Simple R Functions

Junpei Xiao

Feb 2, 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))
}

### simple test

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

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

```
tmpn3 <- function(x,n){
  return (1 + sum((x^(1:n))/(1:n)))
}
### simple test
x = 2
n =2

c<-tmpn3(x,n)
c
```

```
## [1] 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:

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

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

```
tmpn<- function(xVec){
  n = length(xVec)
  return ((xVec[1:(n-2)]+xVec[2:(n-1)]+xVec[3:n])/3)

}

### simple test
a<- c(1:5,6:1)
c <- tmpn(a)
c
```

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

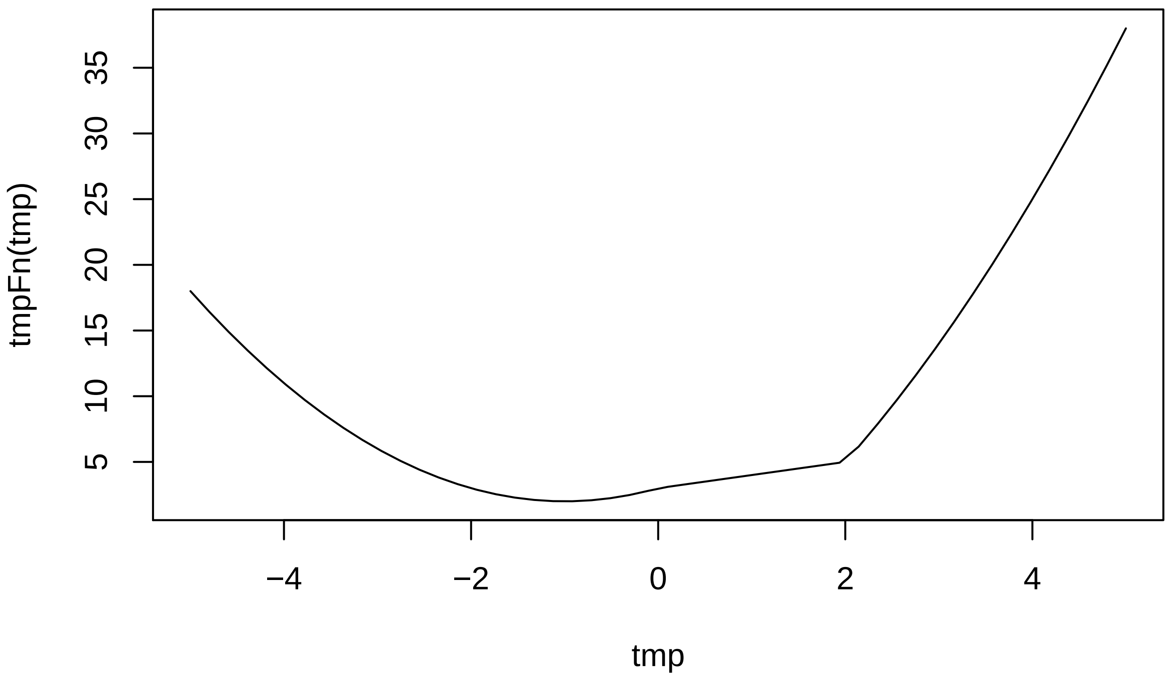
3. Consider the continuous function

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

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))
}
### simple test
tmp <- seq(-5, 5, len=50)
plot(tmp, tmpFn(tmp), 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:

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

```
tmpMn <- function(m){
  m[mat%%2 == 1] <- 2 * m[m%%2 == 1]
  return(m)
}
### simple test

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

```
##      [,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:

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

```r
tmp <- diag(2, nr = 5)
tmp[abs(row(tmp) - col(tmp)) == 1] <- 1
tmp
```

```
##      [,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
tmpQn <- function(n,k){
  tmp <- diag(k, nrow = n,ncol=n)
  tmp[abs(row(tmp) - col(tmp)) == 1] <- 1
  tmp
}
### simple test

n <- 5
k <- 7

a <-tmpQn(n,k)
a
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    7    1    0    0    0
## [2,]    1    7    1    0    0
## [3,]    0    1    7    1    0
## [4,]    0    0    1    7    1
## [5,]    0    0    0    1    7
```

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){
  return (1 + (alpha%%360)%/%90)
}
### simple test

alpha <- 200
b <- quadrant(alpha)
b
```

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

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 $m = 12, k = 21, c = 19, andy = 62$.
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.

(a)

```r
weekday <- function(day,month,year){

  f <- month <= 2
  month <- 12*f + month -2
  year <- year - f
  c <- year %/% 100
  year <- year %% 100
  Zeller <- floor(2.6*month - 0.2) + day + year + year %/% 4 + c %/% 4 - 2 * c
  return(c("Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday")[1+Zeller%%7])

}

### simple test

day<- 01
month <-02
year <- 2018

week <- weekday(day,month,year)
week
```

```
## [1] "Thursday"
```

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

(b). My Function works fine if the input parameters are Vectors. However if the input are invalid, the result may be wrong.

8(a).Suppose $x_1 = 1$ and $x_2 =2 and

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

Write a function `testloop` which takes the single argument n and return the first $n-1$ values of the sequence $\{x_j\}j \geq 0$: that means the values of $x_0, x_2, ...x_n - 2$

(a)

```
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)

}
n <- 5
b <- testloop(n)
b
```

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

(b)Now write a function *testloop*2 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

(b)

```
testLoop2 <- function(yVec){

  n <- length(yVec)
  return (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 you have written:

- for $r = 2$ and $0 < x_1 < 1$ you should get $x_n {\rightarrow} 0.5$ as $n {\rightarrow}$infinity
- try $tmp < -quadmap(start = 0.95, rho = 2.99, niter = 500)$

Now switch back to the Commands window and type: $plot(tmp, type = "1")$ Also try the plot $plot(tmp[300 : 500], type = "1")$

```
quadmap <- function(start, rho, niter)
{
  xVec <- rep(NA,niter)
  xVec[1] <- start
  for(i in 1:(niter-1)) {
    xVec[i + 1] <- rho * xVec[i] * (1 - xVec[i])
  }
  x
}
```

(b)Now write a function which determines the number of iteratons 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
}
```

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

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

Thus

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

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 $r1$ and $r2$ for the vector(2,5,8,. . . .53,56).

```
tmpFn <- function(xVec){

  y <- xVec - mean(xVec)
  z <- sum(y^2)
  n <- length(xVec)
  r1 <- sum( y[2:n] * y[1:(n-1)] )/z
  r2 <- sum( y[3:n] * y[1:(n-2)] )/z
  list(r1 = r1, r2 = r2)
}
xVec <- seq(2,56,2)
```

```
test <- tmpFn(xVec)
test
```

```
## $r1
## [1] 0.8928571
##
## $r2
## [1] 0.7862616
```

(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.)

```
tmpFn2 <- function(x, k){
  y <- x - mean(x)
  z <- sum(y^2)
  n <- length(x)
  tmpFn <- function(j){ sum( y[(j+1):n] * y[1:(n-j)] )/z }
  c(1, sapply(1:k, tmpFn))
}
```