# Individual Homework

*Petar Petrov h11729352*

## Regression Analysis

After loading the data from the csv file, the first thing that I did was to use the summary() function and see the structure of the data with str().

```
> summary(df)
     price         Distribution    Advertising        Sales
 Min.   : 4.99   Min.   :32.00   Min.   : 7867   Min.   :1653
 1st Qu.: 8.99   1st Qu.:45.00   1st Qu.: 9482   1st Qu.:2284
 Median : 9.99   Median :49.50   Median :10079   Median :2824
 Mean   : 9.89   Mean   :49.08   Mean   :10051   Mean   :3128
 3rd Qu.:10.99   3rd Qu.:53.00   3rd Qu.:10694   3rd Qu.:3630
 Max.   :13.99   Max.   :70.00   Max.   :12161   Max.   :8537
```

At a first glance price column seems reasonable, with good quartile ranges, mean and median. Probably the minimum price is a bit low, but it is still possible to have such data, so I won't remove it from the data set.

The Distribution column also looks fine, with good interquartile range and min/max values.

The Advertising column is reasonable as well.

When we see the Sales column, we can see, that there is probably an outlier, because the 3rd Quartile (75% of the data) is below 3630 and the max is 8537, which is kind of high, but I will keep it for now and take a look at it on a later stage, when I plot it.

```
> str(df)
'data.frame':  100 obs. of  4 variables:
 $ price       : num  5.99 6.99 8.99 6.99 10.99 ...
 $ Distribution: int  53 60 37 53 40 43 46 53 47 46 ...
 $ Advertising : int  8378 10603 9916 10068 9430 10947 11981 9140 9866 8032 .
..
 $ Sales       : int  6056 5836 2651 5169 2046 1782 2902 2909 2728 1653 ...
```

The structure of all the values seems fine, all of them have the required data type, while we are lacking a factor variable here.

As a next step, I decided to check the different distributions and plot them all against each other, using the gpairs() function.

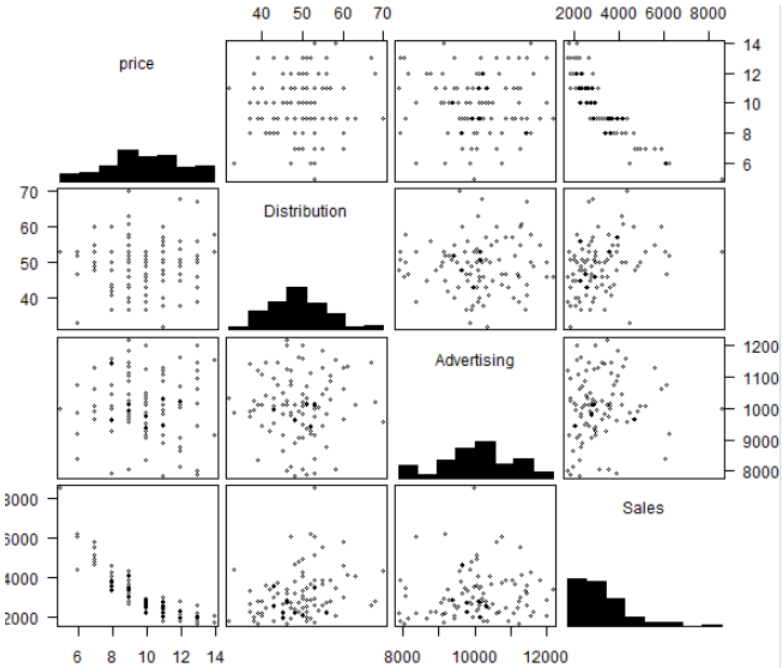



**Fig 1.1 - Scatterplot**

We can see here, that most of the variables have close to normal distribution, except for sales. That is why, I've decided to create a new plot and include in the data frame also the log of Sales, because it has right-skewed distribution.

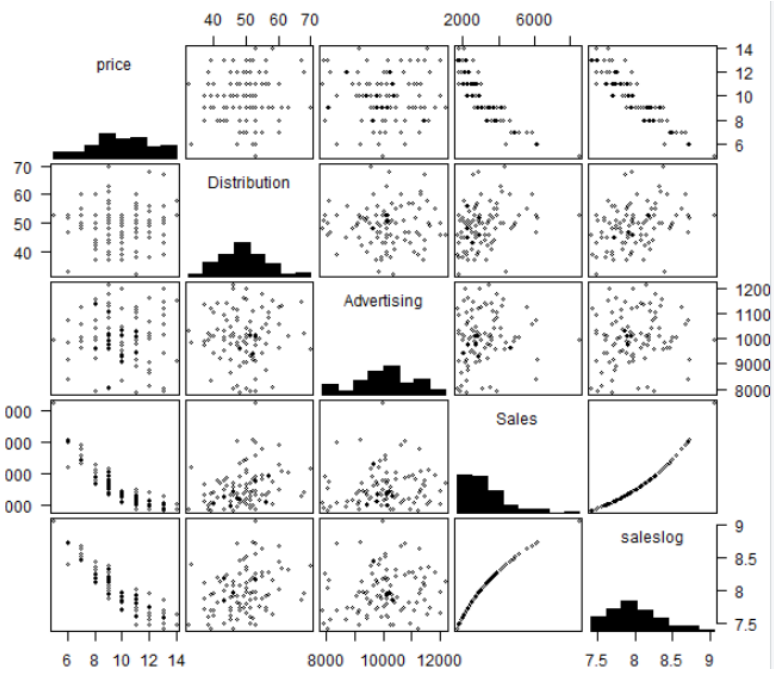



**Fig. 1.2 - Scatterplot**

Here, on this figure, we can see, that now sales has clearly more normal distribution, but the difference between Sales and saleslog's scatterplots with the other variables stays relatively the same.

We can also see, that except for the price variable, all the others seem quite scattered and random with regards to sales and saleslog.

Next, I am using the corrplot.mixed() function in order to check the correlation coefficients between all the possible variables, that I may use to build a regression model and pick the right ones to start with. I am using ellipse forms and the thinner the ellipse, the bigger the correlation. Also the colors mean: the darker the blue – stronger positive correlation; the darker the red – stronger negative correlation.


**Fig 1.3 – Correlation Matrix**

Here we can see that the Sales are strongly negatively correlated to the price of a product(-0.88). The more the price grows, the more the sales drop. The coefficient is even better for the saleslog~price, =-0.92.

Also we can see that the Distribution is medium and positively correlated to Sales and saleslog.

As for Advertising, the correlation is weak, almost non-existent, but positive.

Next, I move on and build the first linear model using the function lm(). The first will be using only one of the variables – the price. I have built **2 models (Sales~price) and (saleslog~price)**. The R squared(0.7681/0.8383) and Adjusted R squared(0.7657/ 0.8367) are both bigger, when using the saleslog, so I will stick with it and provide explanations only for it.

Here is the output:

```
> m5<-lm(saleslog~price,data=df)
> summary(m5)

Call:
lm(formula = saleslog ~ price, data = df)

Residuals:
     Min       1Q   Median       3Q      Max
-0.33775 -0.09168 -0.00394  0.08749  0.35932

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  9.540229   0.070130  136.04   <2e-16 ***
price       -0.156846   0.006958  -22.54   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1354 on 98 degrees of freedom
Multiple R-squared:  0.8383,   Adjusted R-squared:  0.8367
F-statistic: 508.2 on 1 and 98 DF,  p-value: < 2.2e-16
```

When looking at the coefficients, we can notice the small numbers, this is because we are using the logarithm function and they have to be interpreted in a different way. We can see, that both

the intercept and the price are statisically significant, because of the three stars, which are standing at the end of the rows.

To interpret the b1 coefficient, we can **say that by increasing the price by 1 unit, we expect the sales to decrease by ~15%.**

The standard error for both the intercept and the price looks reasonable.

The R-squared is 0.83, which means, that 83% of the sales can be explained by changes in price.

The F-statistic shows a *p*-value << .05, so we reject the null hypothesis and accept h1, which means that price has significant effect on the sales.

In order to further examine the correctness of the model, I use the command par(mfrow=c(2,2)) and again plot the existing model, but this time 4 different plots are shown, which measure the residuals, the distribution and also help us find outliers.

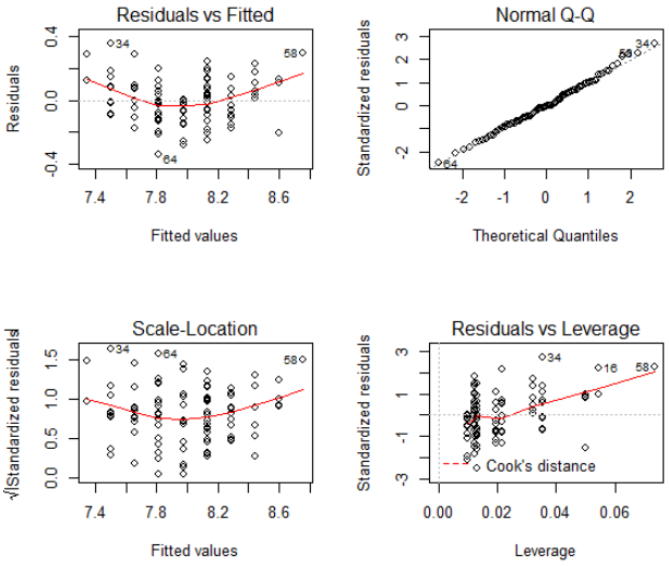



**Fig 1.4 – Checking Model Fit**

Here each graph shows different thing, we can see that our residuals' distribution is almost normal, which is good(Q-Q plot).

Also the other graphs show rather random distribution of the residuals and there is no pattern we can notice.

The only thing that bothered me, is the last graph(Cook's distance), which shows a odd turn of the red line, which can be explained by the outlier(nr 58).

So I've decided to check the row with the suggested outlier(also other possible outliers were shown, but they seem ok).

```
> df[58,]
   price Distribution Advertising Sales saleslog
58  4.99           53        9968  8537 9.052165
```

The Sales value here is an outlier, so I've decided to cut it and check the model without it.

```
> dftest<-df[-58,]
> m5<-lm(saleslog~price,data=dftest)
> summary(m5)

Call:
lm(formula = saleslog ~ price, data = dftest)

Residuals:
     Min       1Q   Median       3Q      Max
-0.33910 -0.09237 -0.00539  0.08295  0.34975

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  9.496400   0.071208   133.4   <2e-16 ***
price       -0.152736   0.007038   -21.7   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1326 on 97 degrees of freedom
Multiple R-squared:  0.8292,   Adjusted R-squared:  0.8275
F-statistic:   471 on 1 and 97 DF,  p-value: < 2.2e-16
```

Here is the output, but as we can see, the R-squared is a bit less, so I’ve just decided to leave it in the dataframe and continue with the next model, this time with all the explanatory variables and check for the outcome. I will interpret only the new information here, not every coefficient and output from the lm() function.

```
> m2<-lm(saleslog~price+Distribution+Advertising,data = df)
> summary(m2)

Call:
lm(formula = saleslog ~ price + Distribution + Advertising, data = df)

Residuals:
     Min       1Q   Median       3Q      Max
-0.09384 -0.02374 -0.01397  0.01799  0.21874

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   8.354e+00  5.911e-02 141.330  < 2e-16 ***
price        -1.594e-01  2.294e-03 -69.482  < 2e-16 ***
Distribution  1.696e-02  6.259e-04  27.090  < 2e-16 ***
Advertising   3.771e-05  4.522e-06   8.338 5.44e-13 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.0446 on 96 degrees of freedom
Multiple R-squared:  0.9828,   Adjusted R-squared:  0.9823
F-statistic:  1832 on 3 and 96 DF,  p-value: < 2.2e-16
```

Here, the first and most obvious thing is that the R-squared is more than 98%, but this can be misleading, because of overfitting of the data. Again the interpretation for all the coefficients is the same, all results are statistically significant, which is quite odd, because I’ve checked earlier the p-value for saleslog~Advertising alone was 0,18, but when included in the multiple predictors model, is signifficant.

So I’ve decided to remove the Advertising from the model and try it again only with price and Distribution as explanatory variables.

```
> m3<-lm(saleslog~price+Distribution,data=df)
> summary(m3)

Call:
lm(formula = saleslog ~ price + Distribution, data = df)

Residuals:
     Min       1Q   Median       3Q      Max
-0.12631 -0.03578 -0.01089  0.03127  0.21346

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   8.7347474  0.0490881  177.94   <2e-16 ***
price        -0.1597988  0.0029961  -53.34   <2e-16 ***
Distribution  0.0170066  0.0008176   20.80   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.05826 on 97 degrees of freedom
Multiple R-squared:  0.9704,   Adjusted R-squared:  0.9698
F-statistic:  1590 on 2 and 97 DF,  p-value: < 2.2e-16
```

Here again we can observe a pretty high value of R-squared, but this time I think the model is appropriate and can serve a good purpose, because earlier, I’ve checked also the significance of saleslog~Distribution and it was 0.00119, which is quite allright and still statistically significant.

**The distribution coefficient means, that if we increase it by 1, Sales will increase with 1,7%.**

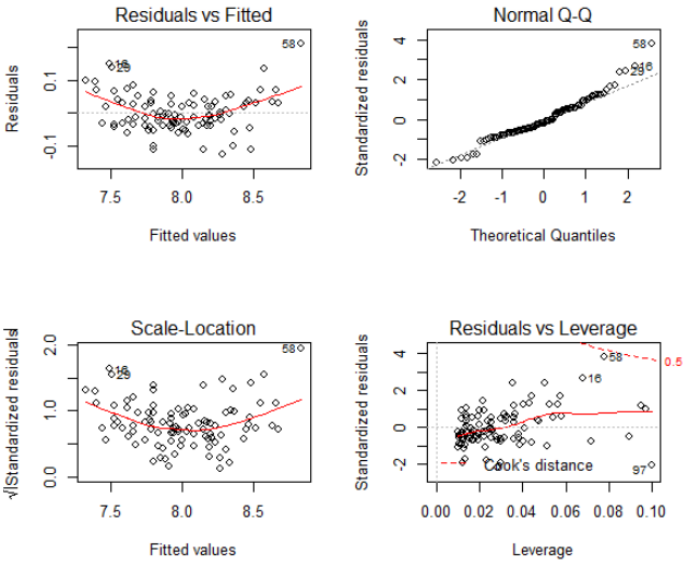



**Fig. 1.5 – Checking Model Fit(2)**

Again the distribution seems fine and random, we don’t observe pattern in the residuals’ position, so we accept this model.

The final thing I did, was to plot both the models on the same scatterplot and compare their distribution:

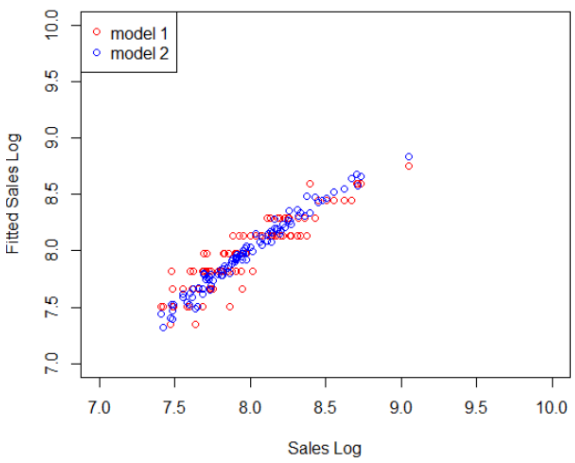



**Fig 1.6 – Model Comparison Plot**

We can see on this plot, that model 2's points (observations) are closer to each other and the red dots are more scattered, which means that the fit for m2 is better.

To conclude, I have decided to go with the model, which does not overfit the data(so much as when we add Advertisments) and which has the largest Adjusted R-squared from all of the models I've tried. The model I chose is lm(saleslog~price+Distribution).

## Segmentation

The first step again is to load the data and check the summary and get to know the structure and the relevant statistics.

The first thing I noticed from the str() function was that the sex variable is not formatted as factor variable, so I decided to transform it, because that is usually how sex is represented for easier computation.

```
$ sex           : Factor w/ 2 levels "0","1": 2 2 1 2 1 2 1 2 1 2 ...
```

Now we see, that sex is coded as factor, with 1 being Female, and 2 being Male. The other information we can get here is that there are 359 observations with 20 different columns.

I think that, regarding all the attributes columns(A_XX) they are inversed again because they can be used as a factor variable. If they were not reversed, R would treat them hierarchically starting with the smallest of them all, which was actually with the highest importance(weight) before factoring, but after the factoring, it will get the factor 1. The problem arises, when for example using the max() function, the max will appear 12, which won't be correct, since this will be the

least important attribute the user placed. However I am not factoring all the columns and am using simply their values in the segmentation.

Another possible explanation is because the log() function is used before the data was given to us.

Next function, I would like to discuss, is the summary(). I won't copy all the output, just a brief summary. The rebuy and the attributes columns seem ok, as well as the sex column with 61 women and 298 men in the data frame. However, I see some possible outliers in the age and price columns:

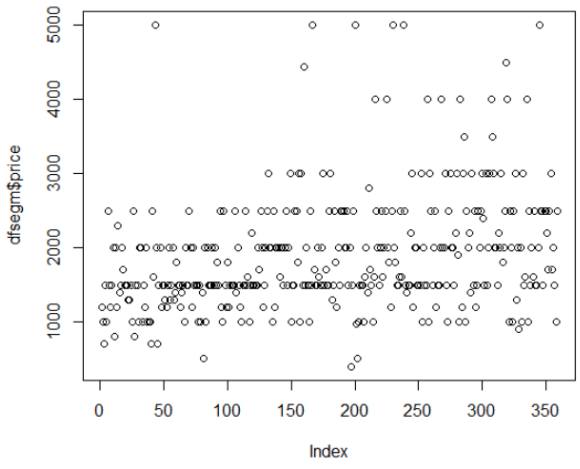
```
> summary(dfsegm$age)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  14.00   21.00   26.00   27.83   31.00  100.00
```

The 3rd Quartile is at 31, but the max value is 100, which seems like a possible error.

```
> summary(dfsegm$price)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    400    1500    1600    1898    2200    5000
```

Same goes for the price column.

I check the plots:



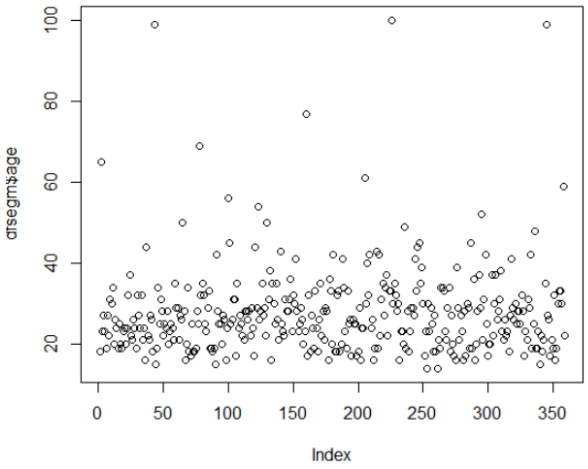




**Fig.2.1 – Price Plot** **Fig.2.2 – Age plot**

The price values seem reasonable, but it is possible that these values for age are wrong. Probably the users didn't want to specify their exact age, but by checking the data in the other columns, it seems reasonable and I chose not to remove them from the data frame.

which(dfsegm$age > 80); dfsegm[c(43,226,345),]

After the initial checking of the data and the transformation and adjustments made, I am continuing with the clustering. The first thing I must do, is to transform the non-numeric values to numeric(factor variable – sex). In order to achieve this, I am using the daisy() function. After converting the values, I am starting with the hierarchical clustering method.

```
seg.hc <- hclust(seg.dist, method="complete")

plot(seg.hc)
```

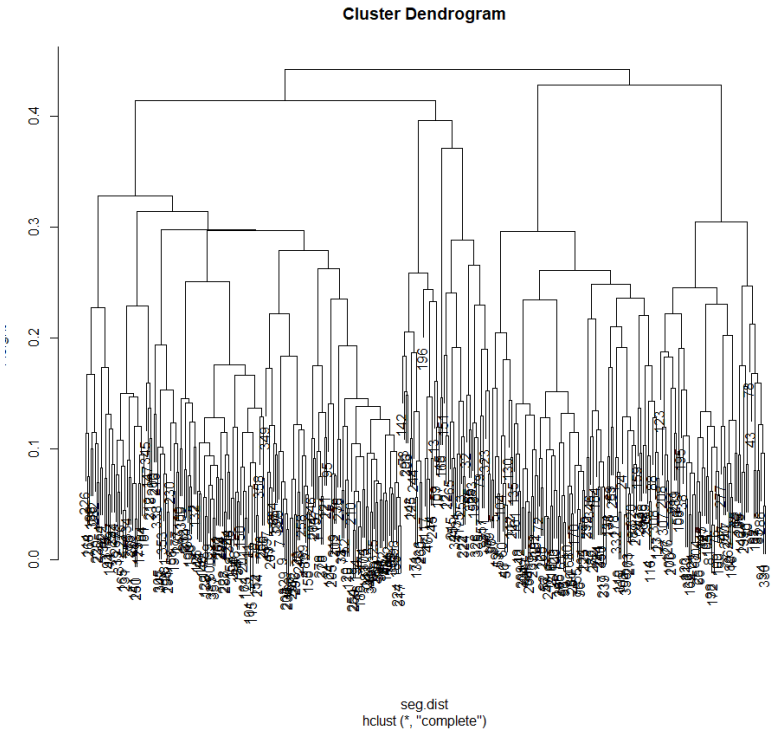



**Fig. 2.3 – Dendrogram Complete**

The output is uninterpretable, but we can at least see where the distances become longer and guess the number of clusters we can have. However we are going to check that later using other method.

First I am going to try to guess the appropriate number of clusters. I will go for four and see what the outcome of the clustering is.

```
plot(seg.hc)

rect.hclust(seg.hc, k=4, border="red")

seg.hc.segment <- cutree(seg.hc, k=4) # membership vector for 4 groups

table(seg.hc.segment)
```

```
seg.hc.segment
  1   2   3   4
167  89  56  47
```

We can see, that there are 4 segments, with the 1st being the biggest one with 167 of the answers.

Next, I will perform a basic check, which will give us information about the means, but this information is very important and can be very useful:

```
seg.summ <- function(data, groups) {
  aggregate(data, list(groups), function(x) mean(as.numeric(x)))
}

seg.summ(dfsegm$sex,seg.hc.segment)
```

```
> seg.summ(dfsegm[,c(1:5)],seg.hc.segment)
  Group.1 rebuy_3months rebuy_6months rebuy_12months rebuy_24months  rebuy_no
1       1     0.0000000     0.0000000               0               0 1.0000000
2       2     0.0000000     0.0000000               0               1 0.0000000
3       3     0.0000000     0.0000000               1               0 0.0000000
4       4     0.4468085     0.5319149               0               0 0.0212766
```

We can clearly see from this output, that the means for rebuy 12/24 months and rebuy_no are all equal to 1. Which means, that this method, with this number of clusters used this as the main criteria for creating the model. This may be misleading, so I am going to check for the appropriate number of clusters by plotting the values of the vector containing the distances between the variables (seg.hc) , but this time I am going to plot them using the column height. It shows the values distributed based on the distance they are away from the nearest one. The following output appears:

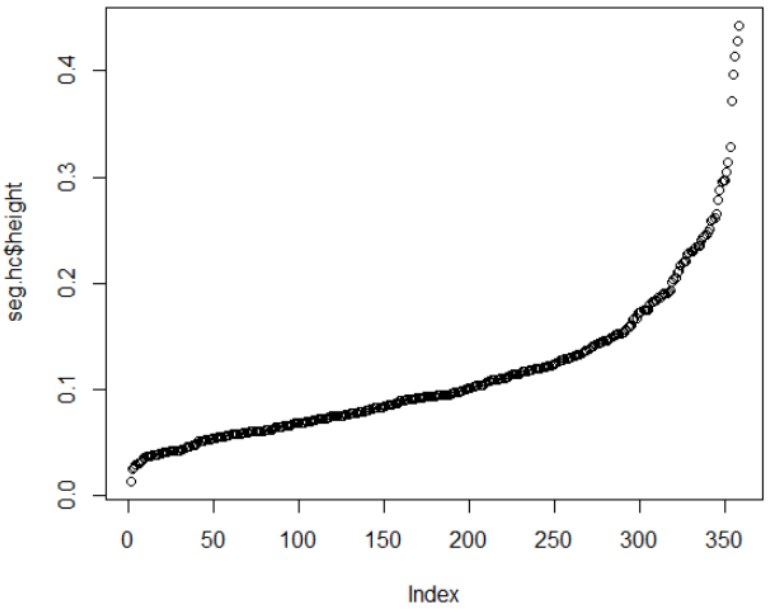



**Fig. 2.4 – Plot of seg.hc$height**

We can see here, that it would be appropriate to create 5-15 clusters, because there the differences are most noticeable. The density of the values with index less than 350 is too large, so they are not relevant for clustering criteria. Therefore I plot only the values from 350 to the end in order to finally decide on the number of clusters.

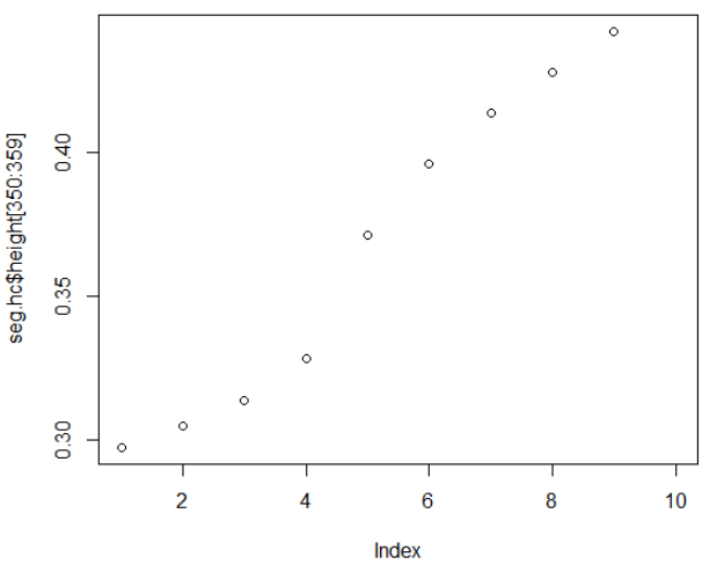



**Fig 2.5 – Plot of seg.hc$height 350:359**

Here we can clearly see the huge gap between the 4th and 5th index. The values before and after that have moderate and acceptable gaps, but this one is bigger, so I decide to go with 5 clusters and try the hclust() method again.

seg.hc.segment2<- cutree(seg.hc,k=5)

table(seg.hc.segment2)

```
seg.hc.segment2
  1   2   3   4   5
167  89  56  27  20
```

Again we see, that the first cluster is with most of the data in it, with the number decreasing, when the cluster index increases.

I run the function to calculate the means again and receive the following output (again just part of the output printed here):

```
> seg.summ(dfsegm[,c(1:5)],seg.hc.segment2)
  Group.1 rebuy_3months rebuy_6months rebuy_12months rebuy_24months   rebuy_no
1       1    0.00000000     0.0000000              0              0 1.00000000
2       2    0.00000000     0.0000000              0              1 0.00000000
3       3    0.00000000     0.0000000              1              0 0.00000000
4       4    0.03703704     0.9259259              0              0 0.03703704
5       5    1.00000000     0.0000000              0              0 0.00000000
```

We can again notice here, that the clustering was based on the timeframe for next buy of the product. This time I would leave it like this, because I have checked based on the distance and this seemed the most appropriate option. We can also see, that the age and sex doesn't matter for the clustering, because the values are close to each other for all the segments.

```
> seg.summ(dfsegm[,c(19:20)],seg.hc.segment2)
  Group.1      age      sex
1       1 28.26347 1.790419
2       2 27.22472 1.865169
3       3 28.32143 1.946429
```

```
4        4 26.55556 1.666667
5        5 27.25000 1.900000
```

After settling on the number of segments, I move on to k-means clustering method to check the groups there. I am using the same number of centroids, as the number of clusters from the dendrogram(5).

The first step to do, is to convert the factor variable to numeric one:
seg.df.num$sex <- ifelse(dfsegm$sex==1, 0, 1)

Then I use the kmeans() function with k=5 and n=50, so it can make 50 iterations and select the model with the least error. The least one that it finds is equal to:

```
> sum(kmeanssegm$withinss)
```

The error is 12%.

```
[1] 1206890
```

Next, I move on to function which calculates the means of all the columns and check the outputs:

```
> seg.summ(dfsegm, kmeanssegm$cluster)
  Group.1 rebuy_3months rebuy_6months rebuy_12months rebuy_24months  rebuy_no    price   A_brand A_CPUbrand A_CPUpower     A_RAM
1       1    0.03225806    0.08064516      0.1612903      0.1612903 0.5645161 1015.613 0.1141935  0.1190323  0.3622581 0.3087097
2       2    0.05555556    0.04166667      0.1388889      0.2500000 0.5138889 2683.333 0.2954167  0.1394444  0.3823611 0.2548611
3       3    0.06250000    0.18750000      0.2500000      0.1250000 0.3750000 4433.938 0.5112500  0.1556250  0.2287500 0.2606250
4       4    0.03125000    0.07031250      0.1484375      0.2656250 0.4843750 1502.344 0.1474219  0.1125000  0.3908594 0.2834375
5       5    0.12345679    0.06172840      0.1604938      0.3086420 0.3456790 1997.531 0.1676543  0.1467901  0.4172840 0.2760494
  A_harddisk A_displaysize A_displayres   A_CD_DVD  A_network  A_weight A_battery   A_price      age      sex
1  0.2166129     0.1424194    0.1393548 0.10677419 0.07096774 0.1238710 0.1922581 0.6135484 29.43548 1.774194
2  0.1829167     0.2347222    0.2113889 0.08541667 0.08902778 0.1965278 0.2848611 0.2654167 25.88889 1.916667
3  0.2375000     0.1968750    0.1506250 0.09812500 0.10687500 0.2137500 0.2462500 0.0787500 39.12500 1.875000
4  0.2139844     0.1865625    0.1293750 0.10976563 0.10078125 0.1075781 0.2691406 0.4578906 26.52344 1.765625
5  0.2206173     0.2538272    0.1966667 0.09407407 0.12185185 0.1660494 0.2996296 0.3140741 28.16049 1.888889
```

Here it is obvious, that now the customers are not grouped based on the next buy columns. It looks like that many factors influence the clusters and there is no pattern like with the hierarchical clustering. However, I can notice, that group nr.3 has significantly higher means for price, age and brand preferences. This looks like the members of the group are older and ready to pay more for a better product, than the members of the other groups. The group is made out of just 16 members, but they can be valuable customers.

```
> length(which(kmeanssegm$cluster==1))
[1] 62
> length(which(kmeanssegm$cluster==2))
[1] 72
> length(which(kmeanssegm$cluster==3))
[1] 16
> length(which(kmeanssegm$cluster==4))
[1] 128
> length(which(kmeanssegm$cluster==5))
[1] 81
```

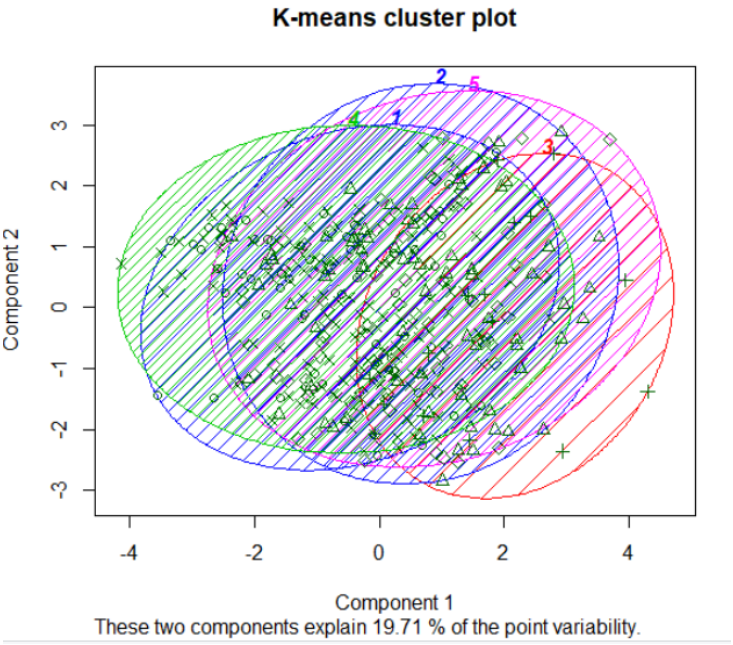



**Fig. 2.6 – K-means cluster plot**

As a next step, I plotted the cluster plot to get an idea of the clusters and check if there is some pattern, or some obvious segments. As said, most of the points are in the middle and the clusters are overlapping, but we can see that, as mentioned, cluster number 3 is a bit to the right with less points near the middle of the plot.

As a last step for this analysis, I am plotting the different variables' distribution in the different clusters with the function boxplot(). This will help us with judging the different segments and the people that are part of them, by visually comparing them to one another. I am going to include only the age and price variables here, because there are 20 variables and it would take too much space.

On Fig. 2.7, we can see, that segment 1 seems to be the price sensitive group, with the smallest value there. The members of the group don't care about the brand, but would like higher CPU power. The group is made out of young people, mostly male.

Next comes segment 2, which is again made out of young persons, mostly males. They prefer to trust specific brands and their price range is significantly higher than the 1st group. They also don't demand high CPU power.

Group 3 – Made out of slightly older people(mostly male). This is the smallest group, but as said earlier, one of the most important ones, because these people are ready to pay big prices in order to buy themselves laptop of specific brand, while the CPU power almost doesn't matter.

Group 4 – The largest group, they are grouped by price. We can see, that the interquartile range and the median are almost identical, with few outliers. They basically buy the average products on the market, without caring so much for the brand.

Group 5 – Again grouped by price preferences like group 4, but slightly better. They tend to require slightly better performance and care about the brand of their laptop slightly more than the members of group 4.

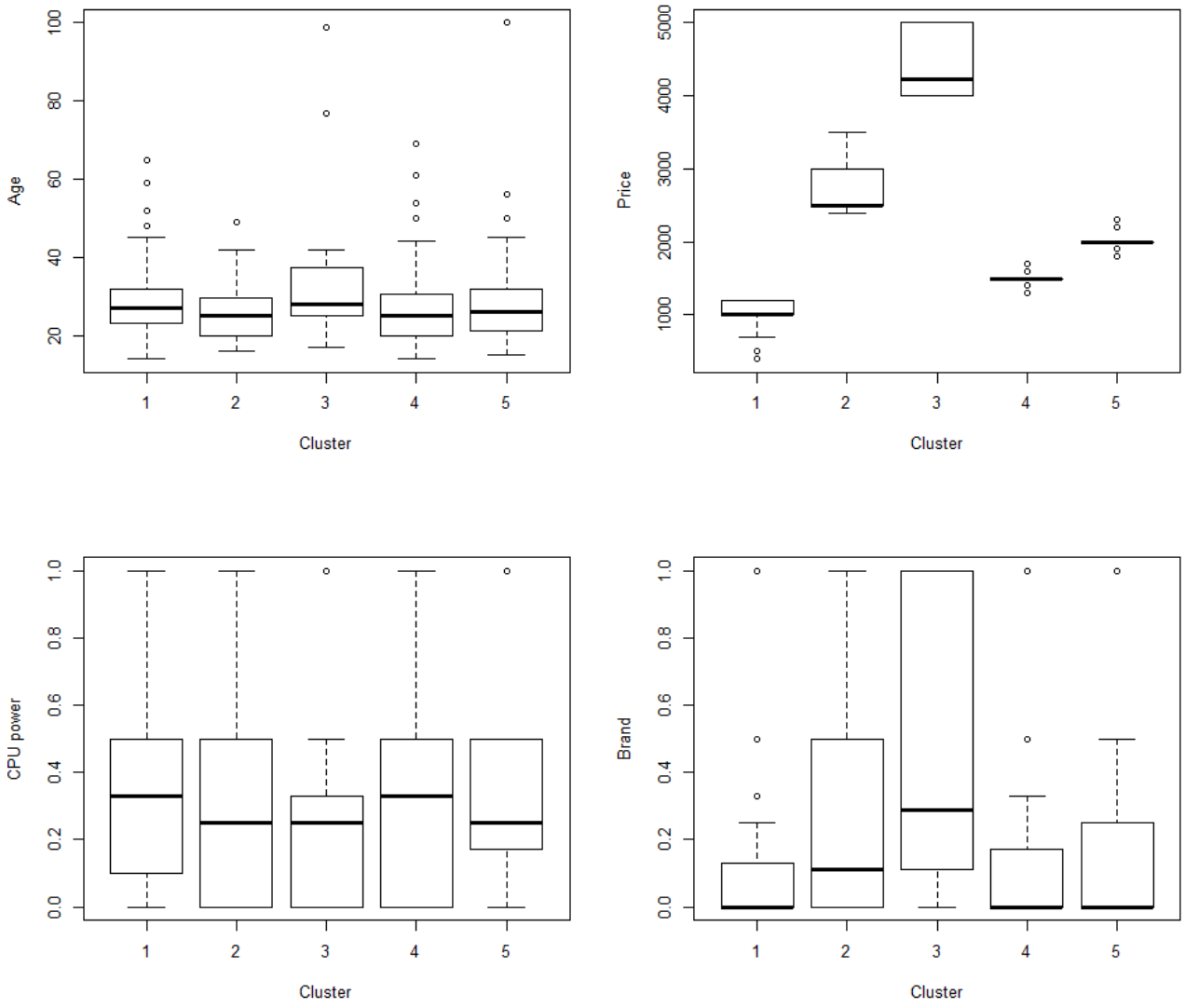



**Fig. 2.7 – Boxplot of Different Variables in the Clusters**

Four P's:

I've decided to propose a marketing strategy for segment #3. I would offer them an expensive, lightweight laptop. It will be from a trusted brand. The other characteristics seem to be almost irrelevant for this group. The target group will be made out of people older than 35 years. The product won't need much advertisement, because the people have their preferences and know what they want from their laptop and are ready to pay for it. Maybe I would occasionally offer promotions, that would include free maintenance of the laptop for 12/24 months, which will increase the sales during the promotion period.

*Note: Just before submitting, I saw, that only the attributes should be used as base values, so I will look at this case here:*

I removed the other variables and created a new dataframe for the clustering, with only the attribute variables in it.

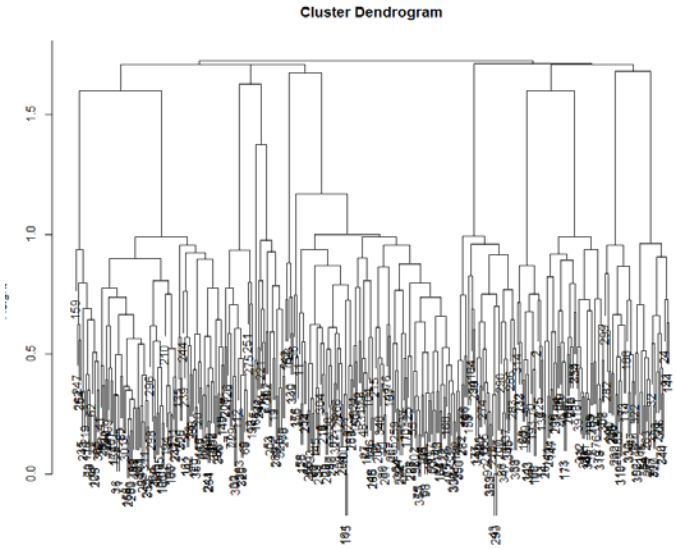



**Fig. 3.1 – Dendrogram of attributes**

Again, this plot is unreadable and we have to decide on the number of clusters. Therefore, I go directly to check the height and decide on number of clusters.

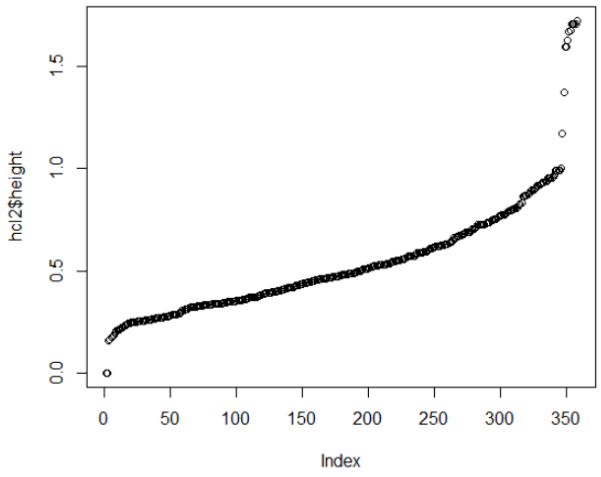



**Fig. 3.2 – Plot of height of attributes' distances**

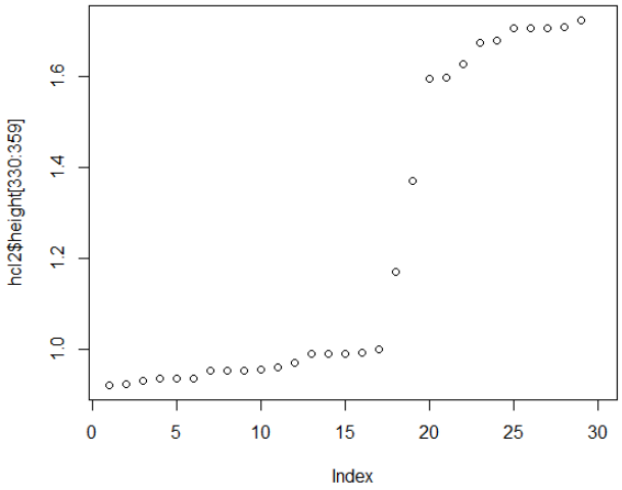



**Fig.3.3 – 340-359 of Fig.3.2**

By counting the number of possible clusters, we can see that the most appropriate one will be 12. I suppose, that this is because the algorithm has grouped the data into 12 clusters, with each cluster representing the people who have answered with highest importance of each of the attributes. This is for me a rather obvious clustering and grouping. Let's now check the output of the seg.summ() function and see, if my guesses were right.

```
> seg.summ(segmdata, seg.hc.segment.new)
```

```
   Group.1    A_brand A_CPUbrand A_CPUpower     A_RAM A_harddisk A_displaysize A_displayres   A_CD_DVD  A_network   A_weight
1        1 0.08235294 0.07882353  0.2847059 1.0000000  0.2447059    0.17352941   0.09235294 0.12470588 0.08294118 0.09529412
2        2 0.09687500 0.08437500  0.1900000 0.1150000  0.0850000    0.16750000   0.15687500 0.05812500 0.05937500 1.00000000
3        3 0.09200000 0.10693333  1.0000000 0.3278667  0.2274667    0.13213333   0.11013333 0.12080000 0.08093333 0.08306667
4        4 0.07783784 0.07405405  0.1762162 0.2170270  0.1497297    0.17540541   0.15594595 0.07378378 0.11324324 0.18918919
5        5 0.06611111 0.04111111  0.2322222 0.3705556  1.0000000    0.08944444   0.08333333 0.08722222 0.10166667 0.11277778
6        6 0.13102041 0.06846939  0.2366327 0.2106122  0.1531633    0.15857143   0.10346939 0.09785714 0.08438776 0.11459184
7        7 0.00000000 0.00000000  0.0000000 0.5000000  0.2350000    0.00000000   0.00000000 1.00000000 0.06500000 0.00000000
8        8 0.08000000 0.04190476  0.2328571 0.1985714  0.1657143    1.00000000   0.27666667 0.10714286 0.06857143 0.08095238
9        9 0.08266667 1.00000000  0.3800000 0.2613333  0.1453333    0.14133333   0.10200000 0.08400000 0.10133333 0.05000000
10      10 1.00000000 0.20054054  0.1908108 0.2170270  0.1318919    0.11837838   0.07891892 0.07729730 0.07243243 0.07675676
11      11 0.09055556 0.08333333  0.1655556 0.1861111  0.1433333    0.34222222   1.00000000 0.07277778 0.10000000 0.15944444
12      12 0.13400000 0.10000000  0.0160000 0.2120000  0.1880000    0.09600000   0.15000000 0.06600000 1.00000000 0.04000000
   A_battery   A_price
1  0.1400000 0.1888235
2  0.3793750 0.1256250
3  0.1932000 0.1881333
4  0.9864865 0.1697297
5  0.1905556 0.1788889
6  0.1746939 0.9798980
7  0.1250000 0.1650000
8  0.1814286 0.1680952
9  0.1026667 0.2393333
10 0.1659459 0.1662162
11 0.1355556 0.1522222
12 0.1840000 0.2520000
```

As we can see here, my guess was correct and each cluster is has mean of 1 for the different attributes. Let's see now the number of people, which are in the different segments.

```
> table(seg.hc.segment.new)
seg.hc.segment.new
 1  2  3  4  5  6  7  8  9 10 11 12
17 16 75 37 18 98  2 21 15 37 18  5
```

Looks like the biggest part of the people, who answered the survey care most about the price of the laptop and the least number of people care about whether their laptops have DVD slot or not.

I continue with the kmeans algorithm and then check the error, which here is 0%:

```
> sum(kmeanssegm2$withinss)
[1] 0
```

Now let's check the age and price distribution for each cluster and see, if we can find some pattern.

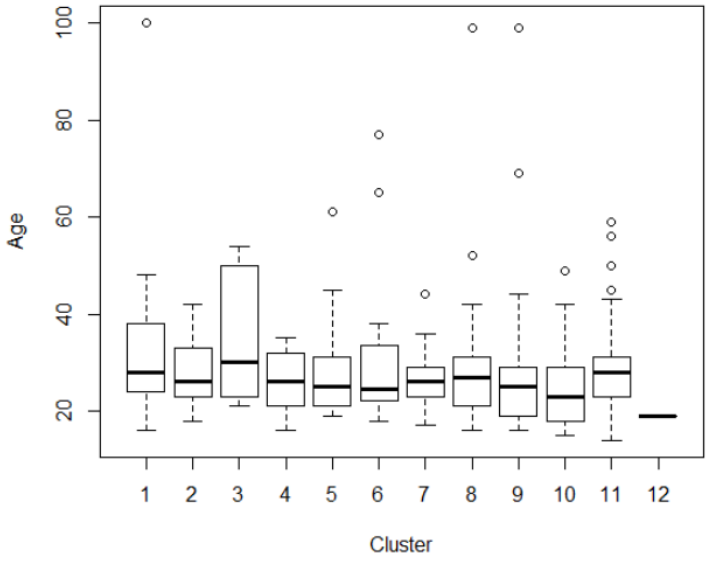



**Fig 3.4 –Age distribution for different clusters**

On this plot we can see, that the older people care about the CPU power of their laptop (cluster 3). All the people who care about the networking of their laptops are around 20 years old. The other clusters have no patterns in their age distribution.

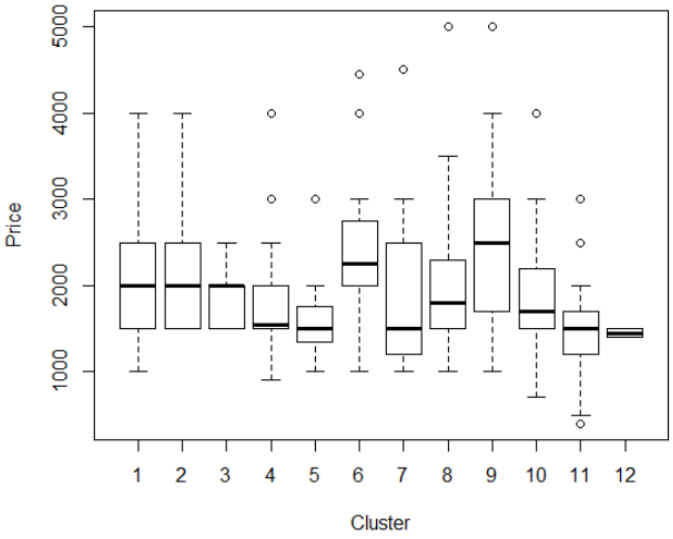



**Fig 3.5 –Price distribution for different clusters**

Here, we can see that, as expected people who paid the least for their laptops, answered that the price is the most important attribute for them. Also people who care about the brand of their CPU are willing to pay higher prices to buy a laptop.

Based on the research conducted, I would try to sell laptops to people who can be part of cluster number 12, because I know what their expectations are, what price they are willing to pay and their age. The other clusters seem more random and it will be harder to define a strategy.