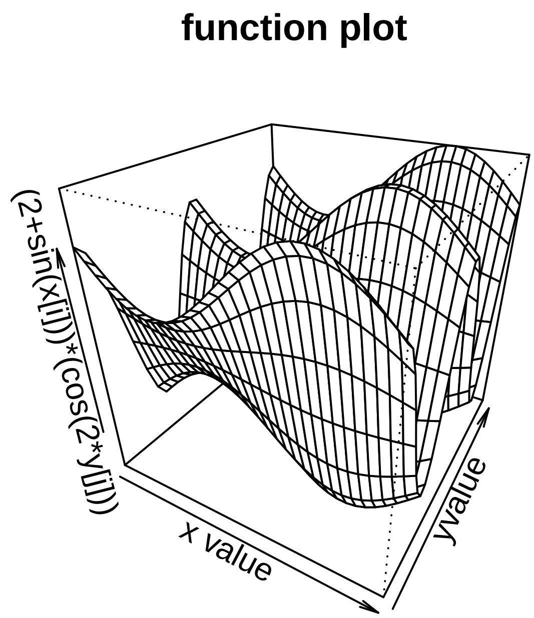
Assignment 3

Bradley Assaly-Nesrallah

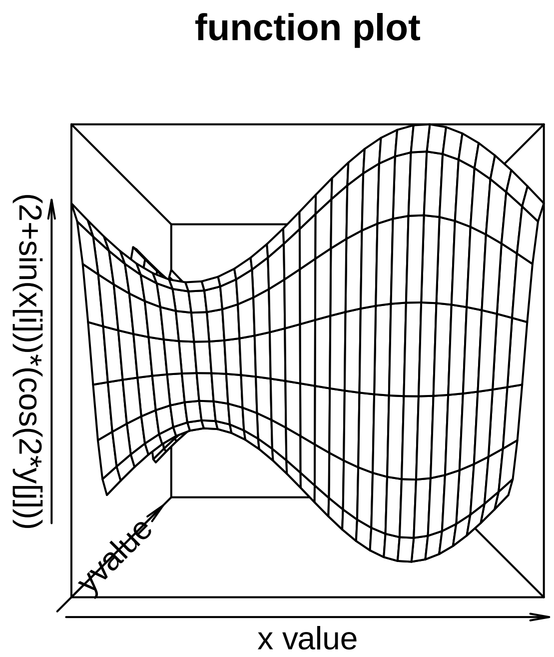
Question 1

We Use the persp function to plot (2+sin(x))(cos(2y)) with values ranging from -pi to pi

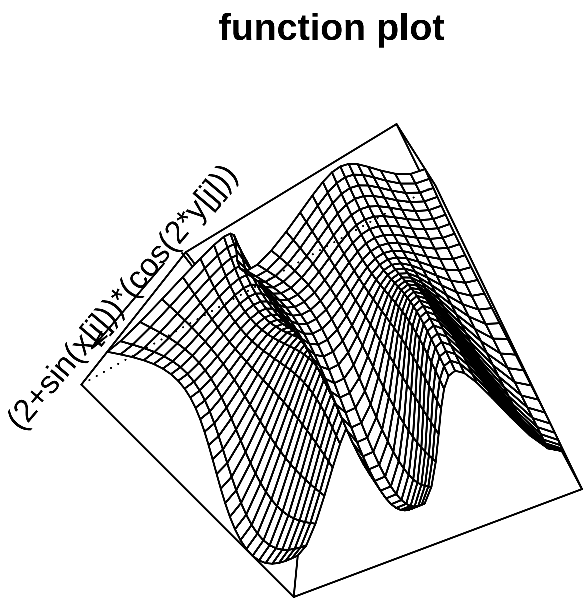
```
# generate x,y values from -pi to pi
x<- seq(-pi,pi, length =30)
y<- seq(-pi,pi, length =30)
#create a matrix for the values
m= matrix(NA, nrow=30, ncol=30)
#populate matrix with the function results
for (i in 1:30){
  for (j in 1:30){
    m[i,j] = (2+sin(x[i]))*(cos(2*y[j]))
  }
}
#plot persp with theta =30 phi =30
persp(m, theta=30, phi=30, xlab="x value",ylab="yvalue",zlab="(2+sin(x[i]))*(cos(2*y[j]))",
      main="function plot")
```




```r
#plot persp with theta =0 phi =0
persp(m, theta=0, phi=0,xlab="x value",ylab="yvalue",zlab="(2+sin(x[i]))*(cos(2*y[j]))",
      main="function plot")
```




```
#plot persp with theta =60 phi =120
persp(m, theta=60, phi=120,xlab="x value",ylab="yvalue",zlab="(2+sin(x[i]))*(cos(2*y[j]))",
      main="function plot")
```



Question 2 ============

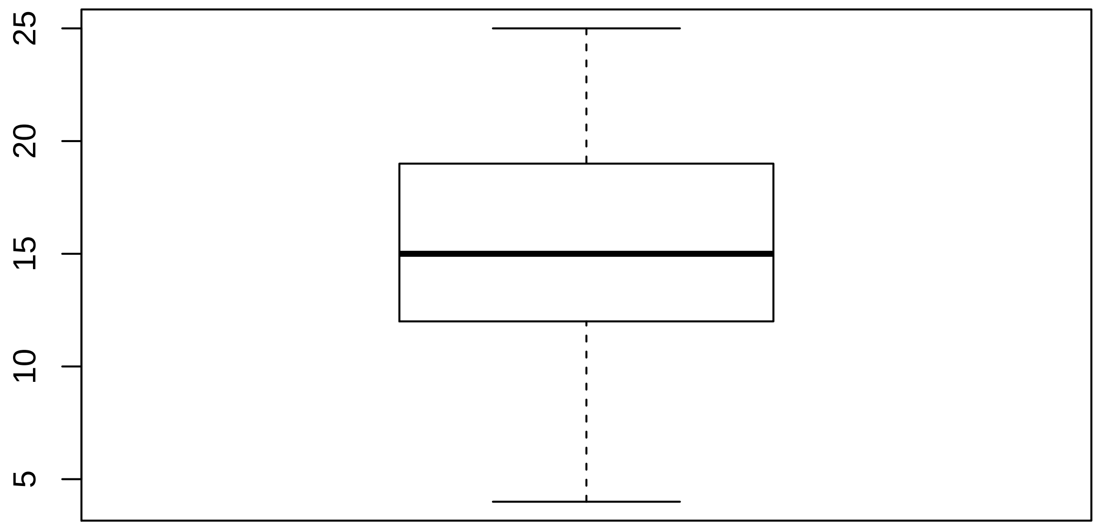
```
#define function to calculate min sum of 2^i and i^3 for input n
my.computemax=function(n){
  s1 =c()
  s2 =c()
  #generate the two sequences for given n
  for( n_i in 1:n){
    s1 =append(s1,2^n_i)
    s2 =append(s2, n_i^3)
  }
  #compute min and max sum using pmin pmax resp
  minsum= sum(pmin(s1,s2))
  maxsum= sum(pmax(s1,s2))
  #return both results in a list
  results = list(minsum,maxsum)
  results
}
#create sequence and use sapply to exceute for the given function
n =seq(200,5000,by= 600)
sapply(n, my.computemax)
```

```
##      [,1]         [,2]          [,3]         [,4]         [,5]
## [1,] 404008996    102656158996  961772488996 4.004001e+12 1.143319e+13
## [2,] 3.213876e+60 1.333603e+241 Inf          Inf          Inf
##      [,6]         [,7]         [,8]        [,9]
```

```
## [1,] 2.623079e+13 5.215584e+13 9.3745e+13 1.563125e+14
## [2,] Inf          Inf          Inf        Inf
```

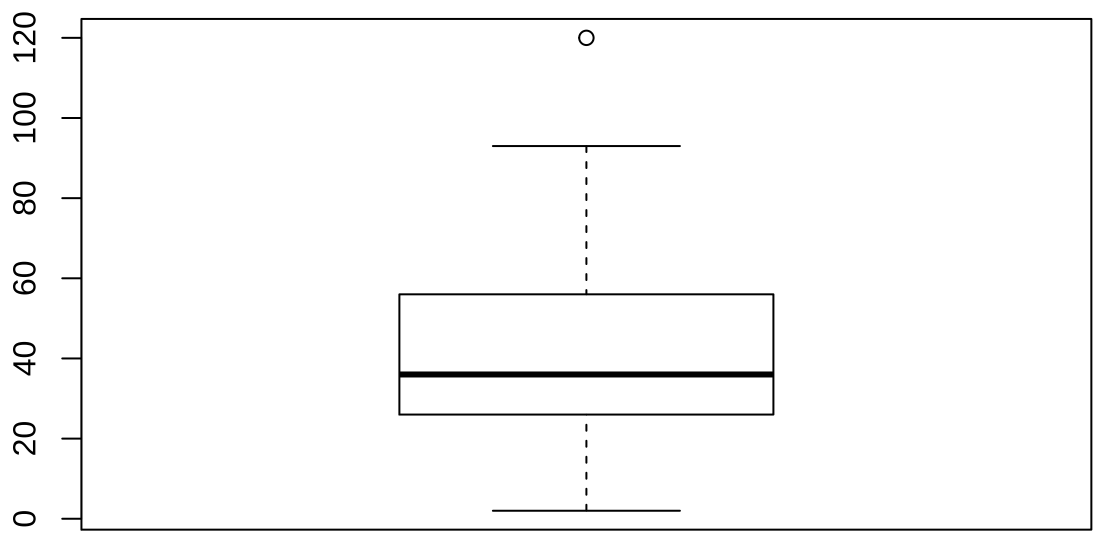
Question 3

```
#define function IQR outliers to compute the IQR, find outlires and boxplot

UQR.outliers=function(x){
  #error checking for the function for na values, and correct type
  if (any(is.na(x))){
    print("Warning, NA values in x")
    return
  }
  if (length(x)<4){
    print("input must have values")
    return
  }
  if (is.null(x)){
    print("input must have values")
    return
  }
  if (!is.numeric(x)){
    print("input must be numeric")
    return
  }
  #compute quantiles for iqr
  q3 = quantile(x,0.75)
  q1 = quantile(x,0.25)
  iqr = q3-q1
  outliers = c()
  n = length(x)
  #populate list of outliers using the criteria
  for (i in 1:n){
    if (x[i] < q1-1.5*iqr){
      outliers =append(outliers,x[i])
    }
    if (x[i] > q3+1.5*iqr){
      outliers =append(outliers,x[i])
    }
  }
  #boxplot for x and list results of the function in a list
  boxplot(x)
  results = list(iqr, outliers)
  results
  }

#compute test cases for speed and distance of cars, and a fail case with lengthn =1
UQR.outliers(cars[,1])
```




```
## [[1]]
## 75%
##   7
##
## [[2]]
## NULL
```

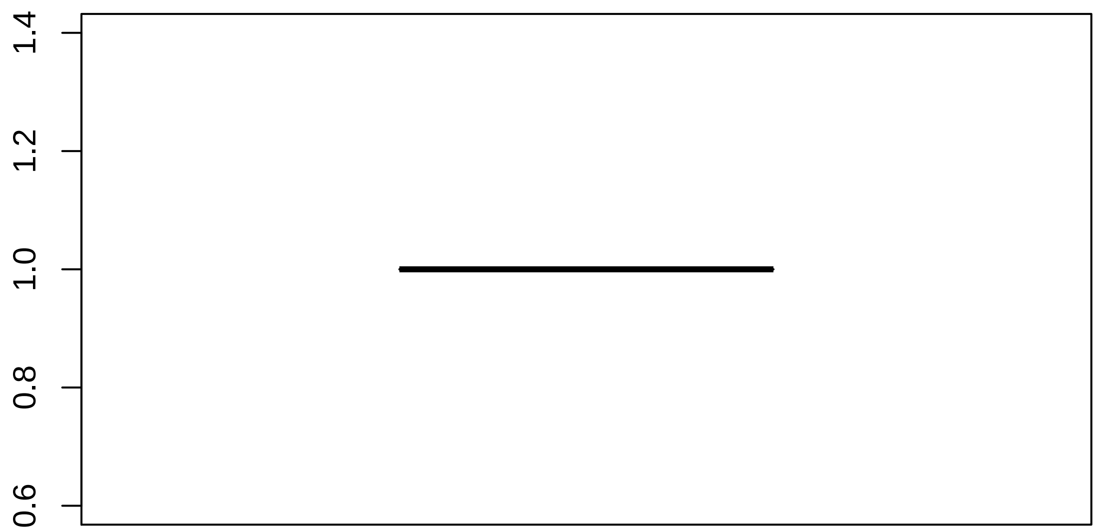
```r
UQR.outliers(cars[,2])
```




```
## [[1]]
## 75%
##  30
##
## [[2]]
## [1] 120
```

```r
UQR.outliers(c(1))
```

```
## [1] "input must have values"
```

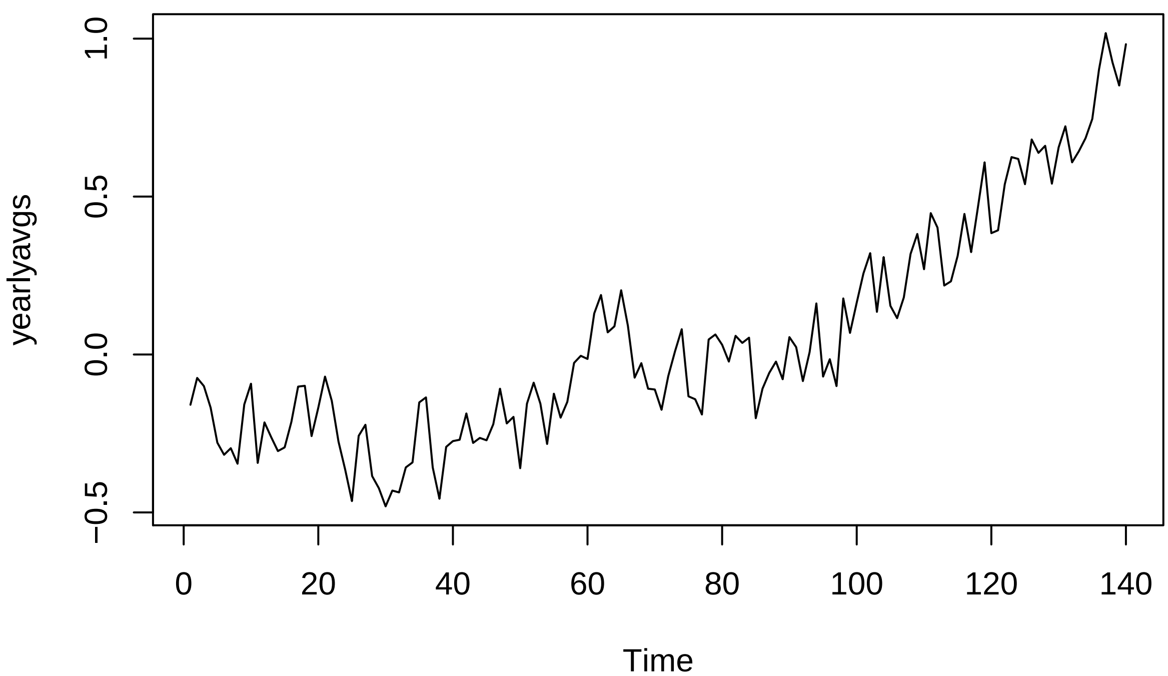



```
## [[1]]
## 75%
##   0
##
## [[2]]
## NULL
```

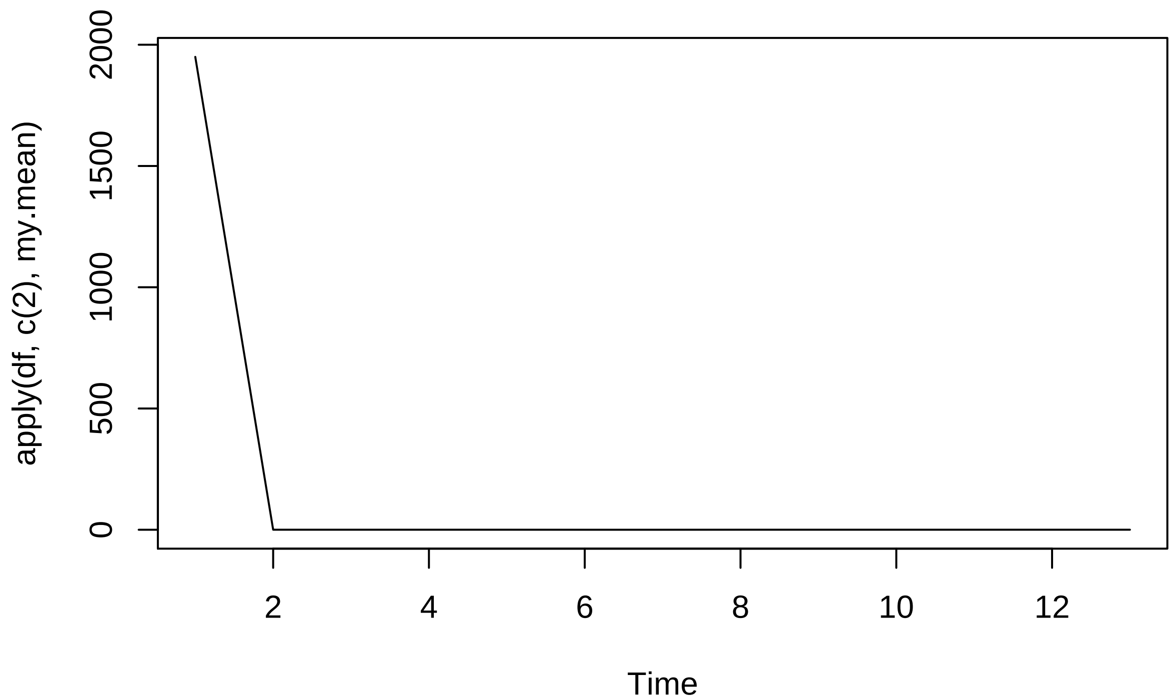
Question 4

```r
#import the data to dataframe
mydata <- read.csv('GLB.Ts_dSST.csv')
#make new dataframe for the desired year, jan-dec cols
df = mydata[c(1:13)]

#define a function to compute the mean of x without the first value
my.mean=function(x){
  if (any(is.na(x))){
    print("warning na values in X")
    return
  }
  n = length(x)
  xnew = x[-1]
  mymean = mean(xnew)
```

```
  mymean
}

# use the apply function to generate a yearly average temperatures from 1880 to 2019
yearlyavgs = apply(df, c(1), my.mean)

#we plot the data as a time series
plot.ts(yearlyavgs)
```




```
#note that the yearly average temperature is trending upwards

#now we plot the monthly tempuratures for jan to dec 1880 as follows

plot.ts(apply(df, c(2), my.mean))
```



Question 5

```
# We write a function to compute the empirical cumulative distribution function as follows

#define the indicator function as follows
my.indicator<-function(a,b){
  n = length(a)
  count = 1
  while (count < n){
    if(a[count] <= y){
      a[count] =1
    }else{
      a[count]=0
    }
    count= count +1
  }
  a
}

#define the edf function with error checking
my.ecdf<-function(x,y){
  if (length(y)!=1){
    print("y must be a single value")
```

```r
    return
  }
  n = length(x)
  Fn = sum(my.indicator(x,y))/n
  Fn
}

#create values for test cases with rnorm and y =-2
x=rnorm(20)
y=-2
#creates second test case x = median(x) and y=2
x1 = median(x)
y1 =2
#compute the test cases
my.ecdf(x,y)
```

```
## [1] -0.01148481
```

```r
my.ecdf(x1,y1)
```

```
## [1] 0.2132042
```