Chapter 14

Missing Data

Missing data is the situation where some values of some cases are missing. This is not uncommon. Dealing with missing data is time consuming. In my experience, fixing up problems caused by missing data sometimes takes longer than the analysis itself.

What can be done? Obviously, finding the missing values is the best option but this is not always possible. Next ask why the data are missing. If the reason for a datum being missing is non-informative, then a fix is easier. For example, if a data point is missed because it was large then this could cause some bias and a simple fix is not possible. Patients may drop out of a drug study because they feel their treatment is not working - this would cause bias.

Here are several fix-up methods to use when data are missing for noninformative reasons:

1. Delete the case with missing observations. This is OK if this only causes the loss of a relatively small number of cases. This is the simplest solution.

2. Fill-in or *impute* the missing values. Use the rest of the data to predict the missing values. Simply replacing the missing value of a predictor with the average value of that predictor is one easy method. Using regression on the other predictors is another possibility. It's not clear how much the diagnostics and inference on the filled-in dataset is affected. Some additional uncertainty is caused by the imputation which needs to be allowed for.

3. Missing observation correlation. Consider just (x_i, y_i) pairs with some observations missing. The means and SDs of x and y can be used in the estimate even when a member of a pair is missing. An analogous method is available for regression problems.

4. Maximum likelihood methods can be used assuming the multivariate normality of the data. The EM algorithm is often used here. We will not explain the details but the idea is essentially to treat missing values as nuisance parameters.

Suppose some of the values in the Chicago Insurance dataset were missing. I randomly declared some the observations missing in this modified dataset. Read it in and take a look:

```
> data(chmiss)
> chmiss
      race fire theft  age involact income
60626 10.0  6.2    29 60.4       NA 11.744
60640 22.2  9.5    44 76.5      0.1  9.323
60613 19.6 10.5    36   NA      1.2  9.948
```

```
60657 17.3  7.7    37   NA      0.5 10.656
--- etc ---
60645  3.1  4.9    27   NA      0.0 13.731
```

There are 20 missing observations denoted by NA here. It's important to know what the missing value code is for the data and/or software you are using. What happens if we try to fit the model?

```
> g <- lm(involact ~ .,chmiss)
> summary(g)
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.11648    0.60576   -1.84  0.07947
race         0.01049    0.00313    3.35  0.00302
fire         0.04388    0.01032    4.25  0.00036
theft       -0.01722    0.00590   -2.92  0.00822
age          0.00938    0.00349    2.68  0.01390
income       0.06870    0.04216    1.63  0.11808

Residual standard error: 0.338 on 21 degrees of freedom
Multiple R-Squared: 0.791,      Adjusted R-squared: 0.741
F-statistic: 15.9 on 5 and 21 degrees of freedom,       p-value: 1.59e-06
```

Any case with at least one missing value is omitted from the regression. You can see there are now only 21 degrees of freedom - almost half the data is lost. We can fill in the missing values by their variable means as in:

```
> cmeans <- apply(chmiss,2,mean,na.rm=T)
> cmeans
    race     fire    theft      age involact   income
35.60930 11.42444 32.65116 59.96905  0.64773 10.73587
> mchm <- chmiss
> for(i in c(1,2,3,4,6)) mchm[is.na(chmiss[,i]),i] <- cmeans[i]
```

We don't fill in missing values in the response because this is the variable we are trying to model. Now refit:

```
> g <- lm(involact ~ ., data=mchm)
> summary(g)
Coefficients:
             Value Std. Error t value Pr(>|t|)
(Intercept)  0.0707  0.5094     0.1387  0.8904
       race  0.0071  0.0027     2.6307  0.0122
       fire  0.0287  0.0094     3.0623  0.0040
      theft -0.0031  0.0027    -1.1139  0.2723
        age  0.0061  0.0032     1.8954  0.0657
     income -0.0271  0.0317    -0.8550  0.3979

Residual standard error: 0.3841 on 38 degrees of freedom
Multiple R-Squared: 0.6819
F-statistic: 16.3 on 5 and 38 degrees of freedom, the p-value is 1.41e-08
```

Compare with the previous results - what differences do you see? Different statistical packages have different ways of handling missing observations. For example, the default behavior in S-PLUS would refuse to fit the model at all.

The regression coefficients are now all closer to zero. The situation is analogous to the error in variables case. The bias introduced by the fill-in method can be substantial and may not be compensated by the attendant reduction in variance.

We can also use regression methods to predict the missing values of the covariates. Let's try to fill-in the missing race values:

```
> gr <- lm(race ~ fire+theft+age+income,chmiss)
> chmiss[is.na(chmiss$race),]
      race fire theft  age involact income
60646   NA  5.7    11 27.9      0.0 16.250
60651   NA 15.1    30 89.8      0.8 10.510
60616   NA 12.2    46 48.0      0.6  8.212
60617   NA 10.8    34 58.0      0.9 11.156
> predict(gr,chmiss[is.na(chmiss$race),])
  60646   60651   60616   60617
-17.847  26.360  70.394  32.620
```

Can you see a problem with filling these values in? Obviously we would need to put more work into the regression models used to fill-in the missing values. One trick that can be applied when the response is bounded between 0 and 1 is the logit transformation:

$$y \to \log(y/(1-y))$$

This transformation maps to the whole real line. We define the logit function and its inverse:

```
> logit <- function(x) log(x/(1-x))
> ilogit <- function(x) exp(x)/(1+exp(x))
```

We now fit the model with a logit-transformed response and then back-transform the predicted values remembering to convert our percentages to proportions and vice versa at the appropriate times:

```
> gr <- lm(logit(race/100) ~ fire+theft+age+income,chmiss)
> ilogit(predict(gr,chmiss[is.na(chmiss$race),]))*100
   60646    60651    60616    60617
 0.41909 14.73202 84.26540 21.31213
```

We can see how our predicted values compare to the actual values:

```
> data(chicago)
> chicago$race[is.na(chmiss$race)]
[1]  1.0 13.4 62.3 36.4
```

So our first two predictions are good but the other two are somewhat wide of the mark.

Like the mean fill-in method, regression fill-in will also introduce a bias towards zero in the coefficients while tending to reduce the variance also. The success of the regression method depends somewhat on the collinearity of the predictors - the filled-in values will be more accurate the more collinear the predictors are.

For situations where there is a substantial proportion of missing data, I recommend that you investigate more sophisticated methods, likely using the EM algorithm. Multiple imputation is another possibility. The fill-in methods described above will be fine when only a few cases need to be filled but will become less reliable as the proportion of missing cases increases.