Tutorial on Loops and Functions

September 28, 2007

This tutorial gives a few examples of typical uses of loops in simple data analysis problems.

Problem 1. In the first example, suppose we are given numerous vectors, say 10, and asked to perform a t test on all possible pairs. As output, produce a table of illustrative test components, sorted by p-value.

In practice we'd be given the vectors; here I'll create some sample data. The 10 vectors will be stored in a list. First create an empty list. Then execute a loop that generates a vector of length 20, randomly from normal distributions with varying means. We start from a mean of -2 and increase it by 0.5 at each step. We'll leave the standard deviation at 1 (the default) in each sample.

```
> X <- vector(mode = "list", length = 10)
> m <- -2
> for (i in 1:10) {
+     X[[i]] <- rnorm(20, mean = m)
+     m <- m + 0.5
+ }
> names(X) <- paste("x", 1:10, sep = "")
```

We're given 10 vectors, $x_1,\ldots,x_{10}$ and asked to perform a t test on each pair. If we execute a t test on x_1,x_2, there is no need to repeat the test on x_2,x_1. Order only matters in how certain numbers are reported (in a two-sided test). The following array displays the pairs on which a t test should be run.

$$
\begin{array}{ccccc}
x_1x_2 & x_1x_3 & x_1x_4 & \cdots & x_1x_{10} \\
 & x_2x_3 & x_2x_4 & \cdots & x_2x_{10} \\
 & & \ddots & \cdots & \vdots \\
 & & & x_8x_9 & x_8x_{10} \\
 & & & & x_9x_{10}
\end{array}
$$

That is, for each i, $1 \le i \le 9$, we execute a t test on x_ix_j, for every j, $i+1 \le j \le 10$. We will need a loop inside a loop to run all these tests. Each iteration of the double loop performs one t test. We'll want to store selected components of the test object as rows in a data frame. To keep track of the variables in that particular t test we need to associate it with a descriptive name. As the row names of the data frame we use the name "1-2" for the t test with x_1, x_2, and similarly for other variables. As useful components of the t-test we select `statistic, p.value, estimate`.

To store to t test results we create data frame with 45 entries (the total number of tests) having 0's as the entries and the characters "1" to "45" as the rownames. The counter `l` keeps track of which of the 45 tests we are running and identifies the row in which data should be stored.

```
> testDat <- data.frame(Statistic = numeric(45), P.value = numeric(45),
+     Estimate = numeric(45))
> rownames(testDat) <- as.character(1:45)
> l <- 1
> for (i in 1:9) {
+     for (j in (i + 1):10) {
+         nm <- paste(i, j, sep = "-")
+         tst <- t.test(X[[i]], X[[j]], var.equal = TRUE)
+         rownames(testDat)[l] <- nm
+         testDat[l, 1:3] <- c(tst$stat, tst$p.val, tst$est)
+         l <- l + 1
+     }
+ }
```

Now sort the rows of `testDat` by p-value.

```
> testDat1 <- testDat[order(testDat$P.value), ]
```

The first 10 rows and the last 10 rows of the table are reported on the following page.

Problem 2. Write a function in two variables, `x` and `n`, that successively takes the exponential of `x` `n` times. NOTE: This function gets very large as `n` increases.

```
> iterExp <- function(x, n) {
+     y <- x
+     for (i in 1:n) {
+         y <- exp(y)
+     }
+     y
+ }
```

Samples:

	Statistic	P.value	Estimate
1-10	−15.40	6.33e−18	−1.74
4-10	−14.00	1.41e−16	−0.89
1-9	−12.85	2.09e−15	−1.74
2-10	−12.79	2.44e−15	−0.96
3-10	−11.72	3.44e−14	−0.99
4-9	−11.13	1.60e−13	−0.89
1-8	−10.61	6.50e−13	−1.74
2-9	−10.43	1.04e−12	−0.96
3-9	−9.73	7.22e−12	−0.99
1-7	−9.27	2.69e−11	−1.74

Table 1: First 10 Rows

	Statistic	P.value	Estimate
1-3	−2.55	1.48e−02	−1.74
3-5	−2.51	1.65e−02	−0.99
7-8	−2.35	2.38e−02	0.94
8-10	−2.24	3.11e−02	1.75
6-7	−1.94	6.00e−02	0.36
5-6	−1.92	6.20e−02	−0.21
8-9	−1.16	2.53e−01	1.75
9-10	−1.07	2.90e−01	2.16
3-4	−0.41	6.83e−01	−0.99
2-4	−0.33	7.42e−01	−0.96
2-3	0.11	9.15e−01	−0.96

Table 2: Last 10 Rows

```
> iterExp(5, 2)

[1] 2.851124e+64

> iterExp(2, 5)

[1] Inf
```