Kevin Chan MA415 Assignment 2: Basic R Exercise 3

Kevin Chan

February 1, 2018

1.

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

Here is `tmpFn1`

```
tmpFn1 <- function(xVec){
  return(xVec^(1:length(xVec)))
}

## simple example
a <- c(2, 5, 3, 8, 2, 4)

b <- tmpFn1(a)
b
```

```
## [1]    2   25   27 4096   32 4096
```

and now `tmpFn2`

```
tmpFn2 <- function(xVec2){

  n = length(xVec2)

  return(xVec2^(1:n)/(1:n))
}




c <- tmpFn2(a)
c
```

```
## [1]    2.0000   12.5000    9.0000 1024.0000    6.4000  682.6667
```

(b) Now write a fuction `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

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

```
tmpFn3 <- function(x, n){
  1+sum(x^(1:n)/(1:n))
}
# As an example, let's try plugging in x=1 and n=5. Should return 3.2833.
tmpFn3(1,5)
```

```
## [1] 3.283333
```

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

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

```
tmpFn <- function(xVec){
  n <- length(xVec)
  return((xVec[-c(n-1,n)]+xVec[-c(1,n)]+xVec[-c(1,2)])/3)
}
```

Try out your function. `tmpFn`(c(1:5,6:1))

```
tmpFn(c(1:5,6:1))
```

```
## [1] 2.000000 3.000000 4.000000 5.000000 5.333333 5.000000 4.000000 3.000000
## [9] 2.000000
```

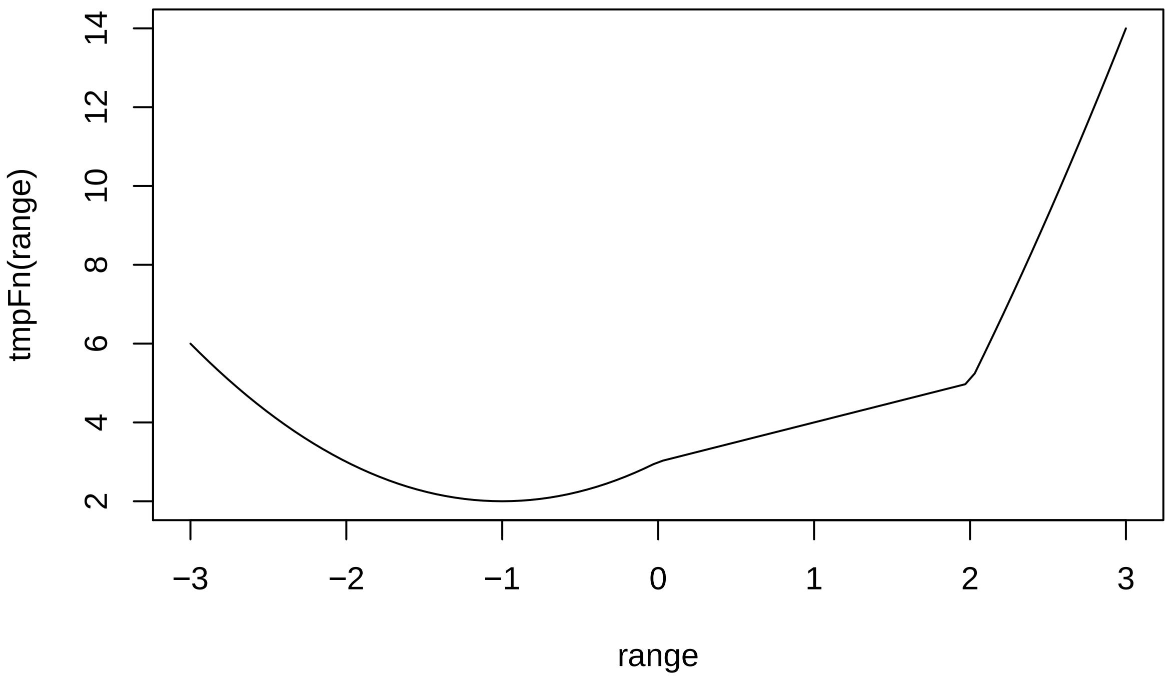
```
# Should return vector (2, 3, 4, 5, 5.333333, 5, 4, 3, 2).
```

3. Consider the continuous function

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

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

```
tmpFn <- function(x){
  ifelse(x < 0, x^2 + 2*x + 3, ifelse(x < 2, x + 3, x^2 + 4*x - 7))
}
range <- seq(-3, 3, len=100)
plot(range, tmpFn(range), type="l")
```



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:

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

```
matFn <- function(matrx){
  matrx[matrx%%2 == 1] <- 2 * matrx[matrx%%2 == 1]
  matrx
}
# Testing matFn on the matrix in the question.
matrxTest <- matrix(c(1,1,3,5,2,6,-2,-1,-3), nrow = 3, ncol = 3, byrow = TRUE)
matFn(matrxTest)
```

```
##      [,1] [,2] [,3]
## [1,]    2    2    6
```

```
## [2,]   10    2    6
## [3,]   -2   -2   -6
```

5. Write a function which takes 2 arguements n and k which are positive integers. It should return the nxn matrix:

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

```r
# Specific case with n=5 and k=2.
matTest <- diag(2, nr = 5)
matTest[abs(row(matTest) - col(matTest)) == 1] <- 1
matTest
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    2    1    0    0    0
## [2,]    1    2    1    0    0
## [3,]    0    1    2    1    0
## [4,]    0    0    1    2    1
## [5,]    0    0    0    1    2
```

```r
# General function.
matKFn <- function(n,k){
  matK <- diag(k, nr = n)
  matK[abs(row(matK) - col(matK)) == 1] <- 1
  return(matK)
}
```

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

If $0 \leq \alpha < 90$ then it is quadrant 1. If $90 \leq \alpha < 180$ then it is quadrant 2.
if $180 \leq \alpha < 270$ then it is quadrant3. if $270 \leq \alpha < 360$ then it is quadrant 4.
if $360 \leq \alpha < 450$ then it is quadrant 1.
And so on ...

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

```r
quadrant <- function(alpha){
  1 + (alpha%%360)%/%90
}
quadrant(120)
```

```
## [1] 2
```

7.

(a) Zeller's congruence is the formula:

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

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 21/07/1‘963 has $m = 5, k = 21, c = 19, y = 63$;
the date 21/2/63 has $m = 12, k = 21, c = 19, andy = 62$.
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.

```
weekday <- function(day, month, year){
  month <- month - 2
  if (month <= 0) {
    month <- month + 12
    year <- year - 1
  }
  cc <- year %/% 100
  year  <- year %% 100
  tmp <- floor(2.6*month - 0.2) + day + year + year %/% 4 + cc %/% 4 - 2 * cc
  c("Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday")[1+tmp%%7]
}
# Testing weekday function.
c(weekday(27,2,1997), weekday(18,2,1940), weekday(21,1,1963))
```

```
## [1] "Thursday" "Sunday"   "Monday"
```

```
# Should return the vector "Thursday", "Sunday", "Monday".
```

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

```
# The function weekday does not work if the input parameters are vectors with the same length and valid
weekdaynew <- function(day, month, year){
  a <- month <= 2
  month <- month - 2 + 12*a
  year <- year - a
  cc <- year %/% 100
  year  <- year %% 100
  tmp <- floor(2.6*month - 0.2) + day + year + year %/% 4 + cc %/% 4 - 2 * cc
  c("Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday")[1+tmp%%7]
}
# Testing weekdaynew on input parameters that are vectors.
weekdaynew(c(27,18,21), c(2,2,1), c(1997,1940,1963))
```

```
## [1] "Thursday" "Sunday"   "Monday"
```

```
# Should return "Thursday", "Sunday", "Monday"
```

8.

(a) Suppose $x_0 = 1$ and $x_1 = 2$ and

$$x_j = x_{j-1} + \frac{2}{x_{j-1}}$$

for $j = 1, 2, ...$ 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 $x_0, x_1, x_2, ..., x_{n-2}$.

```
testLoop <- function(n){
  xVec <- rep(NA, n-1)
  xVec[1] <- 1
  xVec[2] <- 2
  for(j in 3:(n-1))
    xVec[j] <- xVec[j-1] + 2/xVec[j-1]
  return(xVec)
}
# Testing with n=3.
testLoop(3)
```

```
## [1] 1 3 3
```

```
# Gives the incorrect answer. Answer should be the vector c(1,2,3), not c(1,3,3), which R outputs with
# New testLoop function that takes n < 4 into consideration.
testLoop <- function(n){
  xVec <- rep(NA, n-1)
  xVec[1] <- 1
  xVec[2] <- 2
  for(j in 3:(n-1))
    xVec[j] <- xVec[j-1] + 2/xVec[j-1]
  if (n < 4)
    stop("The argument n must be an integer which is at least 4. \n")
  return(xVec)
}
```

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

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

where `n` is the length of `yVec`.

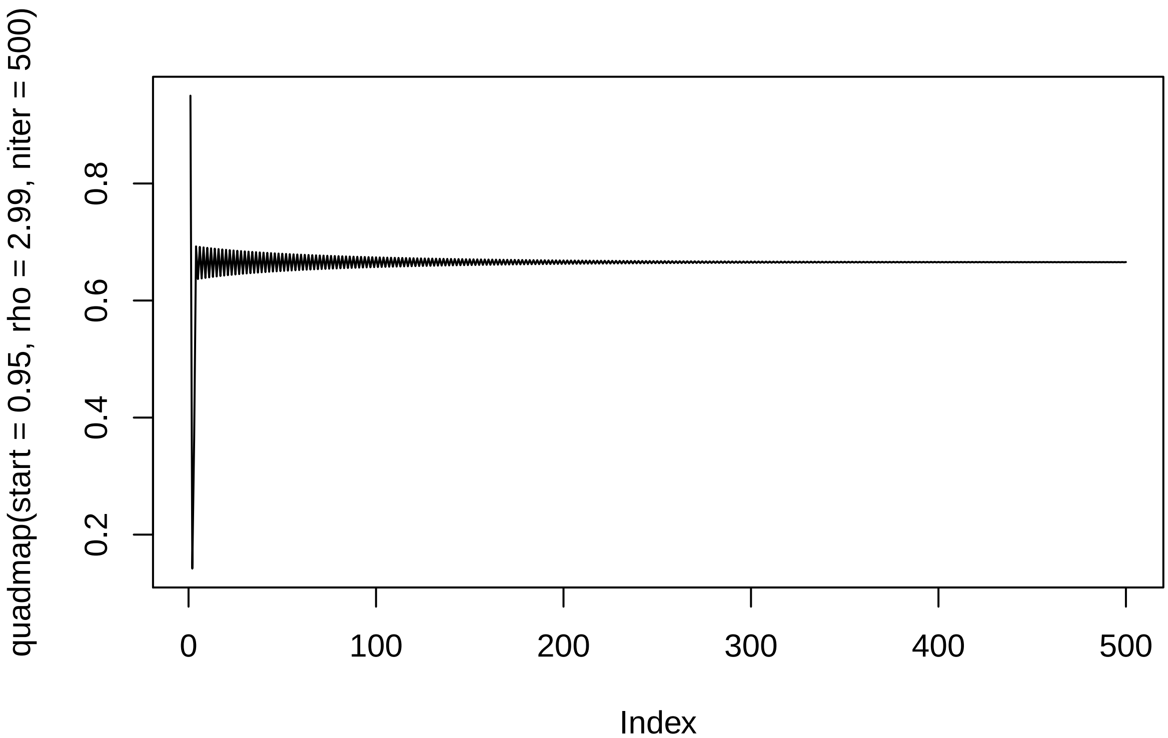
```
testLoop2 <- function(yVec){
  yLen <- length(yVec)
  sum(exp(seq(along=yVec)))
}
```

9.

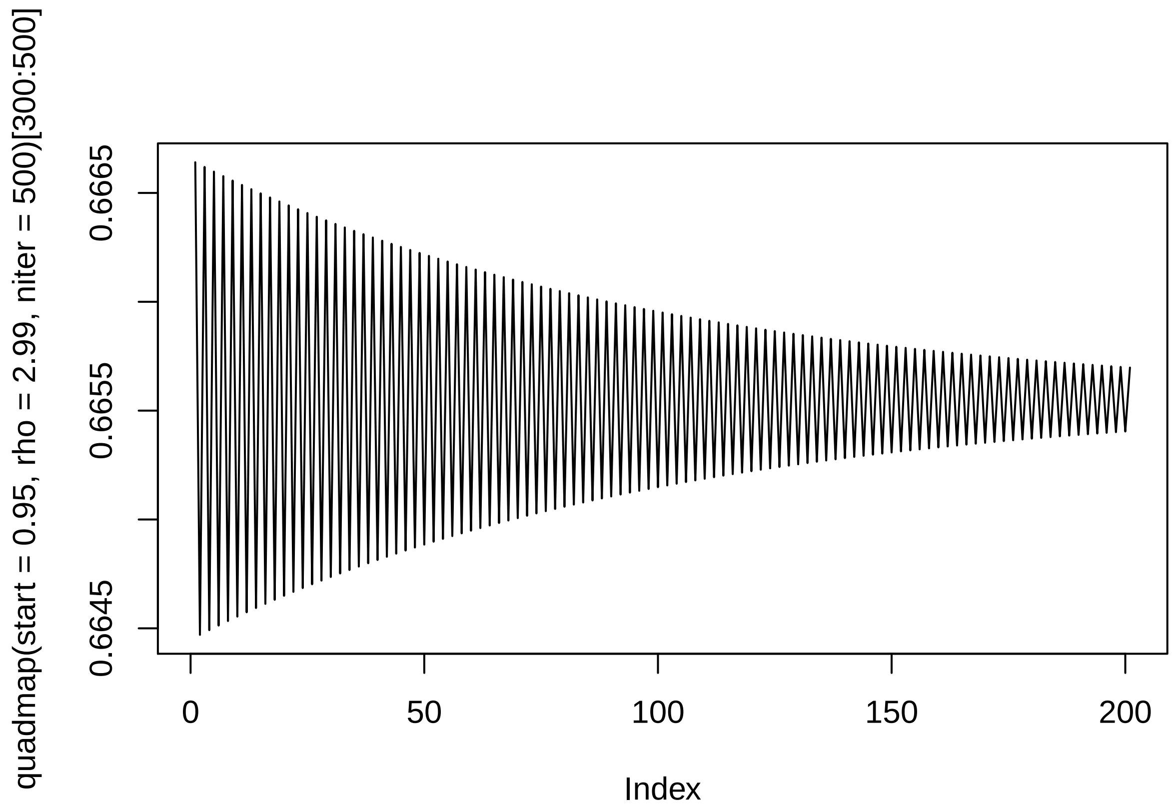
Solution of the difference equation $x_n = rx_{n-1}(1 - x_{n-1})$, with starting value x_1.

(a) Write a function `quadmap(start, rho, niter)` which returns the vector $(x_1, ..., x_n)$ 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 when you have written: Now switch back to the Commands window and type: `plot(tmp, type="l")` Also try the plot `plot(tmp[300:500], type="l")`

```
quadmap <- function(start, rho, niter){
  x <- rep(NA, niter)
  x[1] <- start
  for (i in 1:(niter - 1))
    x[i + 1] <- rho*x[i] * (1 - x[i])
  x
}
# Plotting graphs of the quadmap function with start=0.95, rho=2.99, and niter=500.
plot(quadmap(start=0.95, rho=2.99, niter=500), type="l")
```




```
plot(quadmap(start=0.95, rho=2.99, niter=500)[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.)

```
quadmap2 <- function(start, rho, eps = 0.02){
  x1 <- start
  x2 <- rho*x1*(1 - x1)
  niter <- 1
  while(abs(x1 - x2) >= eps){
    x1 <- x2
    x2 <- rho*x1*(1 - x1)
    niter <- niter + 1
  }
  niter
}
# Testing the function quadmap2 with start=0.95 and rho=2.99.
quadmap2(start = 0.95, rho = 2.99, eps = 0.02)
```

```
## [1] 84
```

10.

(a) Given a vector $(x_1, ..., x_{n-1})$, the sample autocorrelation of lag k is defined to be

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

Thus,

$$r_k = \frac{\sum_{i=2}^{n}(x_i - \overline{x})(x_{i-k} - \overline{x})}{\sum_{i=1}^{n}(x_i - \overline{x})^2} = \frac{(x_2 - \overline{x})(x_1 - \overline{x}) + ... + (x_n - \overline{x})(x_{n-1} - \overline{x})}{\sum_{i=1}^{n}(x_i - \overline{x})^2}$$

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,. . . ,53,56).

```
tmpFn <- function(xVec){
  e <- xVec - mean(xVec)
  den <- sum(e^2)
  xLen <- length(xVec)
  r1 <- sum(e[2:xLen]*e[1:(xLen-1)])/den
  r2 <- sum(e[3:xLen]*e[1:(xLen-2)])/den
  list(r1, r2)
}
# Finding r1 and r2 for the vector (2,5,8,...,53,56).
tmpFn(c(2:56, by = 3))
```

```
## [[1]]
## [1] 0.8547495
##
## [[2]]
## [1] 0.8045096
```

(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 $(r_0 = 1, r_1, ..., r_k)$.

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`.)

```
gentmpFn <- function(xVec,k){
  e <- xVec - mean(xVec)
  den <- sum(e^2)
  xLen <- length(xVec)
  tmpFn <- function(j){
    sum(e[(j+1):xLen]*e[1:(xLen-j)])/den
  }
  c(1, sapply(1:k, tmpFn))
}
# Testing with vector (2,5,8,...,53,56) and k=2.
gentmpFn(c(2:56, by=3), 2)
```

```
## [1] 1.0000000 0.8547495 0.8045096
```