Review Session (MGTA456): Week 1

Simple Newsvendor Problem and Its Implementation in R

Kohei Hayashida

University of California San Diego, Rady School of Management

2024-04-08

Logistics

- Slide available on Canvas after each session
- TA RS
  - Tuesday (every week) 6pm-7pm on zoom (link on Canvas)
- TA Office Hours
  - Every Friday 6pm-7pm on zoom (link on Canvas)

Outline

- Setup of a simple Inventory Problem
- How to solve the Inventory Problem?
  - Analytical
  - Numerical/Simulation-based
- Recap of Optimization
- Coding Practice of the Inventory Problem

[1] Setup of Inventory Problem (Cupcake Problem)

- Suppose you are a manager of the bakery store.
- The stocks need to be determined before the store opens.
- Then, customers arrive and buy them if they want.
- The manager wants to maximize the profit by optimizing the number of stocks.

Assumption (to simplify the analysis)

- Suppose demand is known perfectly.
  - This assumption will be relaxed by introducing forecasting.
- Price is fixed.
  - This is a fair assumption if you think about the short-run problem.
  - But, this is too restrictive in the long run.
  - We can relax this in pricing section.
- The cost of producing and discounting are given.

Setup

- Exogenously given parameters:
  - price r: \$2.49/unit
  - cost c: \$1.24/unit
  - salvage s: \$0.99/unit
- Data
  - Demand: D: unit/day
- Control variable:
  - The number of stocks: q: unit/day

The profit π to maximize:

$$\pi(q) = r \min\{D, q\} + s \max\{q - D, 0\} - cq$$

The problem to solve:

$$\max_q \pi(q)$$

[2] How to Solve?

Approaches:

- *Analytically* solve profit maximization problem.
  - pros: Always accurate and faster.
  - cons: Not applicable to complex setting.
- *Numerically* solve profit maximization problem.
  - pros: Easy to implement and applicable to most problems
  - cons: took time, can be inaccurate than the analytical solution.
  - Implementation:
    - Use Solver
    - Rely on Brute Force Method

Fortunately, we know there is a beautiful analytical solution in this Newsvendor problem.

Analytical Approach to the Inventory Problem

1. One way is to take the derivative (FOC) with respect to q.
2. Another approach is intuitive one taught in class.

Let's say you prepare q units/day.

- $Cu = r - c$: the cost of understocking
- $Co = c - s$: the cost of overstocking
- The problem can be rewritten as $Pr(D > q) * Cu > Prob(D <= q) * Co$, where $Prob(D <= q)$ is called service level (SL).
- If $(1 - SL) * Cu > SL * Co$, you want to increase the q.
- We want to know the target service level that equates the inequality.

$$(1 - SL) \times Cu = SL \times Co$$

In this case, $SL = \frac{Cu}{(Co+Cu)} = 0.83$.

Optimal Inventory

- Using the target SL, the optimal inventory is taken at the point you satisfy the demand distribution.

In other words,

$$q^* = F^{-1}(TSL)$$

where

$$TSL = \frac{Cu}{Cu + Co} = \frac{r - c}{(r - c) + (c - s)} = \frac{r - c}{r - s},$$

and F is the CDF of demand.

- Thus, what you need to do is just calculate SL and F from exogenously given information.
- Then, take the quantile at SL to get the optimal stocking.

How to get the CDF from the demand data?

We will review in the coding part. But, the basic idea is as follows.

- The pdf is the probability distribution.
- You can always transform PDF into CDF.

In the current setting (the CDF is already given in the class slide),

- Target Service Level: $SL = \frac{r-c}{r-s} = \frac{2.49-1.24}{2.49-0.99} = 0.833$
- What is the quantile of CDF of demand distribution at around 83.3 percent?

Numerical Approach to the Inventory Problem

Brute-Force Method (Grid-search)

- This is just one way of doing it.
- Remember the objective function and the control variables.

The problem to solve:

$$\max_q r \min\{D, q\} + s \max\{q - D, 0\} - cq$$

- By changing the control variables, determine the highest value of the objective function [here, profit].
- Try many different values of the control variables to guarantee you reach the best solution.
- The solution **should** match with the analytical one.

Reminder

In the future class, you will learn a more general case such as below:

- You have uncertainty in Demand due to the forecasting error.
- You may also want to optimize the price at the same time.

[3] Some general notes on Optimization

- Curvature of the objective function matters.
- Solution may not exist.
- Always check if your objective function is decent.
- If you defined/coded a wrong objective function, no hope to find the best solution.
- If you encounter the issue in finding the best solution in this simple setup, it is likely that you have some bugs in this part.

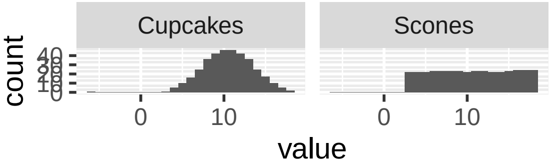
[4] Coding Practice

- General tips to learn how to code:
    - **Do not see the solution first.**
    - But, do not waste too much of your time.
    - Plan ahead staying away from the laptop.
    - Often, there are several ways to reach to the same solution.
- General tips to do data analytics:
    - Check the data first before starting analysis.
        - Missing values, strange values, something that beats your intuition.
    - Keep the hypothesis and verify it step by step.[Do not rush!]

Descriptive Analysis

- Prepare Rstudio and load general packages to use
- Load csv/txt file using `read.csv`

```
demand_df <- read.csv("../data/demand_data_session1_2022.txt", sep = "\t")

# Visualize data
demand_df %>%
  tidyr::pivot_longer(cols = c(Cupcakes, Scones)) %>%
  ggplot(aes(x = value)) +
  geom_histogram(binwidth = 1) + # feel free to change the binwidth if you don't like
  facet_wrap(~name)
```



```r
st(demand_df, out = "latex")
```

Table 1: Summary Statistics

Variable	N	Mean	Std. Dev.	Min	Pctl. 25	Pctl. 75	Max
Cupcakes	365	10	3.1	-6	8	13	18
Scones	365	11	4.6	3	7	15	18

```r
paste0("Number of Missing Values:", sum(is.na(demand_df$Demand)))
```

[1] "Number of Missing Values:0"

Dealing with the strange values

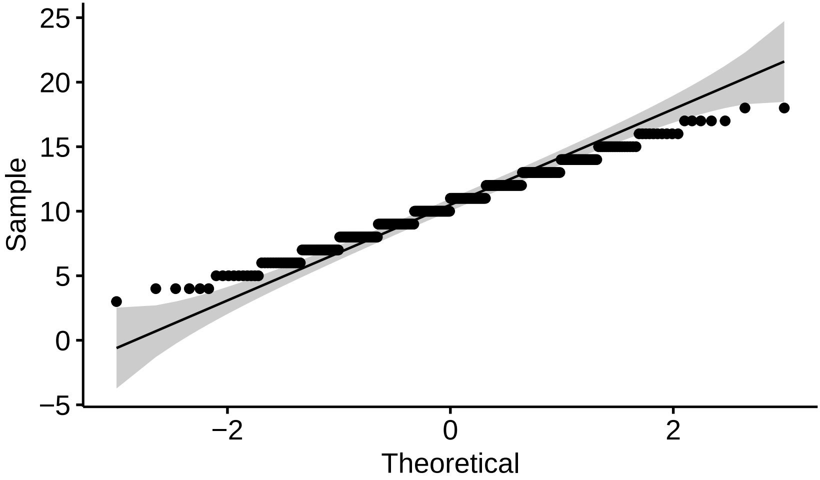
- Removing the observation
- Changing the sign
- Imputing with mean, etc.

```
demand_df$Demand <- demand_df$Cupcakes
demand_df <- demand_df %>%
  mutate(Demand = ifelse(Demand < 0, -Demand, Demand))

st(demand_df, out = "latex")
```

Table 2: Summary Statistics

Variable	N	Mean	Std. Dev.	Min	Pctl. 25	Pctl. 75	Max
Cupcakes	365	10	3.1	-6	8	13	18
Scones	365	11	4.6	3	7	15	18
Demand	365	11	2.9	3	8	13	18

```
ggqqplot(demand_df$Demand)
```



- We usually do not know how data are generated.
- Making distributional assumption to make some calculation easily. [See Wikipedia in the case of Normal, LogNormal]
- The method we are discussing (both analytical one and brute-force one) is non-parametric (meaning, distributional-assumption free)

Set Exogenously Given Values as Constant

```
price <- 2.49
salvage <- 0.99
cost <- 1.24
```

Set the search range for stockings

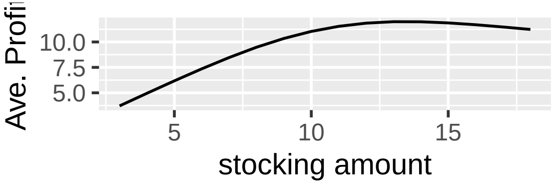
```
stocking <- c(3:18)
```

Try each value of stocking and calculate the profit

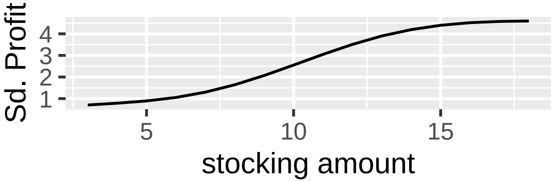
```
# using loop
profit <- matrix(0, nrow = nrow(demand_df), ncol = length(stocking))
for (i in 1:length(stocking)) { # loop for each stocking level
  q <- stocking[i]

  for (j in 1:nrow(demand_df)) { # loop for each realized demand
    profit[j,i] <- price * min(q, demand_df[j,1]) + salvage * max(q - demand_df[j,1], 0) - cost * q
  }
}
# make summary table
summary_table <- data.frame(
  stocking = stocking,
  AvgProf = apply(profit, 2, mean), # column means
  StdProf = apply(profit, 2, sd) # column sd
)
kableExtra::kable(summary_table, format = "latex", digits = 3, align = 'c')
```

stocking	AvgProf	StdProf
3	3.713	0.707
4	4.955	0.789
5	6.176	0.893
6	7.356	1.054
7	8.471	1.300
8	9.482	1.643
9	10.346	2.070
10	11.037	2.553
11	11.539	3.048
12	11.852	3.509
13	11.992	3.901
14	11.985	4.194
15	11.875	4.396
16	11.695	4.516
17	11.473	4.573

```
summary_table %>%
  ggplot(aes(x = stocking, y = AvgProf)) +
  geom_line() +
  xlab("stocking amount") +
  ylab("Ave. Profit")
```




```
summary_table %>%
  ggplot(aes(x = stocking, y = StdProf)) +
  geom_line() +
  xlab("stocking amount") +
  ylab("Sd. Profit")
```



Safety Stock

```r
optimal_profit <- max(summary_table$AvgProf)
optimal_stocking <- summary_table$stocking[which.max(summary_table$AvgProf)]
safety_stock <- optimal_stocking - mean(demand_df$Demand)
print(safety_stock)
```

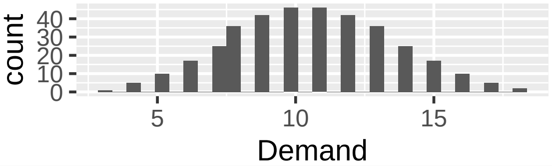
```
## [1] 2.479452
```

Sanity Check

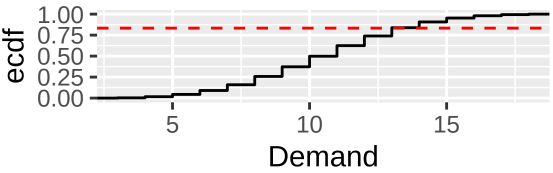
This optimal inventory should matche with the analytical solution

```
demand_df %>% ggplot(aes(Demand)) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
TSL <- (price - cost)/(price - salvage)
demand_df %>% ggplot(aes(Demand)) + stat_ecdf(geom = "step") +
  geom_hline(yintercept = TSL, linetype="dashed", color = "red")
```



```r
quantile(demand_df$Demand, TSL)
```

```
## 83.33333%
##        13
```

Value of Oracle

```r
value_of_oracle <- (price - cost) * mean(demand_df$Demand) - optimal_profit
print(value_of_oracle)
```

```
## [1] 1.158219
```

```r
# Note: the below is the same thing.
value_of_oracle <- mean((price - cost) * demand_df$Demand) - optimal_profit
print(value_of_oracle)
```

```
## [1] 1.158219
```

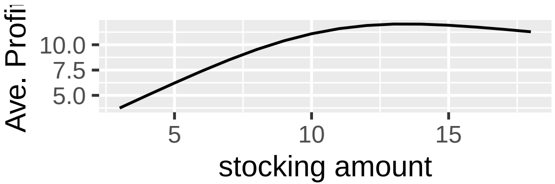
without using loop

```r
# prep grid table
grid_table <- tidyr::expand_grid(demand = demand_df$Demand, stocking = stocking)
head(grid_table)
```

```
## # A tibble: 6 x 2
##   demand stocking
##    <int>    <int>
## 1     12        3
## 2     12        4
## 3     12        5
## 4     12        6
## 5     12        7
## 6     12        8
```

```r
calculate_profit <- function(demand, stocking) {
  price <- 2.49
  salvage <- 0.99
  cost <- 1.24
  q <- stocking

  profit <- price * min(q, demand) + salvage * max(q - demand, 0) - cost * q
  return(profit)
}
grid_table <- grid_table %>%
  dplyr::rowwise() %>%
  # calculate profit for each combination of demand and stocking
  dplyr::mutate(profit = calculate_profit(demand, stocking)) %>%
  ungroup()
```

```r
grid_table %>%
  group_by(stocking) %>%
  summarise(AvgProf = mean(profit)) %>%
  ungroup() %>%
  ggplot(aes(x = stocking, y = AvgProf)) +
  geom_line() +
  xlab("stocking amount") +
  ylab("Ave. Profit")
```




```r
grid_table %>%
  dplyr::group_by(stocking) %>%
  dplyr::summarise(profit = mean(profit)) %>%
  dplyr::ungroup() %>%
  dplyr::arrange(desc(profit)) %>%
  head(5)
```

```
## # A tibble: 5 x 2
##   stocking profit
##      <int>  <dbl>
## 1       13   12.0
## 2       14   12.0
## 3       15   11.9
## 4       12   11.9
## 5       16   11.7
```

Speed Comparison

with loop

```
## 2.91 sec elapsed
```

without loop

```
## 1.039 sec elapsed
```