456 Supply Chain HW 1

Brian Gao

2024-04-19

Table of contents

0.1 Question 1

Consider that the publisher, Scribner, acquired your bookstore, BN, i.e., you manage the whole supply chain, both the Publisher and the bookstore. In this case, what is the cost of understocking, the cost of overstocking, and the target service level of the supply chain (Note that the supply chain here means the Publisher and the bookstore)? How many books the supply chain should print? In this case, what is the supply chain average profit based on the data?

- **cost of understocking(cu)**

  - The lost profit for each book not sold when there is demand is essentially the retail price minus the wholesale price (since if we had the book, we could have sold it at retail price and replenished at wholesale price).
  - *Cu=Retail Price−Wholesale Price=**$22−$2=$20** per book*

- **cost of overstocking(co)**

  - The cost of each unsold book is the cost of printing because once you've purchased the book, if not sold, the loss is what was paid to print it (considering you manage both the printing/publishing and the selling as one entity now).
  - *Co=Cost to Publisher to print =**$2** per book*

- **Target Service Level (Service Factor)**

– The target service level determines the probability that there will be no stockouts, which depends on how the costs of overstocking and understocking relate. The critical ratio (CR) is used to determine the target service level and is defined as:

$$\left(\frac{Cu}{Cu + Co}\right)$$

– here, Cu=$20, and Co=$2, **CR=20/(20+2) ≈ 0.9091**

```
# Constants
P = 22   # Selling price per book
Co = 2    # Cost of production per book (real cost outlay)
W = 14   # Wholesale price per book (internal transfer, not considered here)

# Given values
mu = 20000   # Approximate mean from data
sigma = 2000   # Standard deviation from data
z = 1.34   # CR

# Calculate the optimal order quantity.
Q_print = mu + z * sigma

profits = []
for demand in demand_data['Demand']:
    sold_books = min(demand, Q_print)
    unsold_books = max(Q_print - demand, 0)

    # Total profit calculation
    profit = sold_books * (P - Co) - unsold_books * Co
    profits.append(profit)

# Average profit calculation
average_profit = sum(profits) / len(profits)
(Q_print, average_profit)
```

(22680.0, 393106.072)

Therefore, based on the data, the supply chain should **print 22,680 books** and the supply chain average profit is **approximately $393,106**.

0.2 Question 2

Consider the original case that we studied in class, i.e., the bookstore is BN, and the Publisher is a separate entity. You, as a bookstore owner, decides how many books to order considering the wholesale price and the buy-back price. We discussed the buy-back contract, in which the buy-back price was $12 in class.

Can you find the pair of the wholesale price (note that the status-quo wholesale price is $14) and the buy-back price that will make both the bookstore and the publisher better off compared to the price pair of the wholesale price $14 and the buy-back price $12? As you change the wholesale price and the buy-back price, the order quantity of books will change.

- **First, Calculating the average profits at $14 wholesale and $12 buy-back.**

```python
import scipy.stats as stats
# Constants
P = 22   # Selling price per book
Co = 2   # Cost of overstocking by bookstore
Cu = 8   # Cost of understocking by bookstore
bb = 12 # buy back price
W = 14   # wholesale price
# Given values
mu = 20000   # Approximate mean from data
sigma = 2000   # Standard deviation from data
z_hc = stats.norm.ppf(Cu/(Co+Cu))   # CR

# Calculate the optimal order quantity.
Q = mu + z_hc * sigma

profits_hc = []
profits_bn = []
for demand in demand_data['Demand']:
    sold_books = min(demand, Q)
    unsold_books = max(Q - demand, 0)

    profit_hc = Q * (W-2) - unsold_books * bb
    profit_bn = sold_books * P - 14 * Q + unsold_books * bb
    profits_hc.append(profit_hc)
    profits_bn.append(profit_bn)
# Average profit calculation
average_profit_hc = sum(profits_hc) / len(profits_hc)
average_profit_bn = sum(profits_bn) / len(profits_bn)
```

```
print('average_profit_hc:$', average_profit_hc, 'average_profit_bn:$', average_profit_bn)
```

average_profit_hc:$ 237348.7659324771 average_profit_bn:$ 154424.15334277204

- **Optimizing by using loops and printing out profits greater than \$14 & \$12 pair.**

```python
import numpy as np
from scipy.stats import norm

P = 22  # Selling price per book
mu = 20000  # Mean demand
sigma = 2000  # Standard deviation of demand

# Explore a range of wholesale (W) and buyback (bb) prices to maximize profits
wholesale_prices = np.linspace(10, 15, 50)  # Range from $12 to $18
buyback_prices = np.linspace(9, 14, 50)    # Range from $5 to $12

# Dictionary to store the results for easier comparison
result_summary = []

# Calculate profits for each combination of W and bb
for W in wholesale_prices:
    for bb in buyback_prices:
        Cu = P-W  # Cost of understocking by bookstore
        Co = W-bb  # Cost of overstocking by bookstore

        z_hc = norm.ppf(Cu/(Cu + Co))
        Q = mu + z_hc * sigma

        profits_hc = []
        profits_bn = []
        for demand in demand_data['Demand']:
            sold_books = min(demand, Q)
            unsold_books = max(Q - demand, 0)

            profit_hc = Q * (W - 2) - unsold_books * bb  # Whole chain profit
            profit_bn = sold_books * P - W * Q + unsold_books * bb  # Bookstore profit

            profits_hc.append(profit_hc)
            profits_bn.append(profit_bn)
```

```
        average_profit_hc = sum(profits_hc) / len(profits_hc)
        average_profit_bn = sum(profits_bn) / len(profits_bn)

        result_summary.append((W, bb, average_profit_hc, average_profit_bn))

# Convert results to DataFrame for better visualization
result_df = pd.DataFrame(result_summary, columns=['Wholesale Price', 'Buyback Price', 'Avg P

pd.set_option('display.max_rows', 1000)
print(result_df[(result_df['Avg Profit HC'] > 237349) & (result_df['Avg Profit BN'] > 154425)
```

	Wholesale Price	Buyback Price	Avg Profit HC	Avg Profit BN
2038	14.081633	12.877551	238252.145076	154542.234154
2039	14.081633	12.979592	238093.692554	154794.936936
2040	14.081633	13.081633	237914.915279	155059.251951
2041	14.081633	13.183673	237711.375864	155335.807721
2042	14.081633	13.285714	237476.984090	155625.209784
2096	14.183673	13.693878	238492.020898	154516.500554
2097	14.183673	13.795918	237995.610527	154859.478589