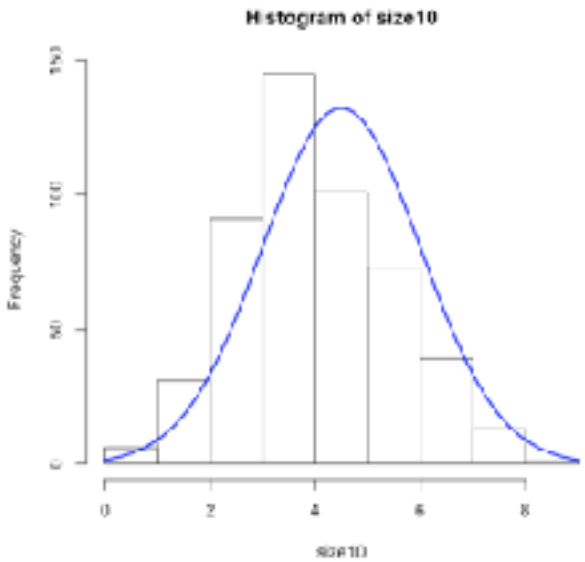
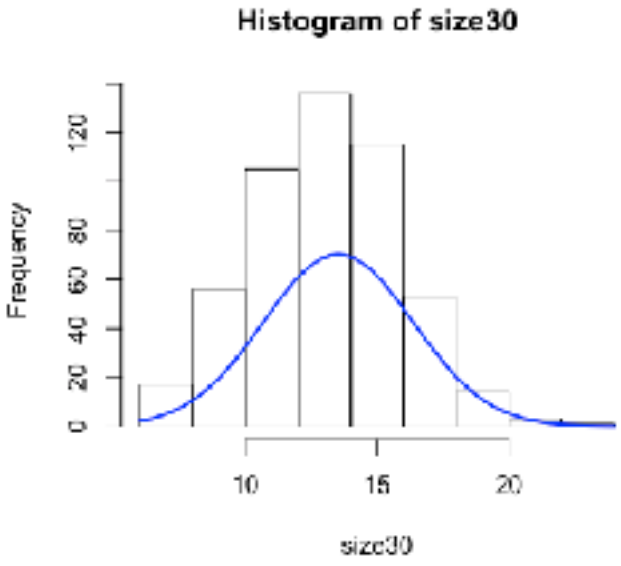
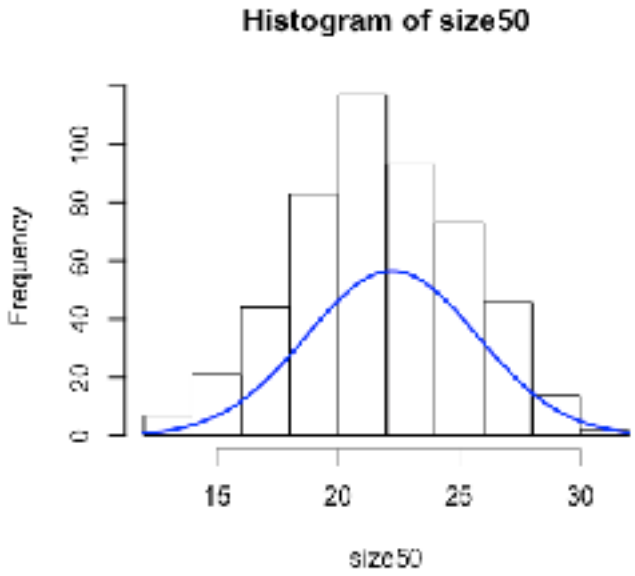
```
# HW 5 - Due Tuesday October 30, 2017 in moodle and hardcopy in class.
# Upload R file to Moodle with name: HW5_490IDS_52.R
# Do Not remove any of the comments. These are marked by #

# Please ensure that no identifying information (other than yur class ID)
# is on your paper copy, including your name

##########################Part 1##############################

#Q1.)

#For this problem we will start with a simulation in order to find out
#how large size needs to be for the binomial distribution to be
#approximated by the normal distribution.


#(a)(4pts) Let's let p=0.45, use the rbinom function to generate
# 500 observations(n = 500)
#Use 10, 30, and 50 for number of trials(size = 10,30,50).
#Add normal curves to all of the plots.
#Display the three histograms as well as your code below.
#Also comment something on the shape of the histogram for each plot
set.seed(000)
size10 = rbinom(500, size = 10, p = 0.45)
size30 = rbinom(500, size = 30, p=0.45)
size50 = rbinom(500,50,0.45)


normalplot = function(size){
  x = seq(min(size),max(size),length=500)
  y = dnorm(x, mean=mean(size),sd=sd(size))
  y = y*diff(his$mids[1:2])*length(size)
  lines(x,y,col="blue", lwd=2)
}

par(mfcol = c(1,1))
hist(size10)
normalplot(size10)
# The size of 10 histogram has little number of bars, and quite difderent from its normal curve,
# unable to clearly and correctly match with a normal distribution.
```



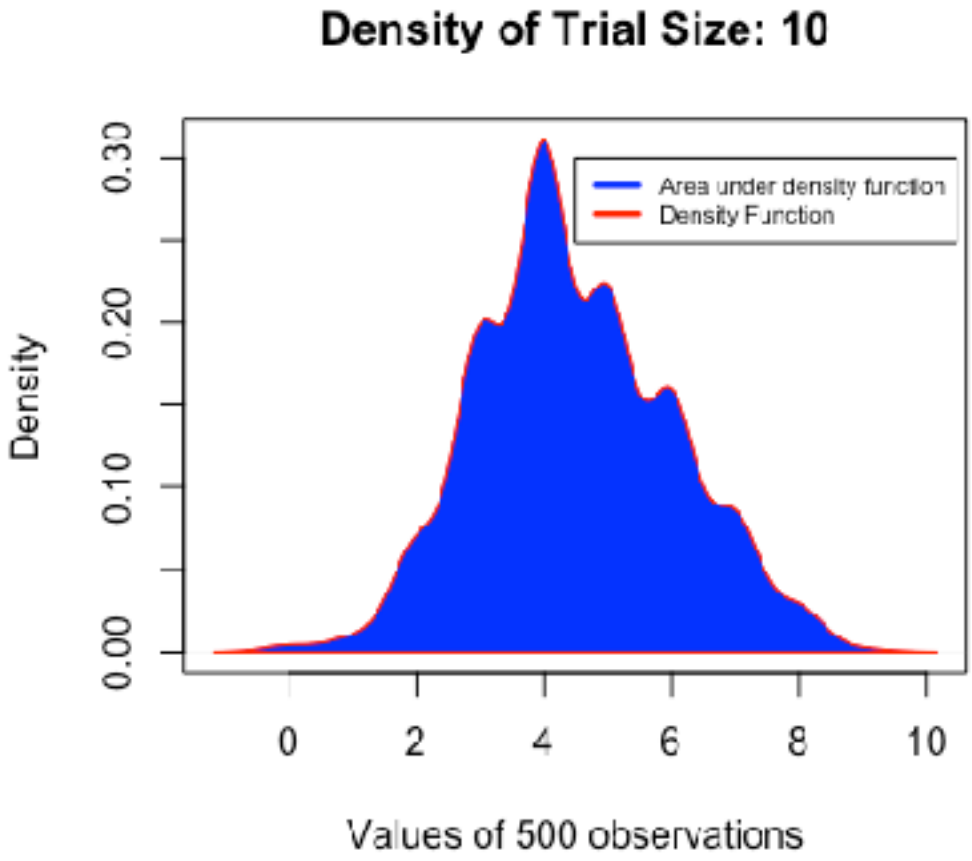
```
hist(size30)
normalplot(size30)
# The size of 30 histogram has better shaping and more bars. We can also se that it is alot more
# similarity to the normal distribution.
```



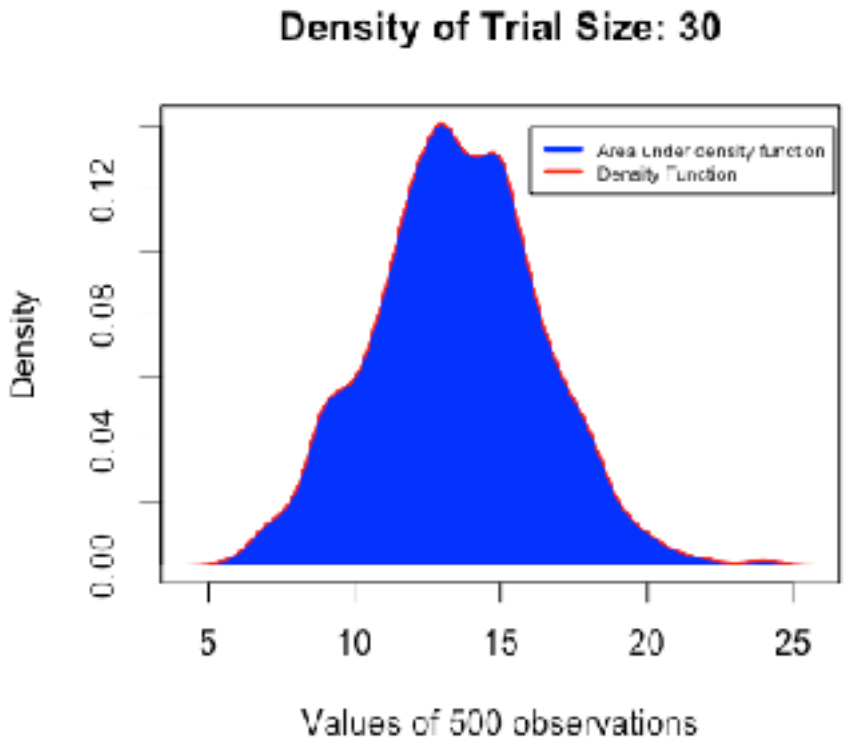

```
hist(size50)
normalplot(size50)
# The size of 50 histogram has the best fit to a normal distribution with the largest x axis.
```



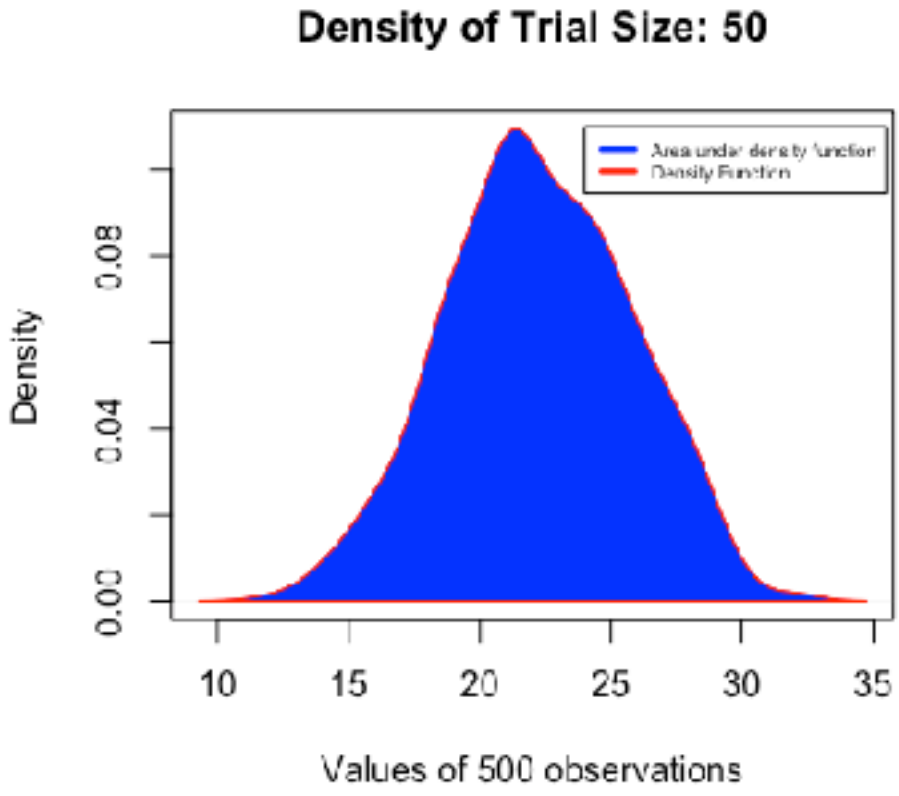
```
#(b)(3pts.) Now use the techniques described in class to improve graphs.
# Explain each step you choose including why you are making the change. You
# might consider creating density plots, changing color, axes,
# labeling, legend, and others for example.

# self define functions so would not need to constantly copy and paste
normalcurves = function(dist, size){
  plot(density(dist), col='red', main = paste(c("Density of Trial Size:", size), collapse = " "),
       xlab = "Values of 500 observations")
  return(density(dist))
}

putlegend = function(x,y,legendsize){
  legend(x,y, c("Area under density function","Density Function"), lty=c(1,1),
         lwd=c(2.5,2.5),col=c("blue","red"),
         cex = legendsize)}

# create just the density plot to show an overview of the distribution, do not want to add the
# histogram bars because it will become clotted and distracting to look at.

# Color the suface beneath the density plot for direct understanding for viewers of the area,
# use different colors for the density function and the area beneath it so it can be easily
# recognized. Added understandable labels for x and y axis and the main plot name that include
the
# trail size. Also add legends in the end so readers can understand what the two colors
represents.


size10improved = normalcurves(size10, 10)
polygon(size10improved, col = "blue", border = “red”)
putlegend(4.5,0.3,0.62)
```



```
size30improved = normalcurves(size30, 30)
polygon(size30improved, col = "blue", border = "red")
putlegend(16,0.14,0.57)
```




```
size50improved = normalcurves(size50, 50)
polygon(size50improved, col="blue", border = “red")
putlegend(24,0.11,0.53)
```



```
#Q2.) (2pts.)
#Why do you think the Data Life Cycle is crucial to understanding the opportunities
#and challenges of making the most of digital data? Give two examples.

# (Q2.)
# Ans:
# 1. The data life cycle goes through the whole process of obtaining the 'correct' data for analysis
# to find underlying information within these data. Exp, businesses obtain client data from within
# its own database in correct format to start applying different statistical or machine learning
# models to use the computation power of nowadays computers to find underlying information
such
# as what kind of clients with what type of attributes may bring business more revenues.

# 2. Every process in the Data life cycle is considered crucial to finding important information.
# Out of all the process, the data visualization can play a important role and happen in every
# stage of the data life cycle. It can quickly give readers insight to the data as a whole and
# what information it can convey. Exp, when data scientists use data visualization to check if there
# are any abnormalities within the processed data, whether extreme outliers exist that need to
reunderstand
# or what distribution does the data have and what kind of model can be applied to them.)


##########################Part 2##############################

#Q3.)Install and Load the library ElemStatLearn
# Load the data prostate into R.
install.packages("ElemStatLearn")
library(ElemStatLearn)
data(prostate)

#(a) (1 pt) Print out the structure of the dataset using str()
str(prostate)
### OutPut ###
# > str(prostate)
# 'data.frame':97 obs. of  10 variables:
# $ lcavol : num  -0.58 -0.994 -0.511 -1.204 0.751 ...
# $ lweight: num  2.77 3.32 2.69 3.28 3.43 ...
# $ age    : int  50 58 74 58 62 50 64 58 47 63 ...
# $ lbph   : num  -1.39 -1.39 -1.39 -1.39 -1.39 ...
# $ svi    : int  0 0 0 0 0 0 0 0 0 0 ...
# $ lcp    : num  -1.39 -1.39 -1.39 -1.39 -1.39 ...
# $ gleason: int  6 6 7 6 6 6 6 6 6 6 ...
# $ pgg45  : int  0 0 20 0 0 0 0 0 0 0 ...
# $ lpsa   : num  -0.431 -0.163 -0.163 -0.163 0.372 ...
# $ train  : logi  TRUE TRUE TRUE TRUE TRUE TRUE ...


#(b)(1 pt) What are the names of the features in the dataset?
names(prostate)

### OutPut ###
# > names(prostate)
# [1] "lcavol"  "lweight" "age"     "lbph"    "svi"     "lcp"     "gleason" "pgg45"   "lpsa"
# [10] "train"


# (c). (2 pts.)
# The first eight variables are the our predictors. Subset the dataset with
# these eight variables and name it X.
```

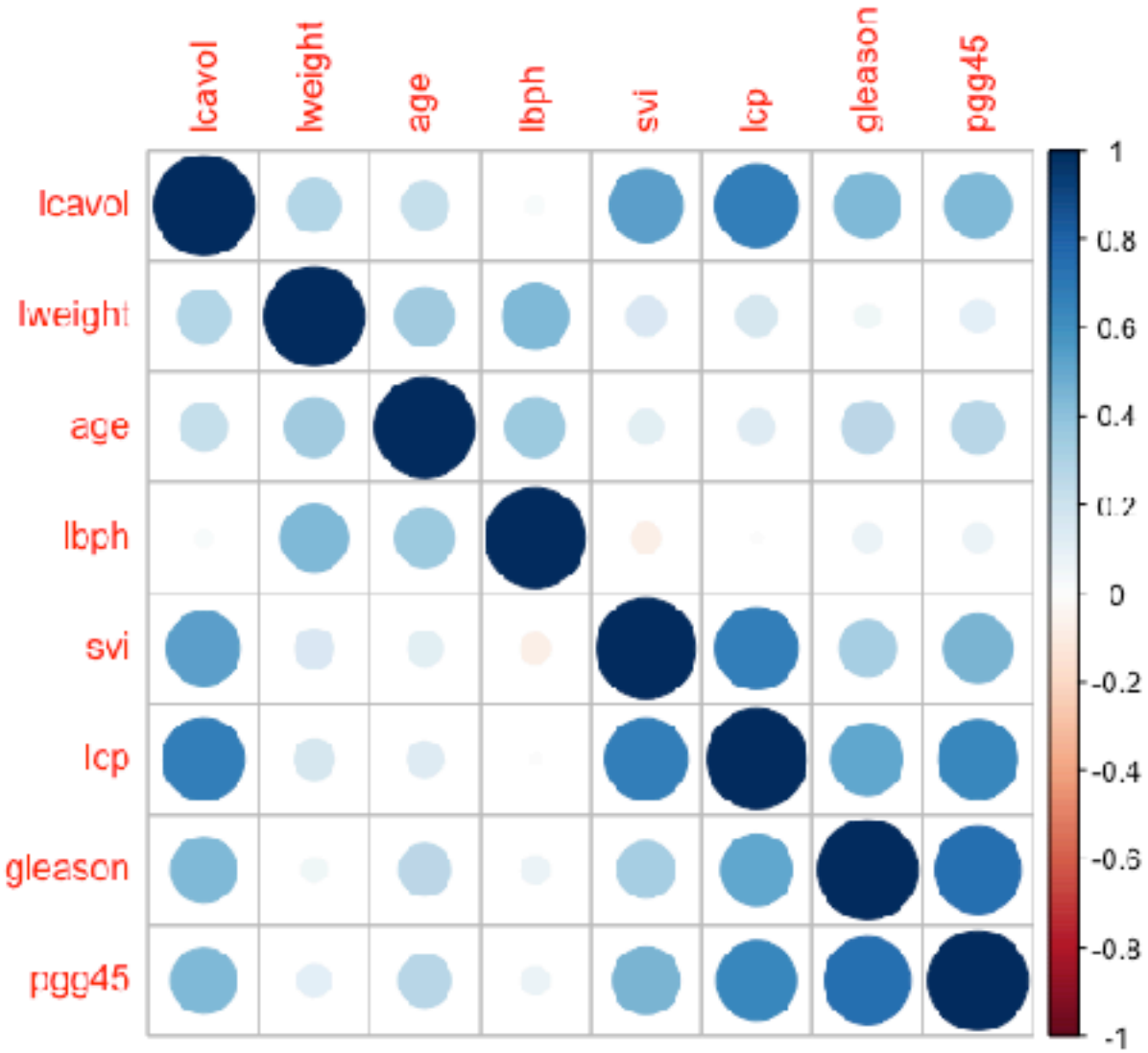
```
X = prostate[,1:8]

### Partial Output ###
# > prostate[,1:8]
#          lcavol  lweight age       lbph svi       lcp gleason pgg45
# 1  -0.579818495 2.769459  50 -1.38629436   0 -1.38629436       6     0
# 2  -0.994252273 3.319626  58 -1.38629436   0 -1.38629436       6     0
# 3  -0.510825624 2.691243  74 -1.38629436   0 -1.38629436       7    20
# 4  -1.203972804 3.282789  58 -1.38629436   0 -1.38629436       6     0
# 5   0.751416089 3.432373  62 -1.38629436   0 -1.38629436       6     0
# 6  -1.049822124 3.228826  50 -1.38629436   0 -1.38629436       6     0
# 7   0.737164066 3.473518  64  0.61518564   0 -1.38629436       6     0
# 8   0.693147181 3.539509  58  1.53686722   0 -1.38629436       6     0
# 9  -0.776528789 3.539509  47 -1.38629436   0 -1.38629436       6     0
# 10  0.223143551 3.244544  63 -1.38629436   0 -1.38629436       6     0

# (d) (2 pts)
# Name the correlation matrix for X CorX and print it out.
CorX = cor(X)
CorX
### Output ###
# > CorX
#           lcavol   lweight       age         lbph         svi         lcp    gleason      pgg45
# lcavol  1.0000000 0.2805214 0.2249999  0.027349703  0.53884500  0.675310484 0.43241706
0.43365225
# lweight 0.2805214 1.0000000 0.3479691  0.442264395  0.15538491  0.164537146 0.05688210
0.10735379
# age     0.2249999 0.3479691 1.0000000  0.350185896  0.11765804  0.127667752 0.26889160
0.27611245
# lbph    0.0273497 0.4422644 0.3501859  1.000000000 -0.08584324 -0.006999431 0.07782045
0.07846002
# svi     0.5388450 0.1553849 0.1176580 -0.085843238  1.00000000  0.673111185 0.32041222
0.45764762
# lcp     0.6753105 0.1645371 0.1276678 -0.006999431  0.67311118  1.000000000 0.51483006
0.63152825
# gleason 0.4324171 0.0568821 0.2688916  0.077820447  0.32041222  0.514830063 1.00000000
0.75190451
# pgg45   0.4336522 0.1073538 0.2761124  0.078460018  0.45764762  0.631528246 0.75190451
1.00000000


# (e) (2 pts)
# Find out the pair of variables with largest correlation
```

Ans:

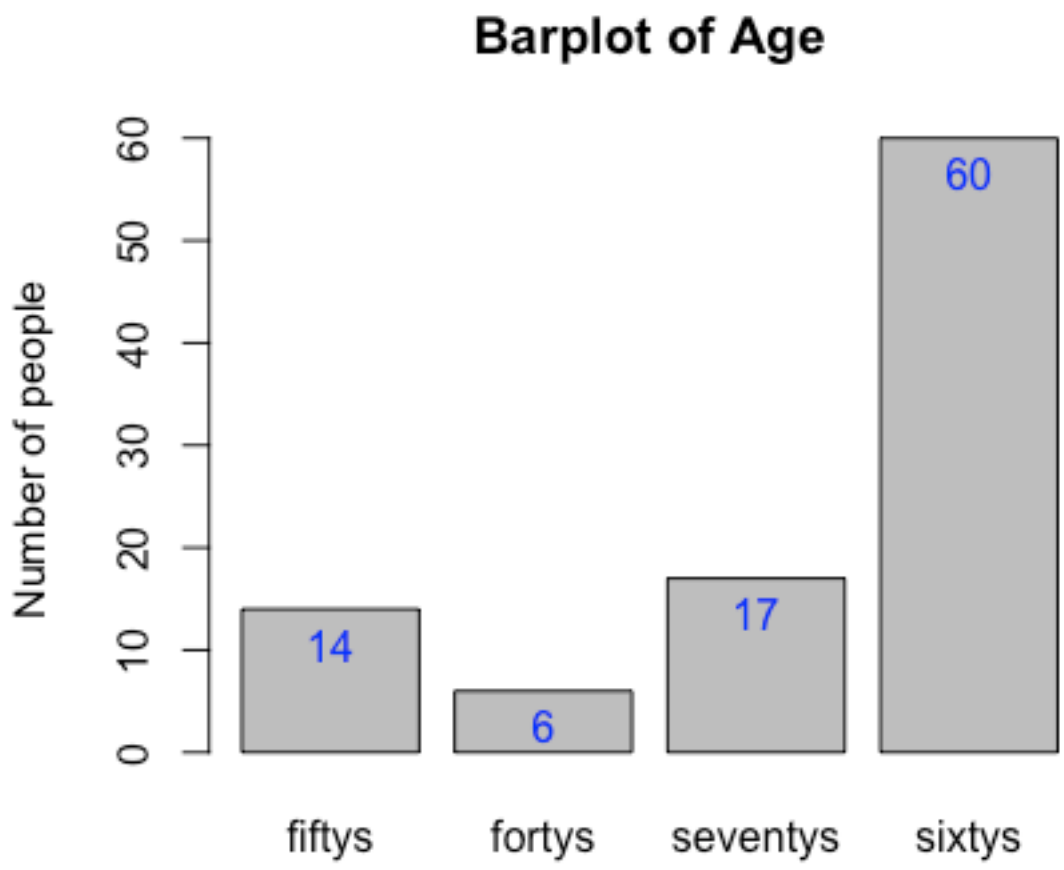
 # From the Above CorX output we can see that in the matrix, the largest value of correlation between
 # two different values is 0.75190451, and the pair of variables that have this value is
 # "pgg45" and "gleason".

```
#### Extra Self Practice to write Code to find the to variable names with largest correlation ####
maxCor = c()
for(i in 1:dim(CorX)[1]){
  maxCor[i] = max(CorX[i,CorX[i,] != 1])
}
maxCorVal = max(maxCor)
maxCorVal
### OutPut ###
# > maxCorVal
# [1] 0.7519045
```

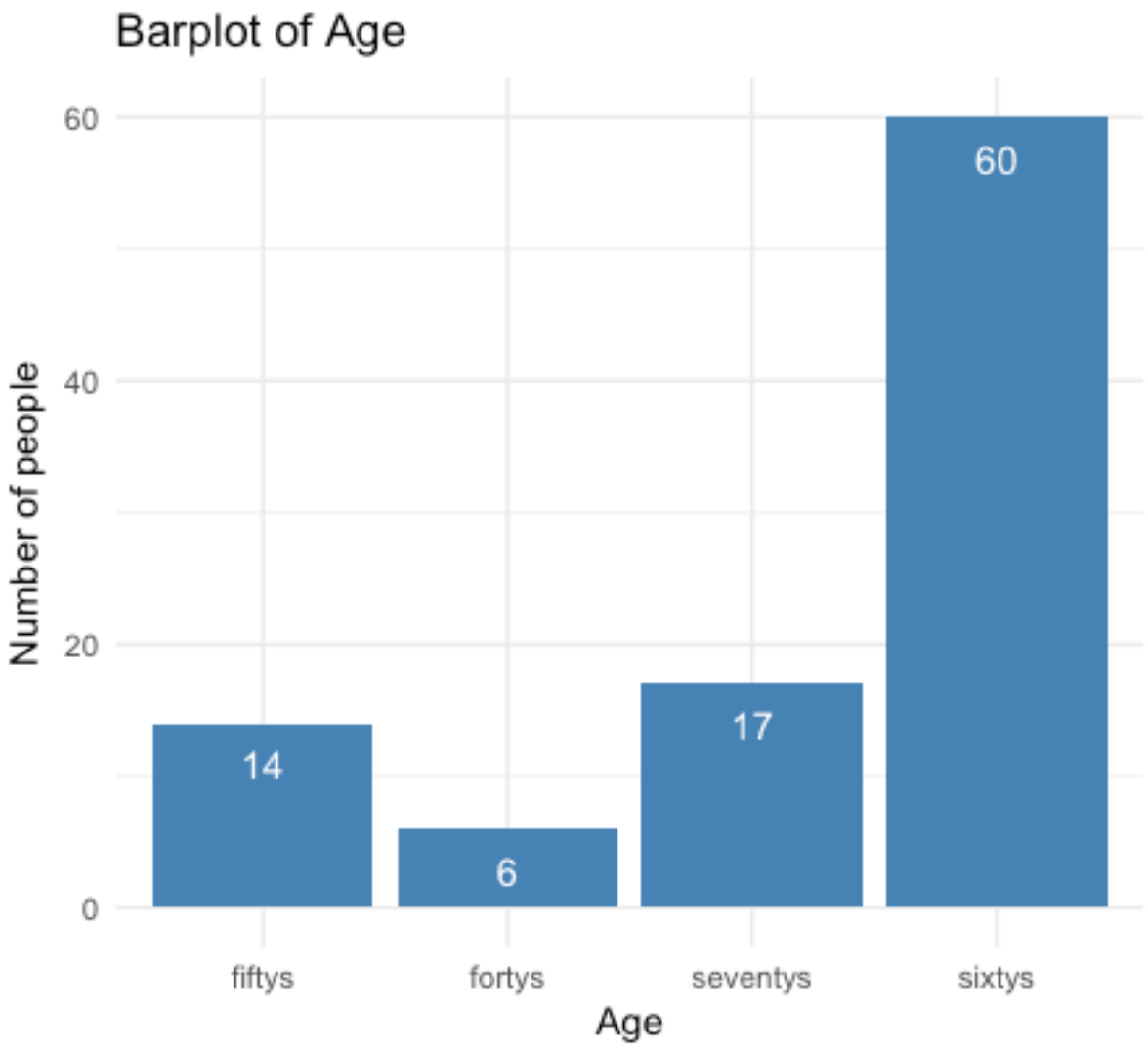
```
for(row in 1:8){
  for(col in 1:8){
    if(CorX[row,col] == maxCorVal){
      maxValuePosition = c(row,col)
    }
  }
}
rownames(CorX)[maxValuePosition[1]]
colnames(CorX)[maxValuePosition[2]]
### OutPut Ans ### ===> The same
# > rownames(CorX)[maxValuePosition[1]]
# [1] "pgg45"
# > colnames(CorX)[maxValuePosition[2]]
# [1] "gleason"


# (f) (2 pts)
# Use corrplot function from package corrplot to make a correlation plot
# install.packages("corrplot")
install.packages("corrplot")
library(corrplot)
corrplot(CorX)
```



```
# (f) (3 pts.)
# Create a new variable called age_group:
# if 40 < age < 50, set it to fortys, if 50 < age < 60, set it to fiftys
# if 60 < age < 70, set it to sixtys, if 70 < age < 80, set it to seventys.
# Make a barplot for this variable.


age_group = X
40 < age_group$age & age_group$age < 50
age_group$age[40 <= age_group$age & age_group$age < 50] = "fortys"
age_group$age[50 <= age_group$age & age_group$age < 60] = "fiftys"
age_group$age[60 <= age_group$age & age_group$age < 70] = "sixtys"
age_group$age[70 <= age_group$age & age_group$age < 80] = "seventys"

counts = table(age_group$age)
x_lab = barplot(counts, main="Barplot of Age", ylab="Number of people")
# Adding values on the bars of barplot for easier readability
text(x = x_lab, y = counts, label = counts, pos = 1, cex = 1, col = “blue")
```

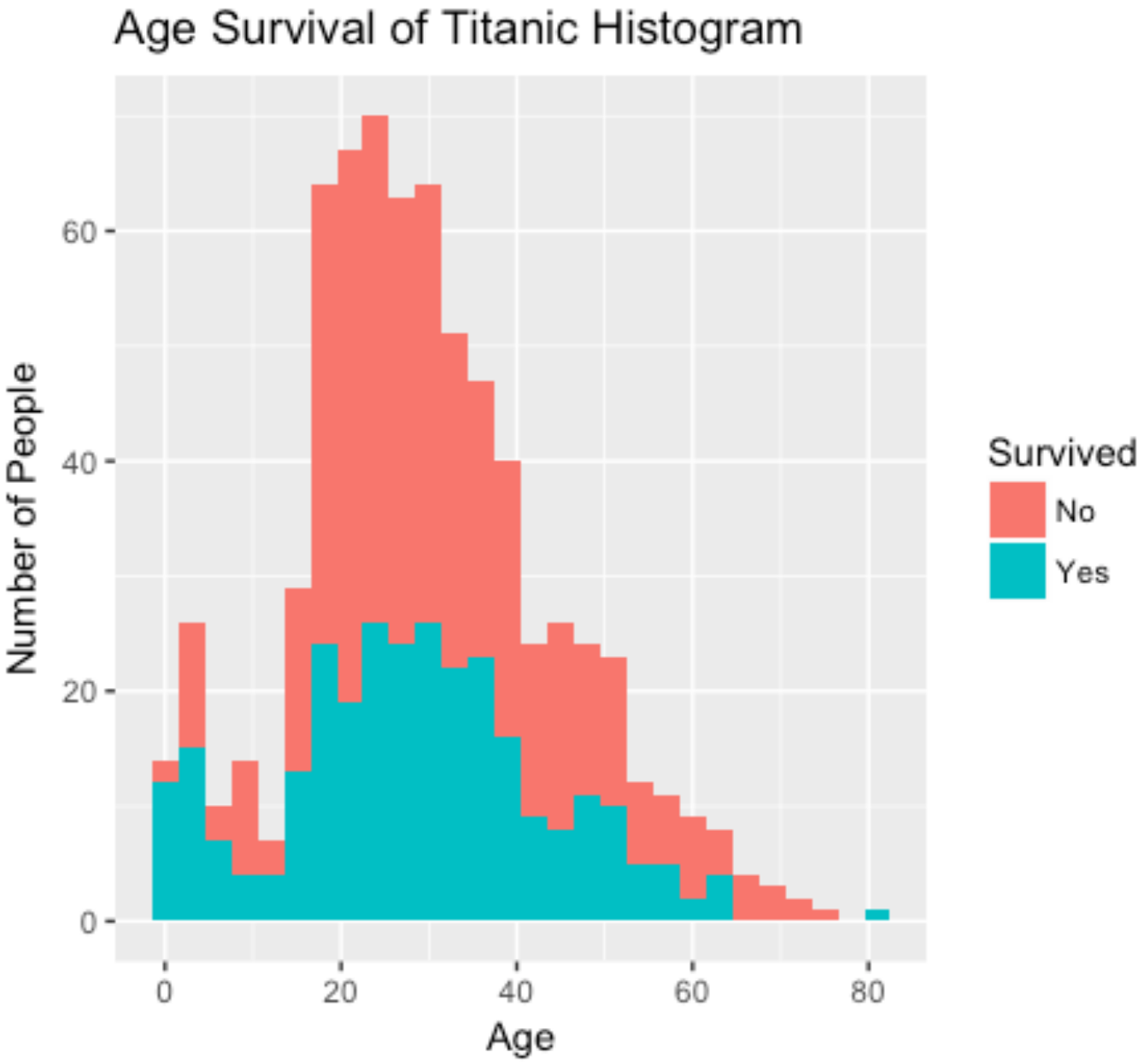


```
#(g) Bonus (1 pt) Can you try to make the same plot using ggplot?
library(ggplot2) ###################
lab = c(sum(age_group$age == "fortys"), sum(age_group$age == "fiftys"),
        sum(age_group$age == "sixtys"), sum(age_group$age == "seventys"))
ggplot(age_group, aes(x = age_group$age)) + geom_bar(fill="steelblue") +
  geom_text(stat='count',aes(label=..count..),vjust=2,color = "white") + theme_minimal()+
  labs(title = "Barplot of Age", x = "Age", y="Number of people")
```



```
##########################Part 3#############################
#You can use either ggplot or normal plot method for plotting.

#4.)

#Load the dataset "titanic.csv" you downloaded from moodle, omit the NA using
#na.omit(). There are 714 observations after removing the missing values.
titanic_data = read.csv("Titanic.csv")
titanic_data = na.omit(titanic_data)
dim(titanic_data)
### OutPut ###
# > dim(titanic_data)
# [1] 714  12

#a.)(1 pt) First print out the str() of the dataset and think about
#          which variable need to be converted to categorical.
str(titanic_data)
### OutPut ###
# > str(titanic_data)
# 'data.frame':714 obs. of  12 variables:
#   $ PassengerId: int  1 2 3 4 5 7 8 9 10 11 ...
# $ Survived   : int  0 1 1 1 0 0 0 1 1 1 ...
# $ Pclass     : int  3 1 3 1 3 1 3 3 2 3 ...
# $ Name       : Factor w/ 891 levels "Abbing, Mr. Anthony",..: 109 191 354 273 16 516 625 413
577 728 ...
# $ Sex        : Factor w/ 2 levels "female","male": 2 1 1 1 2 2 2 1 1 1 ...
# $ Age        : num  22 38 26 35 35 54 2 27 14 4 ...
# $ SibSp      : int  1 1 0 1 0 0 3 0 1 1 ...
# $ Parch      : int  0 0 0 0 0 0 1 2 0 1 ...
# $ Ticket     : Factor w/ 681 levels "110152","110413",..: 524 597 670 50 473 86 396 345 133 617
...
# $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
# $ Cabin      : Factor w/ 148 levels "","A10","A14",..: 1 83 1 57 1 131 1 1 1 147 ...
# $ Embarked   : Factor w/ 4 levels "","C","Q","S": 4 2 4 4 4 4 4 4 2 4 ...
# - attr(*, "na.action")=Class 'omit'  Named int [1:177] 6 18 20 27 29 30 32 33 37 43 ...
# .. ..- attr(*, "names")= chr [1:177] "6" "18" "20" "27" ...
##################
# Ans: I would think of changing the variable "Survived" into categorical data of "Lived" or "Died"


#b.(2pt) If Survived = 0, change it to "No", else change it to "Yes".
#        Replace the "Yes/No" variable to original "Survived" Variable
#       Also convert Pclass to factor.
titanic_data$Survived = ifelse(titanic_data$Survived == 0, "No", "Yes")
### OutPut (Ommitted middle part for space saving) ###
# > titanic_data$Survived
# [1] "No"  "Yes" "Yes" "Yes" "No"  "No"  "No"  "Yes" "Yes" "Yes" "Yes" "No"  "No"  "No"  "Yes"
"No"
# [17] "No"  "No"  "Yes" "Yes" "Yes" "No"  "Yes" "No"  "No"  "No"  "No"  "No"  "No"  "No"  "Yes"
"No"
# .......
# [689] "No"  "Yes" "No"  "Yes" "Yes" "No"  "Yes" "No"  "Yes" "No"  "No"  "Yes" "Yes" "No"  "No"
"Yes"
# [705] "Yes" "No"  "No"  "No"  "No"  "No"  "No"  "Yes" "Yes" "No"
titanic_data$Pclass = as.factor(titanic_data$Pclass)
class(titanic_data$Pclass)
### OutPut ###
# > class(titanic_data$Pclass)
# [1] "factor"
```

```
#c.)(3 pt) Make a histogram for the Age variable and fill the
#         histogram by Survived Variable.
#The plot is similar to
#http://ggplot2.tidyverse.org/reference/geom_freqpoly-9.png

ggplot(titanic_data, aes(x = titanic_data$Age, fill = titanic_data$Survived)) +
  geom_histogram(binwidth = 3) +
  labs(title = "Age Survival of Titanic Histogram", x = "Age",
      y = "Number of People", fill = “Survived")
```



```
#d.) (4 pts.)
#Make a mosaic plot with Sex, Pclass, Survived variables.
#Google is your friend if you find yourself stuck.
library(vcd)
Pclass = titanic_data$Pclass
Sex = titanic_data$Sex
Survived = titanic_data$Survived
mosaic(~ Sex + Survived + Pclass,
       data = titanic_data,
       main = "Titanic MosaicPlot",
       shade = T)
```