mcsSubset: Efficient Computation of Best Subset Linear Regressions in R

Marc Hofmann
Université de Neuchâtel

Cristian Gatu
Université de Neuchâtel

Erricos J. Kontoghiorghes
Birbeck College

Achim Zeileis
Universität Innsbruck

Abstract

todo

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

1. Introduction

The **mcsSubset** package for exact, best variable-subset regression implements the algorithms presented by Gatu and Kontoghiorghes (2006) and Hofmann, Gatu, and Kontoghiorghes (2007). Computationally intensive core code is written in C++. It is available for the R platform for statistical computing (Hofmann, Gatu, Kontoghiorghes, and Zeileis 2011) from the Comprehensive R Archive Network at `http://CRAN.R-project.org/package=mcsSubset`.

`TODO:` somewhere we need to comment on alternative approaches in R, especially the **leaps** package (Lumley and Miller 2009) based on Miller (2002), the **subselect** package (Orestes Cerdeira, Duarte Silva, Cadima, and Minhoto 2009) based on Duarte Silva (2001), and the **glmulti** package (Calcagno and de Mazancourt 2010)

2. Implementation

Consider the standard regression model

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

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

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

Possible criteria include Mallows' C_p , adjusted R^2, or a criterion of the AIC family (Miller 2002). The aforementioned criteria are monotonic functions of the residual sum of squares

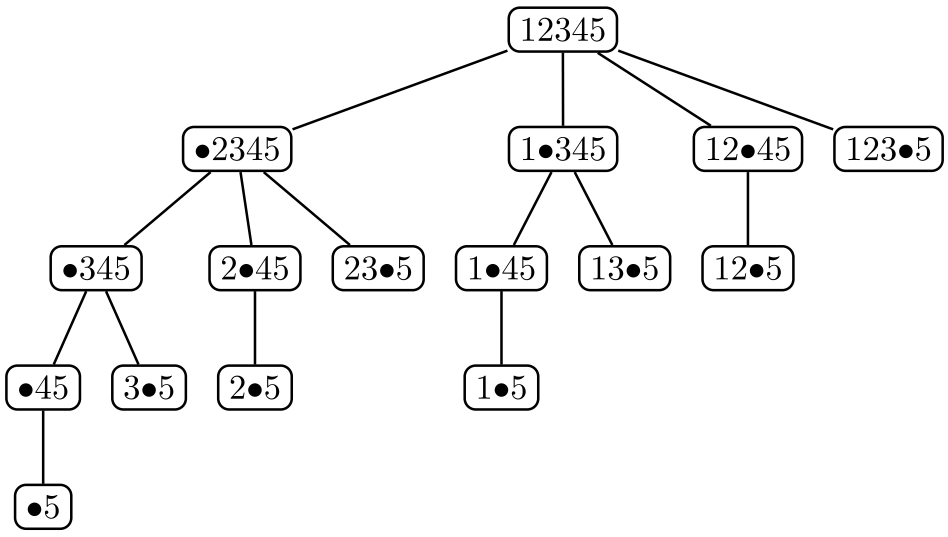

Figure 1: All-subsets regression tree for $N = 5$ variables.

(RSS) for a constant number of parameters, and attain their optimal value when the RSS is minimal. Thus, the search problem may be decomposed as follows: Find

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

There are $2^N - 1$ possible subset models, and their computation is only feasible for small values of N. Regression trees are employed to traverse the search space in a systematic fashion. Figure 1 illustrates a regression tree for $N = 5$ variables. A node in the regression tree is a pair (S, k), where $S = (s_1, \ldots, s_n)$ is a certain subset of n variables and k is an integer, symbolized by a $\bullet$.

Each node corresponds to a unique subset of variables, although not every possible subset gives rise to a node. Thus, the number of nodes is 2^{N-1}. When the algorithm visits node (S, k), it reports the RSS of the models corresponding to the leading variable subsets of size $k+1, \ldots, n$, i.e. the subleading models $(s_1, \ldots, s_{k+1}), \ldots, (s_1, \ldots, s_n)$. It then generates and in turn visits the nodes $(S - \{s_j\}, j-1)$ for $j = n-1, \ldots, k+1$ (Gatu and Kontoghiorghes 2006).

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

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

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

The branch-and-bound algorithm with tolerance parameter is illustrated in Listing 1. The keywords `break` and `next` pertain the same meaning as in the R language; that is, `break` transfers execution to the statement following the inner-most loop, and `next` halts the current iteration and advances the looping index.

Algorithm 1 The branch-and-bound algorithm.

```
 1: procedure BBA($V$, $\tau$, $r$)
 2:     $N \leftarrow |V|$
 3:     $r_{1:N} \leftarrow +\infty$
 4:     $z \leftarrow ((V, 0))$   # node stack
 5:     while not empty $z$ do
 6:         with $z$ pop $(S, k)$
 7:         $n \leftarrow |S|$, $\rho \leftarrow \mathrm{RSS}(S)$
 8:         preorder $(S, k)$
 9:         with $(S, k)$ update $r_{k+1:n}$
10:         for $i = n - 1, \ldots, k + 1$ do
11:             if $(1 + \tau_i) \cdot \rho > r_i$ next
12:             for $j = k + 1, \ldots, i$ do
13:                 with $z$ push $(S - \{s_j\}, j - 1)$
14:             end for
15:             break
16:         end for
17:     end while
18: end procedure
```

The algorithm reports the N subset models with the lowest RSS, one for each subset size. The user can then analyse the list of returned subsets to determine the “best” subset, e.g. by evaluating some criterion function. This approach is practical but not necessarily efficient. The algorithm may be optimized for a particular criterion f under the condition that the latter may be expressed as a function of the subset size n and the RSS ρ, i.e. $f(n, \rho)$, and that f is monotonic with respect to both n and ρ. The modified algorithm is presented in Listing 2. It takes a single tolerance value and returns a single solution, that is the overall (i.e. over all subset sizes) best subset model according to criterion function f.

Algorithm 2 The modified branch-and-bound algorithm.

```
1: procedure mBBA($V$, $\tau$, $f$)
2:     $r \leftarrow +\infty$
3:     $z \leftarrow ((V, 0))$   # node stack
4:     while not empty $z$ do
5:         with $z$ pop $(S, k)$
6:         $n \leftarrow |S|$, $\rho \leftarrow \mathrm{RSS}(S)$
7:         preorder $(S, k)$
8:         with $(S, k)$, $f$ update $r$
9:         for $i = n - 1, \ldots, k + 1$ do
10:            if $(1 + \tau) \cdot f(i, \rho) > r$ next
11:            for $j = k + 1, \ldots, i$ do
12:                with $z$ push $(S - \{s_j\}, j - 1)$
13:            end for
14:            break
15:        end for
16:    end while
17:    return $r$
18: end procedure
```

2.1. C++

Focus has been put on the computational efficiency of the C++ code. The following points have been given special attention: (a) no allocation of dynamic memory in main loop, (b) no (unnecessary) matrix copy operations.

2.2. R interface

3. Illustrations

load package and example data, for convenience already take logs for relative potentials

```
R> library("mcsSubset")
R> data("AirPollution", package = "mcsSubset")
R> for(i in 12:14) AirPollution[[i]] <- log(AirPollution[[i]])
```

then fitting best subsets can be done via

```
R> xs <- mcsSubset(mortality ~ ., data = AirPollution)
R> xs

Call:
mcsSubset(formula = mortality ~ ., data = AirPollution)

Total regressors: 16
Intercept:        Yes
```

```
Include:          1
Exclude:
Size:             2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Criterion:        RSS
Tolerance:        Inf 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
N best:           1
```

To obtain a more complete picture, look at visualization (see Figure 2) or a tabular summary:

```
R> plot(xs)
R> summary(xs)

Call:
mcsSubset(formula = mortality ~ ., data = AirPollution)

Selected variables (best first):
              1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13. 14. 15.
+(Intercept)  x  x  x  x  x  x  x  x  x  x   x   x   x   x   x
precipitation x  x  x  x  x  x  x  x  x  x   x   x   x
temperature1  x  x  x  x  x  x  x  x  x  x   x   x
temperature7  x  x  x              x  x      x   x
age           x  x  x  x           x  x      x   x
household     x  x  x  x  x        x  x      x   x
education     x  x  x  x  x  x  x  x  x      x   x       x
housing                            x  x      x   x
population          x              x  x      x   x
noncauc       x  x  x  x  x  x  x  x  x  x   x   x   x   x   x
whitecollar                                      x
income                                x      x   x
hydrocarbon   x  x  x  x  x  x     x  x      x   x
nox           x  x  x  x  x  x  x  x  x  x   x   x
so2              x  x              x  x      x   x   x
humidity                                     x   x

Model fit:
       1.       2.       3.       4.       5.       6.       7.
AIC       594.1    594.7    596.0    596.3    596.6    596.9    597.1
RSS     48610.2  47471.4  46893.7  52101.6  54128.4  56314.6  58390.6
(size) 10       11       12       9        8        7        6
       8.       9.       10.      11.      12.      13.      14.
AIC       597.3    599.2    600.6    601.1    603.1    610.2    623.3
RSS     46380.2  46280.2  64037.8  46248.6  46248.6  77673.5  99841.1
(size) 13       14       5        15       16       4        3
       15.
AIC       638.8
RSS    133694.5
(size)  2
```

t

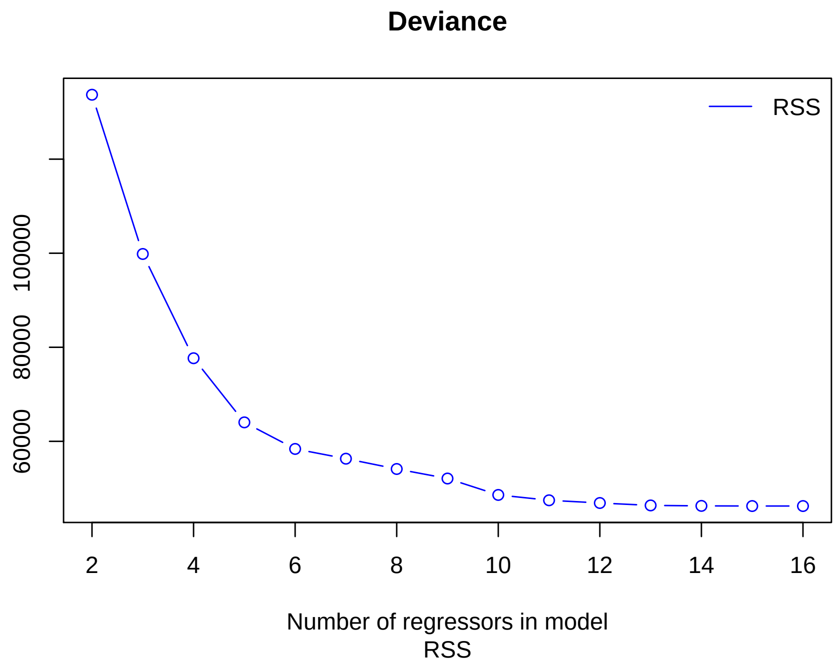

Figure 2: BIC and RSS.

```
AIC: k = 2
```

extract information (by default for best BIC model)

```
R> deviance(xs)

         2          3          4          5          6          7          8
133694.54   99841.07   77673.52   64037.82   58390.63   56314.60   54128.39
         9         10         11         12         13         14         15
 52101.56   48610.18   47471.39   46893.66   46380.24   46280.17   46248.62
        16
 46248.59

R> logLik(xs)

'log Lik.' -316.4054, -307.6460, -300.1141, -294.3228, -291.5533, -290.4672, -289.2794, -2

R> AIC(xs)

   df      AIC
1   3 638.8107
```

```
2   4 623.2920
3   5 610.2281
4   6 600.6457
5   7 597.1066
6   8 596.9345
7   9 596.5588
8  10 596.2690
9  11 594.1072
10 12 594.6849
11 13 595.9502
12 14 597.2897
13 15 599.1601
14 16 601.1192
15 17 603.1191
```

```
R> AIC(xs, k = log(nrow(AirPollution)))
```

```
   df      AIC
1   3 638.8107
2   4 623.2920
3   5 610.2281
4   6 600.6457
5   7 597.1066
6   8 596.9345
7   9 596.5588
8  10 596.2690
9  11 594.1072
10 12 594.6849
11 13 595.9502
12 14 597.2897
13 15 599.1601
14 16 601.1192
15 17 603.1191
```

extract information for all subsets fitted

```
R> deviance(xs, size = 1:16)
```

```
        1         2         3         4         5         6         7 
       NA 133694.54  99841.07  77673.52  64037.82  58390.63  56314.60 
        8         9        10        11        12        13        14 
 54128.39  52101.56  48610.18  47471.39  46893.66  46380.24  46280.17 
       15        16 
 46248.62  46248.59 
```

```
R> AIC(xs, size = 1:16)
```

```
   df      AIC
1   2       NA
2   3 638.8107
3   4 623.2920
4   5 610.2281
5   6 600.6457
6   7 597.1066
7   8 596.9345
8   9 596.5588
9  10 596.2690
10 11 594.1072
11 12 594.6849
12 13 595.9502
13 14 597.2897
14 15 599.1601
15 16 601.1192
16 17 603.1191
```

```
R> AIC(xs, size = 1:16, k = log(nrow(AirPollution)))
```

```
   df      AIC
1   2       NA
2   3 638.8107
3   4 623.2920
4   5 610.2281
5   6 600.6457
6   7 597.1066
7   8 596.9345
8   9 596.5588
9  10 596.2690
10 11 594.1072
11 12 594.6849
12 13 595.9502
13 14 597.2897
14 15 599.1601
15 16 601.1192
16 17 603.1191
```

refit model (best BIC by default)

```
R> lm5 <- refit(summary(xs))
R> summary(lm5)
```

```
Call:
lm(formula = mortality ~ precipitation + temperature1 + temperature7 +
    age + household + education + noncauc + hydrocarbon + nox,
    data = model.frame(formula = summary(xs)))
```

```
Residuals:
    Min      1Q  Median      3Q     Max 
-70.945 -23.333   3.262  16.530  73.411 

Coefficients:
               Estimate Std. Error t value Pr(>|t|)    
(Intercept)   1933.7641   347.8482   5.559 1.05e-06 ***
precipitation    2.6827     0.7582   3.538 0.000881 ***
temperature1    -2.5929     0.5892  -4.400 5.68e-05 ***
temperature7    -3.1549     1.6648  -1.895 0.063875 .  
age            -13.7654     6.7279  -2.046 0.046032 *  
household     -148.8091    57.4629  -2.590 0.012552 *  
education      -20.4739     6.7397  -3.038 0.003782 ** 
noncauc          4.1544     0.9916   4.189 0.000114 ***
hydrocarbon    -33.9532    14.4756  -2.346 0.023007 *  
nox             45.3206    12.6970   3.569 0.000801 ***
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1

Residual standard error: 31.18 on 50 degrees of freedom
Multiple R-squared:  0.7871,	Adjusted R-squared:  0.7488 
F-statistic: 20.54 on 9 and 50 DF,  p-value: 6.521e-14
```

(Note that the p values are not valid due to model selection.)

Acknowledgments

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

References

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. URL http://www.jstatsoft.org/v34/i12/.

Duarte Silva AP (2001). "Efficient Variable Screening for Multivariate Analysis." *Journal of Multivariate Analysis*, **76**, 35–62.

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

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.

Hofmann M, Gatu C, Kontoghiorghes EJ, Zeileis A (2011). ***xsubset**: Variable Subset Selection in Linear Regressions.* R package version 0.4-1, URL `http://CRAN.R-project.org/package=xsubset`.

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

Miller A (2002). *Subset Selection in Regression.* 2nd edition. Chapman & Hall/CRC, Boca Raton, FL.

Orestes Cerdeira J, Duarte Silva AP, Cadima J, Minhoto M (2009). ***subselect**: Selecting Variable Subsets.* R package version 0.9-9993, URL `http://CRAN.R-project.org/package=subselect`.

Affiliation:

Marc Hofmann
Wirtschaftswissenschaftliche Fakultät
Abteilung Quantitative Methoden
Computational Management Science
Universität Basel
Peter Merian-Weg 6
4002 Basel, Switzerland