Assignment1

Dor Kolsky & Chemi Goldstein

11/30/2020

required packages that must be installed:

```
#install.packages('magrittr')
#install.packages('data.table')
#install.packages('ggplot2')
#install.packages("reshape2")
#install.packages('lubridate')
#install.packages('stringr')
#install.packages("corrplot")
#install.packages('dplyr')
#install.packages('gridExtra')
#install.packages('microbenchmark')
#install.packages('chron')
```

required packages that must be loaded:

```
library(magrittr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(data.table)
```

```
##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 4.0.3
```

```
## corrplot 0.84 loaded
```

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:data.table':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(stringr)
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
```

```
## The following objects are masked from 'package:data.table':
##
##     dcast, melt
```

```
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(microbenchmark)
```

```
## Warning: package 'microbenchmark' was built under R version 4.0.3
```

```
library(chron)
```

```
## Warning: package 'chron' was built under R version 4.0.3
```

```
##
## Attaching package: 'chron'
```

```
## The following objects are masked from 'package:lubridate':
##
##     days, hours, minutes, seconds, years
```

Question 1: A:

```
my_aggregation <- function (x,is.truncated=F){
  if(is.truncated==FALSE) {
    result <- list("mean_x" = mean(x),"var_x" = var(x),"median_x" = median(x))
    return(result)
  }
  else{
    x <- x[x<=quantile(x,0.95) & x>=quantile(x,0.05)] # Trim the 5th quantile and the 95th quantile fro
    result <- list("mean_x" = mean(x),"var_x" = var(x),"median_x" = median(x))
    return(result)
  }

}
```

B: There will be a vast difference in said aggregates when the top&bottom 5 percentiles are substantially larger and smaller (respectfully) than the rest of the vector. this will cause a large difference in the variance, mean, and median of our vector. The robustness of our vector's median is quite strong regardless which aggregate is used, since our newly built function removes an equal portion of values from both the high and low edges of our vector - hence we will have around the same median

```
set.seed(1)
testvector <- rlnorm(1000000,1,0.5)
my_aggregation(testvector,F)
```

```
## $mean_x
## [1] 3.080371
##
## $var_x
## [1] 2.69299
##
## $median_x
## [1] 2.718961
```

```
my_aggregation(testvector,T)
```

```
## $mean_x
## [1] 2.936248
##
## $var_x
## [1] 1.348194
##
## $median_x
## [1] 2.718961
```

As we predicted, the MEAN and the MEDIAN for both conditions of our functions our almost identical, since we eliminated an equal proportions of our vector from both ends That being said, our variances are quite different, this is because the extreme values of our vector have been removed. This goes hand-in-hand with what we foresaw, and hence the variance of the trimmed vector is substantially smaller.

C:

```
system.time(my_aggregation(testvector,T))
```

```
##    user  system elapsed
##    0.07    0.02    0.11
```

The time taken of our function to run is 0.08

```
system.time(mean(testvector,trim=0.05))
```

```
##    user  system elapsed
##    0.03    0.00    0.03
```

The time taken of trim function to run is 0.03

The reason there is a substantial difference in run time for the 2 separate functions is that our newly-built function takes a longer route to find our answer, as opposed to the systems built-in function. Hence it takes our function more time.

Question 2: A:

```
airquality <- airquality
listofna <- lapply(airquality,is.na) #create a vector returning true and false. T= na, F= interger.
list(lapply(listofna,sum)) #summed and listed the total number of 'trues' in each category. (number of
```

```
## [[1]]
## [[1]]$Ozone
## [1] 37
##
## [[1]]$Solar.R
## [1] 7
##
## [[1]]$Wind
## [1] 0
##
## [[1]]$Temp
## [1] 0
##
## [[1]]$Month
## [1] 0
##
## [[1]]$Day
## [1] 0
```

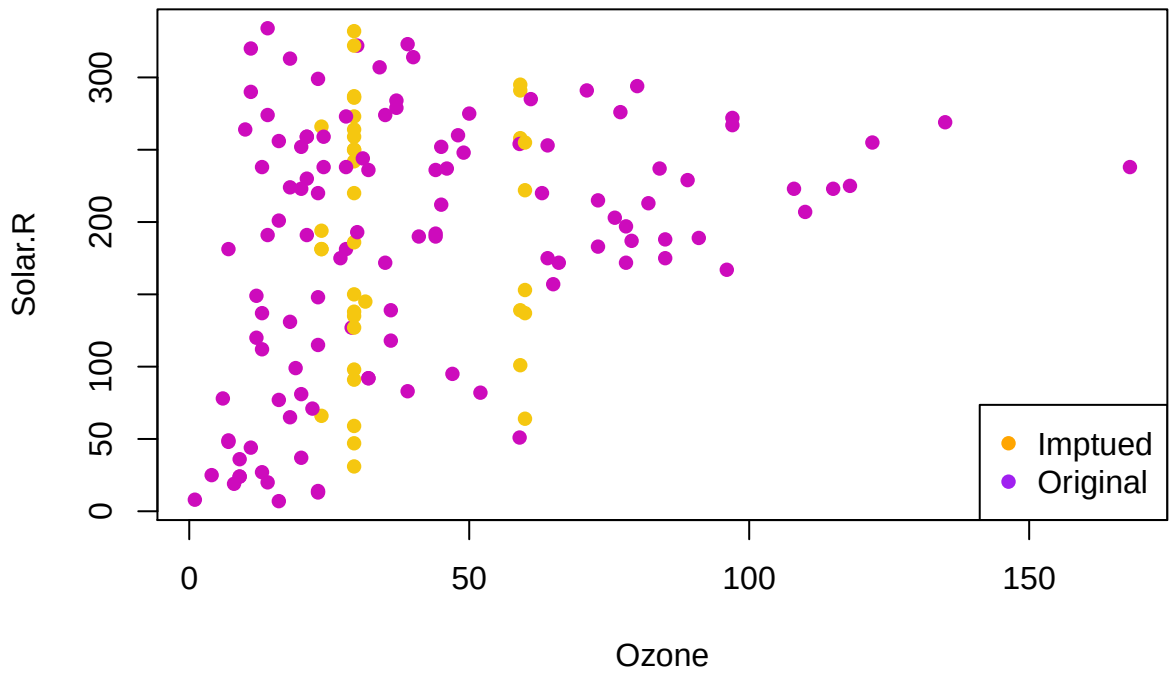
```
is.na_row <- apply(airquality,1 , anyNA)
```

B:

```
airquality_imputed <- as.data.table(airquality)
airquality_imputed <- airquality_imputed[,lapply(.SD,function(x) ifelse(is.na(x),mean(x,na.rm = TRUE),as
```

C:

```
plot(airquality_imputed$Ozone,airquality_imputed$Solar.R, col=15-airquality$Ozone %in%
airquality_imputed$Ozone,xlab="Ozone",ylab="Solar.R",pch=16) # Make different for the imputed informati
legend("bottomright" ,legend=c("Imptued","Original"),col=c("Orange","Purple"),pch=16,cex=1) #Add the le
```



Question 3: A:

```
mydataframe <- data.frame(diamonds)
mydim <-dim(diamonds)
cat("The dimensions of diamonds is " , mydim)
```

```
## The dimensions of diamonds is  53940 10
```

```
myclass <-class(diamonds)
cat("The class of diamonds is ", myclass)
```

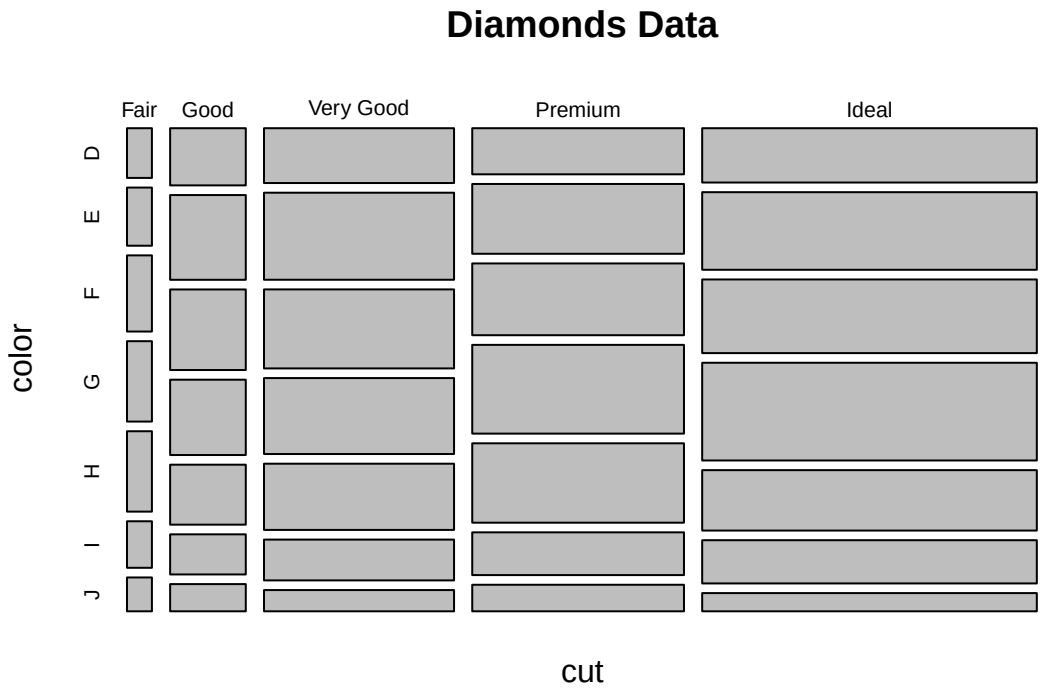
```
## The class of diamonds is  tbl_df tbl data.frame
```

B:

```
set.seed(1)
d25 <- data.table(sample_frac(mydataframe,0.25))
```

C:

```
d25_table_new <- xtabs(~cut+color , d25)
mosaicplot(d25_table_new,main="Diamonds Data")
```



This plot shows us a visualization of the volume for each cut of diamonds in respect to the diamond's color. For example, as we can see, there are a small number of J-colored diamonds who's cut-class is 'fair'. And there is a high volume of G-colored diamonds, who's cut-class are 'ideal'.

D:

```
d25 <-mutate(d25,logp=log10(d25$price))
d25 <- mutate(d25 , v= d25$x*d25$y*d25$z)
median_depth <- median(d25$depth)
d25 <- mutate(d25, cond1 = (d25$cut=="Ideal")+(d25$depth<median_depth)+ (d25$clarity!="I1")+(d25$color
```

E:

```
d25[,list(Vmean = mean(v),Vvar = var(v), LOGPmean = mean(logp), LOGPvar = var(logp)),by = cond1]
```

```
##    cond1    Vmean     Vvar LOGPmean   LOGPvar
## 1: FALSE 146.9135 6924.207 3.440004 0.2019500
## 2:  TRUE 112.8719 4234.052 3.321871 0.1767661
```

F:

```
colorcut <- data.table(expand.grid(unique(d25$color), unique(d25$cut)))
colnames(colorcut)[1]<-"color"
colnames(colorcut)[2]<-"cut"
set.seed(1)
colorcut <- mutate(colorcut, some_feature = rnorm(nrow(colorcut)))


d25<-merge(d25,colorcut,by=c("cut","color"),all = TRUE)
```

1)

```
meanlist <- d25[,list ( Avg_S_Feature = mean(some_feature)), by = clarity]
meanlist <- arrange(meanlist,desc(meanlist$Avg_S_Feature))
print(meanlist)
```

```
##    clarity Avg_S_Feature
## 1:      IF     0.4079421
## 2:    VVS2     0.3588454
## 3:    VVS1     0.3183334
## 4:     VS2     0.2519951
## 5:     SI1     0.2365927
## 6:     VS1     0.2077985
## 7:     SI2     0.1609508
## 8:      I1    -0.1361384
```

2)

```
d25[some_feature>1,list (SDPrice = sd(price), IQRPrice = IQR(price) , MADPrice = mad(price)), by = cut]
```

```
##          cut  SDPrice IQRPrice MADPrice
## 1:      Fair 3368.558   3500.0 2650.889
## 2:      Good 4712.540   6625.5 3857.725
## 3: Very Good 3534.201   3414.5 1747.985
## 4:     Ideal 2781.004   1991.0 1171.254
```

3)

```
d25[((carat>1 & carat<2) | (price>5000 & price<10000)) & ((carat>1 & carat<2) & (price>5000 & price<1000
```

```
##    color   N
## 1:     D 155
## 2:     E 219
## 3:     F 310
## 4:     G 476
## 5:     H 443
## 6:     I 267
## 7:     J 181
```

G:

```
mean_price_by_color_cut<- acast(d25,cut~color,value.var="price",mean)
round(mean_price_by_color_cut,digits=0)
```

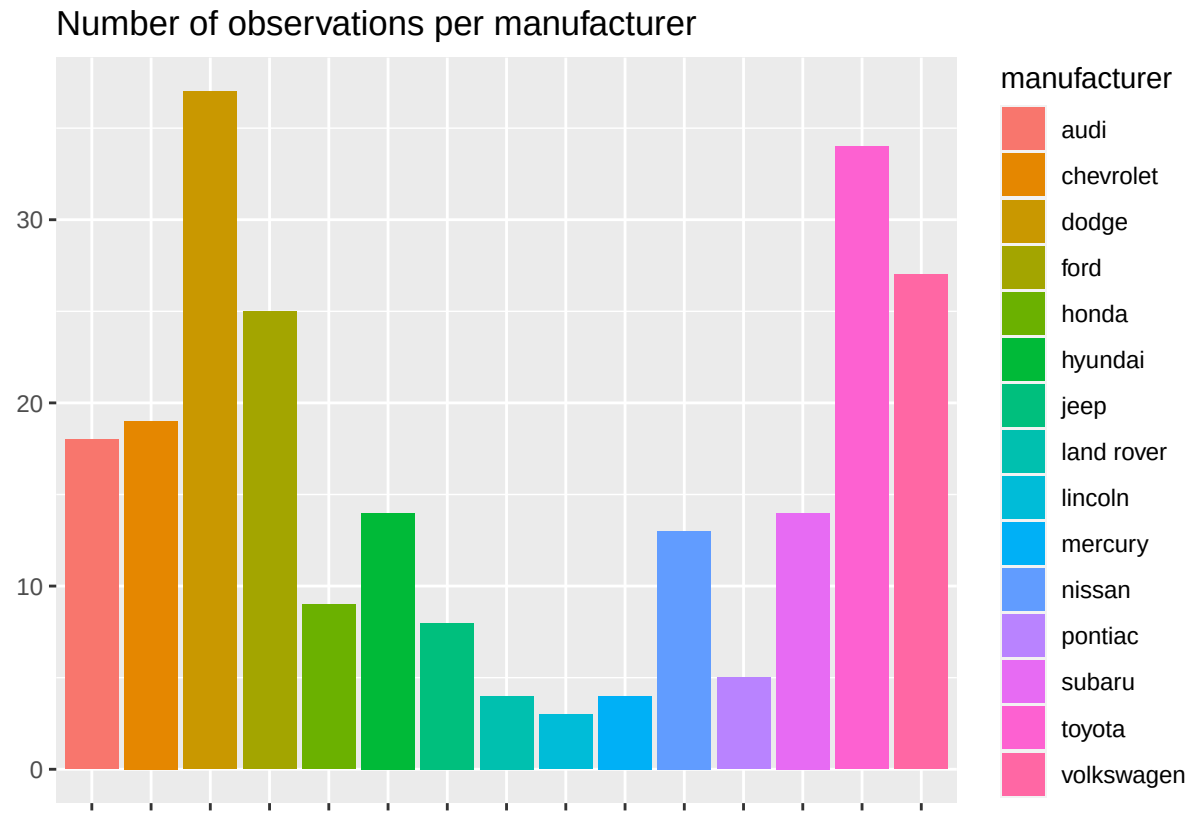
```
##               D    E    F    G    H    I    J
## Fair       3789 4406 3416 3795 4827 4573 5163
## Good       3499 3345 3628 4182 4479 5138 4034
## Very Good 3505 3224 3755 3757 4517 5080 4886
## Premium    3335 3339 4310 4617 5382 5612 6399
## Ideal      2501 2675 3269 3563 3697 4761 4883
```

Question 4: Graph 1:

```
mpg <- data.table(mpg)
g <- ggplot(data=mpg)
g <- g + geom_histogram(aes(x=manufacturer,fill=manufacturer),stat="count")
```

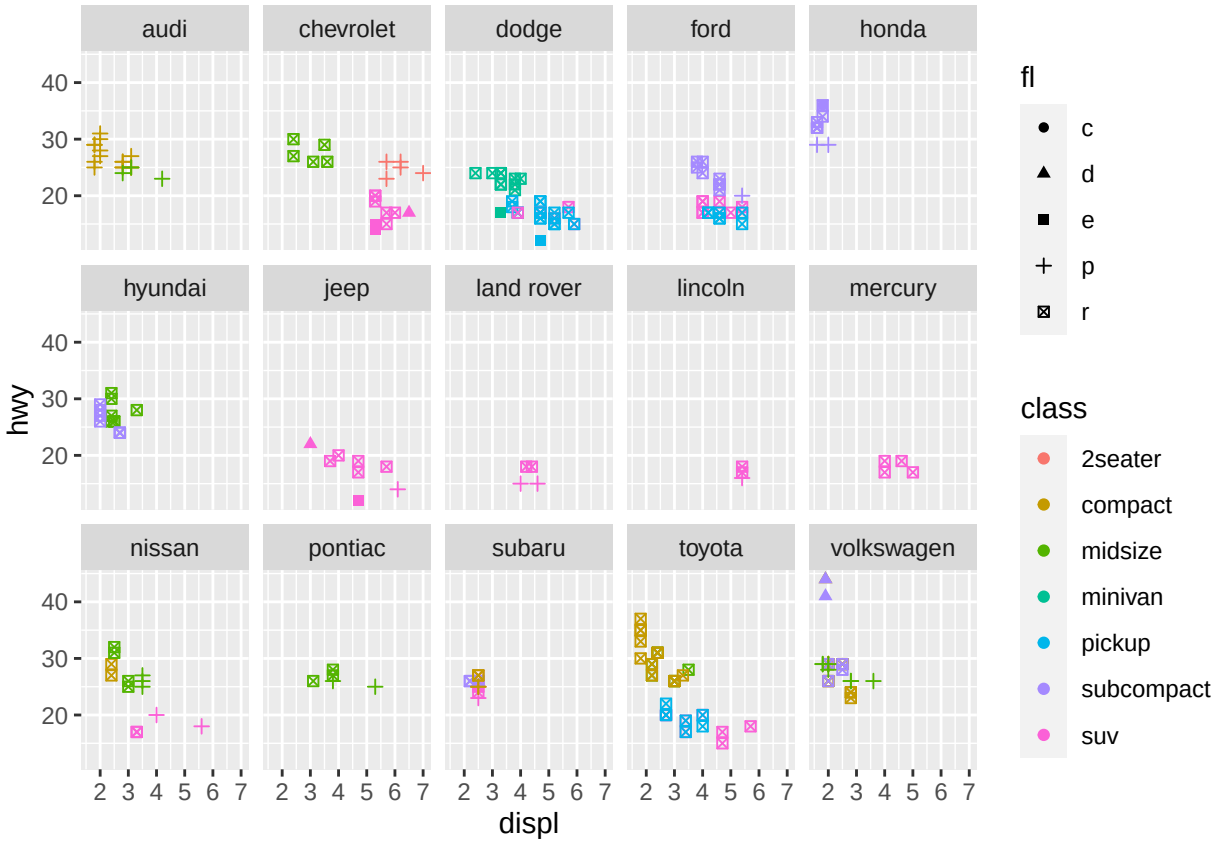
```
## Warning: Ignoring unknown parameters: binwidth, bins, pad
```

```
g <- g + ggtitle("Number of observations per manufacturer")
g <- g + xlab("") + ylab("")+theme(axis.text.x = element_blank())
g
```



Graph 2:

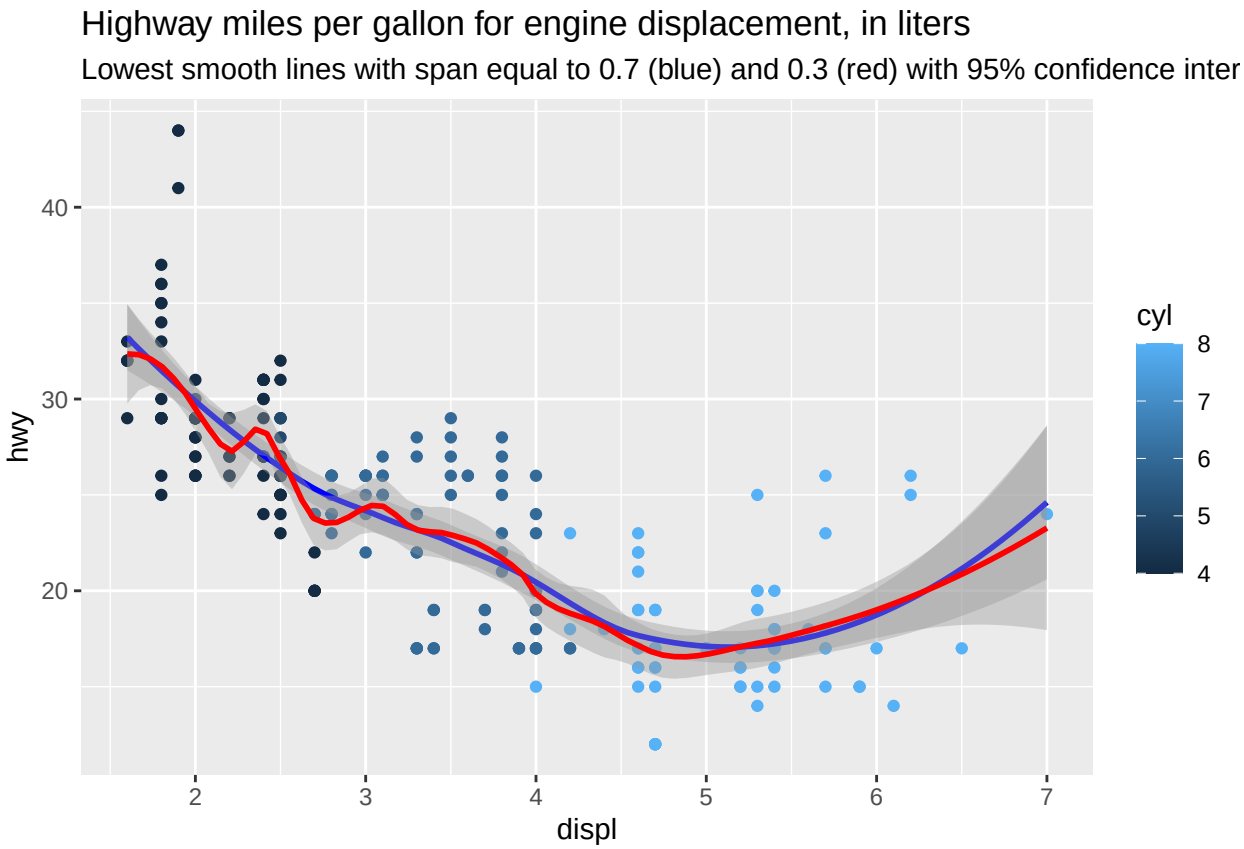
```
g1 <- ggplot(data=mpg)
g1 <- g1 + geom_point(mapping = aes(x=displ , y= hwy , color=class , shape=fl))
g1 <- g1 + facet_wrap(~manufacturer , nrow = 3)
g1
```



Graph 3:

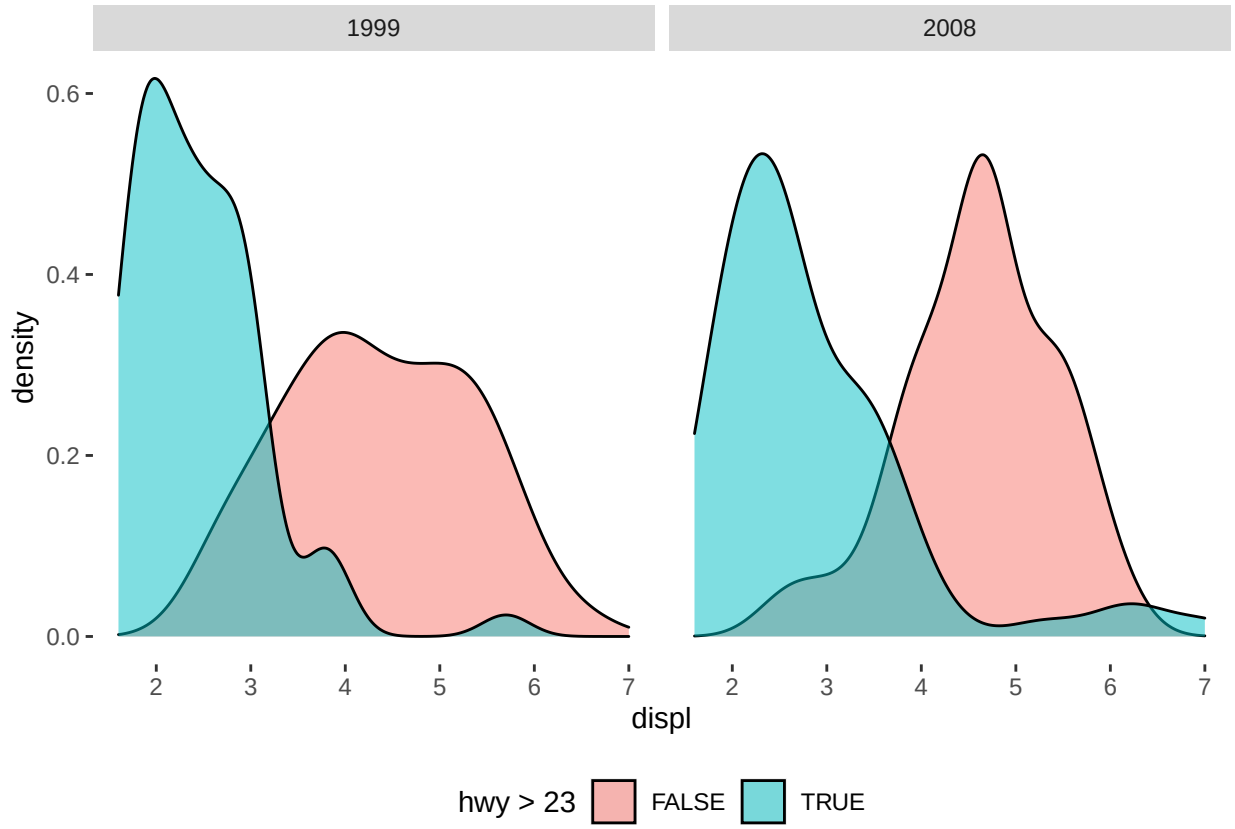
```
g2 <- ggplot(data=mpg,mapping= aes(x=displ , y=hwy , color = cyl ))
g2 <- g2+ geom_point()
g2 <- g2 + geom_smooth(span=0.7, colour="blue",level=0.95)
g2 <- g2 + geom_smooth(span=0.3, colour="red",level=0.95)
g2 <- g2 + ggtitle("Highway miles per gallon for engine displacement, in liters", subtitle = "Lowest sm
g2
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Graph 4:

```
g3 <- ggplot (data = mpg , mapping = aes(x=displ  , fill = hwy > 23  ))
g3 <- g3 + geom_density(alpha=0.5)
g3 <- g3 +  facet_wrap(~year)
g3 <- g3 + theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),panel.backgroun
               legend.position="bottom")
g3
```



Question 5: A:

```
airq <- as.data.table(airquality)
stock <- as.data.table(EuStockMarkets)
```

B:

```
dayvec <- airq$Day
monthvec <- airq$Month
Date <-paste("2019",monthvec,dayvec,sep="-")
Date <- as.Date(Date)
airq <- mutate(airq,date =Date)
date <- seq(as.Date("2019-01-01"), by = "day", length.out = nrow(stock))
stock <- mutate(stock, date =date)
```

C:

```
airq <-merge(x=airq,y=stock,by="date",all.x=TRUE)
```

D:

```
minimum_CAC_Level <- min(airq$CAC)
min_date_in_CAC <- airq[airq$CAC==minimum_CAC_Level,1]
maximum_CAC_Level <- max(airq$CAC)
max_date_in_CAC <- airq[airq$CAC==maximum_CAC_Level,1]
```

```
time_diff_by_days <- max_date_in_CAC-min_date_in_CAC
time_diff_by_hours <- time_diff_by_days * 24
time_diff_by_years <- time_diff_by_days / 365
paste("The diffrence between the 2  dates is " ,time_diff_by_days , "days, or" , time_diff_by_hours , "
```

```
## [1] "The diffrence between the 2  dates is  100 days, or 2400 hours, or 0.273972602739726 years"
```

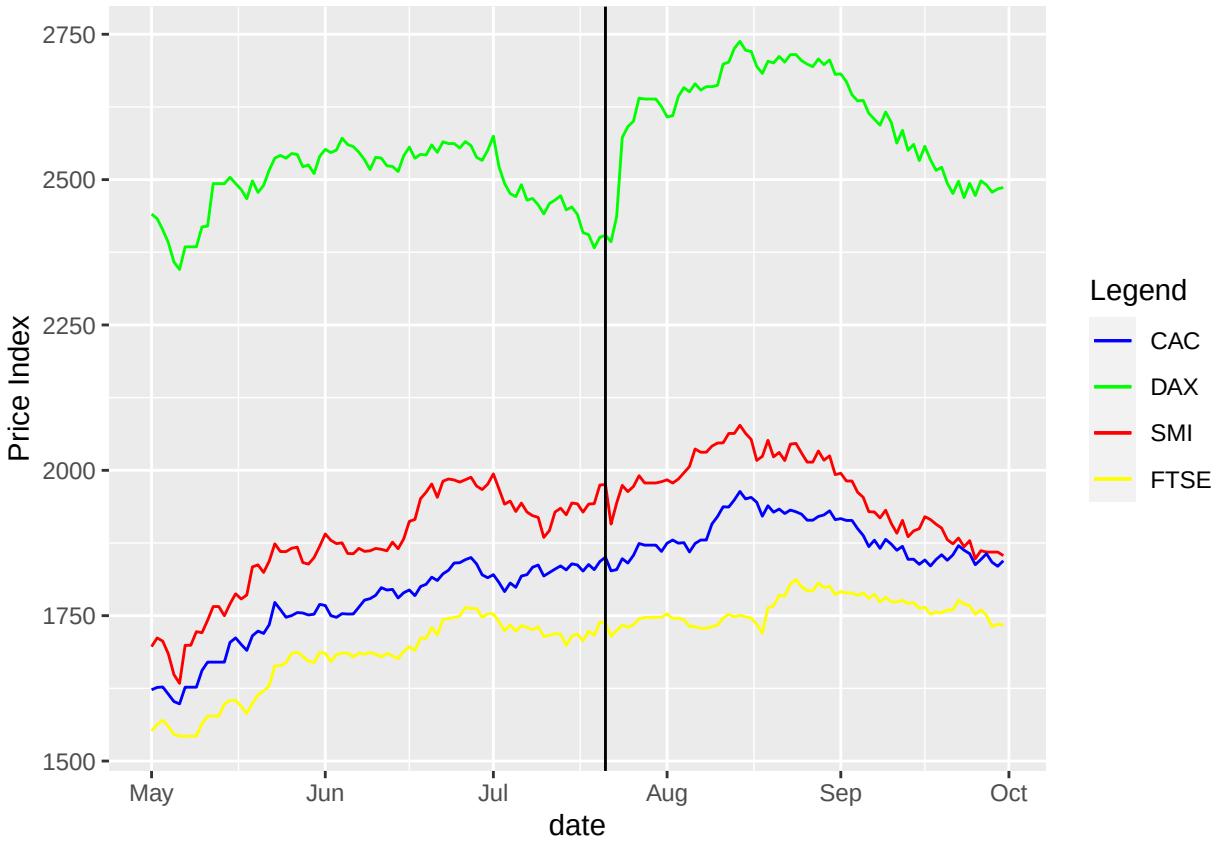
E:

```
time_between_Min_Max <- subset(airq, date>=min_date_in_CAC[1,date] & date<=max_date_in_CAC[1,date])
timeavg <- mean(time_between_Min_Max$Temp)
cat("the average temp throughout the min/max_CAC dates is" , timeavg)
```

```
## the average temp throughout the min/max_CAC dates is 77.94059
```

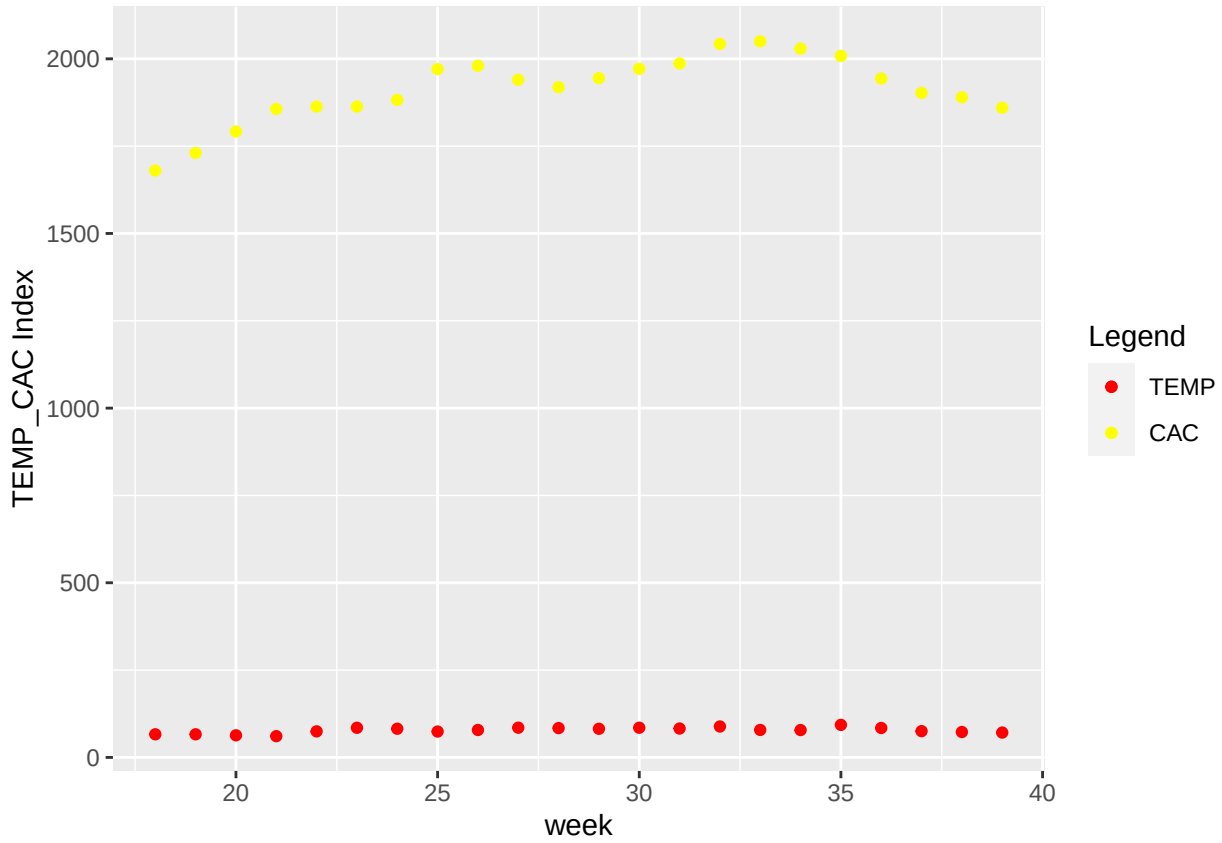
F:

```
minimum_solar <- min(airq$Solar.R,na.rm=TRUE)

date_in_min_solar <- airq[airq$Solar.R==minimum_solar,1]

ourplot <- ggplot(data=airq , aes(x= date))
ourplot <- ourplot + geom_line(aes(date,CAC,color='red'))
ourplot <- ourplot + geom_line(aes(date,DAX, color='yellow'))
ourplot <- ourplot + geom_line(aes(date,SMI, color='blue'))
ourplot <- ourplot + geom_line(aes(date,FTSE, color='green'))
ourplot <- ourplot + geom_vline(xintercept = as.numeric(airq[airq$Solar.R==minimum_solar,1]))
ourplot <- ourplot + ylab("Price Index")
ourplot <- ourplot + scale_colour_manual(name = 'Legend', values =c('red'='red','yellow'='yellow','blue
ourplot
```



G:

```
airq <- mutate(airq,week=week(airq$date))
weekly_merged <- airq[,list(avg_temp = mean(Temp),avg_cac= mean(CAC)), by = week]

plotg <- ggplot(weekly_merged, aes(x=week))
plotg <- plotg + geom_point(aes(x=week, y=avg_temp,color='red'))
plotg <- plotg + geom_point(aes(x=week, y=avg_cac,color='yellow'))
plotg <- plotg + scale_colour_manual(name = 'Legend', values =c('red'='red','yellow'='yellow'), labels =
plotg <- plotg + ylab("TEMP_CAC Index")
plotg
```



Q5-G) The values for the question in hand can be seen in the 'weekly_merged' dataset

Question 6:

```
WHR2017 <- read.csv("C:/Users/user/Downloads/2017.csv")
```

A:

```
print(colnames(WHR2017))
```

```
##  [1] "Country"                          "Happiness.Rank"
##  [3] "Happiness.Score"                  "Whisker.high"
##  [5] "Whisker.low"                      "Economy..GDP.per.Capita."
##  [7] "Family"                           "Health..Life.Expectancy."
##  [9] "Freedom"                          "Generosity"
## [11] "Trust..Government.Corruption."    "Dystopia.Residual"
```

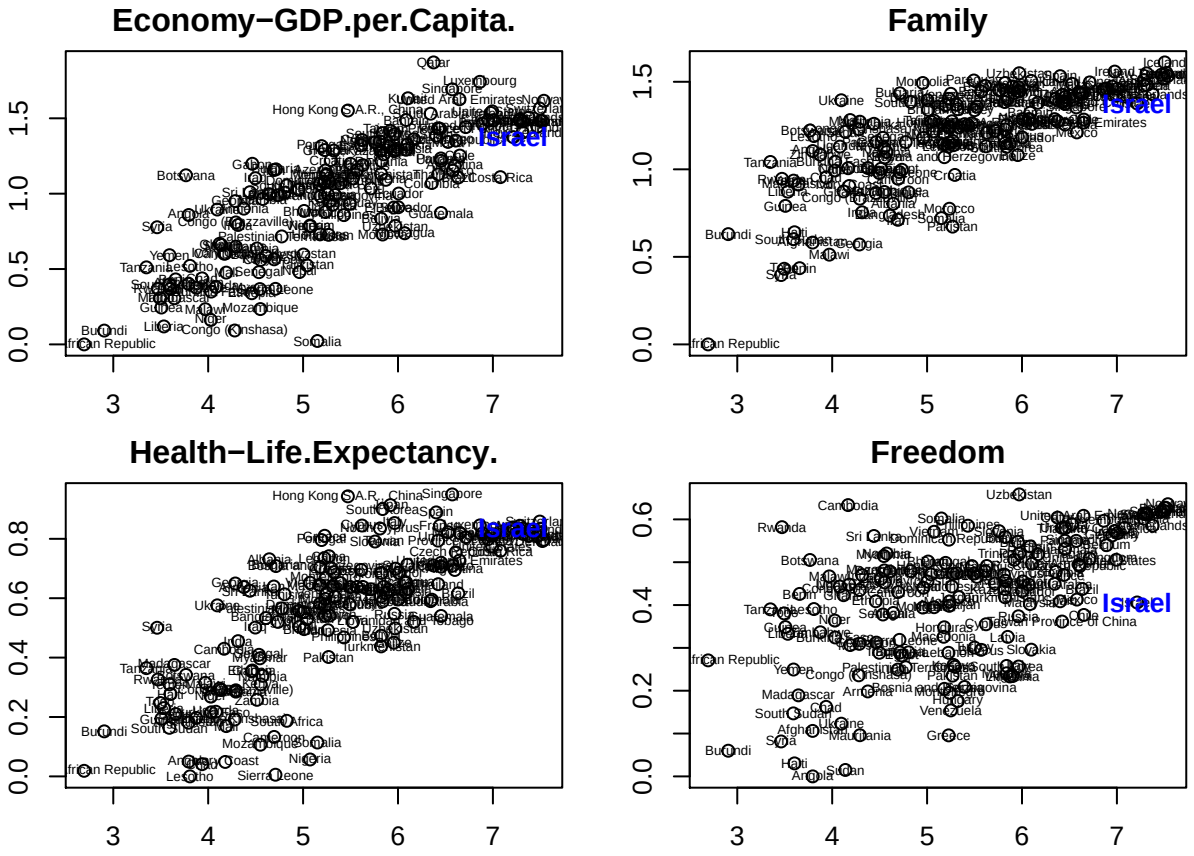
```
for (i  in 1:NROW(colnames(WHR2017))){
colnames(WHR2017)[i]<-gsub("..","-",colnames(WHR2017)[i],fixed = TRUE)}
print(colnames(WHR2017))
```

```
##  [1] "Country"                          "Happiness.Rank"
##  [3] "Happiness.Score"                  "Whisker.high"
##  [5] "Whisker.low"                      "Economy-GDP.per.Capita."
##  [7] "Family"                           "Health-Life.Expectancy."
##  [9] "Freedom"                          "Generosity"
## [11] "Trust-Government.Corruption."     "Dystopia.Residual"
```

We created a loop where each column name is overlooked to see whether or not there is a double-dot in the name (..), in instances where that IS the case, our loop replaces the double-dot with a hyphen. As was asked of us.

B:

```
attach(WHR2017)
plot1<-list(‘Economy-GDP.per.Capita.‘,‘Family‘,‘Health-Life.Expectancy.‘,‘Freedom‘)
title_vector <- c("Economy-GDP.per.Capita.","Family","Health-Life.Expectancy.","Freedom")
par(mfrow = c(2,2), mar= rep (2,4))
j <- 1
for(i in plot1){
  plot(x=‘Happiness.Score‘,y=i, main = title_vector[j])
  graphics::text(‘Happiness.Score‘,i,Country, cex=.5)
  graphics::text(‘Happiness.Score‘,i ,ifelse(Country==’Israel’,’Israel’,""),font = 2, col="blue")
  j=j+1}
```

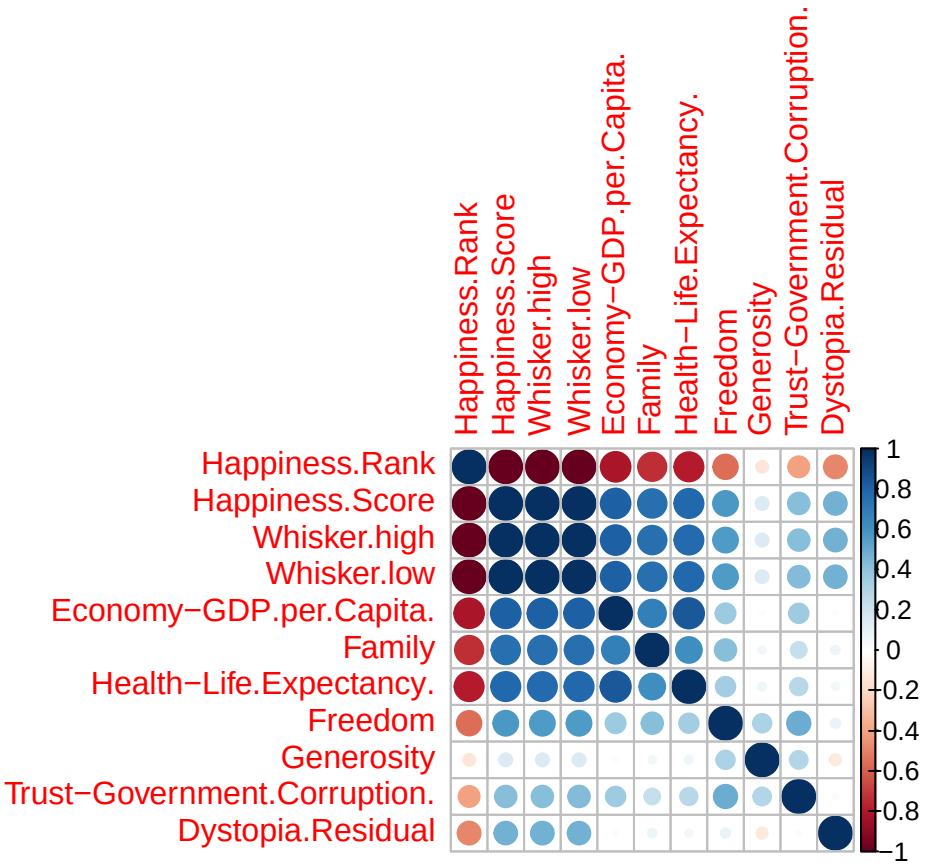



```
  detach(WHR2017)
```

This multi-way graph plots for each graph it's title, versus the Happiness.Score.

c:

```
WHR2017_Titles<-WHR2017[,2:12]
corrplot(cor(WHR2017_Titles), order = "original" )
```



This plot gives us a clear visualization of each factor's correlation to all other factors. where the higher POSITIVELY correlated variables are colored blue, whereas NEGATIVLELY correlated factors are shown in red. uncorrelated variable are not colored. For example, HealthExpectancy-Family are highly-positively correlated. GDPpercCapita-Happiness are highly-negatively correlated. Generosity-Family are not correlated whatsoever.

Question 7:

```
autos <- fread("C:/Users/user/Downloads/autos.csv" )
```

```
## Warning in fread("C:/Users/user/Downloads/autos.csv"): Found and resolved
## improper quoting out-of-sample. First healed line 5263: <<2016-03-29
## 16:46:46,"_SPARDOSE"______Polo_1_4___6N1___60PS___5Tuerer____FESTPREIS,privat,Angebot,
## 500,control,limousine,1999,manuell,60,polo,150000,12,benzin,volkswagen,ja,
## 2016-03-25 00:00:00,0,59581,2016-03-30 11:46:58>>. If the fields are not quoted
## (e.g. field separator does not appear within any field), try quote="" to avoid
## this warning.
```

A:

```
countmazda <- sum(grepl("mazda",autos$name,ignore.case=TRUE))
cat("The number of mazdas is ", countmazda)
```

```
## The number of mazdas is  5463
```

```
mazda <- autos[grepl("mazda",autos$name,ignore.case=TRUE)]
```

B:

```
mazda <- mutate(mazda, is_3 = grepl("3", mazda$name))
```

C:

```
mazda[, list(Mean_Diff_Hours =mean(difftime(as.POSIXct(lastSeen),as.POSIXct(dateCreated),units = "hours
```

```
##     is_3 Mean_Diff_Hours Numofcars DieselRatio
## 1:  TRUE  206.3029 hours      1730  0.09710983
## 2: FALSE  215.8856 hours      3733  0.22769890
```

Question 8 A:

```
ourfunction <- function(d){
  mat <- matrix(data=0,nrow=d,ncol=d)
  mat[,d] <- 1
  mat[d,] <- 1
  mat[1,] <- 1
  mat[,1] <- 1
  return (mat)
}
```

B:

```
ourfunction2 <- function(a,b){
  if(length(a)==length(b)){ # if both vectors are the same length we begin to exam each value by value.
    for (i in 1:length(a)) {
      if(a[i]!=b[i]){return(FALSE)} # if we one of the values is not  identical to its counterpart, we
    }
    return(TRUE) #if all the values in both vectors are identical, we return 'TRUE'
  }
  return(FALSE) #Not Same Length vecots - Can't be identical.
}
```

C:

```
ourfunction3 <- function(a,b){
  count <- str_count(a,b)
  return (count)
}
```

D:

```
ourfunction4 <- function(birthday){
  theday<- wday(as.Date(birthday,"%d/%m/%Y"),label=TRUE)
  timesince <- difftime(Sys.Date(), birthday,units="days")
  nextbirthday <- make_datetime(year=year(Sys.Date())+1,month=month(birthday),day=day(birthday))
  timeuntil <- difftime(nextbirthday,Sys.time() , units="days" )
  return(list(theday,timesince,timeuntil))
}
```

Question 9: A:

```
diamonds <- as.data.table(diamonds)
newdiamonds<- diamonds[,-c(2,3,4)] #We removed the non numeric columns. (columns 2, 3, and 4)
cor(newdiamonds)
```

```
##            carat       depth      table      price           x           y
## carat 1.00000000  0.02822431  0.1816175  0.9215913  0.97509423  0.95172220
## depth 0.02822431  1.00000000 -0.2957785 -0.0106474 -0.02528925 -0.02934067
## table 0.18161755 -0.29577852  1.0000000  0.1271339  0.19534428  0.18376015
## price 0.92159130 -0.01064740  0.1271339  1.0000000  0.88443516  0.86542090
## x     0.97509423 -0.02528925  0.1953443  0.8844352  1.00000000  0.97470148
## y     0.95172220 -0.02934067  0.1837601  0.8654209  0.97470148  1.00000000
## z     0.95338738  0.09492388  0.1509287  0.8612494  0.97077180  0.95200572
##                z
## carat 0.95338738
## depth 0.09492388
## table 0.15092869
## price 0.86124944
## x     0.97077180
## y     0.95200572
## z     1.00000000
```

A correlation matrix is vastly important when analyzing data, since it shows us a clearer idea of which variables are dependent on each other and thus help us to better understand the robustness of our data!

B: The Pearson correlation between 'cut'-'Color' can not be computed, since these are both descriptive variables, and have no true-value.

C:

```
proptable_bycolor <- diamonds %>%
  group_by(color, cut) %>%
  summarise(num = n()) %>%
  mutate(prop = num/(sum(num)))%>%
  dcast(color~cut)
```

```
## `summarise()` regrouping output by 'color' (override with `.groups` argument)
```

```
## Using prop as value column: use value.var to override.
```

D:

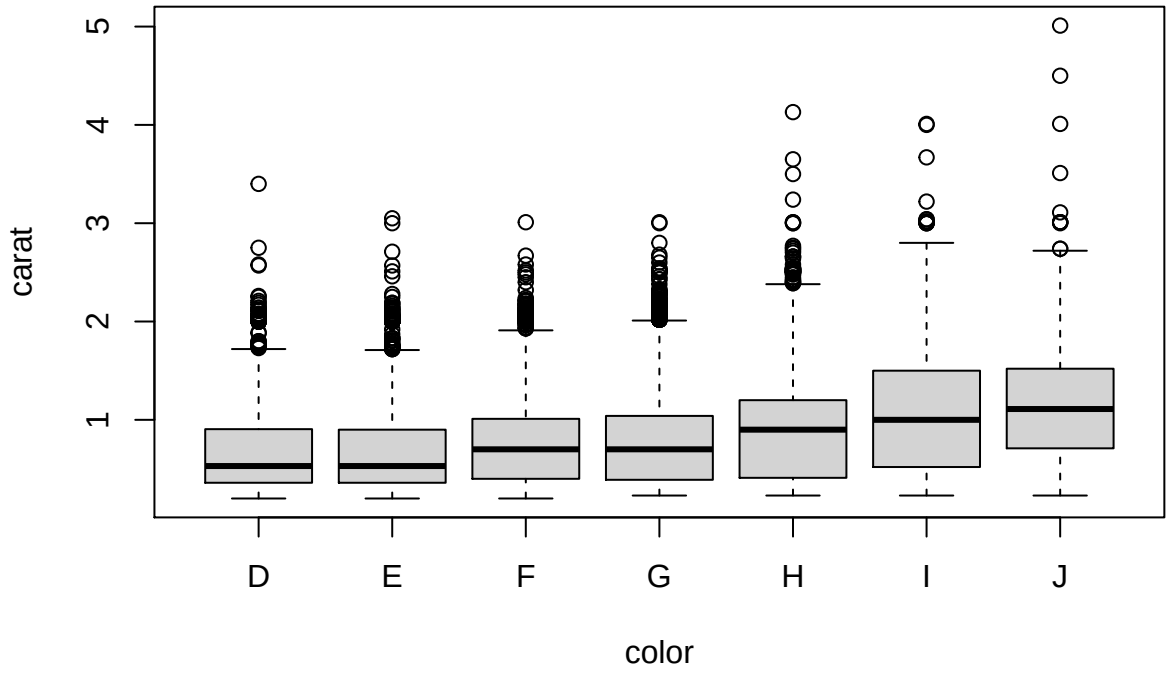
```
diamondsint <- diamonds
diamondsint$color <- as.integer(diamondsint$color)
cor(diamondsint$color,diamondsint$carat)
```

```
## [1] 0.2914368
```

The value of the Color-Carat correlation holds no true significance, since we turned the 'color' variable into an integer, and thus gave each color a reference number (so that we can in turn correlate the two variable) but all-in-all the numbers hold no TRUE value. hence, the correlation we found is not informative at all.

E:

```
boxplot(carat~color, data = diamonds)
```



A great way to display the relationship between EACH color, and it's carat level, is by using a boxplot. This method shows a visualization of each color's carat level, and as we have learned, the boxplot takes into account the median/IQR as well any observations who deviate from the distribution greatly!

Question 10 : A:

```
madfunction <- function(x){
  center<- median(x)
  answer <- 1.4826 * median (abs(x-center))
  return(answer)
}
```

B:

```
set.seed(1)
sampleb<- rnorm(10,1,1)
mad1<- madfunction(sampleb)
sd1 <- sd(sampleb)
```

C:

```
set.seed(1)
samplec<- rexp(10,1)
```

```
mad2 <- madfunction(samplec)
sd2 <- sd(samplec)
```

D:

```
difference_norm <- abs(sd1-mad1)
difference_expo<- abs(sd2-mad2)
cat("The diffrence between the Standard Deviation and the MAD for
    the normal distribution sample is", difference_norm, "whereas the difference between the Standard D
    the Exponential distribution sample is", difference_expo  )
```

```
## The diffrence between the Standard Deviation and the MAD for
##     the normal distribution sample is 0.006881442 whereas the difference between the Standard Deviati
##     the Exponential distribution sample is 0.08832856
```

We would expect that the difference between the Standard Deviation and the MAD would be smaller whilst dealing with Normal distributions. This is intuitive because the normal distribution is SYMMETRICAL, and thus the difference between the SD/MAD is less significant. The MAD in essence takes into account observations who deviate greatly from the median. The difference between the median/mean in exponential distributions is larger than that of normal distributions.

E:

```
exp_vector <- c()
norm_vector <- c()
for (i in 1:1000){
  set.seed(1)
  tempnormsample <- rnorm(10,1,1)
  tempnormsd <- sd(tempnormsample)
  tempnormmad <- mad(tempnormsample)
  absdiffernorm <- abs(tempnormsd-tempnormmad)
  norm_vector<- c(norm_vector,absdiffernorm)
  set.seed(1)
  tempexpsample <- rexp(10,1)
  tempexpsd <- sd(tempexpsample)
  tempexpmad<- mad(tempexpsample)
  absdifferexp <- abs(tempexpsd-tempexpmad)
  exp_vector <- c(exp_vector, absdifferexp)
}

mean(exp_vector)
```

```
## [1] 0.08832856
```

```
mean(norm_vector)
```

```
## [1] 0.006881442
```

F: As we foresaw in Q10-D, the value of the difference between the SD/MAD is clearly larger in Exponential Distributions versus Normal Distributions. Our calculation repeated the process of sampling 10 observations in each of the two distributions, one-thousand times; and then creating a 1000-observation-vector of the difference between each sample's MAD/SD (in absolute value terms). Lastly we took the average difference MAD/SD for each vector in order to see more clearly the robustness of our hypothesis in Q10-D. In conclusion, our results show confidently that the difference between the SD/MAD is smaller in Normal Distributions!

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

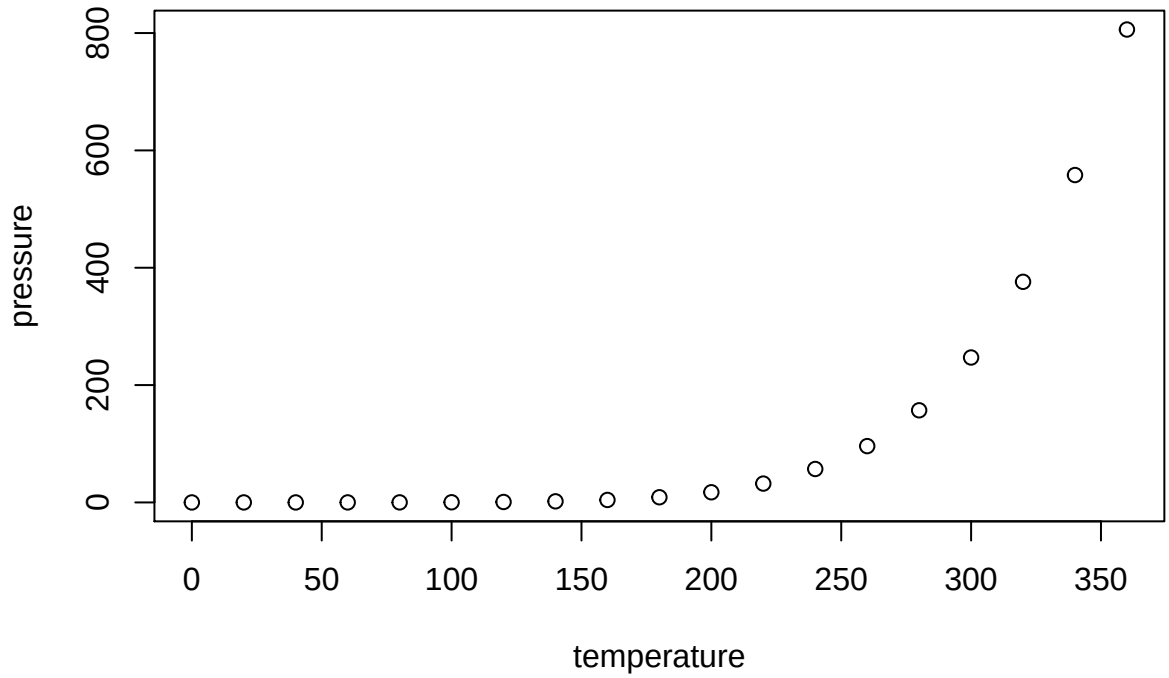
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed           dist
##  Min.   : 4.0   Min.   :  2.00
##  1st Qu.:12.0   1st Qu.: 26.00
##  Median :15.0   Median : 36.00
##  Mean   :15.4   Mean   : 42.98
##  3rd Qu.:19.0   3rd Qu.: 56.00
##  Max.   :25.0   Max.   :120.00
```

Including Plots

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.