lmSubsets: Efficient Computation of Variable-Subsets Linear Regression in R

Marc Hofmann
`TODO: affiliation`

Cristian Gatu
`TODO: affiliation`

Erricos J. Kontoghiorghes
Birbeck College

Ana Colubi
University of Oviedo

Achim Zeileis
Universität Innsbruck

Abstract

`TODO: write abstract`

Keywords: linear regression, model selection, variable selection, best subset regression, R.

1. Introduction

An important problem in statistical modeling is that of subset-selection regression or, equivalently, of finding the best regression equation (Hastie, Tibshirani, and Friedman 2001). Given a set of possible variables to be included in the regression, the problem is to select a subset which optimizes some statistical criterion. The latter originates in the estimation of the corresponding submodel (Miller 2002). Consider the standard regression model

$$y = X\beta + \epsilon, \qquad \epsilon \sim (0, \sigma^2 I_M), \tag{1}$$

where $y \in \mathbb{R}^M$ is the output variable vector, $X \in \mathbb{R}^{M\times N}$ is the exogenous data matrix of full column rank, $\beta \in \mathbb{R}^N$ is the coefficient vector and $\epsilon \in \mathbb{R}^N$ is the noise vector. The columns of X correspond to the set of exogenous variables $V = \{v_1, \ldots, v_N\}$. A submodel S of (1) comprises some of the variables in V. The best-subset selection problem is to determine

$$S^* = \operatorname*{argmin}_{S\subseteq V} f(S), \quad \text{where } f \text{ is some criterion function.} \tag{2}$$

Standard selection criteria include the AIC family (Miller 2002). For constant number of parameters these criteria are monotonic functions of the residual sum of squares (RSS), and attain their optimal value when the RSS is minimal. Thus (2) can be re-written as a two stage problem. In a first stage find

$$S_n^* = \operatorname*{argmin}_{S\subseteq V,\ |S|=n} \text{RSS}(S) \quad \text{for} \quad n = 1, \ldots, N, \tag{3}$$

and in the second stage select

$$S^* = \operatorname*{argmin}_{S_n^*, n=1,\ldots,N} f(S_n^*). \tag{4}$$

Solving (3) means to compute all-subset models corresponding to each submodel size. Since the S_n^* $(n = 1, \ldots, N)$ do not depend on a model size penalty, the solution of (3) can rely on the RSS only. In the second stage the best-submodel S^* can be selected with respect to any standard criterion over the N submodels $S_1^*, \ldots, S_N^*$. Therefore, solving the problem (2) is equivalent in solving the problems (3) and (4). Note that solving directly the best-subset problem (2) can allow a computational strategy to focus on finding the best submodel at lower computational cost. On the other hand solving the problem (3) can require a higher computational effort. Still, in this case the output will be a list of N submodels on which a further selection/analysis can be performed.

Algorithms for subset regression that do exact search but avoid exhaustive fitting of all models can be found on the R package **leaps** (Lumley and Miller 2009) based on Miller (2002). Exhaustive search has been considered also within the context of generalized linear models and summarized on the **bestglm** package (McLeod and Xu 2014). Alternative approximate solutions based on simulated annealing were introduced in the **subselect** package (Orestes Cerdeira, Duarte Silva, Cadima, and Minhoto 2015) based on Duarte Silva (2001). Furthermore approximate solutions via genetic algorithms for generalized linear models and beyond have been implemented in **glmulti** (Calcagno and de Mazancourt 2010) and **kofnGA** (Wolters 2015). Regularized estimation of parametric models with automatic variable selection is performed by lasso or elastic net estimation for generalized linear models (Friedman, Hastie, and Tibshirani 2010).

Here, the **lmSubsets** package for exact, best variable-subset regression is presented. It implements the algorithms presented by Gatu and Kontoghiorghes (2006) and Hofmann, Gatu, and Kontoghiorghes (2007). The package embeds functions that solve both the best-subset problem (2) and the all-subsets problem (3). A branch-and-bound device is employed to prune non-optimal sub-spaces when computing the solution. Exact and approximate strategies are deployed in order to improve the computational performance of the branch-and-bound algorithm. The estimation numerical tool employed is the QR decomposition and its modification. Computationally intensive core code is written in C++. It is available as a package (Hofmann, Gatu, Kontoghiorghes, and Zeileis 2016) for the R system for statistical computing (R Core Team 2016) from the Comprehensive R Archive Network at `https://CRAN.R-project.org/package=lmSubsets`.

2. Computation strategies

In the linear regression model from Equation 1 there are $2^N - 1$ possible subset models and an exhaustive computation of all of them is only feasible for small values of N. However, regression trees can be employed to traverse the search space in a systematic fashion, avoiding the need to explicitly compute every subset combination. Figure 1 illustrates a regression tree for $N = 5$ variables. A node in the regression tree is a pair (S, k), where $S = (s_1, \ldots, s_n)$ is a certain subset of n variables and k is an integer, symbolized by a $\bullet$.

Each node corresponds to a unique subset of variables, although not every possible subset gives rise to a node. Thus, the number of nodes is 2^{N-1}. The root node corresponds to the full set of variables, that is $(V, 1)$. When the algorithm visits node (S, k), it reports the RSS of the models corresponding to the leading variable subsets of size $k + 1$, $\ldots$, n, i.e., the subleading models $(s_1, \ldots, s_{k+1})$, $\ldots$, $(s_1, \ldots, s_n)$. It then generates and in turn visits the

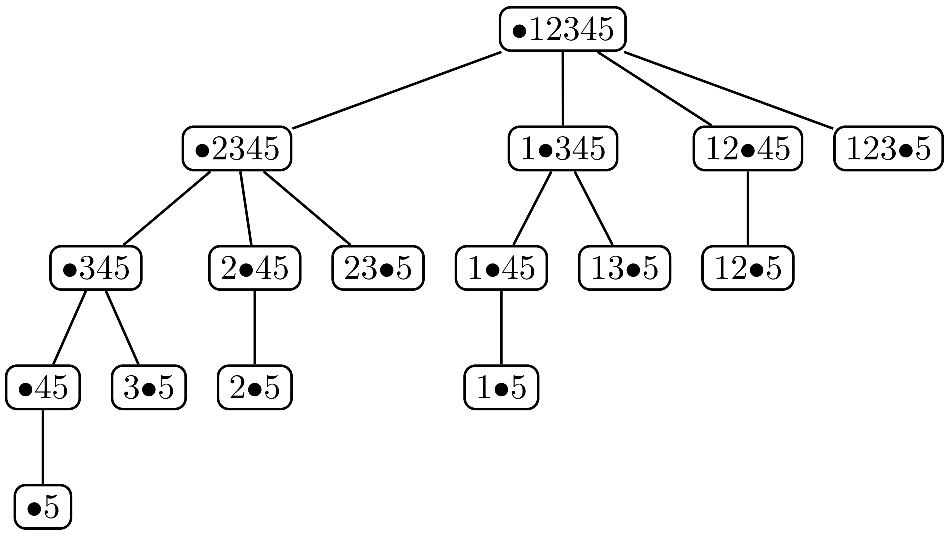

Figure 1: All-subsets regression tree for $N = 5$ variables.

nodes $(S - \{s_j\}, j-1)$ for $j = n-1, \ldots, k+1$. The reported RSS is stored in a subset table r along with its corresponding subset of variables. The entry r_n corresponds to the RSS of the best subset model with n variables found so far (Gatu and Kontoghiorghes 2006).

The algorithm employs a branch-and-bound strategy to reduce the number of generated nodes by cutting subtrees which do not contribute to the best solution. A cutting test is employed to determine which parts of the tree are redundant. That is, a new node $(S - \{s_j\}, j-1)$ is generated only if $\text{RSS}(S) < r_j$ $(j = k+1, \ldots, n-1)$. The quantity $\text{RSS}(S)$ is said to be the *bound* of the subtree rooted in (S, k); that is, no subset model extracted from the subtree can have a lower RSS (Gatu and Kontoghiorghes 2006).

The search can be restricted to find only the submodels of size within a specified range. This yields in spanning only a part of the regression tree in Figure 1, and therefore it requires a reduced computational cost.

In order to encourage the cutting of large subtrees, the regression tree is generated such that large subtrees have greater bounds. The algorithm achieves this by preordering the variables. Computing the bounds of the variables implies a computational overhead. Thus, it is not advisable to preorder the variables in all the nodes. A parameter — the preordering radius p $(0 \leq p \leq N)$ — defines the extent to which variables are preordered (Gatu and Kontoghiorghes 2006; Hofmann *et al.* 2007).

The efficiency of the branch-and-bound strategy is improved by allowing the algorithm to prune non-redundant portions of the regression tree. Thus, the cutting test is relaxed by employing a tolerance parameter $\tau_n \geq 0$ $(n = 1, \ldots, N)$. A node $(S - \{s_j\}, j-1)$ is generated only if there exists at least one i such that $(1 + \tau_i) \cdot \text{RSS}(S) < r_i$ $(i = j, \ldots, n-1)$. The algorithm is non-exhaustive if $\tau_n > 0$ for any n, meaning that the computed solution is not guaranteed to be optimal. The algorithm cuts subtrees more aggressively the greater the value of τ_n; the relative error of the solution is bounded by the employed tolerance (Gatu and Kontoghiorghes 2006; Hofmann *et al.* 2007).

The algorithm reports the N subset models with the lowest RSS, one for each subset size.

Problem	Function	Description
All-subsets regression	`lmSubsets()`	Standard formula interface
	`lmSubsets.fit()`	Workhorse function
Best-subsets regression	`lmSelect()`	Standard formula interface
	`lmSelect.fit()`	Workhorse function
	`lmSubsets.select()`	Conversion function

Table 1: Generator functions.

The user can then analyze the list of returned subsets to determine the "best" subset, e.g., by evaluating some criterion function. This approach is practical but not necessarily efficient. The algorithm may be optimized for a particular criterion f under the condition that the latter may be expressed as a function of the subset size n and the RSS ρ, i.e., $f(n, \rho)$, and that f is monotonic with respect to both n and ρ. It takes a single tolerance value and returns a single solution, that is the overall (i.e., over all subset sizes) best subset model according to criterion function f.

3. Implementation in R

The package provides two generator functions, `lmSubsets()` and `lmSelect()`, for solving the all-subsets and the best-subsets regression problems, respectively. That is, `lmSubsets()` computes the best submodel in terms of deviance for each subset size, while `lmSelect()` computes the overall best submodels according to a statistical criterion of the AIC family.

The generator functions are closely modeled after the `lm()` function and implement R's standard `formula` interface: they can be called with any entity that can be coerced to a `formula` object. After the `formula` instance has been resolved, the raw data and problem-specific parameters are forwarded to the appropriate core function. The workhorse functions `lmSubsets.fit()` and `lmSelect.fit()` implement the core computational logic without any `formula`-related overhead. An S3 object of class "`lmSubsets`" or "`lmSelect`" is returned, for which several standard extractor methods have been defined. Furthermore, "`lmSubsets`" objects can be coerced to "`lmSelect`" objects by means of the `lmSubsets.select()` function. An overview of the various functions is given in Table 1.

3.1. Specifying the problem

The user must first specify a linear model and provide a dataset from which the variables are taken. To perform *best*-subset regression, a *penalty* per parameter must be indicated.

The initial model is typically given in the form of a symbolic description. A `formula` object specifies the dependent and independent variables, which are taken from a `data.frame` object. For example, the call

```
lmSubsets(mortality ~ precipitation + temperature1 + temperature7 + age +
  household + education + housing + population + noncauc + whitecollar +
  income + hydrocarbon + nox + so2 + humidity, data = AirPollution)
```

specifies a response variable (`mortality`) and fifteen predictor variables; all variables are taken from the data frame `AirPollution` (Miller 2002). In this case, the call can be abbreviated to

```
lmSubsets(mortality ~ ., data = AirPollution)
```

where the dot (`.`) stands for "all variables not mentioned in the left-hand side of the formula". By default, an intercept term is included in the model; that is, the previous example is equivalent to

```
lmSubsets(mortality ~ . + 1, data = AirPollution)
```

To discard the intercept, rewrite the call as follows:

```
lmSubsets(mortality ~ . - 1, data = AirPollution)
```

Candidate models can be rejected based on the presence or absence of certain independent variables. To compute all subset models that contain a certain variable, the parameter `include` is employed. In the following example, only submodels containing the variable `noncauc` will be retained:

```
lmSubsets(mortality ~ ., include = "noncauc", data = AirPollution)
```

Conversely, the parameter `exclude` can be employed to discard submodels, as in the following example:

```
lmSubsets(mortality ~ ., exclude = "whitecollar", data = AirPollution)
```

The same effect can be achieved by rewriting the formula as follows:

```
lmSubsets(mortality ~ . - whitecollar, data = AirPollution)
```

The `include` and `exclude` parameters may be used in combination, and both may specify more than one variable (e.g., `include = c("noncauc", "whitecollar")`).

The criterion used for best-subset selection is evaluated following the expression

$$-2 \cdot \texttt{logLik} + \texttt{penalty} \cdot \texttt{npar},$$

where `penalty` is the *penalty per model-parameter*, `logLik` the log-likelihood of the fitted model, and `npar` the number of model-parameters. The `penalty` value indicates how strongly model parameters are penalized. A higher `penalty` will favor the selection of subset models with fewer regressors. When $\texttt{penalty} = 2$, the criterion corresponds to Akaike's information criterion (AIC, Akaike 1974); when $\texttt{penalty} = \log(\texttt{nobs})$, to Schwarz's Bayesian information criterion (BIC, Schwarz 1978), where `nobs` is the number of observations. For example,

```
lmSelect(mortality ~ ., data = AirPollution, penalty = 2)
```

will select the best submodels according to the usual AIC; or, in a more idiomatic style:

```
lmSelect(mortality ~ ., data = AirPollution, penalty = "AIC")
```

Parameter	Description	Representation
`x`	Data matrix	`numeric[nobs,nvar]`
`y`	Response variable	`numeric[nobs]`
`weights`	Model weights	`numeric[nobs]`
`offset`	Model offset	`numeric[nvar]`
`include`	Regressors to force in	`logical[nvar]`
`exclude`	Regressors to force out	`logical[nvar]`
`nmin`	Min. number of regressors (`lmSubsets()` only)	`integer[1]`
`nmax`	Max. number of regressors (`lmSubsets()` only)	`integer[1]`
`tolerance`	BBA tolerance parameters	`numeric[nvar]` (`lmSubsets()`)
		`numeric[1]` (`lmSelect()`)
`pradius`	Preordering radius	`integer[1]`
`nbest`	Number of best subsets	`integer[1]`
`.algo`	Algorithm to execute	`character[1]`

Table 2: Core parameters.

By default, the `lmSelect()` function employs the BIC.

3.2. Core functions

The generator functions process the formula interface to extract the model data, and pass the data — along with any problem specific arguments — to the core functions. The core functions act as wrappers for the C++ library which implements the problem logic. The full signature of the core functions is given by:

```
lmSubsets.fit(x, y, weights = NULL, offset = NULL,
  include = NULL, exclude = NULL, nmin = NULL,
  nmax = NULL, tolerance = 0, pradius = NULL, nbest = 1, ...,
  .algo = "hpbba")

lmSelect.fit(x, y, weights = NULL, offset = NULL,
  include = NULL, exclude = NULL, penalty = "BIC",
  tolerance = 0, pradius = NULL, nbest = 1, ...,
  .algo = "hpbba")
```

The parameters are summarized in Table 2. The model data is given in the form of a numeric matrix `x` and of a numeric vector `y`. The `weights` and `offset` parameters correspond to the homonomic parameters of the `lm` function.

The `include` and `exclude` parameters allow the caller to specify variables that are to be included in or excluded from all candidate models. They are logical vectors – with each entry corresponding to one variable – and are automatically expanded if given in the form of an integer vector (i.e., set of variable indices) or character vector (i.e., set of variable names as shown above).

For a large number of variables (see Section 5), execution times become intractable. In order to speed up the execution, either the search space can be reduced, or one may settle for a non-exhaustive solution. The caller can specify values for the `nmin` and `nmax` parameters,

Component	Description	Representation
`nobs`	Number of observations	`integer[1]`
`nvar`	Number of regressors	`integer[1]`
`weights`	Weights used	`numeric[nobs]`
`offset`	Offset used	`numeric[nobs]`
`intercept`	Intercept flag	`logical[1]`
`include`	Regressors forced in	`logical[nvar]`
`exclude`	Regressors forced out	`logical[nvar]`
`penalty`	Penalty used ("`lmSelect`" only)	`numeric[nvar]`
`tolerance`	Tolerances used	`numeric[nvar]`
`nbest`	Number of best subsets	`integer[1]`
`df`	Degrees of freedom	`integer[nbest,nvar]` (for "`lmSubsets`")
		`integer[nbest]` (for "`lmSelect`")
`rss`	Residual sum of squares	`numeric[nbest,nvar]` (for "`lmSubsets`")
		`numeric[nbest]` (for "`lmSelect`")
`which`	Selected regressors	`logical[nvar,nbest,nvar]` (for "`lmSubsets`")
		`logical[nvar,nbest]` (for "`lmSelect`")

Table 3: Components of "`lmSubsets`" and "`lmSelect`" objects.

in which case submodels with less than `nmin` or more than `nmax` variables are discarded, effectively stripping entire portions from the search space.

The second approach implies that the expectations with respect to the solution quality are lowered, i.e., that non-optimal solutions are *tolerated.* This is indicated by passing an argument to the `tolerance` parameter, which will be used by the BBA cutting test. A tolerance value typically lies between 0 and 1. The solution produced by the algorithm satisfies the following relationship:

$$f(S) \leq (1 + \texttt{tolerance}) \cdot f(S^*),$$

where S is the returned solution, S^* the optimal (theoretical) solution, and f the value of a submodel (i.e., deviance, AIC). The `lmSubsets()` routine accepts a vector of tolerances, with one entry for each subset size.

The `nbest` parameter controls how many submodels are returned. In the case of `lmSubsets()`, `nbest` subsets are returned for each subset size; in the case of `lmSelect()`, a total of `nbest` submodels is returned.

The `pradius` parameter serves to specify the desired preordering radius. The algorithm employs a default value of $\lfloor \texttt{nvar}/3 \rfloor$, and the need to set this parameter directly should rarely arise. The `.algo` parameter serves to specify the computational algorithm to be employed: this parameter is used for testing purposes only and should never be set directly.

3.3. Extracting submodels

The generator functions produce objects of class "`lmSubsets`" and "`lmSelect`", respectively, which implement the solution to a particular instance of a variable selection problem. They encapsulate the computed subset models, as well as problem-specific data that characterize the problem instance that was solved. The object components are summarized in Table 3.

The solution of an all-subsets regression problem is returned as an object of class "`lmSubsets`",

Method	Description
`print()`	Print object
`plot()`	Plot object
`summary()`	Summary statistics
`variable.names()`	Extract variables names
`formula()`	Extract formula object
`model.frame()`	Extract (full) model frame
`model.matrix()`	Extract model matrix
`refit()`	Fit subset model
`coef()`	Extract regression coefficients
`vcov()`	Extract covariance matrix
`fitted()`	Extract fitted values
`residuals()`	Extract residual values
`deviance()`	Extract deviance (RSS)
`logLik()`	Extract log-likelihood
`AIC()`	Extract AIC values
`BIC()`	Extract BIC values

Table 4: S3 methods for "`lmSubsets`" and "`lmSelect`" objects.

which logically represents a `nbest` $\times$ `nvar` table of subset models. Information pertaining to the subset models can be found in the components `df`, `rss`, and `which`. Similarly, the solution of a best-subsets regression problem is returned as an object of class "`lmSelect`" representing a sequence of `nbest` subset models stored in decreasing order of quality.

A wide range of standard methods to visualize, summarize, and extract information is provided (see Table 4). The `print()`, `plot()`, and `summary()` methods all report information about all the models found in the search – especially for "`lmSubsets`" objects this can help to select suitable models from all subsets. The remaining extractor functions (except for `refit()`) all return the kind of information as they would for a fitted "`lm`" object, e.g., the estimated coefficients and corresponding variance-covariance matrix etc. For "`lmSubsets`" objects the `size` of the desired subset needs to be provided and then by default the best model (`best = 1`) for that subset size is employed. Moreover, `best` can be set to higher values to obtain the information for the second best and third best model for a given size, etc. For "`lmSelect`" objects only `best` but not `size` can be modified.

4. Case study

`TODO: by Ana`

5. Benchmark

`TODO: by X`

Acknowledgments

This work was in part supported by the *Förderverein des wirtschaftswissenschaftlichen Zentrums der Universität Basel*, through research project B-123.

References

Akaike H (1974). "A new look at the statistical model identification." *Automatic Control, IEEE Transactions on*, **19**(6), 716–723.

Calcagno V, de Mazancourt C (2010). "**glmulti**: An R package for easy automated model selection with (generalized) linear models." *Journal of Statistical Software*, **34**(12), 1–29. doi:10.18637/jss.v034.i12.

Duarte Silva AP (2001). "Efficient variable screening for multivariate analysis." *Journal of Multivariate Analysis*, **76**, 35–62. doi:10.1006/jmva.2000.19202.

Friedman J, Hastie T, Tibshirani R (2010). "Regularization paths for generalized linear models via coordinate descent." *Journal of Statistical Software*, **33**(1), 1–22. doi:10.18637/jss.v033.i01.

Gatu C, Kontoghiorghes EJ (2006). "Branch-and-bound algorithms for computing the best subset regression models." *Journal of Computational and Graphical Statistics*, **15**, 139–156. doi:10.1198/106186006x100290.

Hastie TJ, Tibshirani RJ, Friedman J (2001). *The elements of statistical learning. Data mining, inference, and prediction.* Springer series in statistics. Springer-Verlag, New York.

Hofmann M, Gatu C, Kontoghiorghes EJ (2007). "Efficient algorithms for computing the best subset regression models for large-scale problems." *Computational Statistics and Data Analysis*, **52**, 16–29. doi:10.1016/j.csda.2007.03.017.

Hofmann M, Gatu C, Kontoghiorghes EJ, Zeileis A (2016). ***lmSubsets**: Variable Subset Selection in Linear Regressions.* R package version 0.1-0, URL https://CRAN.R-project.org/package=lmSubsets.

Lumley T, Miller A (2009). ***leaps**: Regression Subset Selection.* R package version 2.9, URL https://CRAN.R-project.org/package=leaps.

McLeod AI, Xu C (2014). ***bestglm**: Best Subset GLM.* R package version 0.34, URL https://CRAN.R-project.org/package=bestglm.

Miller AJ (2002). *Subset selection in regression*, volume 95 of *Monographs on statistics and applied probability.* 2nd edition. Chapman and Hall.

Orestes Cerdeira J, Duarte Silva AP, Cadima J, Minhoto M (2015). ***subselect**: Selecting variable subsets.* R package version 0.12-5, URL https://CRAN.R-project.org/package=subselect.

R Core Team (2016). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL `https://www.R-project.org/`.

Schwarz G (1978). "Estimating the dimension of a model." *The Annals of Statistics*, **6**(2), 461–464.

Wolters MA (2015). "A genetic algorithm for selection of fixed-size subsets with application to design problems." *Journal of Statistical Software, Code Snippets*, **68**(1), 1–18. `doi:10.18637/jss.v068.c01`.

Affiliation:

Marc Hofmann
`TODO: address`