Inventory Forecasting with SAS/OR®

Tom Taylor
ttaylor@computer.org
Palladian Analysis and Consulting LLC
Houston, TX

Inventory problems are common to product oriented companies. It seems that there are as many solutions to these problems as there are companies with inventory. A lot of systems have evolved as companies have grown. These evolutionary systems are largely designed to make the supply chain easy to manage for the corporation's employees while keeping management and customers happy. This behavior has resulted in systems that are less than optimal in many cases.

In addition to the organization issue mentioned above, we must also realize there are two basic types of inventory. These are **replenishment** inventory and **one shot** inventory.

Replenishment is commonly known and taught in business schools. Replenishment is inventory used year round and demand varies only slightly by season. The classic replenishment inventory is milk. Demand is very predictable and replenishment plans are easy.

One shot goes under many names including seasonal, fashion, special event, close outs, etc. One shot inventory are items that are brought in typically for a season, holiday event, or for a particular fashion. It is called one-shot because typically the supply must be bought pre-season and there is no opportunity to reorder during the season. There is a lot of one-shot inventory; examples include fashion clothing, seasonal clothing, holiday foods, summer toys, Christmas toys, etc. This paper focuses on one shot inventory because it can make or break the company and that was the inventory problem solved with SAS OR.

Inventory Risk and Options

Inventory orders normally have to be placed three months to one-year in advance. These orders have to be correct because the selling season for the item is normally over before additional orders can be placed. Leftovers normally cannot be returned to the manufacturer. Inventory may be delivered to the warehouse just-in-time, but it can still have a long supply chain.

Today, once inventory is purchased, it is largely "yours". If it is a replenishment item, order adjustments over time can generally work out the excess with the impact being only lower inventory turns and lost profit. If it is a one-shot inventory, then it becomes distinctly possible to have "dead" inventory. Dead inventory is a total loss; burning money in the fireplace is probably cheaper because at least the money did not take up lighted display space before becoming dead. As an example, one Houston retailer, Weiners, died twice, once chapter 11 and once chapter 7 primarily due to dead inventory.

The first time the stores became so clogged with dead inventory there was little floor space for new inventory. Weakened by the first bankruptcy, the second time a slight change in brand preferences and poor financial resources finished them off.

So what are the options to avoid "dead" inventory? Limiting selection and stocking only common items is one solution, but then the stores lose the broad selection that attracts most customers. A better solution is just-in-time distribution of a broad selection.

Just-in-time Distribution

Distribution of merchandise involves two steps, the first is adding up what to buy and the second is deciding which stores in the chain should get what amounts. Most of the world likes to do bottom up planning, that is they decide how much of each item each store should have, sum it all up to a top level total, and place an order. This is an OK way to do things and we will not spend much time debating planning techniques. The next step is what we are here to discuss, that is deciding which stores should get what amounts. As mentioned previously, orders for merchandise are placed three months to one year in advance of delivery. If one does the simple task of just taking the plan made one year earlier and makes a distribution, one is going to have a lot of problems with closed stores, new stores, changing neighborhoods and the like. If one really wants to take advantage of every stick of merchandise and make every possible effort to avoid "dead" merchandise, then the best way to distribute merchandise is to throw out the old plan, look at today's inventory by store of similar merchandise, look at today's sales by store, leave something for new stores and then correctly allocate the merchandise in two or three hours down to the last stick, leaving nothing hanging in the warehouse. This is what we do with SAS OR.

We should mention there is a strong desire in distribution today to do cross docking. That is the merchandise comes directly from the manufacturer prepackaged and in some cases pre-labeled ready to go to the stores. At the distribution center, it is cross-docked, out of one truck straight into another. This clearly works for some classes of merchandise. It does not work well for inventory that has little individual item depth and a broad selection such as we are dealing with here. Cross-docking is too rigid for this situation.

The Added Complexity of Sized Clothing

To further complicate the problem, we do clothing with SAS OR. Clothing has different sizes, in the most extreme case I am aware of, men's dress shirts come in 90 possible sizes. If one simply sends a store 1 shirt each of all 90 sizes, one will be quickly out of business as probably 15 sizes cover the majority of the population and there will be a lot of "dead" merchandise. So the store really wants a good selection of these core sizes and a sprinkling of the other sizes if the total of 90 shirts is to be sent to the store.

To further complicate again, size distribution changes by region and neighborhood. In general, California is shorter than New York and there are still a lot of tall Scandinavians

in Minnesota. Ethnic neighborhoods can have very unique size needs. These needs much be matched or the dreaded "dead" merchandise will result.

To solve all these issues we turned to SAS and specifically Integer Programming using SAS OR. We built an application that handles all these issues in a quick manner and nearly eliminates "dead" merchandise.

OR -- Operations Research – Linear Programming --Integer Programming

OR (Operations research) is a branch of math that finds an optimum solution to tradeoff problems. The problems usually involve production system constraints such as finite amounts of material and labor to optimally produce a variety of products. The simple example is; given x amount of labor and y amount of wood and respective selling prices and production costs to make either tables or chairs, how many tables and how many chairs should be made to maximize profit?

OR is a very mathematically elegant solution to problems of production or distribution. Mathematicians like it because it can be solved in a predictable amount of steps for linear problems. Linear means in this case that it is OK to deal in fractions of units such as ½ a gallon or in the above example ½ a chair (which can be ignored in reality). Problems that require whole units in the solutions are termed Integer Programming and are exponentially more difficult to solve in a timely manner. With the advent of computers, recursive, or trial and error techniques such as neural networks and genetic algorithms arose as alternates to OR for problem solving these problems. After a lot of study, the author has concluded that the recursive techniques should be only be used as a second choice. In these recursive systems it is hard to know when one has a good answer or to predict time to results on a repeated basis.

To add further light to the OR versus recursive techniques issue, in the author's experience, professionals in this world are very biased to proposed solutions based on the background of the expert. That is to say mathematician or management science types tend to see every problem in the world as OR problems. On the other hand, computer scientists tend to see solutions to complex problems through neural networks and genetic algorithms. Finally, statisticians see the world as regression is man's best friend. All will claim to have the best solution and most will be only vaguely aware of the existence of the other guy's techniques.

A word about Integer Programming

Dry goods inventory problems must generally be solved with integer programming. Integer programming is tricky because solve time for problems grow at something like the square of the complexity. Problems must be carefully constructed or solve time goes through the roof. The first IP (integer programming) problem the author set up for clothing merchandise, ran for a week, before the computer was killed, being nowhere near finished. That same problem now runs in under 30 minutes but with a lot of careful constraints.

One might try solving linear programming problems and just randomly rounding to the nearest units. At store level quantities, this does not work well. Too many of the choices are between shipping one case or two. IP does a much better job.

How we do Just-in-time Distribution with SAS OR using Integer Programming

The author has always joked that this is his little 80,000 variable algebra problem. That is all it is, unfortunately it has far fewer equations than variables so rigorous algebra is out of the question. But given the constraint system of SAS OR it can be solved accurately and quickly. Our solution for solving quickly is to perfectly optimize the inventory available for each store but only approximately optimize for the entire chain, we call this the 98% solution. The 100% solution would be to run OR-IP for the entire chain. This would be great, but for the solve time of IP over such a large problem is greater than one week. To get the solve time down to under one hour; we break the problem into two steps. First we use fuzzy logic and recursive techniques to determine a quantity each store should receive. Not all stores will receive the merchandise. For those that do, all will receive a minimum, say 20 units, those with high need or high volume might receive 40 units. The second step, the important one, is to pragmatically set up individual IP problems for each store and solve for a solution. On a per store basis, these solve very quick, on the order of a few seconds. Solving IP's for a few hundred stores will cumulatively add up to 20 to 40 minutes depending on the number of stores in this specific allocation.

We have found that IP's for business applications are highly constrained. That is the customer has parameters for every major aspect of the problem they want to set before it runs. User interface wise, this turns into many slider bars and parameter tables. However, it ultimately gives the customer the exact results they are looking for. The downside to this complex set of parameters is the occasional users become intimidated by all the choices and at best run default setting problems. At worst they start avoiding process all together. In a backwards proof that the SAS application is very effective, a supervisor started noticing inventory levels were getting out of whack in certain stores, upon investigation, he discovered one employee had been avoiding using the SAS application in favor of a very simple and inaccurate tool that would also distribute the merchandise. We retrained the employee and the problem went away.

To finish up, the whole process described above can be done in less than an hour. Because this is a company that prides itself in zero inventory shrinkage, only after the inventory has been counted into the warehouse and we know the precise counts do we begin this process. In a couple of hours the resulting allocation is returned to the warehouse and the merchandise is on its way the same day. Nothing is ever left hanging in the warehouse, and nothing ever mysteriously walks out the door using this system.

Strategies to Gain User Acceptance

The monolithic software company from Redmond, Washington has an amazing lock on our perceptions of what desktop user interfaces should look like and the performance of certain keys. The author has found that one of the easiest ways to gain acceptance of a SAS project is to put a Microsoft front end on it. When we originally wrote this application in the mid-90's only rudimentary tools such as DDE were available to interface between SAS and Microsoft. There was no automated way to signal when a data transfer was complete between the two. We resorted to a timer. With today's SAS and **SAS Integration Technologies**, we have tools that allow for a dramatically better programming interface between the SAS and Microsoft cultures.

Our favorite technique is to use an Excel Spreadsheet as the user interface for SAS. It is usually beautiful and users instantly feel they know how to work with the interface.

A second big issue is to make sure the user feels they have power over the computer instead of the computer having power over them. If people are intimidated by an application they will find excuses for not using it. We solved this problem by allowing our users the ability to tweak the results anytime they wished. It turns out they hardly ever tweak the results because they are generally happy with them, but having the power to do so gives them a great deal of confidence. This feature is especially useful when a store manger calls up and makes a special request for specific merchandise. With this it is easy to make the one time exception so the manager can cover his/her special situation without a lot of hand work on the part of the merchandiser.

Payback

Good business people always want to know the payback from an investment. This system was a significant investment and there was a lot of interest in knowing the payback. A problem with business people is they seldom have the discipline or the systems to run a controlled experiment. That is for example, use the new system on ½ the stores and compare the results between the old and new system. No, we like everyone else implemented chain wide while many other changes were going on and still wanted to know the exact impact of this project on the bottom line. We of course never got an exact answer as to payback. We ultimately settled on two answers. The first was conditional, if sales increased 00.2%, the system paid for itself in one year. People felt this had to be true. The second and strongest result was the stores stopped complaining about inventory imbalances or "dead" merchandise.

Durability

During the fad of a few years ago, everyone was replacing their custom software with packages, believing a package to be a better total value. The author helped the client do a nationwide search for this replacement solution. We made two discoveries in the process. First, a lot of companies allow their own bigness and volume of inventory to ignore the many small, but profit enhancing details that we handle with this system.

Second, there was no other detailed allocation package available that could manage one shot inventory with any precision. In most cases, the package software companies did the quick and dirty allocations that a database designer could conceive and the expertise stopped right there. As common, they were experts in their own domain, but not aware of other areas of expertise. They were clearly not aware of the power of SAS-OR and integer programming.

Conclusion

We have shown that one simple and very powerful technique to improve the accuracy of an inventory needs forecast is to keep the forecast at an aggregate nationwide figure as long as possible. Only at the last possible moment should an allocation to individual stores be made. To do allocations in near real-time one needs a powerful and accurate system such as this one using SAS OR. Major benefits of this system are very low inventory shrinkage, happy store mangers, and practically no "dead" merchandise.