Simple R Functions

Nanfang Hong (U92430309)

January 27, 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 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

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

```
tmpFn3 <- function(x, n){
  return(1 + sum(sapply(1:n, function(i) x ^ i / i)))
}
```

2. Write a funciton `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){
  n2 <- 1:(length(xVec) -2)
  return((xVec[n2] + xVec[n2 + 1] + xVec[n2 + 2]) / 3)
}
```

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

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

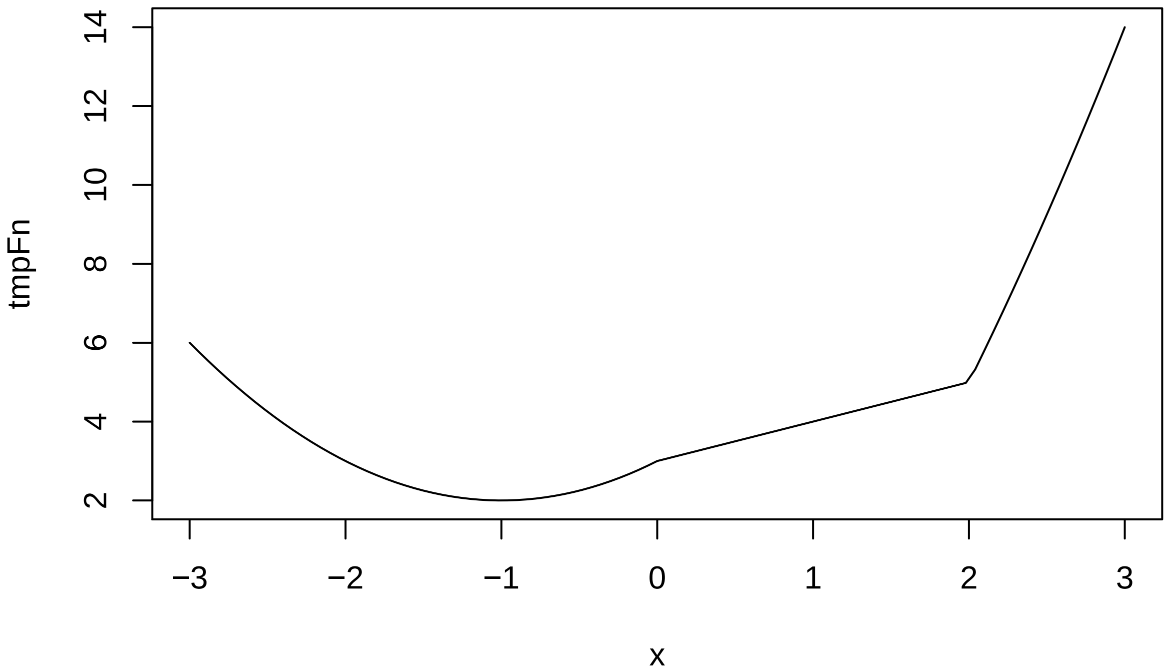
```
## [1] 2.000000 3.000000 4.000000 5.000000 5.333333 5.000000 4.000000 3.000000
## [9] 2.000000
```

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(xVec){
  return(
    (xVec < 0) * (xVec ^ 2 + 2 * xVec + 3)
  + (xVec >= 0 & xVec < 2) * (xVec + 3)
  + (xVec >= 2) * (xVec ^ 2 + 4 * xVec - 7)
  )
}
x <- seq(-3, 3, 0.01)
plot(tmpFn, -3, 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:

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

```
tmpFn <- function(xMat){
  return(
    (xMat %% 2 == 0) * xMat
  + (xMat %% 2 != 0) * (xMat * 2)
  )
}
xMat <- matrix(c(1, 1, 3, 5, 2, 6, -2, -1, -3), nrow = 3, byrow = TRUE)
tmpFn(xMat)
```

```
##      [,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:

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

```
tmpFn <- function(k, n){
  xMat <- matrix(rep(0, n ^ 2), nrow = n)
  xMat <- row(xMat) - col(xMat)
  xMat[abs(xMat) == 1] <- 1
  xMat[abs(xMat) != 1] <- 0
  xMat <- xMat + diag(k, nrow = n)
  return(xMat)
}
tmpFn(2, 5)
```

```
##      [,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
```

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.

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

7.

(a) Zeller's congruence is the formula:

$$f = ([2.6m - 0.2] + k + y + [y/4] + [c/4] - 2c)mod7$$

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 $m = 12, k = 21, c = 19, and y = 62$.
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 valid entries?

```
weekday <- function(day, month, year){
  kVec <- day
  yVec <- ifelse(
    month > 2,
    year - floor(year / 100) * 100,
    year - floor(year / 100) * 100 - 1
  )
  cVec <- floor(year / 100)
  mVec <- ifelse(month > 2, month - 2, month + 10)
  return(
    (
     floor(2.6 * mVec - 0.2)
   + kVec
   + yVec
   + floor(yVec/4)
   + floor(cVec/4)
   - 2 * cVec
    ) %% 7 + 1
  )
}
```

28/1/2018 is Sunday, output should be 1

```
weekday(28, 1, 2018)
```

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

8.

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

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

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 $x_0, x_1, x_2, ..., x_{n-2}$

Professor Haviland admits that here is a typo: if you put $j = 1$, x_0 and x_1 don't satisfy the recursive equation. Professor Haviland asks me in email:

"So, what do you think is wrong with this problem? Imagine corrected statements of the problem. There aren't very many. How do the series produced by the corrected statement differ? How are they the same?"

There are many possible statements that can make problem correct.

1. j begins with 2, instead of 1.
2. $x_1 = 3$, instead of $x_1 = 2$

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

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

Conventionally, in recursive equation indexes go from highest to lowest from left to right. So, there are many other possibilities which we don't cover here. I will do those four possible corrected statements and see if I have extra credit.

1. j begins with 2, instead of 1.

```
tmpFn <-function(n){
  xVec <- c(1, 2)
  for (i in 2: (n - 2)){
    xVec[i + 1] <- xVec[i] + 2 / xVec[i]
  }
  return(xVec)
}

tmpFn(10)
```

```
## [1] 1.000000 2.000000 3.000000 3.666667 4.212121 4.686941 5.113659 5.504768
## [9] 5.868090
```

2. $x_1 = 3$, instead of $x_1 = 2$

```
tmpFn <-function(n){
  xVec <- c(1, 2)
  for (i in 1: (n - 2)){
    xVec[i + 1] <- xVec[i] + 2 / xVec[i]
  }
  return(xVec)
}

tmpFn(10)
```

```
## [1] 1.000000 3.000000 3.666667 4.212121 4.686941 5.113659 5.504768 5.868090
## [9] 6.208916
```

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

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

tmpFn(10)
```

```
## [1] 1.000000 2.000000 3.000000 3.000000 3.666667 3.666667 4.212121 4.212121
## [9] 4.686941
```

4.
$$x_{j+1} = x_j + \frac{2}{x_{j-1}} \text{ for } j = 1, 2, ...$$

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

tmpFn(10)
```

```
## [1] 1.000000 2.000000 4.000000 5.000000 5.500000 5.900000 6.263636 6.602619
## [9] 6.921923
```

(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`.

```
testLoop2 <- function(yVec){
  return(sum(exp(1:length(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 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")`

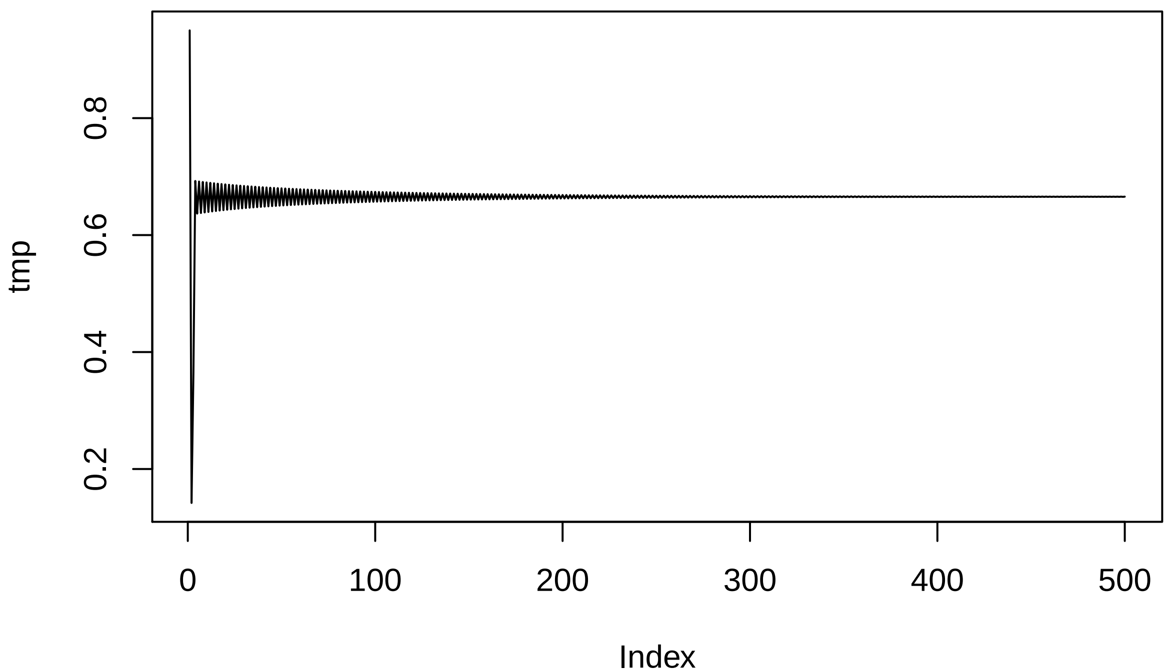
```
quadmap <- function(start, rho, niter){
  yVec <- c(start)
  loop <- 1
  while (loop != niter){
    yVec <- rbind(yVec, rho * yVec[loop] * (1 - yVec[loop]))
    loop <- loop + 1

  }
  return(yVec)
}
```

Try out

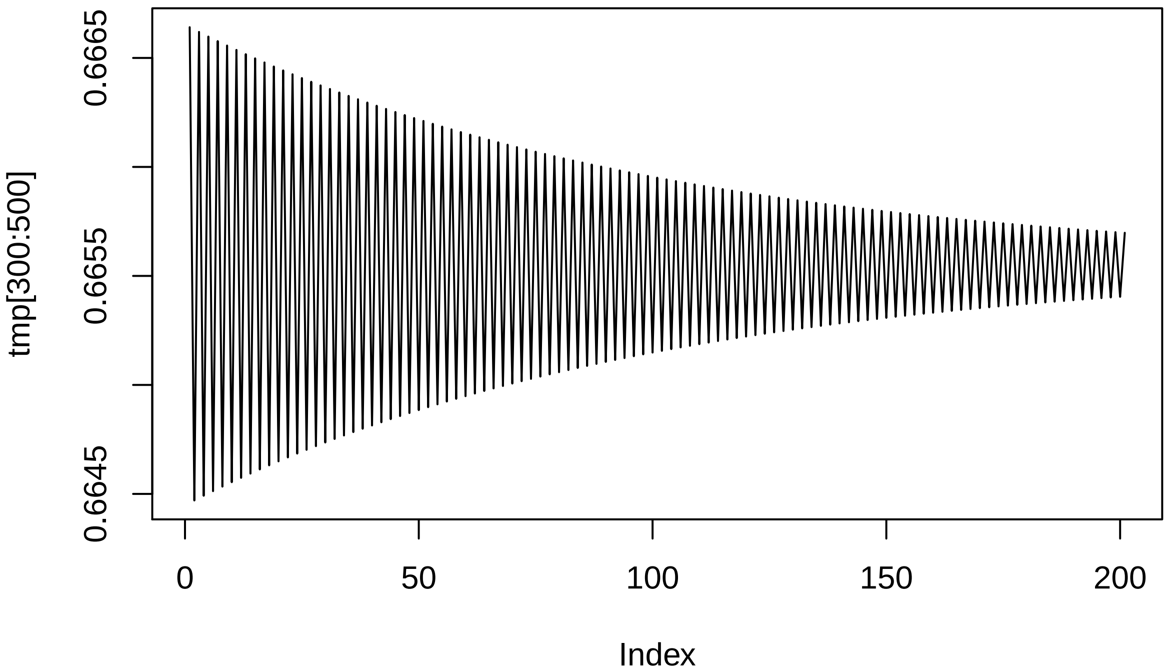
```r
quadmap(0.1, 2, 9)
```

```
##             [,1]
## yVec 0.1000000
##      0.1800000
##      0.2952000
##      0.4161139
##      0.4859263
##      0.4996039
##      0.4999997
##      0.5000000
##      0.5000000
```

```r
tmp <- quadmap(start=0.95, rho=2.99, niter=500)
plot(tmp, type="l")
```




```r
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.)

```
determineNumber <- function(start, rho){
  yVec <- quadmap(start, rho, 2)

  loop <- 2
  while (abs(yVec[length(yVec)] - yVec[length(yVec) - 1]) >= 0.02){
    yVec <- rbind(yVec, rho * yVec[loop] * (1 - yVec[loop]))
    loop <- loop + 1
  }
  return(length(yVec) - 1)
}
determineNumber(0.95, 2.99)
```

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

10.

(a) Given a vector $(x_1, ..., x_n)$, 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(cVec)` 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, k){
  xVecMean <- mean(xVec)
  lowerSum <- sum((xVec - xVecMean) ^ 2)
  upperSum <- sum(sapply((k+1):length(xVec), function(i) (xVec[i] - xVecMean) * (xVec[i-k] - xVecMean))
  return(upperSum / lowerSum)
}

tmpFn(seq(2,56,3), 1)
```

```
## [1] 0.8421053
```

```
tmpFn(seq(2,56,3), 2)
```

```
## [1] 0.6859649
```

(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`.)

```
tmpFn4 <- function(xVec, k){
  outPut <- rep(0, k)
  for (i in 1:k) outPut[i] <- tmpFn(xVec, i)
  return(outPut)
}

tmpFn4(seq(2,56,3), 2)
```

```
## [1] 0.8421053 0.6859649
```