project 2

Sichun Li, Qiqi Liu, Tong Niu, Yunqing Yu, Xiao Yang

R Markdown

```
library("dummies")
```

```
## dummies-1.5.6 provided by Decision Patterns
```

```
library("AER")
```

```
## Loading required package: car
```

```
## Loading required package: carData
```

```
## Loading required package: lmtest
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: sandwich
```

```
## Loading required package: survival
```

```
library("scatterplot3d")
library("rgl")
library("data.table")
library("mlogit")
```

```
## Loading required package: Formula
```

```
library("gmnl")
```

```
## Loading required package: maxLik
```

```
## Loading required package: miscTools
```

```
##
## Please cite the 'maxLik' package as:
## Henningsen, Arne and Toomet, Ott (2011). maxLik: A package for maximum likelihood est
imation in R. Computational Statistics 26(3), 443-458. DOI 10.1007/s00180-010-0217-1.
##
## If you have questions, suggestions, or comments regarding the 'maxLik' package, pleas
e use a forum or 'tracker' at maxLik's R-Forge site:
## https://r-forge.r-project.org/projects/maxlik/
```

```
rm(list = ls());
setwd("~/Desktop/Pricing Analytics/project/project 2")
```

```
data=fread("kiwi_bubbles_P2.csv",stringsAsFactors = F)
#Data cleaning - drop periods with price=99 (stockout).
data=data[!(data$price.KB==99),]
data=data[!(data$price.KR==99),]
data=data[!(data$price.MB==99),]
```

3 Logit model without segmentation

```
#Multinomial logit
#Product-line pricing
#Solve a profit maximization problem over two products

#Write choice probability for both KB and KR as a function
#Notational change - use "para" to represent all parameter inputs,
#instead of separately defining them as "beta0KB,beta0KR,beta0MB,beta1".

demand=function(priceKB,priceKR,priceMB,para){
  probKB=exp(para[1]+para[4]*priceKB)/(1+exp(para[1]+para[4]*priceKB)+exp(para[2]+para[4
]*priceKR)+exp(para[3]+para[4]*priceMB))
  probKR=exp(para[2]+para[4]*priceKR)/(1+exp(para[1]+para[4]*priceKB)+exp(para[2]+para[4
]*priceKR)+exp(para[3]+para[4]*priceMB))
  return(cbind(probKB,probKR))
}

#Write profit as a function of prices we set and model parameters
profit=function(priceKB,priceKR,priceMB,para){
  profitKB=demand(priceKB,priceKR,priceMB,para)[,1]*(priceKB-uc)
  profitKR=demand(priceKB,priceKR,priceMB,para)[,2]*(priceKR-uc)
  return(cbind(profitKB,profitKR))
}
#Unit cost
uc=0.5;

###Estimation of multinomial logit model
#Now columns 4 through 7 contains "Price.something" info.
mlogitdata=mlogit.data(data,id="id",varying=4:7,choice="choice",shape="wide")
#Run MLE.
mle= gmnl(choice ~  price, data = mlogitdata)
summary(mle)
```

```
##
## Model estimated on: Wed Feb 12 22:39:54 2020
##
## Call:
## gmnl(formula = choice ~ price, data = mlogitdata, method = "nr")
##
## Frequencies of categories:
##
##       0      KB      KR      MB
## 0.41564 0.18035 0.20039 0.20362
##
## The estimation took: 0h:0m:0s
##
## Coefficients:
##                Estimate Std. Error z-value  Pr(>|z|)
## KB:(intercept)  4.25316    0.32821  12.959 < 2.2e-16 ***
## KR:(intercept)  4.36240    0.32945  13.241 < 2.2e-16 ***
## MB:(intercept)  4.20440    0.31331  13.419 < 2.2e-16 ***
## price          -3.73793    0.23671 -15.791 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Optimization of log-likelihood by Newton-Raphson maximisation
## Log Likelihood: -1909
## Number of observations: 1547
## Number of iterations: 4
## Exit of MLE: gradient close to zero
```

```
coef=mle$coefficients

#Set parameter
#The first element of "para" is beta0KB,beta0KR,beta0MB,beta1"
para=c(coef[1],coef[2],coef[3],coef[4])

###calculate own- and cross-price elasticities(evaluated at the average prices observed
 in the data)
meanPrice<-c(mean(data$price.KB),mean(data$price.KR),mean(data$price.MB))
demandForAll=function(priceKB,priceKR,priceMB,para){
  probKB=exp(para[1]+para[4]*priceKB)/(1+exp(para[1]+para[4]*priceKB)+exp(para[2]+para[4
]*priceKR)+exp(para[3]+para[4]*priceMB))
  probKR=exp(para[2]+para[4]*priceKR)/(1+exp(para[1]+para[4]*priceKB)+exp(para[2]+para[4
]*priceKR)+exp(para[3]+para[4]*priceMB))
  probMB=exp(para[3]+para[4]*priceMB)/(1+exp(para[1]+para[4]*priceKB)+exp(para[2]+para[4
]*priceKR)+exp(para[3]+para[4]*priceMB))
  return(cbind(probKB,probKR,probMB))
}
prob<-demandForAll(meanPrice[1],meanPrice[2],meanPrice[3],para)

ownElasticity=function(beta1,price,prob){
  ownElasticity=-beta1*price*(1-prob)
  return(ownElasticity)
}
crossElasticity=function(beta1,price,prob){
  crossElasticity=-beta1*price*prob
  return(crossElasticity)
}

#built a matrix for both own- and cross- ealsticity

ElastMatrix<-data.frame(matrix(ncol = 3, nrow = 3))
colnames(ElastMatrix)<- c("KB",'KR','MB')
rownames(ElastMatrix)<- c("KB",'KR','MB')

for(i in 1:3){
  for(j in 1:3){#Products in column change prices and then influence products in rows
    if (rownames(ElastMatrix)[i]==colnames(ElastMatrix)[j]){
      ElastMatrix[i,j]=ownElasticity(para[4],meanPrice[i],prob[i])}
    else{ElastMatrix[i,j]=crossElasticity(para[4],meanPrice[j],prob[j])}
  }
}


###calculate profit
#"demand" function represents each individual consumer's choice probability.
#In order to calculate profit, we multiply the "demand" by the number of consumers.

#Choose space of prices to search for the optimal price over
aux=seq(0.88,1.47,0.01)
#Because we search over two dimensions, create complete combination
#of the two prices
pricespace=expand.grid(aux,aux)
```

```
colnames(pricespace)=c('priceKB','priceKR')
#Compute profit at each realization of this price space.
#write for-loop, take one realization of [P^KB,P^KR] pair and evaluate
#profit at that realization.
profitmat=matrix(0L,nrow(pricespace),1)
for (i in 1:nrow(pricespace)){
    profitmat[i]=sum(profit(pricespace[i,1],pricespace[i,2],1.43,para))
}
expectedProfit=1000*max(profitmat[,1])
optimalPrices=pricespace[which.max(profitmat[,1]),]
optimalPrices
```

```
##      priceKB priceKR
## 1709    1.16    1.16
```

```
#Both optimal KB price and optimal KR are 1.16.
```

4 Logit model with segmentation

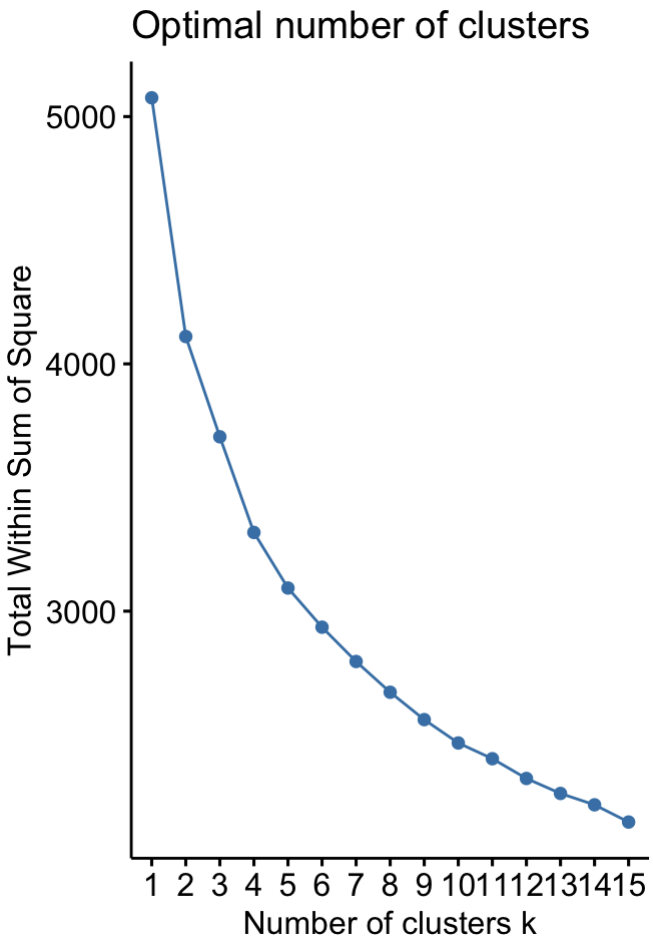
```
library(cluster)
library(fpc)
library(factoextra)
```

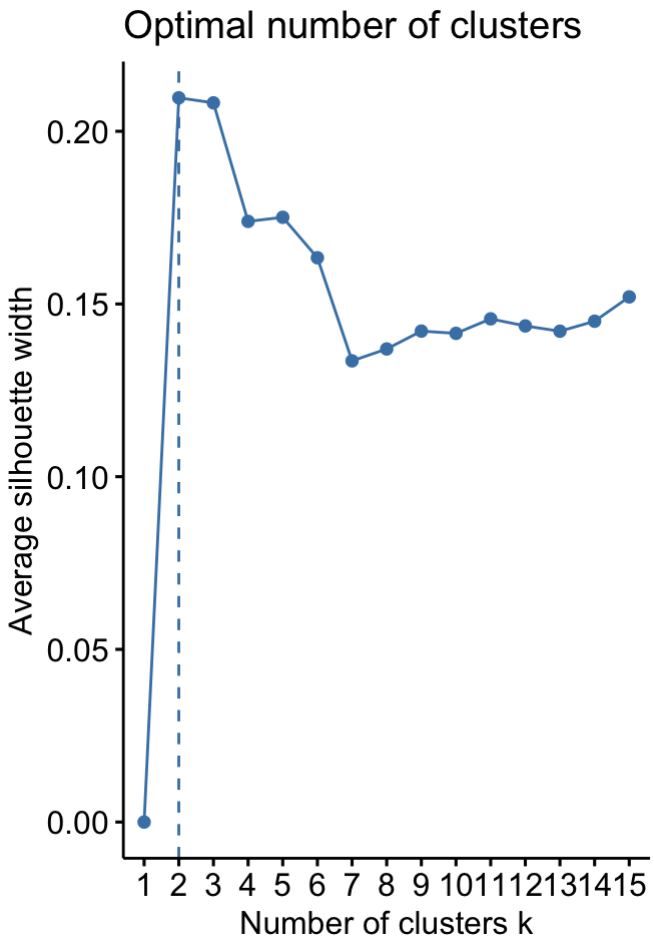
```
## Loading required package: ggplot2
```

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

```
library(gridExtra)
library(conjoint)
demo=fread("demo_P2.csv",stringsAsFactors = F)
#Number of individuals
N = length(unique(data$id))

#Clustering
clustTest = function(toClust,print=TRUE,scale=TRUE,maxClusts=15,seed=12345,nstart=20,ite
r.max=100){
    if(scale){ toClust = scale(toClust);}
    set.seed(seed);   # set random number seed before doing cluster analysis
    wss <- (nrow(toClust)-1)*sum(apply(toClust,2,var))
    for (i in 2:maxClusts) wss[i] <- sum(kmeans(toClust,centers=i,nstart=nstart,iter.max
=iter.max)$withinss)
    ##gpw essentially does the following plot using wss above.
    #plot(1:maxClusts, wss, type="b", xlab="Number of Clusters",ylab="Within groups sum
 of squares")
    gpw = fviz_nbclust(toClust,kmeans,method="wss",iter.max=iter.max,nstart=nstart,k.max
=maxClusts) #alternative way to get wss elbow chart.
    pm1 = pamk(toClust,scaling=TRUE)
    ## pm1$nc indicates the optimal number of clusters based on
    ## lowest average silhoutte score (a measure of quality of clustering)
    #alternative way that presents it visually as well.
    gps = fviz_nbclust(toClust,kmeans,method="silhouette",iter.max=iter.max,nstart=nstar
t,k.max=maxClusts)
    if(print){
        grid.arrange(gpw,gps, nrow = 1)
    }
    list(wss=wss,pm1=pm1$nc,gpw=gpw,gps=gps)
}
##Runs a set of clusters as kmeans
##Arguments:
##  toClust, data.frame with data to cluster
##  nClusts, vector of number of clusters, each run as separate kmeans
##  ... some additional arguments to be passed to clusters
##Return:
##  list of
##    kms, kmeans cluster output with length of nClusts
##    ps, list of plots of the clusters against first 2 principle components
runClusts = function(toClust,nClusts,print=TRUE,maxClusts=15,seed=12345,nstart=20,iter.m
ax=100){
    kms=list(); ps=list();
    for(i in 1:length(nClusts)){
        kms[[i]] = kmeans(toClust,nClusts[i],iter.max = iter.max, nstart=nstart)
        ps[[i]] = fviz_cluster(kms[[i]], geom = "point", data = toClust) + ggtitle(paste
("k =",nClusts[i]))

    }
    library(gridExtra)
    if(print){
        tmp = marrangeGrob(ps, nrow = 2,ncol=2)
        print(tmp)
    }
```

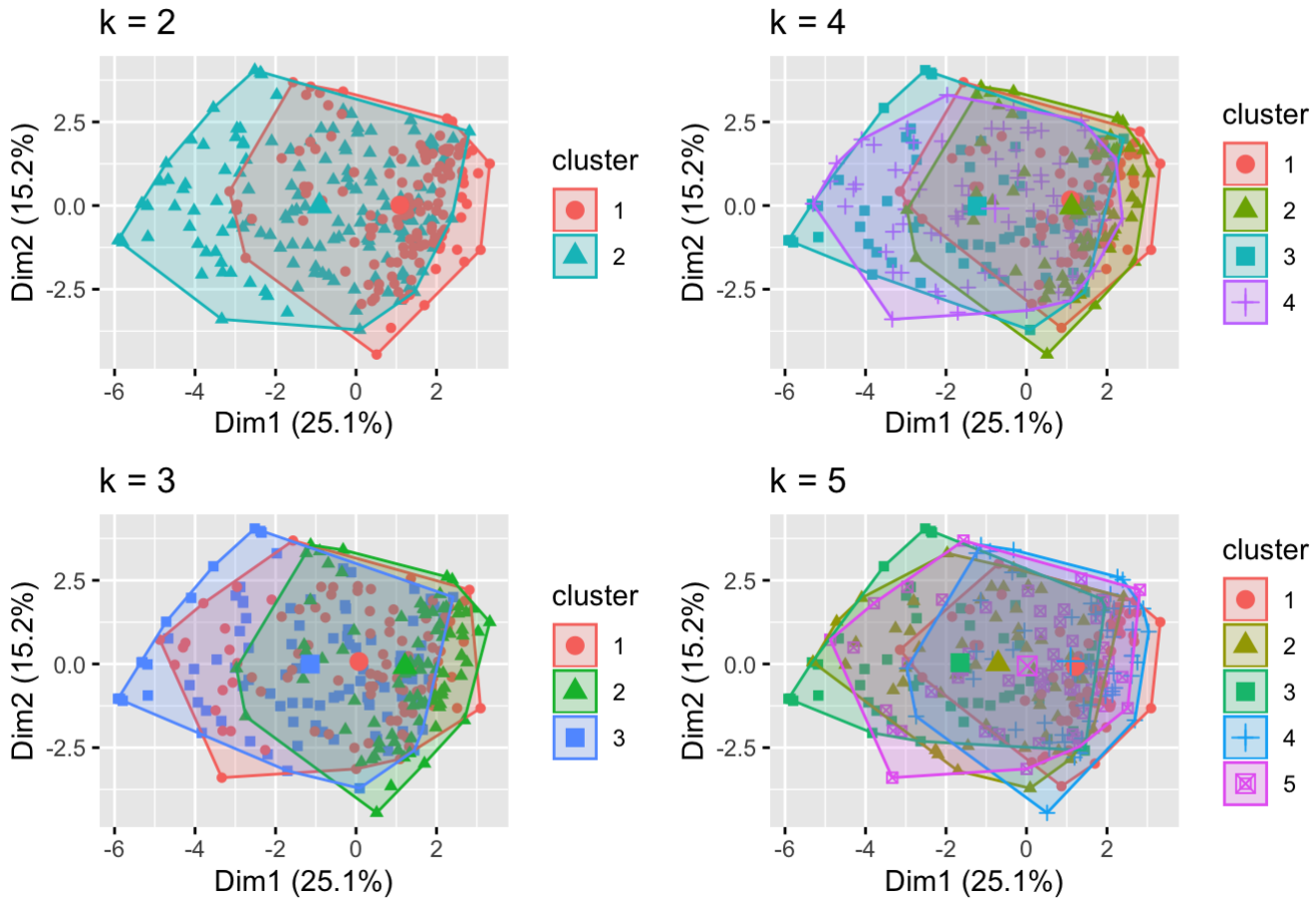
```
    list(kms=kms,ps=ps)
}
##Plots a kmeans cluster as three plot report
##  pie chart with membership percentages
##  ellipse plot that indicates cluster definitions against principle components
##  barplot of the cluster means
plotClust = function(km,toClust,discPlot=FALSE){
    nc = length(km$size)
    if(discPlot){par(mfrow=c(2,2))}
    else {par(mfrow=c(3,1))}
    percsize = paste(1:nc," = ",format(km$size/sum(km$size)*100,digits=2),"%",sep="")
    pie(km$size,labels=percsize,col=1:nc)

    clusplot(toClust, km$cluster, color=TRUE, shade=TRUE,
             labels=2, lines=0,col.clus=1:nc); #plot clusters against principal componen
ts

    if(discPlot){
        plotcluster(toClust, km$cluster,col=km$cluster); #plot against discriminant func
tions ()
    }
    rng = range(km$centers)
    dist = rng[2]-rng[1]
    locs = km$centers+.05*dist*ifelse(km$centers>0,1,-1)
    bm = barplot(km$centers,beside=TRUE,col=1:nc,main="Cluster Means",ylim=rng+dist*c(-.
1,.1))
    text(bm,locs,formatC(km$centers,format="f",digits=1))
}

checks = clustTest(demo)
```





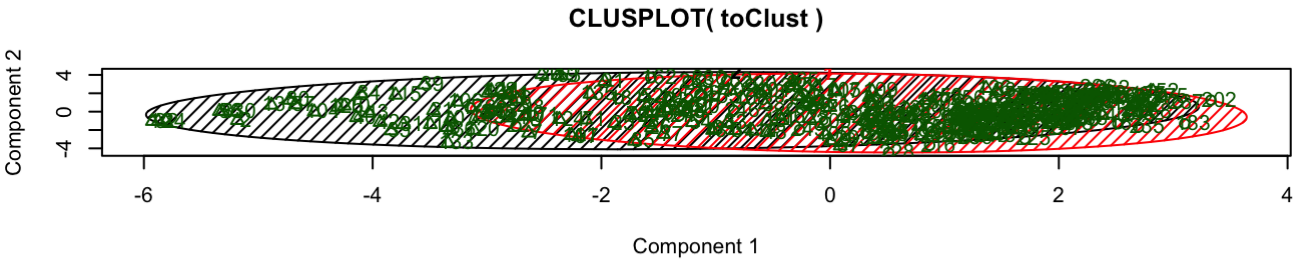

```
clusts=runClusts(demo,2:5)
```

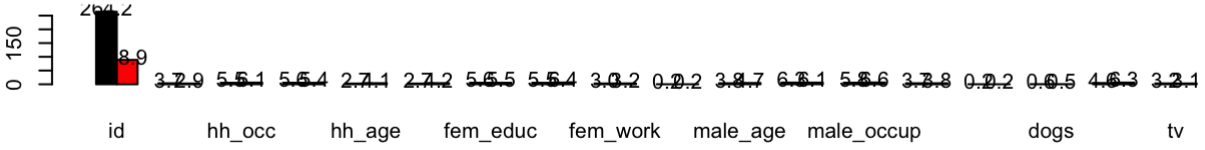
page 1 of 1



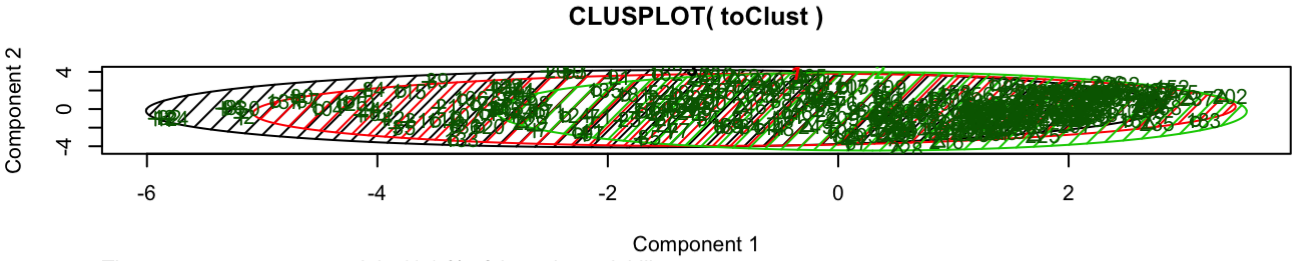

```
for(i in 1:4) {plotClust(clusts[[1]][[i]],demo)}
```

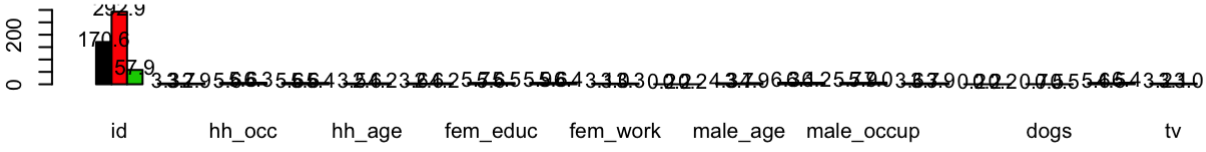

1 = 46%
2 = 54%


CLUSPLOT(toClust)
Component 2
Component 1
These two components explain 40.3 % of the point variability.

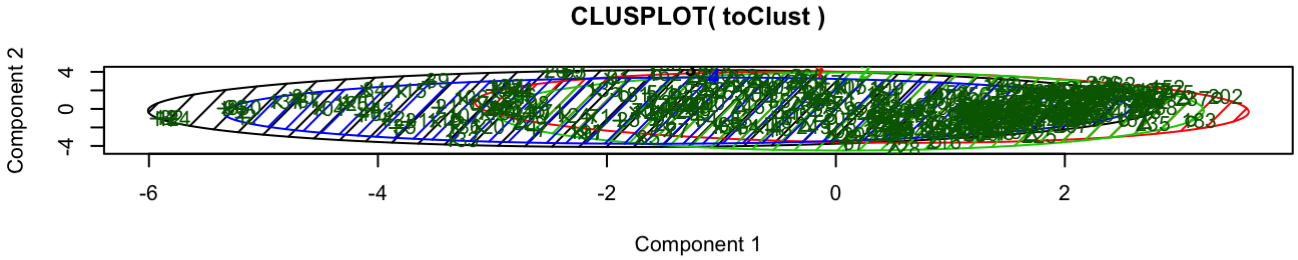

Cluster Means
id
hh_occ
hh_age
fem_educ
fem_work
male_age
male_occup
dogs
tv

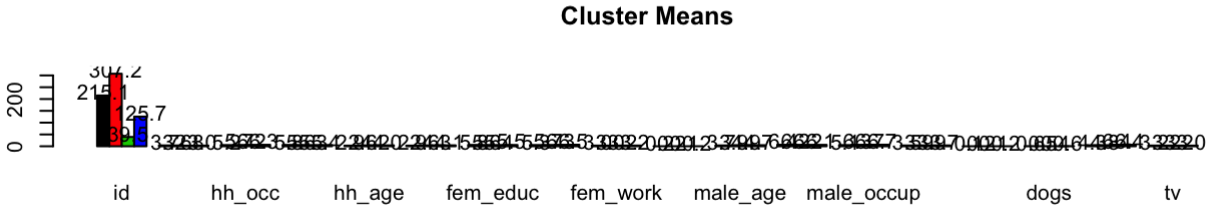

1 = 34%
2 = 31%
3 = 35%


CLUSPLOT(toClust)
Component 2
Component 1
These two components explain 40.3 % of the point variability.

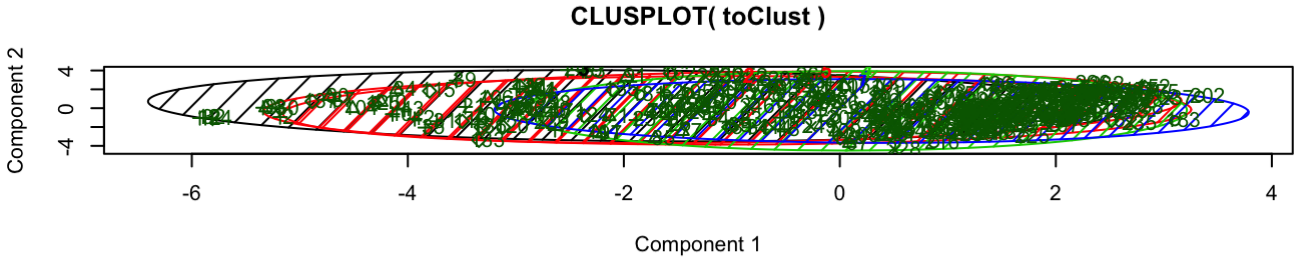

Cluster Means
id
hh_occ
hh_age
fem_educ
fem_work
male_age
male_occup
dogs
tv

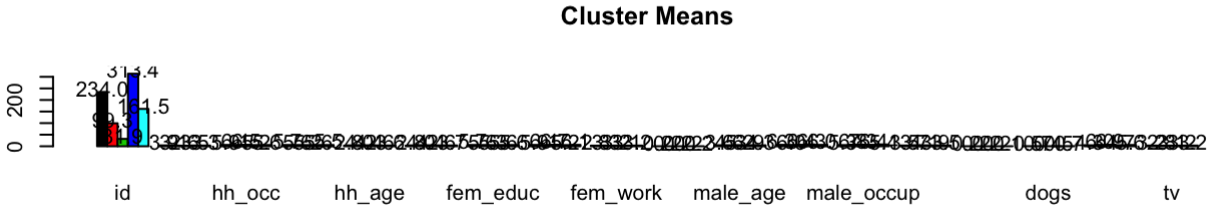

2 = 24%
1 = 23%
3 = 24%
4 = 29%


CLUSPLOT(toClust)
Component 2
Component 1
These two components explain 40.3 % of the point variability.


Cluster Means
id
hh_occ
hh_age
fem_educ
fem_work
male_age
male_occup
dogs
tv


2 = 20%
1 = 20%
3 = 20%
5 = 20%
4 = 20%


CLUSPLOT(toClust)
Component 2
Component 1
These two components explain 40.3 % of the point variability.


Cluster Means
id
hh_occ
hh_age
fem_educ
fem_work
male_age
male_occup
dogs
tv

```
KmeansCluster = function(n){
  set.seed(123)
  demo_cluster = kmeans(x=demo[, 2:18], centers = n, nstart = 1000)
  cluster_id = data.frame(id = demo$id)
  cluster_id$cluster = demo_cluster$cluster
  datafull = merge(data, cluster_id, by = "id", all.x = T)
  datafull$cluster[is.na(datafull$cluster)] = n+1
  N = length(unique(data$id))
  seg.share = c( table(demo_cluster$cluster),N - sum(table(demo_cluster$cluster))) / N

# just store the coefficients (you can store many other things)
  coef.est = data.frame(segment = 1:(n+1), intercept.KB = NA, intercept.KR = NA,
                        intercept.MB = NA, price.coef = NA)
    #Write a for-loop.
  for (seg in 1:(n+1)) {
  # During each loop, pick subset of data of consumers from each segment.
      data.sub = subset(datafull, cluster %in% seg)
      mlogitdata=mlogit.data(data.sub,id="id",varying=4:7,choice="choice",shape="wide")

  #Run MLE.
      mle= gmnl(choice ~  price, data = mlogitdata)
  #Store the outcome in the coef.est matrix.
      coef.est[seg, 2:5] = mle$coefficients

}
      return(list(seg.share,datafull,coef.est))
}

KmeansCluster(2)  #change 0
```

```
## [[1]]
##         1         2
## 0.3617021 0.4984802 0.1398176
##
## [[2]]
##        id week trip price.0 price.KB price.KR price.MB choice cluster
##    1:   1   96    1       0     1.43     1.43     1.43      0       2
##    2:   2   14    1       0     1.43     1.43     1.65      0       1
##    3:   2   25    2       0     1.43     1.43     1.65      0       1
##    4:   2   26    3       0     1.43     1.43     1.65      0       1
##    5:   2   31    4       0     1.43     0.88     1.65     KB       1
##   ---
## 1543: 358  124    8       0     1.43     1.43     1.35     MB       2
## 1544: 358  130    9       0     1.43     1.43     1.14     KB       2
## 1545: 358  140   10       0     1.47     1.43     1.11     KB       2
## 1546: 359   81    1       0     1.43     1.43     1.33      0       3
## 1547: 359   94    2       0     0.90     0.89     1.43      0       3
##
## [[3]]
##   segment intercept.KB intercept.KR intercept.MB price.coef
## 1       1     4.044336     4.340585     4.209052  -3.615491
## 2       2     4.062756     4.366433     4.115583  -3.737853
## 3       3     5.117430     4.509340     4.544963  -4.062526
```

```
KmeansCluster(3)  #change 11.2462%
```

```
## [[1]]
##         1         2         3
## 0.3404255 0.1124620 0.4072948 0.1398176
##
## [[2]]
##        id week trip price.0 price.KB price.KR price.MB choice cluster
##    1:   1   96    1       0     1.43     1.43     1.43      0       3
##    2:   2   14    1       0     1.43     1.43     1.65      0       1
##    3:   2   25    2       0     1.43     1.43     1.65      0       1
##    4:   2   26    3       0     1.43     1.43     1.65      0       1
##    5:   2   31    4       0     1.43     0.88     1.65     KB       1
##   ---
## 1543: 358  124    8       0     1.43     1.43     1.35     MB       3
## 1544: 358  130    9       0     1.43     1.43     1.14     KB       3
## 1545: 358  140   10       0     1.47     1.43     1.11     KB       3
## 1546: 359   81    1       0     1.43     1.43     1.33      0       4
## 1547: 359   94    2       0     0.90     0.89     1.43      0       4
##
## [[3]]
##   segment intercept.KB intercept.KR intercept.MB price.coef
## 1       1     3.855888     4.197239     3.856256  -3.399596
## 2       2     4.244438     4.043417     4.064278  -3.883398
## 3       3     4.202353     4.630657     4.503406  -3.887940
## 4       4     5.117430     4.509340     4.544963  -4.062526
```

```
KmeansCluster(4)  #change 10.6383%
```

```
## [[1]]
##         1         2         3         4
## 0.1063830 0.2127660 0.3829787 0.1580547 0.1398176
##
## [[2]]
##        id week trip price.0 price.KB price.KR price.MB choice cluster
##    1:   1   96    1       0     1.43     1.43     1.43      0       3
##    2:   2   14    1       0     1.43     1.43     1.65      0       4
##    3:   2   25    2       0     1.43     1.43     1.65      0       4
##    4:   2   26    3       0     1.43     1.43     1.65      0       4
##    5:   2   31    4       0     1.43     0.88     1.65     KB       4
##   ---
## 1543: 358  124    8       0     1.43     1.43     1.35     MB       3
## 1544: 358  130    9       0     1.43     1.43     1.14     KB       3
## 1545: 358  140   10       0     1.47     1.43     1.11     KB       3
## 1546: 359   81    1       0     1.43     1.43     1.33      0       5
## 1547: 359   94    2       0     0.90     0.89     1.43      0       5
##
## [[3]]
##   segment intercept.KB intercept.KR intercept.MB price.coef
## 1       1     4.149431     3.947833     3.948372  -3.774973
## 2       2     4.085952     4.316191     4.577172  -3.798219
## 3       3     3.864525     4.347728     4.017122  -3.600028
## 4       4     4.335498     4.674025     4.111057  -3.682239
## 5       5     5.117430     4.509340     4.544963  -4.062526
```

```
KmeansCluster(5)  #change 10.6383%
```

```
## [[1]]
##         1         2         3         4         5
## 0.1063830 0.1246201 0.3647416 0.1033435 0.1610942 0.1398176
##
## [[2]]
##        id week trip price.0 price.KB price.KR price.MB choice cluster
##    1:   1   96    1       0     1.43     1.43     1.43      0       3
##    2:   2   14    1       0     1.43     1.43     1.65      0       5
##    3:   2   25    2       0     1.43     1.43     1.65      0       5
##    4:   2   26    3       0     1.43     1.43     1.65      0       5
##    5:   2   31    4       0     1.43     0.88     1.65     KB       5
##   ---
## 1543: 358  124    8       0     1.43     1.43     1.35     MB       3
## 1544: 358  130    9       0     1.43     1.43     1.14     KB       3
## 1545: 358  140   10       0     1.47     1.43     1.11     KB       3
## 1546: 359   81    1       0     1.43     1.43     1.33      0       6
## 1547: 359   94    2       0     0.90     0.89     1.43      0       6
##
## [[3]]
##   segment intercept.KB intercept.KR intercept.MB price.coef
## 1       1     4.149431     3.947833     3.948372  -3.774973
## 2       2     3.868998     4.354056     4.052386  -3.502896
## 3       3     3.766130     4.246840     3.920580  -3.530290
## 4       4     4.808604     4.606528     5.629481  -4.517302
## 5       5     4.354360     4.692998     4.150623  -3.695810
## 6       6     5.117430     4.509340     4.544963  -4.062526
```

```
KmeansCluster(6)  #change 0
```

```
## [[1]]
##          1          2          3          4          5          6
## 0.09118541 0.17021277 0.10334347 0.12462006 0.20972644 0.16109422
##
## 0.13981763
##
## [[2]]
##        id week trip price.0 price.KB price.KR price.MB choice cluster
##    1:   1   96    1       0     1.43     1.43     1.43      0       2
##    2:   2   14    1       0     1.43     1.43     1.65      0       6
##    3:   2   25    2       0     1.43     1.43     1.65      0       6
##    4:   2   26    3       0     1.43     1.43     1.65      0       6
##    5:   2   31    4       0     1.43     0.88     1.65     KB       6
##   ---
## 1543: 358  124    8       0     1.43     1.43     1.35     MB       5
## 1544: 358  130    9       0     1.43     1.43     1.14     KB       5
## 1545: 358  140   10       0     1.47     1.43     1.11     KB       5
## 1546: 359   81    1       0     1.43     1.43     1.33      0       7
## 1547: 359   94    2       0     0.90     0.89     1.43      0       7
##
## [[3]]
##   segment intercept.KB intercept.KR intercept.MB price.coef
## 1       1     3.761442     3.899505     3.816808  -3.606284
## 2       2     5.823323     5.329806     6.050890  -5.003794
## 3       3     4.808604     4.606528     5.629481  -4.517302
## 4       4     3.868998     4.354056     4.052386  -3.502896
## 5       5     2.521333     3.370624     2.253880  -2.554080
## 6       6     4.354360     4.692998     4.150623  -3.695810
## 7       7     5.117430     4.509340     4.544963  -4.062526
```

```
KmeansCluster(7)  #change 11.550152%
```

```
## [[1]]
##          1          2          3          4          5          6
## 0.11550152 0.16109422 0.11854103 0.14285714 0.12462006 0.09422492
##          7
## 0.10334347 0.13981763
##
## [[2]]
##        id week trip price.0 price.KB price.KR price.MB choice cluster
##    1:   1   96    1       0     1.43     1.43     1.43      0       3
##    2:   2   14    1       0     1.43     1.43     1.65      0       2
##    3:   2   25    2       0     1.43     1.43     1.65      0       2
##    4:   2   26    3       0     1.43     1.43     1.65      0       2
##    5:   2   31    4       0     1.43     0.88     1.65     KB       2
##   ---
## 1543: 358  124    8       0     1.43     1.43     1.35     MB       4
## 1544: 358  130    9       0     1.43     1.43     1.14     KB       4
## 1545: 358  140   10       0     1.47     1.43     1.11     KB       4
## 1546: 359   81    1       0     1.43     1.43     1.33      0       8
## 1547: 359   94    2       0     0.90     0.89     1.43      0       8
##
## [[3]]
##   segment intercept.KB intercept.KR intercept.MB price.coef
## 1       1     7.606395     6.661934     7.477539  -5.897474
## 2       2     4.354360     4.692998     4.150623  -3.695810
## 3       3     2.333681     3.112574     2.925201  -2.896447
## 4       4     2.969402     3.867674     2.727610  -2.909001
## 5       5     3.868998     4.354056     4.052386  -3.502896
## 6       6     3.997297     3.958938     3.883755  -3.715366
## 7       7     4.808604     4.606528     5.629481  -4.517302
## 8       8     5.117430     4.509340     4.544963  -4.062526
```

```
KmeansCluster(8)  #change 11.550152%
```

```
## [[1]]
##          1          2          3          4          5          6
## 0.09422492 0.06990881 0.14285714 0.11854103 0.09422492 0.10334347
##          7          8
## 0.11550152 0.12158055 0.13981763
##
## [[2]]
##        id week trip price.0 price.KB price.KR price.MB choice cluster
##    1:   1   96    1       0     1.43     1.43     1.43      0       4
##    2:   2   14    1       0     1.43     1.43     1.65      0       5
##    3:   2   25    2       0     1.43     1.43     1.65      0       5
##    4:   2   26    3       0     1.43     1.43     1.65      0       5
##    5:   2   31    4       0     1.43     0.88     1.65     KB       5
##   ---
## 1543: 358  124    8       0     1.43     1.43     1.35     MB       3
## 1544: 358  130    9       0     1.43     1.43     1.14     KB       3
## 1545: 358  140   10       0     1.47     1.43     1.11     KB       3
## 1546: 359   81    1       0     1.43     1.43     1.33      0       9
## 1547: 359   94    2       0     0.90     0.89     1.43      0       9
##
## [[3]]
##   segment intercept.KB intercept.KR intercept.MB price.coef
## 1       1    3.9972969     3.958938    3.8837551  -3.715366
## 2       2    7.3034174     7.138563    7.1181389  -5.793619
## 3       3    2.9694016     3.867674    2.7276100  -2.909001
## 4       4    2.3336806     3.112574    2.9252012  -2.896447
## 5       5    0.9169255     1.673183    0.4573439  -1.251711
## 6       6    4.8086045     4.606528    5.6294806  -4.517302
## 7       7    7.6063946     6.661934    7.4775392  -5.897474
## 8       8    3.8689983     4.354056    4.0523865  -3.502896
## 9       9    5.1174301     4.509340    4.5449628  -4.062526
```

```
KmeansCluster(9)  #change 11.550152%
```

```
## [[1]]
##          1          2          3          4          5          6
## 0.11550152 0.01215805 0.12765957 0.09422492 0.06079027 0.11854103
##          7          8          9
## 0.09726444 0.14589666 0.08814590 0.13981763
##
## [[2]]
##        id week trip price.0 price.KB price.KR price.MB choice cluster
##    1:   1   96    1       0     1.43     1.43     1.43      0       6
##    2:   2   14    1       0     1.43     1.43     1.65      0       7
##    3:   2   25    2       0     1.43     1.43     1.65      0       7
##    4:   2   26    3       0     1.43     1.43     1.65      0       7
##    5:   2   31    4       0     1.43     0.88     1.65     KB       7
##   ---
## 1543: 358  124    8       0     1.43     1.43     1.35     MB       8
## 1544: 358  130    9       0     1.43     1.43     1.14     KB       8
## 1545: 358  140   10       0     1.47     1.43     1.11     KB       8
## 1546: 359   81    1       0     1.43     1.43     1.33      0      10
## 1547: 359   94    2       0     0.90     0.89     1.43      0      10
##
## [[3]]
##    segment intercept.KB intercept.KR intercept.MB price.coef
## 1        1     7.606395     6.661934     7.477539  -5.897474
## 2        2     7.095079     8.325625     7.494956  -6.299830
## 3        3     3.929391     4.369639     4.065201  -3.548877
## 4        4     3.997297     3.958938     3.883755  -3.715366
## 5        5     7.064807     6.909717     6.782931  -5.494300
## 6        6     2.333681     3.112574     2.925201  -2.896447
## 7        7     1.194957     1.952011     1.005984  -1.470361
## 8        8     3.001763     3.916981     2.762062  -2.951180
## 9        9     4.791133     4.456635     5.690110  -4.606762
## 10      10     5.117430     4.509340     4.544963  -4.062526
```

```
KmeansCluster(10) #5,8 change 0.11550152+0.07294833=0.1884499 ######seems the best
```

```
## [[1]]
##          1          2          3          4          5          6
## 0.06686930 0.11854103 0.14589666 0.05775076 0.11550152 0.08814590
##          7          8          9         10
## 0.08814590 0.07294833 0.01215805 0.09422492 0.13981763
##
## [[2]]
##        id week trip price.0 price.KB price.KR price.MB choice cluster
##    1:   1   96    1       0     1.43     1.43     1.43      0       2
##    2:   2   14    1       0     1.43     1.43     1.65      0       7
##    3:   2   25    2       0     1.43     1.43     1.65      0       7
##    4:   2   26    3       0     1.43     1.43     1.65      0       7
##    5:   2   31    4       0     1.43     0.88     1.65     KB       7
##   ---
## 1543: 358  124    8       0     1.43     1.43     1.35     MB       3
## 1544: 358  130    9       0     1.43     1.43     1.14     KB       3
## 1545: 358  140   10       0     1.47     1.43     1.11     KB       3
## 1546: 359   81    1       0     1.43     1.43     1.33      0      11
## 1547: 359   94    2       0     0.90     0.89     1.43      0      11
##
## [[3]]
##    segment intercept.KB intercept.KR intercept.MB price.coef
## 1        1    3.0878338     4.006679    3.1802458  -2.826519
## 2        2    2.3336806     3.112574    2.9252012  -2.896447
## 3        3    3.0017626     3.916981    2.7620620  -2.951180
## 4        4    6.6556359     6.515320    6.4858148  -5.043962
## 5        5    7.6063946     6.661934    7.4775392  -5.897474
## 6        6    4.7911334     4.456635    5.6901102  -4.606762
## 7        7    0.8341597     1.673929    0.4294365  -1.233209
## 8        8    8.2287472     6.680553    8.1223860  -6.788933
## 9        9    7.0950794     8.325625    7.4949559  -6.299830
## 10      10    3.9972969     3.958938    3.8837551  -3.715366
## 11      11    5.1174301     4.509340    4.5449628  -4.062526
```

```
KmeansCluster(11) #1,3 change 0.07294833+0.09422492=0.1671732
```

```
## [[1]]
##          1          2          3          4          5          6
## 0.07294833 0.08814590 0.09422492 0.07598784 0.08510638 0.01215805
##          7          8          9         10         11
## 0.09118541 0.05775076 0.06686930 0.12765957 0.08814590 0.13981763
##
## [[2]]
##        id week trip price.0 price.KB price.KR price.MB choice cluster
##    1:   1   96    1       0     1.43     1.43     1.43      0       4
##    2:   2   14    1       0     1.43     1.43     1.65      0       2
##    3:   2   25    2       0     1.43     1.43     1.65      0       2
##    4:   2   26    3       0     1.43     1.43     1.65      0       2
##    5:   2   31    4       0     1.43     0.88     1.65     KB       2
##   ---
## 1543: 358  124    8       0     1.43     1.43     1.35     MB      10
## 1544: 358  130    9       0     1.43     1.43     1.14     KB      10
## 1545: 358  140   10       0     1.47     1.43     1.11     KB      10
## 1546: 359   81    1       0     1.43     1.43     1.33      0      12
## 1547: 359   94    2       0     0.90     0.89     1.43      0      12
##
## [[3]]
##    segment intercept.KB intercept.KR intercept.MB price.coef
## 1        1    8.2287472     6.680553    8.1223860  -6.788933
## 2        2    0.8341597     1.673929    0.4294365  -1.233209
## 3        3    8.3543043     7.050992    7.5755872  -6.105677
## 4        4    4.8261939     5.128052    5.9354245  -4.953841
## 5        5    1.2268129     1.971161    0.7964498  -1.768687
## 6        6    7.0950794     8.325625    7.4949559  -6.299830
## 7        7    3.7614419     3.899505    3.8168079  -3.606284
## 8        8    6.6556359     6.515320    6.4858148  -5.043962
## 9        9    3.0878338     4.006679    3.1802458  -2.826519
## 10      10    3.1233044     4.039871    2.9078149  -2.928321
## 11      11    4.7911334     4.456635    5.6901102  -4.606762
## 12      12    5.1174301     4.509340    4.5449628  -4.062526
```

```
KmeansCluster(12) #6   change 0.07294833
```

```
## [[1]]
##          1          2          3          4          5          6
## 0.05167173 0.06990881 0.04863222 0.14285714 0.06686930 0.07294833
##          7          8          9         10         11         12
## 0.07294833 0.07294833 0.07902736 0.08814590 0.08206687 0.01215805
##
## 0.13981763
##
## [[2]]
##        id week trip price.0 price.KB price.KR price.MB choice cluster
##    1:   1   96    1       0     1.43     1.43     1.43      0       9
##    2:   2   14    1       0     1.43     1.43     1.65      0       7
##    3:   2   25    2       0     1.43     1.43     1.65      0       7
##    4:   2   26    3       0     1.43     1.43     1.65      0       7
##    5:   2   31    4       0     1.43     0.88     1.65     KB       7
##   ---
## 1543: 358  124    8       0     1.43     1.43     1.35     MB       4
## 1544: 358  130    9       0     1.43     1.43     1.14     KB       4
## 1545: 358  140   10       0     1.47     1.43     1.11     KB       4
## 1546: 359   81    1       0     1.43     1.43     1.33      0      13
## 1547: 359   94    2       0     0.90     0.89     1.43      0      13
##
## [[3]]
##    segment intercept.KB intercept.KR intercept.MB price.coef
## 1        1   -0.6858488    0.2551593   -0.3565393 -0.4809951
## 2        2    3.6289901    3.7905660    3.6577125 -3.3942762
## 3        3    6.4935826    6.4118734    6.2248355 -4.9501555
## 4        4    2.9614638    3.8592797    2.7253128 -2.8735580
## 5        5    3.0878338    4.0066788    3.1802458 -2.8265188
## 6        6    8.2287472    6.6805533    8.1223860 -6.7889333
## 7        7    3.1064888    3.5921424    2.5508203 -3.0879894
## 8        8    4.9293282    4.7868075    6.2611317 -5.2083860
## 9        9    2.5411065    3.4178283    2.2300940 -2.5852829
## 10      10    8.5534835    7.2204306    7.7340581 -6.1737253
## 11      11    4.6273679    4.2912871    5.5318158 -4.3774219
## 12      12    7.0950794    8.3256252    7.4949559 -6.2998296
## 13      13    5.1174301    4.5093403    4.5449628 -4.0625262
```

```
KmeansCluster(13) #change 0
```

```
## [[1]]
##          1          2          3          4          5          6
## 0.07598784 0.05471125 0.08814590 0.06686930 0.08206687 0.01215805
##          7          8          9         10         11         12
## 0.05775076 0.05471125 0.03039514 0.08814590 0.09422492 0.02735562
##         13
## 0.12765957 0.13981763
##
## [[2]]
##        id week trip price.0 price.KB price.KR price.MB choice cluster
##    1:   1   96    1       0     1.43     1.43     1.43      0      11
##    2:   2   14    1       0     1.43     1.43     1.65      0       2
##    3:   2   25    2       0     1.43     1.43     1.65      0       2
##    4:   2   26    3       0     1.43     1.43     1.65      0       2
##    5:   2   31    4       0     1.43     0.88     1.65     KB       2
##   ---
## 1543: 358  124    8       0     1.43     1.43     1.35     MB      13
## 1544: 358  130    9       0     1.43     1.43     1.14     KB      13
## 1545: 358  140   10       0     1.47     1.43     1.11     KB      13
## 1546: 359   81    1       0     1.43     1.43     1.33      0      14
## 1547: 359   94    2       0     0.90     0.89     1.43      0      14
##
## [[3]]
##    segment intercept.KB intercept.KR intercept.MB price.coef
## 1        1     4.940191     4.797872     6.270547  -5.231605
## 2        2     2.607921     3.047239     1.878220  -2.771427
## 3        3     3.738379     3.875963     3.794366  -3.573145
## 4        4     3.087834     4.006679     3.180246  -2.826519
## 5        5     4.627368     4.291287     5.531816  -4.377422
## 6        6     7.095079     8.325625     7.494956  -6.299830
## 7        7     6.655636     6.515320     6.485815  -5.043962
## 8        8     7.706112     6.326294     7.724080  -6.450734
## 9        9    -4.566433    -3.395474    -4.709981   2.612757
## 10      10     8.553483     7.220431     7.734058  -6.173725
## 11      11     2.270866     3.114487     1.912202  -2.482877
## 12      12     8.021546     7.391168     7.894349  -6.583017
## 13      13     3.123304     4.039871     2.907815  -2.928321
## 14      14     5.117430     4.509340     4.544963  -4.062526
```

```
KmeansCluster(14) #6 change 0.08814590
```

```
## [[1]]
##          1          2          3          4          5          6
## 0.03039514 0.06079027 0.05471125 0.06686930 0.06990881 0.08814590
##          7          8          9         10         11         12
## 0.08206687 0.08814590 0.05775076 0.01215805 0.08814590 0.07598784
##         13         14
## 0.02127660 0.06382979 0.13981763
##
## [[2]]
##        id week trip price.0 price.KB price.KR price.MB choice cluster
##    1:   1   96    1       0     1.43     1.43     1.43      0       5
##    2:   2   14    1       0     1.43     1.43     1.65      0       2
##    3:   2   25    2       0     1.43     1.43     1.65      0       2
##    4:   2   26    3       0     1.43     1.43     1.65      0       2
##    5:   2   31    4       0     1.43     0.88     1.65     KB       2
##   ---
## 1543: 358  124    8       0     1.43     1.43     1.35     MB      14
## 1544: 358  130    9       0     1.43     1.43     1.14     KB      14
## 1545: 358  140   10       0     1.47     1.43     1.11     KB      14
## 1546: 359   81    1       0     1.43     1.43     1.33      0      15
## 1547: 359   94    2       0     0.90     0.89     1.43      0      15
##
## [[3]]
##    segment intercept.KB intercept.KR intercept.MB price.coef
## 1        1    -4.566433    -3.395474   -4.7099808   2.612757
## 2        2     3.194486     3.723818    2.7569793  -3.214032
## 3        3     7.706112     6.326294    7.7240796  -6.450734
## 4        4     3.087834     4.006679    3.1802458  -2.826519
## 5        5     2.356432     3.161706    2.0491407  -2.290334
## 6        6     8.553483     7.220431    7.7340581  -6.173725
## 7        7     4.627368     4.291287    5.5318158  -4.377422
## 8        8     3.738379     3.875963    3.7943658  -3.573145
## 9        9     6.655636     6.515320    6.4858148  -5.043962
## 10      10     7.095079     8.325625    7.4949559  -6.299830
## 11      11     4.072018     5.782559    4.4224158  -4.068411
## 12      12     4.940191     4.797872    6.2705466  -5.231605
## 13      13     7.932991     6.858227    7.6422339  -6.500594
## 14      14     1.361110     1.227695    0.5486508  -1.527036
## 15      15     5.117430     4.509340    4.5449628  -4.062526
```

```
KmeansCluster(15) #6 change 0.08814590
```

```
## [[1]]
##          1          2          3          4          5          6
## 0.08510638 0.05167173 0.10334347 0.03647416 0.07294833 0.06686930
##          7          8          9         10         11         12
## 0.08206687 0.07294833 0.01823708 0.01215805 0.05471125 0.04255319
##         13         14         15
## 0.08206687 0.02127660 0.05775076 0.13981763
##
## [[2]]
##        id week trip price.0 price.KB price.KR price.MB choice cluster
##    1:   1   96    1       0     1.43     1.43     1.43      0       5
##    2:   2   14    1       0     1.43     1.43     1.65      0       4
##    3:   2   25    2       0     1.43     1.43     1.65      0       4
##    4:   2   26    3       0     1.43     1.43     1.65      0       4
##    5:   2   31    4       0     1.43     0.88     1.65     KB       4
##   ---
## 1543: 358  124    8       0     1.43     1.43     1.35     MB      12
## 1544: 358  130    9       0     1.43     1.43     1.14     KB      12
## 1545: 358  140   10       0     1.47     1.43     1.11     KB      12
## 1546: 359   81    1       0     1.43     1.43     1.33      0      16
## 1547: 359   94    2       0     0.90     0.89     1.43      0      16
##
## [[3]]
##    segment intercept.KB intercept.KR intercept.MB   price.coef
## 1        1     4.333931    4.5887118     4.370333  -3.95076456
## 2        2    -0.559026   -0.0466396    -1.191989  -0.09772677
## 3        3     2.769528    4.5946323     3.195940  -3.16901113
## 4        4     2.637348    4.5553465     3.130577  -3.09441643
## 5        5     2.269350    3.0794199     1.960330  -2.27599941
## 6        6     3.087834    4.0066788     3.180246  -2.82651884
## 7        7     4.627368    4.2912871     5.531816  -4.37742192
## 8        8     4.912553    4.7703675     6.241938  -5.20294289
## 9        9     9.156504    7.3419082     8.842498  -7.40840244
## 10      10     7.095079    8.3256252     7.494956  -6.29982957
## 11      11     7.706112    6.3262944     7.724080  -6.45073356
## 12      12     4.822648    3.3741725     3.533280  -3.75155923
## 13      13     8.409000    7.3151364     7.649112  -6.08515911
## 14      14    21.450192   -0.5361554    19.147735 -16.23512591
## 15      15     6.655636    6.5153203     6.485815  -5.04396201
## 16      16     5.117430    4.5093403     4.544963  -4.06252619
```

```
###So, when centers=10, segment=11, after lauching KB, people's change from MB to KB
 ......
seg.share = KmeansCluster(10)[[1]]
coef.est = KmeansCluster(10)[[3]]
```

4.1 Please see our report.

4.2.1 what are the (product-level, aggregated across segments) own- and cross- elasticities among these products?

```
#Calculate elasticity
prSeg=function(priceKB,priceKR,priceMB) {
  Pr1=demandForAll(priceKB,priceKR,priceMB,as.numeric(coef.est[1,2:5]))
  Pr2=demandForAll(priceKB,priceKR,priceMB,as.numeric(coef.est[2,2:5]))
  Pr3=demandForAll(priceKB,priceKR,priceMB,as.numeric(coef.est[3,2:5]))
  Pr4=demandForAll(priceKB,priceKR,priceMB,as.numeric(coef.est[4,2:5]))
  Pr5=demandForAll(priceKB,priceKR,priceMB,as.numeric(coef.est[5,2:5]))
  Pr6=demandForAll(priceKB,priceKR,priceMB,as.numeric(coef.est[6,2:5]))
  Pr7=demandForAll(priceKB,priceKR,priceMB,as.numeric(coef.est[7,2:5]))
  Pr8=demandForAll(priceKB,priceKR,priceMB,as.numeric(coef.est[8,2:5]))
  Pr9=demandForAll(priceKB,priceKR,priceMB,as.numeric(coef.est[9,2:5]))
  Pr10=demandForAll(priceKB,priceKR,priceMB,as.numeric(coef.est[10,2:5]))
  Pr11=demandForAll(priceKB,priceKR,priceMB,as.numeric(coef.est[11,2:5]))
  return(list(Pr1,Pr2,Pr3,Pr4,Pr5,Pr6,Pr7,Pr8,Pr9,Pr10,Pr11))
}# each c(Pr_KB,Pr_KR,Pr_MB)

agg_choice=function(priceKB,priceKR,priceMB) {
  agg_choice=seg.share[1]*demandForAll(priceKB,priceKR,priceMB,as.numeric(coef.est[1,2:5
]))+
    seg.share[2]*demandForAll(priceKB,priceKR,priceMB,as.numeric(coef.est[2,2:5]))+
    seg.share[3]*demandForAll(priceKB,priceKR,priceMB,as.numeric(coef.est[3,2:5]))+
    seg.share[4]*demandForAll(priceKB,priceKR,priceMB,as.numeric(coef.est[4,2:5]))+
    seg.share[5]*demandForAll(priceKB,priceKR,priceMB,as.numeric(coef.est[5,2:5]))+
    seg.share[6]*demandForAll(priceKB,priceKR,priceMB,as.numeric(coef.est[6,2:5]))+
    seg.share[7]*demandForAll(priceKB,priceKR,priceMB,as.numeric(coef.est[7,2:5]))+
    seg.share[8]*demandForAll(priceKB,priceKR,priceMB,as.numeric(coef.est[8,2:5]))+
    seg.share[9]*demandForAll(priceKB,priceKR,priceMB,as.numeric(coef.est[9,2:5]))+
    seg.share[10]*demandForAll(priceKB,priceKR,priceMB,as.numeric(coef.est[10,2:5]))+
    seg.share[11]*demandForAll(priceKB,priceKR,priceMB,as.numeric(coef.est[11,2:5]))
  return(agg_choice)
}# aggregate c(Pr_KB,Pr_KR,Pr_MB) R,Pr_MB)
segPr=prSeg(meanPrice[1],meanPrice[2],meanPrice[3])
segPr <- data.frame(matrix(unlist(segPr), nrow=length(segPr), byrow=T))
segAggPr=agg_choice(meanPrice[1],meanPrice[2],meanPrice[3])

# own-price elasticities
segOwnElasticity=function(price,Pr,segShare,beta0,segPr){
  SegOwnElasticity=-price*sum(segShare*beta0*segPr*(1-segPr))/Pr
  return(SegOwnElasticity)
}
# cross-price elasticities j's price to i
segCrossElasticity=function(price,Pr,segShare,beta1,segPr,i,j){
  SegCrossElasticity=-price/Pr*sum(segShare*beta1*segPr[[i]]*segPr[[j]])
  return(SegCrossElasticity)
}

#built a matrix for both own- and cross- ealsticity
segElastMatrix<-data.frame(matrix(ncol = 3, nrow = 3))
colnames(segElastMatrix)<- c("KB",'KR','MB')
rownames(segElastMatrix)<- c("KB",'KR','MB')

for(i in 1:3){
  for(j in 1:3){#Products in column change prices and then influence products in rows
```

```
    if (rownames(segElastMatrix)[i]==colnames(segElastMatrix)[j]){
      segElastMatrix[i,j]=segOwnElasticity(meanPrice[i],segAggPr[i],seg.share,coef.est[,
5],segPr[[i]])
    }else{segElastMatrix[i,j]=segCrossElasticity(meanPrice[j],segAggPr[i],seg.share,coe
f.est[,5],segPr,i,j)}
  }
}
```

4.2.2 How does the underlying customer segmentation explain the substitution pattern you see in the elasticity? From the substitution pattern and underlying segmentation, where (i.e. which segment(s)) should Kiwi Bubbles be positioned?

Please see our report.

4.2.3

```
#If not lauching KB
#"newpara" is beta0KB,beta0KR,beta0MB,beta1
demandPrevious=function(priceKR,priceMB,newpara){
  probKR=exp(newpara[2]+newpara[4]*priceKR)/(1+exp(newpara[2]+newpara[4]*priceKR)+exp(ne
wpara[3]+newpara[4]*priceMB))
  probMB=exp(para[3]+para[4]*priceMB)/(1+exp(para[2]+para[4]*priceKR)+exp(para[3]+para[4
]*priceMB))
  return(cbind(probKR,probMB))
}

agg_choicePrevious=function(priceKR,priceMB) {
  agg_choice=seg.share[1]*demandPrevious(priceKR,priceMB,as.numeric(coef.est[1,2:5]))+
    seg.share[2]*demandPrevious(priceKR,priceMB,as.numeric(coef.est[2,2:5]))+
    seg.share[3]*demandPrevious(priceKR,priceMB,as.numeric(coef.est[3,2:5]))+
    seg.share[4]*demandPrevious(priceKR,priceMB,as.numeric(coef.est[4,2:5]))+
    seg.share[5]*demandPrevious(priceKR,priceMB,as.numeric(coef.est[5,2:5]))+
    seg.share[6]*demandPrevious(priceKR,priceMB,as.numeric(coef.est[6,2:5]))+
    seg.share[7]*demandPrevious(priceKR,priceMB,as.numeric(coef.est[7,2:5]))+
    seg.share[8]*demandPrevious(priceKR,priceMB,as.numeric(coef.est[8,2:5]))+
    seg.share[9]*demandPrevious(priceKR,priceMB,as.numeric(coef.est[9,2:5]))+
    seg.share[10]*demandPrevious(priceKR,priceMB,as.numeric(coef.est[10,2:5]))+
    seg.share[11]*demandPrevious(priceKR,priceMB,as.numeric(coef.est[11,2:5]))
  return(agg_choice)
}

uc=0.5
pricespaceForKR=seq(0.88,1.43,0.01)
profit_previous=1000*(agg_choicePrevious(pricespaceForKR,1.43)[,1])*(pricespaceForKR-uc)
max(profit_previous)
```

```
## [1] 285.592
```

```
priceKR_previous=pricespaceForKR[which.max(profit_previous)]
priceKR_previous
```

```
## [1] 1.06
```

```
#if we do not launch KB, optimal price is $1.06, best profit is 285.592

#If lauching KB
#The first element of "newpara" is beta0KB,beta0KR,beta0MB,beta1"
ProfitAfter=function(priceKB,priceKR,priceMB){
  profitKB=agg_choice(priceKB,priceKR,priceMB)[,1]*(priceKB-0.5)*1000
  profitKR=agg_choice(priceKB,priceKR,priceMB)[,2]*(priceKR-0.5)*1000
  profitMB=agg_choice(priceKB,priceKR,priceMB)[,3]*(priceMB-0.5)*1000
  return(cbind(profitKB,profitKR,profitMB))
}
#Choose space of prices to search for the optimal price over
aux=seq(0.88,1.47,0.01)
#Because we search over two dimensions, create complete combination
#of the two prices
pricespace=expand.grid(aux,aux)
#At each iteration of the loop, I take one realization of [P^KB,P^KR] pair and evaluate
#profit at that realization.
profitmat=matrix(nrow(pricespace),1)
for (i in 1:nrow(pricespace)){
    profitmat[i]=sum(ProfitAfter(pricespace[i,1],pricespace[i,2],1.43)[,c(1,2)])
}
priceKB_seg = pricespace[profitmat==max(profitmat),][,1];
priceKB_seg #1.13
```

```
## [1] 1.13
```

```
priceKR_seg = pricespace[profitmat==max(profitmat),][,2];
priceKR_seg #1.2
```

```
## [1] 1.2
```

```
profit_after = max(profitmat);
profit_after #395.6119
```

```
## [1] 395.6119
```

```
#MB change, before launch KB
profitMB_previous=1000*agg_choicePrevious(priceKR_previous,1.43)[,2]*(1.43-uc)
profitMB_previous #105.6955
```

```
## [1] 105.6955
```

```
profitMB_after=ProfitAfter(priceKB_seg,priceKR_seg,1.43)[,3]
profitMB_after #86.57259
```

```
## [1] 86.57259
```

5 Understanding strategic responses

```
# price war

#As Mango, I need to react to KB and KR's new prices.
KB1=priceKB_seg
KR1=priceKR_seg
uc=0.5
pricespace1=seq(0,2,0.01)
profit1=1000*agg_choice(KB1,KR1,pricespace1)[,3]*(pricespace1-uc)
max(profit1)
```

```
## [1] 183.0472
```

```
MB1=pricespace1[profit1==max(profit1)];
MB1 #0.95
```

```
## [1] 0.95
```

```
#As Kiwi, I need to react to MB's new price.
aux2=seq(0.8,2,0.01)
pricespace2=expand.grid(aux2,aux2)
profitmat=matrix(0L,nrow(pricespace2),1)
for (i in 1:nrow(pricespace2)){
    profitmat[i]=sum(ProfitAfter(pricespace2[i,1],pricespace2[i,2],MB1)[,1:2])
    }
KB2 = pricespace2[profitmat==max(profitmat),][,1];
KB2 #0.99
```

```
## [1] 0.99
```

```
KR2 = pricespace2[profitmat==max(profitmat),][,2];
KR2 #1.1
```

```
## [1] 1.1
```

```
#Then, as mango, I need to react to KB and KR's newer prices
profit=1000*agg_choice(KB2,KR2,pricespace1)[,3]*(pricespace1-uc)
MB2=pricespace1[profit==max(profit)];
MB2 #0.91
```

```
## [1] 0.91
```

```
#Then, as Kiwi, I need to react to MB's newer price.
profitmat=matrix(0L,nrow(pricespace2),1)
for (i in 1:nrow(pricespace2)){
    profitmat[i]=sum(ProfitAfter(pricespace2[i,1],pricespace2[i,2],MB2)[,c(1,2)])
    }

KB3 = pricespace2[profitmat==max(profitmat),][,1];KB3 #0.98
```

```
## [1] 0.98
```

```
KR3 = pricespace2[profitmat==max(profitmat),][,2];KR3 #1.09
```

```
## [1] 1.09
```

```
#Then,then, as mango, I need to react to KB and KR's newerer prices
profit=1000*agg_choice(KB3,KR3,pricespace1)[,3]*(pricespace1-uc)
MB4=pricespace1[profit==max(profit)];
MB4 #0.91
```

```
## [1] 0.91
```