Part 17: Multiple Linear Regression

Text references: §3.2 in ISL, Chapters 12 and 13 in Ruppert.

In this part, we will extend from the simple linear regression model to **multiple regression**.

These models comprise the familiar linear models that take the form

$$Y = \beta_0 + \sum_{j=1}^{p} \beta_j x_j + \epsilon$$

where ϵ are assumed to be uncorrelated with mean zero and variance σ^2. Assuming that the ϵ are normally distributed brings additional nice properties.

We will go through the fundamental steps of fitting such a model via an example.

Example: Factor Models

References to the "beta" for a stock derive from the slope of the regression line fit to a scatter plot where excess return for that equity is the response, and excess market return is the predictor. Models of this form are an important component of Capital Asset Pricing Model (CAPM).

The term **Factor Model** is used to indicate any linear model for excess return on an equity. The predictors in the model are the **factors**.

Quoting Ruppert, examples of factors include

1. returns on the market portfolio;
2. growth rate of the GDP;
3. interest rate on short term Treasury bills or changes in this rate;
4. inflation rate or changes in this rate;
5. interest rate spreads;
6. return on some portfolio of stocks;
7. the difference between the returns on two portfolios.

The CAPM only includes the factor for excess market return.

The **Fama-French Three Factor Model** adds two factors.

The first is **small minus big (SMB)**, the difference in returns between portfolios of small and large stocks.

The second is **high minus low (HML)**, the difference in returns between portfolios of high and low book-to-market stocks.

See Ruppert for more details on these factors, and motivation on their inclusion in the model.

Kenneth French maintains a web site with data on the three factors

http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html

I wrote a simple R function that will allow for easy access to the daily factor data.

This function, named `getFamaFrench()`, is located at

http://www.stat.cmu.edu/~cschafer/MSCF/getFamaFrench.txt

The syntax is much like the functions in `quantmod`:

```
> ffhold = getFamaFrench(from="2016-1-1",
       to="2016-6-30")
```

(Note that data is typically unavailable for the most recent months.)

The first few rows of the data returned are as follows:[1]

```
            Date Mkt.RF   SMB   HML    RF
23638 2016-01-04  -1.59 -0.83  0.53 0.000
23639 2016-01-05   0.12 -0.21  0.00 0.000
23640 2016-01-06  -1.35 -0.13  0.01 0.000
23641 2016-01-07  -2.44 -0.28  0.12 0.000
23642 2016-01-08  -1.11 -0.47 -0.04 0.000
23643 2016-01-11  -0.06 -0.65  0.35 0.000
...
```

[1] These factors are periodically revised, and hence the exact numbers may be different at the present time. This will affect the subsequent analysis, as well. The analysis shown here was performed on January 8, 2017.

Let's get the data for PNC for the same time period

```
> PNC = getSymbols("PNC", from="2016-1-1",
      to="2016-6-30", auto.assign=F)
```

Now find the excess returns for PNC. Note that the Fama-French data file includes the risk free rate, with name `RF`:

```
> ffhold$PNCexret = 100*dailyReturn(PNC) - ffhold$RF
```

The three factor model we fit is

$$\text{PNC.Ex.Ret.} = \beta_0 + \beta_1 \text{ Mkt.Ex.Ret.} + \beta_2 \text{ SMB} + \beta_3 \text{ HML} + \epsilon$$

The R command for fitting a linear regression is `lm()`:

```
> ff3modPNC = lm(PNCexret ~ Mkt.RF + SMB + HML,
      data=ffhold)
```

Note the use of the `data` argument to specify the data frame from which the variables are taken.

The key information from the fit of the model is stored in the object `ff3modPNC`. In particular, note that

```
> attributes(ff3modPNC)
$names
 [1] "coefficients"  "residuals"     "effects"       "rank"
 [5] "fitted.values" "assign"        "qr"            "df.residual"
 [9] "xlevels"       "call"          "terms"         "model"
```

For example, you can obtain the residuals from `as.numeric(ff3modPNC$residuals)`.[2]

[2] It is necessary to transform `ff3modPNC$residuals` using `as.numeric()` because it inherits the time series format from the original data.

Diagnostic Plots

Figure 1 shows three key diagnostic plots:

1. Plot of residuals versus fitted values.
2. Normal probability plot.
3. Plot of residuals versus time.

Below are the R commands to create this figure.

```
par(mfrow=c(2,2))

# Plot of residuals versus fitted values
plot(as.numeric(ff3modPNC$fit), as.numeric(ff3modPNC$resid),
   pch=16, xlab="Fitted Values", ylab="Residuals",
   cex.axis=1.3, cex.lab=1.3)

# Normal probability plot
qqnorm(as.numeric(ff3modPNC$resid), cex.axis=1.3, cex.lab=1.3,
   pch=16, main="")
qqline(as.numeric(ff3modPNC$resid))

# Plot of residuals versus time
plot(ff3modPNC$resid, xlab="Time", ylab="Residuals", cex.axis=1.3,
   cex.lab=1.3, pch=16, main="")
```

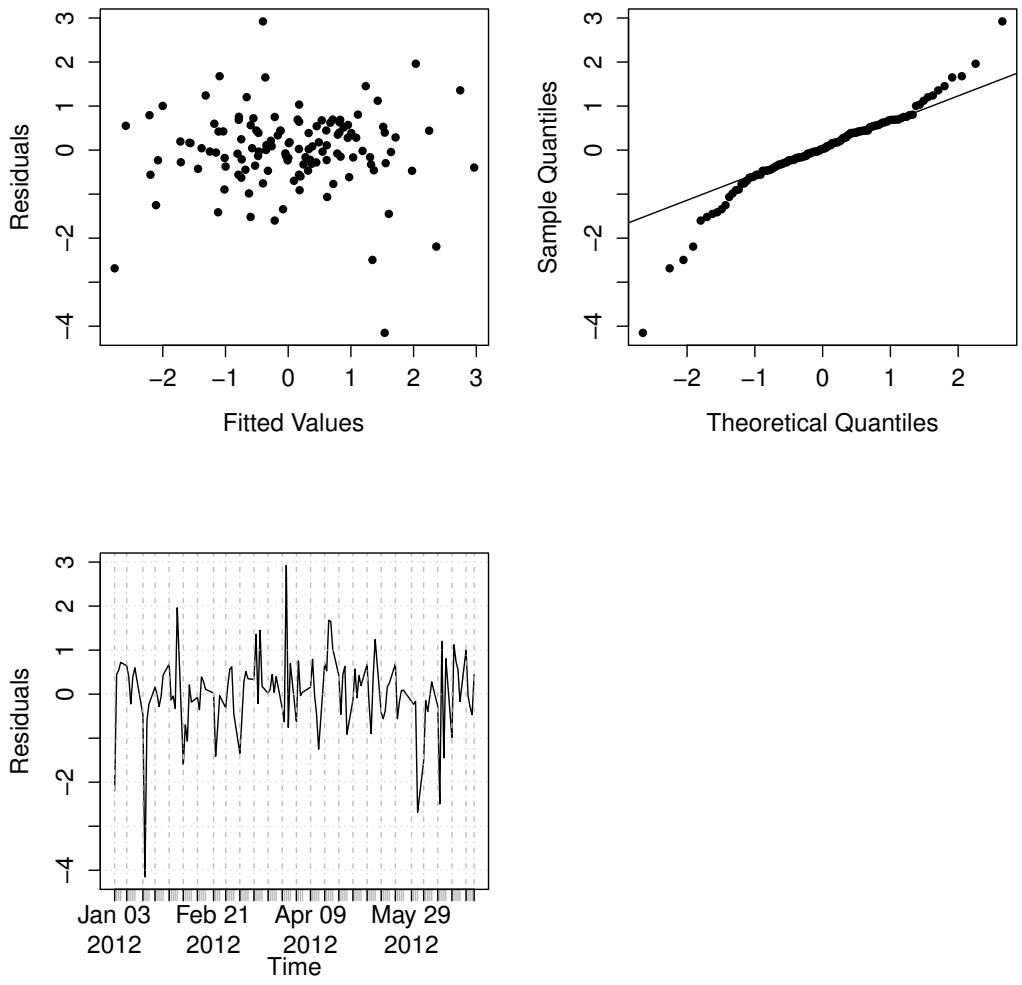

Figure 1: Diagnostic plots for the three factor model for PNC.

Exercise: Comment on the quality of the fit.

Use the `summary()` function to see the parameter estimates and their standard errors:[3]

```
Call:
lm(formula = PNCexret ~ Mkt.RF + SMB + HML, data = ffhold)

Residuals:
     Min       1Q   Median       3Q      Max
-2.35913 -0.51226  0.09069  0.41264  3.08356

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.16153    0.06945  -2.326   0.0217 *
Mkt.RF       1.20089    0.06952  17.275  < 2e-16 ***
SMB         -0.03428    0.13890  -0.247   0.8055
HML          0.78117    0.11937   6.544 1.52e-09 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 0.7755 on 121 degrees of freedom
Multiple R-squared:  0.7699,Adjusted R-squared:  0.7642
F-statistic:   135 on 3 and 121 DF,  p-value: < 2.2e-16
```

[3] Again, a reminder that your results could be slightly different since the Fama-French factors are periodically updated.

Exercise: There is strong evidence that β_3, the coefficient for HML, is larger than zero. How did I reach this conclusion, and what is the interpretation?

Linear Regression in Matrix Notation

First, recall our simple linear regression model:

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i, \quad i = 1, 2, \ldots, n$$

We can rewrite this model in matrix notation by letting

$$\mathbf{Y} = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} 1 & X_1 \\ 1 & X_2 \\ \vdots & \vdots \\ 1 & X_n \end{bmatrix} \quad \boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} \quad \boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

and then noting that

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

When moving to multiple regression, the main change we must make is to add columns to the matrix $\mathbf{X}$, commonly called the **design matrix**, and add entries to $\boldsymbol{\beta}$:

$$\mathbf{X} = \begin{bmatrix} 1 & X_{11} & X_{12} & \cdots & X_{1,p} \\ 1 & X_{21} & X_{22} & \cdots & X_{2,p} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & X_{n1} & X_{n2} & \cdots & X_{n,p} \end{bmatrix} \quad \boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix}$$

One can show that the least squares estimators (now a vector) are

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}.$$

and can now define the vector of fitted values and residuals as

$$\widehat{\mathbf{Y}} = \begin{bmatrix} \widehat{Y}_1 \\ \widehat{Y}_2 \\ \vdots \\ \widehat{Y}_n \end{bmatrix} \quad \text{and} \quad \hat{\boldsymbol{\epsilon}} = \begin{bmatrix} \hat{\epsilon}_1 \\ \hat{\epsilon}_2 \\ \vdots \\ \hat{\epsilon}_n \end{bmatrix}$$

and it follows that

$$\widehat{\mathbf{Y}} = \mathbf{X}\hat{\boldsymbol{\beta}} \quad \text{and} \quad \hat{\boldsymbol{\epsilon}} = \mathbf{Y} - \widehat{\mathbf{Y}}.$$

The Hat Matrix

We note that we could write

$$\widehat{\mathbf{Y}} = \mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} = \mathbf{H}\mathbf{Y}$$

where

$$\mathbf{H} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$$

is called the **hat matrix** (because it puts a "hat" on $\mathbf{Y}$).

The diagonal entries of $\mathbf{H}$ are referred to as the **leverages**.

Comment: Mathematically, $\mathbf{H}$ is a **projection matrix**, as it "projects" the vector of observed responses $\mathbf{Y}$ onto the space of all vectors which are linear combinations of the columns of $\mathbf{X}$. In fact, we note that $\mathbf{H}$ is symmetric and **idempotent**, meaning that $\mathbf{HH} = \mathbf{H}$.

Exercise: Show that $\mathbf{I} - \mathbf{H}$ is also symmetric and idempotent.

Additional Results

A range of important results can be proven:

1. The variance of $\hat{\boldsymbol{\beta}}$ is $\sigma^2(\mathbf{X}^T\mathbf{X})^{-1}$.
2. $\widehat{\mathbf{Y}} = \mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{H}\mathbf{Y}$, where the (symmetric, idempotent) hat matrix $\mathbf{H}$ is defined as above.
3. $\hat{\boldsymbol{\epsilon}} = \mathbf{Y} - \widehat{\mathbf{Y}} = (\mathbf{I} - \mathbf{H})\mathbf{Y}$.
4. If $\boldsymbol{\epsilon}$ is assumed normal with mean zero and variance $\sigma^2\,\mathbf{I}$,
   (a) $\hat{\boldsymbol{\beta}}$ is the maximum likelihood estimator.
   (b) $\mathbf{Y}$ is multivariate normal with mean $\mathbf{X}\boldsymbol{\beta}$ and covariance $\sigma^2\,\mathbf{I}$.
   (c) $\hat{\boldsymbol{\beta}}$ is multivariate normal with mean $\boldsymbol{\beta}$ and covariance $\sigma^2(\mathbf{X}^T\mathbf{X})^{-1}$.
   (d) $\widehat{\mathbf{Y}}$ is multivariate normal with mean $\mathbf{X}\boldsymbol{\beta}$ and covariance $\sigma^2\,\mathbf{H}$.
   (e) $\hat{\boldsymbol{\epsilon}}$ is multivariate normal with mean zero and covar. $\sigma^2(\mathbf{I}-\mathbf{H})$.

Degrees of Freedom

One key thing that does change when moving to multiple regression is the number of **degrees of freedom** in the residual vector.

Recall that we had seen previously that with simple linear regression, it held that

$$\sum_i \widehat{\epsilon}_i = 0 \quad \text{and} \quad \sum_i \widehat{\epsilon}_i X_i = 0.$$

This result is one way of seeing why the residuals only have $n - 2$ degrees of freedom in the case of simple linear regression: There are two constraints placed on the vector of residuals.

But, these facts could be written more compactly using matrix notation: $\mathbf{X}^T \widehat{\boldsymbol{\epsilon}} = \mathbf{0}$, where $\mathbf{0}$ is a vector consisting entirely of zeros.

In fact, this result extends to multiple regression: The vector of residuals $\widehat{\boldsymbol{\epsilon}}$ is orthogonal to **every column** of $\mathbf{X}$, i.e., $\mathbf{X}^T \widehat{\boldsymbol{\epsilon}} = \mathbf{0}$.

We now can state that there are $n-(p+1)$ degrees of freedom in the residuals, and hence need to update some of our previous results. (You will note that simple linear regression corresponds to the case where $p = 1$.) In particular:

1. The unbiased estimator of σ^2 is

$$\hat{\sigma}^2 = \mathrm{RSS} \Big/ (n - p - 1)$$

2. The statistic

$$\frac{\hat{\beta}_i - \beta_i}{\widehat{\mathrm{SE}}(\hat{\beta}_i)}$$

has the t-distribution with $n - p - 1$ degrees of freedom. Hence, the following hold:

   (a) A $100(1-\alpha)\%$ confidence interval for β_i is formed as

$$\hat{\beta}_i \pm t_{\alpha/2, n-p-1} \, \widehat{\mathrm{SE}}(\hat{\beta}_i)$$

   (b) Hypothesis tests concerning β_i should compare the test statistic (the "t value") with the t-distribution with $n-p-1$ degrees of freedom.

Basic Variable Selection

The basic, general strategy for avoiding overfitting with models is to compare the values of AIC across different number/choices of covariates

Recall that we prefer the model which has the smallest value of AIC

Unfortunately, when the number of covariates is large, the number of possible models is enormous. (There are 2^p possible models if there are p covariates.)

We'll utilize the regression problem introduced earlier in which we sought to estimate the forward-rate function based on the current prices of zero-coupon bonds. The data are shown again as Figure 2.

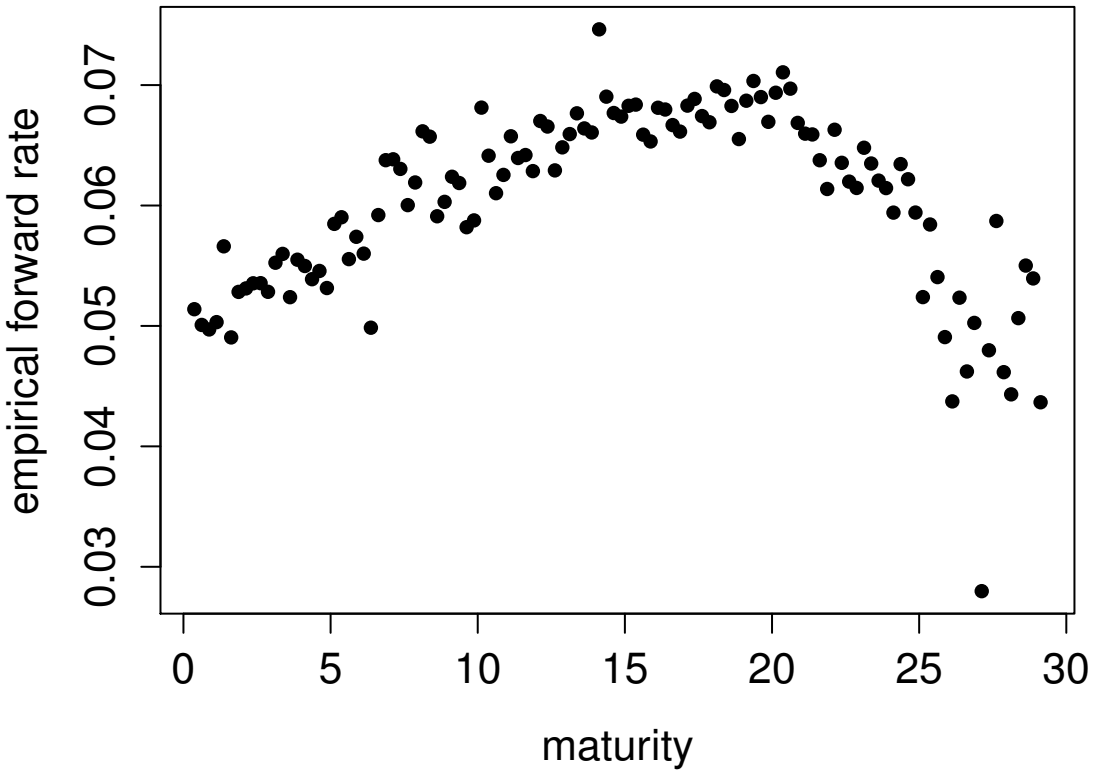

Figure 2: The response versus time for the estimation of the forward rate function.

You can read in this data set using the command

```
> forratedat =
    read.table("http://www.stat.cmu.edu/~cschafer/MSCF/forratedat.txt",
    header=T)

> attach(forratedat)
```

We are considering polynomial models of the form

$$Y_i = \sum_{j=0}^{p} \beta_j T_i^j + \epsilon_i$$

but our goal is to find the best subset of the p polynomial terms to include in the model. (As is usually the case, we do not consider removing the intercept β_0.)

The "full" model (largest possible model) we'll consider includes up to the ninth power.

We will scale the predictor first, so that it has mean zero and standard deviation one:

```
> maturity = scale(maturity)
```

Now, we can fit the full model:

```
> fullmod = lm(emp ~ maturity + I(maturity^2) +
    I(maturity^3) + I(maturity^4) + I(maturity^5) +
    I(maturity^6) + I(maturity^7) + I(maturity^8) +
    I(maturity^9))
```

Note the use of `I()` function to wrap the polynomial terms is required by R.

Exhaustive Search

In cases where the number of predictors is small, an exhaustive search over all possible models is possible.

One implementation of this in R is the function `bestglm()` which is part of the package `bestglm`.

Unfortunately, the syntax for this function requires that the predictors and response be in a single data frame, with the response as the last column.

Consider the following commands

```
> allpreds = cbind(maturity, maturity^2,
      maturity^3, maturity^4, maturity^5,
      maturity^6, maturity^7, maturity^8,
      maturity^9)
> Xyframe = data.frame(cbind(allpreds, emp))
```

Note that now `Xyframe` holds the predictors and the response.

```
> bestmod = bestglm(Xyframe, IC="AIC")
```

By specifying `IC = "AIC"` we are telling the function to use AIC to choose the model.

The R command

```
> print(bestmod)
```

will show the output below

```
AIC
BICq equivalent for q in (0.0443666836953486, 0.908095644854053)
Best Model:
                Estimate   Std. Error    t value      Pr(>|t|)
(Intercept)  0.066820498 0.0004770183 140.079524 8.381181e-126
V1           0.004950250 0.0008628244   5.737262  8.591912e-08
V4          -0.008479843 0.0007636180 -11.104824  1.134645e-19
V5          -0.004005042 0.0008106249  -4.940684  2.801999e-06
V6           0.002275548 0.0002881148   7.898061  2.313625e-12
V7           0.001183985 0.0002729549   4.337659  3.206729e-05
```

We see that the optimal model takes the form

$$Y_i = \beta_0 + \beta_1 T_i + \beta_4 T_i^4 + \beta_5 T_i^5 + \beta_6 T_i^6 + \beta_7 T_i^7 + \epsilon_i$$

Stepwise Regression

An exhaustive search is not feasible when there is a large number of predictors.

A classic strategy for dealing with the large number of possible models is to use a **stepwise variable selection** procedure.

With this method, one (typically) starts with the largest model under consideration, and then takes a sequence of **steps**: At each step one covariate is either **added** or **dropped**.

The decision for which step to take (i.e., which covariate to add or drop) is based on AIC: You take the step which reduces AIC the most.

Once AIC no longer changes, the process stops, and you have your final model.

This algorithm is far from perfect, but is useful in the appropriate situations.

This is built into R using the function `step()`

Now we will use the `step()` function on the example above.

The syntax is simple:

```
> finalmod = step(fullmod, direction="both")
```

Recall that `fullmod` holds the output of the full model fit.

The use of `direction = "both"` tells R to consider not only removing predictors, but also adding predictors to the model. (Once a predictor is removed, it could be returned to the model in a later step.)

The output of the first step of algorithm appears below.

```
Start:  AIC=-1296.97
emp ~ maturity + I(maturity^2) + I(maturity^3) + I(maturity^4) +
    I(maturity^5) + I(maturity^6) + I(maturity^7) + I(maturity^8) +
    I(maturity^9)

                 Df  Sum of Sq       RSS     AIC
- I(maturity^6)  1 3.0370e-07 0.0013612 -1298.9
- I(maturity^9)  1 3.3910e-07 0.0013613 -1298.9
- I(maturity^8)  1 9.2100e-07 0.0013619 -1298.9
- I(maturity^3)  1 9.7950e-07 0.0013619 -1298.9
- I(maturity^7)  1 1.7992e-06 0.0013627 -1298.8
- I(maturity^2)  1 2.1071e-06 0.0013630 -1298.8
- I(maturity^5)  1 3.4433e-06 0.0013644 -1298.7
- I(maturity^4)  1 4.7036e-06 0.0013656 -1298.6
- maturity       1 2.2945e-05 0.0013839 -1297.0
<none>                        0.0013609 -1297.0
```

Note how the table lists each of the covariates currently in the model, and gives what AIC would be if that covariate were removed.

We also see the line labelled `<none>`, indicating that the AIC would stay at -1297.0 if none were removed.

The covariates are ordered by increasing values of AIC. (Unfortunatley, due to rounding it's not possible to see some distinctions in this output.)

So, the term `I(maturity^6)` is removed at the first step.

The second step is shown as:

```
Step:  AIC=-1298.94
emp ~ maturity + I(maturity^2) + I(maturity^3) + I(maturity^4) +
    I(maturity^5) + I(maturity^7) + I(maturity^8) + I(maturity^9)

                 Df  Sum of Sq       RSS     AIC
- I(maturity^9)   1 3.3900e-07 0.0013616 -1300.9
- I(maturity^3)   1 9.8000e-07 0.0013622 -1300.9
- I(maturity^7)   1 1.7990e-06 0.0013630 -1300.8
- I(maturity^5)   1 3.4430e-06 0.0013647 -1300.7
- I(maturity^2)   1 1.9184e-05 0.0013804 -1299.3
- maturity        1 2.2945e-05 0.0013842 -1299.0
<none>                         0.0013612 -1298.9
+ I(maturity^6)   1 3.0400e-07 0.0013609 -1297.0
- I(maturity^4)   1 8.0262e-05 0.0014415 -1294.3
- I(maturity^8)   1 1.2814e-04 0.0014894 -1290.5
```

The decision of this step is to remove `I(maturity^9)`. Note that also under consideration was to add back in the term that was removed.

This process continues until a final model is reached:

```
Step:  AIC=-1303.2
emp ~ maturity + I(maturity^4) + I(maturity^5) + I(maturity^7) +
    I(maturity^8)

                 Df  Sum of Sq       RSS     AIC
<none>                          0.0013818 -1303.2
+ I(maturity^2)   1 0.00001918 0.0013626 -1302.8
+ I(maturity^6)   1 0.00001738 0.0013644 -1302.7
+ I(maturity^3)   1 0.00000105 0.0013808 -1301.3
+ I(maturity^9)   1 0.00000041 0.0013814 -1301.2
- I(maturity^7)   1 0.00023342 0.0016152 -1287.1
- I(maturity^5)   1 0.00030284 0.0016847 -1282.2
- maturity        1 0.00040836 0.0017902 -1275.2
- I(maturity^8)   1 0.00075672 0.0021385 -1254.5
- I(maturity^4)   1 0.00212230 0.0035041 -1197.3
```

So, we have arrived (after only four steps) at a model that includes powers 1, 4, 5, 7, and 8.

Exercise: How does this model compare to the one we stated as being optimal using the exhaustive search? What happened?

Cross Validation

We've discussed the use of AIC for the purpose of **model selection**, i.e. the process of determining which covariates should be included in the model.

An important alternative is **leave-one-out cross validation**.

In this approach, one imagines refitting the model n times, each time excluding one of the n observations.

At iteration i, observation i is excluded. The fitted model is then used to predict the response for this observation. Call this $\widehat{y}_{(-i)}$.

Finally, we calculate the quantity

$$\text{PRESS} = \sum_{i=1}^{n} \left(y_i - \widehat{y}_{(-i)}\right)^2$$

where **PRESS** stands for **Prediction Error Sum of Squares**.

The motivation for this is as follows: Once observation i is removed from the fit, we have removed the **influence** that this observation has on the parameter estimates.

Now, we can think of observation i as being a "new" observation, and we can ask: "How well does the model, containing only these covariates, predict the response for this case?"

The quantity $(y_i - \widehat{y}_{(-i)})^2$ quantifies the amount of error in this prediction.

Exercise: Which is larger: PRESS or RSS?

Fortunately, it is not actually necessary to fit n models in order calculate PRESS.

The remarkable result is as follows:

Let $\mathbf{Y}$ be a vector consisting of the n responses, and let $\widehat{\mathbf{Y}}$ be a vector consisting of the n fitted values (from the full model).

If it is the case that $\widehat{\mathbf{Y}} = \mathbf{HY}$, then

$$y_i - \widehat{y}_{(-i)} = \frac{\widehat{\epsilon}_i}{1 - h_{ii}}$$

where h_{ii} is the i^{th} diagonal element of $\mathbf{H}$.

In the case of linear regression, it is the case that

$$\mathbf{H} = \mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T$$

where $\mathbf{X}$ whose columns hold the values of the predictors. See Ruppert for the details.

This matrix $\mathbf{H}$ is previously mentioned **hat matrix**.

The diagonal elements h_{ii} are called the **leverages**.

The leverages can be found in R using

```
> levs = hatvalues(fullmod)
```

Then we can find PRESS using

```
> PRESS = sum((fullmod$resid/(1-levs))^2)
```

We find PRESS to be 0.001974 for this model.

Of course, our goal is to find the model with lowest PRESS. In R we can do this using the function `bestglm()` with `IC="LOOCV"`:

```
> bestmod2 = bestglm(Xyframe, IC="LOOCV")
```

```
LOOCV
BICq equivalent for q in (0.0443666836953498, 0.908095644854043)
Best Model:
                Estimate   Std. Error    t value      Pr(>|t|)
(Intercept)  0.066820498 0.0004770183 140.079524 8.381181e-126
V1           0.004950250 0.0008628244   5.737262  8.591912e-08
V4          -0.008479843 0.0007636180 -11.104824  1.134645e-19
V5          -0.004005042 0.0008106249  -4.940684  2.801999e-06
V6           0.002275548 0.0002881148   7.898061  2.313625e-12
V7           0.001183985 0.0002729549   4.337659  3.206729e-05
```

We arrive at the same model as when we used AIC.

AIC versus PRESS

AIC and PRESS will often give similar, if not the same, result for the model choice.

There are a couple main reasons to prefer PRESS to AIC: First, the quantity being estimated is by PRESS is a very natural measure of prediction performance. Second, the theory behind AIC is based on the assumption that the disributional assumption for the response is correct.

A main reason to choose AIC over PRESS is that AIC is more stable. The estimate that PRESS provides of the expected prediction error is subject to large variance. Hence, PRESS may not perform well for small to moderate sample sizes.

Although PRESS is simple to calculate in the case of linear regression, this will not always be the case. In general, AIC will be easier to calculate than PRESS.

As is often the case, our past experience will often guide our choice of criterion.

Comment: AIC Calculations in R

One must use great care when comparing values of AIC reported by different functions in R. There are additive constants that are sometimes arbitrarily discarded. Of course, when using the same R function for comparing two models, there will be no problem.

Here is an important example. Suppose that the response values $Y_1, Y_2, \ldots, Y_n$ are modeled such that they are indpendent, normally-distributed random variables with mean $m(\boldsymbol{\beta}, \mathbf{x}_i)$ and variance σ^2. Here, $\mathbf{x}_i$ is the vector of predictors.

Exercise: Show that the value of the log-likelihood is

$$\ell(\boldsymbol{\beta}, \sigma^2) = -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log(\sigma^2) - \sum_{i=1}^{n}\left[\frac{(y_i - m(\boldsymbol{\beta}, \mathbf{x}_i))^2}{2\sigma^2}\right].$$

in this case.

Then, one can show that when you evaluate the log-likelihood at its peak,

$$-2\ell(\widehat{\boldsymbol{\beta}}, \widehat{\sigma}^2) = n\log(2\pi) + n\log(\mathrm{RSS}/n) + n.$$

The R function `AIC()` returns this full expression, plus two times the number of **total** number of parameters (those in $\boldsymbol{\beta}$, and σ^2).

The R function `extractAIC()` returns only

$$n\log(\mathrm{RSS}/n)$$

plus two times the number of parameters in $\boldsymbol{\beta}$.

Hence, the values returned by `AIC()` and `extractAIC()` differ by

$$n\log(2\pi) + n + 2.$$

This is constant across all models being considered, and hence is of no practical importance. Continuing our example from above, (where $n = 125$),

```
> AIC(ff3modPNC)
[1] 297.1191
> extractAIC(ff3modPNC)
[1]   4.00000 -59.61554
```

The function `step()` uses `extractAIC()` when calculating AIC.

Analysis of Variance

The term **analysis of variance (ANOVA)** is somewhat old-fashioned, but refers broadly to tools and tests which attempt to decompose the **total variability in the response into different components**.

First, we will define the **total sum of squares (SSTO or SST)** as

$$\text{SSTO} = \sum_{i=1}^{n} \left(Y_i - \overline{Y}\right)^2$$

Exercise: Interpret the quantity SSTO in the context of regression. Think about the simplest possible model for predicting the response.

We have already defined the **residual sum of squares (RSS)** or **sum of squared errors (SSE)** as

$$\text{SSE} = \sum_{i=1}^{n} \left(Y_i - \widehat{Y}_i\right)^2$$

We will also define the **regression sum of squares (**SSR**)** as

$$\text{SSR} = \sum_{i=1}^{n} \left(\widehat{Y}_i - \overline{Y}\right)^2$$

Exercise: Show that $\text{SSTO} = \text{SSR} + \text{SSE}$.

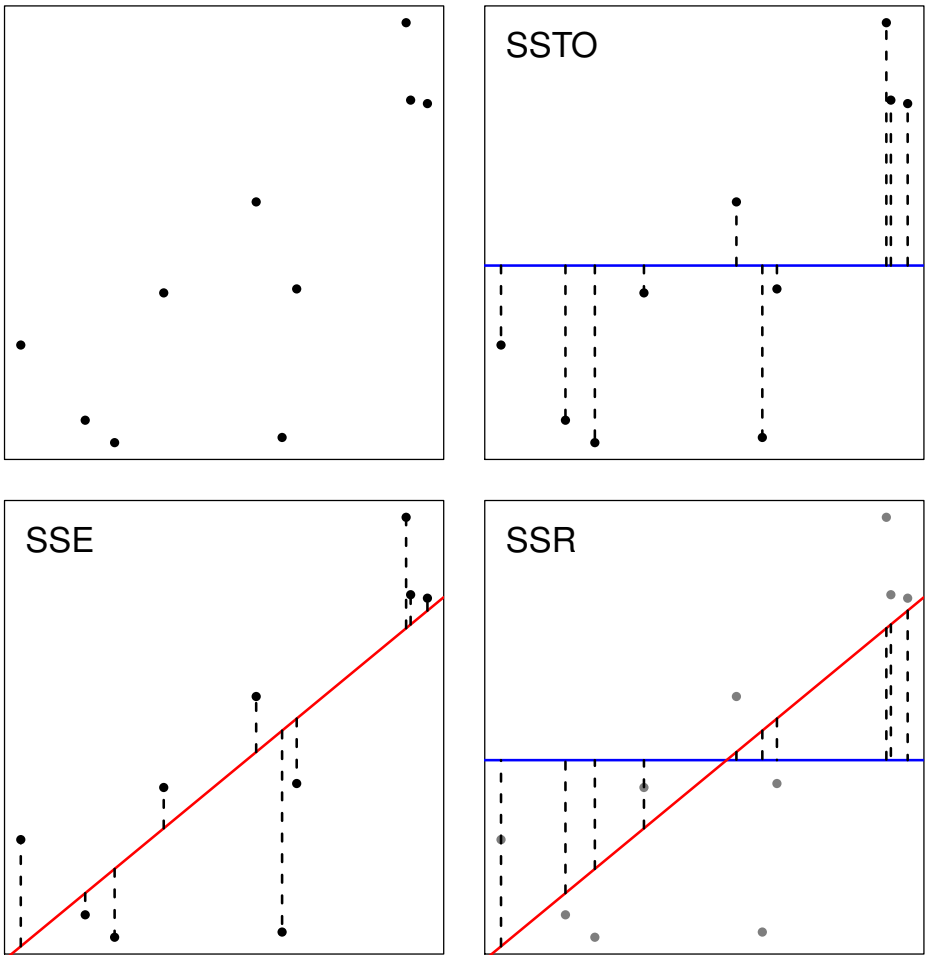

Figure 3: This figure shows the decomposition of the total sum of squares (SSTO) into the sum of squared errors (SSE) and the regression sum of squares (SSR). The top left plot shows the original scatter plot. The top right plot depicts how SSTO is calculated from the deviations of the points around the mean of the response values. The bottom left plot shows the residuals, these are squared and summed to find SSE. Finally, the bottom right plot shows how SSR is calculated from the deviations from the regression line to the mean line.

The ANOVA Table

Each of the three "sums of squares" defined previously has associated with it a certain number of **degrees of freedom**.

In particular, we say that "SSTO has $n-1$ degrees of freedom" because the list of n deviations

$$Y_1 - \overline{Y},\ Y_2 - \overline{Y},\ \ldots,\ Y_n - \overline{Y}$$

is subject to one restriction, namely that they must sum to zero.

We say that "SSE has $n-p-1$ degrees of freedom" because the n residuals

$$Y_1 - \widehat{Y}_1,\ Y_2 - \widehat{Y}_2,\ \ldots,\ Y_n - \widehat{Y}_n$$

is subject to $p+1$ restrictions, i.e. they are orthogonal to each of the columns of $\mathbf{X}$.

Finally, the remaining p degrees of freedom are attributed to SSR because there are p free parameters in the regression model.

This information is classically summarized in the **analysis of variance table**, as shown below.

Source	**SS**	**DF**	**MS**	**F**
Regression	SSR	p	SSR/p	MSR/MSE
Error	SSE	$n-p-1$	$\mathrm{SSE}/(n-p-1)$	
Total	SSTO	$n-1$		

The column **MS** provides the **mean squares** for each of regression and error. You will note that the **mean squared error (MSE)** is simply our unbiased estimator of σ^2, i.e., $E(\mathrm{MSE}) = \sigma^2$.

Global F Test for Regression

Assume that the irreducible errors are i.i.d. normal with mean zero and variance σ^2, as is standard. Then we can prove the following:

Important Theoretical Result: If it is the case that

$$\beta_1 = \beta_2 = \cdots = \beta_p = 0$$

then SSR/σ^2 has the chi-squared distribution with p degrees of freedom. Further, SSE/σ^2 has the chi-squared distribution with $n - p - 1$ degrees of freedom. Finally, SSE and SSR are independent.

Exercise: What does this result tell us about the distribution of

$$F = \frac{\mathrm{MSR}}{\mathrm{MSE}}$$

when $\beta_1 = \beta_2 = \cdots = \beta_p = 0$?

Exercise: Explain how the results on the previous page form the basis for a useful hypothesis test.

Testing a Subset of the β's

The Global F Test described above takes an "all-or-nothing" approach to testing β parameters. Suppose instead one wants to test the null hypothesis

$$H_0 : \beta_1 = \beta_2 = \cdots = \beta_k = 0$$

versus

$$H_1 : \text{at least one of } \beta_1, \beta_2, \ldots, \beta_k \neq 0.$$

where $k < p$.[4]

Exercise: Explain why this test would be particularly useful when working with factors as predictors.

[4] The ordering of the predictors is arbitrary for the purposes of this test, so we can, without affecting anything, just assume that we want to test the first k β coefficients.

To see how this test is constructed, first refer to the regression model with $\beta_1 = \beta_2 = \cdots = \beta_k = 0$ as the **reduced model**. It is "reduced" in the sense that it has fewer parameters than the **full model**.

Imagine fitting this reduced model, and saving the SSE from this fit as $\mathrm{SSE}_{\text{red}}$. Then, under the normality assumption for the irreducible errors, and when the null hypothesis

$$H_0 \colon \beta_1 = \beta_2 = \cdots = \beta_k = 0.$$

is true, the statistic

$$F = \frac{(\mathrm{SSE}_{\text{red}} - \mathrm{SSE}_{\text{full}})/k}{\mathrm{MSE}_{\text{full}}}$$

has the F-distribution with k and $n - p - 1$ degrees of freedom.

The test is conducted as above: If the value of F is too large, reject H_0 in favor of the alternative hypothesis that at least one of the β_i under consideration is not equal to zero.

Conducting the Test in R

R has a built-in function `anova()` which, among other things, will report the results of this test. The syntax is simple:

```
proslmfull = lm(log(PSA)~log(CV)+log(PW)+Age+sqrt(BPH)+
    sqrt(CP), data=prosdat)

proslmred = lm(log(PSA)~log(CV)+log(PW)+Age, data=prosdat)

> anova(proslmred, proslmfull)
Analysis of Variance Table

Model 1: log(PSA)~log(CV)+log(PW)+Age
Model 2: log(PSA)~log(CV)+log(PW)+Age+sqrt(BPH)+sqrt(CP)

  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1     93 52.600
2     91 50.729  2    1.8707 1.6779 0.1925
```

Exercise: Describe the conclusion of the test shown above.

Example: Predicting Insurance Claims Severity

This challenge comes from Kaggle, with details that can be found at

`https://www.kaggle.com/c/allstate-claims-severity/`

Figure 4: Screen shot from Kaggle.

The full training set can be downloaded from their site. We will work with a reduced version, consisting of 5000 observations, each with 76 predictor values. Sixty-two of these predictors are categorical, while the remaining 14 can be treated as continuous.

This data can be read in using

```
> insdat = read.table(
  "http://www.stat.cmu.edu/~cschafer/MSCF/Allstatesub.txt",
  sep=",", header=T)
```

By looking at `names(insdat)`, one can see that the predictors have been given generic names to protect the intellectual property of Allstate. The final column, named `loss`, is the response.

Our objective at this point is to learn some more advanced R syntax for dealing with cases when there are a large number of predictors.

Consider the R code below.

```
fulllm = lm(loss ~ ., data=insdat)

predlist = attributes(terms(formula(fulllm)))$term.labels

pvalue = numeric(length(predlist))

for(i in 1:length(predlist))
{
   holdout = drop1(fulllm, predlist[i], test="F")
   pvalue[i] = holdout$`Pr(>F)`[2]
}

cbind(predlist, signif(pvalue))
```

Exercise: Go through the above code line-by-line to explore and understand the R syntax.