


Minimizing Vaccine Variance
Reducing Excess Inventory in Prevenar

Pfizer Team:
Jonathan Lowe, Abby Garrett, Michelle Ong

Faculty Advisor:
Prof. Daniel Freund

Capstone Team:
Kyle Mana & Ryan Trusler

Problem Background and Scope

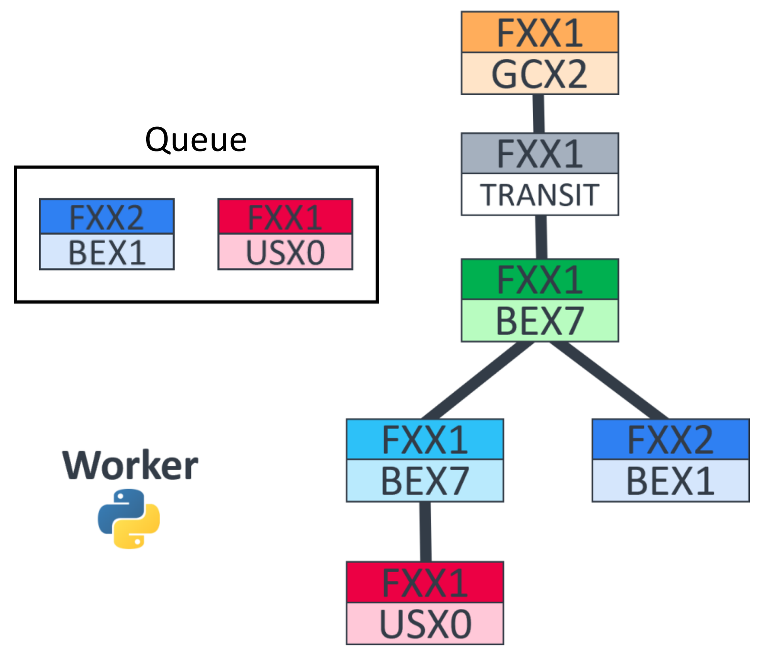
- *Problem Statement*
  - Develop a robust, flexible, and scalable tool for inventory management and supply planning that provides optimal inventory suggestions and results in a significant reduction in excess inventory for Prevenar
- *Prevenar Brand*
  - Pneumococcal conjugate vaccine
  - World and Pfizer's largest vaccine brand before COVID
  - $5.95 Billion in sales in 2021
  - Vaccine used in virtually every country worldwide across 6 continents

Project Scope:

Replicate supply chain logic in a stable environment	Python OOP environment
	End-to-end data pipeline
	Methods generalizable to other Pfizer brands
Develop an optimization for inventory prescription	Minimize the variance of inventory from *target*
	Reduce excess inventory
	Integer optimization for batch-size ordering
Display results in a dashboard	Provide an interpretable view of results and inventory prescriptions
	Flexible visualizations for supply chain manager to consider multiple scenarios
	Provide actionable insights for supply planning

Replicating Supply Chain Logic

- **Graph Representation of Supply Chain**
  - Nodes in the Pfizer supply chain are Item and Location pairs
  - Demand is pulled through the network by the terminal nodes in the market
  - Each node satisfies the demand of its child nodes and is supplied by its parent nodes
  - Consider only nodes below the 'drug-product' line – all supply chains are trees



- **Historical vs Forecasting Logic**
  - Brand managers want to compare historical and forecasted supply
  - For historical movements, actual data exists in the data warehouse
  - Looking forward, we imitate the heuristics used by the supply manager to generate a supply plan

Brand Manager sets Cycle Target — Takes into account lead times, manufacturing times, safety stock → Calculate Target Inventory — Node should have enough inventory to cover one cycle worth of demand → Calculate Required Supply — Given current inventory, demand and target inventory, heuristic used to find required supply

- **Round up Heuristic**
  - Current supply planning heuristic is to round up the supply to the nearest batch size

$$supply_{n,d} = \max\left(\left\lceil \frac{K_{n,d} - \sigma_{n,d-1} + demand_{n,d}}{u_n} \right\rceil, 0\right)$$

  - $K_{n,d}$ is the target inventory at the node and date
  - Ensures inventory never drops below target
  - Can create large variance from target with large batch sizes

- **Tree Search Algorithm**
  - Initialize a queue of terminal nodes
  - Compile historical and forecasting information for next node in the queue using supply chain logic
  - Add node to the queue only after all its children have been visited and information gathered
  - Multiple trees run in parallel

References

- Bertsimas, Dimitris, and John N. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1997.
- Conforti, Michele, et al. *Integer Programming*. Springer International Publishing, 2014.
- Letchford, Adam N., and Andrea Lodi. "Strengthening Chvátal–Gomory Cuts and Gomory Fractional Cuts." *Operations Research Letters*, vol. 30, no. 2, 2002, pp. 74–82., https://doi.org/10.1016/s0167-6377(02)00112-8.

Optimization

In Greek:

$$\min_{\alpha,\sigma,\omega} \alpha_{dn}(1-\lambda)\theta + \sigma_{dn}\lambda \quad (1)$$

subject to

$$s_n + \sum_{t=0}^{d} \sum_{j:(j,n)\in\mathcal{A}} \omega_{tjn}u_n - \sum_{t=0}^{d} \sum_{j:(n,j)\in\mathcal{A}} \omega_{tnj}u_j = \sigma_{dn} \quad \forall d \in D, \forall n \in N \setminus Sink$$

$$\sum_{j:(j,n)\in\mathcal{A}} \omega_{djn}u_n = f_{dn} \quad d \in D, \forall n \in Sink$$

$$\sigma_{dn} \geq K_{dn}e_n \quad \forall d \in D, \forall n \in N$$

$$\alpha_{dn} \geq K_{dn} - \sigma_{dn} \quad \forall d \in D, n \in N$$

$$\alpha_{dn} \geq -(K_{dn} - \sigma_{dn}) \quad \forall d \in D, n \in N$$

$$K_{dn} = \sum_{j:(n,j)\in\mathcal{A}} \sum_{t=d+1}^{d+c_{dn}} \omega_{tnj}u_j \quad \forall d \in D, n \in N \setminus Sink$$

$$\omega_{djn} \in \mathbb{Z}_+, \alpha_{dn} \in \mathbb{R}_+, \sigma_{dn} \in \mathbb{R}_+$$

In English:

1. Minimize a weighted sum of variance from an inventory target and the total planned inventory subject to
2. inventory cannot flow out of a location unless it has already flowed in **and**
3. the customer demand must always be satisfied **and**
4. inventory must remain above a threshold.

Warm-start Heuristics:

- We implemented two warm-start heuristics to make the problem tractable:
  1. A round-up heuristic, which sequentially orders inventory and always rounds up to discrete values.
  2. A greedy optimization heuristic, which traverses the supply chain and generates locally optimal round-up/round-down decisions before moving to the next node.
- Each heuristic made the problem converge to a tolerable gap within three minutes
- Greedy Heuristic > Round-up Heuristic > Cold-start

Strengthening Constraints:

Strengthening constraints further decrease run-times by enforcing integrality at several variables per branch:

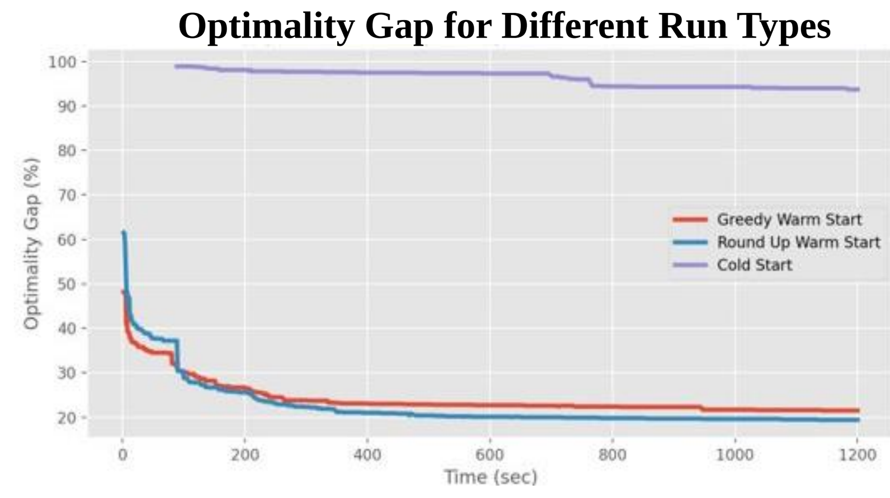
$$\sum_{j:(j,n)\in\mathcal{A}} \omega_{djn}u_n < (x_{dn}+1)u_n + M(1-z_{dn})$$

$$\sum_{j:(j,n)\in\mathcal{A}} \omega_{djn}u_n \leq Mz_{dn}$$

$$-x_{dn} \leq M(1-z_{dn})$$

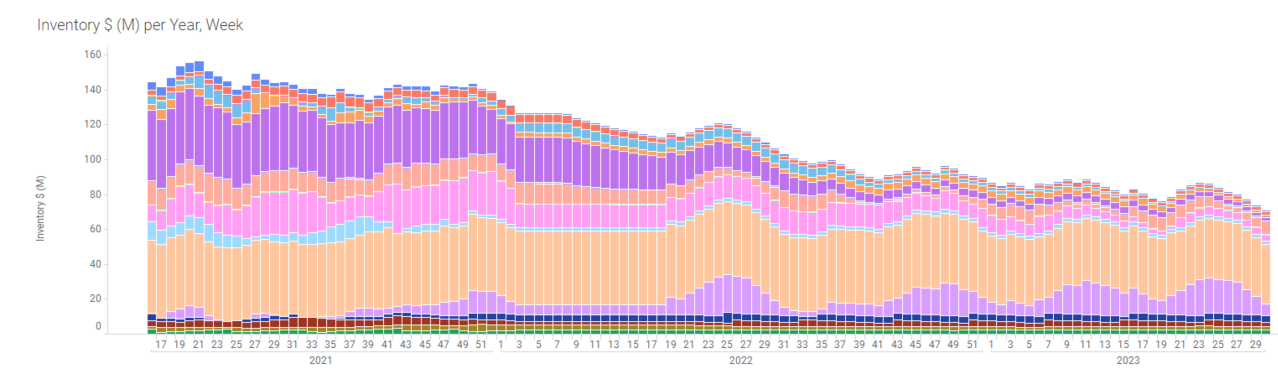
$$x_{dn} \leq Mz_{dn}$$

$$z_{dn} \in \{0,1\}$$

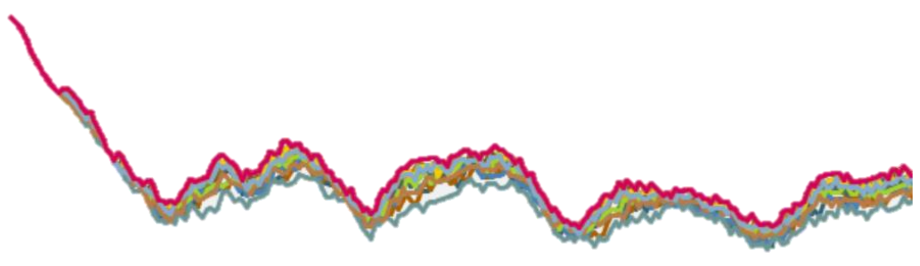


Results and Impact

Plan to a ***previously infeasible level of detail*** by leveraging the scalable nature of advanced analytics



Quickly compare ***optimal inventory plans*** with differing inventory strategies



Each line represents an optimal plan with a different inventory strategy.

Save money on investments in planned inventory with ***limited change in operating model***

Inventory ⬇11%
⬇$50M In Annual Investment
⇧ $4M Profit

Save time by ***automating data pipelines*** that extract and transform data for immediate insights