Model selection

1 Introduction

- Today's topic is model selection. Suppose we are given, or have generated, a "large" number of predictors $X \in \mathbb{R}^d$. ("Large" means different things to different people.) Our goal is to select a small subset to use in our final regression. In many cases, having fewer variables results in a more interpretable model, with more accurate parameter estimates and better predictions. (Of course, we will not always be so lucky!) This is also known as **variable selection** for obvious reasons.
- Broadly speaking, there are at least two ways of thinking about the problem of model selection.
  - Assume that the full model – i.e. the model with *all* of the predictors – is correct, but that most of the slopes are equal to 0. The goal then becomes to find the variables with non-zero slopes.
  - Assume nothing. Try to find the set of variables from which we can predict the response (using OLS) as accurately as possible.
- In this class we will go the second route. Our focus will be on selecting models that predict well. To make this concrete, we model our observed data as being realized values of random variables $(X_1, Y_1), \ldots, (X_n, Y_n)$ sampled i.i.d from an unknown distribution P_{XY}. In addition we suppose there is a test point (X_{n+1}, Y_{n+1}) coming from the same distribution P_{XY}, for which we observe only the predictors and not the response. Our goal is to use the observed data to form a predictive rule $\hat{m}$, such that $\hat{m}(X_{n+1})$ is a good prediction for Y_{n+1}.

2 Prediction error, training error, and cross-validation

- We begin by defining a notion of prediction error that allows us to measure how well a given fitted model predicts. As usual, we will focus on models fit by ordinary least squares.
- Notation: Let d be the total number of variables. Let $\mathcal{C} \subset \{1, ..., d\}$ refer to a **candidate model**, i.e. a subset of the overall variables. Call the number of variables in the model $p = |\mathcal{C}|$. Write $\hat{\beta} \in \mathbb{R}^p$ for the OLS coefficients computed using only this subset of variables and $\hat{Y}$ for the fitted values. (Notice that the notation hides how these depend on $\mathcal{C}$.)
- We would like to know how well the response can be predicted using the predictors in $\mathcal{C}$. At a given value of the selected predictors $x \in \mathbb{R}^p$, the fitted model predicts

$$\hat{m}(x) := \hat{\beta}_0 + \sum_{j=1}^{p} \hat{\beta}_j x_j.$$

- To measure the predictive accuracy of $\hat{m}$, we use mean-squared error. The expected **prediction (mean-squared) error** is

$$\mathrm{Err}(\mathcal{C}) = \mathbb{E}\Big[(Y_{n+1} - \hat{m}(X_{n+1}))^2\Big]$$

  We can now distill model selection into a precise mathematical goal: find the set of variables $\mathcal{C}$ that minimizes the prediction error $\mathrm{Err}(\mathcal{C})$.
- Of course $\mathrm{Err}(\mathcal{C})$ it itself an unknown quantity. So our strategy for selecting a good model will be to find an estimate $\widehat{\mathrm{Err}}$ of Err and then pick the model which minimizes that estimate.

- A natural first estimate for prediction error is the empirical mean-squared error at the observed data, sometimes called **training error** in the context of model selection:

$$\mathrm{MSE}_n(\mathcal{C}) := \frac{1}{n}\sum_{i=1}^{n}\big(Y_i - \hat{m}(X_i)\big)^2. \tag{1}$$

  However, there are (at least) two reasons why using $\mathrm{MSE}_n(\mathcal{C})$ as an estimate for $\mathrm{Err}(\mathcal{C})$ is a bad idea.

  - First, $\mathrm{MSE}_n(\mathcal{C})$ is always a systematic underestimate of $\mathrm{Err}(\mathcal{C})$. This is because $\hat{m}$ was fitted to the same data on which we are using to evaluate error.
  - Second, including more variables always results in a smaller (or at least, no bigger) MSE. So the **full model** with $\mathcal{C} = \{1, \ldots, d\}$ will always achieve the minimum MSE, even though it typically will *not* have the smallest prediction error.

- We can get a much better estimate of prediction error by leaving one point out at a time, i.e. by using the **jackknife**. Remember we introduced the notation $\hat{Y}_{(i)i}$ to refer the predicted value at predictor X_i of a regression trained on all points except (X_i, Y_i). The **jackknife** estimate of prediction error is simply:

$$\widehat{\mathrm{Err}}_{CV}(\mathcal{C}) := \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_{(i)i})^2 \tag{2}$$

  This is more commonly known as the leave-one-out **cross-validation** estimate of prediction error, abbreviated LOOCV. The only difference between (2) and (1) is that (2) uses leave-one-out predicted values instead of fitted values. This is no small difference. The LOOCV is a (nearly) unbiased estimate of the true generalization error $\mathrm{Err}(\mathcal{C})$, whereas the training error can be very biased.

- On its face, computing the LOOCV estimate of prediction error requires computing n OLS estimates, which is daunting when n and p are large. When discussing jackknife residuals I mentioned that there was a leave-one-out trick that allowed us to avoid doing this. The leave-one-out trick, also known as the *shortcut formula*, is $Y_i - \hat{Y}_{(i)i} = \frac{e_i}{1-\mathbf{H}_{ii}}$. Therefore

$$\widehat{\mathrm{Err}}_{CV}(\mathcal{C}) = \frac{1}{n}\sum_{i=1}^{n}\frac{(Y_i - \hat{Y}_i)^2}{(1-\mathbf{H}_{ii})^2}. \tag{3}$$

  We then choose the set of variables that makes the estimated prediction error as small as possible: $\widehat{\mathcal{C}} = \mathrm{argmin}_{\mathcal{C}}\, \widehat{\mathrm{Err}}_{CV}(\mathcal{C})$. Since there are 2^d number of models, heuristics such as **stepwise** regression are typically used to greedily search through the collection of possible models rather than looking at every single one. These are covered in the data analysis section of the notes.

- It is possible to do K-fold cross-validation rather than LOOCV. The idea is similar in spirit: hold n/K points out for evaluation and use the rest for training. This sometimes produces better estimates of $\mathrm{Err}(\mathcal{C})$ (higher bias, but lower variance) but there is no convenient leave-one-out shortcut. For statistical purposes, conventional wisdom often says to take $K = 10$.

- One of the great advantages of CV is that it can be applied to any estimator, not merely one based on linear regression. In the specific case of linear regression, and using some hand-wavy approximations characteristic of the second half of this course, we will see how LOOCV is approximately correcting for the downward bias in MSE.

3 The downward bias of training error, and optimism

- Cross-validation is only one – though arguably the most useful – estimate of prediction. Although nowadays CV is the most common method for estimating prediction error, it is still useful to discuss other estimates, particularly because a lot of the software still uses them. Most of these other estimates come from taking the training error $\mathrm{MSE}_n(\mathcal{C})$ and trying to correct the downwards bias. Therefore, to understand how the estimates work, we need to get a better handle on the bias $\mathbb{E}[\mathrm{MSE}_n(\mathcal{C})] - \mathrm{Err}(\mathcal{C})$.

- In general this bias calculation is quite difficult. One helpful simplification comes from our usual trick of conditioning on the predictors. We will also make the additional fudge of replacing the average over X_{n+1} by an empirical average over $X_1, \ldots, X_n$, that is we will look at the quantity

$$\mathrm{Err}_{|X}(\mathcal{C}) := \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\Big[\big(Y_{n+1} - \hat{m}(X_{n+1})\big)^2 | X_1, \ldots, X_n, X_{n+1} = X_i\Big]. \tag{4}$$

  This is sometimes called **in-sample prediction error**. It is not an intrinsically interesting quantity per se, but it is easier to work with than $\mathrm{Err}(\mathcal{C})$, and will usually be quite similar to $\mathrm{Err}(\mathcal{C})$ when n is large (by the law of large numbers).

- How badly biased is training error as an estimate of $\mathrm{Err}_{|X}(\mathcal{C})$? We do some calculations to find out. First we calculate the expectation of each term in the summand of (1):

$$\begin{aligned}
\mathbb{E}[(\hat{Y}_i - Y_i)^2 | X_1, \ldots, X_n] &= \mathbb{E}[(\hat{Y}_i - \beta_0 - X_i^\top \beta - \epsilon_i)^2 | X_1, \ldots, X_n] \\
&= \mathbb{E}[(\hat{Y}_i - \beta_0 - X_i^\top \beta)^2 | X_1, \ldots, X_n] + \mathbb{E}[\epsilon_i^2] - 2\mathbb{E}[(\hat{Y}_i - \beta_0 - X_i^\top \beta)\epsilon_i | X_1, \ldots, X_n] \\
&= \mathbb{E}[(\hat{Y}_i - \beta_0 - X_i^\top \beta)^2 | X_1, \ldots, X_n] + \sigma^2 - 2\mathrm{Cov}[\hat{Y}_i, \epsilon_i | X_1, \ldots, X_n]. \\
&= \mathbb{E}[(\hat{Y}_i - \beta_0 - X_i^\top \beta)^2 | X_1, \ldots, X_n] + \sigma^2 - 2\sigma^2 \mathbf{H}_{ii}.
\end{aligned}$$

  Then we calculate the expectation of each term in (4). This is simpler as the covariance between $\hat{Y}_i$ and Y_{n+1} is 0, and so

$$\mathbb{E}\Big[(\hat{Y}_i - Y_{n+1})^2 | X_1, \ldots, X_n, X_{n+1} = X_i\Big] = \mathbb{E}\Big[(\hat{Y}_i - \beta_0 - X_i^\top \beta)^2 | X_1, \ldots, X_n\Big] + \sigma^2.$$

  Taking the difference between these two and averaging over $i = 1, \ldots, n$ gives

$$\mathbb{E}\Big[\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 | X_1, \ldots, X_n\Big] = \mathrm{Err}_{|X}(\mathcal{C}) - \frac{2\sigma^2}{n}\sum_{i=1}^{n}\mathbf{H}_{ii} = \mathrm{Err}_{|X}(\mathcal{C}) - \frac{2\sigma^2(p+1)}{n}.$$

  As expected we see that the training error is biased low for the in-sample prediction error. The amount of this bias $2\sigma^2(p+1)/n$ is referred to as the **optimism** of the training error. The bias increases with the number of variables p.

- There is an interesting connection between LOOCV and optimism, which can be seen from the shortcut (3). Let us assume that each leverage is close to the average leverage: $\mathbf{H}_{ii} \approx \gamma = p/n$. Using this approximation, and the Taylor approximation $\frac{1}{(1-\gamma)^2} \approx 1 + 2\gamma$, we see that the LOOCV estimate of prediction error is reasonably close to

$$\begin{aligned}
\frac{1}{n}\sum_{i=1}^{n}\frac{(Y_i - \hat{Y}_i)^2}{(1 - \mathbf{H}_{ii})^2} &\approx \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 + \frac{2\gamma}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \\
&= \mathrm{MSE}(\mathcal{C}) + \frac{2p}{n} \cdot \mathrm{MSE}(\mathcal{C}).
\end{aligned}$$

  Viewing $\mathrm{MSE}(\mathcal{C})$ as a (biased) estimator for σ^2, we see how LOOCV is correcting for the optimism in training error.

4 Mallow's C_p

- Now we have calculated the optimism of training error, an obvious thing to do is try and estimate it. This is exactly the idea behind **Mallow's** C_p which is simply

$$C_p(\mathcal{C}) = \mathrm{MSE}_n(\mathcal{C}) + \frac{2(p+1)\hat{\sigma}^2}{n}.$$

  In this case the estimator $\hat{\sigma}^2$ used for error variance is typically based on the residuals of the full model, meaning the model trained on all d predictors. This will be unbiased so long as the usual assumptions are true, no matter how small the "true" model is (i.e. no matter how many values of $\beta_j = 0$.)

- If the estimator $\hat{\sigma}^2$ is an unbiased estimate of σ^2 then C_p is an unbiased estimate of the in-sample prediction error $\text{Err}_{|X}(\mathcal{C})$. Here we've shifted the goalposts a little bit since our original goal was to find a model with small prediction error, which is Err rather than $\text{Err}_{|X}$. But estimating prediction error is a harder problem.
- Faraway defines Mallow's C_p differently:[1]

$$C_p(\mathcal{C}) = \frac{n\text{MSE}_n(\mathcal{C})}{\hat{\sigma}^2} + 2(p+1) - n.$$

  You should convince yourself that the model chosen by either definition of Mallow's C_p will be the same.
- Here we have motivated Mallow's C_p by viewing it as an estimate of in-sample prediction error. It may be useful to take a broader view. Looking at Mallow's C_p we see that it is the sum of two terms: one involving the fit of the model to the data, the other a "penalty" which discourages larger models (models with larger values of p.) Suppose we are comparing two models $\mathcal{C}_1$ and $\mathcal{C}_2$, which differ by exactly one parameter. In order for the larger model to be preferred by Mallow's, the improvement in fit due to the inclusion of one extra variable must be larger than a certain amount ($2\hat{\sigma}^2/n$ to be precise). Otherwise Mallow's says that the extra variable isn't worth it.

5 Other criteria

- There are many other criteria, and we mention only a few of the other big ones.
- **Akaike's information criterion** (AIC) is defined for the more general problem of maximum likelihood estimation. If $\ell(\theta)$ is the log-likelihood of parameters β, then

$$\text{AIC}(\mathcal{C}) = -\ell(\hat{\theta}) + p + 1. \tag{5}$$

  Here ℓ is the log-likelihood of the data, evaluated at the maximum likelihood estimator $\hat{\theta}$ for model $\mathcal{C}$, and p is again the size of the model. In the context of linear models with Gaussian errors the MLE is exactly the least squares estimator, and the log-likelihood depends closely on MSE. In fact if we assume σ^2 is know, then $AIC(\mathcal{C})$ becomes

$$\text{AIC}(\mathcal{C}) = \frac{\text{MSE}_n(\mathcal{C})}{2n\sigma^2} + p + 1 + const$$

  where *const* is a term that does not depend on the model $\mathcal{C}$. This is exactly equivalent to Mallow's C_p, in the sense that they will always result in the same ordering of models. If σ^2 is not known the two criteria are not the same, but they are generally close.
- **Bayesian Information Criterion** (BIC) looks similar to AIC but with a stronger penalty:

$$\text{BIC}(\mathcal{C}) = -\ell(\hat{\theta}) + \frac{\log n}{2}(p+1)$$

  BIC will tend to prefer smaller models than AIC due to its harsher penalty. This comes with a tradeoff. On the one hand, BIC will tend to choose models that predict a little bit worse than AIC. On the other hand, if one of the candidate models is correct and under a pile of other conditions, it has been shown that BIC will tend to choose the correct model whereas AIC will not.
- R^2. Remember that R^2 can be defined in terms of training error: recalling that s_Y^2 is the empirical variance of Y,

$$R^2 = 1 - \frac{\text{MSE}_n(\mathcal{C})}{s_Y^2}$$

  So choosing the model with smallest R^2 is just as bad as choosing the model with largest training error.

[1] Faraway also uses p instead of $p+1$ to refer to the number of parameters.

- **Adjusted** R^2. Adjusted R^2 is defined similarly to R^2 but adjusts $\text{MSE}_n(\mathcal{C})$ upwards:

$$R^2_{adj} = 1 - \frac{\text{MSE}_n(\mathcal{C})}{s^2_Y} \cdot \frac{n}{n-p-1}.$$

The precise nature of the adjustment is chosen because if the model is correct then $\text{MSE}_n(\mathcal{C}) \cdot \frac{n}{n-p-1}$ is an unbiased estimate of σ^2. So maximizing adjusted R^2 corresponds to minimizing an unbiased estimate of the error variance. Using another Taylor expansion, we can write this as another "fit + penalty" criterion:

$$\text{MSE}_n(\mathcal{C}) \cdot \frac{n}{n-p-1} = \text{MSE}_n(\mathcal{C}) \cdot \frac{1}{1-\gamma} \approx \text{MSE}_n(\mathcal{C}) + \gamma \text{MSE}_n(\mathcal{C}) = \text{MSE}_n(\mathcal{C}) + \frac{p+1}{n}\text{MSE}_n(\mathcal{C}).$$

We see that the penalty is only about half as large as was the case for LOOCV, or Mallow's C_p. So although R^2_{adj} is more reasonable than R^2 it will still tend to select models that are too large from the standpoint of optimal prediction.

Highway data

We try out a dataset from Weisberg's *Applied Linear Regression*.

TABLE 10.5 Definition of Terms for the Highway Accident Data

Variable	Description
log(*Rate*)	Base-two logarithm of 1973 accident rate per million vehicle miles, the response
log(*Len*)	Base-two logarithm of the length of the segment in miles
log(*ADT*)	Base-two logarithm of average daily traffic count in thousands
log(*Trks*)	Base-two logarithm of truck volume as a percent of the total volume
Slim	1973 speed limit
Lwid	Lane width in feet
Shld	Shoulder width in feet of outer shoulder on the roadway
Itg	Number of freeway-type interchanges per mile in the segment
log(*Sigs1*)	Base-two logarithm of (number of signalized interchanges per mile in the segment + 1)/(length of segment)
Acpt	Number of access points per mile in the segment
Hwy	A factor coded 0 if a federal interstate highway, 1 if a principal arterial highway, 2 if a major arterial, and 3 otherwise

According to Weisberg "The goal of this analysis was to understand the impact of the design variables, acpts, slim, sigs, and shld that are under the control of the highway department, on accidents. The other variables are thought to be important determinants of accidents but are more or less beyond the control of the highway department and are included to reduce variability due to these uncontrollable factors. We have no particular reason to believe that rate will be a linear function of the predictors, or any theoretical reason to prefer any particular form for the mean function."

EDA

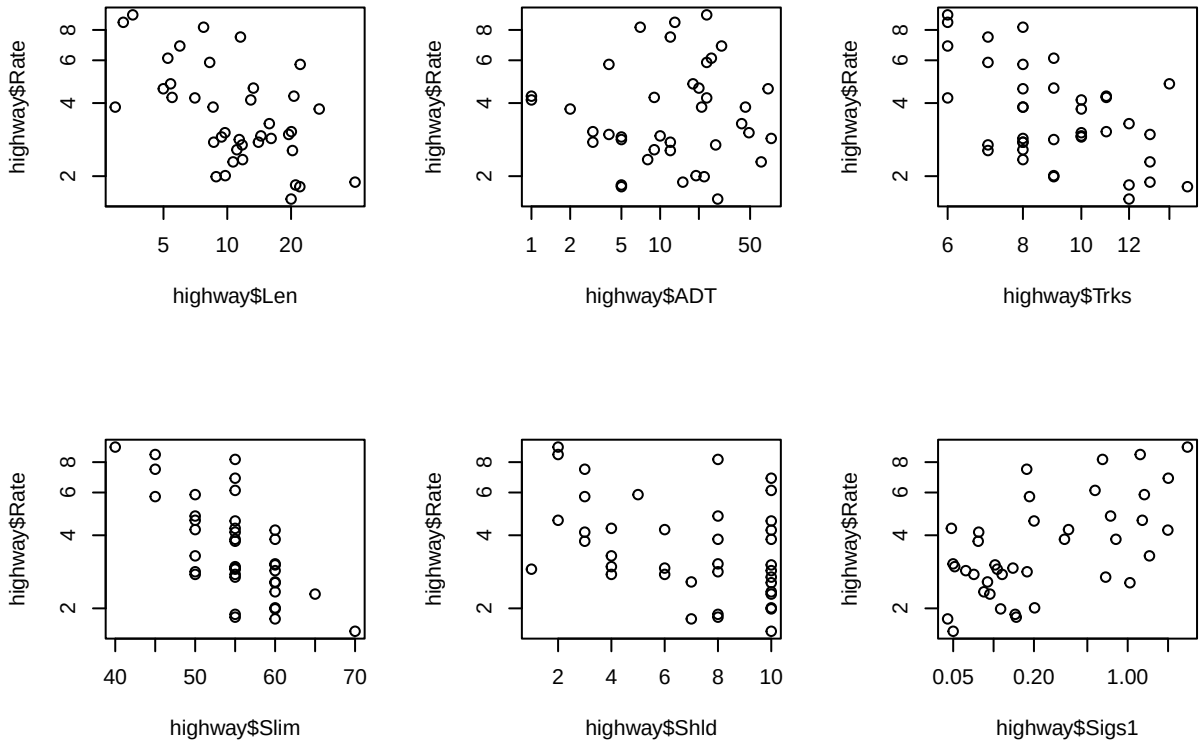
Weisberg suggests many logarithmic transformations. Let's verify that they make empirical sense.

```
# Read in the data
highway <- read.table("highway.txt",header = T)

# Transformations and coding
highway$Sigs1 <- (highway$Sigs*highway$Len + 1)/highway$Len
highway$hiwy <- as.factor(highway$Hwy)

# EDA
par(mfrow=c(2,3))
plot(highway$Len, highway$Rate, log = "xy")
plot(highway$ADT, highway$Rate, log = "xy")
plot(highway$Trks, highway$Rate, log = "xy")
```

```
plot(highway$Slim, highway$Rate, log = "y")
plot(highway$Shld, highway$Rate, log = "y")
plot(highway$Sigs1, highway$Rate, log = "xy")
```



Full model

Let's try out the **full model** including all of the possible predictors.

```
# Full model
m1 <- lm(log2(Rate) ~ log2(Len) + log2(ADT) + log2(Trks) +
                      log2(Sigs1) + Slim + Shld +
                      Lane + Acpt + Itg + Lwid + hiwy, highway)
summary(m1)
```

```
##
## Call:
## lm(formula = log2(Rate) ~ log2(Len) + log2(ADT) + log2(Trks) +
##     log2(Sigs1) + Slim + Shld + Lane + Acpt + Itg + Lwid + hiwy,
##     data = highway)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.64635 -0.14705 -0.00998  0.17645  0.60761
##
```

```
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.704639   2.547137   2.240   0.0342 *
## log2(Len)   -0.214470   0.099986  -2.145   0.0419 *
## log2(ADT)   -0.154625   0.111893  -1.382   0.1792
## log2(Trks)  -0.197560   0.239812  -0.824   0.4178
## log2(Sigs1)  0.192322   0.075367   2.552   0.0172 *
## Slim        -0.039327   0.024236  -1.623   0.1172
## Shld         0.004291   0.049281   0.087   0.9313
## Lane        -0.016061   0.082264  -0.195   0.8468
## Acpt         0.008727   0.011687   0.747   0.4622
## Itg          0.051536   0.350312   0.147   0.8842
## Lwid         0.060769   0.197391   0.308   0.7607
## hiwy1        0.342705   0.576821   0.594   0.5578
## hiwy2       -0.412295   0.393960  -1.047   0.3053
## hiwy3       -0.207358   0.336809  -0.616   0.5437
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3761 on 25 degrees of freedom
## Multiple R-squared:  0.7913, Adjusted R-squared:  0.6828
## F-statistic: 7.293 on 13 and 25 DF,  p-value: 1.247e-05
```

Seems like a huge mess. Not many significant predictors, coefficients are impossible to interpret, etc. Not surprising as we have $d = 13$ potential predictors and only $n = 39$ observations. So we try to pick a subset of the variables.

Forward selection

General idea behind model selection:

- score models according to some criteria,
- search over many different models,
- pick the one with the best (i.e. lowest) score.

Example: using AIC as the score and forward selection as the search, only 4 variables are retained.

```
# Intercept-only model
m0 <- lm(log2(Rate) ~ 1, data=highway)

# Forwrad selection
forward <- step(m0,
                scope=list(lower= ~ 1,
                           upper= ~ log2(Len) + log2(ADT) + log2(Trks) +
                                    log2(Sigs1) + Slim + Shld + Lane + Acpt + Itg +
                                    Lwid + hiwy),
                    direction="forward", data=highway, trace = 0)
forward
```

```
##
```

```
## Call:
## lm(formula = log2(Rate) ~ Slim + log2(Len) + Acpt + log2(Trks),
##     data = highway)
##
## Coefficients:
## (Intercept)         Slim    log2(Len)         Acpt   log2(Trks)
##     6.01105     -0.04595     -0.23573      0.01588     -0.32904
```

Note: rather strange interpretation of the chosen model here...

Of course the question is: what criterion and search should we use? Multiple answers for each. The notes above discuss criteria, and we dive into search strategies.

Forward selection in all its glory

Almost never wise to look at (much less report!) all this output, but...

```
step(m0, scope=list(lower= ~ 1,
             upper=~ log2(Len) + log2(ADT) + log2(Trks) + log2(Sigs1) +
                    Slim + Shld + Lane + Acpt + Itg + Lwid + hiwy),
                    direction="forward", data=highway, trace = 1)
```

```
## Start:  AIC=-30.5
## log2(Rate) ~ 1
##
##                Df Sum of Sq     RSS     AIC
## + Slim          1    8.0771  8.8740 -53.737
## + Acpt          1    7.4345  9.5166 -51.011
## + log2(Sigs1)   1    6.1742 10.7768 -46.160
## + log2(Len)     1    5.5373 11.4138 -43.921
## + log2(Trks)    1    5.0418 11.9092 -42.264
## + Shld          1    2.7536 14.1974 -35.410
## <none>                      16.9510 -30.496
## + hiwy          3    1.8164 15.1346 -28.916
## + Lane          1    0.0138 16.9373 -28.528
## + log2(ADT)     1    0.0131 16.9379 -28.526
## + Itg           1    0.0117 16.9394 -28.523
## + Lwid          1    0.0082 16.9428 -28.515
##
## Step:  AIC=-53.74
## log2(Rate) ~ Slim
##
##                Df Sum of Sq    RSS     AIC
## + log2(Len)     1   2.76182 6.1122 -66.278
## + log2(Trks)    1   2.00977 6.8642 -61.752
## + log2(Sigs1)   1   1.74304 7.1309 -60.266
## + Acpt          1   1.16460 7.7094 -57.224
## <none>                      8.8740 -53.737
## + Lane          1   0.43269 8.4413 -53.687
## + log2(ADT)     1   0.35790 8.5161 -53.343
```

```
## + Itg          1   0.35427 8.5197 -53.326
## + Shld         1   0.16994 8.7040 -52.491
## + Lwid         1   0.13918 8.7348 -52.354
## + hiwy         3   0.36259 8.5114 -49.364
##
## Step:  AIC=-66.28
## log2(Rate) ~ Slim + log2(Len)
##
##               Df Sum of Sq    RSS     AIC
## + Acpt         1   0.60035 5.5118 -68.310
## + log2(Trks)   1   0.54776 5.5644 -67.940
## <none>                     6.1122 -66.278
## + log2(Sigs1)  1   0.30535 5.8068 -66.277
## + hiwy         3   0.70029 5.4119 -65.024
## + Shld         1   0.06796 6.0442 -64.714
## + log2(ADT)    1   0.05335 6.0588 -64.620
## + Lwid         1   0.03464 6.0775 -64.500
## + Lane         1   0.00714 6.1050 -64.324
## + Itg          1   0.00551 6.1067 -64.313
##
## Step:  AIC=-68.31
## log2(Rate) ~ Slim + log2(Len) + Acpt
##
##               Df Sum of Sq    RSS     AIC
## + log2(Trks)   1   0.35995 5.1519 -68.944
## <none>                     5.5118 -68.310
## + log2(Sigs1)  1   0.24989 5.2619 -68.120
## + Shld         1   0.07200 5.4398 -66.823
## + log2(ADT)    1   0.03162 5.4802 -66.534
## + Lane         1   0.03095 5.4809 -66.530
## + Itg          1   0.02810 5.4837 -66.509
## + Lwid         1   0.02632 5.4855 -66.497
## + hiwy         3   0.45265 5.0592 -65.652
##
## Step:  AIC=-68.94
## log2(Rate) ~ Slim + log2(Len) + Acpt + log2(Trks)
##
##               Df Sum of Sq    RSS     AIC
## <none>                     5.1519 -68.944
## + Shld         1   0.13591 5.0159 -67.987
## + log2(Sigs1)  1   0.10525 5.0466 -67.749
## + log2(ADT)    1   0.06498 5.0869 -67.439
## + hiwy         3   0.54012 4.6117 -67.263
## + Lwid         1   0.03957 5.1123 -67.245
## + Itg          1   0.02282 5.1290 -67.117
## + Lane         1   0.00687 5.1450 -66.996
##
```

```
## Call:
## lm(formula = log2(Rate) ~ Slim + log2(Len) + Acpt + log2(Trks),
##     data = highway)
##
## Coefficients:
## (Intercept)         Slim    log2(Len)         Acpt   log2(Trks)
##     6.01105     -0.04595     -0.23573      0.01588     -0.32904
```

Searching for the best model

There are 2^d different models to compare. The brute force approach (try out all the models, called **best subsets**) becomes unworkable when d is even moderately large. A few alternatives:

- **Forward selection**: start from a "small" model (such as the intercept-only model). Add variables one-by-one until no remaining variable improves AIC (say).
- **Backward selection**: such from a big model (such as a model with all the variables in). Delete variables one-by-one until no deletion would improve AIC (say).
- **Stepwise regression**: At each step, either delete one variable or add one variable, whichever improves AIC the most.

Each of these are greedy algorithms and there is no guarantee they will in fact find the model with the smallest AIC. If you really insist on finding the best model, the "leaps and bounds" algorithm saves a little bit of time versus. the brute force approach by "skipping'' certain models which cannot be best.

Backward selection

Look how different the results of backward and forward regression are!

```
back <- step(m1, scope=list(lower= ~ 1,
         upper=~ log2(Len) + log2(ADT) + log2(Trks) + log2(Sigs1) +
                Slim + Shld + Lane + Acpt + Itg + Lwid + hiwy),
                direction="backward", data=highway, trace = 0)
back
```

```
##
## Call:
## lm(formula = log2(Rate) ~ log2(Len) + log2(ADT) + log2(Sigs1) +
##     Slim + hiwy, data = highway)
##
## Coefficients:
## (Intercept)    log2(Len)    log2(ADT)  log2(Sigs1)         Slim        hiwy1
##      6.2768      -0.2616      -0.1269       0.2084      -0.0429       0.1786
##       hiwy2        hiwy3
##     -0.5361      -0.2058
```

```
forward
```

```
##
## Call:
## lm(formula = log2(Rate) ~ Slim + log2(Len) + Acpt + log2(Trks),
```

```
##     data = highway)
##
## Coefficients:
## (Intercept)         Slim    log2(Len)         Acpt   log2(Trks)
##     6.01105     -0.04595     -0.23573      0.01588     -0.32904
```

Stepwise regression

Starting from a small model.

```
step_from_small <- step(m0, scope=list(lower= ~ 1,
             upper=~ log2(Len) + log2(ADT) + log2(Trks) + log2(Sigs1) +
                    Slim + Shld + Lane + Acpt + Itg + Lwid + hiwy),
                    direction="both", data=highway, trace = 0)
step_from_small
```

```
##
## Call:
## lm(formula = log2(Rate) ~ Slim + log2(Len) + Acpt + log2(Trks),
##     data = highway)
##
## Coefficients:
## (Intercept)         Slim    log2(Len)         Acpt   log2(Trks)
##     6.01105     -0.04595     -0.23573      0.01588     -0.32904
```

Starting from a big model.

```
step_from_big <- step(m1, scope=list(lower= ~ log2(Len),
         upper=~ log2(Len) + log2(ADT) + log2(Trks) + log2(Sigs1) +
                Slim + Shld + Lane + Acpt + Itg + Lwid + hiwy),
                direction="both", data=highway, trace = 0)
step_from_big
```

```
##
## Call:
## lm(formula = log2(Rate) ~ log2(Len) + log2(ADT) + log2(Sigs1) +
##     Slim + hiwy, data = highway)
##
## Coefficients:
## (Intercept)    log2(Len)    log2(ADT)  log2(Sigs1)         Slim        hiwy1
##      6.2768      -0.2616      -0.1269       0.2084      -0.0429       0.1786
##       hiwy2        hiwy3
##     -0.5361      -0.2058
```

We see the influence that the starting model plays. When stepwise regression begins at the small model it ends up with the same result as forward selection. When stepwise regression begins at the large model it ends up with the same result as backward selection.

Best subsets

The function `regsubsets` from the library `leaps` runs best subsets and reports the best model among all models with the same number of variables.

```
library(leaps)
msubs <- regsubsets(log2(Rate) ~ log2(Len) + log2(ADT) + log2(Trks) +
                      log2(Sigs1) + Slim + Shld +
                      Lane + Acpt + Itg + Lwid + hiwy, highway, nvmax = 14)
msumm <- summary(msubs)
msumm$which
```

```
##    (Intercept) log2(Len) log2(ADT) log2(Trks) log2(Sigs1) Slim  Shld  Lane
## 1         TRUE     FALSE     FALSE      FALSE       FALSE TRUE FALSE FALSE
## 2         TRUE      TRUE     FALSE      FALSE       FALSE TRUE FALSE FALSE
## 3         TRUE     FALSE     FALSE      FALSE        TRUE TRUE FALSE FALSE
## 4         TRUE      TRUE     FALSE      FALSE        TRUE TRUE FALSE FALSE
## 5         TRUE      TRUE      TRUE      FALSE        TRUE TRUE FALSE FALSE
## 6         TRUE      TRUE      TRUE      FALSE        TRUE TRUE FALSE FALSE
## 7         TRUE      TRUE      TRUE       TRUE        TRUE TRUE FALSE FALSE
## 8         TRUE      TRUE      TRUE       TRUE        TRUE TRUE FALSE FALSE
## 9         TRUE      TRUE      TRUE       TRUE        TRUE TRUE FALSE FALSE
## 10        TRUE      TRUE      TRUE       TRUE        TRUE TRUE FALSE FALSE
## 11        TRUE      TRUE      TRUE       TRUE        TRUE TRUE FALSE  TRUE
## 12        TRUE      TRUE      TRUE       TRUE        TRUE TRUE FALSE  TRUE
## 13        TRUE      TRUE      TRUE       TRUE        TRUE TRUE  TRUE  TRUE
##     Acpt   Itg  Lwid hiwy1 hiwy2 hiwy3
## 1  FALSE FALSE FALSE FALSE FALSE FALSE
## 2  FALSE FALSE FALSE FALSE FALSE FALSE
## 3  FALSE FALSE FALSE FALSE  TRUE FALSE
## 4  FALSE FALSE FALSE FALSE  TRUE FALSE
## 5  FALSE FALSE FALSE FALSE  TRUE FALSE
## 6  FALSE FALSE FALSE FALSE  TRUE  TRUE
## 7  FALSE FALSE FALSE FALSE  TRUE  TRUE
## 8   TRUE FALSE FALSE FALSE  TRUE  TRUE
## 9   TRUE FALSE FALSE  TRUE  TRUE  TRUE
## 10  TRUE FALSE  TRUE  TRUE  TRUE  TRUE
## 11  TRUE FALSE  TRUE  TRUE  TRUE  TRUE
## 12  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## 13  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
```

Best subsets

The best model overall according to Mallow's C_p?

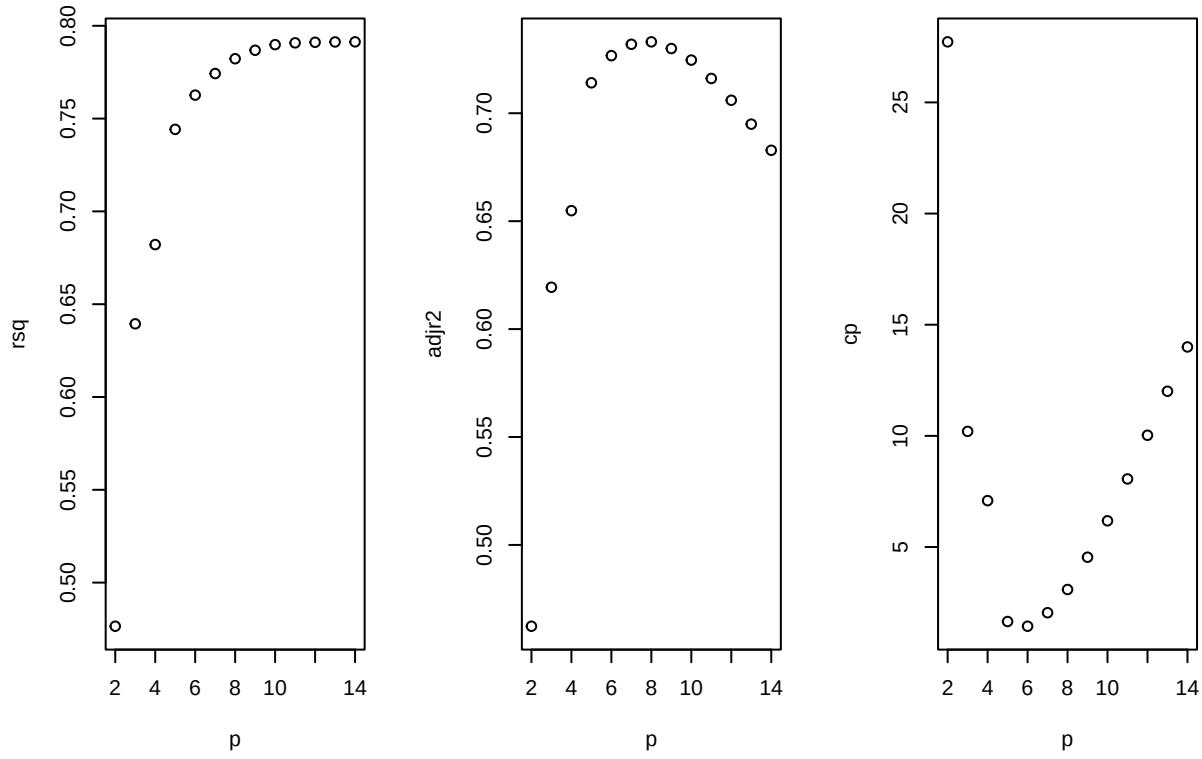
```
best_p = which.min(msumm$cp)
coef(msubs,best_p)
```

```
## (Intercept)   log2(Len)   log2(ADT) log2(Sigs1)        Slim       hiwy2
##   5.6279809  -0.2645638  -0.0756442   0.1864711  -0.0365637  -0.4648045
```

Plots

Model selection methods tend to produce a ton of output. The best way to summarize this output is by plotting the criteria as a function of model size.

```
par(mfrow = c(1,3))
p = 2:14
rsq = msumm$rsq
adjr2 = msumm$adjr2
cp = msumm$cp
plot(p, rsq)
plot(p, adjr2)
plot(p, cp)
```



R^2 and (to a much lesser extent) adjusted R^2 pick too many variables.

Miscellaneous points

- For the highway example, were we really interested in prediction? If so did it make sense to use AIC and C_p?
  - Answers: probably not. What the highway department seemed to really be interested in was the effect of (say) decreasing speed limit on accident rate. This is a causal question and linear regression alone cannot answer it.
- Generally speaking it is better to use stepwise regression than forward or backward selection.

- If p isn't too large then it is computationally possible (and statistically wise) to search over all possible models.
- Unlike `step` the `regsubsets` function has the undesirable property that it can include some (but not all) levels of a factor.
- If the variables in your model were selected according to any of the procedures we've talked about today, you **cannot** use the usual T-tests and confidence intervals.