R ASSIGNMENT 1

NWUDO CHIKAEZE

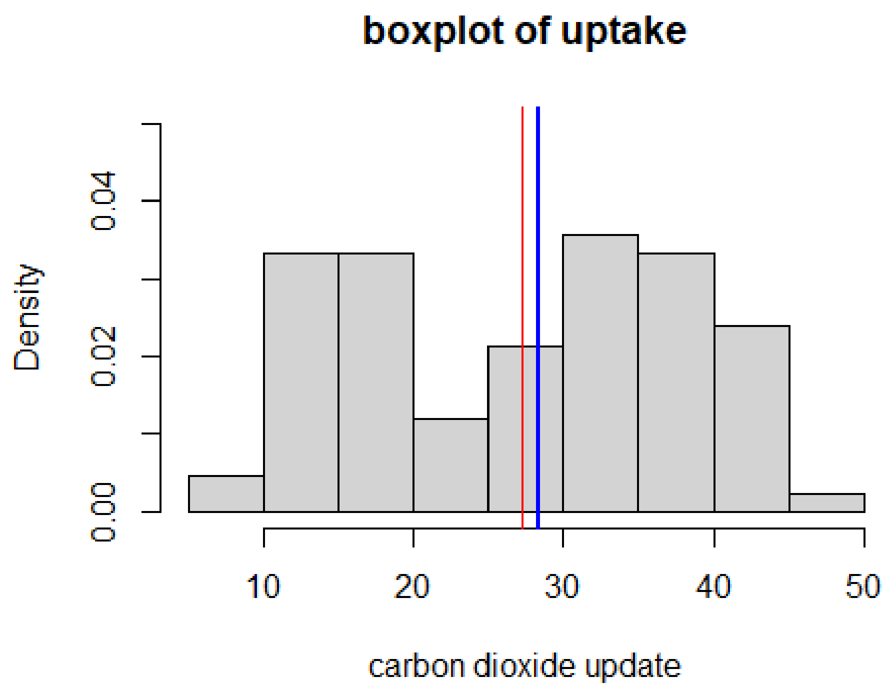
2022-10-02

Question 1

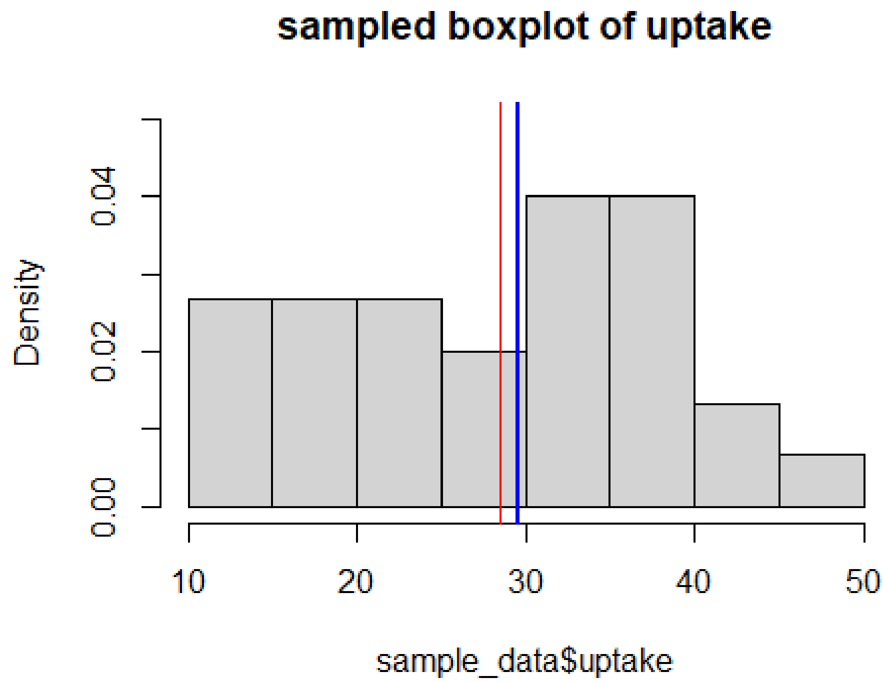
The CO2 data set shows the CO2 concentrations of six plants from Quebec and six plants from Mississippi. Hint: The CO2 data set is available in package datasets. # install.packages("datasets") # library(datasets) head(CO2)

a) Plot the density histogram of the 'uptake' variable and show the mean and median values (with different colours) on the histogram. Lable the x-axis by 'Carbon Dioxide Update'.
b) Take a sample of size n = 30 from the 'uptake' variable. Plot the density histogram of the sampled 'uptake' values and show the sample mean and median (with the same colours as (a)) on the histogram.
c) Does the sample in part (b) represent well the uptake population?
d) Show both the histograms of parts (a) and (b) side by side as the left and right panels of one figure.

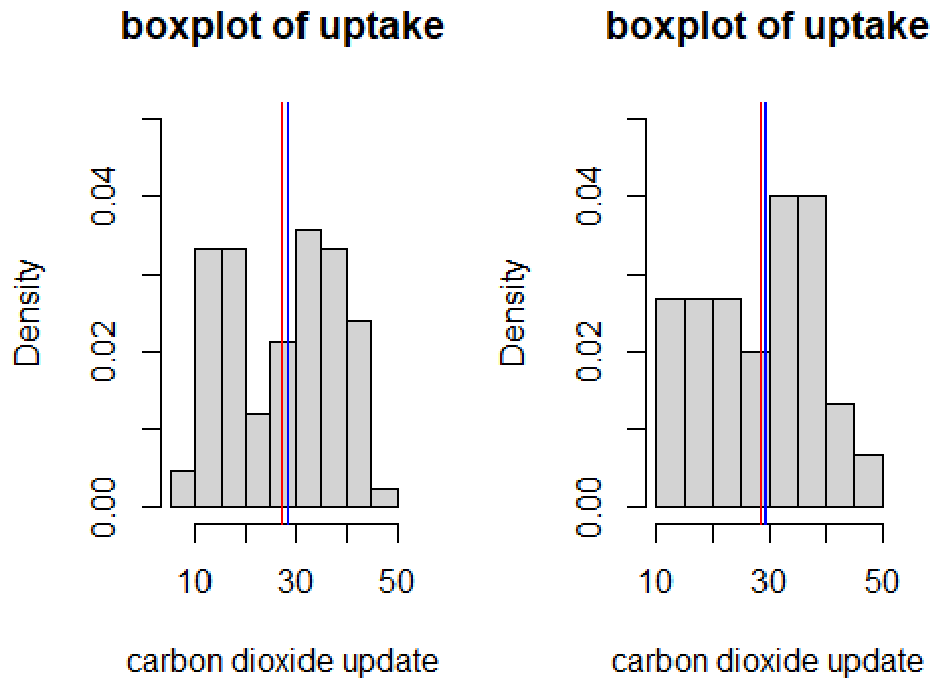
```
#density histogram of uptake variable

library(datasets)
mydata<-CO2
mean(mydata$uptake)

## [1] 27.2131

median(mydata$uptake)

## [1] 28.3

hist(mydata$uptake,freq = FALSE, ylim = c(0,0.05), main="boxplot of
uptake",xlab='carbon dioxide update')
abline(v=mean(mydata$uptake),col='red')
abline(v=median(mydata$uptake), col= "blue", lwd=2)
```




```r
#density histogram of sampled uptake  values

set.seed(1234)
sample_data<-mydata[sample(nrow(mydata), 30), ]
mean(sample_data$uptake)

## [1] 28.50667

median(sample_data$uptake)

## [1] 29.5

hist(sample_data$uptake,freq = FALSE, ylim = c(0,0.05), main = "sampled
boxplot of uptake")
abline(v=mean(sample_data$uptake),col='red')
abline(v=median(sample_data$uptake),col='blue', lwd=2)
```




```
#Histogram side by side?

par(mfrow = c(1,2))
hist(mydata$uptake,freq = FALSE, ylim = c(0,0.05), main="boxplot of
uptake",xlab='carbon dioxide update')
abline(v=mean(mydata$uptake),col='red')
abline(v=median(mydata$uptake), col= "blue", lwd=1)
hist(sample_data$uptake, breaks = 8, freq = FALSE, ylim =
c(0,0.05),main="boxplot of uptake",xlab='carbon dioxide update')
abline(v=mean(sample_data$uptake),col='red')
abline(v=median(sample_data$uptake),col='blue', lwd=1)
```



Question 2

Recall the variable letters (it is already defined in R base).

a) Generate a random vector Z of 1000 letters (from 'a' to 'z'). Print a summary of Z in the form of a frequncy table.
b) Print the list of letters that appear on even number of times in Z.

```
letters

##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q"
"r" "s"
## [20] "t" "u" "v" "w" "x" "y" "z"

Z<-sample(letters[1:26], size = 1000, replace = TRUE)
my_tab<-table(Z)
class(my_tab)

## [1] "table"

t<-as.data.frame(my_tab)
t
```

```
##    Z Freq
## 1  a   43
## 2  b   38
## 3  c   37
## 4  d   34
## 5  e   36
## 6  f   45
## 7  g   40
## 8  h   36
## 9  i   34
## 10 j   38
## 11 k   46
## 12 l   23
## 13 m   36
## 14 n   30
## 15 o   39
## 16 p   40
## 17 q   39
## 18 r   30
## 19 s   53
## 20 t   33
## 21 u   41
## 22 v   59
## 23 w   40
## 24 x   37
## 25 y   38
## 26 z   35
```

```
class(t)
```

```
## [1] "data.frame"
```

```
subset(t,(Freq%%2)==0)
```

```
##    Z Freq
## 2  b   38
## 4  d   34
## 5  e   36
## 7  g   40
## 8  h   36
## 9  i   34
## 10 j   38
## 11 k   46
## 13 m   36
## 14 n   30
## 16 p   40
## 18 r   30
## 23 w   40
## 25 y   38
```

Question 3

Generate a random vector Z2 of 1000 random numbers between 15 to 153. Find the Z2 values that are divisible to 3 and store them in Z3. Print a summary of Z3 in the form of a frequency table.

```
set.seed(12345)
Z2 <- sample(x=15:153,size = 1000, replace = T)
Z2

##    [1]  65  72 107  89 110  16 100  89  52 117 108  24  54  52  44  15  86
26
##   [19]  17 151  28 120 133  30  94  76 112 137  74 119  46  39  50 151 147
141
##   [37]  72 120  27  81  88  70 121 135 105 130  48 123  69  37 109 149  60
104
##   [55] 138 113  19 121  82 135  25  63  81  88  29  21  56 104  69 146 146
123
##   [73]  46  71 100  69  17  32 149 109 112  82 148 134  37 125  82  44  45
113
##   [91]  39  90 106 114  22  92  99  35  61  25  65  85  77  76 119 124  70
51
##  [109]  25 151  24  72  91 112  96 143 119  96 107  27  93  76 133 136  28
45
##  [127] 150 125 100  74  76  80  15 127 100  52 102  70 120 152  96 137 108
137
##  [145]  95  18 136  48 114 124  77  81  73  59 109  54 150  38 151  29  80
34
##  [163]  73 110  30  45 139  80  97  86  35 146 148 125  41  79 141  92 149
106
##  [181]  31  47 109 149  68  29 149  87  72 140 118 103  89  99 100  72  84
137
##  [199]  97  38  38  21  91  94  53 100  67  23 147  66  86  82  70 132  16
51
##  [217]  24 126 106  80  97 128  33  30  38 152 127  55 107  58 106 129 100
23
##  [235] 121  73 135  87  73  65  71  24  20  46  63  37 152 120  77  76 150
149
##  [253]  15 127 105  79  26  48  41  19  55  86  74 127  67  26  55 131  49
17
##  [271] 115  71 122  94  55  75  48  39  30  73 123 125 149  97  77 134  51
81
##  [289] 137 132  99  87 124 136 103 149 145  29 114  46 112  48  15  21 100
127
##  [307] 126 151  88  66  15  72  24  82  49 118  39  17  15  53 123  27 147
47
##  [325]  22 121  87 108 129  98  15  26  40  40  95  68  37  71  87 104  22
125
##  [343]  18 122 117 111  73 116  42  51 117  47  73  77  44  71  48 131 136
```

```
39
##  [361] 139 131  15 126 131  55  84  32 128 108 130  80  85 153  94  86  28
100
##  [379]  81  41  62 118 141  54  54 150  78  60  83  36 133 143  93  33  26
153
##  [397]  21  62  98 116  19  50 121  37  71  94  19  99 130 106  26  27 153
62
##  [415]  42  58  18  99 145 135  96  56  61  97  65 139 149  72  28  55 150
71
##  [433]  61 100 144  67  73 104 143  70 103 123 103 126 107 123  97 140 134
30
##  [451]  50  17 130 141  90  73 147 125  49  16  90 117  96  83  45  63  29
49
##  [469]  41 103  99  73  78  91  58  94 149 104  47  72  77 152  38 132  18
129
##  [487]  82  55  38 137  97 115  84  40  43  64 127  22 115  70 110  33  52
84
##  [505] 115 145 138 109  73  85  82  86  59 108  75 146  67 144  15  97  76
21
##  [523]  21  21  45  67  46  23 142  83  39  93  21  87 125  86 113  56 134
37
##  [541] 110  76  47 104 146  39  16 142  24 153 102  82  26  20  44 125  79
100
##  [559]  20  77  27 102  49  65  79  71  48 128  96  93  68  25  18  84  46
50
##  [577] 131  78  77  20  20  89  76  76 107  95  78  93 113  44  27 123  31
146
##  [595] 122  32  60  54 104 135  91 147  69  57  48  98 119 113  85 108 126
99
##  [613]  71 143  49  90  19 153 122 146  67 100  30  24  21  77  82 129  79
113
##  [631]  46  40 147  72  92 129 105 130 108 129  40 105 152  54 124 118 147
106
##  [649]  17  71  40  99 115  52  36  35  23  22  94  73  19 129  61 107 133
107
##  [667] 140  23  99  69  99  58  50  40 147  46 145  43  83  52  23  84  24
123
##  [685] 135 121 139  46 114  77  61  21  23 104 145 103  38  89  84  69  74
21
##  [703]  20  44 110  33  35  60 116 138  51  41 105  57 111 118  60  89  85
148
##  [721] 109  52  84  22 115 114 132  80 113 116  94  33 115  98  91  94  95
34
##  [739] 152  64  29  61  64  28  72  30  97  50  76 119  54 103  36 100  79
35
##  [757]  64 106 145  81 149 119 107  71 104  56  51  56  81  81 130 138  86
52
##  [775] 132  38 117 118 146  18  86  23  96  26 131  24 111 101 132  53  80
43
##  [793]  99  32  46 113  65 123 119 109  15  93  64  26 116 125  54  76 115
```

```
102
##  [811] 123  62  81 151 106  43 122  49  39  82 112 104 137 123  54  71 104
152
##  [829] 135  46 120  43  70 153  97  55  47  34 133 120  26 102  50 132 116
81
##  [847]  20  70  22  89  35 108  18 120  83 124 111 121  32  68  84  89  35
76
##  [865]  75  87 119  91  68  25  82  87  76 149  48  79  71  17 124  75  27
121
##  [883]  85 102  63 120 110 121  16 145  72  35  54 100  66  50  40 114  51
108
##  [901]  21  33  31 124  82  55  74  92 139  76 121 132  87 101 108  28  69
145
##  [919] 101  75 145  26  41  16  79  45 131 128  45  55  53  28 123 131  50
48
##  [937] 122 119  31  29  85  21 133  44 121  34  24 143 139  67  32  45  55
56
##  [955]  90 103 107  72  69  89 130 118 112  45 139 151 140 147  39  19  75
44
##  [973]  93  72  21 122  70  50  93  94 128 108  47 101  24 124  15 134  24
36
##  [991] 137 147  62  34 103 133  40 118  56 108

my_tab1 <- table(Z2)
t1 <- as.data.frame(my_tab1)
Z3 <- c()
for (val in Z2)
{if(val %% 3 == 0)
    Z3 <- append(Z3,val)
}
Z3

##   [1]  72 117 108  24  54  15 120  30  39 147 141  72 120  27  81 135 105
48
##  [19] 123  69  60 138 135  63  81  21  69 123  69  45  39  90 114  99  51
24
##  [37]  72  96  96  27  93  45 150  15 102 120  96 108  18  48 114  81  54
150
##  [55]  30  45 141  87  72  99  72  84  21 147  66 132  51  24 126  33  30
129
##  [73] 135  87  24  63 120 150  15 105  48  75  48  39  30 123  51  81 132
99
##  [91]  87 114  48  15  21 126  66  15  72  24  39  15 123  27 147  87 108
129
## [109]  15  87  18 117 111  42  51 117  48  39  15 126  84 108 153  81 141
54
## [127]  54 150  78  60  36  93  33 153  21  99  27 153  42  18  99 135  96
72
## [145] 150 144 123 126 123  30 141  90 147  90 117  96  45  63  99  78  72
132
```

```
## [163]  18 129  84  33  84 138 108  75 144  15  21  21  21  45  39  93  21
87
## [181]  39  24 153 102  27 102  48  96  93  18  84  78  78  93  27 123  60
54
## [199] 135 147  69  57  48 108 126  99  90 153  30  24  21 129 147  72 129
105
## [217] 108 129 105  54 147  99  36 129  99  69  99 147  84  24 123 135 114
21
## [235]  84  69  21  33  60 138  51 105  57 111  60  84 114 132  33  72  30
54
## [253]  36  81  51  81  81 138 132 117  18  96  24 111 132  99 123  15  93
54
## [271] 102 123  81  39 123  54 135 120 153 120 102 132  81 108  18 120 111
84
## [289]  75  87  87  48  75  27 102  63 120  72  54  66 114  51 108  21  33
132
## [307]  87 108  69  75  45  45 123  48  21  24  45  90  72  69  45 147  39
75
## [325]  93  72  21  93 108  24  15  24  36 147 108
```

```
my_tab <- table(Z3)
my_tab
```

```
## Z3
##  15  18  21  24  27  30  33  36  39  42  45  48  51  54  57  60  63  66
69  72
##  11   7  14  12   7   7   6   4   9   2   9  10   7  10   2   5   4   3
8  13
##  75  78  81  84  87  90  93  96  99 102 105 108 111 114 117 120 123 126
129 132
##   6   4  10   9   9   5   8   7  11   6   5  12   4   6   5   8  12   5
7   8
## 135 138 141 144 147 150 153
##   7   4   4   2  10   5   6
```

```
t2 <- as.data.frame(my_tab)
t2
```

```
##     Z3 Freq
## 1   15   11
## 2   18    7
## 3   21   14
## 4   24   12
## 5   27    7
## 6   30    7
## 7   33    6
## 8   36    4
## 9   39    9
## 10  42    2
## 11  45    9
## 12  48   10
```

```
## 13  51    7
## 14  54   10
## 15  57    2
## 16  60    5
## 17  63    4
## 18  66    3
## 19  69    8
## 20  72   13
## 21  75    6
## 22  78    4
## 23  81   10
## 24  84    9
## 25  87    9
## 26  90    5
## 27  93    8
## 28  96    7
## 29  99   11
## 30 102    6
## 31 105    5
## 32 108   12
## 33 111    4
## 34 114    6
## 35 117    5
## 36 120    8
## 37 123   12
## 38 126    5
## 39 129    7
## 40 132    8
## 41 135    7
## 42 138    4
## 43 141    4
## 44 144    2
## 45 147   10
## 46 150    5
## 47 153    6
```

Question 4

The mtcars data set reports the fuel consumption and 10 aspects of automobile design and performance for automobiles (1973–74 models). [10 points] # library(datasets) head(mtcars) a) In regression analysis, the coefficients of the regression model

$$y = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p$$

, are estimated by

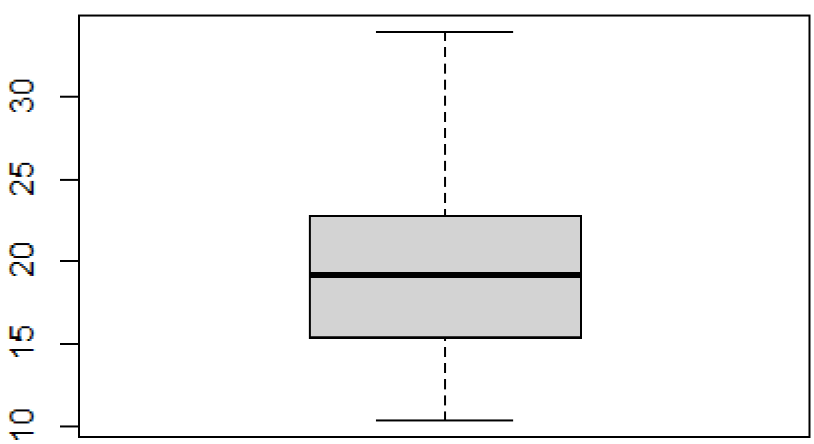
$$\hat{\beta} = (X^T X)^{-1} X^T y$$

where X is n × (p + 1) design matrix (i.e., n observations with p + 1 columns) and y is the response vector of size n. Note that the first column of X is 1 values to accommodate the intercept of the model. Let 'mpg' be the response variable and dsign matrix includes 'cyl, disp, wt, qsec' variables where p = 4. Write an R script that computes $\hat{\beta}$ of the model as described above. Hint: Do not use 'lm' function. You have to estimate them via the matrix computation.

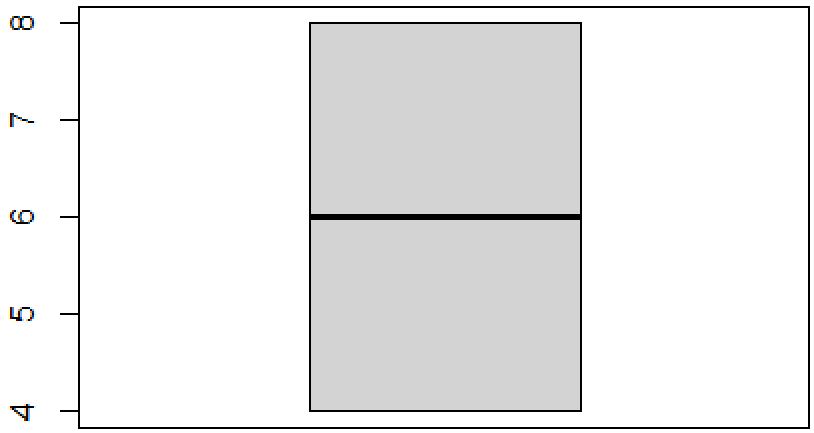
b) Consider the response and four explanatory variables from part (a). Display the boxplot of the variables separably. Then explain which measure (mean vs median) should be used to describe the center of the variables.

```
library(datasets)
class(mtcars)

## [1] "data.frame"

head(mtcars)

##                    mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1

int<-rep(1)
x1 <- mtcars$cyl
x2 <- mtcars$disp
x3 <- mtcars$wt
x4 <- mtcars$qsec

#print out x matrix
xmat <- cbind(int, x1, x2, x3,x4)
y <- mtcars$mpg
xtx <-t(xmat) %*% xmat
xty <- t(xmat) %*% y
Bhat <- solve(xtx) %*% xty
Bhat

##            [,1]
## int 30.17771379
## x1  -1.24109194
## x2   0.01029241
## x3  -4.55318282
## x4   0.55276758

# to check the values we get from Bhat is same with summary estimate
xy.lm <- lm(y~ x1 + x2 + x3 + x4)
summary(xy.lm)
```

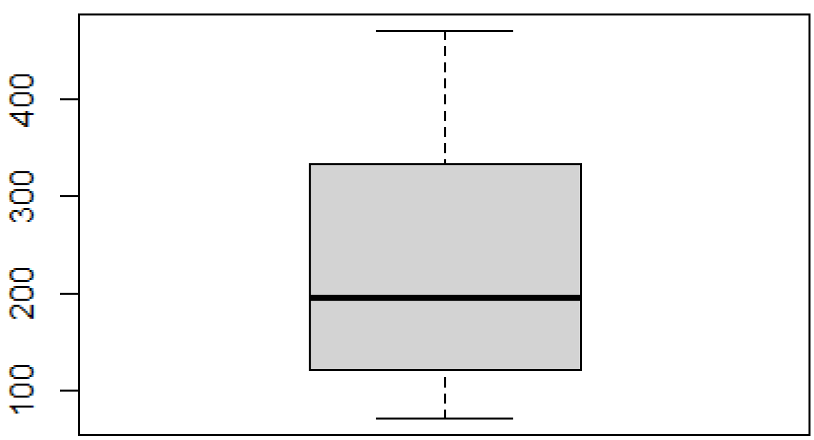
```
##
## Call:
## lm(formula = y ~ x1 + x2 + x3 + x4)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.7867 -1.5997 -0.2629  1.2263  5.6313
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 30.17771    8.27155   3.648  0.00111 **
## x1          -1.24109    0.71154  -1.744  0.09249 .
## x2           0.01029    0.01182   0.871  0.39142
## x3          -4.55318    1.21356  -3.752  0.00085 ***
## x4           0.55277    0.39373   1.404  0.17174
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.551 on 27 degrees of freedom
## Multiple R-squared:  0.844,  Adjusted R-squared:  0.8209
## F-statistic: 36.52 on 4 and 27 DF,  p-value: 1.587e-10

##question 4(b)

boxplot(y)
```



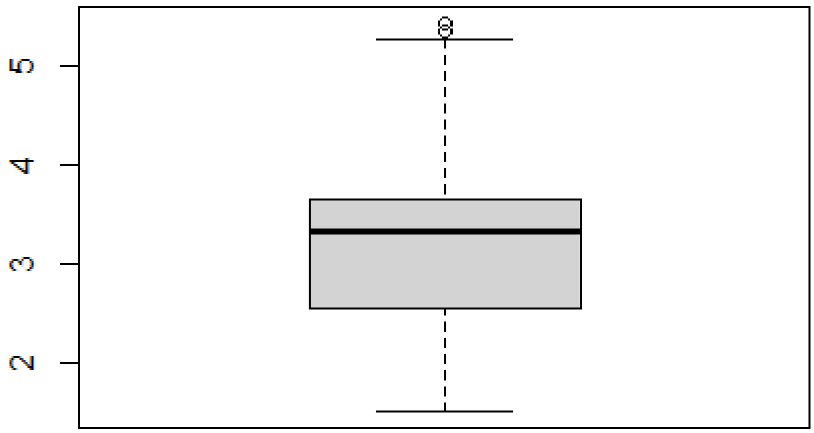

```
boxplot(x1)
```



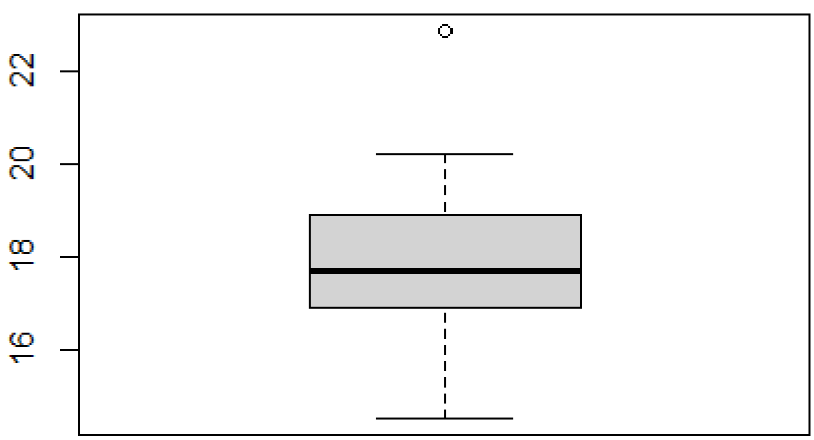

```
boxplot(x2)
```




```
boxplot(x3)
```




```
boxplot(x4)
```




```
summary(y)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   10.40   15.43   19.20   20.09   22.80   33.90

summary(x1)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   4.000   4.000   6.000   6.188   8.000   8.000

summary(x2)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    71.1   120.8   196.3   230.7   326.0   472.0

summary(x3)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.513   2.581   3.325   3.217   3.610   5.424

summary(x4)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   14.50   16.89   17.71   17.85   18.90   22.90
```

For variables (y, X1, x3) we can see that the median 19.20 is slightly lower than the mean 20.09 and with the absence of outliers, the mean(20.09) can be used as the center of the variable "y".It's best to use the mean to describe the center of a dataset when the distribution is mostly symmetrical and there are no outliers.

For variables (x3, x4),the median will do a better job of capturing the central location of a distribution when there is an outlier present in the data.This is because the large values on the tail end of the distribution tend to pull the mean away from the center and towards the long tail. So, It is best to use the median when the distribution is either skewed or there are outliers present.

Question 5

a) Create 25 random numbers between 1 to 51 and store them in an square matrix Y of size 5.

b) Add normal noises (from normal distribution with mean 0 and standard deviation 1) to the diagonal elements of Y.

c) Find the inverse matrix of Y (obtained from part b).

d) Show numerically that the matrix product of Y (part b) and its inverse (part c) is an identity matrix.

```
#creation of five random numbers

set.seed(123456)
R<-c(sample(x=1:51, size = 25,replace = F))
R

##  [1] 42 51 49  7 45 36 38 10  3 43 30 23 50 28  2 16 21  1 24 31 37 14 29
15 41

Y<-matrix(R, nrow = 5,ncol = 5)
Y

##      [,1] [,2] [,3] [,4] [,5]
## [1,]   42   36   30   16   37
## [2,]   51   38   23   21   14
## [3,]   49   10   50    1   29
## [4,]    7    3   28   24   15
## [5,]   45   43    2   31   41

#adding normal noises

normal_noises<-rnorm(5,mean = 0,sd=1)
normal_noises

## [1]  1.66821097  0.55968789 -0.75397477  1.25655419  0.03849255

normal_noisesMatrix<-normal_noises
diag(normal_noisesMatrix)

##          [,1]      [,2]       [,3]     [,4]       [,5]
## [1,] 1.668211 0.0000000  0.0000000 0.000000 0.00000000
## [2,] 0.000000 0.5596879  0.0000000 0.000000 0.00000000
## [3,] 0.000000 0.0000000 -0.7539748 0.000000 0.00000000
## [4,] 0.000000 0.0000000  0.0000000 1.256554 0.00000000
## [5,] 0.000000 0.0000000  0.0000000 0.000000 0.03849255

Y_noise<-Y+diag(normal_noisesMatrix)
Y_noise

##          [,1]     [,2]     [,3]     [,4]     [,5]
## [1,] 43.66821 36.00000 30.00000 16.00000 37.00000
## [2,] 51.00000 38.55969 23.00000 21.00000 14.00000
## [3,] 49.00000 10.00000 49.24603  1.00000 29.00000
## [4,]  7.00000  3.00000 28.00000 25.25655 15.00000
## [5,] 45.00000 43.00000  2.00000 31.00000 41.03849
```

```r
#inverse of matrix
Y_noiseInverse<-solve(Y_noise)
Y_noiseInverse

##             [,1]         [,2]          [,3]         [,4]        [,5]
## [1,] -0.05464423  0.014778354  0.0314434815 -0.010746751  0.02593370
## [2,]  0.06804190  0.010789201 -0.0397790975 -0.009410432 -0.03347708
## [3,]  0.03211617  0.004295913 -0.0074939036  0.013091987 -0.02991089
## [4,] -0.03782688  0.011466982 -0.0003188779  0.030288387  0.01934716
## [5,]  0.01563385 -0.036381205  0.0078077319 -0.001873159  0.01785050

#showing numerically that dot product of m,atrix and inverse gives the
identity matrix
identityMatrix<-Y_noise%*%Y_noiseInverse
identityMatrix

##               [,1]          [,2]          [,3]          [,4]          [,5]
## [1,]  1.000000e+00  0.000000e+00 -5.551115e-17 -3.053113e-16 -1.110223e-16
## [2,]  8.326673e-17  1.000000e+00  5.551115e-17 -4.857226e-17 -1.387779e-16
## [3,] -2.775558e-16  0.000000e+00  1.000000e+00 -3.469447e-17 -3.330669e-16
## [4,] -5.551115e-17 -1.110223e-16  2.775558e-17  1.000000e+00 -1.110223e-16
## [5,] -5.551115e-16 -2.220446e-16 -3.330669e-16 -1.110223e-16  1.000000e+00

#I am not getting back exactly zero for the off diagonals because of floating
point precision errors. So i will have to round up using round()

round(identityMatrix)

##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    0    0    0    0
## [2,]    0    1    0    0    0
## [3,]    0    0    1    0    0
## [4,]    0    0    0    1    0
## [5,]    0    0    0    0    1
```

Question 6

Create the following data frame and name it "exams". set.seed(123) d3 <- data.frame(student = c("Alice", "Sarah", "Harry", "Ron", "Kate"), score = sample(80:100, 5), letter = sample(c("A","B"), 5, replace = TRUE), late = sample(c(T, F), 5, replace = TRUE))

a) Compute the mean score for this exam and print it
b) Find the student with the highest score and print the row corresponding to the student. Hint: you can use command 'which.max'.
c) Write an R script to re-arrange the rows of the data set based on the score variable (i.e, the student with maximum score in the first row and student with minimum score in the last row).

```r
set.seed(123)
d3<-
```

```
data.frame(student=c("Alice","Sarah","Harry","Ron","Kate"),score=sample(80:10
0,5),letter=sample(c("A","B"),5,replace =TRUE),late=sample(c(T,F),5,replace =
TRUE))
exams<-d3
exams

##   student score letter  late
## 1   Alice    94      B FALSE
## 2   Sarah    98      B FALSE
## 3   Harry    93      A  TRUE
## 4     Ron    82      A FALSE
## 5    Kate    89      B  TRUE

#calculating the mean score for the exam

mean_score <- mean(exams$score)
mean_score

## [1] 91.2

#row corresponding to student with highest score

which.max(exams$score)

## [1] 2

exams[2,]

##   student score letter  late
## 2   Sarah    98      B FALSE

#arrangement of rows of data set based on score variable.

newdata<-exams[order((exams$score),decreasing=TRUE),]
newdata

##   student score letter  late
## 2   Sarah    98      B FALSE
## 1   Alice    94      B FALSE
## 3   Harry    93      A  TRUE
## 5    Kate    89      B  TRUE
## 4     Ron    82      A FALSE
```

Question 7

From a survey of the clerical employees of a large financial organization, the data are aggregated from the questionnaires of the approximately 35 employees. # library(datasets) head(attitude)

a) Take a sample of size 150 from the employees (with replacement). Show the histograms of the sampled 'complaints' and 'learning'. This distribution is called the sampling distribution of the variable.

b) Compare the sampling distributions (part a) with the population distribution of the variables (i.e., the distribution based on all the values in the data set)? Are the sample distributions similar to the population distributions? Why?

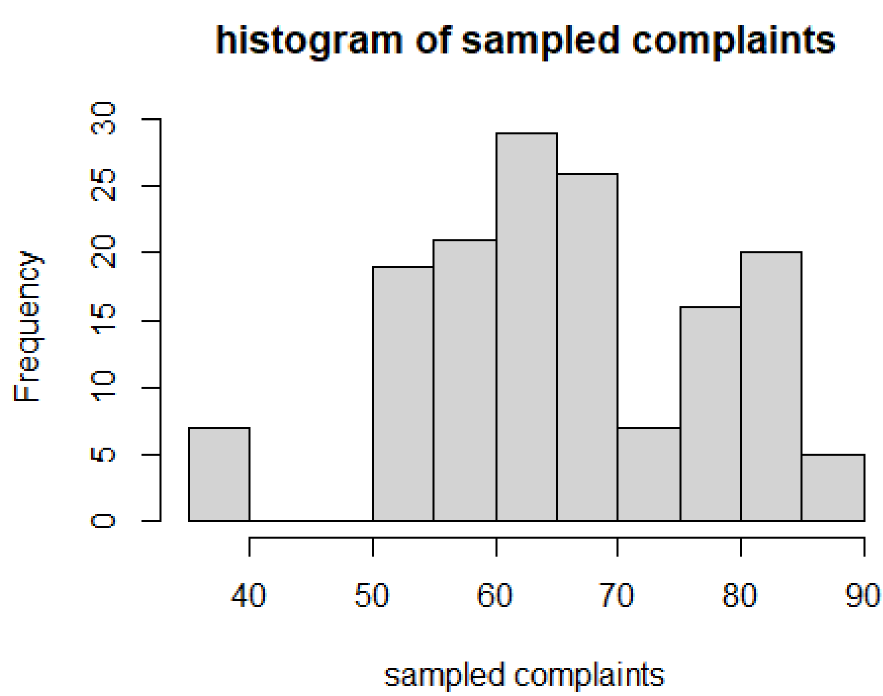
```
library(datasets)
data <- attitude
set.seed(12345)
data_s <- data[sample(1:nrow(data), size = 150, replace = T),]
data_s

##        rating complaints privileges learning raises critical advance
## 14         68         83         83       45     59       77      35
## 19         65         70         46       57     75       85      46
## 16         81         90         50       72     60       54      36
## 26         66         77         66       63     88       76      72
## 28         48         57         44       45     51       83      38
## 24         40         37         42       58     50       57      49
## 26.1       66         77         66       63     88       76      72
## 29         85         85         71       71     77       74      55
## 11         64         53         53       58     58       67      34
## 24.1       40         37         42       58     50       57      49
## 2          63         64         51       54     63       73      47
## 22         64         61         52       62     66       80      41
## 11.1       64         53         53       58     58       67      34
## 6          43         55         49       44     54       49      34
## 7          58         67         42       56     66       68      35
## 30         82         82         39       59     64       78      39
## 10         67         61         45       47     62       80      41
## 17         74         85         64       69     79       79      63
## 8          71         75         50       55     70       66      41
## 7.1        58         67         42       56     66       68      35
## 6.1        43         55         49       44     54       49      34
## 30.1       82         82         39       59     64       78      39
## 1          43         51         30       39     61       92      45
## 12         67         60         47       39     59       74      41
## 20         50         58         68       54     64       78      52
## 8.1        71         75         50       55     70       66      41
## 26.2       66         77         66       63     88       76      72
## 12.1       67         60         47       39     59       74      41
## 3          71         70         68       69     76       86      48
## 9          72         82         72       67     71       83      31
## 14.1       68         83         83       45     59       77      35
## 13         69         62         57       42     55       63      25
## 20.1       50         58         68       54     64       78      52
## 10.1       67         61         45       47     62       80      41
## 23         53         66         52       50     63       80      37
```

```
## 16.1         81          90          50       72     60       54      36
## 16.2         81          90          50       72     60       54      36
## 30.2         82          82          39       59     64       78      39
## 2.1          63          64          51       54     63       73      47
## 20.2         50          58          68       54     64       78      52
## 27           78          75          58       74     80       78      49
## 28.1         48          57          44       45     51       83      38
## 9.1          72          82          72       67     71       83      31
## 25           63          54          42       48     66       75      33
## 4            61          63          45       47     54       84      35
## 8.2          71          75          50       55     70       66      41
## 11.2         64          53          53       58     58       67      34
## 6.2          43          55          49       44     54       49      34
## 9.2          72          82          72       67     71       83      31
## 5            81          78          56       66     71       83      47
## 15           77          77          54       72     79       77      46
## 19.1         65          70          46       57     75       85      46
## 17.1         74          85          64       69     79       79      63
## 17.2         74          85          64       69     79       79      63
## 5.1          81          78          56       66     71       83      47
## 26.3         66          77          66       63     88       76      72
## 10.2         67          61          45       47     62       80      41
## 3.1          71          70          68       69     76       86      48
## 19.2         65          70          46       57     75       85      46
## 13.1         69          62          57       42     55       63      25
## 3.2          71          70          68       69     76       86      48
## 17.3         74          85          64       69     79       79      63
## 10.3         67          61          45       47     62       80      41
## 24.2         40          37          42       58     50       57      49
## 11.3         64          53          53       58     58       67      34
## 25.1         63          54          42       48     66       75      33
## 24.3         40          37          42       58     50       57      49
## 27.1         78          75          58       74     80       78      49
## 20.3         50          58          68       54     64       78      52
## 2.2          63          64          51       54     63       73      47
## 12.2         67          60          47       39     59       74      41
## 3.3          71          70          68       69     76       86      48
## 13.2         69          62          57       42     55       63      25
## 22.1         64          61          52       62     66       80      41
## 19.3         65          70          46       57     75       85      46
## 23.1         53          66          52       50     63       80      37
## 10.4         67          61          45       47     62       80      41
## 23.2         53          66          52       50     63       80      37
## 26.4         66          77          66       63     88       76      72
## 23.3         53          66          52       50     63       80      37
## 7.2          58          67          42       56     66       68      35
## 15.1         77          77          54       72     79       77      46
## 5.2          81          78          56       66     71       83      47
## 14.2         68          83          83       45     59       77      35
## 23.4         53          66          52       50     63       80      37
```

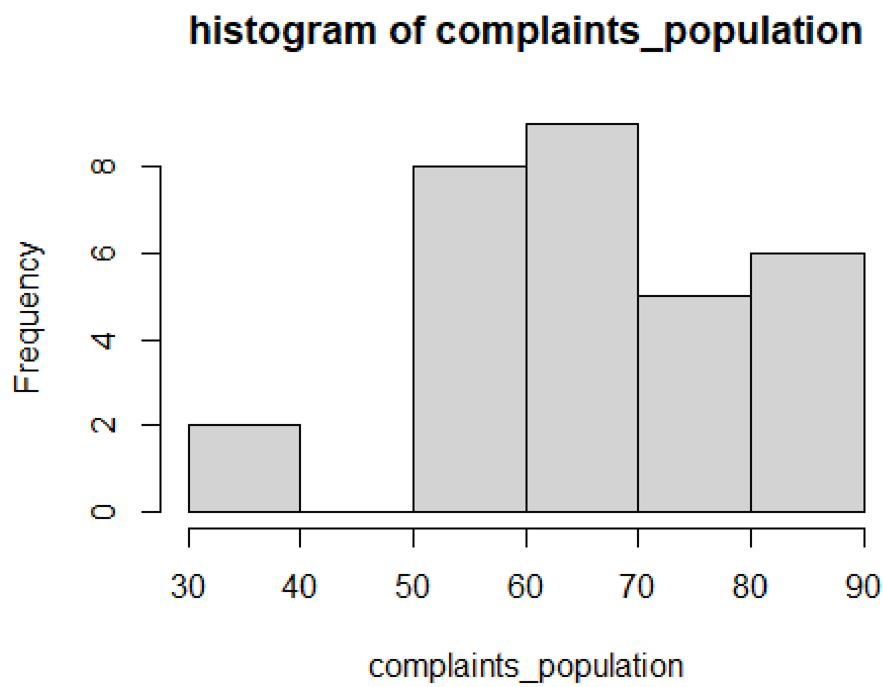
```
## 28.2      48      57      44      45      51      83      38
## 12.3      67      60      47      39      59      74      41
## 26.5      66      77      66      63      88      76      72
## 29.1      85      85      71      71      77      74      55
## 28.3      48      57      44      45      51      83      38
## 24.4      40      37      42      58      50      57      49
## 3.4       71      70      68      69      76      86      48
## 5.3       81      78      56      66      71      83      47
## 11.4      64      53      53      58      58      67      34
## 4.1       61      63      45      47      54      84      35
## 25.2      63      54      42      48      66      75      33
## 11.5      64      53      53      58      58      67      34
## 17.4      74      85      64      69      79      79      63
## 3.5       71      70      68      69      76      86      48
## 10.5      67      61      45      47      62      80      41
## 15.2      77      77      54      72      79      77      46
## 7.3       58      67      42      56      66      68      35
## 10.6      67      61      45      47      62      80      41
## 26.6      66      77      66      63      88      76      72
## 23.5      53      66      52      50      63      80      37
## 4.2       61      63      45      47      54      84      35
## 28.4      48      57      44      45      51      83      38
## 10.7      67      61      45      47      62      80      41
## 4.3       61      63      45      47      54      84      35
## 13.3      69      62      57      42      55      63      25
## 25.3      63      54      42      48      66      75      33
## 18        65      60      65      75      55      80      60
## 24.5      40      37      42      58      50      57      49
## 10.8      67      61      45      47      62      80      41
## 1.1       43      51      30      39      61      92      45
## 28.5      48      57      44      45      51      83      38
## 16.3      81      90      50      72      60      54      36
## 9.3       72      82      72      67      71      83      31
## 22.2      64      61      52      62      66      80      41
## 23.6      53      66      52      50      63      80      37
## 17.5      74      85      64      69      79      79      63
## 3.6       71      70      68      69      76      86      48
## 18.1      65      60      65      75      55      80      60
## 5.4       81      78      56      66      71      83      47
## 10.9      67      61      45      47      62      80      41
## 22.3      64      61      52      62      66      80      41
## 7.4       58      67      42      56      66      68      35
## 25.4      63      54      42      48      66      75      33
## 10.10     67      61      45      47      62      80      41
## 20.4      50      58      68      54      64      78      52
## 12.4      67      60      47      39      59      74      41
## 2.3       63      64      51      54      63      73      47
## 4.4       61      63      45      47      54      84      35
## 3.7       71      70      68      69      76      86      48
## 6.3       43      55      49      44      54      49      34
```

```
## 24.6       40          37          42        58       50        57        49
## 23.7       53          66          52        50       63        80        37
## 15.3       77          77          54        72       79        77        46
## 4.5        61          63          45        47       54        84        35
## 27.2       78          75          58        74       80        78        49
## 30.3       82          82          39        59       64        78        39
## 18.2       65          60          65        75       55        80        60
## 3.8        71          70          68        69       76        86        48
## 8.3        71          75          50        55       70        66        41
## 29.2       85          85          71        71       77        74        55
## 25.5       63          54          42        48       66        75        33
## 16.4       81          90          50        72       60        54        36
## 25.6       63          54          42        48       66        75        33
## 28.6       48          57          44        45       51        83        38
## 12.5       67          60          47        39       59        74        41
```

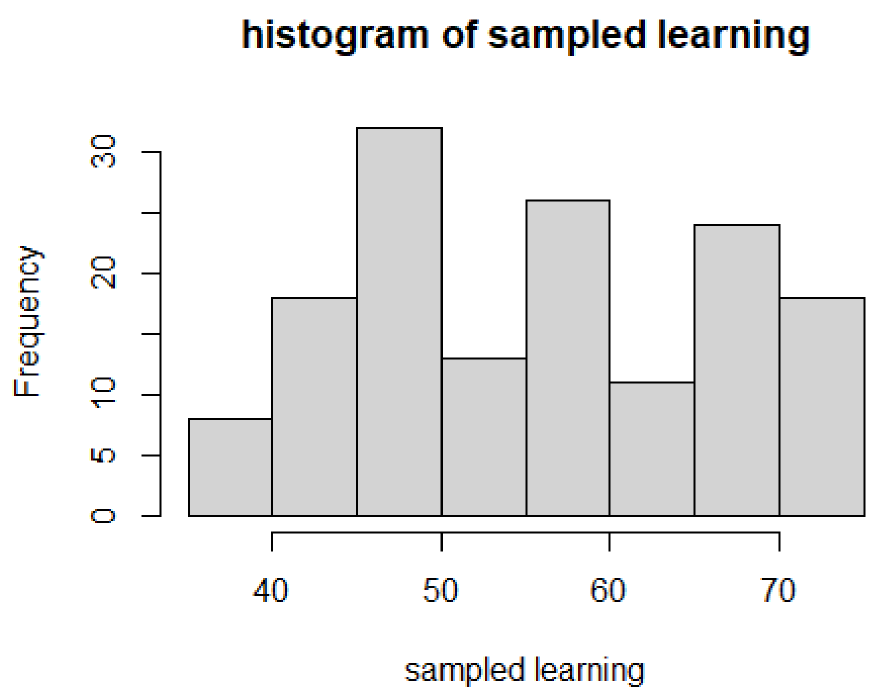
```
hist(data_s$complaints, main = "histogram of sampled complaints", xlab =
"sampled complaints")
```



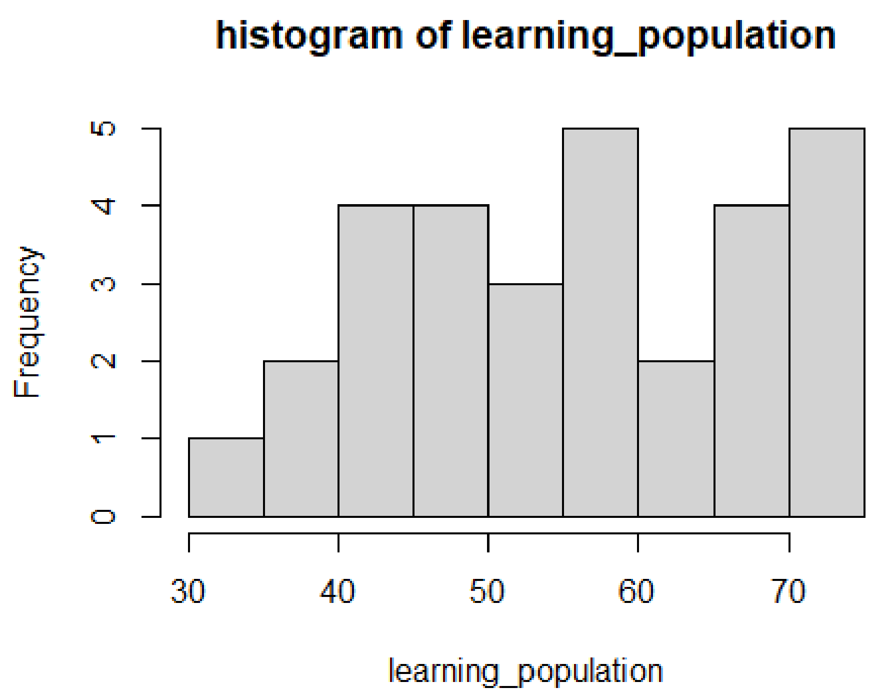

```
hist(data$complaints, main = "histogram of complaints_population", xlab =
"complaints_population")
```




```
hist(data_s$learning, main = "histogram of sampled learning", xlab = "sampled
learning")
```



```r
hist(data$learning, main = "histogram of learning_population", xlab =
"learning_population")
```



Question 8

Find all numbers not greater than 10,000 that are divisible by 5,7, and 11 and print them.

```r
n <- 1
while( n < 10000)
{
   if(n %% 5 == 0 && n %% 7 == 0 && n %% 11 == 0){
        print(n)
        }
        n <- n + 1
}
```

```
## [1] 385
## [1] 770
## [1] 1155
## [1] 1540
## [1] 1925
## [1] 2310
## [1] 2695
## [1] 3080
```

```
## [1] 3465
## [1] 3850
## [1] 4235
## [1] 4620
## [1] 5005
## [1] 5390
## [1] 5775
## [1] 6160
## [1] 6545
## [1] 6930
## [1] 7315
## [1] 7700
## [1] 8085
## [1] 8470
## [1] 8855
## [1] 9240
## [1] 9625
```

Question 9

Print for each of the numbers x = 2, . . ., 20, all numbers that divide x (all factors) excluding 1 and x. For example, for 18, it should print 2 3 6 9.

```r
for (i in 2:20){
  fac <- c()
  for (j in 2:i) {
    if(i%%j==0 && j!=1 && j!=i){
      fac<-c(fac,j)
      }
  }
  cat("factors of", i, "are", fac, "\n")
}
```

```
## factors of 2 are
## factors of 3 are
## factors of 4 are 2
## factors of 5 are
## factors of 6 are 2 3
## factors of 7 are
## factors of 8 are 2 4
## factors of 9 are 3
## factors of 10 are 2 5
## factors of 11 are
## factors of 12 are 2 3 4 6
## factors of 13 are
## factors of 14 are 2 7
## factors of 15 are 3 5
## factors of 16 are 2 4 8
## factors of 17 are
```

```
## factors of 18 are 2 3 6 9
## factors of 19 are
## factors of 20 are 2 4 5 10
```

Question 10

Write a function that takes in a data set as an input, returns a list including the column names, the dimensions of the data and the range of variables. Then apply your function to sleep data set. [10 points]

Library(datasets)

head(sleep)

```
library(datasets)
#attach(sleep)

myfunction <- function(a){
  y <- as.vector(a)
  mylist <- list(colnames(a), dim(a), range(y))
   return(mylist)

}
myfunction(sleep)

## [[1]]
## [1] "extra" "group" "ID"
##
## [[2]]
## [1] 20  3
##
## [[3]]
## [1] -1.6 10.0
```