Variable Selection and Collinearity in Linear Regression

1 The Trouble with Collinearity

Consider the linear regression problem

$$\underline{y}_{n\times 1} = X_{n\times k'}\underline{\beta}_{k'\times 1} + \underline{\epsilon}_{n\times 1}$$

where k' is the total number of regressors (for regression with an intercept, $k' = k+1$), and of course $n > k'$.

One can show that the rank r of the model matrix $X_{n\times k'}$ (number of linearly independent columns of X) is the same as the rank of $X^TX_{k'\times k'}$; if $r < k'$ then X^TX will not be invertible and the usual equations for computing regression estimates and their standard errors will not be usable.

This is the case of *perfect collinearity* and, as we have seen before (especially in the context of ANOVA models), essentially the only fix is to reparametrize the model and/or throw away columns of X, so that the new model matrix X_{new} has the same column space as X, but X_{new} has exactly r columns; for then $(X_{new}^T X_{new})_{r\times r}$ is invertible and the usual regression estimation recipies work.

Partial collinearity, or just *collinearity* for short, occurs when some of the columns of the X matrix are *nearly* linearly dependent. As the columns become more and more nearly linearly dependent, the behavior of the estimated regression coefficients, and especially their standard errors, degrades.

Consider a simple case where we have a y variable and two predictors, x_1 and x_2, with the model:

$$\underline{y} = \underline{x_1}\beta_1 + \underline{x_2}\beta_2 + \epsilon$$

(i.e., no constant term). Let

$$X = \begin{bmatrix} x_{11} & x_{21} \\ \vdots & \vdots \\ x_{1n} & x_{2n} \end{bmatrix}$$

then we know that $\hat{\beta} = (X^TX)^{-1}X^T\underline{y}$ and $\mathrm{Var}\,(\hat{\beta}) = \sigma_\epsilon^2 (X^TX)^{-1}$. Now when X is nearly singular (i.e. near collinearity), then $(X^TX)^{-1}$ get big (like $1/0$) and the variances explode.

Consider the following numerical example. Let

$$\underline{x_1} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} \qquad \text{and} \qquad \underline{x_2} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4+\epsilon \end{bmatrix}$$

then

$$X^TX = \begin{bmatrix} 30 & 30+4\epsilon \\ 30+4\epsilon & 30+8\epsilon+\epsilon^2 \end{bmatrix}$$

and

$$(X^TX)^{-1} = \frac{1}{900+240\epsilon+30\epsilon^2-900-240\epsilon-16\epsilon^2}\cdot\begin{bmatrix} 30+8\epsilon+\epsilon^2 & -30+4\epsilon \\ -30+4\epsilon & 30 \end{bmatrix}$$

so as $\epsilon \longrightarrow 0$,

$$\text{Var}\,(\hat{\beta}_2) \propto \frac{1}{14\epsilon^2} \longrightarrow \infty.$$

Of course the variance estimate for β_1 explodes in just the same manner.

Similar behavior occurs in all cases where collinearity is a problem. This behavior is mostly of concern for the parameter estimates—the fitted values $\hat{y}_i$ are (or should be) quite stable. Numerical problems can (potentially) upset the stability of fitted values unless the software is *very* careful.

More broadly, one can show that, for regression *with* an intercept,

$$\text{Var}\,(\hat{\beta}_j) = [\text{j}^{th} \text{ diag element of } \sigma_\epsilon^2(X^TX)^{-1}] = \frac{1}{1-R_j^2} \cdot \frac{\sigma_\epsilon^2}{(n-1)S_j^2}$$

(where R_j^2 is the squared multiple correlation between X_j and all the other columns of X, and S_j^2 is the sample variance of X_j, the j^{th} column of X), though the algebra can be quite tedious if you are not careful (see Weisberg, 1985, Exercise 8.10, pp. 223–224; or Rawlings et al, 1998, pp. 372–373).

Since R_j^2 measures the degree of collinearity between X_j and the other columns of X, the "variance inflation factor"

$$VIF_j = \frac{1}{1-R_j^2} \geq 1$$

measures the inflation of the variance of $\hat{\beta}_j$ due to collinearity in the columns of X; note that $VIF_j \equiv 1$ iff X_j is orthogonal to the other columns of X (after centering X_j and all the other columns of X by their column means). Sometimes VIF's are a useful diagnostic tool to determine whether collinearity is a problem. More often, though, you don't worry too much about collinearity until you realize that it's getting in the way of some inference that you want to make.

Fixing partial collinearity is like fixing perfect collinearity: you either have to reparameterize the model or throw out some columns of the X matrix. As we have seen with reparametrizing ANOVA models, the effect of this can force a non-trivial change in the interpretation of the model parameters.

Example

The data are from an unpublished Master's Thesis in Civil Engineering by Carl Hoffstedt, as reported by Weisberg (1985, p. 206). They relate the accident report rate, in accidents per million vehicle miles (y) to 13 potential independent variables. The data include 39 sections of large highways in the state of Minnesota in 1973. The variables are:

```
Rate - 1973 accident rate per million vehicle miles
Len  - length of segment in miles
Adt  - average daily traffic count in thousands (estimated)
Trk  - truck volume as a percent of total volume
Sp   - speed limit (in 1973, before the 55 mph limits)
lw   - lane width in feet
shl  - width in feet of outer shoulder on the roadway
itg  - number of freeway-type interchanges per mile in the segment
sigs - number of signalized interchanges per mile in the segment
acpt - number of access points per mile in the segment
lan  - total number of lanes of traffic in both directions
type -  1 if Federal aid interstate highway,
        2 if principal arterial interstate highway,
        3 if minor arterial interstate highway,
        4 otherwise
```

(The full dataset is `highway.dat` on the class website).

We start with a naive regression analysis:

```
> high _ read.table("highway.dat",header=T)
> high[1:5,]
  Rate   Len Adt Trk Sp lw shl  itg sigs acpt lan type
1 4.58  4.99  69   8 55 12  10 1.20    0  4.6   8    1
2 2.86 16.11  73   8 60 12  10 1.43    0  4.4   4    1
3 3.02  9.75  49  10 60 12  10 1.54    0  4.7   4    1
4 2.29 10.65  61  13 65 12  10 0.94    0  3.8   6    1
5 1.61 20.01  28  12 70 12  10 0.65    0  2.2   4    1

> X _ high[,-1]
> y _ high[,1]

> rawlm _ lm(Rate ~ . , data=high)
> summary(rawlm)

Call: lm(formula = Rate ~ ., data = high)
Residuals:
    Min      1Q  Median     3Q   Max
 -1.867 -0.8856 0.09851 0.6906 2.887

Coefficients:
               Value Std. Error  t value Pr(>|t|)
(Intercept)  12.5190   6.5290     1.9175   0.0658
        Len  -0.0564   0.0328    -1.7222   0.0965
        Adt   0.0093   0.0326     0.2841   0.7785
        Trk  -0.1271   0.1057    -1.2026   0.2396
         Sp  -0.0685   0.0637    -1.0756   0.2916
         lw  -0.3433   0.4967    -0.6912   0.4954
        shl  -0.0947   0.1245    -0.7606   0.4535
        itg   0.2868   1.1426     0.2510   0.8037
       sigs   0.4908   0.3941     1.2453   0.2237
       acpt   0.0684   0.0241     2.8376   0.0085
        lan  -0.0131   0.2743    -0.0477   0.9623
       type   0.2338   0.4893     0.4778   0.6366

Residual standard error: 1.172 on 27 degrees of freedom
Multiple R-Squared: 0.7527
F-statistic: 7.472 on 11 and 27 degrees of freedom, the p-value is
    1.051e-05

[ ... ]
```

```
> source("collin.q") # This function is available in the data area of
                     # the course website.  It calculates R^2 for each
                     # X regressed on all the other X's, providing
                     # essentially the same information as VIF=1/(1-R^2).
                     # (Thanks to Howard Seltman.)

> collin(X,cutoff=0.7)  # R^2 for regressing each X on the other X's;
                        # report only those R^2's above 0.7.
Adt 0.902
Sp 0.739
shl 0.747
itg 0.836
lan 0.741
type 0.743
```

Clearly there is a lot of collinearity!

2 Variable Selection and Specification Error

A more pleasant-sounding name for "throwing out some of the columns of X" is *variable selection* or *model selection*, so automatic methods of variable selection almost always get discussed in the same breath as collinearity problems. However it is important to note that variable selection is really a separate issue from fixing collinearity. As illustrated above, *collinearity is primarily a technical problem*. On the other hand, *variable selection is often at least partially a scientific problem*, related to the process of sifting through independent variables to find ones that are causally related to the dependent variable in a particular scientific context.

Variable selection and model selection must deal with two fundamental obstacles, whether in a classical context like least-squares regression, or in a Bayesian or other context:

- *There are a lot of models to consider.* For a regression with k' predictors, there are $2^{k'} - 1$ additive models to consider, and many many more if we allow interactions between two or more predictors at a time. For example, if you have $k' = 8$ predictors, there are 255 models to consider, and if you have $k' = 50$ regressor variables, as with some of your end-of-semester projects, there are $2^{50} - 1 = 1.1259 \times 10^{15}$ models to consider. Even with fast computers, we aren't going to find the best model by exhaustive search.

- *There are not enough models to consider.* Even if you were able to consider all $2^{50} - 1 = 1.1259 \times 10^{15}$ models in a model with 50 known regressors, you know or suspect now that the "right" variables to use to predict differences in yield and so forth are probably not even in the data set.

The first problem is primarily a computational one, and automatic variable selection methods are primarily heuristic methods to reduce the number of models considered, but still generate some reasonably good models. Alternatively, and often more effectively, you can select one or more good models by hand if you have some subject area knowledge to guide you.

The second problem is not a theoretical or computational problem; it is fundamentally a scientific problem. Statistical theory can shed light on the consequences of this problem but not on how to solve it.

From our earlier work this semester (especially Exercise 7.3, pp. 233-234 in Rawlings et al., 1998), we know:

- *Overspecifying the model,* i.e. including X's that aren't necessary, can *increase our uncertainty (higher SE's)* in parameter estimates and predictions, but will not affect the unbiasedness of estimates.
- *Underspecifying the model,* i.e. failing to include X's that are needed and are collinear with X's already in the model, can *increase bias* (including having completely the wrong sign!) and *reduce standard errors* for estimates of coefficients of X's already in the model.

Variable selection tries to achieve a specification that is "not too hot" and "not too cold" in the above senses.

We will consider two or three statistically-motivated approaches to variable selection:

- *Automatic, "stepwise" procedures* that "score" each model and select the model with the highest (or lowest) score.
- *Principal component transformations* of the X matrix that produces a new regressor matrix W. Variable selection is "easier" with W because the columns of W will be uncorrelated; and in many cases
- *Other methods* such as minimizing prediction error and so-called "biased regression" methods.

Note that *all* of these methods finess the scientific question: we assume that the largest model we consider has enough variables in it so that there is no bias in estimating parameters due to omitting relevant regressors from the model. If X is missing some important regressors, then the final model you get from these methods will be underspecified, and your $\hat{\beta}_j$ estimates will be biased, possibly severely (albeit of lower variance).

3 Automatic Variable Selection

In this section we will look at two or three automatic methods for variable selection. These are essentially all heuristic methods for searching through the space of possible models in a regression problem.

Each heuristic can be combined with several different ways of "scoring" the models, so we begin by considering some different ways of scoring models. For each criterion that follows, p is the number of regression parameters (regressors *plus intercept*) in the (current) model. The maximum value for p is $k' = k + 1$, the total number of regressors (plus intercept) available in the X matrix.

- $\mathbf{R^2 = 1 - RSS/TSS}$
  R^2 can be (and is) used to compare models with the *same number* of parameters p. Thus for the highway data we could compare all models which include `Len` and `acpt` and (say) one

extra variable, using the appropriate R^2 values. The model with the largest value of R^2 would be considered as (a candidate for) the "best". However, since R^2 must grow as regressors are added to the model, R^2 is not a good way to compare models with different numbers of regressors.

- **Extra-sum-of-squares F-test**
  The F-test can be used to compare a model and a subset of that model. For example the model (x_1, x_3, x_5) and the model $(x_1, x_3, x_5, x_7, x_8)$ can be compared using an F-test.
  - Because the log-likelihood for normal-errors linear regression is proportional to the RSS for the model, the extra-sum-of-squares F-test

    $$F = \frac{(RSS_{small} - RSS_{big})/(p_{big} - p_{small})}{RSS_{big}/(n - p_{big})}$$

    (in our current notation p_{big} and p_{small} count the intercept too) makes the same decisions (and asymptotically with the same level) as the likelihood ratio test for comparing nested models

    $$\begin{aligned} \lambda &= -2\left\{\log(L_{big}(\underline{y}; \hat{\underline{\beta}}_{big}, \hat{\sigma}^2_{big})) - \log(L_{small}(\underline{y}; \hat{\underline{\beta}}_{small}, \hat{\sigma}^2_{small}))\right\} \\ &= n\log\frac{\hat{\sigma}^2_{small}}{\hat{\sigma}^2_{big}} + \frac{(\underline{y} - X_{small}\hat{\underline{\beta}}_{small})^T(\underline{y} - X_{small}\hat{\underline{\beta}}_{small})}{\hat{\sigma}^2_{small}} - \frac{(\underline{y} - X_{big}\hat{\underline{\beta}}_{big})^T(\underline{y} - X_{big}\hat{\underline{\beta}}_{big})}{\hat{\sigma}^2_{big}} \\ &= n\log\frac{RSS_{small}}{RSS_{big}} + \frac{(\underline{y} - X_{small}\hat{\underline{\beta}}_{small})^T(\underline{y} - X_{small}\hat{\underline{\beta}}_{small})}{RSS_{small}/n} - \frac{(\underline{y} - X_{big}\hat{\underline{\beta}}_{big})^T(\underline{y} - X_{big}\hat{\underline{\beta}}_{big})}{RSS_{big}/n} \\ &= n\log\frac{RSS_{small}}{RSS_{big}} + n - n \;=\; n\log\left(F \cdot \frac{p_{big} - p_{small}}{n - p_{big}} + 1\right) \end{aligned}$$

  - Since the classical distribution theory for the likelihood ratio test and F-test of non-nested models is unwieldy, these tests are not directly useful for comparing non-nested models.
  - Comparing two models with $R^2 = 1 - RSS/TSS$ is equivalent to looking at the difference $RSS_{small} - RSS_{big}$ (or the ratio of RSS's), i.e. it is like doing the F or likelihood ratio test without the benefit of distribution theory to help decide when the difference in RSS's is big enough to care about. All of the other methods below replace distribution theory with a numerical penalization for more parameters, so that model comparisons for which classical distribution theory isn't available (especially non-nested models) may be performed.

- **Adjusted $\mathbf{R^2}$**

  $$R^2_{adj,p} = 1 - \left(\frac{n-1}{n-p}\right)\left(1 - R^2\right)$$

  where p is the number of regression parameters (including intercept) included and R^2 is for the model with p parameters. This is an adjustment which is designed to compensate for the fact that otherwise as $p \nearrow$ then $R^2 \nearrow$. Note that there are some problems with this in that $R^2_{adj,p}$ can be negative, but this is generally not a problem when the model fits reasonably well. The idea is to choose the model with the largest value of $R^2_{adj,p}$.

- **Mallows' $\mathbf{C_p}$**

$$C_p = \frac{RSS_p}{\hat{\sigma}^2_\epsilon} + 2p - n \tag{1}$$

where RSS_p is computed from the "current" model with p parameters (including intercept), but $\hat{\sigma}^2_\epsilon$ is estimated from the full model or the largest model which will be considered (and which is consequently a superset of all the other models). Clearly this is related to the F-statistic, but it is adjusted. Since $E[RSS_p] = (n-p)\sigma^2$ under the "correct" model, it follows that $E[C_p] \approx p$; and in fact $E[C_p] \approx p$ for every model containing the "correct" model as well. The idea then is to select models with C_p near p as candidates for further investigation. A useful method for selecting such models is to plot C_p against p.

The full model always has $C_{k+1} = k+1$, but notice some of the simpler models may also have reasonable C_p's. We usually try to either minimize C_p, or choose the smallest p for which $C_p \approx p$.

- **Akaike's Information Criterion (AIC)**
This is one of the most popular "penalized likelihood" methods for scoring goodness-of-fit.

$$\begin{aligned} AIC &= -2\log(L_p(\underline{y}; \underline{\hat{\beta}}_p, \hat{\sigma}^2_p)) + 2p \\ &= n\log\hat{\sigma}^2_p + \frac{(\underline{y} - X_p\underline{\hat{\beta}}_p)^T(\underline{y} - X_p\underline{\hat{\beta}}_p)}{\hat{\sigma}^2_p} - 2p \\ &= n\log RSS_p - n\log(n) + n + 2p \end{aligned}$$

We want to choose the model with the smallest AIC (sometimes -AIC is reported, so that big values are for good models).

Note that for comparing different models on the same data set (so additive terms involving n only cancel out). Also, for comparing models with the same n, AIC and $Cp = RSS_p/\hat{\sigma}^2_\epsilon + 2p - n$ are clearly monotone functions of each other: smallest AIC and smallest C_p choose the same model.

For nested models

$$AIC_{small} - AIC_{big} = n\log\frac{RSS_{small}}{RSS_{big}} - 2(p_{big} - p_{small})$$

so that this is effectively a likelihood ratio comparison, penalized for extra parameters in the model. It is known that AIC tends to choose models with more parameters (regressors) than the true model; but it is very good at minimizing prediction error.

- **Schwarz's Bayesian Information Criterion (BIC, a.k.a. SBC)**
This is the other popular "penalized likelihood" methods for scoring goodness-of-fit.

$$\begin{aligned} BIC &= -2\log(L_p(\underline{y}; \underline{\hat{\beta}}_p, \hat{\sigma}^2_p)) + p\log(n) \\ &= n\log RSS_p - n\log(n) + n + p\log(n) \end{aligned}$$

Again the goal is to choose the model with the smallest BIC (sometimes -BIC is reported, so that big values are for good models), and again the terms involving only n cancel out in model comparisons. Note that the penalty for extra parameters is now multiplied by $\log(n)$ rather than by 2 as in AIC, therefore BIC will tend to pick smaller models than AIC.

The penalized likelihood ratio statistic is now

$$BIC_{small} - BIC_{big} = n \log \frac{RSS_{small}}{RSS_{big}} - (p_{big} - p_{small}) \log(n)$$

Perhaps more importantly, $BIC_{small} - BIC_{big}$ can be shown to be a reasonable asymptotic approximation to the *Bayes factor* comparing the two models, as n grows, and therefore it can be shown to select the "true" model with probability tending to one as n grows (Schwarz, 1978, *Ann. Stat.*; Kass and Raftery, 1995, *JASA*).

We now have some tools for comparing models, but they seem to require that we fit all (or most) of the models. This would not be much use, unless there was a simple way of fitting models, or deciding which models to think carefully about. In the next several subsections we describe and illustrate several such methods using the highway accident data.

These automatic methods do not absolve you (the analyst) from looking at the data and thinking about the substantive problem that the data came from. You should still

1. Examine the data before selecting model, to assure there are no glaring errors
2. Carefully examine the best modesl for influential or outlying observations
3. Make sure you can interpret and justify the best model (and explain the results to others) in terms of the substantive/scientific inquiry.

Continuation of Example

Figure 1 gives an example of a C_p plot, for the highway data. The solid line is $C_p = p$.

The code for generating the plot follows. It exploits the "leaps()" function in Splus (and R), which exhaustively[1] fits all models and reports the n best (lowest C_p) models with $p = 1$ predictor, $p = 2$ predictors, $p = 3$ predictors, etc.

```
> result _ leaps(X,y,nbest=2) # generates Cp for the 2 best (lowest Cp)
                              # models with 1 predictor, 2 predictors, etc.
> names(result)
[1] "Cp"    "size"  "label" "which" "int"
> plot(result$size,result$Cp)
> abline(0,1)
> title("Cp Plot for the Highway Data")
```

In this case, models of size $p = 4$ or 5 are suggested; a quick look at `help(leaps)` tells us that this means 3 or 4 regressors, plus the intercept. The suggested models are

```
> data.frame(terms=result$label,Cp=result$Cp,size=result$size)[5:8,]
            terms       Cp size
5      Len,Sp,acpt 2.457994    4
6      Trk,Sp,acpt 3.106290    4
7 Len,Sp,sigs,acpt 1.563455    5
8  Len,Trk,Sp,acpt 1.930997    5
```

[1] `leaps()` does not actually have to fit all models, since it can sometimes guess that if a particular model fits badly, all submodels of that model will fit even worse.

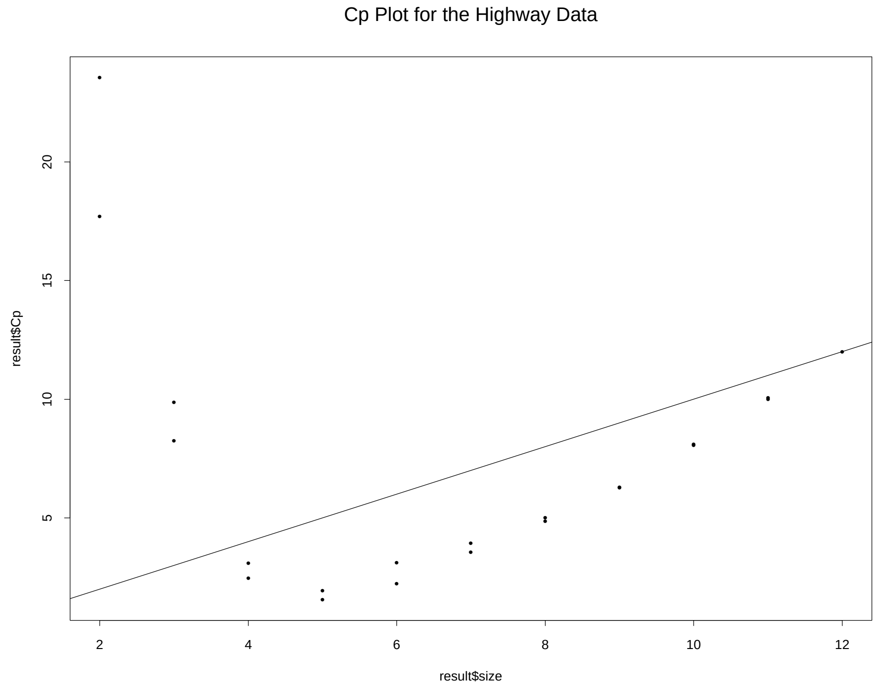


Figure 1: Plot of C_p vs p. Best model has least C_p close to p.

Note that $C_p/n \approx \hat{\sigma}_p^2 + 2p/n$, so one way to think about C_p is that it is a way of minimizing residual standard error, with a penalty for including "too many" parameters in the model. It is also related to a penalized likelihood method, AIC, as we shall see next.

Another function (available only in Splus) called `stepwise()`, can be used to generate the n best (in terms of minimizing RSS, which is equivalent to maximizing R^2) models with $p = 1$ predictor, $p = 2$ predictors, $p = 3$ predictors, etc.[2] The function `sum.step.q` available on the course website (thanks to Howard Seltman!), can summarize the output of `stepwise` in terms of AIC, BIC, etc., to choose the best model.

```
> sw _ stepwise(X,y,meth="exhaustive")
> sum.step(sw,high,"Rate",includeAIC=T,includeR2=T)
          BIC   AIC    R2   R2a Len Adt Trk Sp lw shl itg sigs acpt lan type
 3(#1) -66.0 -62.6 0.694 0.667   1   0   0  1  0   0   0    0    1   0    0
 4(#1) -66.1 -61.9 0.720 0.687   1   0   0  1  0   0   0    1    1   0    0
 4(#2) -66.3 -62.2 0.717 0.683   1   0   1  1  0   0   0    0    1   0    0
 3(#2) -66.3 -63.0 0.688 0.661   0   0   1  1  0   0   0    0    1   0    0
 4(#3) -67.0 -62.8 0.707 0.673   0   0   1  1  0   0   0    1    1   0    0
 5(#1) -67.1 -62.1 0.732 0.692   1   0   1  1  0   0   0    1    1   0    0
 3(#3) -67.3 -64.0 0.671 0.643   0   0   0  1  0   0   0    1    1   0    0
 5(#3) -67.7 -62.8 0.724 0.682   1   0   0  1  0   0   0    1    1   0    1
 5(#2) -67.7 -62.7 0.724 0.682   1   0   0  1  0   1   0    1    1   0    0
 2(#1) -68.2 -65.7 0.622 0.601   0   0   0  1  0   0   0    0    1   0    0
 6(#1) -68.6 -62.8 0.739 0.689   1   0   1  1  0   1   0    1    1   0    0
 6(#2) -68.8 -63.0 0.736 0.686   1   0   0  1  0   0   1    1    1   0    1
```

[2] Like `leaps()`, `stepwise()` does not actually have to fit all models, since it can sometimes guess that if a particular model fits badly, all submodels of that model will fit even worse.

```
 6(#3) -68.9 -63.0 0.735 0.685   1   0   1  1  1   0   0    1    1   0    0
 2(#2) -68.9 -66.4 0.607 0.586   0   0   1  0  0   0   0    0    1   0    0
 2(#3) -69.3 -66.8 0.600 0.578   0   0   0  0  0   0   0    1    1   0    0
 7(#1) -70.0 -63.4 0.746 0.688   1   0   1  1  0   0   1    1    1   0    1
 7(#2) -70.1 -63.4 0.745 0.687   1   0   1  1  1   1   0    1    1   0    0
 7(#3) -70.2 -63.5 0.744 0.686   1   1   1  1  0   0   0    1    1   0    1
 1(#1) -71.1 -69.4 0.517 0.504   0   0   0  0  0   0   0    0    1   0    0
 8(#1) -71.6 -64.2 0.750 0.684   1   0   1  1  1   1   1    1    1   0    0
 8(#2) -71.7 -64.2 0.750 0.683   1   1   1  1  1   1   0    1    1   0    0
 8(#3) -71.8 -64.4 0.748 0.680   1   0   1  1  0   1   1    1    1   0    1
 1(#2) -73.1 -71.5 0.464 0.449   0   0   0  1  0   0   0    0    0   0    0
 9(#2) -73.5 -65.2 0.752 0.675   1   0   1  1  1   1   1    1    1   0    1
 9(#1) -73.5 -65.2 0.752 0.675   1   1   1  1  1   1   0    1    1   0    1
 9(#3) -73.6 -65.3 0.751 0.673   1   1   1  1  1   1   1    1    1   0    0
10(#3) -75.5 -66.4 0.752 0.663   1   0   1  1  1   1   1    1    1   1    1
10(#2) -75.5 -66.4 0.752 0.664   1   1   1  1  1   1   0    1    1   1    1
10(#1) -75.5 -66.3 0.753 0.664   1   1   1  1  1   1   1    1    1   0    1
11(#1) -77.5 -67.5 0.753 0.652   1   1   1  1  1   1   1    1    1   1    1
 1(#3) -77.8 -76.1 0.319 0.300   0   0   0  0  0   0   0    1    0   0    0
 0(#3) -83.4 -82.6 0.000 0.000   0   0   0  0  0   0   0    0    0   0    0
 1(#2) -83.7 -82.0 0.077 0.052   0   0   0  0  0   0   0    0    0   0    1
```

Note that `sum.step()` reports $-AIC/2$ and $-BIC/2$ (and omits terms like $n\log(n)$ that depend only on n) so large values (values near zero) are good; the output is sorted in order from best BIC (nearest zero) to worst BIC (farthest from zero).

The variables in the best several models here are:

```
Len  - length of segment in miles
Trk  - truck volume as a percent of total volume
Sp   - speed limit (in 1973, before the 55 mph limits)
sigs - number of signalized interchanges per mile in the segment
acpt - number of access points per mile in the segment
```

Generally speaking a change of 2 to 6 is considered moderate evidence of improved fit in BIC (this translates to a 1 to 3 difference in what `sum.step` reports), so there isn't much difference among the first five or so models listed here.

3.1 Stepwise Selection Procedures

The exhaustive procedures used to illustrate C_p, AIC, BIC, R^2 and R^2_{adj} above are not useful when there are many (more than 30 or so) regressors to consider in the problem. In this case we need one or another heuristic algorithm to search the model space for a few good candidate models to explore further.

The standard search algorithms are variations on "greedy" strategies, that is, they search among models closely related to the "current" model that maximize the "score" (whether F-test, C_p, or whatever), and jump to that model. In statistics, these methods are all called "stepwise" methods.

There are (at least) three basic strategies for stepwise selection of models:

- Start with a model an keep adding until you can't add anymore—called Forward Selection

- Start with a (big) model and take away variables while you can—called Backward Elimination
- Do both at the same time—a true stepwise technique

Even within these there are several possible ways of choosing a model. Note that all of these methods are very heuristic; there is no guarantee that the best model—even in terms of the specific criterion used—will be found!

Forward Selection.

Start with the intercept-only straw-man model; and add one variable at a time.

1. Stop after a given number of steps or (equivalently) when a subset of a certain size has been reached; **or**
2. Use F-tests to decide which variable to add next, and stop when no variable has a big F-test; **or**
3. Stop when adding the next variable would result in collinearity $\Longleftrightarrow$ Stop when R^2_{new} (squared multiple correlation for regressing X_{new} on the other X's already in the model) gets too big.

Backward Elimination.

Start with all the variables in the model; and remove one variable at a time.

1. Stop after a given number of steps; **or**
2. Use F-tests to decide which variable to remove next, and stop when removing each variable produces a big F-test.

True Stepwise.

One needs to provide criteria for including and deleting terms from the model. This can be done by specifying the values of the F-statistic, F-IN and F-OUT, required for entry into or removal from the the model. Then the stepwise procedure repeatedly applies the following rules, until no further changes in the model occur.

1. Remove a variable is F-test is less than F-OUT
2. Remove if $<$ F-OUT and R^2 is bigger than the previous model with this number of predictors
3. Swap two variables (in and out) if this improves R^2
4. Add a variable if F-test is greater than F-IN

A common choice is F-IN = F-OUT = 4. Making F-IN large and F-OUT small means that the current model is hard to get into and easy to get out of. By making F-OUT = 0 one obtains a forward selection method. F-OUT should generally be (the same or) smaller than F-IN to stop the method from getting stuck in a loop where the same variables are repeatedly added in and taken out. Since F-IN and F-OUT are fixed numbers, the "add" and "remove" decisions are not F-tests with fixed levels.

The stepwise procedure is probably preferable to just backwards elimination or forward selection, as it allows one to enter and delete variables. Due to correlations among the X variables, it is quite possible that the variable entered at the first step (i.e., the BEST single variable) may be removed at some later step.

Stepwise regression with `step()`

Splus has a function `step` that will perform stepwise regression, though it is based on minimizing AIC rather than using entry and removal criteria with F statistics. When adding variables, `step` chooses the variable that reduces AIC the most. Similarly, when removing variables, `step` removes the variable that reduces AIC the most. It continues in adding and removing cycles until any further move increases AIC.

`step()` generates a lot of output, so I will abbreviate it here.

```
> step(rawlm,scope=list(upper= ~ ., lower ~ 1))

Start: AIC= 70.0038
 Rate ~ Len + Adt + Trk + Sp + lw + shl + itg + sigs + acpt + lan + type

Step:  AIC= 67.2617
 Rate ~ Len + Adt + Trk + Sp + lw + shl + itg + sigs + acpt + type

Step:  AIC= 64.6104
 Rate ~ Len + Adt + Trk + Sp + lw + shl + sigs + acpt + type

Step:  AIC= 62.1651
 Rate ~ Len + Adt + Trk + Sp + lw + shl + sigs + acpt

Step:  AIC= 60.2145
 Rate ~ Len + Trk + Sp + lw + shl + sigs + acpt

Step:  AIC= 58.4081
 Rate ~ Len + Trk + Sp + shl + sigs + acpt

Step:  AIC= 56.5758
 Rate ~ Len + Trk + Sp + sigs + acpt

Step:  AIC= 55.6784
 Rate ~ Len + Sp + sigs + acpt
```

In this case, it appears that we could never reduce AIC by adding variables after some were already taken out. [Note that AIC as reported here is scaled yet another way.]

The `drop1()` and `add1()` functions can be used to perform individual steps of the `step()` procedure, to see what happens "by hand". For example,

```
> final  _ lm(formula = Rate ~ Len + Sp + sigs + acpt, data = high)
> as.formula(paste("~",paste(names(high),collapse="+"))
 ~ Rate + Len + Adt + Trk + Sp + lw + shl + itg + sigs + acpt + lan + type
> add1(final,scope=as.formula(paste("~",paste(names(high),collapse="+"))))
Single term additions

Model:
```

```
Rate ~ Len + Sp + sigs + acpt
       Df Sum of Sq      RSS       Cp
<none>              41.95214 54.29100
   Adt  1  0.135480 41.81666 56.62330
   Trk  1  1.847865 40.10427 54.91091
    lw  1  0.319123 41.63302 56.43965
   shl  1  0.622958 41.32918 56.13582
   itg  1  0.351065 41.60107 56.40771
   lan  1  0.025793 41.92635 56.73298
  type  1  0.564620 41.38752 56.19416
```

shows that there are no variables that can be added to the final model to improve (lower) AIC (which is listed in the “Cp” column (!?) in the output).

One nice feature of `step()` is that it can also be used with logistic regression and other `glm()` objects. The Venables and Ripley MASS library (`library(MASS)`) contains a function `stepAIC()` which is supposed to be an improvement on `step()`, especially for glm's where computation of AIC is not as straightforward as in least-squares regression. See `help(stepAIC)` for details.

Stepwise regression with `stepwise()`

The `stepwise()` function that we used previously also has options for forward, backwards, and stepwise regression. The stepwise method is Efroymson's, which adds variables when the F-statistic is greater than F-IN (default: 2), and the criterion for removing a variable is just heavy collinearity. See `help(stepwise)` for details.

```
> swe _ stepwise(X,y,method="efroymson")
> sum.step(swe,high,"Rate",includeAIC=T,includeR2=T,sortcol=F)
        BIC   AIC    R2   R2a Len Adt Trk Sp lw shl itg sigs acpt lan type
1(+9) -71.1 -69.4 0.517 0.504   0   0   0  0  0   0   0    0    1   0    0
2(+4) -68.2 -65.7 0.622 0.601   0   0   0  1  0   0   0    0    1   0    0
3(+1) -66.0 -62.6 0.694 0.667   1   0   0  1  0   0   0    0    1   0    0
4(+8) -66.1 -61.9 0.720 0.687   1   0   0  1  0   0   0    1    1   0    0
```

Here we add first `acpt`, then `Sp`, then `Len`, and finally `sigs`; after this no more variables could be added according to the F-IN criterion, and at no stage could any variables be dropped based on collinearity. Note that the final model (by AIC) is the same one achieved by `step` above; the best of these by BIC omits `sigs`.

Model summaries

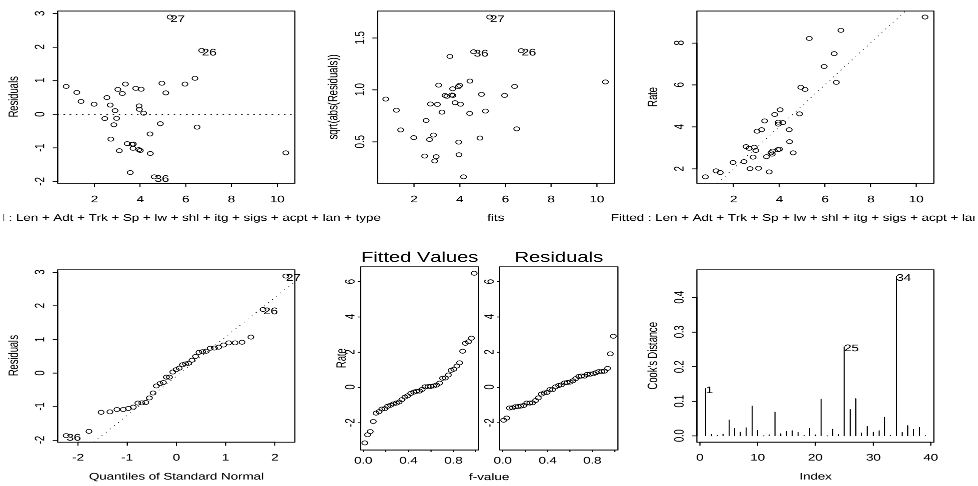
Let's take a look at the parameter estimates and diagnostic plots of the original model including all variables, as well as the two models that were identified by our stepwise procedures.

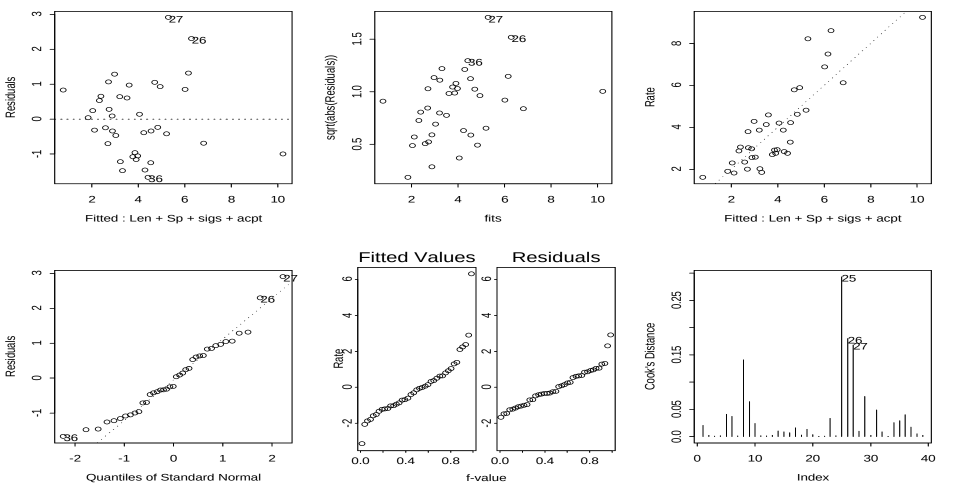
```
> summary(rawlm _ lm(Rate ~ ., data =high))$coefficients
                  Value Std. Error     t value    Pr(>|t|)
(Intercept)  12.519017130 6.52897506  1.91745519 0.065816656
        Len  -0.056444936 0.03277522 -1.72218346 0.096474882
        Adt   0.009261974 0.03260618  0.28405580 0.778534668
        Trk  -0.127078208 0.10567014 -1.20259335 0.239574239
         Sp  -0.068470234 0.06365773 -1.07559977 0.291624232
         lw  -0.343297057 0.49669188 -0.69116704 0.495360764
        shl  -0.094711453 0.12451762 -0.76062688 0.453471734
        itg   0.286810535 1.14261457  0.25101250 0.803704939
       sigs   0.490779227 0.39409530  1.24533133 0.223709754
       acpt   0.068408278 0.02410784  2.83759426 0.008521754
        lan  -0.013096558 0.27430020 -0.04774535 0.962270618
       type   0.233826847 0.48934626  0.47783516 0.636613497
> plot(rawlm)
```

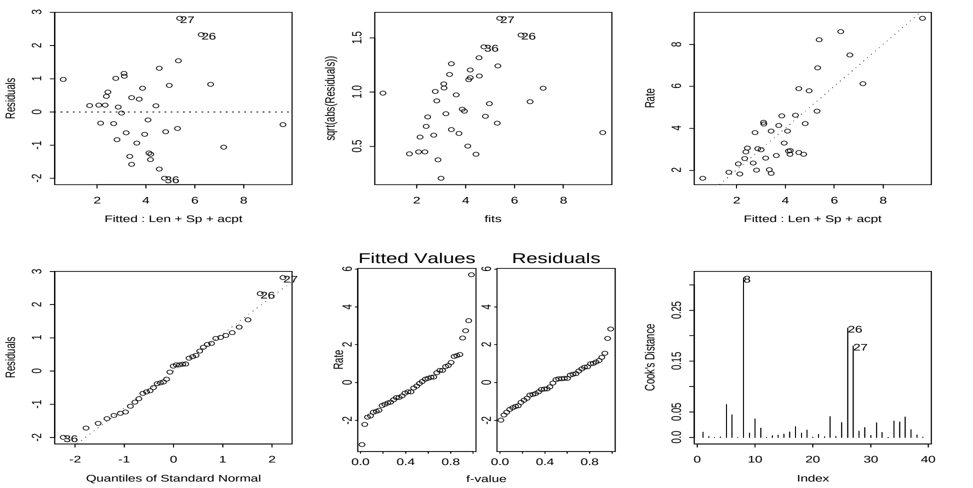
```
> summary(mod1 _ lm(Rate ~ Len + Sp +sigs + acpt, data =high))$coefficients
                  Value Std. Error   t value     Pr(>|t|)
(Intercept)  9.97563578 2.35441161  4.236997 0.0001633434
        Len -0.06188818 0.02535342 -2.441019 0.0200061400
         Sp -0.11573228 0.03881110 -2.981937 0.0052653982
       sigs  0.59434719 0.33121662  1.794436 0.0816397324
       acpt  0.06824634 0.02109316  3.235472 0.0027058274
> plot(mod1)
```

```
> summary(mod2 _ lm(Rate ~ Len + Sp + acpt, data =high))$coefficients
                  Value Std. Error   t value      Pr(>|t|)
(Intercept)  11.10415391 2.33971260  4.745948 0.00003447674
        Len  -0.07254915 0.02541716 -2.854338 0.00719988687
         Sp  -0.13122745 0.03901986 -3.363094 0.00187820573
       acpt   0.07591309 0.02130094  3.563837 0.00107925360
> plot(mod2)
```

Our task next time is to come up with a reasonable explanation for what these variables mean.


Residuals
l : Len + Adt + Trk + Sp + lw + shl + itg + sigs + acpt + lan + type
sqrt(abs(Residuals))
fits
Rate
Fitted : Len + Adt + Trk + Sp + lw + shl + itg + sigs + acpt + lai
Quantiles of Standard Normal
Fitted Values
Residuals
f-value
Cook's Distance
Index


Residuals
Fitted : Len + Sp + sigs + acpt
sqrt(abs(Residuals))
fits
Rate
Fitted : Len + Sp + sigs + acpt
Quantiles of Standard Normal
Fitted Values
Residuals
f-value
Cook's Distance
Index


Residuals
Fitted : Len + Sp + acpt
sqrt(abs(Residuals))
fits
Rate
Fitted : Len + Sp + acpt
Quantiles of Standard Normal
Fitted Values
Residuals
f-value
Cook's Distance
Index

4 Cross-validation and Prediction Error Sum of Squares (PRESS)

We have seen earlier in the course (e.g. in the first course project) that one way to evaluate a model is to hold out part of the data, do model selection and fitting with what's left, and then calculate how well the best model we can find does in predicting the hold-out data (e.g. calculate the "prediction" RSS $\sum_{i \in \text{hold out data}} (y_i - \hat{y}_i)^2$, where the $\hat{y}_i$'s are the predictions of the hold out data from the model fitted on the remainder. We can even do this several times with several different hold-out sets, to get a feel for (a) how much the regression coefficients change; and (b) how much the prediction RSS changes (both are kind of *bootstrapping* ideas).

The most extreme version of this would be to hold one one observation at a time, and predict it from the model fitted to the rest of the data (this is an idea related to *jackknifing*).

By using the magic formula

$$(X_{(-i)}^T X_{(-i)})^{-1} = (X^T X)^{-1} + \frac{(X^T X)^{-1} \underline{x}_i \underline{x}_i^T (X^T X)^{-1}}{1 - h_{ii}}$$

from our study of unusual or influential observations, we can calculate

$$E_{(-i)} = y_i - \underline{x}_i^T \hat{\underline{\beta}}_{(-i)} = \frac{E_i}{1 - h_{ii}}$$

where E_i and h_{ii} are the residuals and diagonal entries in the hat matrix of the full regression, and $\hat{\underline{\beta}}_{(-i)}$ os calculated from the model without observation i, $(y_i, \underline{x}_i^T)$. This suggests a criterion like

$$PRESS = \sum_{i=1}^{n} E_{(-i)}^2 = \sum_{i=1}^{n} \left[\frac{E_i}{1 - h_{ii}} \right]^2$$

PRESS has the aesthetic advantage of measuring how well y_i is predicted from all the data *except* the i^{th} observation, for all i.

Clearly, different models can be compared based on the size of their PRESS, so that PRESS becomes another way to "score" models in model selection. Often C_p and AIC choose models that minimze PRESS (since they are basically minimizing penalized RSS, which is like PRESS, to begin with).

Outside of ordinary linear regression, where the "magic formula" above no longer holds, PRESS can be expensive to calculate. Therefore, one often finds a few candidate models using a cheaper criterion like C_p, AIC, or BIC, and then PRESS is used for final model selection (if that is the goal).

5 Other Ways of Coping with Collinearity: Biased Regression Methods

Recall that the principal problem with collinearity in the regression problem

$$\underline{y} = X\underline{\beta} + \underline{\epsilon} \tag{2}$$

is the technical difficulty that $(X^TX)^{-1}$ becomes difficult to compute. If we still wish to work with this regression problem and not some other one, then we have to modify our estimation procedure somehow to obtain, for example, more stable estimates $\underline{\hat{\beta}}$.

5.1 Principal Components

The least-squares estimate of $\underline{\beta}$ is

$$\underline{\hat{\beta}} = (X^TX)^{-1}X^T\underline{y}$$

but if X has nearly collinear columns this is difficult to compute. To make the problem easier to think about let us imagine that X has a column of 1's, to fit an intercept, and let us write

$$X = [\ \underline{1}\ \ \underline{x}_1\ \ \cdots\ \ \underline{x}_k\ ]$$

where $\underline{1}$ is the intercept column and the columns $\underline{x}_1, \ldots, \underline{x}_k$ are the independent variables of interest. We define

$$Z = [\ \underline{z}_1\ \ \cdots\ \ \underline{z}_k\ ] = [\ (\underline{x}_1 - \overline{x}_1)/s_1\ \ \cdots\ \ (\underline{x}_k - \overline{x}_k)/s_k\ ]$$

where $\overline{x}_j$ is the j^{th} column mean and s_j^2 is the sample variance of the j^{th} column. It is fairly easy to show that

$$\underline{y} = \underline{1}\beta_0 + Z\underline{\alpha} + \underline{\epsilon} \tag{3}$$

is the same regression problem as Equation (2), and

$$\begin{aligned} \underline{\hat{\alpha}} &= (Z^TZ)Z^T\underline{y} \\ \text{Var}\,(\underline{\hat{\alpha}}) &= \sigma_\epsilon^2(Z^TZ)^{-1} \end{aligned}$$

There will be collinearity in X if and only if there is collinearity in Z, and Z has the pleasant property that

$$R_{XX} = \frac{1}{n-1} \cdot Z^TZ$$

is the matrix of sample correlations of the $\underline{x}_j$'s (with 1's down the main diagonal).

For any $\underline{u}$, $\underline{u}^TZ^TZ\underline{u} = \|Z\underline{u}\|^2 \geq 0$ so Z^TZ is symmetric and positive-definite; we can then write the eigenvalue-eigenvector decomposition

$$Z^TZ = VLV^T$$

where $dim(V) = dim(Z) = n \times k$ and the columns of V are *orthonormal*, i.e. $V^TV = I$, and

$$L = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_k \end{bmatrix}$$

Because the columns of V are orthonormal, it follows that

$$Z^T ZV = VLV^T V = VL$$

so that the j^{th} column of V is the j^{th} *eigenvector* of Z, with associated *eigenvalue* λ_j; we can arrange that $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p$.

The *principal components* of Z are the columns of the matrix

$$W = ZV$$

Note that Equation (3) can be written

$$\begin{aligned} \underline{y} &= \underline{1}\beta_0 + Z\underline{\alpha} + \underline{\epsilon} \\ &= \underline{1}\beta_0 + (ZV)(V^T\underline{\alpha}) + \underline{\epsilon} \\ &= \underline{1}\beta_0 + W\underline{\gamma} + \underline{\epsilon} \end{aligned} \tag{4}$$

and this is equivalent to our original Equation (2).

Finally, note that

$$\text{Var}\,(\hat{\underline{\gamma}}) = \sigma_\epsilon^2 (W^T W)^{-1} = \sigma_\epsilon^2 (V^T Z^T Z V)^{-1} = \sigma_\epsilon^2 L^{-1}/(n-1)$$

is a diagonal matrix, and $\text{Var}\,(\hat{\gamma}_k) = \sigma^2 \lambda_k^{-1}/(n-1)$, so that (1) the $\hat{\gamma}$'s are uncorrelated; and (2) the data tells us the least about the γ_k for the smallest eigenvalue λ_k.

This suggests that we can "improve" the model by dropping columns of W corresponding to high $\text{Var}\,(\hat{\gamma}_k)$ i.e. low λ_j. If the original model in Equation (2) was the correct model for the data, then the estimates $\hat{\underline{\beta}}$, $\hat{\underline{\alpha}}$ and $\hat{\underline{\gamma}}$ in these problems should all be unbiased.

To see what the effect of dropping columns of W is, consider minimizing the sum of mean square errors (SMSE) in the coefficients, instead of in the predicted values,

$$\begin{aligned} SMSE &= \sum_{j=1}^{k} E[(\hat{\gamma}_j - \gamma_j)^2] \\ &= \sum_{j=1}^{k} \{\text{Var}\,(\hat{\gamma}_j) + [bias(\hat{\gamma}_j)]^2\} \\ &= \frac{\sigma_\epsilon^2}{n-1} \sum_{j=1}^{k} \lambda_j^{-1} + \sum_{j=1}^{k} \{E[\hat{\gamma}_i - \gamma_j]\}^2 \end{aligned} \tag{5}$$

where $\hat{\gamma}_j$ are the least-squares estimates. When the full set of least-squares estimates are used, $\hat{\gamma}_j$ will be unbiased and the "bias" term on the right will be zero, so

$$SMSE_{OLS} = \frac{\sigma_\epsilon^2}{n-1} \sum_{j=1}^{k} \lambda_j^{-1}$$

Because the columns of W are orthogonal, when we drop columns we do not change the least-squares estimates of the remaining coefficients. Dropping the last m columns for example we obtain

$$SMSE_{PC(k-m)} = \frac{\sigma_\epsilon^2}{n-1} \sum_{j=1}^{k-m} \lambda_j^{-1} + \sum_{j=k-m+1}^{k} \gamma_j$$

This is a classic case of the *bias/variance tradeoff* in statistics: by dropping only a few terms we can reduce SMSE by paying for a large drop in variance with a small increase in bias; clearly we can minimize $SMSE_{PC(k-m)}$ over m if desired.

5.2 Ridge Regression

Ridge regression is another way to play the bias-variance tradeoff game. The idea is that $Z^T Z$ may have a numerically unstable inverse, but

$$(Z^T Z + kI)^{-1}$$

will be quite stable, for $k > 0$. The ridge regression estimator for $\underline{\beta}$ is then

$$\underline{\hat{\alpha}}^{(k)} = (Z^T Z + kI)^{-1} Z^T \underline{y} \qquad (*)$$

Again,

$$SMSE_{RR(k)} = \sum_{j=1}^{k} \{\text{Var}\,(\hat{\alpha}_j^{(k)}) + [bias(\hat{\alpha}_j^{(k)})]^2\}$$

and when $k = 0$ this reduces to the least-squares expression, with no bias term. As k grows (not necc. integer values!), the variance term decreases since it is determined largely by $(Z^T Z + kI)^{-1}$; and of course the bias term is no longer zero. We can if we wish estimate the optimal k by directly minimizing $SMSE_{RR(k)}$, or by employing a criterion like PRESS, or Mallows' C_p (where $p =$ trace$\,(H_{RR(k)})$, the trace of the ridge regression hat matrix); see the bullets below for a bit more information.

If we work with the coefficients $\underline{\gamma}$ in the principal components regression (4), we can show that

$$\hat{\gamma}_j^{(k)} = \frac{\lambda_j}{\lambda_j + k} \hat{\gamma}_j$$

so we see that as k grows the estimates of the γ_j's are being "shrunk to zero". (This corresponds to a certain Bayesian formulation of linear regression which we may be able to look at briefly later in the semester).

Notes:

- In the formulation of ridge regression above we worked with the normalized matrix of predictors Z. In fact there is no particular reason to use this normalized version, and we might prefer to use
  - The matrix X^* of regressors variables after subtracting the column means; or
  - The original, un-normalized matrix X of regressors.

  For example if the X matrix has 0/1 dummy variables to code for levels of some discrete factor, the normalized version is more difficult to interpret and we'd better stick to the un-normalized X. In that case the ridge regression estimates are just as in $(*)$, but using X in place of Z.

- How do we choose the value of the ridge constant k?
  - One possibility is to plot the parameter estimates $\hat{\beta}_j$ as a function of k: in general $|\hat{\beta}_j|$ decreases with k. Then choose the largest value of k that doesn't affect the values of the $\hat{\beta}_j$'s very much. Such plots are called *ridge trace plots.*
  - Another possibility is to plot a measure of overall "goodness" of the model, such as PRESS, as a function of k. Then choose the value of k that minimizes PRESS in the plot. This is an example of choosing k *by cross-validation.*