Data Analysis & Graphics Using R – Solutions to Exercises (April 24, 2004)

Preliminaries

```
> library(DAAG)
```

Exercise 2
The final three sentences have been reworded
For each of the data sets `elastic1` and `elastic2`, determine the regression of stretch on distance. In each case determine

(i) fitted values and standard errors of fitted values and

(ii) the R^2 statistic. Compare the two sets of results. What is the key difference between the two sets of data?

Use the robust regression function `rlm()` from the *MASS* package to carry out the analysis in Section 5.2. Plot residuals from this analysis against fitted values. What differences do you notice, in the regression coefficients, in their standard errors, or in the residuals?

The required regressions are as follows:

```
> data(elastic1)
> data(elastic2)
> e1.lm <- lm(distance ~ stretch, data = elastic1)
> e2.lm <- lm(distance ~ stretch, data = elastic2)
```

The fitted values and standard errors of the fits are then:

```
> predict(e1.lm, se.fit = TRUE)

$fit
    1     2     3     4     5     6     7
183.1 235.7 196.3 209.4 170.0 156.9 222.6

$se.fit
[1]  6.587 10.621  5.892  6.587  8.332 10.621  8.332

$df
[1] 5

$residual.scale
[1] 15.59
```

The R^2 statistic, in each case, is obtained as follows:

```
> summary(e1.lm)$r.squared

[1] 0.7992

> summary(e2.lm)$r.squared

[1] 0.9808
```

The standard errors are somewhat smaller for the second data set than for the first, while the R^2 value is much larger for the second than the first. The main reason for the difference in R^2 is the larger range of `stretch` for the second data set. There is more variation to explain. More specifically

$$\begin{aligned} R^2 &= 1 - \frac{(n-2)s^2}{\sum(y-\bar{y}^2)} && (1) \\ &= 1 - \frac{s^2}{\sum(y-\bar{y}^2)/(n-2)} && (2) \\ & && (3) \end{aligned}$$

Increasing the range of values greatly increases the denominator. If the line is adequate over the whole of the range, s^2 will, as here, not change much. (For these data, in spite of the greater range, it reduces somewhat.)

The robust regression fits can be obtained as follows:

```
> library(MASS)
> e1.rlm <- rlm(distance ~ stretch, data = elastic1)
> e2.rlm <- rlm(distance ~ stretch, data = elastic2)
```

The robust regression fits can be obtained as follows:

The residual plots can be obtained for `rlm` in the same was as for `lm`:

```
> par(mfrow = c(1, 4))
> plot(e1.rlm, which = 1, panel = panel.smooth)
> plot(e1.lm, which = 1, panel = panel.smooth)
> plot(e2.rlm, which = 1, panel = panel.smooth)
> plot(e2.lm, which = 1, panel = panel.smooth)
> par(mfrow = c(1, 1))
```

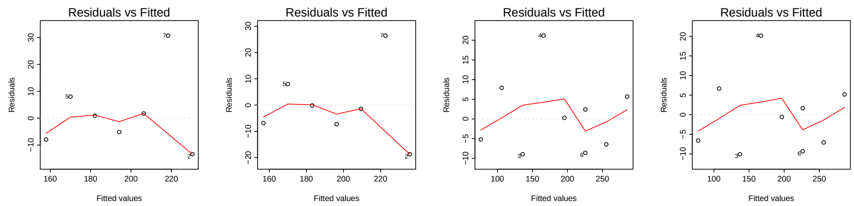

Figure 1: Plots of residuals versus fitted values, for the models `e1.rlm`, `e1.lm`, `e2.rlm` and `e2.lm`

For comparison purposes, we include residual plots for the ordinary regression fits. Note, in particular, how the robust regression has reduced the weight of the outlying observation in the first data set. The residual at that point is larger than it was using ordinary least-squares. The residual plots for the ordinary and robust fits are very similar for the second data set, since there are no outlying observations.

As can be seen in the summaries below, the ordinary and robust fits for the first data set give quite different estimates of the slope and intercept. The robust fit is more in line with both sets of results obtained for the second data set.

Note also the downward effect of the robust regression on the residual standard error. This is again due to the down-weighting of the outlying observation.

```
> summary(e1.rlm)

Call: rlm(formula = distance ~ stretch, data = elastic1)
Residuals:
     1      2      3      4      5      6      7
  0.92 -13.40  -5.16   1.76   8.00  -7.92  30.68

Coefficients:
            Value   Std. Error t value
(Intercept) -95.747  60.690     -1.578
stretch       6.040   1.260      4.793

Residual standard error: 11.7 on 5 degrees of freedom

Correlation of Coefficients:

          (Intercept)
  stretch -0.997

> summary(e1.lm)

Call:
lm(formula = distance ~ stretch, data = elastic1)

Residuals:
      1       2       3       4       5       6       7
 -0.143 -18.714  -7.286  -1.429   8.000  -6.857  26.429

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -119.14      70.94   -1.68   0.1539
stretch         6.57       1.47    4.46   0.0066

Residual standard error: 15.6 on 5 degrees of freedom
Multiple R-Squared: 0.799,        Adjusted R-squared: 0.759
F-statistic: 19.9 on 1 and 5 DF,  p-value: 0.00663

> summary(e2.rlm)

Call: rlm(formula = distance ~ stretch, data = elastic2)
Residuals:
   Min     1Q Median     3Q    Max
-8.951 -6.454  0.297  5.670 21.173

Coefficients:
            Value    Std. Error t value
(Intercept) -103.055   14.496     -7.109
stretch        5.975    0.292     20.438

Residual standard error: 9.57 on 7 degrees of freedom

Correlation of Coefficients:

          (Intercept)
  stretch -0.975
```

```
> summary(e2.lm)

Call:
lm(formula = distance ~ stretch, data = elastic2)

Residuals:
    Min      1Q  Median      3Q     Max
-10.083  -7.083  -0.583   5.167  20.167

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -100.917     15.610   -6.46  0.00035
stretch        5.950      0.315   18.90  2.9e-07

Residual standard error: 10.4 on 7 degrees of freedom
Multiple R-Squared: 0.981,        Adjusted R-squared: 0.978
F-statistic:  357 on 1 and 7 DF,  p-value: 2.89e-07
```

Exercise 3
Using the data frame `cars` (in the *base* package), plot `distance` (i.e. stopping distance) versus `speed`. Fit a line to this relationship, and plot the line. Then try fitting and plotting a quadratic curve. Does the quadratic curve give a useful improvement to the fit? [Readers who have studied the relevant physics might develop a model for the change in stopping distance with speed, and check the data against this model.]

The data can be plotted using

```
> data(cars)
> plot(dist ~ speed, data = cars, xlab = "stopping distance", pch = 16)
```

The linear model can be fit, and a line added, as follows:

```
> cars.lm <- lm(dist ~ speed, data = cars)
> abline(cars.lm)
```

One way of fitting a quadratic curve to the data is as follows:

```
> cars.lm2 <- lm(dist ~ speed + I(speed^2), data = cars)
```

The following overlays the quadratic curve:

```
> xval <- pretty(cars$speed, 50)
> hat2 <- predict(cars.lm2, newdata = list(speed = xval))
> lines(xval, hat2, col = "red", lty = 2, lwd = 2)
```

Here is the graph

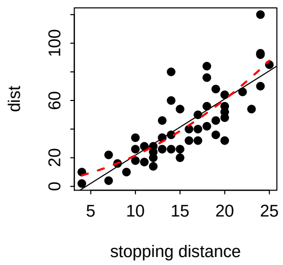

Figure 2: Quadratic curve fitted to car data.

Based on what we've seen so far, the quadratic curve does not appear to fit the data much better than the line. Checking the summary and the p-value might lead us to believe that the quadratic term is not needed:

```
> summary(cars.lm2)

Call:
lm(formula = dist ~ speed + I(speed^2), data = cars)

Residuals:
   Min     1Q Median     3Q    Max
-28.72  -9.18  -3.19   4.63  45.15

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    2.470     14.817    0.17     0.87
speed          0.913      2.034    0.45     0.66
I(speed^2)     0.100      0.066    1.52     0.14

Residual standard error: 15.2 on 47 degrees of freedom
Multiple R-Squared: 0.667,      Adjusted R-squared: 0.653
F-statistic: 47.1 on 2 and 47 DF,  p-value: 5.85e-12
```

The relevant physics suggests that stopping distance is, in fact, a nonlinear function of speed. An over-simplified model is

$$\text{distance} = k \text{ speed}^2$$

where k is a constant, which is inversely related to the acceleration (actually deceleration), which is assumed constant here. Because of the unrealistic assumption that k is independent of the deceleration, this model should be used only as a start. The actual deceleration will not be constant, and there is likely a fair bit of noise associated with it. Note that the error term, which we have not specified, is likely to be a function of `speed`.

Also, we have not consulted a residual plot. In view of the non-significant quadratic term, we examine the residual plot for the model with a linear term.

```
> plot(cars.lm, which = 1, panel = panel.smooth)
```

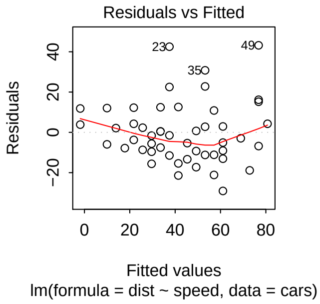

Figure 3: Plot of residuals versus fitted values, for the cars data.

In view of the clear trend in the plot of residuals, it might be safest to include the quadratic term.

Note however that the error variance (even after the trend from the residuals is taken out) is not constant, but increases with the fitted values. Alternatives are to try a weighted least-squares fit, or to try a variance-stabilizing transformation. If we are fortunate, a variance-stabilizing transformation may also reduce any trend that may be present. In particular, a square-root transformation seems to work well:

```
> cars.lm3 <- lm(sqrt(dist) ~ speed, data = cars)
> plot(cars.lm3, which = 1, panel = panel.smooth)
```

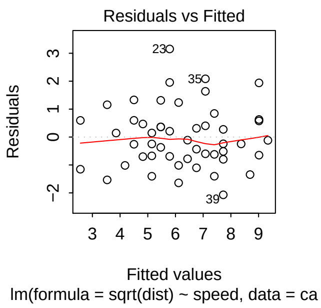

Figure 4: Residuals from the regression of the square root of distance on speed, for the car data.

Incidentally, the square root transformation is also indicated by the Box-Cox procedure (see exercise 5). This is seen from the output to either of

```
> boxcox(dist ~ speed, data = cars)
> boxcox(dist ~ I(speed^2), data = cars)
```

Exercise 4
In the data set `pressure` (*base* package), examine the dependence of pressure on temperature.
[The relevant theory is that associated with the Claudius-Clapeyron equation. Search on the internet, or look in a library, for information on this equation.]

First we ignore the Claudius-Clapeyron equation, and try to transform `pressure`. When the logarithmic transformation is too extreme, as happens in this case, a power transformation with a positive exponent may be a candidate. A square root transformation is a possibility:

```
> data(pressure)
> pressure$K <- pressure$temperature + 273
> p.lm <- lm(I(pressure^0.5) ~ K, data = pressure)
> plot(p.lm, which = 1)
```

A systematic search for a smaller exponent is clearly required.

The Clausius-Clapeyron equation suggests that `log(pressure)` should be a linear function of `1/K`, where `K` is degrees kelvin.

```
> p.lm2 <- lm(log(pressure) ~ I(1/K), data = pressure)
> plot(p.lm2, which = 1)
```

Consulting the residual plot, we see too much regularity. One point appears to be an outlier, and should be checked against other data sources. Some improvement is obtained by considering polynomials in the inverse of temperature. For example, the quadratic can be fit as follows:

```
> p.lm4 <- lm(log(pressure) ~ poly(1/K, 2), data = pressure)
> plot(p.lm4, which = 1)
```

The residual plot still reveals some unsatisfactory features, particularly for low temperatures. However, such low pressure measurements are notoriously inaccurate. Thus, a weighted least-squares analysis would probably be more appropriate.

*Exercise 5**
Look up the help page for the function `boxcox()` from the *MASS* package, and use this function to determine a transformation for use in connection with Exercise 4. Examine diagnostics for the regression fit that results following this transformation. In particular, examine the plot of residuals against temperature. Comment on the plot. What are its implications for further investigation of these data?

The Box-Cox procedure can be applied to the pressure data as follows:

```
> boxcox(pressure ~ K, data = pressure)
```

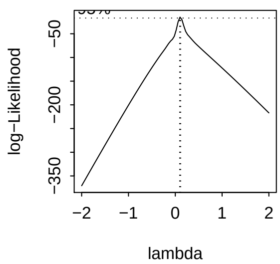

Figure 5: Boxcox plot, for pressure versus degrees kelvin

This suggests a power of around 0.1.

Thus, we might fit the model using

```
lm(I(pressure^.1) ~ K, data=pressure)
```

However, remembering that the physics suggests a transformation of temperature, we should really look at

```
> boxcox(pressure ~ I(1/K), data = pressure)
```

The result is

```
> boxcox(pressure ~ I(1/K), data = pressure)
```

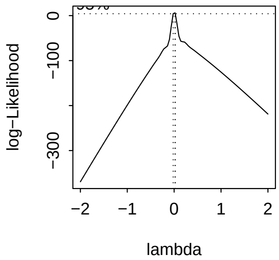

Figure 6: Boxcox plot, for pressure versus $1/K$

This shows clearly that the logarithmic transformation is likely to be helpful. (However check the effect of the Box-Cox transformation on the trend.)

Exercise 6
The following code gives panels B and D of Figure 5.4. Annotate this code, explaining what each function does, and what the parameters are: ......
The code is omitted, as the annotated version appears below ......

```
library(DAAG)      # loads the DAAG library
data(ironslag)     # loads the ironslag data frame
attach(ironslag)   # attaches data frame contents to search path
par(mfrow=c(2,2)) # enables a 2x2 layout on the graphics window
ironslag.lm <- lm(chemical ~ magnetic)
                   # regress chemical on magnetic
chemhat <- fitted(ironslag.lm)  # assign fitted values to chemhat
res <- resid(ironslag.lm)       # assign residuals to res
## Figure 5.4B
plot(magnetic, res, ylab = "Residual", type = "n")  # type = "n"
                   # Set up axes with correct ranges, do not plot
panel.smooth(magnetic, res, span = 0.95)  # plots residuals
                   # vs predictor, & adds a lowess smooth; f=span
## Figure 5.4D
sqrtabs <- sqrt(abs(res))  # square root of abs(residuals)
plot(chemhat, sqrtabs, xlab = "Predicted chemical",
     ylab =  expression(sqrt(abs(residual))), type = "n")
                   # suppressed plot again, as above
panel.smooth(chemhat, sqrtabs, span = 0.95)
                   # plot sqrt(abs(residuals)) vs fitted values
                   # add lowess smooth, with f=span
detach(ironslag)   # remove data frame contents from search path
```

Exercise 7
The following code gives the values that are plotted in the two panels of Figure 5.5.

```
library(modreg)
ironslag.loess <- loess(chemical ~ magnetic, data=ironslag)
chemhat <- fitted(ironslag.loess)
res2 <- resid(ironslag.loess)
sqrtabs2 <- sqrt(abs(res2))
```

Using this code as a basis, create plots similar to Figure 5.5A and 5.5B. Why have we preferred to use `loess()` here, rather than `lowess()`? [Hint: Is there a straightforward means for obtaining residuals from the curve that `lowess()` gives? What are the x-values, and associated y-values, that `lowess()` returns?]

Obtaining residuals from `lowess()` is problematic because the fitted data are sorted according to the predictor variable upon output.

One way of obtaining residuals upon use of `lowess()` is to sort the data beforehand as below:

```
> data(ironslag)
> ironsort <- ironslag[order(ironslag$magnetic), ]
> attach(ironsort)
> ironsort.lw <- lowess(magnetic, chemical)
> ironsort.resid <- chemical - ironsort.lw$y
```

Once we have the residuals (either from `loess()` or from `lowess()`), we may proceed to obtain the plots in Figure 5.5. One way is as follows:

```
> plot(ironsort.resid ~ magnetic, lwd = 2, xlab = "Magnetic", ylab = "Residual")
> lines(lowess(magnetic, ironsort.resid, f = 0.8), lty = 2)
```

To obtain the plot in Figure 5.5B, we could then do the following:

```
> sqrtabs2 <- sqrt(abs(ironsort.resid))
> plot(sqrtabs2 ~ ironsort.lw$y, xlab = "Predicted chemical", ylab = expression(sqrt(Residual))
> lines(lowess(ironsort.lw$y, sqrtabs2, f = 0.8))
> detach(ironsort)
```

One could also use `loess()` instead of `lowess()`.