

Simple R Functions

ShihChing, Huang

January 26, 2018; 20 April, 2018 revised

1.

- (a) Write functions `tmpFn1` and `tmpFn2` such that if `xVec` is the vector (x_1, x_2, \dots, x_n) , then `tmpFn1(xVec)` returns vector $(x_1, x_2^2, \dots, x_n^n)$ and `tmpFn2(xVec)` returns the vector $(x_1, \frac{x_2^2}{2}, \dots, \frac{x_n^n}{n})$.

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 function `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

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

```
tmpFn3 <- function(x,n){  
  
  n <- length(tmpFn2)  
  
  sum(  
    (x^(1:n)/(1:n))  
  )+1  
}
```

```
}
```

```
tmpFn3(a)
```

```
## [1] 25
```

2. Write a function `tmpFn(xVec)` such that if `xVec` is the vector $x = (x_1, \dots, x_n)$ then `tmpFn(xVec)` returns the vector of moving averages:

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

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

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

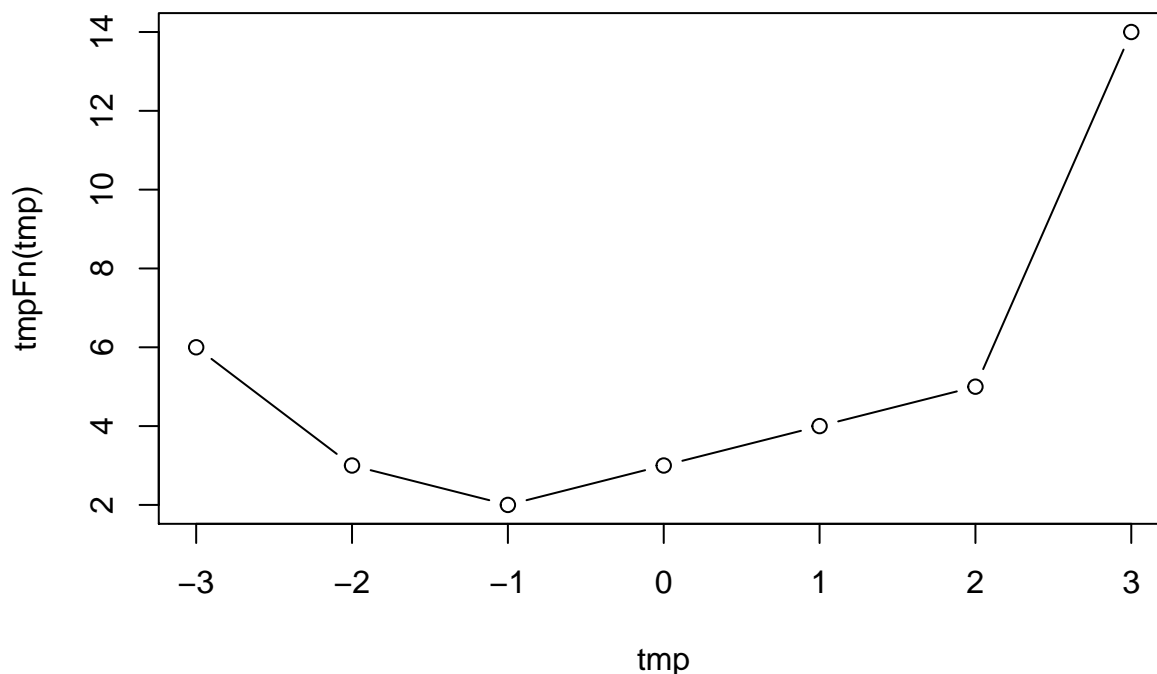
$$f(x) = \begin{cases} x^2 + 2x + 3 & \text{if } x < 0 \\ x + 3 & \text{if } 0 \leq x < 2 \\ x^2 + 4x - 7 & \text{if } 2 \leq x \end{cases}$$

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, by=1)
plot(tmp, tmpFn(tmp), type="b")
```



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:

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

```
A5 <- function(a){
  a[a%%2==1] <- 2*a[a%%2==1]
  a
}
a <- matrix(c(1,1,3,5,2,6,-2,-1,-3),nrow = 3,byrow = TRUE)
a
```

```
##      [,1] [,2] [,3]
## [1,]    1    1    3
## [2,]    5    2    6
## [3,]   -2   -1   -3
```

A5(a)

```
##      [,1] [,2] [,3]
## [1,]    2    2    6
## [2,]   10    2    6
## [3,]   -2   -2   -6
```

5. Write a function which takes 2 arguments n and k which are positive integers. It should return the $n \times n$ matrix:

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

```
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

tmpFn <- function(n, k){
  tmp <- diag(k, nr = n)
  tmp[abs(row(tmp) - col(tmp)) == 1] <- 1
  tmp
}
tmpFn(5,2)
```

```
##      [,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
```

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 quadrant 3. 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:

$$f = ([2.6m - 0.2] + k + y + [y/4] + [c/4] - 2c) \bmod 7$$

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 21/07/1963 has $m = 5$, $k = 21$, $c = 19$, $y = 63$;

the date 21/2/63 has $m = 12$, $k = 21$, $c = 19$, $y = 62$.

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
  }
}
```

```

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

```

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

It would not work with some problems, so I made this weekday2 function.

```

weekday2 <- function(day, month, year) {
  flag <- month <= 2
  month <- month - 2 + 12*flag
  year <- year - flag
  cc <- year %/% 100
  year <- year %% 100
  tmp <- floor(2.6*month - 0.2) + day + year + year %/% 4 + cc %/% 4 - 2 * cc
  c("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday")[1+tmp%7]
}

weekday2( c(27,18,21), c(2,2,1), c(1997,1940,1963) )

## [1] "Thursday" "Sunday"    "Monday"

```