Simple R Functions

Yue Wu

Feburary 1, 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

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

******

Here is `tmpFn3`

```
tmpFn3 <- function(x, n){
  return(sum(1+x^(1:n)/(1:n)))
}

## simple example
```

```r
b <- tmpFn3(5,5)
b
```

```
## [1] 845.4167
```

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:

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

******

Here is `tmpFn`

```
tmpFn <- function(xVec){
  l = length(xVec)
  return(sum((xVec[1:(l-2)]+xVec[2:(l-1)]+xVec[3:l])/3))
}

## example
b <- tmpFn(c(1:5,6:1))
b
```

```
## [1] 33.33333
```

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 $-3 < x < 3$. ******

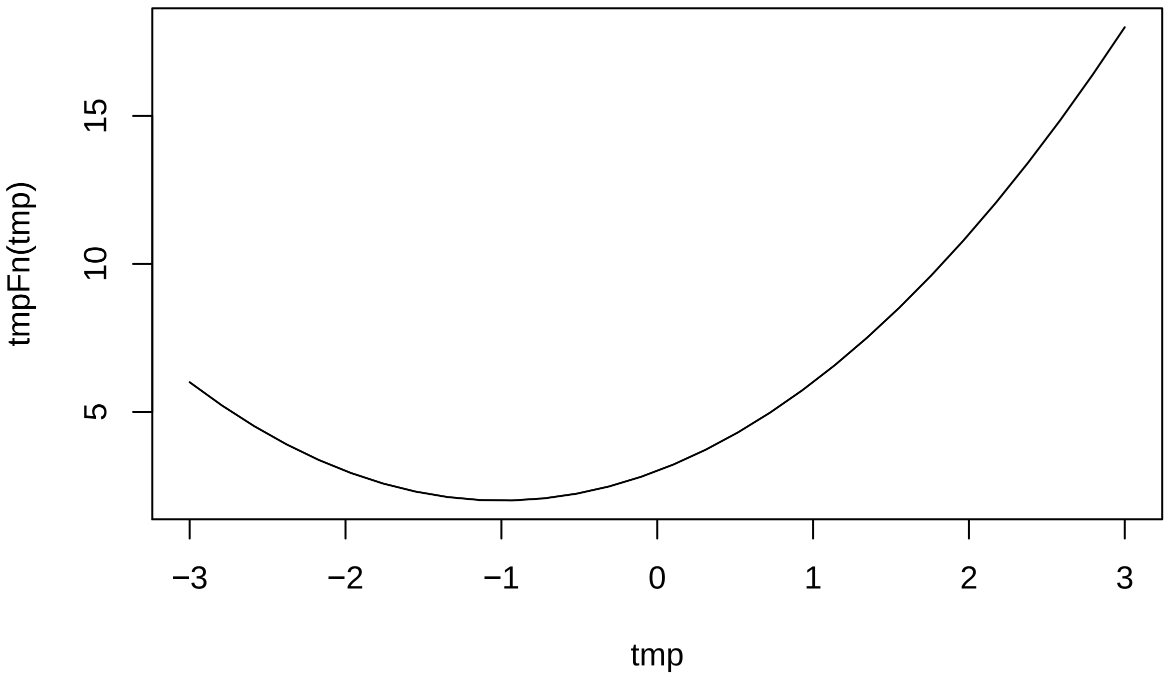
Here is `tmpFn`

```
tmpFn <- function(xVec){

  if (xVec<0) {
    return(xVec^2+2*xVec+3)
  }
  else if ((xVec>=0) & (xVec<2)){
    return (xVec+3)
  }
  else if (xVec>=2) {
    return (xVec^2+4*xVec-7)
  }
}

## example
tmp <- seq(-3, 3, len=30)
plot(tmp, tmpFn(tmp), type="l")
```

```
## Warning in if (xVec < 0) {: 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:

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

******

Here is `dodd`

```
dodd <- function(mat){
  mat[mat%%2==1] <- mat[mat%%2==1]*2
  return(mat)
}

## example
tmp <- matrix(c(1,1,3,5,2,6,-2,-1,-3),ncol=3,nrow=3,byrow=TRUE)
dodd(tmp)
```

```
##      [,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:

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

******

Here is `mat`

```
mat <- function(n,k){
  m <- matrix(0,ncol=n,nrow=n)
  m[col(m)==row(m)] <- k
  m[col(m)==row(m)+1] <- 1
  m[col(m)==row(m)-1] <- 1
  return(m)
}

## example
tmp <- mat(5,3)
tmp
```

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

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. ******

Here is `quad`

```
quad <- function(a){
  if (((a+90)%/%90)%%4 ==0) {
    return(4)
  }
  else{
    return(((a+90)%/%90)%%4)
  }
}
## example
tmp <- quad(271)
tmp
```

```
## [1] 4
```

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

Here is `weekday`

```r
weekday <- function(day,month,year){
  month <- month - 2
  year[month<=0] <- year-1
  month[month<=0] <- month+12
  c <- year %/% 100
  y <- year - 100*c
  d <- ((2.6*month-0.2)%/%1+day+y+y%/%4+c%/%4-2*c)%%7
  return(c("Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday")[1+d])
}
## example
weekday(21,07,1963)
```

```
## [1] "Sunday"
```

```r
## b: with vector input
weekday(c(21,07,1963),c(22,07,1963),c(31,01,2018))
```

```
## [1] "Friday"   "Saturday" "Saturday"
```

8. (a) Suppose $x_0 = 1$ and $x_1 = 2$ and

$$x_j = x_{j-1} + \frac{2}{x_{j-1}}$$

Write a function testLoop which takes the single argument n and returns the first $n1$ values of the sequence $x_{j_j}0$: that means the values of $x_0, x_1, x_2, ..., x_{n2}$. ## (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`.

Here is `testLoop`

```
# (a)
testLoop <- function(n){
  v <- rep(NA, n-1)
  v[1] <- 1
  v[2] <- 2
  for( j in 3:(n-1) )
    v[j]= v[j-1]+2/v[j-1]
  return(v)
  }
# test example
testLoop(8)
```

```
## [1] 1.000000 2.000000 3.000000 3.666667 4.212121 4.686941 5.113659
```

```
# (b)
testLoop2 <- function(yVec){
  n= length(yVec)
  v <- c(1:n)
  v[1:n] <- exp(v[1:n])
  return(sum(v))
  }
# test example
testLoop2(c(1:8))
```

```
## [1] 4714.224
```

9. Solution of the difference equation $x_n = rx_{n1}(1x_{n1})$, with starting value x_1.

(a) Write a function `quadmap( start, rho, niter )` which returns the vector $(x_1, ..., x_n)$ where $xk = rx_{k1}(1x_{k1})$ and

`niter` denotes n, `start` denotes x_1, and `rho` denotes r. Try out the function you have written: • for $r = 2$ and $0 < x1 < 1$ you should get $x_n 0.5$ as $n\infty$. • 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 $|x_n x_{n1}| < 0.02$. So this function has only 2 arguments: $start$ and rho. $(Forstart = 0.95$ and $rho = 2.99$, the answer is 84.) ******

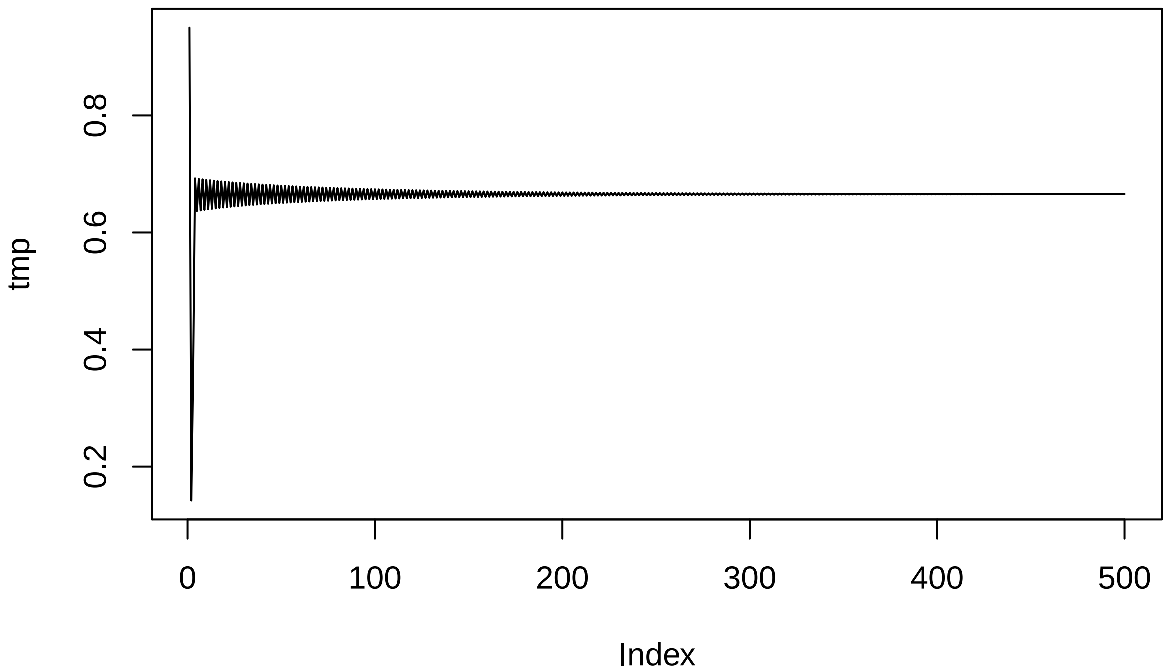
Here is `quadmap`

```
# (a)
quadmap <- function(start, rho, niter)
  {
  v <- rep(NA,niter)
  v[1] <- start
```

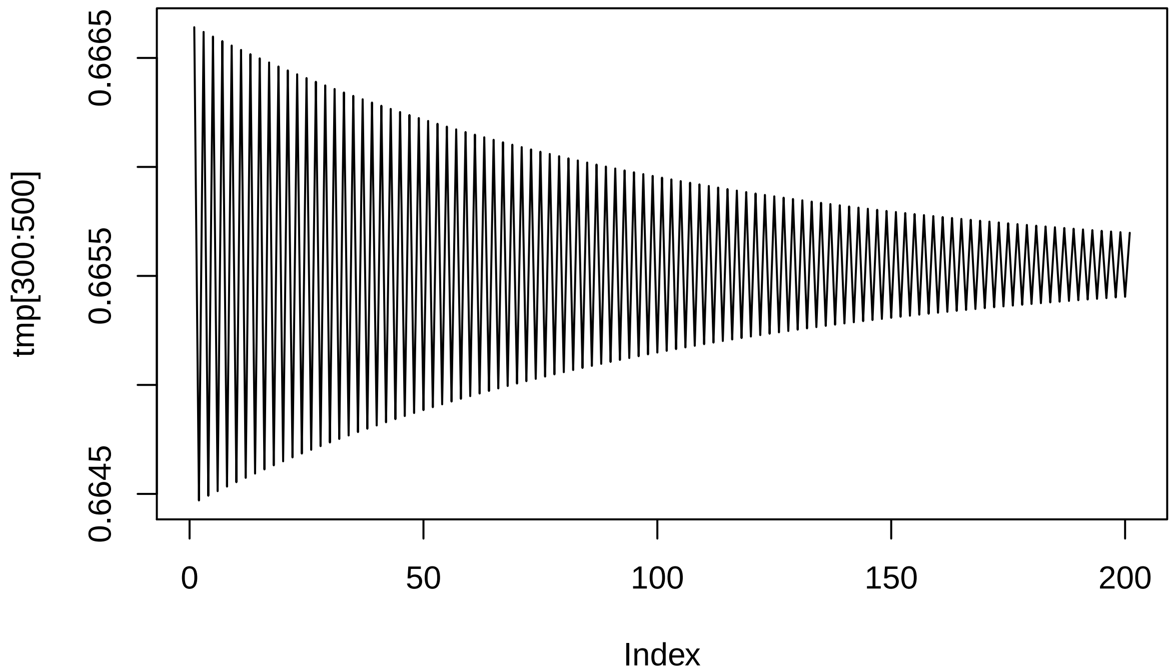
```r
  for(i in 1:(niter-1)) {
  v[i + 1] <- rho*v[i]*(1-v[i])
  }
  return(v)
  }
# test example
quadmap(0.5,2,100)[90:100]
```

```
##  [1] 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5
```

```r
tmp <- quadmap(start=0.95, rho=2.99, niter=500)
plot(tmp, type="l")
```




```r
plot(tmp[300:500], type="l")
```




```r
# (b)
quadmap2 <- function(start, rho)
  {
  x1 <- start
  niter <- 1
  while(abs(x1-rho*x1*(1- x1))>= 0.02) {
  x1 <- rho*x1*(1- x1)
  niter <- niter+ 1
  }
  return(niter)
}
# test example
quadmap2(0.95,2.99)
```

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

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

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

Thus

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

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 r_1 and r_2 for the vector $(2, 5, 8, ..., 53, 56)$ ##(b) (Harder.) Generalise the function so that it takes two arguments: the vector `xVec` and an integer `k` which lies between 1 and $n1$ 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`.) ******

Here is `tmpFn`

```r
# (a)
tmpFn <- function(xVec)
  {
  xbar <- xVec - mean(xVec)
  xbar2 <- sum(xbar^2)
  n <- length(xVec)
  r1 <- sum( xbar[2:n] * xbar[1:(n-1)] )/xbar2
  r2 <- sum( xbar[3:n] * xbar[1:(n-2)] )/xbar2
  return(list(r1 = r1, r2 = r2))
  }
# test example
tmpFn(seq(2,56,by=3))
```

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

```r
# (b)
tmpFn <- function(x,k)
  {
  xbar <-  x - mean(x)
  xbar2 <- sum(xbar^2)
  n <- length(x)
  return(c(1,sapply(1:k, function(y) sum(xbar[(y+1):n]*xbar[1:(n-y)] )/xbar2)))
  }
# test example
tmpFn(seq(2,56,by=3),3)
```

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