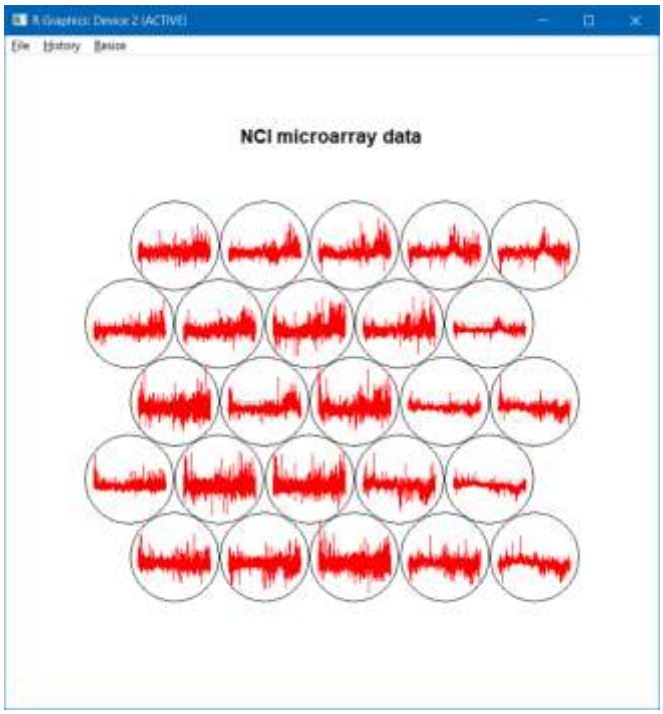
Kunal Mehta
kunalkam
50352960

Homework 3

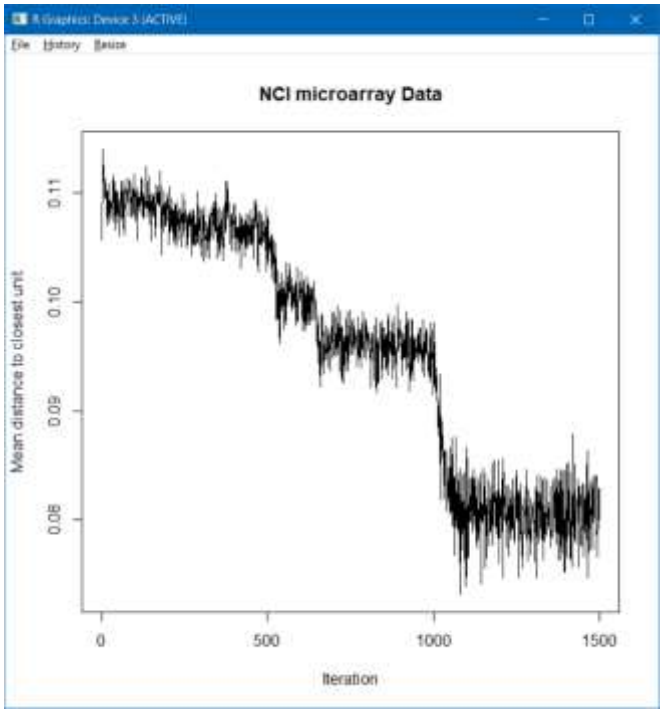
Problem 1

a. First step, I downloaded the ElemStatLearn from the website archives and installed it so it could be used when coding. I also included the kohonen library that we need for the Self-Organizing Maps.
b. The next step was to import the NCI dataset which was fairly straightforward. After importing it, I made sure to scale the data.
c. I set a seed of 100 so that for the same seed value all the following code would give the same result every time it was executed.
d. To build the self-organizing maps, we need to have a grid built with x and y dimensions as well as topology. After trying different dimensions and observing the result, I settled on x and y dimensions of 6x6. The topology is hexagonal. The plot for it is displayed below.
e. I extracted the code vector from the SOM and stored it in the codes variable.
f. Following this, I plotted the SOM, the mean average deviation from code vectors of the map, the no of points assigned to each cell and the map based on average distance, all of which are attributes of the self-organizing map.
   SOM:

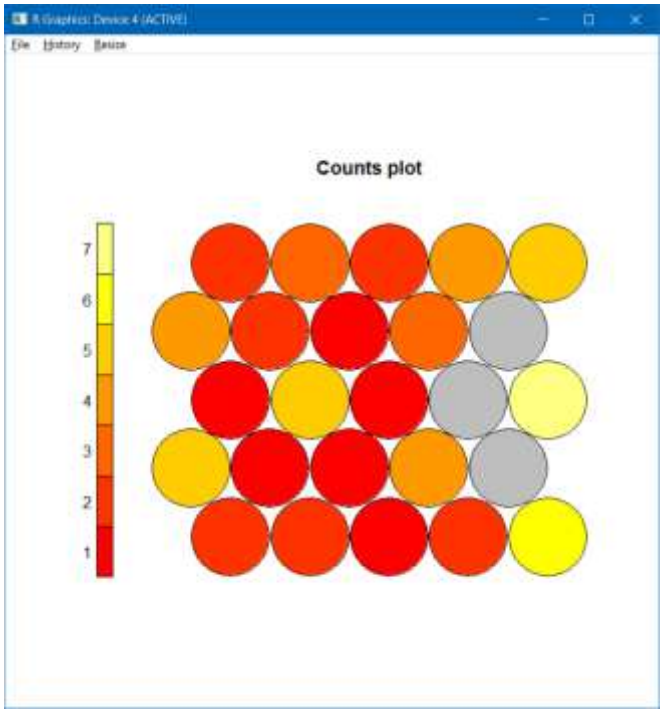


Kunal Mehta
kunalkam
50352960

Mean average deviation:

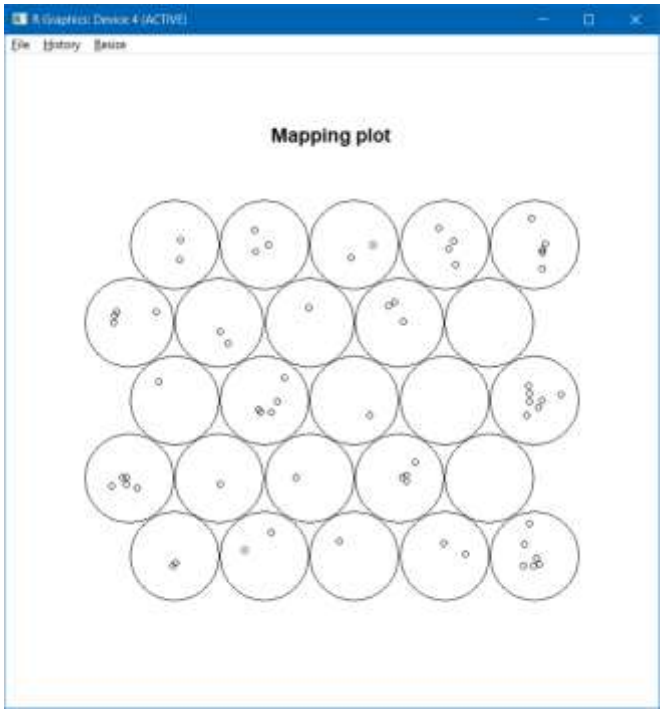


Count:

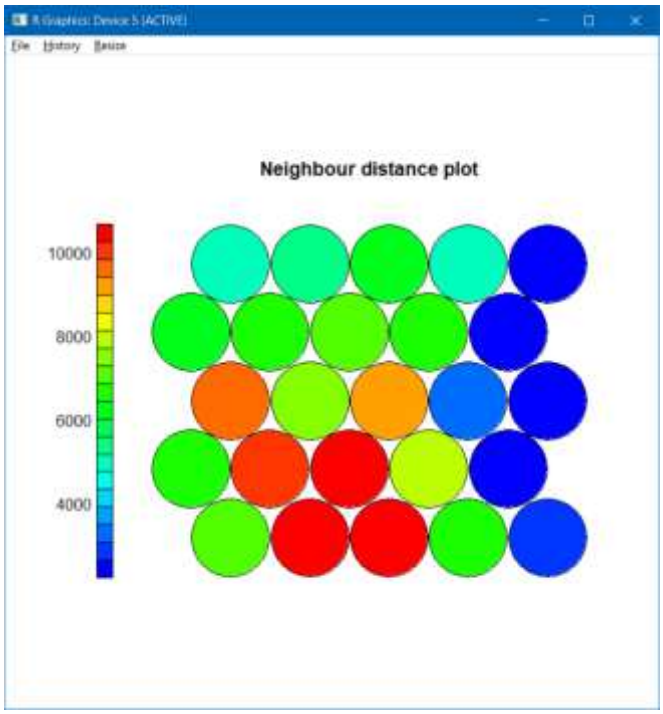


Kunal Mehta
kunalkam
50352960

Mapping:



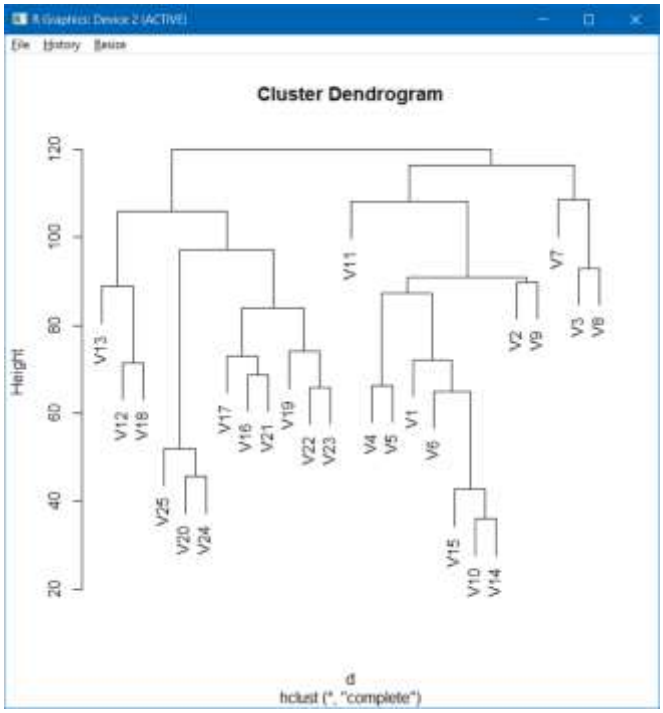
g. The next step was to set up the color palette as described in the SOM lab video. I used the same color palette and plotted the distance between the neighbours of the SOM.



h. For the component plane plots, on scrutinizing the dataset, it is evident that despite 64 columns, there are 14 unique ones. I was reasonably sure that the mapping would not be affected if the plots were done for only the unique columns, but since I was not sure, I have displayed the component plane plots for all of the 64 columns. Since they are 64

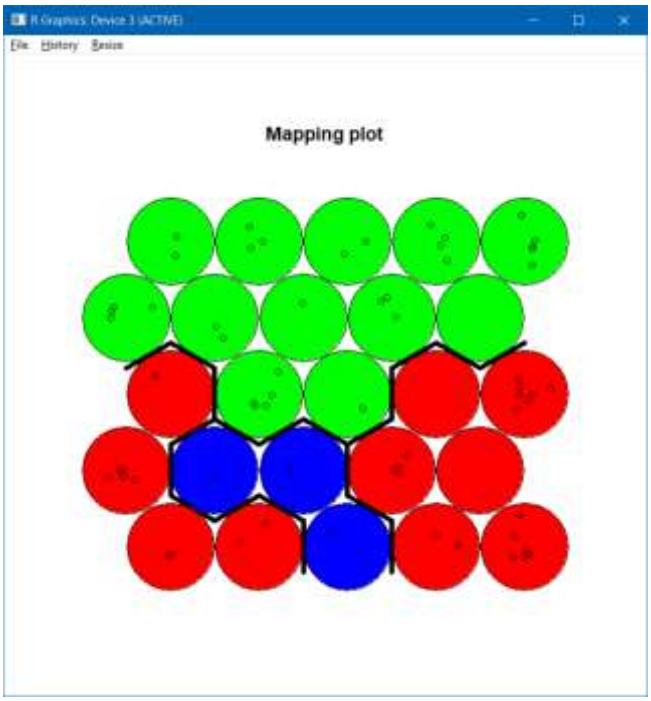
in number, I am not plotting them in the writeup since they would take up way too much space.

i. The next step was to hierarchically cluster the distance between the code vectors, which I did simply by the hclust command.



j. As can be seen, the dendrogram if cut at around 110 would yield 3 clusters. Therefore I cut the dendrogram at a height that gives three clusters using cutree with the argument k = 3.

k. The last step was to set up a color palette and plot the boundaries of distinction between clusters.

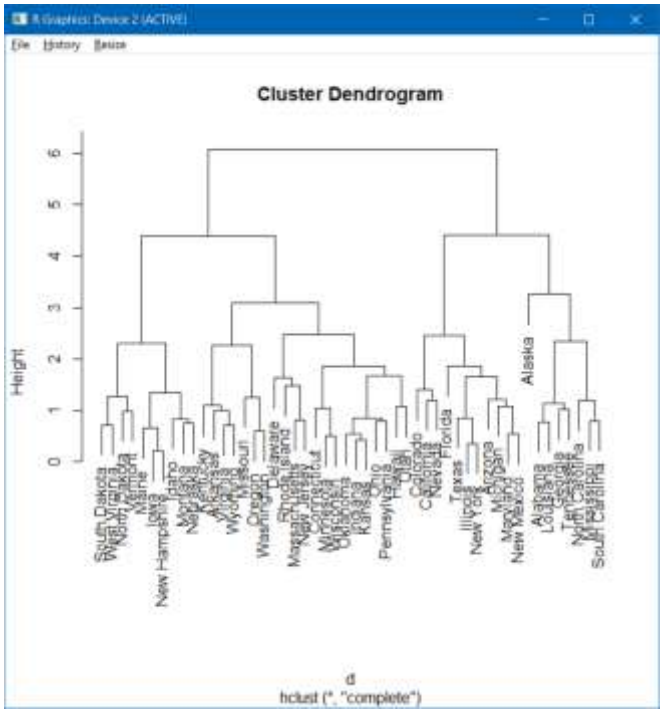


Kunal Mehta
kunalkam
50352960

Self-organizing maps as well cluster the data into 3 clusters. As such, it is not able to characterize the data into the 14 subtypes that exist. There is much overlapping of subtypes within the same cluster, and from the dendrogram it is clear to see that self-organizing maps in this case are not necessarily very good at characterizing tumor cells into groups.

Problem 2

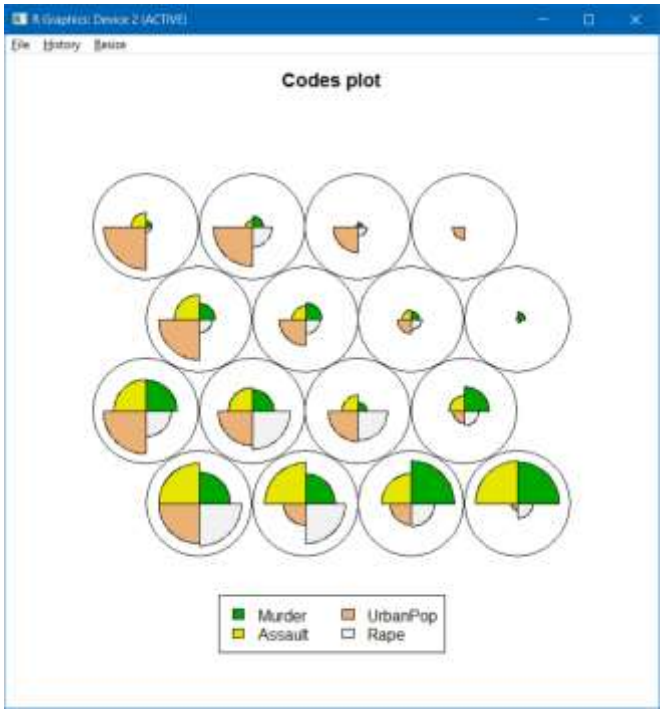
a. As usual, I loaded the ISLR and kohonen libraries, and then accessed the USArrests dataset. In variable scaled, I stored the centred and scaled USArrests data.
b. As per the question the first step is to hierarchically cluster the data with complete linkage. I did so using the hclust command and got the following dendrogram.



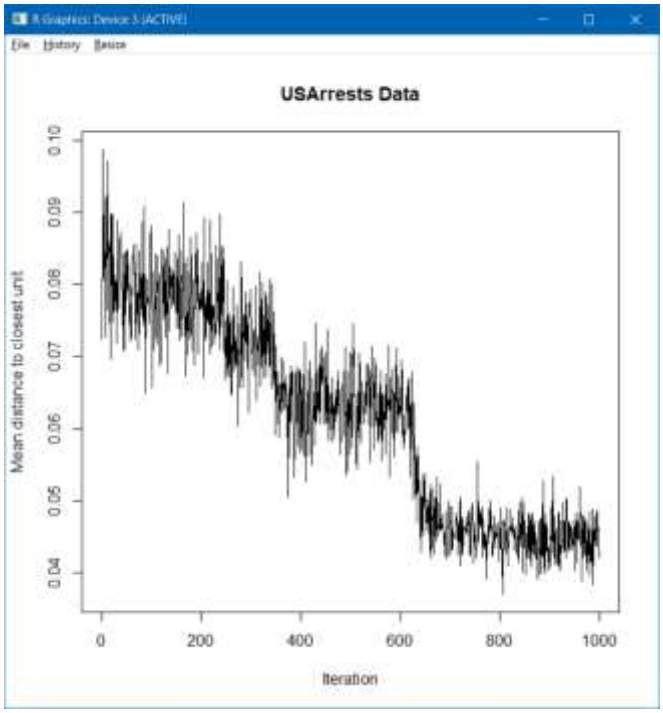
c. As can be seen from the dendrogram, 4 seems to be a good number of clusters to cut the tree for. Therefore I cut the tree such that it results in 4 clusters.
d. I set the seed once again to 100 so the results could be replicated. Considering that the data is 50 observations of 4 variables, I created a grid of 4x4 dimensions and got a self organizing map with 1000 iterations.
e. The plots are as follows:

Kunal Mehta
kunalkam
50352960

SOM:

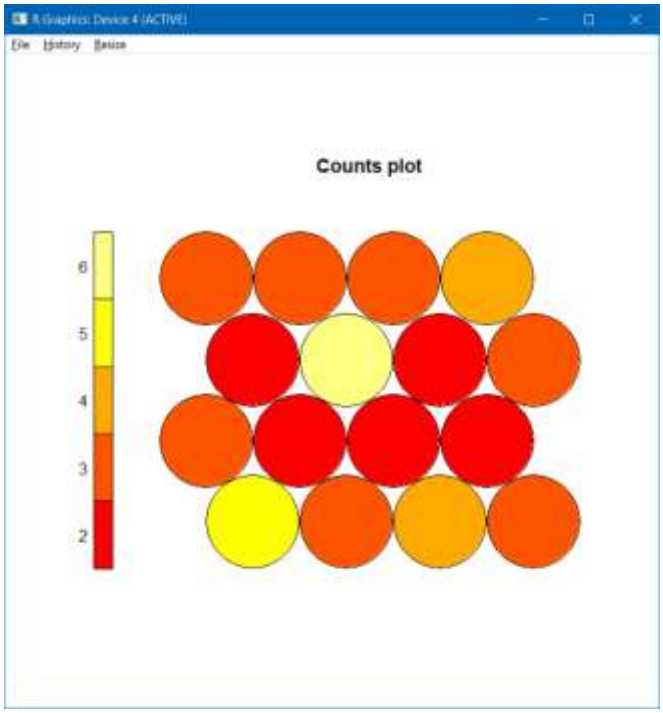


Mean Average deviation:

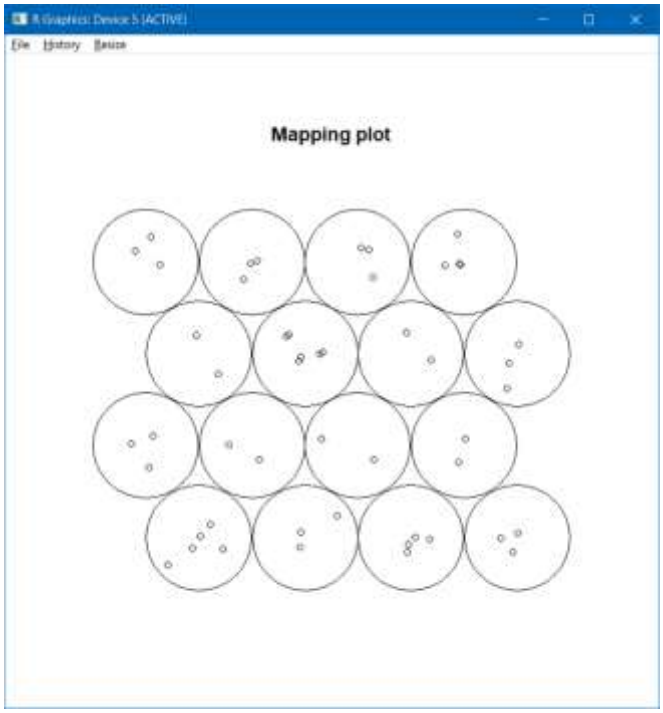


Kunal Mehta
kunalkam
50352960

Count:

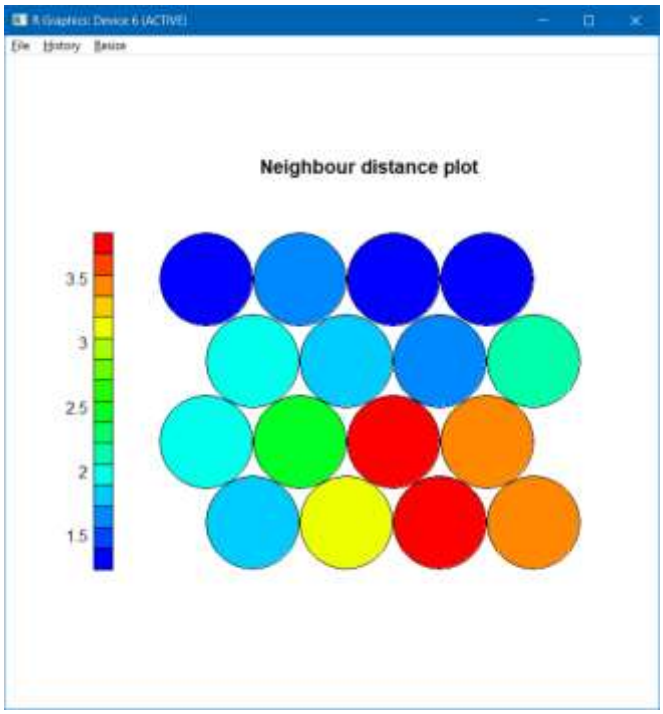


Mapping:



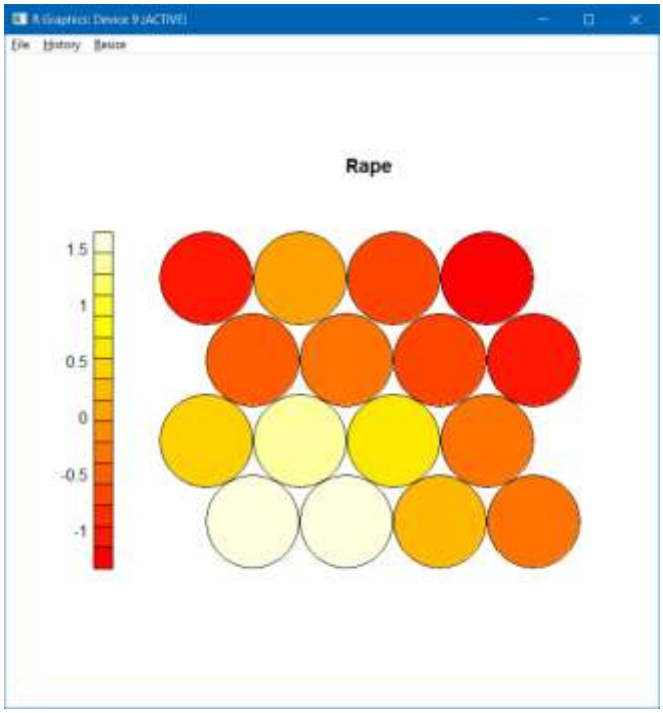
f. The next step, I defined the coolbluehotred color palette again, and plotted the distance between neighbours

Kunal Mehta
kunalkam
50352960



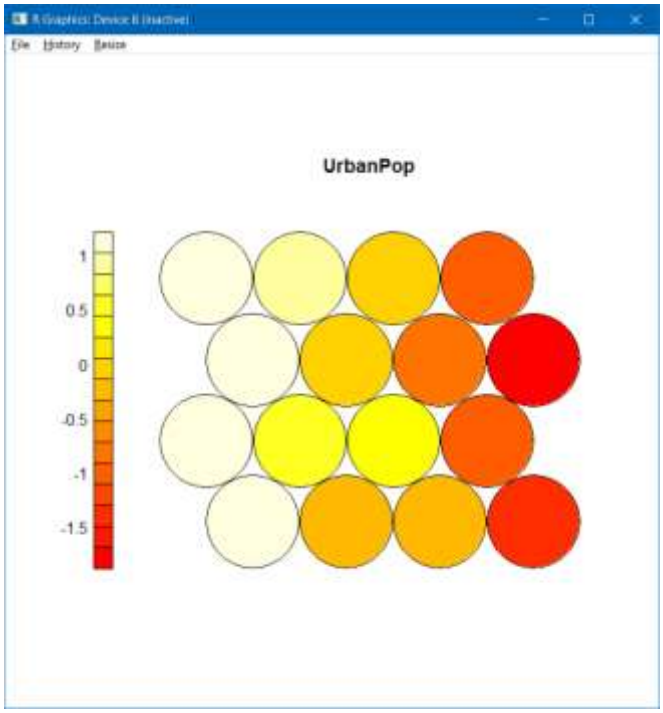
g. The next step was to plot the component plane plots. There being only 4 variables, this was pretty straightforward.

Rape:

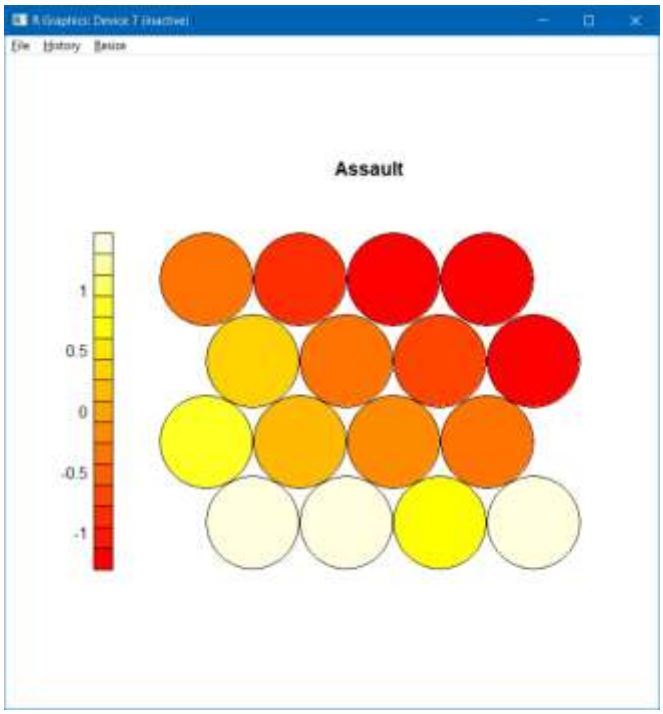


Kunal Mehta
kunalkam
50352960

UrbanPop:

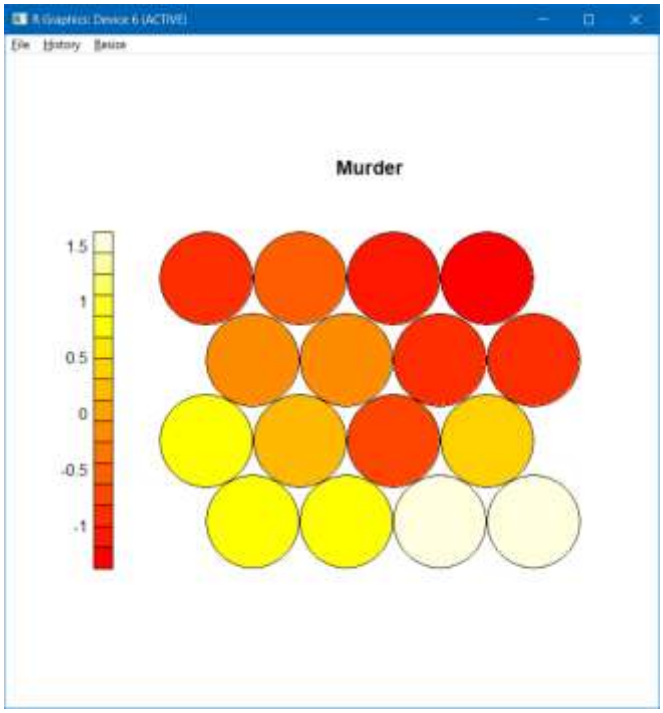


Assault:

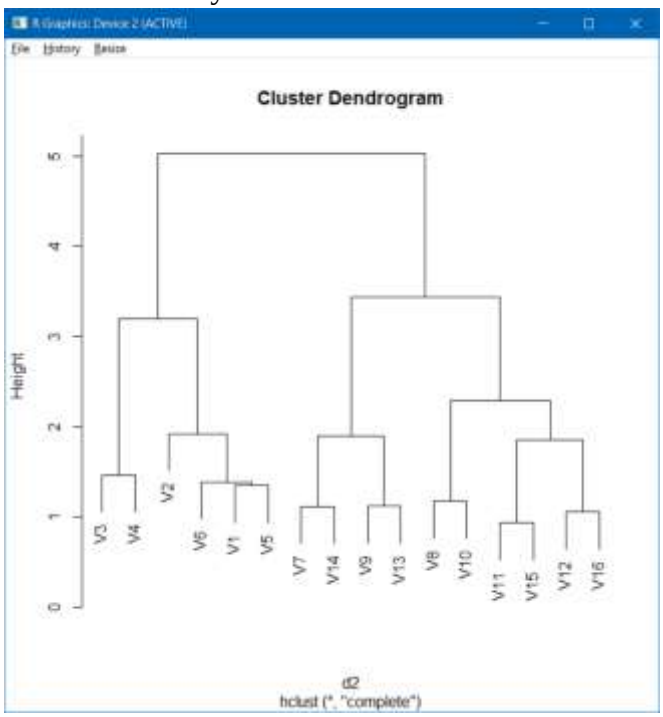


Kunal Mehta
kunalkam
50352960

Murder:



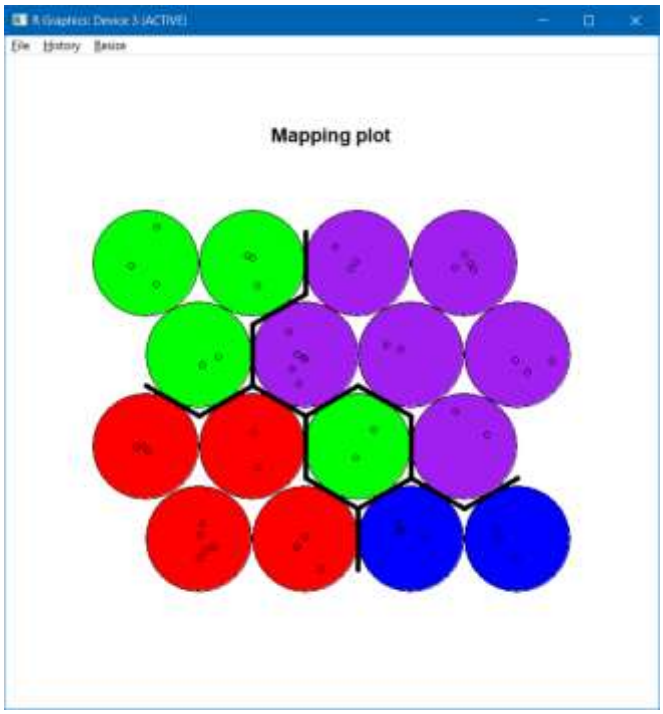
h. The next step was to hierarchically cluster the distance between code vectors.



i. As can be seen from the dendrogram, cutting it into 4 clusters is the best way to go. Therefore I used the cutree command to cut it such that k = 4

j. The last step was to define a color palette and plot the boundaries of distinction between the clusters

Kunal Mehta
kunalkam
50352960



k. The SOM results in 4 clusters which is what I did get in part a of the problem. This is pretty much what I expected since for a dataset that is this small, the advantages of SOM over hierarchical clustering is not immediately apparent in this example. SOMs allow much more flexibility in visual analysis as compared to hierarchical clustering. SOMs also preserve the topology of the dataset. However, hierarchical clustering is much easier to implement than SOMs, which also are prone to human error in visual analysis. One disadvantage of SOMs is that they vary every time due to the random element.

Problem 3

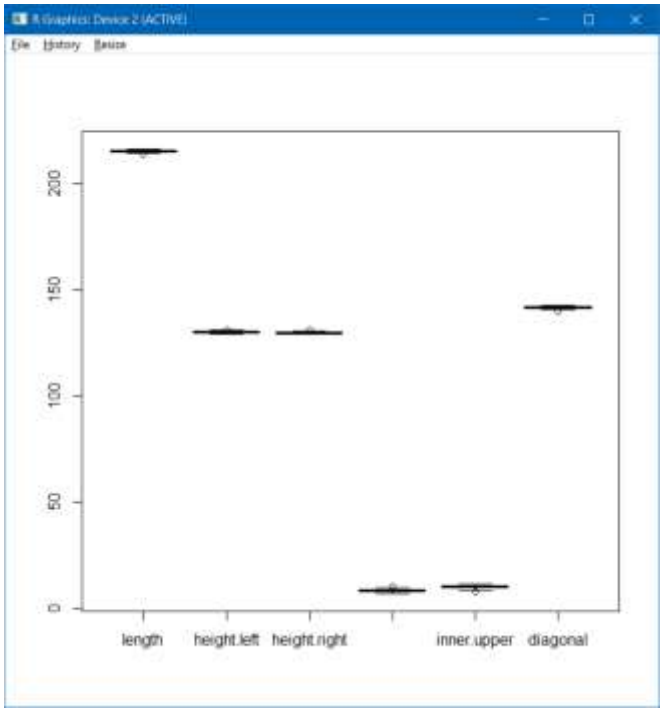
a. The first step was to load the SwissBankNotes dataset provided in UBLearns in R. I downloaded the file and accessed it using the load command.
b. Given that the first 100 notes are real and the next 100 are counterfeit, I stored the first 100 rows of the dataset into variable real, and the last 100 in the variable fake.

```
> head(real)
  length height.left height.right inner.lower inner.upper diagonal
1  214.8       131.0        131.1         9.0         9.7    141.0
2  214.6       129.7        129.7         8.1         9.5    141.7
3  214.8       129.7        129.7         8.7         9.6    142.2
4  214.8       129.7        129.6         7.5        10.4    142.0
5  215.0       129.6        129.7        10.4         7.7    141.8
6  215.7       130.8        130.5         9.0        10.1    141.4
> head(fake)
    length height.left height.right inner.lower inner.upper diagonal
101  214.4       130.1        130.3         9.7        11.7    139.8
102  214.9       130.5        130.2        11.0        11.5    139.5
103  214.9       130.3        130.1         8.7        11.7    140.2
104  215.0       130.4        130.6         9.9        10.9    140.3
105  214.7       130.2        130.3        11.8        10.9    139.7
106  215.0       130.2        130.2        10.6        10.7    139.9
>
```

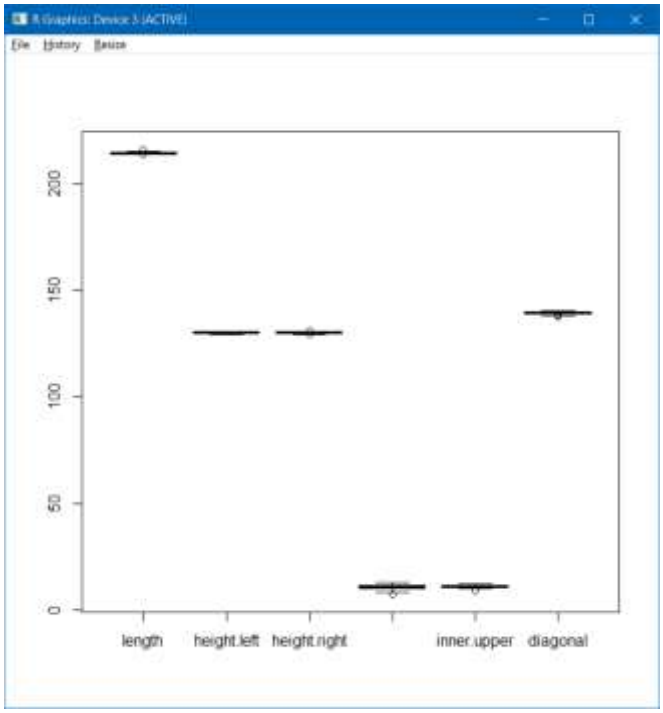
c. As a diagnostic, I boxplotted the entire dataset, and the real and counterfeit ones.

Kunal Mehta
kunalkam
50352960

Real:

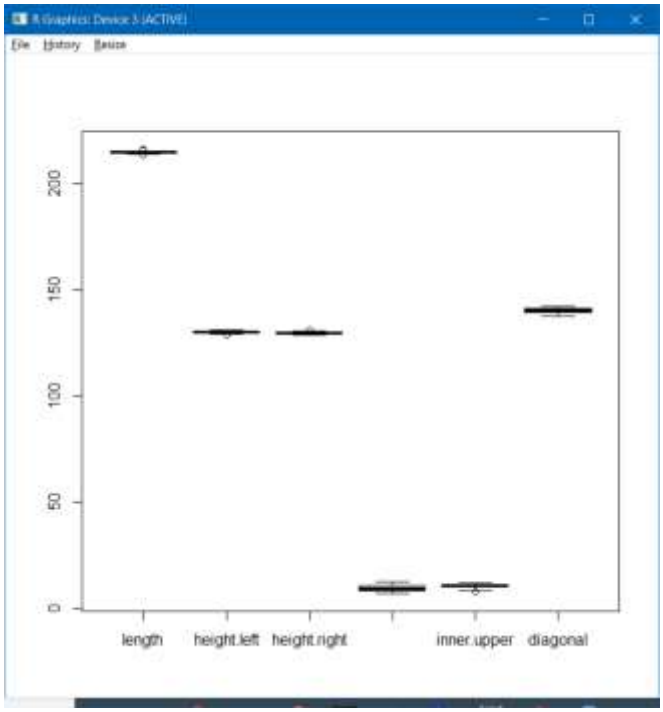


Counterfeit:



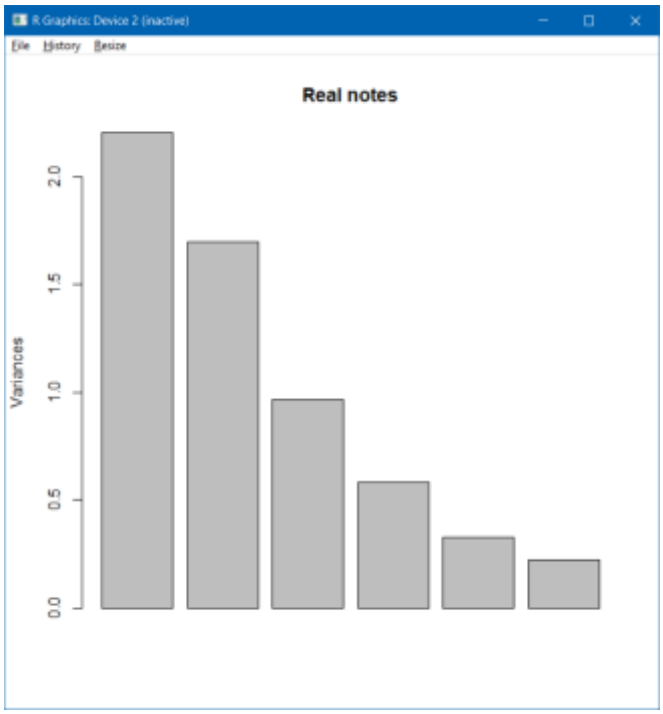
Kunal Mehta
kunalkam
50352960

All:



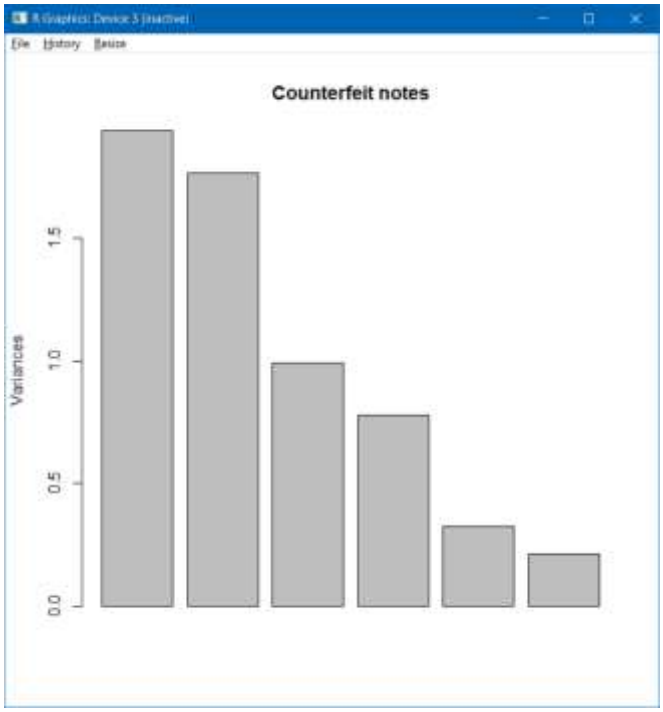
d. The next step was to perform PCA on the real dataset, the fake dataset and all the notes as one. Therefore using the prcomp command, I did so. The score plots of the three are as follows

Real:

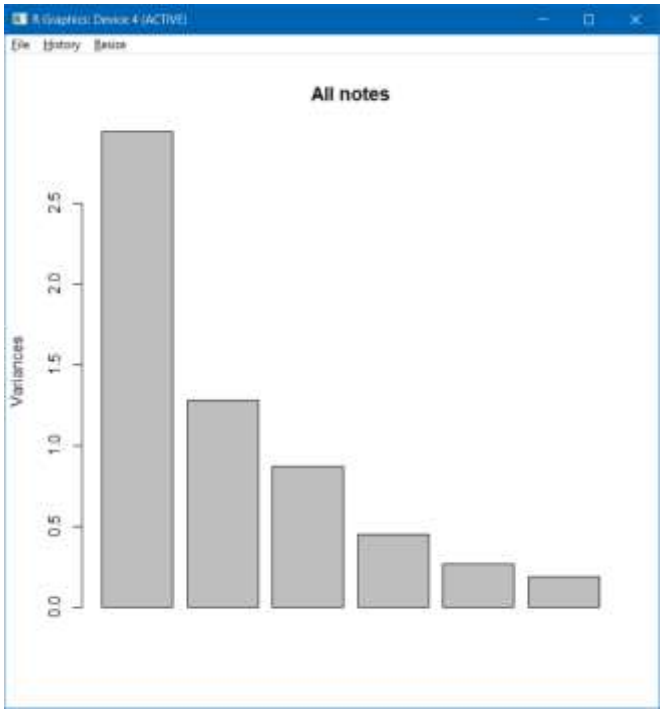


Kunal Mehta
kunalkam
50352960

Counterfeit:



All Notes:



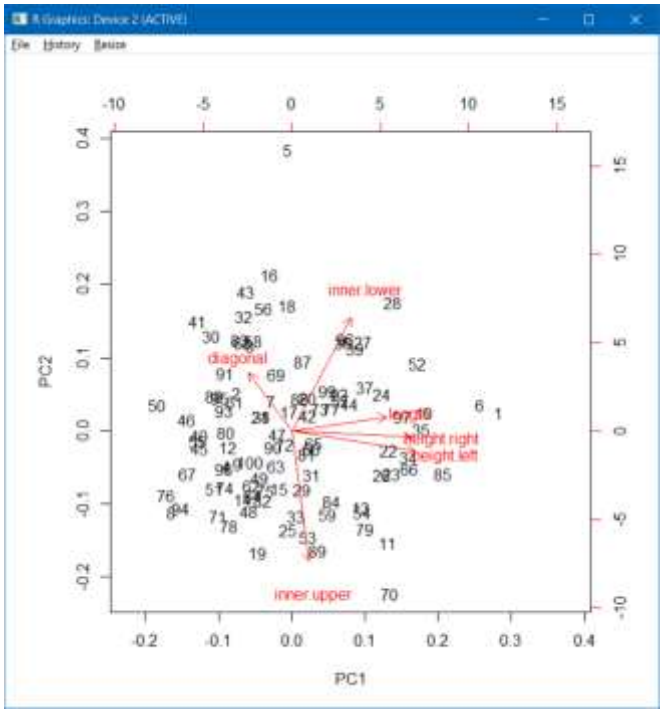
As can be seen, all the three have the first two principal components together accounting for a significantly huge percentage of the variance. This observation in empirical terms is given by the summary of all the 3 PCAs.

Kunal Mehta
kunalkam
50352960

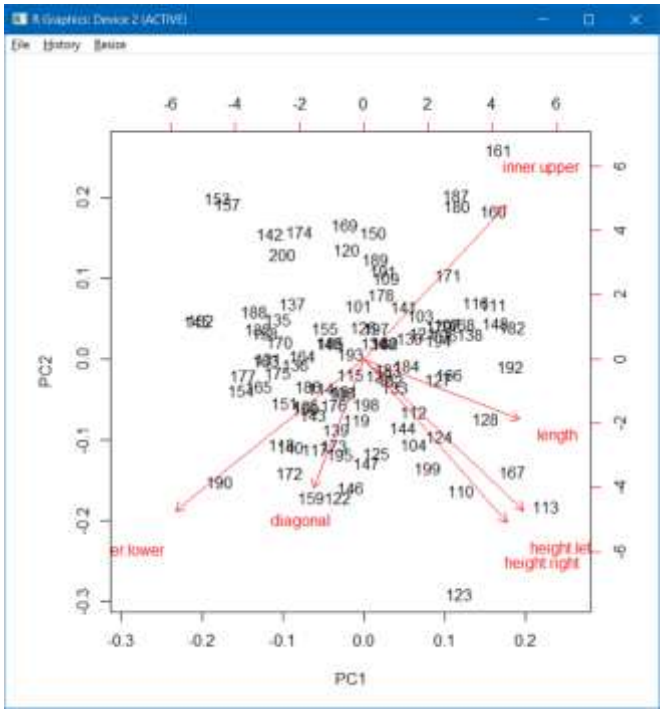
```
> summary(pca1)
Importance of components:
                          PC1    PC2    PC3     PC4     PC5     PC6
Standard deviation     1.4845 1.3026 0.9827 0.76348 0.57156 0.47340
Proportion of Variance 0.3673 0.2828 0.1610 0.09715 0.05445 0.03735
Cumulative Proportion  0.3673 0.6501 0.8111 0.90820 0.96265 1.00000
> summary(pca2)
Importance of components:
                          PC1    PC2    PC3    PC4     PC5     PC6
Standard deviation     1.3915 1.3285 0.9941 0.8823 0.56755 0.45840
Proportion of Variance 0.3227 0.2941 0.1647 0.1297 0.05368 0.03502
Cumulative Proportion  0.3227 0.6169 0.7816 0.9113 0.96498 1.00000
> summary(pca3)
Importance of components:
                          PC1    PC2    PC3    PC4     PC5     PC6
Standard deviation     1.7163 1.1305 0.9322 0.67065 0.51834 0.43460
Proportion of Variance 0.4909 0.2130 0.1448 0.07496 0.04478 0.03148
Cumulative Proportion  0.4909 0.7039 0.8488 0.92374 0.96852 1.00000
>
```

e. The last step was to plot the biplots of all the PCAs. This I did simply using the biplot command.

Real:

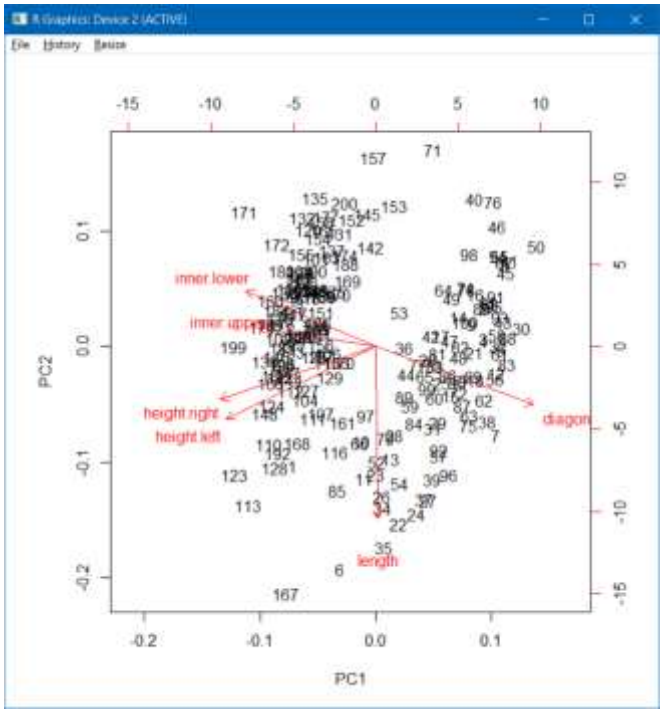


Counterfeit:



Kunal Mehta
kunalkam
50352960

All:



Each of the biplots showcases the effect of the variables of the dataset on observations. As regards the biplot of the entire dataset, a rough separation between two clusters can be seen which is probably the distinction between the fake and real notes. A cursory viewing also shows that most of the observations in one of the clusters are in the 1-100 range and the one to the left are in the 101-200 range.