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

Marc Hofmann
University of Oviedo, Spain
Cyprus University of Technology, Cyprus

Cristian Gatu
"Alexandru Ioan Cuza"
University of Iasi, Romania

Erricos J. Kontoghiorghes
Cyprus University of Technology, Cyprus
Birkbeck University of London, UK

Ana Colubi
University of Oviedo, Spain

Achim Zeileis
University of Innsbruck, Austria

Abstract

An R package for computing the all-subsets regression problem is presented. The proposed algorithms are based on computational strategies recently developed. A novel algorithm for the best-subset regression problem selects subset models based on a predetermined criterion. The package user can choose from exact and from approximation algorithms. The core of the package is written in C++ and provides an efficient implementation of all the underlying numerical computations. A case study and benchmark results illustrate the usage and the computational efficiency of the package.

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 (Clarke 1981; Hastie *et al.* 2001). Given a set of possible variables to be included in the regression, the problem consists in selecting a subset that optimizes some statistical criterion. The evaluation of the criterion function typically involves the estimation of the corresponding submodel (Miller 2002). Consider the standard regression model

$$y = X\beta + \epsilon, \tag{1}$$

where $y \in \mathbb{R}^M$ is the output variable, $X \in \mathbb{R}^{M\times N}$ is the regressor matrix of full column rank, $\beta \in \mathbb{R}^N$ is the coefficient vector, and $\epsilon \in \mathbb{R}^M$ is the noise vector. The ordinary least squares (OLS) estimator of β is the solution of

$$\widehat{\beta}_{\text{OLS}} = \underset{\beta}{\operatorname{argmin}} \operatorname{RSS}(\beta), \tag{2}$$

where the residual sum of squares (RSS) of β is given by

$$\mathrm{RSS}(\beta) = \|y - X\beta\|_2^2. \tag{3}$$

That is, $\widehat{\beta}_{\mathrm{OLS}}$ minimizes the norm of the residual vector. The regression coefficients β do not need to be explicitly computed in order to determine the RSS, which can be obtained through numerically stable orthogonal matrix decomposition methods (Golub and Van Loan 1996).

Let $V = \{1, \ldots, N\}$ denote the set of all independent variables. A subset model (or submodel) is denoted by S, $S \subseteq V$. Given a criterion function f, the *best-subset selection problem* consists in solving

$$S^* = \operatorname*{argmin}_{S \subseteq V} f(S). \tag{4}$$

Here, the value $f(S) = F(n, \rho)$ is seen as a function of $n = |S|$ and $\rho = \mathrm{RSS}(S)$, the number of selected variables and the RSS of the OLS estimator for S, respectively. Furthermore, it is assumed that $f(S)$ is monotonic with respect to $\mathrm{RSS}(S)$ for fixed n, that is

$$\mathrm{RSS}(S_1) \leq \mathrm{RSS}(S_2) \Rightarrow f(S_1) \leq f(S_2), \quad \text{when} \quad |S_1| = |S_2|. \tag{5}$$

Common selection criteria exhibit this property, such as those belonging to the AIC family defined by the formula

$$\mathrm{AIC}_k = M + M \log 2\pi + M \log(\mathrm{RSS}/M) + k(n+1), \tag{6}$$

where the scalar k represents a *penalty per parameter* ($k > 0$). The usual AIC and BIC are obtained for $k = 2$ and $k = \log M$, respectively (Miller 2002). It follows that (4) is equivalent to

$$S^* = S^*_\nu, \quad \text{where} \quad \nu = \operatorname*{argmin}_n f(S^*_n)$$

and

$$S^*_n = \operatorname*{argmin}_{|S|=n} \mathrm{RSS}(S) \quad \text{for} \quad n = 1, \ldots, N. \tag{7}$$

Finding the solution to (7) is called the all-subsets selection problem. Thus, solving (4) can be seen as an indirect, two-stage procedure:

Stage 1 For each size n, find the subset S^*_n ($|S^*_n| = n$) with the smallest RSS.

Stage 2 Compute $f(S^*_n)$ for all n, and determine ν such that $f(S^*_\nu)$ is minimal.

Note that the computational strategy may be optimized for a specific selection criterion when solving the best-subset selection problem (4) directly, thus lowering the computational cost. On the other hand, by explicitly solving the all-subsets regression problem (7) once and for all (Stage 1), the list of all N submodels is made readily available for further exploration: evaluating multiple criterion functions (e.g., AIC and BIC), or conducting a more elaborate statistical inference, can be performed at a negligible cost (Stage 2). Thus, it can be advisable to adopt a two-stage approach within the scope of a broader and more thorough statistical investigation.

Brute-force (or exhaustive) search procedures that enumerate all possible subsets are often intractable even for a modest number of variables. Exact algorithms must employ techniques

to reduce the size of the search space – i.e., the number of enumerated subsets – in order to tackle larger problems. Heuristic algorithms renounce optimality in order to decrease execution times: they are designed for solving a problem more quickly, but make no guarantees on the quality of the solution produced; genetic algorithms and simulated annealing count among the well-known heuristic algorithms. The solution returned by an approximation algorithm, on the other hand, can be prroven to lie within well specified bounds of the optimum.

Several packages that deal with variable subset selection are available on the R platform. The package **leaps** (Lumley and Miller 2009) implements exact, non-exhaustive algorithms for subset regression based on Miller (2002). Exhaustive algorithms have been considered within the context of generalized linear models (package **bestglm**, McLeod and Xu 2014). The package **subselect** proposes simulated annealing algorithms based on the work of Duarte Silva (2001). Furthermore, genetic algorithms for generalized linear models have been implemented by Calcagno and de Mazancourt (2010, package **glmulti**) and Wolters (2015, package **kofnGA**). Non-exact algorithms for regularized estimation of parametric models with automatic variable selection performed by lasso or elastic net estimation for generalized linear models have been investigated by Friedman *et al.* (2010).

Here, the **lmSubsets** package (Hofmann *et al.* 2016) for exact variable-subset regression is presented. It offers methods for solving both the best-subset (4) and the all-subsets (7) selection problems. It implements the algorithms presented by Gatu and Kontoghiorghes (2006) and Hofmann *et al.* (2007). A branch-and-bound strategy is employed to reduce the size of the search space. A similar approach has been employed for exact least-trimmed-squares regression Hofmann *et al.* (2010). The package further proposes approximation methods that compute non-exact solutions very quickly while giving guarantees on the quality of the result. The core of the package is written in C++. The package is available 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`.

Section 2 reviews the theoretical background and the underlying algorithms. The package's R interface is presented in Section 3. A usage example is given in Section 4, while benchmark results are illustrated in Section 5.

2. Computational strategies

The linear regression model (1) has 2^N possible subset models which can be efficiently organized in a regression tree. A dropping column algorithm (DCA) was devised as a straightforward approach to solve the all-subsets selection problem (7). The DCA evaluates all possible variable subsets by traversing a regression tree consisting of $2^{(N-1)}$ nodes (Gatu and Kontoghiorghes 2003; Gatu *et al.* 2007; Smith and Bremner 1989).

Each node of the regression tree can be represented by a pair (S, k), where $S = \{s_1, \ldots, s_n\}$ corresponds to a subset of n variables, $n = 0, \ldots, N$, and $k = 0, \ldots, n-1$. The subleading models are defined as $\{s_1, \ldots, s_{k+1}\}, \ldots, \{s_1, \ldots, s_n\}$, the RSS of which are computed for each visited node. The root node $(V, 0)$ corresponds to the full model. Child nodes are generated by dropping (deleting) a single variable:

$$\mathsf{drop}(S, j) = (S \setminus \{s_j\}, j-1), \quad j = k+1, \ldots, n-1.$$

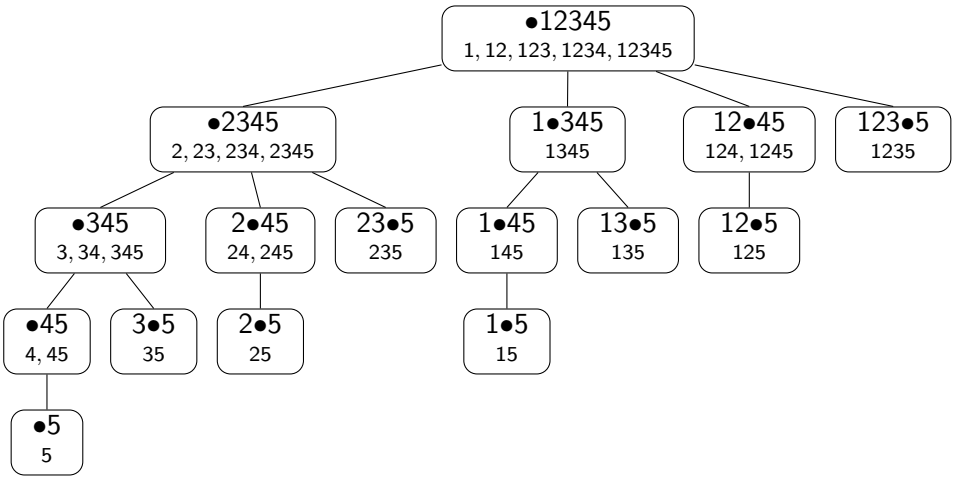

Figure 1: All-subsets regression tree for $N = 5$ variables. Nodes are shown together with their subleading models.

Numerically, this is equivalent to downdating an orthogonal matrix decomposition after a column has been deleted (Golub and Van Loan 1996; Kontoghiorghes 2000; Smith and Bremner 1989). Givens rotations are employed to efficiently move from one node to another. The DCA maintains a subset table r with N entries, where entry r_n contains the RSS of the current-best submodel of size n (Gatu and Kontoghiorghes 2006; Hofmann *et al.* 2007). Figure 1 illustrates a regression tree for $N = 5$ variables. The index k is symbolized by a bullet (•). The subleading models are listed in each node.

The DCA is computationally demanding, with a theoretical time complexity of $O(2^N)$. A branch-and-bound algorithm (BBA) has been devised to reduce the number of generated nodes by cutting subtrees which do not contribute to the current-best solution. It relies on the fundamental property that the RSS increases when variables are deleted from a regression model, that is:

$$S_1 \subseteq S_2 \Rightarrow \mathrm{RSS}(S_1) \geq \mathrm{RSS}(S_2).$$

A cutting test is employed to determine which parts of the DCA tree are redundant: A new node $\mathsf{drop}(S, j)$ is generated only if $\mathrm{RSS}(S) < r_j$ $(j = k+1, \ldots, n-1)$. The quantity $\mathrm{RSS}(S)$ is called the bound of the subtree rooted in (S, k): no subset model extracted from the subtree can have a smaller RSS (Gatu and Kontoghiorghes 2006). Note that the BBA is an exact algorithm, i.e., it computes the optimal solution of the all-subsets regression problem (7).

To further reduce the computational cost, the all-subsets regression problem can be restricted to a range of submodel sizes (Hofmann *et al.* 2007). In this case, the problem (7) is reformulated as

$$S_n^* = \underset{|S|=n}{\operatorname{argmin}} \mathrm{RSS}(S) \quad \text{for} \quad n = n_{\min}, \ldots, n_{\max}, \tag{8}$$

where $n_{\min}$ and $n_{\max}$ are the subrange limits $(1 \leq n_{\min} \leq n_{\max} \leq N)$. The search will span only a part of the DCA regression tree. Specifically, nodes (S, k) are not computed if $|S| < n_{\min}$ or $k \geq n_{\max}$.

The size of subtrees rooted in the same level decreases exponentially from left to right. In order to encourage the pruning of large subtrees by the BBA cutting test, the variables in a

given node can be ordered such that a child node will always have a larger RSS (i.e., bound) than its right siblings (Gatu and Kontoghiorghes 2006). This strategy can be applied in nodes of arbitrary depth. However, computing the variable bounds incurs a computational overhead. Thus, it is not advisable to indiscriminately preorder variables. A parameter – the preordering radius p – has been introduced to control the degree of preordering (Hofmann *et al.* 2007). It accepts a value between $p = 0$ (no preordering) and $p = N$ (preordering in all nodes); when $p = 1$, preordering is performed in the root node only. Typically, $p = \lfloor N/3 \rfloor$ produces better results in terms of execution time.

The computational efficiency of the BBA is improved by allowing the algorithm to prune non-redundant branches of the regression tree. The approximation branch-and-bound algorithm (ABBA) relaxes the cutting test by employing a set of tolerance parameters $\tau_n \geq 0$ ($n = 1, \ldots, N$), one for every submodel size. A node $\mathsf{drop}(S, j)$ is generated only if there exists at least one i such that

$$(1 + \tau_i) \cdot (\mathrm{RSS}(S) - \mathrm{RSS}_{\mathrm{full}}) < (r_i - \mathrm{RSS}_{\mathrm{full}}), \quad i = j, \ldots, n - 1, \tag{9}$$

where $\mathrm{RSS}_{\mathrm{full}} = \mathrm{RSS}(V)$ is the RSS of the full model. The algorithm is non-exact if $\tau_n > 0$ for any n, meaning that the computed solution is not guaranteed to be optimal. The greater the value of τ_n, the more aggressively the regression tree will be pruned, thus decreasing the computational load. The advantage of the ABBA over heuristic algorithms is that the relative error of the solution is bounded by the tolerance parameter (Gatu and Kontoghiorghes 2006; Hofmann *et al.* 2007), thus giving the user control over the tradeoff between solution quality and speed of execution.

The DCA and its derivatives report the N subset models with the lowest RSS, one for each subset size. The user can then analyze the list of returned subsets to determine the "best" subset, for example by evaluating some criterion function. This approach is practical but not necessarily the most efficient. Let f be a criterion function such that $f(S) = F(n, \rho)$, where $n = |S|$ and $\rho = \mathrm{RSS}(S)$, satisfying the monotonicity property (5). The f-BBA specializes the standard cutting test for f under the additional condition that F is non-decreasing in n. Specifically, a node $\mathsf{drop}(S, j)$ is generated if and only if

$$F(j, \mathrm{RSS}(S)) < r_f,$$

where r_f is the single current-best solution. This results in a more "informed" cutting test, and in a smaller number of generated nodes.

3. Implementation in R

The R package **lmSubsets** provides a library of methods for variable subset selection in linear regression. Two S3 classes are defined, namely "`lmSubsets`" and "`lmSelect`", that address all-subsets (7) and best-subset (4) selection, respectively. The package offers R's standard formula interface: linear models can be specified by means of a symbolic formula, and possibly a data frame. The model specification is resolved into a regressor matrix and a response vector, which are forwarded to low-level functions for actual processing, together with optional arguments that serve to further specify the selection problem. Routines to extract the best subsets from an all-subsets regression solution (i.e., to convert an "`lmSubsets`" to an "`lmSelect`" object) are also provided. An overview of the package structure is given in Table 1.

S3 class	Methods and functions	Description
"lmSubsets"	lmSubsets()	all-subsets selection (generic function)
	lmSubsets.matrix()	matrix interface
	lmSubsets.default()	standard formula interface
	lmSubsets_fit()	low-level function
"lmSelect"	lmSelect()	best-subset selection (generic function)
	lmSelect.lmSubsets()	conversion method
	lmSelect.matrix()	matrix interface
	lmSelect.default()	standard formula interface
	lmSelect_fit()	low-level function

Table 1: Package structure.

3.1. Specifying the selection problem

The default methods are closely modeled after R's standard `lm()` function: they can be called with any entity that can be coerced to a `formula` object (Chambers and Hastie 1992). The `formula` object declares the dependent and independent variables, which are typically taken from a `data.frame` specified by the user. 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 taken from the `AirPollution` dataset (Miller 2002). It is common to shorten the call by emloying R's practical "dot-notation":

```
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 call in the previous example is equivalent to

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

To discard the intercept, the call may be rewritten as follows:

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

Submodels can be rejected based on the presence or absence of certain independent variables. The parameter `include` specifies that all submodels must contain one or several variables. In the following example, only submodels containing the variable `noncauc` are retained:

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

Conversely, the `exclude` parameter can be employed to discard a specific set of variables, 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 defined in (6), `logLik` the log-likelihood of the fitted model, and `npar` the number of model parameters (including the error variance). The `penalty` value indicates how strongly model parameters are penalized, with large values favoring parsimonious models. When `penalty` $= 2$, the criterion corresponds to Akaike's information criterion (AIC, Akaike 1974); when `penalty` $= \log($`nobs`$)$, to Schwarz's Bayesian information criterion (BIC, Schwarz 1978), where `nobs` is the number of observations. For example, either one of

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

and

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

will select the best submodel according to the usual AIC; by default, `lmSelect()` employs the BIC. The user may also specify a custom criterion function

```
lmSelect(..., penalty = function (size, rss) ...),
```

where `size` is the number of regressors, and `rss` the residual sum of squares of the corresponding submodel. The user-specified function must be non-decreasing in both parameters.

3.2. Core functions

The high-level interface methods process the model specification before dispatching the call to one of two low-level core functions, passing along a regressor matrix `x` and a response vector `y`, together with problem-specific arguments. The core functions act as wrappers around the C++ library, and are declared as

```
lmSubsets_fit(x, y, weights = NULL, offset = NULL, include = NULL,
  exclude = NULL, nmin = NULL, nmax = NULL, tolerance = 0,
  nbest = 1, ..., pradius = NULL)
```

and

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

The parameters are summarized in Table 2.

The `weights` and `offset` parameters correspond to the homonymous parameters of the `lm()` function. The `include` and `exclude` parameters allow the user to specify variables that are to be included in, or excluded from all candidate models. They are either logical vectors – with each entry corresponding to one variable – or 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).

For a large number of variables (see Section 5), execution times may become prohibitive. In order to speed up the execution, either the search space can be reduced, or one may settle for a non-exact solution. In the first approach, the user may specify values for the `nmin` and `nmax` parameters as defined in (8), in which case submodels with less than `nmin` or more than `nmax` variables are discarded. Well-defined regions of the regression tree can be ignored by the selection algorithm, and the search space is thus reduced.

In the second approach, expectations with respect to the solution quality are lowered, i.e., non-optimal solutions are tolerated. The numeric value – typically between 0 and 1 – passed as the `tolerance` argument indicates the degree of over-pruning performed by the ABBA cutting test (9). The solution produced by the ABBA satisfies the following relationship:

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

where S is the returned solution, V the full model, S^* the optimal (theoretical) solution, and f the cost of a submodel (e.g., deviance, AIC). The `lmSubsets_fit()` function accepts a vector of tolerances, with one entry for each subset size.

The `nbest` parameter controls how many submodels (per subset size) are retained. In the case of `lmSubsets_fit()`, a two-dimensional result set is constructed with `nbest` submodels for each subset size, while in the case of `lmSelect_fit()`, a one-dimensional sequence of `nbest` submodels is handed back to the user.

The `pradius` parameter serves to specify the desired preordering radius. The algorithm employs a default value of $\lfloor \texttt{nvar}/3 \rfloor$. The need to set this parameter directly should rarely arise; please refer to Section 2 for further information.

3.3. Extracting submodels

The user is handed back a result object that encapsulates the solution to an all-subsets (class "`lmSubsets`") or best-subset (class "`lmSelect`") selection problem. An object of class "`lmSubsets`" represents a two-dimensional `nbest` × `nvar` set of submodels; an object of class "`lmSelect`", a linear sequence of `nbest` submodels. Problem-specific information is stored alongside the selected submodels. Table 3 summarizes the components of the result objects.

A wide range of standard methods to visualize, summarize, and extract information are provided (see Table 4). The `print()`, `plot()`, and `summary()` methods give the user a compact overview – either textual or graphical – of the information gathered on the selected submodels in order to help identify "good" candidates. The remaining extractor functions

Parameter	Description	Canonical 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	`integer[1]`	for `lmSubsets()` only
`nmax`	max. number of regressors	`integer[1]`	for `lmSubsets()` only
`penalty`	penalty per parameter	`numeric`	for `lmSelect()` only
	or criterion function	`function`	
`tolerance`	BBA tolerance parameters	`numeric[nvar]`	for `lmSubsets()`
		`numeric[1]`	for `lmSelect()`
`nbest`	number of best subsets	`integer[1]`	
`pradius`	preordering radius	`integer[1]`	

Table 2: Core parameters.

Component	Description	Canonical representation	
`nobs`	number of observations	`integer[1]`	
`nvar`	number of regressors	`integer[1]`	
`intercept`	intercept flag	`logical[1]`	
`include`	regressors forced in	`logical[nvar]`	
`exclude`	regressors forced out	`logical[nvar]`	
`sizes`	covered subset sizes	`numeric[]`	
`tolerance`	tolerances used	`numeric[nvar]`	
`nbest`	number of best subsets	`integer[1]`	
`rank`	submodel rank	`integer[nbest,nvar]`	for "`lmSubsets`"
		`integer[nbest]`	for "`lmSelect`"
`df_residual`	residual deg. of freedom	`integer[nbest,nvar]`	for "`lmSubsets`"
		`integer[nbest]`	for "`lmSelect`"
`rss`	residual sum of squares	`numeric[nbest,nvar]`	for "`lmSubsets`" only
`penalty`	subset penalties	`numeric[nbest]`	for "`lmSelect`" only
`which`	selected regressors	`logical[nvar,nbest,nvar]`	for "`lmSubsets`"
		`logical[nvar,nbest]`	for "`lmSelect`"

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

behave in the usual way, and can be used to extract variable names, coefficients, covariance matrices, fitted values, etc.

In order to designate a submodel, "`lmSubsets`" methods provide the parameters `size` and `best` to specify the number of regressors in and the ranking of the desired submodel, respectively. The user must always indicate the desired `size`, while `best` defaults to `1`. For "`lmSelect`" methods, the `size` parameter has no meaning and is not defined. Lastly, methods that return scalar values – i.e., `deviance()`, `log_lik()`, `aic()`, `bic()` – can extract more than one submodel at a time if passed a numeric vector as an argument to either `size` (e.g., `size = 5:10`) or `best` (e.g., `best = 1:3`).

Method	Description
`format()`	format object
`print()`	print object
`plot()`	plot RSS or penalty
`image()`	heatmap of selected regressors
`summary()`	summary statistics
`variable.names()`	extract variables names
`formula()`	extract formula object
`model.frame()`	extract (full) model frame
`model.matrix()`	extract model matrix
`model_response()`	extract model response
`refit()`	fit sub-"`lm`"
`deviance()`	extract deviance (RSS)
`sigma()`	extract residual standard deviation
`log_lik()`	extract log-likelihood
`aic()`	extract AIC values
`bic()`	extract BIC values
`coef()`	extract regression coefficients
`vcov()`	extract covariance matrix
`fitted()`	extract fitted values
`residuals()`	extract residual values

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

4. Case study: Variable selection in weather forecasting

Statistical weather forecasting is a branch of objective weather forecasting – the other being numerical weather forecasting (NWP) –, and relies on variable-subset selection (also known as "screening regression") as an essential instrument. Model output statistics (MOS) is a multiple linear regression technique that builds a statistical relationship between a predictand and variables forecast by an NWP (dynamic) model at some projection time (Glahn and Lowry 1972). The output model is derived by relating past observations of the predictand with archived dynamic forecasts. Variable-subset selection is employed to determine which quantities forecast by the dynamic model enter the regression for a particular predictand and projection time.

Here, as an application example, **lmSubsets** is used to build an MOS model for the daily temperature at Innsbruck Airport (Austria), based on data provided by the Global Ensemble Forecast System (Hamill *et al.* 2013). The data frame `IbkTemperature` contains 1824 daily cases for 42 variables. The variables include the temperature at Innsbruck Airport (observed), as well as 36 NWP outputs (forecasted), among which quantities pertaining to temperature (two-meters-above-ground, minimum, maximum, and soil temperatures), precipitation, wind, pressure, radiative and heat fluxes. Terms for deterministic trend and seasonal components are provided for convenience. See `?IbkTemperature` for further details.

In order to commence the analysis, the dataset is loaded and cases with missing values are removed.

	MOS0		MOS1	
(Intercept)	-345.252**	(109.212)	-661.700***	(95.225)
t2m	0.318***	(0.016)		
time	0.132*	(0.054)	0.147**	(0.047)
sin	-1.234***	(0.126)	0.811***	(0.120)
cos	-6.329***	(0.164)		
sin2	0.240*	(0.110)	-0.870***	(0.118)
cos2	-0.332**	(0.109)	-1.128***	(0.097)
sshnf			0.018***	(0.004)
mslp			-0.000***	(0.000)
vsmc			20.181***	(3.106)
tmax2m			0.181***	(0.023)
st			1.142***	(0.043)
wr			0.505***	(0.103)
t2pvu			0.149***	(0.028)
p2pvu			-0.000**	(0.000)
AIC	9493.602		8948.182	
BIC	9537.650		9025.267	
Deviance	19506.469		14357.943	
Sigma	3.281		2.820	
R-squared	0.803		0.855	

Table 5: Estimated regression coefficients (along with standard errors) and summary statistics for the output models `MOS0` and `MOS1`.

```
R> data("IbkTemperature", package = "lmSubsets")
R> IbkTemperature <- na.omit(IbkTemperature)
```

A simple output model (`MOS0`) for the observed temperature (`temp`) is constructed, which will serve as a reference model. It consists of the temperature forecast by the dynamic model (`t2m`), a linear trend component (`time`), as well as annual (`sin`, `cos`) and bi-annual (`sin2`, `cos2`) seasonal components.

```
R> MOS0 <- lm(temp ~ t2m + time + sin + cos + sin2 + cos2,
+             data = IbkTemperature)
```

The estimated coefficients (and standard errors) are shown in Table 5. It can be observed that despite the inclusion of the dynamic variable `t2m`, the coefficients for the deterministic components remain significant, which indicates that the seasonal temperature fluctuations are not fully resolved by the dynamic model.

Next, the variable-subset models with the lowest RSS for all submodel sizes (2–42 regressors, including the intercept) are computed (`MOS1.all`). The "`lm`" object corresponding to the submodel with the lowest BIC score is extracted (`MOS1`).

```
R> MOS1.all <- lmSubsets(temp ~ ., data = IbkTemperature)
R> MOS1 <- refit(lmSelect(MOS1.all, penalty = "BIC"))
```

The RSS and BIC of the variable-subset models in `MOS1.all` are illustrated in Figure 2. The BIC-best model `MOS1` has 13 regressors, which are listed in Table 5. Eight dynamic variables not included in `MOS0` are selected, among which quantities relating to temperature (`tmax2m`, `st`, `t2pvu`), pressure (`mslp`, `p2pvu`), hydrology (`vsmc`, `wr`), and heat flux (`sshnf`). The deterministic components (linear trend, annual and bi-annual seasonal patterns) are

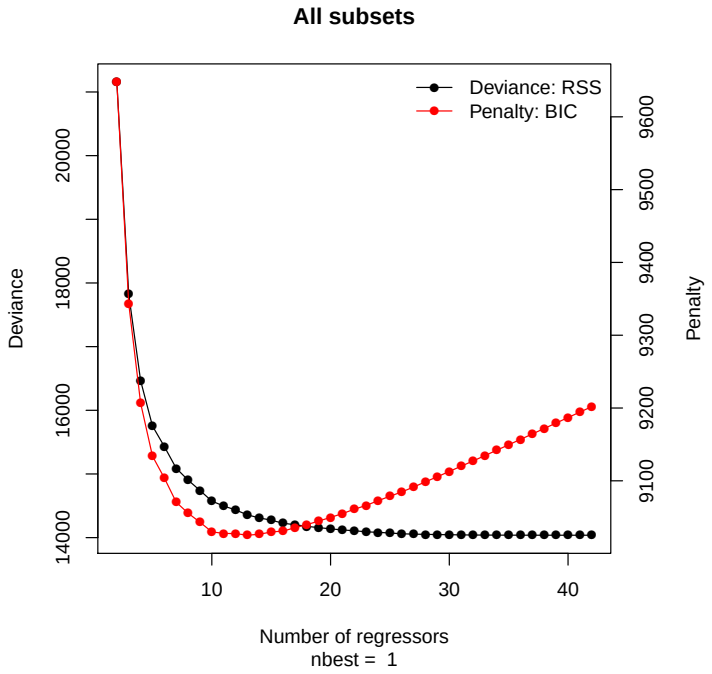

Figure 2: RSS and BIC for the best subset models.

present in `MOS1`. However, and most remarkably, `MOS1` does not include the temperature forecast by the dynamic model (`t2m`). The summary statistics reveal that `MOS1` significantly improves over the reference model `MOS0`.

The 20 overall best MOS subset models are computed (`MOS1.best`) and illustrated by a selection heatmap in Figure 3.

```
R> MOS1.best <- lmSelect(temp ~ ., data = IbkTemperature,
+                        penalty = "BIC", nbest = 20)
```

A selected variable is indicated by a dark cell. The number of selected variables per submodel ranges from 11 to 14. It is noticeable that `t2m` has not been picked up by any of the submodels. Per contra, the deterministic components are always present.

In summary, this case study illustrates how **lmSubsets** can be used in the context of MOS regression, to identify dynamic variables which improve the predictive skill of the output model. In a fully fledged meteorological application, the constructed model would have to be validated using cross-validation or other out-of-sample assessment techniques.

5. Benchmark

Datasets which contain a "true" model are simulated, with `nobs` observations and `nvar` independent variables. The dependent variable `y` is constructed from a linear combination of `ntrue` randomly selected independent variables, a noise vector, and the intercept. The `ntrue` independent variables are assumed to belong to the "true" model. The detailed procedure follows:

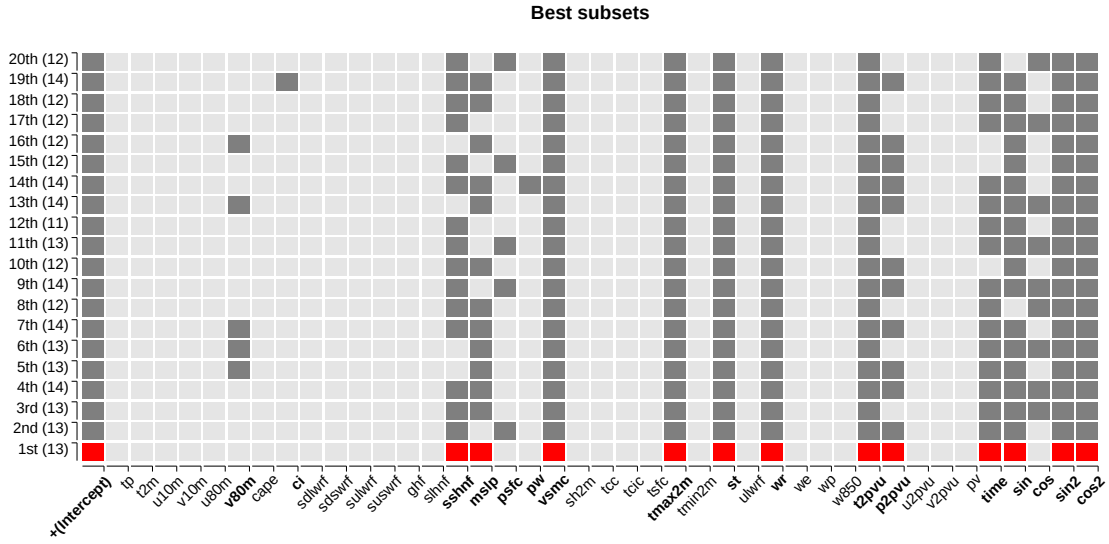

Figure 3: Subset selection for `MOS1.best`.

```
# INPUT PARAMETERS:  nobs, nvar, ntrue, sigma
# (A) DATA MATRIX:
X <- rnorm(nobs * nvar, sd = 1.0)
# (B) SELECTION VECTOR:
beta <- rep_len(0.0, length.out = nvar)
beta[sample.int(nvar, size = ntrue)] <- 1.0
# (C) NOISE VECTOR:
e <- rnorm(nobs, sd = sigma)
# (D) DEPENDENT VARIABLE:
y <- X %*% beta + e + 1.0
```

For the purpose of the benchmark, different data configurations are considered. The simulated datasets have `nobs` $= 1000$ observations, and `nvar` $= 20, 40, 60, 80$ independent variables. The "true" model contains `ntrue` $=$ `nvar`$/2$ variables, and the intercept. Five different levels of noise `sigma` $= 0.05, 0.10, 0.50, 1.00, 5.00$ are considered, giving rise to 20 different data configurations. For each configuration, five datasets are generated, for a total of 100 datasets. The `lmSubsets` and the `lmSelect` (with the default search criterion, i.e., BIC) algorithms are executed for each dataset, once with `tolerance` $= 0.0$ (exact), and once with `tolerance` $= 0.1$ (approximation). Minimum (`tmin`), maximum (`tmax`), and average (`tavg`) execution times (in seconds) for each configuration are reported in Table 6.

The solutions for datasets with 20 and 40 variables are obtained in less than 1 second, in all cases. For 60 variables, the `lmSubsets` procedure completes in 5 minutes or less (exact); the approximation algorithm performs about two times faster. For 80 variables, the computed subset sizes are restricted to between 30 and 50 variables; the average execution times are less then 20 minutes (exact) and 8 minutes (approximation). The `lmSelect` procedure completes in less than 1 second in all cases, with one exception: the worst and average case running times, respectively, for 80 variables with a high level of noise (`sigma` $= 5.00$), are 24 and 9

seconds (exact), and 9 and 4 seconds (approximation).

6. Conclusions

An R package for all-subsets variable selection is presented. It is based on threoretical strategies that have been recently developed. A novel algorithm for best-subset variable selection is proposed, which selects the best variable-subset model according to a pre-determined search criterion. It performs considerably faster than all-subsets variable selection algorithms that rely on the residual sum of squares only. Approximation algorithms allow to further increase the size of tackled datasets. The package implements R's standard formula interface. A case study is presented, and the performance of the package is illustrated in a benchmark with various configurations of simulated datasets.

Acknowledgements

`TODO: Oviedo project, IFMSE`

This work was in part supported by the CRoNoS COST Action (IC1408), the *Förderverein des wirtschaftswissenschaftlichen Zentrums der Universität Basel* through research project B-123, and the IFMSE. The authors are grateful to Jakob Messner for sharing the GEFS forecast data in `IbkTemperature`.

The authors would particularly like to thank Prof. Manfred Gilli for his continued support and encouragements throughout the years.

References

Akaike H (1974). "A New Look at the Statistical Model Identification." *IEEE Transactions on Automatic Control*, **19**(6), 716–723. doi:10.1109/tac.1974.1100705.

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.

Chambers JM, Hastie TJ (eds.) (1992). *Statistical Models in S*. Chapman & Hall, London.

Clarke MRB (1981). "Statistical algorithms: Algorithm AS 163: A Givens Algorithm for Moving from One Linear Model to Another without Going Back to the Data." *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, **30**(2), 198–203.

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 (2003). "Parallel Algorithms for Computing All Possible Subset Regression Models Using the QR Decomposition." *Parallel Computing*, **29**(4), 505–521.

sigma		nvar	nmin	nmax	tolerance = 0.0 tmin	tmax	tavg	tolerance = 0.1 tmin	tmax	tavg
0.05	lmSubsets	20	-	-	0.006	0.076	0.022	0.005	0.006	0.005
		40	-	-	0.302	0.379	0.338	0.216	0.251	0.232
		60	-	-	213.567	275.740	239.592	100.715	128.292	111.694
		80	30	50	784.364	1219.488	961.900	335.535	461.974	390.447
	lmSelect	20	-	-	0.004	0.072	0.018	0.004	0.005	0.005
		40	-	-	0.009	0.011	0.010	0.010	0.012	0.011
		60	-	-	0.015	0.018	0.017	0.016	0.018	0.017
		80	-	-	0.024	0.028	0.026	0.023	0.028	0.025
sigma		nvar	nmin	nmax	tolerance = 0.0 tmin	tmax	tavg	tolerance = 0.1 tmin	tmax	tavg
0.10	lmSubsets	20	-	-	0.005	0.010	0.007	0.005	0.006	0.006
		40	-	-	0.307	0.359	0.328	0.222	0.231	0.227
		60	-	-	179.847	298.430	230.644	87.746	132.034	106.971
		80	30	50	740.274	1321.999	1079.937	313.509	490.188	419.043
	lmSelect	20	-	-	0.004	0.005	0.005	0.004	0.005	0.005
		40	-	-	0.010	0.012	0.011	0.010	0.012	0.011
		60	-	-	0.016	0.018	0.017	0.015	0.018	0.017
		80	-	-	0.024	0.034	0.027	0.025	0.029	0.027
sigma		nvar	nmin	nmax	tolerance = 0.0 tmin	tmax	tavg	tolerance = 0.1 tmin	tmax	tavg
0.50	lmSubsets	20	-	-	0.005	0.010	0.008	0.005	0.005	0.005
		40	-	-	0.297	0.422	0.348	0.196	0.278	0.237
		60	-	-	167.291	303.223	218.706	84.314	135.472	103.185
		80	30	50	982.042	1351.623	1156.535	391.846	495.175	444.296
	lmSelect	20	-	-	0.004	0.005	0.005	0.004	0.005	0.005
		40	-	-	0.010	0.011	0.011	0.010	0.012	0.011
		60	-	-	0.016	0.018	0.017	0.015	0.018	0.017
		80	-	-	0.026	0.028	0.027	0.025	0.027	0.026
sigma		nvar	nmin	nmax	tolerance = 0.0 tmin	tmax	tavg	tolerance = 0.1 tmin	tmax	tavg
1.00	lmSubsets	20	-	-	0.005	0.010	0.007	0.005	0.006	0.005
		40	-	-	0.278	0.415	0.326	0.195	0.250	0.214
		60	-	-	202.200	286.419	255.339	96.492	129.877	117.297
		80	30	50	698.940	1018.715	851.472	291.820	404.323	348.461
	lmSelect	20	-	-	0.004	0.005	0.005	0.005	0.005	0.005
		40	-	-	0.009	0.011	0.010	0.011	0.011	0.011
		60	-	-	0.015	0.018	0.017	0.016	0.020	0.018
		80	-	-	0.025	0.028	0.026	0.025	0.027	0.026
sigma		nvar	nmin	nmax	tolerance = 0.0 tmin	tmax	tavg	tolerance = 0.1 tmin	tmax	tavg
5.00	lmSubsets	20	-	-	0.006	0.009	0.008	0.004	0.007	0.005
		40	-	-	0.153	0.290	0.193	0.120	0.191	0.143
		60	-	-	80.560	112.290	98.981	45.133	58.601	52.173
		80	30	50	70.523	1014.621	358.406	29.615	336.587	129.344
	lmSelect	20	-	-	0.004	0.005	0.005	0.004	0.005	0.004
		40	-	-	0.013	0.015	0.014	0.012	0.014	0.013
		60	-	-	0.098	0.289	0.199	0.062	0.156	0.111
		80	-	-	3.302	23.989	8.811	1.353	8.429	3.267

Table 6: Minimum, maximum, and average execution times (in seconds).

Gatu C, Kontoghiorghes EJ (2006). "Branch-and-Bound Algorithms for Computing the Best Subset Regression Models." *Journal of Computational & Graphical Statistics*, **15**, 139–156. doi:10.1198/106186006x100290.

Gatu C, Yanev P, Kontoghiorghes EJ (2007). "A Graph Approach to Generate All Possible Regression Sub-Models." *Computational Statistics and Data Analysis*, **52**(2), 799–815.

Glahn HR, Lowry DA (1972). "The Use of Model Output Statistics (MOS) in Objective Weather Forecasting." *Journal of Applied Meteorology*, **11**(8), 1203–1211. doi:10.1175/1520-0450(1972)011<1203:TUOMOS>2.0.CO;2.

Golub GH, Van Loan CF (1996). *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences, 3rd edition. Johns Hopkins University Press, Baltimore, Maryland.

Hamill TM, Bates GT, Whitaker JS, Murray DR, Fiorino M, Jr TJG, Zhu Y, Lapenta W (2013). "NOAA's Second-Generation Global Medium-Range Ensemble Reforecast Data Set." *Bulletin of the American Meteorological Society*, **94**(10), 1553–1565. doi:10.1175/BAMS-D-12-00014.1.

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 & Data Analysis*, **52**, 16–29. doi:10.1016/j.csda.2007.03.017.

Hofmann M, Gatu C, Kontoghiorghes EJ (2010). "An exact least-trimmed-squares algorithm for a range of coverage values." *Journal of Computational and Graphical Statistics*, **19**, 191–204.

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.

Kontoghiorghes EJ (2000). *Parallel Algorithms for Linear Models: Numerical Methods and Estimation Problems*, volume 15 of *Advances in Computational Economics*. Kluwer Academic Publishers, Boston. doi:10.1007/978-1-4615-4571-2.

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. doi:10.1201/9781420035933.

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. doi:10.1214/aos/1176344136.

Smith DM, Bremner JM (1989). "All Possible Subset Regressions Using the QR Decomposition." *Computational Statistics & Data Analysis*, **7**(3), 217–235. doi:10.1016/0167-9473(89)90023-6.

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
Institute of Natural Resources and Territorial Planning
University of Oviedo
33600 Mieres, Spain
E-mail: marc.hofmann@gmail.com
E-mail: marc.indurot@uniovi.es
URL: http://www.indurot.uniovi.es