Simple R Functions

Ziran Min

January 26, 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){
  return (1 + sum((x^(1:n))/(1:n)))
}
d <- tmpFn3(2, 5)
d
```

```
## [1] 18.06667
```

---

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:

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

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

```
tmpFn <- function(xVec){
  n <- length(xVec)
  first <- (1:(n-2))
  second <- first + 1
  third <- first + 2
  return ((xVec[first] + xVec[second] + xVec[third])/3)
}
e <- tmpFn(c(1:5, 6:1))
e
```

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

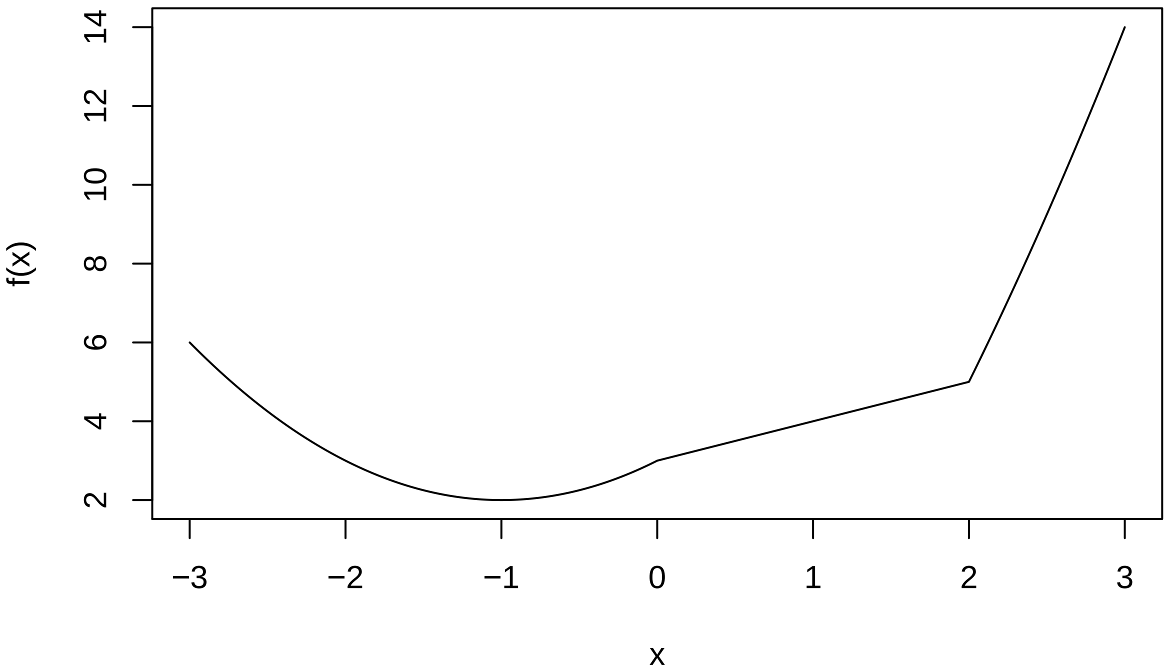
---

3. Consider the continuous function

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

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)
  )
}
tmp <- seq(-3, 3, len=10000)
plot(tmp, tmpFn(tmp), type = "l", xlab = "x", ylab = "f(x)")
```



******

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(m){
  m[m%%2 == 1] <- 2* m[m%%2 == 1]
  m
}

test <- matrix(c(1,1,3,5,2,6,-2,-1,-3), nrow=3, byrow=TRUE)
test
```

```
##      [,1] [,2] [,3]
## [1,]    1    1    3
## [2,]    5    2    6
## [3,]   -2   -1   -3
```

```
tmpFn(test)
```

```
##      [,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:

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

******

```
tmpFn <- function(n, k) {
  tmp <- diag(k, nrow = n)
  tmp[abs(row(tmp) - col(tmp)) == 1] <- 1
  tmp
}
tmpFn(6,3)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    3    1    0    0    0    0
## [2,]    1    3    1    0    0    0
## [3,]    0    1    3    1    0    0
## [4,]    0    0    1    3    1    0
## [5,]    0    0    0    1    3    1
## [6,]    0    0    0    0    1    3
```

---

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) {
 1 + (alpha %% 360) %/% 90
}
quadrant(45)
```

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

```
quadrant(105)
```

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

```
quadrant(250)
```

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

```
quadrant(300)
```

```
## [1] 4
```

```r
quadrant(1000)
```

```
## [1] 4
```

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 $m = 12, k = 21, c = 19, and y = 62$.
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. ******

```r
weekday <- function(day, month, year) {
 k <- day
 m <- month - 2
  if (m <= 0) {
    m <- m + 12
    year <- year - 1
  }

 c <- year %/% 100
 y <- year %% 100
 tmp <- (floor(2.6*m - 0.2) + k + y + y %/% 4 + c %/% 4 - 2 * c)
 week <- c("Sunday","Monday","Tuesday", "Wednesday", "Thursday", "Friday", "Saturday")
 return (week[tmp%% 7 + 1])
}
weekday(30,1,2018)
```

```
## [1] "Tuesday"
```

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

```r
days <- c(1,6,10)
months <- c(2,2,2)
years <- c(2018,2018,2018)
weekday(days, months,years)
```

```
## Warning in if (m <= 0) {: the condition has length > 1 and only the first
## element will be used

## [1] "Thursday" "Tuesday"  "Saturday"
```

```
# This returns "the condition has length > 1 and only the first element will be used" but still gives m
```

8.

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

$$x_j = x_{j-1} + 2/x_{j-1}$$

for $j = 1, 2, ...$

Write a function testLoop(n) that returns the first n - 1 elements of the sequence *****

```
testLoop <- function(n){
  x <- rep(0, n-1)
  x[1] <- 1
  x[2] <- 2
  for(i in 3:length(x)){
    x[i] <- x[i-1] + 2/x[i-1]
  }
  return(x)
}
#test
testLoop(10)
```

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

(b) define testLoop2 that calculates

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

for a vector y

```
testLoop2 <- function(yVec){
  n <- length(yVec)
  tmp <- exp(1:n)
  return(sum(tmp))
}
testLoop2(c(1:8))
```

```
## [1] 4714.224
```

```
testLoop2(c(2,59,8,3,10,34,55,4))
```

```
## [1] 4714.224
```

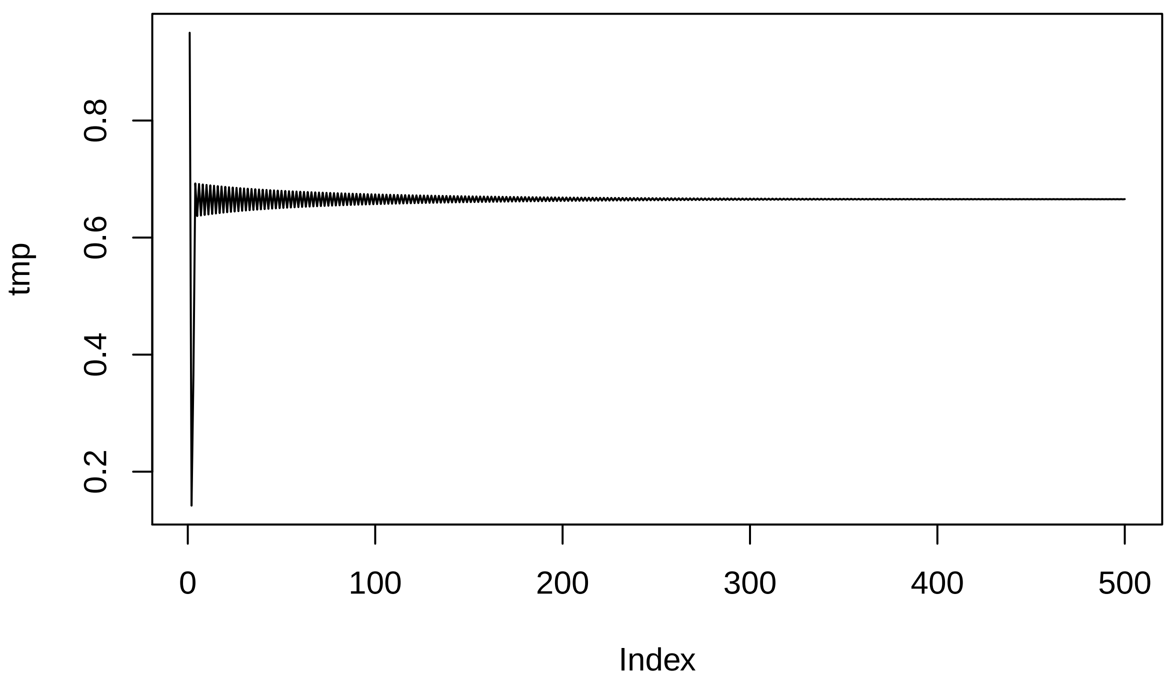
9.

(a) ____________________

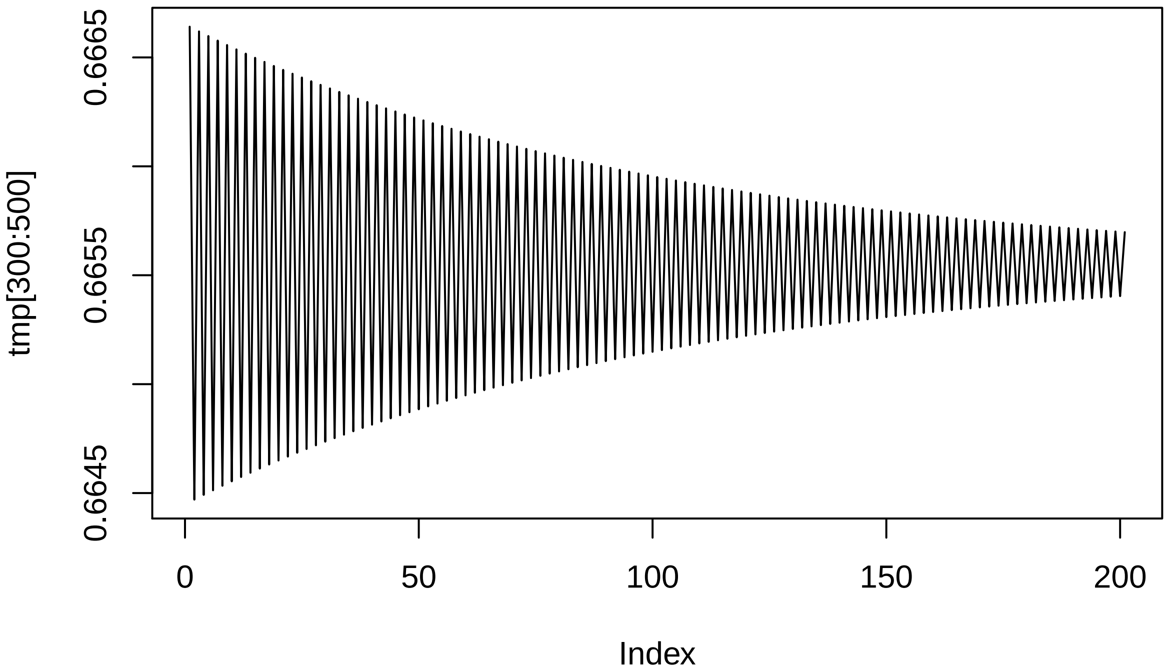
```
quadmap <- function(start, rho, niter){
  x <- rep(0, niter)
  x[1] <- start
  for(i in 2:niter){
    x[i] = rho * x[i-1] * (1- x[i-1])
  }
  return(x)
}

quadmap(start=0.8, rho=2, niter=100)
```

```
##   [1] 0.8000000 0.3200000 0.4352000 0.4916019 0.4998589 0.5000000 0.5000000
##   [8] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000
##  [15] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000
##  [22] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000
##  [29] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000
##  [36] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000
##  [43] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000
##  [50] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000
##  [57] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000
##  [64] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000
##  [71] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000
##  [78] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000
##  [85] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000
##  [92] 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000
##  [99] 0.5000000 0.5000000
```

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



```r
plot(tmp[300:500], type="l")
```



(b)

```r
quadmap2 <- function(start, rho){
  x1 <- start
  n <- 1
  while(abs(x1 - rho * x1 * (1 - x1)) >= 0.02 ){
    x1 <- rho * x1 * (1 - x1)
    n <- n + 1
  }
  return(n)
}
quadmap2(start=.95, rho=2.99)
```

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

---

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

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

Where x_s is the sample mean write a function to solve for r1 and r2

```r
tmpFn <- function(xVec){
  x <- xVec - mean(xVec)
  sxx <- sum(x^2)
  n <- length(xVec)
  r1 <- sum ( x[2:n] * x[1:(n-1)] ) / sxx
  r2 <- sum ( x[3:n] * x[1:(n-2)] ) / sxx
```

```
  return (c(r1, r2))
}

#testing function over given interval
tmpFn(seq(2,56, by=3))
```

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

since I started out with nearly the generalised code, It is simple to allow for my result to list $r_0, ..., r_k$ (b)

```
tmpFn2 <- function(xVec, k){
  x <- xVec - mean(xVec)
  sxx <- sum(x^2)
  n <- length(x)
  return(c(1, sapply((1:k), function(a) sum( x[(a+1):n] * x[1:(n-a)] / sxx  ))))
}

tmpFn2(seq(2,56, by=3), 5)
```

```
## [1] 1.0000000 0.8421053 0.6859649 0.5333333 0.3859649 0.2456140
```
