Exercises 3. Simple Functions

Tiffany Cheng

1.

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

```
tmpFn1 <- function(xVec) {
  return(xVec^(1:length(xVec)))
}
tmpFn1(c(1,2,3)) # example
```

```
## [1]  1  4 27
```

```
tmpFn2 <- function(xVec2) {
  n = length(xVec2)
  return(xVec2^(1:n)/(1:n))
}
tmpFn2(c(1,2,3)) # example
```

```
## [1] 1 2 9
```

(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))
}
tmpFn3(2,3) # example
```

```
## [1] 7.666667
```

---

2. Write a function `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; for example, try `tmpFn(c(1:5,6:1))`.

```
tmpFn <- function(xVec) {
  n <- length(xVec)
  for (i in 1:n)
    MA <- (xVec[1:(n-2)]+xVec[2:(n-1)]+xVec[3:n])/3
    print(MA)
}
tmpFn(c(1:5,6:1)) # example
```

```
## [1] 2.000000 3.000000 4.000000 5.000000 5.333333 5.000000 4.000000 3.000000
## [9] 2.000000
```

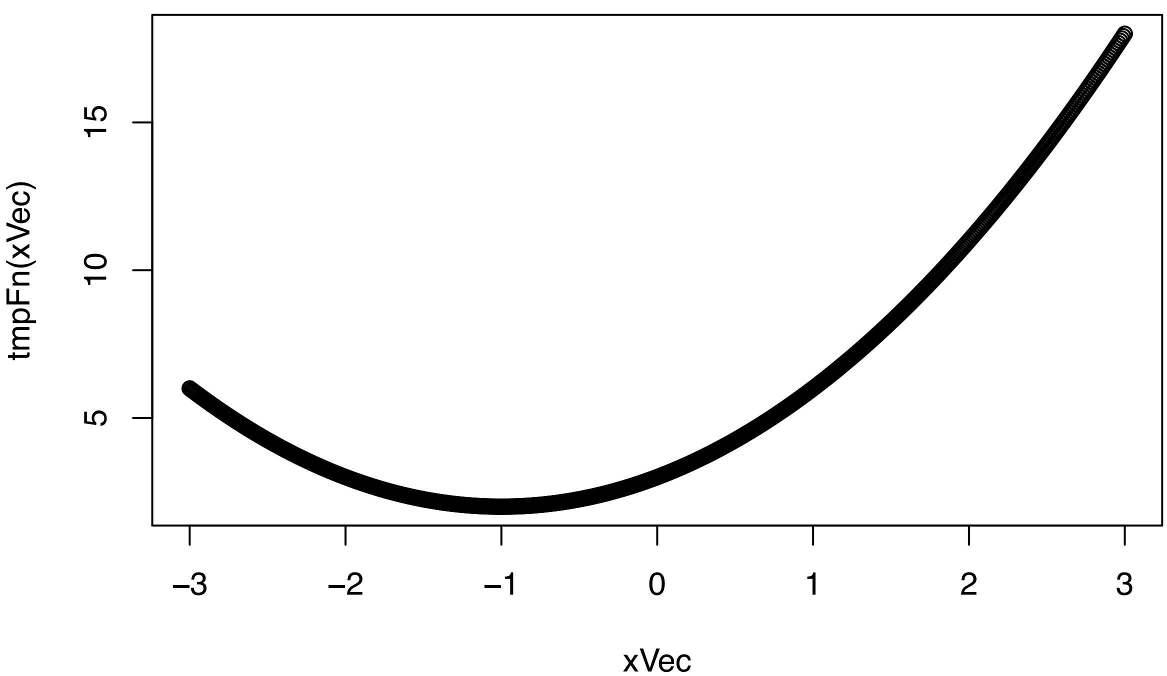
3. Consider the continuous function

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

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

tmpFn <- function(xVec) {
  if (xVec<0) {
    xVec^2+2*xVec+3
  } else if (xVec>=2) {
    xVec^2+4*xVec-7
  } else {
    xVec+3
  }
}
xVec <- seq(-3,3,by=0.01)
plot(xVec,tmpFn(xVec))
```



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:

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

Hint: First try this for a specific matrix on the Command Line.

```
double <- function(my_matrix) {
  for (row in 1:nrow(my_matrix)) {
    for (col in 1:ncol(my_matrix)) {
      if (my_matrix[row,col]%%2==1) my_matrix[row,col] <- my_matrix[row,col]*2
    }
  }
print(my_matrix)}
my_matrix <- matrix(c(1,1,3,5,2,6,-2,-1,-3), nrow = 3, byrow = TRUE)
double(my_matrix) # example
```

```
##      [,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:

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

Hint: First try to do it for a specific case such as n = 5 and k = 2 on the Command Line.

```
kdiagonal <- function(n,k) {
  matN <- matrix(rep(0,n^2), nrow = n, byrow = TRUE)
  matN[abs(row(matN)-col(matN))==1] <- 1
  for (row in 1:nrow(matN)) {
    for (col in 1:ncol(matN)) {
      if (row==col) matN[row,col] <- k
    }
  }
```

```r
print(matN)}
kdiagonal(5,2) # example
```

```
##      [,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 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.

```r
quadrant <- function(alpha) {
  if (0<=alpha & alpha<90) {
    1
  } else if (90<=alpha & alpha<180) {
    2
  } else if (180<=alpha & alpha<270) {
    3
  } else {
    4
  }
}
quadrant(45) # example
```

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

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 21/07/1963 has $m = 5, k = 21, c = 19, y = 63$;
whilst the date 21/2/1963 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.

```r
weekday <- function(day,month,year) {
  k <- day
  m <- month-2
  y <- year
  if (m<=0) {
    m <- m+12
    y <- y-1 }
  c <- trunc(y/100)
  y <- y%%100
  f <- (trunc(2.6*m-0.2)+k+y+trunc(y/4)+trunc(c/4)-2*c)%%7+1
  days <- c("Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday")
  days[f]
}
weekday(30,1,1975) # example
```

```
## [1] "Thursday"
```

```r
weekday(30,3,1975) # example
```

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

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

```r
# to check whether the entries are valid or not, more code will need to be added at the top
weekday.valid <- function(day,month,year) {
  if (day<0|day>31) {
    break
  } else if (month<0|month>12) {
    break
  } else if (year<0) {
    break
  } else {
    k <- day
  m <- month-2
  y <- year
  if (m<=0) {
    m <- m+12
    y <- y-1 }
  c <- trunc(y/100)
  y <- y%%100
  f <- (trunc(2.6*m-0.2)+k+y+trunc(y/4)+trunc(c/4)-2*c)%%7
  days <- c("Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday")
  days[1+f]
  }
}
```

8.

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

```r
testLoop <- function(n) {
  if (n<=3) {
    break
  } else {
    sequence <- rep(0,n-1)
  sequence[1] <- 1
  sequence[2] <- 2
  for (i in 3:(n-1))
    sequence[i] <- sequence[i-1]+2/(sequence[i-1])
    sequence
  }
}
testLoop(5) # example
```

```
## [1] 1.000000 2.000000 3.000000 3.666667
```

(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`.

```r
testLoop2 <- function(yVec) {
  sum <- 0
  n <- length(yVec)
  for (j in 1:n)
    sum <- sum+sum(exp(j))
    sum
}
testLoop2(c(1,2,3)) # example
```

```
## [1] 30.19287
```

---

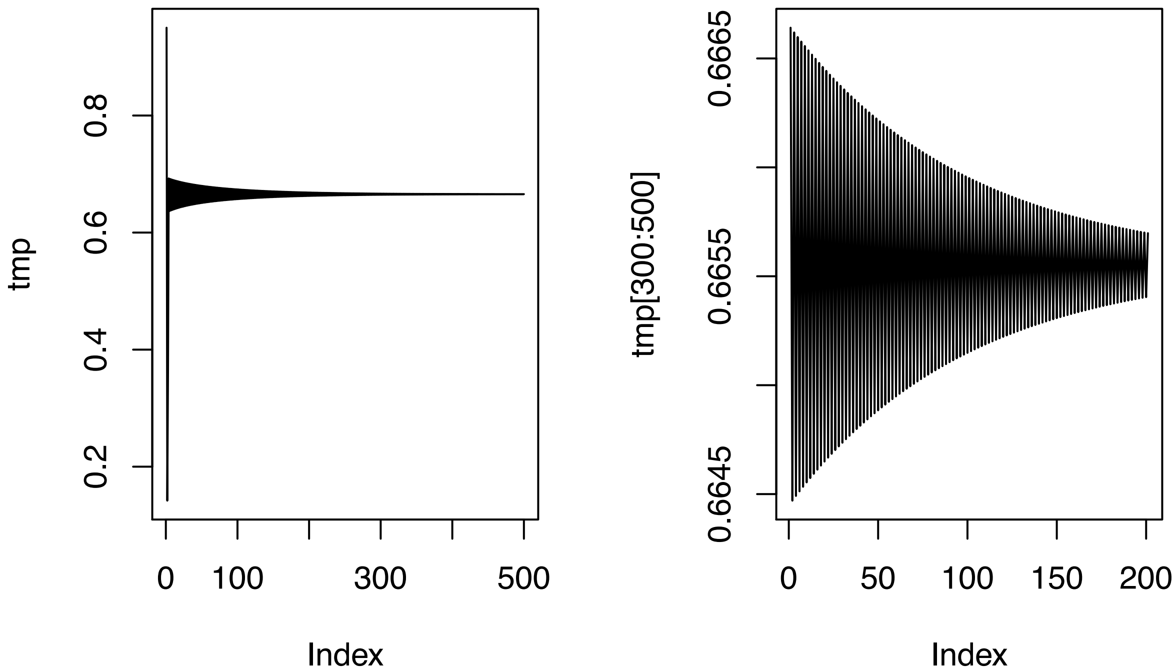
9. *Solution of the difference equation* $x_n = rx_{n-1}(1 - x_{n-1})$, with starting value x_1.

(a) Write a function `quadmap(start, rho, niter)` which returns the vector $(x_1, ..., x_n)$ where $x_k = rx_{k-1}(1 - x_{n-1})$ 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 $x_n \to 0.5$ as $n \to \infty$. • 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")`

```r
quadmap <- function(start,rho,niter) {
  n <- niter
  vec <- rep(0,n)
  vec[1] <- start
  r <- rho
  for (i in 1:(n-1))
    vec[i+1] <- r*vec[i]*(1-vec[i])
    vec
}
quadmap(0.3,2,5) # for the first bullet point
```

```
## [1] 0.3000000 0.4200000 0.4872000 0.4996723 0.4999998
```

```r
tmp <- quadmap(start=0.95,rho=2.99,niter=500) # for the second bullet point
par(mfrow=c(1,2))
plot(tmp, type="l")
plot(tmp[300:500], type="l")
```



(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`. (For `start=0.95` and `rho=2.99`, the answer is 84.)

```r
quadmap.iterations <- function(start,rho) {
  x1 <- start
  x2 <- rho*x1*(1-x1)
  count <- 1
  while (abs(x2-x1)>=0.02) {
    x1 <- x2
    x2 <- rho*x1*(1-x1)
    count <- count+1
  }
print(count)}
quadmap.iterations(start=0.95,rho=2.99) # example
```

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

10.

(a) Given a vector $(x_1, ..., x_n)$, 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).

```
tmpFn <- function(xVec) {
  xbar <- mean(xVec)
  n <- length(xVec)
  square <- sum((xVec-xbar)^2)
  r1 <- 0
  for (i in 2:n) {
    r1 <- r1+(sum((xVec[i]-xbar)*(xVec[i-1]-xbar))/square)
  }
  r2 <- 0
  for (j in 3:n) {
    r2 <- r2+(sum((xVec[j]-xbar)*(xVec[j-2]-xbar))/square)
  }
print(list(r1,r2))}
tmpFn(seq(2,56,by=3)) # example
```

```
## [[1]]
## [1] 0.8421053
##
## [[2]]
## [1] 0.6859649
```

(b) (Harder.) Generalize the function so that it takes two arguments: the vector `xVex` 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 $(r_0 = 1, r_1, ..., r_k)$. 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`.)

```
tmpFn.general <- function(xVec,k) {
  xbar <- mean(xVec)
  n <- length(xVec)
  square <- sum((xVec-xbar)^2)
  values <- c(1,rep(0,k))
  for (k in 2:(k+1)) {
    values[k] <- sum((xVec[k]-xbar)*(xVec[k-1]-xbar))/square
  }
print(values)}
tmpFn.general(seq(2,56,by=3),3) # example
```

```
## [1] 1.00000000 0.12631579 0.09824561 0.07368421
```