Updates and Corrections (as of May 28, 2012)

Data Analysis and Graphics Using R – An Example-Based Approach, 3rd edn

John Maindonald (email: john.maindonald@anu.edu.au) and John Braun

Webpage: `http://www.maths.anu.edu.au/~johnm/r-book/r-book.html`.

First (2010) and second (2011) printing

Chapter 1, p.31, line -11: Omit `pch=c(4,11),`
[To get Figure 1.3 as shown, specify:

```
xyplot(ht ~ wt | sport, groups=sex, par.settings=simpleTheme(pch=c(4,1)),
       aspect=1, data=ais, auto.key=list(columns=2, space="right"),
       subset=sport%in%c("Row","Swim"))
```

Use of `par.settings` to supply values for `pch` ensures that the settings affect the key as well as the points on the panels.]

Chapter 3, p.88, lines -6 and -5 (final 2 lines of Section 3.3.2): Replace with

```
matplot(roller$weight, roller.sim, pch=1,
        ylim=range(roller$depression))
points(roller, pch=16)
```

The text is all there, but the formatting has been scrambled.

p.101, line 2 below the matrix *Pb* in Exercise 13: The stationary distribution values are given wrongly. They are, to 3 decimal places:

Sun	Cloud	Rain
0.429	0.286	0.286

Chapter 6, p.177, line 2:: Replace 0.00070 by 0.0070

p.191: A further bullet point that may be added following line -10 is:

- If observations follow a time sequence, check for sequential correlation in the residuals. The function `acf()` (see Section 9.2) may be used for such a check.

Chapter 7, p.239, lines 8 and 9: Omit
In Figure 7.11, both $f_1(x_1)$ and $f_2(x_2)$ are modeled by spline functions with five degrees of freedom.

Both $f_1(x_1)$ and $f_2(x_2)$ are determined, in the model `ds.gam`, using the roughness penalty method that was described in Subsections 7.5.2 and 7.5.3.

Chapter 10 (Section 10.5), p.332, line 4: The model incorporates a term that allows for normally distributed random variation, additional to the poisson variation at each observation. Technically, this is an example of the use of "observation level random effects".

Chapter 12, Section 12.5, p.407: In *DAAG* version 1.12 and earlier, the names of the columns `longitude` and `latitude` are interchanged. The following code ensures that meaningful names are used for the columns:

```
names(possumsites)[1:2] <- c("long", "lat")
with(possumsites, {
    points(long, lat)
    text(long, lat, row.names(possumsites), pos=c(2,4,2,2,4,2,2))
})
```

First (2010) printing only

Chapter 6, p.187, lines -13 to -12: Delete: "differs from the AIC statistic only by subtraction of n, and by omission of the constant term. It"

line -11: Replace with:

$$C_p = \frac{\text{RSS}}{\sigma^2} + 2p - n$$

Here, σ^2 is replaced by s^2 if the variance has to be estimated. If the variance is known, the C_p statistic differs from AIC only by omission of the constant term and subtraction of n.

p.210, line 3: Starting values can be obtained by fitting the log-linear equation:

```
nihills.lm <- lm(log(time) ~ log(dist) + log(climb.mi), data = nihills)
```

The coefficients are:

```
> coef(nihills.lm)
  (Intercept)     log(dist) log(climb.mi)
      -0.9688        0.6814        0.4658
```

Then suitable starting values for the nonlinear equation are $\hat{\alpha} = \exp(-0.9688) \simeq 0.38$, $\widehat{\beta_1} = 0.68$, $\widehat{\beta_2} = 0.47$

p.210 (Subsection 6.8.4), line -5: Replace "$y = x_1^{\alpha} x_2^{\beta} + \epsilon$" by "$y = \alpha\, {x_1}^{\beta_1}\, {x_2}^{\beta_2} + \epsilon$"

p.211, lines 5-6: Replace with

```
nihills.nls0 <- nls(time~alpha*(dist^beta1)*climb.mi^beta2,
  start=c(alpha=0.38, beta1=0.68, beta2=0.47), data=nihills)
```

Replace lines 11-12 by:

```
alpha   0.3602     0.0601    6.00  7.3e-06
beta1   0.7179     0.0655   10.96  6.6e-10
beta2   0.4948     0.0524    9.45  8.1e-09
```

p.211, line 13: Replace "substantially" by "noticeably".

Chapter 7, p.238 (Subsection 7.6.1), line -2: Replace "`lm`" by "`gam`"

Chapter 9, p.295 (Section 9.2), footnote 5, line 2: Replace "0.0427" by "0.040"

p.298, final computer output in Section 9.2 Using version 2.04 of the *forecast* package, the call to `auto.arima()` fits an `ARIMA(0,1,2)` model, thus:

```
> (mdb2.arima <- with(xbomsoi, auto.arima(mdb3rtRain,
+                       xreg=poly(SOI,2))))
Series: mdb3rtRain
ARIMA(0,1,2)

Call: auto.arima(x = mdb3rtRain, xreg = poly(SOI, 2))

Coefficients:
         ma1    ma2      1      2
      -0.984  0.050  2.899  0.950
s.e.   0.110  0.111  0.510  0.551

sigma^2 estimated as 0.266:  log likelihood = -82.87
AIC = 175.7   AICc = 176.3   BIC = 189.2
```

Chapter 10, p.308, (Section 10.1.2), lines -11 an -10 Replace "$\sqrt{\sigma_L^2 n + \sigma_W^2} = \sqrt{2.37n + 0.578}$" by "$n\sqrt{\sigma_L^2 + \sigma_W^2/n} = n\sqrt{2.37 + 0.578/n}$"

p.350, (Section 10.10), Exercise 5 For assessing the accuracy of the components of variance, consider using `mcmcsamp()` as demonstrated on p.316.

Chapter 15, pp.483-484 (Section 15.5.3) p.483, lines -4 to -1, and p.484, lines 1-2, should be deleted. It repeats p.484, lines -9 to -1, and is out of place on p.483.

Additional note: The function `layer()` (in *latticeExtra*) provides a mechanism for fitting parallel lines that is simpler than creation of a panel function, as describes on lines -21 to -5 (under the heading *A panel function that fits and plots parallel lines*).

The function `layer()` creates a "layer" that can be added to a trellis graphics object. Use the operator "`+`" ("add") to add a layer. For example:

```
## Create graphics object that has the points.
gph <- xyplot(Brainwt ~ Bodywt,  data=primates, xlim=c(0,270))
## Add a second layer that has the labels
gph2 <- gph + layer(panel.text(x,y, labels=rownames(primates), pos=4))
print(gph2)
```

Such "addition" of another layer is often easier than use of a user created panel function.

The function `layer()` allows as arguments, passed via the `...` argument, any sequence of statements that might appear in a panel function. Such statements can refer to panel function arguments, including 'x', 'y' and 'subscripts'. Additionally, statements can refer to names of columns of an optional `data` argument. The new layer can either be overlaid (the default for `layer()`) or underlaid (specify `under=TRUE` or use `layer_()`).

The following adds a new layer to `basic2`, used for Figure 15.4 in Subsection 15.5.2 above, to add separate and parallel lines for the two sports, as in Plate 13:

```
## Create new layer that has the parallel lines
layer2 <- layer(parallel.fit <- fitted(lm(y ~ groups[subscripts] + x)),
                panel.superpose(x, parallel.fit, type = "a", ...))
## Enhanced version of graph, with parallel lines added
print(basic2 + layer2)
```

The function `as.layer()` creates a layer from a trellis graphics object. This can then be "added" in the same way as above.

p.491 (Section 15.6), Table 15.2 Note also the function `opts()`. For example:

```
quickplot(ht, wt, data=ais, facets=. ~ sex) +
  opts(axis.text.x=theme_text(size=14),
       axis.text.y=theme_text(size=10),
       axis.title.y=theme_text(size=14, angle=90),
       legend.text=theme_text(size=14, hjust=0.5),
       legend.title=theme_blank(),
       legend.position=c(.5,.915),
       title="Body Dimensions of Australian Athletes")
```