Data Analysis & Graphics Using R – Solutions to Exercises (December 11, 2006)

Preliminaries

```
> library(DAAG)
```

Exercise 2
Draw graphs that show, for degrees of freedom between 1 and 100, the change in the 5% critical value of the t-statistic. Compare a graph on which neither axis is transformed with a graph on which the respective axis scales are proportional to log(t-statistic) and log(degrees of freedom). Which graph gives the more useful visual indication of the change in the 5% critical value of the t-statistic changes with increasing degrees of freedom?

```
> par(mfrow = c(1, 2))
> nu <- 1:100
> plot(nu, qt(0.975, nu), type = "l")
> plot(log(nu), qt(0.975, nu), type = "l", xaxt = "n")
> axis(1, at = log(nu), labels = paste(nu))
> par(mfrow = c(1, 1))
```

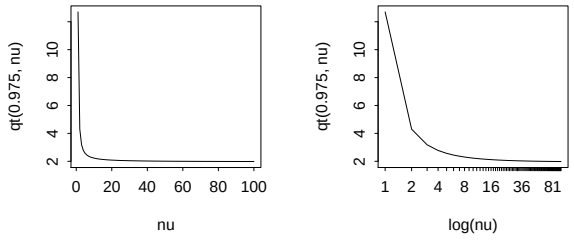

Figure 1: Plot of two-sided 95% critical value for a t-statistic (a) against degrees of freedom and (b) against log(degrees of freedom).

The second graph, because it makes it possible to see the large changes with low degrees of freedom, gives the more useful visual indication.

Exercise 6
Here we generate random normal numbers with a sequential dependence structure.

```
y1 <- rnorm(51)
y <- y1[-1] + y1[-51]
acf(y1)              # acf is `autocorrelation function' (see Ch. 9)
acf(y)
```

Repeat this several times. There should be no consistent pattern in the acf plot for different random samples `y1`. There will be a fairly consistent pattern in the acf plot for `y`, a result of the correlation that is introduced by adding to each value the next value in the sequence.

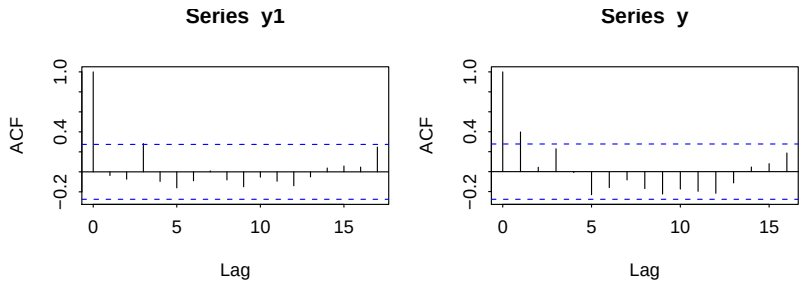

Figure 2: Autocorrelation function (a) for independently and identically distributed normal deviates and (b) for sequentially correlated deviates.

Exercise 7
Create a function that does the calculations in the first two lines of the previous exercise. Put the calculation in a loop that repeats 25 times. Calculate the mean and variance for each vector `y` that is returned. Store the 25 means in the vector `av`, and store the 25 variances in the vector `v`. Calculate the variance of `av`.

```
> corfun <- function(n = 51) {
+     y1 <- rnorm(n)
+     y <- y1[-1] + y1[-n]
+     y
+ }
> av <- numeric(25)
> sdev <- numeric(25)
> for (i in 1:25) {
+     z <- corfun()
+     av[i] <- mean(z)
+     sdev[i] <- sd(z)
+ }
> var(av)

[1] 0.07033
```

Note: The variance of the values that are returned by `corfun()` is $\text{var}(y1_i = \text{var}(y_i + \text{var}(y_{i+1}) = 2$ Thus, compare `var(av)` as calculated above with $\text{var}(y1_i)/50 = 2/50 = 0.04$. As a result of the correlation between successive values, `var(av)` will, on average, be greater than this.

Exercise 10
Use `mosaicplot()` to display the table `rareplants` (Subsection 4.4.1) that was created using code in Footnote 13. Annotate the mosaic plot to draw attention to the results that emerged from the analysis in Subsection 4.4.1.

```
> oldpar <- par(mar = c(3.1, 3.1, 2.6, 1.1))
> rareplants <- matrix(c(37, 190, 94, 23, 59, 23, 10, 141, 28,
+     15, 58, 16), ncol = 3, byrow = T, dimnames = list(c("CC",
+     "CR", "RC", "RR"), c("D", "W", "WD")))
> mosaicplot(t(rareplants), color = TRUE, main = NULL)
```

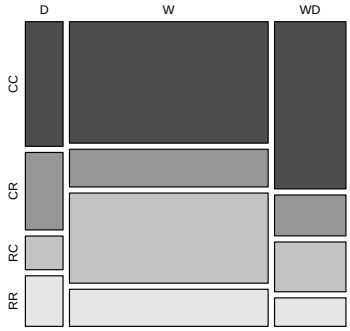

Figure 3: Mosaicplot for the `rareplants` data. Observe that the matrix rareplants has been transposed so that the layout of the graph reflects the layout of the table in Section 4..4.1.

For each color, i.e., row of the table, compare the heights of the rectangles. Large positive residuals in the table of residuals on page 87 correspond to rectangles that are tall relative to other rectangles with the same color, while large negative residuals correspond to rectangles that are short relative to other rectangles with the same color.

Exercise 11
The table `UCBAdmissions` was discussed in Subsection 2.2.1. The following gives a table that adds the 2×2 tables of admission data over all departments:

```
## UCBAdmissions is in the datasets package
## For each combination of margins 1 and 2, calculate the sum
UCBtotal <- apply(UCBAdmissions, c(1,2), sum)
```

What are the names of the two dimensions of this table?

(a) From the table `UCBAdmissions`, create mosaic plots for each faculty separately. (If necessary refer to the code given in the help page for `UCBAdmissions`.)

(b) Compare the information in the table `UCBtotal` with the result from applying the function `mantelhaen.test()` to the table `UCBAdmissions`. Compare the two sets of results, and comment on the difference.

(c) The Mantel–Haenzel test is valid only if the male to female odds ratio for admission is similar across departments. The following code calculates the relevant odds ratios:

```
apply(UCBAdmissions, 3, function(x)
      (x[1,1]*x[2,2])/(x[1,2]*x[2,1]))
```

Is the odds ratio consistent across departments? Which department(s) stand(s) out as different? What is the nature of the difference?

[For further information on the Mantel–Haenszel test, see the help page for `mantelhaen.test`.]

Use `dimnames(UCBAdmissions)[1:2]` to get the names of the first two dimensions, which are `Admit` and `Gender`.

(a) First note the code needed to give a mosaic plot for the totals; the question does not ask for this. There is an excess of males and a deficit of females in the `Admitted` category.

```
> par(mar = c(3.1, 3.1, 2.6, 1.1))
> UCBtotal <- apply(UCBAdmissions, c(1, 2), sum)
> mosaicplot(UCBtotal, col = TRUE)
```

Now obtain the mosaic plots for each department separately.

```
> oldpar <- par(mfrow = c(2, 3), mar = c(3.1, 3.1, 2.6, 1), cex.main = 0.8)
> for (i in 1:6) mosaicplot(UCBAdmissions[, , i], xlab = "Admit",
+     ylab = "Sex", main = paste("Department", LETTERS[i]), color = TRUE)
> par(mfrow = c(1, 1))
> par(oldpar)
```

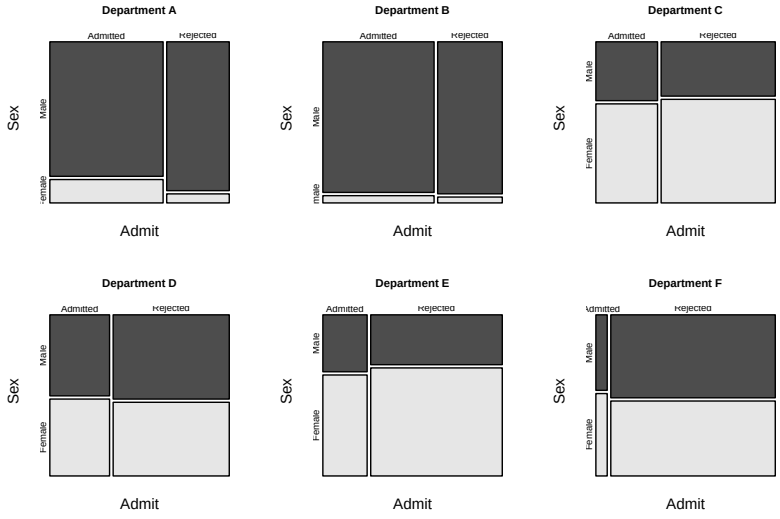

Figure 4: Mosaicplots, for each department separately. The greatest difference in the proportions in the two vertical columns is for Department A.

(b)
```
> apply(UCBAdmissions, 3, function(x) (x[1, 1] * x[2, 2])/(x[1,
+     2] * x[2, 1]))
```

```
     A      B      C      D      E      F
0.3492 0.8025 1.1331 0.9213 1.2216 0.8279
```

The odds ratio (male to female admissions) is much the lowest for Department A.

Exercise 12
*Table 3.3 (Chapter 3) contained fictitious data that illustrate issues that arise in combining data across tables. Table 1 is another such set of fictitious data, designed to demonstrate how biases that go in different directions in the two subtables may cancel in the table to totals.

	Engineering			Sociology			Total	
	Male	Female		Male	Female		Male	Female
Admit	30	20	Admit	10	20	Admit	40	40
Deny	30	10	Deny	5	25	Deny	35	35

Table 1: In these data, biases that go in different directions in the two faculties have canceled in the table of totals.

To enter the data for Table 3.3, type:

```
admissions <- array(c(30,30,10,10,15,5,30,10),
                    dim=c(2,2,2))
```

Similarly for Table 1. The third dimension in each table is faculty, as required for using faculty as a stratification variable for the Mantel–Haenzel test. From the help page for `mantelhaen.test()`, extract and enter the code for the function `woolf()`. Apply the function `woolf()`, followed by the function `mantelhaen.test()`, to the data of each of Tables 3.3 and 1. Explain, in words, the meaning of each of the outputs. Then apply the Mantel–Haenzel test to each of these tables.

```
> admissions <- array(c(30, 30, 10, 10, 15, 5, 30, 10), dim = c(2,
+     2, 2))
> woolf <- function(x) {
+     x <- x + 1/2
+     k <- dim(x)[3]
+     or <- apply(x, 3, function(x) (x[1, 1] * x[2, 2])/(x[1, 2] *
+         x[2, 1]))
+     w <- apply(x, 3, function(x) 1/sum(1/x))
+     1 - pchisq(sum(w * (log(or) - weighted.mean(log(or), w))^2),
+         k - 1)
+ }
> woolf(admissions)

[1] 0.9696
```

The differences from homogeneity (equal odds ratios for males and females in each of the two departments) are well removed from statistical significance.

```
> admissions1 <- array(c(30, 30, 20, 10, 10, 5, 20, 25), dim = c(2,
+     2, 2))
> woolf(admissions1)

[1] 0.04302
```

There is evidence of department-specific biases.

```
> mantelhaen.test(admissions)

        Mantel-Haenszel chi-squared test without continuity correction
```

```
data:  admissions
Mantel-Haenszel X-squared = 0, df = 1, p-value = 1
alternative hypothesis: true common odds ratio is not equal to 1
95 percent confidence interval:
 0.4566 2.1902
sample estimates:
common odds ratio
                1
```

The estimate of the common odds ratio is 1.

```
> mantelhaen.test(admissions1)
```

```
        Mantel-Haenszel chi-squared test with continuity correction

data:  admissions1
Mantel-Haenszel X-squared = 0.0141, df = 1, p-value = 0.9053
alternative hypothesis: true common odds ratio is not equal to 1
95 percent confidence interval:
 0.4481 1.8077
sample estimates:
common odds ratio
              0.9
```

The common odds ratio is given as 0.9. However, because the odds ratio is not homogeneous within each of the two departments, this overall figure can be misleading.

Exercise 14
The function `overlapDensity()` in the *DAAG* package can be used to visualize the unpaired version of the *t*-test. Type in

```
attach(two65)
overlapDensity(ambient, heated)    # Included with our DAAG package
detach(two65)
```

in order to observe estimates of the stretch distributions of the ambient (control) and heated (treatment) elastic bands.

```
> attach(two65)
> overlap.density(ambient, heated)
> detach(two65)
```

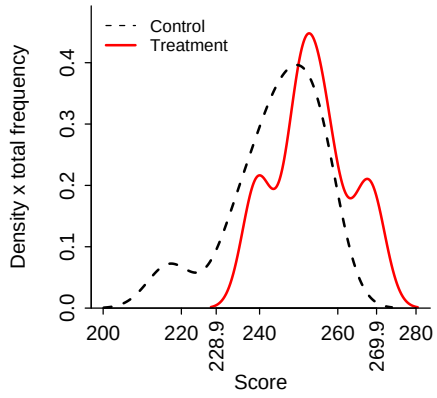

Figure 5: Estimated densities for ambient and heated bands, overlaid.

The densities look quite similar, except that the controls are more widely spread. To get an idea of the sorts of differences that may appear in repeated random sampling, try

```
> par(mfrow = c(2, 2))
> for (i in 1:4) {
+     y1 <- rnorm(10)
+     y2 <- rnorm(11)
+     overlap.density(y1, y2)
+ }
> par(mfrow = c(1, 1))
```

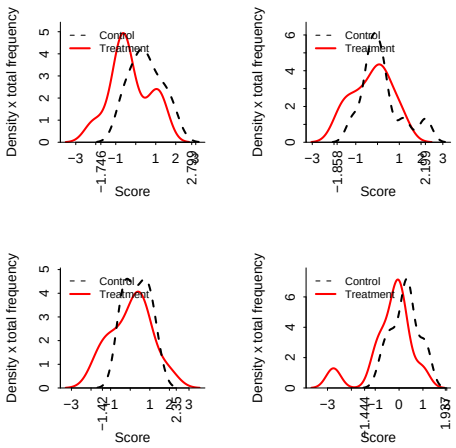

Figure 6: Estimated densities for simulated random normal samples, of the same size as the ambient and heated samples.

*Exercise 15**
*For constructing bootstrap confidence intervals for the correlation coefficient, it is advisable to work with the Fisher z-transformation of the correlation coefficient. The following lines of R code show how to obtain a bootstrap confidence interval for the z-transformed correlation between `chest` and `belly` in the `possum` data frame. The last step of the procedure is to apply the inverse of the z-transformation to the confidence interval to return it to the original scale. Run the following code and compare the resulting interval with the one computed without transformation. Is the z-transform necessary here?

```
z.transform <- function(r) .5*log((1+r)/(1-r))
z.inverse <- function(z) (exp(2*z)-1)/(exp(2*z)+1)
possum.fun <- function(data, indices) {
chest <- data$chest[indices]
belly <- data$belly[indices]
z.transform(cor(belly, chest))}
possum.boot <- boot(possum, possum.fun, R=999)
z.inverse(boot.ci(possum.boot, type="perc")$percent[4:5])
  # See help(bootci.object).  The 4th and 5th elements of
  # the percent list element hold the interval endpoints.
```

```
> library(boot)
```

```
> z.transform <- function(r) 0.5 * log((1 + r)/(1 - r))
> z.inverse <- function(z) (exp(2 * z) - 1)/(exp(2 * z) + 1)
> possum.fun <- function(data, indices) {
+     chest <- data$chest[indices]
+     belly <- data$belly[indices]
+     z.transform(cor(belly, chest))
+ }
> possum.boot <- boot(possum, possum.fun, R = 999)
> z.inverse(boot.ci(possum.boot, type = "perc")$percent[4:5])
```

```
[1] 0.4630 0.7117
```

Exercise 16
The 24 paired observations in the data set `mignonette` were from five pots. The observations are in order of pot, with the numbers 5, 5, 5, 5, 4 in the respective pots. Plot the data in a way that shows the pot to which each point belongs. Also do a plot that shows, by pot, the differences between the two members of each pair. Do the height differences appear to be different for different pots?

```
> mignonette$pot <- rep(1:5, c(5, 5, 5, 5, 4))
> attach(mignonette)
> plot(cross ~ self, type = "n")
> text(cross ~ self, labels = paste(pot), col = pot, lwd = 2)
> detach(mignonette)
```

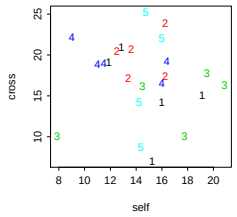

Figure 7: Plot of `cross` versus `self`, with points identified by pot.

```
> "Alternative to above"
> plot(cross ~ self, col = pot, lwd = 2)
> text(cross ~ self, labels = paste(pot), pos = 1, cex = 0.8)
```

```
> library(lattice)
> print(stripplot(pot ~ I(cross - self), data = mignonette, pch = 15))
```

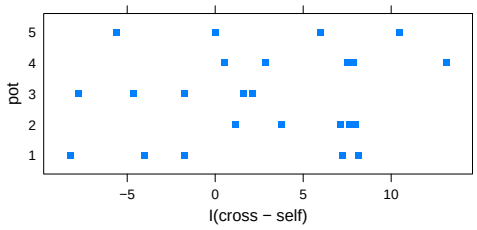

Figure 8: Alternative plot of `cross` versus `self`, with points identified by pot.

x

```
> summary(lm(I(cross - self) ~ factor(pot), data = mignonette))

Call:
lm(formula = I(cross - self) ~ factor(pot), data = mignonette)

Residuals:
   Min     1Q Median     3Q    Max
-8.525 -3.694  0.725  3.386  7.850

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)     0.275      2.418    0.11     0.91
factor(pot)2    5.250      3.419    1.54     0.14
factor(pot)3   -2.350      3.419   -0.69     0.50
factor(pot)4    6.100      3.419    1.78     0.09
factor(pot)5    2.444      3.626    0.67     0.51

Residual standard error: 5.41 on 19 degrees of freedom
Multiple R-Squared: 0.311,        Adjusted R-squared: 0.166
F-statistic: 2.14 on 4 and 19 DF,  p-value: 0.115
```

The evidence for a difference between pots is not convincing. Nevertheless a careful analyst, when checking for a systematic difference between crossed and selfed plants, would allow for a pot effect. (This requires the methods that are discussed in Chapter 10.)

Exercise 18
Use the function `rexp()` to simulate 100 random observations from an exponential distribution with rate 1. Use the bootstrap (with 99999 replications) to estimate the standard error of the median. Repeat several times. Compare with the result that would be obtained using the normal approximation, i.e. $\sqrt{\pi/(2n)}$.

```
> med.fn <- function(x, index) median(x[index])
> x <- rexp(100)
> x.boot <- boot(x, statistic = med.fn, R = 99999)
> x.boot

ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = x, statistic = med.fn, R = 99999)


Bootstrap Statistics :
    original  bias    std. error
t1*   0.6141 0.02902      0.1356
```

We see that the normal approximation is over-estimating the standard error slightly. For large exponential samples (with rate 1), the standard error of the median is $1/\sqrt{n}$.

Exercise 19
Low doses of the insecticide toxaphene may cause weight gain in rats. A sample of 20 rats are given toxaphene in their diet, while a control group of 8 rats are not given toxaphene. Assume further that weight gain among the treated rats is normally distributed with a mean of 60g and standard deviation 30g, while weight gain among the control rats is normally distributed with a mean of 10g and a standard deviation of 50g. Using simulation, compare confidence intervals for the difference in mean weight gain, using the pooled variance estimate and the Welch approximation. Which type of interval is correct more often?
Repeat the simulation experiment under the assumption that the standard deviations are 40g for both samples. Is there a difference between the two types of intervals now? Hint: Is one of the methods giving systematically larger confidence intervals? Which type of interval do you think is best?

```
> "Welch.pooled.comparison" <- function(n1 = 20, n2 = 8, mean1 = 60,
+     mean2 = 10, sd1 = 30, sd2 = 50, nsim = 1000) {
+     Welch.count <- logical(nsim)
+     pooled.count <- logical(nsim)
+     Welch.length <- numeric(nsim)
+     pooled.length <- numeric(nsim)
+     mean.diff <- mean1 - mean2
+     for (i in 1:1000) {
+         x <- rnorm(n1, mean = mean1, sd = sd1)
+         y <- rnorm(n2, mean = mean2, sd = sd2)
+         t1conf.int <- t.test(x, y)$conf.int
+         t2conf.int <- t.test(x, y, var.equal = TRUE)$conf.int
```

```
+         t1correct <- (t1conf.int[1] < mean.diff) & (t1conf.int[2] >
+             mean.diff)
+         t2correct <- (t2conf.int[1] < mean.diff) & (t2conf.int[2] >
+             mean.diff)
+         Welch.count[i] <- t1correct
+         pooled.count[i] <- t2correct
+         Welch.length[i] <- diff(t1conf.int)
+         pooled.length[i] <- diff(t2conf.int)
+     }
+     c(Welch.proportion.correct = mean(Welch.count), pooled.proportion.correct = mean(pooled.c
+         Welch.length.avg = mean(Welch.length), pooled.length.avg = mean(pooled.length))
+ }
> Welch.pooled.comparison()

 Welch.proportion.correct pooled.proportion.correct          Welch.length.avg
                    0.954                      0.885                    82.808
       pooled.length.avg
                   62.009

> Welch.pooled.comparison(sd1 = 40, sd2 = 40)

 Welch.proportion.correct pooled.proportion.correct          Welch.length.avg
                    0.941                      0.940                    70.565
       pooled.length.avg
                   67.770
```

*Exercise 21**
Experiment with the `pair65` example and plot various views of the likelihood function, either as a surface using the `persp()` function or as one-dimensional profiles using the `curve()` function. Is there a single maximizer: Where does it occur?

First, check the mean and the SD.

```
> with(pair65, heated - ambient)

[1] 19  8  4  1  6 10  6 -3  6

> mean(with(pair65, heated - ambient))

[1] 6.333

> sd(with(pair65, heated - ambient))

[1] 6.103
```

Now create and use a function that calculates the likelihood, given mu and sigma

```
> funlik <- function(mu, sigma, x = with(pair65, heated - ambient)) prod(dnorm(x,
+     mu, sigma))
```

Next, calculate a vector of values of mu, and a vector of values of sigma

```
> muval <- seq(from = 2, to = 12, by = 0.5)
> sigval <- seq(from = 1, to = 15, by = 0.5)
```

Now calculate an array of logliklihoods

```
> loglikArray <- function(mu, sigma, d = with(pair65, heated -
+     ambient)) {
+     xx <- matrix(0, nrow = length(mu), ncol = length(sigma))
+     for (j in seq(along = sigma)) for (i in seq(along = mu)) xx[i,
+         j] <- log(funlik(mu[i], sigma[j], d))
+     xx
+ }
> loglik <- loglikArray(mu = muval, sigma = sigval)
```

Now create a perspective plot

```
> persp(x = muval, y = sigval, loglik)
```

A wider range of values of `mu`, and a narrower range of values of `sigma`, seems preferable:

```
> muval <- seq(from = -1, to = 14, by = 0.5)
> sigval <- seq(from = 3, to = 12, by = 0.2)
> loglik <- loglikArray(mu = muval, sigma = sigval)
> persp(x = muval, y = sigval, loglik)
```

Try also

```
> contour(muval, sigval, loglik)
> filled.contour(muval, sigval, loglik)
```

*Exercise 22**
Suppose the mean reaction time to a particular stimulus has been estimated in several previous studies, and it appears to be approximately normally distributed with mean 0.35 seconds with standard deviation 0.1 seconds. On the basis of 10 new observations, the mean reaction time is estimated to be 0.45 seconds with an estimated standard deviation of 0.15 seconds. Based on the sample information, what is the maximum likelihood estimator for the true mean reaction time? What is the Bayes' estimate of the mean reaction time.

Following Section 4.2.2 the posterior density of the mean is normal with mean

$$\frac{n\bar{y} + \mu_0\sigma^2/\sigma_0^2}{n + \sigma^2/\sigma_0^2}$$

and variance

$$\frac{\sigma^2}{n + \sigma^2/\sigma_0^2}$$

where, here

$$\mu_0 = 0.35, \sigma_0 = 0.1, \quad \bar{y} = 0.45, n = 10, \sigma = 0.15$$

Thus the posterior mean and variance of the mean are:

```
> print(c(mean = (10 * 0.45 + 0.35 * 0.15^2/0.1^2)/(10 + 0.15^2/0.1^2)))

  mean
0.4316

> print(c(variance = 0.1^2/(10 + 0.15^2/0.1^2)))

 variance
0.0008163
```

The posterior mean is the Bayes' estimate of the mean.