DSC 465 HomeWork1 Assignment

Ronaldlee Ejalu

4/1/2021

#load the necessary packages

```
#library(plyr) # for data wrangling
library(dplyr)     # for data wrangling

## Warning: package 'dplyr' was built under R version 4.0.5

##
## 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(lubridate) # for dealing with dates

## Warning: package 'lubridate' was built under R version 4.0.3

##
## Attaching package: 'lubridate'

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

library(ggplot2)   # for dealing with visualization

## Warning: package 'ggplot2' was built under R version 4.0.3
```

#set the working directory

```
setwd("C:/Users/rejalu1/OneDrive - Henry Ford Health System/DSC
465/HomeWork/HomeWork1")
```

#Load the Intel Stock data set

```
intelstock <- read.csv(file = "../HomeWork1/datasets/Intel-1998.csv", sep =
"," , header = TRUE )
```

#view the first 6 records of the data set

```
head(intelstock)
```

```
##       Date Trading.Day  Open  High   Low Close   Volume Adj.Close
## 1 1/2/1998           1 70.69 72.62 70.50 72.62 40927200     16.94
## 2 1/5/1998           4 73.06 75.14 72.00 74.50 78368400     17.37
## 3 1/6/1998           5 73.87 74.31 72.69 73.12 48313600     17.05
## 4 1/7/1998           6 72.75 73.62 71.56 72.75 55380800     16.97
## 5 1/8/1998           7 72.25 74.81 72.12 74.31 75730400     17.33
## 6 1/9/1998           8 74.56 75.00 71.23 71.87 88028800     16.76
```

#Find any missing values

```
sum(is.na(intelstock))
```

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

#derive weekday

```
intelstockds <- intelstock %>%
  select(date = Date,
         trainingday = Trading.Day,
         open = Open,
         high = High,
         low = Low,
         close = Close,
         volume = Volume,
         adjclose = Adj.Close, date1 = Date,
         ) %>%
  transmute(date = mdy(date),
            date1 = date1,
            trainingday = trainingday,
            open=open,
            high=high,
            low=low,
            close=close,
            volume=volume,
            adjclose=adjclose,
            range = (high - low)
            ) %>%
  mutate(wday = wday(date, label = TRUE, abbr = TRUE), date1 = as.Date(date)

         )
```

An ordinary line graph of closing price vs date

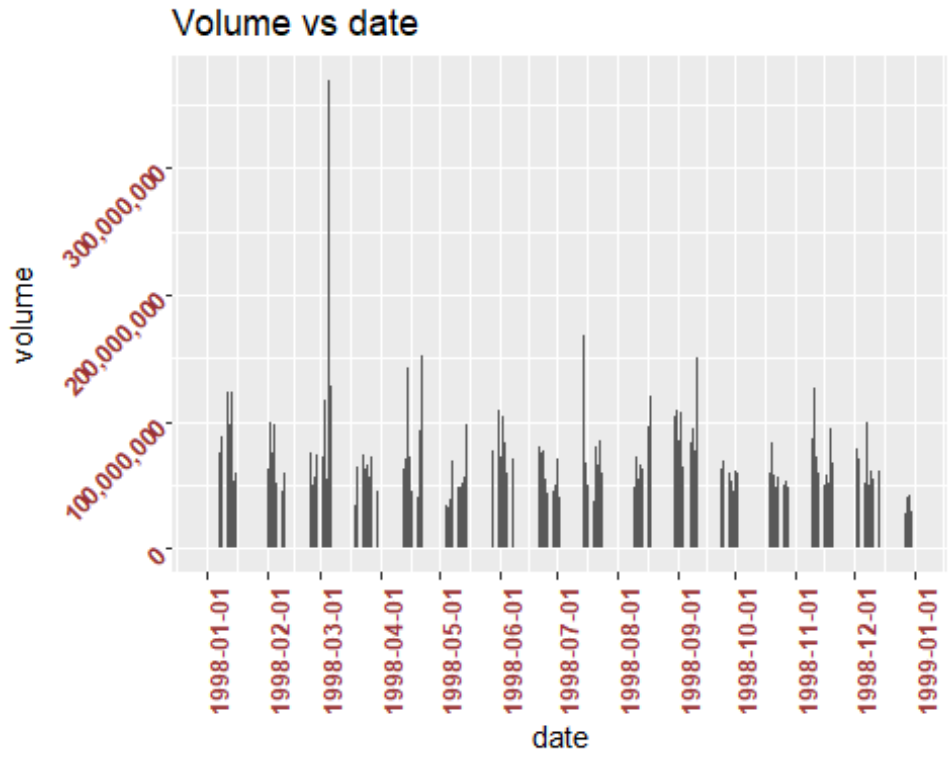
```
ggplot(data = intelstockds, aes(x = date, y = close)) +
  geom_line(size = 1.5, alpha = .6, position = position_dodge(0.2)) +
  ylim(0, max(intelstockds$close)) + # have the y range start from zero.
  annotate("text",
           x = mean(range(intelstockds$date)),
           y = Inf,
```

```
          label = "closing price vs date",
          vjust = 1.5,
          size = 4.5,
          colour = "red"
          )
```

#Bar graph of Volume Vs the exact Date

```
#specify breaks as a Date vector
datebreaks <- seq(as.Date("1998-01-01"), as.Date("1999-01-01"), by = "1
month")
p <- ggplot(intelstockds, aes(x = date, y = volume)) +
  geom_col(width = 0.5, position = "dodge") +
  ggtitle("Volume vs date")

p <- p + scale_x_date(breaks = datebreaks) +
  scale_y_continuous(labels = scales::comma) + # format the y axis and
include , in the volume values
  theme(axis.text.x = element_text(face="bold",
                                   color="#993333"
                                   #,  size=14
                                   , angle=90)
              , axis.text.y = element_text(face="bold",
                                    color="#993333"
                                   #, size=14
                                   , angle=45)
              )
```

```
p
```



normalize the volume

```
library(scales)
intelstockds <- intelstockds %>%
  mutate(norm.volume = scale(volume))
```

structure of intelstockds

```
str(intelstockds)
```

```
## 'data.frame':    252 obs. of  12 variables:
##  $ date       : Date, format: "1998-01-02" "1998-01-05" ...
##  $ date1      : Date, format: "1998-01-02" "1998-01-05" ...
##  $ trainingday: int  1 4 5 6 7 8 11 12 13 14 ...
##  $ open       : num  70.7 73.1 73.9 72.8 72.2 ...
##  $ high       : num  72.6 75.1 74.3 73.6 74.8 ...
##  $ low        : num  70.5 72 72.7 71.6 72.1 ...
##  $ close      : num  72.6 74.5 73.1 72.8 74.3 ...
##  $ volume     : int  40927200 78368400 48313600 55380800 75730400 88028800
123747600 97348000 123962000 53458400 ...
##  $ adjclose   : num  16.9 17.4 17.1 17 17.3 ...
##  $ range      : num  2.12 3.14 1.62 2.06 2.69 ...
##  $ wday       : Ord.factor w/ 7 levels "Sun"<"Mon"<"Tue"<..: 6 2 3 4 5 6 2
```

```
3 4 5 ...
##  $ norm.volume: num [1:252, 1] -0.89 0.301 -0.655 -0.431 0.217 ...
##   ..- attr(*, "scaled:center")= num 68917848
##   ..- attr(*, "scaled:scale")= num 31444442
```

summary statistics of norm.volume

```
summary(intelstockds$norm.volume)
```

```
##        V1
##  Min.   :-1.6422
##  1st Qu.:-0.5989
##  Median :-0.1882
##  Mean   : 0.0000
##  3rd Qu.: 0.3017
##  Max.   : 9.5538
```

determine the bin width

using the formula 2 * IQR/cube root of n

```
# Q3 = 0.3017
# Q1 = -0.5989
# IQR = Q3 - Q1
# bin width  = 2 * IQR/cube root of n
```

number of observations

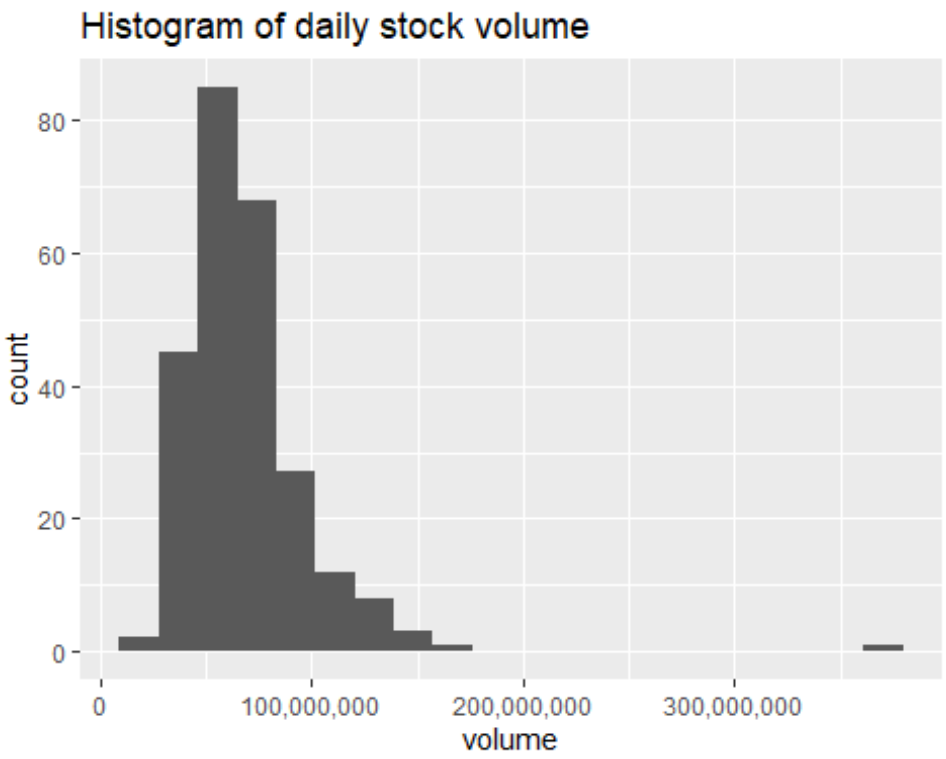
```
dim(intelstockds)
```

```
## [1] 252  12
```

Filt

#Creating a histogram of the daily stock Volume

```
ggplot(intelstockds, aes(x=volume)) +
  geom_histogram(bins = 20) +
  ggtitle("Histogram of daily stock volume") +
  scale_x_continuous(labels = scales::comma)
```

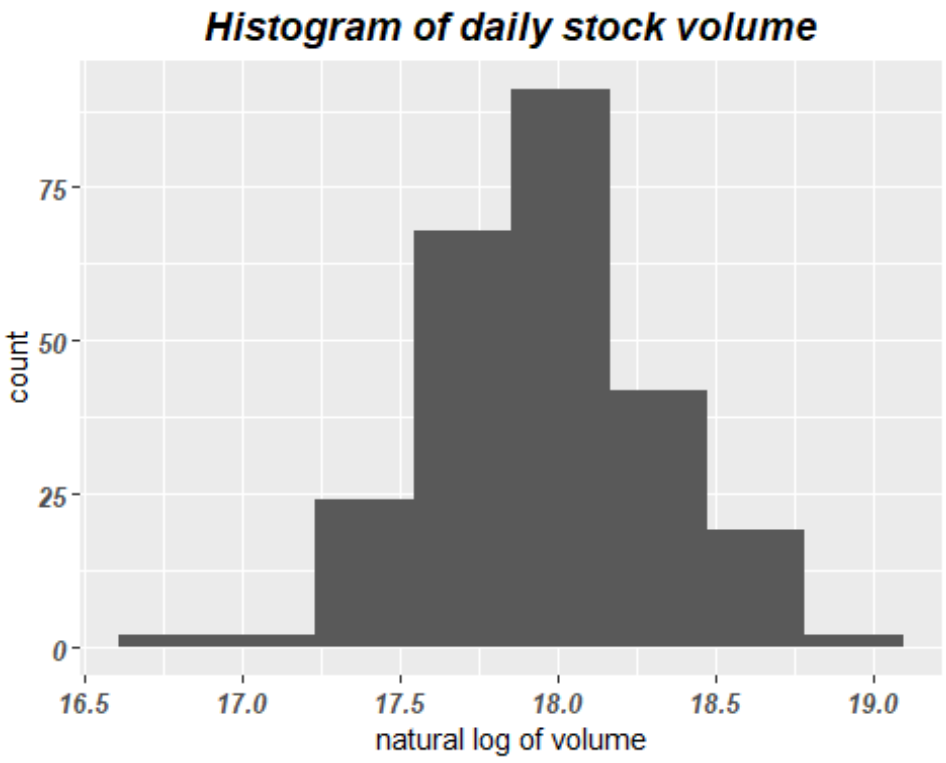


Remove entries with the means greater than 150086000

```
intelstockds.sortedf  <- intelstockds %>% filter(volume <= 151875200)
```

#Repeat or corrections for 1C

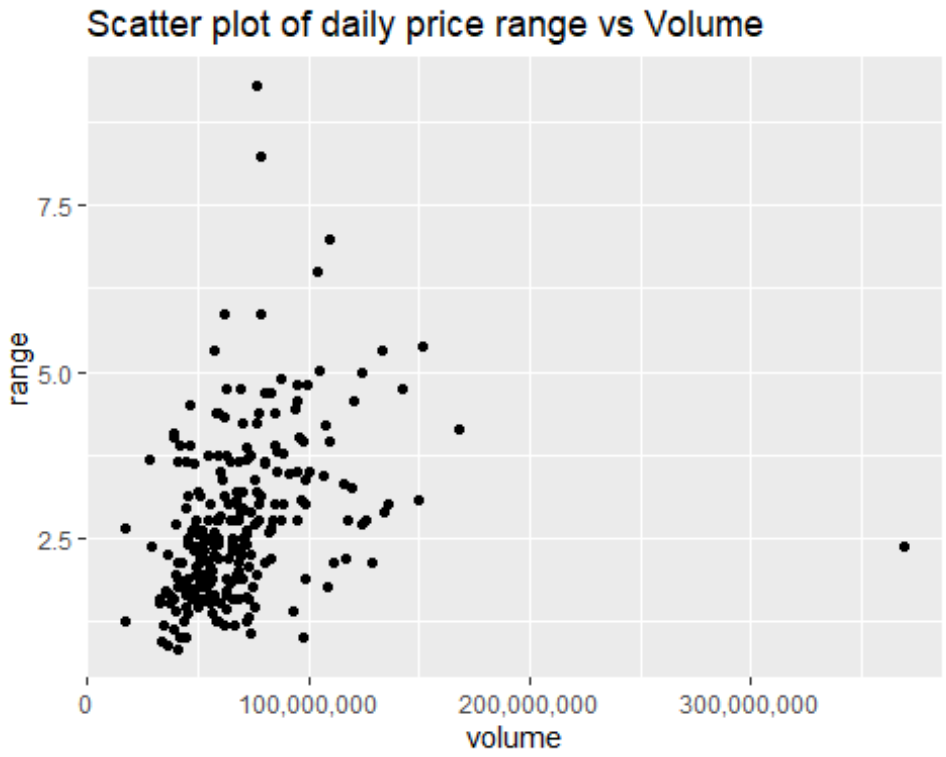
```
ggplot(intelstockds.sortedf, aes(x=log(volume))) +
  geom_histogram(bins = (log2(252)) + 1) +
  expand_limits(y=0) +
  theme(axis.text.x = element_text(face="bold.italic",

                                   size=10
                                   ),
        axis.text.y = element_text(face="bold.italic",

                                   size=10
                                    )
        ,
        plot.title = element_text(hjust = 0.5,
                                  size = 14,
                                  face = "bold.italic"
                                  ) # format the title

        )  + labs(x = "natural log of volume", title = "Histogram of daily
stock volume")
```



Scatter plot that graphs the Volume on the x-axis and

the daily price on the y-axis.

```
ggplot(intelstockds, aes(x = volume, y = range)) +
  geom_point(size = 1.5) + # using the size aesthetic to control the size of
the points.
  ggtitle("Scatter plot of daily price range vs Volume ") +
  scale_x_continuous(labels = scales::comma)
```



#Question 2
##
##
###############

#Load the data set

```
perceptionExperiment <- read.csv(file =
"../HomeWork1/datasets/PerceptionExperiment1.csv", sep = "," , header = TRUE
)
```

#View the first 6 rows

```
head(perceptionExperiment)
```

```
##                          Test Display TestNumber Trial Subject Response
## 1 Veritcal Distance, Aligned       1          1     B       1     0.60
## 2 Veritcal Distance, Aligned       1          1     B       2     0.55
## 3 Veritcal Distance, Aligned       1          1     B       3     0.70
## 4 Veritcal Distance, Aligned       1          1     B       4     0.60
## 5 Veritcal Distance, Aligned       1          1     B       5     0.65
## 6 Veritcal Distance, Aligned       1          1     B       6     0.60
##   TrueValue
## 1       0.6
## 2       0.6
```

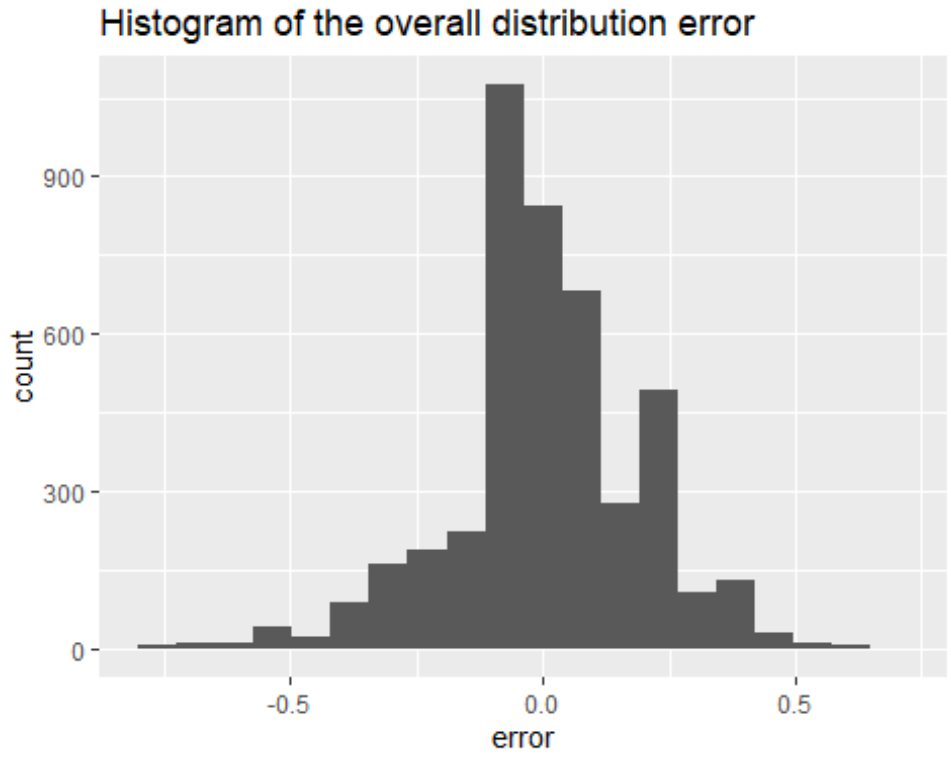
```
## 3       0.6
## 4       0.6
## 5       0.6
## 6       0.6
```

#derive a new column that contains the amount of error.

```
perceptionExperiment <- perceptionExperiment %>%
  mutate(error = Response - TrueValue,
         absoluteError = abs(Response - TrueValue), Test = as.factor(Test)
         )
```

#a histogram of the overall distribution of Error

```
ggplot(perceptionExperiment, aes(x=error)) +
  geom_histogram(bins = 20) +
  ggtitle("Histogram of the overall distribution error") #+
```




```
  #scale_x_continuous(labels = scales::comma)
```

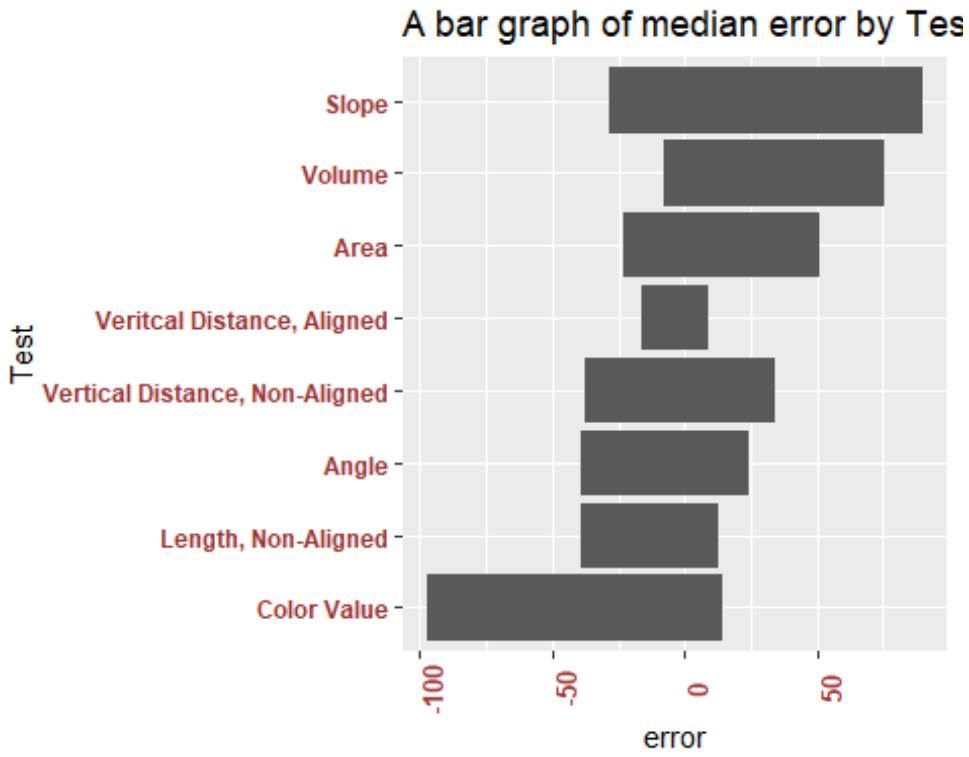
To be discussed in class.

a bar graph of the median Error by Test

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse
1.3.0 --

## v tibble  3.1.1     v purrr   0.3.4
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1

## Warning: package 'tibble' was built under R version 4.0.5

## Warning: package 'readr' was built under R version 4.0.3

## Warning: package 'stringr' was built under R version 4.0.3

## Warning: package 'forcats' was built under R version 4.0.3

## -- Conflicts ------------------------------------------
tidyverse_conflicts() --
## x lubridate::as.difftime() masks base::as.difftime()
## x readr::col_factor()      masks scales::col_factor()
## x lubridate::date()        masks base::date()
## x purrr::discard()         masks scales::discard()
## x dplyr::filter()          masks stats::filter()
## x lubridate::intersect()   masks base::intersect()
## x dplyr::lag()             masks stats::lag()
## x lubridate::setdiff()     masks base::setdiff()
## x lubridate::union()       masks base::union()
```

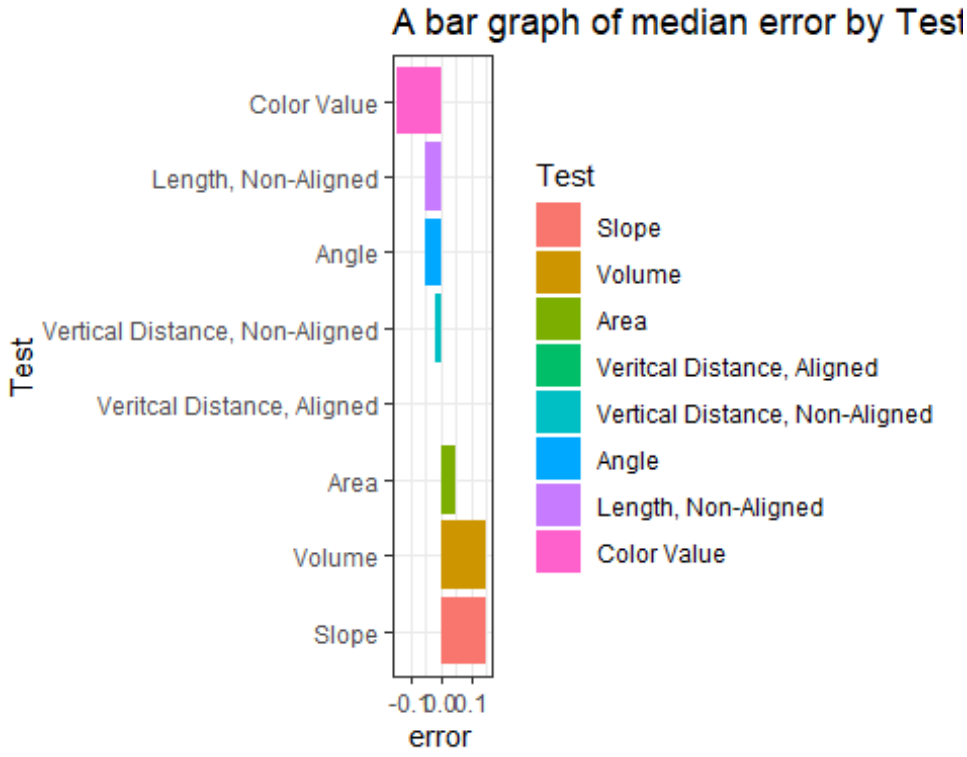
```
ggplot(perceptionExperiment, aes(x = fct_reorder(Test, error), y = error)) +
  geom_col() +
  theme(axis.text.x = element_text(face="bold",
                                           color="#993333"
                                           ,  size=10
                                           , angle=90
                                    ),
        axis.text.y = element_text(face="bold",
                                    color="#993333"
                                    )
  ) +
  labs(title = "A bar graph of median error by Test", x = "Test") +
  coord_flip()
```



#Repeat of 2b # a bar graph of the median Error by Test

```
library(tidyverse)
p <- ggplot(perceptionExperiment, aes(x=fct_reorder(Test, desc(error)),
y=error, fill=Test))
p <- p + stat_summary(fun.y="median", geom="bar") + theme_grey(base_size =
14)

## Warning: `fun.y` is deprecated. Use `fun` instead.

p <- p +  theme(axis.text.x = element_text(face="bold"
                                           ,  size=10
                                           , angle=90
                                    ),
        axis.text.y = element_text(face="bold"
                                    ),
        panel.grid.major.x = element_blank(), # Hide the vertical grid lines
        panel.grid.minor.x = element_blank(),

        panel.grid.major.y = element_blank(), # Hide the horizontal grid
lines
        panel.grid.minor.y = element_blank(),

        # Change the background color (fill) and border (colour)
        # and border thickness (size)of the legend box
        # We control the title text color, font size and font face
```

```
        legend.background = element_rect(fill = "white", colour = "grey",
size = 1),
        legend.title = element_text(colour = "blue", face = "bold", size =
14),
        legend.text = element_text(colour = "black"),
        legend.key = element_rect(colour = "black", size = 0.1)
  ) +
  labs(title = "A bar graph of median error by Test", x = "Test")
p <- p + scale_fill_discrete(limits = c("Slope", "Volume", "Area",  #
controlling the order of the legend
                                        "Veritcal Distance, Aligned",
                                        "Vertical Distance, Non-Aligned",
                                        "Angle", "Length, Non-Aligned", "Color
Value"))
p + scale_y_continuous(breaks = c(0.1, 0, -0.1)) + theme_bw()  + coord_flip()
```



#get variables of interest to derive the standard deviation

```
perceptionExperimentsdErrorbyTestdf <- perceptionExperiment %>%
  select(Test = Test,
         #TestNumber = TestNumber,
         error = error
         )
```

#Function to calculate the mean and the standard deviation for each group

```
data_summary <- function(data, varname, groupnames){
  require(plyr)
  summary_func <- function(x, col){
    c(mean = mean(x[[col]], na.rm=TRUE),
      sd = sd(x[[col]], na.rm=TRUE))
  }
  data_sum<-ddply(data, groupnames, .fun=summary_func,
                  varname)
  data_sum <- rename(data_sum, c("mean" = varname))
 return(data_sum)
}
```

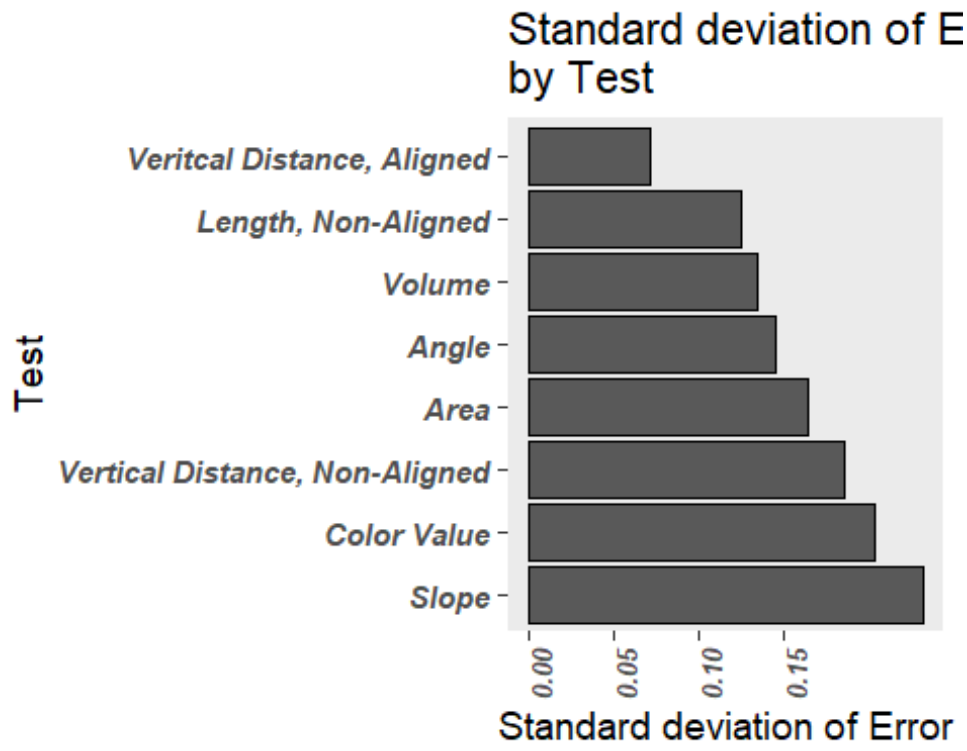
#Summarize the data set

```
perceptiondf <- data_summary(perceptionExperimentsdErrorbyTestdf,
                             varname = "error",
                             groupnames = c("Test")
                             )

## Loading required package: plyr

## ------------------------------------------------------------------------
----

## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first,
then dplyr:
## library(plyr); library(dplyr)

## ------------------------------------------------------------------------
----

##
## Attaching package: 'plyr'

## The following object is masked from 'package:purrr':
##
##     compact

## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

perceptiondf$Test <- as.factor(perceptiondf$Test)
```

#plot a bar graph of Standard deviation of the Error by Test

```
p <- ggplot(perceptiondf, aes(x = fct_reorder(Test, desc(sd)), y = sd)) +
  geom_col(colour="black") +
  expand_limits(y=0) + # Make the y-axis go to zero.
  theme_grey(base_size = 14) +
```

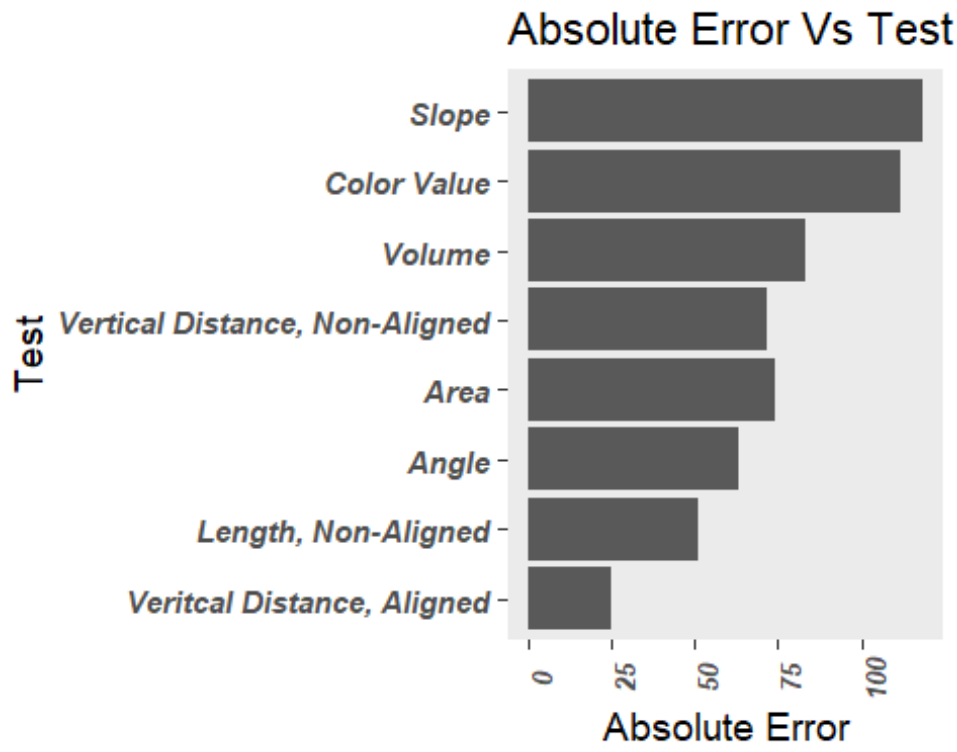
```
  theme(axis.text.x = element_text(face="bold.italic",
                                   size=10,
                                   angle=90),

        axis.text.y = element_text(face="bold.italic",
                                   #color="#993333"
                                   ),
        panel.grid.major.x = element_blank(), # Hide the vertical grid lines
        panel.grid.minor.x = element_blank(),

        panel.grid.major.y = element_blank(), # Hide the horizontal grid
lines
        panel.grid.minor.y = element_blank(),
        legend.position = "none"
        ) +
  labs(y  = "Standard deviation of Error", x = "Test", title = "Standard
deviation of Error\nby Test")
p <- p +  coord_flip() + scale_y_continuous(breaks = c(0.00, 0.05, 0.10,
0.15, 2.0))
p
```



a bar graph of the Absolute Error by Test

```
library(tidyverse)
perceptionExperiment <- perceptionExperiment %>% mutate(Test =
as.factor(Test)) %>% mutate(results = fct_reorder(Test, absoluteError)) #
```

```r
ordering the data

p <- ggplot(perceptionExperiment, aes(x = fct_reorder(Test, absoluteError), y
= absoluteError)) +
  geom_col() +
  expand_limits(y=0) + # Make the y-axis go to zero.
  theme_grey(base_size = 14) +
  theme(axis.text.x = element_text(face="bold.italic",
                                   size=10,
                                   angle=90),

        axis.text.y = element_text(face="bold.italic",
                                   ),
        panel.grid.major.x = element_blank(), # Hide the vertical grid lines
        panel.grid.minor.x = element_blank(),

        panel.grid.major.y = element_blank(), # Hide the horizontal grid
lines
        panel.grid.minor.y = element_blank(),
        legend.position = "none"
        )
p <- p +
  labs(x = "Test", title = "Absolute Error Vs Test", y = "Absolute Error") +
  coord_flip() + scale_y_continuous(breaks = c(0, 25, 50, 75, 100))
p
```



#Question number 3
##
##
##

#Load the Infant data set.

```
infantdata <- read.csv(file = "../HomeWork1/datasets/InfantData.csv", sep =
"," , header = TRUE )
```

#find an missing variables

```
sum(is.na(infantdata))

## [1] 0
```

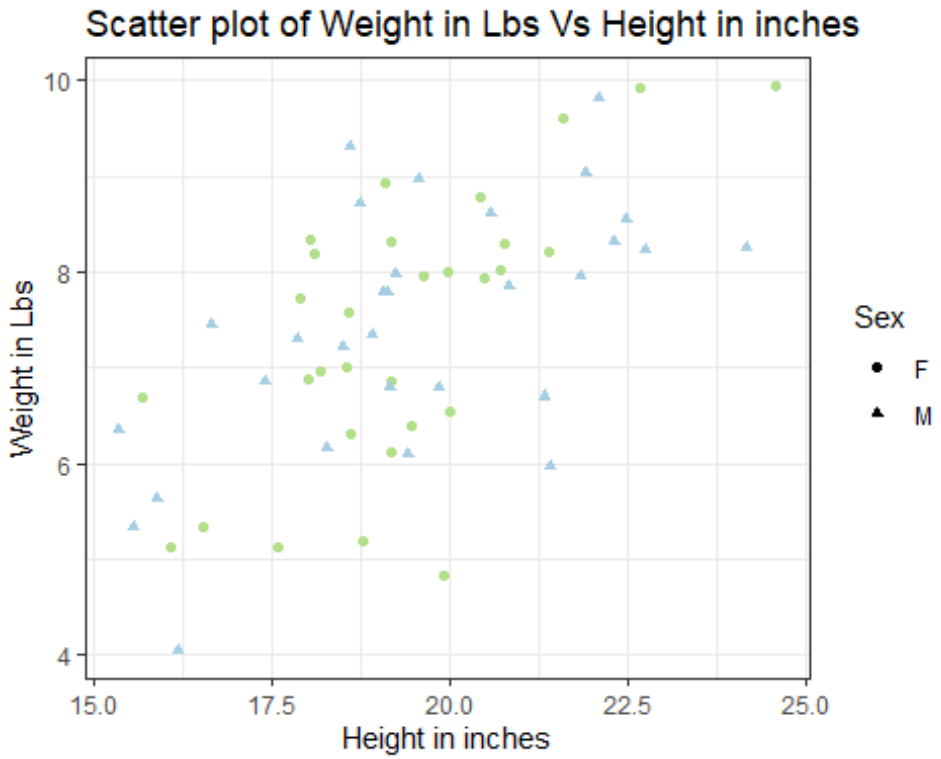
#Convert sex to a factor

```
infantdatads <- infantdata %>%
  transmute(Sex = Sex,
            HeightIn = HeightIn,
            WeightLbs = WeightLbs
  ) %>%
  mutate(Sex = as.factor(Sex),
         color = as.factor(case_when(Sex == "M" ~ "#a6cee3",
                                     Sex == "F"  ~  "#b2df8a")
                           )
  )
```

Scatter plot of Weightlbs Vs HeightIn

```
colr  <- as.character("blue", "green")
p <- ggplot(infantdatads,
       aes(x = HeightIn,
           y = WeightLbs
           , shape = Sex
           )
       ) +
  geom_point(color=infantdatads$color) + # set the size of the points to 1.5

  theme_grey(base_size = 14)# +  scale_fill_manual(values = c("#a6cee3",
"#1f78b4"))
p <- p + theme(axis.text.x = element_text(face="bold.italic",

                                     size=10
                                     ),
         axis.text.y = element_text(face="bold.italic",
                                     #color="#993333",
                                     size=10
```

```
                                    ),
        panel.grid.major.x = element_blank(), # Hide the vertical grid lines
        panel.grid.minor.x = element_blank(),

        panel.grid.major.y = element_blank(), # Hide the horizontal grid
lines
        panel.grid.minor.y = element_blank(),
        plot.title = element_text(hjust = 0.5,
                                  size = 14,
                                  face = "bold.italic"
          )
        )  + labs(title = "Scatter plot of Weight in Lbs Vs Height in
inches", x = "Height in inches", y = "Weight in Lbs") +
scale_color_manual(values = levels(infantdatads$color))
p <- p + scale_x_continuous(breaks = seq(15.0, 25.0, by = 2.5)) + theme_bw()
p
```



Scatter plot of Weightlbs Vs HeightIn

Adding separate single trend lines

```
p <- ggplot(infantdatads,
       aes(x = HeightIn,
           y = WeightLbs
           , colour = Sex
```

```
            , shape = Sex
            )
       ) +
  geom_point() + # set the size of the points to 1.5
  stat_smooth(method = "lm", se = FALSE) +
  labs(title = "Scatter plot of Weight in lbs Vs Height in inches\nwith
separate trend lines", x = "Height in inches", y = "Weight in Lbs") +
  theme_grey(base_size = 14)
p <- p + theme(axis.text.x = element_text(face="bold.italic",

                                          size=10
                                          ),
         axis.text.y = element_text(face="bold.italic",

                                          size=10
                                          ),
         panel.grid.major.x = element_blank(), # Hide the vertical grid lines
         panel.grid.minor.x = element_blank(),

         panel.grid.major.y = element_blank(), # Hide the horizontal grid
lines
         panel.grid.minor.y = element_blank(),
         )  + theme_bw()
p

## `geom_smooth()` using formula 'y ~ x'
```