MA 415: Assignment 4

Rachel Feingold

2/5/2017

Problem 1: Warm Up a. Write a function which takes a numeric vector x and returns a named list containing the mean, median and variance of the values in x

```
stats <- function(x, plot = FALSE)
{
  if(is.numeric(x)==FALSE) # need to filter out character vectors!\
    stop("Invalid Input. summ() only accepts numeric vectors")
  st =  list(Mean=mean(x), Median=median(x), Variance =var(x))
  if(plot==TRUE)
  {
    plot(x)
    abline(h=st[["Mean"]])
    }
   return (st)
}

stats(1:9)
```

```
## $Mean
## [1] 5
##
## $Median
## [1] 5
##
## $Variance
## [1] 7.5
```

b. Write a function with arguments x and n which evaluates sum(i=0 to n):(($e^{-x)(x}$i))/ (i!)

```
partb <- function(x,n)
{
  for(i in 0:n){
    s <- sum(((exp(-x)*(x^i))/ (factorial(i))))
    return(s)
  }
}
partb(2,2)
```

```
## [1] 0.1353353
```

c. Write a fucntion which goes through every entry in a list, checks whether it is a character vector

```
is.char <- function(l)
{
  if (is.list(l)==FALSE)
    stop("Invalid Input. Must enter a list")
for (x in l){
  if(is.character(x)==TRUE)
    print(x)
}
```

```
}
list <- list(c(4:6), "cat", "dog", 6, "pie")
is.char(list)
```

```
## [1] "cat"
## [1] "dog"
## [1] "pie"
```

```
list <- list("cat", 5, 7, "dog")
is.char(list)
```

```
## [1] "cat"
## [1] "dog"
```

d. Write a function with an argument k which stimulates a symmetric walk on the intergers, stopping when the walk reaches k (or -k). A random walk on the integers is a sequence X1, X2, . . . , with X0 =0 and Xi = Xi+1 + Di where Di are independent with P(Di=1)=P(Di=-1)= 0.5

```
r.walk <- function(k)
{
  r=0
  print(r)
  x <-  c(-1,1)
  while (abs(r)<k){

    y <-  sample(x,1)
    r <- r + y
    print(r)
  }
  }
r.walk(4)
```

```
## [1] 0
## [1] 1
## [1] 2
## [1] 1
## [1] 2
## [1] 1
## [1] 2
## [1] 1
## [1] 0
## [1] -1
## [1] 0
## [1] -1
## [1] -2
## [1] -1
## [1] 0
## [1] -1
## [1] 0
## [1] 1
## [1] 2
## [1] 1
## [1] 0
## [1] 1
## [1] 2
## [1] 3
```

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

```
r.walk(2)
```

```
## [1] 0
## [1] 1
## [1] 2
```

Problem 2. Moving Averages

a. Write a function to calculate the moving averages of length 3 of a vector (x1,. . . ,xn)ˆT. (The function returns (z1,. . . ,zn)ˆT) where zi = (1/3) (xi+xi+1+xi+2), i=1, ..., n-2 Call this function ma3

```
ma3 <- function(xVec){
  if (is.numeric(xVec)==FALSE){
    stop("Invalid Input. Please enter a numeric vector")}
  n <- length(xVec)
  z <- ( xVec[1:(n-2)]+xVec[2:(n-1)]+xVec[3:n])/3
  return(z)
}
ma3(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
```

b. Write a function which takes two arguments, x and k, and calculates the moving average of x for length k

```
ma.k <- function(x,k){
  if (is.numeric(x)==FALSE){
    stop("Invalid Input. Please enter a numeric vector")}
  z <- filter(x,rep(1/k,k))
if (k==1){
  stop("Invalid Input. K must be greater than 1.")
}
if ((length(k)<length(x))==TRUE){
  stop("Invalid Input. K cannot be larger than or equal to x.")
}
  return(z)
}
#ma.k(c(1,2,3,4,5,6),3)

#ma.k(c(1:3),1)
```

c. How does your function behave if k is larger than (or equal to) the length of x? There is an error message that says: Error in filter(x, rep(1/k, k)) : 'filter' is longer than time series.

d. You should return an error, use the stop() function. Are there other choices? Error message included.

e. How does your function behave if k=1? What should you do? Fit it if necessary? If k =1, then the error message pops up saying that k cannot be larger than or equal to x. To fix it, we should say that k must be greater than 1.

Problem 3. Optional Plot

Continuous functions: f(x) = xˆ2 + 2x+3 if x<0 = x+3 if 0 <=x<2 = xˆ2 +4x-7 if x>=2

Write a function which takes a vector and returns a vector of the values f(x). The function should be valid for inputs where -4 < x < 4. The function should check the input for validity and offer the user the option of plotting the values the function returns

```
prob2 <- function(x, plot = FALSE)
  {
  if (-4 < x & x < 0){
    z = ((x^2)+2*x+3)
    return(z)
  }
  if (0 <= x & x < 2){
    z = (x+3)
    return(z)
  }
  if (2 <= x & x < 4){
    z = ((x^2)+4*x-7)
    return(z)
  }
  if(plot==TRUE){
    plot(x)
    }
  else{
    stop("Input Invalid. Please enter a number between -4 and 4.")
  }
}
prob2(0)
```

```
## [1] 3
```

```
#prob2(-4)
prob2(2,TRUE)
```

```
## [1] 5
```

```
prob2(1)
```

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

```
prob2(3)
```

```
## [1] 14
```

```
#prob2(5)
```

Problem 4. Matrix Input

Write a function which takes a single argument, a matrix or an arguments that can be coerced into a matrix and return a matrix which is the same as the function of the argument, but every odd number is doubled

```
double <- function(mat)
{
  x = nrow(mat)
  y = ncol(mat)
  for(i in 1:x){
    for(j in 1:y){
      if(mat[i,j]%%2==1){
      mat[i,j] = mat[i,j]*2
      }
      }
  }
  return(mat)
```

```
}
matrix1 <- matrix(c(2,-4,-5,7), nrow = 4,ncol = 4 ,byrow = TRUE)
double(matrix1)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    2   -4  -10   14
## [2,]    2   -4  -10   14
## [3,]    2   -4  -10   14
## [4,]    2   -4  -10   14
```