Ryan Ballenger
CSCIE63 Big Data Analytics
Assignment 1

Assignment 1 Solutions

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).

Solution:

```
# First, I use the c command to create a vector from the given values.
> v <- c(7,2,1,0,3,-1,-3,4)
> v
 [1]  7  2  1  0  3 -1 -3  4

# matrix() is used create a matrix from the values in the vector v that is 4x2
> A <- matrix(v, nrow=4, ncol=2)
> A
     [,1] [,2]
[1,]    7    3
[2,]    2   -1
[3,]    1   -3
[4,]    0    4
```

- Create a matrix transpose to the above matrix A. Call that matrix AT.

Solution:

```
# The transpose function is used on the matrix A.
> AT <- t(A)
> AT
     [,1] [,2] [,3] [,4]
[1,]    7    2    1    0
[2,]    3   -1   -3    4
```

- Calculate matrix products: A*AT and AT*A. Present the results. What are the dimensions of those two product matrices.

Solution:

```
# Multiplication is done using R's matrix multiplication command %*%
> product <- A%*% AT
> product
     [,1] [,2] [,3] [,4]
[1,]   58   11   -2   12
[2,]   11    5    5   -4
[3,]   -2    5   10  -12
[4,]   12   -4  -12   16
```

The dimensions are 4x4 for A * AT.

```
> product <- AT %*% A
> product
     [,1] [,2]
[1,]   54   16
[2,]   16   35
```
The dimensions are 2x2 for AT * A

- 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.

The solve function is used to find the inverse of the matrices.
```
> solve(A %*% AT)
```
A*AT is a singular matrix, meaning its determinant is zero and it cannot be inverted.
```
> solve(AT %*% A)
             [,1]         [,2]
[1,]  0.021419829 -0.009791922
[2,] -0.009791922  0.033047736
```

- Extend the above vector V with the ninth number of value -2. Do it elegantly by concatenating two vectors ().

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

- Transform that extended vector into a 3X3 matrix B.

Solution:
```
> B <- matrix(v, nrow=3, ncol=3)
> B
     [,1] [,2] [,3]
[1,]    7    0   -3
[2,]    2    3    4
[3,]    1   -1   -2
```

- 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?

Solution:
The inverse is calculated using the solve function on the matrix B.
```
> Binv = solve(B)
> Binv
     [,1] [,2] [,3]
[1,]   -2    3    9
[2,]    8  -11  -34
```

```
[3,]   -5    7   21

> B %*% Binv
     [,1]         [,2]          [,3]
[1,]    1 0.000000e+00  0.000000e+00
[2,]    0 1.000000e+00 -1.421085e-14
[3,]    0 1.776357e-15  1.000000e+00
> Binv %*% B
              [,1] [,2]          [,3]
[1,]  1.000000e+00    0  0.000000e+00
[2,] -7.105427e-15    1 -1.421085e-14
[3,] -3.552714e-15    0  1.000000e+00

# By rounding the matrices above to the 6th decimal place, both result in the matrix below
# which is the identity matrix.
      [,1]    [,2]   [,3]
[1,]  1.0     0.0     0.0
[2,]  0.0     1.0     0.0
[3,]  0.0     0.0     1.0
```

- Determine the eigenvectors of matrices B.

```
# The eigenvectors are found by using the eigen function and then by extracting the vectors.
# The eigenvectors are the columns 1, 2, and 3 in the matrix C below.
> C <-eigen(B)$vectors
> C
           [,1]       [,2]       [,3]
[1,] 0.86822600  0.1825742  0.2159107
[2,] 0.49436902 -0.9128709 -0.8426423
[3,] 0.04222416  0.3651484  0.4932914
```

- 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.

```
# C is found in the previous step and is the $vectors portion of the result of eigen(B)
> B %*% C
          [,1]       [,2]        [,3]
[1,] 5.9509095  0.1825742  0.03150095
[2,] 3.3884557 -0.9128709 -0.12293986
[3,] 0.2894087  0.3651484  0.07197025

> C %*% B
```

```
          [,1]       [,2]       [,3]
[1,] 6.658641  0.3318118 -2.3062028
[2,] 0.792199 -1.8959704 -3.4493061
[3,] 1.519157  0.6021537  0.3473382
```

The two matrices above don't have significant values. The matrix below of C-inverse * B * C
produces a diagonal matrix with the eigenvalues of B in the diagonal.

```
> solve(C) %*% B %*% C
              [,1]          [,2]          [,3]
[1,]  6.854102e+00 -1.665335e-15 -1.776357e-15
[2,]  3.108624e-15  1.000000e+00 -1.776357e-15
[3,] -1.762479e-15 -1.221245e-15  1.458980e-01
```

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

```
# the rows and columns are named by assigning the dimnames of B
> dimnames(B) <- list(c("row1", "row2", "row3"), c("col1", "col2", "col3"))
> B
     col1 col2 col3
row1    7    0   -3
row2    2    3    4
row3    1   -1   -2
```

- Transformed that fully "named" matrix into a data.frame.

```
# the transformation is done with the data.frame() function on the matrix B
> dFrame = data.frame(B)
> dFrame
     col1 col2 col3
row1    7    0   -3
row2    2    3    4
row3    1   -1   -2
```

- Ask the object you just created what is its class().

```
> class(dFrame)
[1] "data.frame"
```

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.

Solution:
The read.csv() function is used to import the data

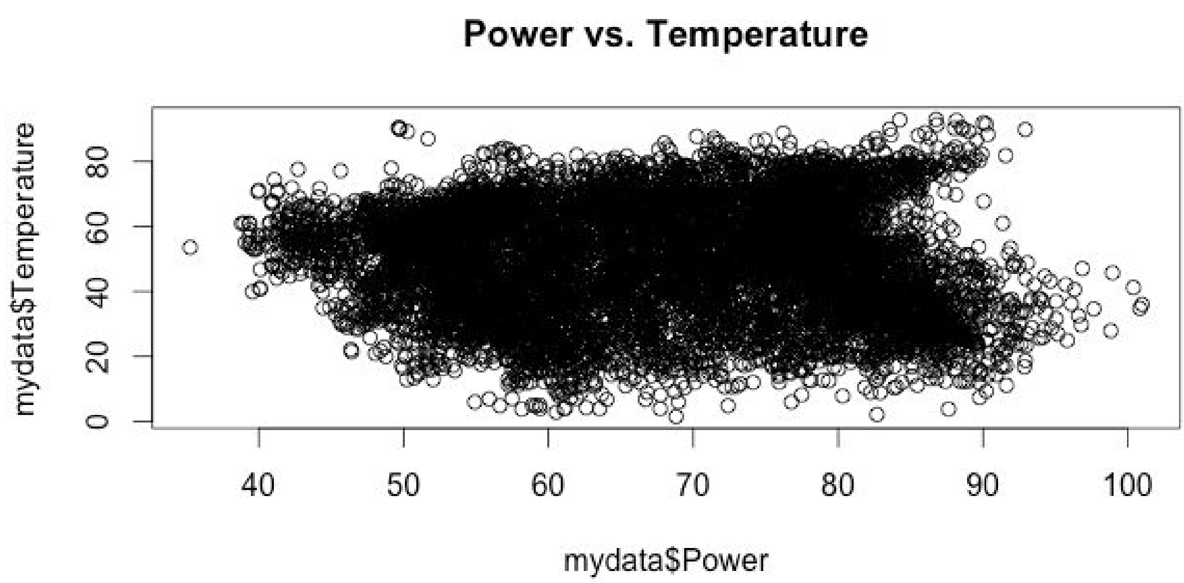
```
> mydata = read.csv("2006Data.csv")
> head(mydata)
```

	Month	Day	Hour	DayOfWeek	Holiday	Power	Temperature	X
1	1	1	1	7	0	54.5448	19.0000	NA
2	1	1	2	7	0	52.3898	18.8500	NA
3	1	1	3	7	0	51.6344	17.8650	NA
4	1	1	4	7	0	51.5597	17.2800	NA
5	1	1	5	7	0	51.7148	15.9182	NA
6	1	1	6	7	0	52.6898	16.2400	NA

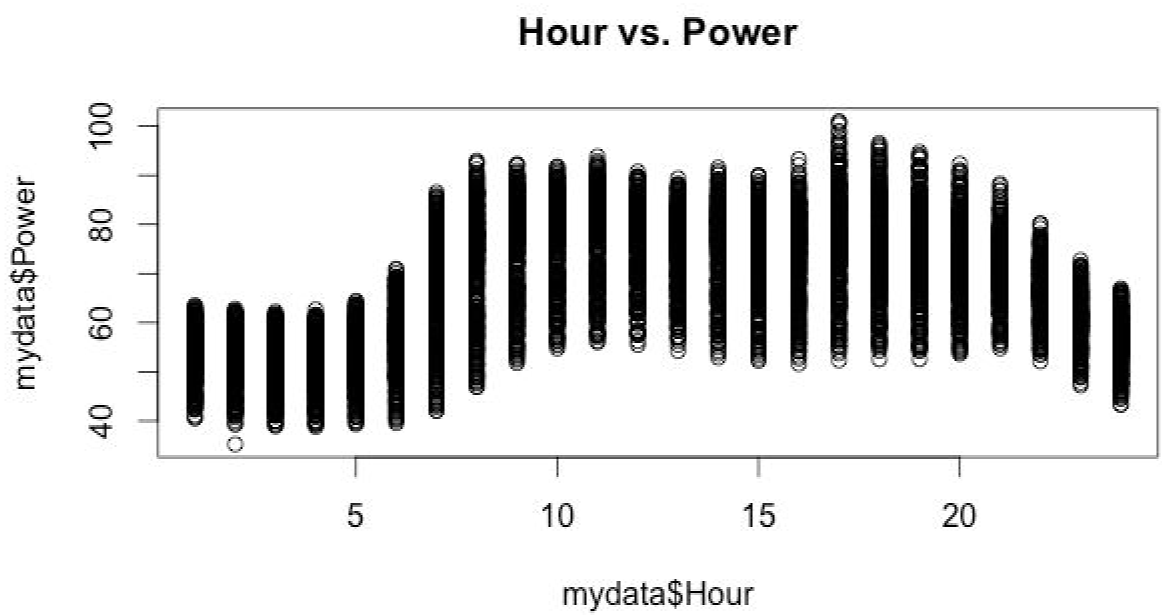
Create a scatter plot of power consumption vs. temperature and power consumption vs. hour of the day.

Solution:
I used the plot function and selected the appropriate data from the whole data frame mydata.

```
> plot(mydata$Power, mydata$Temperature, main="Power vs. Temperature")
```




```
> plot(mydata$Hour, mydata$Power, main="Hour vs. Power")
```

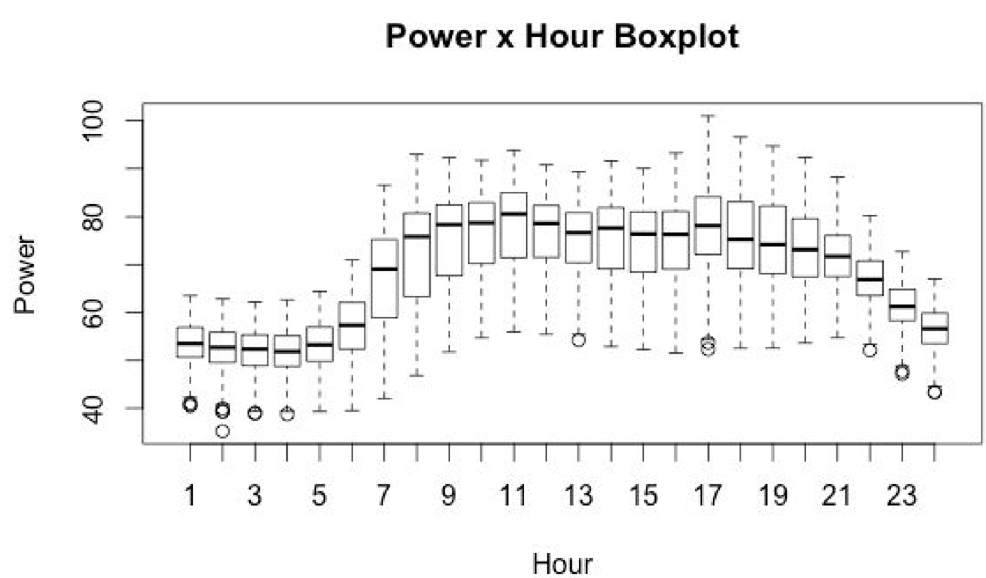


Subsequently create a boxplot with power on the vertical axis and hour of the day on the horizontal axis. The objective is to present the distribution (variation) of power consumption for every hour of the day.

Solution:

I used the boxplot method and selected the appropriate data from the data frame mydata

> boxplot(mydata$Power ~ mydata$Hour, main="Power x Hour Boxplot", xlab="Hour", ylab="Power")



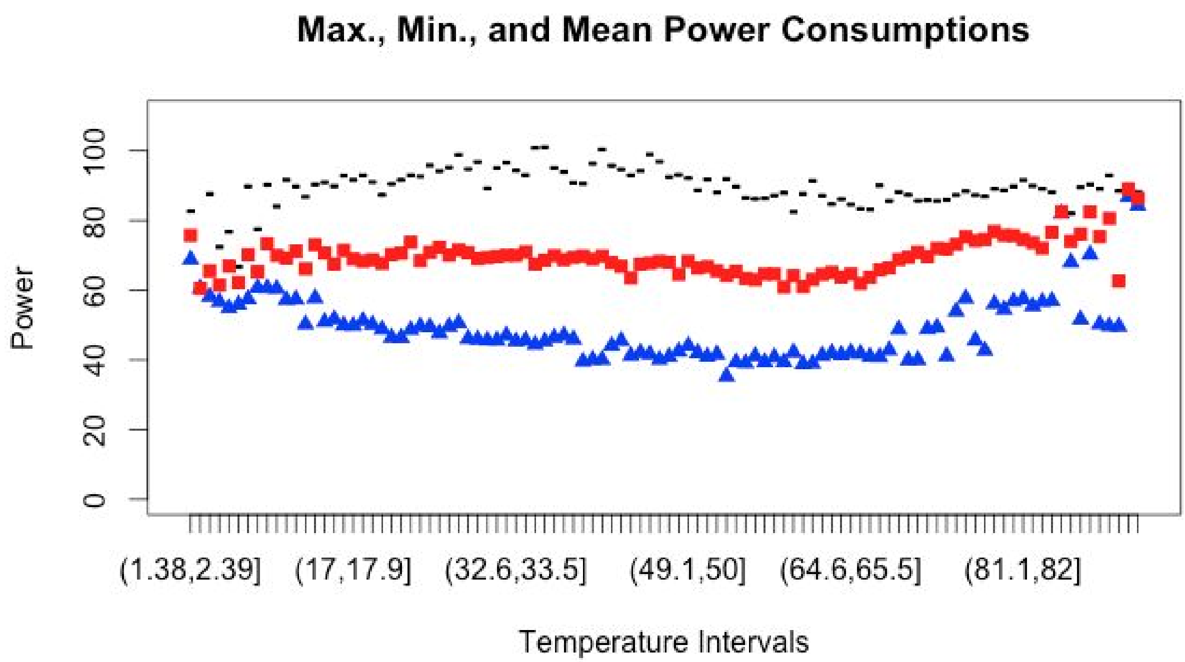
Problem 3. Separate temperature scale in a reasonable number of intervals: 50 or 100. Calculate average power consumption, minimum power consumption and maximum power consumptions for every interval.

Solution:

```
# 100 temperature intervals are made with the cut function. Then the power data is aggregated
# by which temperature interval it is in. The mean, min., and max. power values for each
# interval are found by setting the FUN parameter in aggregate() to the respective function.
> intervals= cut(mydata$Temperature, 100)
> storeMean = aggregate(mydata$Power, by=list(intervals), FUN=mean)

# First, the mean power value for each temperature interval is found.
> head(storeMean)
     Group.1        x
1 (1.38,2.39] 75.75205
2 (2.39,3.31] 60.53260
3 (3.31,4.22] 65.52837
4 (4.22,5.14] 61.51257
5 (5.14,6.05] 66.88447
6 (6.05,6.97] 62.21377

# Next the max. power value for each temperature interval is found.
> storeMax = aggregate(mydata$Power, by=list(intervals), FUN=max)
> head(storeMax)
     Group.1       x
1 (1.38,2.39] 82.6960
2 (2.39,3.31] 60.5326
3 (3.31,4.22] 87.6270
4 (4.22,5.14] 72.3954
5 (5.14,6.05] 76.7704
6 (6.05,6.97] 66.7038

# Lastly, the minimum power value for each temperature interval is found.
> storeMin = aggregate(mydata$Power, by=list(intervals), FUN=min)
> head(storeMin)
     Group.1       x
1 (1.38,2.39] 68.8081
2 (2.39,3.31] 60.5326
3 (3.31,4.22] 58.1601
4 (4.22,5.14] 56.6477
5 (5.14,6.05] 54.9205
6 (6.05,6.97] 55.9044
```

Note: Complete listings of mean, max., and min. powers by interval are listed at the end.

Present those three sets of values on a single scatter graph (perhaps in different colours).
Solution:
The plot function is used and additional data sets for the min. and mean power are added with
the points function. Also each data set is colored differently with the par function.
> plot(storeMax, ylim=c(0, 110), xlab="Temperature Intervals", ylab="Power", main="Max., Min., and Mean Power Consumptions")
> par(pch=17, col="blue")
> points(storeMin)
> par(pch=15, col="red")
> points(storeMean)



Calculate three covariance matrixes between temperature and each of those power indicators (min, average, max).
Solution:
The intervals are extracted and saved as mp. The intervals are converted to a string and split
until the beginning and ending value of each interval is reached. Those values are converted
to a numeric and lastly, the average of the start and end of each interval is calculated.
> mp = storeMean$Group.1
> midpoints = (aggregate(as.data.frame(mp), by=list(mp), FUN=function(x)
(as.numeric(substr(unlist(strsplit(as.character(x), ","))[1], 2,
10))+as.numeric(unlist(strsplit(unlist(strsplit(as.character(x), ","))[2], "]")[1])[1]))/2))$mp
> head(midpoints)

```
[1] 1.885 2.850 3.765 4.680 5.595 6.510
```

Temperature and minimum power covariance

```
# the interval midpoints and the min. power values are put into a matrix.
# The resulting covariance of this 100x2 matrix is a 4x4 matrix.
# 705.190794 is the covariance of the midpoints with themselves.
# 7.906748 is the covariance of the midpoints with the min. power values, meaning there is a
# positive correlation between these two variables.
# 92.406717 is the covariance of min. power values with themselves.
> cov(matrix(c(midpoints, storeMin$x), nrow=100, ncol=2))
           [,1]      [,2]
[1,] 705.190794  7.906748
[2,]   7.906748 92.406717
```

Temperature and mean power covariance

```
# The midpoints and the mean power are put into a matrix for the covariance calculation.
# 705.19079 again represents the same covariance of the midpoints with themselves.
# 49.85253 is the covariance between the mean power values and the midpoints.
# This suggest a strong positive correlation between these two variables that is stronger than
# that between the min. power points and the midpoints.
# 27.35601 is the covariance of the mean power values with themselves.
> cov(matrix(c(midpoints, storeMean$x), nrow=100, ncol=2))
          [,1]     [,2]
[1,] 705.19079 49.85253
[2,]  49.85253 27.35601
```

Temperature and maximum power covariance

```
# The midpoints and the max. power are put into a matrix for the covariance calculation.
# 705.19079 again represents the same cov. of the midpoints with themselves.
#  3.382451 is the covariance between the max. power values and the midpoints.
# This suggest a weak positive correlation between these two variables that is weaker than
# that between the min. power or mean power and the midpoints.
# 37.496919 is the covariance of the max. power values with themselves.
> cov(matrix(c(midpoints, storeMax$x), nrow=100, ncol=2))
           [,1]      [,2]
[1,] 705.190794  3.382451
[2,]   3.382451 37.496919
```

Appendix:

```
> storeMean
      Group.1        x
1   (1.38,2.39] 75.75205
2   (2.39,3.31] 60.53260
3   (3.31,4.22] 65.52837
4   (4.22,5.14] 61.51257
5   (5.14,6.05] 66.88447
6   (6.05,6.97] 62.21377
7   (6.97,7.88] 70.20718
8    (7.88,8.8] 65.26590
9    (8.8,9.71] 73.39637
10  (9.71,10.6] 69.99766
11  (10.6,11.5] 69.16680
12  (11.5,12.5] 71.18797
13  (12.5,13.4] 66.18224
14  (13.4,14.3] 73.04802
15  (14.3,15.2] 70.67881
16  (15.2,16.1] 67.54408
17    (16.1,17] 71.44375
18    (17,17.9] 68.94623
19  (17.9,18.9] 68.31896
20  (18.9,19.8] 68.71435
21  (19.8,20.7] 67.73369
22  (20.7,21.6] 70.25384
23  (21.6,22.5] 70.61595
24  (22.5,23.4] 73.80868
25  (23.4,24.4] 68.44159
26  (24.4,25.3] 70.88065
27  (25.3,26.2] 72.30138
28  (26.2,27.1] 70.10909
29    (27.1,28] 71.55183
30    (28,28.9] 70.75086
31  (28.9,29.8] 69.03743
32  (29.8,30.8] 69.37282
33  (30.8,31.7] 69.71369
34  (31.7,32.6] 70.01482
35  (32.6,33.5] 70.00666
36  (33.5,34.4] 70.95129
37  (34.4,35.3] 67.51793
38  (35.3,36.3] 68.55177
39  (36.3,37.2] 69.91988
```

40	(37.2,38.1]	68.72625
41	(38.1,39]	69.43315
42	(39,39.9]	69.81562
43	(39.9,40.8]	68.87853
44	(40.8,41.7]	69.81403
45	(41.7,42.7]	67.93662
46	(42.7,43.6]	66.83172
47	(43.6,44.5]	63.54442
48	(44.5,45.4]	67.49919
49	(45.4,46.3]	67.80343
50	(46.3,47.2]	68.30028
51	(47.2,48.2]	68.03177
52	(48.2,49.1]	64.55237
53	(49.1,50]	68.31634
54	(50,50.9]	66.38534
55	(50.9,51.8]	66.74457
56	(51.8,52.7]	65.55811
57	(52.7,53.6]	64.22405
58	(53.6,54.6]	65.38439
59	(54.6,55.5]	63.35608
60	(55.5,56.4]	62.99458
61	(56.4,57.3]	64.62322
62	(57.3,58.2]	64.63714
63	(58.2,59.1]	60.87621
64	(59.1,60.1]	64.16369
65	(60.1,61]	60.99559
66	(61,61.9]	63.04476
67	(61.9,62.8]	64.55285
68	(62.8,63.7]	65.11167
69	(63.7,64.6]	63.79745
70	(64.6,65.5]	64.67096
71	(65.5,66.5]	61.96465
72	(66.5,67.4]	63.73172
73	(67.4,68.3]	65.90883
74	(68.3,69.2]	66.40768
75	(69.2,70.1]	68.76676
76	(70.1,71]	69.48880
77	(71,71.9]	70.85030
78	(71.9,72.9]	69.67759
79	(72.9,73.8]	72.02108
80	(73.8,74.7]	71.83831
81	(74.7,75.6]	73.19687
82	(75.6,76.5]	75.34647

```
83  (76.5,77.4] 74.23531
84  (77.4,78.4] 74.56746
85  (78.4,79.3] 76.81973
86  (79.3,80.2] 75.74894
87  (80.2,81.1] 75.70026
88    (81.1,82] 74.57160
89    (82,82.9] 73.64694
90  (82.9,83.8] 72.06534
91  (83.8,84.8] 76.58159
92  (84.8,85.7] 82.48933
93  (85.7,86.6] 74.05720
94  (86.6,87.5] 76.09365
95  (87.5,88.4] 82.44285
96  (88.4,89.3] 75.38355
97  (89.3,90.3] 80.57612
98  (90.3,91.2] 62.62437
99  (91.2,92.1] 89.02307
100 (92.1,93.1] 86.39373
```

```
> storeMax
       Group.1        x
1   (1.38,2.39]  82.6960
2   (2.39,3.31]  60.5326
3   (3.31,4.22]  87.6270
4   (4.22,5.14]  72.3954
5   (5.14,6.05]  76.7704
6   (6.05,6.97]  66.7038
7   (6.97,7.88]  89.7771
8    (7.88,8.8]  77.4909
9    (8.8,9.71]  90.2336
10  (9.71,10.6]  84.0427
11  (10.6,11.5]  91.6283
12  (11.5,12.5]  89.7539
13  (12.5,13.4]  86.8044
14  (13.4,14.3]  90.3355
15  (14.3,15.2]  90.9417
16  (15.2,16.1]  89.8396
17    (16.1,17]  92.8717
18    (17,17.9]  91.6216
19  (17.9,18.9]  92.9389
20  (18.9,19.8]  91.0105
```

21	(19.8,20.7]	87.3819
22	(20.7,21.6]	90.4748
23	(21.6,22.5]	91.6367
24	(22.5,23.4]	92.9723
25	(23.4,24.4]	92.6620
26	(24.4,25.3]	95.8191
27	(25.3,26.2]	94.1587
28	(26.2,27.1]	95.1873
29	(27.1,28]	98.8383
30	(28,28.9]	94.7358
31	(28.9,29.8]	96.7866
32	(29.8,30.8]	89.1891
33	(30.8,31.7]	95.0473
34	(31.7,32.6]	96.6031
35	(32.6,33.5]	94.3725
36	(33.5,34.4]	93.0051
37	(34.4,35.3]	100.8437
38	(35.3,36.3]	100.9896
39	(36.3,37.2]	95.0553
40	(37.2,38.1]	93.9437
41	(38.1,39]	90.8336
42	(39,39.9]	90.5679
43	(39.9,40.8]	96.3287
44	(40.8,41.7]	100.3930
45	(41.7,42.7]	95.7247
46	(42.7,43.6]	94.6591
47	(43.6,44.5]	92.9579
48	(44.5,45.4]	94.2719
49	(45.4,46.3]	98.9550
50	(46.3,47.2]	96.8664
51	(47.2,48.2]	92.4966
52	(48.2,49.1]	93.0825
53	(49.1,50]	92.1957
54	(50,50.9]	88.6777
55	(50.9,51.8]	91.7737
56	(51.8,52.7]	88.0476
57	(52.7,53.6]	91.9065
58	(53.6,54.6]	89.7125
59	(54.6,55.5]	86.5059
60	(55.5,56.4]	86.2429
61	(56.4,57.3]	86.3998
62	(57.3,58.2]	87.1483
63	(58.2,59.1]	87.9378

```
64  (59.1,60.1]  82.5046
65    (60.1,61]  87.6648
66    (61,61.9]  91.3635
67  (61.9,62.8]  87.1151
68  (62.8,63.7]  84.7115
69  (63.7,64.6]  86.1453
70  (64.6,65.5]  84.5971
71  (65.5,66.5]  83.3536
72  (66.5,67.4]  83.1391
73  (67.4,68.3]  90.0549
74  (68.3,69.2]  85.5745
75  (69.2,70.1]  88.1541
76    (70.1,71]  87.3791
77    (71,71.9]  85.6309
78  (71.9,72.9]  85.8301
79  (72.9,73.8]  85.6128
80  (73.8,74.7]  85.9253
81  (74.7,75.6]  87.3303
82  (75.6,76.5]  88.4328
83  (76.5,77.4]  87.2949
84  (77.4,78.4]  86.9132
85  (78.4,79.3]  89.0667
86  (79.3,80.2]  88.6840
87  (80.2,81.1]  89.7736
88    (81.1,82]  91.5709
89    (82,82.9]  89.9356
90  (82.9,83.8]  89.0597
91  (83.8,84.8]  88.1065
92  (84.8,85.7]  82.7765
93  (85.7,86.6]  82.0439
94  (86.6,87.5]  89.5421
95  (87.5,88.4]  90.3311
96  (88.4,89.3]  89.0548
97  (89.3,90.3]  92.9252
98  (90.3,91.2]  88.5200
99  (91.2,92.1]  90.2133
100 (92.1,93.1]  88.1772


> storeMin
       Group.1       x
1   (1.38,2.39] 68.8081
2   (2.39,3.31] 60.5326
```

3	(3.31,4.22]	58.1601
4	(4.22,5.14]	56.6477
5	(5.14,6.05]	54.9205
6	(6.05,6.97]	55.9044
7	(6.97,7.88]	57.4954
8	(7.88,8.8]	60.7024
9	(8.8,9.71]	60.6043
10	(9.71,10.6]	60.4221
11	(10.6,11.5]	57.2195
12	(11.5,12.5]	57.4083
13	(12.5,13.4]	50.2237
14	(13.4,14.3]	57.7248
15	(14.3,15.2]	50.9279
16	(15.2,16.1]	51.7148
17	(16.1,17]	49.9430
18	(17,17.9]	49.8641
19	(17.9,18.9]	51.2449
20	(18.9,19.8]	50.1434
21	(19.8,20.7]	48.7796
22	(20.7,21.6]	46.3978
23	(21.6,22.5]	46.3474
24	(22.5,23.4]	48.6415
25	(23.4,24.4]	49.6580
26	(24.4,25.3]	49.5164
27	(25.3,26.2]	47.6825
28	(26.2,27.1]	49.6092
29	(27.1,28]	50.6035
30	(28,28.9]	46.0882
31	(28.9,29.8]	45.9109
32	(29.8,30.8]	45.6591
33	(30.8,31.7]	45.6151
34	(31.7,32.6]	47.0494
35	(32.6,33.5]	45.3238
36	(33.5,34.4]	45.6437
37	(34.4,35.3]	44.4238
38	(35.3,36.3]	45.3269
39	(36.3,37.2]	46.4085
40	(37.2,38.1]	47.0528
41	(38.1,39]	45.9313
42	(39,39.9]	39.5643
43	(39.9,40.8]	40.0857
44	(40.8,41.7]	40.0671
45	(41.7,42.7]	44.1056

46	(42.7,43.6]	45.4897
47	(43.6,44.5]	41.2311
48	(44.5,45.4]	42.1209
49	(45.4,46.3]	41.6288
50	(46.3,47.2]	40.1007
51	(47.2,48.2]	40.8894
52	(48.2,49.1]	42.4239
53	(49.1,50]	44.1051
54	(50,50.9]	41.9975
55	(50.9,51.8]	40.9541
56	(51.8,52.7]	41.6193
57	(52.7,53.6]	35.2605
58	(53.6,54.6]	39.3752
59	(54.6,55.5]	39.0662
60	(55.5,56.4]	41.0658
61	(56.4,57.3]	39.3408
62	(57.3,58.2]	40.7643
63	(58.2,59.1]	39.3739
64	(59.1,60.1]	42.1571
65	(60.1,61]	38.8142
66	(61,61.9]	38.9765
67	(61.9,62.8]	41.2297
68	(62.8,63.7]	41.9802
69	(63.7,64.6]	41.4718
70	(64.6,65.5]	42.1162
71	(65.5,66.5]	41.8363
72	(66.5,67.4]	40.9116
73	(67.4,68.3]	40.8431
74	(68.3,69.2]	42.7762
75	(69.2,70.1]	48.7543
76	(70.1,71]	39.9362
77	(71,71.9]	39.9917
78	(71.9,72.9]	48.8566
79	(72.9,73.8]	49.3744
80	(73.8,74.7]	41.0651
81	(74.7,75.6]	53.9464
82	(75.6,76.5]	57.6834
83	(76.5,77.4]	45.6383
84	(77.4,78.4]	42.7008
85	(78.4,79.3]	56.0804
86	(79.3,80.2]	54.4724
87	(80.2,81.1]	56.8079
88	(81.1,82]	57.5287

89	(82,82.9]	55.4768
90	(82.9,83.8]	56.6712
91	(83.8,84.8]	56.9069
92	(84.8,85.7]	81.9917
93	(85.7,86.6]	68.0002
94	(86.6,87.5]	51.6770
95	(87.5,88.4]	70.2406
96	(88.4,89.3]	50.2647
97	(89.3,90.3]	49.7290
98	(90.3,91.2]	49.6333
99	(91.2,92.1]	86.8271
100	(92.1,93.1]	84.2437