(SMM641) - Revenue Management & Pricing. Price Optimization, Setting Differentiated Prices - 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:

```r
install.packages("nloptr",repos = "http://cran.us.r-project.org")
```

```
##
##   There is a binary version available but the source version is later:
##          binary  source needs_compilation
## nloptr 1.2.2.1 1.2.2.2              TRUE
```

```
## installing the source package 'nloptr'
```

```
## Warning in install.packages("nloptr", repos = "http://cran.us.r-project.org"):
## installation of package 'nloptr' had non-zero exit status
```

```r
library(nloptr)
```

```
## Warning: package 'nloptr' was built under R version 3.5.2
```

2 Basic Price Optimization

2.1 Profit Maximization - Nonlinear Optimization in R

```r
# NonLinear Optimization

library("nloptr")

# Profit Maximization

# Constructing the Objective Function
eval_f <- function(x){
  price=x
  demand=max(0,10000-800*price)
  cost=5
  profit=(price-cost)*demand
  objfunction=-profit
  return(objfunction)
}
```

```r
# Setting Initial values for Decision Variables
x0 <- 10

# Setting Lower and Upper Bounds for Decision Variables
lb <- 5
ub <- 12.5

# Setting Optimization Options
opts <- list( "algorithm" = "NLOPT_LN_COBYLA",
              "xtol_rel"  = 1.0e-6,
              "maxeval"   = 1000)

# Running the Optimization Problem

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

# Viewing the Results
print(result)
```

```
##
## Call:
## nloptr(x0 = x0, eval_f = eval_f, lb = lb, ub = ub, opts = opts)
##
##
## Minimization using NLopt version 2.4.2
##
## NLopt solver status: 4 ( NLOPT_XTOL_REACHED: Optimization stopped because
## xtol_rel or xtol_abs (above) was reached. )
##
## Number of Iterations....: 37
## Termination conditions:  xtol_rel: 1e-06 maxeval: 1000
## Number of inequality constraints:  0
## Number of equality constraints:    0
## Optimal value of objective function:  -11249.9999999985
## Optimal value of controls: 8.750001
```

```r
# Optimal Decision Variables
priceOpt<-result$solution
print(paste("Optimal value of Decision Variable(s):",
            priceOpt))
```

```
## [1] "Optimal value of Decision Variable(s): 8.75000136581421"
```

```r
# Optimal Profit (Revenue)
profitOpt<- -result$objective
print(paste("Optimal Profit:",
            profitOpt))
```

```
## [1] "Optimal Profit: 11249.9999999985"
```

3 Concert Ticket Pricing

3.1 Single Price for Two Segments

```r
# Single Price

# Constructing the Objective Function
eval_f <- function(x){
  price=x
  memberDemand=max(0,10000-100*price)
  publicDemand=max(0,25000-125*price)
  revenue=price*(memberDemand+publicDemand)
  objfunction=-revenue
  return(objfunction)
}

# Constructing the Constraints
eval_g_ineq <- function(x) {
  price=x
  cap=10000
  memberDemand=max(0,10000-100*price)
  publicDemand=max(0,25000-125*price)
  # Constraint 1: total tickets <= capacity
  # Constraint 2: member tickets >= 0
  # Constraint 3: public tickets >= 0
  constraint <- c(memberDemand+publicDemand-cap,
                  -memberDemand,
                  -publicDemand)
  return(constraint)
}
```

```r
# Initial Value(s) for Decision Variable(s):
x0 <- 80
# Lower and Upper Bounds for Decision Variable(s):
lb <- 0
ub <- 200
# Optimization options
opts <- list( "algorithm" = "NLOPT_LN_COBYLA",
              "xtol_rel"  = 1.0e-9,
              "maxeval"   = 1000)

# Running the Optimization
result <- nloptr(x0=x0,eval_f=eval_f,lb=lb,ub=ub,
                 eval_g_ineq=eval_g_ineq,opts=opts)

# print(result)

priceOpt<-result$solution
RevenueOpt<- -result$objective
soldMember=max(0,10000-100*priceOpt)
soldPublic=max(0,25000-125*priceOpt)
soldTickets=soldMember+soldPublic

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

```
## [1] "Optimal Price: 120"
```

```r
print(paste("Optimal Revenue:",RevenueOpt))
```

```
## [1] "Optimal Revenue: 1200000"
```

```r
print(paste("Member Tickets Sold:",soldMember))
```

```
## [1] "Member Tickets Sold: 0"
```

```r
print(paste("Public Tickets Sold:",soldPublic))
```

```
## [1] "Public Tickets Sold: 10000"
```

```r
print(paste("Total Tickets Sold:",soldTickets))
```

```
## [1] "Total Tickets Sold: 10000"
```

3.2 Setting Differentiated Prices for Two Segments

```r
# Differentiated Prices

eval_f <- function(x){
    # Now the variable x has two dimensions for each price
    # Let's set first element as member price
    # Set second element as public price
  memberPrice=x[1]
  publicPrice=x[2]
  memberDemand=max(0,10000-100*memberPrice)
  publicDemand=max(0,25000-125*publicPrice)
  revenue=memberPrice*memberDemand+publicPrice*publicDemand
  objfunction=-revenue
  return(objfunction)
}

eval_g_ineq <- function(x) {
  memberPrice=x[1]
  publicPrice=x[2]
  memberDemand=max(0,10000-100*memberPrice)
  publicDemand=max(0,25000-125*publicPrice)
  cap=10000
  # Add Constraint 4: Member Price <= Public Price
  constraint <- c(memberDemand+publicDemand-cap,
                  -memberDemand,
                  -publicDemand,
                  x[1]-x[2])
  return(constraint)
}


# initial values
x0 <- c(80,150)
# lower and upper bounds of control
lb <- c(0,0)
ub <- c(100,200)
opts <- list( "algorithm" = "NLOPT_LN_COBYLA",
              "xtol_rel"  = 1.0e-9,
               "maxeval"   = 1000)
result <- nloptr(x0=x0,eval_f=eval_f,lb=lb,ub=ub,
                 eval_g_ineq=eval_g_ineq,opts=opts)
# print(result)

priceOpt<-result$solution
RevenueOpt<- -result$objective
soldMember=max(0,10000-100*priceOpt[1])
soldPublic=max(0,25000-125*priceOpt[2])
soldTickets=soldMember+soldPublic

print(paste("Optimal Price for Members:",priceOpt[1]))
```

```
## [1] "Optimal Price for Members: 83.3333336497098"
```

```r
print(paste("Optimal Price for Public:",priceOpt[2]))
```

```
## [1] "Optimal Price for Public: 133.333333080232"
```

```r
print(paste("Optimal Revenue:",RevenueOpt))
```

```
## [1] "Optimal Revenue: 1250000"
```

```r
print(paste("Member Tickets Sold:",soldMember))
```

```
## [1] "Member Tickets Sold: 1666.66663502902"
```

```r
print(paste("Public Tickets Sold:",soldPublic))
```

```
## [1] "Public Tickets Sold: 8333.33336497098"
```

```r
print(paste("Total Tickets Sold:",soldTickets))
```

```
## [1] "Total Tickets Sold: 10000"
```

3.3 Differentiated Prices for Two Segments with a Limit on Price Difference

```r
# Differentiated Prices (max Price Difference is 25)
eval_f <- function(x){
  memberPrice=x[1]
  publicPrice=x[2]
  memberDemand=max(0,10000-100*memberPrice)
  publicDemand=max(0,25000-125*publicPrice)
  revenue=memberPrice*memberDemand+publicPrice*publicDemand
  objfunction=-revenue
  return(objfunction)
}

eval_g_ineq <- function(x) {
  memberPrice=x[1]
  publicPrice=x[2]
  memberDemand=max(0,10000-100*memberPrice)
  publicDemand=max(0,25000-125*publicPrice)
  cap=10000
  constraint <- c(memberDemand+publicDemand-cap,
                  -memberDemand,
                  -publicDemand,
                  x[1]-x[2],
                  x[2]-x[1]-25)
  return(constraint)
}


# initial values
x0 <- c(80,100)
# lower and upper bounds of control
lb <- c(0,0)
ub <- c(100,200)
opts <- list( "algorithm" = "NLOPT_LN_COBYLA",
              "xtol_rel"  = 1.0e-9,
              "maxeval"   = 1000)
result <- nloptr(x0=x0,eval_f=eval_f,lb=lb,ub=ub,
                 eval_g_ineq=eval_g_ineq,opts=opts)
print(result)
```

```
##
## Call:
##
## nloptr(x0 = x0, eval_f = eval_f, lb = lb, ub = ub, eval_g_ineq = eval_g_ineq,
##     opts = opts)
##
##
## Minimization using NLopt version 2.4.2
```

```
##
## NLopt solver status: 4 ( NLOPT_XTOL_REACHED: Optimization stopped because
## xtol_rel or xtol_abs (above) was reached. )
##
## Number of Iterations....: 39
## Termination conditions:  xtol_rel: 1e-09 maxeval: 1000
## Number of inequality constraints:  5
## Number of equality constraints:    0
## Optimal value of objective function:  -1215277.77777778
## Optimal value of controls: 97.22222 122.2222
```

```
priceOpt<-result$solution
RevenueOpt<- -result$objective
soldMember=max(0,10000-100*priceOpt[1])
soldPublic=max(0,25000-125*priceOpt[2])
soldTickets=soldMember+soldPublic

print(paste("Optimal Price for Members:",priceOpt[1]))
```

```
## [1] "Optimal Price for Members: 97.2222222222222"
```

```
print(paste("Optimal Price for Public:",priceOpt[2]))
```

```
## [1] "Optimal Price for Public: 122.222222222222"
```

```
print(paste("Optimal Revenue:",RevenueOpt))
```

```
## [1] "Optimal Revenue: 1215277.77777778"
```

```
print(paste("Member Tickets Sold:",soldMember))
```

```
## [1] "Member Tickets Sold: 277.777777777783"
```

```
print(paste("Public Tickets Sold:",soldPublic))
```

```
## [1] "Public Tickets Sold: 9722.22222222222"
```

```
print(paste("Total Tickets Sold:",soldTickets))
```

```
## [1] "Total Tickets Sold: 10000"
```

3.4 Differentiated Prices for Three Segments (Members/Public Standing/Public Seated)

```
# Differentiated Prices with Seats/Standing for Public
eval_f <- function(x){
  memberPrice=x[1]
  publicStandPrice=x[2]
  publicSeatPrice=x[3]
  memberDemand=max(0,10000-100*memberPrice)
  publicStandDemand=max(0,12000-75*publicStandPrice)
  publicSeatDemand=max(0,15000-50*publicSeatPrice)
  revenue=memberPrice*memberDemand+
      publicStandPrice*publicStandDemand+
      publicSeatPrice*publicSeatDemand
  objfunction=-revenue
  return(objfunction)
}

eval_g_ineq <- function(x) {
  memberPrice=x[1]
  publicStandPrice=x[2]
  publicSeatPrice=x[3]
  memberDemand=max(0,10000-100*memberPrice)
  publicStandDemand=max(0,12000-75*publicStandPrice)
  publicSeatDemand=max(0,15000-50*publicSeatPrice)
  cap=10000
  constraint <- c(memberDemand+publicStandDemand+1.6*publicSeatDemand-cap,
                  -memberDemand,
                  -publicStandDemand,
                  -publicSeatDemand,
                  x[1]-x[2],
                  x[1]-x[3])
  return(constraint)
}


# initial values
x0 <- c(80,110,120)
# lower and upper bounds of control
```

```r
lb <- c(0,0,0)
ub <- c(100,160,300)
opts <- list( "algorithm" = "NLOPT_LN_COBYLA",
              "xtol_rel"  = 1.0e-9,
              "maxeval"   = 1000)
result <- nloptr(x0=x0,eval_f=eval_f,lb=lb,ub=ub,
                 eval_g_ineq=eval_g_ineq,opts=opts)
# print(result)

priceOpt<-result$solution
RevenueOpt<- -result$objective
soldMember=max(0,10000-100*priceOpt[1])
soldPublicStand=max(0,12000-75*priceOpt[2])
soldPublicSeat=max(0,15000-50*priceOpt[3])
soldTickets=soldMember+soldPublicStand+1.6*soldPublicSeat

print(paste("Optimal Price for Members:",priceOpt[1]))
```

```
## [1] "Optimal Price for Members: 92.9042891777248"
```

```r
print(paste("Optimal Price for Standing Public:",priceOpt[2]))
```

```
## [1] "Optimal Price for Standing Public: 122.904290463361"
```

```r
print(paste("Optimal Price for Seated Public:",priceOpt[3]))
```

```
## [1] "Optimal Price for Seated Public: 218.646866218443"
```

```r
print(paste("Optimal Revenue:",RevenueOpt))
```

```
## [1] "Optimal Revenue: 1297244.22442244"
```

```r
print(paste("Member Tickets Sold:",soldMember))
```

```
## [1] "Member Tickets Sold: 709.571082227521"
```

```r
print(paste("Standing Public Tickets Sold:",soldPublicStand))
```

```
## [1] "Standing Public Tickets Sold: 2782.17821524792"
```

```r
print(paste("Seated Public Tickets Sold:",soldPublicSeat))
```

```
## [1] "Seated Public Tickets Sold: 4067.65668907785"
```

```r
print(paste("Total Tickets Sold:",soldTickets))
```

```
## [1] "Total Tickets Sold: 9999.99999999999"
```