15.10: Model Selection

One fairly major problem that remains is the problem of "model selection". That is, if we have a data set that contains several variables, which ones should we include as predictors, and which ones should we not include? In other words, we have a problem of ***variable selection***. In general, model selection is a complex business, but it's made somewhat simpler if we restrict ourselves to the problem of choosing a subset of the variables that ought to be included in the model. Nevertheless, I'm not going to try covering even this reduced topic in a lot of detail. Instead, I'll talk about two broad principles that you need to think about; and then discuss one concrete tool that R provides to help you select a subset of variables to include in your model. Firstly, the two principles:

- It's nice to have an actual substantive basis for your choices. That is, in a lot of situations you the researcher have good reasons to pick out a smallish number of possible regression models that are of theoretical interest; these models will have a sensible interpretation in the context of your field. Never discount the importance of this. Statistics serves the scientific process, not the other way around.
- To the extent that your choices rely on statistical inference, there is a trade off between simplicity and goodness of fit. As you add more predictors to the model, you make it more complex; each predictor adds a new free parameter (i.e., a new regression coefficient), and each new parameter increases the model's capacity to "absorb" random variations. So the goodness of fit (e.g., R^2) continues to rise as you add more predictors no matter what. If you want your model to be able to generalise well to new observations, you need to avoid throwing in too many variables.

This latter principle is often referred to as ***Ockham's razor***, and is often summarised in terms of the following pithy saying: *do not multiply entities beyond necessity*. In this context, it means: don't chuck in a bunch of largely irrelevant predictors just to boost your R^2. Hm. Yeah, the original was better.

In any case, what we need is an actual mathematical criterion that will implement the qualitative principle behind Ockham's razor in the context of selecting a regression model. As it turns out there are several possibilities. The one that I'll talk about is the ***Akaike information criterion*** (AIC; Akaike 1974) simply because it's the default one used in the R function `step()`. In the context of a linear regression model (and ignoring terms that don't depend on the model in any way!), the AIC for a model that has K predictor variables plus an intercept is:[227]

$$\text{AIC} = \frac{\text{SS}_{res}^2}{\hat{\sigma}} + 2K$$

The smaller the AIC value, the better the model performance is. If we ignore the low level details, it's fairly obvious what the AIC does: on the left we have a term that increases as the model predictions get worse; on the right we have a term that increases as the model complexity increases. The best model is the one that fits the data well (low residuals; left hand side) using as few predictors as possible (low K; right hand side). In short, this is a simple implementation of Ockham's razor.

15.10.1 Backward elimination

Okay, let's have a look at the `step()` function at work. In this example I'll keep it simple and use only the basic ***backward elimination*** approach. That is, start with the complete regression model, including all possible predictors. Then, at each "step" we try all possible ways of removing one of the variables, and whichever of these is best (in terms of lowest AIC value) is accepted. This becomes our new regression model; and we then try all possible deletions from the new model, again choosing the option with lowest AIC. This process continues until we end up with a model that has a lower AIC value than any of the other possible models that you could produce by deleting one of its predictors. Let's see this in action. First, I need to define the model from which the process starts.

```
full.model <- lm( formula = dan.grump ~ dan.sleep + baby.sleep + day,
                  data = parenthood
 )
```

That's nothing terribly new: yet another regression. Booooring. Still, we do need to do it: the `object` argument to the `step()` function will be this regression model. With this in mind, I would call the `step()` function using the following command:

```
step( object = full.model,     # start at the full model
      direction = "backward"   # allow it remove predictors but not add them
)
```

```
## Start:  AIC=299.08
## dan.grump ~ dan.sleep + baby.sleep + day
##
##              Df Sum of Sq    RSS    AIC
## - baby.sleep  1       0.1 1837.2 297.08
## - day         1       1.6 1838.7 297.16
## <none>                    1837.1 299.08
## - dan.sleep   1    4909.0 6746.1 427.15
##
## Step:  AIC=297.08
## dan.grump ~ dan.sleep + day
##
##             Df Sum of Sq    RSS    AIC
## - day        1       1.6 1838.7 295.17
## <none>                   1837.2 297.08
## - dan.sleep  1    8103.0 9940.1 463.92
##
## Step:  AIC=295.17
## dan.grump ~ dan.sleep
##
##             Df Sum of Sq    RSS    AIC
## <none>                   1838.7 295.17
## - dan.sleep  1    8159.9 9998.6 462.50
```

```
##
## Call:
## lm(formula = dan.grump ~ dan.sleep, data = parenthood)
##
## Coefficients:
## (Intercept)    dan.sleep
##     125.956       -8.937
```

although in practice I didn't need to specify `direction` because `"backward"` is the default. The output is somewhat lengthy, so I'll go through it slowly. Firstly, the output reports the AIC value for the current best model:

```
Start:  AIC=299.08
dan.grump ~ dan.sleep + baby.sleep + day
```

That's our starting point. Since small AIC values are good, we want to see if we can get a value smaller than 299.08 by deleting one of those three predictors. So what R does is try all three possibilities, calculate the AIC values for each one, and then print out a short table with the results:

```
             Df Sum of Sq    RSS    AIC
- baby.sleep  1       0.1 1837.2 297.08
- day         1       1.6 1838.7 297.16
<none>                   1837.1 299.08
- dan.sleep   1    4909.0 6746.1 427.15
```

To read this table, it helps to note that the text in the left hand column is telling you what *change* R made to the regression model. So the line that reads `<none>` is the actual model we started with, and you can see on the right hand side that this still corresponds to an AIC value of 299.08 (obviously). The other three rows in the table correspond to the other three models that it looked at: it tried removing the `baby.sleep` variable, which is indicated by `- baby.sleep`, and this produced an AIC value of 297.08. That was the best of the three moves, so it's at the top of the table. So, this move is accepted, and now we start again. There are two predictors left in the model, `dan.sleep` and `day`, so it tries deleting those:

```
Step:  AIC=297.08
dan.grump ~ dan.sleep + day

            Df Sum of Sq    RSS    AIC
- day        1       1.6 1838.7 295.17
<none>                   1837.2 297.08
- dan.sleep  1    8103.0 9940.1 463.92
```

Okay, so what we can see is that removing the `day` variable lowers the AIC value from 297.08 to 295.17. So R decides to keep that change too, and moves on:

```
Step:  AIC=295.17
dan.grump ~ dan.sleep

            Df Sum of Sq    RSS    AIC
<none>                   1838.7 295.17
- dan.sleep  1    8159.9 9998.6 462.50
```

This time around, there's no further deletions that can actually improve the AIC value. So the `step()` function stops, and prints out the result of the best regression model it could find:

```
Call:
lm(formula = dan.grump ~ dan.sleep, data = parenthood)

Coefficients:
(Intercept)    dan.sleep
    125.956       -8.937
```

which is (perhaps not all that surprisingly) the `regression.1` model that we started with at the beginning of the chapter.

15.10.2 Forward selection

As an alternative, you can also try ***forward selection***. This time around we start with the smallest possible model as our start point, and only consider the possible additions to the model. However, there's one complication: you also need to tell `step()` what the largest possible model you're willing to entertain is, using the `scope` argument. The simplest usage is like this:

```
null.model <- lm( dan.grump ~ 1, parenthood )   # intercept only.
step( object = null.model,     # start with null.model
      direction = "forward",   # only consider "addition" moves
      scope =  dan.grump ~ dan.sleep + baby.sleep + day  # largest model allowed
)
```

```
## Start:  AIC=462.5
## dan.grump ~ 1
##
##              Df Sum of Sq    RSS    AIC
## + dan.sleep   1    8159.9 1838.7 295.17
## + baby.sleep  1    3202.7 6795.9 425.89
## <none>                    9998.6 462.50
## + day         1      58.5 9940.1 463.92
##
## Step:  AIC=295.17
## dan.grump ~ dan.sleep
##
##              Df Sum of Sq    RSS    AIC
## <none>                    1838.7 295.17
## + day         1   1.55760 1837.2 297.08
## + baby.sleep  1   0.02858 1838.7 297.16
```

```
##
## Call:
## lm(formula = dan.grump ~ dan.sleep, data = parenthood)
##
## Coefficients:
## (Intercept)    dan.sleep
##     125.956       -8.937
```

If I do this, the output takes on a similar form, but now it only considers addition (`+`) moves rather than deletion (`-`) moves:

```
Start:  AIC=462.5
dan.grump ~ 1

             Df Sum of Sq    RSS    AIC
+ dan.sleep   1    8159.9 1838.7 295.17
+ baby.sleep  1    3202.7 6795.9 425.89
<none>                    9998.6 462.50
+ day         1      58.5 9940.1 463.92

Step:  AIC=295.17
dan.grump ~ dan.sleep

             Df Sum of Sq    RSS    AIC
<none>                    1838.7 295.17
+ day         1   1.55760 1837.2 297.08
+ baby.sleep  1   0.02858 1838.7 297.16

Call:
lm(formula = dan.grump ~ dan.sleep, data = parenthood)

Coefficients:
(Intercept)    dan.sleep
    125.956       -8.937
```

As you can see, it's found the same model. In general though, forward and backward selection don't always have to end up in the same place.

15.10.3 caveat

Automated variable selection methods are seductive things, especially when they're bundled up in (fairly) simple functions like `step()`. They provide an element of objectivity to your model selection, and that's kind of nice. Unfortunately, they're sometimes used as an excuse for thoughtlessness. No longer do you have to think carefully about which predictors to add to the model and what the theoretical basis for their inclusion might be… everything is solved by the magic of AIC. And if we start throwing around phrases like Ockham's razor, well, it sounds like everything is wrapped up in a nice neat little package that no-one can argue with.

Or, perhaps not. Firstly, there's very little agreement on what counts as an appropriate model selection criterion. When I was taught backward elimination as an undergraduate, we used F-tests to do it, because that was the default method used by the software. The default in the `step()` function is AIC, and since this is an introductory text that's the only method I've described, but the AIC is hardly the Word of the Gods of Statistics. It's an approximation, derived under certain assumptions, and it's guaranteed to work only for large samples when those assumptions are met. Alter those assumptions and you get a different criterion, like the BIC for instance. Take a different approach again and you get the NML criterion. Decide that you're a Bayesian and you get model selection based on posterior odds ratios. Then there are a bunch of regression specific tools that I haven't mentioned. And so on. All of these different methods have strengths and weaknesses, and some are easier to calculate than others (AIC is probably the easiest of the lot, which might account for its popularity). Almost all of them produce the same answers when the answer is "obvious" but there's a fair amount of disagreement when the model selection problem becomes hard.

What does this mean in practice? Well, you *could* go and spend several years teaching yourself the theory of model selection, learning all the ins and outs of it; so that you could finally decide on what you personally think the right thing to do is. Speaking as someone who actually did that, I wouldn't recommend it: you'll probably come out the other side even more confused than when you started. A better strategy is to show a bit of common sense… if you're staring at the results of a `step()` procedure, and the model that makes sense is close to having the smallest AIC, but is narrowly defeated by a model that doesn't make any sense… trust your instincts. Statistical model selection is an inexact tool, and as I said at the beginning, *interpretability matters*.

15.10.4 Comparing two regression models

An alternative to using automated model selection procedures is for the researcher to explicitly select two or more regression models to compare to each other. You can do this in a few different ways, depending on what research question you're trying to answer. Suppose we want to know whether or not the amount of sleep that my son got has any relationship to my grumpiness, over and above what we might expect from the amount of sleep that I got. We also want to make sure that the day on which we took the measurement has no influence on the relationship. That is, we're interested in the relationship between `baby.sleep` and `dan.grump`, and from that perspective `dan.sleep` and `day` are nuisance variable or ***covariates*** that we want to control for. In this situation, what we would like to know is whether `dan.grump ~ dan.sleep + day + baby.sleep` (which I'll call Model 1, or `M1`) is a better regression model for these data than `dan.grump ~ dan.sleep + day` (which I'll call Model 0, or `M0`). There are two different ways we can compare these two models, one based on a model selection criterion like AIC, and the other based on an explicit hypothesis test. I'll show you the AIC based approach first because it's simpler, and follows naturally from the `step()` function that we saw in the last section. The first thing I need to do is actually run the regressions:

```
M0 <- lm( dan.grump ~ dan.sleep + day, parenthood )
M1 <- lm( dan.grump ~ dan.sleep + day + baby.sleep, parenthood )
```

Now that I have my regression models, I could use the `summary()` function to run various hypothesis tests and other useful statistics, just as we have discussed throughout this chapter. However, since the current focus on model comparison, I'll skip this step and go straight to the AIC calculations. Conveniently, the `AIC()` function in R lets you input several regression models, and it will spit out the AIC values for each of them:[228]

```
AIC( M0, M1 )
```

```
##    df      AIC
## M0  4 582.8681
## M1  5 584.8646
```

Since Model 0 has the smaller AIC value, it is judged to be the better model for these data.

A somewhat different approach to the problem comes out of the hypothesis testing framework. Suppose you have two regression models, where one of them (Model 0) contains a *subset* of the predictors from the other one (Model 1). That is, Model 1 contains all of the predictors included in Model 0, plus one or more additional predictors. When this happens we say that Model 0 is ***nested*** within Model 1, or possibly that Model 0 is a ***submodel*** of Model 1. Regardless of the terminology what this means is that we can think of Model 0 as a null hypothesis and Model 1 as an alternative hypothesis. And in fact we can construct an F test for this in a fairly straightforward fashion. We can fit both models to the data and obtain a residual sum of squares for both models. I'll denote these as $SS_{res}^{(0)}$ and $SS_{res}^{(1)}$ respectively. The superscripting here just indicates which model we're talking about. Then our F statistic is

$$F = \frac{\left(\mathrm{SS}_{res}^{(0)} - \mathrm{SS}_{res}^{(1)}\right)/k}{\left(\mathrm{SS}_{res}^{(1)}\right)/(N-p-1)}$$

where N is the number of observations, p is the number of predictors in the full model (not including the intercept), and k is the difference in the number of parameters between the two models.[229] The degrees of freedom here are k and N−p−1. Note that it's often more convenient to think about the difference between those two SS values as a sum of squares in its own right. That is:

$$\mathrm{SS}_{\Delta} = \mathrm{SS}_{res}^{(0)} - \mathrm{SS}_{res}^{(1)}$$

The reason why this his helpful is that we can express SSΔ a measure of the extent to which the two models make different predictions about the the outcome variable. Specifically:

$$SS_{\Delta} = \sum_i (\hat{y}_i^{(1)} - \hat{y}_i^{(0)})^2$$

where $\hat{y}_i^{(0)}$ is the fitted value for y_i according to model M0 and $\hat{y}_i^{(1)}$ is the is the fitted value for yi according to model M_1.

Okay, so that's the hypothesis test that we use to compare two regression models to one another. Now, how do we do it in R? The answer is to use the `anova()` function. All we have to do is input the two models that we want to compare (null model first):

```
anova( M0, M1 )
```

```
## Analysis of Variance Table
##
## Model 1: dan.grump ~ dan.sleep + day
## Model 2: dan.grump ~ dan.sleep + day + baby.sleep
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     97 1837.2
## 2     96 1837.1  1  0.063688 0.0033 0.9541
```

Note that, just like we saw with the output from the `step()` function, R has used the acronym `RSS` to refer to the residual sum of squares from each model. That is, RSS in this output corresponds to SS_{res} in the formula above. Since we have p>.05 we retain the null hypothesis (`M0`). This approach to regression, in which we add all of our covariates into a null model, and then *add* the variables of interest into an alternative model, and then compare the two models in hypothesis testing framework, is often referred to as ***hierarchical regression***.