Part V
Linear Models in R

For background, see SPDM: Sections 9-11.

1 Linear and Other Models in R

In this laboratory, the major attention will be on the model fitting function is `lm()`, where the `lm` stands for linear model. An aim is to encourage an expansive view of linear models, modeling reponses that may be highly non-linear in the explanatory variables.

R's implementation of linear models uses a symbolic notation, described in (Wilkinson & Rogers , 1973), that gives a straightforward means for describing elaborate and intricate models. A very similar notation is used throughout the modeling functions in R, with modification as required. Important ideas are *basis function*, *factor*, *interaction*, and *model formula*.

2 Fitting Straight Lines to Data

Exercise 1
In each of the data frames `elastic1` and `elastic2`, fit straight lines that show the dependence of `distance` on `stretch`. Plot the two sets of data, using different colours, on the same graph. Add the two separate fitted lines. Also, fit one line for all the data, and add this to the graph.

Exercise 2
In the data set pressure (*datasets*), the relevant theory is that associated with the Claudius-Clapeyron equation, by which the logarithm of the vapor pressure is approximately inversely proportional to the absolute temperature. Transform the data in the manner suggested by this theoretical relationship, plot the data, fit a regression line, and add the line to the graph. Does the fit seem adequate?
[For further details of the Claudius-Clapeyron equation, search on the internet, or look in a suitable reference text.]

Exercise 3
Run the function `plot.intersalt()`; data frame `intersalt`. Data are population average values of blood pressure, from 52 different studies. Is the fitted line reasonable? Or is it a misinterpretation of the data? Suggest alternatives to regression analysis, for getting a sense of how these results should be interpreted? What are the populations where levels are very low? What is special about these countries?
[The function `plot.intersalt()`, and the data frame `intersalt1`, are available from `http://www.maths.anu.edu.au/~johnm/courses/dm/math3346/`]

Exercise 4
Install the package *gamair* (from CRAN) and examine the help page for the data frame *hubble*. Type `data(hubble)` to bring the data frame into the workspace. (This is necessary because the `gamair` package differs from most other packages in not using the *lazy loading* mechanism for data sets.)

(a) Plot `y` (Velocity in km $\sec^{-1}$) versus `x` (Distance in Mega-parsec).

(b) Fit a line, omitting the constant term; for this the `lm()` function call is

```
lm(y ~ -1 + x, data=hubble)
```

A Mega-parsec (Mpc) is 3.09×10^{-19} km, so that we need to divide the slope by this amount, so that distances in both cases are in km. The inverse of the slope is then the age of the universe, in seconds. Divide this by $60^2 \times 24 \times 365$ to get an estimate for the age of the earth in years.
[The answer should be around 13×10^9 years.]

(c) Repeat the plot, now using logarithmic scales for both axes. Fit a line, now insisting that the coefficient of `log(x)` is 1.0 (Why?) For this, specify

```
lm(log(y) ~ 1 + offset(log(x)), data=hubble)
```

Add this line to the plot. Again, obtain an estimate of the age of the universe. Does this give a substantially different estimate for the age of the universe?

(d) In each of the straight line fits, on an untransformed scale and using logarithmic scales, do any of the points seem outlers? Investigate the effect of omitting any points that seem to be outliers?

(e) Does either plot, on an untransformed scale or using logarithmic scales, seem to show evidence of curvature?
[See further the note at the end of this set of exercises.]

3 Multiple Explanatory Variables

Exercise 5
For the data frame `oddbooks` (*DAAG*),

(a) Add a further column that gives the density.

(b) Use the function `pairs()`, or the *lattice* function `splom()`, to display the scatterplot matrix. Which pairs of variables show evidence of a strong realtionship?

(c) In each panel of the scatterplot matrix, record the correlation for that panel. (Use `cor()`) to calculate correlations).

(d) Fit the following regression relationships:

  (a) `log(weight)` on `log(thick)`, `log(height)` and `log(breadth)`.

  (b) `log(weight)` on `log(thick)` and `0.5*(log(height) + log(breadth))`. What feature of the scatterplot matrix suggests that this might make sense to use this form of equation?

(e) Take the fitted regression equation (take whichever of the two forms of equation seems preferable) and rewrite it in a form that as far as possible separates effects that arise from changes in the linear dimensions from effects that arise from changes in linear dimensions.

4 Errors in Variables

Exercise 6
Run the accompanying function `errorsINx()` several times. Comment on the results. The underlying relationship between y and x is the same in all cases. The error in x is varied, from values of x that are exact to values of x that have random errors added with a variance that is twice that of the variance of x.

4.1 Note re exercise 4

According to the relevant cosmological model, the velocity of recession of any galaxy from any other galaxy has been constant, independent of time. Those parts of the universe that started with the largest velocities of recession from our galaxy have moved furthest, with no change from the velocity just after after time 0. Thus the time from the beginning should be s/v, where s is distance, and v is velocity. The slope of the least squares line gives a combined estimate, taken over all the galaxies included in the data frame gamair. More recent data suggests, in fact, that the velocity of recession is not strictly proportional to distance.

4.2 Functions used

These are available from my web page

```
"errorsINx" <-
  function(mu = 8.25, n = 100, a = 5, b = 2.5, SDx=1, sigma = 2,
           timesSDx=(1:5)/2.5){
    mat <- matrix(0, nrow=n, ncol=length(timesSDx)+2)

    x0 <- mu*exp(rnorm(n,0,SDx/mu))/exp(0)

    y <- a + b*x0+rnorm(n,0,sigma)
    mat[, length(timesSDx)+2] <- y
    mat[,2] <- x0
    mat[,1] <- y
    sx <- sd(x0)
    k <- 2
    for(i in timesSDx){
      k <- k+1
      xWITHerror <- x0+rnorm(n, 0, sx*i)
      mat[, k] <- xWITHerror
    }
    df <- as.data.frame(mat)
    names(df) <- c("y", "x", paste("x",timesSDx,sep=""))
    df
  }

## Now use function to simulate y vs x relationships, with several
## different values of timesSDx, which specifies the ratio of the
## errors in x variance to SD[x]
    oldpar <- par(mar=c(3.6,3.1,1.6,0.6), mgp=c(2.5,0.75,0),
                  oma=c(1,1,0.6,1),
                  mfrow=c(2,3), pty="s")
    mu <- 20; n <- 100; a <- 15; b <- 2.5; sigma <- 12.5; timesSigma<-(1:5)/2.5
    mat <- errorsINx(mu = 20, n = 100, a = 15, b = 2.5, sigma = 5,
                     timesSDx=(1:5)/2.5)
    beta <- numeric(dim(mat)[2]-1)
    sx <- sd(mat[,2])
    y <- mat[, 1]
    for(j in 1:length(beta)){
      xj <- mat[,j+1]
      plot(y ~ xj, xlab="", ylab="", col="gray30")
      if(j==1)
        mtext(side=3, line=0.5, "No error in x") else{
          xm <- timesSigma[j-1]
          mtext(side=3, line=0.5, substitute(tau == xm*s[z], list(xm=xm)))
        }
      if(j>=4)mtext(side=1, line=2, "x")
      if(j%%3 == 1)mtext(side=2, line=2, "y")
      errors.lm <- lm(y ~ xj)
      abline(errors.lm)
      beta[j] <- coef(errors.lm)[2]
      bigsigma <- summary(errors.lm)$sigma
      print(bigsigma/sigma)
      abline(a, b, lty=2)
    }
    print(round(beta, 3))

plot.intersalt <-
function (dset = intersalt1, figno = 2)
{
    oldpar <- par(oma = c(6.5, 0, 0, 0), mar = par()$mar - c(0,
        0, 3.5, 0))
    on.exit(par(oldpar))
    lowna <- c(4, 5, 24, 28)
    plot(dset$na, dset$bp, pch = 15, ylim = c(50, 85),
         xlab = "Median sodium excretion (mmol/24hr)",
         ylab = "Median diastolic BP (mm Hg)", type = "n")
    points(dset$na[-lowna], dset$bp[-lowna], pch = 16)
    points(dset$na[lowna], dset$bp[lowna], pch = 1, lwd = 3)
    u <- lm(bp ~ na, data = dset)
    abline(u$coef[1], u$coef[2])

    figtxt <-
       paste("Fig. ", figno, ": Plot of median blood pressure versus salt",
             "\\n(measured by sodium excretion) for 52 human",
             "\\npopulations. Four results (open circles) are for",
             "\\nnon-industrialised societies with very low salt intake,",
             "\\nwhile other results are for industrialised societies.",
       sep = "")
    mtext(side = 1, line = 6, figtxt, cex = 1.1, adj = 0, at = -20)
}
```

Part VI
Exploiting the Model Matrix

For background, see SPDM: Sections 12-13.

It is straightforward to model qualitative effects. In R, factors, with one level for each qualitative category, are typically used to account for qualitative effects. Additonally, and perhaps more surpringingly, it is straightforward to use linear models to model curvilinear effects,

1 A One-way classification – eggs in the cuckoo's nest

This demonstrates the use of linear models to fit qualitative effects.

Like many of nature's creatures, cuckoos have a nasty habit. They lay their eggs in the nests of other birds. First, let's see how egg length changes with the host species. This will use the graphics function `stripplot()` from the *lattice* package. The data frame `cuckoos` is in the *DAAG* package.

```
> par(mfrow = c(1, 2))
> library(DAAG)
> library(lattice)
> names(cuckoos)[1] <- "length"
> table(cuckoos$species)

hedge.sparrow  meadow.pipit  pied.wagtail         robin    tree.pipit
           14            45            15            16            15
         wren
           15

> stripplot(species ~ length, data = cuckoos)
> cuckoo.strip <- stripplot(breadth ~ length, groups = species,
+     data = cuckoos, auto.key = list(columns = 3))
> print(cuckoo.strip)
> par(mfrow = c(1, 1))
```

Now estimate the means for each of the groups. We will use two forms of the model. The first gives the species means directly, as parameters for the model. In the second parameterization, the first parameter estimate is the mean for the first species, while later estimates are differences from the first species.

Exercise 1
In examining how cuckoo egg length varies between host species, the following forces all parameters to be species means

```
> lm(length ~ -1 + species, data = cuckoos)
```

The following makes the first species the baseline

```
> lm(length ~ species, data = cuckoos)
```

Use the function `fitted()` to calculate the fitted values in each case, and check that they are the same. Reconcile the two sets of results.

Exercise 2
Use the `termplot()` function to show the effects of the different factor levels. Be sure to call the function with `partial.resid=TRUE` and with `se=TRUE`. Does the variability seem similar for all host species?

Exercise 3
Obtain the standard deviation for each species:

```
> attach(cuckoos)

> sapply(split(length, species), sd)

hedge.sparrow  meadow.pipit  pied.wagtail         robin    tree.pipit
    1.0494373     0.9195849     1.0722917     0.6821229     0.8800974
         wren
    0.7542262
```

[The function `split()` splits the lengths into six sublists, one for each species. The function `sapply()` applies the function calculation to the vectors of lengths in each separate sublist.]
Check that the SD seems smallest for those species where the SD seems, visually, to be smallest.

Exercise 4
Obtain, for each species, the standard error of the mean. This is obtained by dividing the standard deviation by the square root of the number of values:

```
> sdev <- sapply(split(length, species), sd)
> n <- sapply(split(length, species), length)
> sdev/sqrt(n)

hedge.sparrow  meadow.pipit  pied.wagtail         robin    tree.pipit
    0.2804739     0.1370836     0.2768645     0.1705307     0.2272402
         wren
    0.1947404
```

Alternatively, create a function that calculates the standard error of the mean in a single calculation:

```
> se <- function(x) sd(x)/sqrt(length(x))
> sapply(split(length, species), se)

hedge.sparrow  meadow.pipit  pied.wagtail         robin    tree.pipit
    0.2804739     0.1370836     0.2768645     0.1705307     0.2272402
         wren
    0.1947404
```

This can be done in a single line of code. The function `se()` can be inserted as an anonymous function. In this case, it does not need a name; the function definition is inserted where the function name would otherwise appear:

```
> sapply(split(length, species), function(x) sd(x)/sqrt(length(x)))

hedge.sparrow  meadow.pipit  pied.wagtail         robin    tree.pipit
    0.2804739     0.1370836     0.2768645     0.1705307     0.2272402
         wren
    0.1947404
```

2 Regression Splines – one explanatory variable

Exercise 5
The following is based on the `fruitohms` data frame (in *DAAG*)

(a) Plot `ohms` against `juice`.

(b) Try the following:

```
> plot(ohms ~ juice, data = fruitohms)
> with(fruitohms, lines(lowess(juice, ohms)))
> with(fruitohms, lines(lowess(juice, ohms, f = 0.2), col = "red"))
```

Which of the two fitted curves best captures the pattern of change?

(c) Now fit a natural regression spline. Try for example:

```
> library(splines)
> plot(ohms ~ juice, data = fruitohms)
> hat <- with(fruitohms, fitted(lm(ohms ~ ns(juice, 4))))
> with(fruitohms, lines(juice, hat, col = 3))
> attributes(ns(fruitohms$juice, 4))$knots

   25%    50%    75%
18.625 41.000 48.000
```

Experiment with different choices for the number of degrees of freedom. How many degrees of freedom seem needed to adequately capture the pattern of change? Plot the spline basis functions. Add vertical lines to the plot that show the knot locations.

Exercise 6
In the `geophones` data frame (*DAAG*), plot `thickness` against distance. Use regression splines, as in Exercise 6, to fit a suitable curve through the data. How many degrees of freedom seem needed? Add vertical lines to the plot that show the locatons of the knots.

3 Regression Splines – Two or More Explanatory Variables

We begin by using the `hills2000` data (*DAAG* version 0.84 or later) to fit a model that is linear in `log(dist)` and `log(climb)`, thus

```
> lhills2k <- log(hills2000[, 7:9])
> names(lhills2k) <- c("ldist", "lclimb", "ltime")
> lhills2k.lm <- lm(ltime ~ ldist + lclimb, data = lhills2k)
```

Use `termplot()` to check departures from linearity in `lhills2k.lm`. Note whether there seem to be any evident outliers.

Exercise 7
Now use regression splines to take out the curvature that was evident when `ltime` was modeled as a linear function of `ldist` and `lclimb`. Use `termplot()` to guide your choice of degrees of freedom for the spline bases. For example, you might try

```
> lhills2k.ns <- lm(ltime ~ ns(ldist, 4) + ns(lclimb, 3), data = lhills2k)
```

Again, examine the diagnostic plots? Are there any points that should perhaps be regarded as outliers?

Exercise 8

The *MASS* package has the function `lqs()` that can be used for a *resistant* regression fit, i.e., the effect of outlying points is attenuated. Try the following

```
> library(MASS)
> lhills2k.lqs <- lqs(ltime ~ ns(ldist, 4) + ns(lclimb, 3), data = lhills2k)
> plot(resid(lhills2k.lqs) ~ fitted(lhills2k.lqs))
> big4 <- order(abs(resid(lhills2k.lqs)), decreasing = TRUE)[1:4]
> text(resid(lhills2k.lqs)[big4] ~ fitted(lhills2k.lqs)[big4],
+     labels = rownames(lhills2k)[big4], pos = 4)
```

Try the plot without the two largest outliers. Does this make any difference of consequence to the fitted values?