7.6.2 Appendix: Using R to Find Confidence Intervals
by EV Nordheim, MK Clayton & BS Yandell, October 30, 2003

The `tinterval` command of R is a useful one for finding confidence intervals for the mean when the data are normally distributed with unknown variance. We illustrate the use of this command for the lizard tail length data.

```
> lizard = c(6.2, 6.6, 7.1, 7.4, 7.6, 7.9, 8, 8.3, 8.4, 8.5, 8.6,
+     8.8, 8.8, 9.1, 9.2, 9.4, 9.4, 9.7, 9.9, 10.2, 10.4, 10.8,
+     11.3, 11.9)
```

If we use the `t.test` command listing only the data name, we get a 95% confidence interval for the mean after the significance test.

```
> t.test(lizard)

        One Sample t-test


data:  lizard
t = 30.4769, df = 23, p-value = < 2.2e-16
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 8.292017 9.499649
sample estimates:
mean of x
 8.895833
```

Note here that R reports the interval using more decimal places than was used in Subsection 7.1.2. Because the data were recorded to a single decimal, this extra precision is unnecessary.

The `t.test` command can also be used to find confidence intervals with levels of confidence different from 95%. We do so by specifying the desired level of confidence using the `conf.level` option.

```
> t.test(lizard, conf.level = 0.9)

        One Sample t-test


data:  lizard
t = 30.4769, df = 23, p-value = < 2.2e-16
alternative hypothesis: true mean is not equal to 0
90 percent confidence interval:
 8.395575 9.396092
sample estimates:
mean of x
 8.895833
```

The R command `prop.test` can be used similarly to construct confidence intervals for the normal approximation to the binomial.

```
> prop.test(83, 100, 0.75)

        1-sample proportions test with continuity correction

data:  83 out of 100, null probability 0.75
X-squared = 3, df = 1, p-value = 0.08326
alternative hypothesis: true p is not equal to 0.75
95 percent confidence interval:
 0.7389130 0.8950666
sample estimates:
   p
0.83
```

R does not have a command to find confidence intervals for the mean of normal data when the variance is known. Because this arises rarely in practice, we could skip this. For those interested, the following command lines create a new command `norm.interval` based on material from this chapter. We apply it to the lizard data, assuming we know ahead that the variance is 2.

```
> norm.interval = function(data, variance = var(data), conf.level = 0.95) {
+     z = qnorm((1 - conf.level)/2, lower.tail = FALSE)
+     xbar = mean(data)
+     sdx = sqrt(variance/length(data))
+     c(xbar - z * sdx, xbar + z * sdx)
+ }
> norm.interval(lizard, 2)

[1] 8.330040 9.461626
```

Similar calculations, or a similar function, could be developed for confidence intervals for the variance of a normal distribution. We illustrate this for the variance of the lizard data.

```
> var.interval = function(data, conf.level = 0.95) {
+     df = length(data) - 1
+     chilower = qchisq((1 - conf.level)/2, df)
+     chiupper = qchisq((1 - conf.level)/2, df, lower.tail = FALSE)
+     v = var(data)
+     c(df * v/chiupper, df * v/chilower)
+ }
> var.interval(lizard)

[1] 1.235162 4.023559
```

2

Sampling Confidence Intervals

Here is a way to see how confidence intervals are random. Based on the lizard data, we draw 100 random samples with mean 9 and SD the same as the lizards.

```
> n.draw = 100
> mu = 9
> n = 24
> SD = sd(lizard)
> SD

[1] 1.429953

> draws = matrix(rnorm(n.draw * n, mu, SD), n)
```
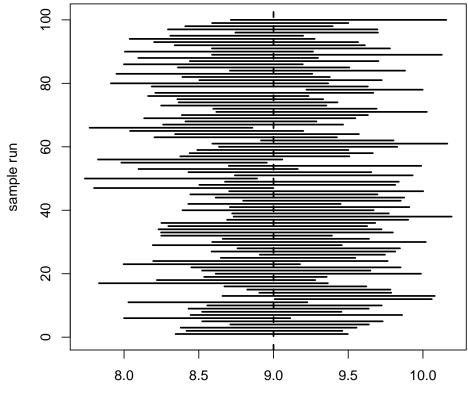
Now we construct 95% confidence intervals for each sample. The first line creates a local command `get.conf.int` to extract the confidence interval (`conf.int`) from the `t.test` command. The second line uses the `apply` command to apply `get.conf.int` to every column of `draws`. Finally, we count the number of confidence intervals that cover $\mu = 9$.

```
> get.conf.int = function(x) t.test(x)$conf.int
> conf.int = apply(draws, 2, get.conf.int)
> sum(conf.int[1, ] <= mu & conf.int[2, ] >= mu)

[1] 94
```

Here is a figure showing the 100 confidence intervals as horizontal lines, with a vertical line at the population mean of 9.

```
> plot(range(conf.int), c(0, 1 + n.draw), type = "n", xlab = "mean tail length",
+       ylab = "sample run")
> for (i in 1:n.draw) lines(conf.int[, i], rep(i, 2), lwd = 2)
> abline(v = 9, lwd = 2, lty = 2)
```