Product Design Segmentation via Conjoint Analysis

Nelson Shilman

10/8/2021

The complete project can be cloned from Github! at this link

Study Design

In this study I'll use conjoint analysis in order to expose the latent personal preferences regarding what we look after when buying an e-reader (Kindle branded).

In this fashion, I first assumed which where the main attributes most customers look after when deciding to buy an e-reader, which I guessed where the screen size, backlit screen, water resistance, internal memory and price.

After choosing those attributes, I identified possible levels for each attribute:

- Screen size: *6-inches, 8-inches, 10-inches*
- Backlit screen: *with, without*
- Water resistance: *with, without*
- Internal memory: *4GB, 8GB, 16GB, 32GB*
- Price: *80 USD, 120 USD, 160 USD, 200 USD*

Then I proceded to elaborate an orthogonal experimental design which is documented at the file *orthogonal_design.R* and can be consumed from the file *factorialDesign.csv*.

Survey

The file *orthogonal_design.R* also prints each alternative shown on the factorial design and its used to populate a survey made with Google Forms.

The survey is closed but an exact copy can be accessed at the following link.

Additionally, the 39 obtained responses can be downloaded from this link

Data Analysis

Regression Analysis

First of all I imported the data

Table 1: Dummies for each configuration and values given by first surveyed person

price_2	price_3	price_4	size_2	size_3	backlight_2	water_resistant_2	internal_memory_2	internal_memory_3	internal_memory_4	value
0	1	0	0	0	1	0	0	0	0	3
1	0	0	0	1	1	0	0	0	0	2
0	0	1	1	0	0	1	0	0	0	3
0	0	0	0	1	0	1	0	0	0	4
0	0	1	0	1	0	0	1	0	0	5
0	0	0	0	1	1	0	1	0	0	5
0	1	0	0	0	0	1	1	0	0	6
1	0	0	1	0	1	1	1	0	0	4
1	0	0	0	0	0	0	0	1	0	1
0	0	0	1	0	0	0	0	1	0	1
0	0	1	0	0	1	1	0	1	0	9
0	1	0	0	1	1	1	0	1	0	7
0	1	0	1	0	0	0	0	0	1	7
0	0	1	1	0	1	0	0	0	1	6
1	0	0	0	1	0	1	0	0	1	8
0	0	0	0	0	1	1	0	0	1	2

```
path <-
  'https://docs.google.com/spreadsheets/d/1xcZ50RHmDu-X2GYyOh7OUkJLhqBkNmJ9chydSBOcag4/export?gid=12229

survey_data <- read_csv(path) %>%
  select(-Timestamp)
```

Afterwards, I created dummy variables for each attribute, excluding the first level from each since they'll be represented as the base values from the regression.

```
dummified <- dummy_cols(code %>% mutate_all(as.factor),
                        remove_first_dummy = T,
                        remove_selected_columns = T)
```

Then, proceed to transform data, building a list of dataframes where each dataframe corresponds to a surveyed person and each row represents a product configuration and columns represent attributes and value given to that configuration.

```
scores_dfs <-  map(1:nrow(survey_data), ~dummified %>%
                     cbind(value = survey_data %>% slice(.x) %>%
                             t() %>%
                             as.vector()))
```

In order to make it clear, the first of those dataframes is shown at Table 1:

```
kbl(scores_dfs[[1]], booktabs = T,
    caption = " Dummies for each configuration and values given by first surveyed person") %>%
  kable_styling(latex_options = c("striped", "scale_down"))
```

Additionally, I binded all the dataframes by their rows, obtaining a general dataframe from the survey

```
scores_total <- map_df(scores_dfs, ~.x)
```

In order to apply particular regressions to each user, I developed a function that applies the statistical model required and gets the coefficients which are later coherced into a tabular structure

Table 2: partial values for each person

(Intercept)	price_2	price_3	price_4	size_2	size_3	backlight_2	water_resistant_2	internal_memory_2	internal_memory_3	internal_memory_4
-1.042986	0.75	3.3201357	3.2760181	0.1764706	2.2805430	0.3970588	1.6470588	2.00	2.0701357	3.2760181
9.711538	-0.75	0.7403846	-0.2596154	0.0000000	-0.0384615	-2.3750000	-1.3750000	-3.50	-2.5096154	-3.2596154
5.824661	-1.50	-2.0033937	-5.6504525	0.5882353	1.9864253	1.3235294	-0.1764706	-0.50	0.7466063	1.5995475
-1.992081	2.00	1.8291855	3.0938914	1.9411765	4.3167421	-0.6323529	1.8676471	1.50	0.5791855	0.8438914
1.510181	-0.25	-0.1481900	-1.4864253	2.3529412	1.4072398	-1.2058824	-1.4558824	0.50	4.3518100	3.2635747
5.369910	-3.50	-4.5509050	-4.7567873	0.8235294	1.7963801	-1.6470588	0.6029412	-0.75	0.4490950	0.4932127
3.833710	-1.25	-0.4128959	-2.7217195	0.2352941	2.3484163	-1.4705882	-0.4705882	-0.25	-1.1628959	2.0282805
5.472851	-0.75	0.4785068	-1.0361991	-0.9411765	-1.0859729	-0.6176471	-0.1176471	1.25	1.9785068	3.4638009
5.106335	-0.25	-0.5616516	-3.1498869	0.3529412	0.7533937	-2.5808824	-0.5808824	0.50	1.4383484	2.1001131
2.986425	1.50	0.6142534	-0.7680995	1.5294118	1.4570136	-0.3088235	0.4411765	0.75	0.3642534	2.2319005
6.075792	0.75	0.8829186	-0.9406109	0.2941176	1.5316742	-1.7132353	-0.9632353	-0.75	0.3829186	1.0593891
5.493213	-2.00	-2.0678733	-1.5090498	0.7647059	1.7285068	-3.7794118	-0.5294118	-0.25	0.6821267	-0.0090498
6.510181	0.00	1.1018100	0.5135747	0.3529412	0.4072398	-0.2058824	-0.2058824	-0.50	0.3518100	-0.2364253
6.617647	-1.50	-1.3235294	-1.6764706	0.4117647	-1.2941176	-1.3235294	-1.0735294	1.75	2.4264706	1.3235294
6.489819	-3.25	-5.4768100	-6.3885747	0.6470588	0.0927602	-0.6691176	0.8308824	0.50	0.7731900	0.6114253
10.151584	1.75	-1.2341629	-0.6312217	-1.4117647	-2.9366516	-0.6764706	-0.9264706	-2.00	-1.4841629	-0.6312217
7.814480	-0.25	-0.8552036	-1.6640271	-0.7647059	-1.4208145	-5.7205882	-0.7205882	0.75	1.6447964	1.8359729
6.290724	-3.75	-4.0927602	-4.6957014	-0.5882353	-0.3710407	1.1764706	0.1764706	1.25	-0.3427602	1.5542986
5.874434	-2.00	-0.6306561	-2.2924208	1.6470588	1.4773756	-0.4191176	0.3308824	1.00	0.3693439	-0.2924208
4.884615	-2.50	-2.4038462	-2.9038462	1.0000000	2.3846154	-0.5000000	-0.7500000	0.25	0.3461538	2.5961538
5.585973	1.75	0.8597285	0.1979638	-1.3529412	-0.5610860	-6.0441176	-1.2941176	2.50	2.6097285	2.6979638
10.214932	-6.25	-5.9756787	-4.5050905	-2.8823529	-2.9027149	0.3897059	-0.1102941	1.00	1.5243213	0.7449095
8.180995	0.00	-1.8150452	-2.3003394	-0.0588235	-0.2601810	-0.7573529	-1.2573529	-0.25	-0.8150452	-0.0503394
5.196833	0.75	1.4683258	0.0124434	-0.1764706	0.8733032	-1.8970588	-0.8970588	-0.25	-0.5316742	1.0124434
1.815611	-1.00	-2.5938914	-4.3291855	2.9411765	4.6244344	0.8676471	0.8676471	0.00	1.1561086	0.9208145
5.109729	-0.50	-3.0277149	-3.8953620	0.4705882	0.8891403	-0.5661765	-1.0661765	1.25	0.2222851	2.1046380
4.980769	1.75	1.0576923	1.3076923	0.0000000	-0.7692308	-1.0000000	-0.2500000	0.50	-0.4423077	-0.9423077
8.833710	-2.25	-2.5378959	-3.5967195	0.2352941	-1.1515837	-2.5955882	-1.3455882	-1.25	-0.0378959	2.9032805
6.933258	-0.50	-2.4174208	-4.0056561	-0.6470588	0.3303167	-1.9558824	0.0441176	0.50	1.3325792	2.2443439
6.176471	-0.25	-0.4852941	-2.7647059	2.1176471	1.0588235	0.2647059	0.0147059	-1.00	0.7647059	1.7352941
-1.992081	2.00	1.8291855	3.0938914	1.9411765	4.3167421	-0.6323529	1.8676471	1.50	0.5791855	0.8438914
6.621041	-1.25	-1.7895928	-1.9219457	1.5294118	1.8416290	-1.0588235	-0.3088235	0.50	0.7104072	0.8280543
9.338235	-2.00	-4.3676471	-5.6323529	0.0588235	-0.4705882	-0.3676471	-2.3676471	-1.00	-0.1176471	0.1176471
5.557692	-1.00	-1.5480769	-2.2980769	1.0000000	0.8076923	-1.3750000	-0.1250000	1.00	0.9519231	1.2019231
3.675339	0.50	-0.3716063	-0.4745475	1.4117647	1.5135747	-0.1985294	-0.6985294	-0.25	2.3783937	2.5254525
4.762443	-0.50	-3.0005656	-2.1917421	0.7647059	2.9977376	-0.6544118	-1.9044118	1.00	2.2494344	3.0582579
5.377828	1.25	-1.5967195	-2.0378959	-0.2352941	-0.3868778	-2.4044118	-2.1544118	4.25	2.1532805	4.2121041
6.696833	-1.75	-1.6566742	-1.8625566	2.8235294	1.3733032	0.4779412	-0.5220588	-1.00	-2.1566742	-0.1125566
4.825792	0.50	-0.7420814	-2.0656109	0.2941176	1.0316742	-0.5882353	-1.0882353	0.00	1.2579186	2.4343891

```
get_coeffs <- function(df){
  model <- lm(value~.,df)
  coeffs <- model$coefficients %>%
            t() %>%
             as.data.frame()

  return(coeffs)
}
```

Afterwards, I apply that function to each Dataframe, binding results by row, showing all partial values at Table 2

```
partial_values <- map_dfr(scores_dfs, get_coeffs)
kbl(partial_values, booktabs = T, caption = "partial values for each person") %>%
  kable_styling(latex_options = c("striped", "scale_down"))
```

Define relative importance for each person

First we build a function that obtains the range of coefficients and zero for each level of the attributes after applying multiple regression model

```
range_calculation <- function(coef_numbers){
  range <- abs(max(0,model$coefficients[coef_numbers]) - min(0,model$coefficients[coef_numbers]))
  return(range)
}
```

Now I initialize an empty Dataframe, defining a vector of attributes and a list with the positions of coefficients from each attribute

```
relative_importance_df <- data.frame()
attributes <- c('price', 'size', 'backlight', 'water_resistant','internal_memory')
coeff_positions <- list(2:4,5:6,7,8,9:11)
```

Finally I iterate over the dataframes applying the chosen statistical model, extracting coeffs and estimating their ranges, showing relative importance and populate the empty Dataframe. showing the relative importance for each attribute for each person at Table 3

```
for(i in 1:length(scores_dfs)){

  model <- lm(value~., scores_dfs[[i]])
  ranges <- map_dfc(coeff_positions, ~ range_calculation(.x)) %>% set_names(str_c(attributes,'_range'))
  ranges_sum <- sum(ranges)
  relative_importance <- map_dfc(ranges, ~.x/ranges_sum) %>%
    mutate(id = i) %>%
    select(id, everything())

  relative_importance_df <- rbind(relative_importance_df, relative_importance)

}

kbl(relative_importance_df, booktabs = T,
    caption = "Relative importance for each attribute for each person") %>%
kable_styling(latex_options = "striped")
```

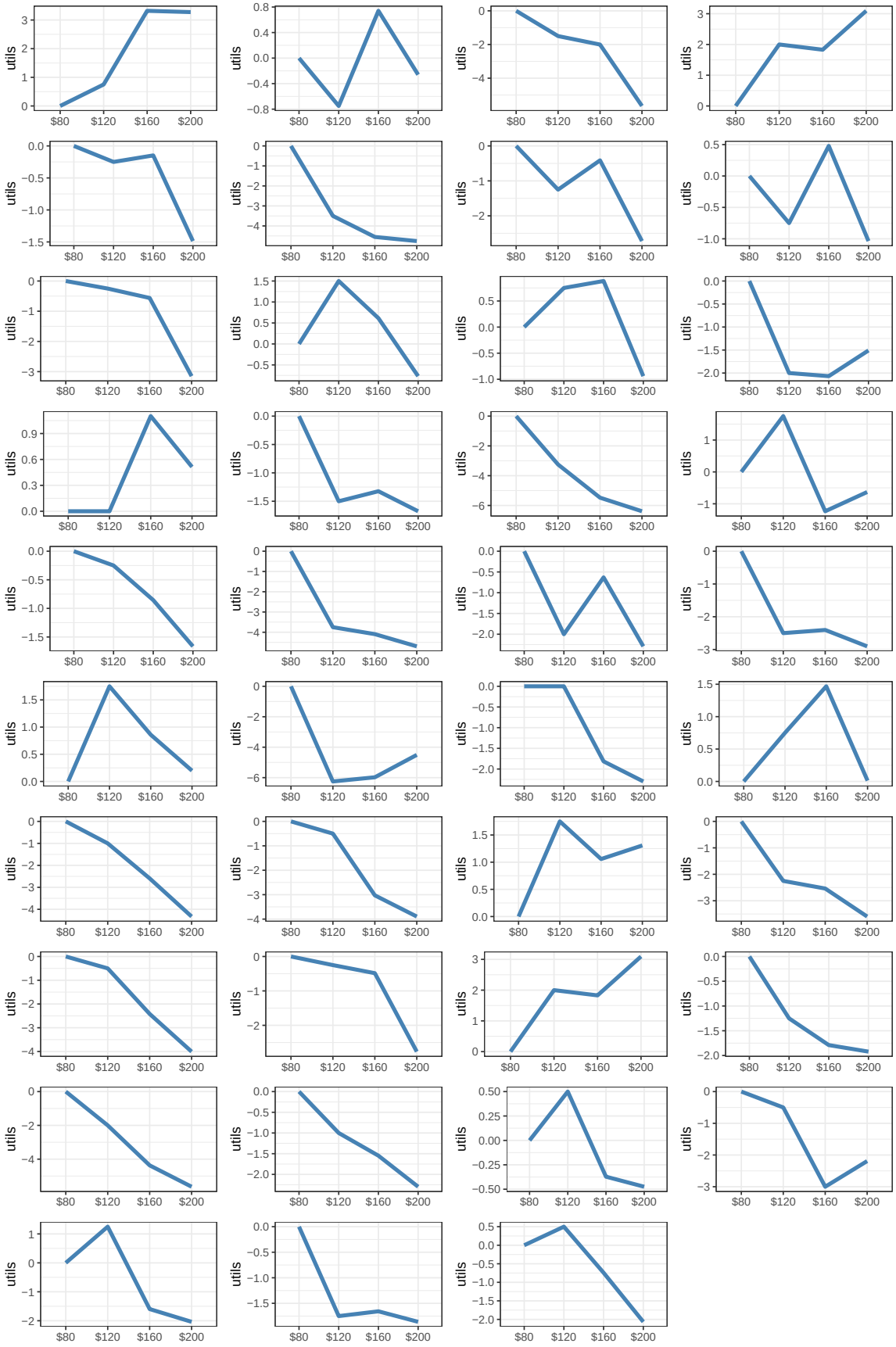
Partial value plots

```
plist <- list()
for(i in 1:length(scores_dfs)){

  model <- lm(value~., scores_dfs[[i]])
  plot_df <- data.frame(utils = c(0,model$coefficients[2:4]) %>% as.vector(),
                        label = c('$80', '$120', '$160', '$200'))

  plist[[i]] <- ggplot(plot_df)+
    geom_line(aes(x = factor(label, levels = c('$80', '$120', '$160', '$200')),
                  y = utils, group = 1),
              color = 'steelblue', size = 1.5)+
    theme_bw()+
    xlab(NULL)
```

Table 3: Relative importance for each attribute for each person

id	price_range	size_range	backlight_range	water_resistant_range	internal_memory_range
1	0.3040191	0.2088254	0.0363580	0.1508183	0.2999793
2	0.1697700	0.0043812	0.2705367	0.1566265	0.3986857
3	0.5028692	0.1767845	0.1177892	0.0157052	0.1868519
4	0.2711411	0.3783087	0.0554179	0.1636760	0.1314563
5	0.1369606	0.2168022	0.1111111	0.1341463	0.4009798
6	0.4734827	0.1788087	0.1639455	0.0600158	0.1237473
7	0.2667702	0.2301807	0.1441401	0.0461248	0.3127841
8	0.2227583	0.1597072	0.0908335	0.0173016	0.5093994
9	0.3436806	0.0822019	0.2815971	0.0633794	0.2291410
10	0.3345570	0.2255965	0.0455531	0.0650759	0.3292174
11	0.2325615	0.1953401	0.2184953	0.1228450	0.2307581
12	0.2288146	0.1912630	0.4182000	0.0585805	0.1031418
13	0.3973888	0.1468788	0.0742554	0.0742554	0.3072215
14	0.2043011	0.2078853	0.1612903	0.1308244	0.2956989
15	0.6862924	0.0695103	0.0718799	0.0892575	0.0830599
16	0.3133389	0.3083502	0.0710298	0.0972800	0.2100012
17	0.1464556	0.1250498	0.5034847	0.0634209	0.1615890
18	0.5502386	0.0689290	0.1378579	0.0206787	0.2222959
19	0.3832262	0.2753404	0.0700643	0.0553139	0.2160552
20	0.3178947	0.2610526	0.0547368	0.0821053	0.2842105
21	0.1331898	0.1029703	0.4600086	0.0984933	0.2053379
22	0.5591822	0.2597035	0.0348667	0.0098679	0.1363797
23	0.4267576	0.0482686	0.1405037	0.2332634	0.1512067
24	0.2141561	0.1531100	0.2766870	0.1308365	0.2252104
25	0.3654856	0.3904116	0.0732499	0.0732499	0.0976029
26	0.4571220	0.1043409	0.0664410	0.1251162	0.2469800
27	0.3357934	0.1476015	0.1918819	0.0479705	0.2767528
28	0.2750195	0.1060462	0.1984690	0.1028890	0.3175763
29	0.4341057	0.1059213	0.2119652	0.0047812	0.2432267
30	0.3500931	0.2681564	0.0335196	0.0018622	0.3463687
31	0.2711411	0.3783087	0.0554179	0.1636760	0.1314563
32	0.3225133	0.3090357	0.1776765	0.0518223	0.1389522
33	0.5624082	0.0528634	0.0367107	0.2364170	0.1116006
34	0.3830128	0.1666667	0.2291667	0.0208333	0.2003205
35	0.1581895	0.2456849	0.0322255	0.1133860	0.4505141
36	0.2583268	0.2580834	0.0563401	0.1639560	0.2632937
37	0.2633773	0.0309909	0.1926057	0.1725794	0.3404467
38	0.2374874	0.3600173	0.0609404	0.0665657	0.2749892
39	0.3328441	0.1338421	0.0763135	0.1411799	0.3158204

```
}
do.call("grid.arrange", c(plist, ncol=4))
```

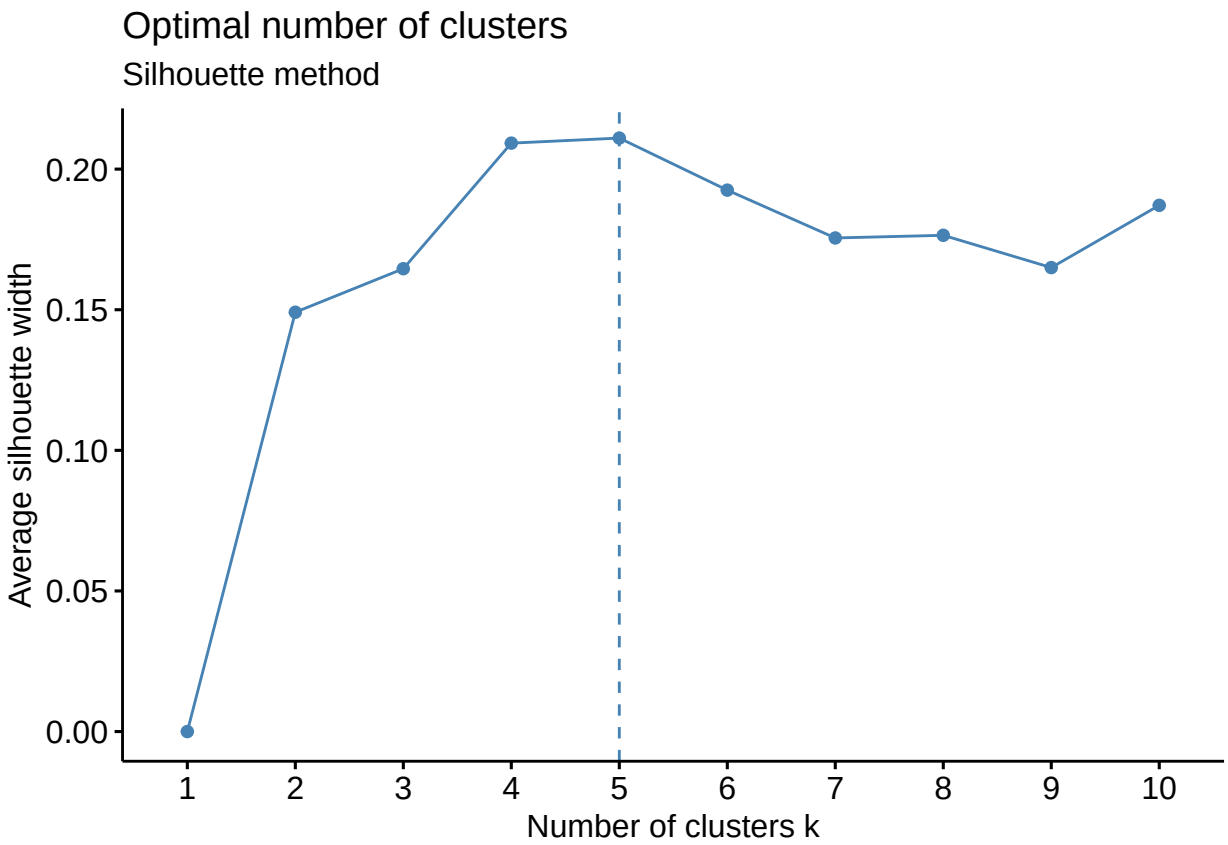

utils
$80
$120
$160
$200

Being the base price the lowest price, the most seen behavior is that after a surge in price, utility falls showing high sensitivity to price that must be offset by different levels from other attributes

Segmentation

using a k-means alogrithm optimized by the silhouette method, it is advisable to divide the surveyed people into 5 groups

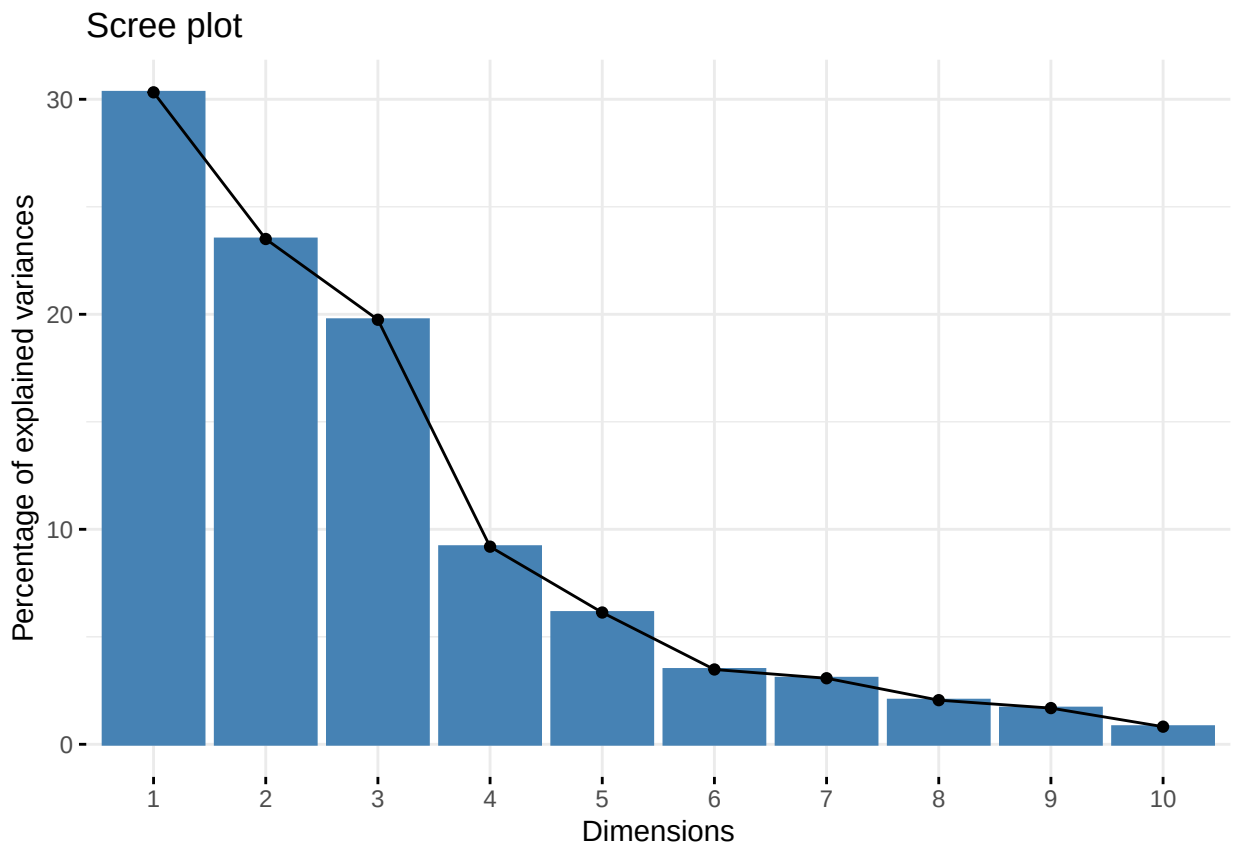
```
data_scaled <- partial_values %>%
  select(-1) %>%
  scale() %>%
  data.frame()

fviz_nbclust(data_scaled,nstart = 50, kmeans, method = "silhouette")+
  labs(subtitle = "Silhouette method")
```



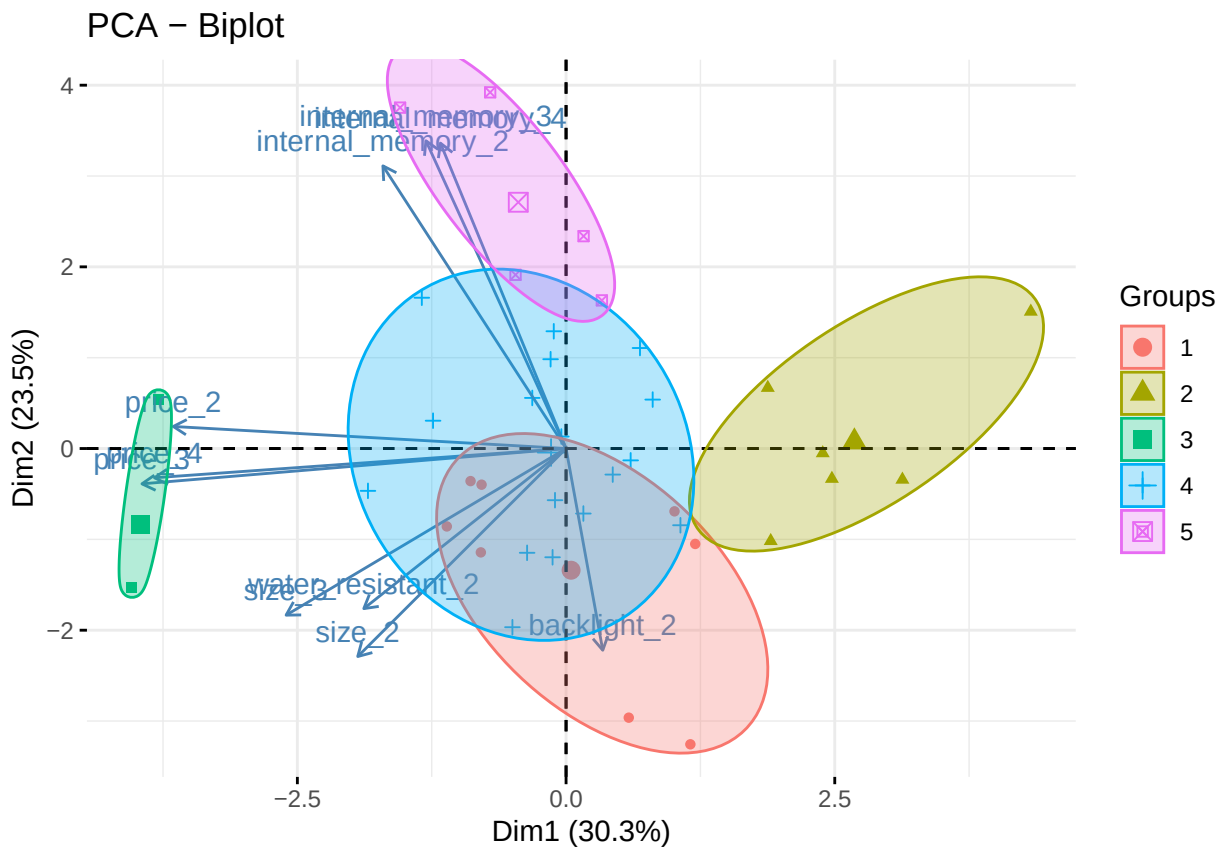
Now I run the k-means algorithm with the scaled values and reproject output data into 2 synthetic features built with PCA

```
kmeans_clustering <- kmeans(data_scaled,centers = 5,nstart = 50)
```

```
res.pca <- prcomp(data_scaled, scale = TRUE)
fviz_eig(res.pca)
```




```
fviz_pca_biplot(res.pca, label="var", habillage=kmeans_clustering$cluster)+
  ggforce::geom_mark_ellipse(aes(fill = Groups,
                                 color = Groups),expand = 1e-2)
```



Conclusion

From a visual standpoint, first group prioritize internal memory and back-lit panels, the second one has very high sensitivity to price so they demand a cheaper product no matter what. In contrast, group 3 has low price sensitivity so its advisable to offer them an uncompromised product. group 4 seems pretty equilibrated and lots of memory it is for group 5.