Jones, Maillardet, and Robinson 2009. Introduction to Scientific Programming and Simulation Using R.

3.9 Exercises

1. Consider the function $y = f(x)$ defined by

x	≤ 0	$\in (0, 1]$	> 1
$f(x)$	$-x^3$	x^2	$\sqrt{x}$

Supposing that you are given x, write an R expression for y using `if` statements.

Add your expression for y to the following program, then run it to plot the function f.

```
# input
x.values <- seq(-2, 2, by = 0.1)

# for each x calculate y
n <- length(x.values)
y.values <- rep(0, n)
for (i in 1:n) {
    x <- x.values[i]
    # your expression for y goes here
    y.values[i] <- y
}

# output
plot(x.values, y.values, type = "l")
```

Your plot should look like Figure 3.2. Do you think f has a derivative at 1? What about at 0?

We remark that it is possible to vectorise the program above, using the `ifelse` function.

2. Let $h(x, n) = 1 + x + x^2 + \cdots + x^n = \sum_{i=0}^{n} x^i$. Write an R program to calculate $h(x, n)$ using a `for` loop.

3. The function $h(x, n)$ from Exercise 2 is the finite sum of a geometric sequence. It has the following explicit formula, for $x \neq 1$,

$$h(x, n) = \frac{1 - x^{n+1}}{1 - x}.$$

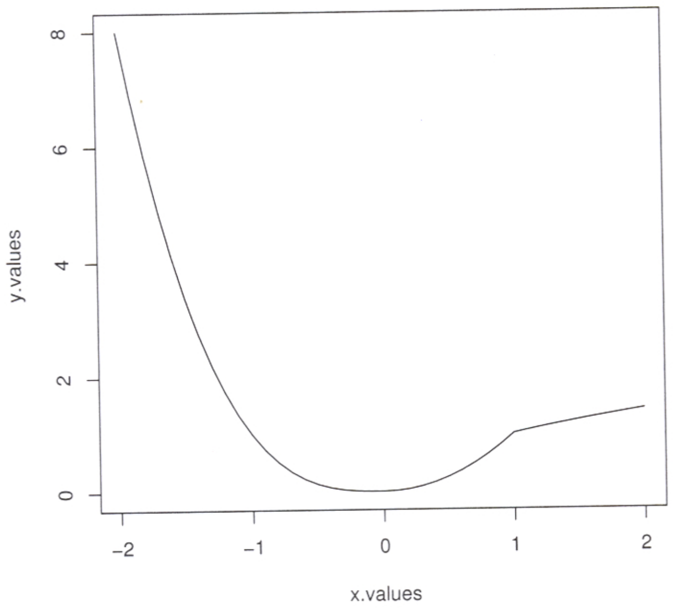

Figure 3.2 *The graph produced by Exercise 1.*

Test your program from Exercise 2 against this formula using the following values

x	n	$h(x,n)$
0.3	55	1.428571
6.6	8	4243335.538178

You should use the computer to calculate the formula rather than doing it yourself.

4. First write a program that achieves the same result as in Exercise 2 but using a `while` loop. Then write a program that does this using vector operations (and no loops).

   If it doesn't already, make sure your program works for the case $x = 1$.

5. To rotate a vector $(x, y)^T$ anticlockwise by θ radians, you premultiply it by the matrix

$$\begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$$

   Write a program in R that does this for you.

6. Given a vector `x`, calculate its geometric mean using both a for loop and vector operations. (The geometric mean of $x_1, \ldots, x_n$ is $\left(\prod_{i=1}^n x_i\right)^{1/n}$.)

You might also like to have a go at calculating the harmonic mean, $\left(\sum_{i=1}^{n} 1/x_i\right)^{-1}$, and then check that if the x_i are all positive, the harmonic mean is always less than or equal to the geometric mean, which is always less than or equal to the arithmetic mean.

7. How would you find the sum of every third element of a vector `x`?

8. How does program `quad2.r` (Exercise 3.2.1) behave if `a2` is 0 and/or `a1` is 0? Using `if` statements, modify `quad2.r` so that it gives sensible answers for all possible (numerical) inputs.

9. Chart the flow through the following two programs.

(a). The first program is a modification of the example from Section 3.6, where x is now an array. You will need to keep track of the value of each element of x, namely $x[1]$, $x[2]$, etc.

```
# threexplus1array.r
x <- 3
for (i in 1:3) {
  show(x)
  if (x[i] %% 2 == 0) {
    x[i+1] <- x[i]/2
  } else {
    x[i+1] <- 3*x[i] + 1
  }
}
show(x)
```

(b). The second program implements the Lotka-Volterra model for a 'predator-prey' system. We suppose that $x(t)$ is the number of prey animals at the start of a year t (rabbits) and $y(t)$ is the number of predators (foxes), then the Lotka-Volterra model is:

$$\begin{aligned} x(t+1) &= x(t) + b_r \cdot x(t) - d_r \cdot x(t) \cdot y(t); \\ y(t+1) &= y(t) + b_f \cdot d_r \cdot x(t) \cdot y(t) - d_f \cdot y(t); \end{aligned}$$

where the parameters are defined by:

b_r is the natural birth rate of rabbits in the absence of predation;
d_r is the death rate per encounter of rabbits due to predation;
d_f is the natural death rate of foxes in the absence of food (rabbits);
b_f is the efficiency of turning predated rabbits into foxes.

```
# program spuRs/resources/scripts/predprey.r
# Lotka-Volterra predator-prey equations
br <- 0.04   # growth rate of rabbits
dr <- 0.0005 # death rate of rabbits due to predation
df <- 0.2    # death rate of foxes
bf <- 0.1    # efficiency of turning predated rabbits into foxes
x <- 4000
y <- 100
while (x > 3900) {
    # cat("x =", x, " y =", y, "\n")
    x.new <- (1+br)*x - dr*x*y
    y.new <- (1-df)*y + bf*dr*x*y
    x <- x.new
    y <- y.new
  }
```

Note that you do not actually need to know anything about the program to be able to chart its flow.

10. Write a program that uses a loop to find the minimum of a vector `x`, without using any predefined functions like `min(...)` or `sort(...)`.

    You will need to define a variable, `x.min` say, in which to keep the smallest value you have yet seen. Start by assigning `x.min <- x[1]` then use a `for` loop to compare `x.min` with `x[2]`, `x[3]`, etc. If/when you find `x[i] < x.min`, update the value of `x.min` accordingly.

11. Write a program to merge two sorted vectors into a single sorted vector.

    Do not use the `sort(x)` function, and try to make your program as efficient as possible. That is, try to minimise the number of operations required to merge the vectors.

12. The game of craps is played as follows. First, you roll two six-sided dice; let x be the sum of the dice on the first roll. If $x = 7$ or 11 you win, otherwise you keep rolling until either you get x again, in which case you also win, or until you get a 7 or 11, in which case you lose.

    Write a program to simulate a game of craps. You can use the following snippet of code to simulate the roll of two (fair) dice:

    ```
    x <- sum(ceiling(6*runif(2)))
    ```

13. Suppose that $(x(t), y(t))$ has polar coordinates $(\sqrt{t}, 2\pi t)$. Plot $(x(t), y(t))$ for $t \in [0, 10]$. Your plot should look like Figure 3.3.

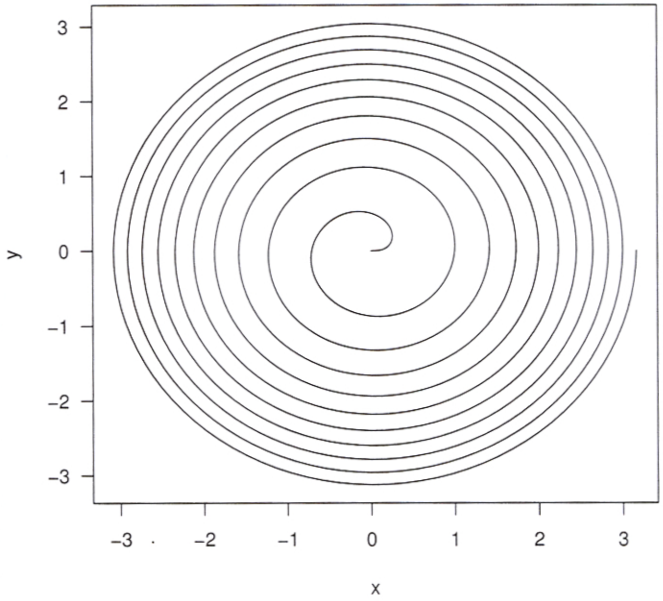

Figure 3.3 *The output from Exercise 13.*