Project 3 Unit 3

Ujjwal Gautam

2024-04-30

Q1

```
aq<-airquality
aq$Temp
```

```
##   [1] 67 72 74 62 56 66 65 59 61 69 74 69 66 68 58 64 66 57 68 62 59 73 61 61 57
##  [26] 58 57 67 81 79 76 78 74 67 84 85 79 82 87 90 87 93 92 82 80 79 77 72 65 73
##  [51] 76 77 76 76 76 75 78 73 80 77 83 84 85 81 84 83 83 88 92 92 89 82 73 81 91
##  [76] 80 81 82 84 87 85 74 81 82 86 85 82 86 88 86 83 81 81 81 82 86 85 87 89 90
## [101] 90 92 86 86 82 80 79 77 79 76 78 78 77 72 75 79 81 86 88 97 94 96 94 91 92
## [126] 93 93 87 84 80 78 75 73 81 76 77 71 71 78 67 76 68 82 64 71 81 69 63 70 77
## [151] 75 76 68
```

Hint 1: Divide the "Temp" variable into different class intervals using a statistical rule and get number of frequencies in each class interval

```
num_intervals <- ceiling(log2(length(aq$Temp)) + 1)
```

- Defining the number of intervals using a statistical rule (Sturges' formula)

```
aq_intervals <- cut(aq$Temp, breaks = num_intervals)
```

- Creating intervals using the cut() function

```
aq_interval_freq <- table(aq_intervals)
aq_interval_freq
```

```
## aq_intervals
##   (56,60.6] (60.6,65.1] (65.1,69.7] (69.7,74.2] (74.2,78.8] (78.8,83.3]
##           8          10          14          16          26          35
## (83.3,87.9] (87.9,92.4]   (92.4,97]
##          22          15           7
```

- Calculating frequencies in each interval using the table() function

Hint 2: Get less than frequency data for less than ogive

```
cumulative_freq <- cumsum(aq_interval_freq)
cumulative_freq
```

```
##   (56,60.6] (60.6,65.1] (65.1,69.7] (69.7,74.2] (74.2,78.8] (78.8,83.3]
##           8          18          32          48          74         109
## (83.3,87.9] (87.9,92.4]   (92.4,97]
##         131         146         153
```

```
cumulative_freq2 <- cumsum(rev(aq_interval_freq))
cumulative_freq2
```

```
##   (92.4,97] (87.9,92.4] (83.3,87.9] (78.8,83.3] (74.2,78.8] (69.7,74.2]
##           7          22          44          79         105         121
## (65.1,69.7] (60.6,65.1]   (56,60.6]
##         135         145         153
```

- Calculate cumulative frequencies

```
upper_boundaries <- as.numeric(sub("\\((.*),.*\\]", "\\1", names(cumulative_freq)))
upper_boundaries
```

```
## [1] 56.0 60.6 65.1 69.7 74.2 78.8 83.3 87.9 92.4
```

- Geting the upper class boundaries of the intervals using regular expression

```
max_upper_boundary <- max(upper_boundaries)
max_upper_boundary
```

```
## [1] 92.4
```

```
min_upper_boundary<-min(upper_boundaries)
min_upper_boundary
```

```
## [1] 56
```

- Get the maximum and minimum upper boundary

hint 3 : Get more than frequency data for more than ogive

```
rev_upper_boundaries<-rev(upper_boundaries)
rev_cumulative_freq<-rev(cumulative_freq2)
```

hint 4: Plot less than and more than ogives in a single plot

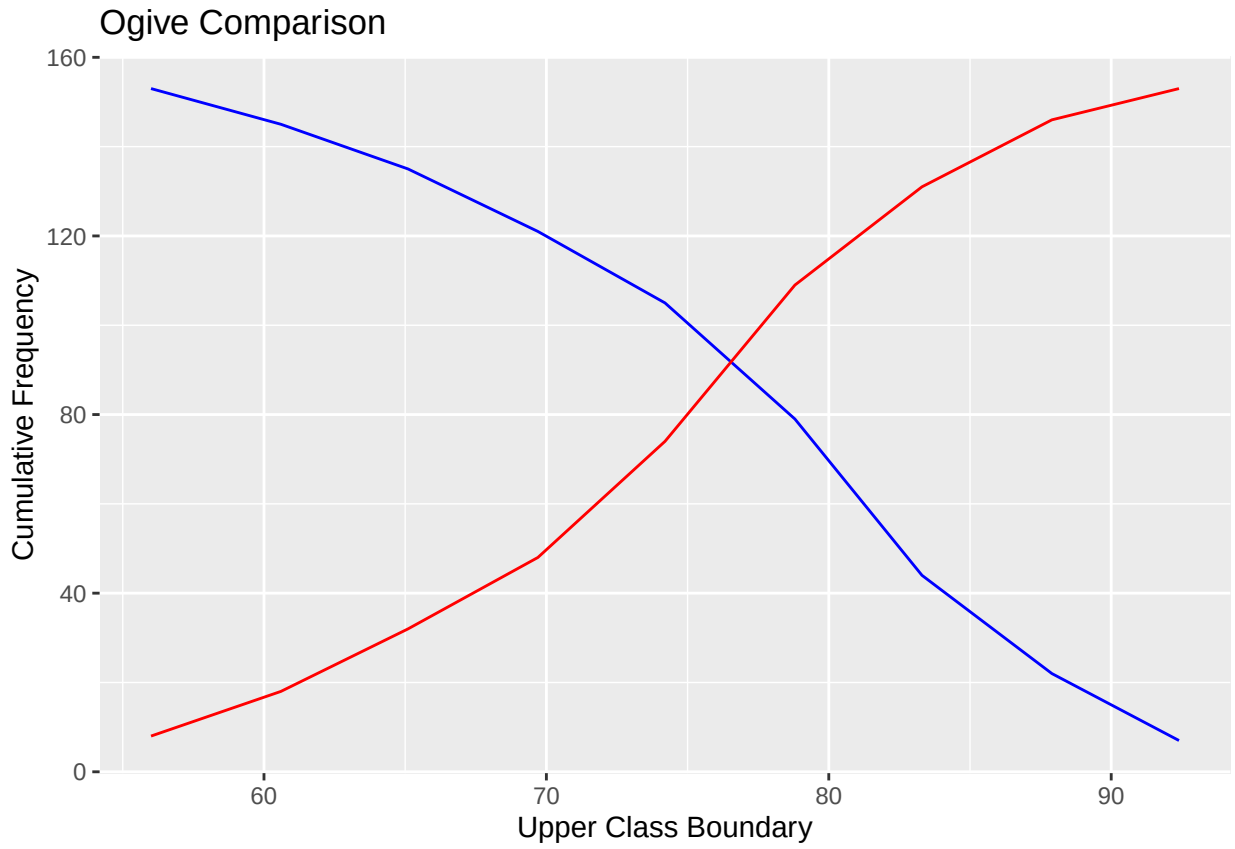
```r
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.3.3
```

```r
df_more_than <- data.frame(Upper_Class_Boundary = upper_boundaries, Cumulative_Frequency = rev_cumulati
df_less_than <- data.frame(Upper_Class_Boundary = upper_boundaries, Cumulative_Frequency = cumulative_f

common_xlim <- c(min(min_upper_boundary, rev_upper_boundaries), max(max_upper_boundary, upper_boundarie

ggplot() +
  geom_line(data = df_more_than, aes(x = Upper_Class_Boundary, y = Cumulative_Frequency), color = "blue
  geom_line(data = df_less_than, aes(x = Upper_Class_Boundary, y = Cumulative_Frequency), color = "red"
  labs(x = "Upper Class Boundary", y = "Cumulative Frequency",
       title = "Ogive Comparison") +
  xlim(common_xlim)
```



- Create data frames
- Plot the Ogives
- Determine the common xlim for both plots
- Plot the Ogives with the same xlim

hint 5 : Intersection of less than and more than ogive in the x-axis is the median

```
median(aq$Temp)
```

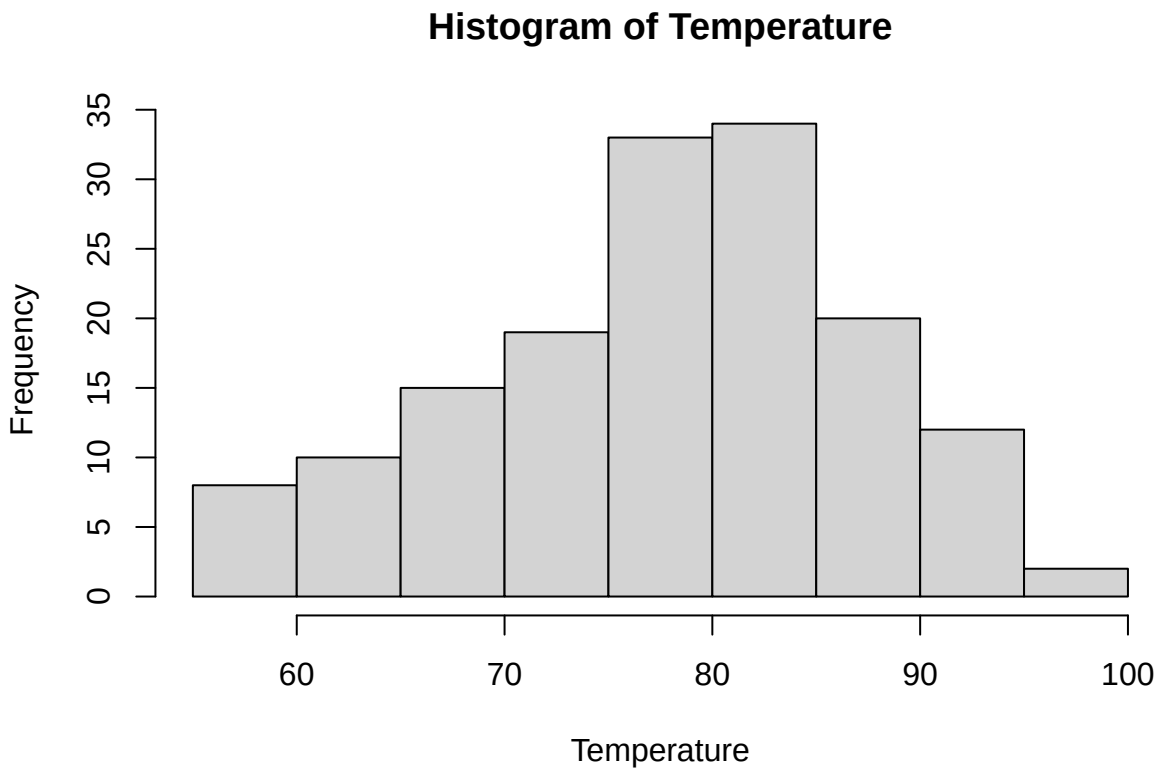
```
## [1] 79
```

- from ogive comparison the intersection was found around 77 which is not exact as median itself.

Q2

hint 7 : Get histogram of "Temp" variable

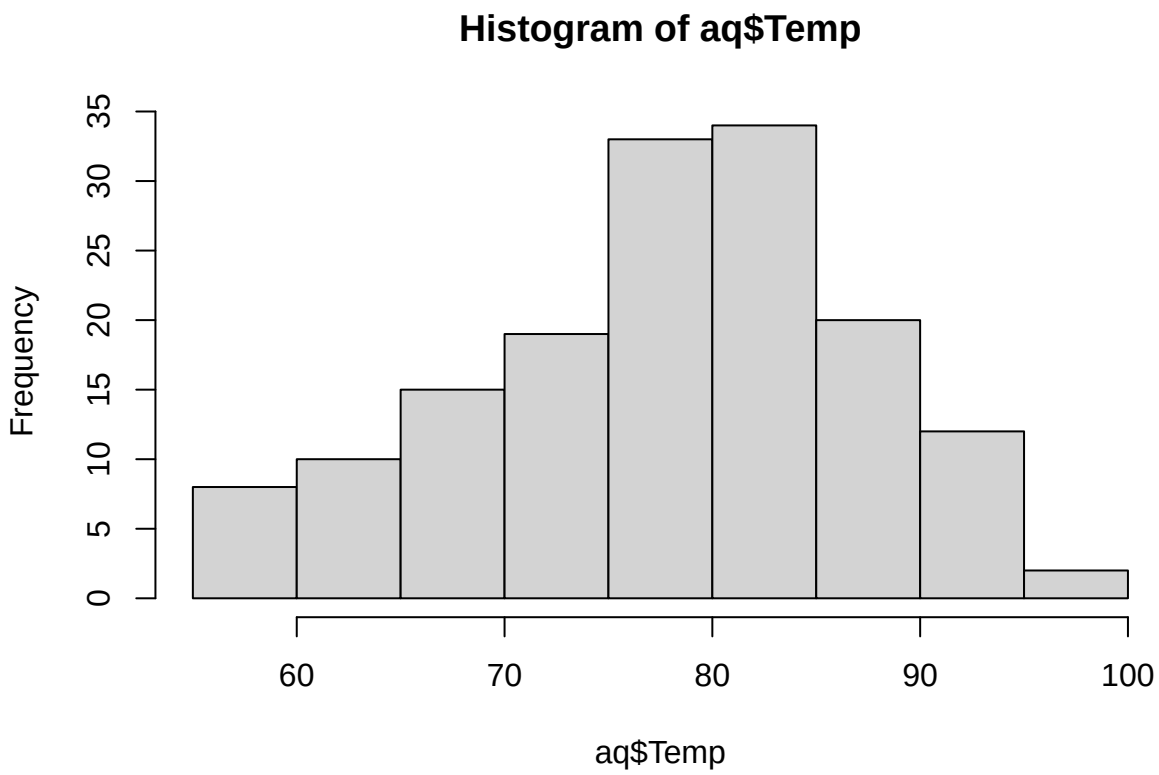
```
hist(aq$Temp, xlab = "Temperature", ylab = "Frequency", main = "Histogram of Temperature")
```



- Plotting histogram of 'Temp' variable

hint 8 : Draw a diagonal line from en edge of the largest bar to the tip of the opposite adjacent bar

```
hist_data <- hist(aq$Temp)
```



- Creating histogram

```
max_freq_index <- which.max(hist_data$counts)
max_freq_index
```

```
## [1] 6
```

- Finding the index of the bin with the highest frequency

```
left_bin <- max_freq_index - 1
left_bin
```

```
## [1] 5
```

```
right_bin <- max_freq_index + 1
right_bin
```

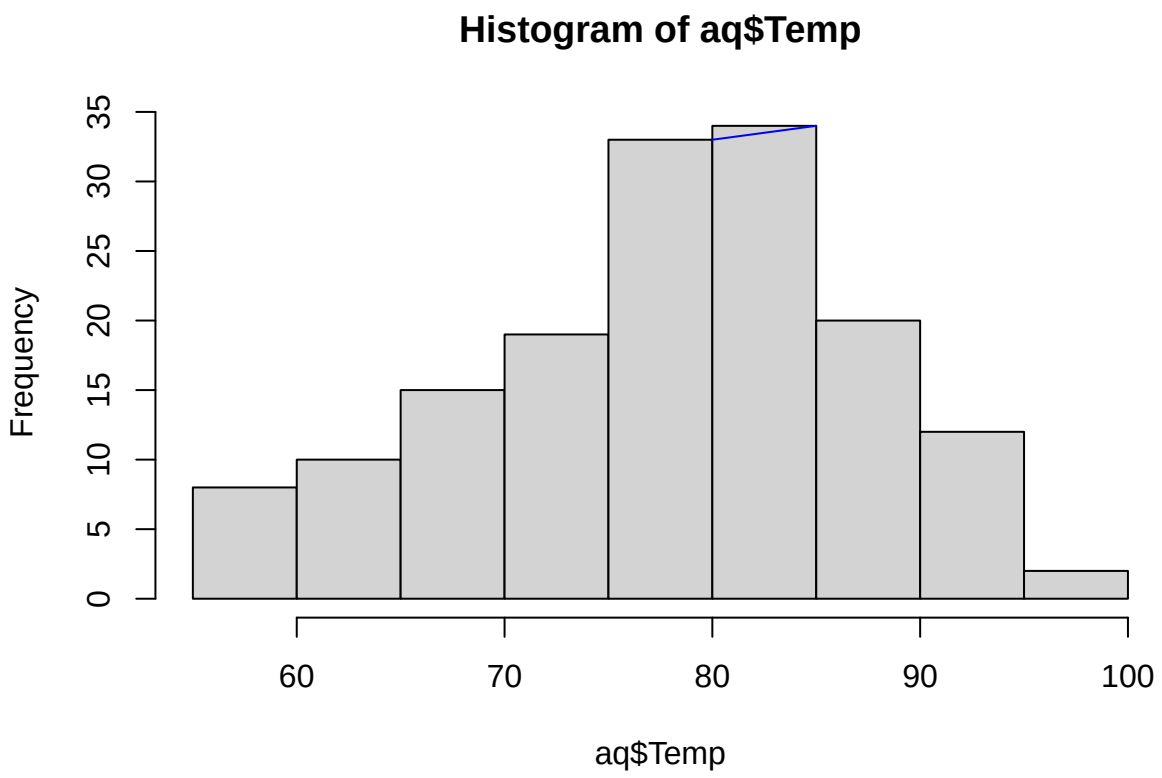
```
## [1] 7
```

- Determining the adjacent bins to the left and right of the highest frequency bin

```r
x_left1 <- hist_data$breaks[left_bin + 1]
x_right1 <- hist_data$breaks[max_freq_index + 1]
y_max <- max(hist_data$counts)
y_left1 <- hist_data$counts[left_bin]
y_right1 <- hist_data$counts[max_freq_index]
```

- Calculating coordinates for the diagonal line from the edge of the largest bar to the tip of the opposite adjacent bar

```r
plot.new()
hist_data <- hist(aq$Temp)
segments(x_left1, y_left1, x_right1, y_right1, col = "blue")
```



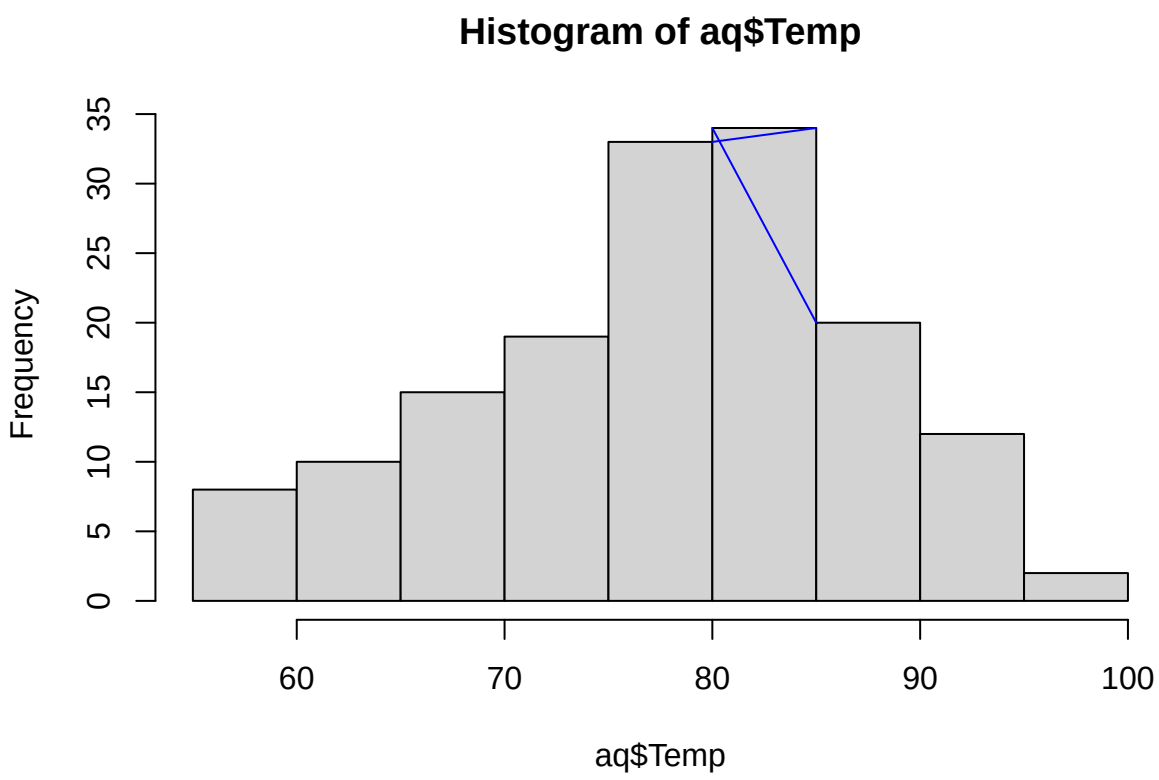
- plotting 1st diagonal line

hint 9: Draw another diagonal line from other edge of the largest bar to the tip of of the opposite adjacent bar

```r
x_left2 <- hist_data$breaks[max_freq_index]  # Other edge of the largest bar
x_right2 <- hist_data$breaks[right_bin]  # Tip of the opposite adjacent bar
y_left2 <- hist_data$counts[max_freq_index]
y_right2 <- hist_data$counts[right_bin]
```

- Calculating coordinates for the diagonal line from the other edge of the largest bar to the tip of the opposite adjacent bar

```
plot.new()
hist_data <- hist(aq$Temp)
segments(x_left1, y_left1, x_right1, y_right1, col = "blue")
segments(x_left2, y_left2, x_right2, y_right2, col = "blue")
```



- plotting 2nd diagonal line

hint 10 : Intersection of the two diagonal lines in the x-axis in the mode

```
slope1 <- (y_right1 - y_left1) / (x_right1 - x_left1)
slope2 <- (y_right2 - y_left2) / (x_right2 - x_left2)
```

- Calculate the slopes of the two lines

```
intercept1 <- y_left1 - slope1 * x_left1
intercept2 <- y_left2 - slope2 * x_left2
```

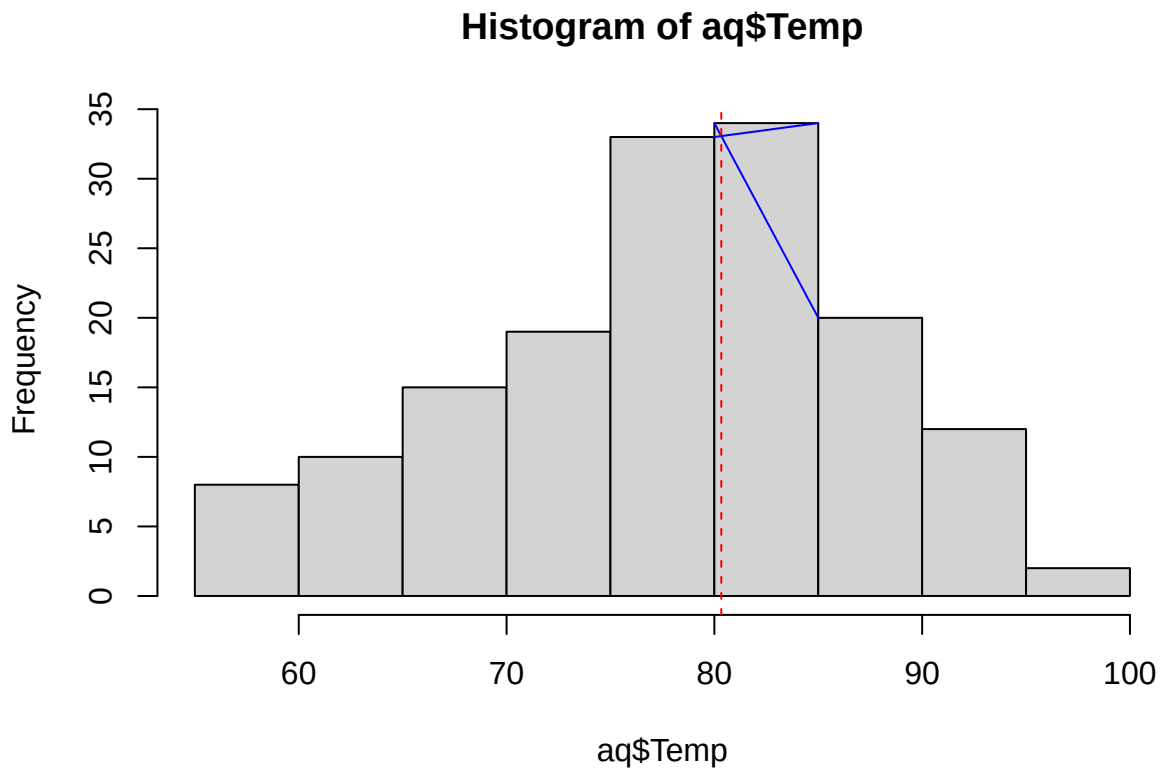
- Calculate the y-intercepts of the two lines

```r
intersection_x <- (intercept2 - intercept1) / (slope1 - slope2)
intersection_x
```

```
## [1] 80.33333
```

- Calculate the intersection point

```r
plot.new()
hist_data <- hist(aq$Temp)
segments(x_left1, y_left1, x_right1, y_right1, col = "blue")
segments(x_left2, y_left2, x_right2, y_right2, col = "blue")
abline(v = intersection_x, col = "red", lty = 2)
```



- Plot vertical line at intersection point

hint 11: Check this value with mode code of R

```r
mode(aq$Temp)
```

```
## [1] "numeric"
```

```
freq_table <- table(aq$Temp)
mode <- as.numeric(names(freq_table)[freq_table == max(freq_table)])
mode
```

```
## [1] 81
```

```
intersection_x
```

```
## [1] 80.33333
```

- two values from graph and median are similar but not exact

Q3

```
library(igraph)
```

```
## Warning: package 'igraph' was built under R version 4.3.3
```

```
##
## Attaching package: 'igraph'
```

```
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
```

```
## The following object is masked from 'package:base':
##
##     union
```

```
sna_school<-read.csv("C:/Users/Ujjwal/Desktop/MDS/R program/R program examples/ujjwal gautam - SNA_Scho
head(sna_school)
```

```
##   first second grade spec
## 1    AA     DD     6    Y
## 2    AB     DD     6    R
## 3    AF     BA     6    Q
## 4    DD     DA     6    Q
## 5    CD     EC     6    X
## 6    DD     CE     6    Y
```

hint 12 : Import and create a data frame "s" with first and second column of the data

```
s <- data.frame(first = sna_school$first, second = sna_school$second)
```

hint 13 : Save it as network graph data object "net" with directed = T argument

```
net<-graph.data.frame(s,directed = T)
```

```
## Warning: ‘graph.data.frame()‘ was deprecated in igraph 2.0.0.
## i Please use ‘graph_from_data_frame()‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

```
net
```

```
## IGRAPH d0826d2 DN-- 52 290 --
## + attr: name (v/c)
## + edges from d0826d2 (vertex names):
##  [1] AA->DD AB->DD AF->BA DD->DA CD->EC DD->CE CD->FA CD->CC BA->AF CB->CA
## [11] CC->CA CD->CA BC->CA DD->DA ED->AD AE->AC AB->BA CD->EC CA->CC EB->CC
## [21] BF->CE BB->CD AC->AE CC->FB DC->BB BD->CF DB->DA DD->DA DB->DD BC->AF
## [31] CF->DE DF->BF CB->CA BE->CA EA->CA CB->CA CB->CA CC->CA CD->CA BC->CA
## [41] BF->CA CE->CA AC->AD BD->BE AE->DF CB->DF AC->DF AA->DD AA->DD AA->DD
## [51] CD->DD AA->DD EE->DD CD->DD DB->AA AA->FC BE->CC EF->FD CF->FE BB->DD
## [61] CD->DD BA->AB CD->EC BE->EE CE->CC CD->CC ED->CC BB->CC BE->CE DD->CE
## [71] AC->CD ED->CD FF->CD AC->CD DD->CD DD->CD AE->GA AE->GA AE->GA AE->GA
## + ... omitted several edges
```

hint 14 : Check number of vertices, edges, degree of “net” and interpret the carefully

```
V(net)
```

```
## + 52/52 vertices, named, from d0826d2:
##  [1] AA AB AF DD CD BA CB CC BC ED AE CA EB BF BB AC DC BD DB CF DF BE EA CE EE
## [26] EF FF FD GB GC GD AD KA KF LC DA EC FA FB DE FC FE GA GE KB KC KD KE LB LA
## [51] LD LE
```

- “AA”, “AB”, “AF”, etc., are some of the names of the vertices in the graph.
- “52/52 vertices” suggests that there are a total of 52 vertices in the graph, and each vertex is named uniquely.

```
E(net)
```

```
## + 290/290 edges from d0826d2 (vertex names):
##   [1] AA->DD AB->DD AF->BA DD->DA CD->EC DD->CE CD->FA CD->CC BA->AF CB->CA
##  [11] CC->CA CD->CA BC->CA DD->DA ED->AD AE->AC AB->BA CD->EC CA->CC EB->CC
##  [21] BF->CE BB->CD AC->AE CC->FB DC->BB BD->CF DB->DA DD->DA DB->DD BC->AF
##  [31] CF->DE DF->BF CB->CA BE->CA EA->CA CB->CA CB->CA CC->CA CD->CA BC->CA
##  [41] BF->CA CE->CA AC->AD BD->BE AE->DF CB->DF AC->DF AA->DD AA->DD AA->DD
##  [51] CD->DD AA->DD EE->DD CD->DD DB->AA AA->FC BE->CC EF->FD CF->FE BB->DD
##  [61] CD->DD BA->AB CD->EC BE->EE CE->CC CD->CC ED->CC BB->CC BE->CE DD->CE
##  [71] AC->CD ED->CD FF->CD AC->CD DD->CD DD->CD AE->GA AE->GA AE->GA AE->GA
##  [81] BA->ED BE->ED EB->ED CD->ED FD->EF FD->EF CD->BB BF->BB BC->BB BB->CF
##  [91] AE->AC DD->DA BE->CA BE->CA CB->CA CB->CA CC->CA BE->CC BE->CC DB->DD
## + ... omitted several edges
```

- The output indicates that there are 290 edges in the graph, each represented by a pair of vertices. Each pair of vertices indicates a directed edge from the first vertex to the second vertex.

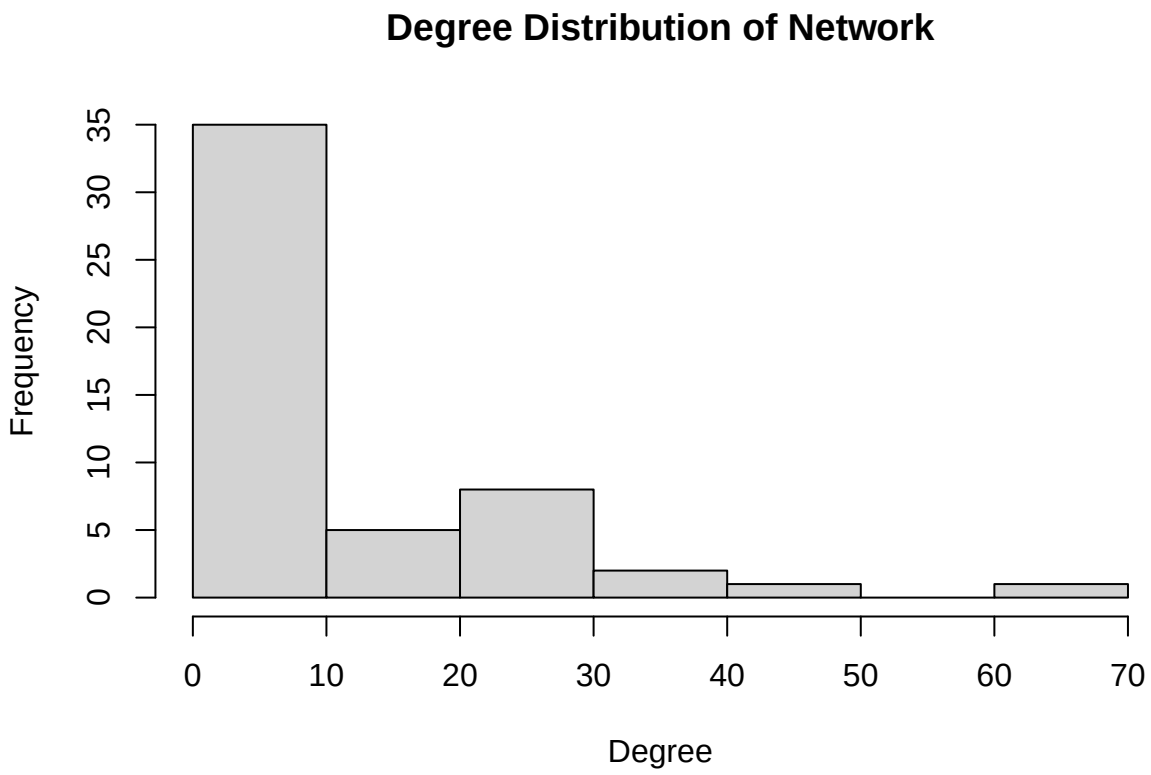
```
degree(net)
```

```
## AA AB AF DD CD BA CB CC BC ED AE CA EB BF BB AC DC BD DB CF DF BE EA CE EE EF
## 18  9 23 36 40 26 24 50 21 27 15 62  7 12 23 27  2  4  8 12 23 20  8 10  6  8
## FF FD GB GC GD AD KA KF LC DA EC FA FB DE FC FE GA GE KB KC KD KE LB LA LD LE
##  1  8  1  1  1  9  3  3  1  7  3  1  1  2  1  2  5  1  1  1  1  1  1  1  1  1
```

- The degree of a vertex provides information about its importance and connectivity within * * the graph.
- For example:
- "AA" has a degree of 18, indicating that there are 18 edges incident to the vertex "AA".
- "AB" has a degree of 9, indicating that there are 9 edges incident to the vertex "AB".

hint 15: Get histogram of net degree and interpret it carefully

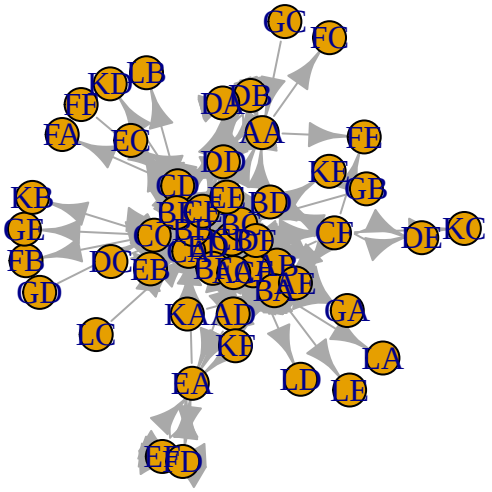
```
degree_values <- degree(net)
hist(degree_values, xlab = "Degree", ylab = "Frequency", main = "Degree Distribution of Network")
```



- Create a histogram of degree distribution

hint 16: Get network diagram of "net" and interpret it carefully

```
set.seed(34)
plot(net)
```



- The plot visualize the vertices and edges of the graph, depending on the plotting method used for the specific graph object.