(SMM641) - Revenue Management & Pricing. Pricing Optimization with Consumer Choice Part 2- R Supplement

Oben Ceryan

Contents

1 Required Packages and Data

1.1 Use Package nloptr

Type the following code to install and activate the **nloptr** package:

```
#Remove below comment to install
#install.packages("nloptr",repos = "http://cran.us.r-project.org")
library(nloptr)
```

1.2 Use Package mlogit

Type the following code to install and activate the **mlogit** package:

```
#Remove below comment to install
#install.packages("mlogit",repos = "http://cran.us.r-project.org")
library(mlogit)
```

1.3 Use Package matrixStats

Type the following code to install and activate the **matrixStats** package:

```
#Remove below comment to install
#install.packages("matrixStats",repos = "http://cran.us.r-project.org")
library(matrixStats)
```

1.4 Use Package stargazer

Type the following code to install and activate the **stargazer** package:

```
#Remove below comment to install
#install.packages("stargazer",repos = "http://cran.us.r-project.org")
library(stargazer)
```

1.5 Data Files

Download and place in the R working directory the data files:

- "NYHCSurvey.csv"
- "NYHCstudentChoiceData.csv"

2 New York Health Club Part B: MNL Model with WTP data

2.1 Data Selection (All/Students/Nonstudents)

```r
# Read the Survey Data
nyhc = read.csv("NYHCSurvey.csv",header=T)

# (1) Uncomment below line to set prices for all client types
#
 nyhcnew = nyhc

# (2) To Set prices only for students
# Extract parts of data that correspond to students, i.e., type is 1.
# Uncomment below line to set prices only for students
#
# nyhcnew = subset(nyhc, Type == 1)

# (3) To Set prices for nonstudents (workers)
# Extract parts of data that correspond to Non students, i.e., type is 2 or 3.
# Uncomment below line to set prices only for nonstudents
#
# nyhcnew = subset(nyhc, Type == 2 | Type == 3)
```

2.2 Estimating the parameters of the MNL model from WTP data

```r
N=nrow(nyhcnew)

# Compute Average Willingness To Pay across Time Slots:
AvgWTPs=colMeans(nyhcnew[3:8])
SdWTPs=colSds(as.matrix(nyhcnew[3:8]))
VarWTPs=SdWTPs^2

AvgVarWTPs=mean(VarWTPs)

# The parameters of the MNL model:
util=AvgWTPs            # u_i
mu=sqrt(6*AvgVarWTPs)/pi  # mu, shape parameter
```

2.3 Finding the Optimal Price

```
eval_f <- function(x){
  price=x

  # An array of prices for all 6 time slots (all equal to the same price)
  prices=rep(price,6)

  # Calculating attraction values
  attractions=exp((util-prices)/mu)

  # Calculating purchase probabilities
  # The +1 corresponds to the no purchase option
  # (no purchase has utility=0, and price=0, so its attraction is exp(0)=1)
  purchaseProbs=attractions/(sum(attractions)+1)

  # Calculating Expected Revenue
  # N: total number of people, for example N=1000
  # Revenue = 1000*( (prob of purchase for timeslot1)*(price of timeslot1)
  #                  +(prob of purchase for timeslot2)*(price of timeslot2)
  #                  + ...
  #                  +(prob of purchase for timeslot6)*(price of timeslot6)

  revenue=N*sum(prices*purchaseProbs)

  objfunction=-revenue
  return(objfunction)
}

# initial values
x0 <- 80
# lower and upper bounds
lb <- 25
ub <- 150
opts <- list( "algorithm" = "NLOPT_LN_COBYLA",
              "xtol_rel"  = 1.0e-6,
              "maxeval"   = 1000)

result <- nloptr(x0=x0,eval_f=eval_f,lb=lb,ub=ub,opts=opts)
```

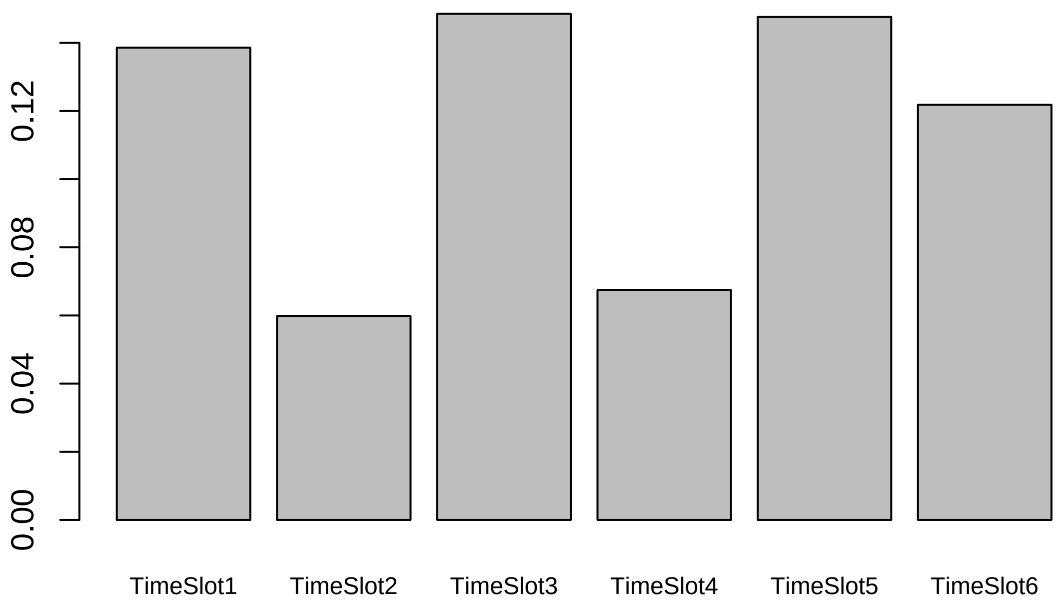
```r
priceOpt<-round(result$solution,2)

print(paste("Optimal Single Price:",priceOpt))
```

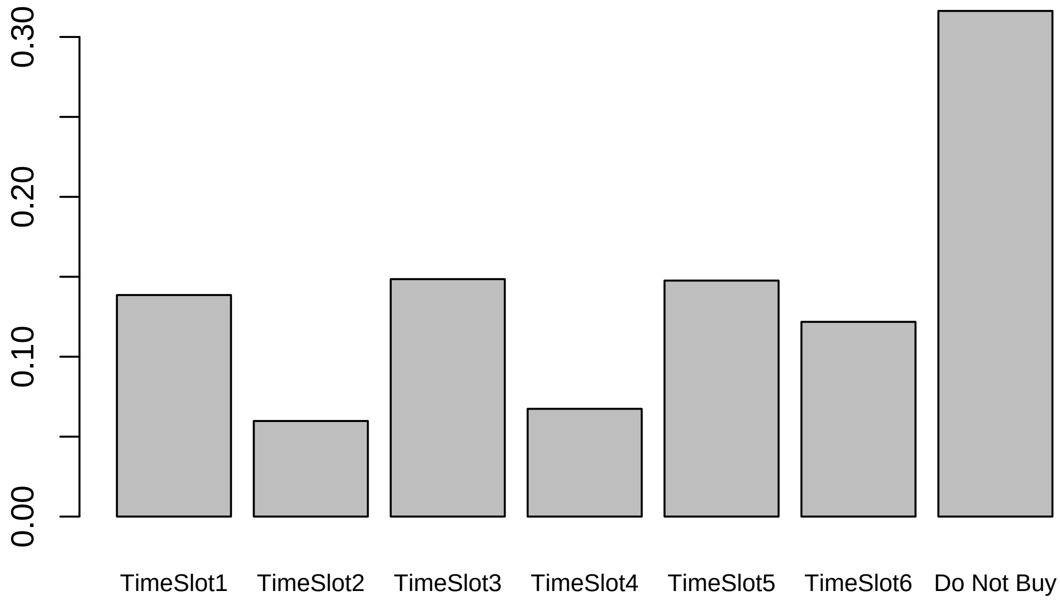
```
## [1] "Optimal Single Price: 82.76"
```

2.4 Visualizing Purchase Probabilities at the Optimal Price

```r
# Plotting Purchase Probabilities at the Optimal Price
prices=rep(priceOpt,6)
attractions=exp((util-prices)/mu)
purchaseProbs=attractions/(sum(attractions)+1)
barplot(purchaseProbs,cex.names=0.75)
```




```r
noPurchaseprob=1/(sum(attractions)+1)
names(noPurchaseprob) <- "Do Not Buy"
barplot(c(purchaseProbs,noPurchaseprob),cex.names=0.75)
```



3 New York Health Club Part B: MNL Model with Choice data

3.1 Structure of the Choice Data and Data Preparation

```r
# Read Choice Data (for students)
mydata <- read.csv("NYHCstudentChoiceData.csv")
head(mydata)
```

```
##   Individual price.T1 price.T2 price.T3 price.T4 price.T5 price.T6 price.DNB
## 1          1       50       50       50       50       50       50         0
## 2          2       50       50       50       50       50       50         0
## 3          3       60       60       60       60       60       60         0
## 4          4       60       60       60       60       60       60         0
## 5          5       50       50       50       50       50       50         0
## 6          6       60       60       60       60       60       60         0
##   choice
## 1     T4
## 2     T3
## 3     T5
## 4     T5
## 5     T6
## 6     T4
```

```r
# mlogit expects the price columns be named in the form of price.productname.
# For example, price.timeslot1, price.timeslot2, etc.
# We have introduced DNB to capture no purchase, i.e., Did Not Buy

# mlogit first needs to transform data into the shape it requires.
# Our original data is in the "wide" format,
# meaning each choice decision is in one row.
# varying means, which columns are varying properties for alternatives
# Here, they are the columns associated with the prices of each alternative.
# Choice means which column contains the overall choice of the customer.
# If the choice is stated as 1s and 0s in each row rather than directly given,
# we can do a simple preprocessing to generate a column with the choice name
# Here I prepared the data so that it is ready for the mlogit.data function

mydata2 <- mlogit.data(mydata, shape = "wide", varying=2:8, choice = "choice")
```

3.2 Estimating the Parameters for the MNL Demand Model

```r
# to obtain the parameters of the MNL demand model
mymodel <- mlogit(choice ~ price, data=mydata2,reflevel="DNB")

# Summarizing the results of the Estimation
stargazer(mymodel, type="text")
```

```
##
## ==========================================
##                     Dependent variable:
##                 ---------------------------
##                           choice
## ------------------------------------------
## (Intercept):T1          10.612**
##                          (4.607)
##
## (Intercept):T2           8.973*
##                          (4.613)
##
## (Intercept):T3           9.704**
##                          (4.609)
##
```

```
## (Intercept):T4          10.097**
##                          (4.608)
##
## (Intercept):T5          10.712**
##                          (4.607)
##
## (Intercept):T6          10.502**
##                          (4.607)
##
## price                   -0.155**
##                          (0.079)
##
## ------------------------------------------
## Observations               292
## R2                        0.005
## Log Likelihood          -512.976
## LR Test              4.993 (df = 7)
## ==========================================
## Note:          *p<0.1; **p<0.05; ***p<0.01
```

```r
# Remember from the MNL model, that the expression is in the form (u_i-p_i) / mu
# Hence (u_i/mu) is the intercept and (-1/mu) is the coefficient for price.
# That is, mu=-1/(price coefficient)

mu=-1/summary(mymodel)$coefficients[7]

print(paste("The shape parameter mu is:", round(mu,2)))
```

```
## [1] "The shape parameter mu is: 6.44"
```

```r
# And, u_i= intercept_i * mu

util=summary(mymodel)$coefficients[1:6]*mu
names(util)=paste0("T",c(1:6))

print("The gross utilities, u_i are:")
```

```
## [1] "The gross utilities, u_i are:"
```

```r
print(round(util,2))
```

```
##    T1    T2    T3    T4    T5    T6
```

```
## 68.30 57.74 62.45 64.98 68.94 67.59
```

3.3 Setting the Optimal Price

```r
# Finding the optimal price

N=nrow(mydata)

eval_f <- function(x){
  price=x

  # An array of prices for all 6 time slots (all equal to the same price)
  prices=rep(price,6)

  # Calculating attraction values
  attractions=exp((util-prices)/mu)

  # Calculating purchase probabilities
  # The +1 corresponds to the no purchase option
  # (no purchase has utility=0, and price=0, so its attraction is exp(0)=1)
  purchaseProbs=attractions/(sum(attractions)+1)

  # Calculating Expected Revenue
  # N: total number of people, for example N=1000
  # Revenue = 1000*( (prob of purchase for timeslot1)*(price of timeslot1)
  #                  +(prob of purchase for timeslot2)*(price of timeslot2)
  #                  + ...
  #                  +(prob of purchase for timeslot6)*(price of timeslot6)

  revenue=N*sum(prices*purchaseProbs)

  objfunction=-revenue
  return(objfunction)
}

# eval_g_ineq <- function(x) {
#   studentPrice=x[1]
#   nonstudentPrice=x[2]
#   constraint <- c(x[1]-x[2])
```

```r
#   return(constraint)
# }

# initial values
x0 <- 80
# lower and upper bounds of control
lb <- 25
ub <- 150
opts <- list( "algorithm" = "NLOPT_LN_COBYLA",
              "xtol_rel"  = 1.0e-6,
              "maxeval"   = 1000)
result <- nloptr(x0=x0,eval_f=eval_f,lb=lb,ub=ub,opts=opts)
#result <- nloptr(x0=x0,eval_f=eval_f,lb=lb,ub=ub,eval_g_ineq=eval_g_ineq,opts=opts)
# print(result)

priceOpt<-round(result$solution,2)

print(paste("Optimal Price:", priceOpt))
```

```
## [1] "Optimal Price: 63.48"
```

4 Assortment Optimization

```r
# Attractions for the products
v=c(3,5,6,4,5)

# Attraction for the outside option
v0=10

# Prices for the products
p=c(7,6,4,3,2)

# Let i1 denote an indicator funtion for whether product 1 will be
# included in the assortment or not.
# If i1=1, product 1 is included in the assortment
# If i1=0, product 1 is not included in the assortment

# Similarly, defined i2,i3,i4,i5.
```

```r
# We will consider all possible combinations of assortments.
# For each assortment, we will compute the purchase probabilities
# within that particular assortment and record the revenue

# For bookkeeping, we will track each case with an index variable.

index=1

Probpurchase=rep(NA,5)
Revenue=rep(NA,2^5)
assortmenttable=data.frame(matrix(nrow=2^5,ncol = 12))
indicatorLabels=c(paste0("Prod", 1:5))
purchaseProbLabels=c(paste0("pBuy", 1:5))
colnames(assortmenttable)=c("N",indicatorLabels,purchaseProbLabels, "Revenue")

for (i1 in 0:1){
  for (i2 in 0:1){
    for (i3 in 0:1){
      for (i4 in 0:1){
        for (i5 in 0:1){

          # Collecting all assortment indicators in a single array
          i=c(i1,i2,i3,i4,i5)

          # Computing the denominator for the purchase probabilities
          Denominator=sum(v*i)+v0

          #Obtaining the purchase probabilities
          for (j in 1:5){
            Probpurchase[j]=v[j]*i[j]/Denominator
          }

          # Revenue from this assortment
          Revenue[index]=sum(Probpurchase*p)

          assortmenttable[index,1]=sum(i)
          assortmenttable[index,2:6]=i
          assortmenttable[index,7:11]=round(Probpurchase,3)
```

```
          assortmenttable[index,12]=round(Revenue[index],3)

          index=index+1
        }
      }
    }
  }
}

assortmenttable2<-assortmenttable[order(assortmenttable$N),]
```