Journal of Statistical Software

April 2020, Volume 93, Issue 3. doi: 10.18637/jss.v093.i03

lmSubsets: Exact Variable-Subset Selection in Linear Regression for R

Marc Hofmann
University of Oviedo
Cyprus University of Technology

Cristian Gatu
"Alexandru Ioan Cuza"
University of Iasi

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

Ana Colubi
University of Oviedo
Justus-Liebig-University Giessen

Achim Zeileis
Universität Innsbruck

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, Tibshirani, and Friedman 2009). 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

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

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

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

That is, $\hat{\beta}_{\text{OLS}}$ 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

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

Here, the value $f(S) = F(n, \rho)$ is seen as a function of $n = |S|$ and $\rho = \operatorname{RSS}(S)$, 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 $\operatorname{RSS}(S)$ for fixed n, that is

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

Common information criteria (IC) exhibit this property, such as those belonging to the AIC family and defined by the formula

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

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

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

and

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

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.

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 may be advisable to adopt a two-stage approach within the scope of a broader and more thorough statistical investigation. On the other hand, precursory knowledge of the search function and of its characteristics

opens up the possibility for a custom-tailored computational strategy to solve the best-subset selection problem (4) in one go; by exploiting more information about the problem at hand, the solution strategy will be rendered more efficient.

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 are among the well-known heuristic algorithms (Goldberg 1989; Otten and Van Ginneken 1989). The solution returned by an approximation algorithm, on the other hand, can be proven to lie within well specified bounds of the optimum.

Several packages that deal with variable subset selection are available for the R environment for statistical computing and graphics (R Core Team 2019). Package **leaps** (Lumley and Miller 2017) implements exact, exhaustive and non-exhaustive algorithms for subset selection in linear models (Miller 2002); it has been extended to generalized linear models by package **bestglm** (McLeod and Xu 2019). An active set algorithm for solving the best subset selection problem in generalized linear models is proposed by package **BeSS** (Wen, Zhang, Quan, and Wang 2019). Package **subselect** (Orestes Cerdeira, Duarte Silva, Cadima, and Minhoto 2018) proposes simulated annealing and genetic algorithms that search for subsets of variables which are optimal under various criteria. Package **glmulti** (Calcagno 2019) provides IC-based automated model selection methods for generalized linear models in the form of exhaustive and genetic algorithms. Package **kofnGA** (Wolters 2015) uses a genetic algorithm to choose a fixed-size subset under a user-supplied objective function. Procedures for regularized estimation of generalized linear models with elastic-net penalties are implemented in package **glmnet** (Friedman, Hastie, and Tibshirani 2010).

Here, the **lmSubsets** package (Hofmann, Gatu, Kontoghiorghes, Colubi, and Zeileis 2020) 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, Gatu, and Kontoghiorghes (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, Gatu, and Kontoghiorghes 2010). The package further proposes approximation methods that compute non-exact solutions very quickly: The exigencies toward solution quality are relaxed by means of a tolerance parameter that steers the permitted degree of error. The core of the package is written in C++. The package is available for the R environment for statistical computing and graphics from the Comprehensive R Archive Network (CRAN) 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 straight-

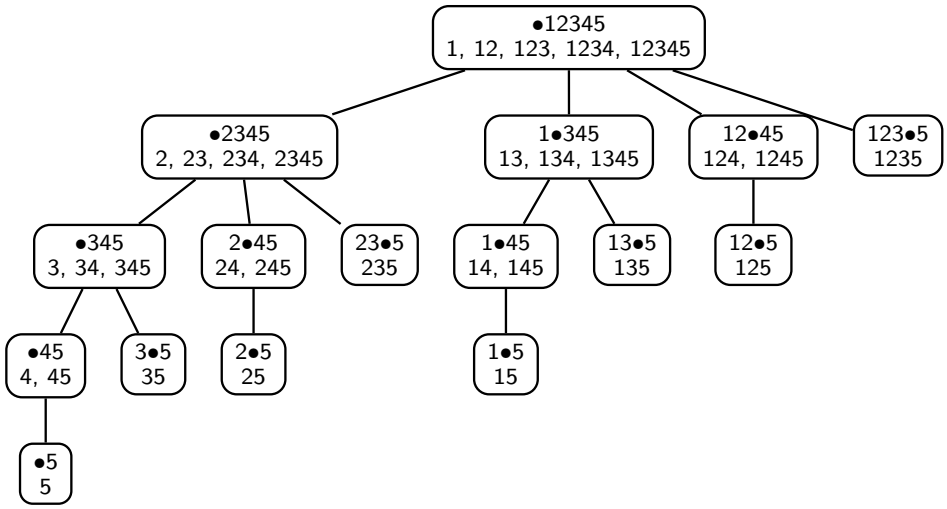


Figure 1: All-subsets regression tree for $N = 5$ variables. Nodes are shown together with their subleading models.

forward 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, Yanev, and Kontoghiorghes 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:

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

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:

$$S_1 \subseteq S_2 \implies \text{RSS}(S_1) \geq \text{RSS}(S_2) \ .$$

A cutting test is employed to determine which parts of the DCA tree are redundant: A new node $\text{drop}(S, j)$ is generated only if $\text{RSS}(S) < r_j$ $(j = k+1, \ldots, n-1)$. The quantity $\text{RSS}(S)$

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

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

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.

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 $\operatorname{drop}(S, j)$ is generated only if there exists at least one $i = j, \ldots, n-1$ such that

$$(1 + \tau_i) \cdot (\operatorname{RSS}(S) - \operatorname{RSS}_{\text{full}}) < (r_i - \operatorname{RSS}_{\text{full}}) \ , \tag{9}$$

where $\operatorname{RSS}_{\text{full}} = \operatorname{RSS}(V)$ 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 to solve the best-subset selection problem (4). Let f be a criterion function such that $f(S) = F(n, \rho)$, where $n = |S|$ and $\rho = \operatorname{RSS}(S)$, 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 $\operatorname{drop}(S, j)$ is generated if and only if

$$F(j, \operatorname{RSS}(S)) < r_f \ , \tag{10}$$

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 which further specify the selection problem. A routine to extract the best submodels from an all-subsets regression solution (i.e., to convert an '`lmSubsets`' to an '`lmSelect`' object) is also provided. An overview of the package structure is given in Table 1.

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 employing 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)
```

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 matrix interface
'`lmSelect`'	`lmSelect()`	Best-subset selection (generic function)
	`lmSelect.lmSubsets()`	Conversion method
	`lmSelect.matrix()`	Matrix interface
	`lmSelect.default()`	Standard formula interface
	`lmSelect_fit()`	Low-level matrix interface

Table 1: Package structure.

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

Parameter	Description	Canonical representation	
`x`	Data matrix	`double[nobs, nvar]`	
`y`	Response variable	`double[nobs]`	
`weights`	Model weights	`double[nobs]`	
`offset`	Model offset	`double[nvar]`	
`include`	Regressors to force in	`logical[nvar]`	
`exclude`	Regressors to force out	`logical[nvar]`	
`nmin`	Min. number of regressors	`integer[1]`	`lmSubsets()` only
`nmax`	Max. number of regressors	`integer[1]`	`lmSubsets()` only
`penalty`	Penalty per parameter	`double[1]`	`lmSelect()` only
	or criterion function	`function[1]`	
`tolerance`	ABBA tolerance parameter	`double[nvar]`	`lmSubsets()`
		`double[1]`	`lmSelect()`
`nbest`	Number of best subsets	`integer[1]`	
`pradius`	Preordering radius	`integer[1]`	

Table 2: Core parameters for `lmSubsets()` and `lmSelect()`.

```
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:

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

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]`
`size`	Covered subset sizes	`integer[]`
`tolerance`	Tolerances used	`double[nvar]`
`nbest`	Number of best subsets	`integer[1]`
`submodel`	Submodel information	`data.frame`
`subset`	Selected variables	`data.frame`

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

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 `nvar` × `nbest` 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 can be used to extract variable names, coefficients, covariance matrices, fitted values, etc.

In order to designate a submodel, '`lmSubsets`' methods provide two parameters to specify the number of regressors and the ranking of the desired submodel, namely `size` and `best`, respectively. For '`lmSelect`' methods, the `size` parameter has no meaning and is not defined. Some methods – i.e., `variable.names()`, `deviance()`, `sigma()`, `logLik()`, `AIC()`, `BIC()`, and `coef()` – 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`). The shape of the return value can be controlled with the `drop` parameter: a `numeric` or `character` vector (in some cases, a `logical` or `numeric` matrix) is returned if `drop` is `TRUE`; otherwise, a `data.frame` object is handed back.

Method	Description
`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
`logLik()`	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

Advances in numerical weather prediction (NWP) have played an important role in the increase of weather forecast skill over the past decades (Bauer, Thorpe, and Brunet 2015). Numerical models simulate physical systems that operate at a large, typically global, scale. The horizontal (spatial) resolution is limited by the computational power available today. Starting from Glahn and Lowry (1972) the NWP outputs are post-processed to correct for local and unresolved effects in order to obtain forecasts for specific locations (see Wilks 2011, Chapter 7, for an overview). So-called model output statistics (MOS) develops a regression relationship based on past meteorological observations of the variable to be predicted and forecasted NWP quantities at a certain lead time. Variable-subset selection is often employed to determine which NWP outputs should be included in the regression model for a specific location.

In the following, package **lmSubsets** is used to build a MOS regression model predicting temperature at Innsbruck Airport, Austria, based on data from the Global Ensemble Forecast System (Hamill, Bates, Whitaker, Murray, Fiorino, Galarneau Jr., Zhu, and Lapenta 2013). The data frame `IbkTemperature` contains 1824 daily cases for 42 variables: the temperature at Innsbruck Airport (observed), 36 NWP outputs (forecasted), and 5 deterministic time trend/season patterns. The NWP variables include quantities pertaining to temperature (e.g., 2-meter above ground, minimum, maximum, soil), precipitation, wind, and fluxes, among others. See `?IbkTemperature` for more details.

First, the dataset is loaded and the few missing values are omitted for simplicity.

```
R> data("IbkTemperature", package = "lmSubsets")
```

	MOS0	MOS1	MOS2
(Intercept)	-345.252** (109.212)	-666.584*** (95.349)	-661.700*** (95.225)
t2m	0.318*** (0.016)	0.055. (0.029)	
time	0.132* (0.054)	0.149** (0.047)	0.147** (0.047)
sin	-1.234*** (0.126)	0.522*** (0.147)	0.811*** (0.120)
cos	-6.329*** (0.164)	-0.812** (0.273)	
sin2	0.240* (0.110)	-0.794*** (0.119)	-0.870*** (0.118)
cos2	-0.332** (0.109)	-1.067*** (0.101)	-1.128*** (0.097)
sshnf		0.016*** (0.004)	0.018*** (0.004)
vsmc		20.200*** (3.115)	20.181*** (3.106)
tmax2m		0.145*** (0.037)	0.181*** (0.023)
st		1.077*** (0.051)	1.142*** (0.043)
wr		0.450*** (0.109)	0.505*** (0.103)
t2pvu		0.064*** (0.011)	0.149*** (0.028)
mslp			-0.000*** (0.000)
p2pvu			-0.000** (0.000)
AIC	9493.602	8954.907	8948.182
BIC	9537.650	9031.992	9025.267
RSS	19506.469	14411.122	14357.943
Sigma	3.281	2.825	2.820
R-sq.	0.803	0.854	0.855

Table 5: Estimated regression coefficients (along with standard errors) and summary statistics for models MOS0, MOS1, and MOS2.

```
R> IbkTemperature <- na.omit(IbkTemperature)
```

A simple output model for the observed temperature (temp) is constructed, which will serve as the reference model. It consists of the 2-meter temperature NWP forecast (t2m), a linear trend component (time), as well as seasonal components with annual (sin, cos) and bi-annual (sin2, cos2) harmonic patterns.

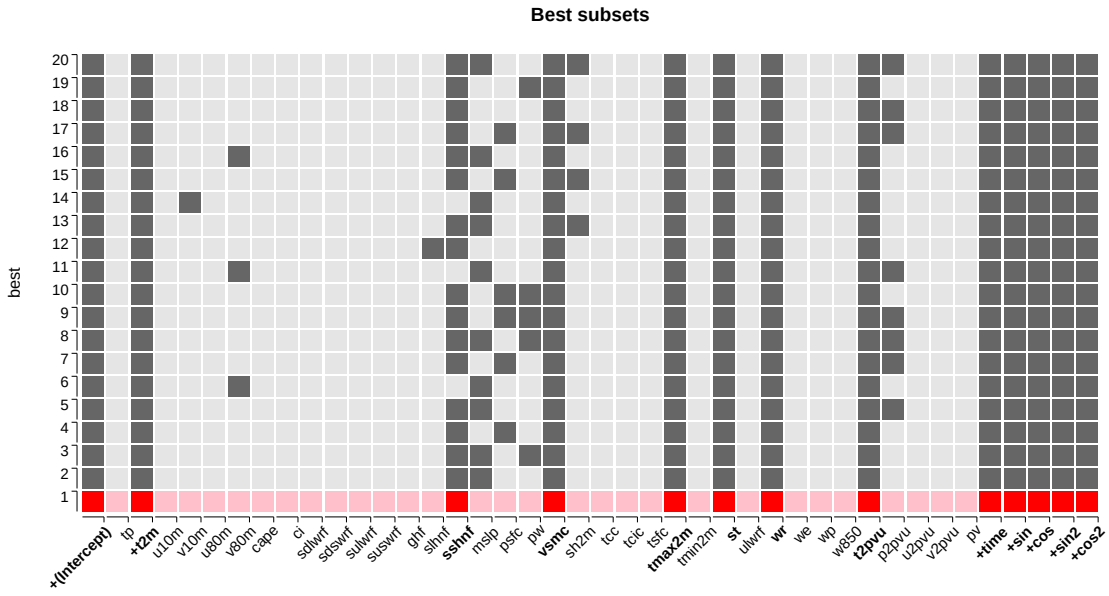
```
R> MOS0 <- lm(temp ~ t2m + time + sin + cos + sin2 + cos2,
+    data = IbkTemperature)
```

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

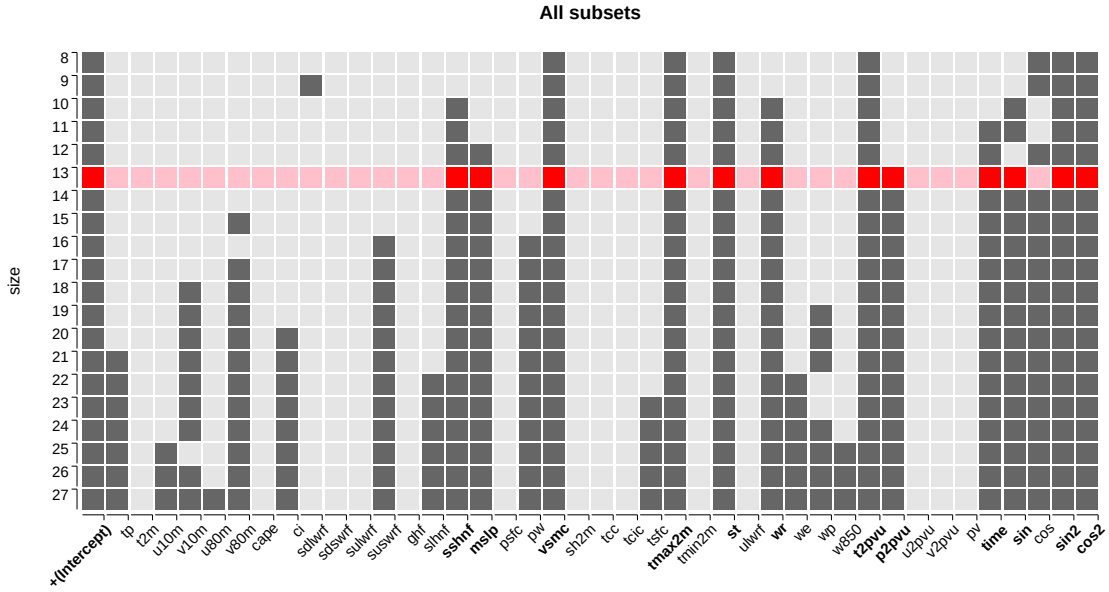
Next, the reference model is extended with selected regressors taken from the remaining 35 NWP variables.

```
R> MOS1_best <- lmSelect(temp ~ ., data = IbkTemperature,
+    include = c("t2m", "time", "sin", "cos", "sin2", "cos2"),
+    penalty = "BIC", nbest = 20)
R> MOS1 <- refit(MOS1_best)
```

Best-subset regression is employed to determine pertinent variables in addition to the regressors already found in MOS0. The 20 best submodels with respect to the BIC are computed. The selected subsets and the corresponding BIC values are illustrated in Figures 2a and 3a,



(a) `MOS1_best`



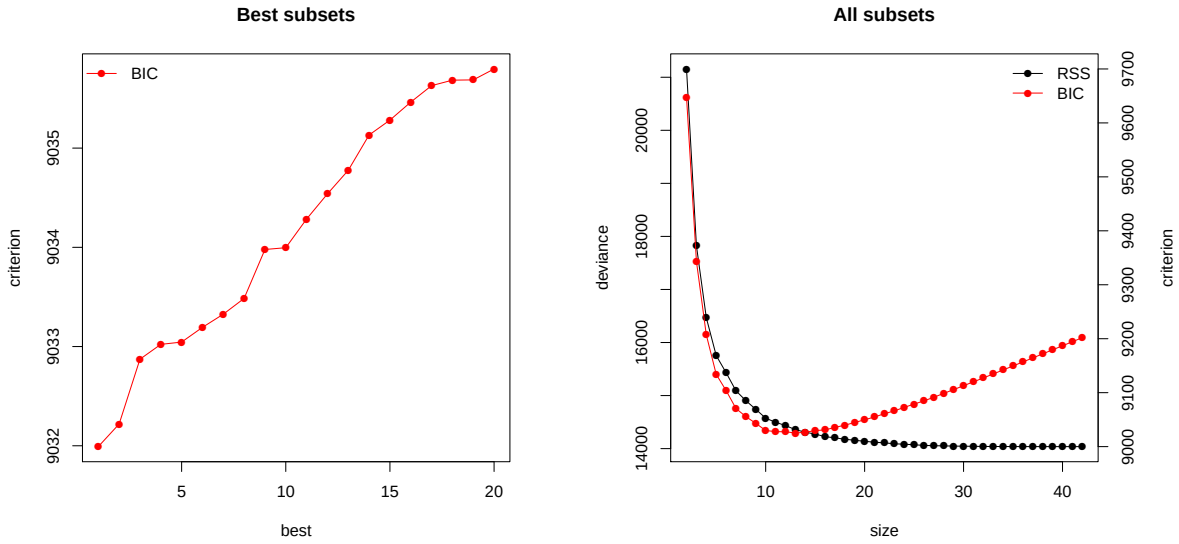
(b) `MOS2_all`

Figure 2: Variables selected in `MOS1_best` and `MOS2_all`. Submodels `MOS1` and `MOS2` are highlighted in red.

respectively. The '`lm`' object for the best submodel is extracted (`MOS1`). Selected coefficients and summary statistics for `MOS1` are listed in Table 5.

Finally, an all-subsets regression is conducted on all 41 variables without any restrictions.

```
R> MOS2_all <- lmSubsets(temp ~ ., data = IbkTemperature)
R> MOS2 <- refit(lmSelect(MOS2_all, penalty = "BIC"))
```



(a) `MOS1_best` (b) `MOS2_all`

Figure 3: BIC (and RSS) for submodels in `MOS1_best` and `MOS2_all`.

The results are illustrated in Figures 2b and 3b. Here, all-subsets regression is employed – instead of the cheaper best-subsets regression – in order to give insights into possible variable selection patterns over a range of submodel sizes. The '`lm`' object for the submodel with the lowest BIC is extracted (`MOS2`). See Table 5 for `MOS2` summary statistics.

The best-BIC models `MOS1` and `MOS2` both have 13 regressors. The deterministic trend and all but one of the harmonic seasonal components are retained in `MOS2`. In addition, `MOS1` and `MOS2` share six NWP outputs relating to temperature (`tmax2m`, `st`, `t2pvu`), pressure (`mslp`, `p2pvu`), hydrology (`vsmc`, `wr`), and heat flux (`sshnf`). However, and most remarkably, `MOS1` does not include the direct 2-meter temperature output from the NWP model (`t2m`). In fact, `t2m` is not included by any of the 20 submodels (sizes 8 to 27) shown in Figure 2b, whereas the temperature quantities `tmax2m`, `st`, `t2pvu` are included by all. The summary statistics reveal that both `MOS1` and `MOS2` significantly improve over the simple reference model `MOS0`, with `MOS2` being slightly better than `MOS1`.

In summary, this case study illustrates how **lmSubsets** can be used to easily identify relevant variables beyond the direct model output for MOS regressions, yielding substantial improvements in forecasting skill. A full meteorological application would require further testing using cross-validation or other out-of-sample assessments. Recently, there has been increasing interest in MOS models beyond least-squares linear regression, e.g., to take into account the effects of heteroscedasticity, censoring, and truncation. In this context, other selection techniques – such as boosting (Messner, Mayr, and Zeileis 2016, 2017) – can be considered.

5. Benchmark tests

Comparative tests are conducted to evaluate the computational efficiency of the proposed methods for exact all-subsets and exact best-subset regression. The `regsubsets()` method from package **leaps**, and the `bestglm()` method from package **bestglm** serve as benchmarks, respectively.

sigma	nvar	leaps		lmSubsets		
		regsubsets()[1]	regsubsets()[2]	lmSubsets()	speedup[1]	speedup[2]
0.05	20	0.009 s	0.004 s	0.021 s	0.4	0.2
	25	0.072 s	0.033 s	0.011 s	6.4	2.9
	30	0.829 s	0.474 s	0.027 s	31.2	17.8
	35	12.309 s	5.084 s	0.067 s	182.6	75.4
	40	172.613 s	82.566 s	0.313 s	550.8	263.5
0.10	20	0.008 s	0.004 s	0.007 s	1.1	0.6
	25	0.064 s	0.031 s	0.010 s	6.2	3.0
	30	0.970 s	0.457 s	0.027 s	36.5	17.2
	35	9.912 s	4.792 s	0.068 s	146.2	70.7
	40	208.998 s	93.101 s	0.334 s	626.5	279.1
0.50	20	0.009 s	0.004 s	0.007 s	1.2	0.6
	25	0.081 s	0.031 s	0.011 s	7.5	2.9
	30	0.995 s	0.462 s	0.026 s	38.0	17.6
	35	12.751 s	4.995 s	0.068 s	187.5	73.4
	40	204.834 s	82.710 s	0.312 s	656.9	265.3
1.00	20	0.008 s	0.004 s	0.007 s	1.2	0.6
	25	0.070 s	0.033 s	0.011 s	6.6	3.1
	30	0.971 s	0.461 s	0.026 s	37.6	17.9
	35	13.066 s	4.560 s	0.066 s	198.6	69.3
	40	171.499 s	62.978 s	0.277 s	620.0	227.7
5.00	20	0.008 s	0.004 s	0.007 s	1.1	0.5
	25	0.058 s	0.019 s	0.010 s	5.7	1.9
	30	0.588 s	0.198 s	0.021 s	28.5	9.6
	35	6.951 s	2.455 s	0.053 s	131.7	46.5
	40	117.859 s	30.252 s	0.193 s	609.4	156.4

[1] `regsubsets()` is executed w/out preliminary preordering of the variables.
[2] `regsubsets()` is executed with preliminary preordering of the variables.

Table 6: Speedup of `lmSubsets()` relative to `regsubsets()`; average execution times in seconds.

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 `e`, and the intercept:

$$\mathtt{y} = \mathtt{X} \cdot \mathbb{1}_{\text{true}} + \mathtt{e} + 1 \ , \quad \text{where} \quad \mathtt{e} \sim (0, \mathtt{sigma}^2) \ ,$$

where `X` is a `nobs` $\times$ `nvar` matrix of random data, and $\mathbb{1}_{\text{true}}$ a (random) indicator function evaluating to 1 if the corresponding column of `X` belongs to the "true" model. All tests were conducted on a Dell XPS15 laptop with 8GB (7.4 GiB) of memory and an Intel Core i7-6700HQ CPU@2.60GHz×8 processor, running a Ubuntu 64bit operating system.

Benchmark 1 concerns itself with all-subsets selection. The `lmSubsets()` method is compared to `regsubsets()`. The complexity mainly depends on the number of variables (`nvar`): The algorithms employ the QR decomposition to compress the data into a square `nvar` $\times$ `nvar` matrix; the initial cost of constructing the QR decomposition is negligible. Data con-

sigma	nvar	bestglm		lmSubsets		
		bestglm()[1]	bestglm()[2]	lmSelect()	speedup[1]	speedup[2]
0.05	20	0.021 s	0.017 s	0.006 s	3.4	2.7
	25	0.083 s	0.046 s	0.008 s	10.4	5.7
	30	0.835 s	0.489 s	0.008 s	99.4	58.2
	35	12.270 s	5.110 s	0.010 s	1202.9	501.0
	40	174.041 s	83.399 s	0.012 s	14503.4	6949.9
0.10	20	0.020 s	0.016 s	0.007 s	3.0	2.5
	25	0.074 s	0.045 s	0.007 s	10.1	6.1
	30	0.974 s	0.471 s	0.009 s	110.7	53.5
	35	9.875 s	4.777 s	0.010 s	949.5	459.3
	40	210.076 s	93.968 s	0.012 s	17219.3	7702.3
0.50	20	0.020 s	0.017 s	0.006 s	3.3	2.7
	25	0.093 s	0.044 s	0.008 s	12.2	5.8
	30	1.004 s	0.474 s	0.009 s	114.1	53.8
	35	12.744 s	5.067 s	0.011 s	1158.5	460.7
	40	205.744 s	83.268 s	0.012 s	17145.4	6939.0
1.00	20	0.021 s	0.017 s	0.006 s	3.2	2.7
	25	0.082 s	0.046 s	0.007 s	11.0	6.3
	30	0.979 s	0.474 s	0.008 s	119.3	57.8
	35	13.002 s	4.568 s	0.011 s	1182.0	415.2
	40	172.923 s	63.283 s	0.012 s	14907.2	5455.4
5.00	20	0.020 s	0.016 s	0.006 s	3.2	2.6
	25	0.070 s	0.032 s	0.008 s	9.3	4.3
	30	0.598 s	0.212 s	0.009 s	63.6	22.6
	35	6.942 s	2.467 s	0.012 s	588.3	209.1
	40	118.004 s	30.404 s	0.018 s	6555.8	1689.1

[1] `bestglm()` is executed w/out preliminary preordering of the variables.
[2] `bestglm()` is executed with preliminary preordering of the variables.

Table 7: Speedup of `lmSelect()` relative to `bestglm()`; average execution times in seconds.

figurations with varying sizes (`nvar` $= 20, 25, 30, 35, 40$) and degrees of noise (`sigma` $= 0.05, 0.10, 0.50, 1.00, 5.00$) are considered; in all cases, `nobs` $= 1000$ and `ntrue` $= \lfloor$`nvar`$/2 \rfloor$. For each configuration, five random datasets are generated, giving rise to five runs per method over which average execution times are determined. The performance of `regsubsets()` can be improved by "manually" preordering the dataset in advance (Hofmann *et al.* 2007). The average running times are summarized in Table 6, along with the relative performance (speedup) of `lmSubsets()`. The same setup is used in Benchmark 2, where methods for best-subset selection are compared, namely `bestglm()` and `lmSelect()`. As in the previous benchmark, average execution times are determined for `bestglm()` with and without preordering. The results are illustrated in Table 7.

It is not surprising that `bestglm()` is very close to `regsubsets()` in terms of execution time, as the former post-processes the results returned by the latter; in fact, `bestglm()` implements the two-stage approach to solving the best-subset selection problem, where Stage 1

sigma	nvar	nmin	nmax	tolerance = 0.0			tolerance = 0.1		
				lmSubsets()	lmSelect()	speedup	lmSubsets()	lmSelect()	speedup
0.05	20	-	-	0.021 s	0.022 s	1.0	0.006 s	0.007 s	1.0
	40	-	-	0.327 s	0.012 s	26.8	0.227 s	0.012 s	18.6
	60	-	-	217.028 s	0.020 s	10961.0	101.557 s	0.020 s	5181.5
	80	30	50	413.982 s	0.032 s	12936.9	163.629 s	0.031 s	5312.6
0.10	20	-	-	0.007 s	0.006 s	1.1	0.007 s	0.007 s	1.0
	40	-	-	0.315 s	0.012 s	26.2	0.220 s	0.012 s	18.4
	60	-	-	208.721 s	0.020 s	10332.7	97.096 s	0.020 s	4953.9
	80	30	50	465.824 s	0.031 s	14835.2	185.665 s	0.031 s	5989.2
0.50	20	-	-	0.007 s	0.007 s	1.0	0.007 s	0.007 s	1.0
	40	-	-	0.336 s	0.012 s	28.0	0.231 s	0.012 s	19.2
	60	-	-	197.147 s	0.020 s	9956.9	93.936 s	0.020 s	4744.2
	80	30	50	486.858 s	0.032 s	15406.9	195.240 s	0.031 s	6217.8
1.00	20	-	-	0.007 s	0.006 s	1.1	0.007 s	0.007 s	1.0
	40	-	-	0.290 s	0.012 s	24.2	0.205 s	0.012 s	17.1
	60	-	-	228.710 s	0.020 s	11668.9	106.009 s	0.019 s	5464.4
	80	30	50	374.452 s	0.032 s	11701.6	148.421 s	0.043 s	3467.8
5.00	20	-	-	0.007 s	0.007 s	1.1	0.007 s	0.007 s	1.0
	40	-	-	0.196 s	0.017 s	11.4	0.146 s	0.015 s	9.5
	60	-	-	89.244 s	0.195 s	458.1	47.399 s	0.114 s	416.5
	80	30	50	154.836 s	8.056 s	19.2	58.246 s	3.012 s	19.3

Table 8: Speedup of `lmSelect()` relative to `lmSubsets()`, with and without tolerance; average execution times in seconds.

is tackled by `regsubsets()` (see Section 1 for further details). Manually preordering the variables improves the performance of `regsubsets()` (and hence, of `bestglm()`) by a factor of approximately 2; for `nvar` = 40 and a high level of noise (`sigma` = 5.00), by a factor of almost 4. In the tests conducted here, `lmSubsets()` is two orders of magnitude faster than `regsubsets()`, even with preordering; `lmSelect()` is three orders of magnitude faster than `bestglm()`.

Benchmark 3 pits all-subsets and best-subset selection, exact and approximation algorithms against one another. The average execution times of `lmSubsets()` and `lmSelect()`, for `tolerance` = 0.0 and 0.1, are illustrated in Table 8. Note that for large datasets (`nvar` = 80), subsets computed by `lmSubsets()` are restricted to sizes between `nmin` = 30 and `nmax` = 50 variables; the restriction does not apply to `lmSelect()`.

In the case of `lmSubsets()`, the approximation algorithm (`tolerance` = 0.1) is 2–3 times faster than the exact algorithm. The speedup of `lmSelect()` with respect to `lmSubsets()` is four orders of magnitude for the exact, three orders of magnitude for the approximation algorithm. It is interesting to note, that the computational performance of `lmSubsets()` increases for high levels of noise (`sigma` = 5.00), contrary to `lmSelect()`. Under these conditions, the relative speedup of `lmSelect()` is significantly lower. As the noise increases, the information in the data is "blurred", rendering the cutting test (10) – which depends on the information criterion – less effective; in this respect, `lmSubsets()` is more robust, as it only depends on the RSS.

In Benchmark 4, the effects of the `nbest` parameter (number of computed best submodels) on the execution times of `lmSelect()` are investigated. Two information criteria are considered (`ic` is `"AIC"` and `"BIC"`). The noise level used in the benchmark is `sigma` = 1.0. Average execution times are reported in Table 9 for `nbest` = $1, 5, 10, 15, 20$. Finally, Benchmark 5 investigates how the AIC penalty per parameter (`penalty`) affects the performance of `lmSelect()`. Table 10 summarizes the results for `penalty` = $1.0, 2.0, 4.0, 8.0, 16.0, 32.0$.

nvar	ic	nbest				
		1	5	10	15	20
100	AIC	2.159 s	2.334 s	2.457 s	2.557 s	2.639 s
	BIC	0.051 s	0.058 s	0.068 s	0.074 s	0.079 s
200	BIC	23.987 s	49.622 s	82.175 s	104.860 s	119.064 s

Table 9: Average execution times (in seconds) of `lmSelect()`, by number of computed subset models (`nbest`).

nvar	penalty					
	1.0	2.0	4.0	8.0	16.0	32.0
80	0.293 s	0.300 s	0.050 s	0.033 s	0.032 s	0.068 s
100	4.223 s	0.894 s	0.084 s	0.048 s	0.054 s	0.341 s
120	14.178 s	6.420 s	0.531 s	0.085 s	0.168 s	3.925 s

Table 10: Average execution times (in seconds) of `lmSelect()`, by AIC penalty per parameter (`penalty`).

Note that `penalty` = 2.0 and `penalty` = `log`(1000) $\approx$ 6.9 correspond to the usual AIC and BIC, respectively (here, `nobs` = 1000). The results reveal that the execution time of `lmSelect()` increases linearly with `nbest`, and – from the values considered here – is minimal for `penalty` = 8.0, which is close to the BIC.

5.1. Shrinkage methods

Genetic algorithms for model selection have been considered for comparative study. However, pertinent R packages have been found to impose restrictions on the class of problems that can be addressed – limited problem size (**glmulti**), fixed submodel size (**kofnGA**), or no immediate support for IC-based search (**subselect**) –, hampering efforts to conduct a meaningful comparison.

LASSO (Tibshirani 1996) can be seen as an alternative to exact variable selection methods, of which package **glmnet** brings an efficient implementation to R. The function `glmnet()` computes an entire regularization path and returns a sequence of sparse estimators. The method is not IC-based; rather, it employs a modified objective function that induces sparsity by penalizing the regression coefficients.

The return value of `glmnet()` can be post-processed for comparison with `lmSelect()`. For each (sparse) estimator contained in the sequence returned by `glmnet()`, the subset model corresponding to the variables with non-zero coefficients is identified; the submodel is fitted (in the OLS sense), and the BIC extracted. The list of submodels thus obtained is sorted in order of increasing BIC, after removal of duplicates.

Comparative results are illustrated in Table 11. For each data configuration, five datasets are simulated. The ten best submodels are computed by `lmSelect()` (i.e., `nbest` = 10). Average execution times of `lmSelect()` and `glmnet()` are reported, as well as the average number of matches – i.e., the number of best subsets correctly identified – and the speedup of the LASSO. Each function returns an ordered sequence of submodels; the number of matches is k if and only if the two sequences are identical in the first k entries and differ in the $(k+1)$th.

sigma	nvar	**lmSubsets**		**glmnet**	
		lmSelect()	glmnet()	speedup	matches
0.05	60	0.020 s	0.006 s	3.6	0.8
	100	0.075 s	0.009 s	8.0	0.6
	140	1.012 s	0.016 s	63.2	0.6
0.10	60	0.019 s	0.005 s	4.0	0.8
	100	0.070 s	0.009 s	7.4	0.6
	140	1.148 s	0.016 s	71.7	0.6
0.50	60	0.020 s	0.006 s	3.4	1.2
	100	0.068 s	0.012 s	5.7	1.2
	140	0.784 s	0.021 s	37.7	0.8
1.00	60	0.019 s	0.006 s	3.5	1.6
	100	0.062 s	0.012 s	5.2	1.2
	140	0.600 s	0.021 s	28.8	0.6

Table 11: Speedup and average number of matches of `glmnet()`; average execution times in seconds.

The takeaway it that the LASSO is computationally very efficient; it is much less affected by the dimension of the problem than `lmSelect()`. On the other hand, while `lmSelect()` always finds the global optimum – or a solution with provable error bounds when a tolerance is employed –, `glmnet()` does not provide any guarantees on the distance of the result from the optimal solution (in the OLS sense).

6. Conclusions

An R package for all-subsets variable selection is presented. It is based on theoretical 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. An extension of the package to handle missing data merits investigation.

Acknowledgments

This work was in part supported by the CRoNoS COST Action (IC1408); GRUPIN14-005 (Principality of Asturias, Spain); and the *Förderverein des wirtschaftswissenschaftlichen Zentrums der Universität Basel* through research project B-123. 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.

Bauer P, Thorpe A, Brunet G (2015). "The Quiet Revolution of Numerical Weather Prediction." *Nature*, **525**(7567), 47–55. doi:10.1038/nature14956.

Calcagno V (2019). ***glmulti****: Model Selection and Multimodel Inference Made Easy*. R package version 1.0.7.1, URL https://CRAN.R-project.org/package=glmulti.

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

Clarke MRB (1981). "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 C*, **30**(2), 198–203. doi:10.2307/2346398.

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. doi:10.1016/s0167-8191(03)00019-x.

Gatu C, Kontoghiorghes EJ (2006). "Branch-and-Bound Algorithms for Computing the Best-Subset Regression Models." *Journal of Computational and Graphical Statistics*, **15**(1), 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 & Data Analysis*, **52**(2), 799–815. doi:10.1016/j.csda.2007.02.018.

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.

Goldberg DE (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. 1st edition. Addison-Wesley Longman Publishing, Boston.

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

Hamill TM, Bates GT, Whitaker JS, Murray DR, Fiorino M, Galarneau Jr TJ, 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 (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd edition. Springer-Verlag, New York. doi:10.1007/978-0-387-84858-7.

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**(1), 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**(1), 191–204. doi:10.1198/jcgs.2009.07091.

Hofmann M, Gatu C, Kontoghiorghes EJ, Colubi A, Zeileis A (2020). ***lmSubsets***: *Exact Variable-Subset Selection in Linear Regression*. R package version 0.5, 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 (2017). ***leaps***: *Regression Subset Selection*. R package version 3.0, URL https://CRAN.R-project.org/package=leaps.

McLeod AI, Xu C (2019). ***bestglm***: *Best Subset GLM*. R package version 0.37.1, URL https://CRAN.R-project.org/package=bestglm.

Messner JW, Mayr GJ, Zeileis A (2016). "Heteroscedastic Censored and Truncated Regression with **crch**." *The R Journal*, **8**(1), 173–181. doi:10.32614/rj-2016-012.

Messner JW, Mayr GJ, Zeileis A (2017). "Non-Homogeneous Boosting for Predictor Selection in Ensemble Post-Processing." *Monthly Weather Review*, **145**(1), 137–147. doi:10.1175/mwr-d-16-0088.1.

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.

Orestes Cerdeira J, Duarte Silva AP, Cadima J, Minhoto M (2018). ***subselect***: *Selecting Variable Subsets*. R package version 0.14, URL https://CRAN.R-project.org/package=subselect.

Otten RHJM, Van Ginneken LPPP (1989). *The Annealing Algorithm*. Kluwer Academic Publishers, Dordrecht.

R Core Team (2019). *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.

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.

Tibshirani R (1996). "Regression Shrinkage and Selection via the Lasso." *Journal of the Royal Statistical Society B*, **58**(1), 267–288. doi:10.1111/j.2517-6161.1996.tb02080.x.

Wen C, Zhang A, Quan S, Wang X (2019). ***BeSS**: Best Subset Selection in Linear, Logistic and Cox PH Models*. R package version 1.0.6, URL https://CRAN.R-project.org/package=BeSS.

Wilks DS (2011). *Statistical Methods in the Atmospheric Sciences*. 3rd edition. Academic Press, Amsterdam.

Wolters MA (2015). "A Genetic Algorithm for Selection of Fixed-Size Subsets with Application to Design Problems." *Journal of Statistical Software*, **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, marc.indurot@uniovi.es
URL: http://www.indurot.uniovi.es/

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
April 2020, Volume 93, Issue 3 *Submitted:* 2018-05-02
doi:10.18637/jss.v093.i03 *Accepted:* 2018-11-11