Errata and Notes

Statistical Computing with R
Chapman & Hall/CRC 2008
Maria L. Rizzo

Errata

p. 30 Last line: $x \geq 0, \nu = 1, 2, \ldots$.

p. 35 In the first two displayed equations: $\lim_{n\to\infty}$.

p. 57 In se calculation (middle of page) `dbeta(Q, 2, 2)` should be `dbeta(Q, 2, 2)^2`.

p. 58 Item 4. $Z_2 = \sqrt{-2 \log U} \sin(2\pi V)$.

p. 121 In step 2: $\overline{g(X)} = \frac{1}{m} \sum_{i=1}^{m} g(X_i)$.

p. 132 In sentence above (5.9): $\hat{\theta}_c = g(X) + c(f(X) - \mu)$.

p. 133 Line 4 from bottom: $100(.2429355 - .003940175)/.2429355$.

p. 158 Table 6.1: $n\widehat{se}$ for $p = 0.95$ and $p = 0.90$ should be different from $p = 1$ case. See note below.

p. 187 Line 1. Remove hat from the first 'se'.

p. 200 Example 7.10. Misplaced right paren.; correction:

```
#normal
print(boot.obj$t0 + qnorm(alpha) * sd(boot.obj$t))
```

p. 204

$$\hat{a} = \frac{\sum_{i=1}^{n} (\overline{\theta_{(.)}} - \theta_{(i)})^3}{6 \left(\sum_{i=1}^{n} (\overline{\theta_{(.)}} - \theta_{(i)})^2 \right)^{3/2}}, \tag{7.11}$$

p. 260 Last paragraph: delete the second sentence "Then an observed sample is generated."

p. 263 Example 9.7.

$$E[X_2|x_1] = \mu_2 + \rho \frac{\sigma_2}{\sigma_1}(x_1 - \mu_1)$$

p. 312 Example 10.15. Correction: The mean vectors are

$$\mu_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad \mu_2 = \begin{bmatrix} 1 \\ 3 \end{bmatrix}, \quad \mu_3 = \begin{bmatrix} 4 \\ -1 \end{bmatrix}.$$

Notes

p. 56 Remark on Example 3.7. Although 6 is an upper bound, it is not the least upper bound. The generator is more efficient if $c = 1.5$, the maximum value of $f(x)/g(x)$ for $0 \le x \le 1$.

p. 57 Code above para. 2: `#See Ch. 2`

p. 71–72 Example 3.16, summary statistics. The `rmvn.eigen` generator takes the covariance matrix `Sigma` as an argument, so in general one may want to display `cov(X)` for comparison with `Sigma` rather than the sample correlation matrix `cor(X)`. (Here our `Sigma` was a correlation matrix.)

p. 158 Table 6.1. With `set.seed(522)` the corresponding table should be:

	Normal		$p = 0.95$		$p = 0.90$	
k	$n\,\widehat{MSE}$	$n\,\widehat{se}$	$n\,\widehat{MSE}$	$n\,\widehat{se}$	$n\,\widehat{MSE}$	$n\,\widehat{se}$
0	0.976	0.140	6.229	0.353	11.485	0.479
1	1.019	0.143	1.954	0.198	4.126	0.287
2	1.009	0.142	1.304	0.161	1.956	0.198
3	1.081	0.147	1.168	0.153	1.578	0.178
4	1.048	0.145	1.280	0.160	1.453	0.170
5	1.103	0.149	1.395	0.167	1.423	0.169
6	1.316	0.162	1.349	0.164	1.574	0.177
7	1.377	0.166	1.503	0.173	1.734	0.186
8	1.382	0.166	1.525	0.175	1.694	0.184
9	1.491	0.172	1.646	0.181	1.843	0.192

p. 178–179 Examples 6.14–6.15. Although mathematically it is not an error, it is unnecessary to subtract the sample means in the expression `tests` of Example 6.14 because the sample means are subtracted in the function `count5test`. Mathematically, if $Z_i = X_i - \overline{X}$, then $\overline{Z} = 0$. The same is true in the expression for `alphahat` in Example 6.15. The `count5test` can be applied without centering the data first, as in Example 6.16.

p. 225–228 Examples 8.4-8.6. The `knnFinder` package with `nn` function for finding nearest neighbors has been withdrawn from CRAN. These examples have been revised using the `ann` function in the `yaImpute` package.

p. 323-333 Example 11.3. In `system.time` the timings are hardware dependent; however, the vectorized version should be faster on all platforms.

pp. 338–339 Example 11.11. The first code snippet will produce a graph similar to Figure 11.3 but with x-axis ranging from about 2 to 8. To produce Figure 11.3 as shown, replace 8 with 15 in `seq(2, 8, .001)`.

p. 341 Example 11.12. The histograms will of course vary slightly from Figure 11.4 because the data is generated at random. According to my notes,

`set.seed(333)` before the first line of code should produce samples matching the histograms as shown on page 341. See the note below concerning `par(ask=TRUE)` to wait for user input before displaying each graph.

p. 342–343 Example 11.13. `set.seed(333)` was set prior to run.

Programming Notes

1. `curve` is convenient for plotting a function. It can replace `lines` in some examples; e.g. in Example 3.2 to add the density curve to the histogram, instead of `lines` we can use:

   ```
   curve(3x^2, add=TRUE)
   ```

2. Displaying a sequence of graphs: `par(ask=TRUE)` has the effect that the user is asked for input before each new figure is drawn. Follow it with `par(ask=FALSE)` to restore to default behavior. Alternately in the R GUI use "Recording" from the graph History menu.

3. `sapply` can be used instead of `apply` in some examples, which eliminates the need for MARGIN and the need for the argument to have a dimension attribute. See e.g. Example 5.4 on page 123. The lines:

   ```
   dim(x) <- length(x)
   p <- apply(x, MARGIN=1, function(x, z) {mean(z < x)}, z=z)
   ```

   can be replaced with either version below:

   ```
   p <- sapply(x, FUN=function(x,z) mean(z<x), z=z)
   p <- sapply(x, function(x) mean(z<x))
   ```

4. A more elegant approach to the comparison of generators in Example 3.19 is to wrap the repeated statements in a function that takes the name of the generator (e.g. rmvn.eigen) as an argument. An example of a function that has a functional argument is `boot`; see Example 7.10 for a typical example.

Acknowledgements Thanks to several readers who sent comments, corrections, and suggestions.