STAT 4360 - Project 2

Yebom Kim

2024-09-26

Section 1

```
# 1-a)
cat("With the graphs we got, we can see most of wines have high clarity  and less aroma. I think they a
has proportion relationship. High body wines have high clarity. ")
```

```
## With the graphs we got, we can see most of wines have high clarity  and less aroma. I think they are
## has proportion relationship. High body wines have high clarity.
```

```
# 1-e)
wine_data <- read.table("/Users/springkim/Downloads/wine.txt", header=TRUE)
final_model <- lm(Quality ~ Aroma + Body + Region, data=wine_data)
summary(final_model)
```

```
##
## Call:
## lm(formula = Quality ~ Aroma + Body + Region, data = wine_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.2580 -0.6929  0.1856  0.7619  2.6041
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   4.6143     1.4626   3.155  0.00335 **
## Aroma         1.0186     0.3047   3.343  0.00203 **
## Body          0.5438     0.3488   1.559  0.12820
## Region        0.1808     0.3499   0.517  0.60876
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.446 on 34 degrees of freedom
## Multiple R-squared:  0.5408, Adjusted R-squared:  0.5002
## F-statistic: 13.34 on 3 and 34 DF,  p-value: 6.43e-06
```

```
cat("Other questions are on the down side with the codes")
```

```
## Other questions are on the down side with the codes
```

Section 2 for Question 1

```
# Load the data
wine_data <- read.table("/Users/springkim/Downloads/wine.txt", header=TRUE)

# Explore the dataset
summary(wine_data)
```

```
##     Clarity           Aroma            Body           Flavor
##  Min.   :0.5000   Min.   :3.300   Min.   :2.600   Min.   :2.900
##  1st Qu.:0.8250   1st Qu.:4.125   1st Qu.:4.150   1st Qu.:4.225
##  Median :1.0000   Median :4.650   Median :4.750   Median :4.800
##  Mean   :0.9237   Mean   :4.847   Mean   :4.684   Mean   :4.768
##  3rd Qu.:1.0000   3rd Qu.:5.450   3rd Qu.:5.375   3rd Qu.:5.500
##  Max.   :1.0000   Max.   :7.700   Max.   :6.600   Max.   :7.000
##     Oakiness        Quality          Region
##  Min.   :2.900   Min.   : 7.90   Min.   :1.000
##  1st Qu.:3.700   1st Qu.:11.15   1st Qu.:1.000
##  Median :4.100   Median :12.45   Median :2.000
##  Mean   :4.255   Mean   :12.44   Mean   :1.868
##  3rd Qu.:4.775   3rd Qu.:13.75   3rd Qu.:3.000
##  Max.   :6.000   Max.   :16.10   Max.   :3.000
```

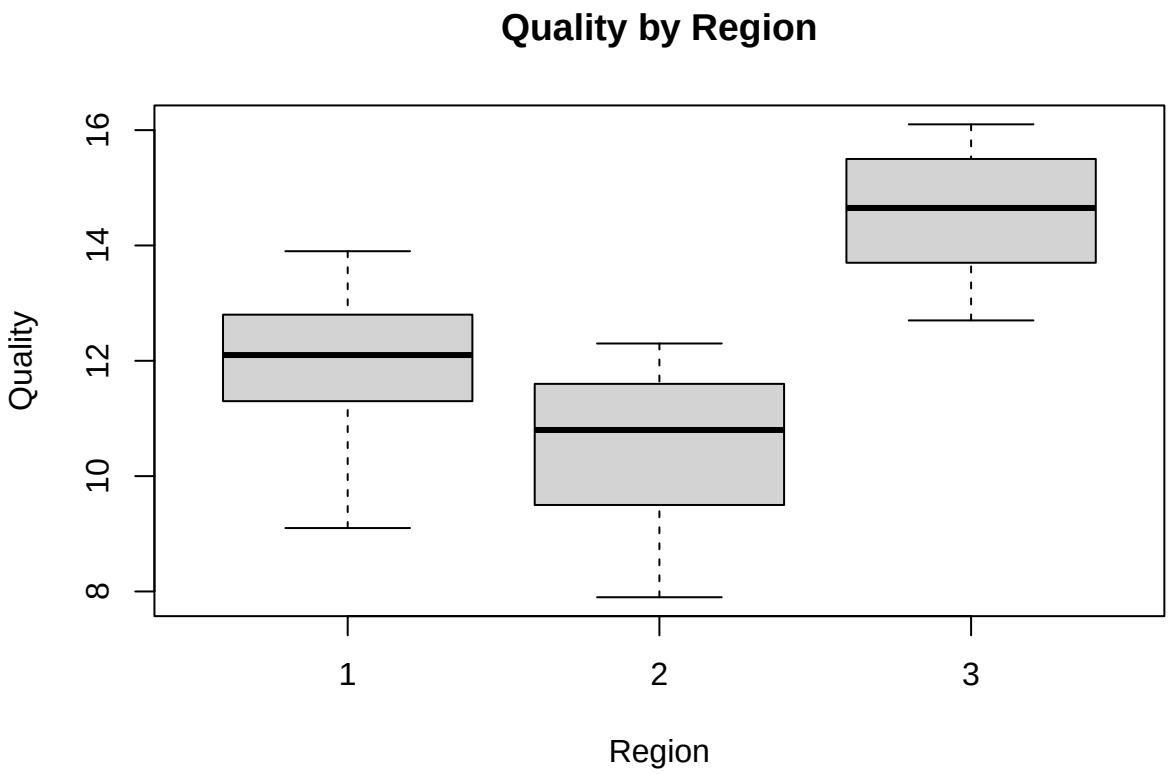
```
str(wine_data)
```

```
## 'data.frame':    38 obs. of  7 variables:
##  $ Clarity : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ Aroma   : num  3.3 4.4 3.9 3.9 5.6 4.6 4.8 5.3 4.3 4.3 ...
##  $ Body    : num  2.8 4.9 5.3 2.6 5.1 4.7 4.8 4.5 4.3 3.9 ...
##  $ Flavor  : num  3.1 3.5 4.8 3.1 5.5 5 4.8 4.3 3.9 4.7 ...
##  $ Oakiness: num  4.1 3.9 4.7 3.6 5.1 4.1 3.3 5.2 2.9 3.9 ...
##  $ Quality : num  9.8 12.6 11.9 11.1 13.3 12.8 12.8 12 13.6 13.9 ...
##  $ Region  : int  1 1 1 1 1 1 1 1 3 1 ...
```

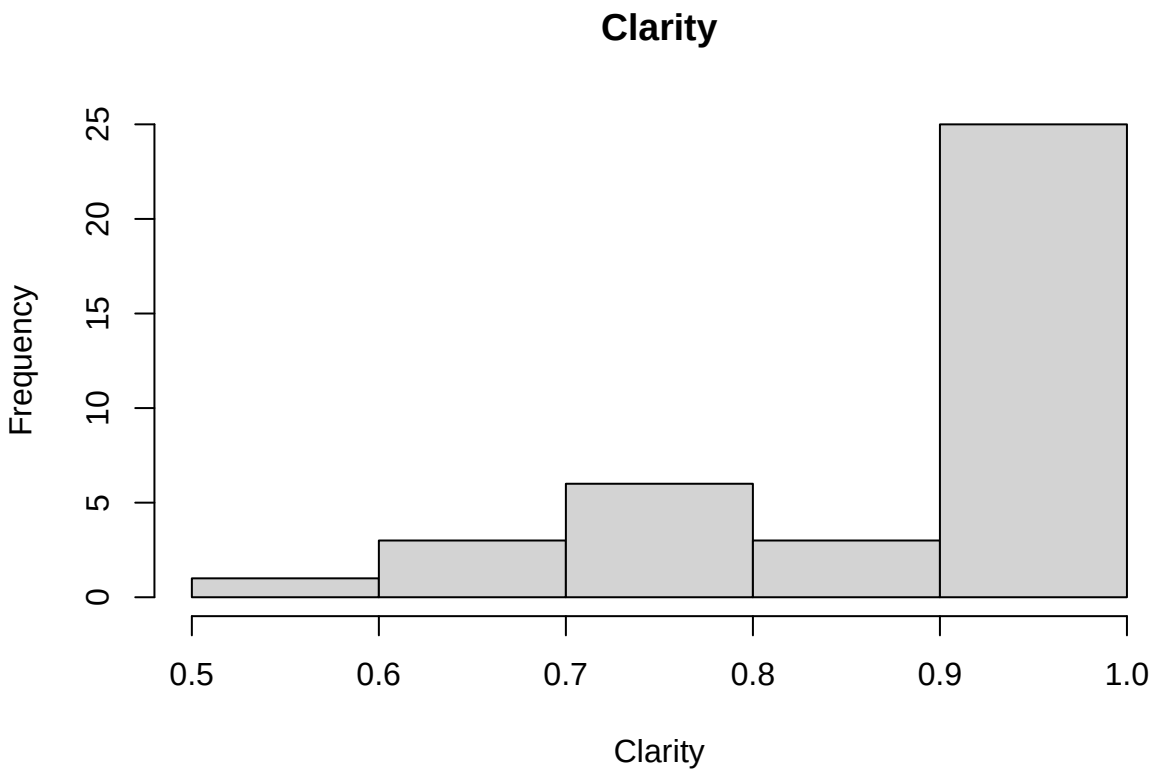
```
# Convert Region to a factor (qualitative predictor)
wine_data$Region <- as.factor(wine_data$Region)

# View the first few rows of data
head(wine_data)
```

```
##   Clarity Aroma Body Flavor Oakiness Quality Region
## 1       1   3.3  2.8    3.1      4.1     9.8      1
## 2       1   4.4  4.9    3.5      3.9    12.6      1
## 3       1   3.9  5.3    4.8      4.7    11.9      1
## 4       1   3.9  2.6    3.1      3.6    11.1      1
## 5       1   5.6  5.1    5.5      5.1    13.3      1
## 6       1   4.6  4.7    5.0      4.1    12.8      1
```

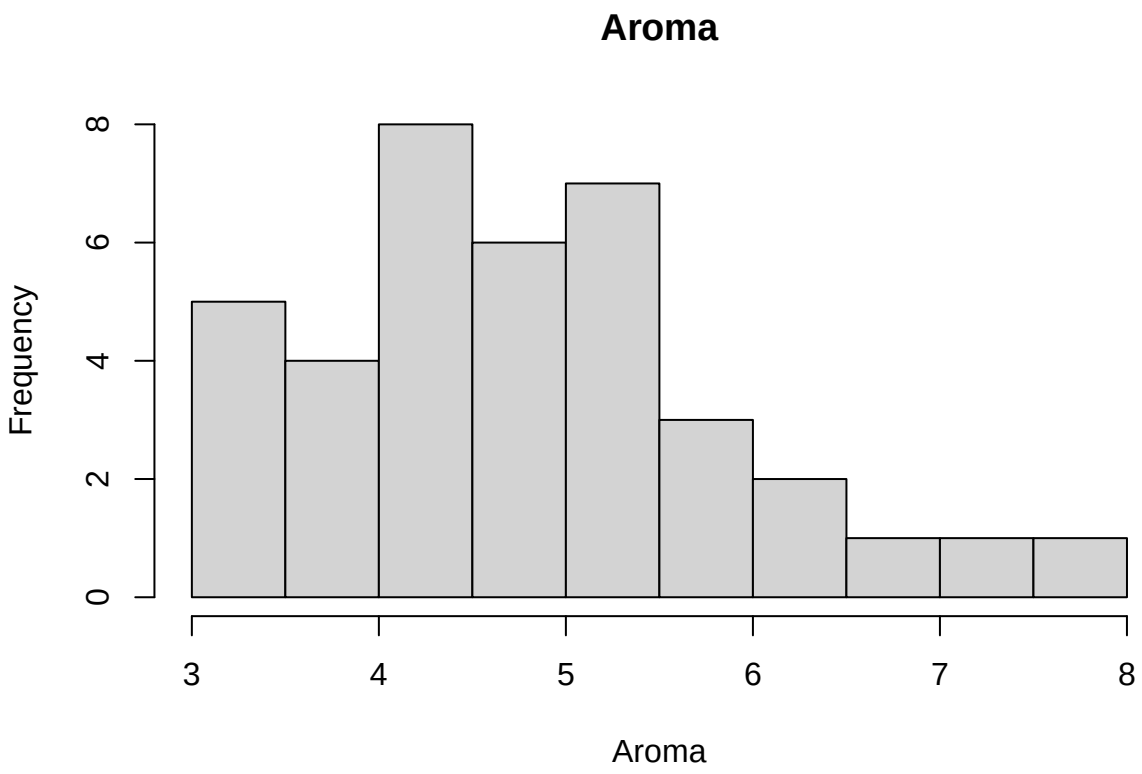
```
## Problem 1-a
boxplot(Quality ~ Region, data=wine_data, main="Quality by Region")
```



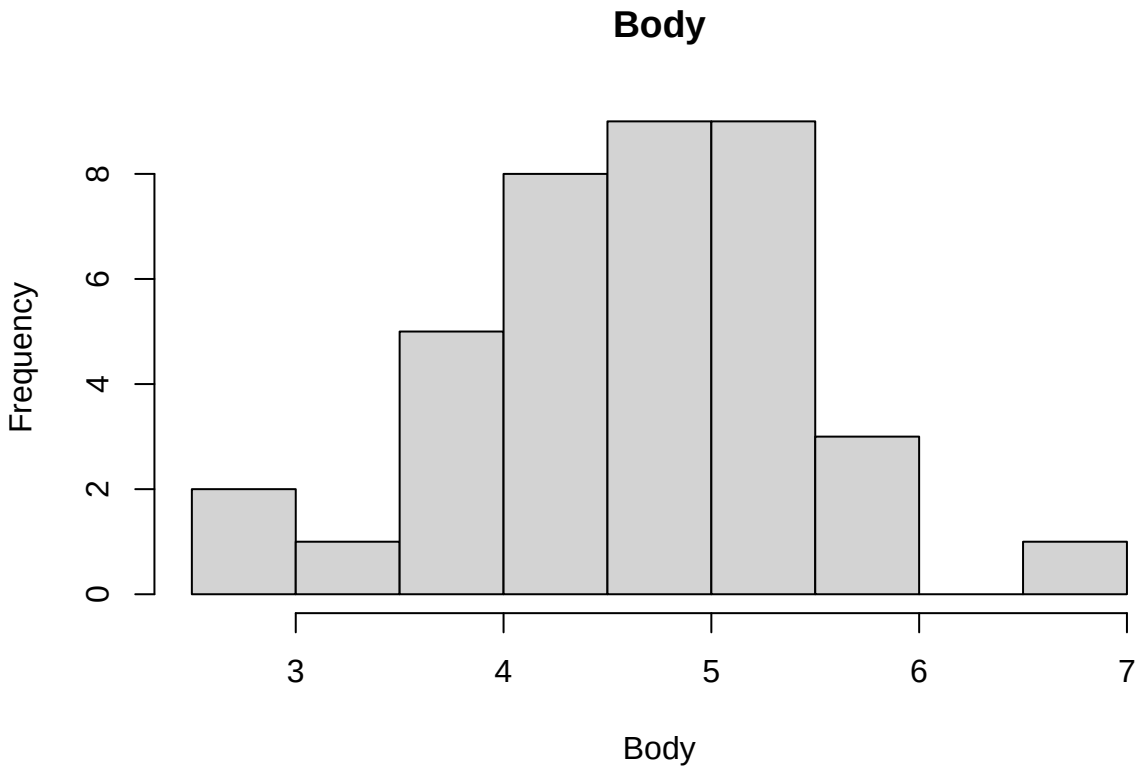

```
hist(wine_data$Clarity, main="Clarity", xlab="Clarity")
```



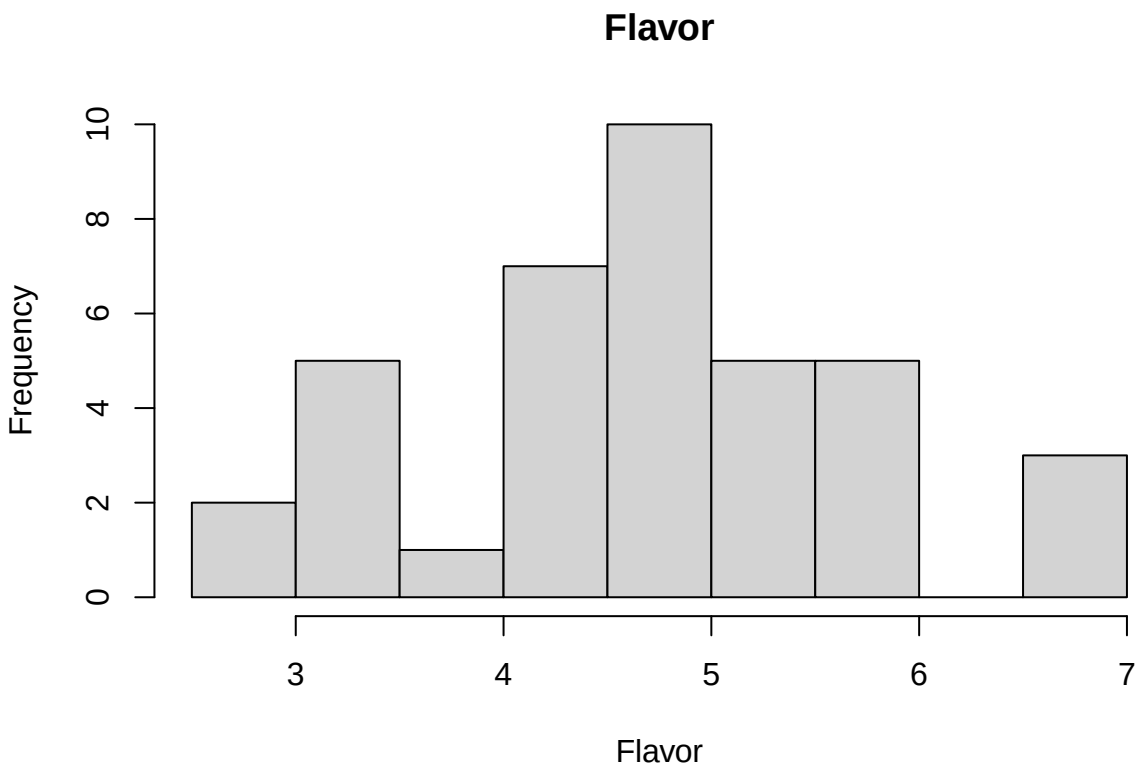

```r
hist(wine_data$Aroma, main="Aroma", xlab="Aroma")
```



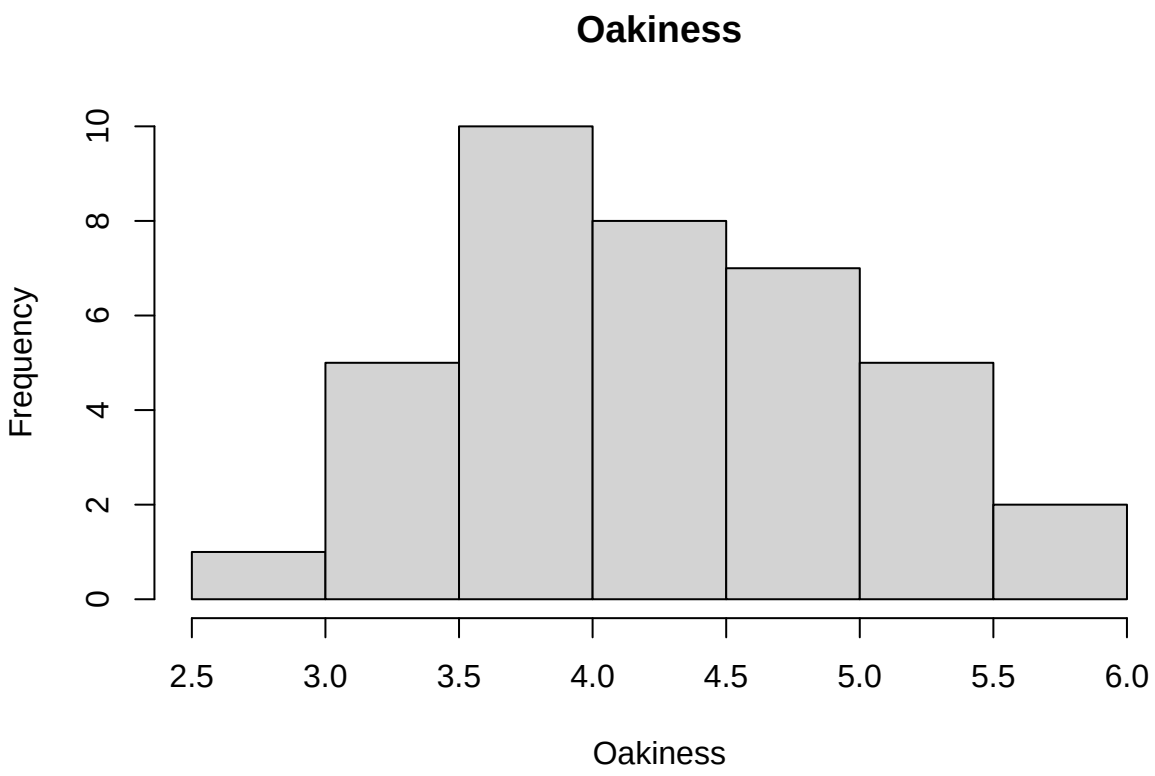

```r
hist(wine_data$Body, main="Body", xlab="Body")
```




```
hist(wine_data$Flavor, main="Flavor", xlab="Flavor")
```




```r
hist(wine_data$Oakiness, main="Oakiness", xlab="Oakiness")
```




```r
## Problem 1-b
# Fit simple linear regression models for each predictor
model_clarity <- lm(Quality ~ Clarity, data=wine_data)
model_aroma <- lm(Quality ~ Aroma, data=wine_data)
model_body <- lm(Quality ~ Body, data=wine_data)
model_flavor <- lm(Quality ~ Flavor, data=wine_data)
model_oakiness <- lm(Quality ~ Oakiness, data=wine_data)
model_region <- lm(Quality ~ Region, data=wine_data)

# Display the summary of each model
summary(model_clarity)
```

```
##
## Call:
## lm(formula = Quality ~ Clarity, data = wine_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5257 -1.3227  0.0947  1.2773  3.7681
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  12.0034     2.5610   4.687 3.89e-05 ***
## Clarity       0.4692     2.7486   0.171    0.865
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 2.073 on 36 degrees of freedom
## Multiple R-squared:  0.0008089,  Adjusted R-squared:  -0.02695
## F-statistic: 0.02914 on 1 and 36 DF,  p-value: 0.8654
```

```
summary(model_aroma)
```

```
##
## Call:
## lm(formula = Quality ~ Aroma, data = wine_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4726 -0.8574 -0.0091  0.8346  2.2563
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   5.9583     1.1050   5.392 4.51e-06 ***
## Aroma         1.3365     0.2226   6.004 6.87e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.466 on 36 degrees of freedom
## Multiple R-squared:  0.5003, Adjusted R-squared:  0.4864
## F-statistic: 36.04 on 1 and 36 DF,  p-value: 6.871e-07
```

```
summary(model_body)
```

```
##
## Call:
## lm(formula = Quality ~ Body, data = wine_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.9669 -0.8386  0.0620  1.2204  3.4502
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   6.0580     1.6441   3.685 0.000748 ***
## Body          1.3618     0.3458   3.938 0.000361 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.734 on 36 degrees of freedom
## Multiple R-squared:  0.3011, Adjusted R-squared:  0.2817
## F-statistic: 15.51 on 1 and 36 DF,  p-value: 0.0003612
```

```
summary(model_flavor)
```

```
##
## Call:
## lm(formula = Quality ~ Flavor, data = wine_data)
```

```
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.38583 -0.72226 -0.00756  0.62006  2.52822
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   4.9414     0.9911   4.986 1.57e-05 ***
## Flavor        1.5719     0.2033   7.732 3.68e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.271 on 36 degrees of freedom
## Multiple R-squared:  0.6242, Adjusted R-squared:  0.6137
## F-statistic: 59.79 on 1 and 36 DF,  p-value: 3.683e-09
```

```
summary(model_oakiness)
```

```
##
## Call:
## lm(formula = Quality ~ Oakiness, data = wine_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.6483 -1.3886 -0.0527  1.2907  3.6429
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  12.9916     1.9918   6.522  1.4e-07 ***
## Oakiness     -0.1304     0.4614  -0.283    0.779
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.071 on 36 degrees of freedom
## Multiple R-squared:  0.002213,   Adjusted R-squared:  -0.0255
## F-statistic: 0.07984 on 1 and 36 DF,  p-value: 0.7791
```

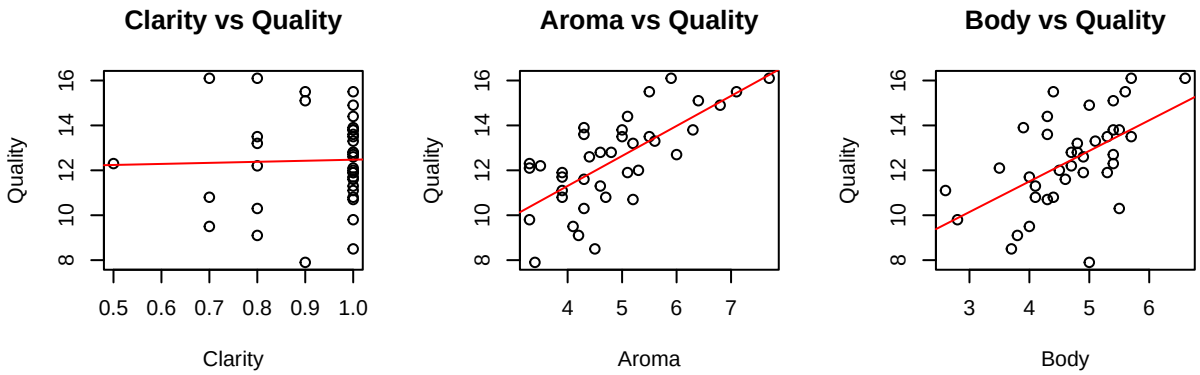
```
summary(model_region)
```

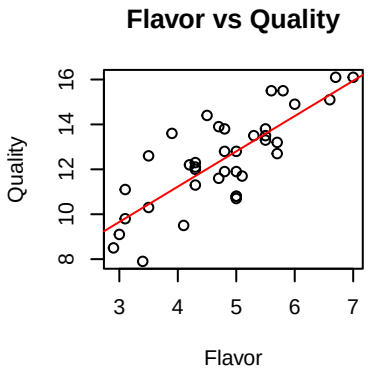
```
##
## Call:
## lm(formula = Quality ~ Region, data = wine_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8765 -0.8532  0.2395  0.9167  1.9235
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  11.9765     0.3180  37.662  < 2e-16 ***
## Region2      -1.5320     0.5405  -2.834  0.00757 **
## Region3       2.6069     0.4944   5.273 7.01e-06 ***
## ---
```

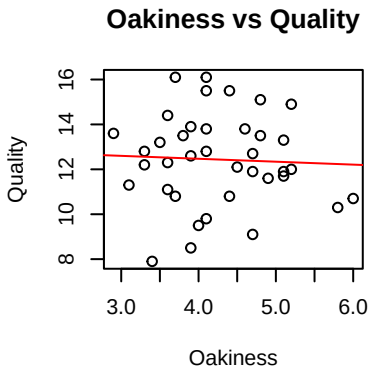
```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.311 on 35 degrees of freedom
## Multiple R-squared:  0.6113, Adjusted R-squared:  0.5891
## F-statistic: 27.52 on 2 and 35 DF,  p-value: 6.587e-08
```

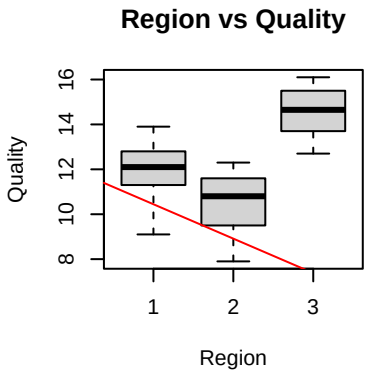
```r
# Create scatter plots with regression lines for each predictor
par(mfrow=c(2, 3)) # Arrange the plots in a grid

plot(wine_data$Clarity, wine_data$Quality, main="Clarity vs Quality", xlab="Clarity", ylab="Quality")
abline(model_clarity, col="red")

plot(wine_data$Aroma, wine_data$Quality, main="Aroma vs Quality", xlab="Aroma", ylab="Quality")
abline(model_aroma, col="red")

plot(wine_data$Body, wine_data$Quality, main="Body vs Quality", xlab="Body", ylab="Quality")
abline(model_body, col="red")

plot(wine_data$Flavor, wine_data$Quality, main="Flavor vs Quality", xlab="Flavor", ylab="Quality")
abline(model_flavor, col="red")

plot(wine_data$Oakiness, wine_data$Quality, main="Oakiness vs Quality", xlab="Oakiness", ylab="Quality")
abline(model_oakiness, col="red")

plot(wine_data$Region, wine_data$Quality, main="Region vs Quality", xlab="Region", ylab="Quality")
abline(model_region, col="red")
```

```
## Warning in abline(model_region, col = "red"): only using the first two of 3
## regression coefficients
```










```
## Problem 1-c
# Fit a multiple regression model using all predictors
multiple_model <- lm(Quality ~ Clarity + Aroma + Body + Flavor + Oakiness + Region, data=wine_data)

# Display the summary of the multiple regression model
summary(multiple_model)
```

```
##
## Call:
## lm(formula = Quality ~ Clarity + Aroma + Body + Flavor + Oakiness +
##     Region, data = wine_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.80824 -0.58413 -0.02081  0.48627  1.70909
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.81437    1.96944   3.968 0.000417 ***
## Clarity      0.01705    1.45627   0.012 0.990736
## Aroma        0.08901    0.25250   0.353 0.726908
## Body         0.07967    0.26772   0.298 0.768062
## Flavor       1.11723    0.24026   4.650 6.25e-05 ***
## Oakiness    -0.34644    0.23301  -1.487 0.147503
## Region2     -1.51285    0.39227  -3.857 0.000565 ***
## Region3      0.97259    0.51017   1.906 0.066218 .
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9154 on 30 degrees of freedom
## Multiple R-squared:  0.8376, Adjusted R-squared:  0.7997
## F-statistic:  22.1 on 7 and 30 DF,  p-value: 3.295e-10
```

```
## Problem 1-d
# Start with full model including all predictors and interaction between Region and others
full_model <- lm(Quality ~ Clarity + Aroma + Body + Flavor + Oakiness + Region +
                 Region:Clarity + Region:Aroma + Region:Body + Region:Flavor + Region:Oakiness, data=wi

# Perform backward stepwise selection to simplify the model
library(MASS)
final_model <- stepAIC(full_model, direction="backward")
```

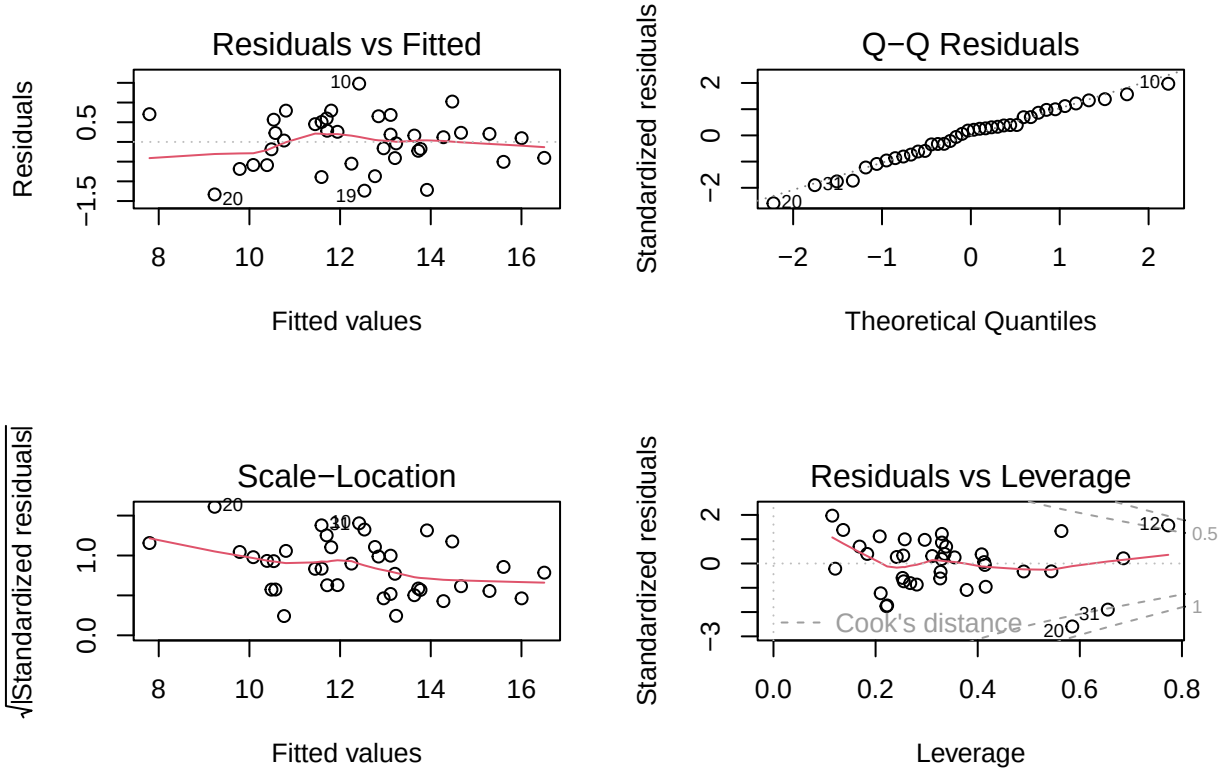
```
## Start:  AIC=-0.12
## Quality ~ Clarity + Aroma + Body + Flavor + Oakiness + Region +
##     Region:Clarity + Region:Aroma + Region:Body + Region:Flavor +
##     Region:Oakiness
##
##                   Df Sum of Sq    RSS     AIC
## - Aroma:Region     2    0.0229 14.712 -4.0591
## - Flavor:Region    2    1.0100 15.699 -1.5913
## - Oakiness:Region  2    1.5643 16.253 -0.2728
## <none>                         14.689 -0.1184
## - Body:Region      2    4.1828 18.872  5.4032
## - Clarity:Region   2    4.7222 19.411  6.4741
##
## Step:  AIC=-4.06
## Quality ~ Clarity + Aroma + Body + Flavor + Oakiness + Region +
##     Clarity:Region + Body:Region + Flavor:Region + Oakiness:Region
##
##                   Df Sum of Sq    RSS     AIC
## - Aroma            1    0.1698 14.882 -5.6230
## - Flavor:Region    2    1.0633 15.775 -5.4073
## <none>                         14.712 -4.0591
## - Oakiness:Region  2    1.9162 16.628 -3.4065
## - Body:Region      2    4.4792 19.191  2.0408
## - Clarity:Region   2    4.7625 19.474  2.5978
##
## Step:  AIC=-5.62
## Quality ~ Clarity + Body + Flavor + Oakiness + Region + Clarity:Region +
##     Body:Region + Flavor:Region + Oakiness:Region
##
##                   Df Sum of Sq    RSS     AIC
## - Flavor:Region    2    1.0526 15.934 -7.0261
## <none>                         14.882 -5.6230
## - Oakiness:Region  2    2.4006 17.282 -3.9402
## - Body:Region      2    4.5777 19.459  0.5684
## - Clarity:Region   2    5.2764 20.158  1.9088
##
## Step:  AIC=-7.03
## Quality ~ Clarity + Body + Flavor + Oakiness + Region + Clarity:Region +
```

```
##     Body:Region + Oakiness:Region
##
##                   Df Sum of Sq    RSS     AIC
## <none>                          15.934 -7.0261
## - Oakiness:Region  2    3.0051 18.939 -4.4608
## - Clarity:Region   2    4.2682 20.203 -2.0074
## - Body:Region      2    4.3696 20.304 -1.8172
## - Flavor           1    9.7000 25.634  9.0412
```

```
# Summary of the final model
summary(final_model)
```

```
##
## Call:
## lm(formula = Quality ~ Clarity + Body + Flavor + Oakiness + Region +
##     Clarity:Region + Body:Region + Oakiness:Region, data = wine_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.3326 -0.4815  0.1080  0.4933  1.4790
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)         4.8700     2.8883   1.686 0.104216
## Clarity             3.4196     2.3579   1.450 0.159404
## Body                0.4582     0.3012   1.521 0.140806
## Flavor              0.8956     0.2296   3.901 0.000638 ***
## Oakiness           -0.4782     0.2697  -1.773 0.088433 .
## Region2            -2.2465     4.3389  -0.518 0.609185
## Region3            19.8445     7.2111   2.752 0.010865 *
## Clarity:Region2    -6.4481     3.1614  -2.040 0.052085 .
## Clarity:Region3   -12.2933     5.2672  -2.334 0.027936 *
## Body:Region2        0.3336     0.6194   0.539 0.594950
## Body:Region3       -1.7208     0.7156  -2.405 0.023921 *
## Oakiness:Region2    1.1637     0.5368   2.168 0.039878 *
## Oakiness:Region3    0.4332     0.5744   0.754 0.457772
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7984 on 25 degrees of freedom
## Multiple R-squared:  0.8971, Adjusted R-squared:  0.8476
## F-statistic: 18.15 on 12 and 25 DF,  p-value: 2.012e-09
```

```
# Diagnostic plots for the final model
par(mfrow=c(2,2))
plot(final_model)
```




```
## Problem 1-f
# Calculate means of other predictors
mean_clarity <- mean(wine_data$Clarity)
mean_aroma <- mean(wine_data$Aroma)
mean_body <- mean(wine_data$Body)
mean_flavor <- mean(wine_data$Flavor)
mean_oakiness <- mean(wine_data$Oakiness)

# Create a new data frame for prediction
new_data <- data.frame(Clarity=mean_clarity, Aroma=mean_aroma, Body=mean_body, Flavor=mean_flavor, Oaki
# Predict Quality with 95% prediction interval
pred <- predict(final_model, newdata=new_data, interval="prediction", level=0.95)
pred
```

```
##        fit      lwr      upr
## 1 12.41075 10.70892 14.11258
```

```
# Predict Quality with 95% confidence interval for mean response
conf <- predict(final_model, newdata=new_data, interval="confidence", level=0.95)
conf
```

```
##        fit      lwr     upr
## 1 12.41075 11.97181 12.8497
```

Section 2 for Question 2

```
# Load the data
diabetes_data <- read.csv("/Users/springkim/Downloads/diabetes.csv", header=TRUE)

# Explore the dataset
summary(diabetes_data)
```

```
##  Pregnancies..      Glucose..     BloodPressure..  SkinThickness..
##  Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   :  0.00
##  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 63.50   1st Qu.:  0.00
##  Median : 3.000   Median :117.0   Median : 72.00   Median : 23.00
##  Mean   : 3.704   Mean   :121.2   Mean   : 69.15   Mean   : 20.93
##  3rd Qu.: 6.000   3rd Qu.:141.0   3rd Qu.: 80.00   3rd Qu.: 32.00
##  Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :110.00
##    Insulin..          BMI..       DiabetesPedigreeFunction..     Age..
##  Min.   :  0.00   Min.   : 0.00   Min.   :0.0780             Min.   :21.00
##  1st Qu.:  0.00   1st Qu.:27.38   1st Qu.:0.2440             1st Qu.:24.00
##  Median : 40.00   Median :32.30   Median :0.3760             Median :29.00
##  Mean   : 80.25   Mean   :32.19   Mean   :0.4709             Mean   :33.09
##  3rd Qu.:130.00   3rd Qu.:36.80   3rd Qu.:0.6240             3rd Qu.:40.00
##  Max.   :744.00   Max.   :80.60   Max.   :2.4200             Max.   :81.00
##     Outcome
##  Min.   :0.000
##  1st Qu.:0.000
##  Median :0.000
##  Mean   :0.342
##  3rd Qu.:1.000
##  Max.   :1.000
```

```
str(diabetes_data)
```

```
## 'data.frame':    2000 obs. of  9 variables:
##  $ Pregnancies..             : int  2 0 0 0 1 0 4 8 2 2 ...
##  $ Glucose..                 : int  138 84 145 135 139 173 99 194 83 89 ...
##  $ BloodPressure..           : int  62 82 0 68 62 78 72 80 65 90 ...
##  $ SkinThickness..           : int  35 31 0 42 41 32 17 0 28 30 ...
##  $ Insulin..                 : int  0 125 0 250 480 265 0 0 66 0 ...
##  $ BMI..                     : num  33.6 38.2 44.2 42.3 40.7 46.5 25.6 26.1 36.8 33.5 ...
##  $ DiabetesPedigreeFunction..: num  0.127 0.233 0.63 0.365 0.536 ...
##  $ Age..                     : int  47 23 31 24 21 58 28 67 24 42 ...
##  $ Outcome                   : int  1 0 1 1 0 0 0 0 0 0 ...
```

```
# Check for missing values
colSums(is.na(diabetes_data))
```

```
##              Pregnancies..                 Glucose..
##                          0                         0
##            BloodPressure..            SkinThickness..
##                          0                         0
##                  Insulin..                      BMI..
```

```
##                           0                           0
## DiabetesPedigreeFunction..                         Age..
##                           0                           0
##                     Outcome
##                           0
```

```r
# View first few rows
head(diabetes_data)
```

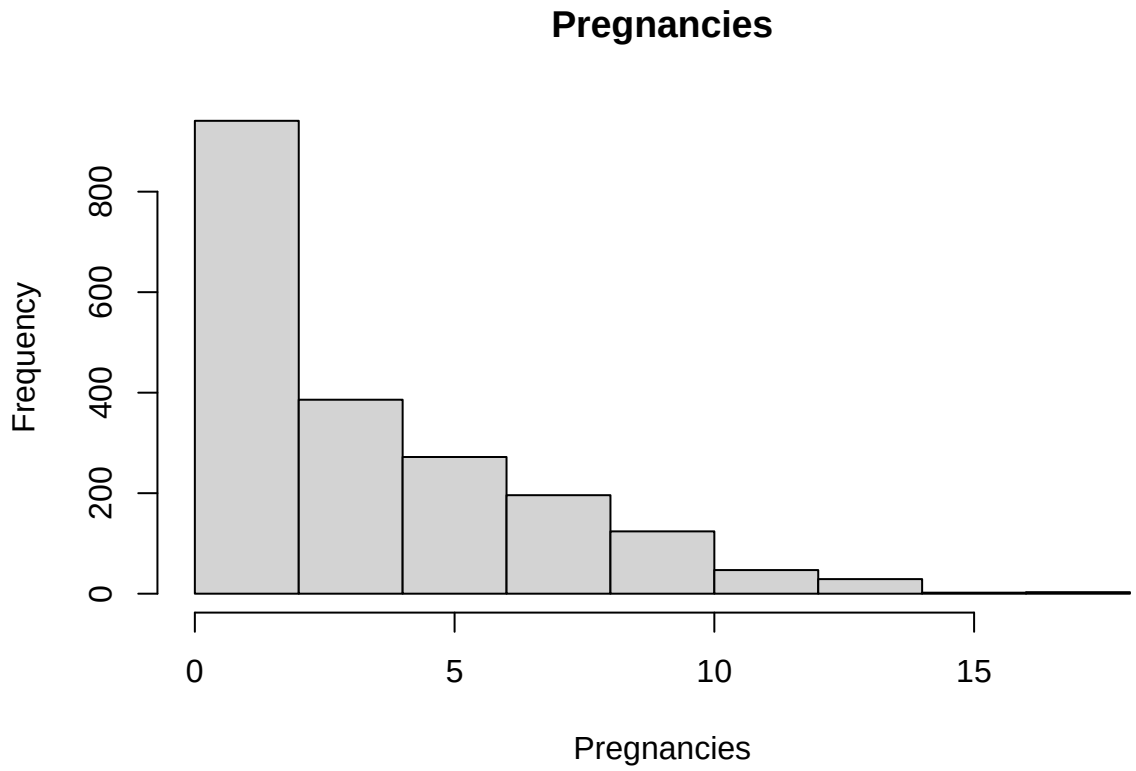
```
##   Pregnancies.. Glucose.. BloodPressure.. SkinThickness.. Insulin.. BMI..
## 1             2       138              62              35         0  33.6
## 2             0        84              82              31       125  38.2
## 3             0       145               0               0         0  44.2
## 4             0       135              68              42       250  42.3
## 5             1       139              62              41       480  40.7
## 6             0       173              78              32       265  46.5
##   DiabetesPedigreeFunction.. Age.. Outcome
## 1                      0.127    47       1
## 2                      0.233    23       0
## 3                      0.630    31       1
## 4                      0.365    24       1
## 5                      0.536    21       0
## 6                      1.159    58       0
```

```r
## 2-a
# Correlation matrix to assess relationships between numeric variables
cor(diabetes_data[, -9]) # Exclude 'Outcome' for correlation
```

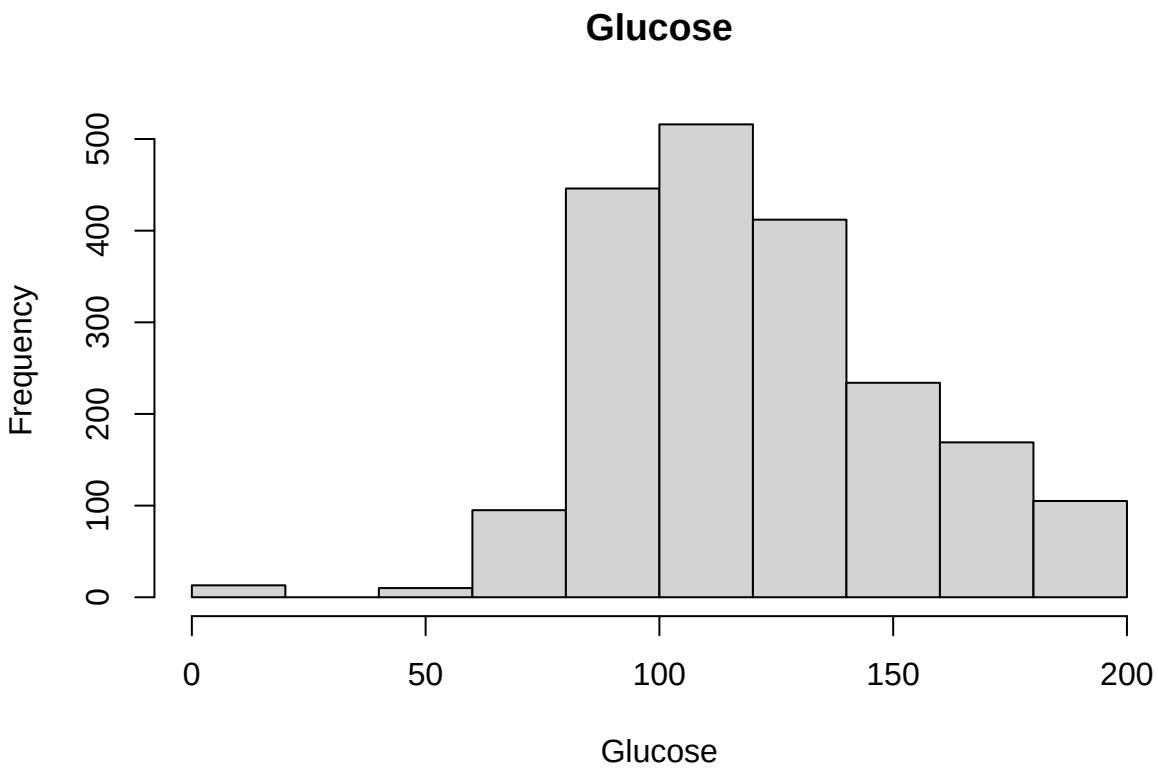
```
##                           Pregnancies..  Glucose.. BloodPressure..
## Pregnancies..                1.00000000 0.12040541      0.14967246
## Glucose..                    0.12040541 1.00000000      0.13804400
## BloodPressure..              0.14967246 0.13804400      1.00000000
## SkinThickness..             -0.06337462 0.06236813      0.19880047
## Insulin..                   -0.07659977 0.32037084      0.08738405
## BMI..                        0.01947503 0.22686443      0.28154513
## DiabetesPedigreeFunction..  -0.02545316 0.12324343      0.05133095
## Age..                        0.53945719 0.25449621      0.23837508
##                           SkinThickness..   Insulin..      BMI..
## Pregnancies..                -0.06337462 -0.07659977 0.01947503
## Glucose..                     0.06236813  0.32037084 0.22686443
## BloodPressure..               0.19880047  0.08738405 0.28154513
## SkinThickness..               1.00000000  0.44885895 0.39376029
## Insulin..                     0.44885895  1.00000000 0.22301161
## BMI..                         0.39376029  0.22301161 1.00000000
## DiabetesPedigreeFunction..    0.17829888  0.19271873 0.12571935
## Age..                        -0.11103369 -0.08587910 0.03898737
##                           DiabetesPedigreeFunction..       Age..
## Pregnancies..                            -0.02545316  0.53945719
## Glucose..                                 0.12324343  0.25449621
## BloodPressure..                           0.05133095  0.23837508
## SkinThickness..                           0.17829888 -0.11103369
## Insulin..                                 0.19271873 -0.08587910
## BMI..                                     0.12571935  0.03898737
```

```
## DiabetesPedigreeFunction..                1.00000000  0.02656950
## Age..                                     0.02656950  1.00000000
```

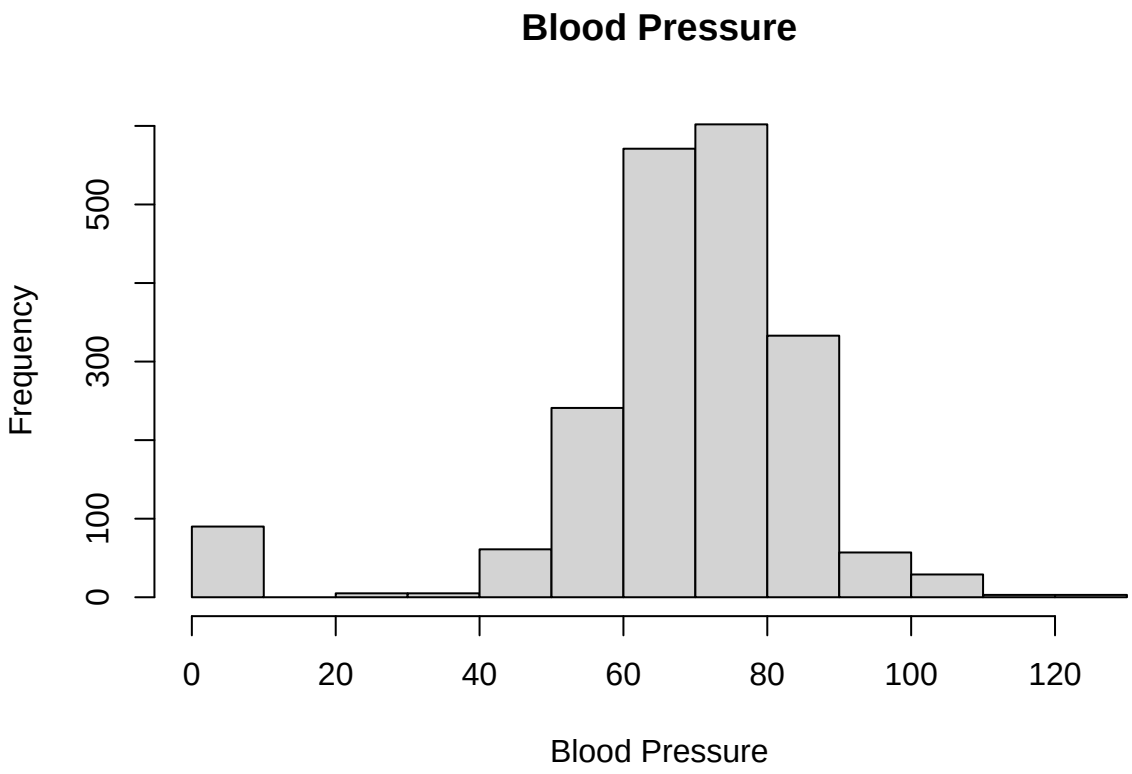
```r
# Histograms to check the distribution of each variable
hist(diabetes_data$Pregnancies, main="Pregnancies", xlab="Pregnancies")
```



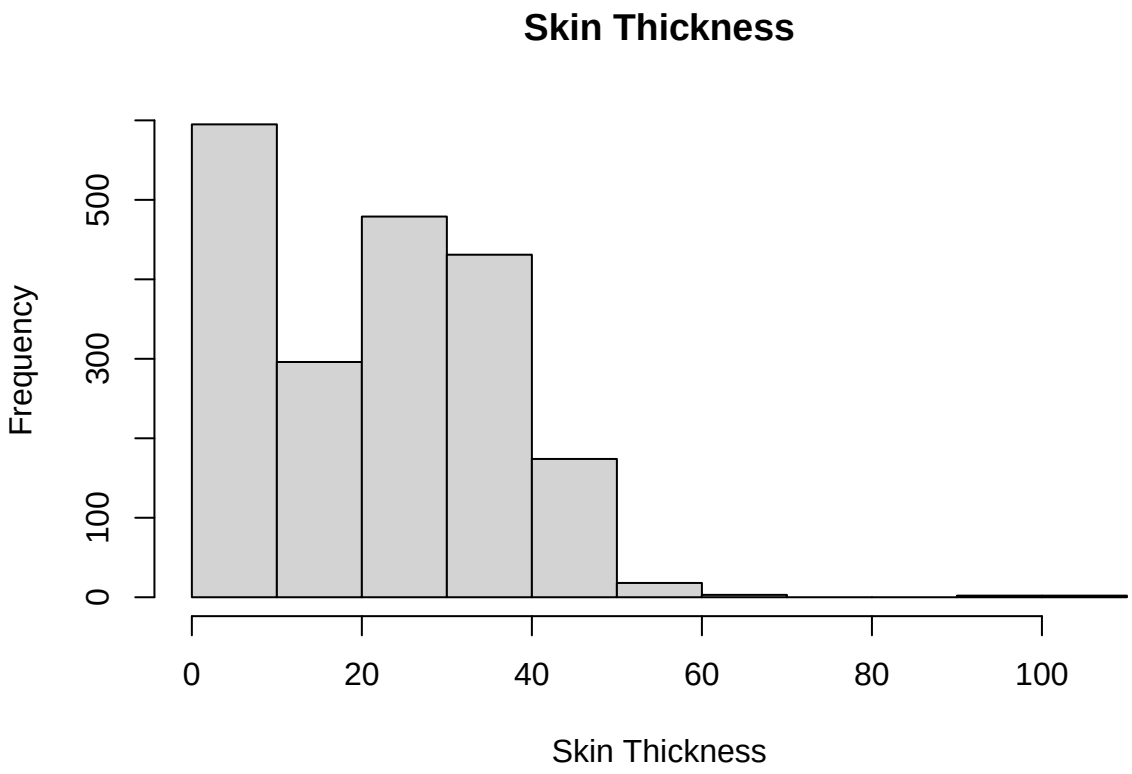

```r
hist(diabetes_data$Glucose, main="Glucose", xlab="Glucose")
```



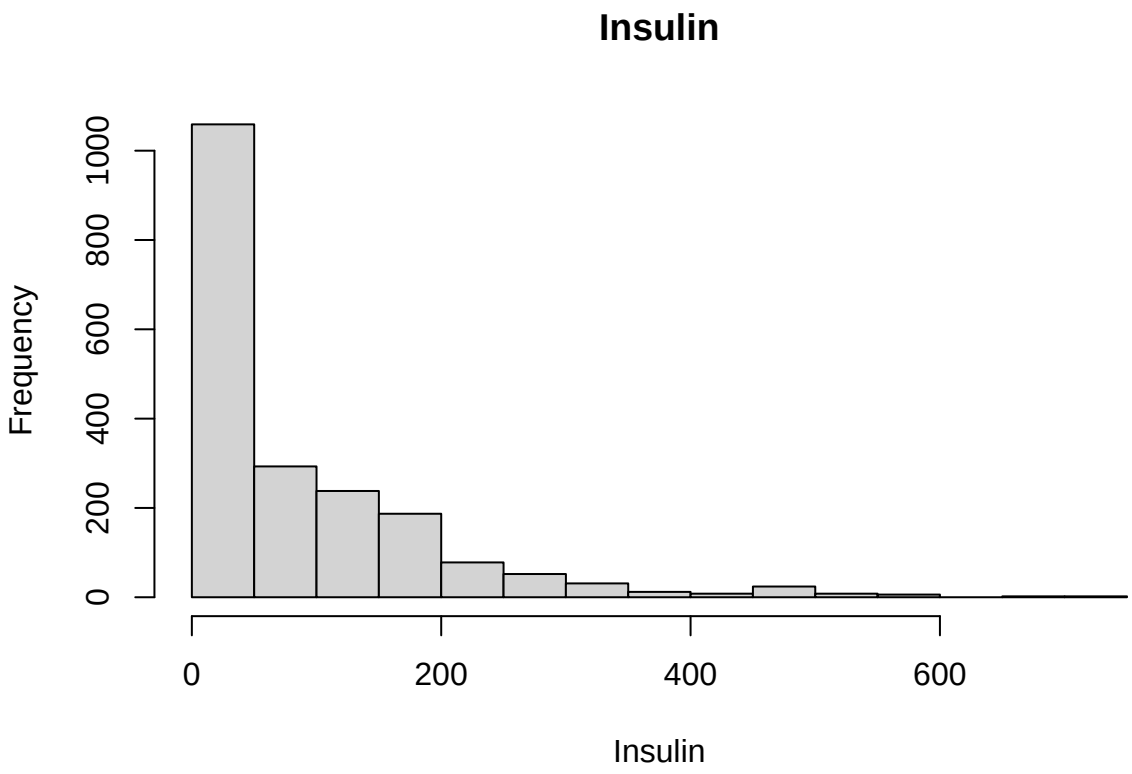

```r
hist(diabetes_data$BloodPressure, main="Blood Pressure", xlab="Blood Pressure")
```



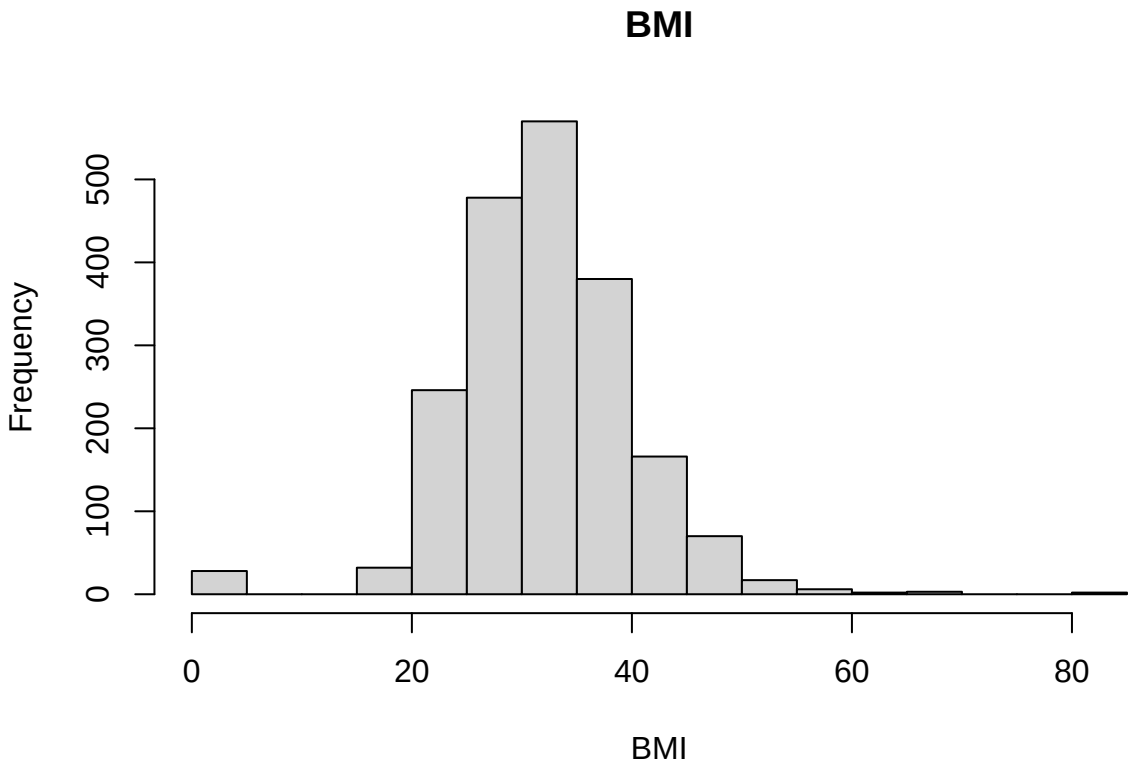

```
hist(diabetes_data$SkinThickness, main="Skin Thickness", xlab="Skin Thickness")
```



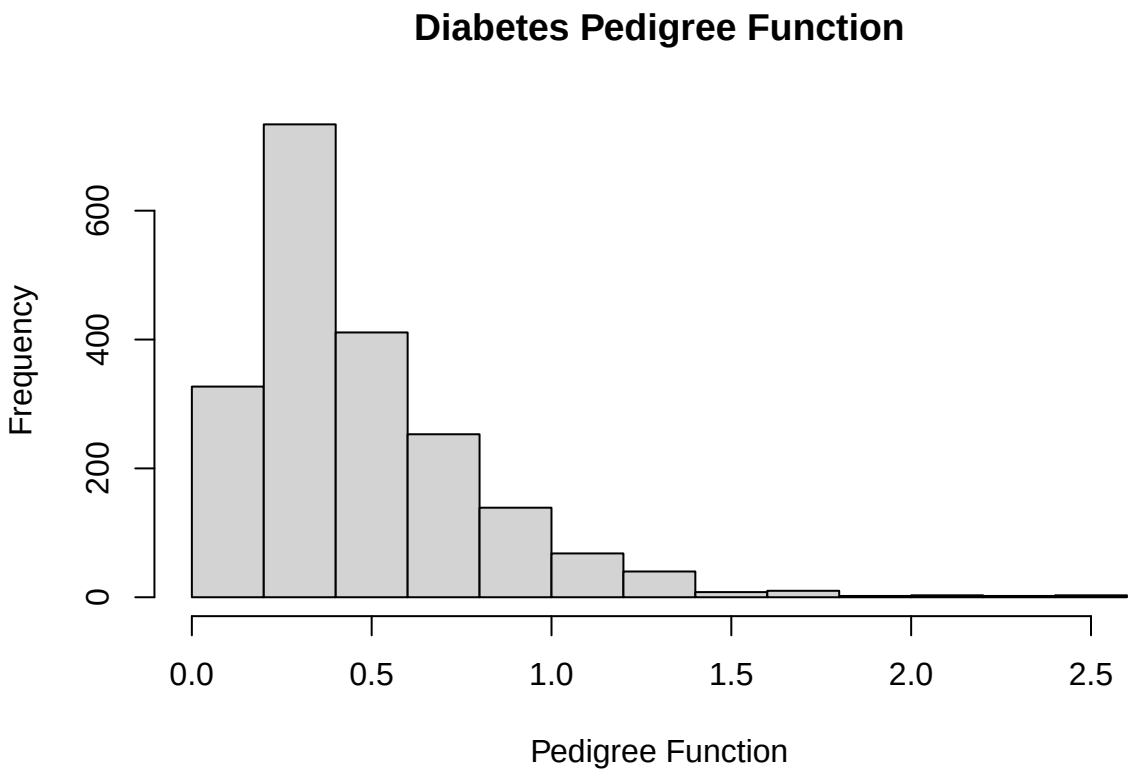

```
hist(diabetes_data$Insulin, main="Insulin", xlab="Insulin")
```



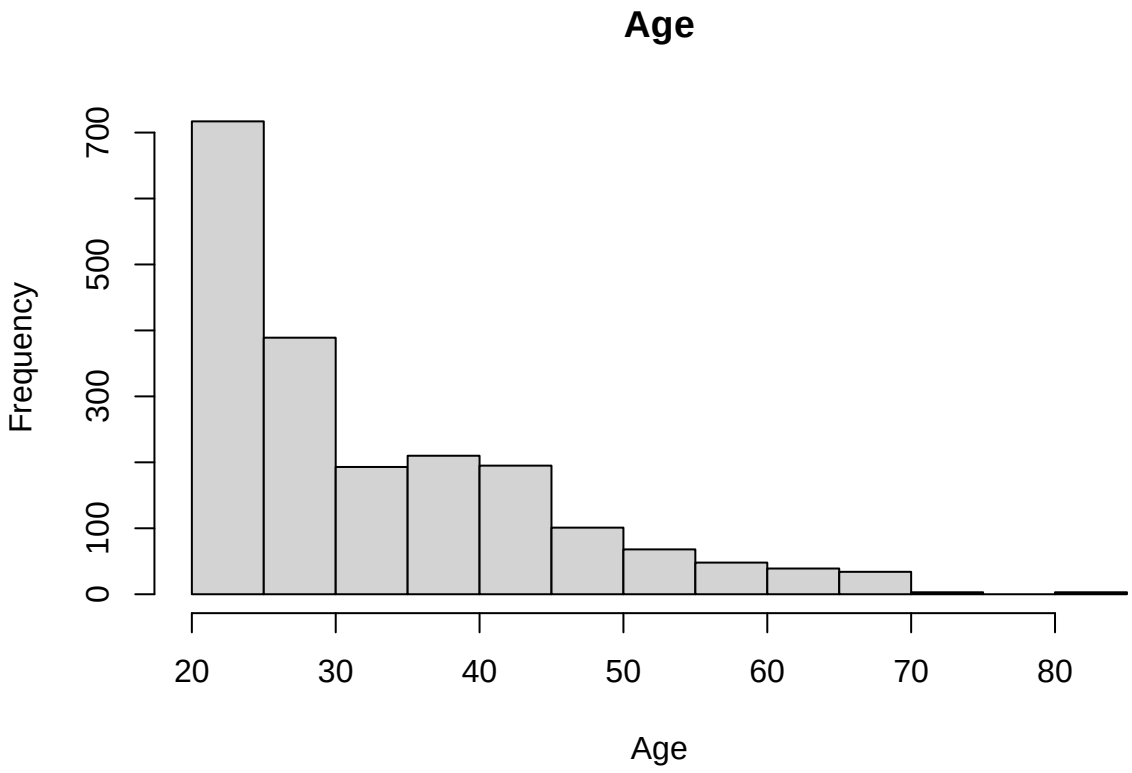

```r
hist(diabetes_data$BMI, main="BMI", xlab="BMI")
```



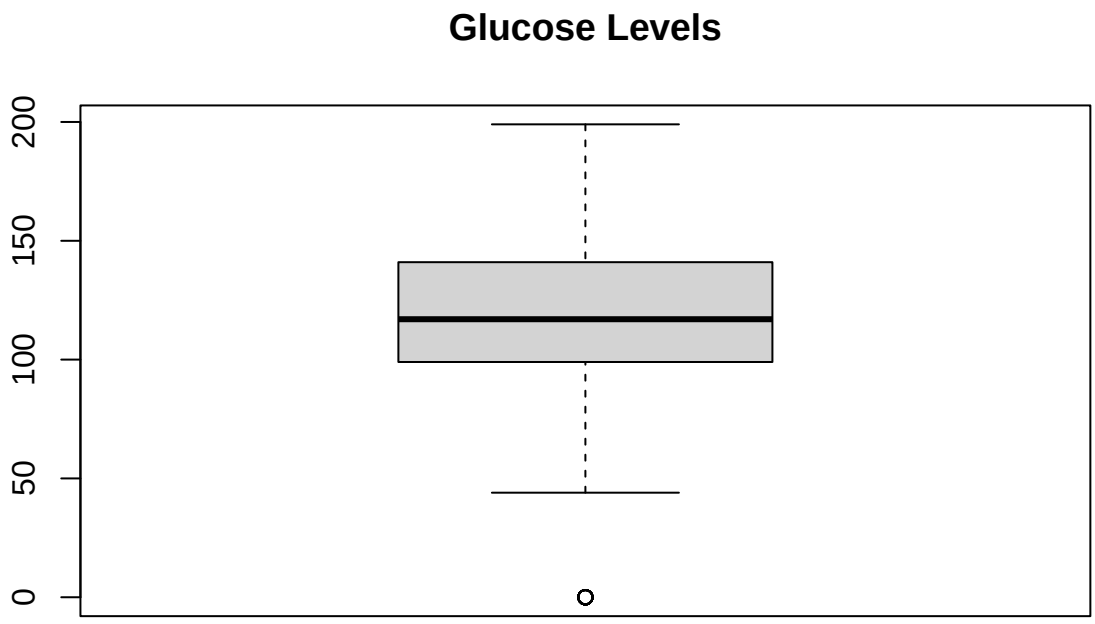

```
hist(diabetes_data$DiabetesPedigreeFunction, main="Diabetes Pedigree Function", xlab="Pedigree Function
```



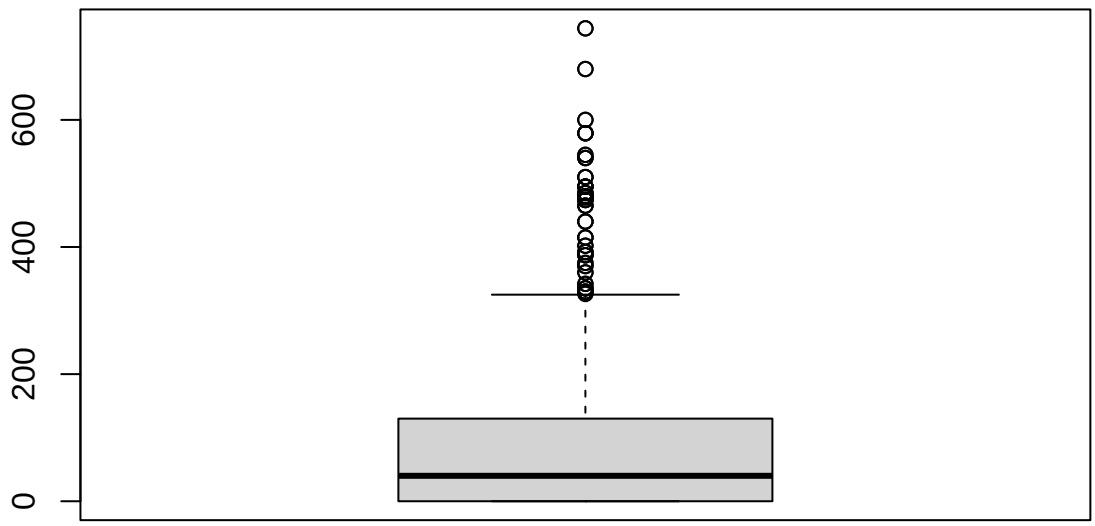

```
hist(diabetes_data$Age, main="Age", xlab="Age")
```




```
# Boxplots to check for outliers
boxplot(diabetes_data$Glucose, main="Glucose Levels")
```



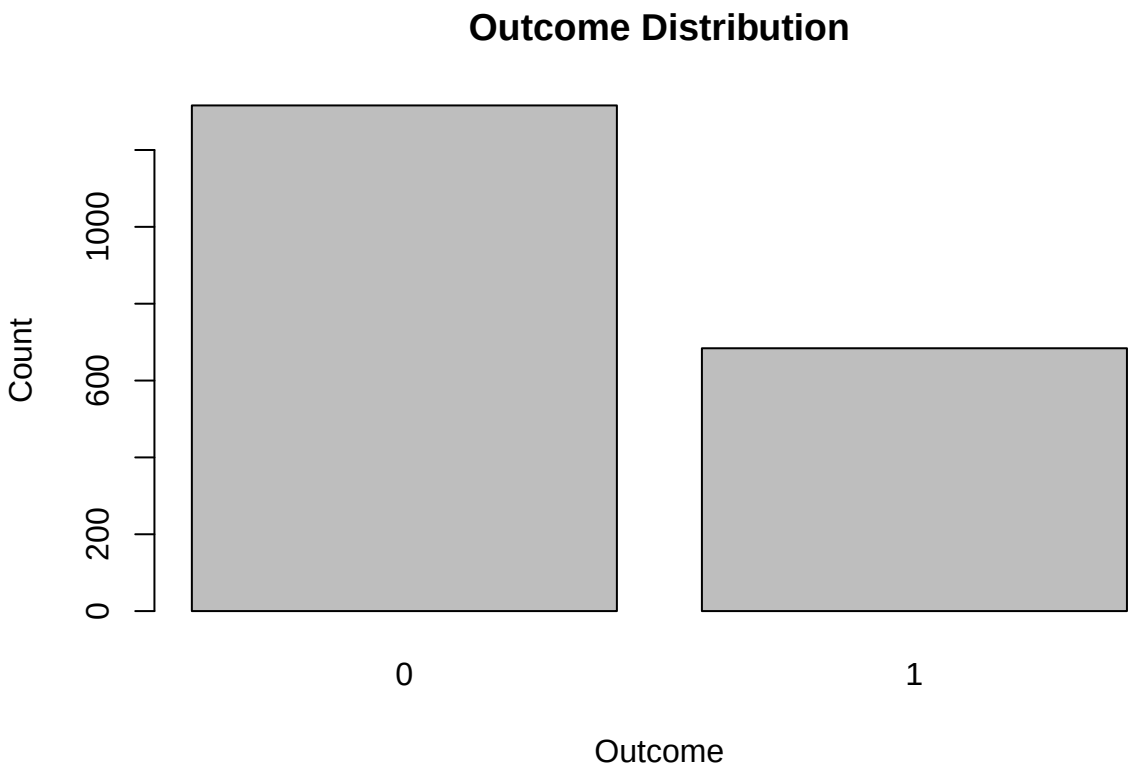

```r
boxplot(diabetes_data$Insulin, main="Insulin Levels")
```




```
# Plot Outcome distribution
table(diabetes_data$Outcome)
```

```
##
##    0    1
## 1316  684
```

```
barplot(table(diabetes_data$Outcome), main="Outcome Distribution", xlab="Outcome", ylab="Count")
```




```
## 2-b
## 1. Perform LDA
# Load the MASS library for LDA
library(MASS)

# Perform LDA
lda_model <- lda(Outcome ~ ., data=diabetes_data)

# View the LDA model
lda_model
```

```
## Call:
## lda(Outcome ~ ., data = diabetes_data)
##
## Prior probabilities of groups:
##     0     1
## 0.658 0.342
##
## Group means:
##   Pregnancies.. Glucose.. BloodPressure.. SkinThickness.. Insulin..    BMI..
## 0      3.168693  110.5866        68.09498        20.05243  70.56383 30.56748
## 1      4.732456  141.5687        71.16667        22.63304  98.89766 35.32047
##   DiabetesPedigreeFunction..    Age..
## 0                  0.4346763 31.08131
## 1                  0.5406813 36.95614
##
```

```
## Coefficients of linear discriminants:
##                                       LD1
## Pregnancies..                0.1015317728
## Glucose..                    0.0271248841
## BloodPressure..             -0.0084340627
## SkinThickness..              0.0012318356
## Insulin..                   -0.0009708906
## BMI..                        0.0537005742
## DiabetesPedigreeFunction..   0.6549925988
## Age..                        0.0109349308
```

```
## 2. Make Prediction
# Predict using the LDA model
lda_pred <- predict(lda_model)

# Posterior probabilities
lda_prob <- lda_pred$posterior[,2]

# Predictions based on 0.5 cutoff
lda_class <- ifelse(lda_prob > 0.5, 1, 0)

# Create confusion matrix
table(Predicted = lda_class, Actual = diabetes_data$Outcome)
```

```
##           Actual
## Predicted    0    1
##         0 1174  298
##         1  142  386
```

```
## 3. Compute Confusion Matrix, Sensitivity, Specificity, and Misclassification Rate
# Confusion matrix
conf_matrix <- table(Predicted = lda_class, Actual = diabetes_data$Outcome)

# Sensitivity (True Positive Rate)
sensitivity <- conf_matrix[2,2] / (conf_matrix[2,2] + conf_matrix[1,2])

# Specificity (True Negative Rate)
specificity <- conf_matrix[1,1] / (conf_matrix[1,1] + conf_matrix[2,1])

# Overall misclassification rate
misclass_rate <- (conf_matrix[1,2] + conf_matrix[2,1]) / sum(conf_matrix)

# Output the metrics
sensitivity
```

```
## [1] 0.5643275
```

```
specificity
```

```
## [1] 0.8920973
```

```
misclass_rate
```

```
## [1] 0.22
```

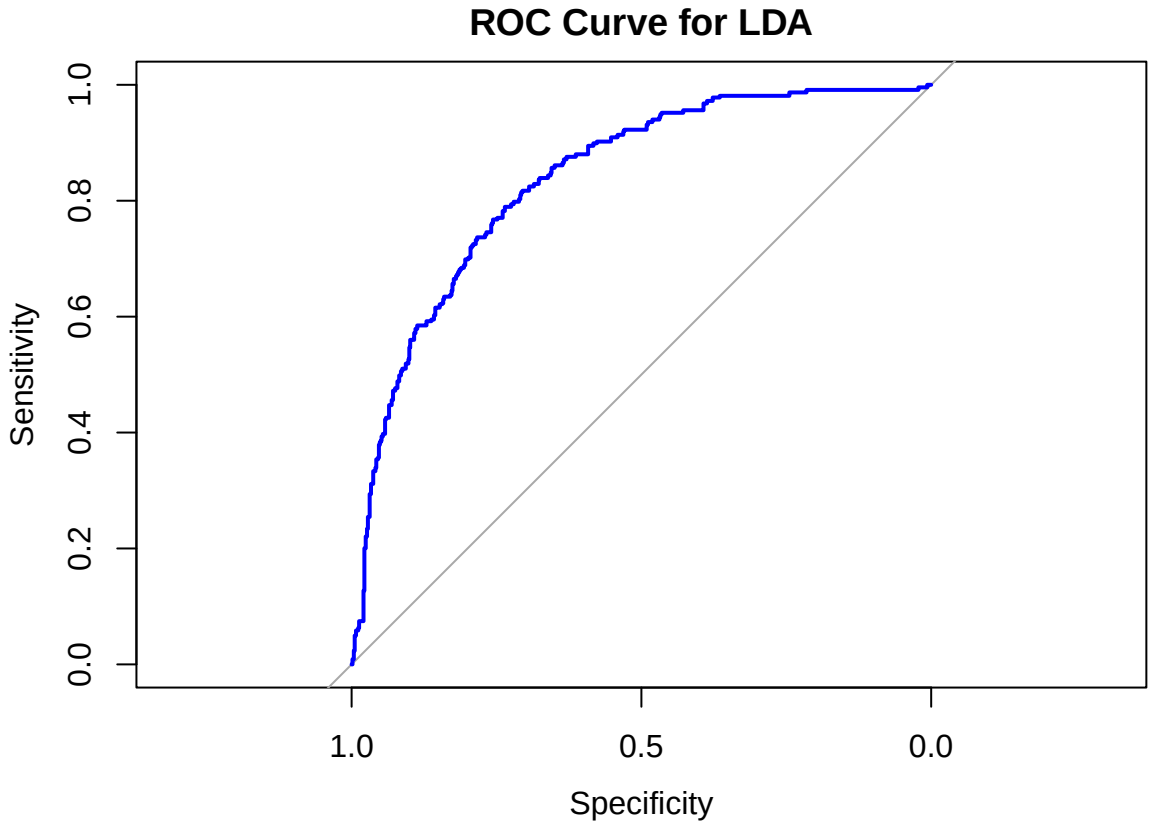
```
## 4. Plot the ROC Curve:
# Load the pROC package to plot the ROC curve
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
# Generate ROC curve
lda_roc <- roc(diabetes_data$Outcome, lda_prob)
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```

```
# Plot ROC curve
plot(lda_roc, col="blue", main="ROC Curve for LDA")
```



```r
## 2-c
## 1. Perform QDA
# Perform QDA
qda_model <- qda(Outcome ~ ., data=diabetes_data)

# View the QDA model
qda_model
```

```
## Call:
## qda(Outcome ~ ., data = diabetes_data)
##
## Prior probabilities of groups:
##     0     1
## 0.658 0.342
##
## Group means:
##   Pregnancies.. Glucose.. BloodPressure.. SkinThickness.. Insulin..    BMI..
## 0      3.168693  110.5866        68.09498        20.05243  70.56383 30.56748
## 1      4.732456  141.5687        71.16667        22.63304  98.89766 35.32047
##   DiabetesPedigreeFunction..    Age..
## 0                   0.4346763 31.08131
## 1                   0.5406813 36.95614
```

```r
## 2. Make Predictions and Evaluate
# Predict using the QDA model
qda_pred <- predict(qda_model)

# Posterior probabilities
qda_prob <- qda_pred$posterior[,2]

# Predictions based on 0.5 cutoff
qda_class <- ifelse(qda_prob > 0.5, 1, 0)

# Create confusion matrix
table(Predicted = qda_class, Actual = diabetes_data$Outcome)
```

```
##          Actual
## Predicted    0    1
##         0 1135  290
##         1  181  394
```

```r
## 3. Compute Confusion Matrix, Sensitivity, Specificity, and Misclassification Rate:
# Confusion matrix
conf_matrix_qda <- table(Predicted = qda_class, Actual = diabetes_data$Outcome)

# Sensitivity (True Positive Rate)
sensitivity_qda <- conf_matrix_qda[2,2] / (conf_matrix_qda[2,2] + conf_matrix_qda[1,2])

# Specificity (True Negative Rate)
specificity_qda <- conf_matrix_qda[1,1] / (conf_matrix_qda[1,1] + conf_matrix_qda[2,1])

# Overall misclassification rate
```

```r
misclass_rate_qda <- (conf_matrix_qda[1,2] + conf_matrix_qda[2,1]) / sum(conf_matrix_qda)

# Output the metrics
sensitivity_qda
```

```
## [1] 0.5760234
```

```r
specificity_qda
```

```
## [1] 0.862462
```

```r
misclass_rate_qda
```

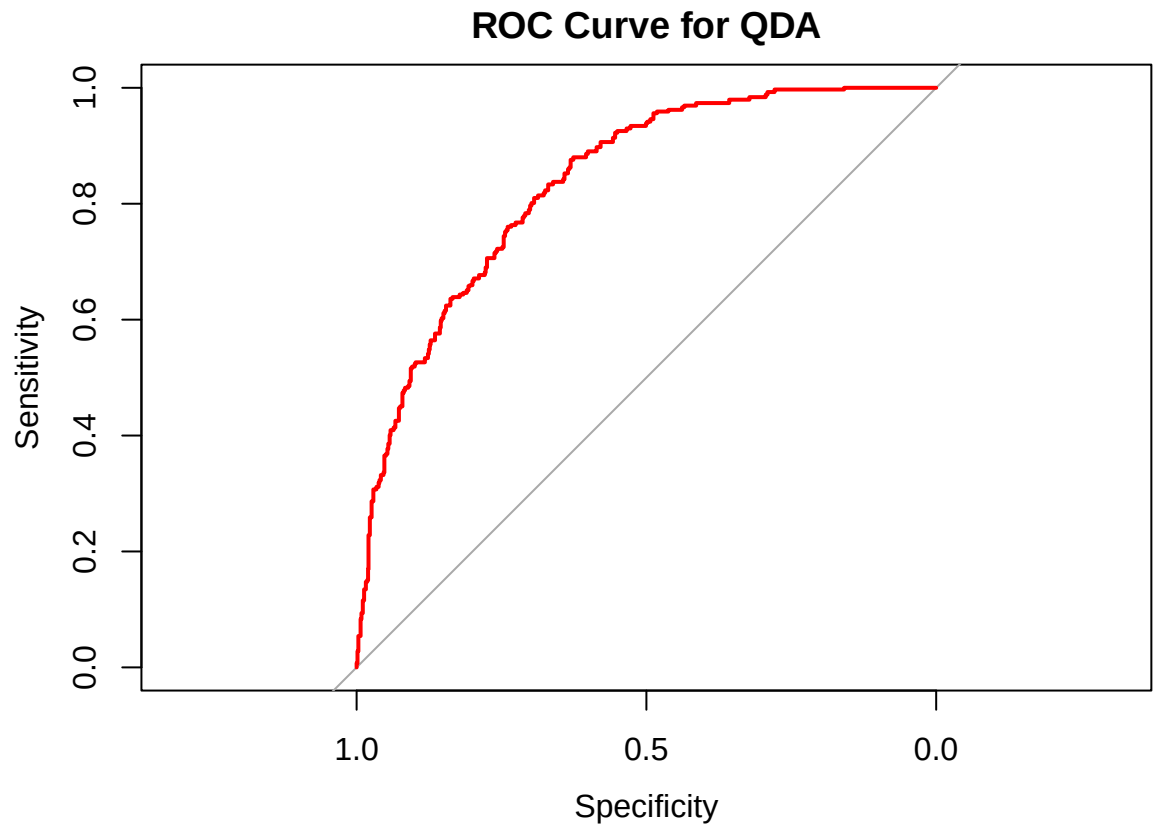
```
## [1] 0.2355
```

```r
## 4. Plot the ROC Curve for QDA:
# Generate ROC curve for QDA
qda_roc <- roc(diabetes_data$Outcome, qda_prob)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
# Plot ROC curve
plot(qda_roc, col="red", main="ROC Curve for QDA")
```



```r
## 2-d
# Optimal cutoff based on ROC curve
optimal_cutoff <- coords(lda_roc, "best", ret="threshold")
optimal_cutoff
```

```
##   threshold
## 1 0.2906709
```

```r
## Question 3
# 3-a
# Load required package
library(MASS)  # for multivariate normal sampling

# Problem parameters
p <- 10
sigma <- 1
mu <- rep(1, p)  # vector of all 1's
N <- 1000  # number of observations

# Function to compute the James-Stein estimator
JS_estimator <- function(Y, p, sigma) {
  norm_Y_sq <- sum(Y^2)  # squared L2 norm of Y
  shrinkage_factor <- 1 - (p - 2) * sigma^2 / norm_Y_sq
  return(shrinkage_factor * Y)
}

# Initialize matrices to store estimates
JS_estimates <- matrix(0, nrow=N, ncol=p)
MLE_estimates <- matrix(0, nrow=N, ncol=p)

# Simulate N observations and compute JS and MLE estimates
set.seed(123)  # for reproducibility
for (i in 1:N) {
  Y_i <- mvrnorm(1, mu=mu, Sigma=sigma^2 * diag(p))  # generate Y_i
  JS_estimates[i, ] <- JS_estimator(Y_i, p, sigma)  # compute JS estimate
  MLE_estimates[i, ] <- Y_i  # MLE is just the observation itself
}

# Compute empirical bias
bias_JS <- norm(colMeans(JS_estimates) - mu, type="2")
bias_MLE <- norm(colMeans(MLE_estimates) - mu, type="2")

# Compute empirical risk
risk_JS <- mean(apply(JS_estimates, 1, function(est) norm(est - mu, type="2")^2))
risk_MLE <- mean(apply(MLE_estimates, 1, function(est) norm(est - mu, type="2")^2))

# Print results
cat("Bias of James-Stein estimator:", bias_JS, "\n")
```

```
## Bias of James-Stein estimator: 1.338441
```

```r
cat("Bias of MLE estimator:", bias_MLE, "\n")
```

```
## Bias of MLE estimator: 0.09435853
```

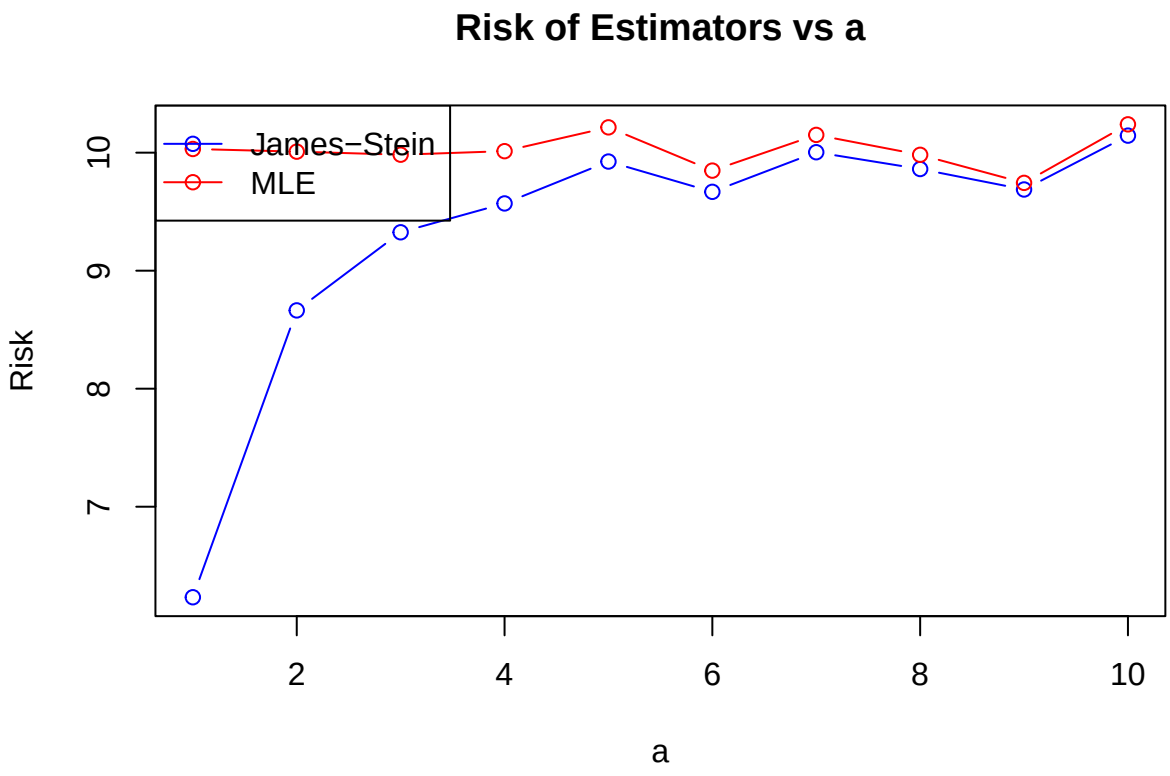
```r
cat("Risk of James-Stein estimator:", risk_JS, "\n")
```

```
## Risk of James-Stein estimator: 6.135585
```

```r
cat("Risk of MLE estimator:", risk_MLE, "\n")
```

```
## Risk of MLE estimator: 9.97181
```

```r
# 3-b
# Varying mu as a * [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
a_values <- 1:10
risk_JS_a <- numeric(length(a_values))
risk_MLE_a <- numeric(length(a_values))

for (j in 1:length(a_values)) {
  a <- a_values[j]
  mu_a <- a * mu  # new mu

  # Recompute JS and MLE estimates
  JS_estimates_a <- matrix(0, nrow=N, ncol=p)
  MLE_estimates_a <- matrix(0, nrow=N, ncol=p)

  for (i in 1:N) {
    Y_i <- mvrnorm(1, mu=mu_a, Sigma=sigma^2 * diag(p))
    JS_estimates_a[i, ] <- JS_estimator(Y_i, p, sigma)
    MLE_estimates_a[i, ] <- Y_i
  }

  # Compute risk for both estimators
  risk_JS_a[j] <- mean(apply(JS_estimates_a, 1, function(est) norm(est - mu_a, type="2")^2))
  risk_MLE_a[j] <- mean(apply(MLE_estimates_a, 1, function(est) norm(est - mu_a, type="2")^2))
}

# Plot the risk vs a
plot(a_values, risk_JS_a, type="b", col="blue", ylim=range(c(risk_JS_a, risk_MLE_a)), ylab="Risk", xlab=
lines(a_values, risk_MLE_a, type="b", col="red")
legend("topleft", legend=c("James-Stein", "MLE"), col=c("blue", "red"), lty=1, pch=1)
```




```r
# 3-c
# Varying sigma
sigma_values <- c(0.1, 0.5, 2, 5, 10)
risk_JS_sigma <- numeric(length(sigma_values))
risk_MLE_sigma <- numeric(length(sigma_values))

for (j in 1:length(sigma_values)) {
  sigma <- sigma_values[j]  # new sigma

  # Recompute JS and MLE estimates
  JS_estimates_sigma <- matrix(0, nrow=N, ncol=p)
  MLE_estimates_sigma <- matrix(0, nrow=N, ncol=p)

  for (i in 1:N) {
    Y_i <- mvrnorm(1, mu=mu, Sigma=sigma^2 * diag(p))
    JS_estimates_sigma[i, ] <- JS_estimator(Y_i, p, sigma)
    MLE_estimates_sigma[i, ] <- Y_i
  }

  # Compute risk for both estimators
  risk_JS_sigma[j] <- mean(apply(JS_estimates_sigma, 1, function(est) norm(est - mu, type="2")^2))
  risk_MLE_sigma[j] <- mean(apply(MLE_estimates_sigma, 1, function(est) norm(est - mu, type="2")^2))
}

# Plot the risk vs sigma
plot(sigma_values, risk_JS_sigma, type="b", col="blue", ylim=range(c(risk_JS_sigma, risk_MLE_sigma)), yl
```

```
lines(sigma_values, risk_MLE_sigma, type="b", col="red")
legend("topleft", legend=c("James-Stein", "MLE"), col=c("blue", "red"), lty=1, pch=1)
```