Simple R Functions

yaru peng

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){
  1+sum((x^(1:n))/(1:n))
}

d <-tmpFn3(1,10)
d
```

```
## [1] 3.928968
```

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))

```
tempFn <- function(xVec){
  x<-length(xVec)
  (xVec[1:(x-2)]+xVec[2:(x-1)]+xVec[3:x])/3
}

tempFn(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
```

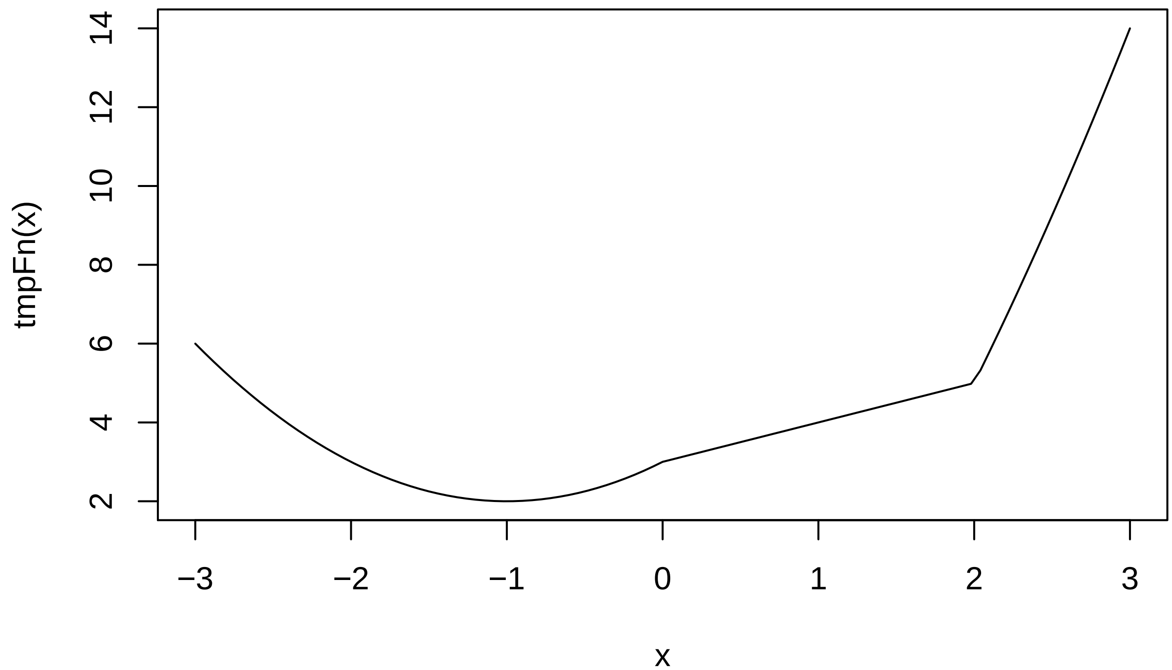
3. Consider the continuous function

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

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(xVec){
  ifelse(xVec<0,xVec^2+2*xVec+3,ifelse(xVec<2,xVec+3,xVec^2+4*xVec-7))
}

curve(tmpFn, from=-3, to=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:

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

```
newMatrix<-function(x){
  x[x%%2==1]<-2*x[x%%2==1]
}
```

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

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

```
tFn<-function(n,k){
  newMat<-matrix(rep(0,n*n), byrow = TRUE)
  newMat <- diag(k,nrow=n)
  newMat[abs(row(newMat)-col(newMat))==1]<-1
  newMat
}
```

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){
  floor(alpha/90)%%4+1
}
```

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.

```
weekday<-function(day,month,year){
  month <- month-2
  if (month<=0){
```

```r
    month <- month+12
    year <- year-1
  }
 century<-as.integer(substring(as.character(year*100),1,2))
 year <- year%%100
 vcs <-floor(2.6*month-0.2)+day+year+year%/%4+century%/%4-2*century
 c("Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday")[1+vcs%%7]
}
weekday(03,14,1996)
```

```
## [1] "Monday"
```

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

```r
weekday2<-function(day,month,year){
  new<-month<=2
  month<-month-2+12*new
  year<-year-new
  century<-as.integer(substring(as.character(year*100),1,2))
 year <- year%%100
 vcs <-floor(2.6*month-0.2)+day+year+year%/%4+century%/%4-2*century
 c("Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday")[1+vcs%%7]
 new
 }
```

8.

(a)

```r
testloop<-function(n){
  x<-rep(NA,n-1)
    x[1]<-1
    x[2]<-2
  for(j in 3:(n-1))
    x[j]<-x[j-1]+2/x[j-1]
  x
}
```

(b)

```r
testloop2<-function(yVec){
  n<-length(yVec)
  if(n<=0){0}
  else{sum(exp(1:n))}
}
```

9.

(a)

```r
quadmap<-function(start,rho,niter){
  x <- rep(NA,niter)
   x[1]<-start
  for(n in 1:(niter-1)){
```

```r
    x[n+1]<-rho*x[n]*(1-x[n])
  }
  x
  }
tmp <-quadmap(start=0.95,rho=2.99,niter=500)
```

(b)

```r
quadmap2<-function(start,rho){
  x1 <- start
  x2 <- rho*x1*(1-x1)
  n <- 1
  while(abs(x2-x1)>=0.02){
    x1 <- x2
    x2 <- rho*x1*(1-x1)
    n<- n+1
  }
  n
}
tmp<-quadmap2(0.95,2.99)
```

10

(a)

```r
tmpFn<-function(xVec){
  len <- length(xVec)
  diff<- xVec-mean(xVec)
  r1 <- sum(diff[2:len]*diff[1:(len-1)])/sum(diff^2)
  r2 <- sum(diff[3:len]*diff[1:(len-2)])/sum(diff^2)
  list(r1=r1,r2=r2)
}
vecs<-seq(2, 56, by=3)
tmpFn(vecs)
```

```
## $r1
## [1] 0.8421053
##
## $r2
## [1] 0.6859649
```

(b)

```r
tmpFn<-function(xVec,k){
  len<-length(xVec)
  diff<-xVec-mean(xVec)
  div<-sum(diff^2)
  tmpfn<-function(i){sum(diff[(i+1):len]*diff[1:(len-i)])/div}
  c(1,sapply(1:k,tmpfn))
}
vecs<-seq(2,56,by=3)
tmpFn(vecs,3)
```

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