Exercises 3. Simple R Functions

Lauren VanValkenburg

1/31/2018

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)))
}
```

and now `tmpFn2`

```
tmpFn2 <- function(xVec2)
{
  n = length(xVec2)
  return(xVec2^(1:n)/(1:n))
}
```

(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))
}
```

---

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)
{
  n <- length(xVec)
  (xVec[-c(n-1,n)]+xVec[-c(1,n)]+xVec[-c(1,2)])/3
}
```

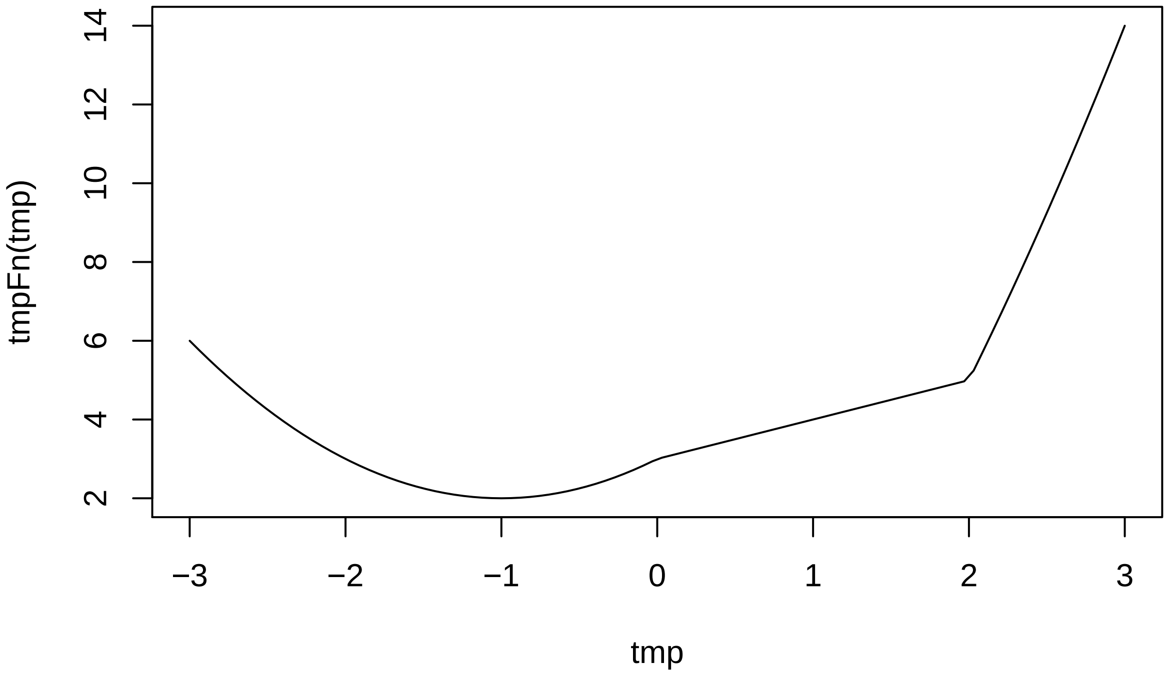
---

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(x)
{
  ifelse(x<0, x^2+2*x+3, ifelse(x<2, x+3, x^2+4*x-7))
}
tmp <- seq(-3, 3, len=100)
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:

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

```
tmpFn <- function(matrix)
{
  matrix[matrix%%2==1] <- 2*matrix[matrix%%2==1]
```

```
  matrix
}
```

---

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

---

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
}
```

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.

```
weekday <- function(day,month,year)
{
  month <- month-2
  if(month<=0){
    month <- month + 12
    year <- year -1
  }
  cc <- year%%100
  year <- year%%100
  tmp <- floor(2.6*month-.2)+day+year+year%%4+cc%%4-2*cc
}
```

(b) Does your function work if the input parameters day, month, and year are vectors with the same length and valid entries? We need to remove the if statement for it to work with vectors

```
weekday <- function(day, month, year)
{
  x <- month <=2
  month <- month-2+12*x
  year <- year-x
  cc <- year%%100
  year <- year%%100
  tmp <- floor(2.6*month-.2)+day+year+year%%4+cc%%4-2*cc
}
```

9.

Solution of the difference equation $x_n = rx_{n-1}(1 - x_{n-1})$, with starting value x1. (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 donates n, start donates x1, and rho donates r.

```
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 iterations needed to get $|x_n - x_{n-1}| < 0.02$ So this function has only 2 arguments: start and rho.

```
quad2 <- 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
}
```
