Model Selection in R

Model (variable) selection methods do not appear to have been implemented in R Commander; the R Console interface must be used. The principal function for this is **`regsubsets`**, which is in the **`leaps`** package. The **`leaps`** function (not surprisingly, also in package **`leaps`**) also is applicable.

Other functions (e.g. **`step`**) do various forms of "stepwise" model selection. As far as I can tell, however, these all select a single model, based on some criterion for stopping the stepwise process, rather than producing sets of candidate models of different sizes.

regsubsets

The minimal invocation of **`regsubsets`** is:

```
> subsets <- regsubsets(y ~ x1 + x2 … + xk, dataframe)
```

where

`subsets` is an object in which the results will be stored (see below); **`regsubsets`** does not itself produce any output so you must assign it to an object.

`y` is the response variable

`xn` are the full set of explanatory variables; if all the variables in the dataframe except **`y`** are to be explanatory variables, this list can be replaced by a period .

`dataframe` is the data frame to be analyzed.

The most useful options are

`nbest = n`	Specifies how many models of each size are to be kept in the result object; default is **`nbest = 1`**
`nvmax = n`	Specifies the maximum size model (number of variables to include); default is **`nvmax = 8`**.
`force.in = n [, n…]`	Specifies one or more variables to be included in all models. Variables are specified not by name but by order in the formula. So for instance **`regsubsets(lambda ~ latit + age + altit + rain, data=loopers, force.in=2)`** would force the second explanatory variable, **`age`**, to be included in all models.

I have not yet succeeded in using it, but there also is a form of **`regsubsets`** in which the Y variable and a matrix of X variables are specified, rather than a model formula:

```
> subsets <- regsubsets(x = Xmat, y = Yvec)
```

where

`Xmat` is a matrix containing (as columns) all the explanatory variables

`Yvec` is a vector containing the response variable

In this case the **`X`** matrix might include columns you do not want included in the models, in which case there is an option to exclude columns:

`force.out = n [, n…]` Specifies one or more variables to be excluded from all models. As for **`force.in`**, columns are specified not by name but by order in the X matrix.

Output

As noted above, **`regsubsets`** does not directly produce any output. Some of its results can be gotten through the **`summary`** and **`plot`** functions. Of these, **`plot`** is the more useful, though its output is strange when first encountered.

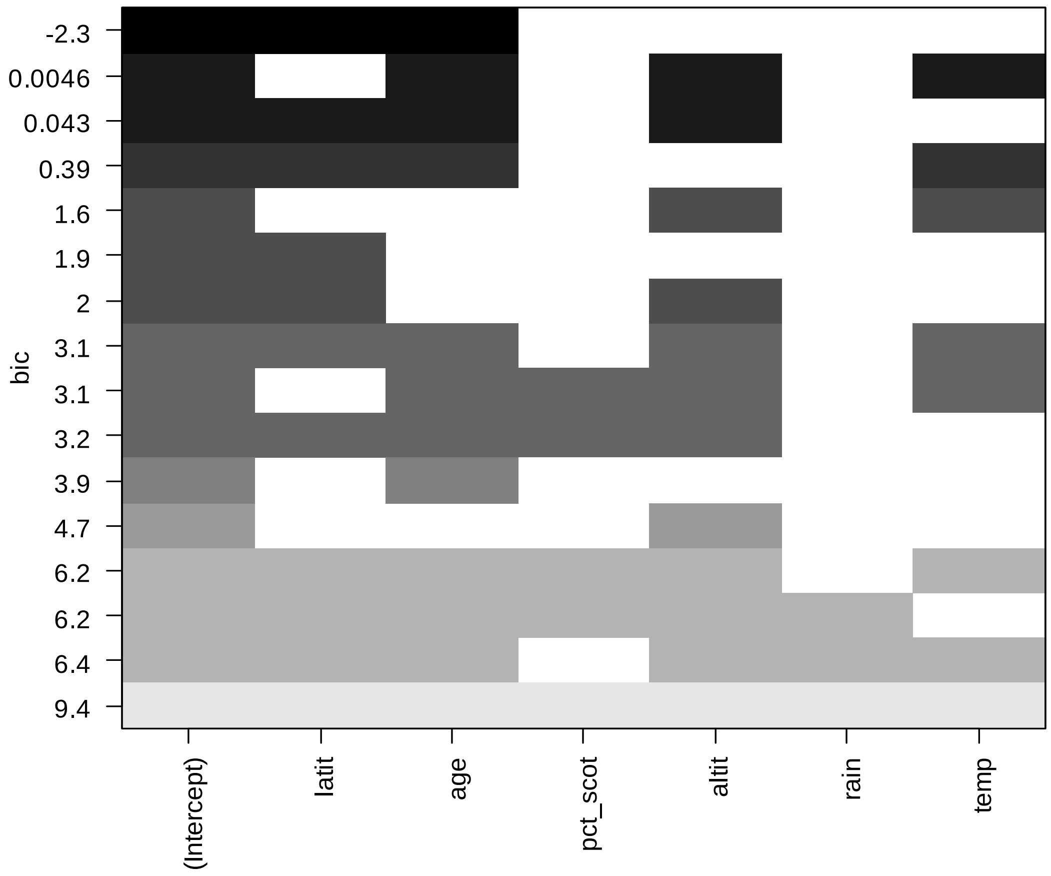
`plot`:

```
> plot(subsets)
```

where

`subsets` is an object created by **`regsubsets`**.

Here is an example of the plot this produces:



Each row in this graph represents a model; the shaded rectangles in the columns indicate the variables included in the given model. The numbers on the left margin are the values of Schwartz' Bayesian Information Criterion; not that the axis is not quantitative but is ordered. The darkness of the shading simply represents the ordering of the BIC values.

In the example above, the model with **`latit`** and **`age`** (and the intercept) has the lowest BIC (= -2.3). The next best model has **`age`**, **`altit`** and **`temp`**, with BIC = 0.0046; the third best has **`latit`**, **`age`** and **`altit`**, with BIC = 0.043; etc.

As shown above, the default criterion statistic produced by **`plot.regsubsets`** is BIC. Available alternatives are C_p, R^2, and R^2_{adj}. These are requested by the **`scale=`** option to the **`plot`** function, e.g.

```
> plot (subsets, scale="Cp")
```

`scale="xx"` where **`"xx"`** is either **`"Cp"`**, **`"adjr2"`**, **`"r2"`** or **`"bic"`**

`summary`:

If you simply enter the name of an object created by **`regsubsets`**

```
> subsets
```

you only get a listing of the command used to create the object.

To get some of the actual results, use

```
> summary(subsets)
```

which produces output like the following:

```
Subset selection object
Call: regsubsets.formula(lambda ~ latit + age + pct_scot +
    altit, data = loopers, nbest = 2)
4 Variables  (and intercept)
         Forced in Forced out
latit        FALSE      FALSE
age          FALSE      FALSE
pct_scot     FALSE      FALSE
altit        FALSE      FALSE
2 subsets of each size up to 4
Selection Algorithm: exhaustive
         latit age pct_scot altit
1  ( 1 ) "*"   " " " "      " "
1  ( 2 ) " "   "*" " "      " "
2  ( 1 ) "*"   "*" " "      " "
2  ( 2 ) "*"   " " " "      "*"
3  ( 1 ) "*"   "*" " "      "*"
3  ( 2 ) "*"   "*" "*"      " "
4  ( 1 ) "*"   "*" "*"      "*"
```

The useful part of this is the last, a "matrix" indicating which variables are included in each of the models. Models are listed in order of size (the first column), and within a size, in order of fit (best model first). The included variables are indicated by asterisks in quotes; variables not in a model have empty quotes. So in the preceding, the first model includes only **`latit`**, while the fourth model (second best two-variable model) includes **`latit`** and **`altit`**.

An option is available which makes this matrix perhaps more readable:

`matrix.logical=TRUE` (the default is FALSE)

When this is included in the call of **`summary`**, the last part of the output looks as follows:

```
          latit   age pct_scot altit
1  ( 1 )  TRUE FALSE    FALSE FALSE
1  ( 2 ) FALSE  TRUE    FALSE FALSE
2  ( 1 )  TRUE  TRUE    FALSE FALSE
2  ( 2 )  TRUE FALSE    FALSE  TRUE
3  ( 1 )  TRUE  TRUE    FALSE  TRUE
3  ( 2 )  TRUE  TRUE     TRUE FALSE
4  ( 1 )  TRUE  TRUE     TRUE  TRUE
```

Other output

As far as I can tell, the BIC values in the plot are the only statistics **`regsubsets`** directly produces for the models it finds as best. To get other statistics you can fit each model separately, using the **`lm`** function (which can be done through R-Commander). E.g.

```
> modeln <- lm(y ~ x1 + x2 …)
```

Then the various other criterion statistics can be gotten as follows:

AIC and BIC:

There is a function that gives AIC or Schwartz' BIC, from an **`lm`** object:

```
> AIC(modeln)
```

The default from this is the AIC; to get BIC add the option

`k=log(n)` where ***n*** is the sample size (there probably is some neat function you could specify in place of ***n*** that would get the sample size for you, e.g. **`length(y)`**)

Cp:

I don't think C_p can be gotten from an **`lm`** object, but since C_p orders the models exactly the same as does AIC, there's no real need to get C_p. If you want C_p nonetheless, see the **`leaps`** function below.

PRESS:

Function **`press`** in package **`DAAG`** will return the PRESS statistic for a model object produced by **`lm`**. It is invoked simply as

```
> press(modeln)
```

Adjusted R-Squared:

This is part of the output produced by

```
> summary(modeln)
```

Residual SS:

Any of the criterion statistics can be calculated from the Residual SS. The RSS can be gotten from the standard output of **`lm`** as the square of the "**`Residual standard`**

`error`" multiplied by its degrees of freedom. Alternatively the RSS of a **`lm`** object can be gotten by

```
aov(modeln)
```

leaps

This function is quite similar to **`regsubsets`** but is less flexible in how it is invoked; on the other hand, more results can be gotten from it more easily. The basic invocation is:

```
leaps(x=Xmat, y=Yvec)
```

where

> ***`Xmat`*** is a matrix of the explanatory variables (as columns)
>
> ***`Yvec`*** is the response variable, as a vector

The default statistic produced by **`leaps`** is C_p. R^2 or R^2_{adj} can be requested instead, using the options

```
method = "adjr2"
method = "r2"
```

Another useful option is

`nbest = n`	which specifies how many models of each size to report; the default is 10 so you probably often will want to lower it.

Output

`leaps` produces an object rather than any immediate output, so as with **`regsubsets`**, you will need to assign it to an object when running the function. This object is a list which can be printed simply by entering its name at the command prompt.

```
> leapmodels <- leaps(z=X, y=Y, nbest=3)
> leapmodels
```

will produce output like

```
$which
      1     2     3     4     5     6
1  TRUE FALSE FALSE FALSE FALSE FALSE
1 FALSE  TRUE FALSE FALSE FALSE FALSE
1 FALSE FALSE FALSE  TRUE FALSE FALSE
2  TRUE  TRUE FALSE FALSE FALSE FALSE
2 FALSE FALSE FALSE  TRUE FALSE  TRUE
2  TRUE FALSE FALSE  TRUE FALSE FALSE
3 FALSE  TRUE FALSE  TRUE FALSE  TRUE
3  TRUE  TRUE FALSE  TRUE FALSE FALSE
3  TRUE  TRUE FALSE FALSE FALSE  TRUE
4 FALSE  TRUE  TRUE  TRUE FALSE  TRUE
4  TRUE  TRUE FALSE  TRUE FALSE  TRUE
4 FALSE  TRUE FALSE  TRUE  TRUE  TRUE
```

```
5  TRUE  TRUE  TRUE  TRUE FALSE  TRUE
5 FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
5  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
6  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE

$label
[1] "(Intercept)" "1"         "2"         "3"         "4"
[6] "5"           "6"

$size
 [1] 2 2 2 3 3 3 4 4 4 5 5 5 6 6 6 7

$Cp
 [1]  4.375338951  6.417823236  7.387245661 -0.002910028
    3.031274707
 [6]  3.460269095  1.236672513  1.394466101  1.662057422
    3.079761089
[11]  3.190632108  3.229284093  5.018985467  5.079724973
    5.136339264
[16]  7.000000000
```

The first part of the output is a matrix like that produced by **`regsubsets`**, indicating which variables are included in each model. The second part gives the names of the variables (to get meaningful names here, you have to give them to **`leaps`** when you run it, as the **`names`** option), and the third part simply lists the sizes of the models.

The useful part of the output is the last one, which gives the values of C_p (or R^2 or R^2_{adj} instead, if one was requested) for the models. This is a simple vector, with entries in the order the models are listed in the matrix.

To get the results in an easier format, use something like

```
> cbind(leapmodels$size,leapmodels$matrix, leapmodels$Cp)
```

(where ***`leapmodel`*** is the object created when **`leap`** was run). This will produce output like

```
      1 2 3 4 5 6
1 2 1 0 0 0 0 0  4.375338951
1 2 0 1 0 0 0 0  6.417823236
1 2 0 0 0 1 0 0  7.387245661
2 3 1 1 0 0 0 0 -0.002910028
2 3 0 0 0 1 0 1  3.031274707
2 3 1 0 0 1 0 0  3.460269095
3 4 0 1 0 1 0 1  1.236672513
3 4 1 1 0 1 0 0  1.394466101
3 4 1 1 0 0 0 1  1.662057422
4 5 0 1 1 1 0 1  3.079761089
4 5 1 1 0 1 0 1  3.190632108
4 5 0 1 0 1 1 1  3.229284093
5 6 1 1 1 1 0 1  5.018985467
```

```
   5 6 0 1 1 1 1 1  5.079724973
   5 6 1 1 1 1 1 0  5.136339264
   6 7 1 1 1 1 1 1  7.000000000
```

Note that in this form the matrix specifying the models uses 1/0 coding for TRUE/FALSE. Also note that the first column of the matrix already gives the number of variables in the model, while the "size" vector gives the number of parameters (i.e. including the intercept).

Plots

The parts of the list produced by **`leaps`** also can be used in plotting functions. Most usefully and simply,

```
> plot(leapmodels$size, leapmodels$Cp)
> abline(0,1)
```

will produce a scatterplot of C_p *vs*. model size, which is very useful for identifying the model(s) for which C_p first approaches p.

Other output

As with **`regsubsets`**, models identified as good candidates by **`leaps`** can then by run in **`lm`** and various other criterion statistics obtained from that.

Mass processing of results of `regsubsets` or `leaps`

The "which" matrix produced by **`regsubsets`** or **`leaps`** can be used to run all (or some) of the models identified by those functions through other processing, e.g. run **`lm`** on them to get AIC and/or PRESS. Here's a sample program to do this (adapted from http://statstics/stanford.edu/~jtaylo/courses/stats191/R/selectionII/selectionII.Rout.html)

```
> submodels <- regsubsets(lambda ~ ., data=loopers,nbest=3)
> nmodel <- nrow(submodels$which)
> AIClist <- numeric(nmodel)
> PRESSlist <- numeric(nmodel)
> for (i in 1:nmodel) {
+
```