Hanjiao Qiu
CSCIE63 Big Data Analytics
Assignment 01

Assignment 01 Solution

Download and install the latest version of R and R Studio.

Problem 1

Create a vector V with 8 elements (7,2,1,0,3,-1,-3,4).

- Transform that vector into a rectangular matrix A of dimensions 4X2 (4- rows, 2-columns).
- Create a matrix transpose to the above matrix A. Call that matrix AT.
- Calculate matrix products: A*AT and AT*A. Present the results. What are the dimensions of those two product matrices.
- Square matrixes sometimes have an inverse matrix. Try calculating inverse matrices (or matrixes, if you prefer) of above matrices (matrixes) A*AT and AT*A.
- Extend the above vector V with the ninth number of value -2. Do it elegantly by concatenating two vectors (☺).
- Transform that extended vector into a 3X3 matrix B.
- Calculate the inverse matrix of matrix B. Call it Binv. Demonstrate that the product of B and Binv is the same as the product of Binv and B and is equal to what?
- Determine the eigenvectors of matrixes B.
- Construct a new matrix C which is made by using each eigenvector of matrix B as a column. Calculate the product of matrix C and matrix B and the product of matrix B and C. Is there any significance to the elements of the product matrixes.
- Transform matrix B into a matrix with names columns and named rows.
- Transformed that fully "named" matrix into a data.frame.
- Ask the object you just created what is its class().

Solution:

1. First we create a vector V with (7, 2, 1, 0, 3, -1, -3, 4).

```
> V <- c(7,2,1,0,3,-1,-3,4)
> V
[1]  7  2  1  0  3 -1 -3  4
```

2. Transform the vector into a 4x2 matrix. "byrow = True" means that the matrix is filled by rows.

```
> A <- matrix(V, nrow = 4, ncol = 2, byrow = TRUE)
> A
     [,1] [,2]
[1,]    7    2
[2,]    1    0
[3,]    3   -1
[4,]   -3    4
```

3. Create a matrix transpose to matrix A.

```
> AT <- t(A)
> AT
     [,1] [,2] [,3] [,4]
[1,]    7    1    3   -3
[2,]    2    0   -1    4
```

4. Calculate the product of A*AT and AT*A. The dimension of A*AT is 4x4. The dimension of AT*A is 2x2.

```
> A %*% AT
     [,1] [,2] [,3] [,4]
[1,]   53    7   19  -13
[2,]    7    1    3   -3
[3,]   19    3   10  -13
[4,]  -13   -3  -13   25
> dim(A %*% AT)
[1] 4 4
> AT %*% A
     [,1] [,2]
[1,]   68   -1
[2,]   -1   21
> dim(AT %*% A)
[1] 2 2
```

5. Calculate the inverse matrix of A*AT and AT*A. The product of A*AT doesn't have inverses. It's a singular matrix. The product of AT*A is invertible.

```
> solve(A %*% AT)
Error in solve.default(A %*% AT) :
  system is computationally singular: reciprocal condition number = 3.4479e-19
> solve(AT %*% A)
             [,1]         [,2]
[1,] 0.0147161878 0.0007007708
[2,] 0.0007007708 0.0476524177
```

6. Extend V with the ninth number of -2.

```
> V <- c(V, -2)
> V
[1]  7  2  1  0  3 -1 -3  4 -2
```

7. Transform the extended matrix into a 3x3 matrix B.

```
> B <- matrix(V, nrow = 3, ncol = 3, byrow = TRUE)
> B
     [,1] [,2] [,3]
[1,]    7    2    1
[2,]    0    3   -1
[3,]   -3    4   -2
```

8. Calculate the inverse matrix of B. Call is Binv. The product of B*Binv is equal to the product of Binv*B. The product is an identity matrix.

```
> Binv <- solve(B)
> Binv
     [,1] [,2] [,3]
[1,]   -2    8   -5
[2,]    3  -11    7
[3,]    9  -34   21
> B %*% Binv
             [,1]          [,2]         [,3]
[1,] 1.000000e+00 -7.105427e-15 3.552714e-15
[2,] 1.776357e-15  1.000000e+00 0.000000e+00
[3,] 3.552714e-15  0.000000e+00 1.000000e+00
> Binv %*% B
             [,1]          [,2] [,3]
[1,] 1.000000e+00 -3.552714e-15    0
[2,] 3.552714e-15  1.000000e+00    0
[3,] 0.000000e+00  1.421085e-14    1
> zapsmall(B %*% Binv) == zapsmall(Binv %*% B)
     [,1] [,2] [,3]
[1,] TRUE TRUE TRUE
[2,] TRUE TRUE TRUE
[3,] TRUE TRUE TRUE
```

9. Calculate the eigenvectors of matrix B. The eigen() will return the eigenvectors and eigenvalues at the same time.

```
> Beigen <- eigen(B)
> Beigen
$values
[1] 6.854102 1.000000 0.145898

$vectors
            [,1]       [,2]       [,3]
[1,]  0.95425723 -0.2857143 -0.2280090
[2,]  0.07508986  0.4285714  0.3219539
[3,] -0.28940397  0.8571429  0.9188893

> Beigenvectors = Beigen$vectors
> Beigenvectors
            [,1]       [,2]       [,3]
[1,]  0.95425723 -0.2857143 -0.2280090
[2,]  0.07508986  0.4285714  0.3219539
[3,] -0.28940397  0.8571429  0.9188893
> Beigenvalues = Beigen$values
```

10. The eigenvector of B is already made by using each eigenvector of matrix B as a column. So Beigenvectors is matrix C. The results below show that the product of the inverse matrix of C*B*C is a diagonal matrix. It has the eigenvalues on the diagonal.

```
> Beigenvectors %*% B
           [,1]      [,2]       [,3]
[1,]  7.3638277 0.1393355  1.6959896
[2,] -0.4402327 2.7237097 -0.9973894
[3,] -4.7824957 5.6681779 -2.9843254
```

```
> B %*% Beigenvectors
           [,1]       [,2]        [,3]
[1,]  6.5405763 -0.2857143 -0.03326607
[2,]  0.5146735  0.4285714  0.04697244
[3,] -1.9836043  0.8571429  0.13406414
```

```
> solve(Beigenvectors) %*% B %*% Beigenvectors
              [,1]         [,2]         [,3]
[1,]  6.854102e+00 2.775558e-16 1.110223e-16
[2,]  1.221245e-15 1.000000e+00 4.440892e-16
[3,] -3.608225e-15 2.664535e-15 1.458980e-01
```

11. Transform matrix B into a matrix with names columns and named rows.

```
> B
     [,1] [,2] [,3]
[1,]    7    2    1
[2,]    0    3   -1
[3,]   -3    4   -2
> dimnames(B) <- list(c("R1", "R2", "R3"), c("C1", "C2", "C3"))
> B
   C1 C2 C3
R1  7  2  1
R2  0  3 -1
R3 -3  4 -2
> class(B)
[1] "matrix"
```

12. Transformed that fully "named" matrix into a data.frame. Ask the object you just created what is its class().

```
> dataframe = data.frame(B)
> class(dataframe)
[1] "data.frame"
> is.data.frame(dataframe)
[1] TRUE
> labels(dataframe)
[[1]]
[1] "R1" "R2" "R3"

[[2]]
[1] "C1" "C2" "C3"

> names(dataframe)
[1] "C1" "C2" "C3"
> row.names(dataframe)
[1] "R1" "R2" "R3"
```

Problem 2

Consider file 2006Data.csv upload to the class site in Assignment 01 folder. File represents actual measurement of power consumption in a country somewhere in a California. Import data contained in that file into a data frame. You are expected to Google and find a function that will let you perform that import. Create a scatter plot of power consumption vs. temperature and power consumption vs. hour of the day.

Subsequently create a boxplot with power on the vertical axis and hour of the day on the horizontal axis. The objective is to present the distribution (variation) of power consumption for every hour of the day.

Solution:

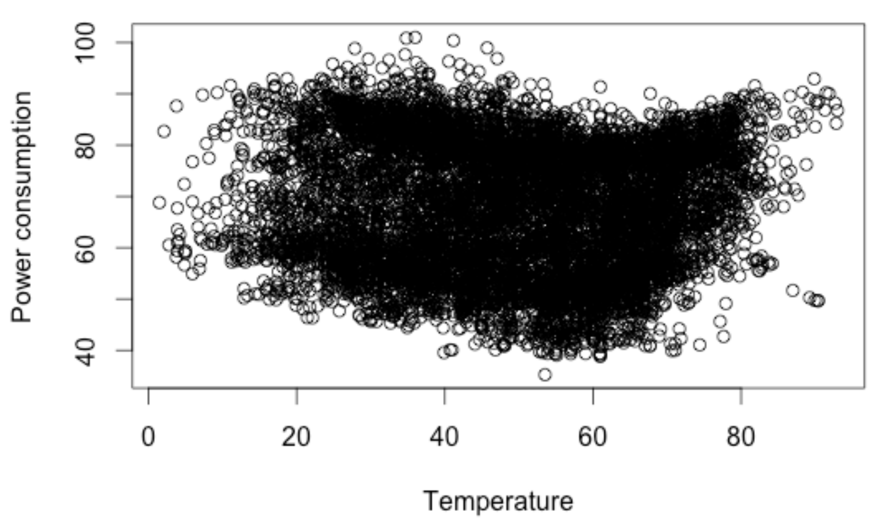
1. Import the data from the csv file into a data frame. Use colClasses here to remove the redundant columns.

```
> mydata <- read.csv("~/Downloads/2006Data.csv", header = TRUE, sep = ",", colClasses = c(NA, N
A, NA, NA, NA, NA, NA, "NULL"))
> summary(mydata)
     Month             Day             Hour         DayOfWeek        Holiday
 Min.   : 1.000   Min.   : 1.00   Min.   : 1.00   Min.   :1.000   Min.   :0.0000
 1st Qu.: 4.000   1st Qu.: 8.00   1st Qu.: 6.75   1st Qu.:2.000   1st Qu.:0.0000
 Median : 7.000   Median :16.00   Median :12.50   Median :4.000   Median :0.0000
 Mean   : 6.526   Mean   :15.72   Mean   :12.50   Mean   :4.008   Mean   :0.0274
 3rd Qu.:10.000   3rd Qu.:23.00   3rd Qu.:18.25   3rd Qu.:6.000   3rd Qu.:0.0000
 Max.   :12.000   Max.   :31.00   Max.   :24.00   Max.   :7.000   Max.   :1.0000
     Power          Temperature
 Min.   : 35.26   Min.   : 1.475
 1st Qu.: 57.19   1st Qu.:35.865
 Median : 67.85   Median :49.835
 Mean   : 67.62   Mean   :49.106
 3rd Qu.: 78.31   3rd Qu.:62.676
 Max.   :100.99   Max.   :93.000
> class(mydata)
[1] "data.frame"
```

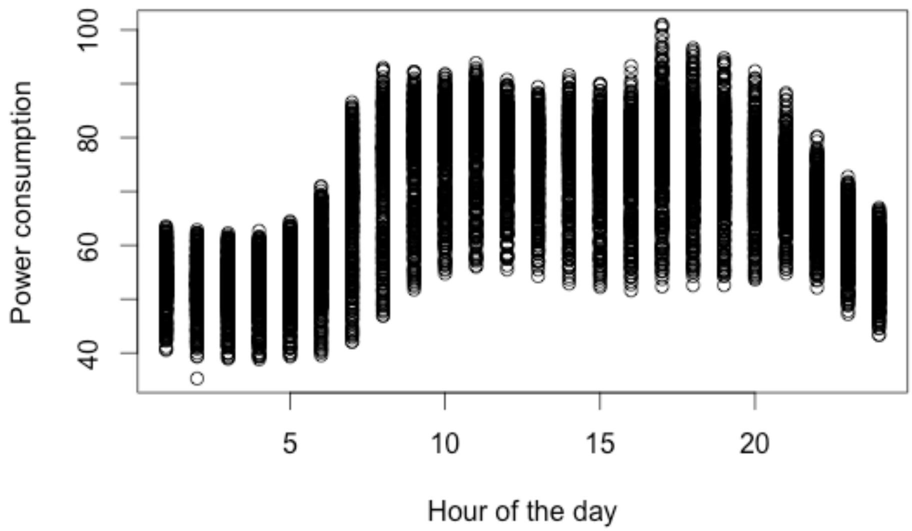
2. Create a scatter plot of power consumptions vs. temperature and power consumptions vs. hour of the day.

```
> power = mydata$Power
> temperature = mydata$Temperature
> hour = mydata$Hour
> plot(temperature, power, xlab = "Temperature", ylab = "Power consumption")
> plot(hour, power, xlab = "Hour of the day", ylab = "Power consumption")
```

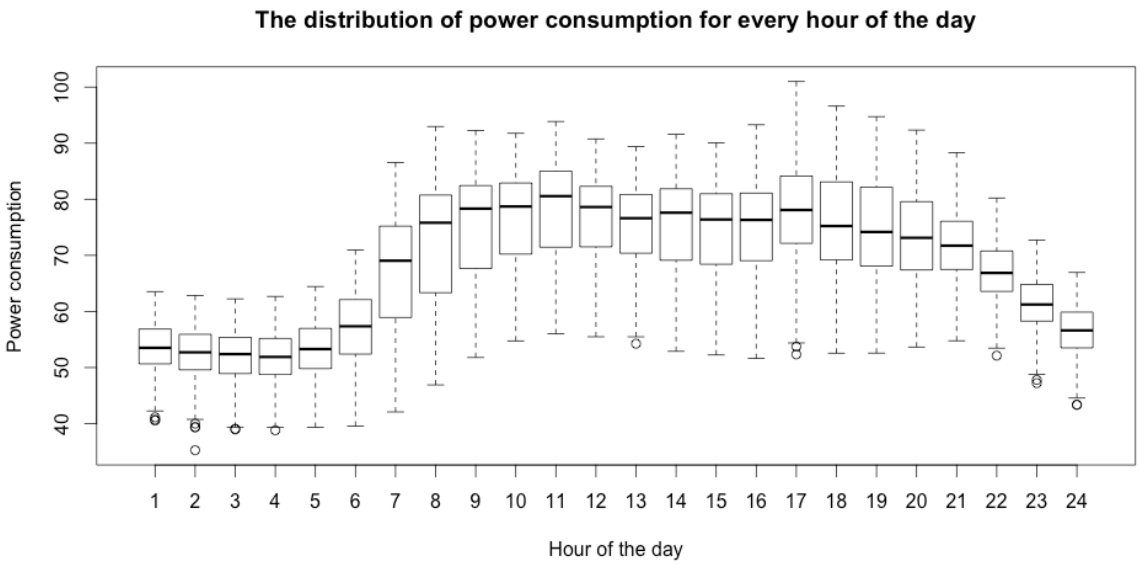
The scatter plot of power consumptions vs. temperature.



The scatter plot of power consumptions vs. hour of the day.



3. Create a boxplot of power consumptions vs. hour of the day to present the distribution of power for every hour of the day. The dots outside the min value are outliers, which are less than 3/2 times of the lower quartile.



Problem 3

Separate temperature scale in a reasonable number of intervals: 50 or 100. Calculate average power consumption, minimum power consumption and maximum power consumptions for every interval. Present those three sets of values on a single scatter graph (perhaps in different colours). Calculate three covariance matrixes between temperature and each of those power indicators (min, average, max).

Solution:

1. Separate the temperature scale in 50 intervals. Assign values from temperature to the intervals delimited by sequence breaks. “right = FALSE” means the intervals are close on the left and open on the right.

```
> breaks = seq(1.0, 94.0, by = 1.86);
> breaks
 [1]  1.00  2.86  4.72  6.58  8.44 10.30 12.16 14.02 15.88 17.74 19.60 21.46 23.32 25.18
[15] 27.04 28.90 30.76 32.62 34.48 36.34 38.20 40.06 41.92 43.78 45.64 47.50 49.36 51.22
[29] 53.08 54.94 56.80 58.66 60.52 62.38 64.24 66.10 67.96 69.82 71.68 73.54 75.40 77.26
[43] 79.12 80.98 82.84 84.70 86.56 88.42 90.28 92.14 94.00
> temperature.cut = cut(temperature, breaks, right=FALSE);
```

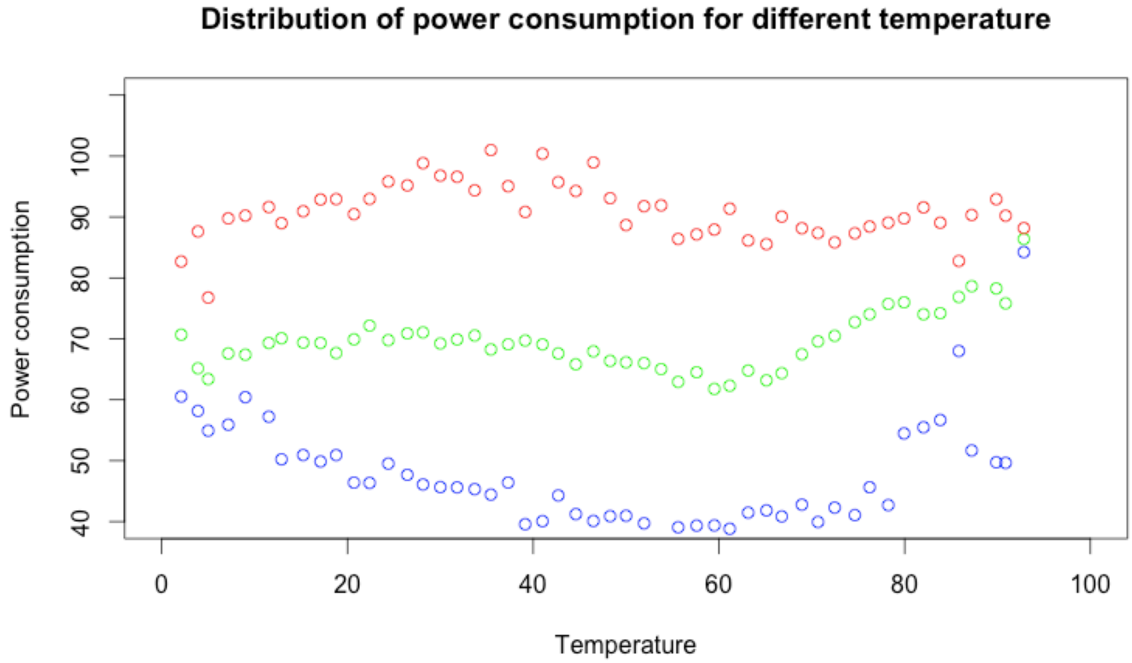
2. Calculate average, minimum and maximum power consumptions for every interval.

```
> maxpower = tapply(power, temperature.cut, max)
> minpower = tapply(power, temperature.cut, min)
> meanpower = tapply(power, temperature.cut, mean)
> maxpower
   [1,2.86) [2.86,4.72) [4.72,6.58) [6.58,8.44) [8.44,10.3) [10.3,12.2)   [12.2,14)
    82.6960     87.6270     76.7704     89.7771     90.2336     91.6283     88.9911
  [14,15.9) [15.9,17.7) [17.7,19.6) [19.6,21.5) [21.5,23.3) [23.3,25.2)   [25.2,27)
    90.9417     92.8717     92.9389     90.4748     92.9723     95.8191     95.1873
  [27,28.9) [28.9,30.8) [30.8,32.6) [32.6,34.5) [34.5,36.3) [36.3,38.2) [38.2,40.1)
    98.8383     96.7866     96.6031     94.3725    100.9896     95.0553     90.8336
[40.1,41.9) [41.9,43.8) [43.8,45.6) [45.6,47.5) [47.5,49.4) [49.4,51.2) [51.2,53.1)
   100.3930     95.7247     94.2719     98.9550     93.0825     88.6777     91.7737
[53.1,54.9) [54.9,56.8) [56.8,58.7) [58.7,60.5) [60.5,62.4) [62.4,64.2) [64.2,66.1)
    91.9065     86.3998     87.1483     87.9378     91.3635     86.1453     85.5433
  [66.1,68)   [68,69.8) [69.8,71.7) [71.7,73.5) [73.5,75.4) [75.4,77.3) [77.3,79.1)
    90.0549     88.1541     87.3791     85.8301     87.3303     88.4328     89.0667
  [79.1,81)   [81,82.8) [82.8,84.7) [84.7,86.6) [86.6,88.4) [88.4,90.3) [90.3,92.1)
    89.7736     91.5709     89.0597     82.7765     90.3311     92.9252     90.2133
  [92.1,94)
    88.1772
> minpower
   [1,2.86) [2.86,4.72) [4.72,6.58) [6.58,8.44) [8.44,10.3) [10.3,12.2)   [12.2,14)
    60.5326     58.1601     54.9205     55.9044     60.4221     57.2195     50.2237
  [14,15.9) [15.9,17.7) [17.7,19.6) [19.6,21.5) [21.5,23.3) [23.3,25.2)   [25.2,27)
    50.9279     49.8641     50.9160     46.3978     46.3474     49.5164     47.6825
  [27,28.9) [28.9,30.8) [30.8,32.6) [32.6,34.5) [34.5,36.3) [36.3,38.2) [38.2,40.1)
    46.0882     45.6591     45.6151     45.3238     44.4238     46.4085     39.5643
[40.1,41.9) [41.9,43.8) [43.8,45.6) [45.6,47.5) [47.5,49.4) [49.4,51.2) [51.2,53.1)
    40.0671     44.2881     41.2311     40.1007     40.8894     40.9541     39.7150
[53.1,54.9) [54.9,56.8) [56.8,58.7) [58.7,60.5) [60.5,62.4) [62.4,64.2) [64.2,66.1)
    35.2605     39.0662     39.3408     39.3739     38.8142     41.4718     41.8363
  [66.1,68)   [68,69.8) [69.8,71.7) [71.7,73.5) [73.5,75.4) [75.4,77.3) [77.3,79.1)
    40.8431     42.7762     39.9362     42.2921     41.0651     45.6383     42.7008
  [79.1,81)   [81,82.8) [82.8,84.7) [84.7,86.6) [86.6,88.4) [88.4,90.3) [90.3,92.1)
    54.4724     55.4768     56.6712     68.0002     51.6770     49.7290     49.6333
  [92.1,94)
    84.2437
```

```
> meanpower
   [1,2.86) [2.86,4.72) [4.72,6.58) [6.58,8.44) [8.44,10.3) [10.3,12.2)   [12.2,14)
   70.67890    65.16306    63.39036    67.59992    67.39706    69.32481    70.09464
  [14,15.9) [15.9,17.7) [17.7,19.6) [19.6,21.5) [21.5,23.3) [23.3,25.2)   [25.2,27)
   69.38817    69.32528    67.67428    69.90888    72.16975    69.78337    70.89329
  [27,28.9) [28.9,30.8) [30.8,32.6) [32.6,34.5) [34.5,36.3) [36.3,38.2) [38.2,40.1)
   71.06004    69.23769    69.89605    70.54779    68.25706    69.10285    69.71936
[40.1,41.9) [41.9,43.8) [43.8,45.6) [45.6,47.5) [47.5,49.4) [49.4,51.2) [51.2,53.1)
   69.08667    67.59824    65.81948    67.94535    66.38124    66.14448    66.01584
[53.1,54.9) [54.9,56.8) [56.8,58.7) [58.7,60.5) [60.5,62.4) [62.4,64.2) [64.2,66.1)
   65.03037    62.93600    64.51590    61.77305    62.30138    64.79231    63.19053
  [66.1,68)   [68,69.8) [69.8,71.7) [71.7,73.5) [73.5,75.4) [75.4,77.3) [77.3,79.1)
   64.35930    67.46678    69.57974    70.49110    72.74420    74.04072    75.73548
  [79.1,81)   [81,82.8) [82.8,84.7) [84.7,86.6) [86.6,88.4) [88.4,90.3) [90.3,92.1)
   75.99739    74.02740    74.20681    76.86791    78.63333    78.26831    75.82372
  [92.1,94)
   86.39373
```

3. Present the above three sets of values on a single scatter graph. For the temperature, take the midpoints of every interval. Use points() to plot the values on the same graph.

```
> midtemperature = tapply(temperature, temperature.cut, median)
> midtemperature
   [1,2.86) [2.86,4.72) [4.72,6.58) [6.58,8.44) [8.44,10.3) [10.3,12.2)   [12.2,14)
    2.09500     3.90000     5.00000     7.16500     9.00000    11.53000    12.90000
  [14,15.9) [15.9,17.7) [17.7,19.6) [19.6,21.5) [21.5,23.3) [23.3,25.2)   [25.2,27)
   15.23500    17.10000    18.78250    20.70000    22.38375    24.40000    26.44620
  [27,28.9) [28.9,30.8) [30.8,32.6) [32.6,34.5) [34.5,36.3) [36.3,38.2) [38.2,40.1)
   28.13500    30.00000    31.80450    33.70000    35.44500    37.30000    39.13500
[40.1,41.9) [41.9,43.8) [43.8,45.6) [45.6,47.5) [47.5,49.4) [49.4,51.2) [51.2,53.1)
   41.00000    42.70000    44.59055    46.48000    48.28500    50.00000    51.93790
[53.1,54.9) [54.9,56.8) [56.8,58.7) [58.7,60.5) [60.5,62.4) [62.4,64.2) [64.2,66.1)
   53.77430    55.61500    57.60000    59.50500    61.16500    63.15000    65.11905
  [66.1,68)   [68,69.8) [69.8,71.7) [71.7,73.5) [73.5,75.4) [75.4,77.3) [77.3,79.1)
   66.76500    68.93500    70.66560    72.46500    74.63500    76.23000    78.23500
  [79.1,81)   [81,82.8) [82.8,84.7) [84.7,86.6) [86.6,88.4) [88.4,90.3) [90.3,92.1)
   79.93500    82.03000    83.83500    85.83500    87.22000    89.86500    90.87500
  [92.1,94)
   92.83500
> plot(midtemperature, meanpower, main = "Distribution of power consumption for different tempe
rature", xlab = "Temperature", ylab = "Power consumption", xlim = c(0, 100), ylim = c(40, 110),
col = "green")
> points(midtemperature, maxpower, col = "red")
> points(midtemperature, minpower, col = "blue")
```



4. Calculate three covariance matrix between temperature and each of those power indicators (min, average, max).

```
> cov(midtemperature, maxpower)
[1] -21.50168
> cov(midtemperature, minpower)
[1] -11.55951
> cov(midtemperature, meanpower)
[1] 57.88333
> M <- cbind(midtemperature, maxpower, minpower, meanpower)
> cov(M)
               midtemperature    maxpower  minpower  meanpower
midtemperature      733.81772 -21.5016801 -11.55951 57.8833333
maxpower            -21.50168  21.3789126 -10.87648  0.8847414
minpower            -11.55951 -10.8764830  77.90368 25.8822440
meanpower            57.88333   0.8847414  25.88224 22.9500587
```

The covariance matrix is a matrix that only concerns the relationship between variables. We can create a matrix from the vectors (temperature, min, average, max). Each value in the covariance matrix represents the covariance (or variance) between two of the vectors. The first column shows the covariance between temperatures and power indicators.