Model Selection in R

We will work again with the data from Problem 6.9, “Grocery Retailer.” Recall that we formed a data table named Grocery consisting of the variables Hours, Cases, Costs, and Holiday. We ran a full linear model which we named Retailer involving Hours as the response variable and Cases, Costs and Holiday as three predictor variables. For this example, we can have a sub-model which includes only *X1*, or only *X2*, or only *X3*, or just *X1* and *X2*, or just *X1* and *X3*, or just *X2* and *X3*, or all three variables. We can also complicate matters by including powers of these variables (appropriately centered), or interactions like *X1X2*, or other transformations (square roots, logs, etc.), but then we would have to obtain another full model which includes all these variables.

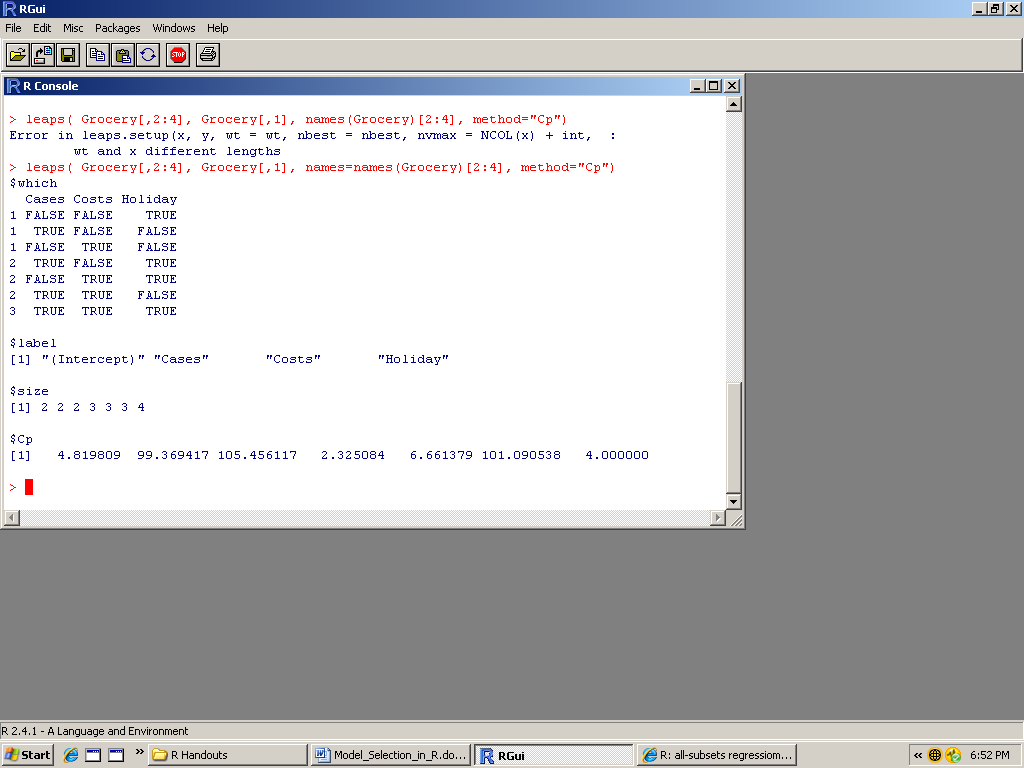
Our first model selection tool is the R function leaps(). This is found in the package leaps, which must first be loaded:

> library(leaps)

If R can't find the package you will need to go to the R repository via the Packages menu and the Install package(s)… option to download it and install it. The leaps() function will search for the best subsets of your predictors using whichever criterion you designate. To use this function, we need to provide it with a matrix consisting of the predictor variables, a vector consisting of the response variable, the names of the predictor variables, and the criterion to use. For this example, the predictor variables are the second through fourth columns of the table Grocery, i.e., Grocery[,2:4], and the response variable is the first column, i.e., Grocery[,1]. The names of the predictors are contained in the second through fourth elements of the vector names(Grocery), i.e., names(Grocery)[2:4]. Suppose we use the Mallows' *Cp* criterion for model selection. Then the R command to use is:

> leaps( x=Grocery[,2:4], y=Grocery[,1], names=names(Grocery)[2:4], method="Cp")

We get the following output:



The first part of the output, denoted $which, lists seven possible sub-models in seven rows. The first column indicates the number of predictors in the sub-model for each row. The variables in each sub-model are those designated TRUE in each row. For example, the first sub-model includes just the variable Holiday (note that Cases and Costs are FALSE and Holiday is TRUE). The next two parts of the output don't give us any new information, but the last part, designated $Cp, gives us the value of the Mallows' *Cp* criterion for each sub-model, in the same order. The best sub-model is that for which the *Cp* value is closest to *p* (the number of parameters in the model, including the intercept). For the full model, we always have *Cp* = *p*. The idea is to find a suitable reduced model, if possible. Here the best reduced model is the third one, consisting of Cases and Holiday, for which *Cp* = 2.325084 and *p =* 3.

Instead of using the Mallows' *Cp* criterion, we can use the *R*2 or the adjusted *R*2 criteria. Just use method="r2" or method="adjr2", respectively, in place of method="Cp" as the last function argument. The highest value for either criteria indicates the best sub-model.

A less-attractive alternative to using the leaps() function would be to make a list of each sub-model you wish to consider, then fit a linear model for each sub-model individually to obtain the selection criteria for that model. One way to make this easier is to start with the full model, the use the update() function to remove and/or add predictors step-by-step. For instance, we could start with our full model Retailer and delete just one variable, Costs. Then we fit a new model named NewMod with only the remaining predictors. The R command is

> NewMod <- update( Retailer, .~. – Costs )

Then if we want to modify NewMod so that we obtain another new model with both Costs and Cases deleted, the R command would be:

> NewMod <- update( NewMod, .~. – Cases)

If you then want to add Costs back into the model (but not Cases), the R command would be:

> NewMod <- update( NewMod, .~. + Costs)

At each step you can obtain the values for *Rp*2 and *Ra*,*p*2 from the model summary, which are given as “Multiple R-Squared” and “Adjusted R-squared,” respectively. So record those next to the corresponding subset. But to obtain the Mallows’ *Cp* criterion for each sub-model, you need your calculator. You also need the value for *MSE* when you fit the full model with all the potential predictor variables (in our example, that would be *MSE* = 20532), and the value of *n* (the number of observations in the data set, which is 52 in our example). Then find the value of *SSE* for the sub-model in the ANOVA table for that sub-model. That will be substituted for *SSEp* in formula **(9.9)** on p.358. Divide this by the *MSE* from the full model, then subtract from the result the value of *n* – 2*p*, where *p* is one more than the number of variables in your sub-model. The result is the value of *Cp* for that sub-model.

To obtain the *AICp* criterion for any sub-model, you will have to obtain a linear fit involving just the predictors for that sub-model, as described above. Then type

> extractAIC(model)

but put the name you have given the sub-model in place of model. Likewise, to obtain the *SBCp* criterion (also called *BICp*), type

> extractAIC(model, k = log(n))

but put the value of *n* for your data set in place of *n*, and put the name of your sub-model in place of model. We would choose the sub-model that minimizes these two values.

To obtain the *PRESSp* criterion for each sub-model, type

> sum((model$residuals/(1-hatvalues(model)))^2)

but put the name of your sub-model in place of model. Be careful with the parentheses. We would choose the sub-model that minimizes this value.

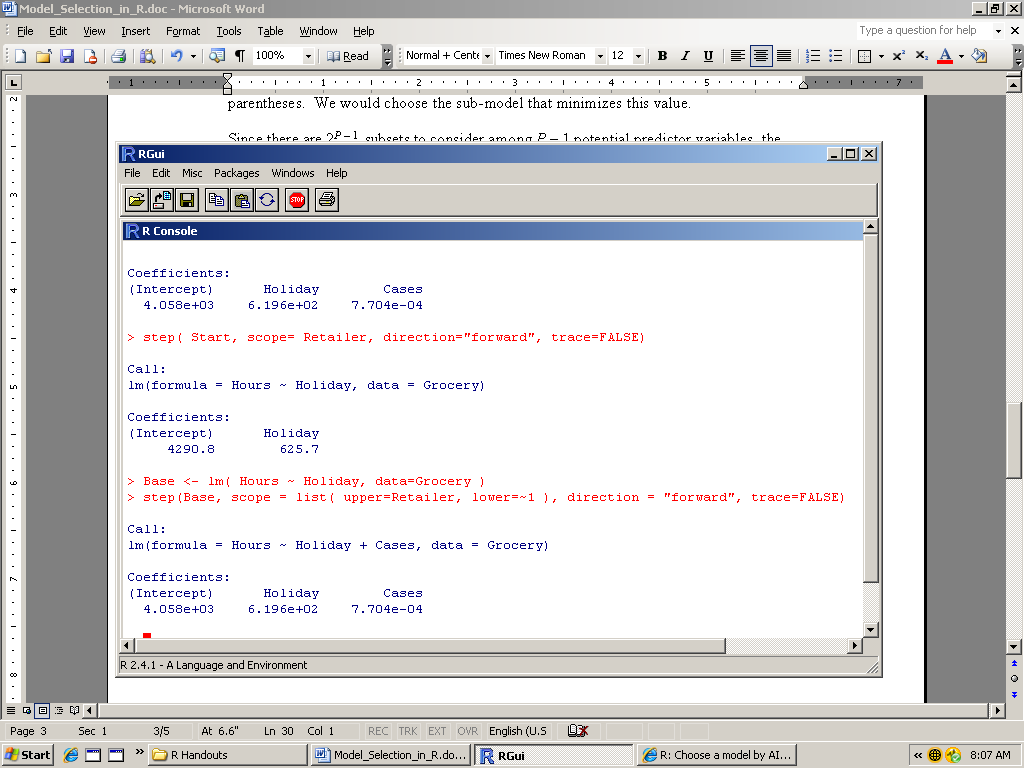
Since there are 2*P* – 1  subsets to consider among *P* – 1 potential predictor variables, the above process can become very tedious and time consuming when there are four or more predictors. One way around this in R is to use stepwise regression. You can do forward stepwise regression, backward stepwise regression, or a combination of both, but R uses the *AICp* criterion at each step instead of the criteria described in the text. To use this procedure in the forward direction, you first must fit a base model (with one predictor) and a full model (with all the predictors you wish to consider). To fit a base model in our example, we will choose Holiday as our predictor, since we are certain this variable should be included in our final model:

Base <- lm( Hours ~ Holiday, data=Grocery )

We will use Retailer as our full model. Then to implement forward stepwise regression, type

> step(Base, scope = list( upper=Retailer, lower=~1 ), direction = "forward", trace=FALSE)

but use the name of *your* base model and full model, respectively, in place of Base and Retailer. The output for our example looks like:

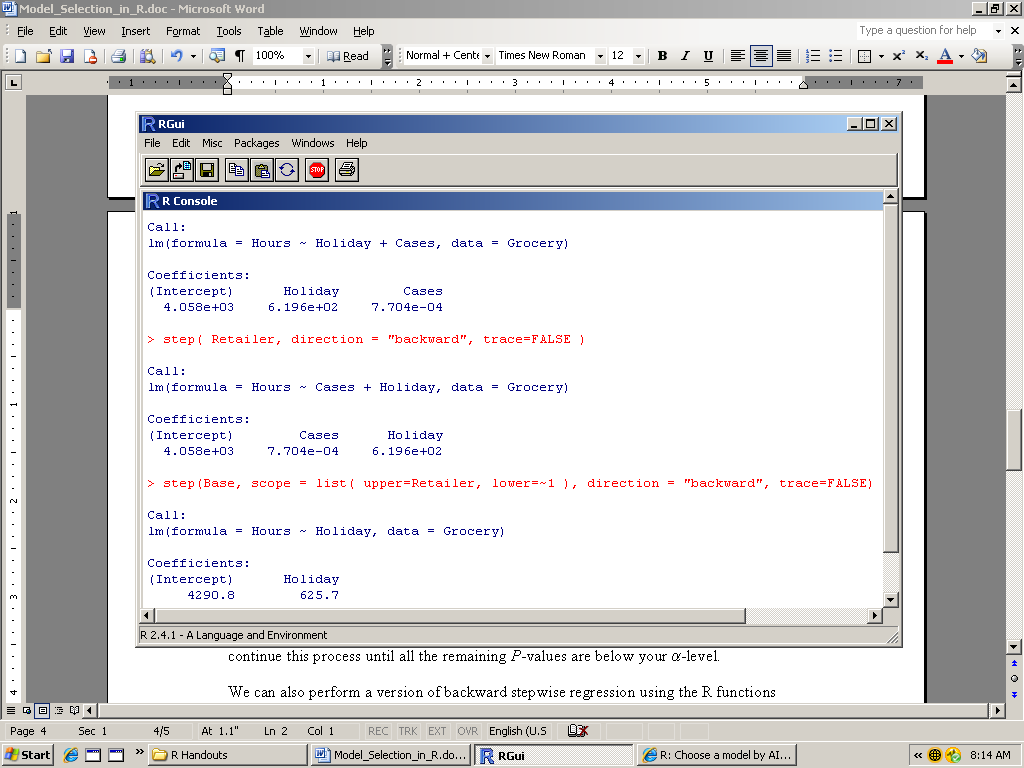


The forward stepwise regression procedure identified the model which included the two predictors Holiday and Cases, but not Costs, as the one which produced the lowest value of AIC.

To use the same procedure in the backward direction, the command is much simpler, since the full model is the base model. We just type:

> step( Retailer, direction = "backward", trace=FALSE )

but use the name of *your* full model, with all potential predictor variables included, in place of Retailer. The output for our example looks like:



The backward elimination procedure also identified the best model as one which includes only Cases and Holiday, not Costs. You can also run both procedures in succession by typing "both" in place of "forward" after direction= in the forward stepwise regression command, i.e.,

step(Base, scope = list( upper=Retailer, lower=~1 ), direction = "both", trace=FALSE)

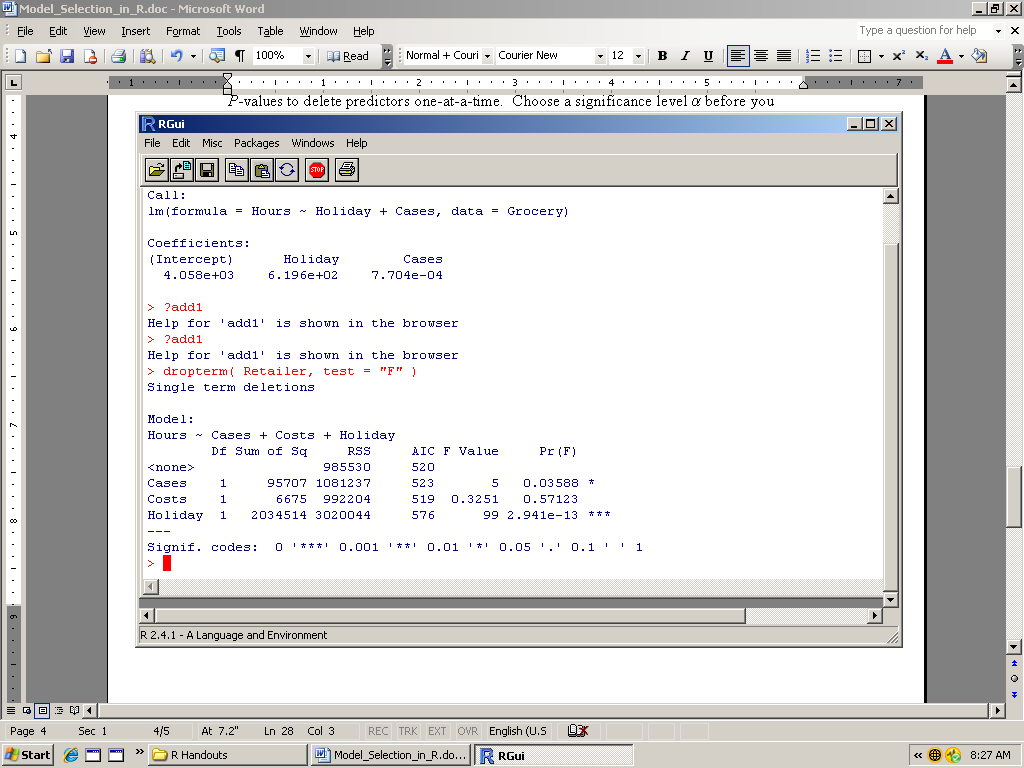
This is probably the best way to go. If you prefer to see the results at each step, regardless of direction, change the last setting to trace=TRUE.

Instead of using the AIC criterion, we can perform a backward stepwise regression using *P*-values to delete predictors one-at-a-time. Choose a significance level before you begin. Then start with the full model, look at the corresponding model summary, and then identify the predictor (if any) which has the largest *P*-value (for the *t* test) above your -level. Then fit a new linear model with that predictor deleted (use the update() function to make this easier). Now look at the model summary corresponding to the new model, and again identify the predictor for which the *P*-value (for the *t* test) is largest (but not smaller than your -level). Fit a new linear model with that predictor deleted, and continue this process until all the remaining *P*-values are below your -level.

We can also perform a version of backward stepwise regression using the R functions addterm() and dropterm() in the MASS package. To load them, use the library(MASS) command. These functions allow you to use an *F*-test criterion or a *P*-value criterion. You should have an *F* limit and/or an -level chosen ahead of time. Start with your full model, and use the R command:

> dropterm( Retailer, test = "F" )

with the name of your full model in place of Retailer. We get:



Then identify and delete the predictor (if any) with the smallest *F*-value below your *F* limit, or the largest *P*-value above your -level. For example, if the *F*-limit to delete a variable is 2.9, the obvious candidate for deletion is Costs, with an *F*-value of 0.3251. Likewise, if we are using an -level of 0.05 for deletion, we would delete Costs. Then we fit a new linear model, call it NewMod, with Costs deleted (use the update() function to make this easier), and use the command:

> dropterm( NewMod, test = "F" )

and repeat the process until all *F*-values are larger than your *F* limit or all *P*-values are below your -level.

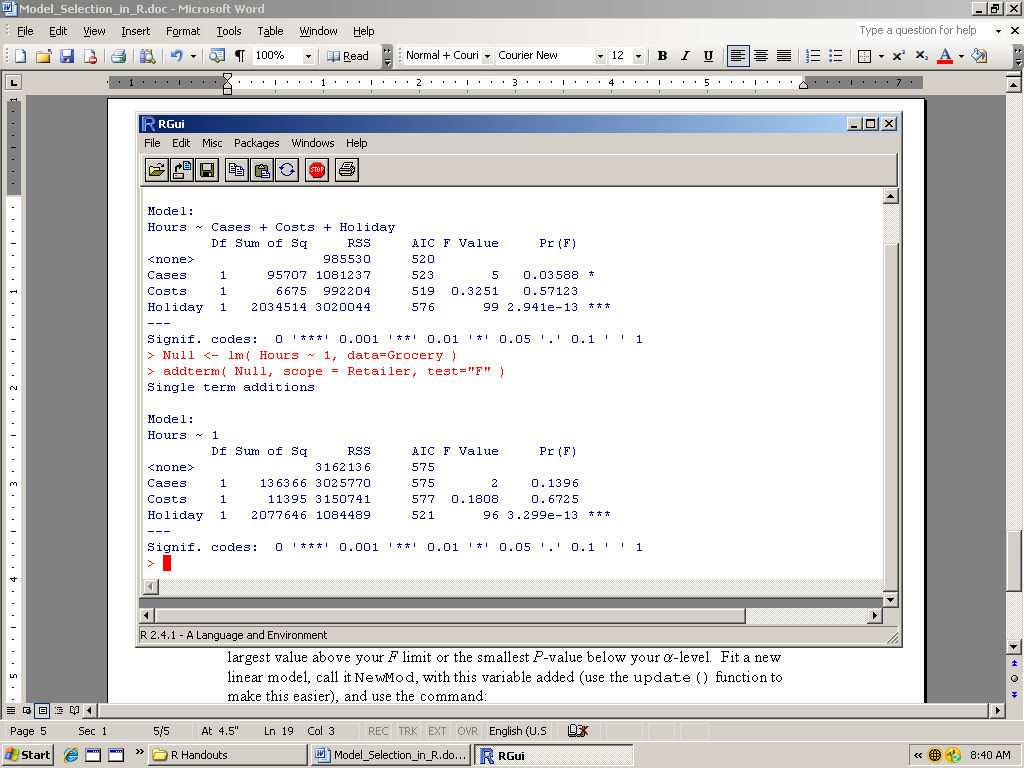
Similarly, one may start with a reduced model and use the R command addterm() to choose a variable for admission. If you want to begin with a null model consisting of no predictors (just the intercept), use ~ 1 in the model formula, i.e., put 1 where you usually put the names of the predictors:

Null <- lm( Hours ~ 1, data=Grocery )

Of course, use the name of your response variable in place of Hours and your data table in place of Grocery. Then use the commnad:

> addterm( Null, scope = Retailer, test="F" )

with the name of your full model in place of Retailer. We get:



Then identify and admit the predictor (if any) with the largest value above your *F* limit or the smallest *P*-value below your -level. For example, if the *F*-limit to admit a variable is 3.0, the obvious candidate for admission is Holiday, with an *F*-value of 96. Likewise, if we are using an -level of 0.05 for admission, we would admit Holiday. Fit a new linear model, call it NewMod, with this variable added (use the update() function to make this easier), and use the command:

> addterm( NewMod, scope = FullModel, test = "F" )

and repeat the process until all *F*-values are larger than your *F* limit or all *P*-values are below your -level.

One can also combine both functions to check as many sub-models as seems reasonable by moving backward and forward. Of course, when there are many variables this becomes impractical. Also, one can use the functions add1() and drop1() instead of addterm() and dropterm(), respectively, in the same manner. You don't need to load any additional packages to call these functions.