Simple R Functions

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))
}



a<- c(1:10)
c <- tmpFn2(a)
c
```

```
##  [1]          1          2          9         64        625       7776
##  [7]     117649    2097152   43046721 1000000000
```

(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

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

```
funct1b <- function(xVec1,nVec1){

  y <- 1:nVec1

  f <- 1 + sum(xVec1^y/y)}



hi <- funct1b(3,2)
hi
```

```
## [1] 8.5
```

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:

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

```
funct2 <- function(xVec)

{
 n <- length(xVec)
  for (i in 1:(n-2)){
xVec[i] <- (xVec[i] + xVec[i+1] + xVec[i+2])/3
}
return(xVec[1:(n-2)])
}
f = c(1,2,3,4,5,6)
funct2(f)
```

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

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 $-3 < x < 3$.

```
funct3 <- function(xVec) {
a <- xVec < 0
b <- xVec < 2
c <- xVec >= 2
if (a) {sum(xVec**2 + xVec * 2 + 3)}
if (b) {sum(xVec + 3)}
if (c) {sum(xVec**2 + xVec*4 -7)}
}
funct3(5)
```

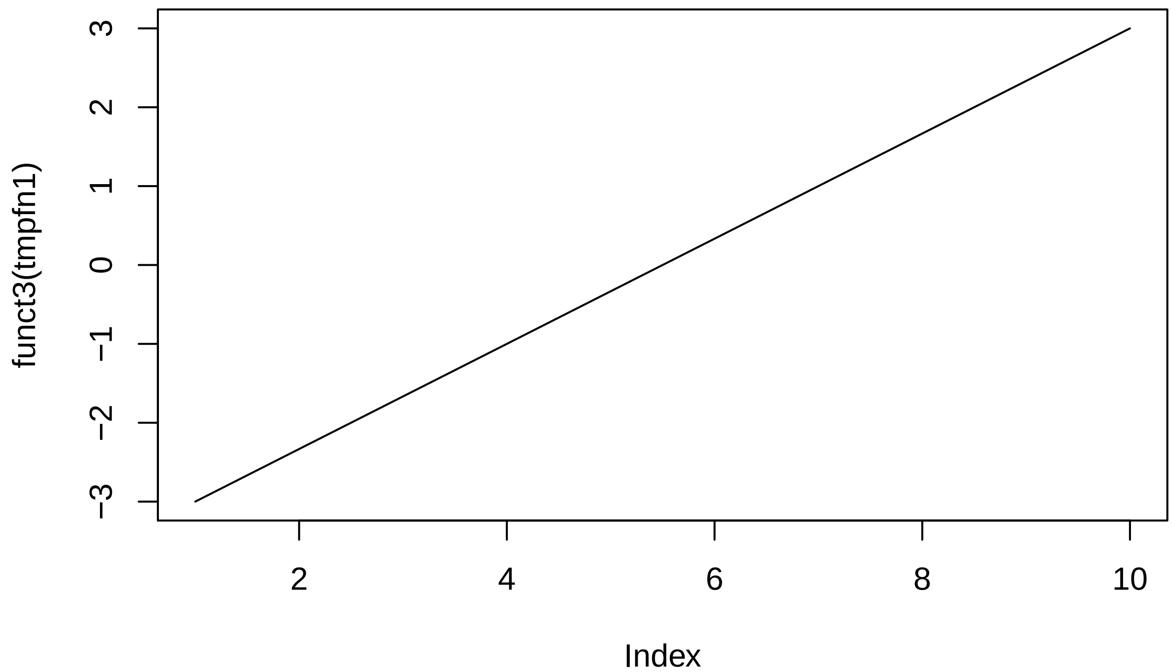
```
## [1] 38
```

```
tmpfn1 <- seq(-3, 3, len=10)
plot(tmpfn1, funct3(tmpfn1), type="l")
```

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

## Warning in if (b) {: the condition has length > 1 and only the first
## element will be used

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



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:

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

```
funct4 <- function(trix){

trix[trix%%2 == 1] <- 2 * trix[trix%%2 == 1]
trix}


T <- matrix(
c(2, 4, 3, 1, 5, 7),
nrow=3,
ncol=3,
```

```
byrow = TRUE)

T
```

```
##      [,1] [,2] [,3]
## [1,]    2    4    3
## [2,]    1    5    7
## [3,]    2    4    3
```

```
h <- funct4(T)
h
```

```
##      [,1] [,2] [,3]
## [1,]    2    4    6
## [2,]    2   10   14
## [3,]    2    4    6
```

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

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

```
Funct5 <- function(n,k){
gh <- diag(2, nr = 5)
gh[abs(row(gh) - col(gh)) == 1] <- 1
gh

}


n = c(4:8)
Funct5(n)
```

```
##      [,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.

```
Funct6 <- function(quad)
{
1 + (quad%%360)%/%90
}

d = c(60,40,50,100)
h = Funct6(d)
h
```

```
## [1] 1 1 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.

```
Funct7 <- function(k,m,y){
  m <- m - 2
  if (m <= 0){
   m <- m +12
  }
  c <- as.integer(y/100)
  y <-as.integer(y%%100)
  x <- as.integer(2.6*m -0.2)
  f <- (x + k +y+ as.integer(y/4)+as.integer(c/4)-2*c)%%7 +1
  return(f)
  }


h <- Funct7(21,07,1963)
h
```

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

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

```
h <- Funct7(07,21,1963)
h
```

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

8(a)Suppose

$$x_0 = 1 and x_1 = 2 and x_j = x_j - 1 + \frac{2}{xj - 1}$$

Write a function testLoop which takes the single argument n and returns the first n ??? 1 values of the sequence

$$xjj???0$$

that means the values of x0, x1, x2, . . . , xn???2

```
Funct8 <- function(x){

  n <- length(x)
  x <- rep(NA, n-1)
  x[1] <- 1
  x[2] <- 2

  for (i in 3:(n-1)) {
    x[i] <- x[i-1] + 2/x[i-1]}
  return(x)
  }

H <- c(1:6)
Funct8(H)
```

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

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.

```
funct8b <- function(xvec){
  total <- 0;
  for(j in 1:length(xvec)){
    print(total)
    total <- total + xvec[j]**j
  }
   return(total)
}

H<- (1:10)
funct8b(H)
```

```
## [1] 0
## [1] 1
## [1] 5
## [1] 32
## [1] 288
## [1] 3413
## [1] 50069
## [1] 873612
## [1] 17650828
## [1] 405071317
```

```
## [1] 10405071317
```

9a Solution of the difference equation

$$x_n = rx_{n-1}(1 - x_{n-1})$$

with starting value x1. b.Write a function quadmap(start, rho, niter) which returns the vector

$$x_1, \cdots x_n$$

where

$$x_k = rx_{n-1}(1 - x_{n-1})$$

and niter denotes n, start denotes x1, and rho denotes r.

Try out the function you have written: . for r = 2 and 0 < x1 < 1 you should get xn ??? 0.5 as n ??? ???. . 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

$$|xn???xn???1| < 0.02$$

. So this function has only 2 arguments: start and rho. (For start=0.95 and rho=2.99, the answer is 84.)

```
funct9b <- function(strt, rho)
{

eps <- 0.02
x1 <- strt
x2 <- rho*x1*(1 - x1)
n  <- 1
while(abs(x1 - x2) >= eps) {
x1 <- x2
x2 <- rho*x1*(1 - x1)
n  <- n  + 1
}
n
}



funct9b(strt=0.95, rho=2.99)
```

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

10a Given a vector (x1, . . . , xn), the sample autocorrelation of lag k is defined to be rk =

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

Thus r1 =

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

Write a function tmpFn(xVec) which takes a single argument xVec which is a vector and returns a list of two values: r1 and r2. In particular, find r1 and r2 for the vector (2, 5, 8, . . . , 53, 56).

```
funct10a <- function(x){
n <- length(x)
y <- x- mean(x)
```

```
div <- sum(y**2)
r1 = 0
r2 = 0

for(i in 2:n){
r1 <- r1 + sum(y[i]*y[i-1])/div
}

for(i in 3:n){
r2 <- r2 + sum(y[i]*y[i-2])/div
}

print(r1)
print(r2)
}

FT <- c(1:4)
funct8b(FT)
```

```
## [1] 0
## [1] 1
## [1] 5
## [1] 32

## [1] 288
```

(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 (r0 = 1, r1, . . . , rk). 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.)

```
funct10b <- function(x,k){
   n <- length(x)
   y <- x- mean(x)
   div <- sum(y**2)

   xvec = integer(k)
   for(j in 1:k){
     for(i in (j+1):n){
       xvec[j]  <- xvec[j] + (y[i]*y[i-j])/div
    }

   }
  print(xvec)
}

funct10b(c(1,2,3,4),2)
```

```
## [1]  0.25 -0.30
```