STA 135 HW 1

Ishita Dutta

3/30/2022

1.6)

a)

```r
#reading the data
air <- read_excel("air_pollution.xlsx", col_names = FALSE, col_types = "numeric")
```

```
## New names:
## * `` -> ...1
## * `` -> ...2
## * `` -> ...3
## * `` -> ...4
## * `` -> ...5
## * ...
```

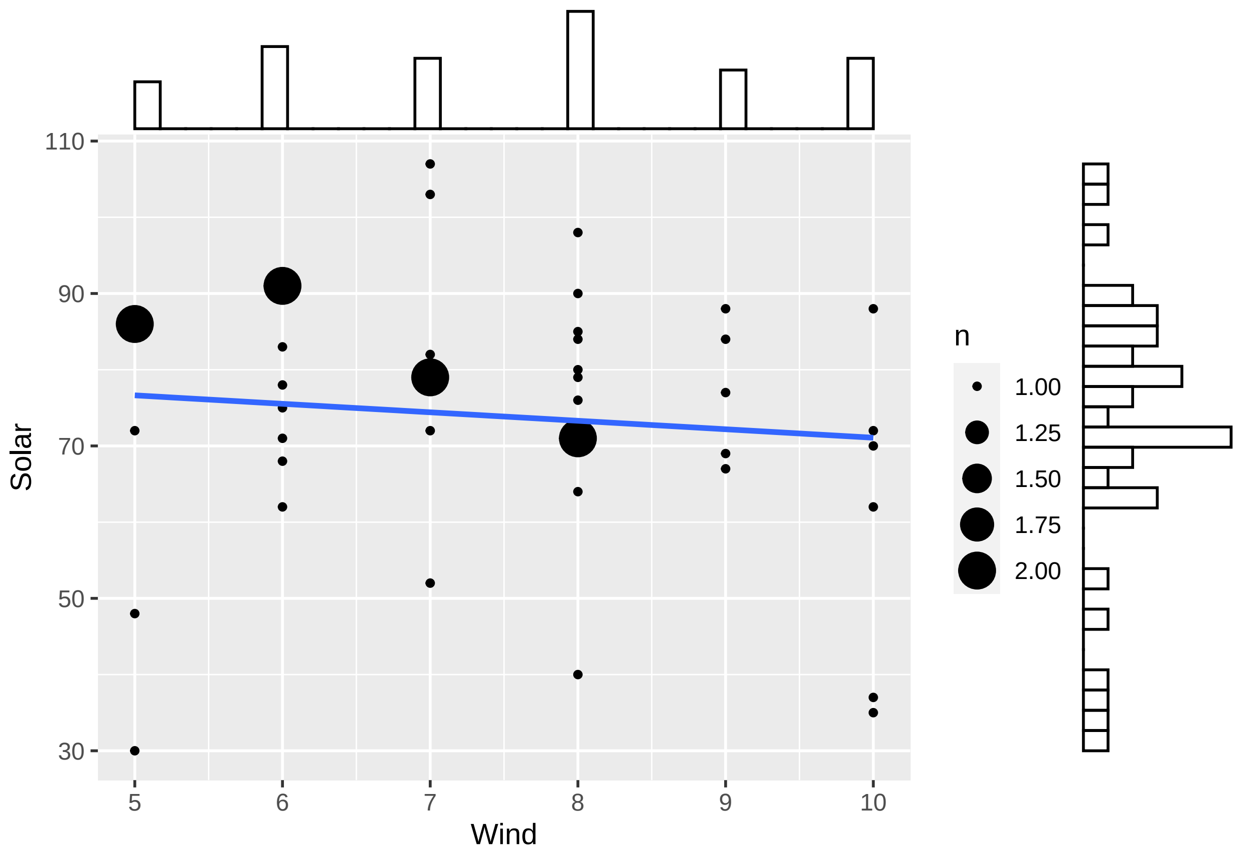
```r
colnames(air) <- c("Wind", "Solar", "CO", "NO", "NO2", "O3", "HC")
head(air)
```

```
## # A tibble: 6 x 7
##    Wind Solar    CO    NO   NO2    O3    HC
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1     8    98     7     2    12     8     2
## 2     7   107     4     3     9     5     3
## 3     7   103     4     3     5     6     3
## 4    10    88     5     2     8    15     4
## 5     6    91     4     2     8    10     3
## 6     8    90     5     2    12    12     4
```

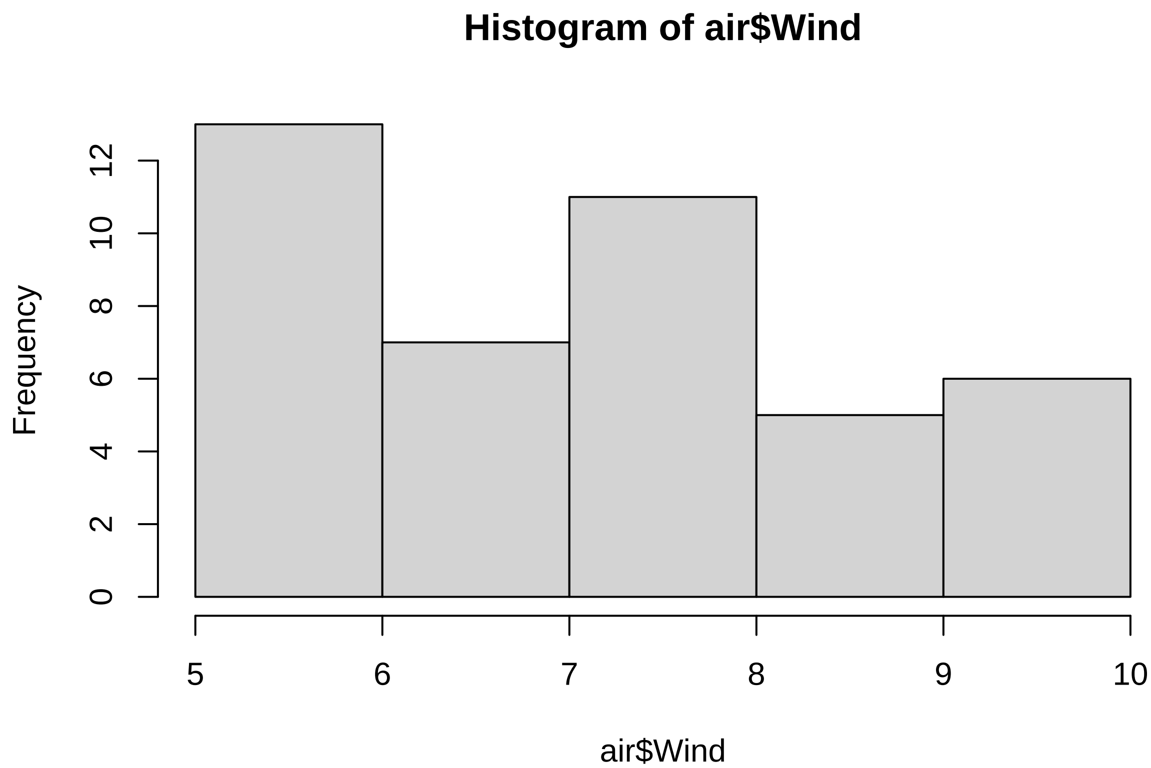
```r
# setting wind and solar values against each other as a graph
g <- ggplot(air, aes(Wind, Solar)) + geom_count() + geom_smooth(method="lm", se=F)

# plotting the marginal plot for wind and solar against each other
ggMarginal(g, type = "histogram", fill="transparent")
```

```
## `geom_smooth()` using formula 'y ~ x'
```

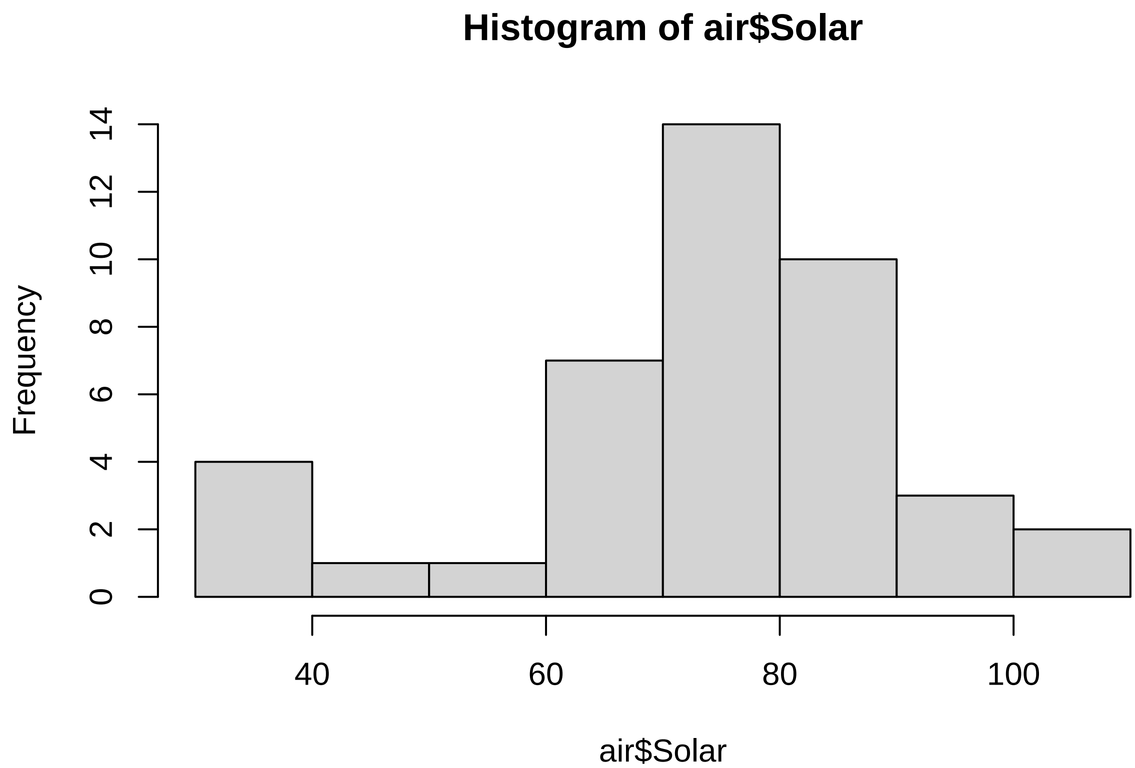
```
## `geom_smooth()` using formula 'y ~ x'
```



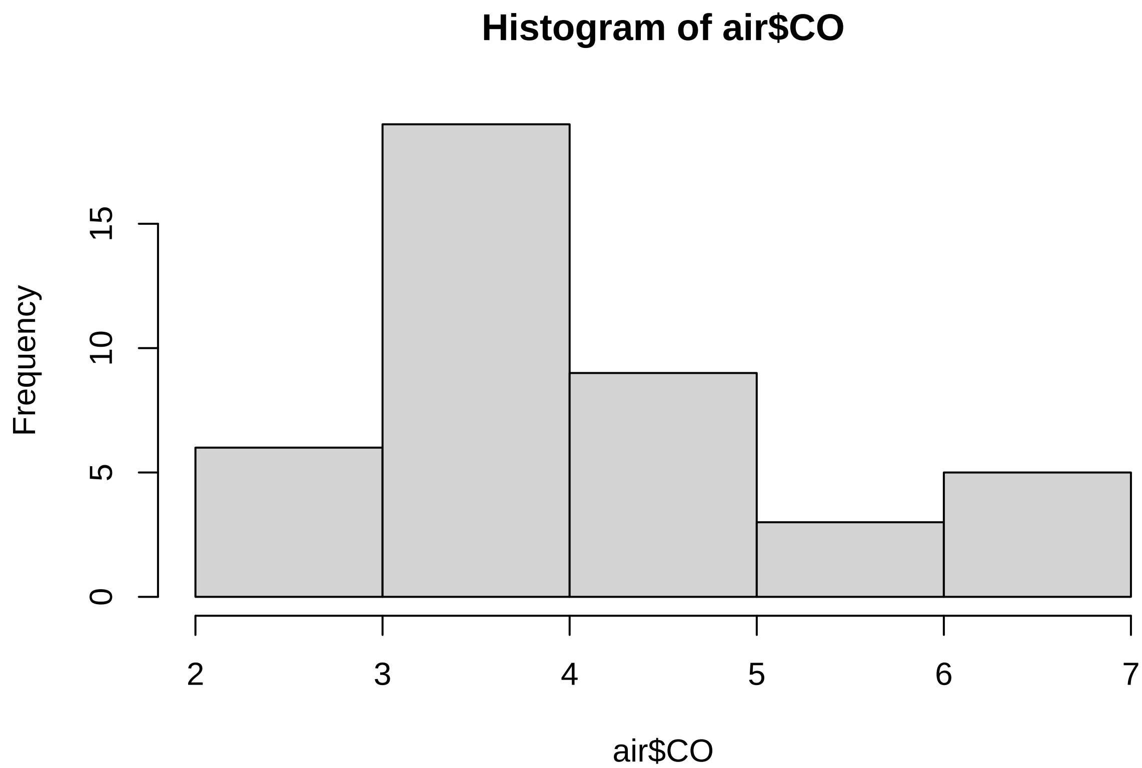

```
#histograms showing the dist of each given variable in the data set
  #You can speculate the graphs on your own :)
hist(air$Wind)
```



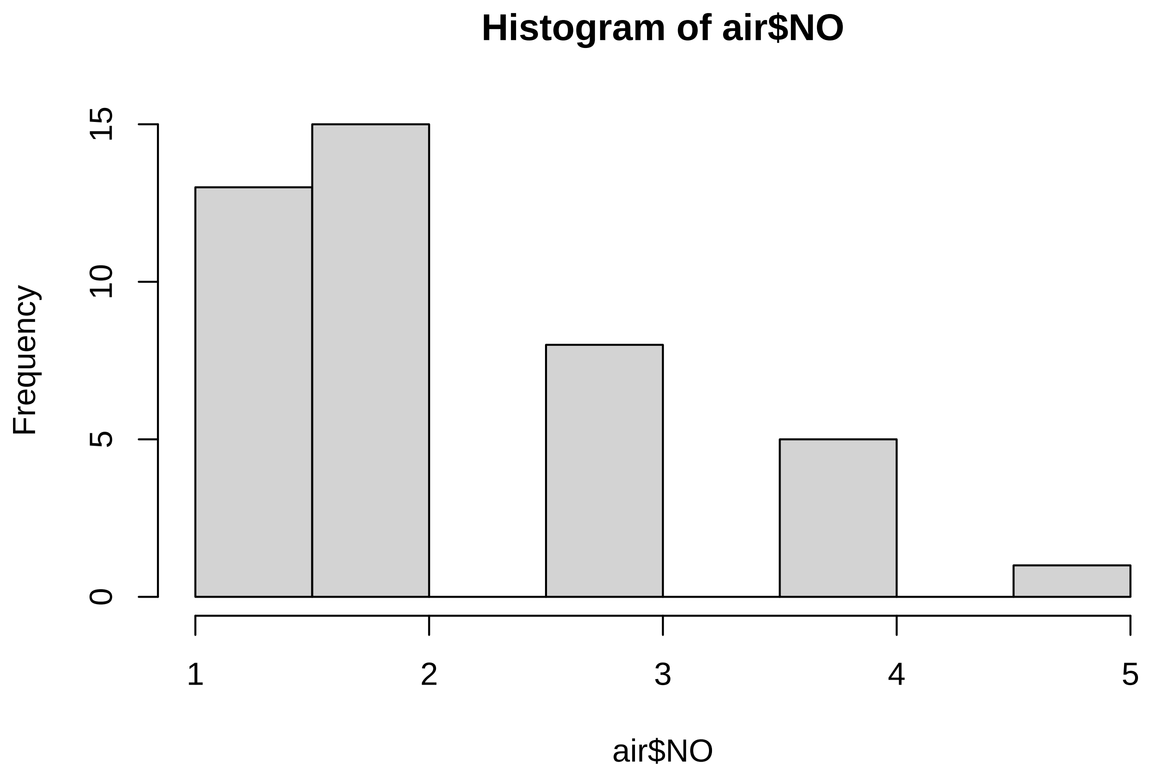

```r
hist(air$Solar)
```


Histogram of air$Solar
Frequency
0
2
4
6
8
10
12
14
40
60
80
100
air$Solar

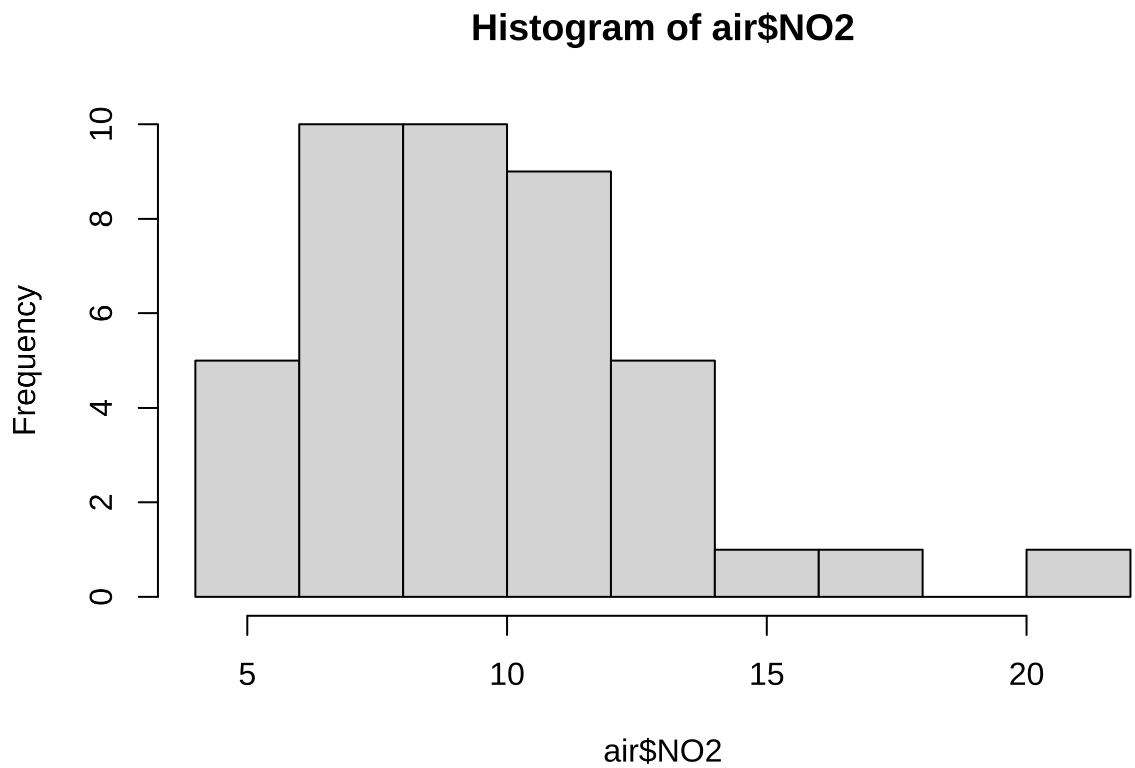
```r
hist(air$CO)
```



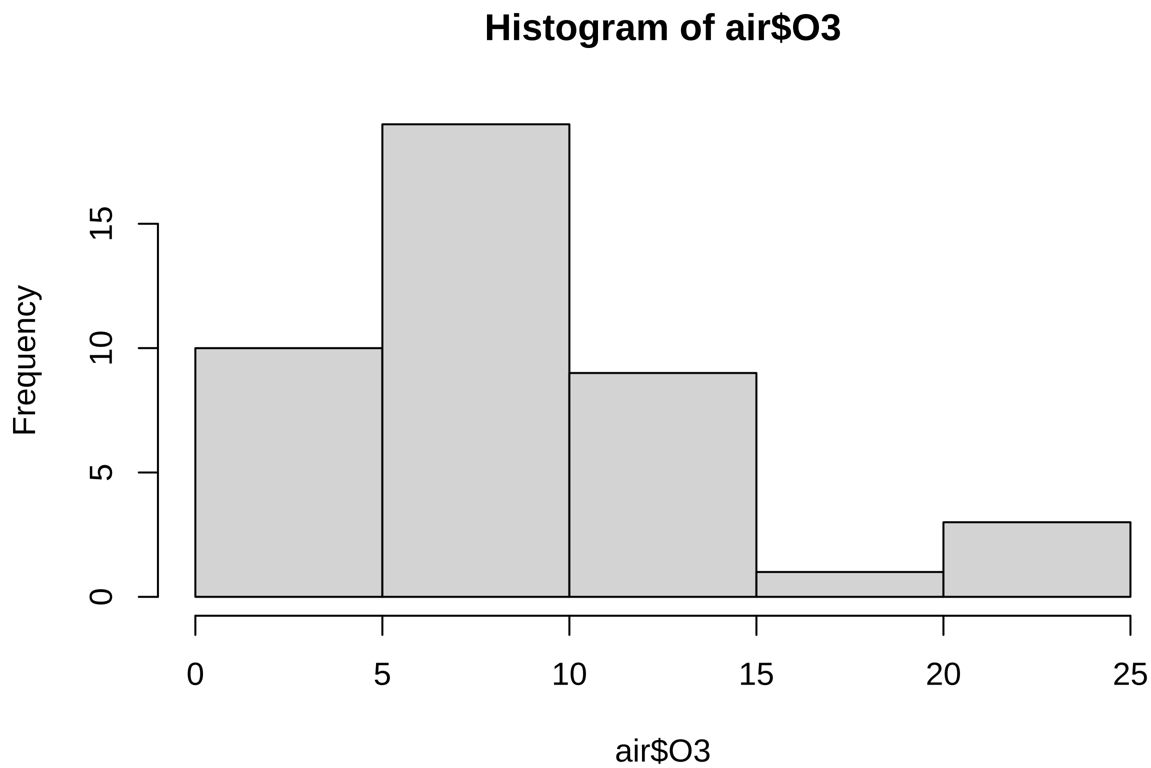

```
hist(air$NO)
```



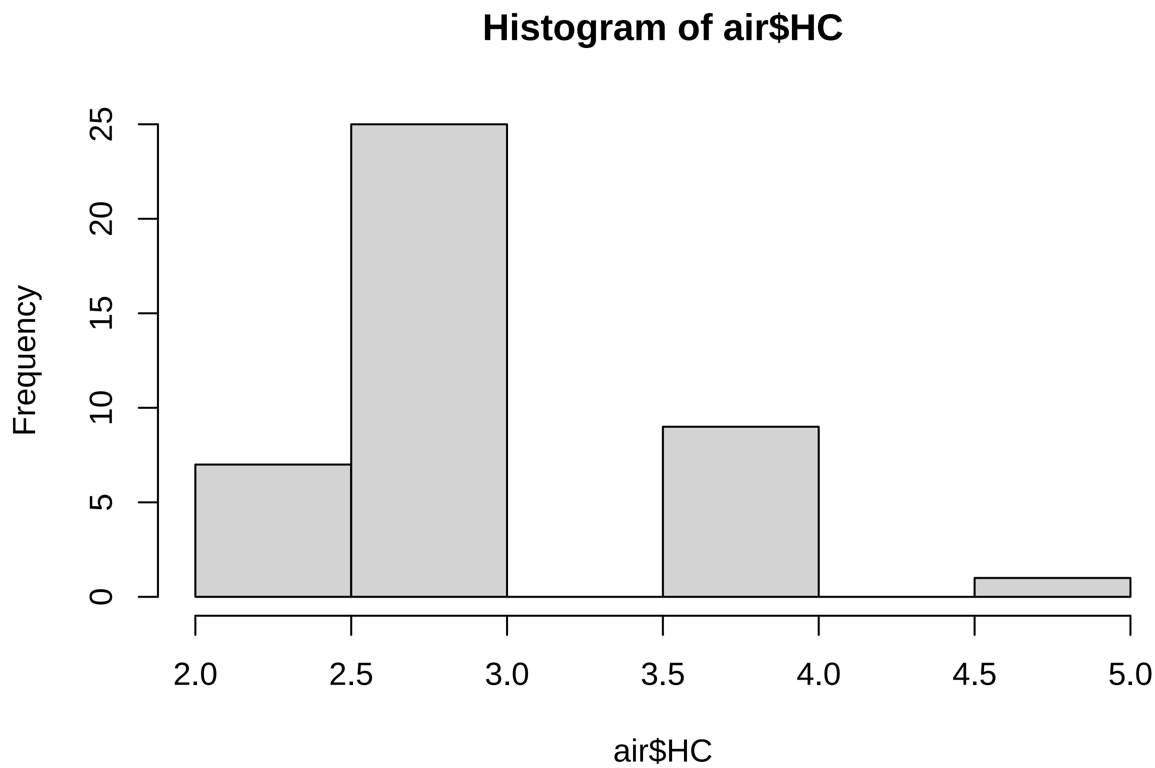

```r
hist(air$NO2)
```



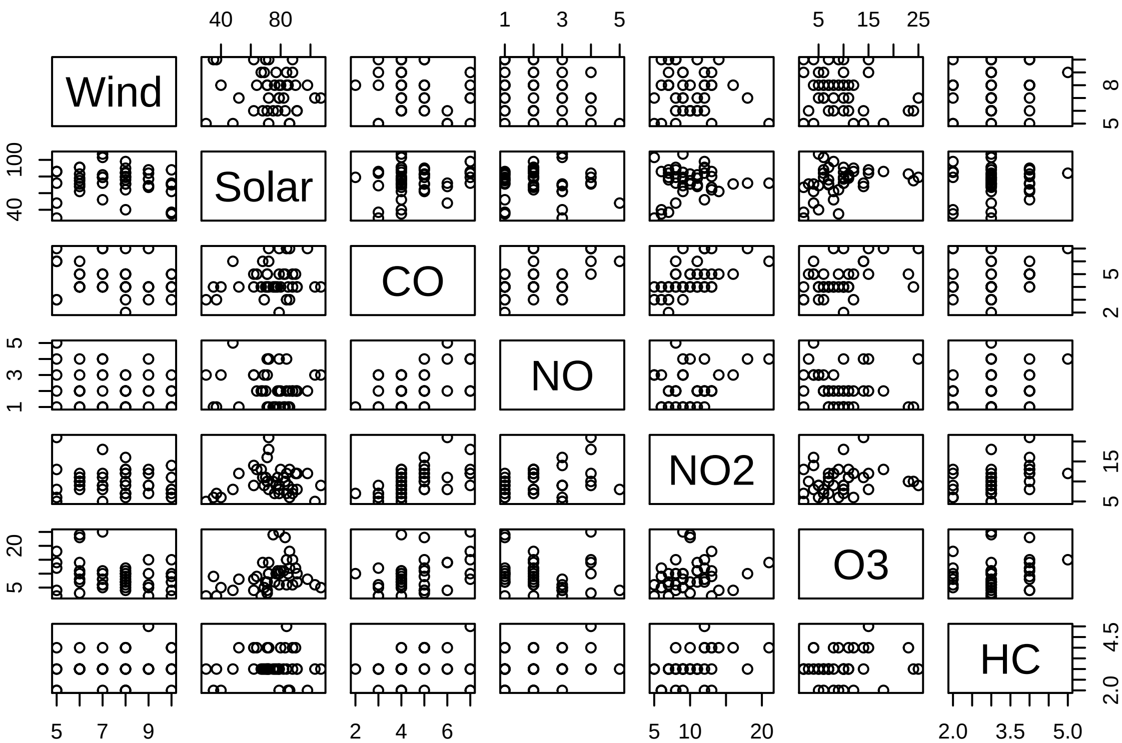

```
hist(air$O3)
```




```r
hist(air$HC)
```


Histogram of air$HC
Frequency
0
5
10
15
20
25
2.0
2.5
3.0
3.5
4.0
4.5
5.0
air$HC

```r
pairs(air)
```




```r
# number of observations
size = length(air$Wind)
```

b)

```r
# means
air = as.matrix(air)
air_mean <- colMeans(air)
cat("Means:\n", air_mean)
```

```
## Means:
##  7.5 73.85714 4.547619 2.190476 10.04762 9.404762 3.095238
```

```r
# standard deviations for air data set
air_sd = apply(air, 2, sd)
cat( "\n\nStandard Deviations: \n", air_sd * (size/(size - 1)))
```

```
##
##
## Standard Deviations:
##  1.619703 17.7582 1.263812 1.113878 3.453203 5.701587 0.7086185
```

```r
# covariance matrix
cat("\n\nCovariance Matrix:\n", cov(air))
```

```
##
##
## Covariance Matrix:
##  2.5 -2.780488 -0.3780488 -0.4634146 -0.5853659 -2.231707 0.1707317 -2.780488 300.5157 3.909408 -1.3
```

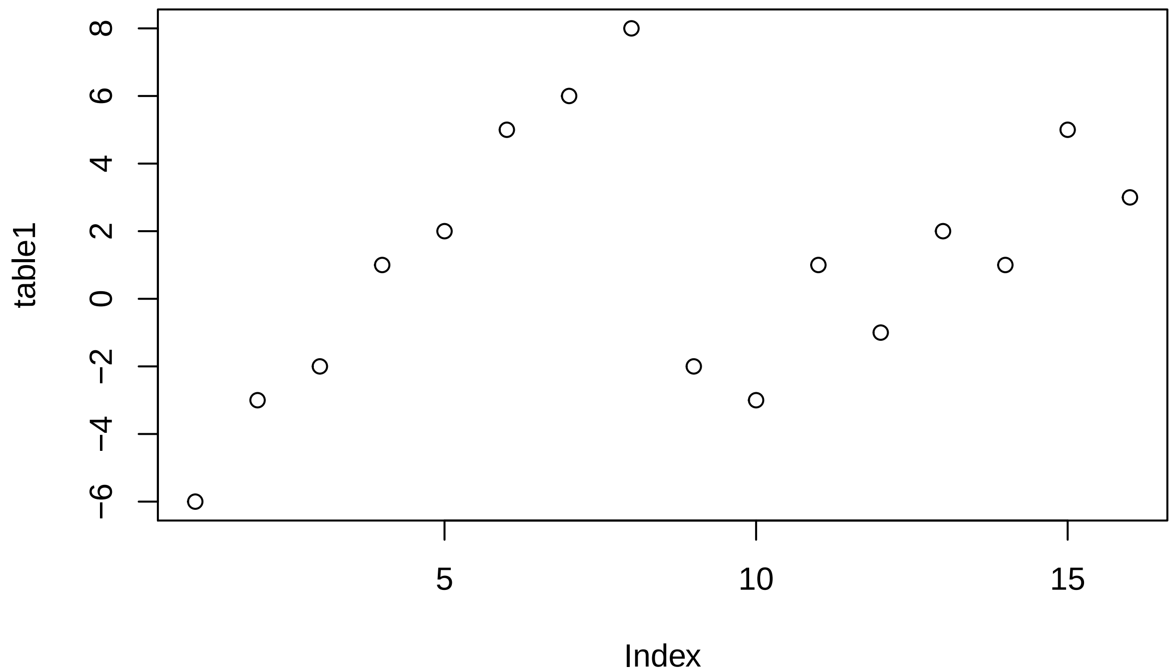
```r
# correlation matrix
cat("\n\nCorrelation Matrix:\n", cor(air))
```

```
##
##
## Correlation Matrix:
##  1 -0.1014419 -0.1938032 -0.2695426 -0.1098249 -0.2535928 0.1560979 -0.1014419 1 0.1827934 -0.073569
```

1.9)

a)

```r
n = 8
# The given table
x1 = c(-6, -3, -2, 1, 2, 5, 6, 8)
x2 = c(-2, -3, 1, -1, 2, 1, 5, 3)
table1 = c(x1, x2)

# The scatterplot
plot(table1)
```




```
# SS values
ss11 = sd(x1) * n/(n - 1)
ss12 = sd(table1) * n/(n - 1)
ss22 = sd(x2) * n/(n - 1)
cat("s 11:", ss11,
    "\ns 12:", ss12,
    "\ns 22:", ss22)
```

```
## s 11: 5.529671
## s 12: 4.326158
## s 22: 3.039104
```

b)

```
#setting the rotating factor into code
theta = (26 * pi)/180

#now rotating both of the data vectors so that we get the question
xt1 = (x1 * cos(theta)) + (x2 * sin(theta))
xt2 = (-x1 * sin(theta)) + (x2 * cos(theta))

#printing final values into a data frame
data.frame(xt1, xt2)
```

```
##          xt1        xt2
## 1 -6.2695066  0.8326388
## 2 -4.0114956 -1.3812687
## 3 -1.3592169  1.7755363
## 4  0.4604229 -1.3371652
## 5  2.6743304  0.9208458
## 6  4.9323414 -1.2930617
## 7  7.5846200  1.8637434
## 8  8.5054658 -0.8105870
```

c)

```r
# the standard deviations for the rotated data
sst11 = sd(xt1) * n/(n - 1)
sst22 = sd(xt2) * n/(n - 1)
cat("s-tilde 11:", sst11,
    "\ns-tilde 22:", sst22)
```

```
## s-tilde 11: 6.096957
## s-tilde 22: 1.62497
```

d)

```r
# new number of observations
m = 9
# new x1, x2 with added value
nx1 = c(x1, 4)
nx2 = c(x2, -2)

# recalculating the rotated data on the new observations
nxt1 = (nx1 * cos(theta)) + (nx2 * sin(theta))
nxt2 = (-nx1 * sin(theta)) + (nx2 * cos(theta))

# printing out the data with the new value and tilted
dat = data.frame(nx1, nx2, nxt1, nxt2)
colnames(dat) =  c("X1", "X2", "Xt1", "Xt2")
dat
```

```
##   X1 X2        Xt1        Xt2
## 1 -6 -2 -6.2695066  0.8326388
## 2 -3 -3 -4.0114956 -1.3812687
## 3 -2  1 -1.3592169  1.7755363
## 4  1 -1  0.4604229 -1.3371652
## 5  2  2  2.6743304  0.9208458
## 6  5  1  4.9323414 -1.2930617
## 7  6  5  7.5846200  1.8637434
## 8  8  3  8.5054658 -0.8105870
## 9  4 -2  2.7184339 -3.5510727
```

```r
# new ss values with tilt and extra val
nsst11 = sd(nxt1) * m/(m - 1)
nsst22 = sd(nxt2) * m/(m - 1)

# applying the formula in code
ntdist = sqrt(((nxt1^2)/nsst11)+((nxt2^2)/nsst22))

# actual answer on the new value with the tilt
ntdist[9]
```

```
## [1] 2.748134
```

e)

```r
# getting the ss values with the added values
ntable1 = c(nx1, nx2)
nss11 = sd(nx1) * m/(m - 1)
nss12 = sd(ntable1) * m/(m - 1)
nss22 = sd(nx2) * m/(m - 1)

# setup for getting the 1-19 formula into code...
ct2 = ((cos(theta))^2)
st2 = ((sin(theta))^2)
cs2 = (2 * sin(theta) * cos(theta))

# making the denominators for formula 1-19 understandable
d1 = (nss11 * ct2) + (nss12 * cs2) + (nss22 * st2)
d2 = (nss22 * ct2) + (nss12 * cs2) + (nss11 * st2)

# getting the dist formula a11, a12, a22 values
na11 = (ct2/(d1)) + (st2/(d2))
na22 = (st2/(d1) + (ct2/(d2)))
na12 = ((cs2 / 2)/(d1)) + ((cs2 / 2)/(d2))

# formula 1-19 into code
ndist = sqrt((na11 * (nx1^2))+(na12 * nx1 * nx2)+(na22 * (nx2^2)))

# the actual answer.
ndist[9]
```

```
## [1] 1.334968
```

1.18)

```r
# This is from the national track records for women data set, table 1.9
```