SulekhaAloorravi_SL_ASmt

Sulekha Aloorravi

Introduction

The methodology used in this assignment is conjoint analysis with ranking. This method helps in identifying the relative importance of an attribute compared to the other, preferences on a product based on Brand, price, color etc.

I chose to make use of Amazon.in to select a product. I just selected two random refrigerator models and imported their Technical details into an excel sheet. Technical details of Amazon's Product Information gave me the data required to decide attributes or factors and their corresponding levels

A. Create a full factorial design for a product with 5 attributes and 2 levels each. Level for each attribute can be labelled 1 and 2.

Answer: Full factorial design: As the name suggests, Full factorial design would consider all permutations and combinations of designing profiles. For each attribute, each level is combined with each other level of the other attributes. There will be 2^5 = 32 (2x2x2x2x2) profiles in this design.

```
library(conjoint)
```

```
##
## This is package 'modeest' written by P. PONCET.
## For a complete list of functions, use 'library(help = "modeest")' or 'help.start()'.
```

```
library(rAmCharts)
```

```
## Important changes: constructors legend() and title() have been replaced by amLegend() a
nd amTitle().
```

```
## For any bug report or feed back see https://github.com/datastorm-open/rAmCharts
```

```
library(optimbase)
```

```
## Loading required package: Matrix
```

```
library(car)
```

Define attributes and levels

```
#Whirlpool -- "WP"
#15290 -- "15k"
#23900 -- "23k"
# Single Door and Scarlet Color -- "1D"
# Double Door and Steel -- "2D"
# Direct Cool -- "DC"
# Frost Free -- "FF"
Brand <- c("LG", "WP")
Capacity_Litres <- c("190","240")
Price <- c("15k","23k")
Appearance <- c("1D","2D")
DefrostSystem <- c("DC", "FF")
```

Expand profile and convert all columns into factors

```
FullFactorial<- expand.grid(Brand,Capacity_Litres,Price,Appearance,DefrostSystem)
```

Create a full factorial design

```
FullFactorialDesign <- caFactorialDesign(FullFactorial,type="full")
colnames(FullFactorialDesign) <- c("Brand", "Capacity in Liters", "Price", "Appearance",
"Defrost System")
FullFactorialDesign$Price <- as.factor(FullFactorialDesign$Price)
caEncodedDesign(FullFactorialDesign) -> EncodedFullProfile
EncodedFullProfile
```

```
##    Brand Capacity.in.Liters Price Appearance Defrost.System
## 1      1                  1     1          1               1
## 2      2                  1     1          1               1
## 3      1                  2     1          1               1
## 4      2                  2     1          1               1
## 5      1                  1     2          1               1
## 6      2                  1     2          1               1
## 7      1                  2     2          1               1
## 8      2                  2     2          1               1
## 9      1                  1     1          2               1
## 10     2                  1     1          2               1
## 11     1                  2     1          2               1
## 12     2                  2     1          2               1
## 13     1                  1     2          2               1
## 14     2                  1     2          2               1
## 15     1                  2     2          2               1
## 16     2                  2     2          2               1
## 17     1                  1     1          1               2
## 18     2                  1     1          1               2
## 19     1                  2     1          1               2
## 20     2                  2     1          1               2
## 21     1                  1     2          1               2
## 22     2                  1     2          1               2
## 23     1                  2     2          1               2
## 24     2                  2     2          1               2
## 25     1                  1     1          2               2
## 26     2                  1     1          2               2
## 27     1                  2     1          2               2
## 28     2                  2     1          2               2
## 29     1                  1     2          2               2
## 30     2                  1     2          2               2
## 31     1                  2     2          2               2
## 32     2                  2     2          2               2
```

Calculate number of profiles

```
NumberOfProfiles = nrow(FullFactorialDesign)
print(c("Number Of Profiles = ",NumberOfProfiles))
```

```
## [1] "Number Of Profiles = " "32"
```

View correlation of attributes in Full Profile

```
cor(EncodedFullProfile)
```

```
##                   Brand Capacity.in.Liters Price Appearance
## Brand                 1                  0     0          0
## Capacity.in.Liters    0                  1     0          0
## Price                 0                  0     1          0
## Appearance            0                  0     0          1
## Defrost.System        0                  0     0          0
##                   Defrost.System
## Brand                          0
## Capacity.in.Liters             0
## Price                          0
## Appearance                     0
## Defrost.System                 1
```

Create consumers and simulate data by generating random rankings for each profile

```
Ravi <- sample(1:32,32)
Dileep <- sample(1:32,32)
Sasi <- sample(1:32,32)
Sailu <- sample(1:32,32)
Praba <- sample(1:32,32)
Sudha <- sample(1:32,32)
```

Combine all consumer ranking records into a single data frame

```
ConsumerRanks <- rbind(Ravi,Dileep,Sasi,Sailu,Praba,Sudha)
```

Combine levels of all attributes into a single data frame

```
Levels <- c(Brand,Capacity_Litres,Price,Appearance,DefrostSystem)
```

Estimate parameters of conjoint analysis model for one consumer

```
caModel(Ravi,EncodedFullProfile)
```

```
##
## Call:
## lm(formula = frml)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -16.5625  -5.0000  -0.5625   6.3750  19.5625
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    16.5000     1.5015  10.989 2.87e-11 ***
## factor(x$Brand)1               -0.5625     1.5015  -0.375  0.71097
## factor(x$Capacity.in.Liters)1  -0.8750     1.5015  -0.583  0.56507
## factor(x$Price)1               -0.6250     1.5015  -0.416  0.68063
## factor(x$Appearance)1          -2.5625     1.5015  -1.707  0.09980 .
## factor(x$Defrost.System)1       4.3125     1.5015   2.872  0.00801 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.493 on 26 degrees of freedom
## Multiple R-squared:  0.3125, Adjusted R-squared:  0.1802
## F-statistic: 2.363 on 5 and 26 DF,  p-value: 0.06786
```

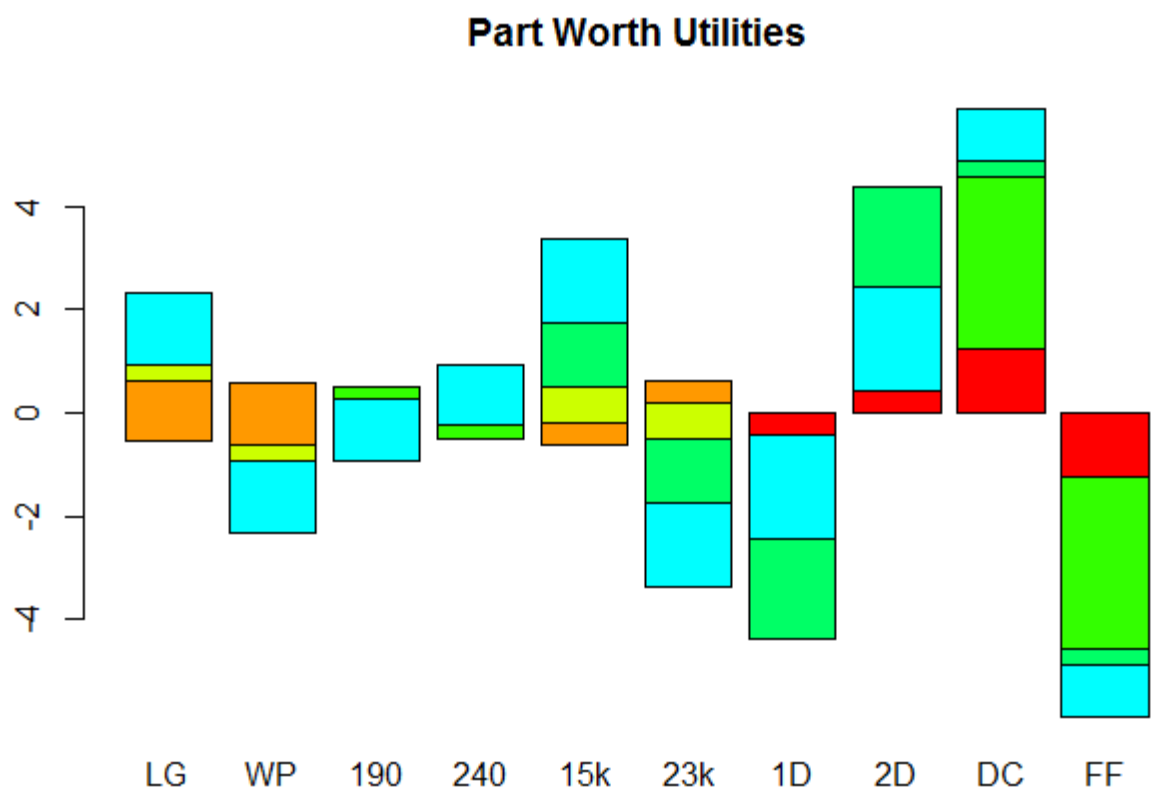
Calculate individual utilities for each consumer at each level of attribute

```
cap<-caPartUtilities(ConsumerRanks,EncodedFullProfile,Levels)
cap
```

```
##      intercept     LG     WP    190    240    15k    23k     1D     2D
## [1,]      16.5 -0.563  0.563 -0.875  0.875 -0.625  0.625 -2.563  2.563
## [2,]      16.5  1.187 -1.187  0.000  0.000  0.437 -0.437  0.688 -0.688
## [3,]      16.5  1.625 -1.625  1.375 -1.375  0.687 -0.687  0.062 -0.062
## [4,]      16.5 -0.813  0.813 -0.500  0.500  0.000  0.000 -2.563  2.563
## [5,]      16.5 -0.500  0.500 -0.938  0.938  2.875 -2.875  1.937 -1.937
## [6,]      16.5  1.375 -1.375  1.187 -1.187 -1.625  1.625  2.000 -2.000
##          DC     FF
## [1,]  4.313 -4.313
## [2,] -1.875  1.875
## [3,] -1.188  1.188
## [4,]  3.312 -3.312
## [5,]  0.313 -0.313
## [6,]  1.000 -1.000
```

Plot Part Worth Utilities

```
barplot(cap[,2:ncol(cap)],col = rainbow(10),main = "Part Worth Utilities")
```



Customers are grouped into clusters using K-means clustering and rated against each profile

```
caSegmentation(ConsumerRanks,EncodedFullProfile)
```

```
## K-means clustering with 3 clusters of sizes 2, 3, 1
##
## Cluster means:
##       [,1]     [,2]   [,3]   [,4]   [,5]     [,6]   [,7]     [,8]   [,9]
## 1 16.06200 17.43700 17.437 18.812 16.687 18.06200 18.062 19.43700 21.187
## 2 18.81233 16.02033 17.104 14.312 19.146 16.35433 17.437 14.64567 16.979
## 3 20.18700 21.18700 22.062 23.062 14.437 15.43700 16.312 17.31200 16.312
##      [,10]    [,11]  [,12]    [,13]    [,14]  [,15]  [,16]    [,17]
## 1 22.56200 22.56200 23.937 21.81200 23.18700 23.187 24.562  8.43700
## 2 14.18733 15.27067 12.479 17.31267 14.52067 15.604 12.812 20.18767
## 3 17.31200 18.18700 19.187 10.56200 11.56200 12.438 13.438 19.56200
##      [,18]    [,19]    [,20]    [,21]    [,22]  [,23]  [,24]    [,25]
## 1  9.81200  9.81200 11.18700  9.06200 10.43700 10.437 11.812 13.56200
## 2 17.39567 18.47933 15.68733 20.52067 17.72933 18.812 16.021 18.35367
## 3 20.56200 21.43700 22.43700 13.81200 14.81200 15.687 16.687 15.68700
##      [,26]    [,27]    [,28]   [,29]  [,30]    [,31]    [,32]
## 1 14.93700 14.93700 16.31200 14.1875 15.562 15.56200 16.93700
## 2 15.56233 16.64533 13.85433 18.6880 15.896 16.97933 14.18733
## 3 16.68700 17.56200 18.56200  9.9380 10.938 11.81200 12.81200
##
## Clustering vector:
## [1] 1 2 2 1 3 2
##
## Within cluster sum of squares by cluster:
## [1]  25.5005 348.6218   0.0000
##  (between_SS / total_SS =  84.2 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

Calculate Average Part worth utilities of each attribute levels

```
worth <- caUtilities(ConsumerRanks,EncodedFullProfile,Levels)
```

```
##
## Call:
## lm(formula = frml)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -17,0417  -7,5990   0,1771   7,6458  17,2708
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   16,50000    0,67222  24,545   <2e-16 ***
## factor(x$Brand)1               0,38542    0,67222   0,573    0,567
## factor(x$Capacity.in.Liters)1  0,04167    0,67222   0,062    0,951
## factor(x$Price)1               0,29167    0,67222   0,434    0,665
## factor(x$Appearance)1         -0,07292    0,67222  -0,108    0,914
## factor(x$Defrost.System)1      0,97917    0,67222   1,457    0,147
## ---
## Signif. codes:  0 '***' 0,001 '**' 0,01 '*' 0,05 '.' 0,1 ' ' 1
##
## Residual standard error: 9,315 on 186 degrees of freedom
## Multiple R-squared:  0,01407,    Adjusted R-squared:  -0,01243
## F-statistic: 0,5309 on 5 and 186 DF,  p-value: 0,7527
```

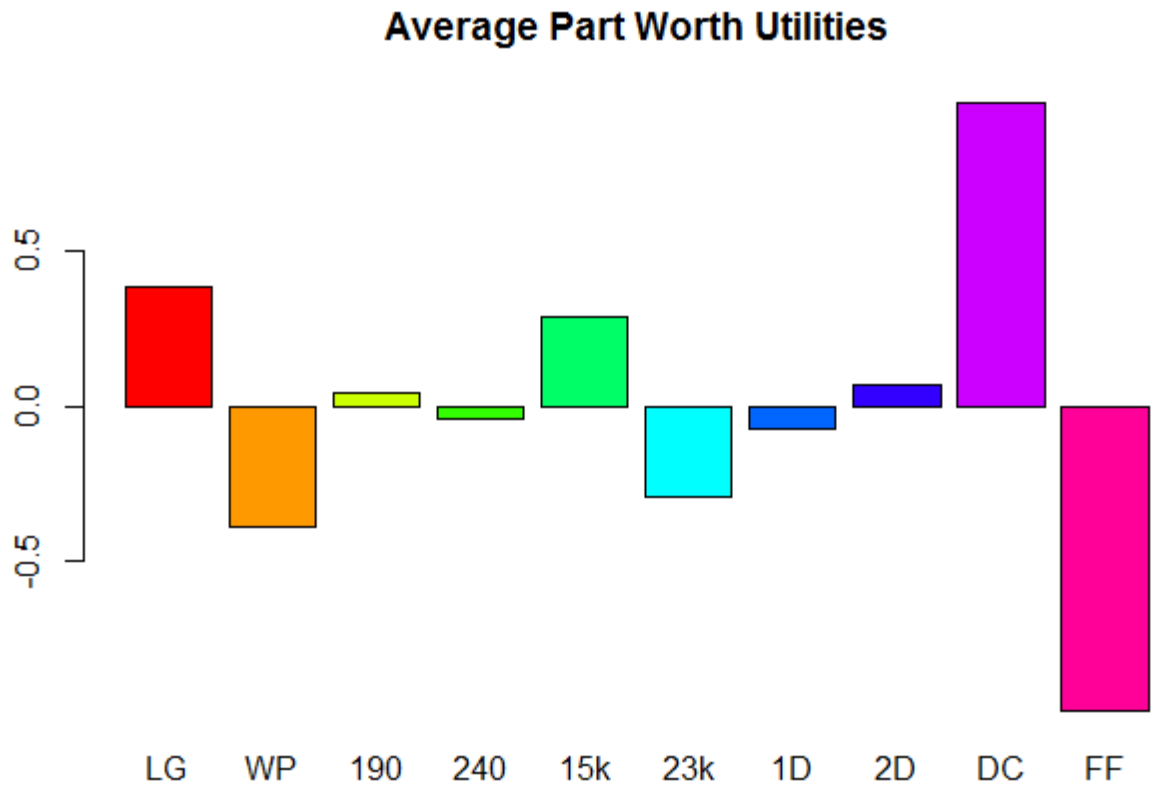
```
worth
```

```
##  [1] 16.50000000  0.38541667 -0.38541667  0.04166667 -0.04166667
##  [6]  0.29166667 -0.29166667 -0.07291667  0.07291667  0.97916667
## [11] -0.97916667
```

```
worth <- worth [2:length(worth)]
names(worth) <- c("LG", "WP","190", "240", "15k", "23k", "1D", "2D", "DC", "FF" )
```

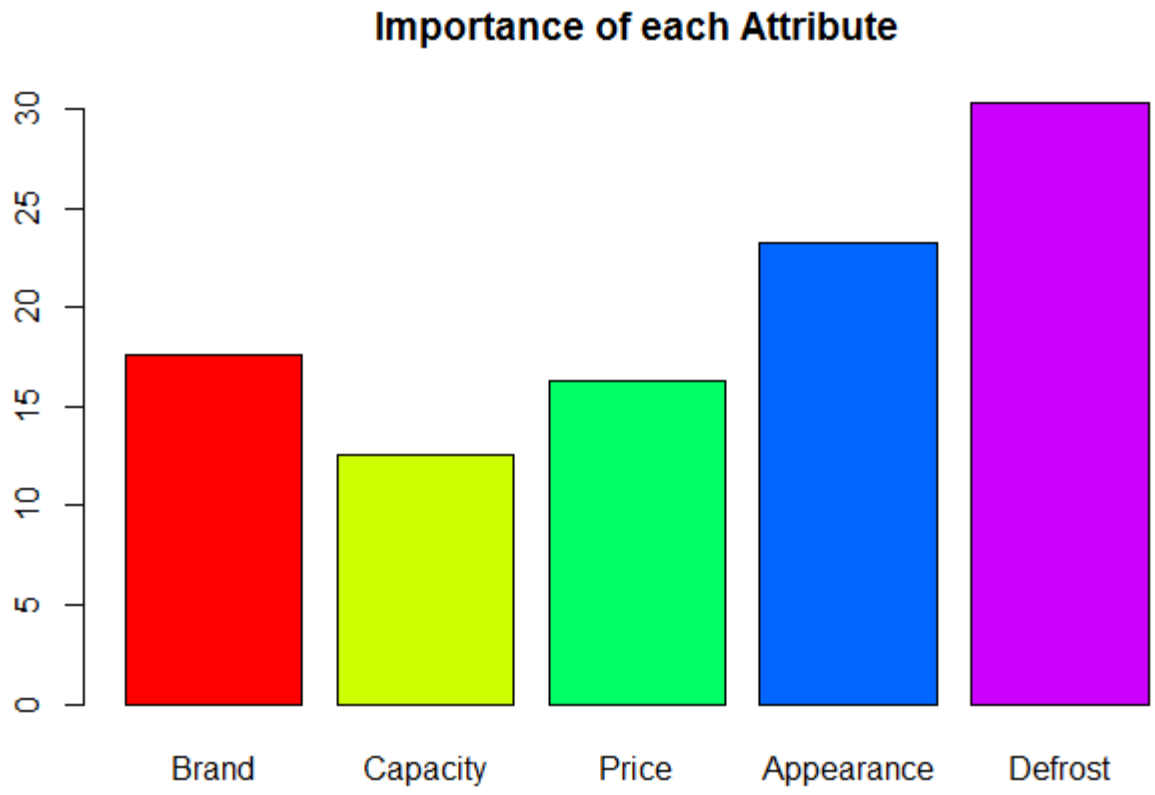
Plot Average Part Worth Utilities

```
barplot(worth,col = rainbow(10),main = "Average Part Worth Utilities")
```



Calculate percentage of importance of each attribute and plot it.

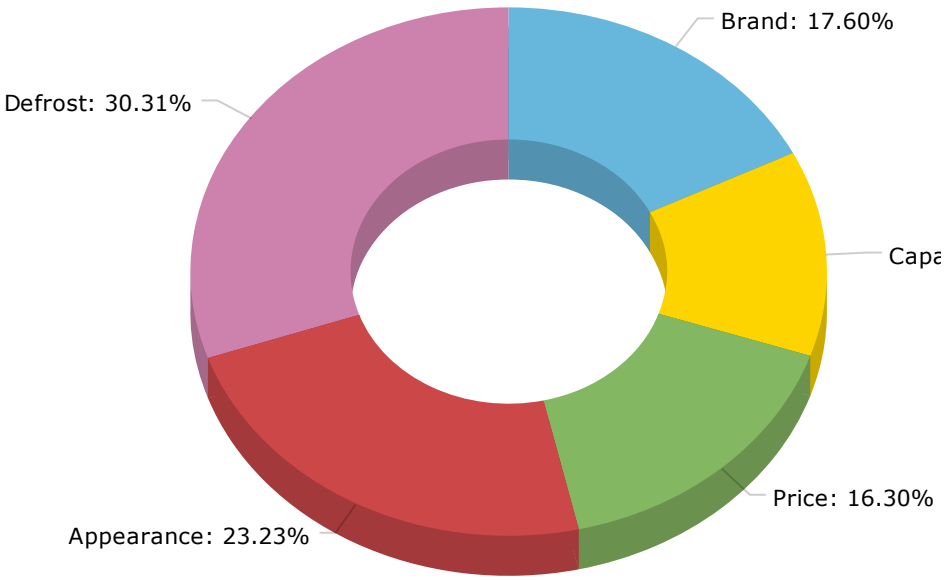
```
imp<-caImportance(ConsumerRanks,EncodedFullProfile)
names(imp) <- c("Brand", "Capacity", "Price", "Appearance", "Defrost")
barplot(imp, col = rainbow(5), main = "Importance of each Attribute")
```



Create a pie chart to visualize Importance of each attribute

```
#Convert imp into a data frame to draw a pie chart
imp<-transpose(imp)
label <- c("Brand", "Capacity", "Price", "Appearance", "Defrost")
imp <- cbind(label,imp)
imp <- as.data.frame(imp)
colnames(imp) <- c("label","value")
imp$label <- as.character(imp$label)
imp$value <- as.character(imp$value)
imp$value <- as.numeric(imp$value)
```

```
amPie(imp,inner_radius = 50, depth = 20, col = rainbow(5))
```



B.Create an orthogonal design, how many profiles are there?

Answer: Orthogonal Design: An experimental design is orthogonal if the effects of any factor balance out (sum to zero) across the effects of the other factors.

Orthogonal design will have 1/4th of the total profiles.

No. of profiles in Orthogonal = 8

Create orthogonal design

```
OrthogonalProfiles <- caFactorialDesign(EncodedFullProfile,type="orthogonal")
OrthogonalProfiles
```

```
##    Brand Capacity.in.Liters Price Appearance Defrost.System
## 2      2                  1     1          1              1
## 3      1                  2     1          1              1
## 14     2                  1     2          2              1
## 15     1                  2     2          2              1
## 21     1                  1     2          1              2
## 24     2                  2     2          1              2
## 25     1                  1     1          2              2
## 28     2                  2     1          2              2
```

View correlation of attributes in Orthogonal profile

```
cor(OrthogonalProfiles)
```

```
##                   Brand Capacity.in.Liters Price Appearance
## Brand                 1                  0     0          0
## Capacity.in.Liters    0                  1     0          0
## Price                 0                  0     1          0
## Appearance            0                  0     0          1
## Defrost.System        0                  0     0          0
##                   Defrost.System
## Brand                          0
## Capacity.in.Liters             0
## Price                          0
## Appearance                     0
## Defrost.System                 1
```

Calculate number of profiles

```
NumberOfProfiles = nrow(OrthogonalProfiles)
print(c("Number Of Profiles = ",NumberOfProfiles))
```

```
## [1] "Number Of Profiles = " "8"
```

Create consumers and simulate data by generating random rankings for each profile

```
Ravi <- sample(1:8,8)
Dileep <- sample(1:8,8)
Sasi <- sample(1:8,8)
Sailu <- sample(1:8,8)
Praba <- sample(1:8,8)
Sudha <- sample(1:8,8)
```

Combine all consumer ranking records into a single data frame

```
ConsumerRanks <- rbind(Ravi,Dileep,Sasi,Sailu,Praba,Sudha)
```

Combine levels of all attributes into a single data frame

```
Levels <- c(Brand,Capacity_Litres,Price,Appearance,DefrostSystem)
```

Estimate parameters of conjoint analysis model for one consumer

```
caModel(Ravi,OrthogonalProfiles)
```

```
##
## Call:
## lm(formula = frml)
##
## Residuals:
##          1          2          3          4          5          6
## -5.551e-16  2.991e-16  4.409e-16 -3.486e-16  2.000e+00 -2.000e+00
##          7          8
## -2.000e+00  2.000e+00
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                       4.50       1.00    4.50    0.046 *
## factor(x$Brand)1                 -0.25       1.00   -0.25    0.826
## factor(x$Capacity.in.Liters)1     0.25       1.00    0.25    0.826
## factor(x$Price)1                 -1.25       1.00   -1.25    0.338
## factor(x$Appearance)1            -0.75       1.00   -0.75    0.531
## factor(x$Defrost.System)1        -1.00       1.00   -1.00    0.423
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.828 on 2 degrees of freedom
## Multiple R-squared:  0.619,  Adjusted R-squared:  -0.3333
## F-statistic:  0.65 on 5 and 2 DF,  p-value: 0.6985
```

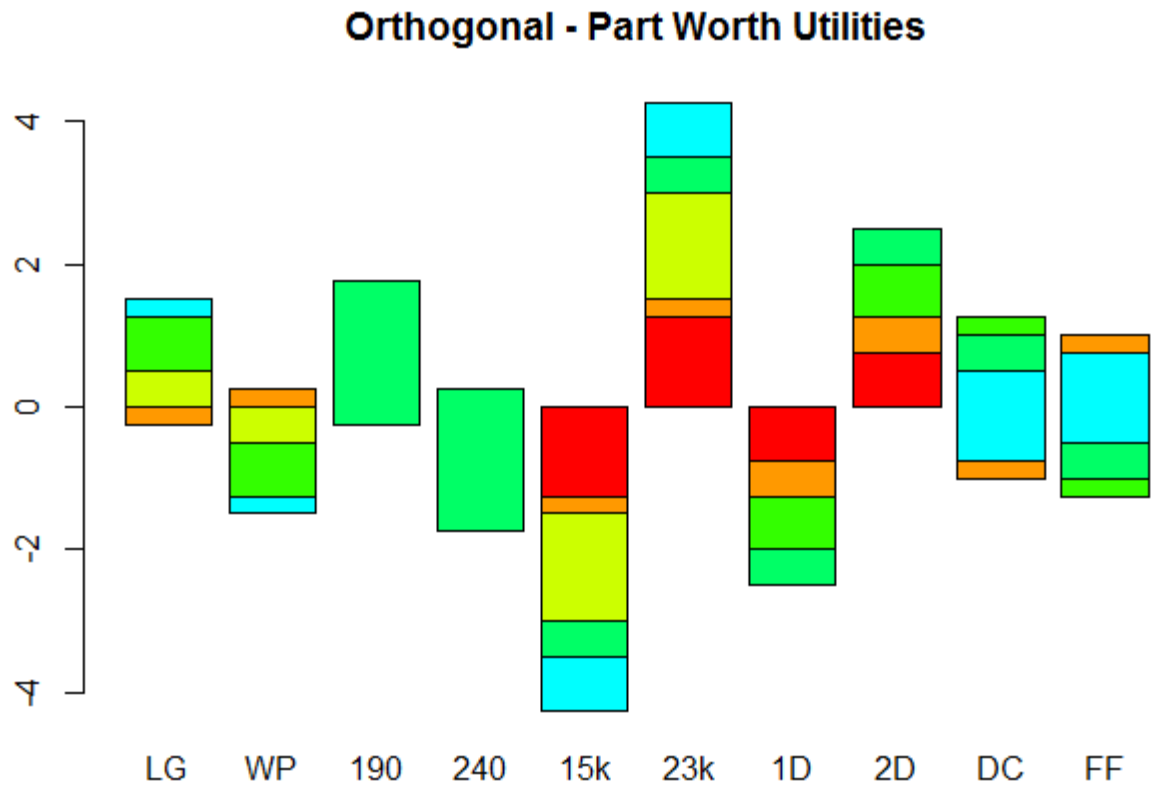
Calculate individual utilities for each consumer at each level of attribute

```
cap2<-caPartUtilities(ConsumerRanks,OrthogonalProfiles,Levels)
cap2
```

```
##      intercept    LG    WP   190   240   15k  23k    1D    2D    DC    FF
## [1,]       4.5 -0.25  0.25  0.25 -0.25 -1.25 1.25 -0.75  0.75 -1.00  1.00
## [2,]       4.5  0.25 -0.25  1.00 -1.00 -0.25 0.25 -1.00  1.00  1.75 -1.75
## [3,]       4.5  0.50 -0.50  0.50 -0.50 -1.50 1.50  0.50 -0.50  0.50 -0.50
## [4,]       4.5  0.75 -0.75  0.00  0.00  0.00 0.00 -1.25  1.25 -0.25  0.25
## [5,]       4.5  0.00  0.00 -2.00  2.00 -0.50 0.50  0.50 -0.50 -0.50  0.50
## [6,]       4.5  0.25 -0.25  0.00  0.00 -0.75 0.75  0.00  0.00 -1.25  1.25
```

Plot Part Worth Utilities

```
barplot(cap2[,2:ncol(cap2)],col = rainbow(10),main = "Orthogonal - Part Worth Utilities")
```



Customers are grouped into clusters using K-means clustering and rated against each profile

```
caSegmentation(ConsumerRanks,OrthogonalProfiles)
```

```
## K-means clustering with 3 clusters of sizes 3, 2, 1
##
## Cluster means:
##       [,1]  [,2]     [,3]     [,4]     [,5]  [,6]     [,7]  [,8]
## 1 2.166667 2.500 4.833333 5.166667 5.666667 5.000 5.666667 5.000
## 2 4.875000 4.125 7.125000 6.375000 5.125000 2.875 3.875000 1.625
## 3 2.000000 6.000 2.000000 6.000000 4.000000 8.000 2.000000 6.000
##
## Clustering vector:
## [1] 1 2 2 1 3 1
##
## Within cluster sum of squares by cluster:
## [1] 21.33333 22.75000  0.00000
##  (between_SS / total_SS =  66.9 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

Calculate Average Part worth utilities of each attribute levels

```
worth <- caUtilities(ConsumerRanks,OrthogonalProfiles,Levels)
```

```
##
## Call:
## lm(formula = frml)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -4,125 -1,646  0,250  1,833  3,208
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    4,50000    0,32940  13,661   <2e-16 ***
## factor(x$Brand)1               0,25000    0,32940   0,759   0,4521
## factor(x$Capacity.in.Liters)1 -0,04167    0,32940  -0,126   0,8999
## factor(x$Price)1              -0,70833    0,32940  -2,150   0,0373 *
## factor(x$Appearance)1         -0,33333    0,32940  -1,012   0,3174
## factor(x$Defrost.System)1     -0,12500    0,32940  -0,379   0,7062
## ---
## Signif. codes:  0 '***' 0,001 '**' 0,01 '*' 0,05 '.' 0,1 ' ' 1
##
## Residual standard error: 2,282 on 42 degrees of freedom
## Multiple R-squared:  0,1319, Adjusted R-squared:  0,0286
## F-statistic: 1,277 on 5 and 42 DF,  p-value: 0,2919
```

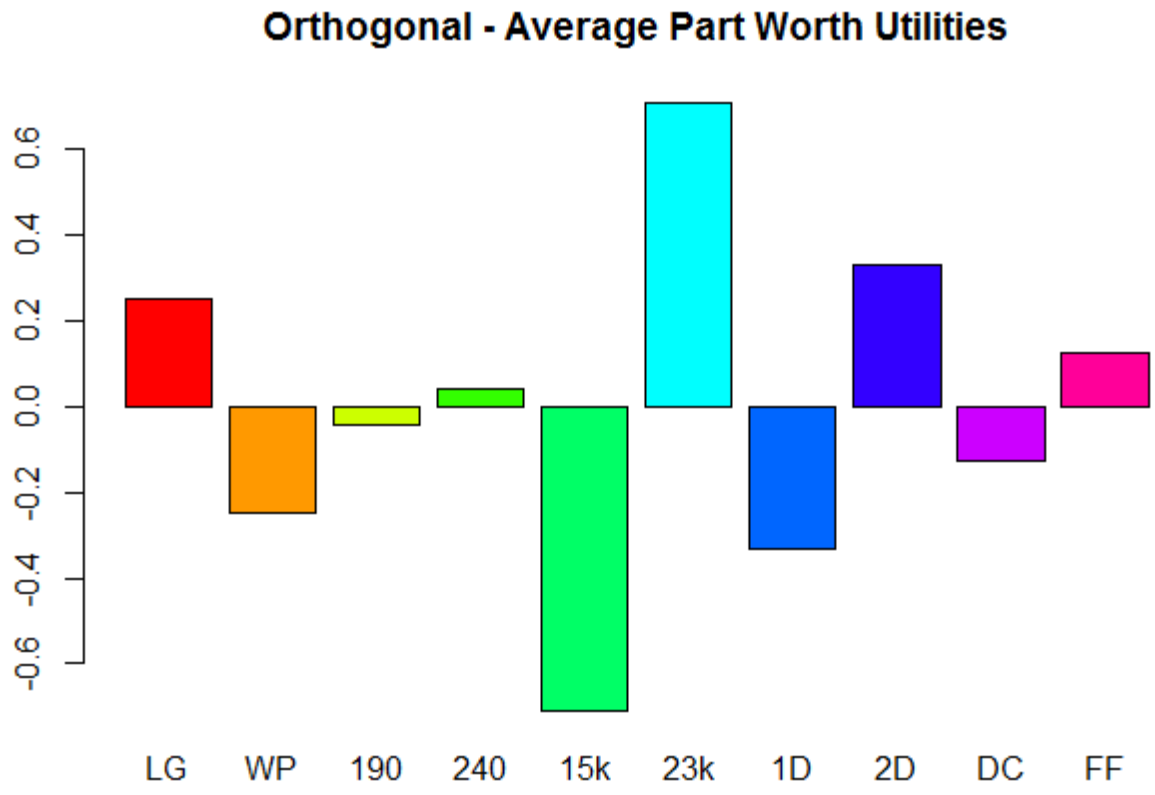
```
worth
```

```
##  [1]  4.50000000  0.25000000 -0.25000000 -0.04166667  0.04166667
##  [6] -0.70833333  0.70833333 -0.33333333  0.33333333 -0.12500000
## [11]  0.12500000
```

```
worth <- worth [2:length(worth)]
names(worth) <- c("LG", "WP","190", "240", "15k", "23k", "1D", "2D", "DC", "FF" )
```

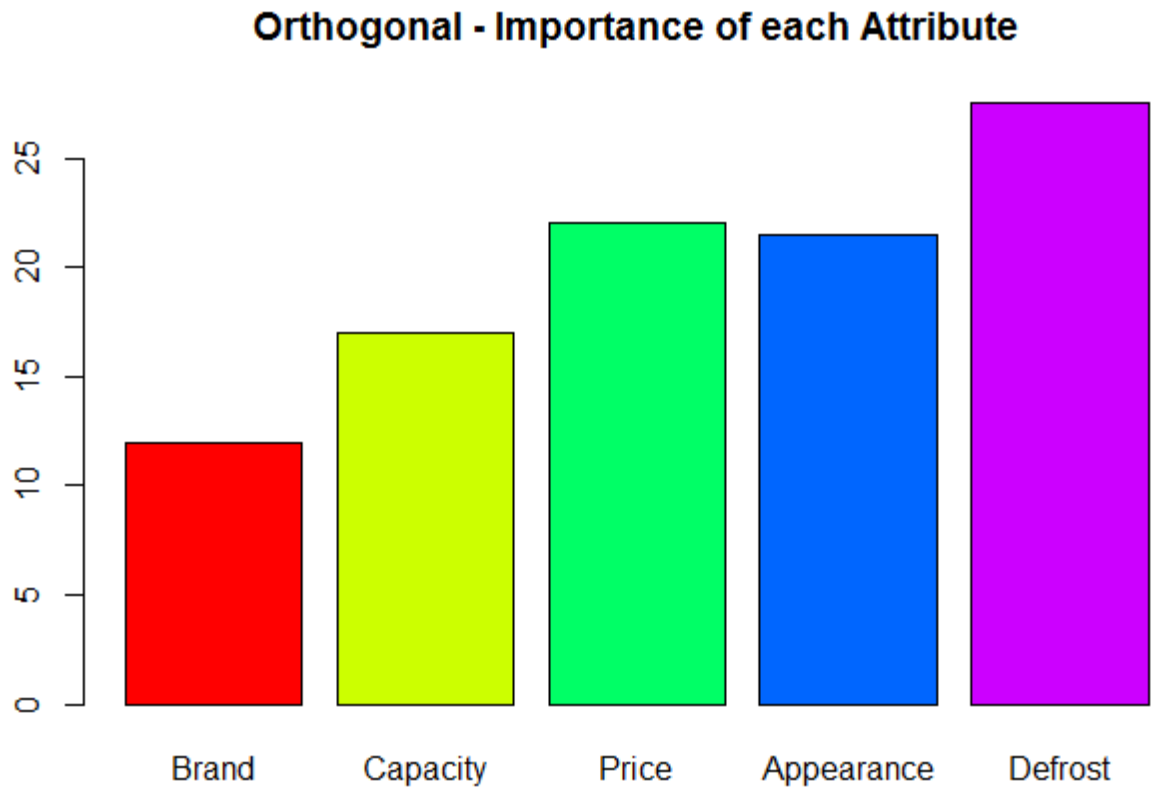
Plot Average Part Worth Utilities

```
barplot(worth,col = rainbow(10),main = "Orthogonal - Average Part Worth Utilities")
```



Calculate percentage of importance of each attribute and plot it.

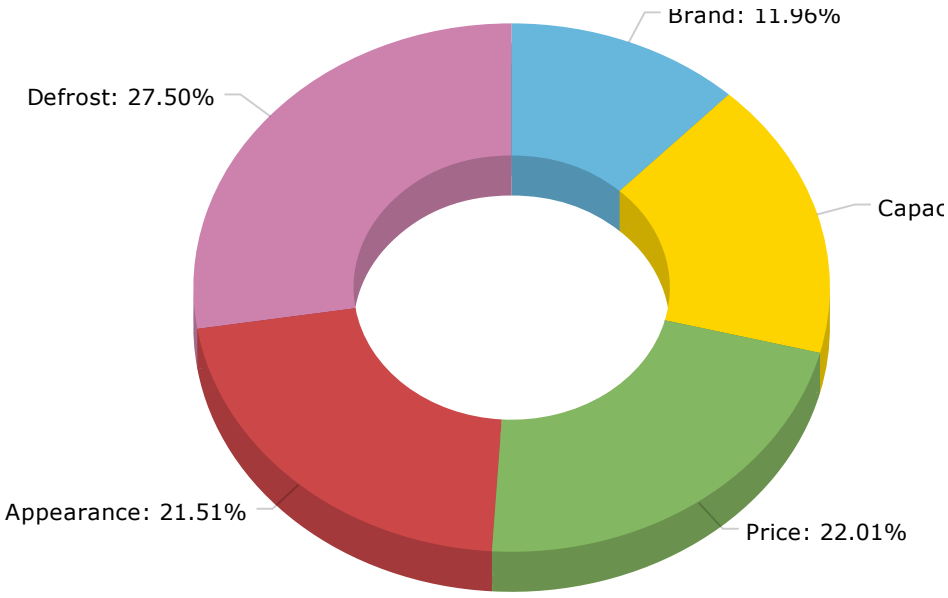
```
imp<-caImportance(ConsumerRanks,OrthogonalProfiles)
names(imp) <- c("Brand", "Capacity", "Price", "Appearance", "Defrost")
barplot(imp, col = rainbow(5), main = "Orthogonal - Importance of each Attribute")
```



Create a pie chart to visualize Importance of each attribute - Orthogonal Profile

```
#Convert imp into a data frame to draw a pie chart
imp<-transpose(imp)
label <- c("Brand", "Capacity", "Price", "Appearance", "Defrost")
imp <- cbind(label,imp)
imp <- as.data.frame(imp)
colnames(imp) <- c("label","value")
imp$label <- as.character(imp$label)
imp$value <- as.character(imp$value)
imp$value <- as.numeric(imp$value)
```

```
amPie(imp,inner_radius = 50, depth = 20, col = rainbow(5))
```



C. consumers' willingness to pay for color given the following output of a conjoint analysis. Price has three levels and color has two levels. Price levels are 50, 100, and 150. Color levels are Black and White.

The coefficients estimated by caModel function are given below. Price 50 7.8
Price 100 0.8 Colour Black 1.4

Answer: Willingness to Pay

In conjoin analysis, sum total of all co-efficients for an attribute should be equal to 0.

In this case, there are 3 levels for Price - 50, 100, 150. Co-efficients for: Price 50 = 7.8 Price 100 = 0.8

```
PriceData = c(50,100,150)
Price50 = 7.8
Price100 = 0.8
```

Calculating Coefficient of Price 150

```
Price150 = 0 - (Price100+Price50)
Price150
```

```
## [1] -8.6
```

Calculating Range of Coefficient

```
Range.CoeffPrice = Price50-Price150
Range.CoeffPrice
```

```
## [1] 16.4
```

Calculating Range of Price

```
Range.Price = max(PriceData) - min(PriceData)
```

Calculating Unit Satisfaction

```
UnitSatisfaction = Range.Price/Range.CoeffPrice
UnitSatisfaction
```

```
## [1] 6.097561
```

Calculating Willingness to pay for color

```
ColorBlack = 1.4
```

Color white as per conjoint analysis sum total of coefficients will be

```
ColorWhite = 0-ColorBlack
ColorWhite
```

```
## [1] -1.4
```

Range of coefficient of color

```
Range.CoeffColor = ColorBlack - ColorWhite
```

Willingess to Pay for Color

```
WIP.Color = UnitSatisfaction * Range.CoeffColor
WIP.Color
```

```
## [1] 17.07317
```