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.

```r
> 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.

\[
\begin{array}{cccccc}
  x_1x_2 & x_1x_3 & x_1x_4 & \cdots & x_1x_{10} \\
  x_2x_3 & x_2x_4 & \cdots & x_2x_{10} \\
  \vdots & \vdots & \ddots & \vdots \\
  x_8x_9 & x_8x_{10} & \quad & \quad \\
  x_9x_{10} & \quad & \quad & \quad \\
\end{array}
\]

That is, for each \(i, 1 \leq i \leq 9\), we execute a t test on \(x_i x_j\), for every \(j, i+1 \leq j \leq 10\). 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.

```r
> 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.

```r
> 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.

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

Samples:
<table>
<thead>
<tr>
<th>Statistic</th>
<th>P.value</th>
<th>Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-10</td>
<td>−15.40</td>
<td>6.33e−18</td>
</tr>
<tr>
<td>4-10</td>
<td>−14.00</td>
<td>1.41e−16</td>
</tr>
<tr>
<td>1-9</td>
<td>−12.85</td>
<td>2.09e−15</td>
</tr>
<tr>
<td>2-10</td>
<td>−12.79</td>
<td>2.44e−15</td>
</tr>
<tr>
<td>3-10</td>
<td>−11.72</td>
<td>3.44e−14</td>
</tr>
<tr>
<td>4-9</td>
<td>−11.13</td>
<td>1.60e−13</td>
</tr>
<tr>
<td>1-8</td>
<td>−10.61</td>
<td>6.50e−13</td>
</tr>
<tr>
<td>2-9</td>
<td>−10.43</td>
<td>1.04e−12</td>
</tr>
<tr>
<td>3-9</td>
<td>−9.73</td>
<td>7.22e−12</td>
</tr>
<tr>
<td>1-7</td>
<td>−9.27</td>
<td>2.69e−11</td>
</tr>
</tbody>
</table>

Table 1: First 10 Rows

<table>
<thead>
<tr>
<th>Statistic</th>
<th>P.value</th>
<th>Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-3</td>
<td>−2.55</td>
<td>1.48e−02</td>
</tr>
<tr>
<td>3-5</td>
<td>−2.51</td>
<td>1.65e−02</td>
</tr>
<tr>
<td>7-8</td>
<td>−2.35</td>
<td>2.38e−02</td>
</tr>
<tr>
<td>8-10</td>
<td>−2.24</td>
<td>3.11e−02</td>
</tr>
<tr>
<td>6-7</td>
<td>−1.94</td>
<td>6.00e−02</td>
</tr>
<tr>
<td>5-6</td>
<td>−1.92</td>
<td>6.20e−02</td>
</tr>
<tr>
<td>8-9</td>
<td>−1.16</td>
<td>2.53e−01</td>
</tr>
<tr>
<td>9-10</td>
<td>−1.07</td>
<td>2.90e−01</td>
</tr>
<tr>
<td>3-4</td>
<td>−0.41</td>
<td>6.83e−01</td>
</tr>
<tr>
<td>2-4</td>
<td>−0.33</td>
<td>7.42e−01</td>
</tr>
<tr>
<td>2-3</td>
<td>0.11</td>
<td>9.15e−01</td>
</tr>
</tbody>
</table>

Table 2: Last 10 Rows

```r
> iterExp(5, 2)
[1] 2.851124e+64

> iterExp(2, 5)
[1] Inf
```