Team Members

Aradhya Mathur, 32384567
Ayush Singla, 32280500
Pradhyumna Rao, 32281507
Richa Yadav, 32381047
Rishabh Kandoi, 32384079

2023-02-09

```
knitr::opts_chunk$set(echo = TRUE, tidy.opts = list(width.cutoff = 90), tidy = TRUE)
```

3. Logit model without segmentation

3.1) Estimate the model and report the estimates. What do those coefficients mean?

```
# Read data
data = read.csv("kiwi_bubbles_P2.csv", stringsAsFactors = F)

# Drop observations with stockout
data = data[!(data$price.KB == 99), ]
data = data[!(data$price.KR == 99), ]
data = data[!(data$price.MB == 99), ]

# 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: Fri Feb 10 22:56:03 2023
##
## 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 (gradtol)
```

```
coef = mle$coefficients
coef
```

```
## KB:(intercept) KR:(intercept) MB:(intercept)          price
##       4.253157       4.362403       4.204396      -3.737931
```

Ans: These coefficients suggest us the relationship between price and demand in the market. As seen above, the popularity of all three products are similar. Also, according to the price coefficient value, the market seems to be price sensitive since magnitude of beta1 coefficient is far away from zero.

3.2) Using the estimated parameters, calculate own- and cross-price elasticities for all combination of products, evaluated at the average prices observed in the data. Do you see any particular patterns? Are the patterns reasonable, or something you are concerned about? Explain why (why not).

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

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))
    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))
}

para = c(coef[1], coef[2], coef[3], coef[4])
demand_output = demand(mean(data$price.KB), mean(data$price.KR), mean(data$price.MB), para)
data$probKB = demand_output[, 1]
data$probKR = demand_output[, 2]
data$probMB = demand_output[, 3]
```

```r
elasticity_KB = coef[4] * mean(data$price.KB) * (1 - demand_output[, 1])
elasticity_KR = coef[4] * mean(data$price.KR) * (1 - demand_output[, 2])
elasticity_MB = coef[4] * mean(data$price.MB) * (1 - demand_output[, 3])

# Validate logic of multi-product cross elasticity
cross_elasticity_KB = -coef[4] * mean(data$price.KB) * (demand_output[, 1])
cross_elasticity_KR = -coef[4] * mean(data$price.KR) * (demand_output[, 2])
cross_elasticity_MB = -coef[4] * mean(data$price.MB) * (demand_output[, 3])

KB_KR = cross_elasticity_KR
KB_MB = cross_elasticity_MB
KR_MB = cross_elasticity_MB

KR_KB = cross_elasticity_KB
MB_KB = cross_elasticity_KB
MB_KR = cross_elasticity_KR

KB_KB = elasticity_KB
KR_KR = elasticity_KR
MB_MB = elasticity_MB

rownames = c("KB", "KR", "MB")
colnames = c("KB", "KR", "MB")

P <- matrix(c(KB_KB, KB_KR, KB_MB, KR_KB, KR_KR, KR_MB, MB_KB, MB_KR, MB_MB), nrow = 3, byrow = TRUE,
    dimnames = list(rownames, colnames))
print(P)
```

```
##            KB        KR         MB
## KB -4.2578474  1.019923  0.9601564
## KR  0.9054743 -4.131270  0.9601564
## MB  0.9054743  1.019923 -4.0695469
```

Ans: From the values of cross-price elasticities seen in the matrix above, the alues are very close to each other, and thus when there is one percent change in price of own product then percent changes of choice probablity among both rivals are same. Similarly, since own price elasticity values are also very close to each other, demand of all products reacts in same amount when there is one percent change in the price of each of the products.

3.3) Calculate optimal prices for KB and KR (note that you have two products) to maximize the profit when Mango price is PMB = 1.43.

```r
pricespace1 = seq(0.7, 3, 0.01)
pricespace2 = seq(0.7, 3, 0.01)
profit_matrix = matrix(, nrow = 231, ncol = 231)
pricespace1_ind = 1
pricespace2_ind = 0
max = 0
uc = 0.5
for (iter in pricespace1) {
    iter_row = c()
```

```
    for (iter2 in pricespace2) {
        profit_KB = (1000 * ((demand(iter, iter2, 1.43, para)[, 1]) * iter - demand(iter, iter2,
            1.43, para)[, 1] * uc))
        profit_KR = (1000 * ((demand(iter, iter2, 1.43, para)[, 2]) * iter2 - demand(iter,
            iter2, 1.43, para)[, 2] * uc))
        profit_total = profit_KB + profit_KR
        iter_row = append(iter_row, profit_total)

    }

    profit_matrix[pricespace1_ind, ] = iter_row
    pricespace1_ind = pricespace1_ind + 1
}

pricespace_ind = which(profit_matrix == max(profit_matrix), arr.ind = TRUE)
paste("Max profit is ", max(profit_matrix), " at price of KB=", pricespace1[pricespace_ind[1]],
    " and price of KR=", pricespace2[pricespace_ind[2]], sep = "")
```

```
## [1] "Max profit is 393.408204542004 at price of KB=1.16 and price of KR=1.16"
```

Ans: Max profit is 393.408204542004 at price of KB=1.16 and price of KR=1.16

4. Logit model with segmentation

4.1) Like what we did in the lecture, group consumers into segments using "kmeans" function. You may use all columns from the demographic data to do clustering. Then estimate multinomial logit model separately for each segment of consumers. I would recommend number of clusters larger than 7 to find meaningful segmentation (but not too large - if you have too many segments you'd start seeing a segment with like only two people in it. Stop adding segments before you get there). Report your estimates.

```
# 8 clusters

# Load demographic data
demo = read.csv("demo_P2.csv", stringsAsFactors = F)
# Clustering
demo_cluster = kmeans(x = demo[, 2:18], centers = 7, nstart = 1000)
demo_cluster
```

```
## K-means clustering with 7 clusters of sizes 47, 41, 53, 38, 39, 31, 34
##
## Cluster means:
##   fam_size    hh_occ   hh_edu   hh_age  fem_age fem_educ fem_occup fem_work
## 1 4.042553  1.702128 6.170213 3.000000 3.000000 6.170213  1.702128 2.659574
## 2 4.024390 13.000000 5.341463 3.121951 3.121951 5.341463 13.000000 3.731707
## 3 2.301887 10.679245 4.641509 4.509434 4.528302 4.735849 10.698113 3.528302
## 4 3.947368  2.184211 5.500000 2.921053 2.921053 5.500000  2.184211 2.763158
## 5 2.871795  2.333333 6.076923 3.641026 3.641026 6.076923  2.333333 2.820513
## 6 2.129032  2.064516 6.000000 3.516129 3.516129 6.000000  2.064516 2.870968
## 7 3.588235  7.029412 5.147059 3.235294 3.617647 5.411765  8.147059 3.382353
```

```
##   fem_smoke male_age male_educ male_occup male_work male_smoke      dogs
## 1 0.1276596 3.127660  6.106383   1.659574  2.851064 0.08510638 0.5957447
## 2 0.1707317 3.512195  5.585366   4.243902  2.853659 0.17073171 0.6829268
## 3 0.2264151 6.037736  7.018868  11.075472  5.339623 0.09433962 0.4528302
## 4 0.2631579 3.210526  5.105263   7.289474  3.000000 0.26315789 0.5789474
## 5 0.1538462 3.923077  5.717949   3.666667  2.897436 0.20512821 0.5128205
## 6 0.1935484 6.774194  8.645161  10.967742  6.516129 0.12903226 0.4838710
## 7 0.1176471 3.617647  5.382353   4.911765  2.882353 0.23529412 0.5882353
##   child_code       tv
## 1   3.021277 3.574468
## 2   5.121951 3.390244
## 3   7.000000 2.509434
## 4   3.447368 3.289474
## 5   7.820513 2.871795
## 6   7.129032 3.000000
## 7   5.323529 3.352941
##
## Clustering vector:
##   [1] 5 3 3 1 1 6 3 3 5 6 3 3 6 6 3 5 5 7 6 6 3 7 2 3 7 2 7 5 4 3 3 3 7 7 3 1 6
##  [38] 5 6 3 3 3 3 7 1 2 4 2 7 4 1 7 6 7 5 7 4 1 5 1 5 2 4 4 2 4 2 1 7 3 6 4 5 6
##  [75] 5 5 3 4 1 3 3 4 5 3 2 4 3 1 3 3 6 5 2 5 7 3 2 6 2 3 5 6 2 3 5 3 3 5 5 3 3
## [112] 4 3 2 6 5 5 1 3 3 3 6 7 2 3 5 2 2 2 1 7 1 3 7 6 3 3 5 5 4 1 5 7 7 1 1 5 4
## [149] 4 2 7 1 7 1 6 4 7 5 5 4 2 6 3 1 1 7 1 7 7 7 3 1 2 5 7 3 1 1 1 5 6 1 4 2 1
## [186] 5 1 1 2 3 2 4 6 4 6 2 1 4 7 1 1 1 3 1 7 2 1 1 2 2 4 3 4 4 1 2 2 7 2 6 7 7
## [223] 2 5 2 1 6 3 3 6 1 4 5 2 7 2 1 4 4 4 2 3 1 3 3 3 2 2 2 7 1 6 6 2 5 5 5 4 4
## [260] 6 6 4 2 1 6 7 4 4 4 5 4 1 4 5 1 4 5 2 3 4 6 1 1
##
## Within cluster sum of squares by cluster:
## [1]  650.2979 1167.7561 2102.7170  685.6316  783.1282  658.9677 1265.5588
##  (between_SS / total_SS =  70.7 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```r
# now combine cluster identity into the raw data
cluster_id = data.frame(id = demo$id)
cluster_id$cluster = demo_cluster$cluster
kmeans_data = merge(data, cluster_id, by = "id", all.x = T)

# for those who don't fit in any cluster, group them into one additional cluster
kmeans_data$cluster[is.na(kmeans_data$cluster)] = 8

# just store the coefficients (you can store many other things)
coef_est_8 = data.frame(segment = 1:8, intercept.KB = NA, intercept.KR = NA, intercept.MB = NA,
    price.coef = NA)
# Write a for-loop.
for (seg in 1:8) {
    # During each loop, pick subset of data of consumers from each segment.
    data.sub = subset(kmeans_data, cluster == seg)

    # Using that data, the rest remains the same.
    mlogitdata = mlogit.data(data.sub, id = "id", varying = 4:7, choice = "choice", shape = "wide")
```

```
    # Run MLE.
    mle = gmnl(choice ~ price, data = mlogitdata)
    mle
    # Store the outcome in the coef.est matrix.
    coef_est_8[seg, 2:5] = mle$coefficients
}
coef_est_8
```

```
##   segment intercept.KB intercept.KR intercept.MB price.coef
## 1       1     2.969402     3.867674     2.727610  -2.909001
## 2       2     3.868998     4.354056     4.052386  -3.502896
## 3       3     4.354360     4.692998     4.150623  -3.695810
## 4       4     7.606395     6.661934     7.477539  -5.897474
## 5       5     2.333681     3.112574     2.925201  -2.896447
## 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
```

```
# 9 clusters

# Load demographic data
demo = read.csv("demo_P2.csv", stringsAsFactors = F)
# Clustering
demo_cluster = kmeans(x = demo[, 2:18], centers = 8, nstart = 1000)
demo_cluster
```

```
## K-means clustering with 8 clusters of sizes 39, 31, 40, 34, 23, 38, 47, 31
##
## Cluster means:
##   fam_size    hh_occ   hh_edu   hh_age  fem_age fem_educ fem_occup fem_work
## 1 2.871795  2.333333 6.076923 3.641026 3.641026 6.076923  2.333333 2.820513
## 2 2.129032  2.064516 6.000000 3.516129 3.516129 6.000000  2.064516 2.870968
## 3 4.050000 13.000000 5.375000 3.075000 3.075000 5.375000 13.000000 3.700000
## 4 3.588235  7.029412 5.147059 3.235294 3.617647 5.411765  8.147059 3.382353
## 5 2.869565 10.913043 4.478261 4.695652 4.739130 4.695652 10.956522 3.565217
## 6 3.947368  2.184211 5.500000 2.921053 2.921053 5.500000  2.184211 2.763158
## 7 4.042553  1.702128 6.170213 3.000000 3.000000 6.170213  1.702128 2.659574
## 8 1.903226 10.580645 4.741935 4.387097 4.387097 4.741935 10.580645 3.548387
##   fem_smoke male_age male_educ male_occup male_work male_smoke      dogs
## 1 0.1538462 3.923077  5.717949   3.666667  2.897436 0.20512821 0.5128205
## 2 0.1935484 6.774194  8.645161  10.967742  6.516129 0.12903226 0.4838710
## 3 0.1750000 3.475000  5.650000   4.150000  2.875000 0.17500000 0.6750000
## 4 0.1176471 3.617647  5.382353   4.911765  2.882353 0.23529412 0.5882353
## 5 0.3043478 4.913043  4.347826  11.086957  3.434783 0.21739130 0.5652174
## 6 0.2631579 3.210526  5.105263   7.289474  3.000000 0.26315789 0.5789474
## 7 0.1276596 3.127660  6.106383   1.659574  2.851064 0.08510638 0.5957447
## 8 0.1612903 6.838710  8.870968  10.967742  6.645161 0.00000000 0.3870968
##   child_code       tv
## 1   7.820513 2.871795
## 2   7.129032 3.000000
## 3   5.050000 3.450000
## 4   5.323529 3.352941
```

```
## 5   6.956522 2.869565
## 6   3.447368 3.289474
## 7   3.021277 3.574468
## 8   7.064516 2.193548
##
## Clustering vector:
##   [1] 1 8 5 7 7 2 5 8 1 2 8 8 2 2 5 1 1 4 2 2 5 4 3 8 4 3 4 1 6 8 5 8 4 4 5 7 2
##  [38] 1 2 8 8 8 5 4 7 3 6 3 4 6 7 4 2 4 1 4 6 7 1 7 1 3 6 6 3 6 3 7 4 8 2 6 1 2
##  [75] 1 1 5 6 7 8 5 6 1 8 3 6 5 7 8 5 2 1 3 1 4 8 5 2 3 5 1 2 3 8 1 8 5 1 1 8 5
## [112] 6 8 3 2 1 1 7 5 5 5 2 4 3 8 1 3 3 3 7 4 7 5 4 2 5 8 1 1 6 7 1 4 4 7 7 1 6
## [149] 6 3 4 7 4 7 2 6 4 1 1 6 3 2 8 7 7 4 7 4 4 4 8 7 3 1 4 8 7 7 7 1 2 7 6 3 7
## [186] 1 7 7 3 8 3 6 2 6 2 3 7 6 4 7 7 7 5 7 4 3 7 7 3 3 6 8 6 6 7 3 3 4 3 2 4 4
## [223] 3 1 3 7 2 5 8 2 7 6 1 3 4 3 7 6 6 6 3 8 7 5 8 8 3 3 3 4 7 2 2 3 1 1 1 6 6
## [260] 2 2 6 3 7 2 4 6 6 6 1 6 7 6 1 7 6 1 3 8 6 2 7 7
##
## Within cluster sum of squares by cluster:
## [1]  783.1282  658.9677 1115.2750 1265.5588  819.4783  685.6316  650.2979
## [8]  856.0645
##  (between_SS / total_SS =  72.6 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```r
# now combine cluster identity into the raw data
cluster_id = data.frame(id = demo$id)
cluster_id$cluster = demo_cluster$cluster
kmeans_data = merge(data, cluster_id, by = "id", all.x = T)

# for those who don't fit in any cluster, group them into one additional cluster
kmeans_data$cluster[is.na(kmeans_data$cluster)] = 9

# just store the coefficients (you can store many other things)
coef_est_9 = data.frame(segment = 1:9, intercept.KB = NA, intercept.KR = NA, intercept.MB = NA,
    price.coef = NA)
# Write a for-loop.
for (seg in 1:9) {
    # During each loop, pick subset of data of consumers from each segment.
    data.sub = subset(kmeans_data, cluster == seg)

    # Using that data, the rest remains the same.
    mlogitdata = mlogit.data(data.sub, id = "id", varying = 4:7, choice = "choice", shape = "wide")

    # Run MLE.
    mle = gmnl(choice ~ price, data = mlogitdata)
    mle
    # Store the outcome in the coef.est matrix.
    coef_est_9[seg, 2:5] = mle$coefficients
}
coef_est_9
```

```
##   segment intercept.KB intercept.KR intercept.MB price.coef
## 1       1    2.3336806     3.112574    2.9252012  -2.896447
```

```
## 2       2    3.9972969     3.958938    3.8837551  -3.715366
## 3       3    3.8689983     4.354056    4.0523865  -3.502896
## 4       4    4.8086045     4.606528    5.6294806  -4.517302
## 5       5    7.3034174     7.138563    7.1181389  -5.793619
## 6       6    7.6063946     6.661934    7.4775392  -5.897474
## 7       7    2.9694016     3.867674    2.7276100  -2.909001
## 8       8    0.9169255     1.673183    0.4573439  -1.251711
## 9       9    5.1174301     4.509340    4.5449628  -4.062526
```

```
# 11 clusters

# Load demographic data
demo = read.csv("demo_P2.csv", stringsAsFactors = F)
# Clustering
demo_cluster = kmeans(x = demo[, 2:18], centers = 10, nstart = 1000)
demo_cluster
```

```
## K-means clustering with 10 clusters of sizes 38, 19, 24, 31, 29, 48, 4, 22, 29, 39
##
## Cluster means:
##    fam_size    hh_occ   hh_edu   hh_age  fem_age fem_educ fem_occup fem_work
## 1  3.947368  2.184211 5.500000 2.921053 2.921053 5.500000  2.184211 2.763158
## 2  2.000000 10.473684 4.631579 5.684211 5.736842 4.894737 10.526316 3.947368
## 3  4.166667 13.000000 5.000000 2.750000 2.750000 5.000000 13.000000 3.583333
## 4  2.129032  2.064516 6.000000 3.516129 3.516129 6.000000  2.064516 2.870968
## 5  4.000000  7.620690 4.931034 3.000000 3.000000 4.931034  7.620690 2.793103
## 6  4.020833  1.791667 6.145833 3.020833 3.020833 6.145833  1.791667 2.666667
## 7  2.250000  1.500000 6.750000 3.750000 7.000000 9.000000 11.000000 7.000000
## 8  3.954545 12.863636 5.590909 3.363636 3.363636 5.590909 12.863636 3.772727
## 9  1.931034 10.724138 4.620690 4.241379 4.241379 4.620690 10.724138 3.413793
## 10 2.871795  2.333333 6.076923 3.641026 3.641026 6.076923  2.333333 2.820513
##    fem_smoke male_age male_educ male_occup male_work male_smoke      dogs
## 1  0.2631579 3.210526  5.105263   7.289474  3.000000 0.26315789 0.5789474
## 2  0.2105263 5.842105  4.894737   9.368421  3.526316 0.15789474 0.3684211
## 3  0.2083333 3.166667  4.875000   8.125000  3.000000 0.29166667 0.8750000
## 4  0.1935484 6.774194  8.645161  10.967742  6.516129 0.12903226 0.4838710
## 5  0.1379310 3.448276  5.137931   6.206897  2.931034 0.20689655 0.6206897
## 6  0.1250000 3.145833  6.145833   1.645833  2.854167 0.10416667 0.6041667
## 7  0.0000000 3.750000  6.750000   1.500000  3.000000 0.25000000 0.7500000
## 8  0.2272727 3.772727  6.045455   2.045455  2.727273 0.09090909 0.4545455
## 9  0.1724138 6.827586  8.862069  11.137931  6.862069 0.00000000 0.4137931
## 10 0.1538462 3.923077  5.717949   3.666667  2.897436 0.20512821 0.5128205
##    child_code       tv
## 1    3.447368 3.289474
## 2    8.000000 2.315789
## 3    4.625000 3.125000
## 4    7.129032 3.000000
## 5    5.344828 3.379310
## 6    3.000000 3.562500
## 7    5.250000 3.750000
## 8    5.363636 3.909091
## 9    7.000000 2.241379
## 10   7.820513 2.871795
##
```

```
## Clustering vector:
##   [1] 10  9  2  6  6  4  2  9 10  4  9  9  4  4  3 10 10  5  4  4  2  6  8  9  5
##  [26]  8  5 10  1  9  2  9  5  5  2  6  4 10  4  2  2  9  2  2  6  8  1  3  5  1
##  [51]  6  7  4  5 10  5  1  6 10  6 10  8  1  1  8  1  3  6  5  9  4  1 10  4 10
##  [76] 10  3  1  6  9  2  1 10  9  8  1  5  6  9  3  4 10  8 10  5  9  2  4  3  3
## [101] 10  4  3  9 10  2  2 10 10  9  2  1  9  8  4 10 10  6  2  2  9  4  7  2  9
## [126] 10  8  3  3  6  5  6  2  5  4  2  9 10 10  1  6 10  8  7  6  6 10  1  1  8
## [151]  7  6  5  6  4  1  5 10 10  1  8  4  9  6  6  5  6  5  5  5  9  6  8 10  5
## [176]  9  6  6  6 10  4  6  1  3  6 10  6  6  3  9  8  1  4  1  4  8  6  1  5  6
## [201]  6  6  3  6  5  3  6  6  8  3  1  9  1  1  6  8  8  5  8  4  5  5  3 10  3
## [226]  6  4  3  9  4  6  1 10  3  5  8  6  1  1  1  3  9  6  5  9  9  8  3  3  5
## [251]  6  4  4  3 10 10 10  1  1  4  4  1  3  6  4  5  1  1  1 10  1  6  1 10  6
## [276]  1 10  8  9  1  4  6  6
##
## Within cluster sum of squares by cluster:
##  [1] 685.6316 396.1053 642.5417 658.9677 749.7931 699.3333  78.7500 575.0000
##  [9] 817.7241 783.1282
##  (between_SS / total_SS =  75.6 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```r
# now combine cluster identity into the raw data
cluster_id = data.frame(id = demo$id)
cluster_id$cluster = demo_cluster$cluster
kmeans_data = merge(data, cluster_id, by = "id", all.x = T)

# for those who don't fit in any cluster, group them into one additional cluster
kmeans_data$cluster[is.na(kmeans_data$cluster)] = 11

# just store the coefficients (you can store many other things)
coef_est_11 = data.frame(segment = 1:11, intercept.KB = NA, intercept.KR = NA, intercept.MB = NA,
    price.coef = NA)
# Write a for-loop.
for (seg in 1:11) {
    # During each loop, pick subset of data of consumers from each segment.
    data.sub = subset(kmeans_data, cluster == seg)

    # Using that data, the rest remains the same.
    mlogitdata = mlogit.data(data.sub, id = "id", varying = 4:7, choice = "choice", shape = "wide")

    # Run MLE.
    mle = gmnl(choice ~ price, data = mlogitdata)
    mle
    # Store the outcome in the coef.est matrix.
    coef_est_11[seg, 2:5] = mle$coefficients
}
coef_est_11
```

```
##    segment intercept.KB intercept.KR intercept.MB price.coef
## 1        1    7.6063946     6.661934    7.4775392  -5.897474
## 2        2    6.6556359     6.515320    6.4858148  -5.043962
```

```
## 3        3    8.2287472     6.680553    8.1223860  -6.788933
## 4        4    3.9972969     3.958938    3.8837551  -3.715366
## 5        5    4.7911334     4.456635    5.6901102  -4.606762
## 6        6    3.0017626     3.916981    2.7620620  -2.951180
## 7        7    7.0950794     8.325625    7.4949559  -6.299830
## 8        8    3.0878338     4.006679    3.1802458  -2.826519
## 9        9    0.8341597     1.673929    0.4294365  -1.233209
## 10      10    2.3336806     3.112574    2.9252012  -2.896447
## 11      11    5.1174301     4.509340    4.5449628  -4.062526
```

```
# 10 clusters

# Load demographic data
demo = read.csv("demo_P2.csv", stringsAsFactors = F)
# Clustering
demo_cluster = kmeans(x = demo[, 2:18], centers = 9, nstart = 1000)
demo_cluster
```

```
## K-means clustering with 9 clusters of sizes 39, 42, 20, 31, 29, 32, 4, 48, 38
##
## Cluster means:
##   fam_size    hh_occ   hh_edu   hh_age  fem_age fem_educ fem_occup fem_work
## 1 2.871795  2.333333 6.076923 3.641026 3.641026 6.076923  2.333333 2.820513
## 2 3.952381 12.857143 5.380952 3.190476 3.190476 5.380952 12.857143 3.714286
## 3 2.700000 11.500000 4.300000 4.750000 4.800000 4.550000 11.550000 3.700000
## 4 2.129032  2.064516 6.000000 3.516129 3.516129 6.000000  2.064516 2.870968
## 5 4.000000  7.620690 4.931034 3.000000 3.000000 4.931034  7.620690 2.793103
## 6 1.906250 10.468750 4.781250 4.437500 4.437500 4.781250 10.468750 3.531250
## 7 2.250000  1.500000 6.750000 3.750000 7.000000 9.000000 11.000000 7.000000
## 8 4.020833  1.791667 6.145833 3.020833 3.020833 6.145833  1.791667 2.666667
## 9 3.947368  2.184211 5.500000 2.921053 2.921053 5.500000  2.184211 2.763158
##   fem_smoke male_age male_educ male_occup male_work male_smoke      dogs
## 1 0.1538462 3.923077  5.717949   3.666667  2.897436  0.2051282 0.5128205
## 2 0.1666667 3.595238  5.642857   4.095238  2.833333  0.1666667 0.6428571
## 3 0.3000000 4.950000  4.100000  10.950000  3.450000  0.2000000 0.5500000
## 4 0.1935484 6.774194  8.645161  10.967742  6.516129  0.1290323 0.4838710
## 5 0.1379310 3.448276  5.137931   6.206897  2.931034  0.2068966 0.6206897
## 6 0.1875000 6.812500  8.781250  10.937500  6.562500  0.0312500 0.3750000
## 7 0.0000000 3.750000  6.750000   1.500000  3.000000  0.2500000 0.7500000
## 8 0.1250000 3.145833  6.145833   1.645833  2.854167  0.1041667 0.6041667
## 9 0.2631579 3.210526  5.105263   7.289474  3.000000  0.2631579 0.5789474
##   child_code       tv
## 1   7.820513 2.871795
## 2   5.190476 3.404762
## 3   6.950000 2.850000
## 4   7.129032 3.000000
## 5   5.344828 3.379310
## 6   7.093750 2.187500
## 7   5.250000 3.750000
## 8   3.000000 3.562500
## 9   3.447368 3.289474
##
## Clustering vector:
##   [1] 1 6 3 8 8 4 3 6 1 4 6 6 4 4 3 1 1 5 4 4 3 8 2 6 5 2 5 1 9 6 3 6 5 5 3 8 4
```

```
##  [38] 1 4 6 6 6 3 2 8 2 9 2 5 9 8 7 4 5 1 5 9 8 1 8 1 2 9 9 2 9 2 8 5 6 4 9 1 4
##  [75] 1 1 3 9 8 6 6 9 1 6 2 9 5 8 6 3 4 1 2 1 5 6 3 4 2 3 1 4 2 6 1 6 3 1 1 6 3
## [112] 9 6 2 4 1 1 8 3 3 3 4 7 2 6 1 2 2 2 8 5 8 3 5 4 3 6 1 1 9 8 1 2 7 8 8 1 9
## [149] 9 2 7 8 5 8 4 9 5 1 1 9 2 4 6 8 8 5 8 5 5 5 6 8 2 1 5 6 8 8 8 1 4 8 9 2 8
## [186] 1 8 8 2 6 2 9 4 9 4 2 8 9 5 8 8 8 3 8 5 2 8 8 2 2 9 6 9 9 8 2 2 5 2 4 5 5
## [223] 2 1 2 8 4 3 6 4 8 9 1 2 5 2 8 9 9 9 2 6 8 5 6 6 2 2 2 5 8 4 4 2 1 1 1 9 9
## [260] 4 4 9 2 8 4 5 9 9 9 1 9 8 9 1 8 9 1 2 6 9 4 8 8
##
## Within cluster sum of squares by cluster:
## [1]  783.1282 1220.6429  620.9500  658.9677  749.7931  908.3438   78.7500
## [8]  699.3333  685.6316
##  (between_SS / total_SS =  74.3 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```r
# now combine cluster identity into the raw data
cluster_id = data.frame(id = demo$id)
cluster_id$cluster = demo_cluster$cluster
kmeans_data = merge(data, cluster_id, by = "id", all.x = T)

# for those who don't fit in any cluster, group them into one additional cluster
kmeans_data$cluster[is.na(kmeans_data$cluster)] = 10

# just store the coefficients (you can store many other things)
coef_est_10 = data.frame(segment = 1:10, intercept.KB = NA, intercept.KR = NA, intercept.MB = NA,
    price.coef = NA)
# Write a for-loop.
for (seg in 1:10) {
    # During each loop, pick subset of data of consumers from each segment.
    data.sub = subset(kmeans_data, cluster == seg)

    # Using that data, the rest remains the same.
    mlogitdata = mlogit.data(data.sub, id = "id", varying = 4:7, choice = "choice", shape = "wide")

    # Run MLE.
    mle = gmnl(choice ~ price, data = mlogitdata)
    mle
    # Store the outcome in the coef.est matrix.
    coef_est_10[seg, 2:5] = mle$coefficients
}
coef_est_10
```

```
##    segment intercept.KB intercept.KR intercept.MB price.coef
## 1        1     2.333681     3.112574     2.925201  -2.896447
## 2        2     3.929391     4.369639     4.065201  -3.548877
## 3        3     7.064807     6.909717     6.782931  -5.494300
## 4        4     3.997297     3.958938     3.883755  -3.715366
## 5        5     4.791133     4.456635     5.690110  -4.606762
## 6        6     1.194957     1.952011     1.005984  -1.470361
## 7        7     7.095079     8.325625     7.494956  -6.299830
## 8        8     3.001763     3.916981     2.762062  -2.951180
```

```
## 9        9     7.606395     6.661934     7.477539  -5.897474
## 10      10     5.117430     4.509340     4.544963  -4.062526
```

Ans: So after trying multiple number of clusters, we came to the conclusion that segmentation with 10 clusters (including the remaining cluster after kmeans) divides the data upto 4 consumers in a segment, and thus we would choose not to move further with more number of segments.

Below are the segment share thus found, for 10 clusters

```
# segment share
N = 359
seg.share = c(demo_cluster$size, N - sum(demo_cluster$size))/N
seg.share
```

```
##  [1] 0.10863510 0.11699164 0.05571031 0.08635097 0.08077994 0.08913649
##  [7] 0.01114206 0.13370474 0.10584958 0.21169916
```

4.2) At the average prices observed in the data, what are the market-level (aggregated across segments) own- and cross- elasticities among these products? How does it differ from the no-segmentation case? Which products are closer substitutes and which products are not as close substitutes?

Ans: For this, we would go with formula based apprach, as shown below.

```
# finding own-elasticities finding elasticity of KB
value_temp = 0
value_temp_prob = 0
for (i in (1:length(seg.share))) {
    w = seg.share[i]
    beta_1 = coef_est_10[i, 5]
    prob_temp = demand(mean(kmeans_data[kmeans_data$cluster == i, ]$price.KB), mean(kmeans_data[kmeans_
        i, ]$price.KR), mean(kmeans_data[kmeans_data$cluster == i, ]$price.MB), coef_est_10[i,
        2:5])[, 1]
    value_temp = value_temp + (w * beta_1 * prob_temp * (1 - prob_temp))
    value_temp_prob = value_temp_prob + (w * prob_temp)
}
KB_elasticity = (mean(kmeans_data$price.KB) * value_temp)/value_temp_prob

# finding elasticity of KR
value_temp = 0
value_temp_prob = 0
for (i in (1:length(seg.share))) {
    w = seg.share[i]
    beta_1 = coef_est_10[i, 5]
    prob_temp = demand(mean(kmeans_data[kmeans_data$cluster == i, ]$price.KB), mean(kmeans_data[kmeans_
        i, ]$price.KR), mean(kmeans_data[kmeans_data$cluster == i, ]$price.MB), coef_est_10[i,
        2:5])[, 2]
    value_temp = value_temp + (w * beta_1 * prob_temp * (1 - prob_temp))
    value_temp_prob = value_temp_prob + (w * prob_temp)
}
```

```
KR_elasticity = (mean(kmeans_data$price.KR) * value_temp)/value_temp_prob

# finding elasticity of MB
value_temp = 0
value_temp_prob = 0
for (i in (1:length(seg.share))) {
    w = seg.share[i]
    beta_1 = coef_est_10[i, 5]
    prob_temp = demand(mean(kmeans_data[kmeans_data$cluster == i, ]$price.KB), mean(kmeans_data[kmeans_
        i, ]$price.KR), mean(kmeans_data[kmeans_data$cluster == i, ]$price.MB), coef_est_10[i,
        2:5])[, 3]
    value_temp = value_temp + (w * beta_1 * prob_temp * (1 - prob_temp))
    value_temp_prob = value_temp_prob + (w * prob_temp)
}
MB_elasticity = (mean(kmeans_data$price.MB) * value_temp)/value_temp_prob
```

```
# Cross Price elasticity of KB-KR

value_temp = 0
value_temp_prob = 0
for (i in (1:length(seg.share))) {
    w = seg.share[i]
    beta_1 = coef_est_10[i, 5]
    prob_temp_KB = demand(mean(kmeans_data[kmeans_data$cluster == i, ]$price.KB), mean(kmeans_data[kmean
        i, ]$price.KR), mean(kmeans_data[kmeans_data$cluster == i, ]$price.MB), coef_est_10[i,
        2:5])[, 1]
    prob_temp_KR = demand(mean(kmeans_data[kmeans_data$cluster == i, ]$price.KB), mean(kmeans_data[kmean
        i, ]$price.KR), mean(kmeans_data[kmeans_data$cluster == i, ]$price.MB), coef_est_10[i,
        2:5])[, 2]
    value_temp = value_temp + (w * beta_1 * prob_temp_KB * prob_temp_KR)
    value_temp_prob = value_temp_prob + (w * prob_temp_KB)
}
KB_KR_elasticity = -(mean(kmeans_data$price.KR) * value_temp)/value_temp_prob

# Cross Price elasticity of KB-MB

value_temp = 0
value_temp_prob = 0
for (i in (1:length(seg.share))) {
    w = seg.share[i]
    beta_1 = coef_est_10[i, 5]
    prob_temp_KB = demand(mean(kmeans_data[kmeans_data$cluster == i, ]$price.KB), mean(kmeans_data[kmean
        i, ]$price.KR), mean(kmeans_data[kmeans_data$cluster == i, ]$price.MB), coef_est_10[i,
        2:5])[, 1]
    prob_temp_MB = demand(mean(kmeans_data[kmeans_data$cluster == i, ]$price.KB), mean(kmeans_data[kmean
        i, ]$price.KR), mean(kmeans_data[kmeans_data$cluster == i, ]$price.MB), coef_est_10[i,
        2:5])[, 3]
    value_temp = value_temp + (w * beta_1 * prob_temp_KB * prob_temp_MB)
    value_temp_prob = value_temp_prob + (w * prob_temp_KB)
}
KB_MB_elasticity = -(mean(kmeans_data$price.MB) * value_temp)/value_temp_prob

# Cross Price elasticity of KR-KB
```

```
value_temp = 0
value_temp_prob = 0
for (i in (1:length(seg.share))) {
    w = seg.share[i]
    beta_1 = coef_est_10[i, 5]
    prob_temp_KR = demand(mean(kmeans_data[kmeans_data$cluster == i, ]$price.KB), mean(kmeans_data[kmean
        i, ]$price.KR), mean(kmeans_data[kmeans_data$cluster == i, ]$price.MB), coef_est_10[i,
        2:5])[, 2]
    prob_temp_KB = demand(mean(kmeans_data[kmeans_data$cluster == i, ]$price.KB), mean(kmeans_data[kmean
        i, ]$price.KR), mean(kmeans_data[kmeans_data$cluster == i, ]$price.MB), coef_est_10[i,
        2:5])[, 1]
    value_temp = value_temp + (w * beta_1 * prob_temp_KR * prob_temp_KB)
    value_temp_prob = value_temp_prob + (w * prob_temp_KR)
}
KR_KB_elasticity = -(mean(kmeans_data$price.KB) * value_temp)/value_temp_prob

# Cross Price elasticity of KR-MB

value_temp = 0
value_temp_prob = 0
for (i in (1:length(seg.share))) {
    w = seg.share[i]
    beta_1 = coef_est_10[i, 5]
    prob_temp_KR = demand(mean(kmeans_data[kmeans_data$cluster == i, ]$price.KB), mean(kmeans_data[kmean
        i, ]$price.KR), mean(kmeans_data[kmeans_data$cluster == i, ]$price.MB), coef_est_10[i,
        2:5])[, 2]
    prob_temp_MB = demand(mean(kmeans_data[kmeans_data$cluster == i, ]$price.KB), mean(kmeans_data[kmean
        i, ]$price.KR), mean(kmeans_data[kmeans_data$cluster == i, ]$price.MB), coef_est_10[i,
        2:5])[, 3]
    value_temp = value_temp + (w * beta_1 * prob_temp_KR * prob_temp_MB)
    value_temp_prob = value_temp_prob + (w * prob_temp_KR)
}
KR_MB_elasticity = -(mean(kmeans_data$price.MB) * value_temp)/value_temp_prob

# Cross Price elasticity of MB-KB

value_temp = 0
value_temp_prob = 0
for (i in (1:length(seg.share))) {
    w = seg.share[i]
    beta_1 = coef_est_10[i, 5]
    prob_temp_MB = demand(mean(kmeans_data[kmeans_data$cluster == i, ]$price.KB), mean(kmeans_data[kmean
        i, ]$price.KR), mean(kmeans_data[kmeans_data$cluster == i, ]$price.MB), coef_est_10[i,
        2:5])[, 3]
    prob_temp_KB = demand(mean(kmeans_data[kmeans_data$cluster == i, ]$price.KB), mean(kmeans_data[kmean
        i, ]$price.KR), mean(kmeans_data[kmeans_data$cluster == i, ]$price.MB), coef_est_10[i,
        2:5])[, 1]
    value_temp = value_temp + (w * beta_1 * prob_temp_MB * prob_temp_KB)
    value_temp_prob = value_temp_prob + (w * prob_temp_MB)
}
MB_KB_elasticity = -(mean(kmeans_data$price.KB) * value_temp)/value_temp_prob

# Cross Price elasticity of MB-KR
```

```
value_temp = 0
value_temp_prob = 0
for (i in (1:length(seg.share))) {
    w = seg.share[i]
    beta_1 = coef_est_10[i, 5]
    prob_temp_MB = demand(mean(kmeans_data[kmeans_data$cluster == i, ]$price.KB), mean(kmeans_data[kmean
        i, ]$price.KR), mean(kmeans_data[kmeans_data$cluster == i, ]$price.MB), coef_est_10[i,
        2:5])[, 3]
    prob_temp_KR = demand(mean(kmeans_data[kmeans_data$cluster == i, ]$price.KB), mean(kmeans_data[kmean
        i, ]$price.KR), mean(kmeans_data[kmeans_data$cluster == i, ]$price.MB), coef_est_10[i,
        2:5])[, 2]
    value_temp = value_temp + (w * beta_1 * prob_temp_MB * prob_temp_KR)
    value_temp_prob = value_temp_prob + (w * prob_temp_MB)
}
MB_KR_elasticity = -(mean(kmeans_data$price.KR) * value_temp)/value_temp_prob


KB_KB_elasticity = KB_elasticity
KR_KR_elasticity = KR_elasticity
MB_MB_elasticity = MB_elasticity

rownames = c("KB", "KR", "MB")
colnames = c("KB", "KR", "MB")

P <- matrix(c(KB_KB_elasticity, KB_KR_elasticity, KB_MB_elasticity, KR_KB_elasticity, KR_KR_elasticity,
    KR_MB_elasticity, MB_KB_elasticity, MB_KR_elasticity, MB_MB_elasticity), nrow = 3, byrow = TRUE,
    dimnames = list(rownames, colnames))
print(P)
```

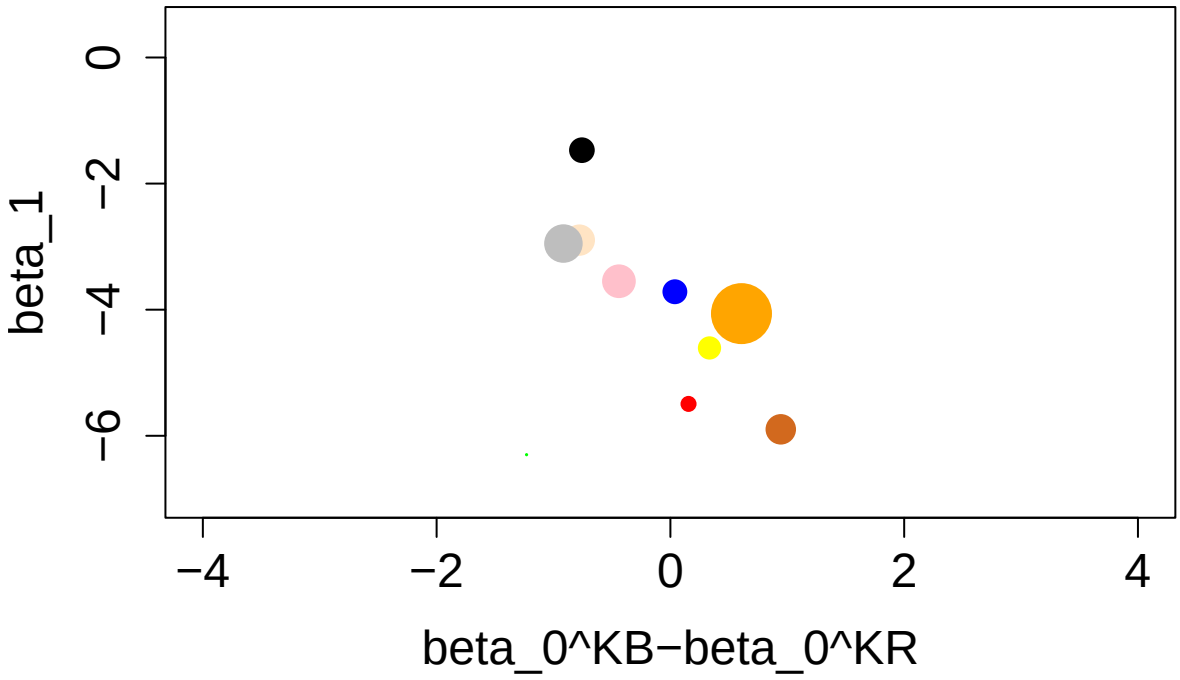
```
##            KB         KR         MB
## KB -4.3946517  0.9435617  1.0548101
## KR  0.8199515 -3.6587531  0.8236847
## MB  1.0161712  0.9131366 -4.2493285
```

```
# In comparison with the no segmentation case, price sensitivity is getting reduced when
# introducing segmentation Confirm about closer substitute being inferred using
# cross-price elasticity among combinations in matrix only?
```

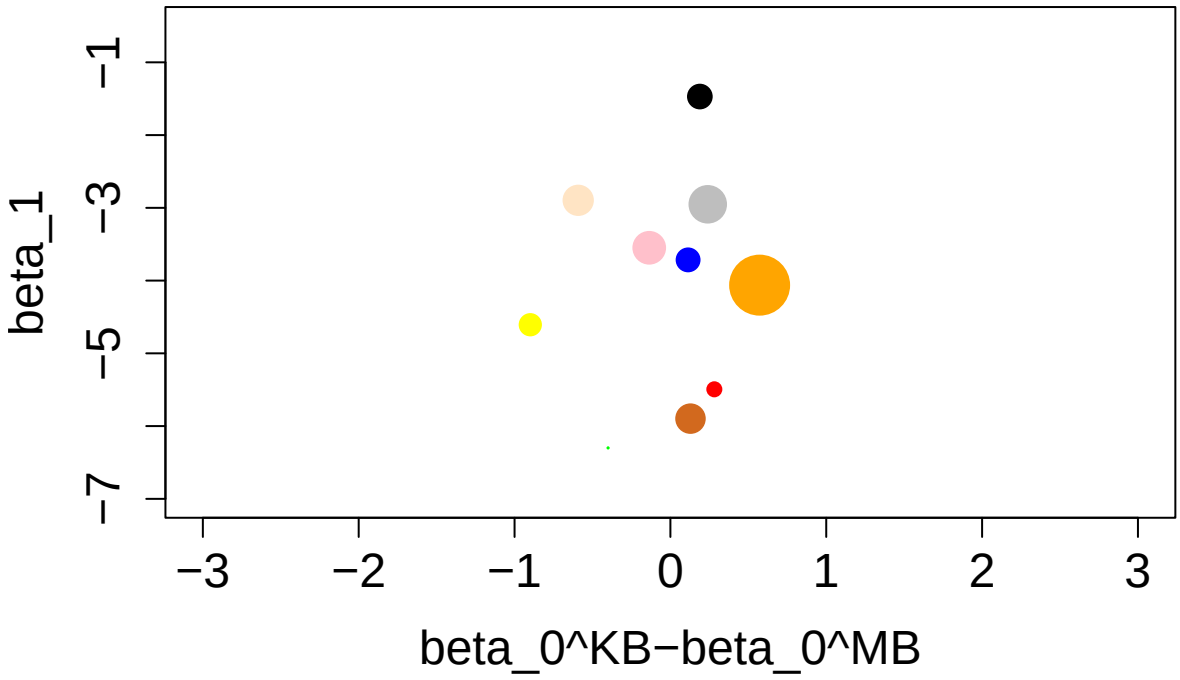
Ans: So as seen above, Own-price elasticity of KB and MB is larger while that of KR is smaller. Cross-price elasticity between KB and KR is a lot smaller, while that of KB and MG is larger. By doing segmentation, we can clearly see the substitution effect among products. After segmentation, the average cross elasticity between KB and MB is the largest, which means that they are closer substitutes. The cross elasticity between MB and KR is the smallest, so they are not as close substitutes. Therefore, launching KB has a strong impact on our competitor, and the cannibalization effect of launching KB on KR is not a big issue since the cross price elasticity between KB and KR is smaller than 1.

4.3) Discuss the preference of the segments you found (who prefers KB over KR, who is more price sensitive, etc.). How does the underlying customer segmentation explain the substitution pattern you see in the elasticity? From the elasticity and underlying segmentation you found, where (i.e. which segment(s)) do you suggest that Kiwi Bubbles should be positioned?

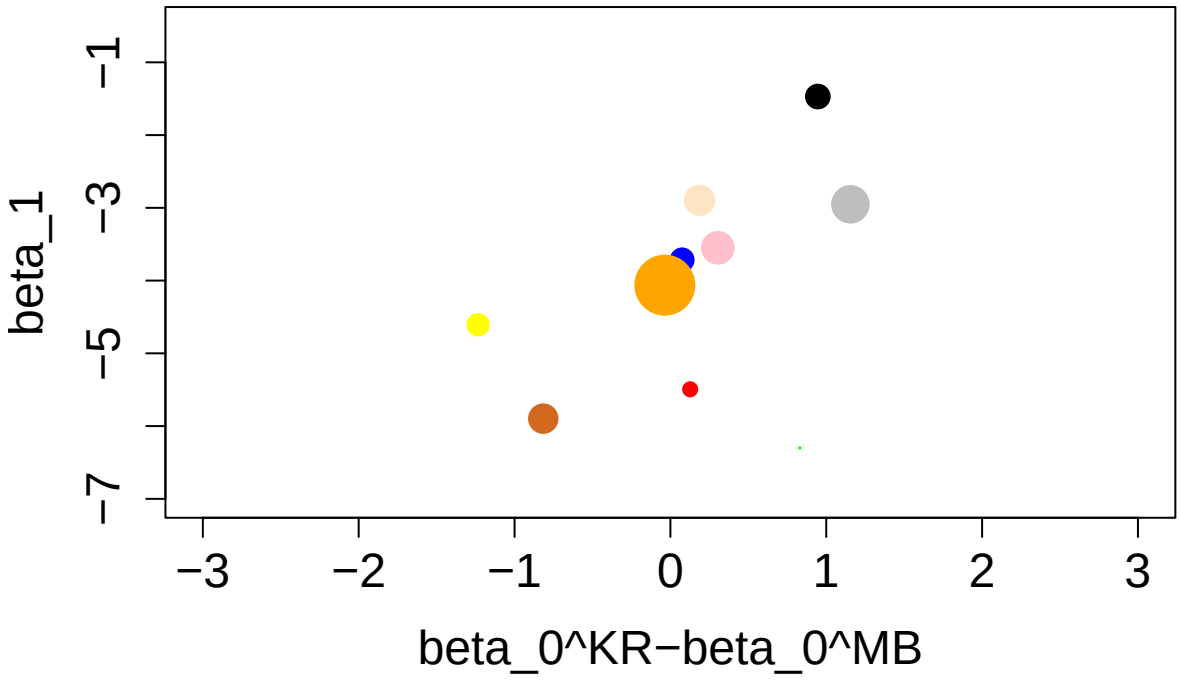
```r
# KB-KR
coef.est = coef_est_10
plot(coef.est[1, 2] - coef.est[1, 3], coef.est[1, 5], cex = 20 * seg.share[1], xlim = c(-4,
    4), ylim = c(-7, 0.5), col = "bisque", pch = 16, cex.lab = 1.5, cex.axis = 1.5, cex.main = 1.5,
    cex.sub = 1.5, xlab = "beta_0^KB-beta_0^KR", ylab = ("beta_1"))
points(coef.est[2, 2] - coef.est[2, 3], coef.est[2, 5], cex = 20 * seg.share[2], col = "pink",
    pch = 16)
points(coef.est[3, 2] - coef.est[3, 3], coef.est[3, 5], cex = 20 * seg.share[3], col = "red",
    pch = 16)
points(coef.est[4, 2] - coef.est[4, 3], coef.est[4, 5], cex = 20 * seg.share[4], col = "blue",
    pch = 16)
points(coef.est[5, 2] - coef.est[5, 3], coef.est[5, 5], cex = 20 * seg.share[5], col = "yellow",
    pch = 16)
points(coef.est[6, 2] - coef.est[6, 3], coef.est[6, 5], cex = 20 * seg.share[6], col = "black",
    pch = 16)
points(coef.est[7, 2] - coef.est[7, 3], coef.est[7, 5], cex = 20 * seg.share[7], col = "green",
    pch = 16)
points(coef.est[8, 2] - coef.est[8, 3], coef.est[8, 5], cex = 20 * seg.share[8], col = "grey",
    pch = 16)
points(coef.est[9, 2] - coef.est[9, 3], coef.est[9, 5], cex = 20 * seg.share[9], col = "chocolate",
    pch = 16)
points(coef.est[10, 2] - coef.est[10, 3], coef.est[10, 5], cex = 20 * seg.share[10], col = "orange",
    pch = 16)
```



```
# KB-MB
coef.est = coef_est_10
plot(coef.est[1, 2] - coef.est[1, 4], coef.est[1, 5], cex = 20 * seg.share[1], xlim = c(-3,
    3), ylim = c(-7, -0.5), col = "bisque", pch = 16, cex.lab = 1.5, cex.axis = 1.5, cex.main = 1.5,
    cex.sub = 1.5, xlab = "beta_0^KB-beta_0^MB", ylab = ("beta_1"))
points(coef.est[2, 2] - coef.est[2, 4], coef.est[2, 5], cex = 20 * seg.share[2], col = "pink",
    pch = 16)
points(coef.est[3, 2] - coef.est[3, 4], coef.est[3, 5], cex = 20 * seg.share[3], col = "red",
    pch = 16)
points(coef.est[4, 2] - coef.est[4, 4], coef.est[4, 5], cex = 20 * seg.share[4], col = "blue",
    pch = 16)
points(coef.est[5, 2] - coef.est[5, 4], coef.est[5, 5], cex = 20 * seg.share[5], col = "yellow",
    pch = 16)
points(coef.est[6, 2] - coef.est[6, 4], coef.est[6, 5], cex = 20 * seg.share[6], col = "black",
    pch = 16)
points(coef.est[7, 2] - coef.est[7, 4], coef.est[7, 5], cex = 20 * seg.share[7], col = "green",
    pch = 16)
points(coef.est[8, 2] - coef.est[8, 4], coef.est[8, 5], cex = 20 * seg.share[8], col = "grey",
    pch = 16)
points(coef.est[9, 2] - coef.est[9, 4], coef.est[9, 5], cex = 20 * seg.share[9], col = "chocolate",
    pch = 16)
points(coef.est[10, 2] - coef.est[10, 4], coef.est[10, 5], cex = 20 * seg.share[10], col = "orange",
    pch = 16)
```



```
# KR-MB
coef.est = coef_est_10
plot(coef.est[1, 3] - coef.est[1, 4], coef.est[1, 5], cex = 20 * seg.share[1], xlim = c(-3,
    3), ylim = c(-7, -0.5), col = "bisque", pch = 16, cex.lab = 1.5, cex.axis = 1.5, cex.main = 1.5,
    cex.sub = 1.5, xlab = "beta_0^KR-beta_0^MB", ylab = ("beta_1"))
points(coef.est[2, 3] - coef.est[2, 4], coef.est[2, 5], cex = 20 * seg.share[2], col = "pink",
    pch = 16)
points(coef.est[3, 3] - coef.est[3, 4], coef.est[3, 5], cex = 20 * seg.share[3], col = "red",
    pch = 16)
points(coef.est[4, 3] - coef.est[4, 4], coef.est[4, 5], cex = 20 * seg.share[4], col = "blue",
    pch = 16)
points(coef.est[5, 3] - coef.est[5, 4], coef.est[5, 5], cex = 20 * seg.share[5], col = "yellow",
    pch = 16)
points(coef.est[6, 3] - coef.est[6, 4], coef.est[6, 5], cex = 20 * seg.share[6], col = "black",
    pch = 16)
points(coef.est[7, 3] - coef.est[7, 4], coef.est[7, 5], cex = 20 * seg.share[7], col = "green",
    pch = 16)
points(coef.est[8, 3] - coef.est[8, 4], coef.est[8, 5], cex = 20 * seg.share[8], col = "grey",
    pch = 16)
points(coef.est[9, 3] - coef.est[9, 4], coef.est[9, 5], cex = 20 * seg.share[9], col = "chocolate",
    pch = 16)
points(coef.est[10, 3] - coef.est[10, 4], coef.est[10, 5], cex = 20 * seg.share[10], col = "orange",
    pch = 16)
```



```
# For KB, we should target segment 9 (chocolate colour in graph)
```

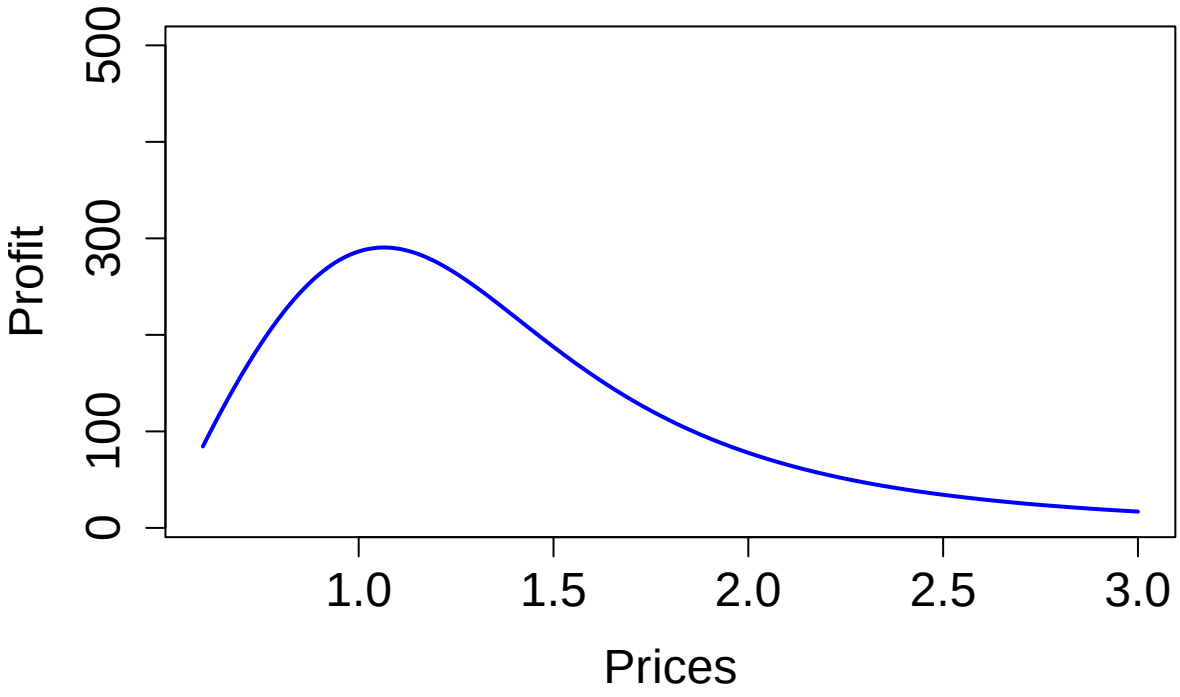
Ans: Inferences from the graphs above:
a) Among KB-KR, customers from segment 1 are loyal towards product KB, but are very price sensitive, which is visible from the other graphs since segment customers seems to have no preference for any product in other two graphs.
b) Consumers in segment 5,6 and 8 are very loyal towards product KR and are less price senitive as well.
c) Consumers from segment 3 and 4 seems to be very loyal towards product MB, but are quite price sensitive as well.
d) The biggest segment 10 shows slight preference for product KB.
e) Other segments, 2, 7 and 9 have consumers with no preference for any single product.

So, as observed from the inferences above, product KB, i,e, Kiwi Bubble should be positioned in the segment 1, 10. Segment 10 being the biggest as well, holds a good number of consumers to generate maximum profit.

\textbf{To justify your suggestion, compare profits with and without Kiwi Bubbles. Assume for now that Mango Bubbles is priced at $1.43 and does not react to Kiwi's pricing. First, assume that you decided not to launch Kiwi Bubbles. What is the optimal price of Kiwi Regular, and what is Kiwi and Mango's profit? Next, assume that you do launch Kiwi Bubbles. What are the optimal prices for Kiwi Regular and Kiwi Bubbles? How does the profit of Kiwi and Mango change as Kiwi launches KB? Does your result justify the launch of KB, and if so, does it support your positioning suggestions in the previous question?}

```
# Demand function without consideration of KB
demand_q4 = function(priceKR, priceMB, para) {
    probKR = exp(para[2] + para[4] * priceKR)/(1 + exp(para[2] + para[4] * priceKR) + exp(para[3] +
        para[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))
}

pricespace3 = seq(0.6, 3, 0.01)
profit_KR = 0
profit_MB = 0
for (i in (1:length(seg.share))) {
    w = seg.share[i]
    prob_temp_KR = demand_q4(pricespace3, 1.43, as.numeric(coef_est_10[i, 2:5]))[, 1]
    # prob_temp_MB = demand_q4(mean(kmeans_data[kmeans_data$cluster ==
    # i,]$price.KR),1.43,coef_est_10[i,2:5])[,2]
    num_customer = seg.share[i] * 1000
    profit_KR = profit_KR + num_customer * prob_temp_KR * (pricespace3 - uc)
    # profit_MB = profit_MB+ num_customer*prob_temp_MB * (1.43 - uc)
}
plot(pricespace3, profit_KR, type = "l", xlab = "Prices", ylab = "Profit", ylim = c(10, 500),
    col = "blue", lwd = 2, cex = 2, cex.lab = 1.5, cex.axis = 1.5, cex.main = 1.5, cex.sub = 1.5)
```




```
for (i in (1:length(seg.share))) {
    w = seg.share[i]
    prob_temp_MB = demand_q4(pricespace3[profit_KR == max(profit_KR)], 1.43, as.numeric(coef_est_10[i,
        2:5]))[, 2]
    num_customer = seg.share[i] * 1000
    profit_MB = profit_MB + num_customer * prob_temp_MB * (1.43 - uc)
}

paste("Profit of KR without KB is ", max(profit_KR), "at price ", pricespace3[profit_KR ==
    max(profit_KR)])
```

```
## [1] "Profit of KR without KB is  290.500080072366 at price  1.06"
```

```
paste("Profit of MB without KB is ", profit_MB, "at price ", 1.43)
```

```
## [1] "Profit of MB without KB is  106.532227061579 at price  1.43"
```

```
# finding demand considering KB too

pricespace3 = seq(0.6, 3, 0.01)
pricespace1 = seq(0.6, 3, 0.01)
pricespace2 = seq(0.6, 3, 0.01)
profit_KR = 0
profit_MB = 0
```

```
profit_KB = 0


profit_matrix = matrix(, nrow = length(pricespace1), ncol = length(pricespace2))
pricespace1_ind = 1
pricespace2_ind = 0

for (iter in pricespace1) {
    iter_row = c()
    for (iter2 in pricespace2) {
        profit_KR = 0
        profit_KB = 0
        for (i in (1:length(seg.share))) {
            w = seg.share[i]
            prob_temp_KB = demand(iter, iter2, 1.43, as.numeric(coef_est_10[i, 2:5]))[, 1]
            num_customer = seg.share[i] * 1000
            profit_KB = profit_KB + num_customer * prob_temp_KB * (iter - uc)
            # print('--') print(profit_KB)
            prob_temp_KR = demand(iter, iter2, 1.43, as.numeric(coef_est_10[i, 2:5]))[, 2]
            profit_KR = profit_KR + num_customer * prob_temp_KR * (iter2 - uc)
            # print('**') print(profit_KR)
        }
        profit_total = profit_KB + profit_KR
        # print('++') print(profit_total)
        iter_row = append(iter_row, profit_total)
        profit_total = 0
    }
    profit_matrix[pricespace1_ind, ] = iter_row
    pricespace1_ind = pricespace1_ind + 1
}
```

```
# finding actual optimal prices of KB and KR: KB is in rows; KR is in columns
max_profit_indices = which(profit_matrix == max(profit_matrix), arr.ind = TRUE)
(pricespace1[max_profit_indices[1]])
```

```
## [1] 1.15
```

```
(pricespace2[max_profit_indices[2]])
```

```
## [1] 1.19
```

```
max(profit_matrix)
```

```
## [1] 395.3726
```

```
paste("After considering KB, the optimal price of KB=", pricespace1[max_profit_indices[1]],
    " and optimal price for KR=", pricespace2[max_profit_indices[2]], " with total maximum profit=",
    max(profit_matrix), sep = "")
```

```
## [1] "After considering KB, the optimal price of KB=1.15 and optimal price for KR=1.19 with total maxi
```

```r
# to find the optimal price of MB, used the optimal prices of KB and KR and added the
# profit of MB for each segment
profit_MB = 0
profit_KR = 0
profit_KB = 0
prob_temp_MB = 0
prob_temp_KR = 0
prob_temp_KB = 0
for (i in (1:length(seg.share))) {
    w = seg.share[i]
    prob_temp_MB = demand(pricespace1[max_profit_indices[1]], pricespace2[max_profit_indices[2]],
        1.43, as.numeric(coef_est_10[i, 2:5]))[, 3]
    prob_temp_KR = demand(pricespace1[max_profit_indices[1]], pricespace2[max_profit_indices[2]],
        1.43, as.numeric(coef_est_10[i, 2:5]))[, 2]
    prob_temp_KB = demand(pricespace1[max_profit_indices[1]], pricespace2[max_profit_indices[2]],
        1.43, as.numeric(coef_est_10[i, 2:5]))[, 1]
    num_customer = seg.share[i] * 1000
    profit_MB = profit_MB + num_customer * prob_temp_MB * (1.43 - uc)
    profit_KR = profit_KR + num_customer * prob_temp_KR * (pricespace2[max_profit_indices[2]] -
        uc)
    profit_KB = profit_KB + num_customer * prob_temp_KB * (pricespace2[max_profit_indices[1]] -
        uc)
}

paste("Profit of KB is ", profit_KB, " at price ", pricespace2[max_profit_indices[1]])
```

```
## [1] "Profit of KB is  201.399928309403  at price  1.15"
```

```r
paste("Profit of KR with KB is ", profit_KR, "at price ", pricespace2[max_profit_indices[2]])
```

```
## [1] "Profit of KR with KB is  193.972639638362 at price  1.19"
```

```r
paste("Profit of MB with KB is ", profit_MB, "at price ", 1.43)
```

```
## [1] "Profit of MB with KB is  90.1612804634099 at price  1.43"
```

```r
paste("Profit of KB+KR is ", profit_KB + profit_KR)
```

```
## [1] "Profit of KB+KR is  395.372567947765"
```

```r
# So after the launch of KB, profit for both KR and MB reduces
```

Thus, launching of KB increases total profit of the company and reduces profit of MB.
If we do not launch kiwi bubble, 3 segments prefer Mango Bubble, and other segments prefer Kiwi Regular. After we launch Kiwi Bubble, because Kiwi Bubble and Mango Bubble are closer substitutes, only 1 segment would still be attracted by Mango Bubble, and we get customers in the other 2 segments to buy our newly launched Kiwi Bubble. Therefore, the market share of our products will increase when we launch Kiwi Bubble, resulting in a higher profit.

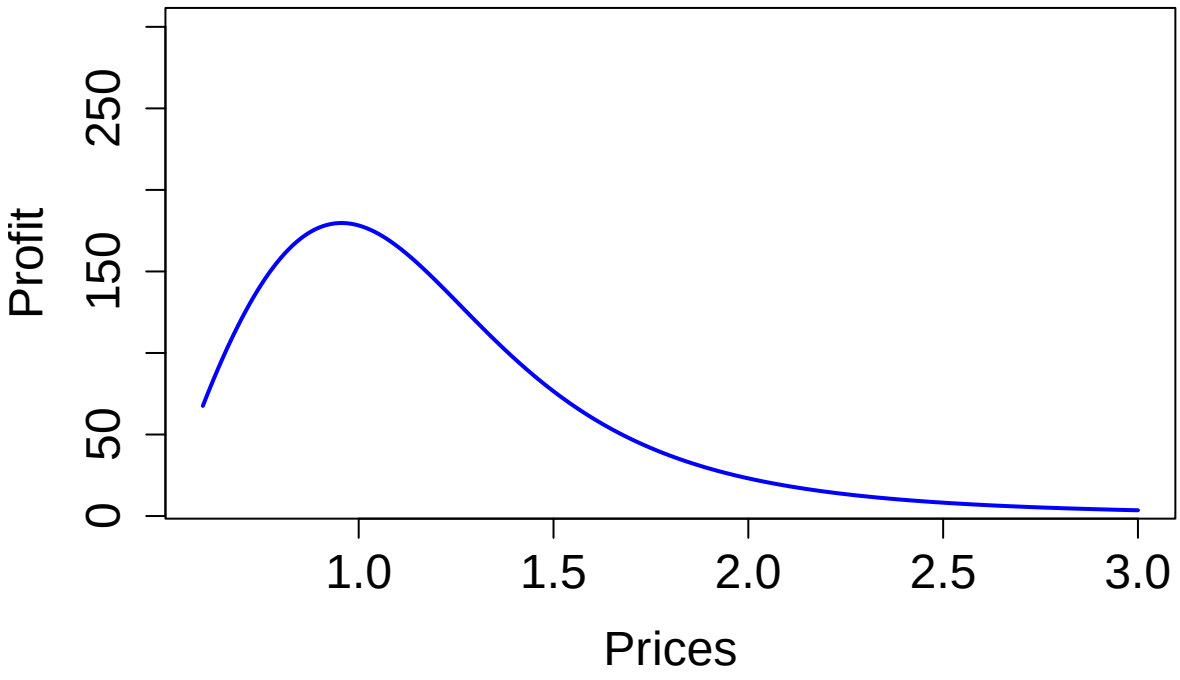
5. Understanding strategic responses

5.1) First, solve Mango's optimal pricing problem, given that Kiwi's price is the one you set from the previous section. What is the new price of MB?

```
# Question 5.1
pricespace4 = seq(0.6, 3, 0.01)
profit_MB_q5 = 0
for (i in (1:length(seg.share))) {
    w = seg.share[i]
    prob_demand_MB = demand(pricespace1[max_profit_indices[1]], pricespace2[max_profit_indices[2]],
        pricespace4, as.numeric(coef_est_10[i, 2:5]))[, 3]
    num_customer = seg.share[i] * 1000
    profit_MB_q5 = profit_MB_q5 + num_customer * prob_demand_MB * (pricespace4 - uc)
}
pricespace4[profit_MB_q5 == max(profit_MB_q5)]
```

```
## [1] 0.96
```

```
plot(pricespace4, profit_MB_q5, type = "l", xlab = "Prices", ylab = "Profit", ylim = c(10,
    300), col = "blue", lwd = 2, cex = 2, cex.lab = 1.5, cex.axis = 1.5, cex.main = 1.5, cex.sub = 1.5)
```



```
paste("Max Profit of MB=", max(profit_MB_q5), " at price=", pricespace4[profit_MB_q5 == max(profit_MB_q5
    sep = "")
```

```
## [1] "Max Profit of MB=179.690463808545 at price=0.96"
```

5.2) As Kiwi, you need to react to Mango's new price. Set prices for KR and KB to respond to the new price of Mango Bubble that we just derived. What is your new price for KR and KB?

```
# question 5.2: adjusting kr and kb prices according to the latest change in price of MB
pricespace3 = seq(0.5, 3, 0.01)
pricespace1 = seq(0.5, 3, 0.01)
pricespace2 = seq(0.5, 3, 0.01)
profit_KR = 0
profit_MB = 0
profit_KB = 0


profit_matrix = matrix(, nrow = length(pricespace1), ncol = length(pricespace2))
pricespace1_ind = 1
pricespace2_ind = 0

for (iter in pricespace1) {
    iter_row = c()
    for (iter2 in pricespace2) {
        profit_KR = 0
        profit_KB = 0
        for (i in (1:length(seg.share))) {
            w = seg.share[i]
            prob_temp_KB = demand(iter, mean(kmeans_data[kmeans_data$cluster == i, ]$price.KR),
                pricespace4[profit_MB_q5 == max(profit_MB_q5)], as.numeric(coef_est_10[i, 2:5]))[,
                1]
            num_customer = seg.share[i] * 1000
            profit_KB = profit_KB + num_customer * prob_temp_KB * (iter - uc)
            # print('--') print(profit_KB)
            prob_temp_KR = demand(mean(kmeans_data[kmeans_data$cluster == i, ]$price.KB), iter2,
                pricespace4[profit_MB_q5 == max(profit_MB_q5)], as.numeric(coef_est_10[i, 2:5]))[,
                2]
            profit_KR = profit_KR + num_customer * prob_temp_KR * (iter2 - uc)
            # print('**') print(profit_KR)
        }
        profit_total = profit_KB + profit_KR
        # print('++') print(profit_total)
        iter_row = append(iter_row, profit_total)
        profit_total = 0
    }
    profit_matrix[pricespace1_ind, ] = iter_row
    pricespace1_ind = pricespace1_ind + 1
}
```

```r
# finding actual optimal prices of KB and KR: KB is in rows; KR is in columns
max_profit_indices = which(profit_matrix == max(profit_matrix), arr.ind = TRUE)

paste("New price for KB=", (pricespace1[max_profit_indices[1]]), " and KR=", (pricespace1[max_profit_in
    " with maximum profit of ", max(profit_matrix), sep = "")
```

```
## [1] "New price for KB=0.94 and KR=0.96 with maximum profit of 344.07320102957"
```

5.3) Repeat the previous two steps iteratively, until neither Kiwi nor Mango has an incentive to set a different price (you can be as accurate as one cent, but no need to be more accurate than that). These set of prices are the new "equilibrium price" where you and Mango compete with each other. Please report the sequence of prices until you reach here.

```r
# Question 5.3

# After launch of KB

pricespace4 = seq(0.5, 3, 0.01)
profit_MB_q5 = 0
for (i in (1:length(seg.share))) {
    w = seg.share[i]
    prob_demand_MB = demand(pricespace1[max_profit_indices[1]], pricespace2[max_profit_indices[2]],
        pricespace4, as.numeric(coef_est_10[i, 2:5]))[, 3]
    num_customer = seg.share[i] * 1000
    profit_MB_q5 = profit_MB_q5 + num_customer * prob_demand_MB * (pricespace4 - uc)
}

paste("After launch of KB, with KB=", pricespace1[max_profit_indices[1]], " and KR=", pricespace1[max_p
    " profit of MB=", max(profit_MB_q5), " at price=", pricespace4[profit_MB_q5 == max(profit_MB_q5)])
```

```
## [1] "After launch of KB, with KB= 0.94  and KR= 0.96  profit of MB= 121.620985437181  at price= 0.89
```

```r
pricespace3 = seq(0.5, 3, 0.01)
pricespace1 = seq(0.5, 3, 0.01)
pricespace2 = seq(0.5, 3, 0.01)
profit_KR = 0
profit_MB = 0
profit_KB = 0


profit_matrix = matrix(, nrow = length(pricespace1), ncol = length(pricespace2))
pricespace1_ind = 1
pricespace2_ind = 0

for (iter in pricespace1) {
    iter_row = c()
    for (iter2 in pricespace2) {
        profit_KR = 0
        profit_KB = 0
        a = pricespace4[profit_MB_q5 == max(profit_MB_q5)]
        for (i in (1:length(seg.share))) {
```

```
            w = seg.share[i]
            prob_temp_KB = demand(iter, mean(kmeans_data[kmeans_data$cluster == i, ]$price.KR),
                a, as.numeric(coef_est_10[i, 2:5]))[, 1]
            num_customer = seg.share[i] * 1000
            profit_KB = profit_KB + num_customer * prob_temp_KB * (iter - uc)
            # print('--') print(profit_KB)
            prob_temp_KR = demand(mean(kmeans_data[kmeans_data$cluster == i, ]$price.KB), iter2,
                a, as.numeric(coef_est_10[i, 2:5]))[, 2]
            profit_KR = profit_KR + num_customer * prob_temp_KR * (iter2 - uc)
            # print('**') print(profit_KR)
        }
        profit_total = profit_KB + profit_KR
        # print('++') print(profit_total)
        iter_row = append(iter_row, profit_total)
        profit_total = 0
    }
    profit_matrix[pricespace1_ind, ] = iter_row
    pricespace1_ind = pricespace1_ind + 1
}

max_profit_indices = which(profit_matrix == max(profit_matrix), arr.ind = TRUE)

paste("After launch of KB and MB=", pricespace4[profit_MB_q5 == max(profit_MB_q5)], "total max profit="
    max(profit_matrix), " at KB=", pricespace1[max_profit_indices[1]], " and KR=", pricespace1[max_profi
```

```
## [1] "After launch of KB and MB= 0.89 total max profit= 311.557662174358  at KB= 0.92  and KR= 0.95"
```

```
# pricespace4[profit_MB_q5 == max(profit_MB_q5)]
# plot(pricespace4,profit_MB_q5,type='l',xlab='Prices',
# ylab='Profit',ylim=c(10,300),col='blue',lwd=2, cex=2,cex.lab=1.5, cex.axis=1.5,
# cex.main=1.5, cex.sub=1.5)
```

```
# iter 3

pricespace4 = seq(0.5, 3, 0.01)
profit_MB_q5 = 0
for (i in (1:length(seg.share))) {
    w = seg.share[i]
    prob_demand_MB = demand(pricespace1[max_profit_indices[1]], pricespace2[max_profit_indices[2]],
        pricespace4, as.numeric(coef_est_10[i, 2:5]))[, 3]
    num_customer = seg.share[i] * 1000
    profit_MB_q5 = profit_MB_q5 + num_customer * prob_demand_MB * (pricespace4 - uc)
}

paste("After launch of KB, with KB=", pricespace1[max_profit_indices[1]], " and KR=", pricespace1[max_pi
    " profit of MB=", max(profit_MB_q5), " at price=", pricespace4[profit_MB_q5 == max(profit_MB_q5)])
```

```
## [1] "After launch of KB, with KB= 0.92  and KR= 0.95  profit of MB= 117.601924814113  at price= 0.88
```

```
pricespace3 = seq(0.5, 3, 0.01)
pricespace1 = seq(0.5, 3, 0.01)
pricespace2 = seq(0.5, 3, 0.01)
```

```
profit_KR = 0
profit_MB = 0
profit_KB = 0


profit_matrix = matrix(, nrow = length(pricespace1), ncol = length(pricespace2))
pricespace1_ind = 1
pricespace2_ind = 0

for (iter in pricespace1) {
    iter_row = c()
    for (iter2 in pricespace2) {
        profit_KR = 0
        profit_KB = 0
        a = pricespace4[profit_MB_q5 == max(profit_MB_q5)]
        for (i in (1:length(seg.share))) {
            w = seg.share[i]
            prob_temp_KB = demand(iter, mean(kmeans_data[kmeans_data$cluster == i, ]$price.KR),
                a, as.numeric(coef_est_10[i, 2:5]))[, 1]
            num_customer = seg.share[i] * 1000
            profit_KB = profit_KB + num_customer * prob_temp_KB * (iter - uc)
            # print('--') print(profit_KB)
            prob_temp_KR = demand(mean(kmeans_data[kmeans_data$cluster == i, ]$price.KB), iter2,
                a, as.numeric(coef_est_10[i, 2:5]))[, 2]
            profit_KR = profit_KR + num_customer * prob_temp_KR * (iter2 - uc)
            # print('**') print(profit_KR)
        }
        profit_total = profit_KB + profit_KR
        # print('++') print(profit_total)
        iter_row = append(iter_row, profit_total)
        profit_total = 0
    }
    profit_matrix[pricespace1_ind, ] = iter_row
    pricespace1_ind = pricespace1_ind + 1
}

max_profit_indices = which(profit_matrix == max(profit_matrix), arr.ind = TRUE)

paste("After launch of KB and MB=", pricespace4[profit_MB_q5 == max(profit_MB_q5)], "total max profit="
    max(profit_matrix), " at KB=", pricespace1[max_profit_indices[1]], " and KR=", pricespace1[max_profi
```

```
## [1] "After launch of KB and MB= 0.88 total max profit= 306.85896121596  at KB= 0.92  and KR= 0.95"
```

```
# pricespace4[profit_MB_q5 == max(profit_MB_q5)]
# plot(pricespace4,profit_MB_q5,type='l',xlab='Prices',
# ylab='Profit',ylim=c(10,300),col='blue',lwd=2, cex=2,cex.lab=1.5, cex.axis=1.5,
# cex.main=1.5, cex.sub=1.5)
```

```
# iter 4

pricespace4 = seq(0.5, 3, 0.01)
profit_MB_q5 = 0
for (i in (1:length(seg.share))) {
```

```
    w = seg.share[i]
    prob_demand_MB = demand(pricespace1[max_profit_indices[1]], pricespace2[max_profit_indices[2]],
        pricespace4, as.numeric(coef_est_10[i, 2:5]))[, 3]
    num_customer = seg.share[i] * 1000
    profit_MB_q5 = profit_MB_q5 + num_customer * prob_demand_MB * (pricespace4 - uc)
}

paste("After launch of KB, with KB=", pricespace1[max_profit_indices[1]], " and KR=", pricespace1[max_p
    " profit of MB=", max(profit_MB_q5), " at price=", pricespace4[profit_MB_q5 == max(profit_MB_q5)])
```

```
## [1] "After launch of KB, with KB= 0.92  and KR= 0.95  profit of MB= 117.601924814113  at price= 0.88
```

```
pricespace3 = seq(0.5, 3, 0.01)
pricespace1 = seq(0.5, 3, 0.01)
pricespace2 = seq(0.5, 3, 0.01)
profit_KR = 0
profit_MB = 0
profit_KB = 0


profit_matrix = matrix(, nrow = length(pricespace1), ncol = length(pricespace2))
pricespace1_ind = 1
pricespace2_ind = 0

for (iter in pricespace1) {
    iter_row = c()
    for (iter2 in pricespace2) {
        profit_KR = 0
        profit_KB = 0
        a = pricespace4[profit_MB_q5 == max(profit_MB_q5)]
        for (i in (1:length(seg.share))) {
            w = seg.share[i]
            prob_temp_KB = demand(iter, mean(kmeans_data[kmeans_data$cluster == i, ]$price.KR),
                a, as.numeric(coef_est_10[i, 2:5]))[, 1]
            num_customer = seg.share[i] * 1000
            profit_KB = profit_KB + num_customer * prob_temp_KB * (iter - uc)
            # print('--') print(profit_KB)
            prob_temp_KR = demand(mean(kmeans_data[kmeans_data$cluster == i, ]$price.KB), iter2,
                a, as.numeric(coef_est_10[i, 2:5]))[, 2]
            profit_KR = profit_KR + num_customer * prob_temp_KR * (iter2 - uc)
            # print('**') print(profit_KR)
        }
        profit_total = profit_KB + profit_KR
        # print('++') print(profit_total)
        iter_row = append(iter_row, profit_total)
        profit_total = 0
    }
    profit_matrix[pricespace1_ind, ] = iter_row
    pricespace1_ind = pricespace1_ind + 1
}

max_profit_indices = which(profit_matrix == max(profit_matrix), arr.ind = TRUE)
```

```
paste("After launch of KB and MB=", pricespace4[profit_MB_q5 == max(profit_MB_q5)], "total max profit="
    max(profit_matrix), " at KB=", pricespace1[max_profit_indices[1]], " and KR=", pricespace1[max_profi
```

```
## [1] "After launch of KB and MB= 0.88 total max profit= 306.85896121596  at KB= 0.92  and KR= 0.95"
```

```
# pricespace4[profit_MB_q5 == max(profit_MB_q5)]
# plot(pricespace4,profit_MB_q5,type='l',xlab='Prices',
# ylab='Profit',ylim=c(10,300),col='blue',lwd=2, cex=2,cex.lab=1.5, cex.axis=1.5,
# cex.main=1.5, cex.sub=1.5)
```

```
paste("Init: MB=1.43 -->> KB=1.15, KR=1.19")
```

```
## [1] "Init: MB=1.43 -->> KB=1.15, KR=1.19"
```

```
paste("Iter-1: KB=1.15, KR=1.19 -->> MB=0.96")
```

```
## [1] "Iter-1: KB=1.15, KR=1.19 -->> MB=0.96"
```

```
paste("Iter-1: MB=0.96 -->> KB=0.94, KR=0.96")
```

```
## [1] "Iter-1: MB=0.96 -->> KB=0.94, KR=0.96"
```

```
paste("Iter-2: KB=0.94, KR=0.96 -->> MB=0.89")
```

```
## [1] "Iter-2: KB=0.94, KR=0.96 -->> MB=0.89"
```

```
paste("Iter-2: MB=0.89 --> KB=0.92, KR=0.95")
```

```
## [1] "Iter-2: MB=0.89 --> KB=0.92, KR=0.95"
```

```
paste("Iter-3: KB=0.92, KR=0.95 --> MB=0.88")
```

```
## [1] "Iter-3: KB=0.92, KR=0.95 --> MB=0.88"
```

```
paste("Iter-3: MB=0.88 -->> KB=0.92, KB=0.95")
```

```
## [1] "Iter-3: MB=0.88 -->> KB=0.92, KB=0.95"
```

```
paste("Iter-4: KB=0.92, KR=0.95 --> MB=0.88")
```

```
## [1] "Iter-4: KB=0.92, KR=0.95 --> MB=0.88"
```

```
paste("Iter-4: MB=0.88 -->> KB=0.92, KB=0.95")
```

```
## [1] "Iter-4: MB=0.88 -->> KB=0.92, KB=0.95"
```

Thus we can see that after the launch of Kiwi Bubbles, the price of KB converges to $0.92, KR converges to $0.95 and MB converges to $0.88.

```r
# Without launch of KB Iter-1

pricespace4 = seq(0.6, 3, 0.01)
profit_MB_q5 = 0
for (i in (1:length(seg.share))) {
    w = seg.share[i]
    prob_demand_MB = demand_q4(1.06, pricespace4, as.numeric(coef_est_10[i, 2:5]))[, 2]
    num_customer = seg.share[i] * 1000
    profit_MB_q5 = profit_MB_q5 + num_customer * prob_demand_MB * (pricespace4 - uc)
}

paste("Without KB, in iter1, max profit by MB at KR=1.06 is ", max(profit_MB_q5), " at price ",
    pricespace4[profit_MB_q5 == max(profit_MB_q5)], sep = "")
```

```
## [1] "Without KB, in iter1, max profit by MB at KR=1.06 is 199.506252772117 at price 0.98"
```

```r
# adjusting kr prices according to the latest change in price of MB Iter-1
pricespace2 = seq(0.5, 3, 0.01)
profit_KR = 0

for (i in (1:length(seg.share))) {
    w = seg.share[i]
    prob_temp_KR = demand_q4(pricespace2, pricespace4[profit_MB_q5 == max(profit_MB_q5)], as.numeric(coe
        2:5]))[, 1]
    num_customer = 1000 * w
    profit_KR = profit_KR + num_customer * prob_temp_KR * (pricespace2 - uc)
}

paste("Without KB, in iter1, max profit by KR at MB=", pricespace4[profit_MB_q5 == max(profit_MB_q5)],
    " is ", max(profit_KR), " at price ", pricespace2[profit_KR == max(profit_KR)], sep = "")
```

```
## [1] "Without KB, in iter1, max profit by KR at MB=0.98 is 197.528828343969 at price 0.98"
```

```r
# Iter-2

pricespace4 = seq(0.6, 3, 0.01)
profit_MB_q5 = 0
for (i in (1:length(seg.share))) {
    w = seg.share[i]
    prob_demand_MB = demand_q4(pricespace2[profit_KR == max(profit_KR)], pricespace4, as.numeric(coef_es
        2:5]))[, 2]
    num_customer = seg.share[i] * 1000
    profit_MB_q5 = profit_MB_q5 + num_customer * prob_demand_MB * (pricespace4 - uc)
}

paste("Without KB, in iter-2, max profit by MB at KR=", pricespace2[profit_KR == max(profit_KR)],
    " is ", max(profit_MB_q5), " at price ", pricespace4[profit_MB_q5 == max(profit_MB_q5)])
```

```
## [1] "Without KB, in iter-2, max profit by MB at KR= 0.98  is  179.555220812596  at price  0.96"
```

```r
pricespace2 = seq(0.5, 3, 0.01)
profit_KR = 0

for (i in (1:length(seg.share))) {
    w = seg.share[i]
    prob_temp_KR = demand_q4(pricespace2, pricespace4[profit_MB_q5 == max(profit_MB_q5)], as.numeric(coe
        2:5]))[, 1]
    num_customer = 1000 * w
    profit_KR = profit_KR + num_customer * prob_temp_KR * (pricespace2 - uc)
}

paste("Without KB, in iter2, max profit by KR at MB=", pricespace4[profit_MB_q5 == max(profit_MB_q5)],
    " is ", max(profit_KR), " at price ", pricespace2[profit_KR == max(profit_KR)], sep = "")
```

```
## [1] "Without KB, in iter2, max profit by KR at MB=0.96 is 192.180236161499 at price 0.98"
```

```r
# Iter-3

pricespace4 = seq(0.6, 3, 0.01)
profit_MB_q5 = 0
for (i in (1:length(seg.share))) {
    w = seg.share[i]
    prob_demand_MB = demand_q4(pricespace2[profit_KR == max(profit_KR)], pricespace4, as.numeric(coef_es
        2:5]))[, 2]
    num_customer = seg.share[i] * 1000
    profit_MB_q5 = profit_MB_q5 + num_customer * prob_demand_MB * (pricespace4 - uc)
}

paste("Without KB, in iter-3, max profit by MB at KR=", pricespace2[profit_KR == max(profit_KR)],
    " is ", max(profit_MB_q5), " at price ", pricespace4[profit_MB_q5 == max(profit_MB_q5)])
```

```
## [1] "Without KB, in iter-3, max profit by MB at KR= 0.98  is  179.555220812596  at price  0.96"
```

```r
pricespace2 = seq(0.5, 3, 0.01)
profit_KR = 0

for (i in (1:length(seg.share))) {
    w = seg.share[i]
    prob_temp_KR = demand_q4(pricespace2, pricespace4[profit_MB_q5 == max(profit_MB_q5)], as.numeric(coe
        2:5]))[, 1]
    num_customer = 1000 * w
    profit_KR = profit_KR + num_customer * prob_temp_KR * (pricespace2 - uc)
}

paste("Without KB, in iter3, max profit by KR at MB=", pricespace4[profit_MB_q5 == max(profit_MB_q5)],
    " is ", max(profit_KR), " at price ", pricespace2[profit_KR == max(profit_KR)], sep = "")
```

```
## [1] "Without KB, in iter3, max profit by KR at MB=0.96 is 192.180236161499 at price 0.98"
```

```r
paste("Iter-1: MB=1.43 -->> KR=1.06")
```

```
## [1] "Iter-1: MB=1.43 -->> KR=1.06"
```

```
paste("Iter-1: KR=1.06 -->> MB=0.98")
```

```
## [1] "Iter-1: KR=1.06 -->> MB=0.98"
```

```
paste("Iter-2: MB=0.98 -->> KR=0.98")
```

```
## [1] "Iter-2: MB=0.98 -->> KR=0.98"
```

```
paste("Iter-2: KR=0.98 -->> MB=0.96")
```

```
## [1] "Iter-2: KR=0.98 -->> MB=0.96"
```

```
paste("Iter-3: MB=0.96 -->> KR=0.98")
```

```
## [1] "Iter-3: MB=0.96 -->> KR=0.98"
```

```
paste("Iter-4: KR=0.98 -->> MB=0.96")
```

```
## [1] "Iter-4: KR=0.98 -->> MB=0.96"
```

```
paste("Iter-4: MB=0.96 -->> KR=0.98")
```

```
## [1] "Iter-4: MB=0.96 -->> KR=0.98"
```

Thus we can see that without the launch of Kiwi Bubbles, the price of KR converges to $0.98 and MB converges to $0.96.

4.4) At these prices, how does the strategic advantage of Kiwi Bubbles change from the one you derived in the previous section (if at all)?

\textbf{Thus, comparing with and without launch of Kiwi Bubbles, the optimal price got reduced after the launch of Kiwi bubbles, while convergence occurred in both situations. Talking about our company particularly, our profit was close to $306 with Kiwi Bubbles in the market, while it converged to $179 in case of no Kiwi Bubbles in the market.}

Thus, it is highly recommended to launch Kiwi Bubbles to increase the company's profit share and cover larger consumer market distinguishly from it's existing product KR.