SMM 641
Revenue Management and Pricing

Week 7:

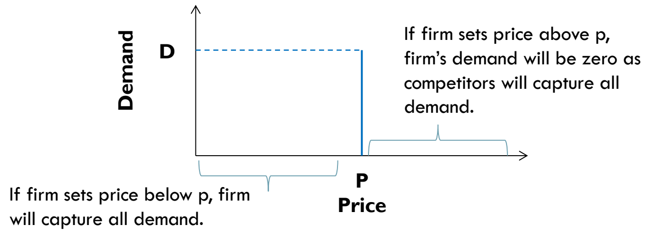
Basic Price Optimization

Price Differentiation

Setting Differentiated Prices under Constrained Capacity

Price Response Function

- A fundamental input to any pricing and revenue optimization analysis is price-response function, which determines:
  - how demand for a product changes as a function of its price p.
- There is one price-response function for each combination of product, channel and market segment.
- In a *perfectly competitive market*, price-response function is a vertical line at the market price:

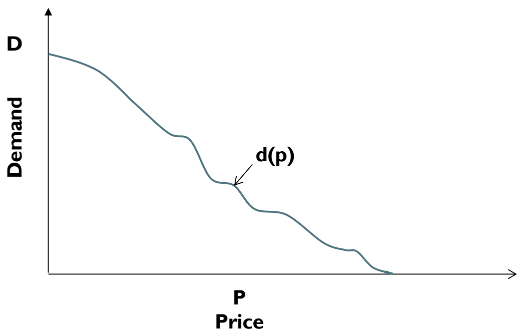


Price Response Function

The price response functions we consider – as faced by most companies - are:

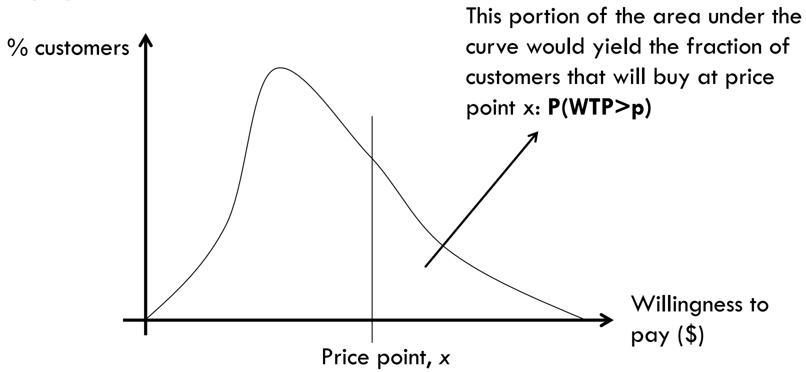
- Non-negative
- Continuous (no gaps, no jumps)
- Differentiable
- Downward sloping

  There are certain cases where price response function may not have downward slope:
  - When price is an indicator of quality
  - Conspicuous consumption



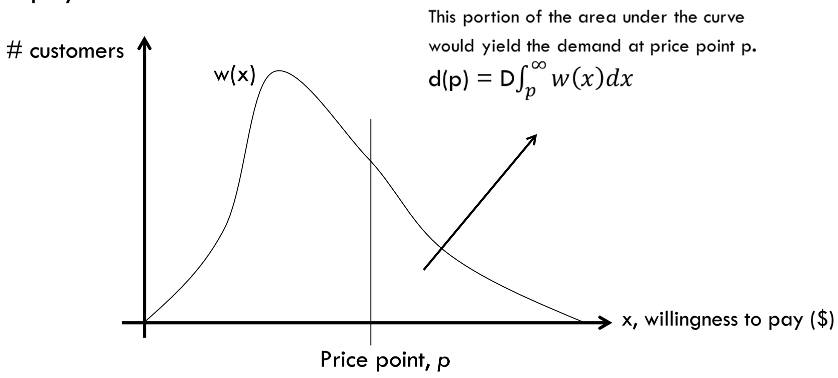
Willingness to pay view of demand functions

- Willingness to pay (aka reservation price): the highest price an individual is willing to pay for a product.
- In theory, a demand curve (aka price response function) is a culmination of several individuals acting out their willingness to pay.



Willingness to pay view of demand functions

- **Willingness to pay (aka reservation price):** the highest price an individual is willing to pay for a product.
- In theory, **a demand curve (aka price response function)** is a culmination of several individuals acting out their willingness to pay.



Estimation Procedure (more on this later)

- Estimate willingness to pay distribution (customer behavior forecast)

  Willingness to pay distribution →**P(WTP>p)**

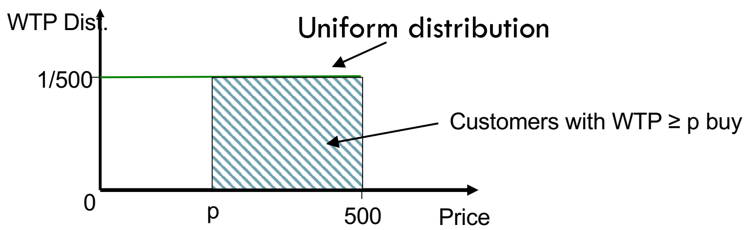
- Estimate number of customers making purchase decisions (volume forecast)

  N

- Form demand function estimate

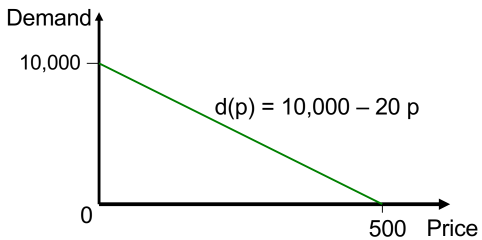
  d(p) = N P(WTP>p)

Linear Demand Curve – Uniform WTP



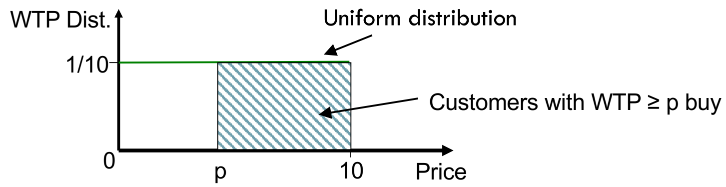
Suppose market size N=10,000.

d(p) = 10,000 · P(WTP ≥ p)
= 10,000 · (1 – p/500)
= 10,000 – 20 p

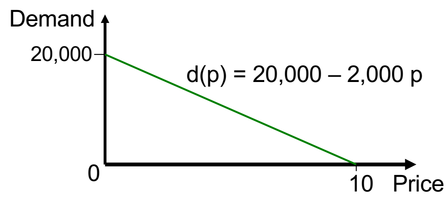


Linear Demand Curve – Uniform WTP

- Example: The total potential market for a notebook is 20000 and willingness to pay is distributed uniformly between $0 and $10. Derive the price-response function.



d(p) = 20,000 · P(WTP ≥ p)
= 20,000 · (1 – p/10)
= 20,000 – 2,000 p



Basic Price Optimization

- The basic price optimization problem is:

$$\max_{p} \ (p - c) \cdot d(p)$$

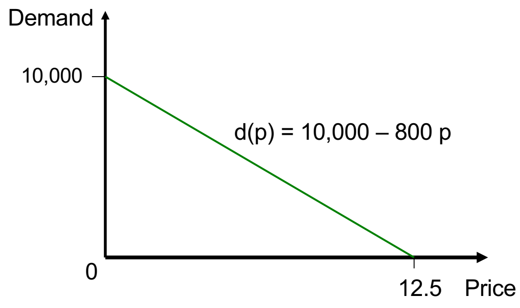
where:

- p is the price of the product
- c is the unit (incremental) cost, e.g.,
  - In airlines, additional meal and fuel for an additional seat sold, *not* the pilot/crew wages (fixed costs), etc.
  - In groceries, cost to replenish an item after it is sold.
- $d(p)$ is the demand at price p, i.e., the price response function.

Basic Price Optimization

- Let the price-response function for a product be given by:

  $d(p)$=10,000 – 800 p
- The unit incremental cost for the product is $5.



Basic Price Optimization

- Let the price-response function for a product be given by:

  $d(p)$=10,000 – 800 p
- The unit incremental cost for the product is $5.

- We can state the problem as:

$$\max_{p} \ (p - 5) \cdot (10{,}000 - 800\, p)$$

- Equating the first derivative of profit with respect to price to zero:

$$(10{,}000 - 800\, p^*) + (p^* - 5) \cdot (-800) = 0$$

$$1600\, p^* = 14{,}000$$

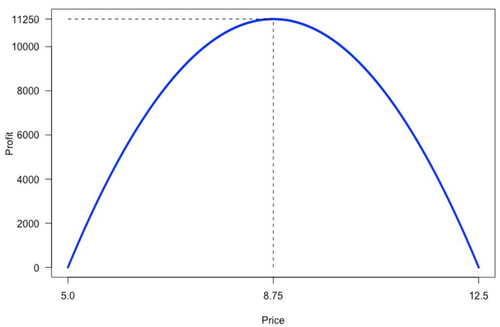
$$p^* = \$8.75$$

  - (Note, the second derivative of profit with respect to price is negative, implying concavity)

Basic Price Optimization

- Let the price-response function for a product be given by:

  $d(p)$=10,000 – 800 p
- The unit incremental cost for the product is $ 5.
- The profit maximizing price is p* = $ 8.75
- The optimal profit is: (8.75-5)*(10000-800*8.75) = $ 11,250.



Price Optimization Using R – Nonlinear Programming

- Notice that profit (revenue) is nonlinear in the decision variable, p
  - E.g., for profit: $\max_p \; (p - 5) * (10{,}000 - 800\,p)$
  - The objective function has a p^2 term.
- We can use R to solve Nonlinear Optimization Problems
  - We will use R package `nloptr`
  - Type the following code to install and activate the `nloptr` package:

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

  - Activate package by typing:

```
library(nloptr)
```

Price Optimization Using R – Nonlinear Programming

- Recall from Week 3, when we discussed the generic form of optimization problems:

$$\begin{aligned} &maximize\,(minimize) \quad f(x) \\ &subject\ to \quad g_1(x) \le \ (=, or \ge) \ A \\ &\qquad\qquad\quad\ g_2(x) \ge \ (=, or \ge) \ B \end{aligned}$$

Decision Variables

Objective function

Constraints

Optimization problems have three components.

1. Decision variables (e.g. quantities for product i)	$x = (x_1, x_2, x_3, \ldots, x_n)$
2. Objective function (e.g. profit, cost)	$f(x) = f(x_1, x_2, \ldots, x_n)$
3. Constraints (e.g. restriction of available resource)	$g_1(x) = g_1(x_1, x_2, \ldots, x_n) \le A$ $g_2(x) = g_2(x_1, x_2, \ldots, x_n) \ge B$

Examples of constraints:

- Room allocations cannot exceed demand.
- Total seat allocation cannot exceed plane capacity.

Price Optimization Using R – Nonlinear Programming

- The package `nloptr` (as many others) requires the optimization problem to be entered in a specific format.
- The objective must be to minimize.
  - !!!??? We were trying to maximize profit or revenue?
  - Workaround: We will minimize -Profit, or -Revenue.
- All inequality constraints should be of the form g(x) ≤ 0
  - Workaround 1: If g(x) ≤ 5, we can write g(x) -5 ≤ 0
  - Workaround 2: If g(x) ≥ 0, we can write -g(x) ≤ 0
  - Workaround 3: If g(x) ≥ 5, we can write -g(x)+5 ≤ 0
- All equality constraints should be of the form h(x) = 0
  - Workaround: If h(x) = 5, we can write h(x) -5 = 0

Price Differentiation

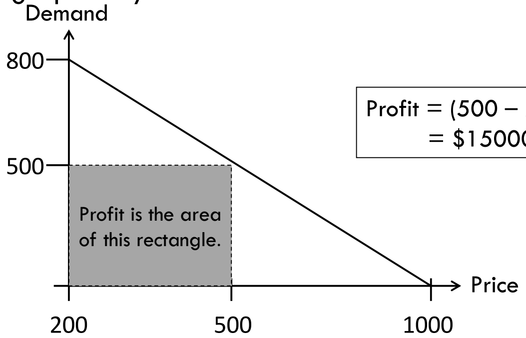
- Charging different prices to different customers
  - either for exactly the same good
  - or for slightly different versions of the same good
- Powerful way to improve profitability, yet complex
- Price differentiation is an art:
  - Finding ways to divide the market into different segments such that higher prices can be charged to the high willingness-to-pay segments and lower prices to the low willingness-to-pay segments
- Price differentiation is a science:
  - Setting and updating the prices in order to maximize overall return from all segments

Price Differentiation

- Suppose demand for a product, *D*, is the following function of price, *p*:

  $D = 1000 - p$

- Suppose this product's unit cost to the seller is \$200. Imagine the seller charges \$500 per unit. The seller's profit can be captured graphically as follows:



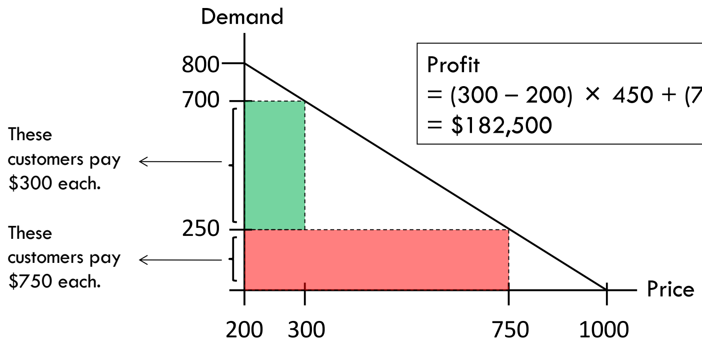
Profit = (500 – 200) × 500
= \$150000

Price Differentiation

- There are 500 people willing-to-pay more than \$500.
  - Potential revenue "left on the table".
- There are 300 people who is willing to pay more than what the product costs the seller (\$200), yet cannot buy it because it is too expensive.
  - Potential untapped customers, hence additional revenue.

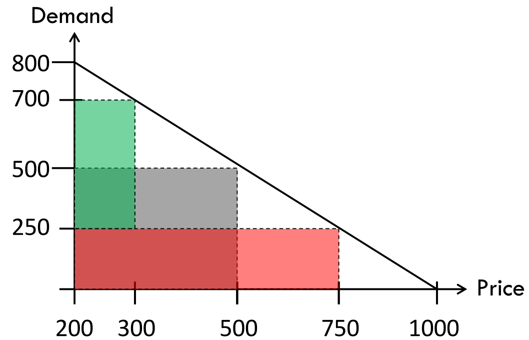
Price Differentiation

- Suppose now:
  - We charge two different prices: \$300 and \$750
  - Anybody who is willing to pay \$750 or more purchase at \$750.
  - Anybody who is willing to pay more than \$300, but less than 750 purchase at \$300.
- The seller's profit will now be as follows:



Profit
= (300 – 200) × 450 + (750 – 200) × 250
= \$182,500

Price Differentiation



With carefully chosen prices and an ability to make customers pay the highest price they are willing to pay, price differentiation can help improve the bottom line substantially.

Limits to Price Differentiation

- Imperfect segmentation / cannibalization:
  - What if we cannot identify the customers who are willing to pay \$750 (imperfect segmentation)?
  - What if customers who are willing to pay \$750 find a way to purchase at the lower price of \$300 (cannibalization)?
- Arbitrage:
  - What if there existed a group of speculators who decided to buy at \$300 and sell to others at \$750?

Limits to Price Differentiation

- Suppose that, due to imperfect segmentation and/or cannibalization 80% of the customers who are willing to pay \$750 end up paying \$300 instead.

Profit before imperfect segmentation / cannibalization
$= (300-200) \times 450 + (750-200) \times 250$
$= \$182,500$

Profit after imperfect segmentation / cannibalization
$= (300-200) \times (450 + 200) + (750-200) \times (250 - 200)$
$= \$92,500 < \$150,000$

- This profit is worse than what we would obtain if we charged a uniform price of \$500!

Tactics for Price Differentiation

- Group pricing
- Channel pricing
- Regional pricing
- Product versioning
- Time-based differentiation

Tactics for Price Differentiation

Group Pricing: Offering different prices to different groups of customers for exactly the same product.

- Student / senior citizen discounts

- Family specials

Spotify

- Discounts for loyalty

Exclusive to myWaitrose members

- Lower prices for government, educational, non-profit organizations

Apple Education Pricing
Available to current and newly accepted college students and their parents, as well as faculty, staff, and homeschool teachers of all grade levels.*

Tactics for Price Differentiation

Conditions for group pricing to work:

- There must be an easy and reliable way to check an individual's group membership.
- Groups must be distinct from one another in their price sensitivity.
- There must exist impediments to a secondary market.
- Not only the practice must be legal, but also it must be perceived as "fair."

Tactics for Price Differentiation

Channel Pricing: Selling the same product for different prices through different distribution channels.

Examples:

- Delta.com vs. Expedia vs. travel agency
- Different prices at brick-and-mortar stores vs. online
- "Fashion malls" charging different prices than outlet stores
- Have you visited a price comparison web site?

Reasons:

- Different distribution costs across different channels.
- Customer's choice of channel is an indicator of price sensitivity.

Tactics for Price Differentiation

Regional Pricing: Selling the product at different prices at different locations.

Examples:

- Gas stations
- Different prices in higher/lower cost of living areas
- Airports

Tactics for Price Differentiation

Product versioning: Designing inferior and superior versions of the product and selling them at different prices.

- Olive crop in Turkey
  - Classify the crop into several categories depending on size
  - Higher prices for bigger olives
- Inferior products
  - Software or hardware with fewer features --- obtained by turning off functionality
  - National brands sell the same product to retailers who sell the product to consumers as a "house brand"
- Product lines
  - *Horizontal differentiation*: products differ in flavor, color, etc --- typically affords little or no opportunity for price differentiation
  - *Vertical differentiation*: products differ in perceived quality, number of features, etc --- enables price differentiation

Tactics for Price Differentiation

Time-based differentiation:

- Plane tickets / hotels
- Dry cleaners: same day service vs. regular
- Fashion apparel
- Due-date-dependent pricing in custom-made products

Setting Differentiated Prices: Concert Price Setting*

- Suppose up to 10,000 tickets are available.
- There are two segments:
  - 10,000 members. At price p, the demand for tickets from members is:

$$10{,}000 - 100p.$$

  - General public. At price p, the demand for tickets from members is:

$$25{,}000 - 125p.$$

* Exercise partially based on a case study by Popescu (INSEAD)

Concert Price Setting

- Suppose a single price will be charged for all tickets. What is the optimal price?

Optimal price =

# of tickets sold =

Member attendance=

General public attendance =

Revenue =

Concert Price Setting – Solving Nonlinear Optimization with R

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

```r
library(nloptr)
```

Concert Price Setting – Solving Nonlinear Optimization with R

```r
# 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)
}
```

Concert Price Setting – Solving Nonlinear Optimization with R

```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)
```

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

Concert Price Setting – Solving Nonlinear Optimization with R

```r
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"
```

Concert Price Setting

- Suppose a single price will be charged for all tickets. What is the optimal price?

  Optimal price = € 120

  # of tickets sold = 10,000

  Member attendance = 0

  General public attendance = 10,000

  Revenue = € 1,200,000

Concert Price Setting

- Does revenue improve if discounts are given to members?

Optimal member price =

Optimal general public price =

# of tickets sold =

Member attendance=

General public attendance =

Revenue =

Concert Price Setting

```
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)
}
```

Concert Price Setting

```
# 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
```

Concert Price Setting

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

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

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

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

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

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

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

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

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

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

Concert Price Setting

- Does the revenue improve if discounts are given to members?

Optimal member price = € 83.3

Optimal general public price = € 133.3

# of tickets sold = 10,000

Member attendance= 1,667

General public attendance =8,333

Revenue = € 1,250,000 (4% increase compared to single price)

Concert Price Setting

- Takeaways:
  - Price differentiation allows the seller to increase overall revenues.
- Who benefits/suffers due to price differentiation?
  - Some members benefit because now they can afford going to the concert
  - General public suffers because either they cannot go, or they have to pay more than they would have
  - The organizers benefit as their revenue increases

Concert Price Setting

- Organizers believe members should not get more than a € 25 discount. What are the optimal prices now?

Optimal member price =

Optimal general public price =

# of tickets sold =

Member attendance=

General public attendance =

Revenue =

Concert Price Setting

```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)
}
```

Concert Price Setting

```r
# 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)
```

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

Concert Price Setting

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

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

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

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

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

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

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

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

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

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

Concert Price Setting

- Organizers believe members should not get more than a € 25 discount. What are the optimal prices now?

Optimal member price = € 97.2

Optimal general public price = € 122.2

# of tickets sold = 10,000

Member attendance = 278

General public attendance = 9,722

Revenue = 1,215,277

Concert Price Setting

- Suppose now we could add a seated zone among the public tickets.
- Every seat takes away 1.6 units of standing space.
- Assume that members go for standing tickets only.
- As for general public, their demand for:
  - the standing tickets is estimated as 12,000 – 75p.
  - the seated zone is estimated as 15,000 – 50p,
- What prices should be charged for members, standing-public and seated-public tickets?
- As a result, how many seat tickets should be sold?

Concert Price Setting

- Suppose now we could add a seated zone among the public tickets.

Optimal member price =

Optimal standing-public price =

Optimal seated-public price =

seated tickets sold =

standing tickets sold to members =

standing tickets sold to general public=

Revenue =

Concert Price Setting

```
# 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)
}
```

Concert Price Setting

```
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)
}
```

Concert Price Setting

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

Concert Price Setting

```r
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"
```

Concert Price Setting

```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"
```

Concert Price Setting

- Suppose now we could add a seated zone among the public tickets.

  Optimal member price = € 92.9

  Optimal standing-public price = € 122.9

  Optimal seated-public price = € 218.6

  # seated tickets sold = 4,068

  # standing tickets sold to members = 710

  # standing tickets sold to general public= 2,782

  Revenue = € 1,297,244

Setting up price fences

If a firm has the ability to set up price fences:

- The firm can separate the customer population into identifiable segments, and
- the firm can target a product of its choice at each segment (some segments may be targeted with the same product), and
- the firm can then charge different prices to different segments (even for the same product, if it is profitable to do so)

In short: Each segment lives in a world of their own, where they have no access to the price / product combination available to other segments.