Simple R Functions

Han Nguyen

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))
}
```

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
}
```

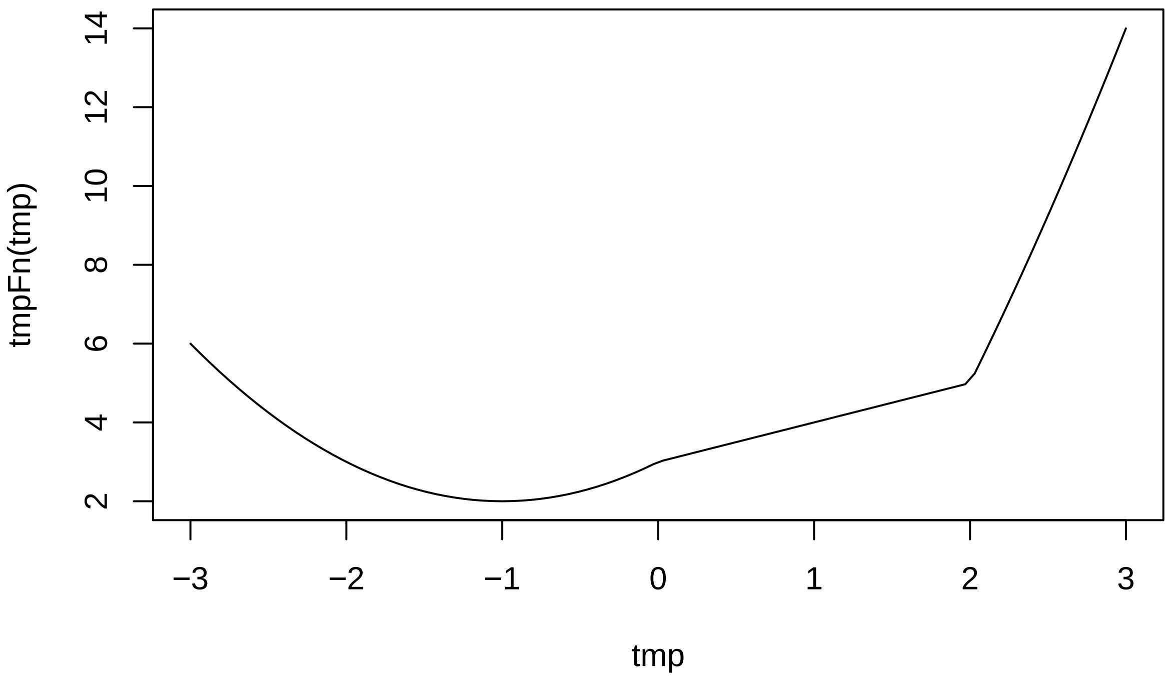
Try out your function. `tmpFn`(c(1:5,6:1))

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

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

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

```
weekday <- function(day, month, year){
  month <- month - 2
  if(month <= 0){
    month <- month + 12
```

```
    year <- year - 1
  }
  c <- year%/%100
  year <- year%/%100
  x <- floor(2.6*month - 0.2) + day + year + year%/%4 + c%/%4 - 2*c
  c("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday")[1 + x%%7]
}
```

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

The function doesn't work on vectors.

8

a) Suppose $x_0 = 1$ and $x_1 = 2$ and $x_j = x_{j-1} + \frac{2}{x_{j-1}}$ for j = 1,2,. . . . Write a function `testLoop` which tkes the single argument `n` and returns the first `n - 1` values of the sequence $x_{j_{j \geq 0}}$: that means the values of $x_0, x_1, x_2, ..., x_{n-2}$.

```
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) Now write a function `testLoop` 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`.

```
testLoop2 <- function(yVec){
  sum(exp(seq(along = yVec)))
}
```

9

Solution of the difference equation $x_n = rx_{n-1}(1 - x_{n-1})$, with starting values 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.

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

10

a) Given a vector $(x_1, ..., x_n)$, the sample autocorrelation of lag k is defined to be $x_k = \frac{\sum_{i=k+1}^{n}(x_i - \overline{x})(x_{i-k} - \overline{x})}{\sum_{i=1}^{n}(x_i - \overline{x})^2}$
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.

```
tmpFn <- function(xVec){
  x <- xVec - mean(xVec)
  denominator <- sum(x^2)
  n <- length(xVec)
  r1 <- sum(x[2:n] * x[1:(n - 1)]) / denominator
  r2 <- sum(x[3:n] * x[1:(n - 2)]) / denominator
  list(r1 = r1, r2 = r2)
}
tmpFn(c(2:56, by=3))
```

```
## $r1
## [1] 0.8547495
##
## $r2
## [1] 0.8045096
```

b) 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 $(r_0 = 1, r_1, ..., r_k)$.

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