STA 135 HW 1

Ishita Dutta, Raina Joby

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
## * ...
```

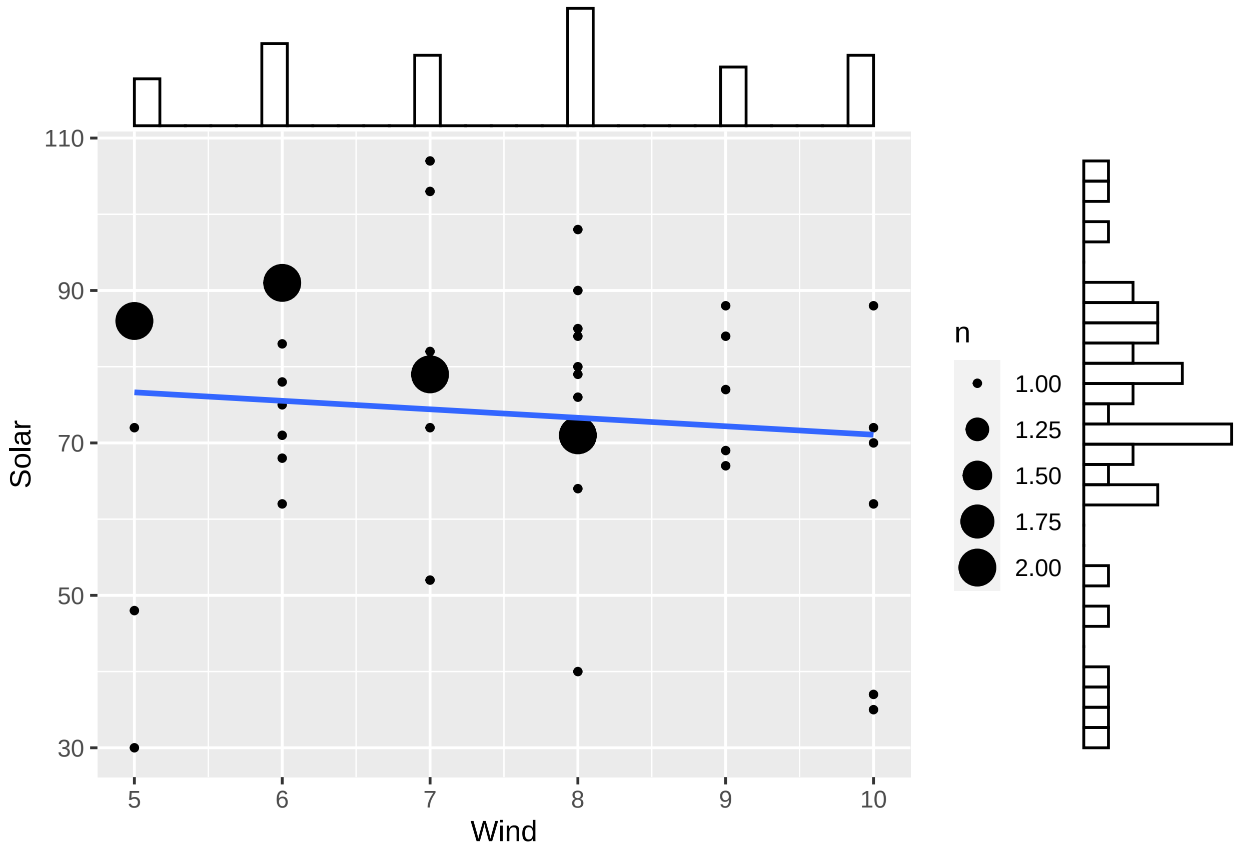
```r
colnames(air) <- c("Wind", "Solar", "CO", "NO", "NO2", "O3", "HC")
head(air)
```

```
## # 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
```

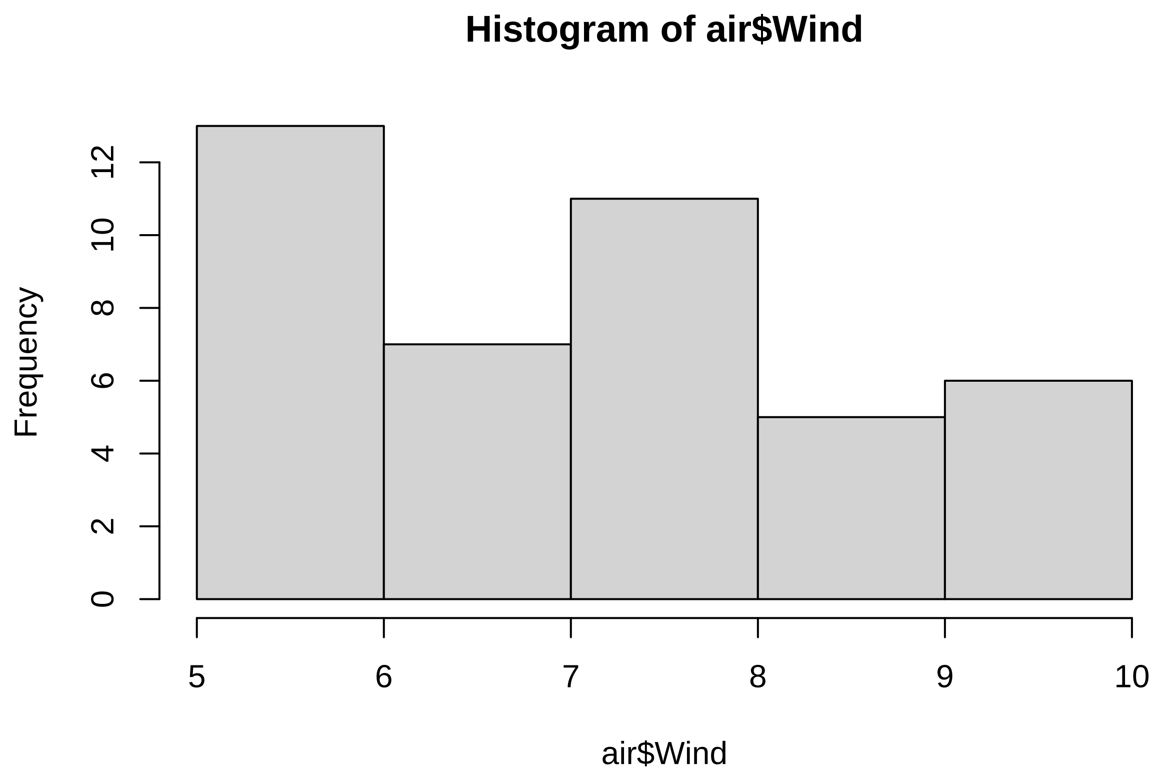
```r
# setting wind and solar values against each other as a graph
g <- ggplot(air, aes(Wind, Solar)) + geom_count() + geom_smooth(method="lm", se=F)

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

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

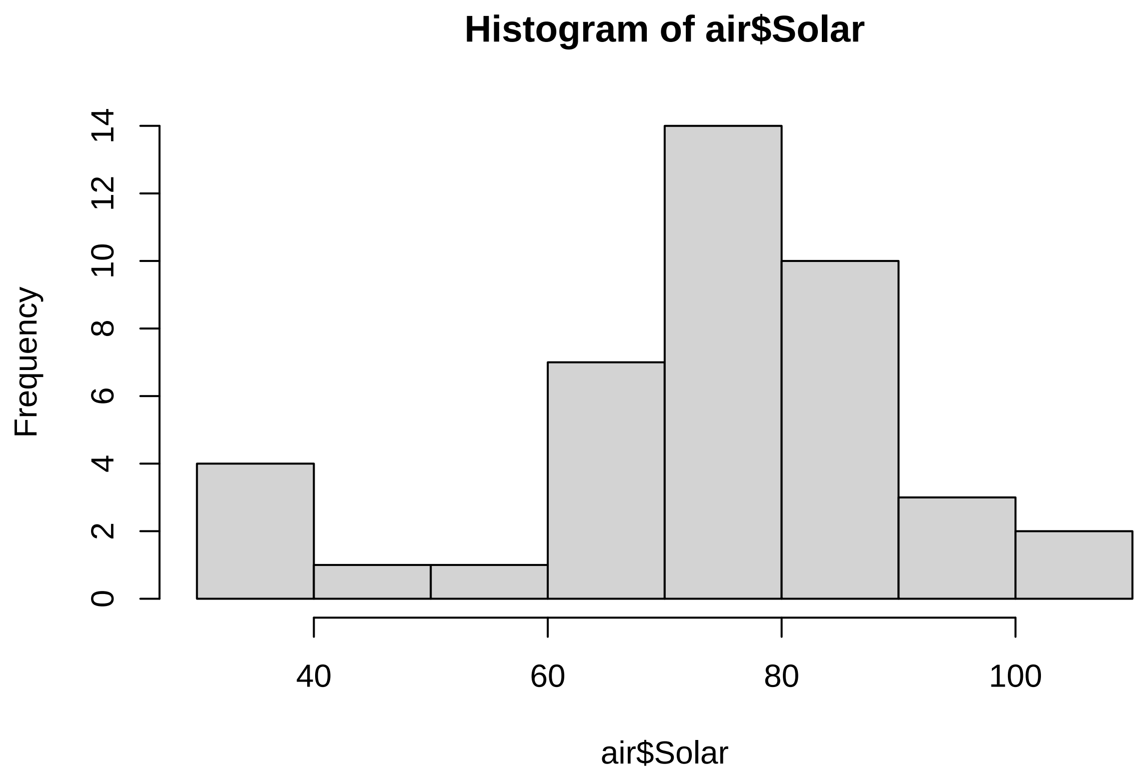
```
## `geom_smooth()` using formula 'y ~ x'
```



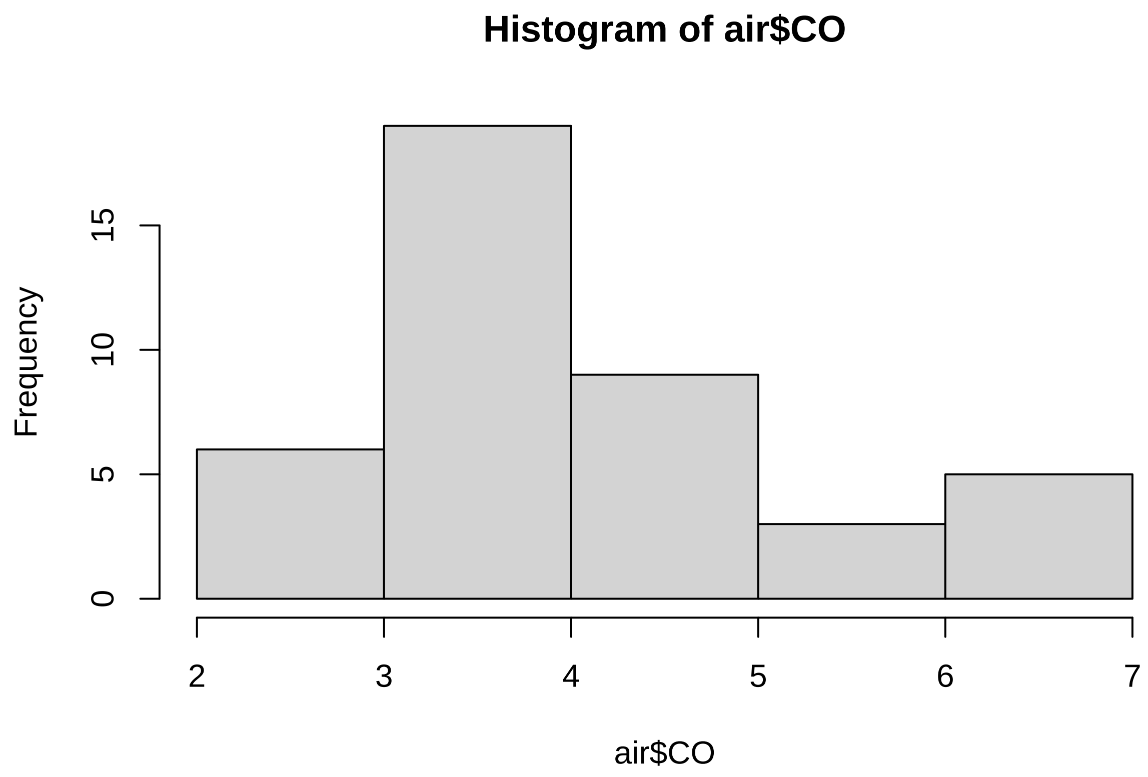
```r
#histograms showing the dist of each given variable in the data set
  #You can speculate the graphs on your own :)
hist(air$Wind)
```



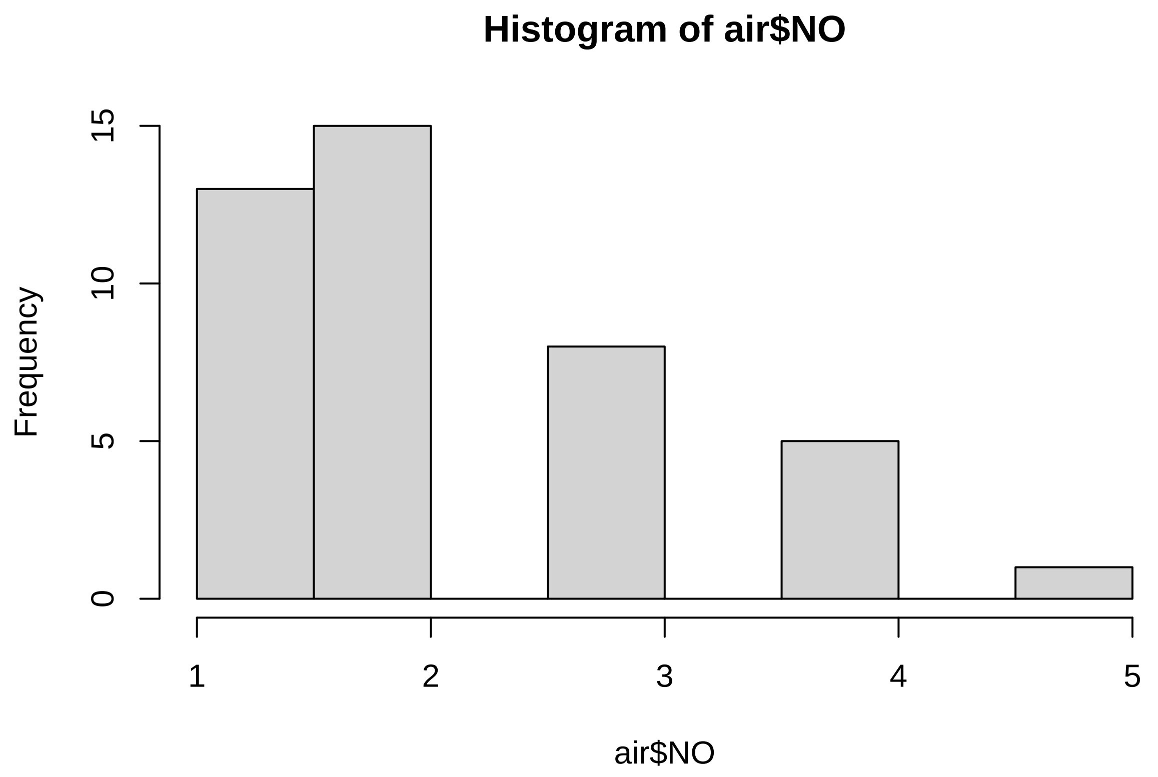
```r
hist(air$Solar)
```


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

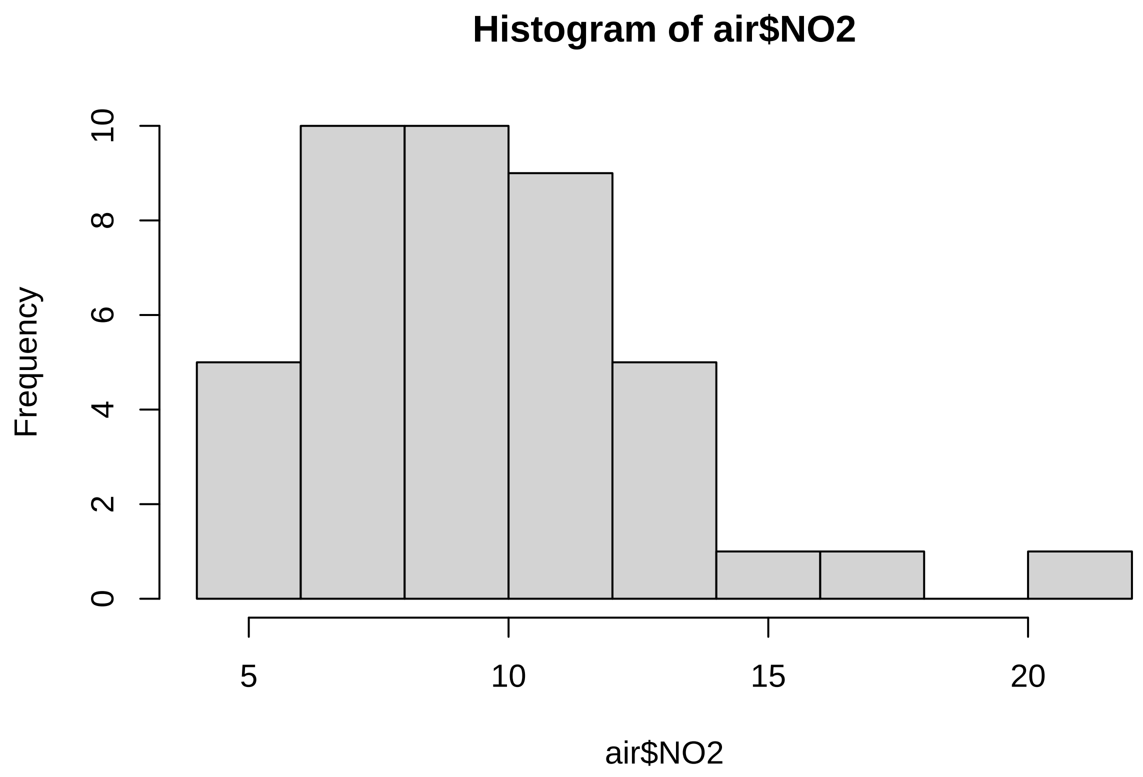
```r
hist(air$CO)
```



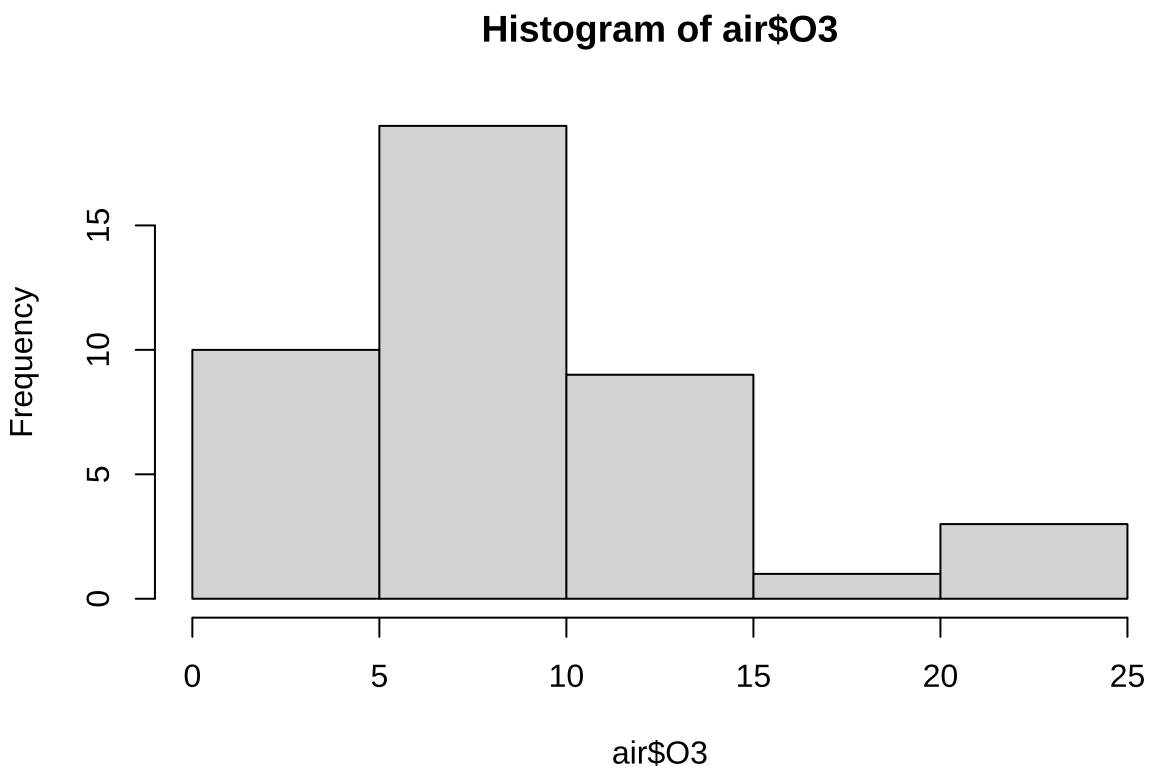
```r
hist(air$NO)
```



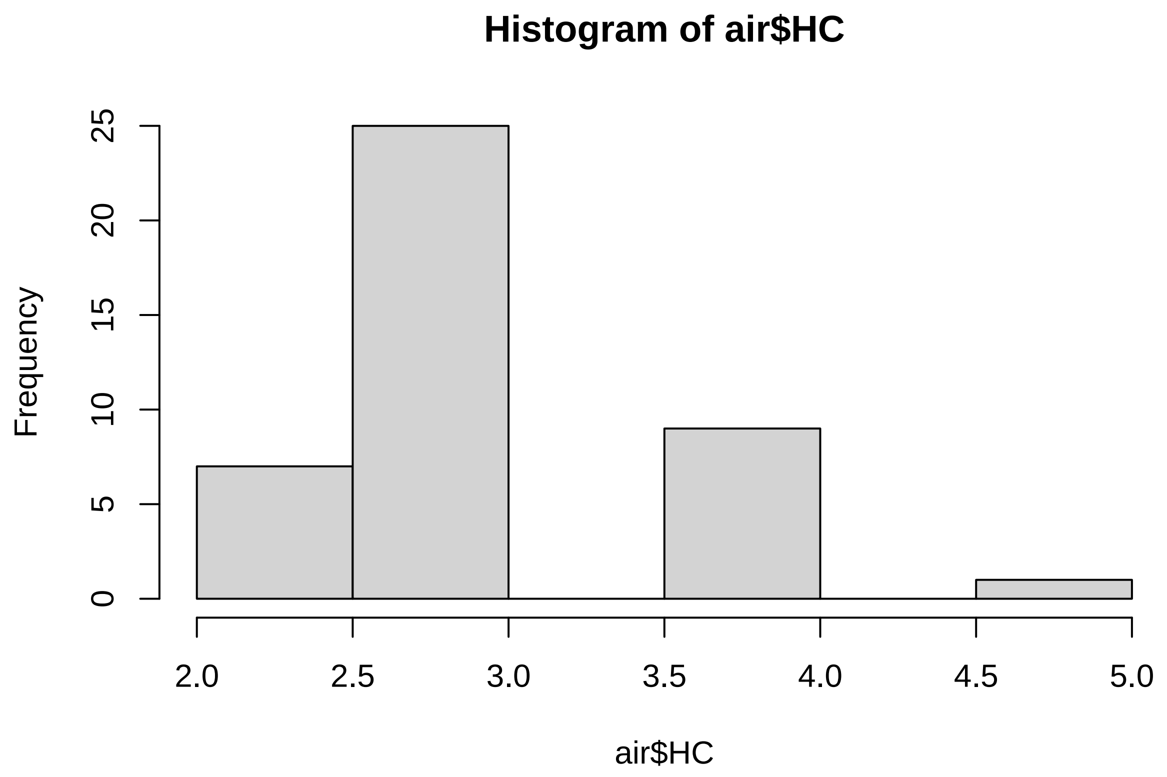
```r
hist(air$NO2)
```



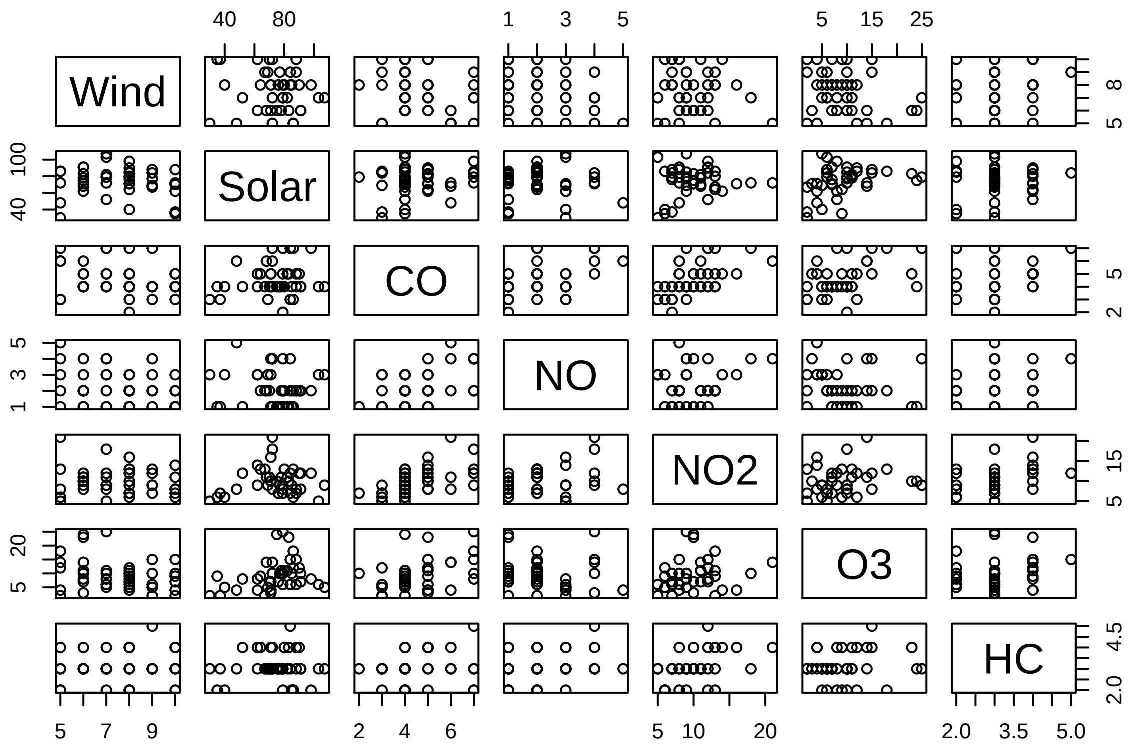
```
hist(air$O3)
```



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


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

```r
pairs(air)
```



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

b)

```
# 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
```

```
# 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")
```

```
##
##
## Covariance Matrix:
```

```r
data.frame(cov(air))
```

```
##             Wind       Solar         CO         NO        NO2         O3
## Wind   2.5000000  -2.7804878 -0.3780488 -0.4634146 -0.5853659 -2.2317073
## Solar -2.7804878 300.5156794  3.9094077 -1.3867596  6.7630662 30.7909408
## CO    -0.3780488   3.9094077  1.5220674  0.6736353  2.3147503  2.8217189
## NO    -0.4634146  -1.3867596  0.6736353  1.1823461  1.0882695 -0.8106852
## NO2   -0.5853659   6.7630662  2.3147503  1.0882695 11.3635308  3.1265970
## O3    -2.2317073  30.7909408  2.8217189 -0.8106852  3.1265970 30.9785134
## HC     0.1707317   0.6236934  0.1416957  0.1765389  1.0441347  0.5946574
##              HC
## Wind  0.1707317
## Solar 0.6236934
## CO    0.1416957
## NO    0.1765389
## NO2   1.0441347
## O3    0.5946574
## HC    0.4785134
```

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

```
##
##
## Correlation Matrix:
```

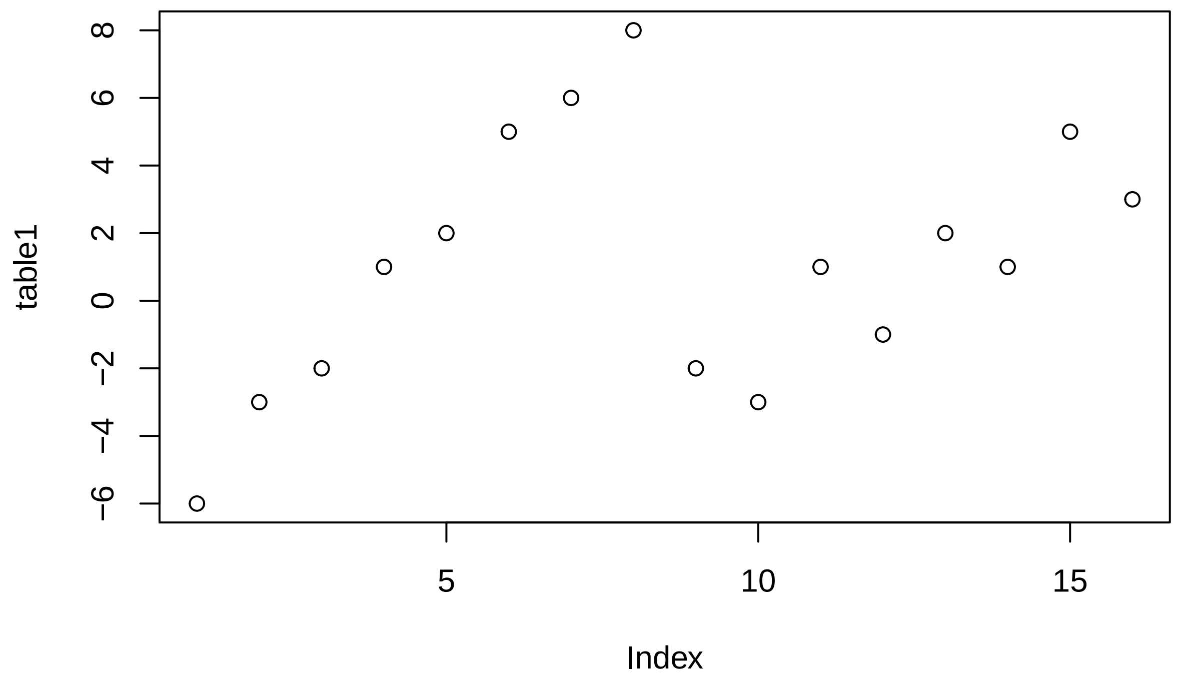
```r
data.frame(cor(air))
```

```
##             Wind       Solar         CO          NO        NO2         O3
## Wind   1.0000000 -0.10144191 -0.1938032 -0.26954261 -0.1098249 -0.2535928
## Solar -0.1014419  1.00000000  0.1827934 -0.07356907  0.1157320  0.3191237
## CO    -0.1938032  0.18279338  1.0000000  0.50215246  0.5565838  0.4109288
## NO    -0.2695426 -0.07356907  0.5021525  1.00000000  0.2968981 -0.1339521
## NO2   -0.1098249  0.11573199  0.5565838  0.29689814  1.0000000  0.1666422
## O3    -0.2535928  0.31912373  0.4109288 -0.13395214  0.1666422  1.0000000
## HC     0.1560979  0.05201044  0.1660323  0.23470432  0.4477678  0.1544506
##               HC
## Wind  0.15609793
## Solar 0.05201044
## CO    0.16603235
## NO    0.23470432
## NO2   0.44776780
## O3    0.15445056
## HC    1.00000000
```

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)
```



```r
# 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)

```r
#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
```

```
# 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)

```
# 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)

```
# This is from the national track records for women data set, table 1.9
## read the table
track = read.table("T1-9.dat", sep="\t")
## set the column names
colnames(track) = c("Country", "s100m", "s200m", "s400m", "min800m", "min1500m", "min3000m", "minMaratho
## print first five vals for checking
head(track)
```

```
##   Country s100m s200m s400m min800m min1500m min3000m minMarathon
## 1     ARG 11.57 22.94 52.50    2.05     4.25     9.19      150.32
## 2     AUS 11.12 22.23 48.63    1.98     4.02     8.63      143.51
## 3     AUT 11.15 22.70 50.62    1.94     4.05     8.78      154.35
## 4     BEL 11.14 22.48 51.45    1.97     4.08     8.82      143.05
## 5     BER 11.46 23.05 53.30    2.07     4.29     9.81      174.18
## 6     BRA 11.17 22.60 50.62    1.97     4.17     9.04      147.41
```

```
# getting the converted values
track$mps100 = 100/track$s100m
track$mps200 = 200/track$s200m
track$mps400 = 400/track$s400m
track$mps800 = 800/(track$min800m * 60)
track$mps1500 = 1500/(track$min1500m * 60)
track$mps3000 = 3000/(track$min3000m * 60)
track$mpsmarathon = 42195/(track$minMarathon * 60)

head(track)
```

```
##   Country s100m s200m s400m min800m min1500m min3000m minMarathon   mps100
## 1     ARG 11.57 22.94 52.50    2.05     4.25     9.19      150.32 8.643042
## 2     AUS 11.12 22.23 48.63    1.98     4.02     8.63      143.51 8.992806
## 3     AUT 11.15 22.70 50.62    1.94     4.05     8.78      154.35 8.968610
## 4     BEL 11.14 22.48 51.45    1.97     4.08     8.82      143.05 8.976661
## 5     BER 11.46 23.05 53.30    2.07     4.29     9.81      174.18 8.726003
## 6     BRA 11.17 22.60 50.62    1.97     4.17     9.04      147.41 8.952551
##     mps200   mps400   mps800  mps1500  mps3000 mpsmarathon
## 1 8.718396 7.619048 6.504065 5.882353 5.440696    4.678353
## 2 8.996851 8.225375 6.734007 6.218905 5.793743    4.900355
## 3 8.810573 7.902015 6.872852 6.172840 5.694761    4.556203
## 4 8.896797 7.774538 6.768190 6.127451 5.668934    4.916113
## 5 8.676790 7.504690 6.441224 5.827506 5.096840    4.037490
## 6 8.849558 7.902015 6.768190 5.995204 5.530973    4.770708
```

```
size = length(track$s100m)
# means
##air = as.matrix(air)
track2 = data.frame(track$s100m, track$s200m, track$s400m, track$min800m, track$min1500m, track$min3000
track2 = as.matrix(track2)

data.frame(colMeans(track2))
```

```
##                     colMeans.track2.
## track.s100m                11.357778
## track.s200m                23.118519
## track.s400m                51.989074
## track.min800m               2.022407
## track.min1500m              4.189444
## track.min3000m              9.080741
## track.minMarathon         153.619259
## track.mps100                8.814772
## track.mps200                8.664408
## track.mps400                7.712067
## track.mps800                6.604214
## track.mps1500               5.989687
## track.mps3000               5.542701
## track.mpsmarathon           4.620264
```

```r
data.frame(cor(track2))
```

```
##                     track.s100m track.s200m track.s400m track.min800m
## track.s100m           1.0000000   0.9410886   0.8707802     0.8091758
## track.s200m           0.9410886   1.0000000   0.9088096     0.8198258
## track.s400m           0.8707802   0.9088096   1.0000000     0.8057904
## track.min800m         0.8091758   0.8198258   0.8057904     1.0000000
## track.min1500m        0.7815510   0.8013282   0.7197996     0.9050509
## track.min3000m        0.7278784   0.7318546   0.6737991     0.8665732
## track.minMarathon     0.6689597   0.6799537   0.6769384     0.8539900
## track.mps100         -0.9985740  -0.9357999  -0.8638768    -0.7977551
## track.mps200         -0.9408909  -0.9984055  -0.9049866    -0.8150717
## track.mps400         -0.8694178  -0.9060984  -0.9968462    -0.8007075
## track.mps800         -0.8067167  -0.8181022  -0.8061430    -0.9982440
## track.mps1500        -0.7887776  -0.8131821  -0.7368626    -0.9116531
## track.mps3000        -0.7430908  -0.7508157  -0.7031975    -0.8849172
## track.mpsmarathon    -0.6740219  -0.6844870  -0.6849276    -0.8635155
##                     track.min1500m track.min3000m track.minMarathon track.mps100
## track.s100m              0.7815510      0.7278784         0.6689597   -0.9985740
## track.s200m              0.8013282      0.7318546         0.6799537   -0.9357999
## track.s400m              0.7197996      0.6737991         0.6769384   -0.8638768
## track.min800m            0.9050509      0.8665732         0.8539900   -0.7977551
## track.min1500m           1.0000000      0.9733801         0.7905565   -0.7662085
## track.min3000m           0.9733801      1.0000000         0.7987302   -0.7096688
## track.minMarathon        0.7905565      0.7987302         1.0000000   -0.6520648
## track.mps100            -0.7662085     -0.7096688        -0.6520648    1.0000000
## track.mps200            -0.7905857     -0.7177712        -0.6662173    0.9383028
## track.mps400            -0.7113733     -0.6623060        -0.6588348    0.8655248
## track.mps800            -0.8947562     -0.8511288        -0.8368936    0.7974687
## track.mps1500           -0.9934981     -0.9542162        -0.7939339    0.7764777
## track.mps3000           -0.9730416     -0.9868079        -0.8207653    0.7287297
## track.mpsmarathon       -0.8149283     -0.8226685        -0.9883670    0.6601124
##                     track.mps200 track.mps400 track.mps800 track.mps1500
## track.s100m           -0.9408909   -0.8694178   -0.8067167    -0.7887776
## track.s200m           -0.9984055   -0.9060984   -0.8181022    -0.8131821
## track.s400m           -0.9049866   -0.9968462   -0.8061430    -0.7368626
## track.min800m         -0.8150717   -0.8007075   -0.9982440    -0.9116531
## track.min1500m        -0.7905857   -0.7113733   -0.8947562    -0.9934981
## track.min3000m      -0.7177712   -0.6623060   -0.8511288    -0.9542162
## track.minMarathon   -0.6662173   -0.6588348   -0.8368936    -0.7939339
## track.mps100         0.9383028    0.8655248    0.7974687     0.7764777
## track.mps200         1.0000000    0.9058875    0.8159945     0.8057456
## track.mps400         0.9058875    1.0000000    0.8041737     0.7306437
## track.mps800         0.8159945    0.8041737    1.0000000     0.9060324
## track.mps1500        0.8057456    0.7306437    0.9060324     1.0000000
## track.mps3000        0.7409469    0.6944025    0.8754795     0.9718385
## track.mpsmarathon    0.6748635    0.6722005    0.8518052     0.8244153
##                    track.mps3000 track.mpsmarathon
## track.s100m           -0.7430908        -0.6740219
## track.s200m           -0.7508157        -0.6844870
## track.s400m           -0.7031975        -0.6849276
## track.min800m         -0.8849172        -0.8635155
## track.min1500m        -0.9730416        -0.8149283
## track.min3000m        -0.9868079        -0.8226685
## track.minMarathon     -0.8207653        -0.9883670
## track.mps100           0.7287297         0.6601124
## track.mps200           0.7409469         0.6748635
## track.mps400           0.6944025         0.6722005
## track.mps800           0.8754795         0.8518052
## track.mps1500          0.9718385         0.8244153
## track.mps3000          1.0000000         0.8541900
## track.mpsmarathon      0.8541900         1.0000000
```

```
data.frame(cov(track2))
```

```
##                    track.s100m track.s200m track.s400m track.min800m
## track.s100m         0.15531572   0.3445608   0.8912960   0.027703564
## track.s200m         0.34456080   0.8630883   2.1928363   0.066165898
## track.s400m         0.89129602   2.1928363   6.7454576   0.181807932
## track.min800m       0.02770356   0.0661659   0.1818079   0.007546925
## track.min1500m      0.08389119   0.2027633   0.5091768   0.021414570
## track.min3000m      0.23388281   0.5543502   1.4268158   0.061379315
## track.minMarathon   4.33417757  10.3849876  28.9037314   1.219654647
## track.mps100       -0.11841427  -0.2615934  -0.6751086  -0.020853104
## track.mps200       -0.12556674  -0.3140959  -0.7959312  -0.023977752
## track.mps400       -0.12718717  -0.3124712  -0.9610388  -0.025820579
## track.mps800       -0.08620648  -0.2060850  -0.5677132  -0.023514344
## track.mps1500      -0.10939403  -0.2658562  -0.6734783  -0.027870580
## track.mps3000      -0.12306248  -0.2931143  -0.7674659  -0.032304552
## track.mpsmarathon -0.10845956  -0.2596444  -0.7263342  -0.030629598
##                    track.min1500m track.min3000m track.minMarathon track.mps100
## track.s100m            0.08389119     0.23388281          4.334178  -0.11841427
## track.s200m            0.20276331     0.55435017         10.384988  -0.26159336
## track.s400m            0.50917683     1.42681579         28.903731  -0.67510856
## track.min800m          0.02141457     0.06137932          1.219655  -0.02085310
## track.min1500m         0.07418270     0.21615514          3.539837  -0.06279345
## track.min3000m         0.21615514     0.66475793         10.706091  -0.17410191
## track.minMarathon      3.53983732    10.70609113        270.270150  -3.22556542
## track.mps100          -0.06279345    -0.17410191         -3.225565   0.09053826
## track.mps200          -0.07291685    -0.19817343         -3.708878   0.09560635
## track.mps400          -0.07192104    -0.20044605         -4.020524   0.09667244
## track.mps800          -0.06607957    -0.18816472         -3.730615   0.06506402
## track.mps1500       -0.09522472    -0.27378516         -4.593193   0.08221980
## track.mps3000       -0.11136758    -0.33809634         -5.670144   0.09214221
## track.mpsmarathon   -0.09062685    -0.27386894         -6.634428   0.08109987
##                   track.mps200 track.mps400 track.mps800 track.mps1500
## track.s100m        -0.12556674  -0.12718717  -0.08620648   -0.10939403
## track.s200m        -0.31409588  -0.31247122  -0.20608498   -0.26585620
## track.s400m        -0.79593115  -0.96103875  -0.56771316   -0.67347827
## track.min800m      -0.02397775  -0.02582058  -0.02351434   -0.02787058
## track.min1500m     -0.07291685  -0.07192104  -0.06607957   -0.09522472
## track.min3000m     -0.19817343  -0.20044605  -0.18816472   -0.27378516
## track.minMarathon  -3.70887815  -4.02052374  -3.73061490   -4.59319332
## track.mps100        0.09560635   0.09667244   0.06506402    0.08221980
## track.mps200        0.11467144   0.11386990   0.07492487    0.09601895
## track.mps400        0.11386990   0.13778886   0.08094090    0.09544299
## track.mps800        0.07492487   0.08094090   0.07352284    0.08645423
## track.mps1500       0.09601895   0.09544299   0.08645423    0.12384050
## track.mps3000       0.10543645   0.10831645   0.09975466    0.14371481
## track.mpsmarathon   0.09331033   0.10188073   0.09430563    0.11845777
##                   track.mps3000 track.mpsmarathon
## track.s100m         -0.12306248       -0.10845956
## track.s200m         -0.29311431       -0.25964440
## track.s400m         -0.76746591       -0.72633425
## track.min800m       -0.03230455       -0.03062960
## track.min1500m      -0.11136758       -0.09062685
## track.min3000m      -0.33809634       -0.27386894
## track.minMarathon   -5.67014432       -6.63442756
## track.mps100         0.09214221        0.08109987
## track.mps200         0.10543645        0.09331033
## track.mps400         0.10831645        0.10188073
## track.mps800         0.09975466        0.09430563
## track.mps1500        0.14371481        0.11845777
## track.mps3000        0.17658433        0.14656043
## track.mpsmarathon    0.14656043        0.16671409
```

Solution: The correlation magnitude gets smaller as the distances increase. This can be due to how a longer distance would require a higher amount of endurance on the runner's part.