# 基于Newsvendor模型的Reebok订货决策研究

# 摘要

本文针对单季订货决策问题，构建了基于 **Newsvendor 模型** 的库存优化方法。首先提出**简单二分法模型**，将明星球员与普通球员分开处理，前者直接订购成衣球衣，后者通过空白球衣满足需求。进一步，设计**三阶段复杂模型**，引入空白球衣作为风险缓冲库存，从整体需求出发，依次确定总订货量、明星成衣比例与空白球衣配置。通过蒙特卡洛模拟验证，两种模型均能获得较高利润，但复杂模型能够更好地平衡缺货与积压风险，显著提升整体利润水平。

**关键词：**Newsvendor模型；蒙特卡洛模拟

# 问题重述

作为耐克进入NFL之前的官方供应商，Reebok 需要在赛季前为球队和市场订购合适数量的球衣，其内部供应链如图1所示：

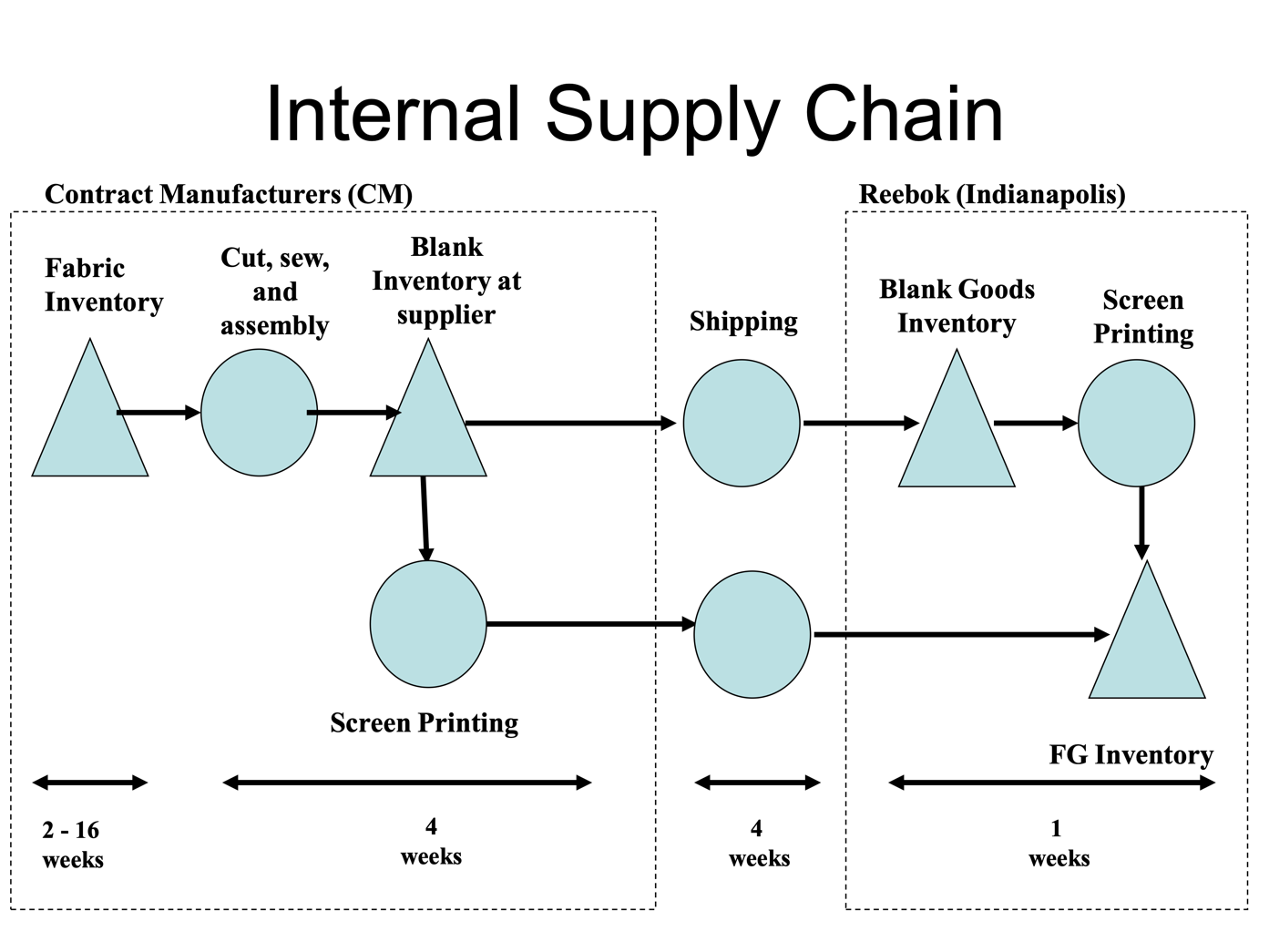


图1:Reebok 内部供应链

同时，球衣需求存在以下特点：

1. 需求不确定性：明星球员的球衣销量可能远超普通球员，但也存在销量骤减的风险。
2. 产品结构复杂：球衣分为成衣和空白球衣。成衣需赛季前由 CM 加工；空白球衣仅含球队元素，赛季中可由 Reebok 加印球员信息，灵活性更强。
3. 成本与利润差异：成衣成本较高，赛季末残值有限；空白球衣需额外加印成本。

因此，目标是在需求随机的情况下，确定成衣和空白球衣的单季最佳订货量，平衡缺货损失和积压损失，以最大化预期利润。

# 模型假设

1. 球队总需求服从正态分布。
2. 各明星球员需求和其他球员需求相互独立，均服从正态分布。
3. 库存周期为单赛季，赛季中不可向 CM 追加订单。
4. 空白球衣可在赛季内转印成任意球员球衣。
5. 赛季结束后剩余库存按固定残值处理。

各类参数与成本如表1所示：

|  |  |  |
| --- | --- | --- |
| 符号 | 说明 | 数值 |
|  | Reebok售出收入按批发价 | $24 |
|  | Blank cost | $9.5 |
|  | 已订成衣视为赛前在 CM 加工的成本 | $9.5+$1.4 = $10.9 |
|  | 空白球衣在赛中加印的成本 | $9.5+$2.4 = $11.9 |
|  | 未售成衣的残值 | $7 |
|  | 空白球衣的单赛季库存成本 | $1.04 |
|  | 空白球衣的残值 | $9.5-$1.04 = $8.46 |

表1:参数与成本

# 模型建立

## Newsvendor模型

Newsvendor 模型用于单周期、随机需求下的订货决策，其订货量满足：

其中符号如表2所示：

|  |  |  |
| --- | --- | --- |
| 符号 | 说明 | 单位 |
|  | 决策变量，即订货量 | 件 |
|  | 积压成本 | $ |
|  | 缺货成本 | $ |
|  | 需求分布函数均值与标准差 | \ |
|  | 标准正态分布分位数 | \ |

表2:Newsvendor模型符号说明

## 简单模型：二分法

明星球员（#12, #24, #80, #4, #54, #32）在CM直接配置成衣：

其他球员统一用空白球衣，并在Reebok加印：

由excel计算得出，成衣订货量为87468，空白球衣订货量为37991，总计125459。

## 复杂模型：三阶段法

### 阶段1：团队总量决策

由于空白球衣能转印成任何球员，考虑到供应链的灵活性，先以空白球衣为边际单位计算团队订货总量。

得到空白球衣的服务水平（不缺货概率）为0.92，利用正态分布得到球队总订货量为114672。

### 阶段2：明星球员成衣决策

仍旧以空白球衣为边际单位，从团队总量中先拨给明星球员一部分做成衣，缺货包括两种情况：空白球衣足够时，可在赛季内转印，只损失从CM和Reebok转印的利润差；空白球衣不够时，不可以转印，损失全部。

当空白球衣足够时：

当空白球衣不足时：

在空白球衣的服务水平下，期望缺货成本为：

同时，积压成本为：

临街比率为：

由excel计算得出，成衣总订货量为50669。

### 阶段3:其余球员以及明星球员溢出空白球衣决策

由总订货量减去成衣订货量得到空白球衣订货量为64003。空白球衣既可补足明星缺口，也可用于普通球员需求。

# 模拟与结果分析

通过excel采用1次蒙特卡洛模拟，生成 500 组需求样本，计算总利润。进一步通过python生成20000次蒙特卡洛模拟，代码在附录。得到利润分布图如图2所示：

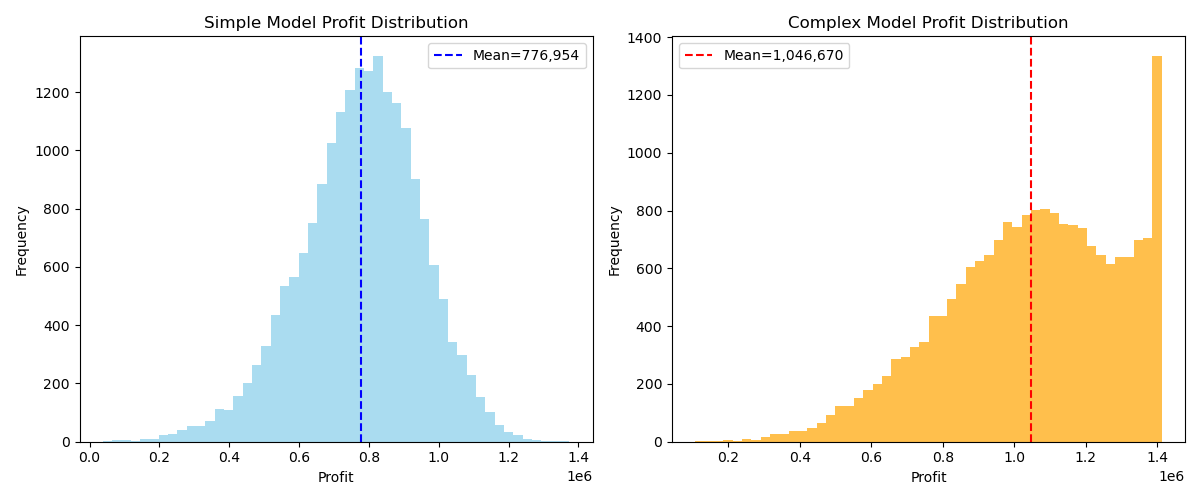


图2:两个模型的利润分布直方图

在简单模型中（左图），利润分布近似呈正态分布，均值约为 77.7 万美元。原因在于需求服从正态分布，且成本与残值函数线性可加，整体利润波动主要由需求随机性驱动。

相较之下，复杂模型（右图）表现出更高的期望利润（均值约 104.7 万美元），分布呈现明显的右偏特征。这一差异源于复杂模型允许利用空白球衣进行转印，以满足明星球员需求的缺口，从而有效减少了因缺货导致的机会损失。该机制不仅提升了利润的整体水平，同时降低低利润结果的概率，使得利润分布的左尾被显著压缩。

同时，复杂模型在极高需求场景下会触及“订货量上限”，具体而言，当市场需求远高于订货量时，所有明星球员的成衣均被售出，且空白球衣几乎完全转印并销售，从而使得剩余库存接近零。此时模拟的利润结果高度集中在理论最大值附近，形成直方图中的陡峭峰值。这种结果意味着在大多数模拟情境中，订货量与需求匹配较好，大部分产品能够销售出去，几乎没有滞销或残值损失。

# 附录

|  |
| --- |
| import numpy as np |
| import matplotlib.pyplot as plt |
|  |
| # ====================== |
| # 输入参数 |
| # ====================== |
|  |
| # 每个明星和 Others 的需求分布参数 (mean, stdev) |
| demand\_params = { |
| "Brady\_12": (30763, 13843), |
| "Law\_24": (10569, 4765), |
| "Brown\_80": (8159, 3671), |
| "Vinatieri\_4": (7270, 4362), |
| "Bruschi\_54": (5526, 3316), |
| "Smith\_32": (2118, 1271), |
| "Others": (23275, 10474) |
| } |
|  |
| # 订货量 |
| order\_qty = { |
| "Brady\_12": 24673, |
| "Law\_24": 6476, |
| "Brown\_80": 6544, |
| "Vinatieri\_4": 5351, |
| "Bruschi\_54": 4067, |
| "Smith\_32": 1558, |
| "Blanks": 64003 |
| } |
|  |
| # 成本和价格参数 |
| price = 24.0 |
| cost\_dressed = 10.9 # dressed jersey cost |
| cost\_blank = 9.5 # blank jersey cost |
| cost\_convert = 2.4 # converting blank -> dressed |
| salvage\_dressed = 7.0 |
| salvage\_blank = 8.46 |
|  |
| # 模拟次数 |
| N = 20000 |
|  |
| # ====================== |
| # 模拟函数 |
| # ====================== |
|  |
| def simulate\_once(model="simple"): |
| # 随机生成需求 |
| demand = {k: max(0, int(np.random.normal(mu, sigma))) |
| for k, (mu, sigma) in demand\_params.items()} |
|  |
| # ====================== |
| # 简单模型 |
| # ====================== |
| if model == "simple": |
| # 明星球员 dressed |
| sales\_dressed = {k: min(demand[k], order\_qty[k]) |
| for k in demand if k != "Others"} |
| leftover\_dressed = {k: order\_qty[k] - sales\_dressed[k] |
| for k in sales\_dressed} |
|  |
| # Others 用 blank |
| sales\_blank = min(demand["Others"], order\_qty["Blanks"]) |
| leftover\_blank = order\_qty["Blanks"] - sales\_blank |
|  |
| # 成本：明星成衣 + 空白 + 转印 |
| cost = sum(order\_qty[k] \* cost\_dressed for k in sales\_dressed) \ |
| + order\_qty["Blanks"] \* cost\_blank \ |
| + sales\_blank \* cost\_convert # Others 全部需要转印 |
|  |
| # 收入 |
| revenue = price \* (sum(sales\_dressed.values()) + sales\_blank) |
|  |
| # 残值：dressed、blank 各自 salvage |
| salvage = sum(v \* salvage\_dressed for v in leftover\_dressed.values()) \ |
| + leftover\_blank \* salvage\_blank |
|  |
| profit = revenue - cost + salvage |
| return profit |
|  |
| # ====================== |
| # 复杂模型 |
| # ====================== |
| elif model == "complex": |
| sales\_dressed = {} |
| leftover\_dressed = {} |
| convert\_blank = 0 |
|  |
| leftover\_blank = order\_qty["Blanks"] |
|  |
| # 明星先用 dressed，缺口用 blank 转印 |
| for k in demand: |
| if k == "Others": |
| continue |
| demand\_i = demand[k] |
| dressed\_stock = order\_qty[k] |
|  |
| # 先卖 dressed |
| sales\_d = min(demand\_i, dressed\_stock) |
| leftover\_d = dressed\_stock - sales\_d |
| shortage = demand\_i - sales\_d |
|  |
| # 缺口用 blank 转印 |
| convert = min(shortage, leftover\_blank) |
| convert\_blank += convert |
| leftover\_blank -= convert |
|  |
| sales\_dressed[k] = sales\_d + convert |
| leftover\_dressed[k] = leftover\_d |
|  |
| # Others 用剩余 blank |
| sales\_blank = min(demand["Others"], leftover\_blank) |
| leftover\_blank -= sales\_blank |
|  |
| # 成本：dressed + blank + 转印 |
| cost = sum(order\_qty[k] \* cost\_dressed for k in order\_qty if k != "Blanks") \ |
| + order\_qty["Blanks"] \* cost\_blank \ |
| + (convert\_blank + sales\_blank) \* cost\_convert # 所有转印 |
|  |
| # 收入 |
| revenue = price \* (sum(sales\_dressed.values()) + sales\_blank) |
|  |
| # 残值：dressed、blank 各自 salvage |
| salvage = sum(v \* salvage\_dressed for v in leftover\_dressed.values()) \ |
| + leftover\_blank \* salvage\_blank |
|  |
| profit = revenue - cost + salvage |
| return profit |
|  |
| # ====================== |
| # 跑模拟 |
| # ====================== |
| profits\_simple = [simulate\_once("simple") for \_ in range(N)] |
| profits\_complex = [simulate\_once("complex") for \_ in range(N)] |
|  |
| # 均值 |
| mean\_simple = np.mean(profits\_simple) |
| mean\_complex = np.mean(profits\_complex) |
| print(f"Simple Model Mean Profit: {mean\_simple:,.0f}") |
| print(f"Complex Model Mean Profit: {mean\_complex:,.0f}") |
|  |
| # ====================== |
| # 画直方图 |
| # ====================== |
| plt.figure(figsize=(12,5)) |
|  |
| plt.subplot(1,2,1) |
| plt.hist(profits\_simple, bins=50, color="skyblue", alpha=0.7) |
| plt.axvline(mean\_simple, color="blue", linestyle="--", label=f"Mean={mean\_simple:,.0f}") |
| plt.title("Simple Model Profit Distribution") |
| plt.xlabel("Profit") |
| plt.ylabel("Frequency") |
| plt.legend() |
|  |
| plt.subplot(1,2,2) |
| plt.hist(profits\_complex, bins=50, color="orange", alpha=0.7) |
| plt.axvline(mean\_complex, color="red", linestyle="--", label=f"Mean={mean\_complex:,.0f}") |
| plt.title("Complex Model Profit Distribution") |
| plt.xlabel("Profit") |
| plt.ylabel("Frequency") |
| plt.legend() |
|  |
| plt.tight\_layout() |
| plt.show() |