**Demand models**

About pricing optimization, we discussed a little about linear demand and how to estimate optimal prices in that case. In this post we are going to compare three different types of demand models for homogeneous products and how to find optimal prices for each one of them.

For the linear model, demand is given by:

\displaystyle d(p) = \alpha p + \beta,

where \alphais the slope of the curve and \betathe intercept. For the linear model, the elasticity goes from zero to infinity. Another very common demand model is the constant-elasticity model, given by:

\displaystyle \ln d(p) = \alpha \ln p + \beta,  
or

\displaystyle d(p) = d_0 e^\beta p^\alpha = Cp^\alpha,

where \alphais the elasticity of the demand and Cis a scale factor. A much more interesting demand curve is given by the logistic/sigmoide function:

\displaystyle d(p) = C\frac{e^{\alpha p + \beta}}{1 + e^{\alpha p + \beta}} = \frac{C}{1+e^{-\alpha(p - p_0)}},

where Cis a scale factor and \alphameasures price sensitivity. We also can observe p_0 = -\alpha/\betaas the inflection point of the demand.

Some books changes the signs of the coefficients using the assumption that \alphais a positive constant and using a minus sign in front of it. However, it does not change the estimation procedure or final result, it is just a matter of convenience. Here, we expect \alphato be negative in the three models.

In the Figure below we can check a comparison among the shapes of the demand models:

library(ggplot2)

library(reshape2)

library(magrittr)

linear = function(p, alpha, beta) alpha\*p + beta

constant\_elast = function(p, alpha, beta) exp(alpha\*log(p)+beta)

logistic = function(p, c, alpha, p0) c/(1+exp(-alpha\*(p-p0)))

p = seq(1, 100)

y1 = linear(p, -1, 100)

y2 = constant\_elast(p, -.5, 4.5)

y3 = logistic(p, 100, -.2, 50)

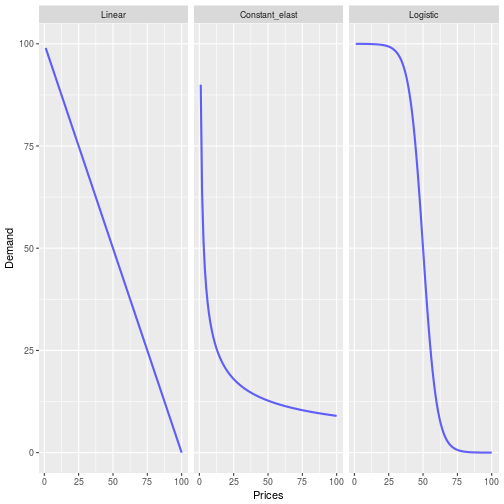
df = data.frame('Prices' = p, 'Linear' = y1, 'Constant\_elast' = y2, 'Logistic' = y3)

df.plot = melt(df, id = 'Prices') %&gt;% set\_colnames(c('Prices', 'Model', 'Demand'))

ggplot(df.plot) + aes(x = Prices, y = Demand) +

geom\_line(color = 'blue', alpha = .6, lwd = 1) +

facet\_grid(~Model)



Of course that in practice prices does not change between 1 and 100, but the idea is to show the main differences in the shape of the models.

All the models presented above have positive and negative points. Although local linear approximation may be reasonable for small changes in prices, sometimes this assumption is too strong and does not capture the correct sensitivity of bigger price changes. In the constant elasticity model, even though it is a non-linear relationship between demand and price, the constant elasticity assumption might be too restrictive. Moreover, it tends to over estimate the demand for lower and bigger prices. In a fist moment, I would venture to say that the logistic function is the most robust and realistic among the three types.

**Pricing with demand models**

In a general setting, one have for the total profit function:

\displaystyle L(p) = d(p)(p-c),

where, Lgives the profit, dis the demand function that depends of the price and cis the marginal cost. Taking the derivative with respect to price we have:

\displaystyle L'(p) = d'(p)(p - c) + d(p).

Making L'(p) = 0to calculate the optimum price (first order condition), we have:

\displaystyle d'(p^\star)(p^\star - c) + d(p^\star) = 0  
\displaystyle d'(p^\star)p^\star + d(p^\star) = d'(p^\star)c,

which is the famous condition that in the optimal price, marginal cost equals marginal revenue. Next, let’s see how to calculate the optimum prices for each demand functions.

**Linear model**

For the linear model d'(p) = \alpha. Hence:

\displaystyle d'(p^\star)p + d(p^\star) = d'(p^\star)c,  
\displaystyle \alpha p^\star + \alpha p^\star + \beta = \alpha c,  
\displaystyle p^\star = \frac{\alpha c - \beta}{2\alpha}.

Example:

library(tidyverse)

# Synthetic data

p = seq(80,130)

d = linear(p, alpha = -1.5, beta = 200) + rnorm(sd = 5, length(p))

c = 75

profit = d\*(p-c)

# Fit of the demand model

model1 = lm(d~p)

profit.fitted = model1$fitted.values\*(p - c)

# Pricing Optimization

alpha = model1$coefficients[2]

beta = model1$coefficients[1]

p.max.profit = (alpha\*c - beta)/(2\*alpha)

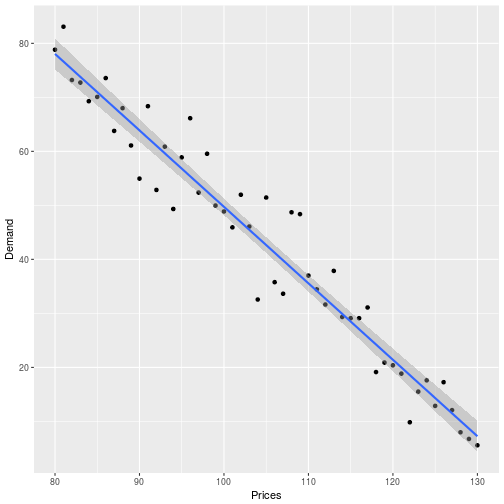
# Plots

df.linear = data.frame('Prices' = p, 'Demand' = d,

'Profit.fitted' = profit.fitted, 'Profit' = profit)

ggplot(select(df.linear, Prices, Demand)) + aes(x = Prices, y = Demand) +

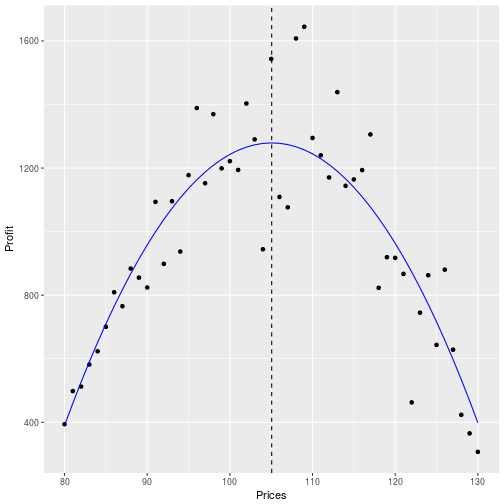
geom\_point() + geom\_smooth(method = lm)



ggplot(select(df.linear, Prices, Profit)) + aes(x = Prices, y = Profit) +

geom\_point() + geom\_vline(xintercept = p.max.profit, lty = 2) +

geom\_line(data = df.linear, aes(x = Prices, y = Profit.fitted), color = 'blue')



**Constant elasticity model**

For the constant elasticity model, since \lim_{\Delta \rightarrow 0}\frac{\Delta D}{\Delta p} = d'(p), we have that:

\displaystyle \epsilon = \frac{\%D}{\%p} = \frac{p\Delta D}{D\Delta p} = -\frac{d'(p)p}{D}.

Therefore,

\displaystyle d'(p^\star)p^\star + d(p^\star) = d'(p^\star)c,  
\displaystyle \frac{d'(p^\star)p^\star}{d(p^\star)} + 1 = \frac{d'(p^\star)c}{d(p^\star)},  
\displaystyle -\epsilon + 1 = \epsilon \frac{c}{p^\star},  
\displaystyle p^\star = \frac{\epsilon c}{1-\epsilon} = \frac{c}{1-1/\epsilon}.

Moreover, knowing that \frac{\%D}{\%p} \sim \frac{\Delta \ln D}{\Delta \ln p}and using the constant elasticity model, we have that:

\displaystyle \epsilon \sim \lim_{\Delta \rightarrow0} \frac{\Delta \ln D}{\Delta \ln P} = \frac{d\ln D}{d\ln p} = \alpha.

Thus, we can calculate the optimum profit price for the constant elasticity model as:

\displaystyle p^\star = \frac{c}{1 - \frac{1}{|\alpha|}}

It is interesting to note that one needs |\alpha| > 1, otherwise the profit function will be convex with respect to price and the optimal price will be \infty. If one have a monopolistic market, normally this assumption holds.

Example:

# Synthetic data

p = seq(80,130)

d = constant\_elast(p, alpha = -3, beta = 15)\*exp(rnorm(sd = .15, length(p)))

c = 75

profit = d\*(p-c)

# Fitting of demand model

model2 = lm(log(d)~log(p))

profit.fitted = exp(model2$fitted.values)\*(p - c)

# pricing optimization

alpha = model2$coefficients[2]

p.max.profit = c/(1-1/abs(alpha))

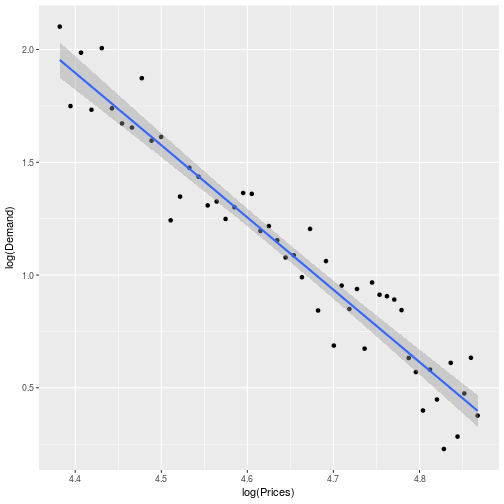
# Plots

df.const\_elast = data.frame('Prices' = p, 'Demand' = d,

'Profit.fitted' = profit.fitted, 'Profit' = profit)

ggplot(select(df.const\_elast, Prices, Demand)) + aes(x = log(Prices), y = log(Demand)) +

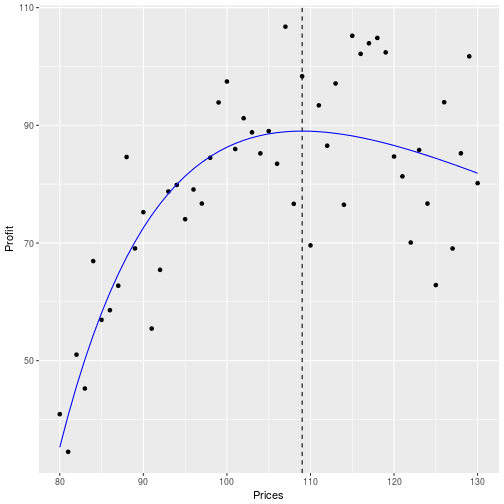
geom\_point() + geom\_smooth(method = lm)



ggplot(select(df.const\_elast, Prices, Profit)) + aes(x = Prices, y = Profit) +

geom\_point() + geom\_vline(xintercept = p.max.profit, lty = 2) +

geom\_line(data = df.const\_elast, aes(x = Prices, y = Profit.fitted), color = 'blue')



**Logistic model**

For the logistic function, one can check that d'(p) = \alpha d(p)(1-d(p)/C). Thus:

\displaystyle d'(p^\star)(p^\star - c) + d(p^\star) = 0,  
\displaystyle \alpha d(p^\star)(1-d(p^\star)/C)(p^\star-c) + d(p^\star) = 0,  
\displaystyle \alpha(1-d(p^\star)/C)(p^\star-c) + 1 = 0,  
\displaystyle \frac{\alpha e^{-\alpha(p^\star - p_0)}(p^\star - c) + 1+ e^{-\alpha(p^\star - p_0)}}{1+ e^{-\alpha(p^\star - p_0)}} = 0,  
\displaystyle \alpha(p^\star-c)+1]e^{-\alpha(p^\star - p_0)} + 1 = 0.

Since the last equation above does not have an analytical solution (at least we couldn’t solve it), one can easily find the result with a newton-step algorithm or minimization problem. We will use the second approach with the following formulation:

\displaystyle \min_{p \in \mathbb{R}} \big{(}[\alpha(p-c)+1]e^{-\alpha(p - p_0)} + 1\big{)}^2

Example:

# Objective functions for optimization

demand\_objective = function(par, p, d) sum((d - logistic(p, par[1], par[2], par[3]))^2)

price\_objective = function(p, alpha, c, p0) (exp(-alpha\*(p-p0))\*(alpha\*(p-c)+1) + 1)^2

# A cleaner alternative for pricing optimization is to min:

price\_objective2 = function(p, c, alpha, C, p0) -logistic(p, C, alpha, p0)\*(p-c)

# synthetic data

p = seq(80,130)

c = 75

d = logistic(p, 120, -.15, 115) + rnorm(sd = 10, length(p))

profit = d\*(p-c)

# Demand fitting, we can't use lm anymore

par.start = c(max(d), 0, mean(d)) # initial guess

demand\_fit = optim(par = par.start, fn = demand\_objective, method = 'BFGS',

p = p, d = d)

par = demand\_fit$par # estimated parameters for demand function

demand.fitted = logistic(p, c = par[1], alpha = par[2], p0 = par[3])

profit.fitted = demand.fitted\*(p - c)

# Pricing Optimization, we don't have a closed expression anymore

price\_fit = optim(mean(p), price\_objective, method = 'BFGS',

alpha = par[2], c = c, p0 = par[3])

# or

price\_fit2 = optim(mean(p), price\_objective2, method = 'BFGS',

c = c, C = par[1], alpha = par[2], p0 = par[3])

# both results are almost identical

p.max.profit = price\_fit$par

# Graphics

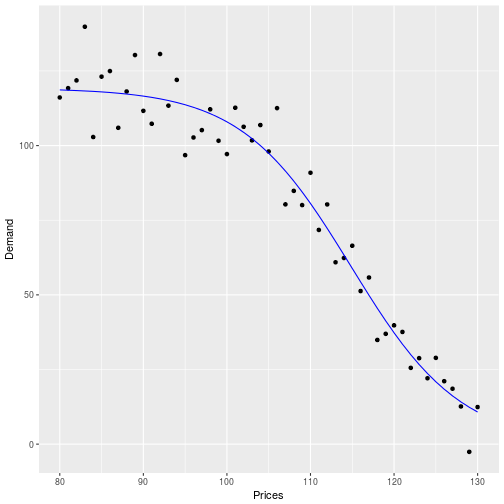
df.logistic = data.frame('Prices' = p, 'Demand' = d, 'Demand.fitted' = demand.fitted,

'Profit.fitted' = profit.fitted, 'Profit' = profit)

ggplot(select(df.logistic, Prices, Demand)) + aes(x = Prices, y = Demand) +

geom\_point() +

geom\_line(data = df.logistic, aes(x = Prices, y = Demand.fitted), color = 'blue')



ggplot(select(df.logistic, Prices, Profit)) + aes(x = Prices, y = Profit) +

geom\_point() + geom\_vline(xintercept = p.max.profit, lty = 2) +

geom\_line(data = df.logistic, aes(x = Prices, y = Profit.fitted), color = 'blue')

