Toy Hourse Case

This is a .Rmd file that documents the specific use of the R file of software for this case.

Read datasets first

```
load("~/Desktop/GBA424 - Toy Horse Case Data (1).RData")
```

Part 1: A Priori Segmentation (lm output)

Ratingij = β0i+ β1iX1ij + β2iX2ij + … + βMiXMij + eij Here, We run regression to get β estimates called part-utility. - Aggregate (using all the individuals for one set of coefficient)

```
summary(lm(ratings~price+size+motion+style,data = conjointData))
```

```
##
## Call:
## lm(formula = ratings ~ price + size + motion + style, data = conjointData)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.0380 -1.8983  0.0433  2.0433  7.5739
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   7.7366     0.1339  57.759  < 2e-16 ***
## price         2.0584     0.1255  16.400  < 2e-16 ***
## size          1.7736     0.1202  14.759  < 2e-16 ***
## motion        0.3014     0.1202   2.508   0.0122 *
## style        -0.6119     0.1202  -5.092 3.82e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.899 on 2395 degrees of freedom
##   (800 observations deleted due to missingness)
## Multiple R-squared:  0.2145, Adjusted R-squared:  0.2132
## F-statistic: 163.5 on 4 and 2395 DF,  p-value: < 2.2e-16
```

- Segment (a priori segments using interations or cell means)

```
data <- merge(conjointData,respondentData,all = TRUE) # merge conjointData and responden
tData by ID
summary(lm(ratings~(price+size+motion+style)*age, data = data))
```

```
##
## Call:
## lm(formula = ratings ~ (price + size + motion + style) * age,
##     data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.6898  -1.2999   0.1558   1.3102   9.6093
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   9.8418     0.1764  55.808  < 2e-16 ***
## price         2.3211     0.1652  14.046  < 2e-16 ***
## size         -0.1266     0.1582  -0.800   0.4238
## motion        1.8480     0.1582  11.681  < 2e-16 ***
## style        -0.7227     0.1582  -4.568 5.17e-06 ***
## age          -3.3152     0.2213 -14.980  < 2e-16 ***
## price:age    -0.4136     0.2074  -1.994   0.0462 *
## size:age      2.9924     0.1985  15.072  < 2e-16 ***
## motion:age   -2.4356     0.1985 -12.267  < 2e-16 ***
## style:age     0.1745     0.1985   0.879   0.3795
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.306 on 2390 degrees of freedom
##   (800 observations deleted due to missingness)
## Multiple R-squared:  0.5041, Adjusted R-squared:  0.5022
## F-statistic: 269.9 on 9 and 2390 DF,  p-value: < 2.2e-16
```

```
summary(lm(ratings~(price+size+motion+style)*gender, data = data))
```

```
##
## Call:
## lm(formula = ratings ~ (price + size + motion + style) * gender,
##     data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.4544 -1.9007  0.0089  2.0089  7.2030
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    7.98472    0.18438  43.306  < 2e-16 ***
## price          2.03690    0.17277  11.790  < 2e-16 ***
## size           1.53671    0.16542   9.290  < 2e-16 ***
## motion         0.46964    0.16542   2.839  0.00456 **
## style         -0.65734    0.16542  -3.974 7.28e-05 ***
## gender        -0.52233    0.26753  -1.952  0.05100 .
## price:gender   0.04533    0.25068   0.181  0.85651
## size:gender    0.49871    0.24001   2.078  0.03783 *
## motion:gender -0.35418    0.24001  -1.476  0.14016
## style:gender   0.09561    0.24001   0.398  0.69041
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.891 on 2390 degrees of freedom
##   (800 observations deleted due to missingness)
## Multiple R-squared:  0.2203, Adjusted R-squared:  0.2174
## F-statistic: 75.03 on 9 and 2390 DF,  p-value: < 2.2e-16
```

Because gender is not significant, so here we just use age segment.

```
summary(lm(ratings~price+size+motion+style,subset=age==1,data = data)) # subset 3-4 years old customer
```

```
##
## Call:
## lm(formula = ratings ~ price + size + motion + style, data = data,
##     subset = age == 1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.9390 -1.7123  0.0217  1.3342  9.6093
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   6.5266     0.1370  47.654  < 2e-16 ***
## price         1.9075     0.1283  14.863  < 2e-16 ***
## size          2.8658     0.1229  23.324  < 2e-16 ***
## motion       -0.5876     0.1229  -4.782 1.90e-06 ***
## style        -0.5482     0.1229  -4.462 8.73e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.362 on 1519 degrees of freedom
##   (508 observations deleted due to missingness)
## Multiple R-squared:  0.4019, Adjusted R-squared:  0.4003
## F-statistic: 255.1 on 4 and 1519 DF,  p-value: < 2.2e-16
```

```
summary(lm(ratings~price+size+motion+style,subset=age==0, data = data)) # subset 2 years old customer
```

```
##
## Call:
## lm(formula = ratings ~ price + size + motion + style, data = data,
##     subset = age == 0)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.6898  -0.8408   0.3102   1.3102   4.1595
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   9.8418     0.1686  58.373  < 2e-16 ***
## price         2.3211     0.1580  14.692  < 2e-16 ***
## size         -0.1266     0.1513  -0.837    0.403
## motion        1.8480     0.1513  12.218  < 2e-16 ***
## style        -0.7227     0.1513  -4.778 2.08e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.204 on 871 degrees of freedom
##   (292 observations deleted due to missingness)
## Multiple R-squared:  0.2985, Adjusted R-squared:  0.2953
## F-statistic: 92.66 on 4 and 871 DF,  p-value: < 2.2e-16
```

From the priori segments, we can know that for 2-years-old children, they prefer P6 (1010), and 3 to 4-years-old children, they prefer P4 (1100).

- Aggregate by Individual

```
b = cbind(rep(1,nrow(conjointData)),conjointData[,c(4:7)])
partworths = matrix(nrow=nrow(respondentData),ncol=ncol(b))
for(i in 1:200){ # for each individual run the regression
  partworths[i,]=lm(ratings~price+size+motion+style,subset=ID==i,data = conjointData)$co
ef
}
colnames(partworths) = c("Intercept","price","size","motion","style")
# Coefficient for each individual
```

Part 2: Post-hoc Segmentation (cluster output)

```
toClust = partworths
source("ClusterCode.R")
```

```
## Loading required package: cluster
```

```
## Loading required package: fpc
```

```
## Loading required package: factoextra
```

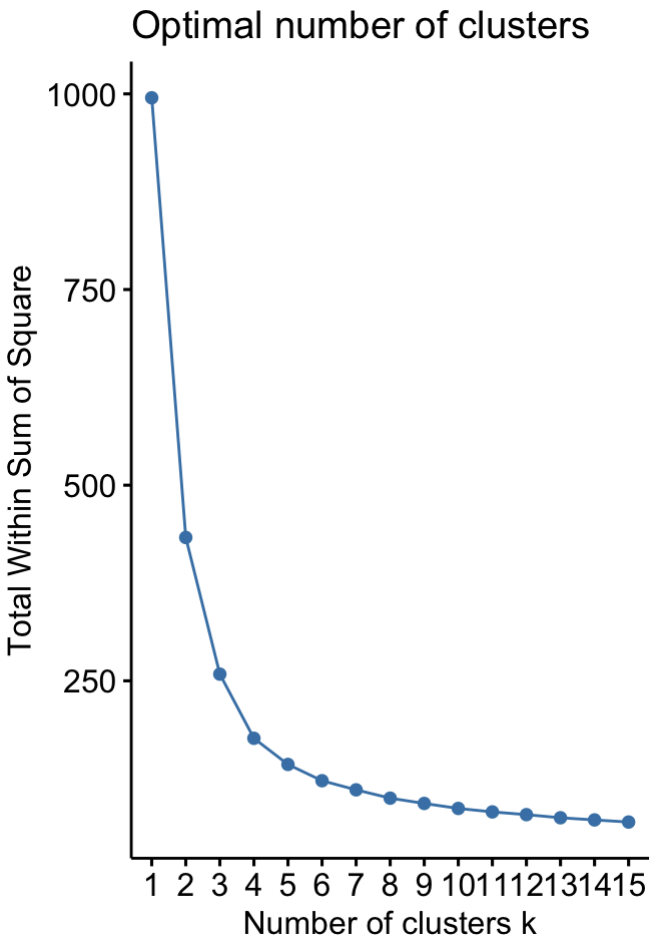
```
## Loading required package: ggplot2
```

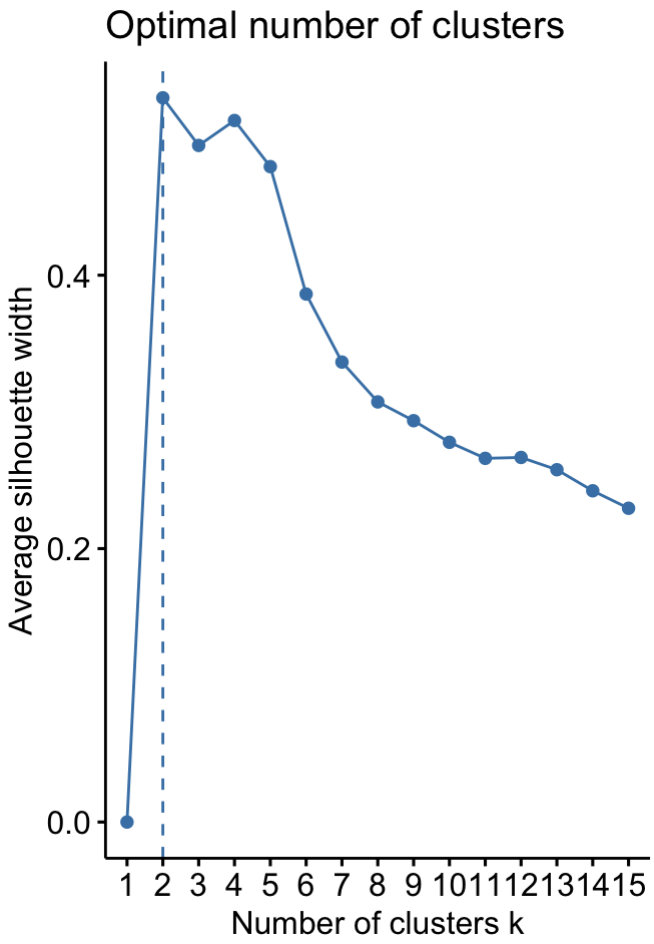
```
## Registered S3 methods overwritten by 'tibble':
##   method     from
##   format.tbl pillar
##   print.tbl  pillar
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WB
a
```

```
## Loading required package: gridExtra
```

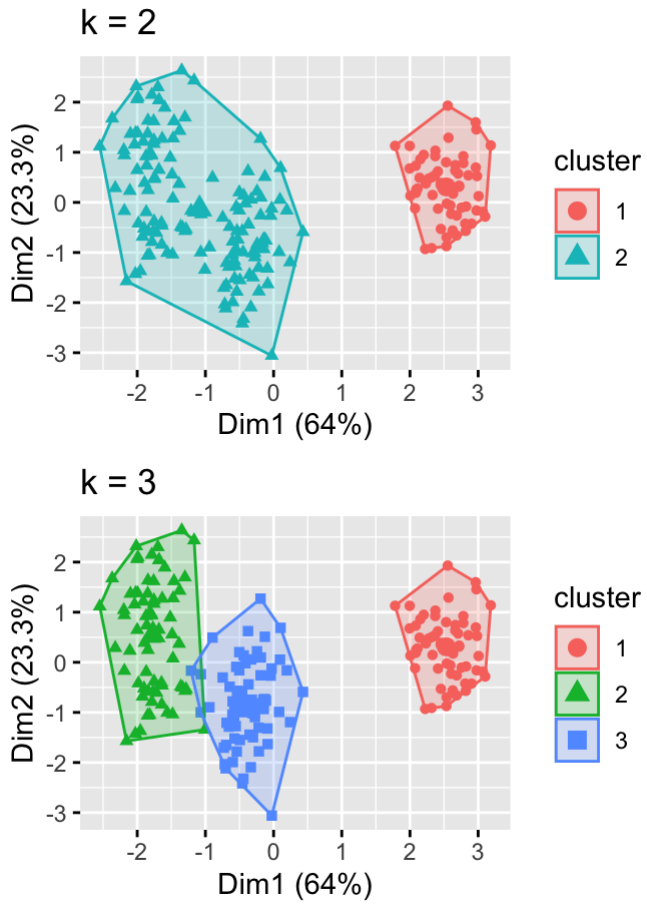
```
tmp <-  clustTest(toClust)
```



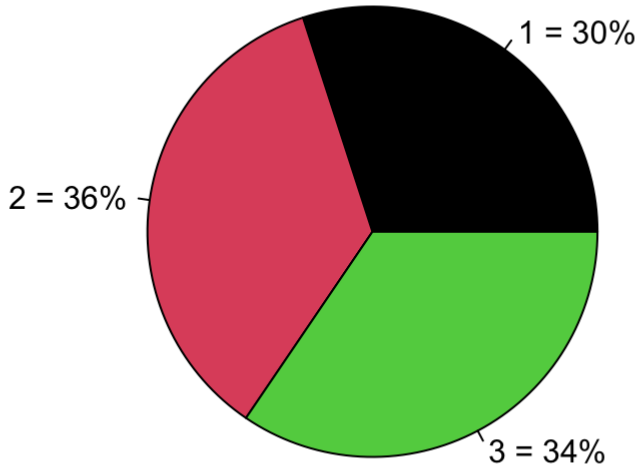


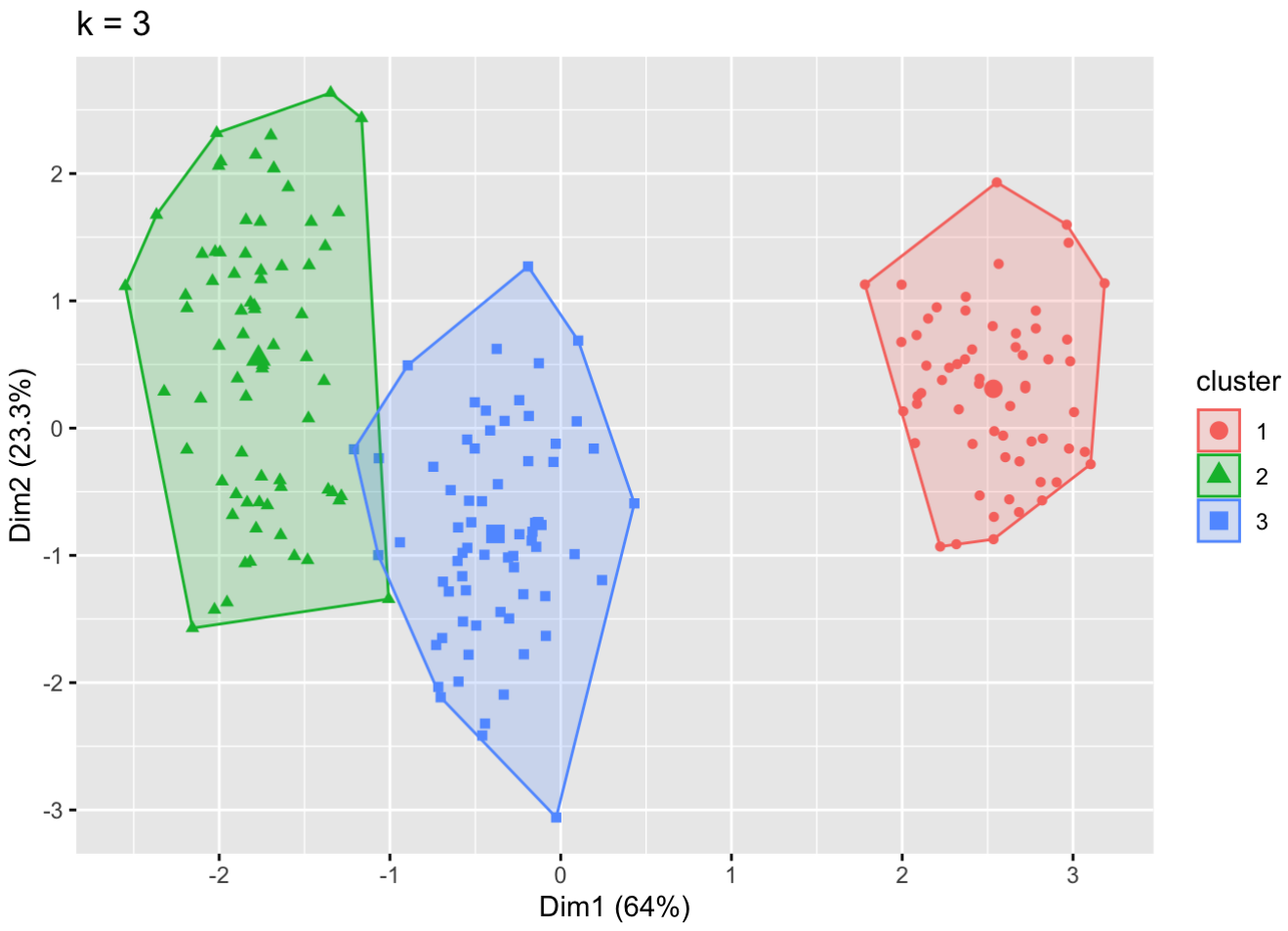

```
clusts <- runClusts(toClust, 2:3)
```

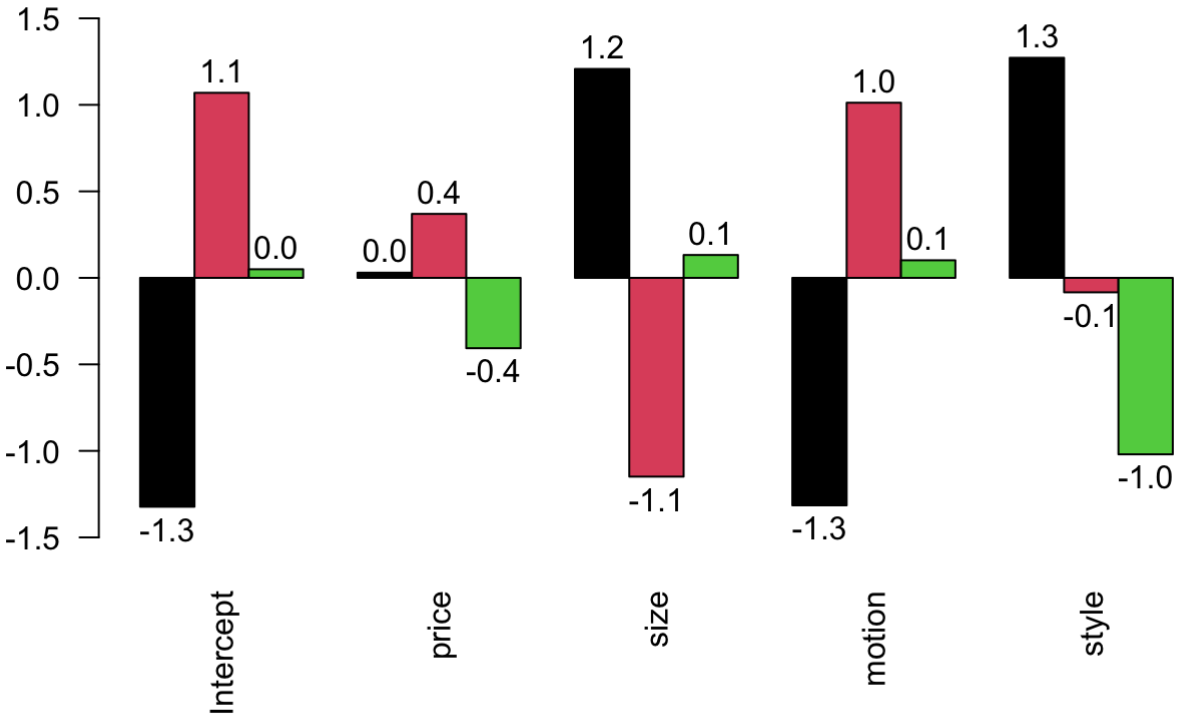
page 1 of 1




```
plotClust(clusts$kms[[2]],toClust)
```


1 = 30%
2 = 36%
3 = 34%


k = 3
Dim2 (23.3%)
Dim1 (64%)
cluster
1
2
3



Segment <int>	Intercept <dbl>	price <dbl>	size <dbl>	motion <dbl>	style <dbl>	Size <dbl>
2	10.588322	2.339789	-0.8289319	2.4392606	-0.7344484	0.355
3	7.869414	1.749094	2.0732186	0.5161534	-2.1070350	0.345
1	4.209375	2.081250	4.5086806	-2.4753472	1.2524306	0.300

3 rows

Part 3: PREDICT MISSING CELLS RATINGS

```
# predict missing cells (preparing for market simulation)
# repeat individual level partworths for multiplication
partworths.full = matrix(rep(partworths,each=16),ncol=5)
pratings = rowSums(b*partworths.full)
finalratings = ifelse(is.na(conjointData$ratings),pratings,conjointData$ratings)
finaldata = cbind(data,finalratings)[,c("ID","profile","finalratings")]
library(reshape)
```

```
##
## Attaching package: 'reshape'
```

```
## The following object is masked from 'package:data.table':
##
##     melt
```

```
finaldata <- cast(finaldata, ID ~ profile)
```

```
## Using finalratings as value column.  Use the value argument to cast to override this
choice
```

```
colnames(finaldata) <- c("ID","P1","P2","P3","P4","P5","P6","P7","P8","P9","P10","P11",
"P12","P13","P14","P15","P16")
```

Part 4: Generate profits given market simulation

```
scens = list()
scens[[1]]=c(6,14,8)    # current scenario (p5,p13; competitor:p7)
scens[[2]]=c(6,14,9)    # competitor reduces price (p5,p13; competitor:p8)

# A priori Segmentation when the competitor reduces price
scens[[3]]=c(5,9)       # Only launch segment(age == 1) preference (p4,competitor:p8)
scens[[4]]=c(7,9)       # Only launch segment(age == 0) preference (p6,competitor:p8)
scens[[5]]=c(5,7,9)     # Launch two product (p4,p6;competitor:p8)

# A Post-hoc Segmentation when the competitor reduces price
# p6,p12,p8
scens[[6]]=c(7,13,9)
# p6,p7,p8
scens[[7]]=c(7,8,9)
# p12,p7,p8
scens[[8]]=c(13,8,9)
# p6,p11,p8
scens[[9]]=c(7,12,9)
# p11,p7,p8
scens[[10]]=c(12,8,9)

# Since local retailers only sell three models so we do not consider more scenarios
```

- Calculate the market share, if there is more than 2 same highest ratings then separate the market share

```
library(matrixStats)
simFCShares = function(scens,data){
  inmkt = finaldata[,scens]
  inmkt$rowMax <- rowMaxs(as.matrix(inmkt))
  decs <- as.data.frame(ifelse(inmkt==rowMaxs(as.matrix(inmkt)),1,0))
  decs$rowMax <- NULL
  decs$rowSum <- rowSums(decs)
  decs <- as.matrix(decs)
  for (i in 1:nrow(decs)){
    if (decs[i,ncol(decs)] == 1){
      decs[i,] <- decs[i,]
    }else {
      decs[i,(1:ncol(decs)-1)][which(decs[i,(1:ncol(decs)-1)]==1)] <- 1/decs[i,ncol(dec
s)]
    }
  }
  decs <- as.data.frame(decs)
  decs <- decs[,1:length(decs)-1]
  shs = colSums(decs)/sum(decs)
  shs
}
```

- Get the market share for each senario

```
simFCShares(scens[[1]],finaldata)
```

```
##        P5       P13        P7
## 0.2233333 0.1058333 0.6708333
```

```
simFCShares(scens[[2]],finaldata)
```

```
##   P5  P13   P8
## 0.04 0.01 0.95
```

```
simFCShares(scens[[3]],finaldata)
```

```
##     P4     P8
## 0.3925 0.6075
```

```
simFCShares(scens[[4]],finaldata)
```

```
##   P6   P8
## 0.29 0.71
```

```
simFCShares(scens[[5]],finaldata)
```

```
##    P4    P6    P8
## 0.375 0.290 0.335
```

```
simFCShares(scens[[6]],finaldata)
```

```
##   P6  P12   P8
## 0.29 0.31 0.40
```

```
simFCShares(scens[[7]],finaldata)
```

```
##   P6   P7   P8
## 0.29 0.05 0.66
```

```
simFCShares(scens[[8]],finaldata)
```

```
##        P12         P7         P8
## 0.31416667 0.05416667 0.63166667
```

```
simFCShares(scens[[9]],finaldata)
```

```
##   P6  P11   P8
## 0.29 0.24 0.47
```

```
simFCShares(scens[[10]],finaldata)
```

```
##    P11     P7     P8
## 0.2375 0.0550 0.7075
```

- Calculate the profit

```
simProfit = function(scens,data,prices,vcosts,year,fcosts=20000,newProductCost=7000,mkts
ize=4000){
  mktshr = simFCShares(scens,data)
  profit <- ifelse(year == 1,
                   mktshr*mktsize*(prices-vcosts)-fcosts-newProductCost,
                   mktshr*mktsize*(prices-vcosts)-fcosts)
  profit
}
```

```
simProfit(scens[[1]],finaldata,c(139.99,139.99,139.99),c(33,33,41),c(0,0,0))
```

```
## [1]  75577.73  25292.43 245623.17
```

```
simProfit(scens[[2]],finaldata,c(139.99,139.99,119.99),c(33,33,41),c(0,0,0),20000)
```

```
## [1]  -2881.6 -15720.4 280162.0
```

```
simProfit(scens[[3]],finaldata,c(119.99,119.99),c(46,41),c(1,0),20000)
```

```
## [1]  89164.3 171945.7
```

```
simProfit(scens[[4]],finaldata,c(119.99,119.99),c(33,41),c(0,0),20000)
```

```
## [1]  80908.4 204331.6
```

```
simProfit(scens[[5]],finaldata,c(119.99,119.99,119.99),c(46,33,41),c(1,0,0),20000)
```

```
## [1] 83985.0 80908.4 85846.6
```

```
simProfit(scens[[6]],finaldata,c(119.99,119.99,119.99),c(33,46,41),c(0,1,0),20000)
```

```
## [1]  80908.4  64747.6 106384.0
```

```
simProfit(scens[[7]],finaldata,c(119.99,139.99,119.99),c(33,41,41),c(0,1,0),20000)
```

```
## [1]  80908.4  -7202.0 188533.6
```

```
simProfit(scens[[8]],finaldata,c(119.99,139.99,119.99),c(46,41,41),c(1,1,0),20000)
```

```
## [1]  65980.767  -5552.167 179581.400
```

```
simProfit(scens[[9]],finaldata,c(119.99,139.99,119.99),c(33,46,41),c(0,1,0),20000)
```

```
## [1]  80908.4  63230.4 128501.2
```

```
simProfit(scens[[10]],finaldata,c(139.99,139.99,119.99),c(46,41,41),c(1,1,0),20000)
```

```
## [1]  62290.5  -5222.2 203541.7
```