A Software Tool for Intermittent Demand Analysis

Hugh R. Medal, Manuel D. Rossetti, Vijith M. Varghese, Edward A. Pohl
Department of Industrial Engineering
University of Arkansas, Fayetteville, AR 72701, USA

Abstract

The forecasting of intermittent demand is a difficult task because of the irregular behavior of the demand process. As a result, the selection of an effective forecasting technique can be challenging. We present an object-oriented software framework for intermittent demand forecasting and inventory analysis. The object-oriented structure of this framework allows easy implementation and integration of new data sources, forecasting techniques, and forecast metrics. This framework is implemented in Java within a user interface and is named the CID (CELDi Intermittent Demand) Forecaster. Unique aspects of the final software product include the implementation of a large number of forecast metrics, the inclusion of most of the intermittent forecasting techniques found in the forecasting literature, and the inclusion of meta forecasting (see [11]).

Keywords

forecasting, intermittent demand, inventory, inventory control, logistics

1 Introduction

Demand forecasting is an important area of inventory management. Good forecasts can help managers choose stocking policies for items. Intermittent demand is characterized by demand data that has many time periods with zero demands. Other characteristics include zero demands being clustered together, and a high variability in order size. Examples include spare parts and slow moving products. These characteristics make intermittent demand difficult to describe using conventional models (e.g. ARIMA) as well as to forecast. Traditional forecasting methods like simple exponential smoothing (SES) and moving average are often unsuitable in intermittent demand scenarios. This makes intermittent demand forecasting an active area of research. Techniques designed specifically for intermittent demand include Croston's method [3], the Syntetos and Boylan method [10], the Average Demand method [9], and the MCARTA method [12].

There are a number of existing software programs with intermittent demand forecasting capability. Commercial programs include Minitab, SAS, JMP, and ForecastPro among others. The only free or open-source packages that we found were R statistical computing software [13] and OpenForecast. Among these, only SAS has algorithms specific to intermittent demand. For a more extensive list of forecasting packages, see [14].

In this paper we describe a intermittent demand forecasting software framework and a computer program called the CID (CELDi Intermittent Demand) forecaster. The CID program implements the framework in Java within a user interface and was created for a CELDi (Center for Engineering Logistics and Distribution) project at the University of Arkansas. The CID program is especially suited for addressing the problem of selecting an appropriate forecasting technique and stocking policy for one or more intermittent demand items. The remainder of this paper is as follows: section 2 describes the intermittent demand forecasting software framework, section 3 describes the inventory analysis functionality of CID, section 4 describes the functionality of CID, and section 5 gives a demonstration of the software. Finally, section 6 gives concluding remarks and suggests extensions to the software.

2 Software Framework

The main functionality of this software is based on a sub-package of the Java Simulation Library [7] called the forecasting package. The forecasting package is a utility package that contains tools for performing time series forecasting, especially that of intermittent demand. The forecasting packages include three sub-packages as well as various other useful classes for time series forecasting. The subpackages include demandgeneration, techniques, and metrics:

- Data analysis classes: The *IntermittentDemandStatistic.java* class calculates statistics for a time series. Standard time-series statistics (mean, variance, correlation, covariance) are calculated on the nonzero demands series and the interval between nonzero demands series. In addition, Markov-chain transition (Pr(non-zero demand after zero demand) and Pr(zero demand after non-zero demand)) and state probabilities (Pr(non zero demand) and Pr(zero demand)) are calculated.

- Data-related classes: The forecasting package contains a number of classes useful for analyzing time series data. The base class is *DataSource.java*, which represents a data source, which can be an array of data, a text file, a database, or something else. *DataSource.java* implements the *Observable.java* interface so observers may be attached and notified of updates. In addition, *IterativeDataSource.java* extends *DataSource.java* and provides a framework for iteratively obtaining observations and notifying observers.

- Techniques package: This package provides implementations of various forecasting techniques. The base class of this package is *ForecastTechnique.java*. This class subclasses *Observer.java* so it can be attached to an instance of *DataSource.java* and notified when a new datum point is collected. When a *ForecastTechnique* is notified of a new data point, it computes a next period forecast based on some forecasting algorithm.

- Metrics package: This package contains classes for calculating measures of forecast accuracy (see [6]). The base class in this package is *ForecastMetric.java*, which provides the basic functionality of being able to be attached to a *ForecastTechnique* and be notified when a new forecast had been made. In this framework, a metric uses the object-oriented concept of delegation, which means that each *ForecastMetric* object is composed of other objects. The idea is that most metrics are composed of three parts: the type of error measured (e.g. error, percentage error, relative error, etc.), the manner in which the collected error is manipulated (e.g, squared, or absolute value), and the statistic that represents the vector of manipulated error values collected (e.g., mean, median, etc.). Thus, each *ForecastMetric* object is composed of a sub-class of *ForecastMetric.java* that defines how error is collected, a class for manipulating the collected error (*MathFunctionIfc.java*), and a class for computing a statistic based on the manipulated error values collected (*StatCalculatorIfc.java*). An exception is *TheilU.java*, which implements Theil's U-Statistic and does not fit into this framework.

A useful aspect of our software framework is the object-oriented design. This allows the user to build off of pre-existing code for forecast techniques and metrics. In addition, the framwork is partly based on the observer pattern of object-oriented design (see [5]). Thus, forecast techniques are attached to data objects and 'observe' updates. Further, forecast metric objects 'observe' updates to forecast techniques. This allows for quick implementation of the packages.

3 Inventory Analysis

In addition to time series forecasting, the CID forecaster also gives analytical and simulated inventory performance. Results are given for an (r,Q) policy and supplied cost parameters (e.g. holding cost, reorder cost, etc.). The analytical model uses the $Gamma(\alpha,\beta)$ distribution as the demand during lead time distribution. Once the mean and variance of the demand distribution are known α and β can be obtained. The demand mean is set as the next period forecast value. The demand variance is set as $1.25\sqrt{MAD}$, where *MAD* denotes the mean absolute deviation between forecasts and actual observations for the time series (see [1]). Once the demand series mean and cost parameters are supplied, performance metrics for an (r,Q) policy can be calculated. Performance metrics calculated include but are not limited to expected customer wait time, expected total cost, expected number of backorders, and fill rate.

Because of the intermittency of the time series data, the *Gamma* distribution may not provide an adequate model for the demand arrival process. Consequently, the previous analytical model is lacking. Another method of modeling inventory performance for a single item is through discrete-event simulation. This method sacrifices closed-form results for a more accurate model for a more general demand arrival process. In the framework of the Java Simulation Library, an instance of *IterativeDataSource.java* must be supplied as an input to the (r,Q) simulation model. The advantage of this is that an *IterativeDataSource* may be an array of data, a text file, a database, or a complicated stochastic process. Once again, performance metrics for a policy can be calculated for a particular (r,Q) policy with given cost parameters and demand source definition.

4 Software Functions

The CID Forecaster has many of the same functions as existing statistical and forecasting softwares as well as functions suited for intermittent demand. To start, the user must import time series data from a file or database or by typing. The data may be analyzed graphically with a time series plot, an empirical PMF, or a frequency chart. The statistics mentioned in section 2 may also be calculated.

The CID forecaster is also able to make a forecast for a particular technique, compare techniques, and make forecasts in batch mode. Analysis options include plotting the forecast along with the dataset, obtaining the residuals, and obtaining the error metric values. The individual forecast mode makes a forecast for one dataset and one technique using multiple metrics to assess accuracy. The compare techniques mode allows the user to compare multiple techniques for a particular time series using multiple error metrics. The batch mode allows the user to compare technique peformance with a single error metric for multiple time series in the same execution. One unique feature of this software is the 'add technique group' function for when the user wants to test different parameter settings for a technique. The user specifies upper and lower bounds on each of the parameters for the technique as well as the number of parameter points to examine within the range. Then the forecast technique instances resulting in all combinations of the specified parameter settings are added to be analyzed.

In addition to standard forecasting, the CID forecaster also can do meta-forecasting for a time series (see [11]). Meta-forecasting is a classifier-based forecast technique selection method. In the CID software, a multinomial regression based meta-forecaster can recommend a forecasting technique for a particular time series based on a training dataset. As with standard forecasting, meta-forecasting can also be done in batch mode.

Finally, a user may choose to obtain analytical or simulated inventory performance for a given (r,Q) policy. For the analytical model, the demand per period mean and variance must either be set by the user or automatically obtained for given time series and forecast technique. At this point the inventory performance metrics discussed in section 2 can be obtained. For the simulation model, the user must set a particular demand generator (see section 2) as well as simulation experiment parameters (length of replication, warm-up length, number of replications). Once the parameters are set, the model can be run and inventory performance metrics obtained.

5 Software Demonstration

In this section, we give a short demonstration of the CID Forecaster. We start with a dataset, which can be imported from a text file or database and is show in figure 1. Next, we want to choose between two techniques, the Average Demand method [9] and Croston's method [3]. We select parameters for each of the techniques aribitrarily. To compare the two techniques, we asses various error metrics for their performance on our dataset. The software allows many different metrics to be chosen for the comparison. Figure 1 shows the compare techniques dialog. Figure 2 shows the results of the comparison. As the error summary statistics shows, the Croston outperfroms the Average Demand method for this dataset. Thus, we will proceed in our analysis using the Average Demand method.

At this point we could make more comparisons using the batch mode or the meta forcast modes. However, in this demonstration we will proceed to assessing the inventory performance for a particular (r,Q) policy based on the forecast generated by the Average Demand method. To do this, we use an analytical (r,Q) inventory model. Figure 3 shows the dialog used to set up this analysis. First, we need the demand during lead time distribution. We assume a constant lead time and then use the method described in section 3 to obtain a demand mean and variance from the next period forecast and forecast mean absolute deviation. Now, we set the parameters for our model (cost per stockout, etc.), choose a policy (the reorder point and reorder quanitity), and run the model. Figure 4 shows inventory performance predictions using our (r,Q) policy. At this point we would typically proceed to using the (r,Q) simulation model within the software to analyze our stocking policy further.

6 Conclusions and Future Research

In summary, we introduced the CID Forecaster for forecasting intermittent demand. In particular, this software is useful for aiding managers in choosing a forecasting method and stocking policy for a particular intermittent demand item. We discussed the importance of intermittent demand and the lack of available software. Then we discussed the Java package on which the CID Forecaster is based as well as some relevant classes. Finally, we described the functionality of the software.

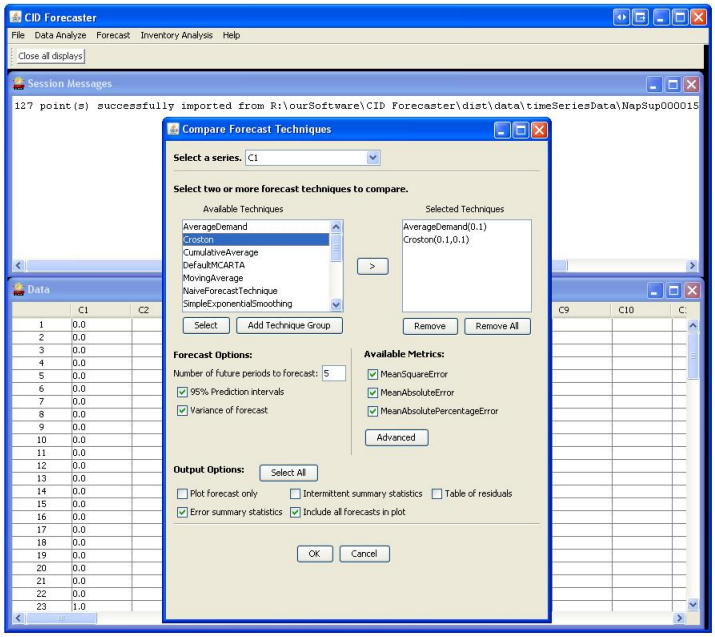

Figure 1: Compare Forecast Techniques Mode

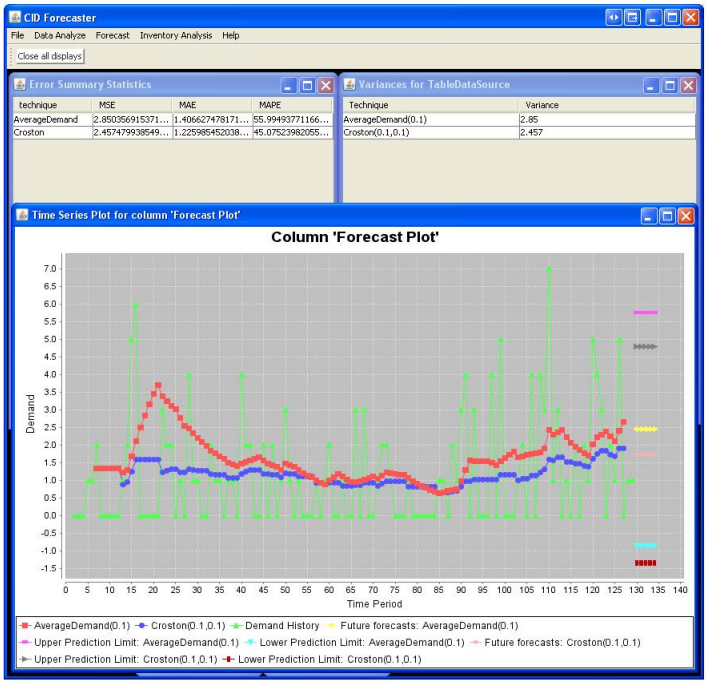

Figure 2: Results of Forecast Techniques Comparison

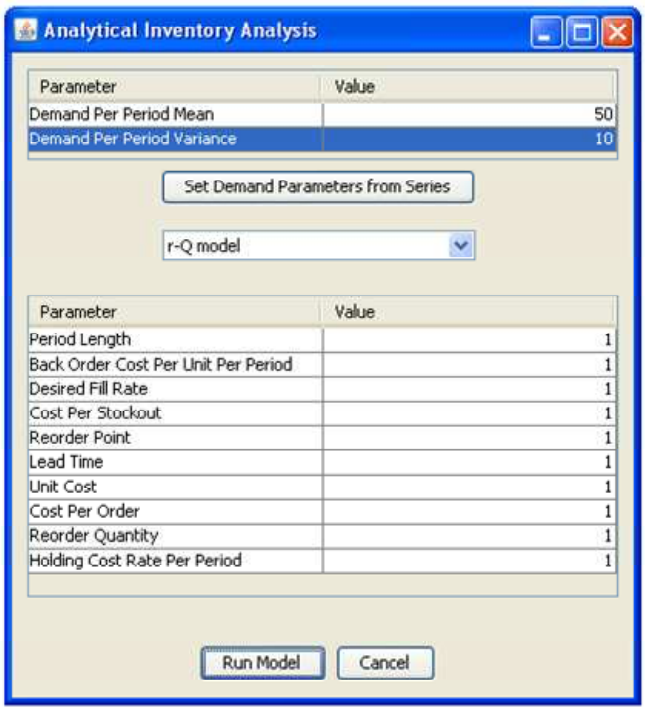

Figure 3: Inventory Analysis Dialog

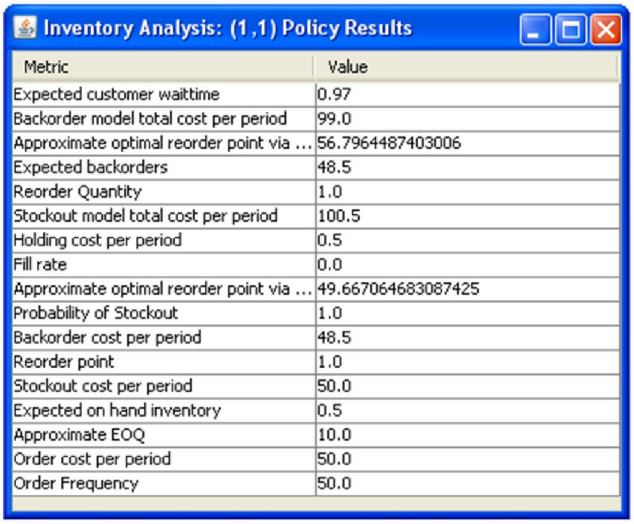

Figure 4: Results from the (r, Q) inventory model.

Possible extensions to this software include: integration of the open-source R statistical computing software to facilitate further plots and ARIMA models, the automated selection of forecast technique parameters, as well as continued development of the inventory analysis module. Another helpful extension to this software would be the selection of optimal or approximately optimal r and Q values for both the analytical and simulation model. For the analytical model an exact algorithm is available (see [15]) while the optimization of the simulation model is more complicated. Finally, we would like to extend the the simulation model to include multiple items and multiple echelons.

Acknowledgements

This work was supported by the National Science Foundation and the U. S. Navy through the auspices of the Center for Engineering Logistics and Distribution at the University of Arkansas under project number CDP07-UA. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation and the U. S. Navy.

References

[1] Axsäter, S. 2006. Inventory Control, 2nd ed. Springer, New York.

[2] Collopy, F., Armstrong, J.S., 2000. "Another Error Measure for Selection of the Best Forecasting Method: The Unbiased Absolute Percentage Error," Unpublished article, available at: `http://hops.wharton.upenn.edu/forecast/paperpdf/armstrong-unbiasedAPE.pdf`.

[3] Croston, J.D., 1972. "Forecasting and Stock Control for Intermittent Demands," Operational Research Quarterly, vol. 23, no. 3, pp. 289-303.

[4] Galmes, S., Puigjaner, R., 2003. "Correlation analysis of a discrete-time flexible arrival process," Computer Networks, vol. 41, no. 6, pp. 795-814.

[5] Gamma, E., Helm, R., Johnson, R., and Vlissides, J.M. 1994. Design Patterns: elements of reusable object-oriented software, Addison Wesley, Upper Saddle River, NJ.

[6] Hyndman, R.J., Koehler, A.B., 2005. "Another Look at Measures of Forecast Accuracy," Working Paper, Monash University.

[7] Rosetti, M.D. 2008. "JSL: An Open-Source Object-Oriented Framework for Discrete-Event Simulation in Java." International Journal for Simulation and Process Modeling. vol. 4, no. 1, pp. 69-87.

[8] SAS Institute Inc., 2007. "SAS High-Performance Forecasting 2.3: User's Guide," Cary, NC: SAS Institute Inc.

[9] SAS Institute Inc., 2007. "Average Demand Method," SAS High-Performance Forecasting 2.3: User's Guide," Cary, NC: SAS Institute Inc., pp. 432-433.

[10] Syntetos, A.A., Boylan, J.E., 2001. "On the Bias of Intermittent Demand Estimates," International Journal of Production Economics, vol. 71, no. 1-3, pp. 457-466.

[11] Varghese, V.M., Rossetti, M.D., 2009. "A Meta Forecasting Methodology for Large Scale Inventory Systems with Intermittent Demand," Working Paper. University of Arkansas.

[12] Varghese, V.M., Rossetti, M.D., 2008. "A Parametric Bootstrapping Approach to Forecast Interimittent Demand," Proceedings of the 2008 Industrial Engineering Research Conference, Vancouver, Canada, May 17-21.

[13] Venables, W.N., Smith, D.M, 2008. An Introduction to R. Available at: `http://www.r-project.org/`.

[14] Yurkiewicz, J. "Forecasting at Steady State?", OR/MS Today. June 2008. Pg. 54-63.

[15] Zipkin, P.H. 2000. Foundations of Inventory Management. 1st ed. McGraw-Hill, New York.