Random Subspace Method for high-dimensional regression with the R *package* regRSM

Paweł Teisseyre, Robert A. Kłopotek & Jan Mielniczuk

Computational Statistics

ISSN 0943-4062

Comput Stat
DOI 10.1007/s00180-016-0658-2

ONLINE FIRST

Computational Statistics 31 · 2

2016

ORIGINAL PAPERS

A genetic algorithm for designing microarray experiments
A.H.M.M. Latif · E. Brunner **409**

Composite likelihood and maximum likelihood methods for joint latent class modeling of disease prevalence and high-dimensional semicontinuous biomarker data
B. Zhang · W. Liu · H. Zhang · Q. Chen · Z. Zhang **425**

On high-dimensional two sample mean testing statistics: a comparative study with a data adaptive choice of coefficient vector
S. Kim · J.Y. Ahn · W. Lee **451**

Correcting statistical models via empirical distribution functions
A. Munteanu · M. Wornowizki **465**

Confidence interval estimation by a joint pivot method
W. Xu · W. Yu · Z. Li **497**

Boosting in Cox regression: a comparison between the likelihood-based and the model-based approaches with focus on the R-packages *CoxBoost* and *mboost*
R. De Bin **513**

Selectiongain: an R package for optimizing multi-stage selection
X. Mi · H.F. Utz · A.E. Melchinger **533**

Introducing libeemd: a program package for performing the ensemble empirical mode decomposition
P.J.J. Luukko · J. Helske · E. Räsänen **545**

(*Contents continued on outside back cover*)

Springer

Springer

Comput Stat
DOI 10.1007/s00180-016-0658-2

ORIGINAL PAPER

Random Subspace Method for high-dimensional regression with the R package regRSM

Paweł Teisseyre[1] · Robert A. Kłopotek[1] · Jan Mielniczuk[1,2]

Received: 18 May 2015 / Accepted: 7 April 2016

Abstract Model selection and variable importance assessment in high-dimensional regression are among the most important tasks in contemporary applied statistics. In our procedure, implemented in the package regRSM, the Random Subspace Method (RSM) is used to construct a variable importance measure. The variables are ordered with respect to the measures computed in the first step using the RSM and then, from the hierarchical list of models given by the ordering, the final subset of variables is chosen using information criteria or validation set. Modifications of the original method such as the weighted Random Subspace Method and the version with initial screening of redundant variables are discussed. We developed parallel implementations which enable to reduce the computation time significantly. In this paper, we give a brief overview of the methodology, demonstrate the package's functionality and present a comparative study of the proposed algorithm and the competitive methods like lasso or CAR scores. In the performance tests the computational times for parallel implementations are compared.

Keywords Random Subspace Method · High-dimensional regression · Variable importance measure · Generalized Information Criterion · MPI · R

✉ Paweł Teisseyre
teisseyrep@ipipan.waw.pl

[1] Institute of Computer Science, Polish Academy of Sciences, 5 Jana Kazimierza St., Warsaw, Poland

[2] Faculty of Mathematics and Information Science, Warsaw University of Technology, 75 Koszykowa St., Warsaw, Poland

Published online: 29 April 2016

1 Introduction

1.1 Motivation

In recent years considerable attention has been devoted to model selection and variable importance assessment in high-dimensional statistical learning. This is due to ubiquity of data with a large number of variables in a wide range of research fields. Moreover, nowadays there is a strong need to discover functional relationships in the data and to build predictive models. In many applications the number of variables significantly exceeds the number of observations ("small n large p problem"). However, very often functional relationships are sparse in the sense that among thousands of available variables only a few of them are informative and it is crucial to identify them correctly. Examples include microarray data containing genes activities, Quantitative Trait Loci (QTL) data, drug design data, high-resolution images, high-frequency financial data and text data among others (see e.g. Donoho 2000 for an extensive list of references). In such situations the standard methods like ordinary least squares cannot be applied directly. In view of this a variety of dimension reduction techniques and regression methods tailored to the high-dimensional framework have been developed recently.

1.2 Related work

There are two mainstream methodologies for the dimension reduction in regression: variable extraction methods and variable selection methods. The aim of variable extraction methods is to identify functions of variables that can replace the original ones. Examples include Principal Component Regression (see e.g. Jolliffe 1982) and Partial Least Squares Regression (see e.g. Martens 2001; Wold 2001). In contrast, variable selection methods identify an active set of original variables which affect the response. A significant number of such methods used for high-dimensional data consists of two steps: in the first step the variables are ordered using some forms of regularization or variable importance measures (see below for examples). In the next step the final model is chosen from the hierarchical list of models given by the ordering. Usually in the second step cross-validation, thresholding or information criteria are used to determine the active set.

An important and intensively studied line of research is devoted to regularization methods like lasso, SCAD and MCP (see e.g. Tibshirani 1996; Zou and Hastie 2005; Zhang and Zhang 2012). They perform estimation of parameters and variable selection simultaneously. The final model depends on the regularization parameter which is usually tuned using cross-validation (Friedman et al. 2010). Many algorithms for computing the entire regularization path efficiently have been developed (see e.g. Friedman et al. 2010 for references).

An alternative direction employs ordering of variables based on their appropriately defined importance. Many variable importance measures have been proposed. The `R` package `relaimpo` (RELAtive IMPOrtance of regressors in linear models) implements different metrics for assessing relative importance of variables in the linear models (Grömping 2006). Unfortunately, some of them, e.g. `pmvd` (Proportional

Marginal Variance Decomposition, Feldman 2005) and `lmg` (Latent Model Growth, Lindemann et al. 1980) require a substantial computational effort and usually cannot be applied in the high-dimensional setting. Package `relaimpo` does not provide functions to select the final subset of variables based on their ordering. Another `R` package `caret` (Classification And REgression Training) presented in Kuhn (2008) is convenient for computing the variable importance metrics based on the popular methods like Random Forests (Breiman 2001) or Multivariate Adaptive Regression Splines (Friedman 1991). It also contains several functions that can be used to assess the performance of classification models and choose the final subset of variables. We also mention CAR scores (Correlation-Adjusted coRrelation) proposed in Zuber and Strimmer (2011) and implemented in `R` package `care`. Zuber and Strimmer (2011) proposed to use information criteria to select the final subset of variables.

1.3 Contribution

Recently a novel approach based on the Random Subspace Method (RSM) has been developed in Mielniczuk and Teisseyre (2014). Originally, the RSM was proposed by Ho (1998) for classification purposes and independently by Breiman (2001) for the case when a considered prediction method is either a classification or a regression tree. In our algorithm the ranking of variables is based on fitting linear models on small randomly chosen subsets of variables. The method does not impose any conditions on the number of candidate variables.

The aims of this paper are to present novel variants of the RSM (the weighted Random Subspace Method and the version with initial screening of the redundant variables), describe their implementations and discuss the results of extensive experiments on real and artificial data. We also present a new way of choosing the final model, which is based on Generalized Information Criterion (GIC) and does not require an additional validation set as originally proposed in Mielniczuk and Teisseyre (2014). We show that this step can be performed very fast using properties of QR decomposition of the design matrix.

For discussion of the theoretical properties of the original RSM we refer to Mielniczuk and Teisseyre (2014). The package `regRSM`, containing an implementation of the discussed procedure, is available from the Comprehensive `R` Archive Network at http://cran.r-project.org/web/packages/regRSM/index.html.

This paper is organized as follows. In Sect. 2 we describe briefly the methodology and present new variants of the original algorithm and in Sect. 3 we present the implementation and the package functionality. In Sect. 4 the efficacy experiments are discussed and Sect. 5 contains the analysis of the experiments on artificial and real datasets obtained using package regRSM.

2 Methodology

We consider the usual setup for linear regression. We have n observations and p explanatory variables which serve a complete feature space and are used to predict a response. The model can be written in matrix notation

$$Y = X\beta + \varepsilon,$$

where Y is $n \times 1$ response vector, X is $n \times p$ design matrix whose rows describe objects corresponding to the observations and ε is $n \times 1$ vector of zero mean independent errors with unknown variance σ^2. We allow that $p \geq n$. X_m will denote submatrix of X with columns corresponding to variable set $m \subset \{1, \dots, p\}$ of cardinality $|m|$. In the RSM for regression a random subset of variables X_m with a number of variables $|m| < \min(n, p)$ is chosen. Then, the corresponding model is build based on variables X_m. Selected variables are assigned weights describing their relevance in the considered submodel. In order to cover the large portion of variables in the dataset, the selection is repeated B times and the cumulative weights (called final scores) are computed. The results of all iterations are combined in a list of p variables ordered according to their final scores. The final model is constructed based on selection method applied to the nested list of models, consisting of less than p variables, given by the ordering. We stress that since ranking of variables in the RSM is based on fitting small linear models, the method does not impose any conditions on the number of candidate variables p. Below, the pseudo code of the procedure is outlined.

Algorithm 1 (RSM procedure)

1. Input: observed data (Y, X), a number of subset draws B, a size of the subspace $|m| < \min(p, n)$. Choice of weights $w_n(i, m)$ is described below.
2. Repeat the following procedure for $k = 1, \dots, B$ starting with the counter $C_{i,0} = 0$ for any variable i.
   - Randomly draw a subset of variables m^* (without replacement) from the original variable space with the same probability for each variable.
   - Fit model to data (Y, X_{m^*}) and compute weight $w(i, m^*) \geq 0$ for each variable $i \in m^*$. Set $w(i, m^*) = 0$ if $i \notin m^*$.
   - Update the counter $C_{i,k} = C_{i,k-1} + I\{i \in m^*\}$.
3. For each variable i compute the final score W_i^* defined as

$$W_i^* = \frac{1}{C_{i,B}} \sum_{m^*: i \in m^*} w(i, m^*).$$

4. Sort the list of variables according to scores W_i^*: $W_{i_1}^* \geq W_{i_2}^* \cdots \geq W_{i_p}^*$.
5. Output: Ordered list of variables $\{i_1, \dots, i_p\}$.

The performance of the above procedure crucially depends on a choice of weights $w(i, m^*)$. It is intuitively clear that the weight $w(i, m^*)$ should reflect the importance of the variable i in the randomly chosen model m^* and the goodness of fit of a model m^* should be also taken into account. In the following we will justify our choice of weights. Let

$$T_{i,m} = \hat{\beta}_{i,m}[\hat{\sigma}_m^2 (X_m^\top X_m)_{i,i}^{-1}]^{-1/2}, \quad i \in m$$

be the t-statistic corresponding to the variable i when the model m is fitted to the data. In the above formula $\hat{\beta}_{i,m}$ is the ith coordinate of least squares estimator $\hat{\beta}_m$ based on model m and $\hat{\sigma}_m^2$ is noise variance estimator based on model m. It is argued in Mielniczuk and Teisseyre (2014) that $w(i, m^*) = T^2_{i,m^*}$ is a reasonable choice. A simple motivation is given by the following formula (see e.g., Section 8.5.1 in Rencher and Schaalje 2008)

$$\frac{T^2_{i,m}}{n - |m|} = \frac{R^2_m - R^2_{m\setminus\{i\}}}{1 - R^2_m},$$

where R^2_m is a coefficient of determination for the model m. Namely, it indicates that up to a multiplicative factor, $T^2_{i,m}$ is a decrease in R^2_m due to leaving out ith variable, multiplied by a measure of goodness-of-fit $(1-R^2_m)^{-1}$ of model m and thus it combines two characteristics: importance of a variable within the model m and the importance of the model itself. It is shown in Mielniczuk and Teisseyre (2014) that W_i^* converges to the relative increase of expected prediction error when variable i is omitted from the model averaged over all models of size $|m|$ containing it. We stress that weights W_i^* are not necessarily contained in the interval (0, 1). In order to compare them with other variable importance measures, simple normalization such as division by their maximal value, is necessary.

Observe that Algorithm 1 is generic in nature, i.e., other choices of weights $w(i, m^*)$ are also possible. Two parameters need to be set in the RSM: the number of selections B and the subspace size $|m|$. The smaller the size of a chosen subspace (i.e., a subset of variables chosen) the larger the chance of missing informative variables or missing dependencies between variables. On the other hand, for large $|m|$ many spurious variables can be included adding noisy dimensions to the subspace. Note that if the choice of the weights $w(i, m^*)$ is based on least squares fit then the subspace size is limited by $\min(n, p)$. In the following the value of parameter $|m|$ is chosen empirically. We concluded from numerical experiments in Mielniczuk and Teisseyre (2014) that the reasonable choice is $|m| = \min(n, p)/2$.

It follows from the description above that a parallel version of the algorithm is very easy to implement. Two parallel versions are provided in the package (see Sect. 3 for details). Figure 1 shows a block diagram of the algorithm.

In addition to the main algorithm, we consider a weighted version of the RSM (called WRSM). In the WRSM the additional initial step is performed in which we fit univariate linear models for each variable. Based on the univariate models we compute the individual relevance for each of the variables. Then in the main step the variables are drawn to the random subspaces with probabilities proportional to the relevances determined in the initial step. Thus variables whose individual influence on response is more significant, have larger probability of being chosen to any of the random subspaces. WRSM implements the heuristic premise that variables more correlated with the response variable should be chosen to the final model with larger probability. On the other hand, variables, which are weakly correlated with the response but useful when considered jointly with other variables, have still a chance to be drawn. So WRSM can be seen as a milder version of Sure Independence Screening method proposed in Fan and Lv (2008). Since in the WRSM the relevant variables are more likely to be selected, we can limit the number of repetitions B in the main loop and reduce the computational cost of the procedure. The pseudo code of the WRSM is shown below.

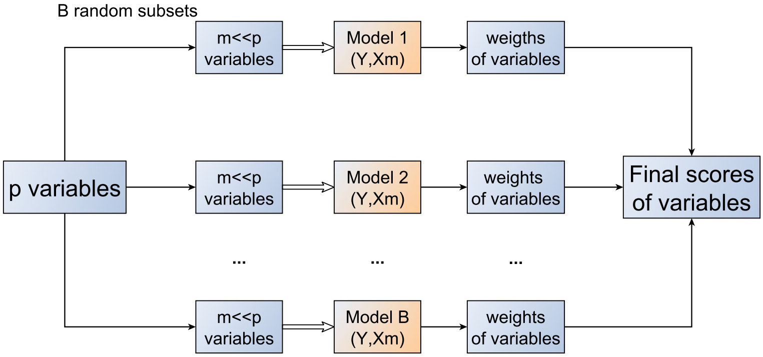

Fig. 1 Block diagram of the RSM procedure

Algorithm 2 (WRSM procedure)

1. Input: observed data (Y, X), number of subset draws B, size of the subspace $|m| < \min(p, n)$. Choice of initial weights $w^0(i)$ is described below.
2. For each variable i fit the univariate regression model and compute a weight of ith variable $w^0(i) \geq 0$.
3. For each variable i compute $\pi_i = w^0(i)/\sum_{l=1}^{p} w^0(l)$.
4. Perform the RSM procedure in such a way that probability of choosing the ith variable to the random subspace is equal π_i.
5. Output: Ordered list of variables $\{i_1, \ldots, i_p\}$.

Observe that when drawing model m^*, probabilities π_i are applied sequentially, that is the probability of choosing the next variable is proportional to the probabilities amongst variables not chosen till that moment. The natural choice of initial weights (used in our package) is $w^0(i) = T^2_{i,\{i\}}$ (i.e., squared t-statistic based on univariate model), however other choices (e.g., mutual information) are also possible. Figure 2 shows a block diagram of the algorithm.

The package also allows for an initial screening of variables. Namely, in the first step, we fit univariate linear models for each variable. Based on the univariate models we compute the individual relevance for each of the variables ($T^2_{i,\{i\}}$ can be used again here). Then the least relevant variables are discarded and the RSM (or WRSM) procedure is performed on the remaining ones. This approach is similar to Sure Independence Screening (SIS) proposed in Fan and Lv (2008) and enable to reduce data dimensionality by filtering out the most spurious variables. This version is particularly recommended for datasets with thousands of variables.

The procedures described above give as an output the ordered list of variables $\{i_1, \ldots, i_p\}$. They also provide importance measures for all variables. In the second step one would like to select the final model from the nested list of models

$$\{\emptyset, \{i_1\}, \{i_1, i_2\} \ldots, \{i_1, \ldots, i_p\}\}$$

given by the ordering. The list should be cut off at some level $h \leq \min(n, p)$ in order to avoid fitting models with more parameters than observations. Thus we select the final set of variables from

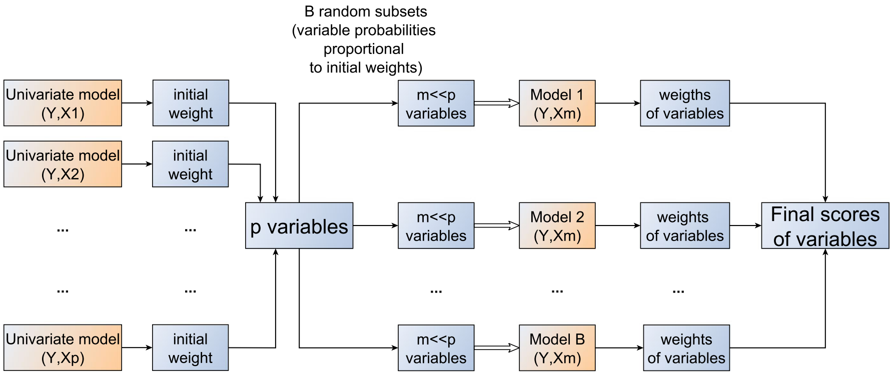

Fig. 2 Block diagram of the WRSM procedure

$$\mathscr{L}_h := \{\emptyset, \{i_1\}, \{i_1, i_2\} \dots, \{i_1, \dots, i_h\}\}. \tag{1}$$

We use two methods of final model selection. The first one is based on supplied validation set (Y^v, X^v). Namely, from family $\mathscr{L}_h$ we select the model m for which the prediction error on validation set $||Y^v - X^v\hat{\beta}_m||$ is minimal ($\hat{\beta}_m$ is the least squares estimator for model m computed on the training set). The drawback of this approach is the need to split data into training and validation sets (it can be a serious problem when the number of available observations is small). The second variant, which does not require splitting the data, is based on information criteria. From $\mathscr{L}_h$ we select model m which minimizes the GIC

$$\text{GIC}(m) := -2l(\hat{\beta}_m) + |m|a_n, \tag{2}$$

where $l(\hat{\beta}_m)$ is log-likehood function at $\hat{\beta}_m$, $|m|$ is the number of parameters in model m and a_n is penalty. For example $a_n = \log(n)$ corresponds to the Bayesian Information Criterion (BIC) and $a_n = 2$ to the Akaike Information Criterion (AIC). When the normal distribution of errors is assumed, GIC can be written as

$$\text{GIC}(m) := n\log(RSS(m)) + |m|a_n, \tag{3}$$

where $RSS(m) = ||Y - X_m\hat{\beta}_m||^2$ is the residual sum of squares for the model m. Obviously other criteria are also possible. The drawback of information criteria is that they can fail when models having number of parameters close to n are fitted. In such situations GIC usually selects models which are close to the saturated one. Thus when using information criteria in the second step, the parameter h should be significantly smaller than n. For example in the simulation experiments, in which p was much larger than n, we set $h = n/2$.

It is crucial for the computational cost that in the second step of the procedure fitting all h nested models from $\mathscr{L}_h$ is not necessary. It suffices to fit the largest one, which makes the second step computationally fast. This follows from the following fact. Let $i_1, \dots, i_h$ be the ordering of variables (cut off at level h) given by the RSM procedure. Let $X_{\{i_1,\dots,i_h\}}$ be a matrix consisting of the h columns of the design matrix X, ordered according to $i_1, \ldots, i_h$. In the following QR decomposition (see e.g., Gentle 2007, p. 182) of matrix $X_{\{i_1,\ldots,i_h\}}$ is used. Let

$$X_{\{i_1,\ldots,i_h\}} = Q_{\{i_1,\ldots,i_h\}} R_{\{i_1,\ldots,i_h\}},$$

where $Q_{\{i_1,\ldots,i_h\}}$ $(n \times h)$ is orthogonal matrix, $Q^{\top}_{\{i_1,\ldots,i_h\}} Q_{\{i_1,\ldots,i_h\}} = I$ (I is identity matrix) and $R_{\{i_1,\ldots,i_h\}}$ $(h \times h)$ is upper triangular matrix. The following equality holds for $k \in \{1, \ldots, h-1\}$

$$RSS(\{i_1, \ldots, i_{h-k}\}) = RSS(\{i_1, \ldots, i_h\}) + \sum_{l=0}^{k-1} (Q^{\top}_{h-l} Y)^2. \tag{4}$$

Observe that in the second step of the procedure it suffices to fit the full model once and than compute the values of RSS for all models from the nested list using (4). Note that QR decomposition requires nh^2 operations (see e.g. Hastie et al. 2009, p. 93).

Finally let us discuss the computational complexity of the whole RSM procedure. The cost of the first step is $Bn|m|^2$ as we fit B linear models with $|m|$ parameters each. The cost of the second step is nh^2 as was discussed above. To reduce the computational cost of the procedure the execution of the first step is parallelized (see Sect. 3 for details).

3 Implementation

3.1 Package functionality

The main function `regRSM` constructs a linear regression model for possibly heavily unbalanced data sets where p may be much larger than n. It uses methodology outlined in the previous section, allowing to apply the RSM method (Algorithm 1), WRSM method (Algorithm 2) and the version with initial screening. It is controlled by the following parameters:

- `y` (the response vector of length n),
- `x` (the input matrix with n rows and p columns),
- `yval` (the optional response vector from validation set),
- `xval` (the optional input matrix from validation set),
- `m` (the size of the random subspace, defaulted to $\min(n, p)/2$),
- `B` (the number of repetitions in the RSM procedure),
- `parallel` (the choice of parallelization method),
- `nslaves` (the number of slaves),
- `store_data` (to be set to `TRUE` when function `validate.regRSM` is used subsequently),
- `screening` (percentage of variables to be discarded in the screening procedure),
- `initial_weights` (whether or not WRSM should be used),
- `useGIC` (indicates whether GIC should be used in the final model selection),
- `thrs` [the cut-off level h, see (1)],

- penalty (the penalty in GIC).

The function returns a list containing the following elements:

- 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 containing information about input parameters),
- informationCriterion (values of GIC 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).

When screening and weighted version are used together, screening is performed first and then the weighted version (WRSM) is used on the remaining variables. The final model is chosen based on the validation set or the GIC (as indicated by useGIC parameter, in conjunction with yval and xval). For this purpose 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). By default the model that minimizes the GIC is chosen. If ties occur the model containing the minimal number of variables is selected. If the validation set is supplied (yval and xval) and useGIC=FALSE then the final model that minimizes the prediction error on validation set is selected. Missing values are not allowed in regRSM function. If there are some missing values in matrix X, the function signals an error and returns indices of cases with missing values. In the case of multicollinearity, we determine the set of linearly independent variables $\mathscr{A}$ (using QR decomposition) and estimate parameters only for variables in $\mathscr{A}$.

The package contains also the number of auxiliary functions:

- validate—This function selects the final model for another, user provided validation set based on the original RSM final scores. 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 as they are based on different validation sets.
- plot—This function produces a plot showing the values of the GIC (or prediction errors on validation set) against the number of variables included in the model.
- ImpPlot—This function produces a dot plot showing final scores from the RSM procedure. Final scores describe importances of explanatory variables.
- predict—This function makes a prediction for new observations. Prediction is based on a final model which is chosen using validation set or GIC.

- `roc`—This function produces ROC-type curve for ordering and computes the corresponding area under the curve (AUC) parameter. Let $i_1, \ldots, i_p$ be the ordering of variables given by the RSM procedure. Let t be the set of relevant variables (i.e., variables whose corresponding coefficients are nonzero), $|t|$ its cardinality and t^c its complement. ROC-type curve for the ordering is defined as:

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

  where

  $$FPR(s) := \frac{|\hat{t}(s) \setminus t|}{|t^c|},$$

  $$TPR(s) := \frac{|\hat{t}(s) \cap t|}{|t|}$$

  and $\hat{t}(s) := \{i_1, \ldots, i_s\}$.
  This function is useful for the evaluation of the ranking produced by the RSM procedure when the set of significant variables t is known (e.g., in the simulation experiments on artificial datasets). When AUC is equal one it means that all significant variables, supplied by the user in argument `truemodel`, are placed ahead of the spurious ones on the ranking list. A similar idea of ranking evaluation is described in Cheng et al. (2014, Section 4).
- `print`, `summary`—These functions print out information about the selection method, screening, initial weights, version (sequential or parallel), size of the random subspace, number of simulations.

3.2 Package demonstration

In this section we illustrate the usage of `regRSM` package step by step using popular real dataset `Boston Housing` (Lichman 2013), containing 505 observations and 14 variables, including response variable `MEDV`, which denotes median value of owner-occupied homes given in 1000$'s. The goal is to predict `MEDV` based on some variables describing houses. The detailed description of the dataset can be found at http://archive.ics.uci.edu/ml/datasets/Housing. To make the task more challenging, we add 100 additional noise variables drawn from standard normal distribution, which are not correlated with the response variable:

```
R> for(j in 1:100){
R>   data <- cbind(data, rnorm(nrow(data)))
R> }
R> names(data)[15:114] <- paste("noise", 1:100)
```

Then we split date into training (400 observations) and validation (105 observations) sets:

```
R> set.seed(1)
R> samp <- sample(1:nrow(data), 400)
R> train <- data[samp, ]
```

```
R> valid <- data[-samp, ]
```

Now we can use RSM procedure on training data:

```
R> model1 <- regRSM(MEDV~., data = train)
```

By default the final model is chosen using GIC. To see the final scores, please type

```
R> model1$scores
```

and to get basic information about the procedure type

```
R> summary(model1)

 Model summary:

Selection method: Generalized Information
Criterion Screening: no
Initial weights: no
Version: sequential
Subspace size: 56
Number of simulations: 1000
```

It is seen that the number of repetitions B is set to 1000. Use a `predict` method to obtain predictions on independent validation set:

```
R> predict(model1, as.matrix(valid[, -14]))
```

We can plot the values of GIC as a function of the number of variables by using the `plot` method:

```
R> plot(model1)
```

The curve generated by the `plot` is given in Fig. 3a. The final model contains 12 variables:

```
R> model1$model

13 6 11 8 5 1 10 4 3 2 9 12
```

Alternatively, the final model can be chosen using validation set, which should be provided as an additional argument in `regRSM` function:

```
R> model2 <- regRSM(x=as.matrix(train[, -14]),
                y = train[, 14],
                yval = valid[, 14],
                xval = as.matrix(valid[, -14]))
```

In this case the final model contains 15 variables:

```
R> model2$model

13 6 11 8 5 3 10 1 4 9 2 7 12 103 62
```

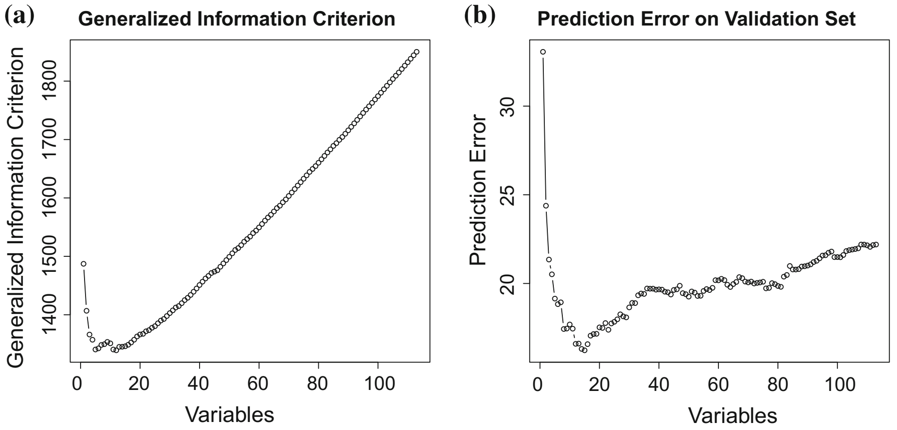

Fig. 3 Generalized Information Criterion (**a**) and prediction error on validation set (**b**) as functions of number of variables

When the final scores are already computed, we can also use `validate` function to select the final subset of variables without recomputing the scores, which is obviously much faster:

```
R> model2 <- validate(model1,
                  yval = valid[, 14],
                  xval = as.matrix(valid[, -14]))
```

For `model2`, the `plot` method gives prediction error on validation set as a function of the number of variables (see Fig. 3b). The weighted version of RSM (WRSM) is called when `init_weights=TRUE`:

```
R> model3 <- regRSM(MEDV~., data = train,
                        init_weights = TRUE)
```

Finally, we can visualize final scores corresponding to RSM (`model1`) and WRSM (`model3`) by using `ImpPlot` method:

```
R> ImpPlot(model1)
R> ImpPlot(model3)
```

The resulting charts (shown for the selected variables) given in Fig. 4a, b indicate that `LSTAT` variable (% lower status of the population) is recognized as the most significant one by both variants (this variable has also the highest absolute value of the t-statistic when standard linear regression model is fitted on data without additional noise variables). On the other hand, the orderings of the remaining variables are slightly different for these two variants of RSM.

3.3 Parallel implementation

The main function `regRSM` performs RSM using methodology outlined in the previous section. The default value `parallel=NO` corresponds to the sequential version.

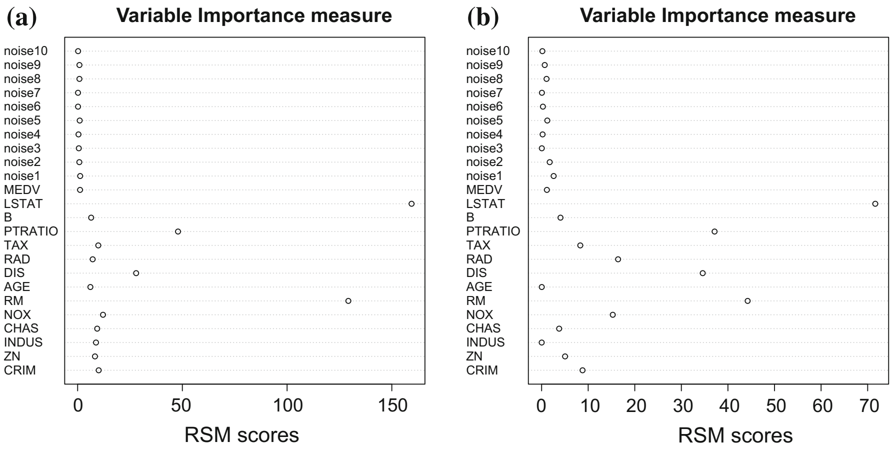

Fig. 4 Final scores of variables obtained using RSM (**a**) and WRSM (**b**)

This option is provided because it is very inefficient to use parallelization on a single processor machine with one core. If hardware for a parallel execution is available, one can choose one of the two parallel versions of the algorithm implemented in the package:

(1) using MPI framework (based on package `Rmpi`), option `parallel=MPI`,
(2) using process forking (based on package `doParallel`), option `parallel=POSIX`.

The execution of the most time consuming step 2 of the RSM algorithm is parallelized. To use the parallel processing, one needs to set the parameter `nslaves` (with default value 4) that indicates how many parallel tasks of partial model building are to be executed.

In order to use `parallel=MPI` option, installation of MPI framework and `Rmpi` package (Hao 2002) are required. A guideline for installing MPI on multiple machines under Ubuntu is given in the "Appendix". `Rmpi` is a wrapper of MPI written in `R` language. The main advantage of this wrapper is that writing `R` programs using MPI is much easier and possible even for non-programmers. The optimal value of `nslaves` under MPI is the number of computing cores of all machines configured in MPI framework. Note that after execution of the `regRSM` function, it is necessary to close MPI framework by calling `mpi.close.Rslaves()` function. Function `regRSM` does not close the MPI framework itself because of efficiency issues as creation of slaves is usually very time consuming. For example if one would like to execute `regRSM` multiple times in a row (e.g., for different datasets or with different settings), the command `mpi.close.Rslaves()` should be used after the last execution of `regRSM`. In this way the process of MPI initialization will be executed only once. Next parallel call will reuse existing slave processes. In order to change the number of slaves, MPI needs to be terminated (using command `mpi.close.Rslaves()`), and only then one can call function `regRSM` with new value of the parameter `nslaves`.

The other parallelization option (parallel=POSIX) does not require any preinstalled software except for the R package doParallel (Revolution Analytics and Weston 2013). The limitation of this parallel mode is that the execution can be performed on one single logical machine only. POSIX uses OpenMP like parallel implementation. The parallel execution is handled by doParallel library. The optimal value of nslaves is the number of processor cores in a machine. In contrast to MPI option, there is no need to shut down the workers (slaves) because the workers will cease to operate if the master R session is completed (or its process dies).

The internal implementations of both parallelization options differ in the way how the parallel processes of variable selection, partial model building and variable evaluation are created and synchronized. The POSIX path delegates the processing of parallelism to the operating system. The MPI path requires elaboration of the proper sequence of messages for starting new slaves, assigning tasks to them, taking over the results and for task reassignment.

In the case of parallel=MPI the Algorithm 1 is replaced by the Algorithm 3 below.

Algorithm 3 (MPI parallelization procedure)

1. Input (for Master): observed data (Y, X), a number of subset draws B, a size of the subspace $|m| < \min(p, n)$.
2. Master: send observed data (Y, X) and parameter $|m|$ to each slave.
3. Master: Compute task_number=B/nslaves.
4. Master: Send task_number to each slave as their B^{local} except for the last one which gets remaining number of tasks B-task_number*(nslaves-1) as its B^{local}.
5. Slave: Repeat the following procedure for $k = 1, \dots, B^{local}$ starting with $C_{i,0}^{local} = 0$ for any variable i.
   - Randomly draw a subset of variables m^* (without replacement) from the original variable space with the same probability for each variable.
   - Fit model to data (Y, X_{m^*}) and compute weight $w(i, m^*) \geq 0$ for each variable $i \in m^*$. Set $w(i, m^*) = 0$ if $i \notin m^*$.
   - Update the counter $C_{i,k}^{local} = C_{i,k-1}^{local} + I\{i \in m^*\}$.
6. Slave: For each variable i compute the partial sum

$$S_i^{local} = \sum_{m^*: i \in m^*} w(i, m^*).$$

7. Slave: send vectors $(S_1^{local}, \dots, S_p^{local})$ and $(C_{1,B^{local}}^{local}, \dots, C_{p,B^{local}}^{local})$ to the Master.
8. Master: Compute final scores:

$$W_i^* = \frac{\sum_{\text{slaves}} S_i^{local}}{\sum_{\text{slaves}} C_{i,B^{local}}^{local}}.$$

9. Master: Sort the list of variables according to scores W_i^*: $W_{i_1}^* \geq W_{i_2}^* \cdots \geq W_{i_p}^*$.
10. Output: Ordered list of variables $\{i_1, \dots, i_p\}$.

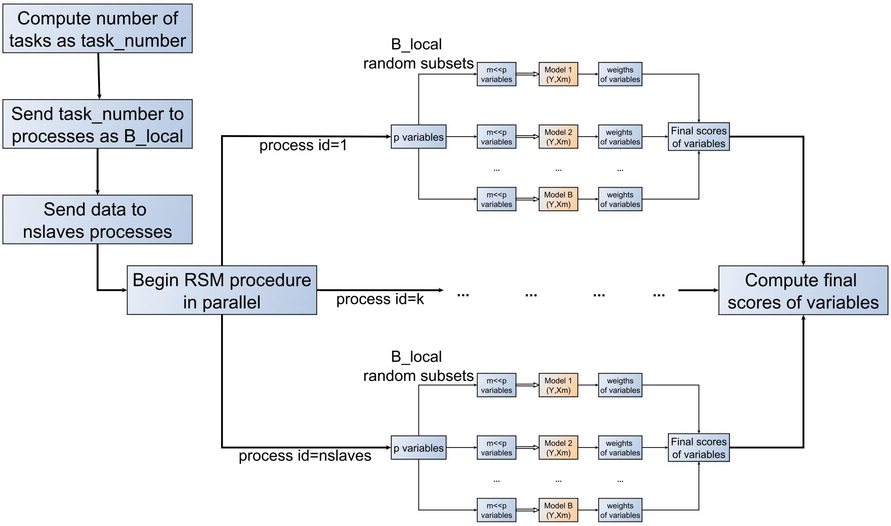
Compute number of tasks as task_number
Send task_number to processes as B_local
Send data to nslaves processes
Begin RSM procedure in parallel
process id=1
process id=k
process id=nslaves
B_local random subsets
p variables
m<<p variables
Model 1 (Y,Xm)
Model 2 (Y,Xm)
Model B (Y,Xm)
weigths of variables
weights of variables
Final scores of variables
Compute final scores of variables

Fig. 5 Block diagram of the parallel version of RSM procedure

In the case of `parallel=POSIX` the algorithm is executed in an analogous way with the difference that the observed data (Y, X), the number of subset draws B, the size of the subspace $|m| < \min(p, n)$ are shared (in common addressing memory space) by the master (in POSIX language: parent) and slaves (in POSIX language: children). Secondly, in this version there is also no explicit message communication. The above features are two main advantages over MPI version when only one logical machine is available. Experimental results comparing MPI and POSIX versions on single machine are shown in the next section. Figure 5 shows a block diagram of the parallel algorithm.

When calling `regRSM` with `parallel=MPI` or `parallel=POSIX` options, the software will check for presence of the `Rmpi` or `doParallel` packages. If they are not installed, the `regRSM` will not be executed and an error message will be displayed.

4 Efficiency

A series of experiments was performed evaluating practical efficiency of the algorithm and its parallelization. The experiments were performed under different parallelization settings: POSIX and MPI for two different hardware settings:

(1) one physical computer with 16 cores (4 × 4 core processor Intel(R) Xeon(R) CPU X7350 @ 2.93 GHz), 64 GB RAM, Open RTE 1.4.3 mpi, R 3.0.1, Ubuntu 12.04 LTS
(2) four physical computers with 4 cores each (4 core processor Intel(R) Core(TM) i7-2600 CPU @ 3.40 GHz), 16 GB RAM (12 GB for VM) each, Open RTE 1.7.3 mpi, R 3.0.2, Ubuntu 12.04 LTS on Oracle VM VirtualBox

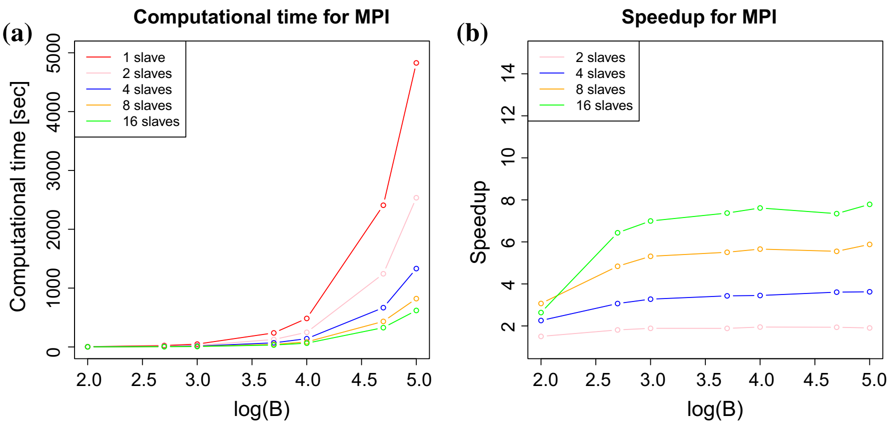

Fig. 6 Execution time (**a**) and speedup (**b**) versus $\log(B)$ (decimal logarithm) for MPI parallel version and hardware setup (1)

In the experiments, the number of slaves varies from 1 to 16. For each number of slaves, computational times are averaged over 5 simulation trials.

In the first experiment, an artificial dataset containing $n = 400$ cases and $p = 1000$ explanatory variables was generated. We set $|m| = \min(n, p)/2 = 200$. We investigated how the computational time depends on the number of simulations B. Figures 6 and 7 show the execution times and speedups against $\log(B)$ for hardware setting (1). Observe that in this case the speedups for POSIX version (Fig. 7b) are larger than for MPI version (Fig. 6b) which makes POSIX version faster. Figure 8 shows the execution times and speedups against $\log(B)$ for hardware setup (2). Observe that MPI version is faster on hardware setup (2) than on setup (1) and is faster than difference in processor frequencies (3.40 vs 2.93 GHz).

In the second experiment we study how the computational time depends on the data size. We set $B = p = 10{,}000$ and change $n = 100, 200, \ldots, 1000$. We set again $|m| = \min(n, p)/2$. Figures 9 and 10 show the execution times and speedups versus n for hardware setting (1) and (2), respectively. For both settings, we can see a breakdown of performance when $n > 600$. This is caused by specific configurations of memory swapping schedules in operating systems. Swapping schedule for hardware setting (1) is more rigorous than for setting (2). Unfortunately this configuration could not be changed. Hardware setting (2), which is based on several physical machines, was able to recover from this downgrade of performance, even when running on a virtual machine and not on a real physical hardware. Here we can see the main advantage of using MPI on different physical machines instead of the single one, which is not possible with POSIX version. In Figs. 6 and 8 we can also observe a slight decrease of performance for hardware setting (2) and $n = 1000$, whereas the performance for setting (1) is constant (except for 16 slaves). The decrease is again caused by lack of memory for hardware setting (2) due to its more limited memory (4×12 GB vs 64 GB).

The observed effects of parallelization seem to be satisfactory. The speedup is not linear with respect to the number of slaves as for MPI overheads occur due to transfer

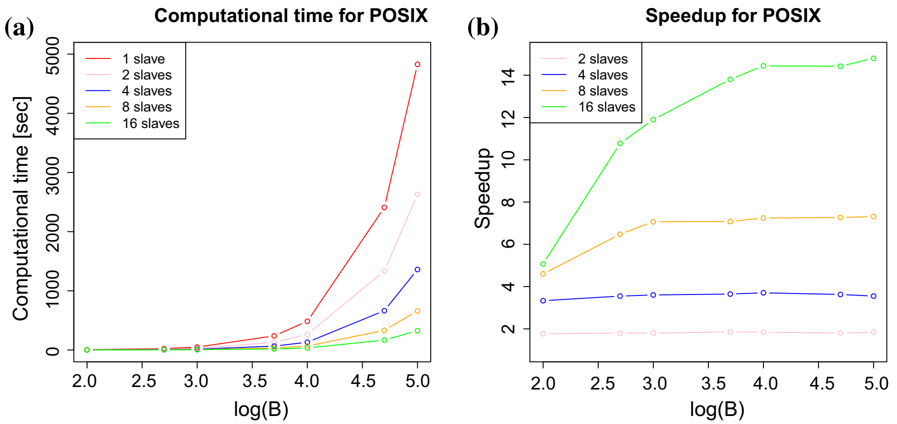

Fig. 7 Execution time (**a**) and speedup (**b**) versus log(B) (decimal logarithm) for POSIX parallel version and hardware setup (1)

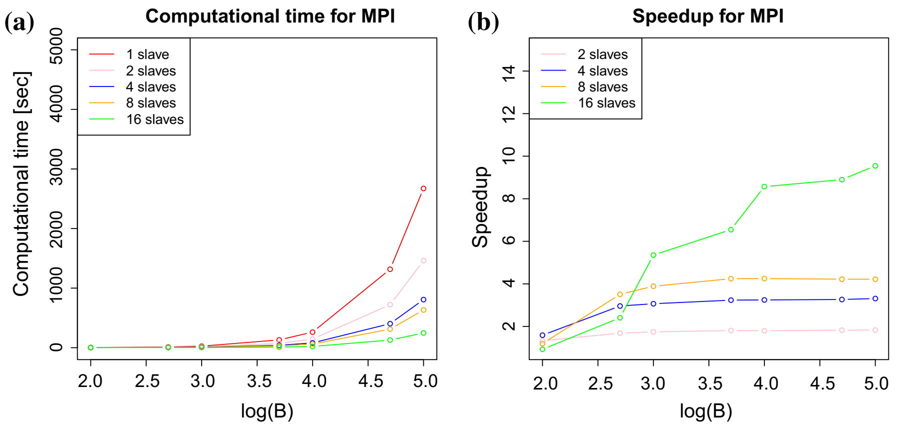

Fig. 8 Execution time (**a**) and speedup (**b**) versus log(B) (decimal logarithm) for MPI parallel version and hardware setup (2)

of the complete data set and MPI-start-ups. Moreover, there are some other tasks which are executed in a sequential way. MPI version on four PC with one processor is faster tan on one server with four processors. This gives us a cheaper solution for speeding up our algorithm.

5 Application examples

5.1 Artificial data example

In this section we study performance of the RSM and the WRSM. We focus on the case $n < p$. First we present the results of experiments on artificial datasets. Let t be the set of relevant variables (i.e., variables whose corresponding coefficients in

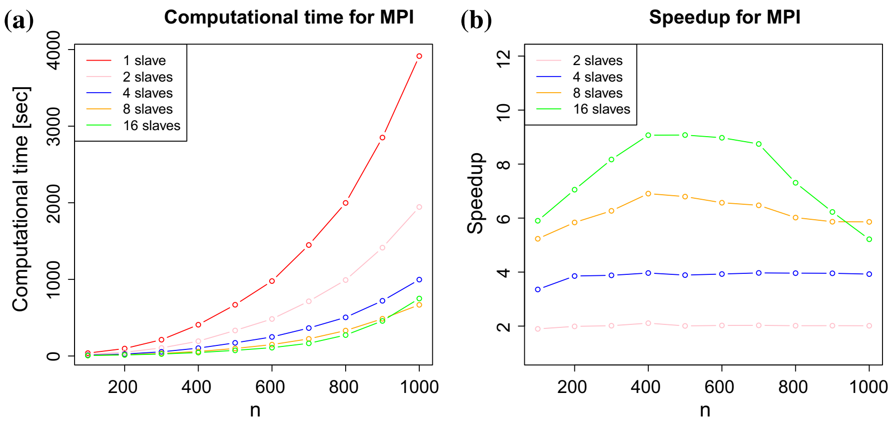

Fig. 9 Execution time (**a**) and speedup (**b**) versus n for MPI parallel version and hardware setup (1)

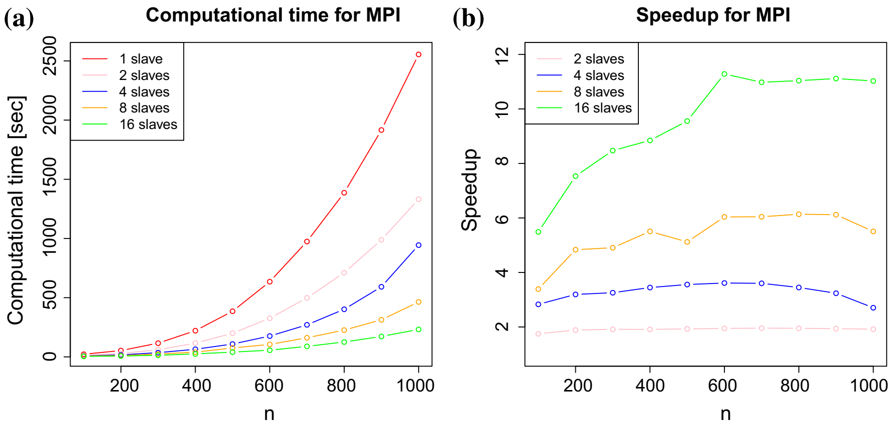

Fig. 10 Execution time (**a**) and speedup (**b**) versus n for MPI parallel version and hardware setup (2)

the model are nonzero), $|t|$ its cardinality and β_t be a subvector of β corresponding to t. Table 1 shows parameters of the considered models. The majority of models are chosen from the examples available in the literature to represent wide variety of structures. Models (M1)–(M4) are considered in Zheng and Loh (1997), models (M5) and (M6) in Huang et al. (2008), model (M7) in Shao and Deng (2012), and model (M8) in Chen and Chen (2008). The last two examples are included in order to consider models with significantly larger number of relevant variables. The rows of X are generated independently from the standard normal p-dimensional distribution with zero mean and the covariance matrix with (i, j)th entry equal $\rho^{|i-j|}$. This type of $AR(1)$ dependence, used for models (M1)–(M8) in the original papers, is frequently considered in the literature devoted to model selection in linear models. We set $\rho = 0.5$ which corresponds to moderate dependence between variables. The outcome is $Y = X_t\beta_t + \varepsilon$, where ε has zero-mean normal distribution with covariance matrix $\sigma^2 I$. As in the original papers, we set $\sigma = 1$ for all models except models 5 and

Table 1 Models summary

Model	$\|t\|$	t	β_t
1	1	$\{10\}$	(0.2)
2	3	$\{2, 4, 5\}$	$(1, 1, 1)$
3	10	$\{2k + 7 : k = 3, \dots, 12\}$	$(1, \dots, 1)$
4	5	$\{k^2 : k = 1, \dots, 5\}$	$(1, \dots, 1)$
5	15	$\{1, \dots, 15\}$	$(2.5, \dots, 2.5, 1.5, \dots, 1.5, 0.5, \dots, 0.5)$
6	15	$\{1, \dots, 5, 11, \dots, 15, 21, \dots, 25\}$	$(2.5, \dots, 2.5, 1.5, \dots, 1.5, 0.5, \dots, 0.5)$
7	20	$\{1, \dots, 20\}$	$(1.1, 1.2, \dots, 3)$
8	8	$\{1, \dots, 8\}$	$(0.7, 0.9, 0.4, 0.3, 1, 0.2, 0.2, 0.1)$
9	50	$\{1, \dots, 25, 51, \dots 75\}$	$(0.5, \dots, 0.5)$
10	50	$\{1, \dots, 25, 51, \dots 75\}$	$(1, \dots, 1)$

6 for which $\sigma = 1.5$. We use the following values of parameters: the number of observations $n = 200$, the number of variables $p = 1000$, the number of repetitions in the RSM $B = 1000$, the subspace size $|m| = \min(n, p)/2 = 100$, the cut-off level $h = \min(n, p)/2 = 100$. The experiments are repeated $L = 500$ times. In the first step the RSM/WRSM is used to obtain the ranking list of variables, in the second step BIC is used to select the final model as described in Sect. 2. The proposed methods are compared with the lasso (Tibshirani 1996), CAR scores (Zuber and Strimmer 2011) and univariate method. To make the comparison with RSM/WRSM fair, the penalty parameter for lasso is chosen using BIC. This approach was described in Zhang et al. (2012) and Fan and Tang (2013). We also tested cross-validation for choosing penalty parameter in lasso. The results were slightly worse than for BIC and thus they are not presented in the tables. For CAR scores we use BIC to select the model. In the univariate method (UNI, sometimes called marginal regression) the prediction strength of each variables is evaluated individually. Here, for each variable $i \in \{1, \dots, p\}$ we compute squared value of its t-statistic based on a simple univariate regression model. Then the variables are ordered with respect to the squared t-statistics and the same procedure on hierarchical list of models as in the RSM is performed.

Let $\hat{t}$ denote the model selected by a given method. As the measures of performance we use the following indices:

- true positive rate (TPR): $|\hat{t} \cap t|/|t|$,
- false discovery rate (FDR): $|\hat{t} \setminus t|/|\hat{t}|$,
- prediction error (PE) equal to root mean squared error computed on independent dataset having the same number of observations as the training set.

The above measures are averaged over $L = 500$ simulations. Observe that TPR calculates a fraction of correctly chosen variables with respect to all significant ones whereas FDR measures a fraction of false positives (selected spurious variables) with respect to all chosen variables. TPR $= 1$ means that all significant variables are included in the chosen model. FDR $= 0$ means that no spurious variables are present in the final model.

Table 2 Mean values of 100 × PE/min(PE) based on 500 simulation trials

Model	|t|	lasso	RSM	WRSM	UNI	CAR	Min
1	1	130.06	112.50	111.79	110.80	111.85	UNI
		(0.634)	(0.523)	(0.836)	(0.472)	(0.496)	
2	3	105.43	100.34	110.61	100.26	100.31	UNI
		(0.27)	(0.073)	(0.565)	(0.068)	(0.069)	
3	10	115.88	100.52	108.03	102.08	101.32	RSM
		(0.543)	(0.101)	(0.535)	(0.291)	(0.258)	
4	5	111.72	100.36	109.63	100.52	100.40	RSM
		(0.462)	(0.054)	(0.53)	(0.089)	(0.069)	
5	15	109.36	114.75	102.03	118.43	117.67	WRSM
		(0.513)	(0.812)	(0.311)	(0.941)	(0.895)	
6	15	112.68	125.33	101.04	129.83	127.94	WRSM
		(0.606)	(1.14)	(0.183)	(1.252)	(1.214)	
7	20	119.14	122.21	100.75	139.89	133.59	WRSM
		(0.629)	(1.908)	(0.228)	(3.258)	(2.637)	
8	8	106.96	100.96	114.95	100.62	100.61	CAR
		(0.362)	(0.154)	(0.525)	(0.101)	(0.099)	
9	50	121.67	115.98	102.85	144.62	132.96	WRSM
		(0.918)	(1.112)	(0.362)	(1.717)	(1.529)	
10	50	130.09	156.73	101.64	225.23	196.56	WRSM
		(1.279)	(3.277)	(0.397)	(4.744)	(4.326)	

Standard deviations of means are provided in brackets

Table 2 shows values of $100 \times \mathrm{PE}/\min(\mathrm{PE})$ averaged over 500 simulations, where $\min(\mathrm{PE})$ is the minimal value of prediction error of 5 considered methods. The last column pertains to the method for which PE was minimal. It is seen that for 5 models the WRSM outperforms all other competitors with respect to PE. Table 3 shows values of TPR; the last column pertains to the method for which the maximal TPR is attained. Observe that in the majority of models, TPR for the lasso is close to one, which indicates that lasso selects most relevant variables. However, the differences between the lasso and the WRSM are negligible. Table 4 shows values of FDR; the last column pertains to the method for which the minimal FDR is obtained. Here, the WRSM outperforms other methods in the four out of ten cases. Table 5 contains information about sizes of chosen models: RSM usually selects much smaller models than lasso.

The clear advantage of the WRSM over other methods can be seen for models with large number of relevant variables (e.g., 7, 9, 10). For these models, the significant variables are usually placed on the top of the ranking list when the WRSM is used. It is not necessarily the case for other methods. For example in the case of model 7, the position of the last relevant variable in the ranking list (averaged over simulation trials) is: 20 (lasso), 64.76 (RSM), 20 (WRSM), 124.88 (UNI), 99.68 (CAR). This indicates that for the lasso and the WRSM, all relevant variables are placed ahead of spurious ones in all simulations. On the other hand in some situations (usually for models with

Table 3 Mean values of true positive rates (TPR) based on 500 simulation trials

Model	$\|t\|$	Lasso	RSM	WRSM	UNI	CAR	Max. TPR
1	1	0.785	0.570	0.310	0.575	0.575	Lasso
		(0.03)	(0.035)	(0.033)	(0.035)	(0.035)	
2	3	1.000	1.000	1.000	1.000	1.000	All
		(0)	(0)	(0)	(0)	(0)	
3	10	1.000	1.000	1.000	0.998	0.998	Lasso, RSM, WRSM
		(0)	(0)	(0)	(0.001)	(0.001)	
4	5	1.000	1.000	1.000	1.000	1.000	All
		(0)	(0)	(0)	(0)	(0)	
5	15	0.993	0.811	0.986	0.786	0.791	Lasso
		(0.001)	(0.005)	(0.002)	(0.005)	(0.005)	
6	15	0.989	0.747	0.981	0.727	0.737	Lasso
		(0.002)	(0.005)	(0.002)	(0.005)	(0.005)	
7	20	1.000	0.979	1.000	0.962	0.968	Lasso, WRSM
		(0)	(0.002)	(0)	(0.003)	(0.003)	
8	8	0.858	0.838	0.702	0.831	0.834	Lasso
		(0.006)	(0.005)	(0.007)	(0.005)	(0.005)	
9	50	0.998	0.918	0.954	0.840	0.876	Lasso
		(0.001)	(0.003)	(0.003)	(0.005)	(0.005)	
10	50	1.000	0.951	0.992	0.881	0.910	Lasso
		(0)	(0.003)	(0.001)	(0.006)	(0.005)	

Standard deviations of means are provided in brackets

small number of significant variables, e.g., 2, 4, 8) the WRSM have quite large FDRs compared to other methods. In these cases, the relevant variables are also placed on the top of the ranking list (TPRs are close to one) but the Bayesian Information Criterion used in the second step selects too many spurious variables to the final model. As this behaviour occurs for small values of $|t|$ the number of false positives is also small in absolute terms, however.

There is also an important difference between the RSM and the WRSM. Consider spurious variables which are strongly correlated with the relevant ones (e.g., variable 1, 3, 6 in model 2). In the case of the RSM, such variables are usually placed right behind relevant ones in the ranking list. The top 10 variables for the RSM and model 2 (determined for one example simulation) are:

$$\mathbf{4}, \mathbf{5}, \mathbf{2}, 3, 6, 1, 778, 767, 423, 519$$

(the significant variables are in bold). On the other hand, for the WRSM, spurious variables correlated with relevant ones (thus correlated also with the response) are usually drawn together with relevant variables. Therefore they are assigned smallweights

Table 4 Mean values of false discovery rates (FDR) based on 500 simulation trials

Model	$\|t\|$	Lasso	RSM	WRSM	UNI	CAR	Min. FDR
1	1	0.996	0.935	0.946	0.925	0.930	UNI
		(0)	(0.005)	(0.009)	(0.006)	(0.005)	
2	3	0.223	0.035	0.643	0.059	0.050	RSM
		(0.015)	(0.006)	(0.014)	(0.007)	(0.007)	
3	10	0.374	0.316	0.348	0.396	0.365	RSM
		(0.009)	(0.008)	(0.014)	(0.007)	(0.008)	
4	5	0.344	0.060	0.506	0.135	0.091	RSM
		(0.014)	(0.009)	(0.014)	(0.012)	(0.011)	
5	15	0.202	0.164	0.213	0.181	0.167	RSM
		(0.008)	(0.012)	(0.011)	(0.014)	(0.013)	
6	15	0.247	0.240	0.150	0.246	0.252	WRSM
		(0.008)	(0.015)	(0.011)	(0.015)	(0.014)	
7	20	0.203	0.273	0.026	0.342	0.315	WRSM
		(0.007)	(0.015)	(0.006)	(0.016)	(0.016)	
8	8	0.154	0.065	0.605	0.047	0.053	UNI
		(0.011)	(0.008)	(0.012)	(0.006)	(0.007)	
9	50	0.639	0.222	0.188	0.257	0.243	WRSM
		(0.01)	(0.007)	(0.007)	(0.008)	(0.008)	
10	50	0.463	0.308	0.193	0.333	0.319	WRSM
		(0.01)	(0.008)	(0.008)	(0.008)	(0.01)	

Standard deviations of means are provided in brackets

$w(i, m^*)$ and consequently have smaller final scores. The top 10 variables for the WRSM and model 2 (determined for one example simulation) are:

$$\mathbf{2}, \mathbf{4}, \mathbf{5}, 423, 442, 514, 601, 441, 263, 745$$

(the significant variables are in bold).

We also analyse the effect of the subspace size in our method. We run the experiments with different subspace sizes $|m| \in \{1, 5, 25, 50, 75, \ldots, \min(n, p) - 1\}$. Let $PE(|m|)$, $TRP(|m|)$, $FDR(|m|)$ and $LEN(|m|)$ be respectively: the prediction error, the true positive rate, the false discovery rate and the length of the chosen model, for a given subspace size $|m|$. Let $|m_{def}| = \min(n, p)/2 = 100$ be the default subspace size used in all experiments and also recommended in Mielniczuk and Teisseyre (2014). Table 6 shows the ratio of the performance measure for $|m_{def}|$ to the performance measure corresponding to the optimal subspace size. Value 1 means that the performance measure is optimal for $|m_{def}|$. Observe that we obtain the smallest prediction errors for $|m_{def}|$ in the case of 6 models. This additionally confirms the validity of our choice.

Table 5 Mean values of model sizes based on 500 simulation trials

Model	$\|t\|$	Lasso	RSM	WRSM	UNI	CAR	Min. size
1	1	221.295	11.540	11.705	9.415	10.375	UNI
		(0.457)	(0.361)	(0.989)	(0.293)	(0.297)	
2	3	4.205	3.215	11.025	3.285	3.255	RSM
		(0.11)	(0.031)	(0.429)	(0.032)	(0.032)	
3	10	16.600	15.135	16.900	17.255	16.190	RSM
		(0.258)	(0.196)	(0.389)	(0.22)	(0.207)	
4	5	8.335	5.420	12.585	6.065	5.685	RSM
		(0.208)	(0.077)	(0.402)	(0.122)	(0.098)	
5	15	19.035	15.540	19.870	15.430	15.190	CAR
		(0.207)	(0.362)	(0.348)	(0.418)	(0.39)	
6	15	20.300	16.145	18.225	16.030	16.400	UNI
		(0.228)	(0.48)	(0.308)	(0.473)	(0.459)	
7	20	25.590	30.290	20.660	33.470	31.935	WRSM
		(0.257)	(0.907)	(0.195)	(1.012)	(1.019)	
8	8	8.405	7.330	16.840	7.105	7.190	WRSM
		(0.164)	(0.116)	(0.541)	(0.083)	(0.091)	
9	50	155.550	60.315	59.480	58.655	59.605	UNI
		(2.869)	(0.73)	(0.632)	(1.013)	(0.932)	
10	50	101.120	70.975	62.825	68.955	69.665	WRSM
		(1.944)	(0.991)	(0.73)	(1.143)	(1.218)	

Standard deviations of means are provided in brackets

Table 6 Ratio of the performance measure for the default subspace size $|m_{def}|$ to the performance measure corresponding to the optimal subspace size

	Model									
	1	2	3	4	5	6	7	8	9	10
$\frac{PE(\|m_{def}\|)}{\min_{\|m\|} PE(\|m\|)}$	1.13	1.00	1.00	1.00	1.00	1.00	1.03	1.00	1.02	1.12
$\frac{TPR(\|m_{def}\|)}{\max_{\|m\|} TPR(\|m\|)}$	0.92	1.00	1.00	1.00	1.00	1.00	1.00	0.98	0.99	0.99
$\frac{FDR(\|m_{def}\|)}{\min_{\|m\|} FDR(\|m\|)}$	1.03	1.33	1.19	2.09	1.26	1.27	1.15	1.00	1.04	1.02
$\frac{LEN(\|m_{def}\|)}{\min_{\|m\|} LEN(\|m\|)}$	1.28	1.01	1.08	1.04	1.08	1.12	1.10	1.00	1.03	1.04

5.2 Real data example

The algorithms are also compared on real high-dimensional dataset described in Hannum et al. (2013). The dataset is available at http://www.ipipan.eu/~teisseyrep/SOFTWARE. In the dataset, there are 657 observations representing individuals, aged 19–101 and 473,034 variables representing methylation states of CpG markers. Methylation was recorded as a fraction between zero and one, representing the frequency of methylation of a given CpG marker across the population of blood cells taken from a single individual. To estimate the prediction error (PE) we use 3-fold cross-validation. To reduce the dimensionality we select 15 % of variables that are most correlated with the response for each cross-validation split. We build models using the remaining 70,955 variables. As a baseline we use naive method for which the prediction is a sample mean of the response calculated on the training set. For the RSM, the WRSM, CAR and univariate method (UNI), the final model is chosen using BIC or the validation set. In the latter case, the validation set is separated from the training part in such a way that 50 % of observations are used to build a model and the remaining 50 % for validation. In the case of the lasso, the final model is selected using cross-validation. For the RSM and the WRSM we use the following values of parameters: number of repetitions $B = 2000$, subspace size $|m| = \min(n, p)/2$, cut-off level $h = \min(n, p)/2$.

Table 7 shows the prediction errors and the sizes of selected models averaged over 3 cross-validation splits. The value in bold pertains to the minimal value in each column. Observe that for all methods there is a significant improvement over the naive method. Note that for the lasso we obtain larger prediction errors and much larger models than for other considered models. When the final model is chosen using BIC, we get the

Table 7 Results for 3-fold cross-validation

Method	PE (BIC)	$\|\hat{t}\|$ (BIC)	PE (VAL)	$\|\hat{t}\|$ (VAL)
RSM	**6.45**	**10**	6.85	44.33
WRSM	6.59	28.33	6.92	22.33
CAR	6.55	10.66	**6.79**	**21.33**
UNI	6.69	14.66	7.08	27.33
Lasso	7.69	189.33	7.69	189.33
Naive	14.75	1	14.75	1

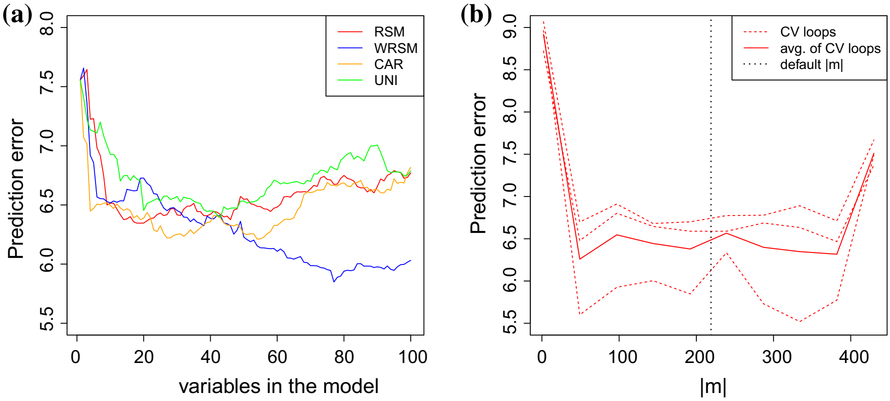

Fig. 11 Prediction errors with respect to the number of variables included in the model (**a**). Prediction errors with respect to the subspace size $|m|$ for RSM (**b**)

Table 8 Top 50 variables for real dataset

	RSM	RL	WRSM	RL	CAR	RL	UNI	RL	Lasso
1	cg16867657	1	cg16867657	1	cg08097417	6	cg16867657	1	cg16867657
2	cg08097417	6	cg08097417	6	cg14361627	4	cg06639320	3	cg10501210
3	cg06639320	3	cg14361627	4	cg16867657	1	cg24724428		cg06639320
4	cg14361627	4	cg10501210	2	cg22454769	5	cg22454769	5	cg14361627
5	cg10501210	2	cg06639320	3	cg06639320	3	cg10501210	2	cg22454769
6	cg22454769	5	cg22454769	5	cg07955995		cg24079702		cg08097417
7	cg23606718	46	cg07955995		cg09499629		cg07553761		cg19283806
8	cg07955995		cg24079702		cg24079702		cg21572722		cg14692377
9	cg24079702		cg02650266	12	cg10501210	2	cg19283806	7	cg16054275
10	cg24724428		cg10591771		cg04875128		cg06784991		cg08234504
11	cg00748589		cg24955895		cg19283806	7	cg08234504	10	cg07082267
12	cg09499629		cg22285878		cg22285878		cg04875128		cg02650266
13	cg17110586		cg24724428		cg03607117	39	cg14692377	8	cg03399905
14	cg02650266	12	cg26690592		cg22736354	20	cg01974375	17	cg20426994
15	cg07850154		cg09499629		cg03032497	21	cg16054275	9	cg23091758
16	cg04875128		cg23606718	46	cg06493994		cg22736354	20	cg04400972
17	cg21572722		cg08719712		cg21572722		cg23744638		cg01974375
18	cg17621438		cg21572722		cg20426994	14	cg07547549		cg08128734
19	cg20426994	14	cg07850154		cg08719712		cg08160331		cg18618815
20	cg06493994		cg10804656		cg24724428		cg02650266	12	cg22736354
21	cg19283806	7	cg00748589		cg01528542		cg08128734	18	cg03032497
22	cg14692377	8	cg17621438		cg00748589		cg23500537		cg04581938
23	cg03473532		cg06493994		cg25478614		cg23078123	37	cg22796704
24	cg22796704	23	cg03399905	13	cg25410668		cg03032497	21	cg21801378
25	cg07547549		cg17110586		cg23091758	15	cg22796704	23	cg06240854

Table 8 continued

	RSM	RL	WRSM	RL	CAR	RL	UNI	RL	Lasso
26	cg22285878		cg00058879		cg18473521		cg25994988		cg25428494
27	cg18933331		cg16312514		cg23606718	46	cg15804973		cg26290632
28	cg03399905	13	cg20426994	14	cg23500537		cg08097417	6	cg22016779
29	cg10804656		cg08540945		cg21186299		cg16932827		cg15707833
30	cg07082267	11	cg17802949		cg16008966	40	cg14361627	4	cg16193278
31	cg17802949		cg01528542		cg07547549		cg08262002		cg01820962
32	cg07553761		cg00590036		cg14692377	8	cg16419235		cg04604946
33	cg11693709		cg07553761		cg06419846		cg24466241		cg17183905
34	cg18468088		cg20149168		cg17110586		cg03259243		cg13221458
35	cg16008966	40	cg20275558		cg07927379		cg07082267	11	cg11067179
36	cg01528542		cg18450254		cg17621438		cg01763090		cg22213242
37	cg13327545		cg03607117	39	cg04581938	22	cg00292135		cg23078123
38	cg03607117	39	cg10397932		cg04084157		cg03735592		cg18310639
39	cg26290632	27	cg15707833	29	cg20052760		cg01820374		cg03607117
40	cg02018902		cg18633600		cg08160331		cg01528542		cg16008966
41	cg18450254		cg02452500	48	cg00481951		cg10149533		cg05207048
42	cg02328239		cg26290632	27	cg26290632	27	cg22156456		cg06231995
43	cg20822990		cg21186299		cg04400972	16	cg17110586		cg15894389
44	cg08160331		cg17715419		cg02650266	12	cg07080372		cg02481950
45	cg09310092		cg16008966	40	cg03399905	13	cg09809672		cg02924487
46	cg09809672		cg06419846		cg01763090		cg24711336		cg23606718
47	cg16312514		cg04940570		cg07553761		cg26161329		cg15936446
48	cg08719712		cg05576959		cg11067179	35	cg03431918		cg02452500
49	cg00094518		cg04875128		cg08128734	18	cg22285878		cg10106965
50	cg04084157		cg19283806	7	cg10804656		cg05991454		cg01541867

RL denotes a position of the given variable in the ranking based on lasso smallest error for the RSM (see the first column). When the selection is based on validation set, CAR method is a winner (see the third column).

Figure 11a shows prediction errors (averaged over 3 folds) with respect to the number of variables included in the model. Observe that when the number of variables included in the final model is sufficiently large, the prediction errors for the WRSM are smaller than for competitive models. Figure 11b shows prediction errors with respect to the subspace size $|m|$ for RSM. The vertical line corresponds to the default subspace size.

Table 8 shows rankings of top 50 variables obtained using considered methods. As in the original paper (Hannum et al. 2013), lasso was used to assess the relevance of the variables, we compare rankings obtained by RSM, WRSM, CAR and UNI with the one based on lasso. RL in Table 8 denotes a position of the given variable in the ranking based on lasso (empty space means that the given variable is not ranked in top 50 variables by lasso). Note that the rankings, corresponding to considered methods, are not fully concordant, which may be valuable in biological research as some new relevant variables can be potentially discovered. It is seen that 6 variables, recognized as the most significant ones by lasso are also ranked on the top 6 positions by RSM and WRSM. It is interesting that `cg08097417` is the second most important variable according to RSM/WRSM (and the most important variable according to CAR), whereas it is placed on 6th position by lasso. Finally, observe that `cg14361627` (4th position according to lasso and RSM) is not recognized as very relevant variable by UNI, which may suggest that this variable is relatively weakly correlated with the response but becomes useful when considered jointly with other variables.

6 Summary

In this paper we presented a novel variants of RSM as well as an implementation in `R` package `regRSM`. The method does not impose any conditions on the number of candidate variables. The underlying algorithms are discussed. The first step in our procedure is based on fitting linear models on small randomly chosen subsets of variables and thus it allows for parallelization. Two versions of parallel implementation are presented. Moreover other improvements of the original method are introduced, including an initial screening of variables and their weighting in the sampling process. The article presents the empirical evaluation of our implementation including: its efficiency in identifying the significant variables, its prediction power and acceleration of the processing due to parallel implementations. The method and its weighted variant compare well with other methods tailored to the high-dimensional setup (like lasso or CAR scores) and is amenable to parallelization under various hardware settings (single and multiple physical machines) and parallel softwares (MPI, POSIX).

Acknowledgments Research of Paweł Teisseyre and Robert A. Kłopotek was supported by the European Union from resources of the European Social Fund within project 'Information technologies: research and their interdisciplinary applications' POKL.04.01.01-00-051/10-00.

Appendix

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:

```
R> 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:

```
R> library("Rmpi")

library("Rmpi")

R> 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 have been launched successfully.

References

Breiman L (2001) Random forests. Mach Learn 45(1):5–32

Chen J, Chen Z (2008) Extended Bayesian information criteria for model selection with large model spaces. Biometrika 95:759–771

Cheng J, Levina E, Wang P, Zhu J (2014) A sparse Ising model with covariates. Biometrics 70:943–953

Donoho DL (2000) High-dimensional data analysis: the curses and blessings of dimensionality. Aide-memoire of a lecture at AMS conference on math challenges of the 21st century

Fan J, Lv J (2008) Sure independence screening for ultra-high dimensional feature space (with discussion). J R Stat Soc B 70:849–911

Fan Y, Tang CY (2013) Tuning parameter selection in high dimensional penalized likelihood. J R Stat Soc Ser B (Stat Methodol) 75(3):531–552

Feldman B (2005) Relative importance and value. http://ssrn.com/abstract=2255827

Friedman JH (1991) Multivariate adaptive regression splines. Ann Stat 19(1):1–67

Friedman J, Hastie T, Tibshirani R (2010) Regularization paths for generalized linear models via coordinate descent. J Stat Softw 33(1):1–22

Gentle JE (2007) Matrix algebra: theory, computations, and applications in statistics. Springer, New York

Grömping U (2006) Relative importance for linear regression in R: the package relaimpo. J Stat Softw 17(1):1–27

Hannum G, Guinney J, Zhao L, Zhang L, Hughes G, Sadda S, Klotzle B, Bibikova M, Fan JB, Gao Y, Deconde R, Chen M, Rajapakse I, Friend S, Ideker T, Zhang K (2013) Genome-wide methylation profiles reveal quantitative views of human aging rates. Mol Cell 49(2):359–367

Hao Y (2002) Rmpi: parallel statistical computing in R. R News 2(2):10–14. http://cran.r-project.org/doc/Rnews/Rnews_2002-2.pdf

Hastie T, Tibshirani R, Friedman J (2009) The elements of statistical learning: data mining, inference and prediction. Springer. http://www-stat.stanford.edu/tibs/ElemStatLearn/

Ho TK (1998) The random subspace method for constructing decision forests. IEEE Trans Pattern Anal Mach Intell 20(8):832–844

Huang J, Ma S, Zhang C-H (2008) Adaptive lasso for high-dimensional regression models. Stat Sin 18:1603–1618

Jolliffe IT (1982) A note on the use of principal components in regression. J R Stat Soc Ser C (Appl Stat) 31(3):300–303

Kuhn M (2008) Building predictive models in R using the caret package. J Stat Softw 28(5):1–26

Lichman M (2013) UCI machine learning repository. http://archive.ics.uci.edu/ml

Lindemann R, Merenda P, Gold R (1980) Introduction to bivariate and multivariate analysis. Scott Foresman, Glenview

Martens H (2001) Reliable and relevant modelling of real world data: a personal account of the development of PLS regression. Chemom Intell Lab Syst 58(2):85–95

Mielniczuk J, Teisseyre P (2014) Using random subspace method for prediction and variable importance assessment in regression. Comput Stat Data Anal 71:725–742

Rencher AC, Schaalje GB (2008) Linear models in statistics. Wiley, Hoboken

Revolution Analytics, Weston S (2013) doParallel: foreach parallel adaptor for the parallel package. http://CRAN.R-project.org/package=doParallel. R package version 1.0.6

Shao J, Deng X (2012) Estimation in high-dimensional linear models with deterministic covariates. Ann Stat 40(2):812–831

Tibshirani R (1996) Regression shrinkage and selection via the lasso. J R Stat Soc B 58:267–288

Wold S (2001) Personal memories of the early PLS development. Chemom Intell Lab Syst 58(2):83–84

Zhang C-H, Zhang T (2012) A general theory of concave regularization for high-dimensional sparse estimation problems. Stat Sci 27(4):576–593

Zhang Y, Lia R, Tsaia C-L (2012) Regularization parameter selections via generalized information criterion. J Am Stat Assoc 105:312–323

Zheng X, Loh W-Y (1997) A consistent variable selection criterion for linear models with high-dimensional covariates. Stat Sin 7:311–325

Zou H, Hastie T (2005) Regularization and variable selection via the elastic net. J R Stat Soc B 67(2):301–320

Zuber V, Strimmer K (2011) High-dimensional regression and variable selection using car scores. Stat Appl Genet Mol Biol 10(1):301–320