Case 6: Data Driven Decision Making and Soup Pricing

Dr. Avery Haviv

Winter 2020

Introduction

In this case we will apply a simple demand model to a pricing problem. The reason we care so much about causal analysis is that it allows us to make a business decisions. Here, we will show how to use our model to set prices, and how omitted variables can ruin any decision the firm tries to make with statistics.

Making a pricing decision based on a causal model

For simplicity, I have taken a subset of the data corresponding to the most popular product, which had a `productNum` of 89. First, we load the data and run our demand model, which will control for prices, whether the product is featured or displayed, and the store.

```
setwd('D:/Dropbox/Teaching Lectures/Soup Pricing Case')
caseData = read.csv('Soup Pricing Case Data.csv')
demandModel = lm(log(units)~pricePerCan+isFeature+isDisplay+factor(storeNum),data=caseData)
summary(demandModel)
```

```
##
## Call:
## lm(formula = log(units) ~ pricePerCan + isFeature + isDisplay +
##     factor(storeNum), data = caseData)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7041 -0.5272  0.0455  0.5191  3.6329
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)          2.35743    0.12645  18.643  < 2e-16 ***
## pricePerCan         -1.38687    0.15481  -8.959  < 2e-16 ***
## isFeatureTRUE        0.48988    0.05404   9.065  < 2e-16 ***
## isDisplayTRUE        0.49474    0.05268   9.391  < 2e-16 ***
## factor(storeNum)2   -0.43586    0.07333  -5.944 3.10e-09 ***
## factor(storeNum)3    1.31008    0.06599  19.853  < 2e-16 ***
## factor(storeNum)4   -0.50172    0.07331  -6.844 9.28e-12 ***
## factor(storeNum)5    0.68892    0.06941   9.926  < 2e-16 ***
## factor(storeNum)6   -0.27525    0.11977  -2.298 0.021620 *
## factor(storeNum)7    0.80477    0.06758  11.909  < 2e-16 ***
## factor(storeNum)8   -0.39387    0.08388  -4.696 2.77e-06 ***
## factor(storeNum)9    0.18724    0.06885   2.720 0.006575 **
## factor(storeNum)10  -0.37528    0.07971  -4.708 2.62e-06 ***
## factor(storeNum)11  -0.60233    0.08516  -7.073 1.87e-12 ***
## factor(storeNum)12   0.89247    0.06676  13.368  < 2e-16 ***
## factor(storeNum)13   1.19983    0.06608  18.157  < 2e-16 ***
## factor(storeNum)14   0.28943    0.08096   3.575 0.000356 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7407 on 3008 degrees of freedom
## Multiple R-squared:  0.5248, Adjusted R-squared:  0.5222
## F-statistic: 207.6 on 16 and 3008 DF,  p-value: < 2.2e-16
```

To calculate the optimal price, we will need to generate the expected profit for each potential price the store might charge. Below, I construct a data frame that will store these prices, and the resulting profits. We will be using this data frame to get predictions of demand and profit margin. To get predictions, you need to be *very careful* about how you name the variables. All variable names used in estimating the model must be present in the data frame.

I set `pricePerCan` to take on each value between 0 and 4. The variables `isFeature` and `isDisplay` have been set to FALSE. This means we are predicting demand and optimizing prices when there are no features and displays. If we wanted to price this product when the product was featured and displayed, we would set those variables to TRUE here. I set the store to be the 3rd store.

```
possiblePrices = data.frame(pricePerCan=seq(0,4,.01),isFeature=FALSE,isDisplay=FALSE,storeNum=3)
```

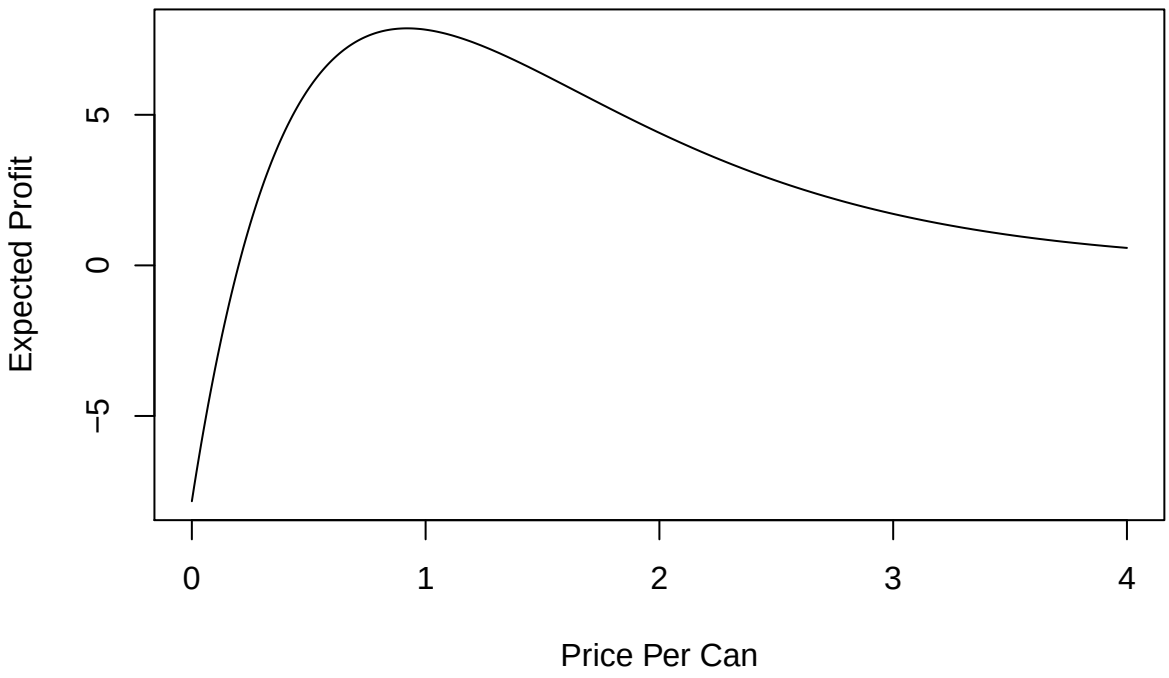
Ultimately the firm cares about profits. Because profit = demand * profit margin, we first calculate the expected demand (according to our demand model) and profit margin for each possible price. This is *exactly* why we care about demand models so much: once we have a good demand model, profits and ultimately pricing are simple calculations.

Note that since our regression predicts log units (rather than units), we have to take the exponential of the prediction from `demandModel`.

```
possiblePrices$predictedDemand = exp(predict(demandModel,possiblePrices))
possiblePrices$profitMargin = possiblePrices$pricePerCan - .2
possiblePrices$profit = possiblePrices$profitMargin * possiblePrices$predictedDemand
```

Now that we have predicted profit for each potential price the firm could charge for this soup, we optimize prices by finding the price that maximizes profit. We can do this using the `which.max` function.

```
plot(possiblePrices$pricePerCan,possiblePrices$profit,xlab='Price Per Can'
     ,ylab='Expected Profit',type='l')
```




```
#What price maximized profit? We can use the which.max function
possiblePrices[which.max(possiblePrices$profit),]
```

```
##    pricePerCan isFeature isDisplay storeNum predictedDemand profitMargin
## 93        0.92     FALSE     FALSE        3        10.93088         0.72
##      profit
## 93 7.870232
```

So, according to this demand model, the optimal price for this can of soup is 92 cents, at which point the store will earn roughly 8 dollars a week for this flavor. The plot of profits makes some intuitive sense. If prices are below marginal costs of .2, the firm loses money on each unit of soup they sell.[1]

1. Suppose we thought that consumers did not respond to price changes. Would we charge high prices or low prices?
2. Suppose we thought that consumers were extremely sensitive to price changes. Would we charge high prices or low prices?

Interactions with features and displays

Interactions between price, feature, and display have important effects. Demand for products that are featured/displayed is more responsive to changes in price. In this next code segment, we will account for these interactions while calculating the optimal price by incorporating them in the demand model.

[1] Formally speaking, prices were optimized with a grid based optimizer. That works well when we are only setting one price, but if you try to optimize multiple prices at the same time it can be very slow. In that case you should look at the `optim` function, which is similar to solver in excel. Feel free to email me if you want my slides on this topic!

```
demandModel = lm(log(units)~pricePerCan*(isFeature+isDisplay)+factor(storeNum),data=caseData)
summary(demandModel)
```

```
##
## Call:
## lm(formula = log(units) ~ pricePerCan * (isFeature + isDisplay) +
##     factor(storeNum), data = caseData)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7841 -0.5210  0.0377  0.5100  3.6256
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  2.20906    0.13370  16.522  < 2e-16 ***
## pricePerCan                 -1.20453    0.16348  -7.368 2.23e-13 ***
## isFeatureTRUE                1.53565    0.30210   5.083 3.94e-07 ***
## isDisplayTRUE                0.83689    0.32864   2.547 0.010930 *
## factor(storeNum)2           -0.42684    0.07353  -5.805 7.10e-09 ***
## factor(storeNum)3            1.32549    0.06605  20.068  < 2e-16 ***
## factor(storeNum)4           -0.48685    0.07334  -6.638 3.75e-11 ***
## factor(storeNum)5            0.70539    0.06946  10.155  < 2e-16 ***
## factor(storeNum)6           -0.29408    0.11962  -2.458 0.014014 *
## factor(storeNum)7            0.81320    0.06755  12.038  < 2e-16 ***
## factor(storeNum)8           -0.40893    0.08384  -4.877 1.13e-06 ***
## factor(storeNum)9            0.19871    0.06882   2.888 0.003910 **
## factor(storeNum)10          -0.38521    0.07962  -4.838 1.37e-06 ***
## factor(storeNum)11          -0.61602    0.08509  -7.239 5.69e-13 ***
## factor(storeNum)12           0.90951    0.06688  13.599  < 2e-16 ***
## factor(storeNum)13           1.21436    0.06610  18.371  < 2e-16 ***
## factor(storeNum)14           0.30652    0.08126   3.772 0.000165 ***
## pricePerCan:isFeatureTRUE   -1.57218    0.44735  -3.514 0.000447 ***
## pricePerCan:isDisplayTRUE   -0.50711    0.48565  -1.044 0.296481
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7392 on 3006 degrees of freedom
## Multiple R-squared:  0.5271, Adjusted R-squared:  0.5242
## F-statistic: 186.1 on 18 and 3006 DF,  p-value: < 2.2e-16
```

```
possiblePrices = data.frame(pricePerCan=seq(0,10,.01),isFeature=FALSE,isDisplay = FALSE,storeNum = 3)
possiblePrices$predictedDemand = exp(predict(demandModel,possiblePrices))
possiblePrices$profitMargin = possiblePrices$pricePerCan - .2
possiblePrices$profit = possiblePrices$profitMargin * possiblePrices$predictedDemand
possiblePrices$pricePerCan[which.max(possiblePrices$profit)]
```

```
## [1] 1.03
```

Just as in the homework, when you include interactions between features, displays, and prices, prices becomes *less* important in the baseline scenario. That is, in this regression, the baseline coefficient on `pricePerCan` is -1.20453, whereas in the previous analysis the coefficient was -1.38687. This is because the baseline scenario no longer contains features or displays, where consumers are more price sensitive.

Because consumers are less responsive to price changes, the optimal price in this analysis increased by 11 cents to 1.03. Improving the demand model leads to a different price!

Discussion Questions:

1. The above code *controlled for* features and displays, but it did not figure out the optimal price when a feature or display is on. Do you expect the optimial price to be higher or lower when the product is featured? What in the analysis tells you that?
2. Modify the code to calculate the optimal price when a feature or display is present

Featured Price

One great thing about pricing using a data driven process is that it is relatively easy to scale up. If we instead wanted the prices when a product is featured, we could only have to slightly modify the above code by setting `isFeature` to TRUE in our `possiblePrices` dataframe.

```
possiblePrices = data.frame(pricePerCan=seq(0,4,.01),isFeature=TRUE,isDisplay=FALSE,storeNum=3)
possiblePrices$predictedDemand = exp(predict(demandModel,possiblePrices))
possiblePrices$profitMargin = possiblePrices$pricePerCan - .2
possiblePrices$profit = possiblePrices$profitMargin * possiblePrices$predictedDemand
possiblePrices$pricePerCan[which.max(possiblePrices$profit)]
```

```
## [1] 0.56
```

The optimal price is now 56 cents, which is much lower than before. This is expected, as consumers who respond to features are more price sensitive.

Pricing and Omitted Variables Bias

On the other hand, if we stop controlling for features and displays, the price coefficient will be biased downwards, making prices seem more important than they actually are. This in turn, will mean the firm tries to charge a price that is too low. We can see this by removing `isFeature` and `isDisplay` from the demand model.

```
demandModel = lm(log(units)~pricePerCan,data=caseData)
possiblePrices = data.frame(pricePerCan=seq(0,10,.01),storeNum = 3)
possiblePrices$predictedDemand = exp(predict(demandModel,possiblePrices))
possiblePrices$profitMargin = possiblePrices$pricePerCan - .2
possiblePrices$profit = possiblePrices$profitMargin * possiblePrices$predictedDemand
possiblePrices$pricePerCan[which.max(possiblePrices$profit)]
```

```
## [1] 0.47
```

The price dropped from 1.03 to 47 cents! This should tell you how serious an issue omitted variable bias can be in making business decisions, which is why I've emphasized this topic. A firm using this seemingly reasonable but ultimately incorrect demand model would get an optimal price that is not even close to the best answer. This should teach you to think carefully about other factors that might affect demand - the model we've estimated can certainly be improved in a number of ways! The follow up course in Pricing Analytics will tackle some improvements that can be made.

Discussion Questions:

1. In our analysis in the previous section, we did not control for time of year. If the price of soup is higher during winter, how will our estimate of the price coefficient be biased? Given that bias, will we get too low of a price or too high of a price?
2. What if instead prices are lower during winter?

Neural Networks and Functional Forms

A complete treatment of neural networks is beyond the scope of this class[2]. However, I want to demonstrate that *any* statistical model, no matter how fancy, is subject to omitted variable bias. Below, I estimate a neural network in R, and use this as a demand model for our price optimization. Notice that the syntax for estimating this model is very similar to the regression syntax.

```
#install.packages('nnet')
library('nnet')
```

```
## Warning: package 'nnet' was built under R version 3.5.3
```

```
set.seed(238)
caseData$isFeature = caseData$isFeature*1
caseData$isDisplay = caseData$isDisplay*1

demandModel = nnet(log(units)~pricePerCan*(isFeature+isDisplay),data=caseData,subset=storeNum==3,
                   size=4,linout=1,maxit=100000,trace=FALSE)
possiblePrices = data.frame(pricePerCan=seq(0,1,.01),isFeature=0,isDisplay = 0,storeNum = 3)
possiblePrices$predictedDemand = (predict(demandModel,possiblePrices))
possiblePrices$profitMargin = possiblePrices$pricePerCan - .2
possiblePrices$profit = possiblePrices$profitMargin * possiblePrices$predictedDemand
possiblePrices$pricePerCan[which.max(possiblePrices$profit)]
```

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

```
set.seed(238)
demandModel2 = nnet(log(units)~pricePerCan,data=caseData,subset=storeNum==3,
                    size=4,linout=1,maxit=100000,trace=FALSE)
possiblePrices = data.frame(pricePerCan=seq(0,2,.01),isFeature=FALSE,isDisplay = FALSE,storeNum = 3)
possiblePrices$predictedDemand = (predict(demandModel2,possiblePrices))
possiblePrices$profitMargin = possiblePrices$pricePerCan - .2
possiblePrices$profit = possiblePrices$profitMargin * possiblePrices$predictedDemand
possiblePrices$pricePerCan[which.max(possiblePrices$profit)]
```

```
## [1] 0.58
```

Again, when you don't control for features and displays, the optimal price drops by nearly a factor of two. Omitted variable bias affect all types of models, and it leads to systematically bad decisions.

[2] I am however happy to provide my presentation slides and potentially a previous video. Email me if interested.