Statistics Project in R

Yuping Qi

Part 1 The CSV data file AmericaCorp shows the Fortune 500 rankings of America's largest corporations for 2010. Next to each corporation are its market capitalization (in billions of dollars) and its total return (in percentage) to investors for the year 2009.

If you have to compare the variation in market capitalization and the total return to investors, which measure of variation will you use and why?

Compute the measure of variation you identified and state which sample data exhibits greater relative variability.

```
AmericaCorp <- read.csv("AmericaCorp.csv")
head(AmericaCorp)
```

```
##            Company MktCap Return
## 1         Wal-Mart    209   -2.7
## 2      Exxon Mobil    314  -12.6
## 3          Chevron    149    8.1
## 4 General Electric    196   -0.4
## 5  Bank of America    180    7.3
## 6   ConocoPhillips     78    2.9
```

I will use coefficient of variation to compare the variation in market capitalization and the total return to investors. I choose cv because it compares dispersion in data sets with dissimilar units of measurement or dissimilar means. It is a unit-free measure which adjusts for differences in the magnitudes of the means.

```
#Create the coefficient of variation function
CV_in_Percent <- function(x) {
  (sd(x, na.rm = TRUE)/mean(x, na.rm = TRUE))*100
}

#Compute the cv for the two groups of sample data
a  <- CV_in_Percent(AmericaCorp$MktCap)
b  <-  CV_in_Percent(AmericaCorp$Return)
print(cat("The market capitalization variation is ", a))
```

```
## The market capitalization variation is  45.09853NULL
```

```
print(cat ("The total return to investors has variation ", b))
```

```
## The total return to investors has variation  258.1025NULL
```

```r
#Compare cv
if(a>b){
  cat("The Market Capitalization has greater variation.")
} else {
  cat("The total return to investors has greater variation.")
}
```

```
## The total return to investors has greater variation.
```

Part 2 The monthly closing values of the Dow Jones Industrial Average (DJIA) for the period beginning in January 1950 are given in the CSV file Dow. According to Wikipedia, the Dow Jones Industrial Average, also referred to as the Industrial Average, the Dow Jones, the Dow 30, or simply the Dow, is one of several stock market indices created by Charles Dow. The average is named after Dow and one of his business associates, statistician Edward Jones. It is an index that shows how 30 large, publicly owned companies based in the United States have traded during a standard trading session in the stock market. It is the second oldest U.S. market index after the Dow Jones Transportation Average, which Dow also created.

The Industrial portion of the name is largely historical, as many of the modern 30 components have little or nothing to do with traditional heavy industry. The average is price-weighted, and to compensate for the effects of stock splits and other adjustments, it is currently a scaled average. The value of the Dow is not the actual average of the prices of its component stocks, but rather the sum of the component prices divided by a divisor, which changes whenever one of the component stocks has a stock split or stock dividend, so as to generate a consistent value for the index.

Along with the NASDAQ Composite, the S&P 500 Index, and the Russell 2000 Index, the Dow is among the most closely watched benchmark indices for tracking stock market activity. Although Dow compiled the index to gauge the performance of the industrial sector within the U.S. economy, the index's performance continues to be influenced not only by corporate and economic reports, but also by domestic and foreign political events such as war and terrorism, as well as by natural disasters that could potentially lead to economic harm.

```r
dow <- read.csv("Dow.csv")
head(dow)
```

```
##    Month ClosingValue
## 1 Jan-50       201.79
## 2 Feb-50       203.44
## 3 Mar-50       206.05
## 4 Apr-50       213.56
## 5 May-50       223.42
## 6 Jun-50       209.11
```

a. Compute the average Dow return over the period (geometric mean) given on the dataset.

```r
# Create a time-series object
timeseries <- ts(dow$ClosingValue, frequency = 1, start = 1)

# Compute the growth factors
One_Plus_Returns <- timeseries/lag(timeseries, - 1)

# Calculate the Geometric Mean and Display
# install.packages(psych)
library(psych)
```

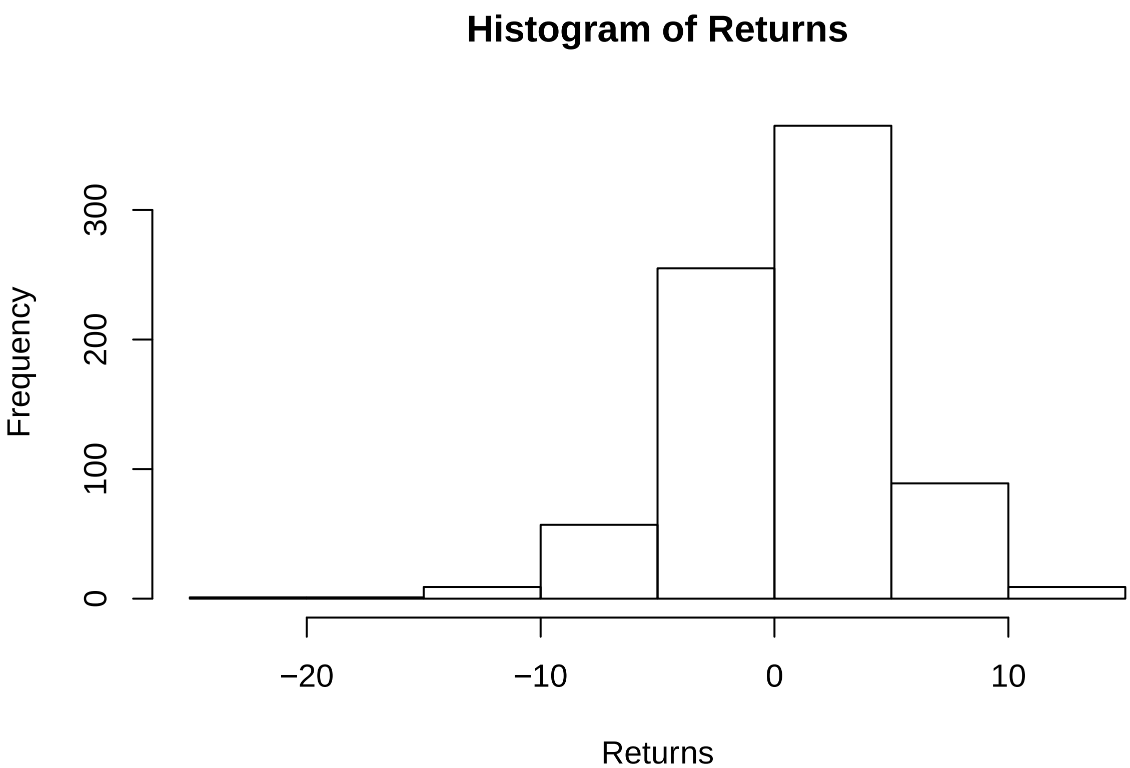
```
## Warning: package 'psych' was built under R version 3.6.2
```

```
Geometric_Mean_Rate <- round(100*(geometric.mean(One_Plus_Returns)-1), digits = 2)
Geometric_Mean_Rate
```

```
## [1] 0.57
```

b. Plot a histogram and intuitively comment on whether or not the returns are normally distributed

```
Returns <- 100*(One_Plus_Returns - 1)
hist(Returns)
```



Intuitively, the returns are normally distributed.

c. Verify if the returns adhere to the empirical rules

```
avg <- mean(Returns)
stddev <- sd(Returns)
TotalSampleSize <- length(Returns)

Percent_Within_a_Range <- function(x){
  Lower_Range = avg - x*stddev
  Upper_Range = avg + x*stddev
  Num_Within <- length(which(Returns >= Lower_Range
                            & Returns <= Upper_Range))
  list(
    Percent_Within <- 100*(Num_Within/TotalSampleSize)
  )
```

```
}

Percent_Within_a_Range(1)
```

```
## [[1]]
## [1] 72.01018
```

```
Percent_Within_a_Range(2)
```

```
## [[1]]
## [1] 95.54707
```

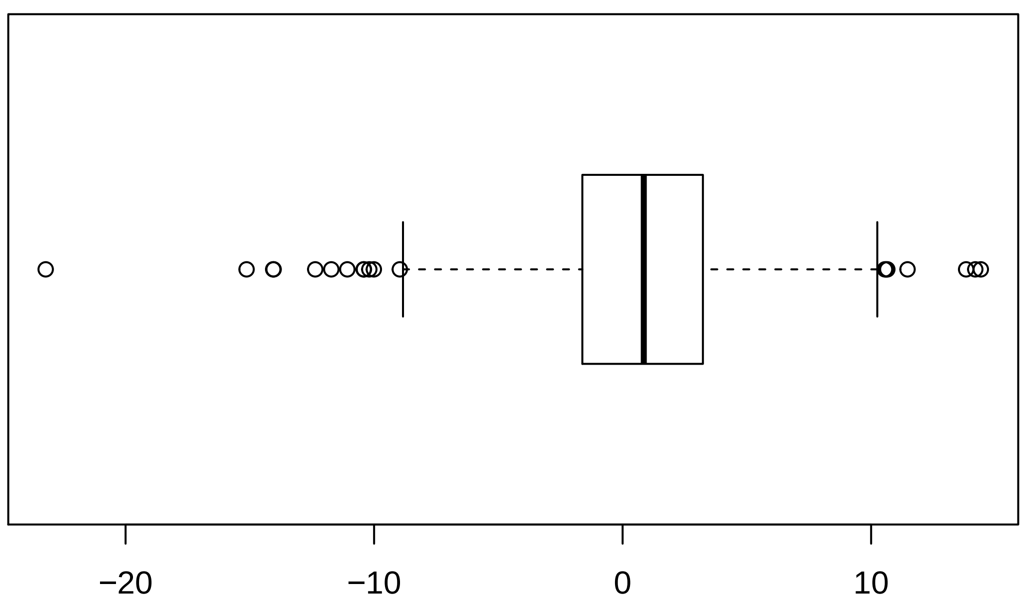
```
Percent_Within_a_Range(3)
```

```
## [[1]]
## [1] 98.85496
```

The result indicates that the return adheres to the empirical rule.

d. Plot a boxplot of returns and the five-number summary, and interpret the first, second, and third quartiles

```
boxplot(Returns, horizontal = TRUE)
```




```
summary(Returns)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -23.2159  -1.6191   0.8520   0.6566   3.2264  14.4144
```

The first quartile is as follows, 25% of the numbers in the data set lie below Q1 and about 75% lie above Q1

```r
(Q1 <- quantile(Returns, 0.25))
```

```
##       25%
## -1.619052
```

The second quartile is as follows, and it is the median of the dataset

```r
(Q2 <- quantile(Returns, 0.50))
```

```
##       50%
## 0.8519632
```

The third quartile is as follows, and 75% of the numbers in the data set lie below Q3 and about 25% lie above Q3

```r
(Q3 <- quantile(Returns, 0.75))
```

```
##     75%
## 3.22641
```

e. Identify the mild and extreme outlier returns using the IQR. Include whether each monthly return is a mild, an extreme outlier, or not an outlier. Sort the results displaying the outliers first.

```r
#Identify mild outliers.
IQR <- Q3-Q1
returns <- data.frame(Returns)
outliervector <- with(returns, returns$Outlier <-
                        ifelse(Returns >= Q3 + 1.5*IQR | Returns <= Q1 - 1.5*IQR, "Yes", "No"))
returns["Outlier"] <- outliervector

#Identify extreme outliers.
extremeoutliervector <- with(returns, returns$ExtremeOutlier <-
                                ifelse(Returns >= Q3+ 3*IQR | Returns <= Q1 - 3*IQR, "Yes", "No"))
returns["ExtremeOutlier"] <- extremeoutliervector

library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.6.2
```

```
##
## 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
```

```r
View_Data_FINAL <- returns %>%
  group_by(Outlier) %>%
  arrange(desc(Outlier)) %>%
  group_by(ExtremeOutlier) %>%
  arrange(desc(ExtremeOutlier))

returns_outlier_first <- as.data.frame(View_Data_FINAL)
head(returns_outlier_first)
```

```
##     Returns Outlier ExtremeOutlier
## 1 -23.21591     Yes            Yes
## 2 -14.04274     Yes             No
## 3 -10.41020     Yes             No
## 4 -10.42029     Yes             No
## 5  14.19090     Yes             No
## 6  14.41442     Yes             No
```

Part 3 The RealEstateMaster and RealEstateMissing CSV datasets for one of the towns in the state of AZ is given. Prices are removed for homes which are greater than 15 miles away from the center (missing values). The variables in the dataset are:

Price: Selling Price in $000 Bedrooms: No. of bedrooms Size: Size of the home in square feet Pool: Pool (1 = yes, 0 = no) Distance: Distance from the center of the city in miles Township Garage: Garage Attached (1 = yes, 0 = no) Baths: No. of bathrooms

a. Use median imputation technique and fill in the missing values. Also provided is the master dataset containing all the prices. See how well median imputes the missing values when compared with the original values.

```r
library(Hmisc)
```

```
## Warning: package 'Hmisc' was built under R version 3.6.2

## Loading required package: lattice

## Warning: package 'lattice' was built under R version 3.6.2

## Loading required package: survival

## Warning: package 'survival' was built under R version 3.6.2

## Loading required package: Formula

## Warning: package 'Formula' was built under R version 3.6.2

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.6.2

##
## Attaching package: 'ggplot2'
```

```
## The following objects are masked from 'package:psych':
##
##     %+%, alpha

##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:dplyr':
##
##     src, summarize

## The following object is masked from 'package:psych':
##
##     describe

## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```r
library(VIM)
```

```
## Loading required package: colorspace

## Warning: package 'colorspace' was built under R version 3.6.2

## Loading required package: grid

## Loading required package: data.table

## Warning: package 'data.table' was built under R version 3.6.2

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

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

## VIM is ready to use.
##  Since version 4.0.0 the GUI is in its own package VIMGUI.
##
##          Please use the package to use the new (and old) GUI.

## Suggestions and bug-reports can be submitted at: https://github.com/alexkowa/VIM/issues

##
## Attaching package: 'VIM'

## The following object is masked from 'package:datasets':
##
##     sleep
```

```r
#Read data
REM <- read.csv("RealEstateMaster.csv")
REMM <- read.csv("RealEstateMissing.csv")

#Determine the number of NAs (missing values) in the HP column
sum(is.na(REMM$Price))
```

```
## [1] 45
```

```r
#Imputation with median
REMM_impute_median <- impute(REMM$Price, median)

#Vector of missing values
x <- as.data.frame(REMM$Price)

#Vector of Price values imputed with median
y <- as.data.frame(REMM_impute_median)

#Juxtapose all three columns in one view for convenient viewing
FINAL_IMPUTED_SUMMARY <- cbind(REM$Price, x, y)
#Writing data to CSV. Not needed but to verify.
#write.table(FINAL_IMPUTED_SUMMARY, file="FINAL_IMPUTED_SUMMARY.csv",sep=",",row.names=F)
head(FINAL_IMPUTED_SUMMARY)
```

```
##   REM$Price REMM$Price REMM_impute_median
## 1     263.1         NA             223.05
## 2     182.4         NA             223.05
## 3     242.1      242.1             242.10
## 4     213.6         NA             223.05
## 5     139.9         NA             223.05
## 6     245.4      245.4             245.40
```

b. Use kNN imputation for k = 5. See how well kNN with k = 5 imputes the missing values.

```r
REMM_impute_knn <- kNN(REMM, variable = c("Price"), k = 5)
```

c. Use kNN imputation for k = 3. See how well kNN with k = 3 imputes the missing values.

```r
REMM_impute_knn2 <- kNN(REMM, variable = c("Price"), k = 3)
```

d. Create a summary csv file

```r
comparison <- cbind(REM$Price, x, y, REMM_impute_knn2$Price, REMM_impute_knn$Price)
names(comparison) <- c("Original", "Missing", "Median_Imputation", "KNN_Imputation_3","KNN_Imputation_5
#Writing data to CSV. Not needed but to verify.
#write.table(comparison, file="imputation_comparison.csv",sep=",",row.names=F)
head(comparison)
```

```
##   Original Missing Median_Imputation KNN_Imputation_3 KNN_Imputation_5
## 1    263.1      NA            223.05            198.9            188.3
```

```
## 2    182.4      NA             223.05            217.8             194.4
## 3    242.1   242.1             242.10            242.1             242.1
## 4    213.6      NA             223.05            194.4             194.4
## 5    139.9      NA             223.05            194.4             194.4
## 6    245.4   245.4             245.40            245.4             245.4
```

e. The objective of imputing missing values is so that we can get our data ready for data analysis. In pursuit of this objective, find the numerical summaries (mean, median, standard deviation, min, max, and the quartiles) of the original Price values. Then find the same numerical summaries for the median imputed and kNN imputed values. In all you should have four (4) sets of summaries.

```r
# Find the numerical summaries of the original Price values
Summary_Original <- as.table(summary(comparison$Original))
var_Original <- sum((Summary_Original -
                       mean(Summary_Original))^2)/ length(Summary_Original)
Summary_Original["SD"] <- sqrt(var_Original)

# Find the numerical summaries of the median imputed values
Summary_Median <- as.table(summary(comparison$Median_Imputation))
```

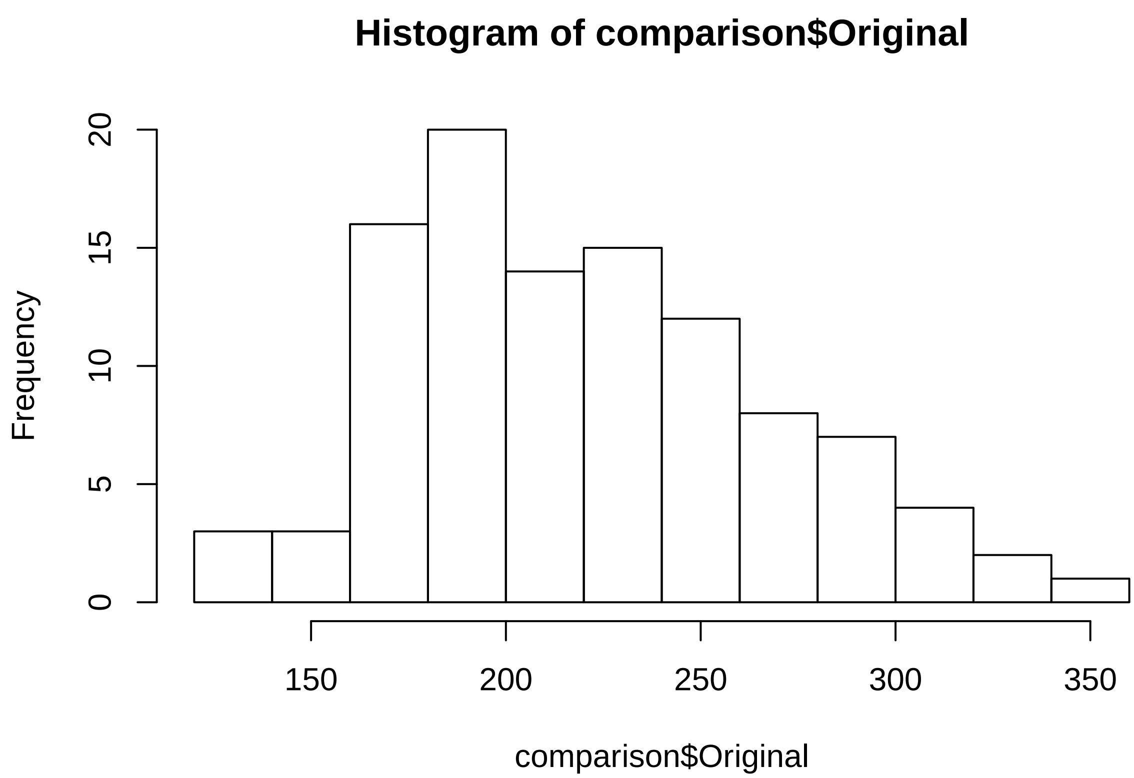
```
##
##  45 values imputed to 223.05
```

```r
var_Median <- sum((Summary_Median -
                       mean(Summary_Median))^2)/ length(Summary_Median)
Summary_Median["SD"] <- sqrt(var_Median)

# Find the numerical summaries of the knn imputed values with k = 3
Summary_KNN3 <- as.table(summary(comparison$KNN_Imputation_3))
var_KNN3 <- sum((Summary_KNN3 -
                       mean(Summary_KNN3))^2)/ length(Summary_KNN3)
Summary_KNN3["SD"] <- sqrt(var_KNN3)

# Find the numerical summaries of the knn imputed values with k = 5
Summary_KNN5 <- as.table(summary(comparison$KNN_Imputation_5))
var_KNN5 <- sum((Summary_KNN5 -
                       mean(Summary_KNN5))^2)/ length(Summary_KNN5)
Summary_KNN5["SD"] <- sqrt(var_KNN5)

summaries <- rbind(Summary_Original, Summary_Median, Summary_KNN3, Summary_KNN5)
summaries
```

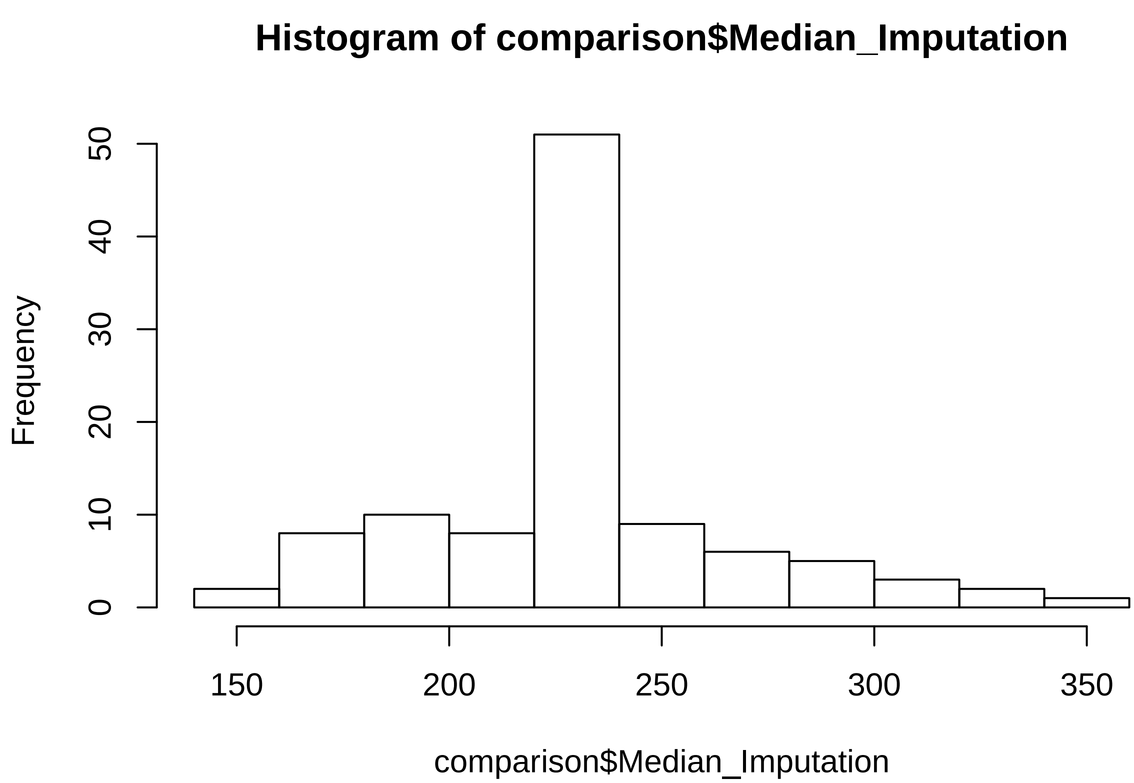
```
##                   Min. 1st Qu. Median     Mean 3rd Qu.  Max.       SD
## Summary_Original 125.0   187.0 213.60 221.1029   251.4 345.3 66.77193
## Summary_Median   147.4   216.8 223.05 228.0167   240.0 345.3 58.27067
## Summary_KNN3     147.4   194.4 217.80 226.4248   254.3 345.3 60.73309
## Summary_KNN5     147.4   192.9 209.00 225.2210   252.3 345.3 61.16105
```

f. Continuing with the analysis, plot separate histograms (four total) for the original Price variable, for the median imputed Prices, the kNN(3) imputed Prices, and the kNN(5) imputed prices. You can use the hist() command in R with the defaults. See which imputation method preserves the shape of the distribution when compared with the original Price data.

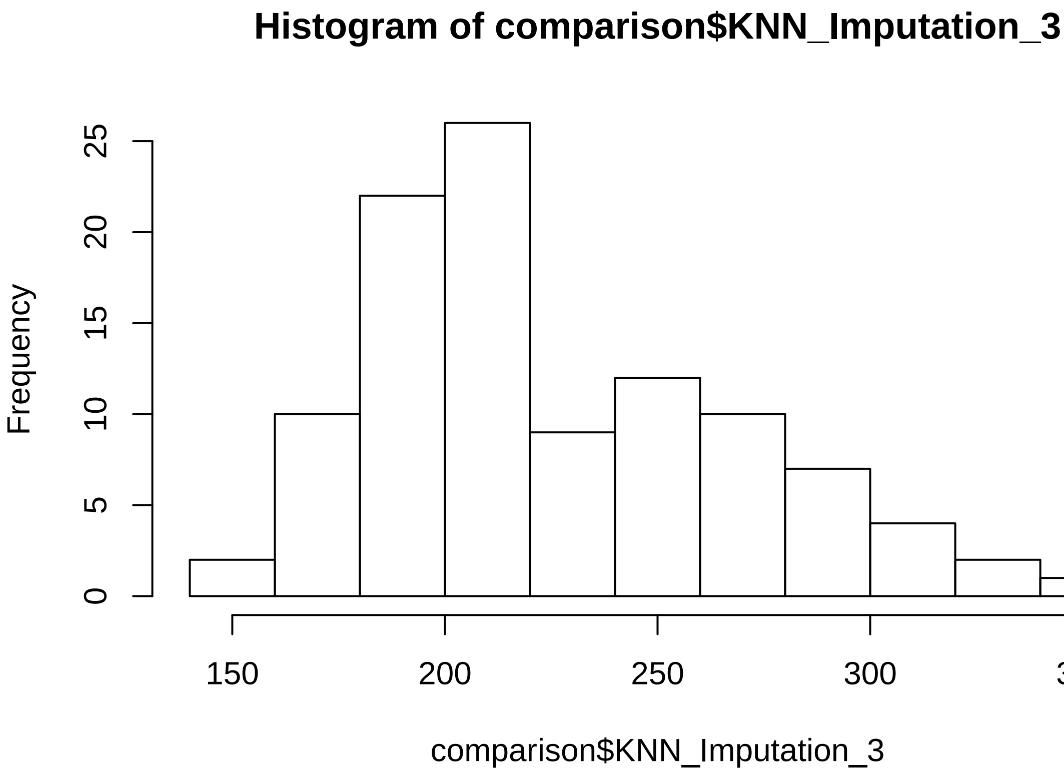
```r
hist(comparison$Original)
```



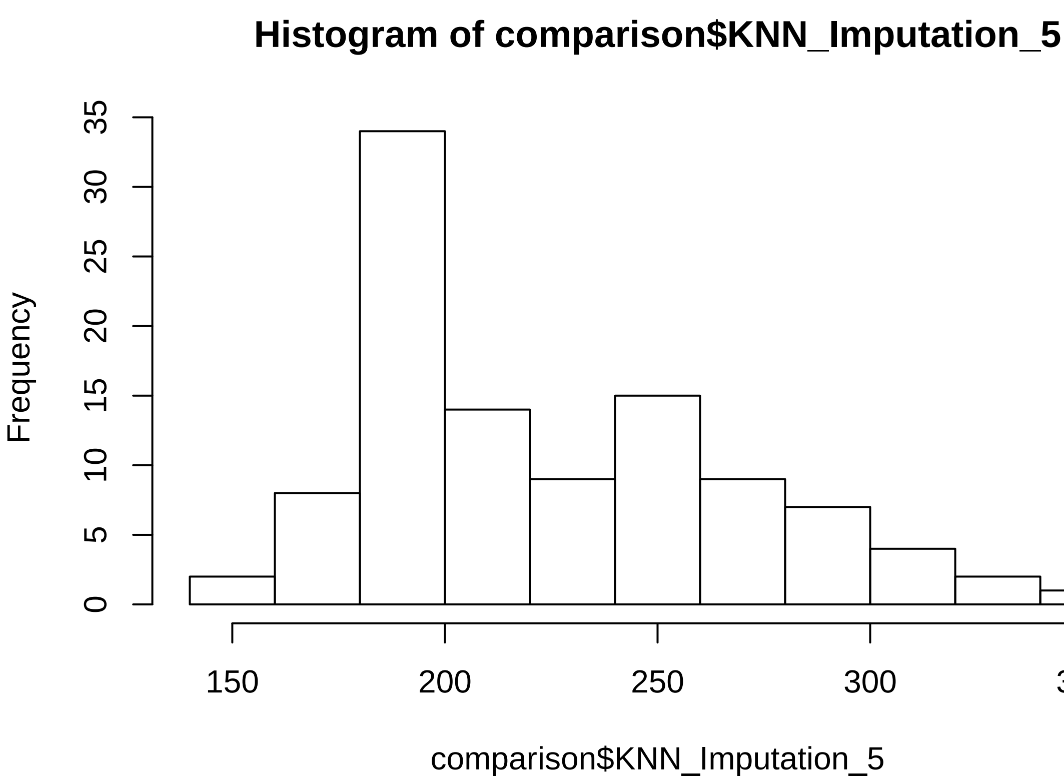

```r
hist(comparison$Median_Imputation)
```



```r
hist(comparison$KNN_Imputation_3)
```




```r
hist(comparison$KNN_Imputation_5)
```



Knn imputation with k = 3 preserves the shape of the distribution of the original the best.

Part 4 A leading pizza vendor has a contract to supply pizza at all home baseball games in Sacramento. Before each game begins, a constant challenge is to determine how many pizzas to make available at the games. Ken Binlard, a business analyst, has determined that his fixed cost of providing pizzas is $1,000. Ken believes that this cost should be equally allocated between two types of pizzas.

Ken will supply only two types of pizzas: plain cheese and veggie-and-cheese combo. It costs Ken $4.50 to produce a plain cheese pizza and $5.00 to produce a veggie-and-cheese pizza. The selling price for both pizzas at the game is $9.00. Left over pizzas will have no value and will be donated to the homeless.

The demand for cheese pizza in quantities of (200, 300, 400, 500, 600, 700, 800, 900) have the probabilities of (0.1, 0.15, 0.15, 0.2, 0.2, 0.1, 0.05, 0.05) respectively

The demand for veggie-cheese pizza in quantities of (300, 400, 500, 600, 700, 800) have the probabilities of (0.1, 0.2, 0.25, 0.25, 0.15, 0.05) respectively

a. For both plain cheese and veggie-and-cheese combo, determine the profit (or loss) associated with producing at different possible demand levels. For instance, determine the profit if 200 plain cheese pizzas are produced and 200 are demanded. What is the profit if 200 plain cheese pizzas are produced but 300 were demanded, and so on? Summarize your results in a two-way data table using R or Python and NOT Excel. A two-way data table is one in which rows correspond to one variable (say, demand) and columns correspond to another variable (say, production). The body of the table contains the data. You will create two such tables – one for plain cheese and another for veggie-and-cheese pizza.

```
library(knitr)
```

```
## Warning: package 'knitr' was built under R version 3.6.2
```

```
Probability_Plain_Cheese  <- c(0.1, 0.15, 0.15, 0.2, 0.2, 0.1, 0.05, 0.05)
Demand_Plain_Cheese <- seq(200,900, by= 100)
Produced_Plain_Cheese <- seq(200,900, by= 100)

Profit_function_Plain <- function(x,y) {
  ifelse(x>y, 9*y -4.5*x - 500, 4.5*x-500)
}

Profit_Plain <- outer(Produced_Plain_Cheese, Demand_Plain_Cheese, Profit_function_Plain)
colnames(Profit_Plain) <- c("Demand 200", "Demand 300", "Demand 400", "Demand 500", "Demand 600", "Demar
rownames(Profit_Plain) <- c("Produce 200", "Produce 300", "Produce 400", "Produce 500", "Produce 600",

kable(Profit_Plain, caption = "Profit for Plain Pizza")
```

Table 1: Profit for Plain Pizza

	Demand 200	Demand 300	Demand 400	Demand 500	Demand 600	Demand 700	Demand 800	Demand 900
Produce 200	400	400	400	400	400	400	400	400
Produce 300	-50	850	850	850	850	850	850	850
Produce 400	-500	400	1300	1300	1300	1300	1300	1300
Produce 500	-950	-50	850	1750	1750	1750	1750	1750

	Demand 200	Demand 300	Demand 400	Demand 500	Demand 600	Demand 700	Demand 800	Demand 900
Produce 600	-1400	-500	400	1300	2200	2200	2200	2200
Produce 700	-1850	-950	-50	850	1750	2650	2650	2650
Produce 800	-2300	-1400	-500	400	1300	2200	3100	3100
Produce 900	-2750	-1850	-950	-50	850	1750	2650	3550

```
Probability_Veggie_Cheese  <- c(0.1, 0.2, 0.25, 0.25, 0.15, 0.05)
Demand_Veggie_Cheese <- seq(300,800, by= 100)
Produced_Veggie_Cheese <- seq(300,800, by= 100)

Profit_function_Veggie <- function(a, b) {
  ifelse(a>b, 9*b - 5*a - 500, (9-5)*a-500)
}

Profit_Veggie <- outer(Produced_Veggie_Cheese, Demand_Veggie_Cheese, Profit_function_Veggie)
colnames(Profit_Veggie) <- seq(300,800, by= 100)
rownames(Profit_Veggie) <- seq(300,800, by= 100)
colnames(Profit_Veggie) <- c("Demand 300", "Demand 400", "Demand 500", "Demand 600", "Demand 700", "Dema
rownames(Profit_Veggie) <- c("Produce 300", "Produce 400", "Produce 500", "Produce 600", "Produce 700",

kable(Profit_Veggie, caption = "Profit for Veggie Pizza")
```

Table 2: Profit for Veggie Pizza

	Demand 300	Demand 400	Demand 500	Demand 600	Demand 700	Demand 800
Produce 300	700	700	700	700	700	700
Produce 400	200	1100	1100	1100	1100	1100
Produce 500	-300	600	1500	1500	1500	1500
Produce 600	-800	100	1000	1900	1900	1900
Produce 700	-1300	-400	500	1400	2300	2300
Produce 800	-1800	-900	0	900	1800	2700

b. Compute the expected profit associated with each possible production level (assuming Ken will only produce at one of the possible demand levels) for each type of pizza. Hint: This would be a vector of expected values. You will need two such vectors of expected values– one for plain cheese and another for veggie-and-cheese pizza.

```
expected_profit_plain <- rowSums(t(t(Profit_Plain) * Probability_Plain_Cheese))
expected_profit_veggie <- rowSums(t(t(Profit_Veggie) * Probability_Veggie_Cheese))
print("The expected profit for plain pizza at the weighted average demand is as follows")
```

```
## [1] "The expected profit for plain pizza at the weighted average demand is as follows"
```

```r
expected_profit_plain
```

```
## Produce 200 Produce 300 Produce 400 Produce 500 Produce 600 Produce 700
##         400         760         985        1075         985         715
## Produce 800 Produce 900
##         355         -50
```

```r
print("The expected profit for veggie pizza at the weighted average demand is as follows")
```

```
## [1] "The expected profit for veggie pizza at the weighted average demand is as follows"
```

```r
expected_profit_veggie
```

```
## Produce 300 Produce 400 Produce 500 Produce 600 Produce 700 Produce 800
##         700        1010        1140        1045         725         270
```

c. If Ken wants to maximize the expected profit from pizza sales at the game, then how many of each type of pizza should he produce?

According to the result above, Ken should produce 500 plain and 500 veggie pizzas.

Part 5 Suppose you have the opportunity to play a game with a "wheel of fortune". When you spin a large wheel, it is equally likely to stop in any position. Depending on where it stops, you win anywhere from \$0 to \$1000 (in \$1 increments, assume). Let us suppose your winnings are actually based on not one spin, but on the average of n spins of the wheel. For example, if n = 2, your winnings are based on the average of two spins. If the first spin results in \$580 and the second spin results in \$320, you win the average, \$450. The question is how does the distribution of your winnings depend on n. To answer this question, perform a simulation in R using the following guidelines.

a. Find the theoretical mean and theoretical standard deviation of the winnings.

```r
winnings <- seq(0, 1000, by = 1)
theor_mean <- mean(winnings)
theor_var <- sum((winnings - theor_mean)^2)/1001
theor_sd <- sqrt(theor_var)

cat("The theoretical mean is", theor_mean,
    "and the theoretical standard deviation is", theor_sd)
```

```
## The theoretical mean is 500 and the theoretical standard deviation is 288.9637
```

b. Consider number of spins as your sample size (n). Perform simulating 1 spin, 2 spins, 3 spins, 4 spins, 5 spins, 6 spins, 7 spins, 8 spins, 9 spins, and 10 spins. For each number of spins, perform 1,000 replications. For example, 1,000 replications containing 1 spin each, 1,000 replications: each replication containing 2 spins 1,000 replications: each replication containing 3 spins 1,000 replications: each replication containing 4 spins, etc.

```r
set.seed(50)

# Sample
rep_spins <- function(sample_size){
  replicate(1000, {
    sample(winnings, sample_size, replace = TRUE)
  })
}

# Store the results in a matrix
matrix_spins_1 <- matrix(rep_spins(1), 1000)
matrix_spins_2 <- matrix(rep_spins(2), 1000)
matrix_spins_3 <- matrix(rep_spins(3), 1000)
matrix_spins_4 <- matrix(rep_spins(4), 1000)
matrix_spins_5 <- matrix(rep_spins(5), 1000)
matrix_spins_6 <- matrix(rep_spins(6), 1000)
matrix_spins_7 <- matrix(rep_spins(7), 1000)
matrix_spins_8 <- matrix(rep_spins(8), 1000)
matrix_spins_9 <- matrix(rep_spins(9), 1000)
matrix_spins_10 <- matrix(rep_spins(10), 1000)
```

c. Find the sample mean for each replication for each spin category

```r
means_1_spin <- rowMeans(matrix_spins_1, na.rm = FALSE)
means_2_spin <- rowMeans(matrix_spins_2, na.rm = FALSE)
means_3_spin <- rowMeans(matrix_spins_3, na.rm = FALSE)
means_4_spin <- rowMeans(matrix_spins_4, na.rm = FALSE)
means_5_spin <- rowMeans(matrix_spins_5, na.rm = FALSE)
means_6_spin <- rowMeans(matrix_spins_6, na.rm = FALSE)
means_7_spin <- rowMeans(matrix_spins_7, na.rm = FALSE)
means_8_spin <- rowMeans(matrix_spins_8, na.rm = FALSE)
means_9_spin <- rowMeans(matrix_spins_9, na.rm = FALSE)
means_10_spin <- rowMeans(matrix_spins_10, na.rm = FALSE)
```

d. Find the mean of the sample means of each replication.

```r
mean_of_sample_mean1 <- mean(means_1_spin)
mean_of_sample_mean2 <- mean(means_2_spin)
mean_of_sample_mean3 <- mean(means_3_spin)
mean_of_sample_mean4 <- mean(means_4_spin)
mean_of_sample_mean5 <- mean(means_5_spin)
mean_of_sample_mean6 <- mean(means_6_spin)
mean_of_sample_mean7 <- mean(means_7_spin)
mean_of_sample_mean8 <- mean(means_8_spin)
mean_of_sample_mean9 <- mean(means_9_spin)
mean_of_sample_mean10 <- mean(means_10_spin)

means_of_sample_means <- cbind(mean(means_1_spin),
      mean(means_2_spin),
      mean(means_3_spin),
      mean(means_4_spin),
      mean(means_5_spin),
```

```r
      mean(means_6_spin),
      mean(means_7_spin),
      mean(means_8_spin),
      mean(means_9_spin),
      mean(means_10_spin)
      )
means_of_sample_means
```

```
##         [,1]    [,2]  [,3]    [,4]     [,5]     [,6]    [,7]    [,8]     [,9]
## [1,] 503.067 503.512 491.9 497.555 494.1392 499.6518 500.229 498.584 499.6303
##         [,10]
## [1,] 503.7255
```

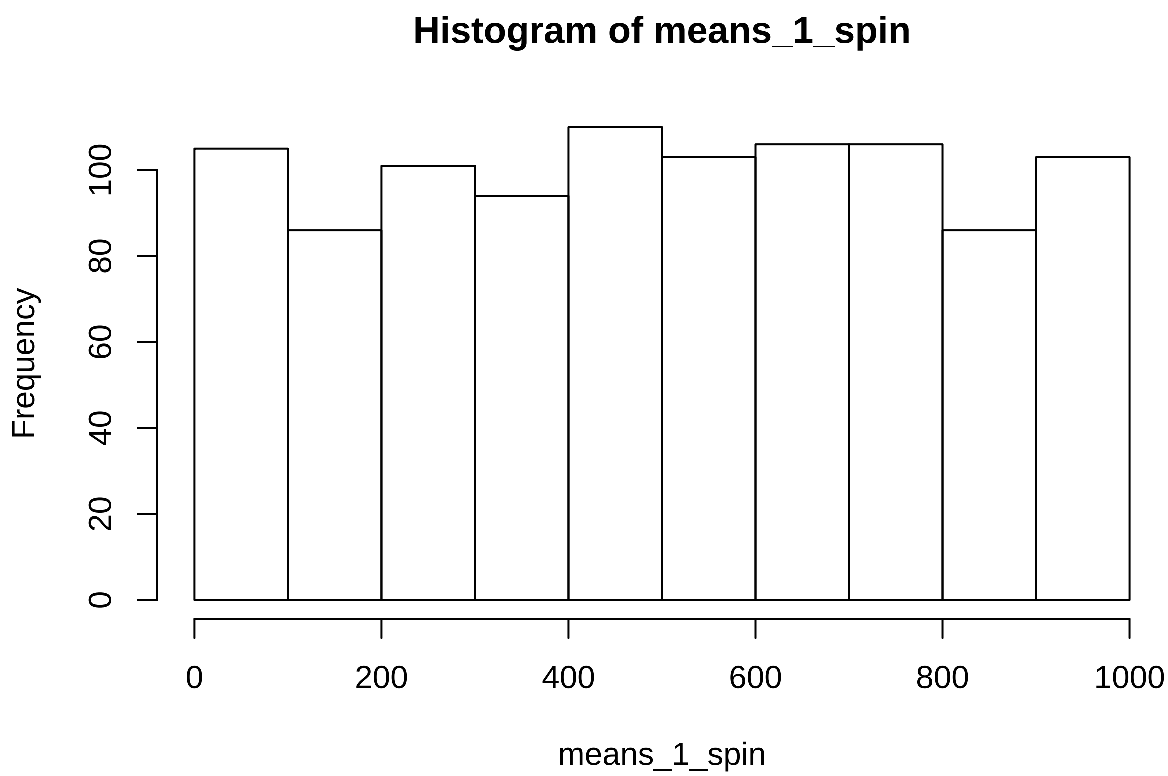
e. Find the standard deviation of sample means of each replication.

```r
sd_of_sample_means <- cbind(
  sd(means_1_spin),
  sd(means_2_spin),
  sd(means_3_spin),
  sd(means_4_spin),
  sd(means_5_spin),
  sd(means_6_spin),
  sd(means_7_spin),
  sd(means_8_spin),
  sd(means_9_spin),
  sd(means_10_spin)
)

sd_of_sample_means
```

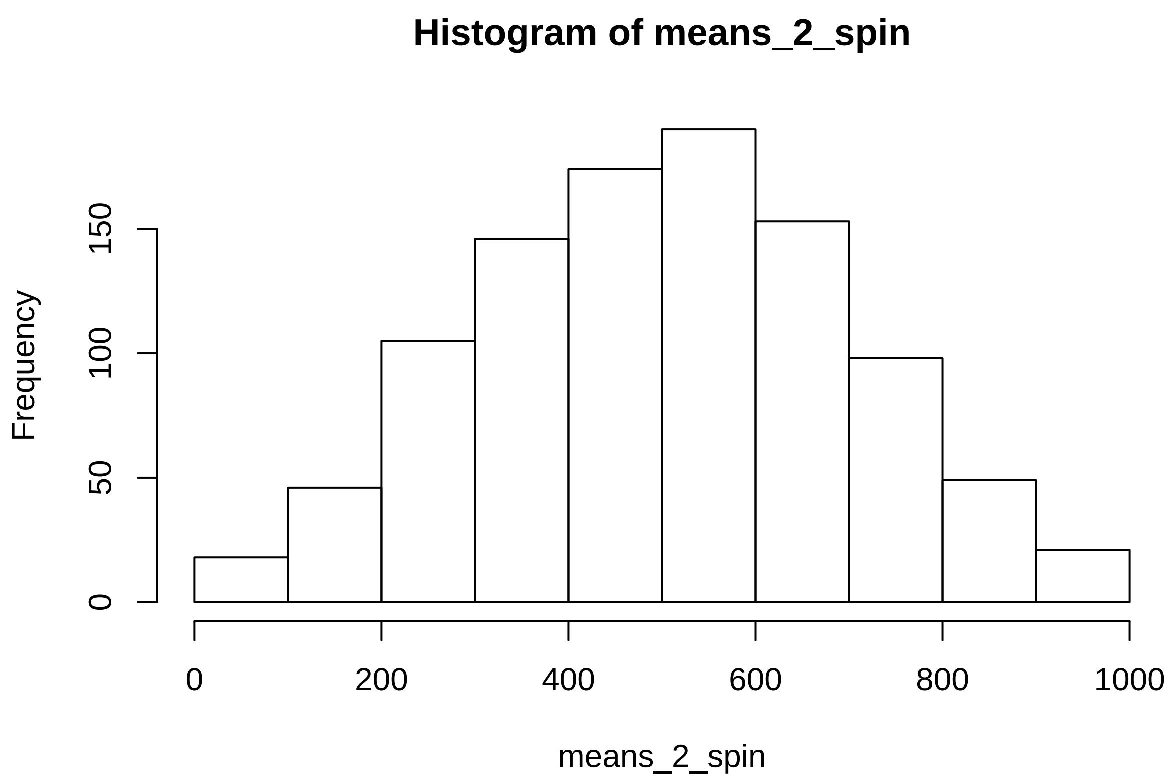
```
##          [,1]     [,2]     [,3]     [,4]   [,5]     [,6]     [,7]     [,8]
## [1,] 287.1736 197.7254 173.6212 143.6107 127.77 118.4016 107.9148 106.3041
##          [,9]    [,10]
## [1,] 99.89172 88.71582
```

f. Plot a histogram for each of the 10 categories of spins. So there will be 10 histograms – one 1 spin, one for 2 spins, one for 3 spins, etc. Comment on how the shape of the histogram changes with increasing the number of spins.

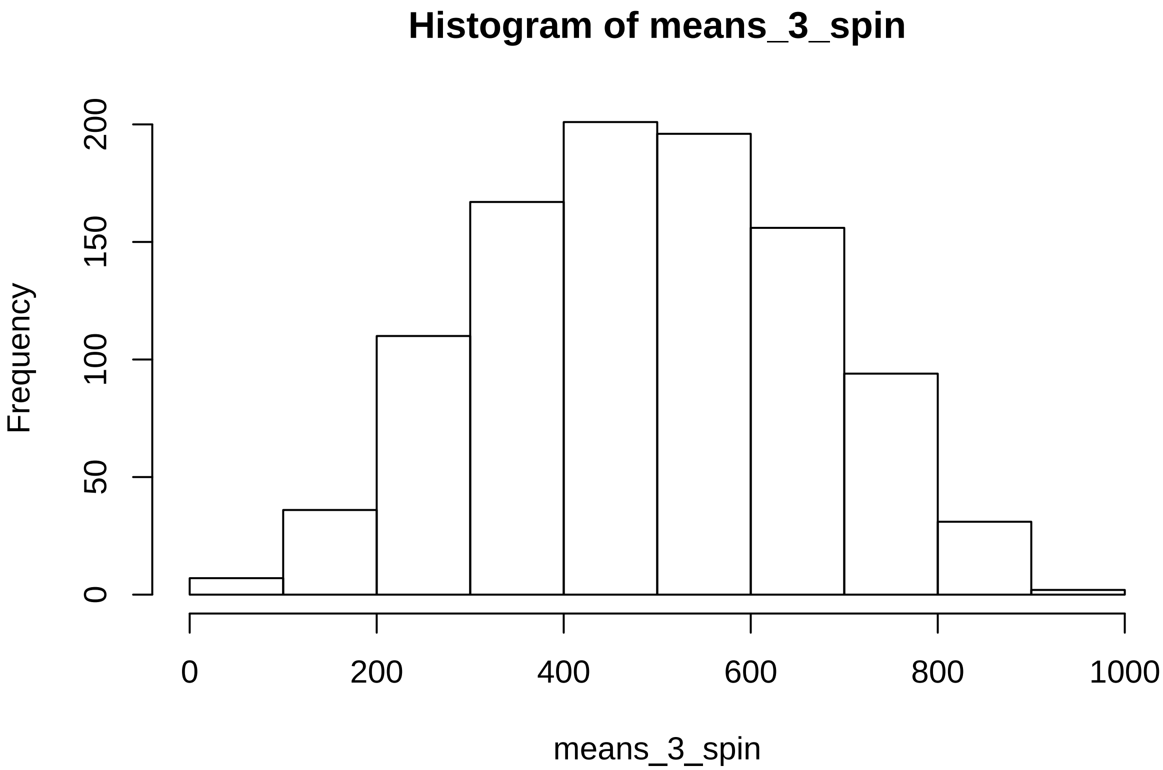
```r
hist(means_1_spin)
```



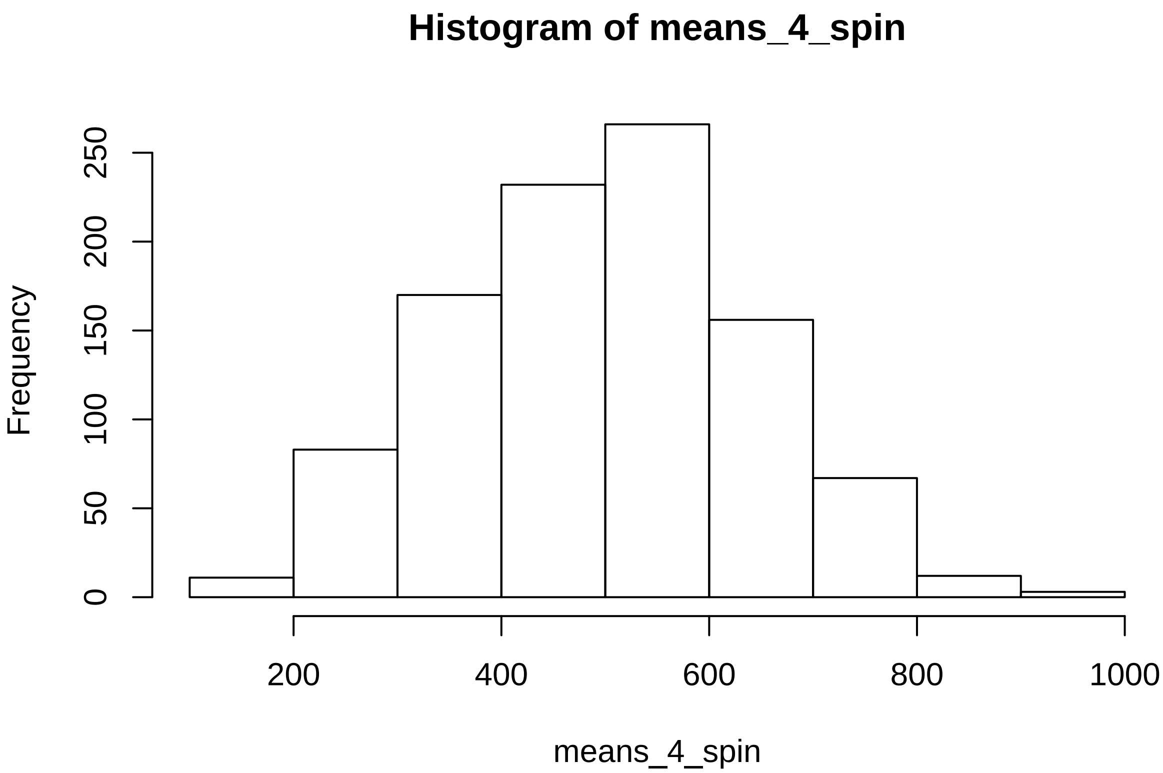

```
hist(means_2_spin)
```



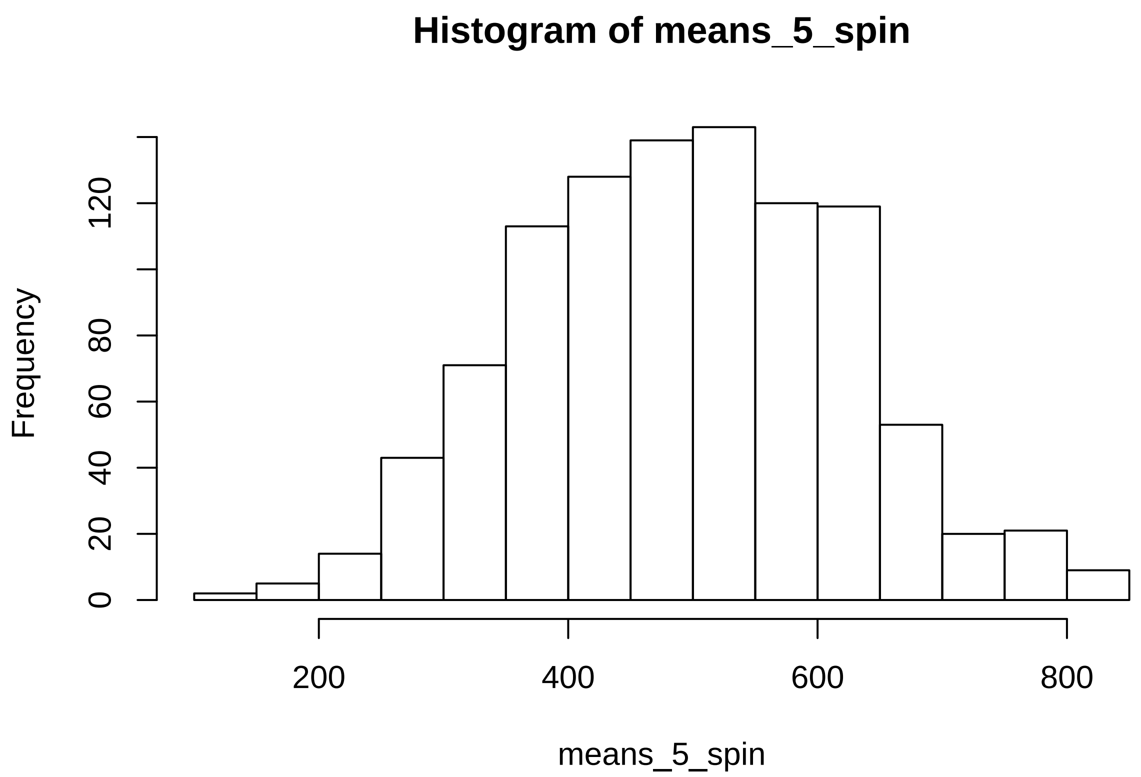
```r
hist(means_3_spin)
```



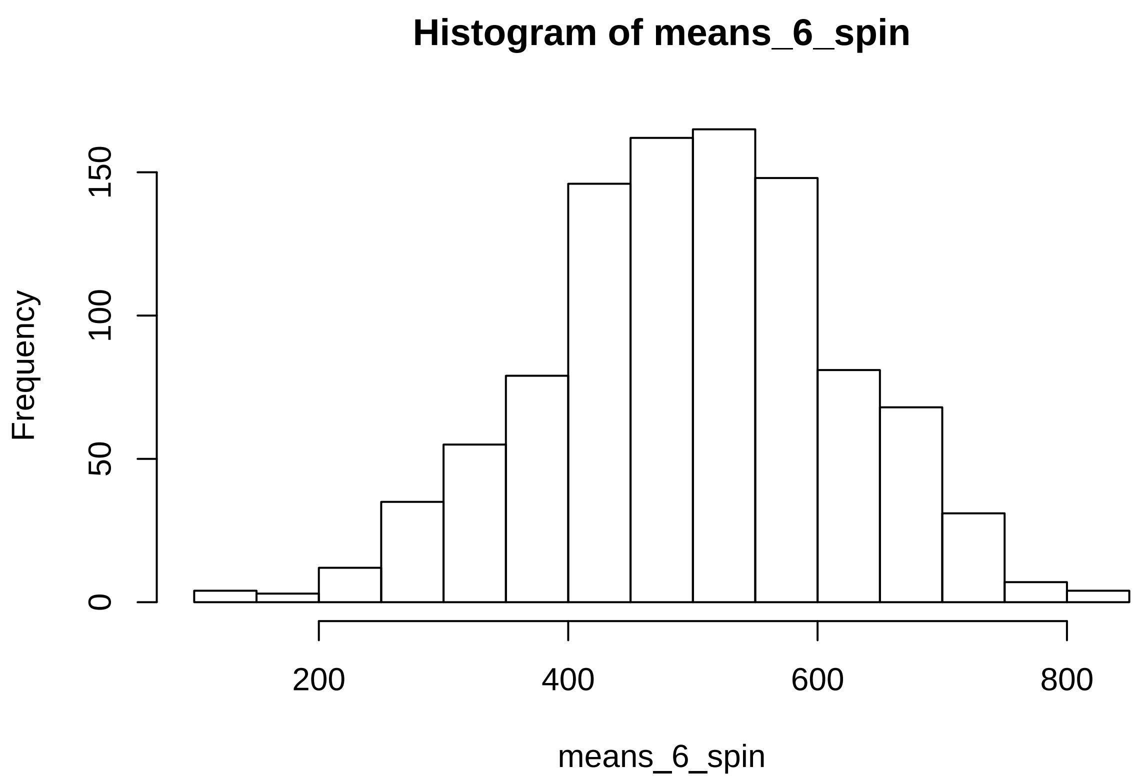

```r
hist(means_4_spin)
```



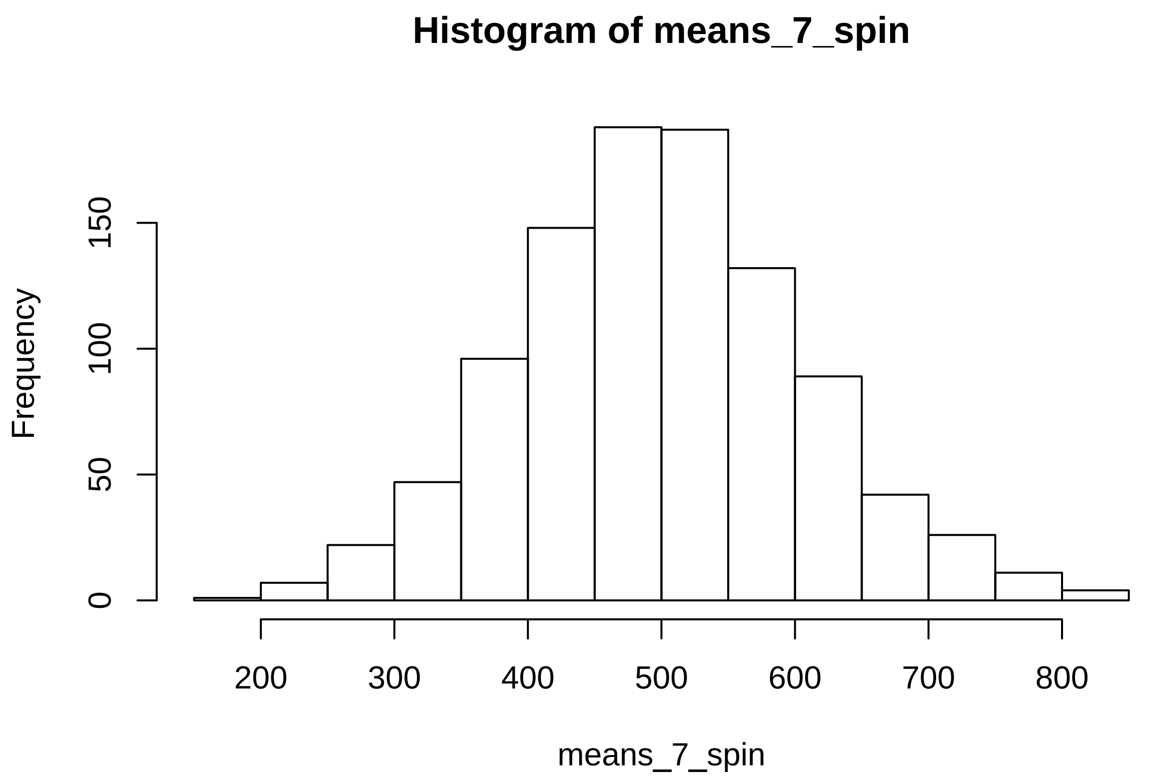
```r
hist(means_5_spin)
```



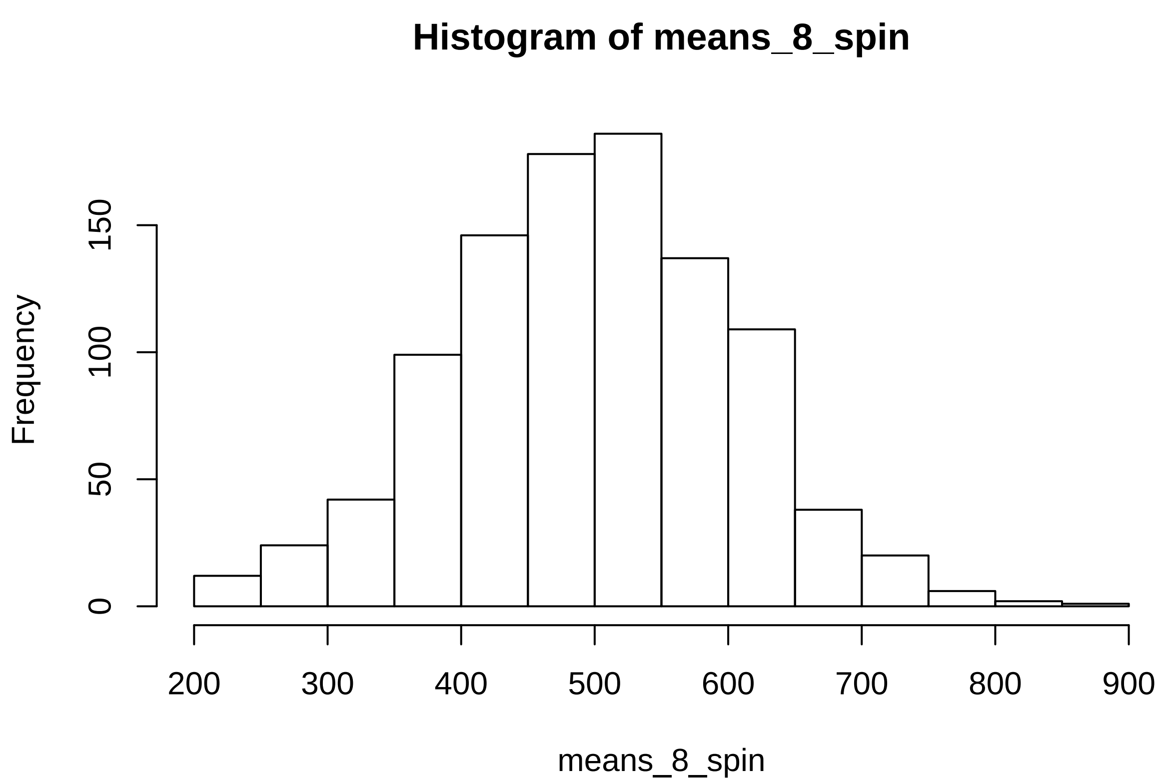

```r
hist(means_6_spin)
```



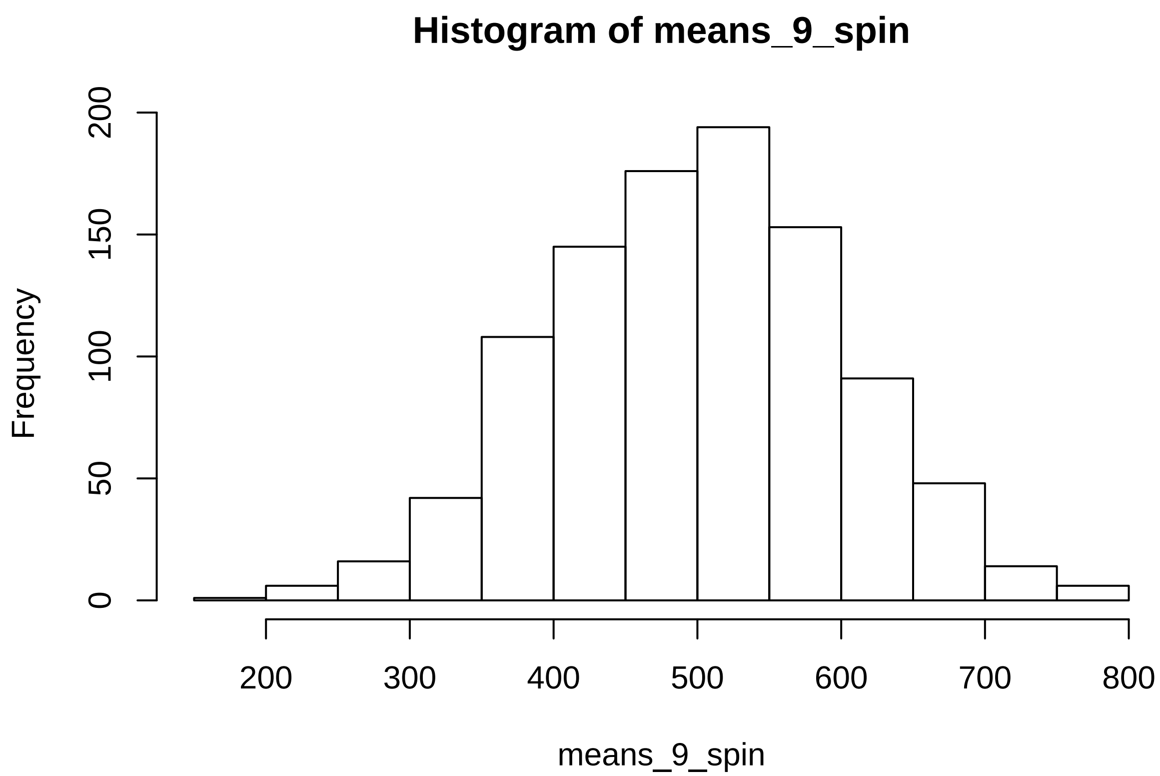
```r
hist(means_7_spin)
```



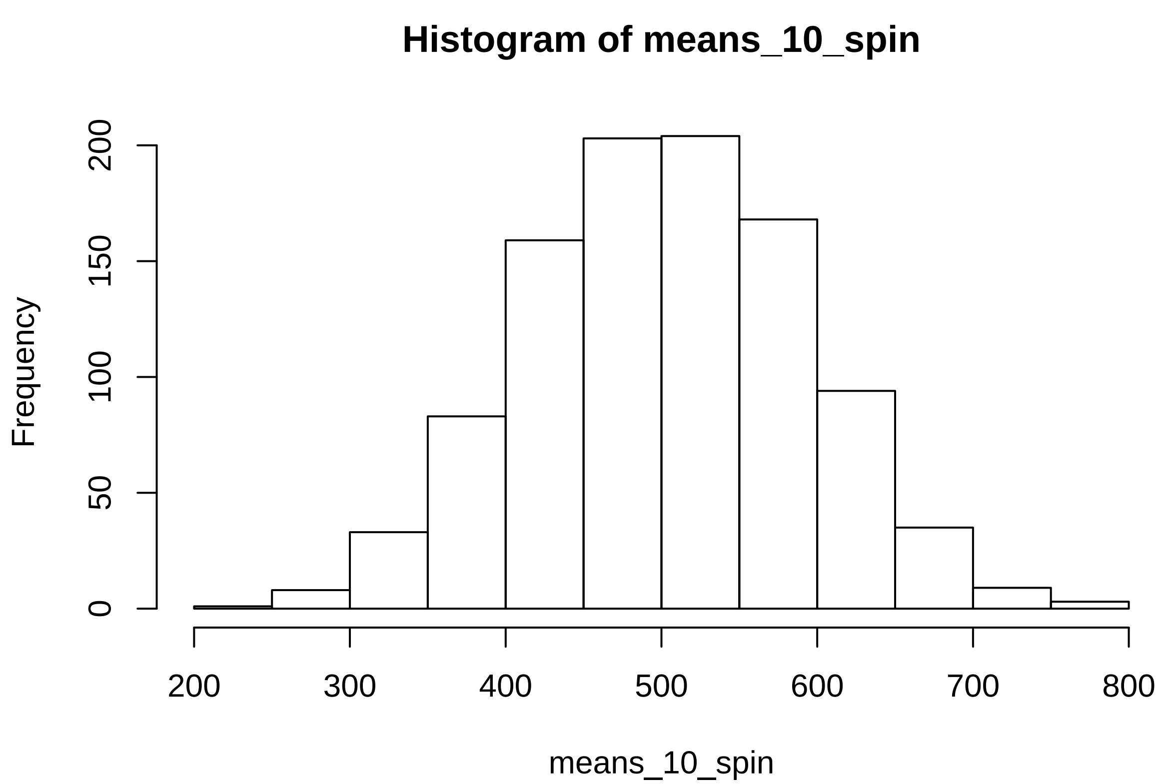

```r
hist(means_8_spin)
```



```r
hist(means_9_spin)
```




```r
hist(means_10_spin)
```



As the number of spins increases, the histogram becomes closer and closer to a normal distribution.

g. Find the theoretical standard error for each set of spins.

```r
theor_se<- cbind(
  theor_sd/sqrt(1),
  theor_sd/sqrt(2),
  theor_sd/sqrt(3),
  theor_sd/sqrt(4),
  theor_sd/sqrt(5),
  theor_sd/sqrt(6),
  theor_sd/sqrt(7),
  theor_sd/sqrt(8),
  theor_sd/sqrt(9),
  theor_sd/sqrt(10))

theor_se
```

```
##          [,1]     [,2]     [,3]     [,4]     [,5]     [,6]    [,7]     [,8]
## [1,] 288.9637 204.3282 166.8333 144.4818 129.2285 117.9689 109.218 102.1641
##          [,9]    [,10]
## [1,] 96.32122 91.37833
```

h. Compare the theoretical mean with the mean of sample means found in part d above.

```r
theoretical_mean <- c(rep(theor_mean, 10))

comparison_of_theormean_and_samplemean <-
  rbind(theoretical_mean, means_of_sample_means)

rownames(comparison_of_theormean_and_samplemean) <- c("Theoretical mean", "Sample mean")
colnames(comparison_of_theormean_and_samplemean) <- c("1 spin", "2 spins", "3 spins", "4 spins", "5 spin
                                    "6 spins", "7 spins", "8 spins", "9 spins", "10 spins")

comparison_of_theormean_and_samplemean
```

```
##                   1 spin 2 spins 3 spins 4 spins  5 spins  6 spins 7 spins
## Theoretical mean 500.000 500.000   500.0 500.000 500.0000 500.0000 500.000
## Sample mean      503.067 503.512   491.9 497.555 494.1392 499.6518 500.229
##                  8 spins  9 spins 10 spins
## Theoretical mean 500.000 500.0000 500.0000
## Sample mean      498.584 499.6303 503.7255
```

Intuitively, the theoretial mean is very close to all the means of sample means.

i. Compare the theoretical standard error (part g above) with the standard deviation of sample means (part e above).

```r
comparison_of_sd_and_therse <- rbind(sd_of_sample_means, theor_se)
rownames(comparison_of_sd_and_therse) <- c("sd of sample means","theor se")
colnames(comparison_of_sd_and_therse) <- c("1 spin", "2 spins", "3 spins", "4 spins", "5 spins",
                                    "6 spins", "7 spins", "8 spins", "9 spins", "10 spins")
comparison_of_sd_and_therse
```

```
##                       1 spin  2 spins  3 spins  4 spins  5 spins  6 spins
## sd of sample means 287.1736 197.7254 173.6212 143.6107 127.7700 118.4016
## theor se           288.9637 204.3282 166.8333 144.4818 129.2285 117.9689
##                     7 spins  8 spins  9 spins 10 spins
## sd of sample means 107.9148 106.3041 99.89172 88.71582
## theor se           109.2180 102.1641 96.32122 91.37833
```

Intuitively, the sd of sample means of each set of spins is very close to the theoretical standard error of it.

j. Find the probability of winning more than $600 for each spin category.

```r
library(dplyr)
mean_1_spin <- matrix(means_1_spin, 1000)
colnames(mean_1_spin) <- c("x")
mean_1_over <- as.data.frame(mean_1_spin) %>%
  filter(x>600) %>%
  summarise(n = n())
prob_of_1_win_over_600 <- mean_1_over/1000

mean_2_spin <- matrix(means_2_spin, 1000)
colnames(mean_2_spin) <- c("x")
mean_2_over <- as.data.frame(mean_2_spin) %>%
  filter(x>600) %>%
  summarise(n = n())
prob_of_2_win_over_600 <- mean_2_over/1000

mean_3_spin <- matrix(means_3_spin, 1000)
colnames(mean_3_spin) <- c("x")
mean_3_over <- as.data.frame(mean_3_spin) %>%
  filter(x>600) %>%
  summarise(n = n())
prob_of_3_win_over_600 <- mean_3_over/1000

mean_4_spin <- matrix(means_4_spin, 1000)
colnames(mean_4_spin) <- c("x")
mean_4_over <- as.data.frame(mean_4_spin) %>%
  filter(x>600) %>%
  summarise(n = n())
prob_of_4_win_over_600 <- mean_4_over/1000

mean_5_spin <- matrix(means_5_spin, 1000)
colnames(mean_5_spin) <- c("x")
mean_5_over <- as.data.frame(mean_5_spin) %>%
  filter(x>600) %>%
  summarise(n = n())
prob_of_5_win_over_600 <- mean_5_over/1000

mean_6_spin <- matrix(means_6_spin, 1000)
colnames(mean_6_spin) <- c("x")
mean_6_over <- as.data.frame(mean_6_spin) %>%
  filter(x>600) %>%
  summarise(n = n())
prob_of_6_win_over_600 <- mean_6_over/1000
```

```r
mean_7_spin <- matrix(means_7_spin, 1000)
colnames(mean_7_spin) <- c("x")
mean_7_over <- as.data.frame(mean_7_spin) %>%
  filter(x>600) %>%
  summarise(n = n())
prob_of_7_win_over_600 <- mean_7_over/1000

mean_8_spin <- matrix(means_8_spin, 1000)
colnames(mean_8_spin) <- c("x")
mean_8_over <- as.data.frame(mean_8_spin) %>%
  filter(x>600) %>%
  summarise(n = n())
prob_of_8_win_over_600 <- mean_8_over/1000

mean_9_spin <- matrix(means_9_spin, 1000)
colnames(mean_9_spin) <- c("x")
mean_9_over <- as.data.frame(mean_9_spin) %>%
  filter(x>600) %>%
  summarise(n = n())
prob_of_9_win_over_600 <- mean_9_over/1000

mean_10_spin <- matrix(means_10_spin, 1000)
colnames(mean_10_spin) <- c("x")
mean_10_over <- as.data.frame(mean_10_spin) %>%
  filter(x>600) %>%
  summarise(n = n())
prob_of_10_win_over_600 <- mean_10_over/1000

prob_of_win_over_600 <- cbind(mean_1_over/1000, mean_2_over/1000,
                              mean_3_over/1000, mean_4_over/1000,
                              mean_5_over/1000, mean_6_over/1000,
                              mean_7_over/1000, mean_8_over/1000,
                              mean_9_over/1000, mean_10_over/1000)

colnames(prob_of_win_over_600) <- c("1 spin", "2 spins", "3 spins", "4 spins", "5 spins",
                                    "6 spins", "7 spins", "8 spins", "9 spins", "10 spins")
prob_of_win_over_600
```

```
##   1 spin 2 spins 3 spins 4 spins 5 spins 6 spins 7 spins 8 spins 9 spins
## 1  0.401   0.321   0.283   0.238   0.222   0.191   0.172   0.176   0.159
##   10 spins
## 1    0.141
```

k. Summarize the results in a table

```r
summary_tbl <- rbind(theoretical_mean, as.vector(means_of_sample_means), as.vector(theor_se), as.vector

rownames(summary_tbl) <- c("Theoretical Mean", "Mean of Sample Means",
                       "Theoretical Standard Error",
                       "Standard Deviation of Sample Means",
                       "P(winning > 600)")
colnames(summary_tbl) <- c("1 spin", "2 spins", "3 spins", "4 spins", "5 spins",
```

```r
                                    "6 spins", "7 spins", "8 spins", "9 spins", "10 spins")
summary_tbl
```

```
##                                     1 spin  2 spins  3 spins  4 spins  5 spins
## Theoretical Mean                  500.0000 500.0000 500.0000 500.0000 500.0000
## Mean of Sample Means              503.0670 503.5120 491.9000 497.5550 494.1392
## Theoretical Standard Error        288.9637 204.3282 166.8333 144.4818 129.2285
## Standard Deviation of Sample Means 287.1736 197.7254 173.6212 143.6107 127.7700
## P(winning > 600)                    0.4010   0.3210   0.2830   0.2380   0.2220
##                                    6 spins  7 spins  8 spins   9 spins
## Theoretical Mean                  500.0000 500.0000 500.0000 500.00000
## Mean of Sample Means              499.6518 500.2290 498.5840 499.63033
## Theoretical Standard Error        117.9689 109.2180 102.1641  96.32122
## Standard Deviation of Sample Means 118.4016 107.9148 106.3041  99.89172
## P(winning > 600)                    0.1910   0.1720   0.1760   0.15900
##                                    10 spins
## Theoretical Mean                  500.00000
## Mean of Sample Means              503.72550
## Theoretical Standard Error         91.37833
## Standard Deviation of Sample Means  88.71582
## P(winning > 600)                    0.14100
```

Observation: 1. Theoretical mean stays constant for all spin values. 2. Mean of sample means approximately equal to the theoretical mean. 3. SD of sample means approximately equal to the theoretical standard error. 4. The SD of sample decreases as the number of spins increases. 5. The probability of winning over $600 decreases as the number of spins increases.

Part 6 The CSV file SupermarketTransactions contains over 14,000 transactions made by supermarket customers over a period of approximately two years. (The data are not real, but real supermarket chains have huge data sets just like this one.) Column B contains the date of the purchase, column C is a unique identifier for each customer, columns D–H contain information about the customer, columns I–K contain the location of the store, columns L–N contain information about the product purchased, and the last two columns indicate the number of items purchased and the amount paid.

For this question, consider this data set the population of transactions.

```r
smtransac <- read.csv("SupermarketTransactions.csv")
head(smtransac)
```

```
##   Transaction PurchaseDate CustomerID Gender MaritalStatus Homeowner Children
## 1           1   12/18/2011       7223      F             S         Y        2
## 2           2   12/20/2011       7841      M             M         Y        5
## 3           3   12/21/2011       8374      F             M         N        2
## 4           4   12/21/2011       9619      M             M         Y        3
## 5           5   12/22/2011       1900      F             S         Y        3
## 6           6   12/22/2011       6696      F             M         Y        3
##    AnnualIncome          City StateOrProvince Country ProductFamily
## 1   $30K - $50K   Los Angeles              CA     USA          Food
## 2   $70K - $90K   Los Angeles              CA     USA          Food
## 3   $50K - $70K     Bremerton              WA     USA          Food
## 4   $30K - $50K      Portland              OR     USA          Food
## 5 $130K - $150K Beverly Hills              CA     USA         Drink
## 6   $10K - $30K Beverly Hills              CA     USA          Food
```

```
##   ProductDepartment      ProductCategory Units.Sold Revenue
## 1       Snack Foods           Snack Foods          5   27.38
## 2           Produce            Vegetables          5   14.90
## 3       Snack Foods           Snack Foods          3    5.52
## 4            Snacks                 Candy          4    4.44
## 5         Beverages Carbonated Beverages          4   14.00
## 6              Deli           Side Dishes          3    4.37
```

```r
str(smtransac)
```

```
## 'data.frame':    14059 obs. of  16 variables:
##  $ Transaction      : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ PurchaseDate     : Factor w/ 742 levels "1/1/2012","1/1/2013",..: 203 210 213 213 216 216 219 224
##  $ CustomerID       : int  7223 7841 8374 9619 1900 6696 9673 354 1293 7938 ...
##  $ Gender           : Factor w/ 2 levels "F","M": 1 2 1 2 1 1 2 1 2 2 ...
##  $ MaritalStatus    : Factor w/ 2 levels "M","S": 2 1 1 1 2 1 2 1 1 2 ...
##  $ Homeowner        : Factor w/ 2 levels "N","Y": 2 2 1 2 2 2 2 2 2 1 ...
##  $ Children         : int  2 5 2 3 3 3 2 2 3 1 ...
##  $ AnnualIncome     : Factor w/ 8 levels "$10K - $30K",..: 5 7 6 5 3 1 5 4 1 6 ...
##  $ City             : Factor w/ 23 levels "Acapulco","Bellingham",..: 8 8 4 12 3 3 13 23 2 15 ...
##  $ StateOrProvince  : Factor w/ 10 levels "BC","CA","DF",..: 2 2 8 6 2 2 6 8 8 2 ...
##  $ Country          : Factor w/ 3 levels "Canada","Mexico",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ ProductFamily    : Factor w/ 3 levels "Drink","Food",..: 2 2 2 2 1 2 2 2 3 3 ...
##  $ ProductDepartment: Factor w/ 22 levels "Alcoholic Beverages",..: 20 18 20 21 4 11 13 6 15 14 ...
##  $ ProductCategory  : Factor w/ 45 levels "Baking Goods",..: 42 45 42 7 15 41 5 13 16 35 ...
##  $ Units.Sold       : int  5 5 3 4 4 3 4 6 1 2 ...
##  $ Revenue          : num  27.38 14.9 5.52 4.44 14 ...
```

a. If you were interested in estimating the mean of Revenue for the population, why might it make sense to use a stratified sample, stratified by product family, to estimate this mean?

Because maybe the management wants to see which product family is growing or decreasing in mean revenue, and decides the inventory purhasing plan based on it. In this perspective, it does not make sense to have the mean revenue of all product families. Also, between different product families there is a relatively high variation in price range. Therefore, it's better to have the mean revenue of stratefied samples

b. Suppose you want to generate a stratified random sample, stratified by product family, and have the total sample size be 250. If you use proportional sample sizes, how many transactions should you sample from each of the three product families?

```r
#library(dplyr)
num <- smtransac %>%
  group_by(ProductFamily) %>%
  summarise(n=n()) %>%
  mutate(prop = n/sum(n))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```r
Numbers <- as.data.frame(num)
Proportions <- Numbers$prop
```

```r
number_in_sample <- matrix(round(Proportions * 250),1)
colnames(number_in_sample) <- c("Drink","Food","Non-Consumable")
rownames(number_in_sample) <- c("Sample Size")
number_in_sample
```

```
##             Drink Food Non-Consumable
## Sample Size    22  181             47
```

c. Using the sample sizes from part b, generate a corresponding stratified random sample. What are the individual sample means from the three product families? What are the sample standard deviations?

```r
# Sample drinks
Drinks <- smtransac %>%
  filter(ProductFamily == "Food" | ProductFamily == "Non-Consumable")
DrinksRevenue <- Drinks$Revenue
DrinksRevenue_Sample <-  sample(DrinksRevenue, as.numeric(number_in_sample[1,1]), replace = TRUE)

mean_drink <- mean(DrinksRevenue_Sample)
sd_drink <- sd(DrinksRevenue_Sample)

# Sample food
Food <- smtransac %>%
  filter(ProductFamily == "Drink" | ProductFamily == "Non-Consumable")
FoodRevenue <- Food$Revenue
FoodRevenue_Sample <-  sample(FoodRevenue, as.numeric(number_in_sample[1,2]), replace = TRUE)

mean_food <- mean(FoodRevenue_Sample)
sd_food <- sd(FoodRevenue_Sample)

# Sample non-consumable
Nonconsum <- smtransac %>%
  filter(ProductFamily == "Drink" | ProductFamily == "Food")
NonconsumRevenue <- Nonconsum$Revenue
NonconsumRevenue_Sample <-  sample(NonconsumRevenue, as.numeric(number_in_sample[1,3]), replace = TRUE)

mean_nc <- mean(NonconsumRevenue_Sample)
sd_nc <- sd(NonconsumRevenue_Sample)

table <- matrix(c(mean_drink, sd_drink, mean_food, sd_food, mean_nc, sd_nc), nrow = 2)
rownames(table) <- c("mean", "standard deviation")
colnames(table) <- c("Drink","Food","Non-Consumable")
table
```

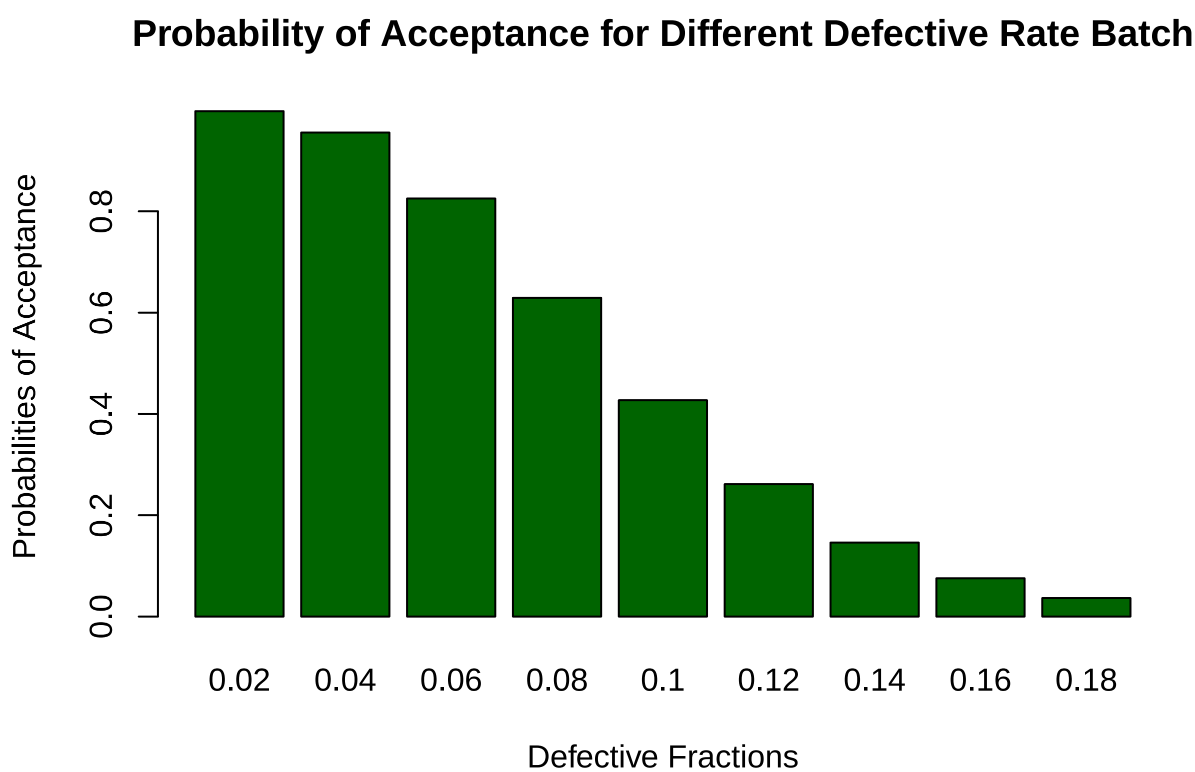
```
##                       Drink      Food Non-Consumable
## mean               8.672273 12.618287      12.999574
## standard deviation 6.480039  8.201987       7.722143
```

Part 7 Sampling is a very common practice in quality control. Sampling plans are created to control the quality of manufactured items that are ready to be shipped. To illustrate the use of a sampling plan, suppose that a chip manufacturing company produces and ships electronic computer chips in lots, each lot consisting of 1000 chips. This company's sampling plan specifies that quality control personnel should randomly sample 50 chips from each lot and accept the lot for shipping if the number of defective chips is four or fewer. The lot will be rejected if the number of defective chips is five or more.

Find the probability of accepting a lot as a function of the actual fraction of defective chips. In particular, let the actual fraction of defective chips in a given lot equal any of 0.02, 0.04, 0.06, 0.08, 0.10, 0.12, 0.14, 0.16, 0.18. Then compute the lot acceptance probability for each of these lot defective fractions.

Create a bar plot showing how the acceptance probability varies with the fraction defective. Comment on what you observe.

```
Def_Frac <- c(0.02, 0.04, 0.06, 0.08, 0.10, 0.12, 0.14, 0.16, 0.18)
probabilities <- phyper(4, Def_Frac*1000, (1-Def_Frac)*1000, 50)

barplot(probabilities, names.arg = Def_Frac, ylab  = "Probabilities of Acceptance",
        xlab  = "Defective Fractions", col = "Dark Green",
        main = "Probability of Acceptance for Different Defective Rate Batch")
```



As the defective fraction increases, the probability of accepting it gets smaller

Part 8 Leslie loves to swim and compete in races. The time it takes Leslie to swim 100 yards in a race follows a normal distribution with the mean of 62 seconds and standard deviation of 2 seconds. In her next five races, what is the probability that she will swim under a minute exactly twice?

```
prob_under_60 <- pnorm(60, 62, 2, lower.tail = TRUE)
prob_under_60_twice  <- dbinom(2, 5, prob_under_60)
prob_under_60_twice
```

```
## [1] 0.1499101
```

Part 9 Many transportation vehicles such as airplanes are built with redundant systems for safety. Many components have backup systems so that if one or more components fail, backup components take over and this assures the safe, uninterrupted operation of the component and the vehicle as a whole. For example, consider one main component of an international trans-Atlantic airplane that has n duplicated systems (i.e., one original system and n - 1 backup systems). Each of these systems functions independently of the others, with probability 0.98. This component functions successfully if at least one of the n systems functions properly.

a. Find the probability that this airplane component functions successfully if n = 2.

```
pbinom(0, 2, 0.98, lower.tail = FALSE)
```

```
## [1] 0.9996
```

b. Find the probability that this airplane component functions successfully if n = 4.

```
pbinom(0, 4, 0.98, lower.tail = FALSE)
```

```
## [1] 0.9999998
```

c. What is the minimum number n of duplicated systems that must be incorporated into this airplane component to ensure at least a 0.9999 probability of successful operation?

```
Prob_Distr <- matrix(pbinom(0, 1:5, 0.98, lower.tail = FALSE), 1)
colnames(Prob_Distr) <- c(1, 2, 3, 4, 5)
rownames(Prob_Distr) <- c("Probability of Success")
Prob_Distr
```

```
##                           1      2        3         4 5
## Probability of Success 0.98 0.9996 0.999992 0.9999998 1
```

According to probability distribution, at least 3 duplicated systems are needed to ensure at least a 0.9999 probability of successful operation

Part 10 Suppose you work for a survey research company. In a typical survey, you mail questionnaires to 150 companies. Some of these companies might decide not to respond. Assume that the nonresponse rate is 45%; that is, each company's probability of not responding, independently of the others, is 0.45. Suppose your company does this survey in two "waves." It mails the 150 questionnaires and waits a certain period for the responses. Assume that the nonresponse rate for this first wave is 45%. However, after this initial period, your company follows up (by telephone, say) on the nonrespondents, asking them to please respond. Suppose that the nonresponse rate on this second wave is 70%; that is, each original nonrespondent now responds with probability 0.3, independently of the others. Your company now wants to find the probability of obtaining at least 110 responses total. What is the probability (fraction of successes) of getting this required number of returns from both waves?

```
required_response <- 110
sample_size <- 150
prob_of_response <- 1 -  0.45 + 0.45 * 0.3
pbinom(required_response-1, sample_size, prob_of_response, lower.tail = FALSE)
```

```
## [1] 0.1167226
```

Part 11 A chain of women's clothing stores operating throughout the US recently ran a promotion in which discount coupons were sent to customers. Data collected from a sample of 100 in-store transactions during one day while the promotion was running are contained in the Dataset 11. The Proprietary Card method of payment refers to charges made using the stores credit card. Customers who made a purchase using a discount coupon are referred to as promotional customers and customers who made a purchase but did not use a discount coupon are referred to as regular customers. Because the promotional coupons were not sent to the regular customers, management considers the sales made to people presenting the promotional coupons as sales it would not otherwise make.

Most of the variables in the dataset are pretty self-explanatory. Two bear some clarification: Items: The total number of items purchased Net Sales: The total amount ($) charged to the credit card

The management would like to use this data to learn more about the customer base and to evaluate the promotion involving discount coupons.

a. Create a percent frequency distribution for key variables. You have to think about which variables should be summarized to provide management some insights about customers.

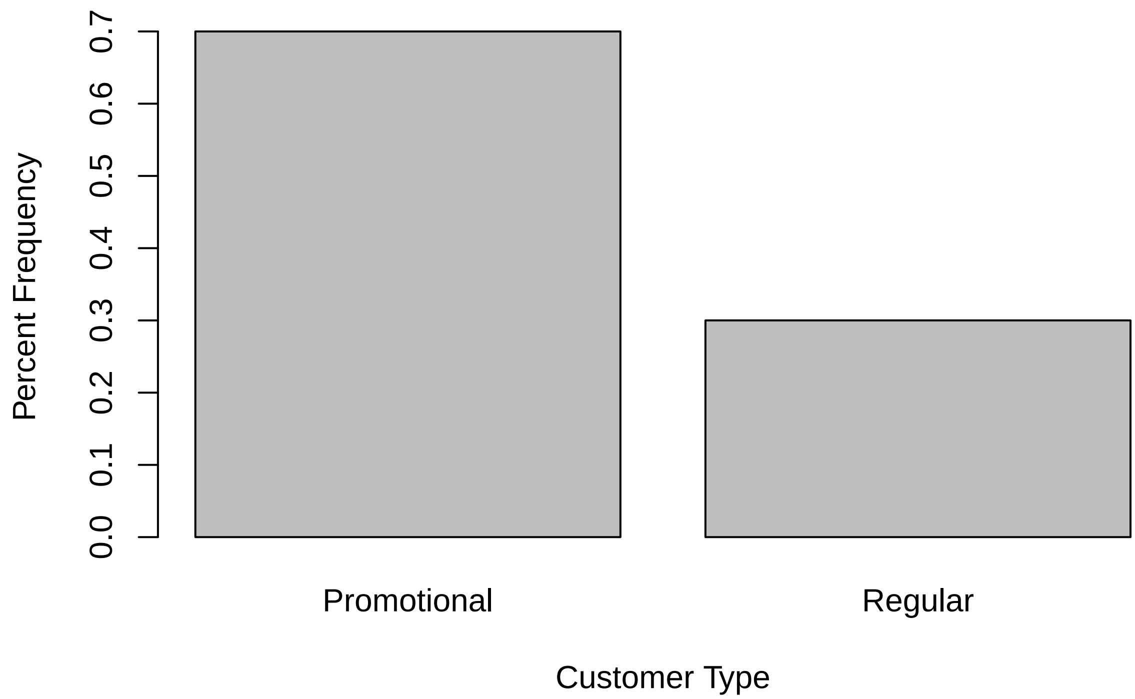
```
q11 <- read.csv("Dataset 11.csv")

Type_of_Customer <- q11 %>%
  group_by(Type.of.Customer) %>%
  summarise(n=n()) %>%
  mutate(prop_type_of_cus = n/sum(n))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
Customer_Type <- as.data.frame(Type_of_Customer)
Customer_Type
```

```
##   Type.of.Customer  n prop_type_of_cus
## 1      Promotional 70              0.7
## 2          Regular 30              0.3
```

```
barplot(Customer_Type[,3], names.arg = c("Promotional","Regular"),
        xlab = "Customer Type", ylab = "Percent Frequency")
```



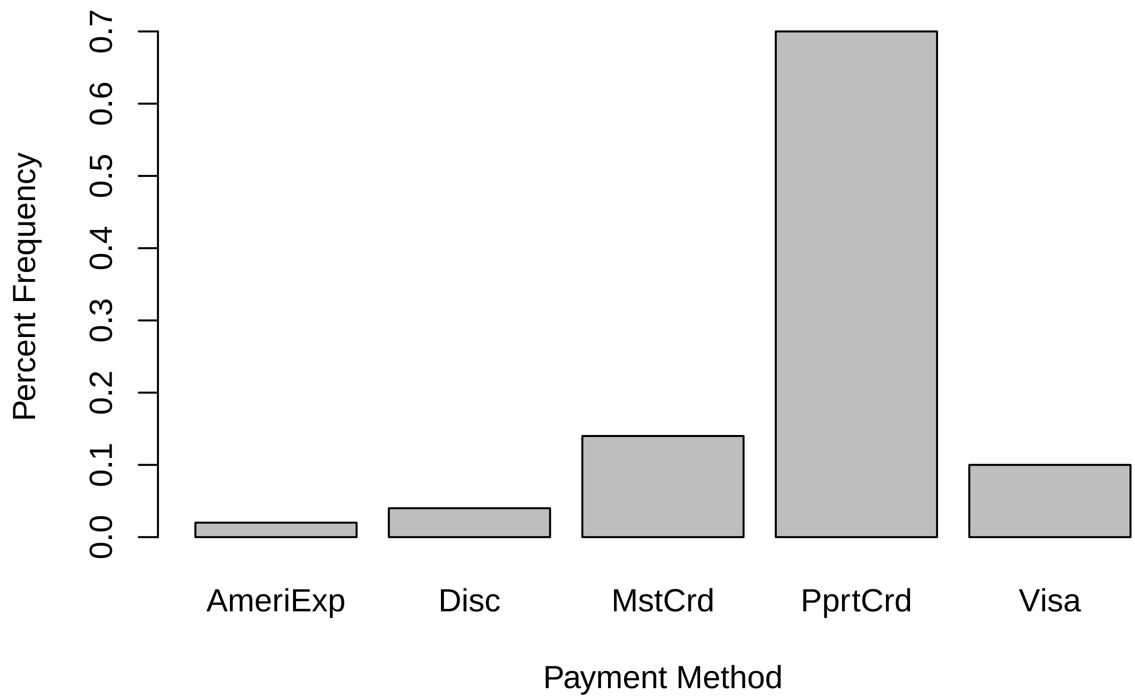

```
Method_of_Payment <- q11 %>%
  group_by(Method.of.Payment) %>%
  summarise(n=n()) %>%
  mutate(prop_payment_method = n/sum(n))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
Payment_Method <- as.data.frame(Method_of_Payment)
Payment_Method
```

```
##   Method.of.Payment  n prop_payment_method
## 1  American Express  2                0.02
## 2          Discover  4                0.04
## 3        MasterCard 14                0.14
## 4  Proprietary Card 70                0.70
## 5              Visa 10                0.10
```

```
barplot(Payment_Method[,3], names.arg = c("AmeriExp","Disc","MstCrd",
                                          "PprtCrd","Visa"),
        xlab = "Payment Method", ylab = "Percent Frequency")
```



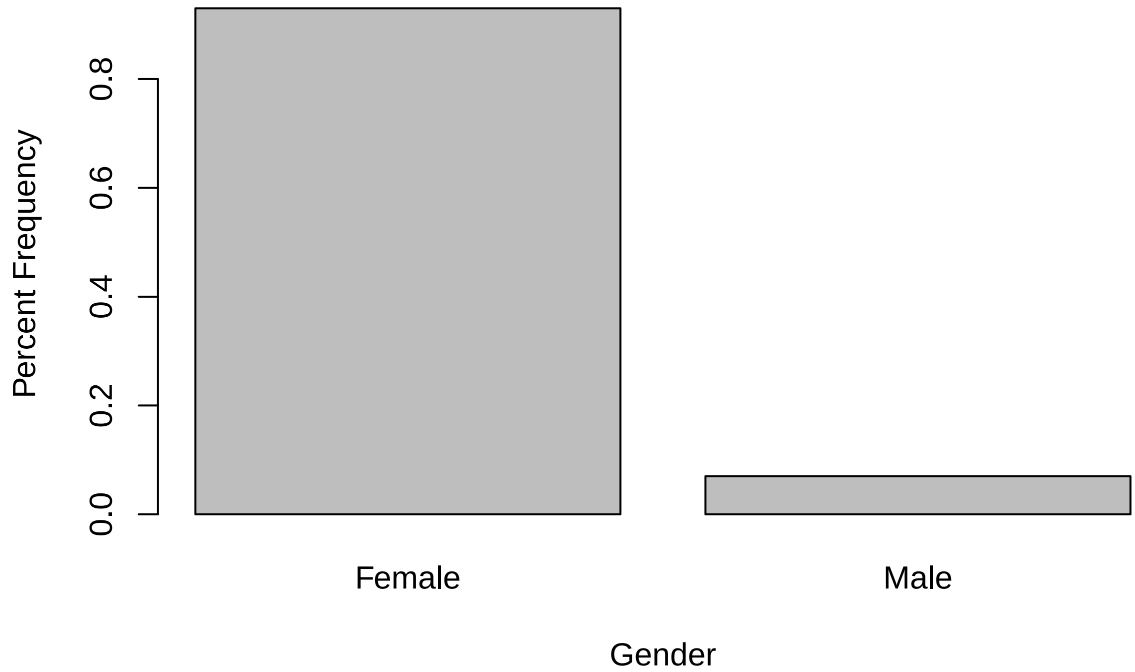

```
Gender <- q11 %>%
  group_by(Gender) %>%
  summarise(n=n()) %>%
  mutate(prop_gender = n/sum(n))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
Gender_New <- as.data.frame(Gender)
Gender_New
```

```
##   Gender  n prop_gender
## 1 Female 93        0.93
## 2   Male  7        0.07
```

```
barplot(Gender_New[,3], names.arg = c("Female", "Male"),
        xlab = "Gender", ylab = "Percent Frequency")
```



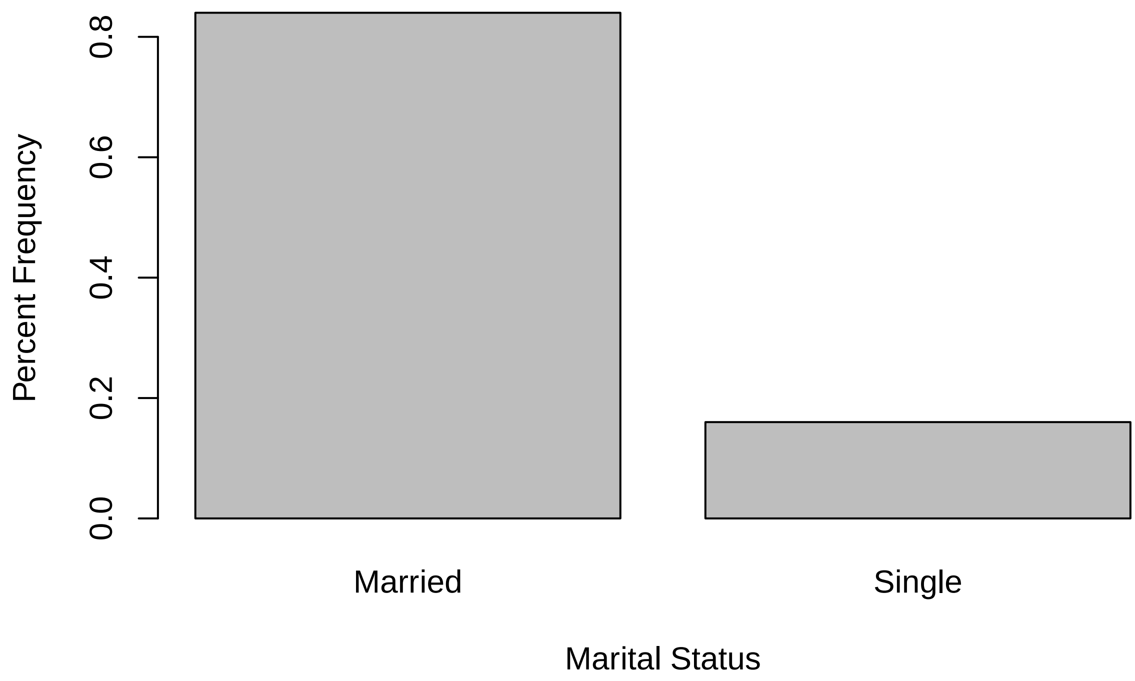

```
Marital_Status <- q11 %>%
  group_by(Marital.Status) %>%
  summarise(n=n()) %>%
  mutate(prop_marital_status = n/sum(n))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
Marital_Status_New <- as.data.frame(Marital_Status)
Marital_Status_New
```

```
##   Marital.Status  n prop_marital_status
## 1        Married 84                0.84
## 2         Single 16                0.16
```

```
barplot(Marital_Status_New[,3], names.arg = c("Married", "Single"),
        xlab = "Marital Status", ylab = "Percent Frequency")
```




```r
Age <- q11 %>%
  group_by(Age) %>%
  summarise(n=n()) %>%
  mutate(prop_age = n/sum(n))
```

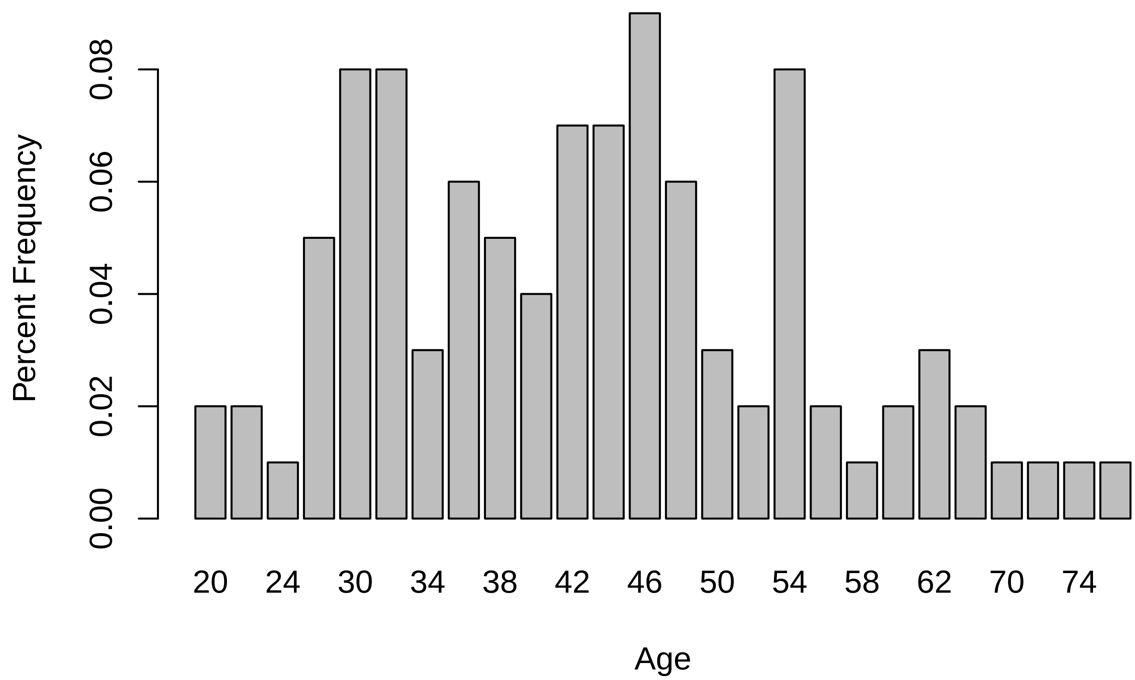
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```r
Age_New <- as.data.frame(Age)
Age_New
```

```
##    Age n prop_age
## 1   20 2     0.02
## 2   22 2     0.02
## 3   24 1     0.01
## 4   28 5     0.05
## 5   30 8     0.08
## 6   32 8     0.08
## 7   34 3     0.03
## 8   36 6     0.06
## 9   38 5     0.05
## 10  40 4     0.04
## 11  42 7     0.07
## 12  44 7     0.07
## 13  46 9     0.09
## 14  48 6     0.06
## 15  50 3     0.03
## 16  52 2     0.02
## 17  54 8     0.08
## 18  56 2     0.02
## 19  58 1     0.01
## 20  60 2     0.02
```

```
## 21   62 3      0.03
## 22   68 2      0.02
## 23   70 1      0.01
## 24   72 1      0.01
## 25   74 1      0.01
## 26   78 1      0.01
```

```
barplot(Age_New[,3], names.arg = Age_New[,1],
        xlab = "Age", ylab = "Percent Frequency")
```

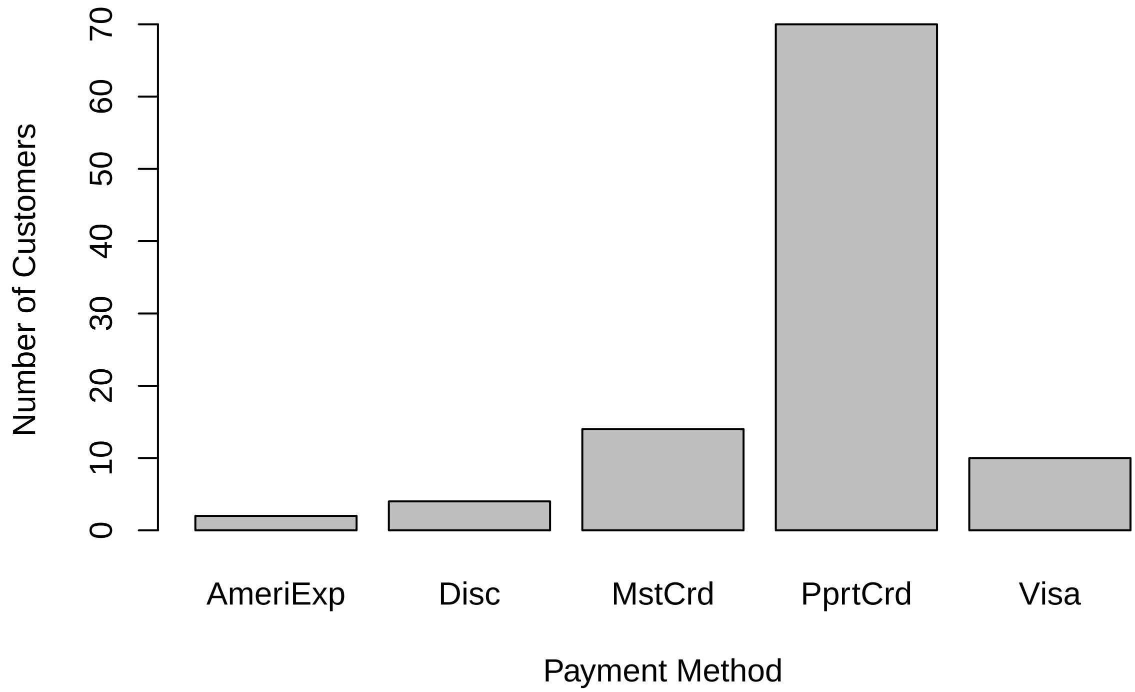


b. A bar chart showing the number of customer purchases attributable to the method of payment.

```
Method_of_Payment_b <- q11 %>%
  group_by(Method.of.Payment) %>%
  summarise(n=n())
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
Payment_Method_b <- as.data.frame(Method_of_Payment_b)
barplot(Payment_Method_b[,2], names.arg = c("AmeriExp","Disc","MstCrd",
                                            "PprtCrd","Visa"),
        xlab = "Payment Method", ylab = "Number of Customers")
```



c. A cross tabulation of type of customer (regular and promotional) versus net sales. Comment on what you observe.

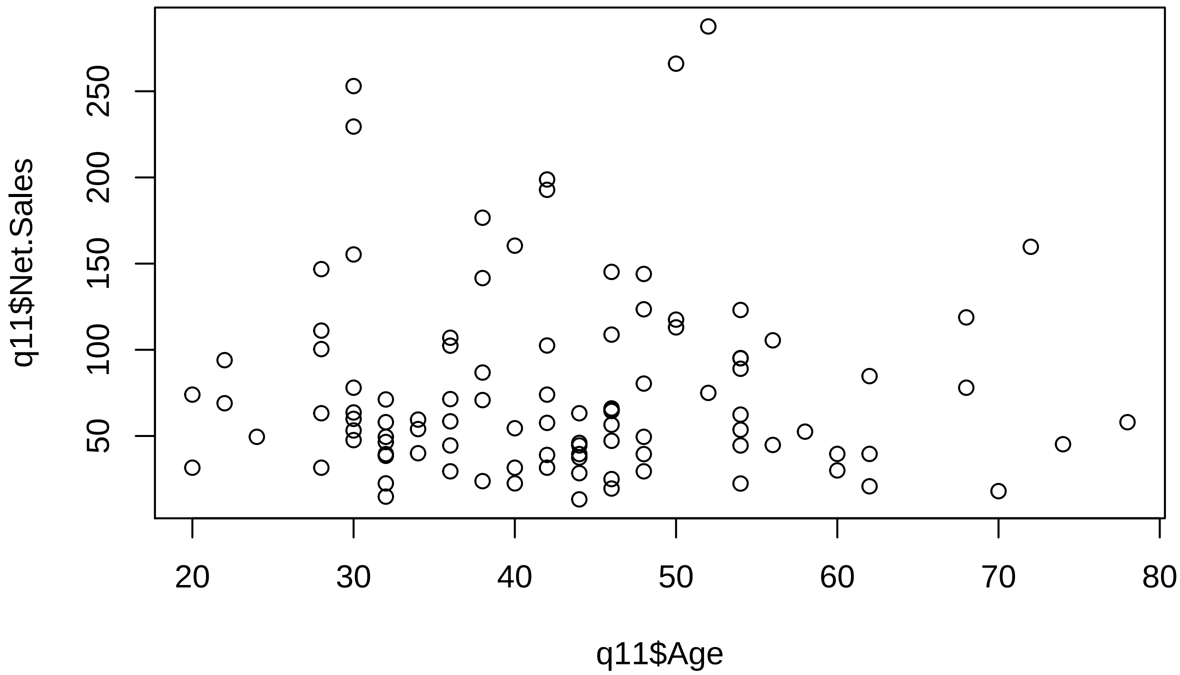
```
q11$Groups <- case_when(
  q11$Net.Sales <= 50 ~ "0-50",
  q11$Net.Sales <= 100 ~ "50-100",
  q11$Net.Sales <= 150 ~ "100-150",
  q11$Net.Sales <= 200 ~ "150-200",
  q11$Net.Sales <= 250 ~ "200-250",
  q11$Net.Sales <= 300 ~ "250-300",
)

xtabs(~Groups + Type.of.Customer, data=q11)
```

```
##          Type.of.Customer
## Groups    Promotional Regular
##   0-50             24      15
##   100-150          12       4
##   150-200           5       1
##   200-250           1       0
##   250-300           3       0
##   50-100           25      10
```

Most customers buy stuff at lower price range, and coupons have encourage customres buy more expensive things.

d. A scatterplot to explore the relationship between net sales and customer age.

```r
plot(q11$Age, q11$Net.Sales)
```



Part 12 Toll booths on the New York State Thruway are often congested because of the large number of cars waiting to pay. A consultant working for the state concluded that if service times are measured from the time a car stops in line until it leaves, service times are exponentially distributed with a mean of 2.7 minutes. What proportion of cars can get through the toll booth in less than 3 minutes?

```r
pexp(3, 1/2.7)
```

```
## [1] 0.670807
```