PROGRAMMING EXERCISES FOR R

by

Nastasiya F. Grinberg & Robin J. Reed

Introduction

These exercises were originally developed for a second year undergraduate module at the University of Warwick. The exercises are graded—the first two sheets are intended to get users thinking in terms of vector and matrix operations whilst the later sheets involve writing functions.

Certain important topics are not included. Depending on the response we get to this first version, we have plans for further exercises on classes, graphics programming and the use of more esoteric functions such as `eval`, `deparse`, `do.call`, etc. We welcome comments, suggestions, improved solutions and notifications of errors.

Anyone is free to make a copy or multiple copies of this document or parts of this document for use within their own organisation—although we hope acknowledgement is given. Because we wish to retain the option of including these exercises in a possible text, we reserve overall copyright and so include the following line:

`N.F.Grinberg@gmail.com`
`R.J.Reed@warwick.ac.uk` *Jul 31, 2012*

Contents

EXERCISES

ANSWERS

Exercises 1. Vectors

1. Create the vectors:

(a) $(1, 2, 3, \ldots, 19, 20)$

(b) $(20, 19, \ldots, 2, 1)$

(c) $(1, 2, 3, \ldots, 19, 20, 19, 18, \ldots, 2, 1)$

(d) $(4, 6, 3)$ and assign it to the name `tmp`.

For parts (e), (f) and (g) look at the help for the function `rep`.

(e) $(4, 6, 3,\ 4, 6, 3, \ldots, 4, 6, 3)$ where there are 10 occurrences of 4.

(f) $(4, 6, 3,\ 4, 6, 3, \ldots, 4, 6, 3, 4)$ where there are 11 occurrences of 4, 10 occurrences of 6 and 10 occurrences of 3.

(g) $(4, 4, \ldots, 4,\ 6, 6, \ldots, 6,\ 3, 3, \ldots, 3)$ where there are 10 occurrences of 4, 20 occurrences of 6 and 30 occurrences of 3.

2. Create a vector of the values of $e^x \cos(x)$ at $x = 3, 3.1, 3.2, \ldots, 6$.

3. Create the following vectors:

(a) $(0.1^3 0.2^1, 0.1^6 0.2^4, \ldots, 0.1^{36} 0.2^{34})$ **(b)** $\left(2, \frac{2^2}{2}, \frac{2^3}{3}, \ldots, \frac{2^{25}}{25}\right)$

4. Calculate the following:

(a) $\displaystyle\sum_{i=10}^{100} \left(i^3 + 4i^2\right)$. **(b)** $\displaystyle\sum_{i=1}^{25} \left(\frac{2^i}{i} + \frac{3^i}{i^2}\right)$

5. Use the function `paste` to create the following character vectors of length 30:

(a) `("label 1", "label 2", ....., "label 30")`.
Note that there is a single space between `label` and the number following.

(b) `("fn1", "fn2", ..., "fn30")`.
In this case, there is no space between `fn` and the number following.

6. Execute the following lines which create two vectors of random integers which are chosen with replacement from the integers 0, 1, . . . , 999. Both vectors have length 250.

```
set.seed(50)
xVec <- sample(0:999, 250, replace=T)
yVec <- sample(0:999, 250, replace=T)
```

Suppose $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ denotes the vector `xVec` and $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ denotes the vector `yVec`.

(a) Create the vector $(y_2 - x_1, \ldots, y_n - x_{n-1})$.

(b) Create the vector $\left(\frac{\sin(y_1)}{\cos(x_2)}, \frac{\sin(y_2)}{\cos(x_3)}, \ldots, \frac{\sin(y_{n-1})}{\cos(x_n)}\right)$.

(c) Create the vector $(x_1 + 2x_2 - x_3, x_2 + 2x_3 - x_4, \ldots, x_{n-2} + 2x_{n-1} - x_n)$.

(d) Calculate $\displaystyle\sum_{i=1}^{n-1} \frac{e^{-x_{i+1}}}{x_i + 10}$.

7. This question uses the vectors `xVec` and `yVec` created in the previous question and the functions `sort`, `order`, `mean`, `sqrt`, `sum` and `abs`.

(a) Pick out the values in `yVec` which are > 600.

(b) What are the index positions in `yVec` of the values which are > 600?

(c) What are the values in `xVec` which correspond to the values in `yVec` which are > 600? (By correspond, we mean at the same index positions.)

(d) Create the vector $(|x_1 - \bar{\mathbf{x}}|^{1/2}, |x_2 - \bar{\mathbf{x}}|^{1/2}, \dots, |x_n - \bar{\mathbf{x}}|^{1/2})$ where $\bar{\mathbf{x}}$ denotes the mean of the vector $\mathbf{x} = (x_1, x_2, \dots, x_n)$.

(e) How many values in `yVec` are within 200 of the maximum value of the terms in `yVec`?

(f) How many numbers in `xVec` are divisible by 2? (Note that the modulo operator is denoted `%%`.)

(g) Sort the numbers in the vector `xVec` in the order of increasing values in `yVec`.

(h) Pick out the elements in `yVec` at index positions $1, 4, 7, 10, 13, \dots$.

8. By using the function `cumprod` or otherwise, calculate

$$1 + \frac{2}{3} + \left(\frac{2}{3}\frac{4}{5}\right) + \left(\frac{2}{3}\frac{4}{5}\frac{6}{7}\right) + \cdots + \left(\frac{2}{3}\frac{4}{5}\cdots\frac{38}{39}\right)$$

Exercises 2. Matrices

1. Suppose

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 3 \\ 5 & 2 & 6 \\ -2 & -1 & -3 \end{bmatrix}$$

(a) Check that $\mathbf{A}^3 = \mathbf{0}$ where $\mathbf{0}$ is a 3×3 matrix with every entry equal to 0.

(b) Replace the third column of $\mathbf{A}$ by the sum of the second and third columns.

2. Create the following matrix $\mathbf{B}$ with 15 rows:

$$\mathbf{B} = \begin{bmatrix} 10 & -10 & 10 \\ 10 & -10 & 10 \\ \cdots & \cdots & \cdots \\ 10 & -10 & 10 \end{bmatrix}$$

Calculate the 3×3 matrix $\mathbf{B}^{\mathrm{T}}\mathbf{B}$. (Look at the help for `crossprod`.)

3. Create a 6×6 matrix `matE` with every entry equal to 0. Check what the functions `row` and `col` return when applied to `matE`. Hence create the 6×6 matrix:

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

4. Look at the help for the function `outer`. Hence create the following patterned matrix:

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 6 \\ 3 & 4 & 5 & 6 & 7 \\ 4 & 5 & 6 & 7 & 8 \end{pmatrix}$$

5. Create the following patterned matrices. In each case, your solution should make use of the special form of the matrix—this means that the solution should easily generalise to creating a larger matrix with the same structure and should not involve typing in all the entries in the matrix.

(a) $$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 & 0 \\ 2 & 3 & 4 & 0 & 1 \\ 3 & 4 & 0 & 1 & 2 \\ 4 & 0 & 1 & 2 & 3 \end{pmatrix}$$

(b) $$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 8 & 9 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 9 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{pmatrix}$$

(c) $$\begin{pmatrix} 0 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 \\ 1 & 0 & 8 & 7 & 6 & 5 & 4 & 3 & 2 \\ 2 & 1 & 0 & 8 & 7 & 6 & 5 & 4 & 3 \\ 3 & 2 & 1 & 0 & 8 & 7 & 6 & 5 & 4 \\ 4 & 3 & 2 & 1 & 0 & 8 & 7 & 6 & 5 \\ 5 & 4 & 3 & 2 & 1 & 0 & 8 & 7 & 6 \\ 6 & 5 & 4 & 3 & 2 & 1 & 0 & 8 & 7 \\ 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 & 8 \\ 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \end{pmatrix}$$

6. Solve the following system of linear equations in five unknowns

$$\begin{aligned} x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 &= 7 \\ 2x_1 + x_2 + 2x_3 + 3x_4 + 4x_5 &= -1 \\ 3x_1 + 2x_2 + x_3 + 2x_4 + 3x_5 &= -3 \\ 4x_1 + 3x_2 + 2x_3 + x_4 + 2x_5 &= 5 \\ 5x_1 + 4x_2 + 3x_3 + 2x_4 + x_5 &= 17 \end{aligned}$$

by considering an appropriate matrix equation $\mathbf{Ax} = \mathbf{y}$.

Make use of the special form of the matrix $\mathbf{A}$. The method used for the solution should easily generalise to a larger set of equations where the matrix $\mathbf{A}$ has the same structure; hence the solution should not involve typing in every number of $\mathbf{A}$.

7. Create a 6×10 matrix of random integers chosen from 1, 2,. . . , 10 by executing the following two lines of code:

```
set.seed(75)
aMat <- matrix( sample(10, size=60, replace=T), nr=6)
```

(a) Find the number of entries in each row which are greater than 4.
(b) Which rows contain exactly two occurrences of the number seven?
(c) Find those pairs of columns whose total (over both columns) is greater than 75. The answer should be a matrix with two columns; so, for example, the row $(1, 2)$ in the output matrix means that the sum of columns 1 and 2 in the original matrix is greater than 75. Repeating a column is permitted; so, for example, the final output matrix could contain the rows $(1, 2)$, $(2, 1)$ and $(2, 2)$.
What if repetitions are not permitted? Then, only $(1, 2)$ from $(1, 2)$, $(2, 1)$ and $(2, 2)$ would be permitted.

8. Calculate

(a) $\displaystyle\sum_{i=1}^{20}\sum_{j=1}^{5}\frac{i^4}{(3+j)}$ **(b)** (Hard) $\displaystyle\sum_{i=1}^{20}\sum_{j=1}^{5}\frac{i^4}{(3+ij)}$ **(c)** (Even harder!) $\displaystyle\sum_{i=1}^{10}\sum_{j=1}^{i}\frac{i^4}{(3+ij)}$

Exercises 3. Simple Functions

1. **(a)** Write functions `tmpFn1` and `tmpFn2` such that if `xVec` is the vector $(x_1, x_2, \ldots, x_n)$, then `tmpFn1(xVec)` returns the vector $\left(x_1, x_2^2, \ldots, x_n^n\right)$ and `tmpFn2(xVec)` returns the vector $\left(x_1, \frac{x_2^2}{2}, \ldots, \frac{x_n^n}{n}\right)$.

(b) Now write a function `tmpFn3` which takes 2 arguments `x` and `n` where `x` is a single number and `n` is a strictly positive integer. The function should return the value of

$$1 + \frac{x}{1} + \frac{x^2}{2} + \frac{x^3}{3} + \cdots + \frac{x^n}{n}$$

2. Write a function `tmpFn(xVec)` such that if `xVec` is the vector $\mathbf{x} = (x_1, \ldots, x_n)$ then `tmpFn(xVec)` returns the vector of moving averages:

$$\frac{x_1 + x_2 + x_3}{3}, \quad \frac{x_2 + x_3 + x_4}{3}, \quad \ldots, \quad \frac{x_{n-2} + x_{n-1} + x_n}{3}$$

Try out your function; for example, try `tmpFn( c(1:5,6:1) )`.

3. Consider the continuous function

$$f(x) = \begin{cases} x^2 + 2x + 3 & \text{if } x < 0 \\ x + 3 & \text{if } 0 \le x < 2 \\ x^2 + 4x - 7 & \text{if } 2 \le x. \end{cases}$$

Write a function `tmpFn` which takes a single argument `xVec`. The function should return the vector of values of the function $f(x)$ evaluated at the values in `xVec`.
Hence plot the function $f(x)$ for $-3 < x < 3$.

4. Write a function which takes a single argument which is a matrix. The function should return a matrix which is the same as the function argument but every odd number is doubled.

Hence the result of using the function on the matrix

$$\begin{bmatrix} 1 & 1 & 3 \\ 5 & 2 & 6 \\ -2 & -1 & -3 \end{bmatrix}$$

should be:

$$\begin{bmatrix} 2 & 2 & 6 \\ 10 & 2 & 6 \\ -2 & -2 & -6 \end{bmatrix}$$

Hint: First try this for a specific matrix on the Command Line.

5. Write a function which takes 2 arguments `n` and `k` which are positive integers. It should return the $n \times n$ matrix:

$$\begin{bmatrix} k & 1 & 0 & 0 & \cdots & 0 & 0 \\ 1 & k & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & k & 1 & \cdots & 0 & 0 \\ 0 & 0 & 1 & k & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & \cdots & k & 1 \\ 0 & 0 & 0 & 0 & \cdots & 1 & k \end{bmatrix}$$

Hint: First try to do it for a specific case such as $n = 5$ and $k = 2$ on the Command Line.

6. Suppose an angle α is given as a positive real number of degrees.

If $0 \le \alpha < 90$ then it is quadrant 1. If $90 \le \alpha < 180$ then it is quadrant 2.
If $180 \le \alpha < 270$ then it is quadrant 3. If $270 \le \alpha < 360$ then it is quadrant 4.
If $360 \le \alpha < 450$ then it is quadrant 1. And so on.

Write a function `quadrant(alpha)` which returns the quadrant of the angle α.

7. **(a)** Zeller's congruence is the formula:

$$f = \big([2.6m - 0.2] + k + y + [y/4] + [c/4] - 2c\big) \bmod 7$$

where $[x]$ denotes the integer part of x; for example $[7.5] = 7$.
Zeller's congruence returns the day of the week f given:
k = the day of the month,
y = the year in the century
c = the first 2 digits of the year (the century number)
m = the month number (where January is month 11 of the preceding year, February is month 12 of the preceding year, March is month 1, etc.)
For example, the date 21/07/1963 has $m = 5$, $k = 21$, $c = 19$, $y = 63$; whilst the date 21/2/1963 has $m = 12$, $k = 21$, $c = 19$ and $y = 62$.

Write a function `weekday(day,month,year)` which returns the day of the week when given the numerical inputs of the day, month and year.
Note that the value of 1 for f denotes Sunday, 2 denotes Monday, etc.

(b) Does your function work if the input parameters `day`, `month` and `year` are vectors with the same length and with valid entries?

8. **(a)** Suppose $x_0 = 1$ and $x_1 = 2$ and

$$x_j = x_{j-1} + \frac{2}{x_{j-1}} \qquad \text{for } j = 1, 2, \ldots.$$

Write a function `testLoop` which takes the single argument n and returns the first $n - 1$ values of the sequence $\{x_j\}_{j\geq 0}$: that means the values of $x_0, x_1, x_2, \ldots, x_{n-2}$.

(b) Now write a function `testLoop2` which takes a single argument `yVec` which is a vector. The function should return

$$\sum_{j=1}^{n} e^j$$

where n is the length of `yVec`.

9. *Solution of the difference equation $x_n = rx_{n-1}(1 - x_{n-1})$, with starting value x_1.*

(a) Write a function `quadmap( start, rho, niter )` which returns the vector $(x_1, \ldots, x_n)$ where $x_k = rx_{k-1}(1 - x_{k-1})$ and

`niter` denotes n,
`start` denotes x_1, and
`rho` denotes r.

Try out the function you have written:
- for $r = 2$ and $0 < x_1 < 1$ you should get $x_n \to 0.5$ as $n \to \infty$.
- try `tmp <- quadmap(start=0.95, rho=2.99, niter=500)`

Now switch back to the Commands window and type:

```
plot(tmp, type="l")
```

Also try the plot `plot(tmp[300:500], type="l")`

(b) Now write a function which determines the number of iterations needed to get $| x_n - x_{n-1} | < 0.02$. So this function has only 2 arguments: `start` and `rho`. (For `start=0.95` and `rho=2.99`, the answer is 84.)

10. **(a)** Given a vector $(x_1, \ldots, x_n)$, the sample autocorrelation of lag k is defined to be

$$r_k = \frac{\sum_{i=k+1}^{n}(x_i - \bar{x})(x_{i-k} - \bar{x})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Thus

$$r_1 = \frac{\sum_{i=2}^{n}(x_i - \bar{x})(x_{i-1} - \bar{x})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} = \frac{(x_2 - \bar{x})(x_1 - \bar{x}) + \cdots + (x_n - \bar{x})(x_{n-1} - \bar{x})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Write a function `tmpFn(xVec)` which takes a single argument `xVec` which is a vector and returns a

`list` of two values: r_1 and r_2.
In particular, find r_1 and r_2 for the vector $(2, 5, 8, \ldots, 53, 56)$.

(b) (Harder.) Generalise the function so that it takes two arguments: the vector `xVec` and an integer `k` which lies between 1 and $n-1$ where n is the length of `xVec`.
The function should return a vector of the values $(r_0 = 1, r_1, \ldots, r_k)$.

If you used a loop to answer part (b), then you need to be aware that much, much better solutions are possible—see exercises 4. (Hint: `sapply`.)

Exercises 4. Harder functions

1. Suppose we are given `xVec` which represents the vector $(x_1, \ldots, x_n)$ and `yVec` which represents the vector $(y_1, \ldots, y_m)$. Suppose further that `zVec` represents the vector $(z_1, \ldots, z_n)$ where

z_1 = number$(y_j < x_1)$ $\qquad$ z_2 = number$(y_j < x_2)$ $\qquad \ldots \qquad$ z_n = number$(y_j < x_n)$

Formally, if I denotes the indicator function, then

$$z_k = \sum_{j=1}^{m} I(y_j < x_k) \qquad \text{for } k = 1,\ 2,\ \ldots,\ n$$

(a) By using the function `outer`, write a function which takes the arguments `xVec` and `yVec` and returns the vector `zVec`.

(b) Repeat part (a) but use `sapply` instead of `outer`.,

(b) Now repeat part (a) but use `vapply` instead of `outer` or `sapply`.

(d) Investigate how the functions when one or both of the arguments is a vector with length 0. What if either or both arguments are matrices? *Always check your functions return sensible values whatever the values of the input parameters. Inserting checks on the values of input parameters is often necessary.*

(e) Investigate the relative speed of your solutions by using `system.time`.

2. **(a)** Suppose `matA` is a matrix containing some occurrences of `NA`. Pick out the submatrix which consists of all columns which contain no occurrence of `NA`. So the objective is to write a function which takes a single argument which can be assumed to be a matrix and returns a matrix.

(b) Now write a function which takes a single argument which can be assumed to be a matrix and returns the submatrix which is obtained by deleting every row and column from the input matrix which contains an `NA`.

3. *The empirical copula.*
Suppose we are given two data vectors $(x_1, \ldots, x_n)$ and $(y_1, \ldots, y_n)$. Then the empirical copula is the function C: $[0,1] \times [0,1] \to [0,1]$ defined by

$$C(u,v) = \frac{1}{n} \sum_{j=1}^{n} I\left(\frac{r_i}{n+1} \leq u, \frac{s_i}{n+1} \leq v \right)$$

where $(r_1, \ldots, r_n)$ denotes the vector of ranks of $(x_1, \ldots, x_n)$ and $(s_1, \ldots, s_n)$ denotes the vector of ranks of $(y_1, \ldots, y_n)$. For example, if $(x_1, x_2, x_3, x_4) = (7, 3, 1, 4)$ then $(r_1, r_2, r_3, r_4) = (4, 2, 1, 3)$, because $x_1 = 7$ is the largest and hence $r_1 = 4$; $x_2 = 3$ which is the second largest when the x-values are ranked in increasing size and hence $r_2 = 2$, etc. The supplied function `rank` returns the vector of ranks of the input vector.

(a) Write a function called `empCopula` which takes four arguments: `u`, `v` `xVec` and `yVec`. You can assume that the values of `u` and `v` lie in $[0,1]$ and `xVec` and `yVec` are numeric vectors with equal non-zero lengths.

(b) Of course, users of `R` legitimately expect that all functions will work on vectors. In particular, users will wish to plot the empirical copula and this involves calculating its value at many points (u, v). Does the function you gave as the answer to part (a) work if `u` and `v` are numeric vectors with the same length and with all values lying in $[0,1]$? If not, can you write a function which does cope with that situation?

4. Experiment with different ways of defining a function which calculates the following double sum for any value of n.

$$f(n) = \sum_{i=1}^{n} \sum_{s=1}^{r} \frac{s^2}{10 + 4r^3}$$

For each function you create, time how quickly it executes by using the function `system.time`.

(a) First use a loop within a loop.

(b) Write a function `funB` which uses the functions `row` and `col` to construct a matrix with suitable entries so that the sum of the matrix gives the required answer.

(c) Write a function `funC` which uses the function `outer` to construct a matrix with suitable entries so that the sum of the matrix gives the required answer.

(d) Create a function which takes a single argument `r` and calculates

$$\sum_{s=1}^{r} \frac{s^2}{10+4r^3}$$

Then write a function `funD` which uses `sapply` to calculate the double sum.
Note that `sapply` is just a combination of `unlist` and `lapply`. Is there any increase in speed gained by using this information (`funE`)?

(e) Write a function which takes two arguments `r` and `s` and calculates

$$\frac{I(s \le r)s^2}{10+4r^3}$$

where I denotes the indicator function. Then write a function `funF` which calculates the double sum by using `mapply` to calculate all the individual terms.

Which is the fastest function?

5. *The waiting time of the n^{th} customer in a single server queue.* Suppose customers labelled $C_0, C_1, \ldots, C_n$ arrive at times $\tau = 0, \tau_1, \ldots, \tau_n$ for service by a single server. The interarrival times $A_1 = \tau_1 - \tau_0, \ldots, A_n = \tau_n - \tau_{n-1}$ are independent and identically distributed random variables with the exponential density

$$\lambda_a e^{-\lambda_a x} \qquad \text{for } x \ge 0.$$

The service times $S_0, S_1, \ldots, S_n$ are independent and identically distributed random variables which are also independent of the interarrival times with the exponential density

$$\lambda_s e^{-\lambda_s x} \qquad \text{for } x \ge 0.$$

Let W_j denote the waiting time of customer C_j. Hence customer C_j leaves at time $\tau_j + W_j + S_j$. If this time is greater than τ_{j+1} then the next customer, C_{j+1} must wait for the time $\tau_j + W_j + S_j - \tau_{j+1}$. Hence we have the recurrent relation

$$W_0 = 0$$
$$W_{j+1} = \max\{0, W_j + S_j - A_{j+1}\} \quad \text{for } j = 0, 1, \ldots, n-1$$

(a) Write a function `queue(n, aRate, sRate)` which simulates one outcome of W_n where `aRate` denotes λ_a and `sRate` denotes λ_s. Try out your function on an example such as `queue(50,2,2)`

(b) Now suppose we wish to simulate many outcomes of W_n in order to estimate some feature of the distribution of W_n. Write a function which uses a loop to repeatedly call the function in part (a) to calculate W_n. Then write another function which uses `sapply` (or `replicate`) to call the function created in part (a). Compare the speed of the two functions by using `system.time`.

(c) Can we do any better? Try writing a vectorised form of the basic recurrence relation $W_{j+1} = \max\{0, W_j + S_j - A_{j+1}\}$ where W_j is treated as a vector. Compare the speed of this new function with the two answers to the previous part.

6. *A random walk.* A symmetric simple random walk starting at the origin is defined as follows. Suppose $X_1, X_2, \ldots$ are independent and identically distributed random variables with the distribution

$$\begin{cases} +1 & \text{with probability } 1/2 \\ -1 & \text{with probability } 1/2 \end{cases}$$

Define the sequence $\{S_n\}_{n\ge 0}$ by

$$S_0 = 0$$
$$S_n = S_{n-1} + X_n \quad \text{for } n = 1, 2, \ldots$$

Then $\{S_n\}_{n\ge 0}$ is a symmetric simple random walk starting at the origin. Note that the position of the walk at time n is just the sum of the previous n steps: $S_n = X_1 + \cdots + X_n$.

(a) Write a function `rwalk(n)` which takes a single argument `n` and returns a vector which is a realisation of $(S_0, S_1, \ldots, S_n)$, the first `n` positions of a symmetric random walk starting at the origin.
Hint: the code `sample( c(-1,1), n, replace=TRUE, prob=c(0.5,0.5) )` simulates n steps.

(b) Now write a function `rwalkPos(n)` which simulates one occurrence of the walk which lasts for a length of time n and then returns the length of time the walk spends above the x-axis.
(Note that a walk with length 6 and vertices at 0, 1, 0, -1, 0, 1, 0 spends 4 units of time above the axis and 2 units of time below the axis.)

(c) Now suppose we wish to investigate the distribution of the time the walk spends above the x-axis. This means we need a large number of replications of `rwalkPos(n)`.
Write two functions: `rwalkPos1(nReps,n)` which uses a loop and `rwalkPos2(nReps,n)` which uses `replicate` or `sapply`. Compare the execution times of these two functions.

(d) In the previous question on the waiting time in a queue, a very substantial increase was obtained by using a vector approach. Is that possible in this case?

Exercises 5. Data frame, list, array and time series

1. *Time Series.* The code

```
ts(datVec, start=c(1960,3), frequency=12)
```

creates a time series with monthly observations (`frequency=12`), with first observation in March 1960 (`start=c(1960,3)`) and with values specified in the vector `datVec`.

Suppose $z_1, z_2, \ldots, z_n$ is a time series. Then we define the exponentially weighted moving average of this time series as follows: select a starting value m_0 and select a discount factor δ. Then calculate m_1, m_2, $\ldots$, m_n recursively as follows: for $t = 1, 2, \ldots, n$

$$e_t = z_t - m_{t-1}$$
$$m_t = m_{t-1} + (1 - \delta)e_t$$

(a) Write a function `tsEwma(tsDat, m0=0, delta=0.7)` where `tsDat` is a time series, `m0` is the starting value m_0 and `delta` is δ. The function should return $m_1, m_2, \ldots, m_n$ in the form of a time series.

(b) In general, *looping over named objects is much slower than looping over objects which do not have names*. This principle also applies to time series: looping over a vector is much quicker than looping over a time series. Use this observation to improve the execution speed of your function which should still return a time series. Investigate the difference in speed between the functions in parts (a) and (b) by using the function `system.time`.

2. **(a)** Write a function, called `myListFn`, which takes a single argument n and implements the following algorithm:

   1. Simulate n independent numbers, denoted $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, from the $N(0, 1)$ distribution.
   2. Calculate the mean $\bar{\mathbf{x}} = \sum_{j=1}^n x_j \Big/ n$.
   3. If $\bar{\mathbf{x}} \geq 0$, then simulate n independent numbers, denoted $\mathbf{y} = (y_1, y_2, \ldots, y_n)$, from the exponential density with mean $\bar{\mathbf{x}}$.
      If $\bar{\mathbf{x}} < 0$, then simulate n independent numbers, denoted $\mathbf{z} = (z_1, z_2, \ldots, z_n)$, from the exponential density with mean $-\bar{\mathbf{x}}$. Set $\mathbf{y} = (y_1, y_2, \ldots, y_n) = -\mathbf{z}$.
   4. Calculate k which is the number of j with $|y_j| > |x_j|$.
   5. Return the list of $\mathbf{x}$, $\mathbf{y}$ and k with names `xVec`, `yVec` and `count` respectively.

(b) Execute the following lines and check the format of the answers:

```
lapply( rep(10,4), myListFn )
sapply( rep(10,4), myListFn )
```

Note that `sapply` is effectively `lapply` followed by `simplify2array`.

If `myListFn` has no arguments, then similar results can be obtained with `replicate(4, myListFn())` and `replicate(4, myListFn(), simplify=F)`.

Now for a simulation study. Use `lapply` to call the function `myListFn` with $n = 10$ for 1,000 times. So the output consists of
10,000 values for x denoted $\{x_{i,j} : i = 1, 2, \ldots, 1{,}000; j = 1, 2, \ldots, 10\}$;
10,000 values for y denoted $\{y_{i,j} : i = 1, 2, \ldots, 1{,}000; j = 1, 2, \ldots, 10\}$;
and 1,000 values for n denoted $n_1, n_2, \ldots, n_{1000}$.
Denote the output by `myList`. This output is used in the remaining parts of this question.

(c) Extract all the vectors with the name `yVec`. The result should be a list of 1000 vectors.

(d) Extract all the vectors with the name `yVec`. The result should be a 10×1000 matrix with one column for each of the vectors `yVec`.

(e) Create a list which is identical to `myList` but all the components called `count` have been removed.

(f) Pick out those lists in `myList` which are such that `count` is greater than 2.

3. *This question uses the list* `myList` *created in the previous question.*

(a) Calculate the vector which consists of the values of
$$\frac{x_{i1} + 2x_{i2} + \cdots + 10x_{i,10}}{y_{i1} + 2y_{i2} + \cdots + 10y_{i,10}}$$
for $i = 1, 2, \ldots, 1{,}000$.

(b) Calculate the $1{,}000 \times 10$ matrix with entries $x_{ij} - y_{ij}$ for $i = 1, 2, \ldots, 1{,}000$ and $j = 1, 2, \ldots, 10$.

(c) Find the value of
$$\frac{\sum_{i=1}^{1000} i x_{i,2}}{\sum_{i=1}^{1000} n_i y_{i,2}} = \frac{x_{12} + 2x_{22} + \cdots + 1000x_{1000,2}}{n_1 y_{12} + n_2 y_{22} + \cdots + n_{1000} y_{1000,2}}$$

4. *Arrays.* In order to test the functions in this question, you will need an `array`. We can create a three-dimensional test `array` as follows:

```
testArray <- array( sample( 1:60, 60, replace=F), dim=c(5,4,3) )
```

The above line creates a $5 \times 4 \times 3$ array of integers which can be represented in mathematics by:
$$\{x_{i,j,k} : i = 1, 2, \ldots, 5;\ j = 1, 2, 3, 4;\ k = 1, 2, 3\}$$

Note that `apply(testArray, 3, tmpFn)` means that the index k is retained in the answer and the function `tmpFn` is applied to the 3 matrices:
$\{x_{i,j,1} : 1 \le i \le 5; 1 \le j \le 4\}$, $\{x_{i,j,2} : 1 \le i \le 5; 1 \le j \le 4\}$ and $\{x_{i,j,3} : 1 \le i \le 5; 1 \le j \le 4\}$.

Similarly `apply(testArray, c(1,3), tmpFn)` means that indices i and k are retained in the answer and the function `tmpFn` is applied to 15 vectors: $\{x_{1,j,1} : 1 \le j \le 4\}$, $\{x_{1,j,2} : 1 \le j \le 4\}$, etc.
The expression `apply(testArray, c(3,1), tmpFn)` does the same calculation but the format of the answer is different: when using `apply` in this manner, it is always worth writing a small example in order to check that the format of the output of `apply` is as you expect.

(a) Write a function `testFn` which takes a single argument which is a 3-dimensional array. If this array is denoted $\{x_{i,j,k} : i = 1, 2, \ldots, d_1;\ j = 1, 2, \ldots, d_2;\ k = 1, 2, \ldots, d_3\}$, then the function `testFn` returns a `list` of the $d_1 \times d_2 \times d_3$ matrix $\{w_{i,j,k}\}$ and the $d_2 \times d_3$ matrix $\{z_{j,k}\}$ where
$$w_{i,j,k} = x_{i,j,k} - \min_{i=1}^{d_1} x_{i,j,k} \quad \text{and} \quad z_{j,k} = \sum_{i=1}^{d_1} x_{i,j,k} - \max_{i=1}^{d_1} x_{i,j,k}$$

(b) Now suppose we want a function `testFn2` which returns the $d_2 \times d_3$ matrix $\{z_{j,k}\}$ where
$$z_{j,k} = \sum_{i=1}^{d_1} x_{i,j,k}^k$$

5. In this question you will study the matrix $\mathbf{A}$ given by
$$\mathbf{A} = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \\ x_5 & y_5 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 3 \\ 2 & 0 \\ 4/9 & 4/3 \\ 14/9 & 4/3 \end{bmatrix}$$

This matrix consists of 5 coordinates which make up the letter 'A' in the two-dimensional Euclidean plane. The function

```
drawA <- function(X)
{
    lines(X[1:3,1], X[1:3,2])
    lines(X[4:5,1], X[4:5,2])
}
```

adds a graph of 'A' to an existing plot, when provided with a correct matrix of coordinates.
Use `plot(c(-10,10), c(-10,10), ann=F, type='n')` to create an empty graph space of appropriate size.

(a) Given an $n \times 2$ matrix $\mathbf{X}$, we can move the shape represented by the coordinates in $\mathbf{X}$ by a in the x-direction and by b in the y-direction by adding to $\mathbf{X}$ the $n \times 2$ matrix

$$\mathbf{S}_{a,b} = \begin{pmatrix} a & b \\ \vdots & \vdots \\ a & b \end{pmatrix}$$

Write a function `shift(X,a,b)` which, given an $n \times 2$ matrix $\mathbf{X}$ of coordinates, returns $\mathbf{X} + \mathbf{S}_{a,b}$. Try it out on $\mathbf{A}$ together with `drawA` to check how your function is working.

(b) Given an $n \times 2$ matrix $\mathbf{X}$ of coordinates we can rotate the shape represented by the coordinates in $\mathbf{X}$ anticlockwise about the origin by r radians by multiplying it by the matrix

$$\mathbf{R}_r = \begin{pmatrix} \cos r & \sin r \\ -\sin r & \cos r \end{pmatrix}$$

Write a function `rotate(X,r)` which takes an $n \times 2$ matrix $\mathbf{X}$ as an argument and returns $\mathbf{X}\mathbf{R}_r$. Try it out on $\mathbf{A}$ together with `drawA` to check how your function is working.

(c) Create a $5 \times 2 \times 25$ array `arrayA`, such that `arrayA[,,1]` is equal to $\mathbf{A} = \mathbf{A}\mathbf{R}_0$ and `arrayA[,,i]` is equal to

$$\mathbf{A}\mathbf{R}_{\frac{2\pi}{24}}^{i-1} = \mathbf{A}\mathbf{R}_{\frac{2\pi}{24}(i-1)} \quad \text{for } i = 2, 3, \ldots, 25$$

i.e. the i^{th} layer of `arrayA` is equal to $\mathbf{A}$ rotated anti-clockwise by $\frac{2\pi}{24}$ radians $(i-1)$ times. (Note that this is the same as rotating $\mathbf{A}$ anti-clockwise by $\frac{2\pi}{24}(i-1)$ radians.)
We can think of each matrix `arrayA[,,i]` as the position of letter 'A' at time i.

- **(1)** Now plot the resulting 25 instances of letter 'A' all on one graph.
- **(2)** Plot all 25 positions of the vertex of 'A' on one plot. (Remember that the coordinates of the vertex are given by the second row of each 5×2 'position' matrix.)
- **(3)** Plot the x-coordinate of the vertex of 'A' against time.

Now, for something a little different, let us create an animation of our rotating 'A'. For that you will need the 'animation' package; on *Windows*, download it by clicking on 'Packages', then on 'Install package(s)' and choosing 'animation' in the window that appears on the screen. Then to install the library click on 'Packages', 'Load package', and choose 'animation'.[1]

Once the package is installed and the library loaded, enter

```
oopt = ani.options(interval = 0.2, nmax = 25)
for(i in 1:ani.options("nmax")) {
    plot(c(-10,10), c(-10,10), ann=F, type='n')
    drawA(arrayA[,,i])
    ani.pause()
}
```

(d) Multiplying any $n \times 2$ matrix $\mathbf{X}$ of coordinates by a matrix

$$\mathbf{T}_{a,b} = \begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix}$$

stretches the shape represented by $\mathbf{X}$ by a in the x-direction and by b in the y-direction. If $a = b > 1$, then $\mathbf{X}$ is enlarged by a; if $a = b < 1$, then $\mathbf{X}$ is shrunk by a.

Write a function `scale(X,a,b)` which, given an $n \times 2$ matrix $\mathbf{X}$ of coordinates, returns $\mathbf{X}\mathbf{T}_{a,b}$. Hence, or otherwise, transform all the instances of 'A' in `arrayA` by $\mathbf{T}_{2,3}$. Plot the results on the same graph as results from (c) part (1) and/or create an appropriate animation.

(e) (Harder) Create a $5 \times 2 \times 25$ array `arArandom`, where `arArandom[,,1]` is equal to $\mathbf{A}$ and for each $i = 2, ..., 25$ the array slice `arArandom[,,i]` is obtained by first scaling, then rotating and moving `arArandom[,,i-1]` by random amounts. Plot the results and/or create an appropriate animation.

Hint: `runif(1,a,b)` generates a random number uniformly distributed in the interval (a, b).

[1] The method for installing packages is system dependent and you will need to consult your local documentation.