sas

# Forecasting with SAS®

## Special Collection


Foreword by
Michael Gilliland

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2020. *Forecasting with SAS®: Special Collection*. Cary, NC: SAS Institute Inc.

**Forecasting with SAS®: Special Collection**



ISBN 978-1-951685-75-1 (Paperback)
ISBN 978-1-951685-73-7 (Web PDF)



SAS Institute Inc., SAS Campus Drive, Cary, NC 27513-2414

April 2020

# Table of Contents

# Free SAS® e-Books: Special Collection

In this series, we have carefully curated a collection of papers that introduces and provides context to the various areas of analytics. Topics covered illustrate the power of SAS solutions that are available as tools for data analysis, highlighting a variety of commonly used techniques.


Discover more free SAS e-books!
**support.sas.com/freesasebooks**

sas.com/books
*for additional books and resources.*

sas
THE POWER TO KNOW®

# FORESIGHT

## Connect with the earned expertise of business forecasters and practical research from top academics around the globe.

**Join the IIF to receive both digital and print editions of *Foresight*!**

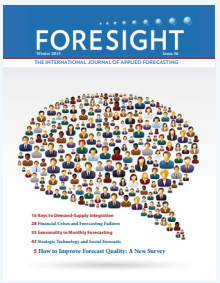
**Foresight *delivers high-quality professional development for those who make and use forecasts in business.***

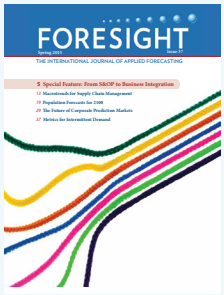
Here's what our readers say:

> ***The information is relevant to practitioners and is presented in a way that is not overly academic but with significant credibility."***
> Thomas Ross, Financial Analyst, ***Brooks Sports***

> ***"...an important forum for practitioners to share their experiences...."***
> Dan Kennedy, Senior Economist, ***Connecticut Department of Labor***

> ***"I find* Foresight *very useful! I use it as a teaching resource to bring theoretical forecasting techniques to life for the students."***
> Dr. Ilsé Botha, Senior Lecturer, ***University of Johannesburg***





***Individual and group memberships available! Corporate memberships are a great way to share important information with your whole team.***

## Join today!

***forecasters.org/membership/join/***

# IBF Membership

## Fostering Innovation in Demand Planning, Forecasting, and S&OP / IBP

- JOURNAL OF BUSINESS FORECASTING (JBF SUBSCRIPTION)
- FREE IBF TRAINING
- FREE ACCESS TO JBF ARTICLE LIBRARY
- RESEARCH REPORTS, BENCHMARK STUDIES
- VALUABLE FORECASTING TEMPLATES
- DEMAND PLANNING, FORECASTING, AND S&OP / IBP SUPPORT
- DISCOUNTED IBF EVENTS, TRAINING, AND CERTIFICATION
- NETWORK WITH 1000'S OF DEMAND PLANNING, FORECASTING, ANALYTICS, S&OP / IBP AND SUPPLY CHAIN PROFESSIONALS GLOBALLY

MASTER DEMAND PLANNING, FORECASTING & S&OP / IBP

# Certification Program

## 3 TYPES OF IBF CERTIFICATION

**CERTIFIED PROFESSIONAL FORECASTER (CPF)**

**ADVANCED CERTIFIED PROFESSIONAL FORECASTER (ACPF)**

**CERTIFIED PROFESSIONAL FORECASTING CANDIDATE (CPFC) FOR STUDENTS & NEW PRACTITIONERS**

*Institute of Business Forecasting & Planning*

PHONE: +1.516.504.7576 | EMAIL: info@ibf.org | www.ibf.org

# Foreword

Nearly every action that we take, and every decision that we make, is based on some kind of forecast – some expectation of what is going to happen in the future. This is particularly true in business where forecasts drive our plans for staffing and resources, inventory and production, and financials. Yet more accurate forecasting is not an end in itself. Rather, more accurate forecasts provide a means to effect better decisions and plans, more appropriate actions, and ultimately, better organizational performance.

The challenge is twofold. First, how do we go about creating better business forecasts? And second, how do we get organizations to recognize and use forecast improvements?

Years of academic research and results from the early M (Makridakis) Forecasting Competitions had two significant findings:

- Complex methods do not necessarily provide more accurate forecasts than simpler ones.
- Combining methods tends to be more accurate than selecting a single method.

It has also been recognized that business forecasting occurs in a highly politicized environment, where the biases and personal agendas of forecasting process participants can tarnish the results. The forecast often ends up reflecting management targets or wishful thinking, instead of being what it should be, an "unbiased best guess at what is really going to happen."

With advances in computational power and data availability, there is renewed interest in applying a broad range of data science methods, including machine learning (ML), to address time series data and forecasting. Yet the impact of ML on forecasting has, so far, been a mixed bag. In the M4 Forecasting Competition held in 2018, the six pure ML entries fared poorly, all falling well below the benchmark combination of three simple time series methods. Yet two entries that used ML methods along with traditional time series methods won the competition. And while these two methods had the most accurate point forecasts and also were the first methods known to produce precise prediction intervals, they were highly complex and resource intensive. For comparison, the M4 benchmark method forecasted 100,000 series in just 33 minutes on a standard hardware configuration. The two "winners" took 6 days and 32 days! (For more results, analysis, and commentary on the M4 competition, see the *International Journal of Forecasting*'s 2020-Q1 issue dedicated to the M4.)

We are in an exciting era of advances in forecasting – both in the technical modeling of time series data, and in our understanding of the effects of forecasting process. SAS offers several solutions to generate high-quality forecasts, including SAS® Visual Forecasting, SAS® Forecast Server, SAS® Forecasting for Desktop, SAS® Econometrics, and SAS/ETS®. And SAS authors have contributed groundbreaking papers to demonstrate the wide range of forecasting techniques available in SAS.

*SAS® for Forecasting: Special Collection* contains a carefully chosen selection of papers on both modeling and process related topics. Some of these applications might surprise you, such as the use of machine learning to guide forecast adjustments, and the use of process control methods to monitor forecasting model performance. Together, these papers (and more listed in the appendix) give you just a sample of what SAS forecasting can offer.

**Ten Underused Features That Improve Your SAS® Forecast Server Workflow**
By Michael Leonard and Evan Anderson

SAS Forecast Server is one of the most feature-rich forecasting products on the market. This paper describes 10 underused features to improve your workflow. First, start with the data: (1) Use SAS® Time Series Studio to become familiar with your data and define a hierarchy, (2) catch data problems through warnings about the time ID variable, (3) look under the covers by using the SAS log, and (4) use adjustment variables and start-up code to fix data issues. Next, improve the forecasts: (5) Define recurring events that influence the time series, (6) add models by importing from an external list, (7) use rolling simulations to evaluate forecast accuracy over the number of periods you need to forecast, (8) evaluate the effects of independent variables by using scenario analysis, and (9) gain insight into results by comparing models. Finally, put your workflow into production: (10) Run the code in batch.

**Using SAS® Forecast Server and the SASEFRED Engine to Enhance Your Forecast**
By Catherine LaChapelle

The SASEFRED interface engine is the best-kept secret in SAS/ETS software. It dramatically reduces the amount of time and effort required to include economic indicator variables in your time series analysis. Using the SASEFRED engine in SAS® Enterprise Guide®, you can directly query the economic database of the Federal Reserve Bank of St. Louis. This public database contains over 529,000 economic time series aggregated from 86 sources. In this paper, we forecast wine demand and enrich predictions via the inclusion of economic variables such as “Retail Sales: Beer, Wine, and Liquor Stores” and “Producer Price Index by Industry: Beer, Wine, and Liquor Stores.” The diversity of economic variables provided in this database ensures that it is useful to virtually every time series analysis and industry. This specific example leverages SAS Enterprise Guide and SAS Forecast Server as interfaces. However, this functionality works on SAS 9.x as well as on SAS Viya technology.

**Regime-Switching Models: Capturing Structural Changes in Time Series**
By Xilong Chen and Ji Shen

Stock market conditions, government policy changes, or even weather patterns can be regarded as stochastic processes that are driven by unobserved regimes. A powerful tool to explore these behavioral patterns is the regime-switching model (RSM) that is offered in the HMM procedure and the associated action in SAS Econometrics software. This model, which is widely used in finance, economics, science, and engineering, has two characteristics: it allows different parameter values for different regimes, and it models the transition probabilities between regimes. These characteristics enable it to fully capture the structural changes in the time series. This paper uses two examples to illustrate how you can use RSMs to better understand the regime patterns in your data and improve your economic analysis.

**Getting More Insight into Your Forecast Errors with the GLMSELECT and QUANTSELECT Procedures**
By Gerhard Svolba

Is it sufficient just to monitor the quality of your forecast models over time? Can data science methods identify the drivers for large forecast errors and provide more insights than descriptive statistics? Do demand planners really improve forecast accuracy with their manual overwrites? Using a real-life case study, this paper answers these questions. It shows how you can study the impact of factors like product group, forecast horizons, seasonality, or the forecast model type on forecast accuracy and convert them into actionable results. You will learn how to use the GLMSELECT, QUANTSELECT, and QUANTREG procedures to identify the most important influential factors on the forecast error. You will also learn how to convert the results from the SAS procedures into actions to improve your forecasting process.

**Monitoring Forecast Models Using Control Charts**
By Joseph Katz

(This article originally appeared in *Foresight: The International Journal of Applied Forecasting* [Issue 56, Winter 2020], and appears with the publisher's permission.)

Inappropriate models result in avoidable forecast bias and error and can consume management resources making manual forecast adjustments. So when should a forecasting model be adjusted or replaced? This article presents a new (patented) application of control charts for automatic monitoring of forecast errors. By using traditional rules for statistical process control along with custom rules, it shows how to determine whether a forecasting model should be maintained, adjusted/refit, or discarded and replaced with a new model.

**Assisted Demand Planning Using Machine Learning for CPG and Retail**
By Charlie Chase, Varunraj Valsaraj, Becky Gallagher, and Roger Baldridge

More than 40 percent of a demand planner's time is spent managing information and data. Another 30 to 40 percent is spent managing and fine-tuning the demand forecast based on new market and customer information, changes in marketing programming (tactics) and coordinating the consensus forecast (plan). Finally, creating and updating KPI reports represents about 10 percent of a demand planner's time. With the introduction of intelligent automation using machine learning, a large portion of the manual, repetitive activities can be automated, which would enable demand planners to be more productive and add real value to the overall process.

**What Management Must Know About Forecasting**
By Michael Gilliland

How many organizations are you aware of – perhaps even your own – that have thrown thousands or even millions of dollars at their forecasting problem, only to end up with the same lousy forecasts? It boils down to two questions: Why do forecasts always seem to be wrong and sometimes terribly wrong? And is there anything we can do about it? This article explores why forecasting is often so poorly done and offers suggestions for improving it. It introduces the concept of Forecast Value Added analysis and shows how the FVA approach can identify and eliminate the worst practices that waste time and harm forecasting performance.

We hope you enjoy this special collection and find valuable ideas to apply in your forecasting challenges. Please join the conversation with your peers by registering in the SAS Forecasting and Econometrics Community where you can ask questions, post answers and comments, and find the latest news and events on SAS forecasting and beyond.

In addition, we encourage you to join the two main professional organizations for forecasting: the International Institute of Forecasters and the Institute of Business Forecasting. These organizations offer conferences, workshops, and certification opportunities, and publish three outstanding journals: *International Journal of Forecasting* (IIF), *Foresight: The International Journal of Applied Forecasting* (IIF), and *Journal of Business Forecasting* (IBF). SAS is closely aligned with both organizations, and we encourage you to check their ads for details about membership and offerings.

For more information and further reading, see the Appendix for recommended books, white papers, and SAS Global Forum papers.

No more bad forecasting!

Michael Gilliland is Marketing Manager for SAS forecasting software, prior to which he held forecasting positions in the food, consumer electronics, and apparel industries. Mike is author of *The Business Forecasting Deal* (2010), principal editor of *Business Forecasting: Practical Problems and Solutions* (2015), writes The Business Forecasting Deal blog, is Associate Editor of *Foresight: The International Journal of Applied Forecasting*, and in 2017 received the Lifetime Achievement Award from the Institute of Business Forecasting. Mike holds a BA in Philosophy from Michigan State University, and master's degrees in Philosophy and Mathematical Sciences from Johns Hopkins University. He is interested in issues relating to forecasting process, such as worst practices and Forecast Value Added analysis, and in applying research findings to real-life improvement in business forecasting.

Paper SAS4440-2020

# Scalable Cloud-Based Time Series Analysis and Forecasting Using Open-Source Software

Javier Delgado, Thiago Quirino, and Michael Leonard, SAS Institute Inc.

## ABSTRACT

Many organizations need to process large numbers of time series for analysis, decomposition, forecasting, monitoring, and data mining. The TSMODEL procedure, available in SAS® Visual Forecasting and SAS Econometrics® software, provides a resilient, distributed, and optimized generic time series analysis environment for cloud computing. PROC TSMODEL offers capabilities such as automatic forecast model generation, automatic variable and event selection, automatic model selection, and parameter optimization. It also provides advanced support for time series analysis (in the time domain or in the frequency domain), time series decomposition, time series modeling, signal analysis and anomaly detection (for IoT), and temporal data mining. In addition, PROC TSMODEL supports open-source integration with external languages Python and R. This paper describes the scripting language that supports cloud-based open-source integration between SAS® software and external languages; examples that demonstrate this use case are provided.

## INTRODUCTION

More information than ever before is being collected with associated timestamps. Computers, mobile phones, smart devices, detectors, and other devices record timestamped data. These timestamped data can be modeled, forecasted, or mined (or any combination of these) for better decision-making. In most cases, the decisions are critical and have immense financial and ethical implications. For example:

- Retailers rely on both seasonal and nonseasonal forecasts of product demand in order to make profitable decisions about staff scheduling and stocking levels for millions of products across thousands of stores.
- Manufacturers rely on accurate forecasts of time to component failure in order to make decisions about the maintenance schedule of critical machinery components.
- Railroad companies rely on accurate time series forecasts of shipping demand per region of the country in order to preemptively stock their railroad cars across different regions. Accurate forecasts enable them to better meet the predicted demand, minimize shipping delays, and improve customer satisfaction.
- Energy companies rely on the ability to both monitor and analyze, in real time, sensor data that stream from wind turbines. Time series of sensor data are analyzed in order to quickly detect and respond to critical anomalous behavior and to maintain their turbines at peak performance over time.
- Hospitals can aggregate patient sensor data, lab results, and physician notes in order to monitor patient progress and better predict patient outcome. Similarly, a physician can monitor a patient's pacemaker remotely in order to quickly determine when the patient's heart is behaving anomalously.
- Governments rely on time series decomposition techniques in order to decompose series of economic variables into their long-term trends and short-term seasonal effects so that they can gain a better insight into the real status of the economy.

In recent years, there has been an enormous increase in the amount of timestamped data being collected. It is now commonplace for companies (such as banks, manufacturers, retailers, websites, hospitals, universities, and governments, in addition to taxi, insurance, stock trading, phone, energy, and many more companies) to maintain large databases of timestamped data whose sizes range from hundreds of gigabytes to hundreds of terabytes. These databases are gold mines for insights into consumer behavior. These insights can help organizations optimize their internal processes to better meet consumer demands.

The amount of timestamped data being collected is expected to further escalate because of the ongoing proliferation of the Internet of Things (IoT). IoT enables all types of objects (cars, toasters, pacemakers, water and gas meters, and so on) to be discovered, monitored, and controlled remotely via the existing internet infrastructure. In short, "big data" has become pervasive in today's society: it is everywhere and in anything, it is here to stay, and it has a lot to say. Processing this ever-increasing amount of timestamped data in an intelligent way poses both architectural and analytical challenges. For example, because of the sheer amount of data and the ever-increasing demand to gain decision-making insights from data in close to real time, time series analysis of big data is inherently a distributed computing problem and is thus an architectural challenge. In addition, big data solutions must be generic enough to accurately handle the time series analysis requirements of different applications and thus are an analytical challenge.

SAS Visual Forecasting provides procedures for some of the most common analyses that are performed on timestamped data: forecasting, decomposition and price analysis, time series monitoring and anomaly detection, and temporal data mining. This paper provides an overview of the SAS Visual Forecasting procedures—in particular of the TSMODEL procedure, which was specifically designed to support advanced, efficient, and cloud-based time series analysis of big data. Particular emphasis is given to integrating Python and R code with PROC TSMODEL in order to enable efficient, massively parallel execution of Python and R programs.

## HOW THE TSMODEL PROCEDURE WORKS

The goal of cloud-based time series analysis and forecasting is to perform an analytical task in a single pass through the data by using a distributed file system or distributed computing environment (or both). Moving data can strain computing resources, whether internal to a node, external (between computing nodes), or both. A single pass through the data allows for enormous performance gains. By providing a system that both moves data and computes efficiently, the TSMODEL procedure makes time series analysis and forecasting possible on an enormous scale. PROC TSMODEL procedure provides a scalable, cloud-based time series analysis environment, which includes a distributed file system, a scripting environment, and parallel data reading, script execution, and data writing. It is designed to run in the SAS Cloud Analytic Services (CAS) run-time environment that is deployed with SAS Visual Forecasting. The following sections describe these elements in more detail.

### DISTRIBUTED FILE SYSTEM

PROC TSMODEL is designed to enable your analysis to use a distributed file system (DFS). A DFS allows for redundant and resilient storage of data; it breaks up large files into chunks and stores each chunk on several storage media. In addition, it makes several redundant copies of each chunk in order to forgo the need for making periodic backup copies. If a particular file system fails, the distributed file system can resiliently heal itself without needing to restore backup copies (which could cause delays). However, the data are not stored contiguously in such a file system, so sorting on a particular file system is not possible. This is particularly problematic for time series analysis, where the ordering of the data is crucial. In addition, the data that are needed for time series analysis might be stored in several files. These distributed files must be read, sorted, and merged with respect to

time in a scalable and efficient way. SAS Visual Forecasting procedures automatically perform all these operations on the input time series data in preparation for the analysis.

Figure 1 illustrates a cluster that consists of four worker nodes and a distributed file system that contains two tables, A and B. Each table is organized by classification (BY) variables that delineate the time series rows, which are grouped into seven BY groups. Each BY group represents one time series. One or more computing (worker) nodes are connected to the distributed file system; neither the tables nor the BY groups are stored on a single machine.

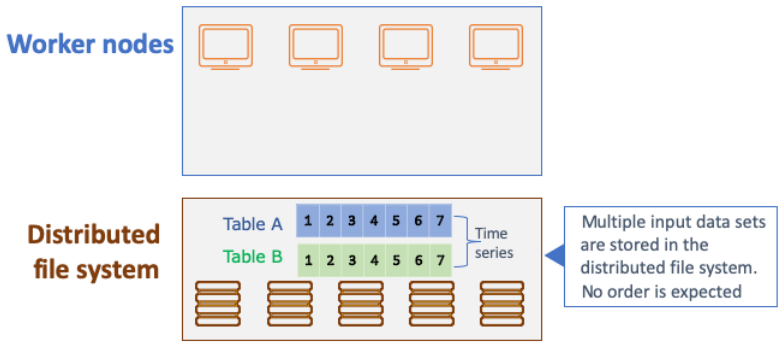



**Figure 1. Distributed File System**

## SCRIPTING LANGUAGE, DISTRIBUTION, AND COMPILATION

The vast amount of data that cloud computing can support calls for a time series analysis environment that allows data to be processed efficiently. SAS Visual Forecasting provides a scripting language that facilitates the use of various capabilities, such as the following:

- automatic forecast model generation, automatic variable and event selection, automatic model selection, and parameter optimization
- advanced support for time series analysis (in the time domain or in the frequency domain), time series decomposition, time series modeling, signal analysis and anomaly detection (for IoT), and temporal data mining
- preparation of the input data prior to analysis and postprocessing of the final results in the same script
- reading of multiple input data files and creation of multiple output data files

These features make the scripting language flexible and useful for numerous applications. Figure 2 illustrates the use of this scripting language. The script is created outside the computing server and can be submitted to the server by SAS, Python, Lua, or R clients.

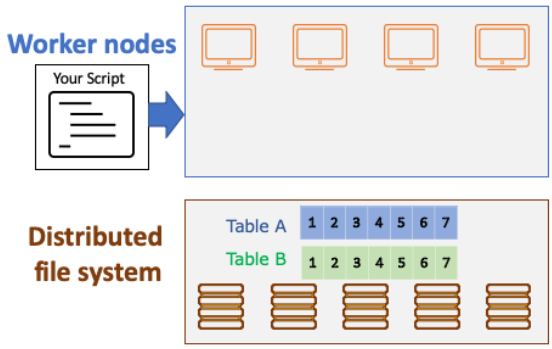



**Figure 2. Scripting Language: User Script Contains SAS Code and Optionally Python and R Code**

The distributed network can consist of one or more computing (worker) nodes. After being submitted to the computing server, the user-specified script is distributed to each worker node to permit parallel execution of the specified analysis, as shown in Figure 3.

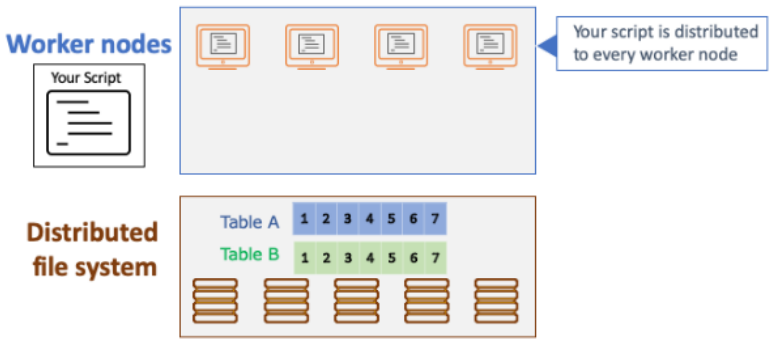
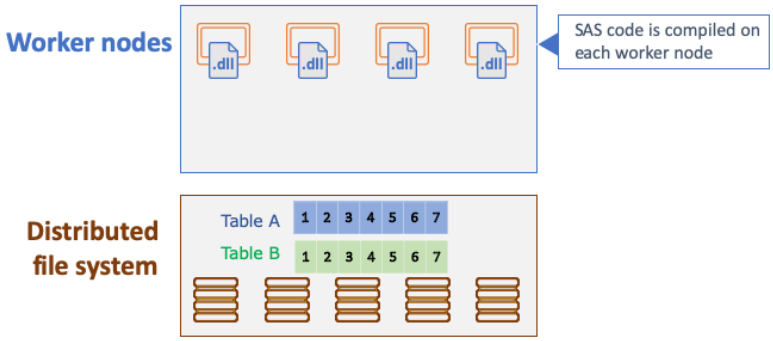



**Figure 3. Script Distribution**

The user-specified script is then compiled on each of the computing nodes. The compiler optimizes the resulting executable for the specific operating system of the computing node (Linux, Windows, and so on). This optimized executable permits very fast execution of the specified analysis. Any external language source code you included in the script is stored in memory. One or more external language interpreters are launched for each thread on each worker node in order to process the external language code at run time.

Figure 4 illustrates the script compilation and execution process when only SAS code is run and when external-language code is integrated. After the script is distributed to the computing (worker) nodes, it is optimally compiled.

*(a) User script contains only SAS code:*




*(b) User script contains SAS and Python code:*

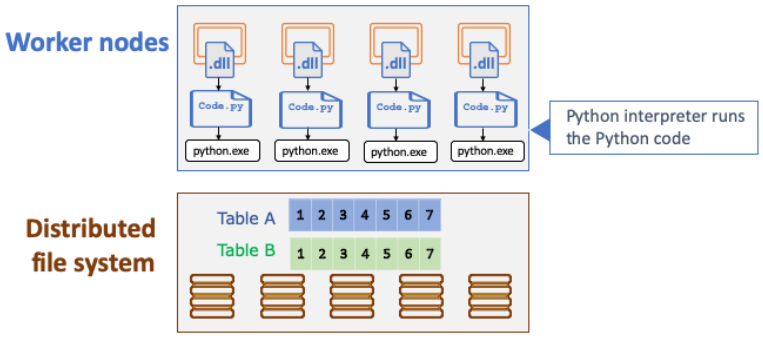



**Figure 4. Script Compilation (a) without and (b) with External-Language Code**

## PARALLEL READ

All the computing nodes read one or more input data files simultaneously. Each input data file contains unsorted, timestamped transactional data that might be recorded at no fixed interval. However, time series analysis algorithms typically require that the input time series data be stored contiguously in memory, in temporal order, and with a fixed-time interval. Therefore, the transactional data must be transformed into a suitable form prior to analysis. PROC TSMODEL relies on the properties of the input data in order to determine how to transform the data for optimal performance. For example, when the input data consist of multiple time series (BY groups), then the transformation occurs via a two-step process that is illustrated in Figure 5 and described in detail in the following sections.

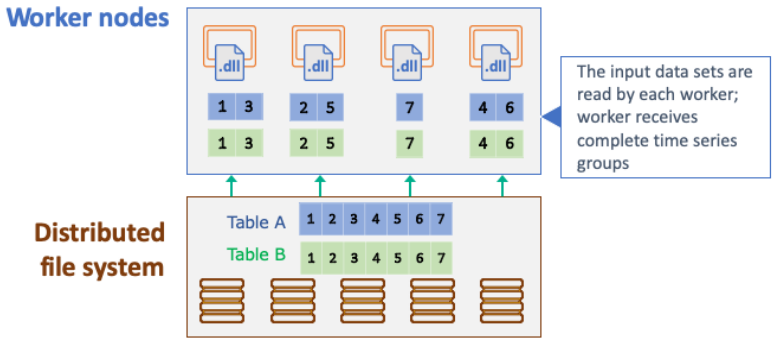



**Figure 5. Parallel Read**

## PARALLEL AND THREADED EXECUTION

Each computing node executes (in parallel) the compiled, optimized script for each time series that has been assigned to it. Each time series is executed on one thread of the computing node. Each of the computing node's threads is kept busy until all the time series that have been assigned to it have been processed. If any problems occur during the execution of a particular time series (BY group), they are logged into an in-memory table so that you can investigate them further. Figure 6 illustrates the parallel execution.

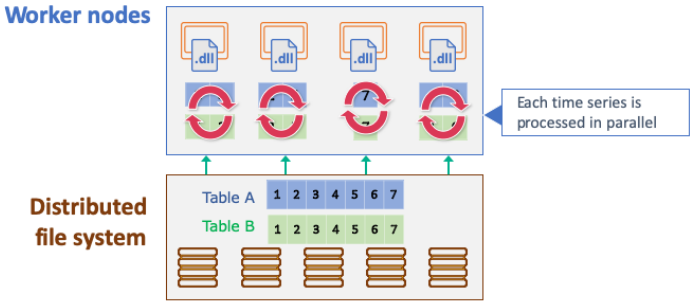



**Figure 6. Parallel Execution**

## PARALLEL AND THREADED EXTERNAL LANGUAGE EXECUTION

The External Languages (EXTLANG) package enables execution of Python and R scripts within the PROC TSMODEL infrastructure. The external-language interpreter is run on the same CAS worker thread where the BY groups data reside, so there is no need for additional internode data transfer. Data are transferred within the worker node and between the SAS process and the external-language interpreter process. Although transfers are backed by a path on disk, the operating system typically uses an in-memory copy of the data, bypassing the need to read the data from disk. On our cluster, we observed a transfer overhead below 2 milliseconds when working with BY-group data sizes of less than 10,000 elements.

### PARALLEL WRITE TO THE DISTRIBUTED FILE SYSTEM

After the specified analysis is executed for a particular time series, the computing nodes write one or more output data sets asynchronously and independently. Multiple output data files can be created simultaneously.

Figure 7 illustrates the parallel write. Each time series analysis result is written back to the distributed file system.

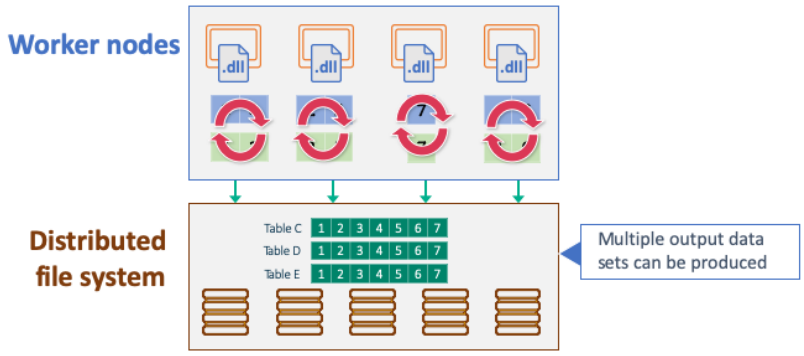



**Figure 7. Parallel Write**

For more information about scalable cloud-based time series analysis and forecasting, see Quirino, Leonard, and Blair (2018).

## IMPLEMENTATION

SAS Visual Forecasting enables you to use a variety of methods (procedures, scripts, packages, and actions) to implement solutions to your time series forecasting problems.

### THE TSMODEL PROCEDURE

The TSMODEL procedure is a SAS® Viya® procedure that executes user-defined programs (scripts) on time series data. PROC TSMODEL analyzes timestamped transactional data with respect to time and accumulates the data into a time series format.

PROC TSMODEL forms time series from timestamped transactional input data and writes the accumulated time series variables to an output table. Time series are delineated by distinct values of the variables that are specified in the BY statement.

Timestamped transactional data are not usually recorded at a fixed interval. Because time series analysis techniques often require fixed-time intervals, the transactional data must be transformed into a fixed-interval time series, such as daily, weekly, or monthly.

PROC TSMODEL forms time series vectors from timestamped data and then provides these vectors as array variables for subsequent processing by program statements, which constitute a script. The script is processed independently for each BY group. The syntax of PROC TSMODEL is the same as that of the TIMEDATA procedure, which is similar to the SAS DATA step for time series data. The SAS DATA step processes data row by row, whereas PROC TSMODEL processes time series vectors (columns) for the BY groups.

For more information about PROC TSMODEL, see *SAS Visual Forecasting: Forecasting Procedures*.

### SCRIPTS

Scripts consist of statements that perform the desired analysis on each time series. For more information about the object-oriented scripting language that PROC TSMODEL supports, see the FCMP procedure in *Base SAS® Procedures Guide*.

## PACKAGES

Packages contain computational services that can be used in your script. A package is a set of related specialized objects and functions (called "methods"), each of which addresses a unique facet of the time series analysis problem. You can use specialized objects and functions to write custom SAS code in order to gain access both to cutting-edge data analysis tools and to utilities that are designed to significantly speed up code development and optimize code quality. Table 1 shows the packages available for PROC TSMODEL.

| Package Name | Description |
|---|---|
| SFS | *Simple Forecast Service:* Tools for automatic forecasting of time series with a simple-to-use interface; these tools use only exponential smoothing (ESM) and ARIMA models |
| ATSM | *Automatic Time Series Modeling And Forecasting:* Tools for automatic modeling and forecasting of time series by using various model families such as exponential smoothing (ESM), ARIMA, intermittent demand (IDM), and unobserved component (UCM) models |
| TSA | *Time Series Analysis:* Tools for efficient statistical analysis of time series (transformations, decompositions, statistical tests for intermittency, seasonality, stationarity, forecast bias, and so on) |
| TSD | *Time Series Distance Measures:* Tools for efficient measure of the distance between two time series or among sequences in temporal data (dynamic time warping, longest common subsequence, and so on) |
| TDR | *Time Series Dimension Reduction:* Tools for efficient time series dimension reduction (symbolic aggregate approximation, discrete Fourier transformation, discrete wavelet transformation, random projection, singular value decomposition) |
| TFA | *Time-Frequency Analysis:* Tools for efficient analysis of time series in both time domain and frequency domain |
| TSM | *Time Series Modeling:* Tools for efficient time series modeling and forecasting |
| SSA | *Singular Spectrum Analysis:* Tools for decomposing a time series into additive components and categorizing those components on the basis of the magnitudes of their contributions |
| MSSA | *Multivariate Singular Spectrum Analysis:* Tools for decomposing one or more time series into additive components and categorizing those components on the basis of the magnitudes of their contributions |
| MTF | *Time Series Motif Discovery:* Tools for the discovery of frequent patterns or repeated subsequences in time series |
| SST | *Subspace Tracking:* Tools for the analysis and decomposition of time series for tracking and monitoring purposes |
| TIMFIL | *Time Series Filters:* Tools for performing various types of filtering and aggregation on time series data |
| UTL | *Utility:* Tools for performing basic statistical computations on pairs of actual and predicted time series |
| EXTLANG | *External Languages:* Tools for enabling seamless integration of external language programs into SAS environments |

**Table 1. Packages Available for the TSMODEL Procedure**

Some of these packages were developed as cloud-based analogues of traditional SAS products and procedures. For example, the ATSM, SFS, and TSM packages carry the features available in SAS® Forecast Server. Similarly, the TSA, TFA, and SSA packages carry various features that are available in SAS/ETS®, albeit with a different scope. For more information about these packages, see *SAS Visual Forecasting: Time Series Packages*.

## ACTIONS

Actions are executed on the CAS workers, using clients available for a variety of languages: SAS, Python, R, and Lua. For more information about actions, see *SAS Visual Forecasting: Programming Guide*.

## OPEN-SOURCE INTEGRATION

In addition to being able to execute actions via SAS, Python, Lua, and R clients, the TSMODEL procedure can now execute R and Python scripts via actions that run on the distributed computing servers (that is, the worker nodes in Figure 4). The computational objects provided by the EXTLANG package to facilitate running Python and R programs on the computational servers are summarized in Table 2. Figure 8 illustrates the object data flow diagram for the EXTLANG package. For more information about the EXTLANG package and other packages, see *SAS Visual Forecasting: Time Series Packages*.

| **Interpreter Object** | **Description** |
|---|---|
| PYTHON2 | Provides support for running code that is written in version 2 of the Python programming language |
| PYTHON3 | Provides support for running code that is written in version 3 of the Python programming language |
| R | Provides support for running code that is written in the R programming language |
| **Output Object** | **Description** |
| OUTEXTCODE | Stores user-supplied external-language source code that is supplied via a PYTHON2, PYTHON3, or R object in a CAS table to "replay" it later |
| OUTEXTLOG | Stores execution and resource usage logs in a table that resides in CAS (a CAS table) |
| OUTEXTVARSTATUS | Collects the status flags of all shared variables and stores them in a CAS table |
| **Input Object** | **Description** |
| INEXTCODE | Reads code from a CAS table and provides it to the external language interpreter for reuse on a per-BY-group basis |

**Table 2. Computational Objects of the EXTLANG Package**

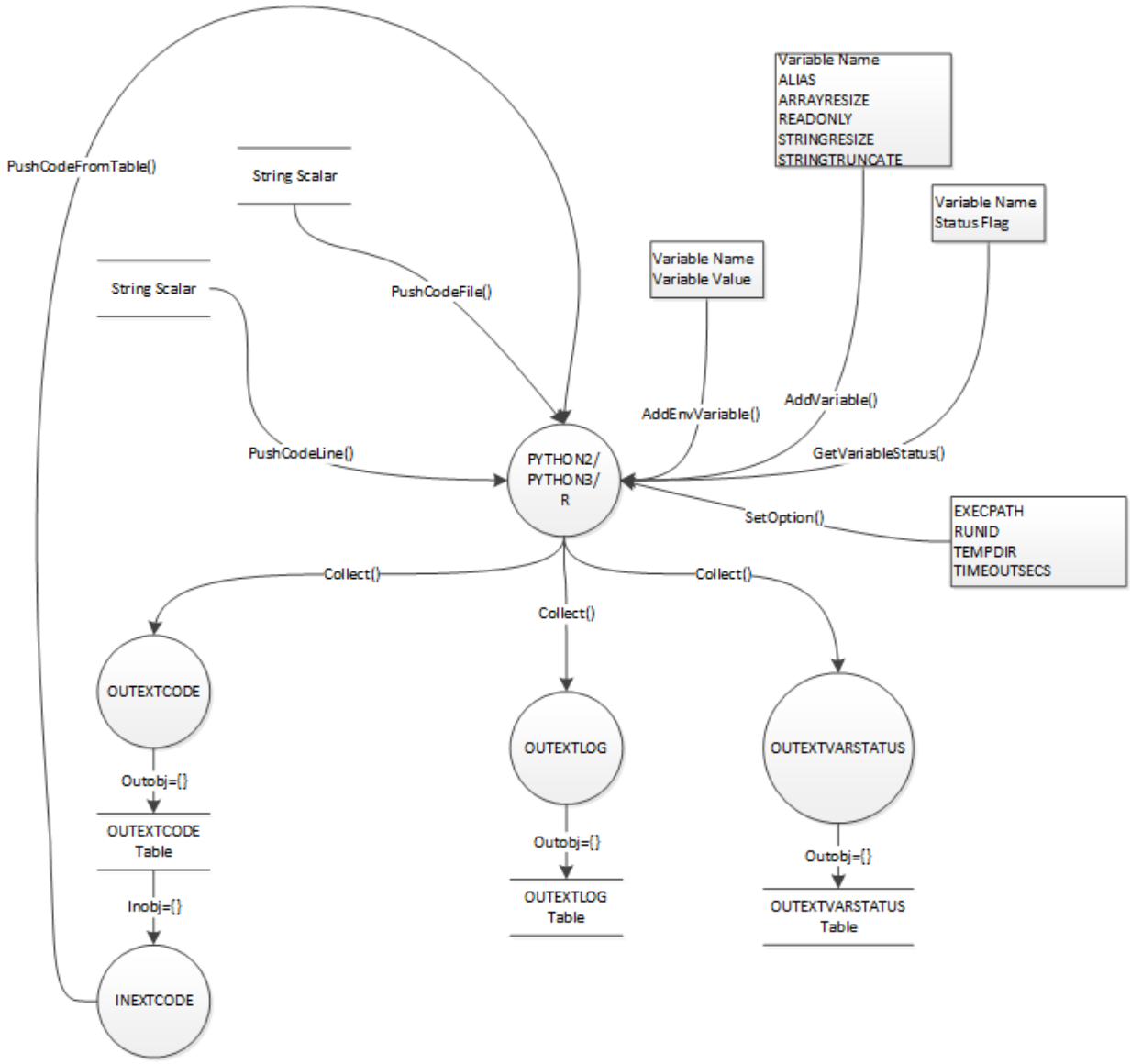



**Figure 8. Object Data Flow Diagram for the EXTLANG Package**

## EXAMPLES

This section provides three examples that demonstrate the capabilities of the TSMODEL procedure, with a specific emphasis on integrating external language programs. The first example illustrates how you can input Python code directly into your SAS script to calculate a simple moving average. The second example shows you can fit an ARIMA model that is implemented in R to your SAS data set. The third example shows the procedure's ability to perform fast time series analysis on big data. All examples use a SAS script as the client to run actions on the CAS server, but you can use any supported CAS client, including Python, R, or Lua.

### EXAMPLE 1: MOVING AVERAGE USING PYTHON

This example demonstrates how to use the EXTLANG package to calculate a simple moving average of a SAS data set (Sashelp.PriceData ) and transfer the value back to your SAS program. This data set consists of simulated monthly sales data that are hierarchically organized by region, line, and product. The Sashelp.PriceData data set contains 1,020

observations, which are divided into 17 BY groups. By virtue of running within PROC TSMODEL, each BY group is processed in parallel in a separate thread on each worker in the CAS cluster. Although this data set is relatively small, the sample code in this section can also readily handle time series data sets that contain millions of BY groups.

First, a connection to CAS (that is, a session) is established, and a CAS library called *mycas* is created. The *mycas* library enables you to transfer data sets to the CAS nodes, where the distributed time series analysis is performed.

```
CAS mycas;
LIBNAME mycas CAS SESSREF = mycas;
```

A DATA step transfers the *Sashelp.PriceData* data set into the CAS *mycas* library:

```
DATA mycas.pricedata;
  SET sashelp.pricedata;
RUN;
```

The PROC TSMODEL statement specifies the input data set (*mycas.pricedata*), an output table in which to store the output data array (*mycas.outarray*), an output table in which to store the scalar variables (*mycas.outscalar*), and an output table in which to store an output object (*mycas.pylog*).

```
PROC TSMODEL DATA=mycas.pricedata OUTARRAY=mycas.outarray
             OUTSCALAR=mycas.outscalar
             OUTOBJ=(pylog=mycas.pylog);
```

The ID statement specifies the variable **date** as the time index variable, and the INTERVAL= option indicates that the data are monthly.

```
ID date INTERVAL = MONTH;
```

The BY statement specifies that each unique combination of the data set variables **regionname**, **productline**, and **productname** corresponds to a unique time series BY group. BY groups are processed independently.

```
BY regionname productline productname;
```

The VAR statement specifies the input data set variable **SALE**. The ACCUMULATE=AVG option specifies an average value accumulation for the **SALE** variable.

```
VAR SALE / ACCUMULATE = AVG;
```

The OUTSCALAR statement specifies the scalar variables that the SAS script is to generate and store. These include a variable in which to store the Python program's execution time (**runtime**), a variable in which to store the exit code (**exitCode),** and variables in which to store the return code from each PYTHON2 object's method call (**rc1–rc6**).

```
OUTSCALAR runtime exitCode rc1 rc2 rc3 rc4 rc5 rc6;
```

The OUTARRAY statement specifies the array variables that the program is to generate and store. The only output array is the moving average (MAVG).

```
OUTARRAY MAVG;
```

The REQUIRE statement specifies the EXTLANG package, which includes all the objects that PROC TSMODEL needs in order to interact with external languages.

```
REQUIRE EXTLANG;
```

The program statements between the SUBMIT and ENDSUBMIT statements use the ATSM package objects to perform the actual analysis on the CAS server:

```
SUBMIT;

    /*
     * Initialize the PYTHON2 object, which is the interface to the
     * Python interpreter.
     */
    declare object py(PYTHON2);
    rc = py.Initialize();

    /*
     * Create the Python program, which simply does the following:
     *  1. Import the NumPy package with alias np
     *  2. Create an array to be used for the moving average computation,
     *     with a window size of 3
     *  3. Compute the moving average and store into variable MAVG
     */
    rc1 = py.PushCodeLine('import numpy as np');
    rc2 = py.PushCodeLine('w = np.ones((3,))/3 ; ');
    rc3 = py.PushCodeLine('MAVG = np.convolve(SALE, w, mode="same")');

    /*
     * Specify variables to share between SAS and Python.
     * The variable SALE is used only as input in the Python program;
     * the default value of READONLY is used to avoid propagating
     * its data back to SAS. MAVG is transferred back to SAS, where
     * it is stored in a CAS table for further analysis.
     */
    rc4 = py.AddVariable(SALE) ;
    rc5 = py.AddVariable(MAVG, 'READONLY', 'FALSE');


    /*
     * Run the program and obtain the run time and exit code.
     */
    rc6 = py.Run();
    runtime = py.GetRuntime();
    exitCode = py.GetExitCode() ;

    /*
     * Store the execution and resource usage statistics logs.
     */
    declare object pylog(OUTEXTLOG);
    rc = pylog.Collect(py,'ALL');

ENDSUBMIT;
RUN;
```

The TSMODEL procedure prints a summary of the time series processing that is performed, as shown in Output 1. This summary includes the number of BY groups that are processed, the total processing time, and some information about the accumulation process.

The SAS System

The TSMODEL Procedure

| Processing Summary | |
|---|---|
| CAS table | PRICEDATA |
| Number of analysis variables | 1 |
| Number of rows read | 2040 |
| Number of groups read | 17 |
| Memory for group packages (KB) | 2 |
| Time to load groups (seconds) | 0.92522192 |
| Minimum time ID | JAN1998 |
| Maximum time ID | DEC2002 |
| Minimum time periods | 60 |
| Maximum time periods | 60 |
| Number of nodes run | 121 |
| Number of nodes with data | 15 |
| Number of nodes with groups | 15 |
| Number of threads budgeted | 40 |
| Minimum thread group count | 0 |
| Maximum thread group count | 1 |
| Minimum threads active | 32 |
| Maximum threads active | 80 |
| Number of groups processed by submitted code | 17 |
| Number of groups failing | 0 |
| Elapsed time to process groups (seconds) | 1.7797710896 |
| Number of array table rows produced | 1020 |
| Number of scalar table rows produced | 17 |

**Output 1. Summary for Processing Example 1**

Output 2 shows a subset of the scalar output, which is produced from the following code:

```
PROC PRINT DATA=mycas.outscalar; RUN;
```

You can verify from Output 2 that all exit and return codes are 0.

| Obs | regionName | productLine | productName | _STATUS_ | runtime | exitCode | rc1 | rc2 | rc3 | rc4 | rc5 | rc6 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Region1 | Line1 | Product3 | 0 | 0.4798159599 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | Region2 | Line2 | Product7 | 0 | 0.4799671173 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | Region2 | Line3 | Product8 | 0 | 0.4737679958 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | Region3 | Line5 | Product16 | 0 | 0.4736320972 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | Region3 | Line4 | Product12 | 0 | 0.416203022 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | Region2 | Line3 | Product11 | 0 | 0.4436099529 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | Region3 | Line4 | Product15 | 0 | 0.4660289288 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | Region1 | Line1 | Product1 | 0 | 0.4725699425 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | Region3 | Line4 | Product13 | 0 | 0.7767958641 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | Region2 | Line2 | Product6 | 0 | 0.45221591 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Output 2. Partial Output of OUTSCALAR Table Produced by Example 1**

Output 3 shows the first 10 lines of the OUTARRAY table from this example. You can see the moving average values (MAVG) in the last column. Note that values are obtained at the

boundaries since the convolution mode was set to "same." This output is generated via the following command:

```
PROC PRINT DATA=mycas.outarray; RUN;
```

| Obs | regionName | productLine | productName | _STATUS_ | _SEASON_ | _CYCLE_ | date | sale | MAVG |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Region1 | Line1 | Product3 | 0 | 1 | 1 | JAN1998 | 300 | 200.33333333 |
| 2 | Region1 | Line1 | Product3 | 0 | 2 | 2 | FEB1998 | 301 | 320 |
| 3 | Region1 | Line1 | Product3 | 0 | 3 | 3 | MAR1998 | 359 | 352.33333333 |
| 4 | Region1 | Line1 | Product3 | 0 | 4 | 4 | APR1998 | 397 | 366.66666667 |
| 5 | Region1 | Line1 | Product3 | 0 | 5 | 5 | MAY1998 | 344 | 362 |
| 6 | Region1 | Line1 | Product3 | 0 | 6 | 6 | JUN1998 | 345 | 344.33333333 |
| 7 | Region1 | Line1 | Product3 | 0 | 7 | 7 | JUL1998 | 344 | 342.33333333 |
| 8 | Region1 | Line1 | Product3 | 0 | 8 | 8 | AUG1998 | 338 | 341.66666667 |
| 9 | Region1 | Line1 | Product3 | 0 | 9 | 9 | SEP1998 | 343 | 334.66666667 |
| 10 | Region1 | Line1 | Product3 | 0 | 10 | 10 | OCT1998 | 323 | 324.66666667 |

**Output 3. Partial Output of OUTARRAY Table of Example 1**

## EXAMPLE 2: ARIMA FORECASTING USING R

This is a more realistic example, which demonstrates how to apply an ARIMA model implemented in R to your data. To keep the example succinct, the R model is used exclusively. However, the example can be extended to work with other objects to do things like include the custom model in the automated model selection process that is provided by the ATSM package objects; see *SAS Visual Forecasting: Time Series Packages*. This example also demonstrates how you can load source code from a file. The freely available forecast package[1] for R is required for this example.

As with the previous example, the first step is to establish a connection to the CAS server and create a CAS library called *mycas*. The *mycas* library enables you to transfer data sets to the CAS server where the distributed time series analysis is performed.

```
CAS mycas;
LIBNAME mycas CAS SESSREF = mycas;
```

A DATA step transfers the *Sashelp.PriceData* data set into the CAS *mycas* library:

```
DATA mycas.pricedata;
  SET sashelp.pricedata;
RUN;
```

The PROC TSMODEL statement specifies the input data set (*mycas.pricedata*), an output table in which to store the output data set (*mycas.outarray*), an output table in which to store one or more scalar variables (*mycas.outscalar*), and an output table in which to store two output objects (the object *mycas.rlog* stores all the output that is generated by the R program and the object *rvars* stores information about shared variables). LEAD=12 requests that 12 time steps into the future be forecasted.

```
PROC TSMODEL DATA=mycas.pricedata OUTARRAY=mycas.outarray
             OUTSCALAR=mycas.outscalar
             OUTOBJ=(rlog=mycas.rlog rvars=mycas.rvars)
             LEAD=12;
```

[1] https://cran.r-project.org/web/packages/forecast/forecast.pdf

The ID statement specifies the variable **date** as the time index variable, and the INTERVAL= option indicates that the data are monthly.

```
ID date INTERVAL = MONTH;
```

The BY statement specifies that each unique combination of the data set variables **regionname**, **productline**, and **productname** correspond to a unique time series BY group. BY groups are processed independently.

```
BY regionname productline productname;
```

The VAR statement specifies the input data set variable **SALE**. The ACCUMULATE=AVG option specifies an average value accumulation for the **SALE** variable.

```
VAR SALE / ACCUMULATE = AVG;
```

The OUTSCALAR statement specifies the scalar variables that the SAS script is to generate and store. These include variables in which to store the R program's run time (**runtime**), exit code (**exitCode**), and the return code from each method called for the R object (**rc1–rc7**):

```
OUTSCALAR runtime exitCode rc1 rc2 rc3 rc4 rc5 rc6 rc7;
```

The OUTARRAY statement specifies the array variables that the SAS script is to generate and store. The only output array is the series that is modeled using the R ARIMA model (**rPred**).

```
OUTARRAY rPred;
```

The REQUIRE statement specifies the EXTLANG package, which includes all the objects that are needed for SAS to interact with external languages.

```
REQUIRE EXTLANG;
```

The program statements between the SUBMIT and ENDSUBMIT statements use the EXTLANG package objects to run the R program on the CAS server:

```
SUBMIT;

    /*
     * Initialize the R object, which is the interface to the
     * R interpreter. The interpreter executable is set via a
     * CAS configuration file.
     */
    declare object robj(R) ;
    rc1 = robj.Initialize() ;

    /*
     * Push code from the filesystem. The R object will dynamically create
     * a file that contains all source code to be run and will autogenerate
     * code for transferring to and from the SAS environment.
     * The file r_arima_code.r has the following contents:
          ------------------------------------------------------
```

```
            library(forecast)
            Y<- Y[1:(NFOR - HORIZON)]
            Y_ts<-ts(Y,frequency=12)
            LOG_Y_ts<-log(Y_ts)
            fit <- stats::arima(LOG_Y_ts, order=c(p=0, d=1, q=1),
            seasonal=list(order=c(0,1,1), frequency=12))
            sse<-sum(fit$residuals^2)
            forecast(fit)
            a <- stats::predict(fit, n.ahead=HORIZON)
            PREDICT <- c( exp(fitted.values(fit)), exp(a$pred) )
            ----------------------------------------------------
         */
        rc2 = robj.PushCodeFile('/path/to/r_arima_code.r') ;

        /*
         * Specify variables to share between SAS and R.
         * SALE is the (READONLY) dependent variable. The ARIMA code
         * uses the generic name Y for the dependent variable, so
         * SALE is aliased to Y.
         * rPred will contain the predicted series, which is returned to the
         * SAS program. The R code that is used stores the predicted series in
         * the variable PREDICT, so rPred is aliased to PREDICT.
         * Two additional read-only variables are needed by the R code:
         * NFOR, which stores the forecast length, and HORIZON, which stores
         * the forecast horizon.
         */
        rc3 = robj.AddVariable(SALE, 'ALIAS', 'Y') ;
        rc4 = robj.AddVariable(rPred, 'ALIAS', 'PREDICT', 'READONLY', 'FALSE');
        rc5 = robj.AddVariable(_LENGTH_, 'ALIAS', 'NFOR') ;
        rc6 = robj.AddVariable(_LEAD_,'ALIAS','HORIZON') ;

         /*
          * Run the model and get the exit code and run time.
          */
        rc7 = robj.Run() ;
        exitCode = robj.GetExitCode() ;
        runtime = robj.GetRunTime() ;

        /*
         * Store the execution and resource usage statistics logs.
         */
        declare object rlog(OUTEXTLOG) ;
        rc16 = rlog.Collect(robj, 'EXECUTION') ;
        declare object rvars(OUTEXTVARSTATUS) ;
        rc17 = rvars.collect(robj) ;

ENDSUBMIT;
RUN;
```

The PROC TSMODEL summary is shown in Output 4.

The TSMODEL Procedure

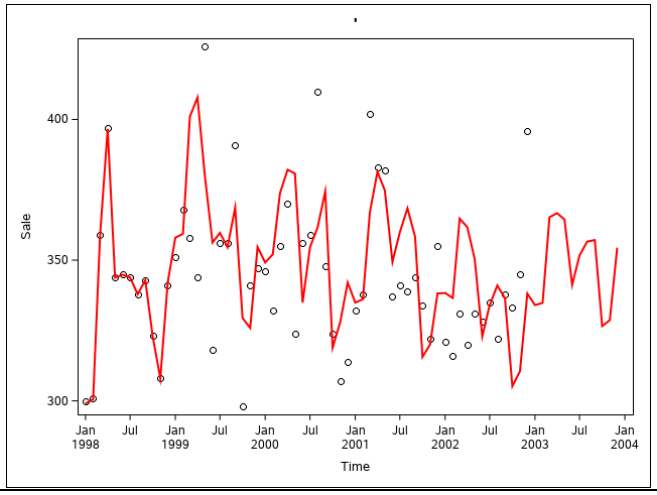
| Processing Summary | |
|---|---|
| CAS table | PRICEDATA |
| Number of analysis variables | 1 |
| Number of rows read | 2040 |
| Number of groups read | 17 |
| Memory for group packages (KB) | 2 |
| Time to load groups (seconds) | 5.2867841721 |
| Minimum time ID | JAN1998 |
| Maximum time ID | DEC2002 |
| Minimum time periods | 60 |
| Maximum time periods | 60 |
| Number of nodes run | 121 |
| Number of nodes with data | 15 |
| Number of nodes with groups | 15 |
| Number of threads budgeted | 40 |
| Minimum thread group count | 0 |
| Maximum thread group count | 1 |
| Minimum threads active | 32 |
| Maximum threads active | 80 |
| Number of groups processed by submitted code | 17 |
| Number of groups failing | 0 |
| Elapsed time to process groups (seconds) | 6.8440380096 |
| Number of array table rows produced | 1224 |
| Number of scalar table rows produced | 17 |

**Output 4. PROC TSMODEL Summary Statistics for Example 2**

The following code produces a plot that shows the actual and forecasted units sold over time for a simple BY group. The forecast values come from the R ARIMA model that is used in Example 2. The DATA step creates a subseries that consists of the BY group that pertains to Region 1, Product Line 1, and Product name "Product 3". The SGPLOT procedure uses the subseries that was obtained in the DATA step to create a scatter plot of the actual SALE values along with a line plot of the values that were obtained by the R ARIMA model. The output is shown in Output 5.

```
DATA mycas.subseries ;
    set mycas.outarray(where=(regionName="Region1" and productLine="Line1"
and productName="Product3")) ;
RUN ;

PROC SGPLOT data=mycas.subseries ;
   scatter x=DATE y=SALE / markerattrs=(color=black) LEGENDLABEL = 'Actual'
Name="act";
   series x=DATE y=rPRED / lineattrs=(color=red thickness=2)
                                LEGENDLABEL = 'Predicted' Name="pred";
   yaxis LABEL="Sale" ;
   xaxis LABEL="Time" ;
   keylegend / POSITION=BOTTOMRIGHT LOCATION=INSIDE ACROSS=1 ;
RUN;
```




**Output 5. Actual and Modeled Sales for R-Based ARIMA Model**

This R program is not expected to produce any text output if no errors occur, so _LOGTEXT_ should be empty. This can be verified by looking at the *mycas.rlog* table. The following code produces this table:

```
PROC PRINT DATA=mycas.rlog; RUN;
```

The output is shown in Output 6.

| Obs | regionName | productLine | productName | _RUNID_ | _EXITCODE_ | _RUNTIMESECONDS_ | _LOGTYPE_ | _LOGLEN_ | _LOGTEXT_ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Region1 | Line1 | Product3 | UNNAMED RUN | 0 | 2.899353981 | EXECUTION | 0 | |
| 2 | Region2 | Line2 | Product7 | UNNAMED RUN | 0 | 2.8814299107 | EXECUTION | 0 | |
| 3 | Region2 | Line3 | Product8 | UNNAMED RUN | 0 | 2.9811868668 | EXECUTION | 0 | |
| 4 | Region3 | Line5 | Product16 | UNNAMED RUN | 0 | 2.980383873 | EXECUTION | 0 | |
| 5 | Region3 | Line4 | Product12 | UNNAMED RUN | 0 | 2.7156209946 | EXECUTION | 0 | |
| 6 | Region2 | Line3 | Product11 | UNNAMED RUN | 0 | 2.7369279861 | EXECUTION | 0 | |
| 7 | Region3 | Line4 | Product15 | UNNAMED RUN | 0 | 2.7010149956 | EXECUTION | 0 | |
| 8 | Region1 | Line1 | Product1 | UNNAMED RUN | 0 | 2.7325601578 | EXECUTION | 0 | |
| 9 | Region3 | Line4 | Product13 | UNNAMED RUN | 0 | 3.2494049072 | EXECUTION | 0 | |
| 10 | Region2 | Line2 | Product6 | UNNAMED RUN | 0 | 2.7260670662 | EXECUTION | 0 | |
| 11 | Region3 | Line5 | Product17 | UNNAMED RUN | 0 | 2.7100639343 | EXECUTION | 0 | |
| 12 | Region2 | Line3 | Product10 | UNNAMED RUN | 0 | 2.7092959881 | EXECUTION | 0 | |
| 13 | Region2 | Line2 | Product4 | UNNAMED RUN | 0 | 2.7104790211 | EXECUTION | 0 | |
| 14 | Region1 | Line1 | Product2 | UNNAMED RUN | 0 | 4.3636190891 | EXECUTION | 0 | |
| 15 | Region2 | Line3 | Product9 | UNNAMED RUN | 0 | 3.3587551117 | EXECUTION | 0 | |
| 16 | Region2 | Line2 | Product5 | UNNAMED RUN | 0 | 2.7048690319 | EXECUTION | 0 | |
| 17 | Region3 | Line4 | Product14 | UNNAMED RUN | 0 | 3.8954088688 | EXECUTION | 0 | |

**Output 6. Output of RLOG Table for Example 2**

## EXAMPLE 3: LIGHTNING-FAST BIG DATA ANALYSIS

This example demonstrates the TSMODEL procedure's ability to perform fast time series analysis on big data. By using the EXTLANG package objects with PROC TSMODEL, any Python or R code can be seamlessly parallelized at a large scale, with minimal overhead. In this example, a large industrial data set that consists of more than 1.5 million BY groups is processed. On average, each BY group contains 4.3 years of weekly historical data. This example was conducted on a cluster of 128 worker nodes and 32 BY-group threads per node. Each worker node contains dual Intel Xeon E5-2680 CPUs; each CPU has 8 cores with Intel Hyper-Threading Technology. Each worker has 252 GB of RAM.

First, a connection to the CAS server is established, and a CAS library called *mycas* is created. The *mycas* library enables you to transfer data sets to the CAS server where the distributed time series analysis is performed:

```
CAS mycas;
LIBNAME mycas CAS SESSREF = mycas;
```

The DATA step is omitted from this example because the data are not publicly available. The PROC TSMODEL statement is similar to the previous examples. The details about the statements before the SUBMIT…END SUBMIT block are omitted for brevity.

```
PROC TSMODEL data = mycas.large_distributor
          OUTARRAY=mycas.outarray
          OUTSCALAR=mycas.outscalar
          LEAD=52
         ;
 ID  period_start_dt
     interval = week
     setmissing = missing
     trimid = none
 ;
 BY customer_id product_id store_location_id;
 VAR demand_qty / acc = total;
 OUTSCALAR pyTime pyExitCode pyProcessingTime
           prc1 prc2 prc3 prc4 prc5 prc6 prc7 prc8 prc9 prc10
           prc11 prc12 prc13 prc14 prc15 prc16
           ;
 OUTARRAY pyPred ;
 REQUIRE extlang ;
```

The code within the SUBMIT…ENDSUBMIT block runs on all the BY groups. The Python version 3 interpreter that is specified in the CAS configuration is launched once for each of the 32 worker threads. The interpreter process is duplicated for each new BY group that the thread processes. Python modules that are used in the user program are preloaded in the duplicated interpreter process to further reduce overhead. Note that Python's indention rules must be obeyed.

```
SUBMIT;

 declare object py(PYTHON3);
 prc1=py.Initialize();

 /* Create the script */
 prc1 = py.PushCodeLine('import numpy as np');
 prc2 = py.PushCodeLine("import os") ;
 prc3 = py.PushCodeLine("import time") ;
 prc4 = py.PushCodeLine('start = time.time()');
 prc5 = py.PushCodeLine('try:');
 /* Moving average with window size = 7 */
 prc6 = py.PushCodeLine('   w = np.ones((7,))/7 ; ');
 prc7 = py.PushCodeLine('   fit = np.convolve(Y, w, mode="same")') ;
 prc8 = py.PushCodeLine('   PREDICT = fit') ;
 prc10 = py.PushCodeLine('except Exception as e:');
 prc11 = py.PushCodeLine('   print("Error occured during computation.
                             Y values: {0}. Error: {1}".format(Y, e))') ;
 prc12 = py.PushCodeLine('PYPROCESSINGTIME = time.time() - start') ;

 /* Add variables */
 prc13 = py.AddVariable(demand_qty, 'ALIAS', 'Y') ;
```

```
 prc14 = py.AddVariable(pyPred, 'ALIAS', 'PREDICT', 'READONLY', 'FALSE');
 prc15 = PY.AddVariable(pyProcessingTime, 'READONLY', 'FALSE');

 /* Run the program */
 prc16 = py.Run();
 pyTime = py.GetRuntime();
 pyExitCode = py.GetExitCode() ;
ENDSUBMIT;
RUN;
```

The results for this example are split into two parts. The first part contains results that are obtained by running without any Python code. These results assess the overhead of just *shuffling* the data among the worker nodes. The second part runs the preceding code. Hence, the second part of the results shows the additional overhead of loading the Python interpreter and transferring data between the interpreter and SAS, in addition to shuffling the data. The output summary of the first part is shown in Output 7, which shows that 1,562,593 BY groups were processed in 29.1 seconds. The output summary from the full example is shown in Output 8, which shows that processing the same BY groups took 153.5 seconds. Given the simplicity of this program, most of this time can be attributed to the overhead involved in duplicating the Python process for each BY group and loading and storing their data. Hence, the penalty for processing 750 million rows of data distributed among 1.5 million BY groups was just 124 seconds, which is quite small.

**The SAS System**

**The TSMODEL Procedure**

| Processing Summary | |
|---|---|
| CAS table | LARGE_DISTRIBUTOR |
| Number of analysis variables | 1 |
| Number of rows read | 750222336 |
| Number of groups read | 1562593 |
| Memory for group packages (KB) | 231948 |
| Time to load groups (seconds) | 20.797576904 |
| Minimum time ID | Sun, 27 Dec 2009 |
| Maximum time ID | Sun, 30 Mar 2014 |
| Minimum time periods | 223 |
| Maximum time periods | 223 |
| Number of nodes run | 121 |
| Number of nodes with data | 120 |
| Number of nodes with groups | 120 |
| Number of threads budgeted | 40 |
| Minimum thread group count | 129 |
| Maximum thread group count | 483 |
| Minimum threads active | 32 |
| Maximum threads active | 80 |
| Number of groups processed by submitted code | 1562593 |
| Number of groups failing | 0 |
| Elapsed time to process groups (seconds) | 29.101961136 |
| Number of array table rows produced | 429713075 |
| Number of scalar table rows produced | 1562593 |

**Output 7. PROC TSMODEL Summary for Loading Large Data Set Using 121 Workers**

The SAS System

The TSMODEL Procedure

| Processing Summary | |
|---|---|
| CAS table | LARGE_DISTRIBUTOR |
| Number of analysis variables | 1 |
| Number of rows read | 750222336 |
| Number of groups read | 1562593 |
| Memory for group packages (KB) | 231948 |
| Time to load groups (seconds) | 141.43509316 |
| Minimum time ID | Sun, 27 Dec 2009 |
| Maximum time ID | Sun, 30 Mar 2014 |
| Minimum time periods | 223 |
| Maximum time periods | 223 |
| Number of nodes run | 121 |
| Number of nodes with data | 120 |
| Number of nodes with groups | 120 |
| Number of threads budgeted | 40 |
| Minimum thread group count | 129 |
| Maximum thread group count | 483 |
| Minimum threads active | 32 |
| Maximum threads active | 80 |
| Number of groups processed by submitted code | 1562593 |
| Number of groups failing | 0 |
| Elapsed time to process groups (seconds) | 153.51236606 |
| Number of array table rows produced | 429713075 |
| Number of scalar table rows produced | 1562593 |

**Output 8. PROC TSMODEL Summary for Example 3**

## CONCLUSION

The TSMODEL procedure enables both scalable and optimized time series analysis in a cloud environment. PROC TSMODEL comes equipped with a generic scripting environment, which enables you to develop custom time series analysis algorithms and prepare your data for analysis (clean, transform, preprocess, and postprocess), all within the same script. This environment helps reduce data movement (because the data remain in the same contiguous memory throughout the analysis) and also optimizes code development. PROC TSMODEL also comes equipped with various specialized time series analysis packages that provide advanced support for time series analysis (in the time domain or in the frequency domain), time series decomposition, time series modeling, signal analysis and anomaly detection (for IoT), and temporal data mining. External language support extends the SAS scripting environment to allow for open-source integration. You can integrate new and existing Python and R programs into your SAS script in order to enhance your processing or analysis (or both). Because of these features, what can be accomplished by PROC TSMODEL is limited only by your imagination. As is illustrated by the third example, PROC TSMODEL's distributed nature (processing BY groups in parallel), efficiency and scalability (minimizing I/O, performing all data operations in memory, and reusing allocated memory efficiently across BY groups), and optimized time series modeling and forecasting capabilities enable big data forecasting problems to be solved with unprecedented speeds. In summary, TSMODEL can efficiently perform time series analysis of big data in close to real time.

## REFERENCES

Quirino, T., Leonard, M., and Blair, E. 2018. "Scalable Cloud-Based Time Series Analysis and Forecasting." *Proceedings of the SAS Global Forum 2018 Conference*. Cary, NC: SAS Institute Inc.

Available http://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2020/2020-4440.pdf

## ACKNOWLEDGMENTS

The authors would like to thank Anne Baxter from SAS for her editing and Ed Blair for his technical insight.

## CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Javier Delgado
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
919-531-4385
Javier.Delgado@sas.com

Thiago Quirino
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
919-531-3721
Thiago.Quirino@sas.com

Michael Leonard
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
919-531-6967
Michael.Leonard@sas.com

**Paper SAS2020-2018**

# Time Series Feature Extraction

Michele A. Trovero and Michael J. Leonard, SAS Institute Inc.

## ABSTRACT

Feature extraction is the practice of enhancing machine learning by finding characteristics in the data that help solve a particular problem. For time series data, feature extraction can be performed using various time series analysis and decomposition techniques. In addition, features can be obtained by sequence comparison techniques such as dynamic time warping and by subsequence discovery techniques such as motif analysis. This paper surveys some of the time series feature extraction methods and demonstrates them through examples that use SAS/ETS® and SAS® Visual Forecasting software.

## INTRODUCTION

In the data mining and machine learning literature, *feature extraction* refers to the process of creating new *features* from an initial set of data. These features encapsulate the central properties of a data set and represent it in a low-dimensional space that facilitates the learning process. The initial data set of raw features might be too large and unwieldy to be effectively managed and might require an unreasonable amount of computing resources. Feature extraction can be used to provide a more manageable, representative subset of input variables.

In recent years, with the growing amount of timestamped data being collected, there has been an explosion of interest in applying machine learning and data mining techniques to timestamped data. For example, websites and transactional databases collect copious amounts of timestamped data that are related to an organization's suppliers or customers (or both) over time. Mining these data can help business leaders make better decisions by enabling them to better understand their relationship with their suppliers or customers via their transactions collected over time. Likewise, a business might have a set of transactions associated with each of its many suppliers and customers. However, each set of transactions might be quite large, making it difficult to perform many traditional data mining tasks.

Most existing data mining tools cannot be used efficiently on time series data. Therefore, a dimension reduction is required through feature extraction techniques that map each time series to a lower-dimensional space.

This paper reviews some commonly used methods of feature extraction for time series. The goal is not to describe them in detail, but rather to provide a brief overview and then point to more information for data scientists who are interested in analyzing time series data. This survey of methods is far from complete; more methods exist than any single paper can cover.

The first main section describes several methods that you can use to decompose a time series signal into components. The first of its subsections covers decomposition of a time series into trend and seasonal components, using either classical decomposition or exponential smoothing models. The second subsection describes single spectrum analysis (SSA), which represents an alternative nonparametric way of decomposing a time series into components by using principal component analysis. The second main section covers the related topic of motif discovery, which is helpful for finding recurrent patterns in a time series. The third main section covers similarity analysis, which is helpful for comparing two sequences or for constructing a similarity matrix among a set of series. You can use a similarity matrix for classification purposes—for example, in a clustering process.

This paper demonstrates how to use these techniques with SAS/ETS and SAS Visual Forecasting software. If your data are in a CAS data table, the TSMODEL procedure, through its dynamic loading of packages of functions, provides a one-stop environment in SAS® Viya® for performing analyses that require several different procedures in SAS 9.4.

## TRANSACTIONAL AND TIME SERIES DATA

Transactional data are timestamped data that collected over time at no particular frequency. Some examples of transactional data are:

- internet data
- point-of-sales (POS) data
- inventory data
- call center data
- trading data

In order to be analyzed, transactional data need to be aggregated into *time series data*, which are timestamped data that are collected over time at a fixed frequency. Following are some examples of time series data:

- website visits per hour
- sales per month
- inventory draws per week
- calls per day
- trades per weekday

As you can see, the frequency that is associated with the time series varies with the problem at hand. The frequency (also called the *time interval*) can be hourly, daily, weekly, monthly, quarterly, yearly, or many other variants of the basic time intervals. The choice of frequency is an important modeling decision.

The aggregation of transactional data into a time series format is often called *time series accumulation* in order to distinguish it from other form of aggregations, such as an aggregation across a hierarchical structure.

Associated with each time series is a seasonal cycle, called s*easonality*. For example, the length of seasonality for a monthly time series is usually assumed to be 12 because there are 12 months in a year. Likewise, the seasonality of a daily time series is usually assumed to be 7. The typical seasonality assumption might not always hold. For example, if a particular business's seasonal cycle is 14 days long, the seasonality is 14 instead of 7.

For the remainder of this paper, $y_t$ denotes a real-valued time series that is observed at regular intervals $t = 1, \ldots, T$.

## TIME SERIES DECOMPOSITION

Time series decomposition is a crucial tool in the analysis of time series. A time series is decomposed into components that represent some patterns of the series. The components can then be combined to recreate the original series, either by adding them together if the decomposition is additive or by multiplying them together if the decomposition is multiplicative.

The components, or the parameters associated with them, represent features of a time series that you can use. For example, you might want to cluster time series that have common patterns.

The following subsections present some common ways of decomposing a time series.

## TREND-SEASON DECOMPOSITION

The decomposition of a series into trend and seasonal components is probably the most widespread practice in time series analysis, especially for business and economic data.

Typically, a time series is decomposed into the following components:

- $T_t$, the *trend* component, which represents the long-term progression of the series
- $C_t$, the *cycle* component, which represents repeated fluctuations around the trend component
- $S_t$ , the *seasonal* component, which represents variations over a fixed and known period
- $I_t$, the *irregular* component, or noise component, which represent random disturbances

Often, the trend and cycle components are combined into one single *trend-cycle* component, $TC_t$.

The seasonal components are typically normalized to sum to 1 for multiplicative decomposition, or to 0 for additive decomposition.

A *seasonally adjusted* (or *deseasonalized*) series is a series whose seasonal component has been removed. Likewise, a *detrended* series is a series whose trend (or *trend-cycle*) component has been removed.

Businesses such as retailers need to distinguish short-term seasonal effects from long-term trends to better plan their stocking decision with enough lead time. Governmental agencies, such as the Federal Reserve or the US Census Bureau, provide the *seasonally adjusted* or *detrended* version of series of economic variables that are used by policy makers to better understand the status of the economy.

Given the importance that trend-season decomposition has in time series analysis, it is not surprising that there are several ways to accomplish it. The following subsections cover two methods: classical decomposition and a model-based decomposition that uses the class of exponential smoothing models. Several other methods are available. For more details and alternative methods, see the chapters about the X11, X12, X13, and UCM procedures in *SAS/ETS 14.3 User's Guide*.

### Classical Decomposition

Classical time series decomposition is a nonparametric method that uses a series of moving averages to decompose the series into *trend-cycle* ($TC_t$), *seasonal* ($S_t$), and *irregular* ($I_t$) components; it is computed as follows:

$f(y_t) = TC_t + S_t + I_t$ for additive decomposition

$f(y_t) = TC_t S_t I_t$ for multiplicative decomposition

where $f(y_t)$ represents a possible functional transformation for the dependent series, such as log, square-root, logistic, or Box-Cox transformation.

The Hodrick-Prescott Filter (Hodrick and Prescott 1980) can further decompose the *trend-cycle* component into *trend* and *cycle* components in an additive fashion:

$TC_t = T_t + C_t$

You can find more details about classical decomposition in the chapter “The TIMESERIES Procedure” in *SAS/ETS 14.3 User's Guide*.

The following and later examples analyze the Air variable in the Sashelp.Air data set. This data set contains a time series that represents international airline passenger data, given as Series G in Box and Jenkins (1976). This series describes monthly totals of international passengers for the period between January 1949 and December 1960. It has been widely used in time series analysis literature as an example of a nonstationary seasonal time series. Figure 1 shows a plot of the series. You can clearly identify an increasing trend and some seasonal patterns: lower in winters and higher in summers.

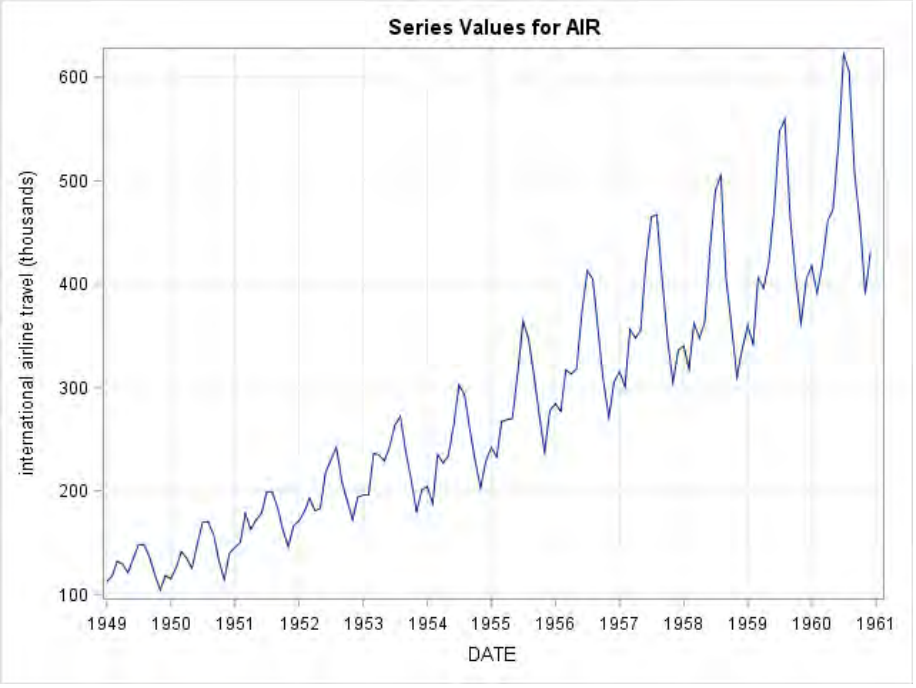



**Figure 1. Air Series**

The following TIMESERIES procedure statements use classical decomposition to decompose the series that is represented by the Air variable in the Sashelp.Air:

```
proc timeseries data=sashelp.air
       out=_NULL_
       outdecomp = decomp
       plot=(decomp);
   id date interval=month;
   var air;
run;
```

The Decomp data set contains the series component and the seasonally adjusted series. Figure 2 shows the panel plot of the series components superimposed over the original series.

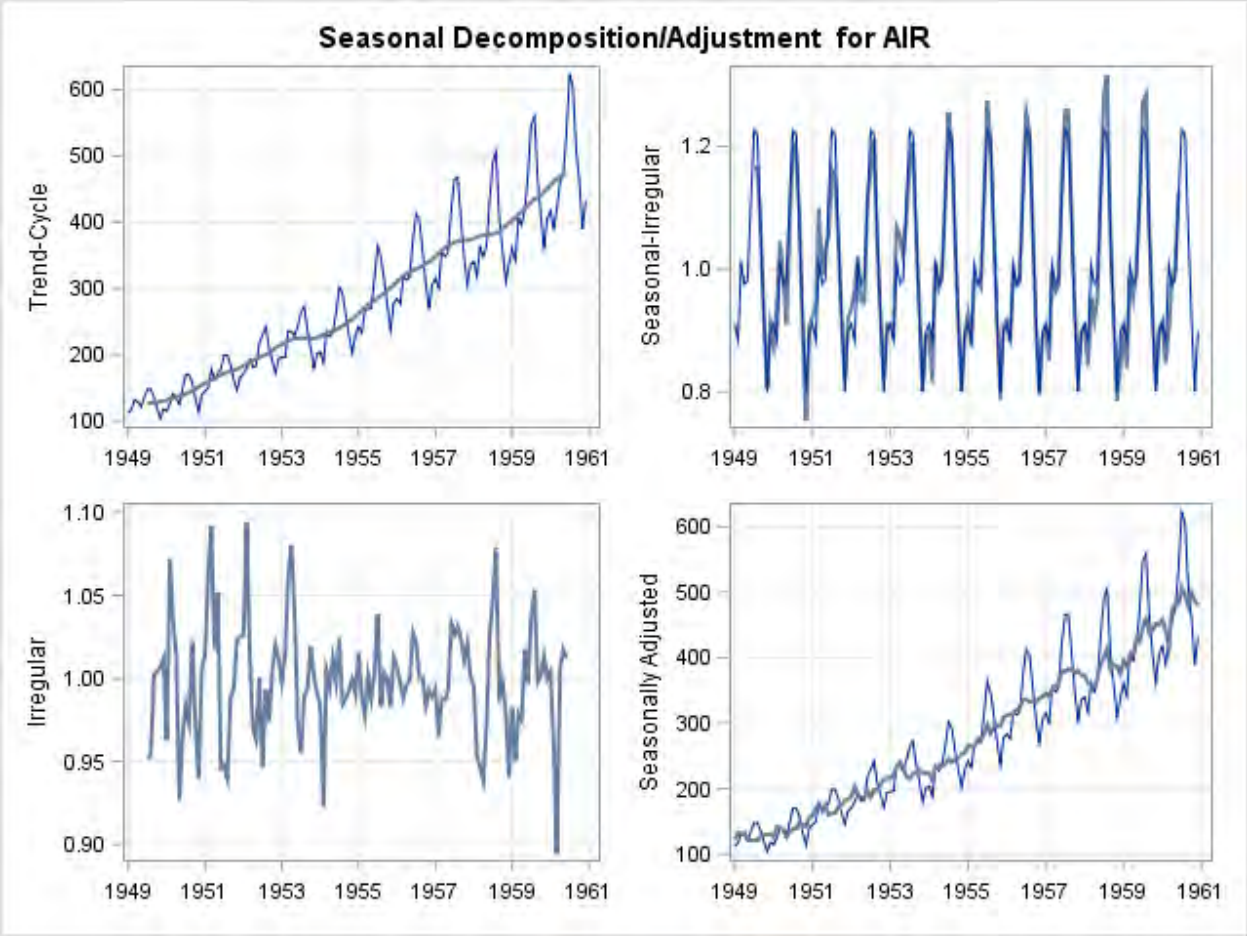



**Figure 2. Trend/Season Decomposition for the Variable Air**

If your data reside in a CAS table, you can use the TSMODEL procedure to perform classical decomposition. The following statements use the TSMODEL procedure to compute the seasonal indices on the time series array Air:

```
proc tsmodel data=mycas.air outarray=mycas.outarray;
   id date interval=month;
   var air;
   outarrays ADJUSTED;
   require tsa;
   submit;
      declare object TSA(tsa);
      rc=TSA.SEASONALDECOMP(air, _SEASONALITY_,
                            'ADD', , , , , , , , ADJUSTED, , , );
   endsubmit;
run;
```

The REQUIRE statement loads the time series analysis (TSA) package, which contains the TSA.SEASONALDECOMP function for classical decomposition.

For more information about the TSMODEL procedure and the TSA package, see the chapter “The TSMODEL Procedure” in *SAS Econometrics 8.2: Econometrics Procedures*.

## Exponential Smoothing Decomposition

Classical trend/season decomposition relies on moving averages to decompose the series. The decomposition can be refined using more complex and flexible classes of models. Although the class of exponential smoothing models is still relatively simple, it provides flexibility in computing the trend and seasonal components.

The following ESM procedure statements fit an additive Holt-Winters model to the log of the series:

```
proc esm data=sashelp.air out=_null_
         lead=0
         back=0
         plot=(trend season);
   id date interval=month;
   forecast air / model=addwinters transform=log;
run;
```

Figure 3 shows the smoothed trend for the Air variable for the additive Winters model.

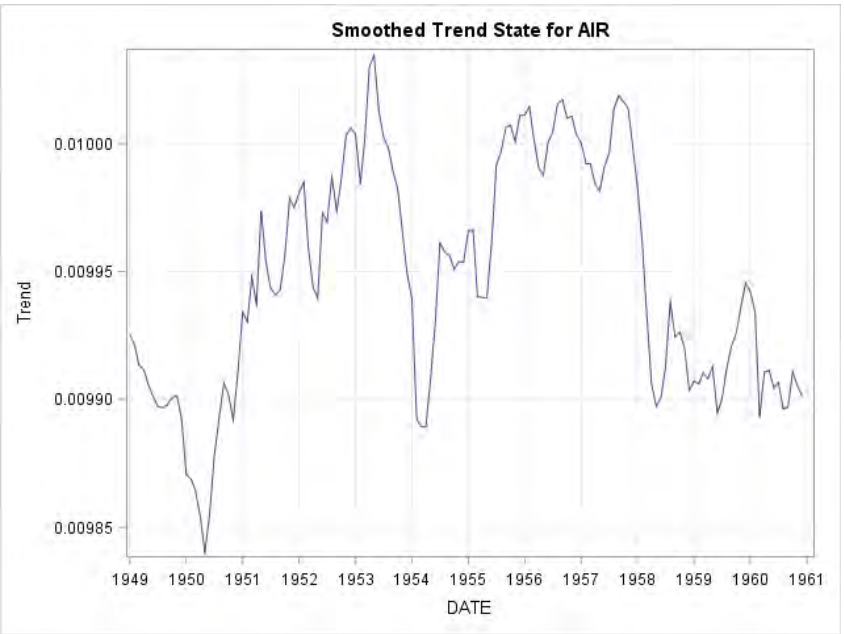



**Figure 3. Additive Winters Method Smoothed Trend**

Figure 4 shows the smoothed season for the Air variable for the additive Holt-Winters model.

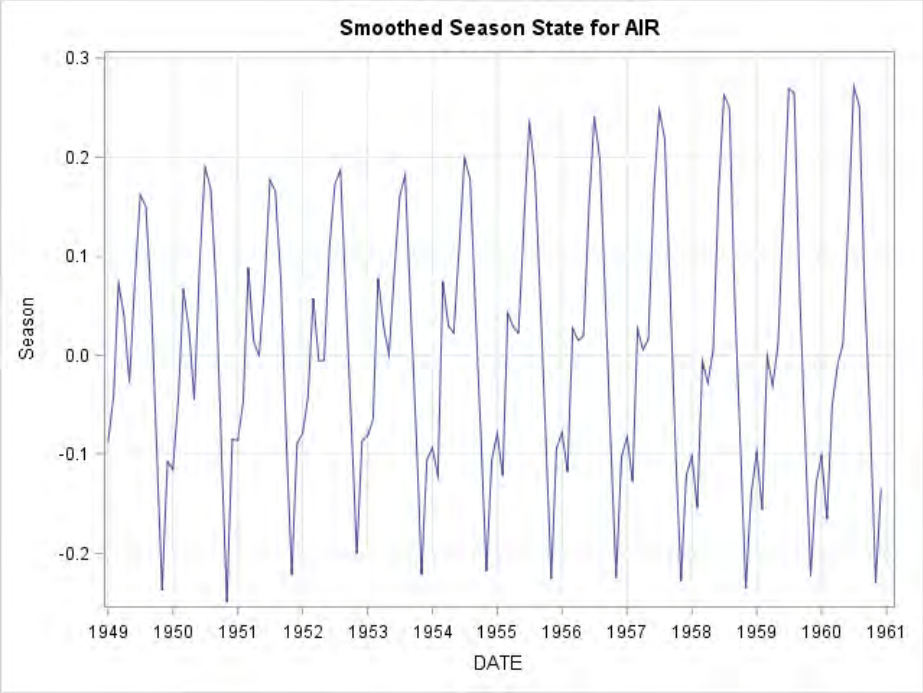



**Figure 4. Additive Winters Method Smoothed Season**

The advantage of a model-based decomposition is that you can use the model parameters as parsimonious summary features of the time series. For example, you can use values of the parameter of the additive Holt-Winters method to cluster series that have similar characteristics.

Table 1 shows the content of the OUTEST= data set with the parameters of the additive Holt-Winters method.

| Obs | _NAME_ | _TRANSFORM_ | _MODEL_ | _PARM_ | _EST_ | _STDERR_ | _TVALUE_ | _PVALUE_ |
|---|---|---|---|---|---|---|---|---|
| 1 | AIR | LOG | ADDWINTERS | LEVEL | 0.37509 | 0.035174 | 10.6637 | 0.00000 |
| 2 | AIR | LOG | ADDWINTERS | TREND | 0.00100 | 0.008803 | 0.1136 | 0.90972 |
| 3 | AIR | LOG | ADDWINTERS | SEASON | 0.73448 | 0.081167 | 9.0490 | 0.00000 |

**Table 1. Additive Winters Method Parameter Estimates**

If your data reside in a CAS table, you can use the TSMODEL procedure to perform a similar analysis:

```
proc tsmodel data=MYCAS.AIR
             outobj=(outFcast=MYCAS.AIRFOR  parEst=MYCAS.AIREST)
             seasonality=12;
   id DATE interval=MONTH;
   var AIR;
   require tsm;
   submit;
      declare object myModel(TSM);
      declare object mySpec(ESMSpec);
      rc=mySpec.open();
      rc=mySpec.SetOption('method', 'addwinters');
      rc=mySpec.SetTransform('log', 'mean');
      rc=mySpec.close();
```

```
        /* Setup and run the TSM model object */
        rc=myModel.Initialize(mySpec);
        rc=myModel.SetY(AIR);
        rc=myModel.SetOption('lead', 0);
        rc=myModel.SetOption('back', 0);
        rc=myModel.Run();

        /* Output model forecasts and estimates */
        declare object outFcast(TSMFor);
        rc=outFcast.Collect(myModel);
        declare object parEst(TSMPEst);
        rc=parEst.Collect(myModel);
    endsubmit;
run;
```

The REQUIRE statement loads the time series model (TSM) package, which contains the object definitions for the exponential smoothing model objects (ESMSpec). The portion of code between the SUBMIT and ENDSUBMIT statements is a SAS language script that is submitted and compiled on each worker node of the cluster on which your SAS Viya installation runs. The DECLARE statements create an instance of the TSM model object (myModel) and an instance of the ESM model object (mySpec). The ESMSpec instance is opened, and the SETOPTION method of the ESMpec object is used to select an additive Holt-Winters model before the instance is closed again.

The outFcast and parEst collector objects store the forecasts and the parameter estimates in the Mycas.Airfor and Mycas.Airest tables, respectively.

## SINGULAR SPECTRUM ANALYSIS DECOMPOSITION

An alternative to trend/season decomposition is singular spectrum analysis (SSA), which applies nonparametric techniques that adapt the commonly used principal component analysis (PCA) for decomposing time series data. The principal components can help you discover and understand the various patterns that the time series contains. After you understand each of these component series, you can model and forecast them separately; then you can aggregate the component series forecasts to forecast the original series under investigation. SSA is particularly valuable for long time series, in which patterns (such as trends and cycles) are difficult to visualize and analyze.

Introductory discussions of SSA can be found in Golyandina, Nekrutkin, and Zhigljavsky (2001), Elsner and Tsonis (1996), and Leonard, Elsheimer, and Kessler (2010).

Given a time series $y_t$ for $t = 1, \dots, T$ and a window length $2 \leq L < T/2$, SSA decomposes the time series into spectral groupings by using the following steps:

1. **Embedding:** Using the time series, form a $K \times L$ trajectory matrix $\boldsymbol{X} = \{x_{k,l}\}_{k=1,l=1}^{K,L}$ such that $x_{k,l} = y_{(k-l+1)}$ for $k = 1, \dots, K$ and $l = 1, \dots, L$, where $K = (T - L + 1)$. By definition, $L \leq K < T$ because $2 \leq L < T/2$.
2. **Decomposition:** Apply singular value decomposition to the trajectory matrix $\boldsymbol{X} = \boldsymbol{UQV}$, where $\boldsymbol{U}$ represents the $K \times L$ matrix that contains the left-hand-side (LHS) eigenvectors, $\boldsymbol{Q}$ represents the diagonal $L \times L$ matrix that contains the singular values, and $\boldsymbol{V}$ represents the $L \times L$ matrix that contains the right-hand-side (RHS) eigenvectors.

   Therefore, $\boldsymbol{X} = \sum_{l=1}^{L} \boldsymbol{X}^{(l)} = \sum_{l=1}^{L} u_l q_l v_l'$, where $\boldsymbol{X}^{(l)}$ represents the $K \times L$ principal component matrix, $u_l$ represents the $K \times 1$ left-hand-side (LHS) eigenvector, $q_l$ represents the singular value, and $v_l$ represents the $L \times 1$ right-hand-side (RHS) eigenvector that is associated with the *l*th window index.

3. **Grouping:** For each group index, $m = 1, \ldots, M$, define a group of window indices $I_m \subset \{1, \ldots, L\}$. Let $\boldsymbol{X}_{\boldsymbol{I_m}} = \sum_{l \in I_m} \boldsymbol{X}^{(l)} = \sum_{i \in I_m} u_l q_l v_l'$ represent the grouped trajectory matrix for group $I_m$.

   Note that if groupings represent a spectral partition, $\bigcup_{m=1}^{M} I_m = \{1, \ldots, L\}$, and $I_m \cap I_n = \emptyset$ for all $m \neq n$, then according to the singular value decomposition theory, $\boldsymbol{X} = \sum_{m=1}^{M} \boldsymbol{X}_{I_m}$.

4. **Averaging:** For each group index, $m = 1, \ldots, M$, compute the diagonal average of

   $\boldsymbol{X}_{I_m} = \left\{x_{k,l}^{(m)}\right\}_{k=1,l=1}^{K,L}$, $\tilde{x}_t^{(m)} = \frac{1}{n_t} \sum_{l=s_t}^{e_t} x_{(t-l+1),l}^{(m)}$

   where
   $s_t = 1, e_t = t, n_t = t$ for $(1 \leq t < L)$
   $s_t = 1, e_t = L, n_t = L$ for $(L \leq t \leq (T - L + 1)$
   $s_t = (T - t + 1), e_t = L, n_t = (T - t + 1)$ for $((T - L + 1) < t \leq T)$

   Note that if groupings represent a spectral partition, $\bigcup_{m=1}^{M} I_m = \{1, \ldots, L\}$, and $I_m \cap I_n = \emptyset$ for all $m \neq n$, then $y_t = \sum_{m=1}^{M} \tilde{x}_t^{(m)}$ by definition. Hence, singular spectrum analysis additively decomposes the original time series, $y_t$, into $m$ component series: $\tilde{x}_t^{(m)}$ for $m = 1, \ldots, M$.

5. **Forecasting (*optional*):** If the groupings represent a spectral partition, then each component series, $\tilde{x}_t^{(m)}$ for $m = 1, \ldots, M$, can be modeled and forecasted independently using an appropriate time series model (ARIMAX, unobserved component model, exponential smoothing model, and others), possibly using different time series models that include different input series (causal factors) and calendar events (interventions). The forecast for the original time series, $\hat{y}_t$, can be derived by simply aggregating the component series forecasts: $\hat{y}_t = \sum_{m=1}^{M} \hat{x}_t^{(m)}$, where $\hat{x}_t^{(m)}$ for $m = 1, \ldots, M$ represent the component series forecasts that are derived from the *m*th independent time series model.

   The SSA forecasting step represents a clever forecast model combination technique.

The following statements extract two additive components from the Sashelp.Air time series by using the THRESHOLDPCT= option to specify that the first component represents 80% of the variability in the series:

```
title "SSA of AIR data";
proc timeseries data=sashelp.air plot=ssa;
   id date interval=month;
   var air;
   ssa / length=12 THRESHOLDPCT=80;
run;
```

The resulting groupings, consisting of the first three and remaining nine singular value components, are presented in Figure 5 through Figure 7.

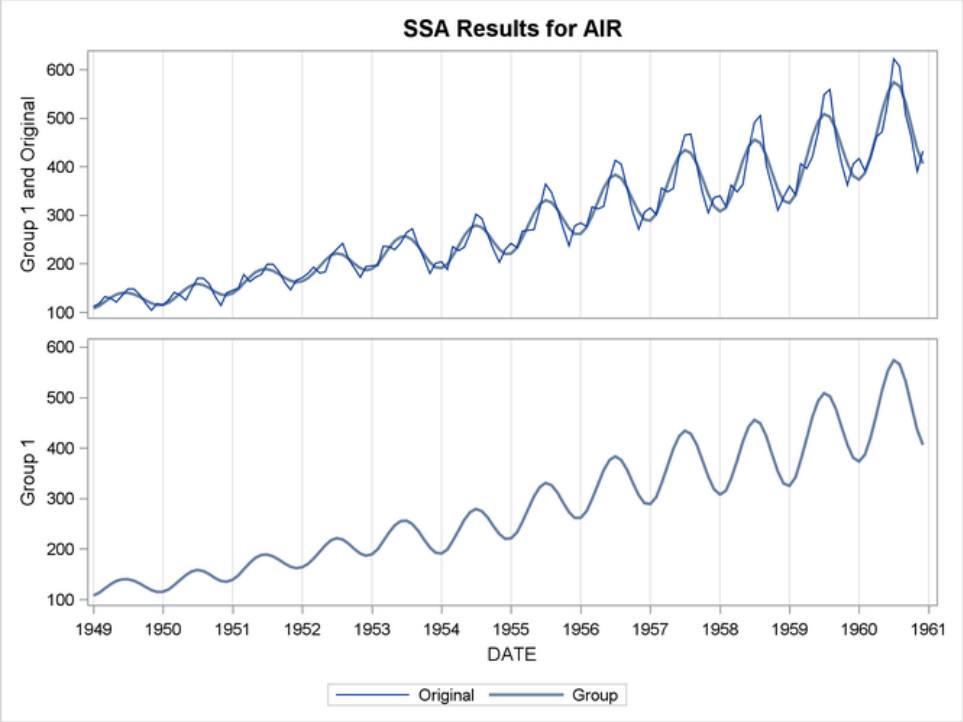



**Figure 5. Plot for Singular Value Grouping 1**

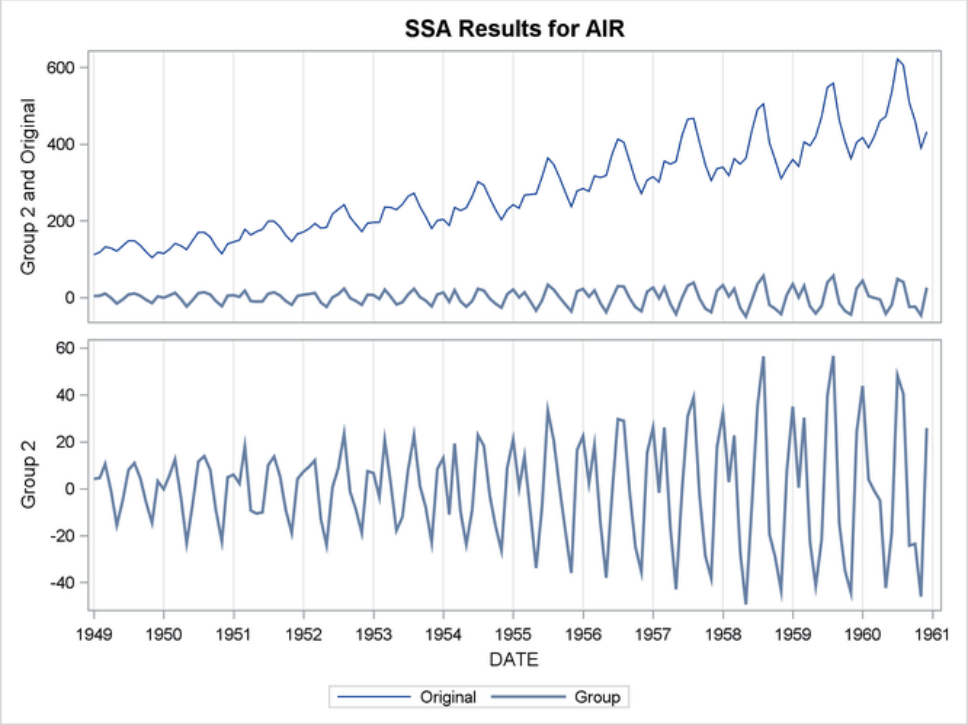



**Figure 6. Plot for Singular Value Grouping 2**

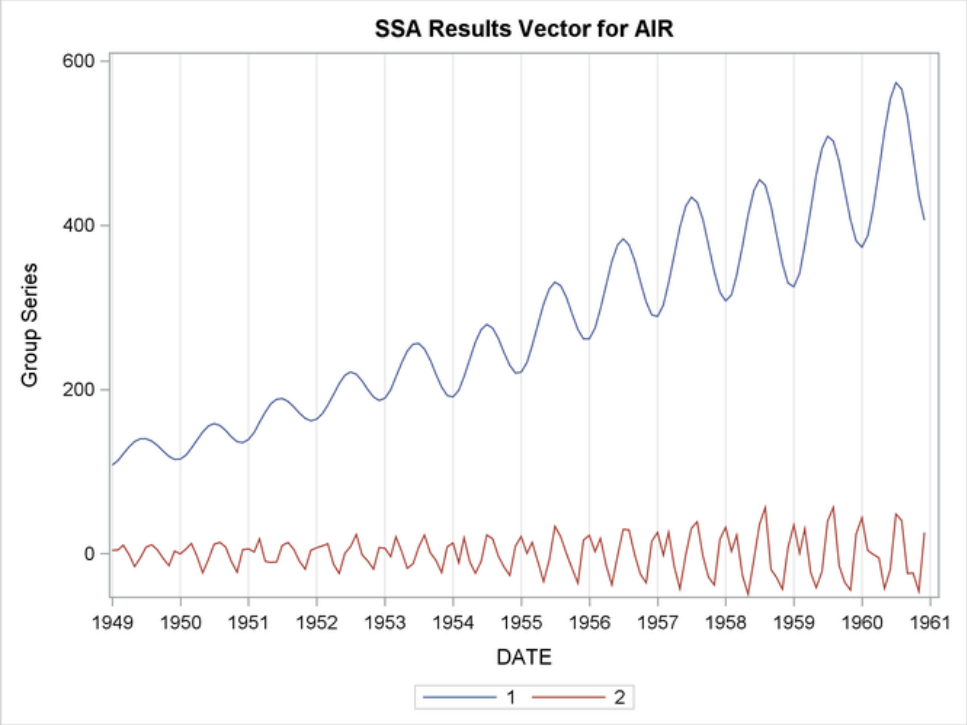



**Figure 7. Plot for Singular Value Components**

The following statements repeat the same analysis by using the TSMODEL procedure for data that are contained in a CAS data table:

```
proc tsmodel data=mycas.air
             outobj=(os=mycas.OUTSSA (replace=YES));
   id date interval=month;
   var air;
   require ssa;
   submit;
      declare object s(ssa);
      declare object os(outssa);
      rc = s.Initialize();
      rc = s.SetY(air);
      rc = s.SetOption('METHOD','THRESHOLD');
      rc = s.SetOption('LENGTH',12);
      rc = s.SetOption('THRESHOLDPCT',80);
      rc = s.Run();
      rc = os.Collect(s);
   endsubmit;
run;
```

The REQUIRE statement loads the singular spectrum analysis (SSA) package, which contains the definitions for the SSA objects. The SetOptions methods of the SSA objects are used to specify the options of the SSA analysis. Finally, the results are collected in a collector object of class Outssa and saved to the Outssa CAS table.

## MOTIF DISCOVERY

Motif discovery is a methodology that is related to the decomposition of a time series. Time series motifs are frequent patterns or repeated subsequences in temporal data; they are primitive shapes and implicit rules of time series data. Discovering motifs helps you understand, interpret, and identify important characteristics of your times series. However, the goal of motif discovery is not to decompose the series into components as it is in time series decomposition. Instead, the goal is to identify the motifs and their occurrence in the time sequence. Because motifs are extracted time series features, they can be used for time series association, classification, and clustering, and also for anomaly detection. Motifs are especially useful for various Internet of Things (IoT) data analyses, including sequence matching from biomedical devices and recognition of activities or gestures from body-worn sensors. The time series motif (MTF) package, used with PROC TSMODEL, provides motif discovery functional objects that perform the following:

- motif discovery by using a brute-force method
- motif discovery by using a probabilistic method based on a temporal topic model
- motif scoring that finds motif instance occurrences of a specified target motif in a new sequence
- motif-based subsequence anomaly detection

This paper demonstrates only the brute-force method of motif discovery. For more information about the other methods, see the *SAS Visual Forecasting: Time Series Packages*.

The following SAS code simulates a time series that has a sine curve motif and uses background data that are sampled from the standard normal distribution. The planted motif instances occur at times 50, 150, and 250. The length of the motif is 10. The length of the time series is 300.

```
%let motif_length = 10;
%let sequence_length = 300;
%let motif_position = (50,150,250);
%let n_motifs = 3;

data SimuData;
        array start {&n_motifs} &motif_position;
        array end {&n_motifs} &motif_position;
        call streaminit(123);
        do j = 1 to dim(start);
                end[j] = start[j] + &motif_length;
        end;
        do i = 1 to &sequence_length;
             time = i;
             signal = rand('NORMAL');
             do j = 1 to dim(start);
                  if i >= start[j] and i<end[j] then do;
                signal = signal+10*sin((i-start[j])/
                         &motif_length * (2*constant('pi')));
                  end;
             end;
        output;
        end;
keep time signal;
run;
```

Figure 8 shows the plot of simulated data, which appear to contain three large sine curves around times 50, 150, and 250.

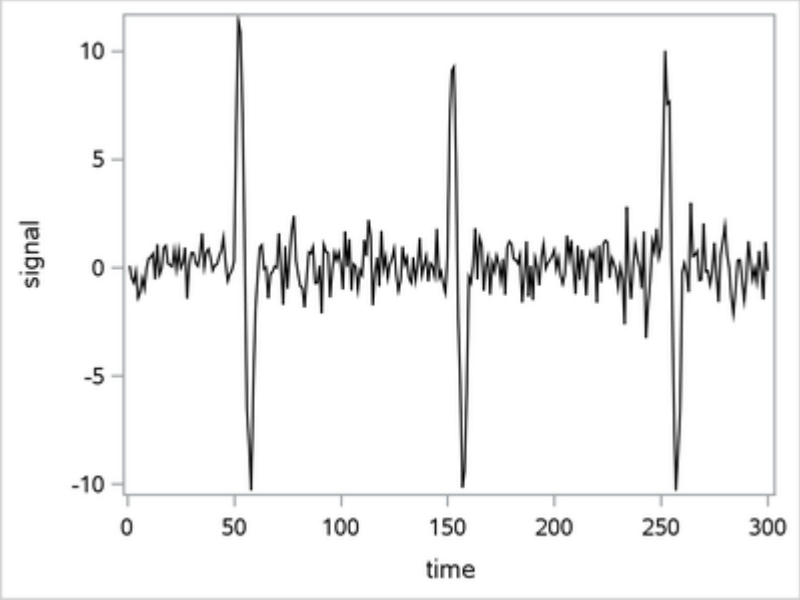



**Figure 8. Plot of Simulated Motif Data**

The following PROC TSMODEL statements use the brute force method to detect the motif:

```
proc tsmodel data=mycas.SimuData
             outobj=(of=mycas.outmotif
             ofms = mycas.outmotifseries);
   var signal;
   id  time interval=day;
   require mtf;
   submit;
           declare object f(MTFBF);
           declare object of(OUTMTF);
           declare object ofms(OUTMTFSERIES);
           rc = f.Initialize();
           rc = f.SetX(signal);
           rc = f.SetOption("NMOTIF", 1,
                            "MOTIFLENGTH",10,
                            "NORMALIZE", "Y",
                            "DISTMARGIN", 1
                           );
           rc = f.Run();
           rc = of.Collect(f);
           rc = ofms.Collect(f);
   endsubmit;
run;
```

The REQUIRE statement loads the time series motif (MTF) package. The MTFBF object executes a brute-force method for motif discovery. The object declaration statement creates a new object, F, of type MTFBF.

The brute-force method finds the exact three time points where the motif instances start, as shown in Table 2 and Figure 9.

| Obs | Variable Name | Motif ID | Start Position | Distance |
|---|---|---|---|---|
| 1 | signal | 1 | 50 | 0.2740376908 |
| 2 | signal | 1 | 150 | 0.3228669954 |
| 3 | signal | 1 | 250 | 0.356297145 |

**Table 2. List of Motifs Discovered by the Brute-Force Method**

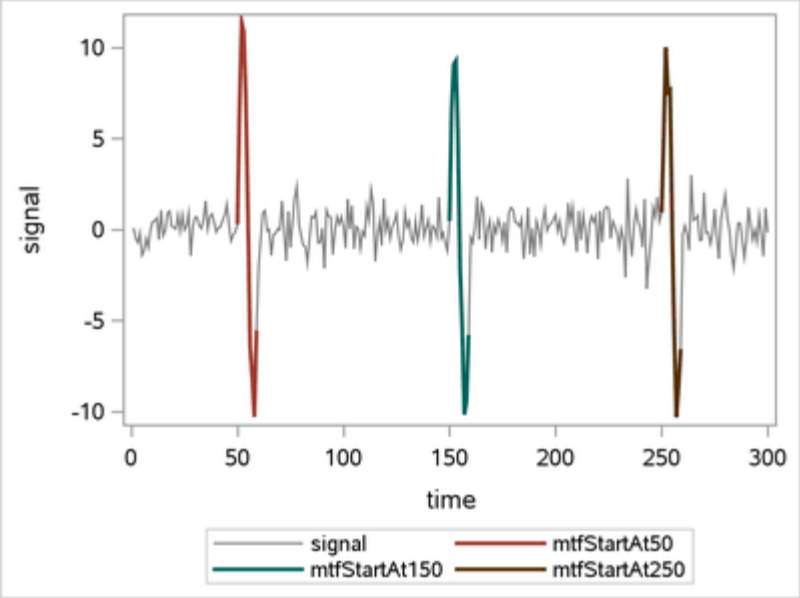



**Figure 9. Plot of Motifs Discovered by the Brute-Force Method**

## SIMILARITY ANALYSIS

Comparing two items or features is often needed in machine learning—for example, in classifying elements in a set. A similarity measure is a metric that measures the distance between an input sequence and a target sequence and that takes ordering into account. The two sequences can be time series or timestamped data that are observed at different time points. In addition, similarity measures can "slide" the target sequence with respect to the input sequence. The "slides" can occur by observation index (in sliding-sequence similarity measures) or by seasonal index (in seasonal-sliding-sequence similarity measures). For computing metrics, you can use fixed and dynamic time-warped metrics for squared and absolute deviations, in addition to other metrics.

Similarity measures can be used to compare a single input sequence to several other representative target sequences. This situation arises in time series classification. For example, given a single input sequence, you can classify the input sequence by finding the "most similar" or "closest" target sequence.

Similarity analysis can be repeated to classify large numbers of input sequences. Similarity measures can also be computed between several sequences to form a similarity matrix. For example, given $K$ time sequences, you can construct a $K \times K$ symmetric matrix in which each element represents the similarity measure between two sequences. You can also use the similarity matrix as a distance matrix in clustering time series.

Sliding similarity measures (such as observational or seasonal indices) can be used to compare a single target sequence to subsequences of many other input sequences on a sliding basis. This situation arises in historical time series analogies. For example, given a single target series, you can find similar times in the history of the input sequence while preserving the ordering or seasonal indices.

Similarity analysis uses *dynamic time warping* techniques to map an input sequence to a target sequence. Several distance measures can be computed by considering the paths that are formed by such a mapping. A full description of the details is beyond the scope of this paper and can be found in Leonard et al. (2008).

This simple example illustrates how to use similarity analysis to compare two time sequences. The following statements create an example data set, Test, which contains two time sequences of differing lengths:

```
data test;
   input i y x;
   datalines;
   1   2  3
   2   4  5
   3   6  3
   4   7  3
   5   3  3
   6   8  6
   7   9  3
   8   3  8
   9  10  .
   10 11  .
   ;
run;
```

The following statements perform similarity analysis on the Test data set:

```
proc similarity data=test out=_null_
   print=all plot=all;
   input x;
   target y / measure=absdev;
run;
```

The DATA=TEST option specifies the input data set Test to be used in the analysis. The OUT=_NULL_ option suppresses the creation of an output time series data set. The PRINT=ALL and PLOTS=ALL options request that all ODS tables and graphs be produced. The INPUT statement specifies that the input variable is X. The TARGET statement specifies that the target variable is Y and requests that the similarity measure be computed using absolute deviation (MEASURE=ABSDEV).

Figure 10 show the plot of the input and target series.

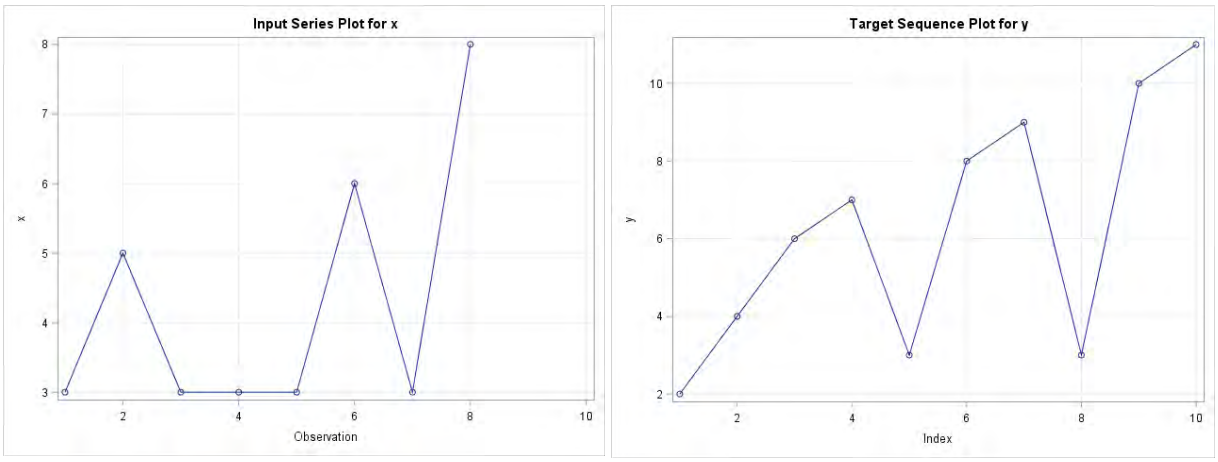



**Figure 10. Input and Target Series**

Notice that the two time sequence plots are somewhat similar but differ in length and timing. Both sequences have two phases of upward movements that are followed by downward movements and a final phase of upward movement.

Figure 11 shows the path plot and warped scaled plot for the input and target series.

In the path plot, the horizontal axis represents the input sequence index, and the vertical axis represents the target sequence index. The dots represent the path coordinates. This plot visualizes the path through the distance matrix. Vertical movements indicate compression, and horizontal movements represent expansion of the target sequence with respect to the input sequence. This plot is useful for visualizing the amount of expansion and compression along the path.

In the warp-scaled plot, the horizontal axis represents the input and target sequence index. The upper line (blue) represents the target sequence. The lower line (red) represents the input sequence. The lines that connect the input and target sequence values represent the mapping between the input and target sequence indices along the path. This plot visualizes the warping of the time index with respect to the input and target sequence values. Expansion of a single target sequence value occurs when it is mapped to more than one input sequence value. Expansion of a single input sequence value occurs when it is mapped to more than one target sequence value. This plot is useful for visualizing the mapping between the input and target sequence values along the path.

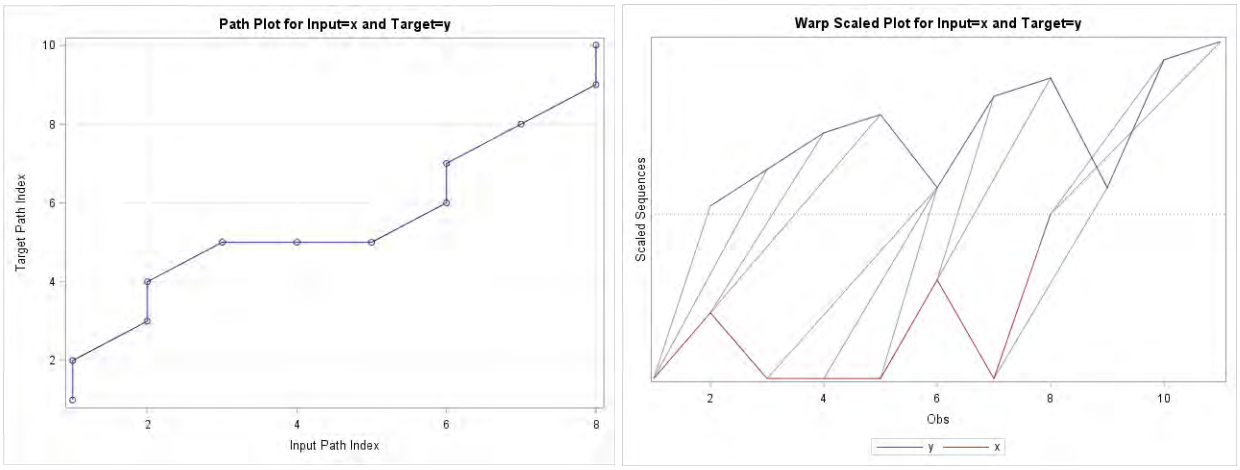



**Figure 11. Path Plot and Warp-Scaled Plot**

The similarity distance is computed by finding the optimal path that minimizes a cost function over all possible paths in the mapping. Different cost functions give rise to different distances. The preceding example uses the absolute deviation.

If your data reside in a CAS data table, you can perform a similar analysis by using the TSMODEL procedure and the SIMILARITY function in the TSA package. The following statements use the TSMODEL procedure to repeat the preceding analysis for data in a CAS data table:

```
proc tsmodel data=mycas.test
      outscalar=mycas.sim_scalar;
   require tsa;
   id i interval=day;
   var x y;
   outscalars measure;
   submit;
   declare object TSA(tsa);
   rc = TSA.SIMILARITY(x, y, 'absdev', 'NONE', , , , ,measure);
   endsubmit;
run;
```

The similarity measure is contained in the Mycas.Sim scalar table.

## CONCLUSION

This paper reviews some analysis methods for time series data that can be used for feature extraction and dimension reduction in the context of data mining and machine learning. The extracted features can give new insights into the time series and their dynamics, and they can be used to describe or classify time series or to build models for such purposes using other machine learning techniques.

The signal that is contained in a time series can be decomposed into components by several methods. This paper covers decomposition of a time series into trend and seasonal components, using either classical decomposition or exponential smoothing models. Singular spectrum analysis (SSA) represents an alternative nonparametric way of decomposing a time series into components by using principal component analysis. You can use motif discovery to find recurrent patterns in a time series. Finally, you can use similarity analysis to compare two sequences or to construct a similarity matrix among a set of series. You can use the similarity matrix for classification purposes (for example, in a clustering process).

You can implement these techniques either by using SAS/ETS procedures, or, if your data are in a CAS data table, the TSMODEL procedure in SAS Visual Forecasting. The TSMODEL procedure, through its dynamic loading of packages of functions, provides a one-stop environment in SAS Viya for performing analyses that would otherwise require several different procedures in SAS 9.4.

## REFERENCES


Box, G. E. P., and Jenkins, G. M. 1976. *Time Series Analysis: Forecasting and Control*. San Francisco: Holden-Day.

Elsner, J. B., and Tsonis, A. A. 1996. *Singular Spectral Analysis: A New Tool in Time Series Analysis*. New York: Plenum Press.

Golyandina, N., Nekrutkin, V., and Zhigljavsky, A. 2001. *Analysis of Time Series Structure: SSA and Related Techniques*. Boca Raton, FL: Chapman and Hall/CRC.

Hodrick, R. J., and Prescott, E. C. 1980. “Postwar U.S. Business Cycles: An Empirical Investigation.” Discussion Paper 451, Carnegie Mellon University.

Leonard, M., Beeman, J., Lee, T., and Elsheimer, B. 2008. “An Introduction to Similarity Analysis Using SAS.” *Proceedings of the SAS Global Forum 2008 Conference*. Cary, NC: SAS Institute Inc. Available http://www2.sas.com/proceedings/forum2008/319-2008.pdf

Leonard, M., John, M., and Elsheimer, B. 2010. “Introduction to Singular Spectrum Analysis with SAS/ETS Software.” *Proceedings of the SAS Global Forum 2010 Conference*. Cary NC: SAS Institute Inc. Available http://support.sas.com/resources/papers/proceedings10/308-2010.pdf

Leonard, M., and Elsheimer, B. “Automatic Singular Spectrum Analysis and Forecasting.” *Proceedings of the SAS Global Forum 2017 Conference*. Cary NC: SAS Institute Inc. Available http://support.sas.com/resources/papers/proceedings17/SAS0586-2017.pdf

Quirino, T., and Leonard, M. “Scalable Cloud-Based Time Series Analysis and Forecasting” *Proceedings of the SAS Global Forum 2018 Conference*. Cary NC: SAS Institute Inc.

## ACKNOWLEDGMENTS

The authors would like to thank Anne Baxter for editing this paper.

## CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Michele Trovero
SAS Institute Inc.
Michele.Trovero@sas.com

Michael Leonard
SAS Institute Inc.
Michael.Leonard@sas.com

Paper SAS3258-2019

# Writing a Gradient Boosting Model Node for SAS® Visual Forecasting

Yue Li, Jingrui Xie, and Iman Vasheghani Farahani, SAS Institute Inc.

## ABSTRACT

SAS Visual Forecasting, the new-generation forecasting product from SAS, includes a web-based user interface for creating and running projects that generate forecasts from historical data. It is designed to use the highly parallel and distributed architecture of SAS® Viya®, a cloud-enabled, in-memory analytics engine that is powered by SAS® Cloud Analytic Services (CAS), to effectively model and forecast time series on a large scale. SAS Visual Forecasting includes several built-in modeling strategies, which serve as ready-to-use models for generating forecasts. It also supports custom modeling nodes, where you can write and import your own code-based modeling strategies. Similar to the ready-to-use models, these custom modeling nodes can also be shared with other projects and forecasters. Forecasters can use SAS Visual Forecasting to create projects by using visual flow diagrams (called pipelines), running multiple built-in or custom models on the same data, and choosing a champion model based on the results. This paper uses a gradient boosting model as an example to demonstrate how you can use a custom modeling node in SAS Visual Forecasting to develop and implement your own modeling strategy.

## INTRODUCTION

SAS Visual Forecasting includes a web-based user interface for creating and running projects to generate forecasts. It provides automation and analytical sophistication to generate millions of forecasts in the fast turnaround time that is necessary to run your business. Forecasters can create projects by using visual flow diagrams (also called pipelines), and can run multiple models on the same data set and choose a champion model on the basis of the forecast results.

Because SAS Visual Forecasting runs in SAS Viya, which has a highly parallel and distributed architecture, is designed to effectively model and forecast time series on a large scale. SAS Cloud Analytic Services (CAS) provides the speed and scalability needed to create the models and generate forecasts for millions of time series. Massive parallel processing within a distributed architecture is one of the key advantages in SAS Visual Forecasting for large-scale time series forecasting.

SAS Visual Forecasting includes a number of modeling strategies that you can use to generate forecasts. These strategies include two hierarchical forecasting models, a panel neural network model, an auto-forecasting model, a naïve model, a multistage forecasting model, a stacked model, and more. Furthermore, you can create your own modeling strategies that you can share with other projects and forecasters. These custom modeling strategies are called pluggable modeling strategies.

The following sections use a gradient boosting model (GBM) as an example to illustrate the steps for writing a pluggable modeling strategy. Gradient boosting is a popular machine learning technique for regression and classification problems. It produces a prediction model in the form of an ensemble of weak prediction models. The example in this paper uses the GRADBOOST procedure in SAS® Visual Data Mining and Machine Learning to implement a GBM. PROC GRADBOOST enables you to build a GBM that consists of multiple decision trees.

## FILES REQUIRED TO DEFINE A PLUGGABLE MODELING STRATEGY

The following files are required to define a pluggable modeling strategy in SAS Visual Forecasting 8.3:

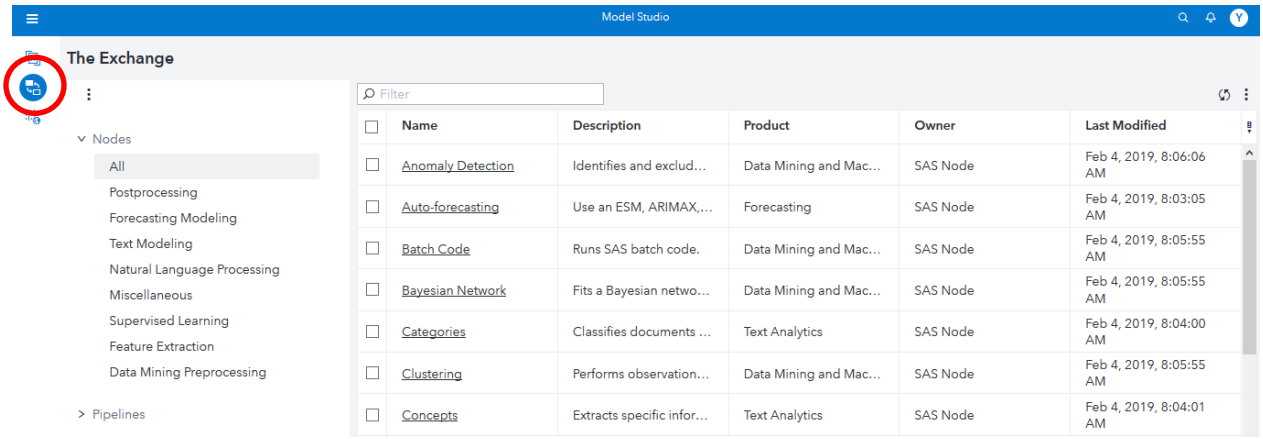
- **code.sas**, which contains the SAS® code to be executed
- **validation.xml**, which defines the validation rules for the strategy specification settings (properties)
- **template.json**, which defines the metadata of the strategy

Starting from SAS Visual Forecasting 8.4, another file is required, **metadata.json**. This file guarantees that the version of the modeling strategies matches the current version of Model Studio. You can download an existing strategy from the exchange and copy the metadata.json file to your modeling strategy to make sure you have the correct version.

These files are packed in a .zip file for uploading to or downloading from **The Exchange** in SAS Visual Forecasting. You can download an existing pluggable modeling strategy and learn from the downloaded example files. Then you can modify the contents accordingly and use the modified files.

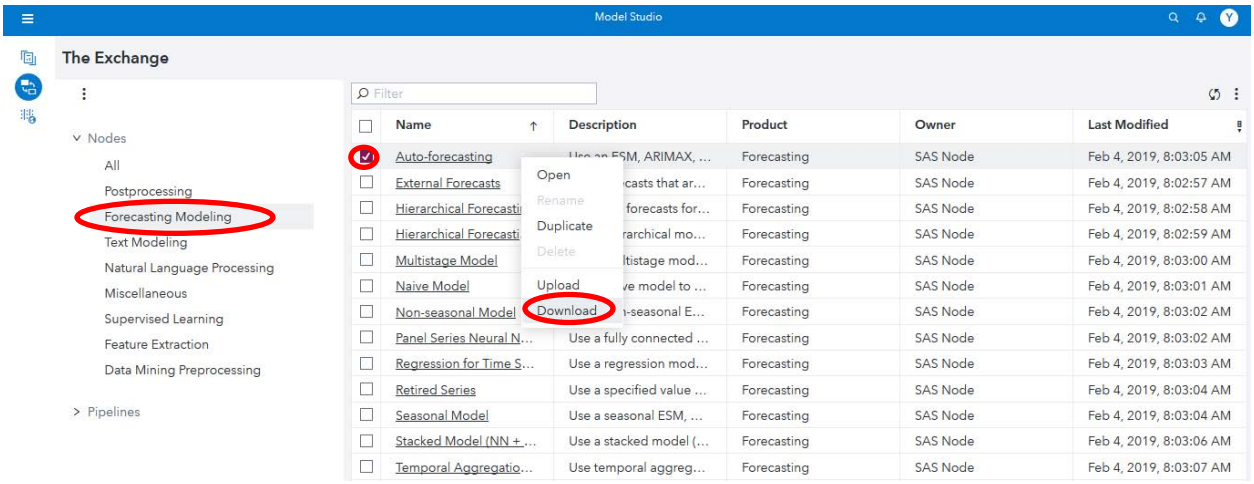
To download an existing pluggable modeling strategy:

1. On the **Build Model** tab of the main page of Model Studio, click the button to open **The Exchange**, as shown in Display 1.




**Display 1. The Exchange in Model Studio**

2. Select **Forecasting Modeling** from the **Nodes** list, right-click an existing modeling strategy (such as **Auto-forecasting** in Display 2) to be downloaded, and then click **Download**.




**Display 2. Download Modeling Strategy**

3. Unzip the downloaded .zip file to view the code.sas, validation.xml and template.json files. The following sections describe these files in more detail.

## CODE.SAS FILE

The code.sas file contains the run-time SAS code that can be executed in a pipeline to generate forecasts. You can use a set of system-defined macro variables and macros in the run-time code to obtain references to the input data, output forecasts, variable roles, settings, and so on from the pipeline.

Table 1, Table 2, and Table 3 in the appendices show the system-defined macro variables and macros that contain project information such as CAS session, caslibs, table names, and variable roles and settings.

When you write the run-time code, you can refer to these macro variables and macros to retrieve information and generate TSMODEL procedure statements.

Figure 1 shows the contents of the code.sas file, which contains the run-time code.

The code first defines a `%fx_prepare_input` macro that uses the built-in data preparation macro code to prepare input data. The input table is prepared by the Data node in the pipeline. For more information about the input data, see the chapter “Setting Up Your Project” in *SAS Visual Forecasting: User’s Guide*. If you want to refer to the input table in code, you can use the macro variable definition, as follows.

```
&vf_libIn.."&vf_inData"n
```

The main code, which is wrapped in a `%gbm_run` macro, first calls the data preparation macro to prepare the input and transform the dependent variable (if needed), and then calls PROC GRADBOOST to build a gradient boosting model. When lag variables for the dependent variable are used in the model, a DATA step is used to recursively score both the historical and future periods according to score code output from the GBM. Another DATA step is then used to prepare the required output. At the end of the file, the main code macro is invoked.

```
/*
    Macro for adding features into input data
        I: &vf_libIn.."&vf_inData"n
        O: &vf_libOut..&vf_tableOutPrefix..&outTblName
*/
%macro fx_prepare_input(outTblName=, byVars=,
                        trendVariable=, seasonalDummy=,
                        seasonalDummyInterval=,
                        esmY=, lagXNumber=, lagYNumber=,
                        holdoutSampleSize=, holdoutSamplePercent=,
                        criteria=, back=);

    %local filerootpath;
    %let filerootpath = &sas_root_location/misc/codegenscrpt/source/sas;
    %include "&filerootpath./vf_data_prep.sas";
    %let temp_work_location = %sysfunc(pathname(work));
    %include "&temp_work_location./vfDataPrepMacro.sas";

%mend;

/*
    Main macro for forecasting using a gradient boosting model
*/
%macro gbm_run;

    /*Protection against problematic _seasonDummy value*/
```

```
    %if (not %symexist(_seasonDummy)) %then
        %let _seasonDummy=&vf_timeIDInterval;
    %if "&_seasonDummy" eq "" %then
        %let _seasonDummy = &vf_timeIDInterval;
    %if %sysfunc(INTTEST( &_seasonDummy )) eq 0 %then %do;
        %put Invalid seasonal dummy interval.
             Use &vf_timeIDInterval instead.;
        %let _seasonDummy = &vf_timeIDInterval;
    %end;

    /*Parepare input data with extracted feature used for modeling*/
    %fx_prepare_input(outTblName=fxInData, byVars=&vf_byVars,
                      trendVariable = &_trend,
                      seasonalDummy = &_seasonDummy,
                      seasonalDummyInterval = &_seasonDummyInterval,
                      esmY =FALSE, lagXNumber=&_lagXNumber,
                      lagYNumber=&_lagYNumber,
                      holdoutSampleSize=&_holdoutSampleSize,
                      holdoutSamplePercent=&_holdoutSamplePercent,
                      criteria=RMSE, back=0);

    /*Dependent variable transformation if needed*/
    %let targetVar=gbmTargetVar;
    %let predictVar=P_&targetVar;
    data &vf_libOut.."&vf_tableOutPrefix..fxInData"n /
        SESSREF=&vf_session;
        set &vf_libOut.."&vf_tableOutPrefix..fxInData"n;
        &targetVar = &vf_depVar;
        %if %upcase(&_depTransform) eq LOG %then %do;
          if not missing(&vf_depVar) and &vf_depVar > 0 then
             &targetVar = log(&vf_depVar);
          else call missing(&targetVar);
        %end;
    run;

    /*Train the gradient boosting model*/
    proc gradboost data=&vf_libOut.."&vf_tableOutPrefix..fxInData"n
                   seed=12345;
        id &vf_byVars &vf_timeID;
        partition rolevar=_roleVar(TRAIN="1" VALIDATE="2" TEST="3");
        input &vf_byVars  /level=NOMINAL;
        %if "&vf_indepVars" ne "" %then %do;
          input &vf_indepVars  /level=INTERVAL;
        %end;
        %if %intervalFeatureVarList ne  %then %do;
          input %intervalFeatureVarList /level=INTERVAL;
        %end;
        %if %nominalFeatureVarList ne  %then %do;
          input %nominalFeatureVarList /level=NOMINAL;
        %end;
        target &targetVar / level=interval;
        autotune maxtime=3600
                 tuningparameters=( ntrees(lb=50 ub=500 init=50)) ;
        %if %eval(&_lagYNumber>0) %then %do;
           code file="&temp_work_location./_gbScore.sas";
        %end;
        %else %do;
```

```
            output out=&vf_libOut.."scored_gb"n
                   copyvars=(&vf_byVars &vf_timeID &vf_depVar);
        %end;
    run;

    %if %eval(&_lagYNumber>0) %then %do;
        /*When lagYNumber is nonzero,
          score the model for the whole data set with
          recurrent dependent variable value for the future periods*/
        %let count=%sysfunc(countw(&vf_byVars,%str( )));
        %let lastByVar=%scan(&vf_byVars, &count, %str( ));
        data &vf_libOut.."scored_gb"n /SINGLE=YES;
            set &vf_libOut.."&vf_tableOutPrefix..fxInData"n;
            by &vf_byVars  &vf_timeID;
            retain copyY . %do i=1 %to &_lagYNumber;
                              copyY_lag&i. . %end;;
            gbmLastByVar = &lastByVar;
            if first.gbmLastByVar  then do;
                copyY=.;
                %do i=1 %to &_lagYNumber; copyY_lag&i.=.; %end;
            end;
            %do i=&_lagYNumber  %to 1 %by -1;
                %if &i eq 1 %then %do;
                    if not missing(copyY) then copyY_lag1=copyY;
                %end;
                %else %do;
                    if not missing(copyY_lag%eval(&i -1)) then
                      copyY_lag&i = copyY_lag%eval(&i -1);
                %end;
            %end;
            %do i=1 %to &_lagYNumber; _lagY&i = copyY_lag&i.; %end;
            %include "&temp_work_location./_gbScore.sas";
            if &vf_timeID >= &vf_horizonStart then copyY = &predictVar;
            else copyY = &targetVar;
            drop copyY %do i=1 %to &_lagYNumber; copyY_lag&i.  %end;;
        run;
    %end;

    /*Prepare the required output tables*/
    data &vf_libOut.."&vf_outFor"n;
       set &vf_libOut.."scored_gb"n;
       actual = &targetVar;
       predict = &predictVar;
       %if %upcase(&_depTransform) eq LOG %then %do;
          if not missing(actual) then actual = exp(actual);
          if not missing(predict) then predict = exp(predict);
       %end;
       %if "&vf_allowNegativeForecasts" eq "FALSE" %then %do;
          if not missing(predict) and predict < 0 then predict = 0;
       %end;
       %if &targetVar ne &vf_depVar or &predictVar ne predict %then %do;
          drop &targetVar  &predictVar;
       %end;
    run;

%mend;
```

```
/*Invoke the main macro */
%gbm_run;
```

**Figure 1. Contents of code.sas File for the Gradient Boosting Model Example**

The only required output table is the OUTFOR table. The other tables are all optional.

The required OUTFOR table contains forecasts from the forecast models; it can be referred to as follows:

```
&vf_libOut..&vf_outFor
```

The OUTFOR table must have the following columns:

- byVars
- timeID
- actual (actual value of the dependent variable)
- predict (predicted value of the dependent variable)

The OUTFOR table can also have the following columns:

- std (standard deviation of errors)
- lower (lower confidence limit of predicted values)
- upper (upper confidence limit of predicted values)

SAS Visual Forecasting automatically validates the OUTFOR table and promotes it so that it can be used as a global CAS table. If the required columns do not exist, the modeling node reports errors. SAS Visual Forecasting also checks the OUTFOR table for invalid values (such as extreme values and negative values) and reports warning messages if it detects any.

The optional OUTSTAT table contains forecast accuracy measures such as MAPE and RMSE; it can be referred to as follows:

```
&vf_libOut..&vf_outStat
```

The OUTSTAT table must have the following columns:

- byVars
- summary statistics, including DFE N NOBS NMISSA NMISSP NPARMS TSS SST SSE MSE RMSE UMSE URMSE MAPE MAE RSQUARE ADJRSQ AADJRSQ RWRSQ AIC AICC SBC

APC MAXERR MINERR MAXPE MINPE ME MPE MDAPE GMAPE MINPPE MAXPPE MPPE MAPPE MDAPPE GMAPPE MINSPE MAXSPE MSPE SMAPE MDASPE GMASPE MINRE MAXRE MRE MRAE MDRAE GMRAE MASE MINAPES MAXAPES MAPES MDAPES GMAPES

If the pluggable modeling strategy run-time code generates this table, the pipeline will validate the table and promote it so that it can be used as a global CAS table. If any required columns or measurements are missing from the table or the run-time code does not output this table, the pipeline will automatically compute the statistics and generate this table on the basis of actual and predicted series from the OUTFOR table.

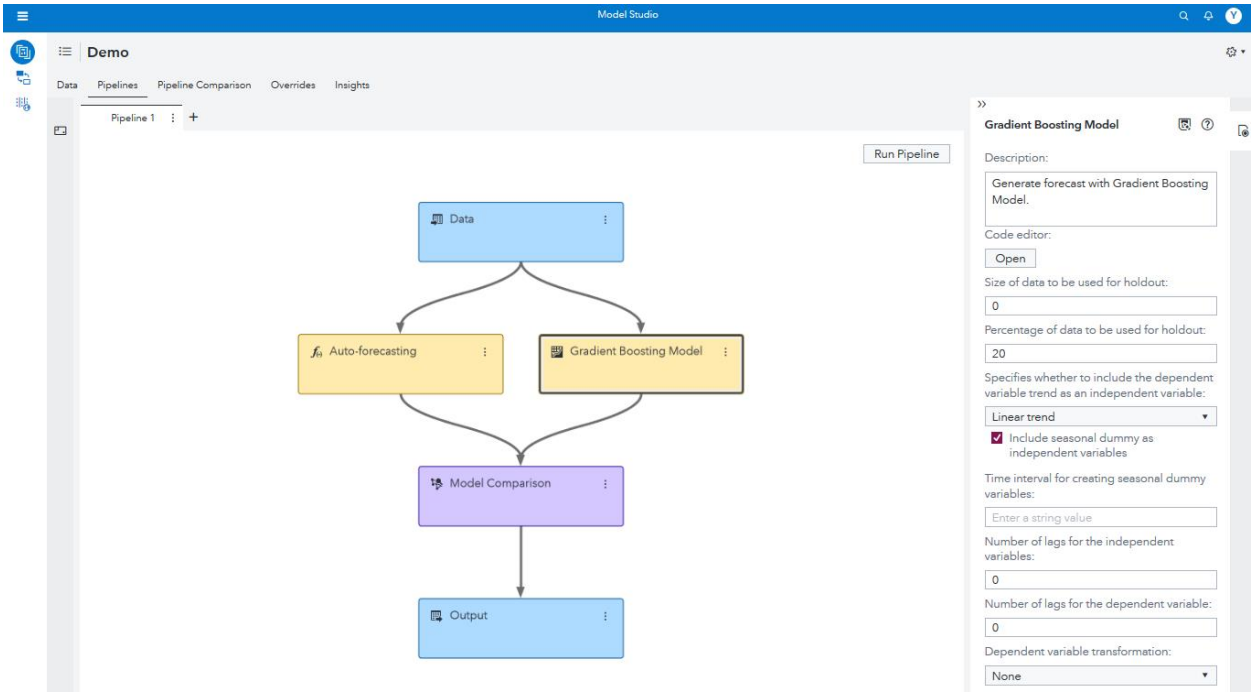
SAS Visual Forecasting automatically promotes the OUTFOR table and the OUTSTAT table (if it exists) so that they can be used as global CAS tables. You can promote any additional output tables in the run-time code by calling the `%vf_promoteCASTable` macro.

## VALIDATION.XML FILE

The validation.xml file validates the specification settings against valid values in XML format. The file conforms to an XML schema that is used to validate any XML you provide to the validation service when you define a new validation model.

When a pluggable modeling strategy is added to a pipeline, the strategy specifications (type, name, displayName, choicelist, and so on) are retrieved from the validation.xml file and are displayed on the right panel of the modeling pipeline interface, with the default values defined in the template.json file and the allowable values defined in the validation.xml file.

A validation model can be best described via an example. Display 3 shows how the GBM example modeling node specifications are displayed in a pipeline.




**Display 3. Modeling Pipeline Interface with the GBM Node Property**

Figure 2 shows the contents of the validation.xml file for this GBM example. The root element in this file is validationModel; it can have a name and a description as shown in Figure 2.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<validationModel
    description="Generate forecast with Gradient Boosting Model."
    name="Gradient Boosting Model" revision="0">
    <links/>
    <version>1</version>
    <properties>
        <property name="_holdoutSampleSize"
                  displayName="Size of data to be used for holdout"
                  type="integer">
            <constraints>
                <range min="0" includeMin="true"/>
            </constraints>
        </property>
        <property name="_holdoutSamplePercent"
                  displayName="Percentage of data to be used for holdout"
                  type="double">
            <constraints>
                <range min="0" max="100"
                 includeMin="true" includeMax="false"/>
            </constraints>
        </property>

        <property name="_trend"
                  displayName="Specifies whether to include the dependent
                               variable trend as an independent variable"
```

```
                type="string">
            <constraints>
                <choicelist>
                  <choice value="None"      displayValue="No trend"/>
                  <choice value="Linear"    displayValue="Linear trend"/>
                  <choice value="Damptrend" displayValue="Damped trend"/>
                </choicelist>
            </constraints>
        </property>
        <property name="_seasonDummy"
                  displayName="Include seasonal dummy
                               as independent variables"
                  type="boolean"/>
        <property name="_seasonDummyInterval"
                  displayName = "Time interval for creating
                                 seasonal dummy variables"
                  type = "string" required="false"
                  enabledWhen="_seasonDummy">
        </property>
        <property name="_lagXNumber"
                  displayName="Number of lags for the independent variables"
                  type="integer">
            <constraints>
                <range min="0" includeMin="true"/>
            </constraints>
        </property>
        <property name="_lagYNumber"
                  displayName="Number of lags for the dependent variable"
                  type="integer">
            <constraints>
                <range min="0" includeMin="true"/>
            </constraints>
        </property>
        <property name="_depTransform"
                  displayName="Dependent variable transformation"
                  type="string">
            <constraints>
                <choicelist>
                    <choice value="NONE" displayValue="None"/>
                    <choice value="LOG" displayValue="Logistic"/>
                </choicelist>
            </constraints>
        </property>

   </properties>
</validationModel>
```

**Figure 2. Contents of validation.xml File for the Gradient Boosting Model Example**

Within the model, properties are defined in a “properties” element, which can contain any number of “property” elements or “group” elements. Each property that is defined in the model describes how values are interpreted by the validation engine when it is asked to validate a map of name, value pairs. The basic required information for a property is as follows:

- **name**: this corresponds to the key used in the map being validated. This is also the macro variable name that can be used in the code.sas file to refer to the property setting value.
- **displayName**: a localizable, user-friendly name for the property. This is provided so that the SAS Visual Forecasting GUI can display meaningful content to instruct users to specify the property values.
- **type**: the type of this property. Currently supported types are string, integer, double, and Boolean.
- **required** (optional): a true or false value that indicate whether a value for this property is required. The default is true.
- **enabledWhen** (optional): a Boolean expression, which, when evaluated, indicates whether this property is enabled. A property that is not enabled is not validated. This field can also be used by the GUI to disable controls that are associated with properties that have been made unavaiable as values in the properties map change. In the GBM example, the **Time interval for creating seasonal dummy variables** box is available only when the **Include seasonal dummy as independent variables** checkbox is checked.
- **Constraints** (optional): constraints on the property value. Properties can have one or more constraints. Typically, only one constraint is needed. Constraints themselves support the "enabledWhen" attribute so that it is possible for a property to have different constraints that depend on some condition. Two types of constraints are currently supported: choice lists and ranges. You can see how these work by referring to the Figure 2. For example, _trend property has a choicelist constraint that requires the choice value to be "None", "Linear", or "Damptrend", and the _holdoutSampleSize property has a range constraint that required its value to be an integer. Note that some properties do not specify any constraints.

## TEMPLATE.JSON FILE

The template.json file contains the metadata about the strategy in JSON format. Figure 3 shows the file contents for the GBM example.

```
{
  "name" : "Gradient Boosting Model",
```

```
  "description" : "Generate forecast with Gradient Boosting Model.",
  "revision" : 0,
  "version" : 1,
  "prototype" : {
    "name" : "Gradient Boosting Model",
    "revision" : 0,
    "executionProviderId" : "Compute",
    "status" : "undefined",
    "componentProperties" : {
      "_holdoutSampleSize": 0,
      "_holdoutSamplePercent": 20,
      "_trend": "Linear",
      "_seasonDummy": true,
      "_lagXNumber": 0,
      "_lagYNumber": 0,
      "_depTransform": "NONE",
      "preProcessTransformationsCode": ""
    }
  },
  "applicationId" : "forecasting",
  "classification" : "pluggable",
  "providerId" : "CustomTemplate",
  "group" : "modeling",
  "hidden" : false
}
```

**Figure 3. Contents of template.json File for the Gradient Boosting Model Example**

The following information is included in this file:

- **name**: name of the strategy; this name will be displayed as the name of the node
- **description**: description of the strategy; this description will be displayed as the description of the node
- **revision**: revision information
- **version**: version information
- **prototype**
    - **name**: Specify the same value as the first name field
    - **revision**: Specify the same value as the first revision field
    - **executionProviderId**: Specify “Compute”
    - **status**: Specify “undefined”

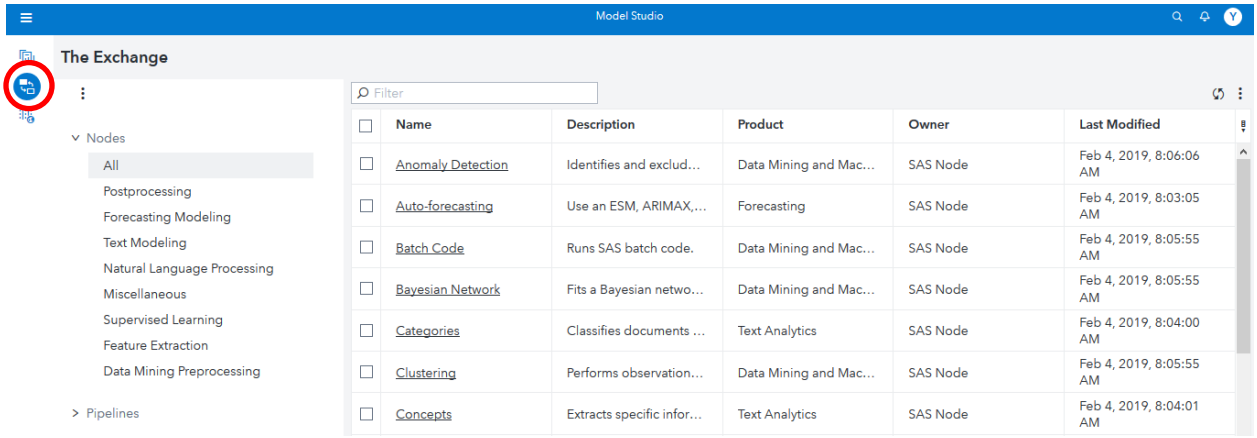
- **componentProperties**: contains the specifications and the corresponding default values. The file can contain two types of properties:
  - Model properties: In this example, seven out of the eight specifications that are defined in the validation.xml file are declared in this file with default values. It is not necessary to declare the "_seasonDummyInterval" option because required="false" is specified in the validation.xml file.
  - Process property: You also need to include one additional property called preProcessTransformationsCode with default value set to "" to make sure the pipeline works as expected.

- **applicationId**: Specify the application, which can be "forecasting", "text", or "datamining". The value "forecasting" should always be used for SAS Visual Forecasting strategies.
- **classification**: Specify "pluggable"
- **providerId**: Specify "CustomTemplate"
- **group**: Specify "modeling"
- **hidden**: Specify false

## PUTTING EVERYTHING TOGETHER

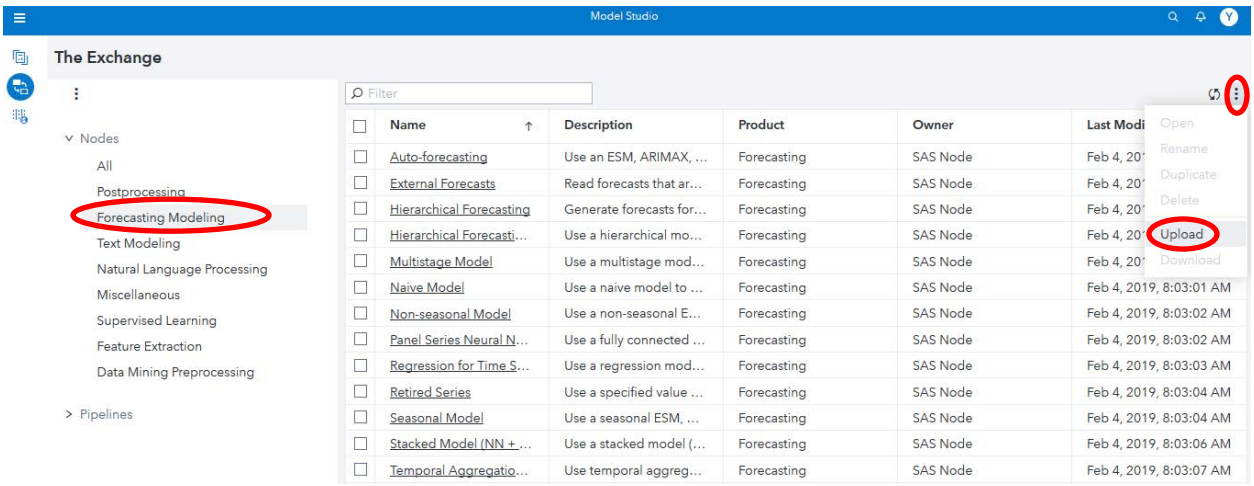
After you create a .zip file from the files of the pluggable modeling strategy, you can upload the .zip file through **The Exchange** as follows:

1. On the **Build Model** tab of the main page of Model Studio, click the  button to open **The Exchange**.



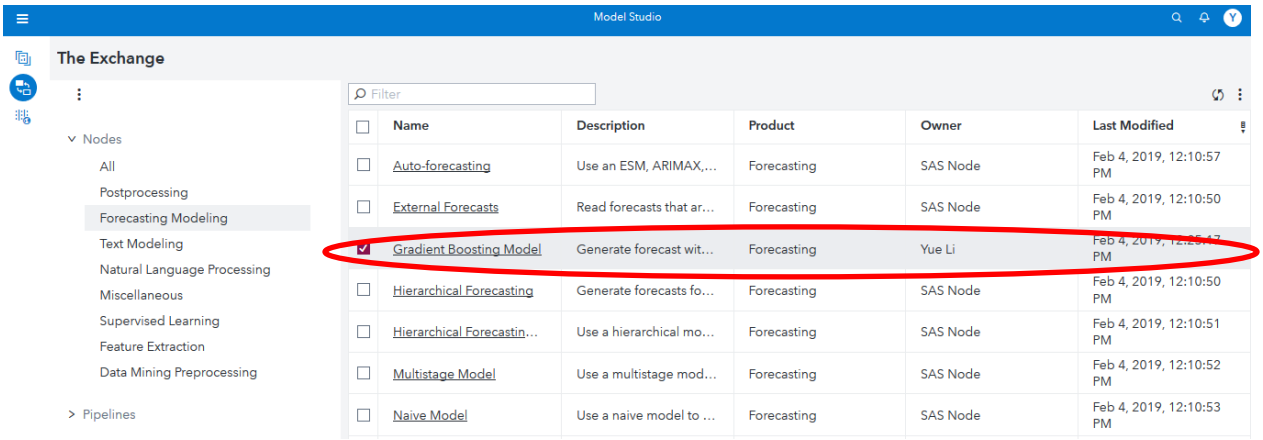

**Display 4. The Exchange in Model Studio**

2. Select **Forecasting Modeling** and click the three dots in upper right corner of the page. Select the **Upload** option to open another window as shown in Display 5. Select the .zip file of the pluggable model that you want to upload and click **OK**.



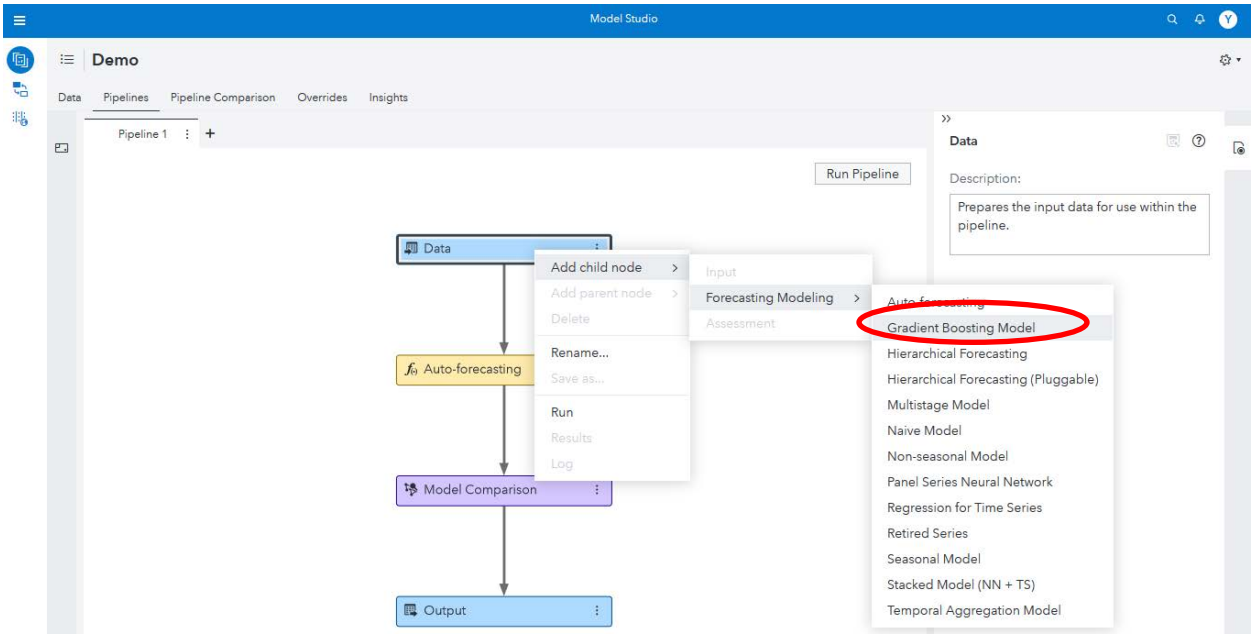

**Display 5. Upload New Modeling Strategy**

After the pluggable modeling strategy has been successfully uploaded, it will show up under the **Forecasting Modeling** node as shown in Display 6.



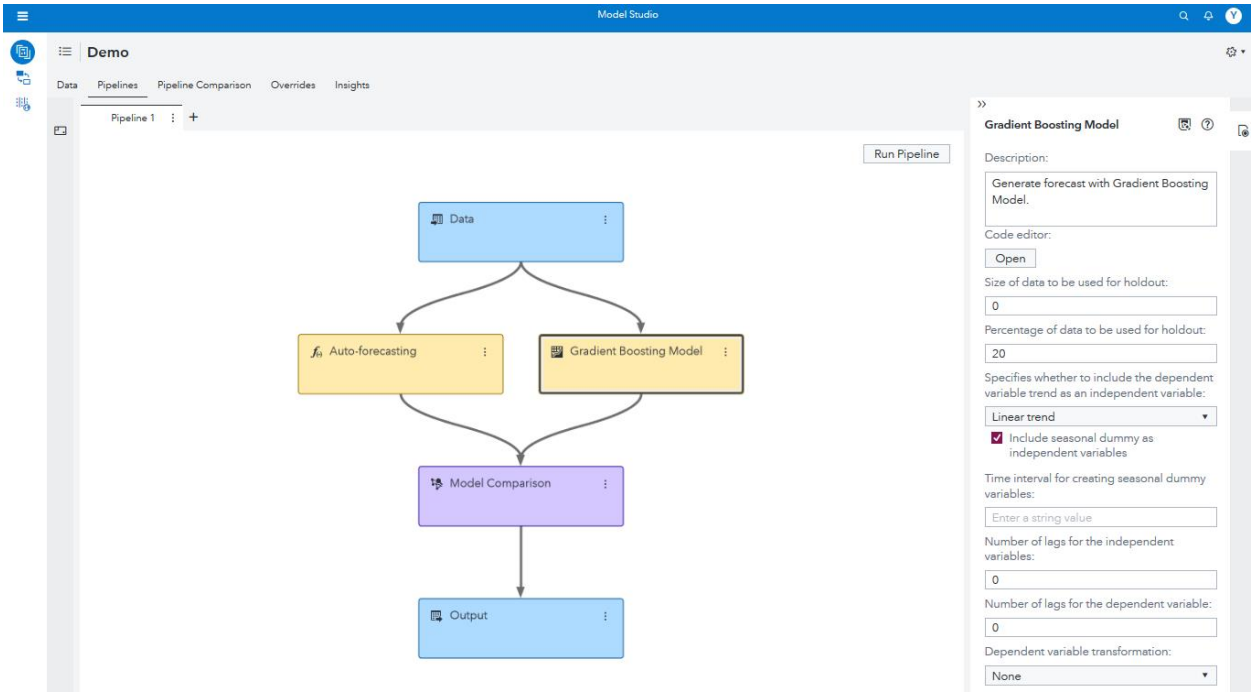

**Display 6. Gradient Boosting Model Appears in The Exchange**

The pluggable model (**Gradient Boosting Model** in this example) becomes available when you build a SAS Visual Forecasting pipeline, as shown in Display 7.




**Display 7. Add the New Gradient Boosting Model Node**

You can use the pluggable model to add a modeling node. When the modeling node is selected, related information is displayed in the side panel. From there, you can view the code or change the parameter settings. In Display 8, all parameter settings that are defined in the GBM model are correctly displayed and are ready to receive your input.




**Display 8. Pipeline with the New Gradient Boosting Model Node**

### Checklist

You can use the following checklist to sort out everything required to build a pluggable modeling node:

1. Create a .zip file that contains the files: template.json, validation.xml, and code.sas.
2. Make sure the code in the code.sas file follows the I/O contracts for input and output tables.
3. Upload the .zip file to **The Exchange**.
4. The pluggable model is ready to use in the pipeline.

## CONCLUSION

This paper introduces how you can build a customized modeling strategy node in SAS Visual Forecasting by going through a step-by-step example of building a gradient boosting modeling node. You can follow the instructions and build your own customized modeling node for forecasting purposes.

## APPENDICES

### APPENDIX I. SYSTEM-DEFINED MACRO VARIABLES

Table 1 shows the system-defined macro variables that are related to input and output.

| Macro Variable | Description |
|---|---|
| vf_session | CAS session name |
| vf_sessionID | CAS session ID |
| vf_caslibIn | caslib where input data is stored |
| vf_caslibOut | caslib where output data is stored |
| vf_libIn | CAS engine libref for input data |
| vf_libOut | CAS engine libref for output data |
| vf_tableInPrefix | Input table name prefix, which is used to differentiate input tables among different projects |
| vf_tableOutPrefix | Output table name prefix, which is used to differentiate output tables among different projects |
| vf_inData | Input table name |
| Vf_inAttribute | Input attribute table name |
| vf_outFor | Output forecast table name |
| vf_outStat | Output statistics table name |
| vf_outInformation | Output information table name |
| vf_outSelect | Output model selection table name |
| vf_outModelInfo | Output model information table name |
| vf_outLog | Output log name |

**Table 1. System-Defined Macro Variables That Are Related to Input and Output**

Table 2 shows the macro variables that are related to data specification.

| Macro Variable | Description |
|---|---|
| vf_depVar | Dependent variable name |
| vf_depVarAcc | Accumulation setting for the dependent variable |
| vf_depVarAgg | Aggregation setting for the dependent variable |
| vf_depVarSetMissing | Missing-value interpretation setting for the dependent variable |
| vf_byVars | BY variables defined for the project |
| vf_timeID | Time ID variable name |
| vf_timeIDInterval | Time series interval |
| vf_timeIDSeasonality | Time series seasonality |
| vf_setMissing | Missing-value interpretation setting for the time ID |
| vf_lead | Forecast lead |
| vf_horizonStart | Forecast horizon start date |
| vf_reconcileLevelNumber | Reconcile level number specified for the project; the _TOP_ level number is 0 |
| vf_reconcileLevel | BY variable that corresponds to the reconcile level specified for the project |
| vf_allowNegativeForecasts | Boolean value indicating whether negative values are allowed in the forecast results |
| vf_indepVars | Strings that specify the independent variable names (for example, promotion, price, and inventory) |
| vf_indepVarsAcc | Accumulation settings for the independent variables |
| vf_indepVarsAgg | Aggregation settings for the independent variables |
| vf_indepVarsSetMissing | Missing-value interpretation settings for the independent variables |

| Macro Variable | Description |
|---|---|
| vf_indepVarsRequired | Specification of whether independent variables are required (YES, NO) |
| vf_indepVarsExtend | Forecast method for extension |
| vf_events | List of events defined in the project |
| vf_eventsRequired | Specification of whether the events are required |
| vf_inEventData | Event definition data |
| vf_inEventObj | inEventObj with all event data definitions specified, which can be used in PROC TSMODEL with the ATSM package |

**Table 2. Macro Variables That Are Related to Data Specification**

## APPENDIX II. SYSTEM-DEFINED MACROS

Table 3 shows the system-defined macros.

| Macro | Description |
|---|---|
| %vf_depVarTSMODEL | Declares the dependent variable for PROC TSMODEL (Note 1) |
| %vf_indepVarsTSMODEL | Declares the independent variables for PROC TSMODEL (Note 1) |
| %vf_varsTSMODEL | Declares statement both the dependent variable and independent variables for PROC TSMODEL (Note 1) |
| %vf_addXTSMODEL(tsdf) | Adds independent variables tothe ATSM package data frame in PROC TSMODEL (Note 2) |
| %vf_addEvents(tsdf, eventsObj) | Declares and adds event specification to the ATSM package data frame and event object in PROC TSMODEL (Note 3) |
| %vf_promoteCASTable(localCASTable = , globalCASTable = ) | Promotes a local CAS table to a global CAS table (Note 4) |

**Table 3. List of Macros**

Notes:

1. The %vf_varsTSMODEL macro generates the VAR statements of the PROC TSMODEL to define the dependent and independent variables. It combines the statements that are generated by the %vf_depVarTSMODEL and %vf_indepVarsTSMODEL macros. The statements also include the ACC= and SETMISS= settings for the corresponding variables. For example, when there is one dependent variable, y, and two independent variables, x1 and x2, the macro generates the following statement:

```
var y /acc=SUM setmiss=MISSING;
var x1/acc=AVG setmiss=AVG;
var x2/acc=AVG setmiss=FIRST;
```

2. The %vf_addXTSMODEL macro generates an addX function call of the Time Series Data Frame (TSDF) object from the Automatic Time Series Modeling (ATSM) package for all the independent variables. The following example shows the use of the %vf_addXTSMODEL macro:

```
declare object dataFrame(tsdf);
%if "&vf_indepVars" ne "" %then %do;
%vf_addXTSMODEL(dataFrame);
%end;
```

3. The %vf_addEvents macro adds event specifications of the TSDF object and the event object from the ATSM package. The following example shows the use of the %vf_addEvents macro:

```
declare object dataFrame(tsdf);
%if "&vf_inEventObj" ne "" or "&vf_events" ne "" %then %do;
declare object ev1(event);
rc = ev1.Initialize();
%vf_addEvents(dataframe, ev1);
%end;
```

4. The %vf_promoteCASTable macro promotes a local CAS table to the global level. If you want to make any table other than the system-defined output tables available for further use, you need to use the %vf_promoteCASTable macro to promote them.

## CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author:

Iman Vasheghani Farahani
SAS Institute Inc
Iman.VasheghaniFarahani@sas.com

Paper SAS4493-2020

# Neural Network–Based Forecasting Strategies in SAS® Viya®

Steven C. Mills, SAS Institute Inc.

## ABSTRACT

Recent literature indicates that hybrids of machine learning and classical time series models are among the top contenders in accurately forecasting the future. Classical linear models are parsimonious and often perform well, but they are unable to capture nonlinear relationships in the data. On the other hand, machine learning models such as neural networks (NNs) are very good at modeling nonlinear effects. Knowing when and how to use machine learning models might seem difficult, but these decisions can be distilled down to best practices that any analyst can use with little experience. This paper discusses several NN-based modeling strategies available in SAS® Visual Forecasting software and the important factors to consider in choosing and training a model. The discussion includes key features of the data that inform the decision to use machine learning models, feature generation options to augment the training process, and best practices to fit a robust model. This knowledge will enable you to leverage the advantages of both NN and linear models to achieve more powerful forecasts.

## INTRODUCTION

Machine learning and hybrid modeling strategies have emerged as top contenders in time series forecasting because of the volume of data and processing power brought about by the information age. Neural networks have become particularly popular because they are able to approximate any functional relationship and they are very well suited for modeling nonlinear relationships between the dependent (target) variable and independent (predictor) variables (Box 1976, Yoshio, Hipel and Mcleod 2005, Taşpınar 2015, Crone and Häger 2016).

SAS Visual Forecasting implements three forecasting strategies that are based on neural networks (NNs): panel series neural network, stacked model, and multistage model. Neural networks might seem mysterious or even intimidating because they have many parameters, but the guidelines in this paper will enable you to successfully apply NNs to forecasting problems and increase your forecasting accuracy. The first section explains how each of the three NN-based modeling strategies is customized for time series forecasting and what types of data work well. Next, a case study shows predictions of ozone levels in Chicago by using an NN-based strategy that easily outperforms classical models. Finally, some effective use strategies not covered in the example are discussed.

After reading this paper, you will understand what types of data work well with these modeling strategies and you will be able to effectively apply these strategies to your own forecasting problems. Whether the volume of data is medium size, big, or huge, these modeling strategies can help identify complex relationships between variables and increase the predictive power of your models.

### BACKGROUND

Some basic knowledge of neural networks is presented here in order to provide a foundation for the concepts discussed later. A neural network is composed of an input layer, one or more hidden layers, and an output layer. An example NN with one hidden layer is shown in Figure 1a. Each input node has a connection to every node in the first hidden layer. Likewise, every node in the hidden layer has a connection to the node in the output layer.

Figure 1b expands the hidden node $n_4$ to illustrate how the output of a node is calculated. (Nodes $n_3$ and $n_5$ are omitted for clarity.) The input layer simply passes through the values in the input vector X such that the output of node $n_1$ is the value $x_1$ and the output of $n_2$ is the value $x_2$. The output of each node is multiplied by a connection, $w_{jk}$, where $j$ and $k$ represent the nodes being connected. The hidden node adds up the input values and a bias parameter, $b_k$, and feeds the sum into a nonlinear activation function to produce the hidden node output. The output from $n_4$ is multiplied by the corresponding connection weight, $w_{46}$, and fed forward to node $n_6$.

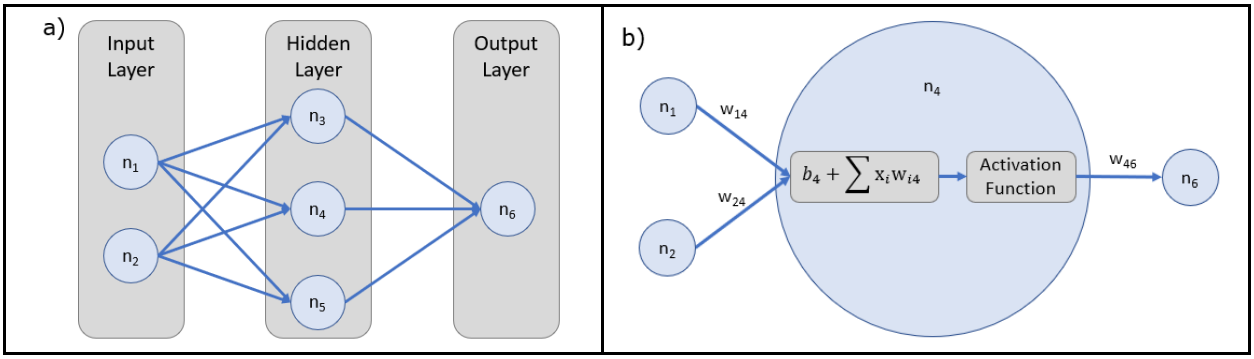



**Figure 1. (a) Example Neural Network Architecture and (b) Functionality of a Node**

Neural networks learn through an iterative cycle of training and validation. Training data are fed forward through the NN to calculate an output value. Then back propagation is used to update the connection weights and reduce the error. The details of back propagation are outside the scope of this paper, but they can be summarized succinctly as follows: The error is measured with respect to the training data, and partial derivatives are calculated with respect to each connection weight. The aggregated partial derivatives are used to update the connection weights and reduce the error (Goodfellow, Yoshua and Courville 2016). The error with respect to the validation data is measured periodically to validate the model training process.

## NEURAL NETWORK MODELING IN SAS VISUAL FORECASTING

Some special considerations are required to successfully use NN-based models in time series forecasting. Fortunately, the modeling strategies in SAS Visual Forecasting take care of a lot of the details by structuring the data and generating additional features before training the model. However, it is important to understand how the data are interpreted and when to use the extracted features. This section describes how the NN-based strategies in SAS Visual Forecasting structure interpret the data compared to classical forecasting and machine learning.

### DATA STRUCTURE

In classical time series forecasting, BY variables are used to delineate a panel of related time series and find the best model for each series individually. This approach is intractable for NNs because they require significantly more data to train than classical models require. A single time series is typically not enough.

The neural network modeling strategies in SAS Visual Forecasting are designed for panel data that consist of multiple related time series. For example, a retail chain might have many time series that are delineated by BY variables (such as STORE and SKU) and independent variables (such as promotions, number of shoppers, and calendar events).

The time series are concatenated together as shown in Figure 2 and modeled as a single series with the BY variables included as categorical independent variables. The resulting input table is a concatenation of all the series.

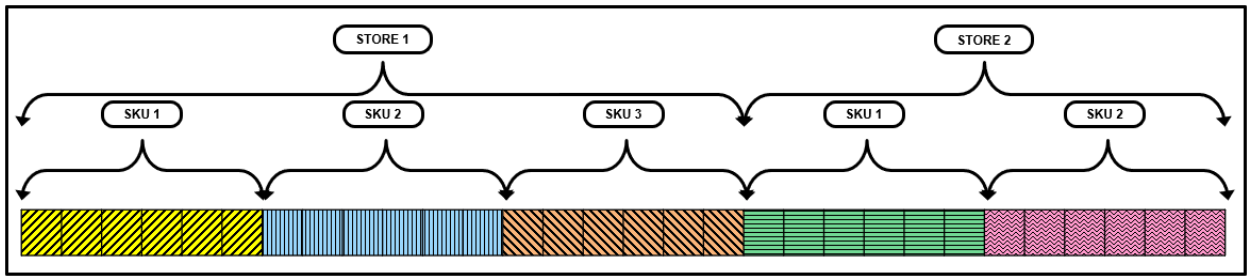



**Figure 2. Five Concatenated Time Series Delineated by the BY Variables STORE and SKU**

## DATA PARTITIONING

A typical machine learning algorithm randomly partitions the data into training, validation, and test partitions. Model fitting is accomplished through alternating cycles of using the training data to minimize the training partition error and validating the progress by checking the validation partition error. This cycle continues until some stopping criterion is met.

The training and validation partitions are analogous to the training and holdout samples in time series forecasting where the holdout portion is used to select the model that best generalizes to new data. The out-of-sample (test) data are used to measure how well the model predictions generalize to new data.

Random sampling in order to partition data is acceptable for machine learning applications, but in time series the most recent data usually have a larger impact on predicted values. To adjust for this, the NN-based forecasting strategies use ordered sampling. Random and ordered sampling are illustrated in Figure 3 for a single time series. In ordered sampling, the oldest data are placed in the training partition and more recent data are placed in the holdout (validation) partition. The out-of-sample (test) partition contains the most recent data leading up to the forecast horizon.

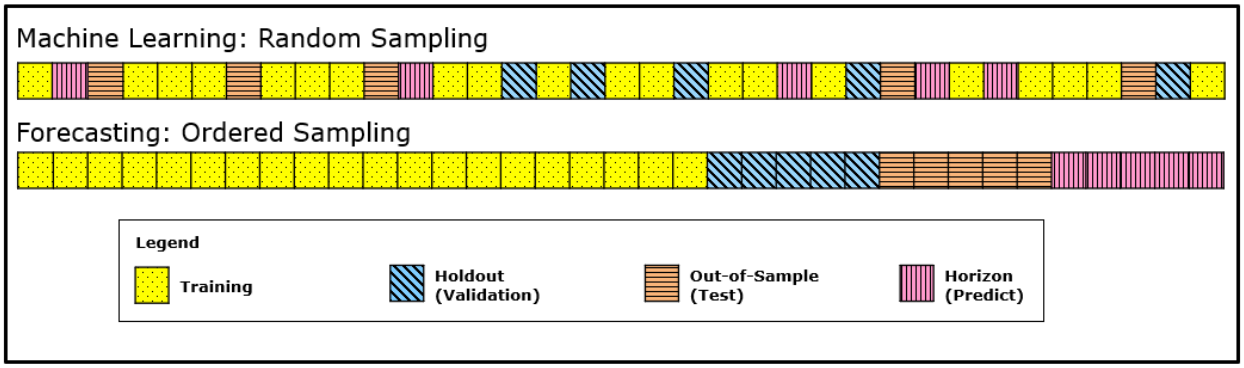



**Figure 3. Partitioning for Machine Learning (Random Sampling) vs. Forecasting (Ordered Sampling)**

## MODELING STRATEGIES

### Panel Series Neural Network

The panel series neural network (PSNN) can be used to implement a neural network like the one described in the introduction. Figure 4 illustrates the general flow of operations in the PSNN modeling strategy. The input data first go through a preprocessing step where data are partitioned, transformed, and/or standardized. Next, salient features are extracted. After preprocessing and feature extraction, the NN learns how to fit a model to the data. Finally, the output data are reverse-transformed and destandardized back to the original scale to produce the final forecast.

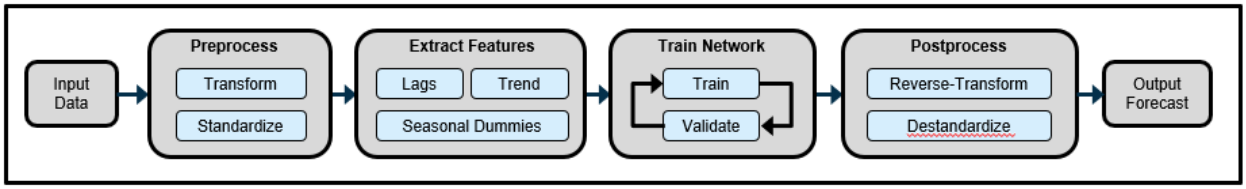



**Figure 4. Panel Series Neural Network Modeling Strategy**

### Stacked Model

The stacked model uses the PSNN to create an initial forecast of the target variable and then models the residuals by using a classical time series approach. The forecasts of the input data and the residuals are then added together to generate the final forecast, as shown in Figure 5.

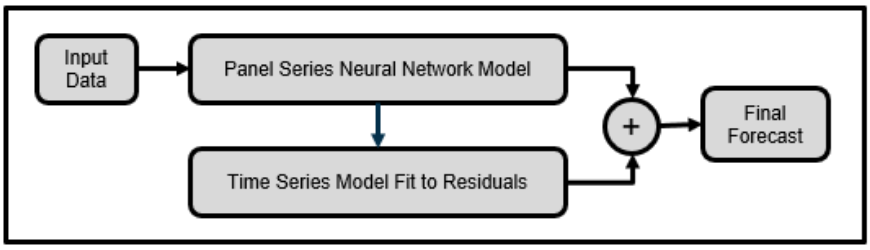



**Figure 5. Stacked Modeling Strategy**

### Multistage Model

The multistage model is a twist on hierarchical forecasting. The lowest levels in a time series hierarchy are often intermittent or display more nonlinear characteristics that challenge time series models. Figure 6a shows the block diagram, and Figure 6b shows an example hierarchy. The lower levels are modeled by using regression or a neural network, whereas the higher levels are fit to time series models. The forecasts are reconciled at a user-specified level of the hierarchy to produce the final forecast.

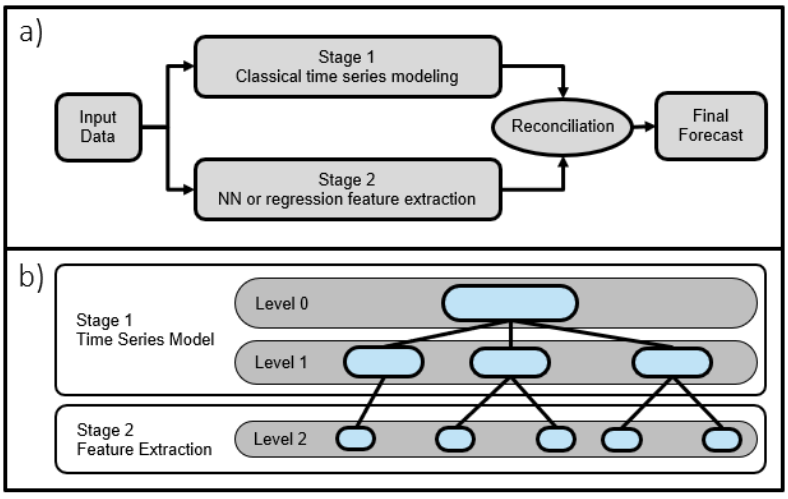



**Figure 6. (a) Block Diagram of Multistage Modeling Strategy and (b) Example Division of Hierarchy Levels**

## WHEN SHOULD YOU USE A NEURAL NETWORK MODEL?

Neural networks are not an all-purpose magic tool to replace classical models (Zhang 2003). To quote George Box, "All models are wrong, but some are useful" (Box and Draper 1987).

For a neural network to be useful, you need more training data than classical time series models require, and you need independent variables that have complex nonlinear relationships to the dependent variable (Box 1976, Yoshio, Hipel and Mcleod 2005, Crone and Häger 2016)

Typical data for which neural networks work well have the following characteristics:

- Large data volume, such as a panel of time series
- Historical data length of at least 300 time ID values
- At least three independent variables, which ideally have nonlinear relationships to the target
- Few (or zero!) missing values

### QUICK-START SETTINGS

If you just want to jump in and start training some models, then the tables in this section offer a good starting point for the PSNN and stacked models. The multistage model uses similar settings if neural networks are chosen to model the lower levels.

| Feature Extraction | |
|---|---|
| Dependent lags | Max($p$) from AR($p$) |
| Independent lags | Equal to dependent lags |
| Seasonal dummies | Yes |
| ESM forecast | Choose one |
| Linear trend | |
| Damped trend | |

| Model Initialization | |
|---|---|
| Input standardization | Z-score |
| Number of hidden layers | 1 |
| Number of neurons | 10 |
| Hidden layer activation function | Rectifier |
| Direct connections | No |
| Dependent var transformation | None |
| Dependent var standardization | Z-score |
| Error function | Normal |
| Output layer activation function | Identity |
| Neuron connection distribution | Xavier |

| Model Training | |
|---|---|
| Algorithm | LBFGS |
| Number of tries | 10 |
| Max training iterations | 300 |
| Max time | 10 |
| L1 regularization | 0 |
| L2 regularization | 0 |
| Enable early stopping | Yes |
| Stagnation limit | 10 |
| Enable Autotune | No |

**Table 1. Quick-Start Settings for PSNN and Stacked Models**

## CASE STUDY

### DATA DESCRIPTION

The data set that is used in this example was obtained from Chicago's Array of Things (AoT) project. The data consist of one month of sensor output from modules that are placed around the city to measure air quality, light, and other environmental conditions.

### DATA PREPARATION

Cleaning the data is an important first step to ensure that the model can make accurate predictions. This process is outside the scope of this paper, but the code can be found here: https://github.com/sascommunities/sas-global-forum-2020/tree/master/papers/4493-2020-Mills. The AoT data had many missing values and broken or malfunctioning sensors.

Columns for irrelevant variables and sensors that were clearly malfunctioning are removed, and the important independent variables are chosen. You might be wondering why you should choose important variables instead of including everything. It is true that NNs require a large amount of data, but the data quality is important too. Extraneous variables and multicollinearity among variables can reduce forecast accuracy and increase the training

time significantly, so unimportant variables should be excluded (Diaconescu 2008, Christ, Kempa-Liehr and and Feindt 2016). Variable importance can be evaluated by using machine learning methods or an ARIMAX model if required, or you can use domain knowledge about ozone formation, which indicates that UV light, temperature, humidity, and the presence of other pollutants (such as nitrogen oxides from combustion engine exhaust) are important factors. Gas sensors require calibration and exhibit signal drift over time, so it would be best not to use other gas sensors to predict the output of the ozone sensor. In this case study, the chosen predictors are temperature, humidity, and infrared, ultraviolet, and visible light.

After cleaning and organizing the data, 27 series of hourly sensor data remain. The series are delineated by the BY variable, and they are accumulated to an hourly interval for a length of 720 observations each. One final step splits the data into two tables: AoT_train and AoT_test. The resulting data set contains eight columns and 19,440 rows. Table 2 describes the variables in the data.

| Variable name | Role | Description |
|---|---|---|
| Timestamp | Time ID | Timestamp for observation |
| Sensor_node_id | BY variable | BY variable from original data |
| O3_concentration | Dependent variable | Ozone sensor output |
| Si1145_ir_intensity | Independent variable | Infrared light sensor output |
| Si1145_uv_intensity | Independent variable | Ultraviolet light sensor output |
| Si1145_visible_light | Independent variable | Visible light sensor output |
| At1_temperature | Independent variable | Temperature sensor output |
| Hih4030_humidity | Independent variable | Humidity sensor output |

**Table 2. Variables in the Input Data Set**

## FEATURE EXTRACTION

Now that the data are prepared, you are ready to choose the features to extract. The following sections describe how you can select the appropriate number of lags, the seasonal dummy variables, and the type of trend component.

It is important to understand how feature extraction impacts the effective number of input variables and model parameters. For example, generating 3 lags of 10 variables results in a total of 40 variables and 40 nodes in the input layer. If there are 5 hidden nodes in the first hidden layer, then there are 200 connection weight parameters to solve. A NN model can quickly become very complex which causes the training time to increase dramatically.

### Lags

Neural network theory assumes that each observation is independent from any other observation. This assumption makes it difficult to learn autocorrelated features that are common in time series, such as trends and seasonality. Generating lags of the dependent and independent variables helps the model understand the local level and trend in a series similarly to the way an ARIMA model uses an autoregressive (AR) term.

It is important to realize how missing values affect the generation of lagged variables. Consider the example shown in Table 3, which consists of six observations. The independent variable, X, has one missing value, and three lags have been generated for both X and Y.

| Date | Y | Lag1(Y) | Lag2(Y) | Lag3(Y) | X | Lag1(X) | Lag2(X) | Lag3(X) |
|---|---|---|---|---|---|---|---|---|
| 1-Jan | 16 | . | . | . | 4 | . | . | . |
| 2-Jan | 25 | 16 | . | . | 6.25 | 4 | . | . |
| 3-Jan | 21 | 25 | 16 | . | . | 6.25 | 4 | . |
| 4-Jan | 22 | 21 | 25 | 16 | 5.5 | . | 6.25 | 4 |
| 5-Jan | 14 | 22 | 21 | 25 | 3.5 | 5.5 | . | 6.25 |
| 6-Jan | 17 | 14 | 22 | 21 | 4.25 | 3.5 | 5.5 | . |

**Table 3. Example of Missing Values Propagating When Lags Are Generated**

Notice that the first three observations contain missing values in the lags of the dependent variable (Y) because the historical data before January 1 is not available. Also notice that the missing value for the independent variable (X) on January 3 propagates downward, resulting in missing values for the last three observations. Every observation in this data set has a missing value, so none of them can be used in the neural network model. If your data have many missing values or you use many lags (or both), then you quickly run out of complete observations for training the NN.

Fitting an AR($p$) model by using the TSMODEL procedure and including independent variables will indicate an appropriate number of lags and show whether a trend component is detected. The code to fit the AR model over the 27 series and generate a histogram of the AR orders can be found here: https://github.com/sascommunities/sas-global-forum-2020/tree/master/papers/4493-2020-Mills. The histogram is shown in Figure 7 and indicates that three lags should be enough. You might notice the total number of series in the histogram is 20 rather than 27. Some of the series could not be fit to an AR($p$) model and are excluded. This is an important point: The AR($p$) model indicates the appropriate number of lags, but a small adjustment to that number at the end of the analysis might result in a better model.

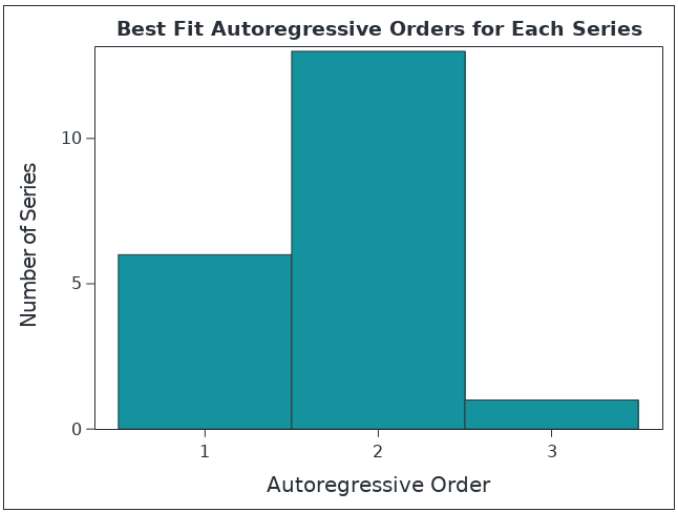



**Figure 7. Histogram of Autoregressive Orders Determined by Fitting an AR($p$) Model with Exogenous Variables**

### Seasonal Dummy Variables

Generating lags of variables quickly increases the volume of data and the time required to train a neural network. In addition, some data are sacrificed at the beginning of each series as shown previously. Imagine generating a seasonal lag for monthly data, which results in a

whole year of data that can't be used because of missing values. Sacrificing a large portion of the historical data is usually a bad idea.

Seasonal dummy variables can be included to help capture seasonal fluctuations in the data without any data loss. In addition, the seasonality can be specified independently of the time series seasonality. For example, if the time series is accumulated to a weekly interval, then the default behavior would generate 52 seasonal dummy variables to cover the 52 weeks in a year. Specifying "month" or "qtr" as the seasonal dummy interval results in fewer dummy variables and faster convergence when training a neural network.

The AoT data in this example are accumulated to an hourly interval, so 24 seasonal dummy variables are generated, one for each hour of the day. This could be very useful because certain times of the day have increased traffic or sunlight to contribute to ozone generation. Seasonal dummy variables allow capturing information about the hour of the day without creating many lagged variables. Avoiding the use of many lagged variables results in a smaller data table and a faster training process.

### Trend Component

A trend component can be extracted from the data to help the model learn. You can choose a linear trend, a damped trend, or an exponential smoothing model (ESM) of the dependent variable to include as an independent variable. The linear and damped trends are special cases of the ESM, so only one of these options should be selected. If an ESM is selected, then the time series forecasting engine chooses the best ESM type on the basis of the training and holdout data partitions. For more control over the extracted model, you can choose between a linear or damped trend on the basis of the length of the holdout and forecast periods. When forecasting further into the future, linear trends often overestimate a series. A damped trend will yield better results.

If you rerun the AR($p$) model code on the AoT data without specifying the trend component, then the outModelInfo table shows that just over half of the series have some type of trend. Since you are forecasting out 24 periods, a damped trend is likely the better choice.

This analysis suggests the following features to include in the NN model:

- three lags of salient variables
- hourly seasonal dummy variables
- damped trend component

## MODEL INITIALIZATION AND TRAINING

You are now prepared to create a project and generate forecasts using NNs in SAS Visual Forecasting. The following steps walk you through the project creation and configuration:

1. Create a new forecasting project in Visual Forecasting with the AOT_train data set.
2. Go to **Project Settings** from the gear icon in the upper right corner and change the forecast horizon from 12 to 24.
3. On the **Data** tab, assign the variable roles as described in Table 2.
4. On the **Pipelines** tab, delete the Auto-forecasting node if you started with one.
5. Add a Hierarchical Forecasting (Pluggable) node and a PSNN node to the pipeline.
6. Set the Feature Extraction options in the PSNN node. For more information about these features, see the section, "Feature Extraction."
7. Under **Model Selection**, set **Holdout Sample Size** to 24 for both modeling nodes.
8. The additional settings for the PSNN nodes are summarized in Table 4.

| Feature Extraction | |
|---|---|
| Dependent lags | 3 |
| Independent lags | 3 |
| Seasonal dummies | Yes |
| ESM forecast | No |
| Linear trend | No |
| Damped trend | Yes |

| Model Initialization | |
|---|---|
| Input standardization | Z-score |
| Number of hidden layers | 1 |
| Number of neurons | 10 |
| Hidden layer activation function | Rectifier |
| Direct connections | No |
| Dependent variable transformation | None |
| Dependent variable standardization | Z-score |
| Error function | Normal |
| Output layer activation function | Identity |
| Neuron connection distribution | Xavier |

| Model Training | |
|---|---|
| Algorithm | LBFGS |
| Number of tries | 10 |
| Max training iterations | 300 |
| Max time | 10 |
| L1 regularization | 0 |
| L2 regularization | 0 |
| Enable early stopping | Yes |
| Stagnation limit | 10 |
| Enable Autotune | No |

**Table 4. PSNN Settings Used to Fit a Model to the AoT Data**

Running the pipeline and viewing the results show that the PSNN performs substantially better than hierarchical forecasting on in-sample data. Keep in mind that the NN training process depends on random initialization and that race conditions in parallel processing threads can also cause variation in output. Therefore, you might see slightly different numbers for the PSNN weighted mean absolute percentage error (WMAPE) measurements.

To view the out-of-sample results, substitute AoT_test for AoT_train on the **Data** tab, change the **Forecasting Task** from **Diagnose** to **Forecast** for each modeling node, and rerun the pipeline. The WMAPE values are summarized in Table 5, and the last two days of forecasts are plotted in Figure 8 for a representative sensor node. The shaded region in the right half of Figure 8 is the forecast horizon.

| Model | In-sample WMAPE | Out-of-sample WMAPE |
|---|---|---|
| Hierarchical forecasting | 23.0136 | 10.9508 |
| PSNN | 10.9310 | 9.7429 |

**Table 5. Weighted MAPE Calculated for the PSNN and Hierarchical Models**

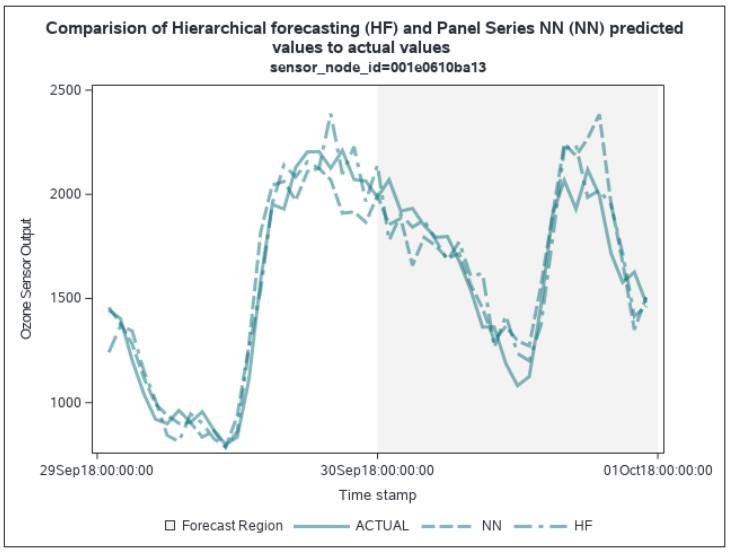



**Figure 8. Forecast Comparison between PSNN and Hierarchical Forecasting**

To demonstrate the stacked model, create a second pipeline with an Auto-forecasting node and a Stacked Model node. These two nodes do not currently support the **Forecast Task** or **Holdout** options, so they are compared against each other based on the in-sample accuracy. Configure the stacked model with the same feature generation and model parameters as were used to configure the PSNN. Then, run the pipeline and view the results. Table 6 shows that the stacked model outperforms auto-forecasting and hierarchical forecasting for this data set.

| Model | In-sample WMAPE |
|---|---|
| Auto-forecasting | 22.1265 |
| Hierarchical forecasting | 20.8441 |
| Stacked model | 10.8316 |

**Table 6. Weighted MAPE Calculated for the Auto-forecasting, Hierarchical, and Stacked Models**

## SCALING

Training time for neural networks is heavily influenced by the amount of data, the available processing power, and the training specifications. Figure 9 shows the training time dependence for the AoT data as the number of BY groups increase 10×, 50×, and 500×. These additional data are simulated by creating another BY variable and duplicating the original data. The plot displays a nice linear trend as the data size increases.

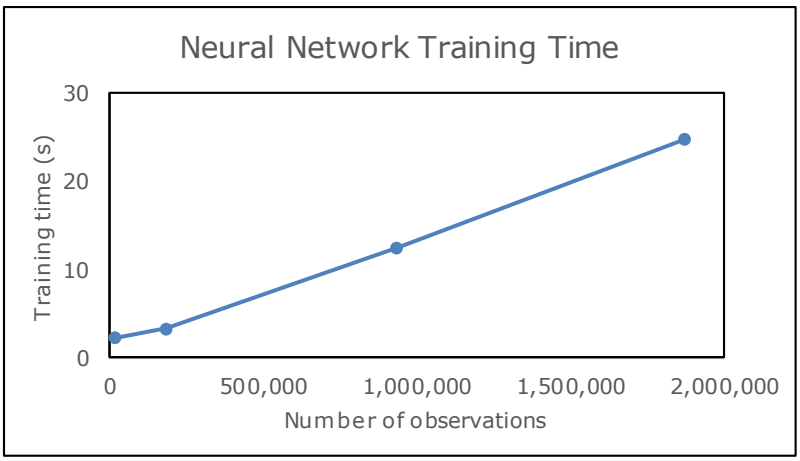



**Figure 9. Scaling of Neural Network Training Time with Number of Observations**

Scaling the data down provides insight into the minimum amount of data required. The effect of reducing the length of the historical record is shown in Figure 10. Using the full month of data (720 observations in each series) provides the data in Table 5. The WMAPE increases approximately linearly as data are discarded until around 50% (360 observations) remain. Further reduction of the historical record length results in a more drastic increase in WMAPE. Be cautious if you have only a few hundred historical time points to work with.

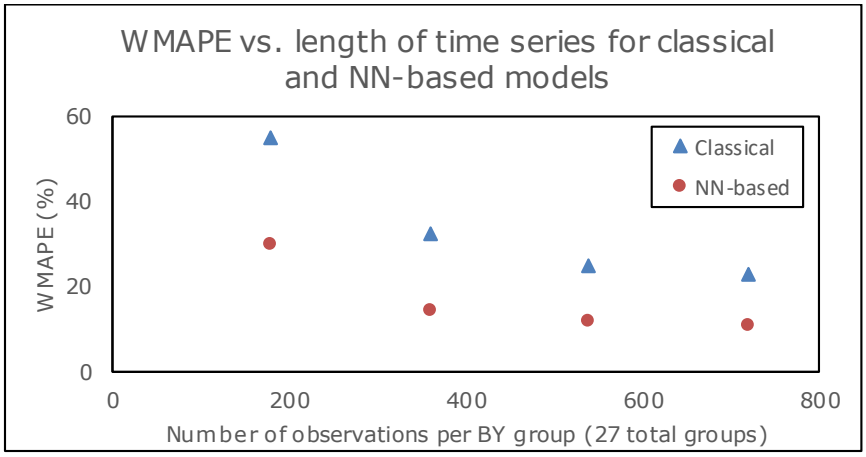



**Figure 10. WMAPE as a Function of Historical Record Length for Classical and NN-Based models**

Reducing the number of BY groups and holding the series length constant does not necessarily impact the error measurement the same way. Figure 11 shows the lowest error measurement for seven BY groups. Increasing the number of BY groups causes an increase in WMAPE for both classical and NN-based models. Further increasing the number of series shows a slight reduction in WMAPE for NN-based models, whereas the classical models display higher WMAPE with no clear trend. These data indicate that additional BY groups might help reduce forecasting error, but also indicate that a small number of similar series might produce better results. The small number of series has less variation (allowing for a tighter fit), but the model would likely be less accurate if it were used to forecast the other series.

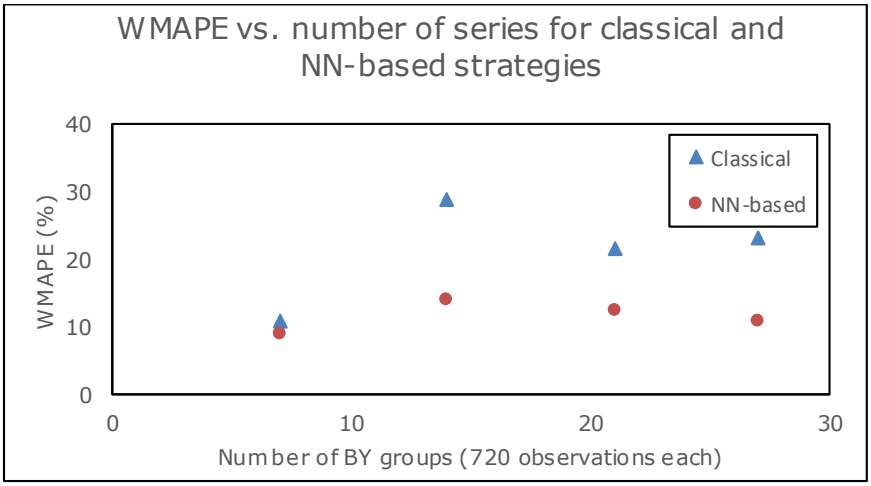



**Figure 11. WMAPE as a Function of the Number of BY Groups for Classical and NN-based models**

## BEST PRACTICES AND OTHER TIPS

Neural network strategies have many parameters to set up and initialize. This section discusses some common pitfalls that are related to the different parameters. Data standardization and activation functions must be chosen along with architectural choices such as the numbers of hidden layers and nodes and the initial neuron connection weights. Model training options such as the optimization algorithm, number of tries, and stopping criteria must also be specified.

## OUT-OF-SAMPLE TESTING

Neural networks and other machine learning models are prone to overfitting during the training process. Overfitting causes the models to make very accurate predictions in the training data but generalize poorly to new data. SAS Visual Forecasting includes a robust early-stopping option that does a good job of preventing overfitting, but reserving part of the data for out-of-sample testing is still a good practice. The following code shows how to subset the data easily in SAS Studio and save the data to the "public" caslib before beginning the modeling process:

```
data public.hourly_data_train;
   set public.hourly_data;
   where timestamp lt "30SEP18:00:00"dt;
run;

data public.hourly_data_test;
  set public.hourly_data;
  where timestamp ge "30SEP18:00:00"dt;
run;

proc cas;
    table.save /
        table={name="hourly_data_train", caslib="public"}
        name="hourly_data_train.sas7bdat"
        caslib="public";
    table.save /
        table={name="hourly_data_test", caslib="public"}
        name="hourly_data_test.sas7bdat"
        caslib="public";
run;
quit;
```

## NEURAL NETWORK ARCHITECTURE

Choosing the number of hidden layers and neurons is the primary way that you can specify the NN architecture. As in time series models, parsimony is your friend here. Smaller neural networks often perform better than those that have many nodes and layers for two reasons: First, the training time and the number of model parameters increase rapidly with the number of nodes and layers, thus requiring longer times to train. Second, as the network size grows, it is more likely to learn more complex interactions, which eventually leads to overfitting (Diaconescu 2008). It is recommended that you start with a single layer containing 10 nodes and adjust from there if you are not satisfied. Autotuning can also be applied, but it has an even more drastic effect on the training time and is often not necessary. The distribution of connection weights should also be specified. The Xavier and normal distributions are good choices.

## ACTIVATION FUNCTIONS AND STANDARDIZATION

Not all activation functions and data standardization methods are compatible with each other. Two options are available for standardization of the input and output data: z-score and mid-range. Z-score standardization scales values to a zero mean and standard deviation equal to 1, whereas mid-range standardization scales values to fall within the range (−1,1). Each activation function also has some range of output values, as shown in Table 7.

The activation function for the output layer must be compatible with the output standardization method. Consider an example that uses the tanh function on the output layer and z-score standardization for the output values. In this case, the output tanh function is trying to map its (−1,1) output to data with a range of (−σ, σ), which results in loss of information for |σ| > 1, where σ is the standard deviation.

| Activation Function | Input Range | Output Range |
|---|---|---|
| sigmoid | [-∞,∞] | (0,1) |
| logistic | [-∞,∞] | (0,1) |
| tanh | [-∞,∞] | (-1,1) |
| rectifier | [-∞,∞] | [0,∞] |
| identity | [-∞,∞] | [-∞,∞] |
| sine | [-∞,∞] | (-1,1) |

**Table 7. Activation Function Ranges**

Neural networks for time series generally work well by using a rectifier activation function for hidden layers and an identity activation function on the output layer. This configuration allows any standardization method for the input and output variables. If you try different activation functions, then you must ensure compatibility between the output layer activation function and the output standardization.

### ALGORITHM

Two algorithm options are available for training the PSNN and stacked model. Historically, stochastic gradient descent (SGD) has been used often for neural networks, but SGD frequently requires subtle tuning of its parameters such as the learning rate and annealing rate. If your data set is large (over 100,000 rows), then SGD might be a good option. However, for most applications of SAS Visual Forecasting, the limited-memory Broyden-Fletcher-Goldfarb-Shanno (LBFGS) algorithm converges faster and is easier to use.

### OVERFITTING

Regularization is one technique to avoid overfitting because it penalizes large connection weights. L1 and L2 regularization terms can be specified, but they should be used sparingly because incorporating lagged variables into the training data can also have a regularizing effect and lead to an underfitted model. You can start with 0 for both L1 and L2 regularization and try increasing them if your out-of-sample fit is much worse than the in-sample fit.

The early-stopping mechanism is very effective at stopping the training/validation cycle when the validation fit stops improving. However, the error that is measured over the input space is almost always nonconvex, which means that multiple local minima can be found. The stagnation property specifies how many training iterations can complete with no improvement in the validation fit before ending the training/validation cycle. This helps the solver find its way out of local minima so that it can continue searching for better solutions.

## CONCLUSION

This paper introduces the neural network-based modeling strategies available in SAS Visual Forecasting and provides guidelines for using them effectively. It explains important considerations for using machine learning strategies for time series forecasting, including structuring of the input data and extraction of features from the input data to aid in

modeling. Guidelines for determining when to use a machine learning model are shown, and an example panel series neural network model was fit to data from Chicago's Array of Things environmental monitoring project. The example was shown to significantly outperform hierarchical forecasting, as evidenced by a WMAPE error measurement about half the size of that for the classical model. Other details and pitfalls not covered in the example are explained, including architecture considerations, regularization, and compatibility of activation functions with data standardization techniques. Finally, the algorithm scaling is evaluated, showing that training time increases linearly with the number of observations.

## REFERENCE


Box, George E P. 1976. "Science and Statistics." *Journal of the American Statistical Association* 71 (356): 791-799.

Box, George E P, and Norman R Draper. 1987. *Empirical Model-Building and Response Surfaces.* New York: Wiley.

Christ, Maximilian, Andreas W Kempa-Liehr, and Michael and Feindt. 2016. "Distributed and Parallel Time Series Feature Extraction for Industrial Big Data Applications." *ArXive Pre-Print.* arXiv:1610.07717v3.

Crone, Sven F, and Stephan Häger. 2016. "Feature Selection of Autoregressive Neural Network Inputs for Trend Time Series Forecasting." *IEEE Internations Joint Conference on Neural Networks.*

Diaconescu, Eugen. 2008. "The Use of NARX Neural Networks to Predict Chaotic Time Series." *WSEAS Transactions on Computer Research* 3 (3): 182-191.

Goodfellow, Ian, Bengio Yoshua, and Aaron Courville. 2016. *Back-Propagation and Other Differentiation Algorithms.* Cambridge: The MIT Press.

Taşpınar, F. 2015. "Improving Artificial Neural Network Model Predictions of Daily Average PM10 Concentrations by Applying Principle Component Analysis and Implementing Seasonal Models." *Journal of the Air & Waste Management Association* 65 (7): 800-809.

Yoshio, Kajitani, Kieth W Hipel, and A Ian Mcleod. 2005. "Forecasting Nonlinear Time Series with Feed-Forward Neural Networks: A Case Study of Canadian Lynx Data." *Journal of Forecasting* 24 (2): 105-117.

Zhang, G. 2003. "Time Series Forecasting Using a Hybrid ARIMA and Neural Network Model." *Neurocomputing* 50: 159-175.


## ACKNOWLEDGMENTS


Special thanks go out to the Array of Things team, the City of Chicago, the National Science Foundation, and any other contributors to the Array of Things project for their hard work and for making the data available to the public.


## CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:


Steven Mills
Steven.Mills@sas.com

Paper SAS3294-2019

# Ten Underused Features That Improve Your SAS® Forecast Server Workflow

Michael Leonard, SAS Institute Inc., Cary, NC, and Evan Anderson

## ABSTRACT

SAS® Forecast Server is one of the most feature-rich forecasting products on the market. This paper describes 10 underused features to improve your workflow. First, start with the data: (1) Use SAS® Time Series Studio to become familiar with your data and define a hierarchy, (2) catch data problems through warnings about the time ID variable, (3) look under the covers by using the SAS® log, and (4) use adjustment variables and start-up code to fix data issues. Next, improve the forecasts: (5) Define recurring events that influence the time series, (6) add models by importing from an external list, (7) use rolling simulations to evaluate forecast accuracy over the number of periods you need to forecast, (8) evaluate the effects of independent variables by using scenario analysis, and (9) gain insight into results by comparing models. Finally, put your workflow into production: (10) Run the code in batch.

## INTRODUCTION

More information than ever before is being collected with associated timestamps. Computers, cell phones, smart devices, detectors, and other devices record timestamped data. These timestamped data can be modeled, forecasted, or mined (or any combination of these) for better decision making.

SAS Forecast Server software provides a large-scale automatic forecasting system. The software provides for the automatic selection of time series models for use in forecasting timestamped data. Given a timestamped data set, the software provides the following automatic forecasting process: It accumulates the timestamped data to form a fixed-interval time series, diagnoses the time series by using time series analysis techniques, creates a list of candidate model specifications that are based on the diagnostics, fits each candidate model specification to the time series, generates forecasts for each candidate fitted model, uses a model selection criterion to select the most appropriate model specification on the basis of either in-sample or holdout-sample forecast performance (fit statistic comparison), refits the selected model specification to the entire range of the time series, creates a forecast score from the selected fitted model, generates forecasts from the forecast score, and evaluates the forecast by using in-sample analysis. SAS Forecast Server also provides for out-of-sample forecast performance analysis.

There are many capabilities associated with SAS Forecast Server; this paper explores some of the lesser-known features.

## DISCOVER FORECAST SERVER CAPABILITIES

The following sections provide details about 10 underused features that you can use to improve your workflow.

### FEATURE 1: USE SAS TIME SERIES STUDIO TO BECOME FAMILIAR WITH YOUR DATA AND DEFINE A HIERARCHY

Many organizations collect large amounts of transactional and time series data, such as sales histories, inventory histories, customer transactions, insurance claim histories, and internet data. Analysts often need to structure the time series data into hierarchical time series at particular frequencies in order to enhance their understanding of the data and improve the accuracy of their analyses. In addition, when analysts have large amounts of data, they often need to segment their time series data to support different analyses on different subsets of their data. SAS Time Series Studio enables you to subset your data by using hierarchical queries, graphical queries, parametric queries, or manual selection.

SAS Time Series Studio enables you to do the following:

- Explore multiple time series simultaneously to better understand your data: You can quickly identify anomalies (such as outliers or missing values) and determine which time series does not look like the others. You can explore the effect of transformations of variables and the effect of different time series.
- Identify series that require specialized methods for analysis: You can identify short series (for example, new products or products that have a short lifecycle) and intermittent time series (for example, a series that contains a large number of zero values).
- Identify and group similar time series: You can quickly identify similar time series and group them to allow for hierarchical modeling.
- Export data source that then can be used by SAS® High-Performance Forecasting, SAS® Forecast Studio, SAS® Enterprise Miner™, and other products.

If you are working with new data or considering a new way of organizing your data into hierarchies, try the powerful SAS Time Series Studio application, which is part of SAS Forecast Server. Like SAS Forecast Studio, SAS Time Series Studio has a **New Project** window, which enables you select a data source that has timestamped observations. If you haven't yet identified an appropriate hierarchy for your project, you've come to the right place. Start with a tentative hierarchy by selecting BY variables, or click **Recommend** to select all character variables in the data source, as shown in Figures 1.1 and 1.2. Next, select a time ID variable that contains date or datetime values. If the automatically detected time interval of the observations is undefined or of higher frequency than you need, specify the desired time interval. SAS Time Series Studio will accumulate the data to this interval for you by using the selected formula (the default is the sum of the values). Assign the role of dependent variable to at least one input variable, and assign other roles if appropriate. Distribution plots of each of the assigned variables can then be viewed, as shown in Figures 1.3 and 1.4.

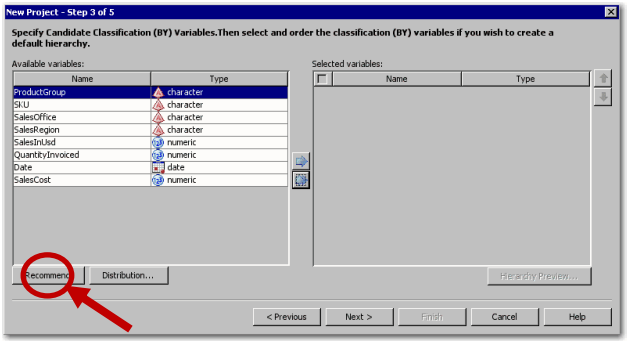



**Figure 1.1: New Project, Recommend Button**

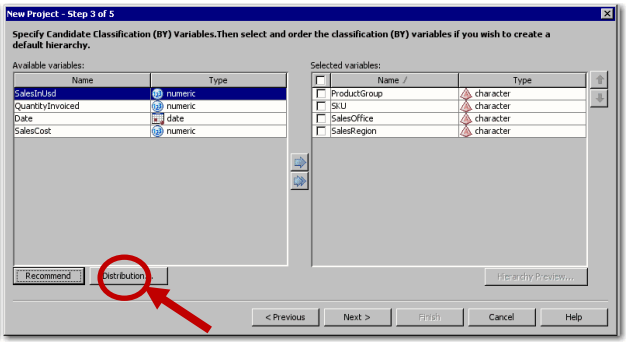



**Figure 1.2: Recommended Variables**

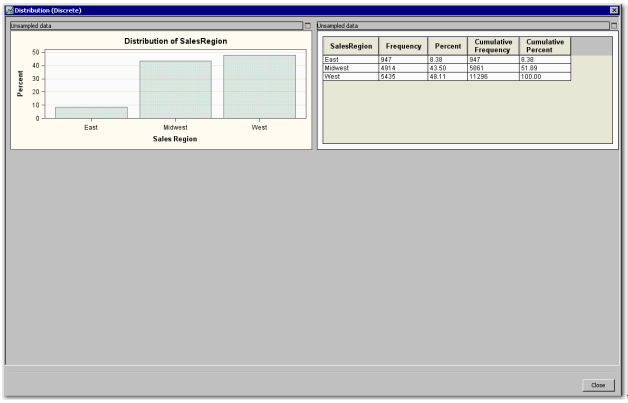



**Figure 1.3: Distribution (Discrete)**

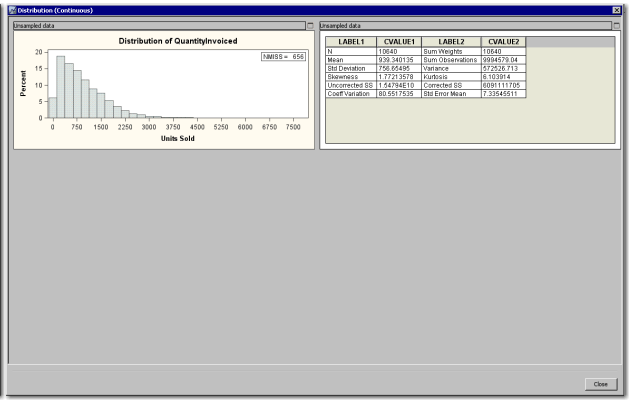



**Figure 1.4: Distribution (Continuous)**

SAS Time Series Studio provides three views of your project, as shown in Figure 1.5. The **Flow Manager** view shows a tree view of the steps you have taken in constructing the project. The **Selection View**

enables you to select elements of interest. The **Details View** shows graphical and tabular analyses of the selected elements. In the **Details View**, you can view a single series or all series that are contained in a selected node. From the **Display** list, select **Envelope** to see series distribution over time (Figure 1.6), select **Time Series** to see actual values over time (Figure 1.6), or select **Combined** to see both at once (Figure 1.8). The **Series Analysis** tab of the **Details View** provides standard graphs and tables for autocorrelation and seasonal decomposition analysis and enables you to apply transformations (Figure 1.9). The **Multi-Variable Time Series** tab of the **Details View** provides cross-series plots of scatter plots of more than one time series (Figure 1.10 and 1.11).

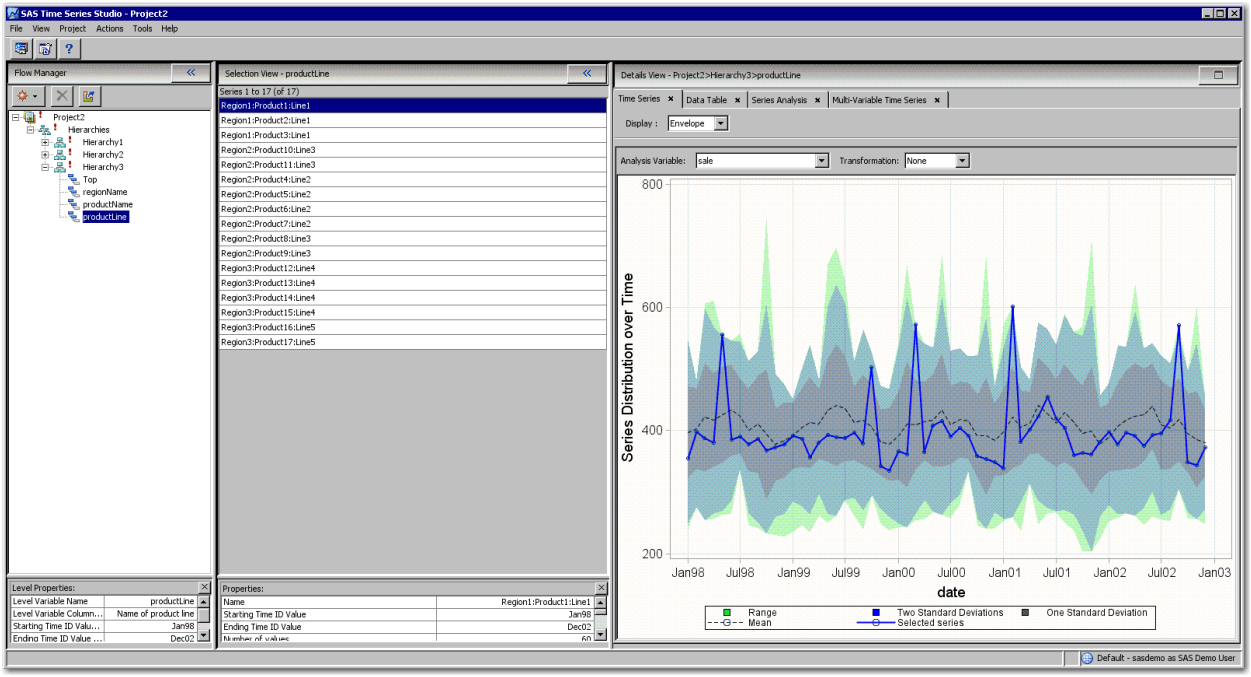



**Figure 1.5: Time Series Studio (Flow Manager, Selection View, and Details View)**

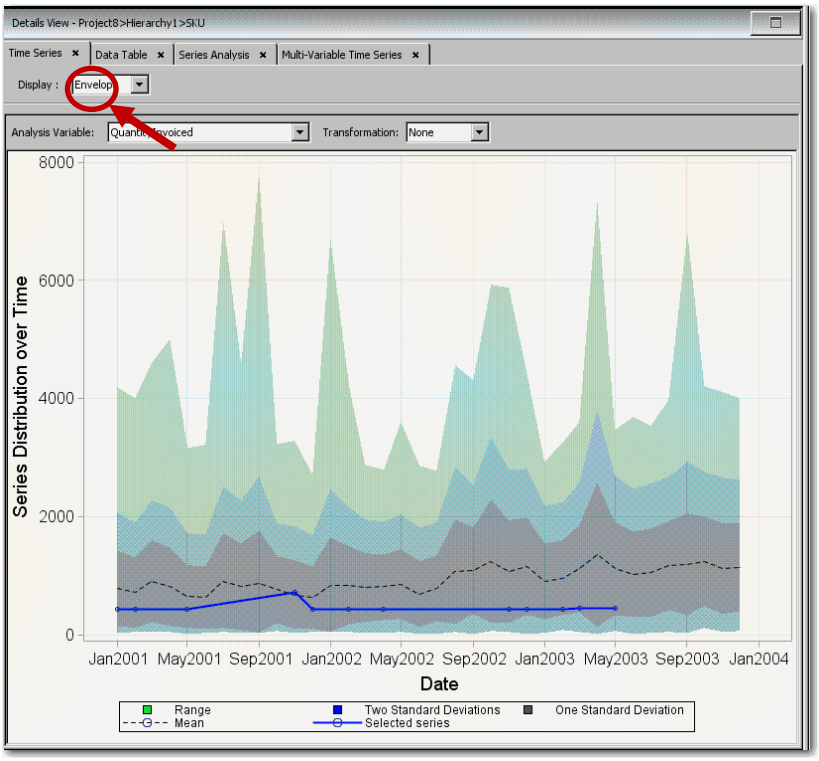



**Figure 1.6: Envelope Plots**

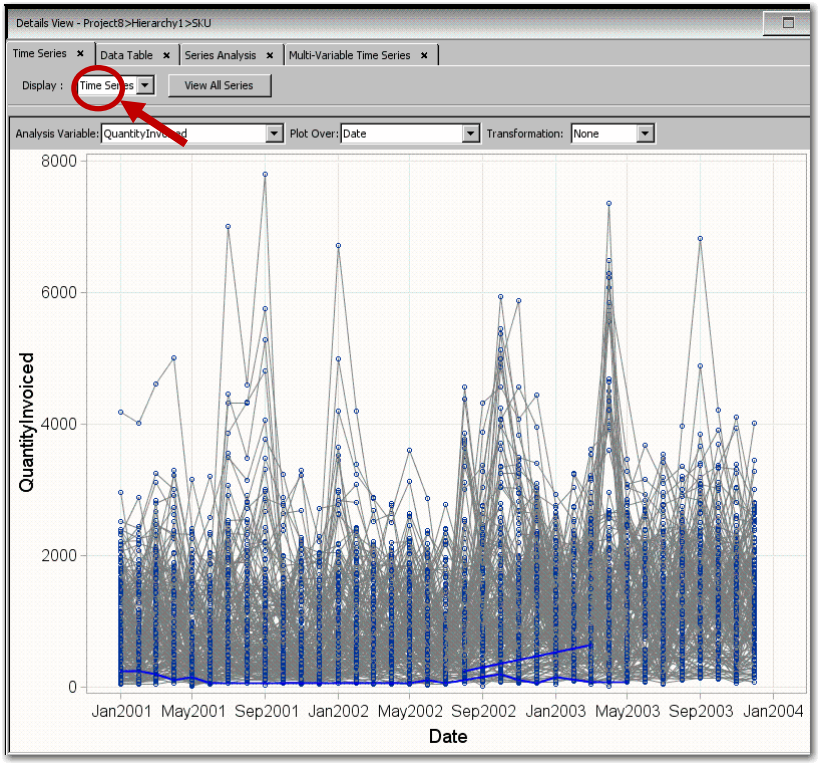



**Figure 1.7: Time Series Plots**

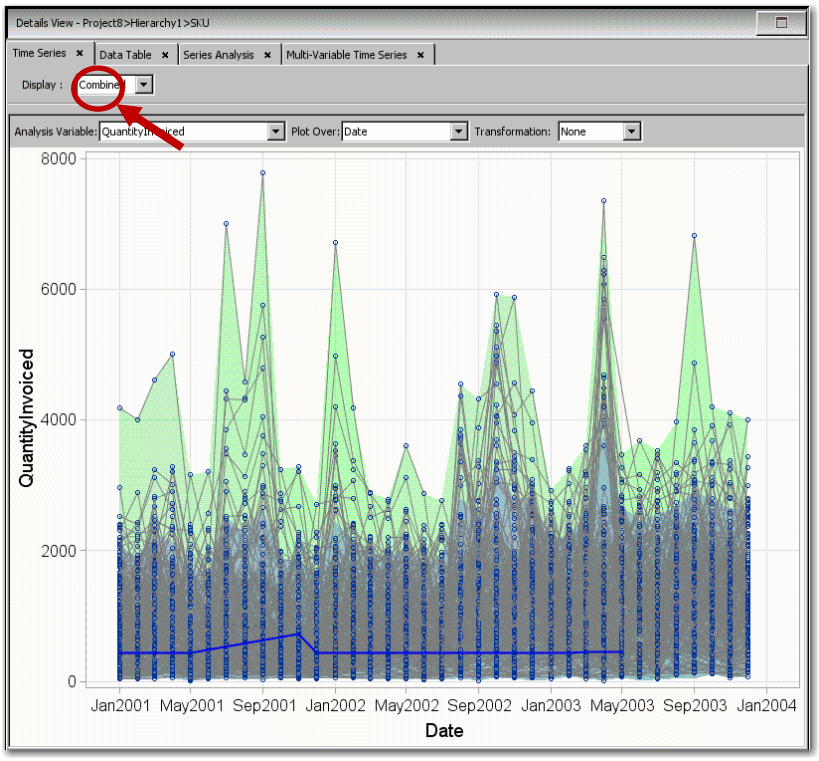



**Figure 1.8: Combined Plots**

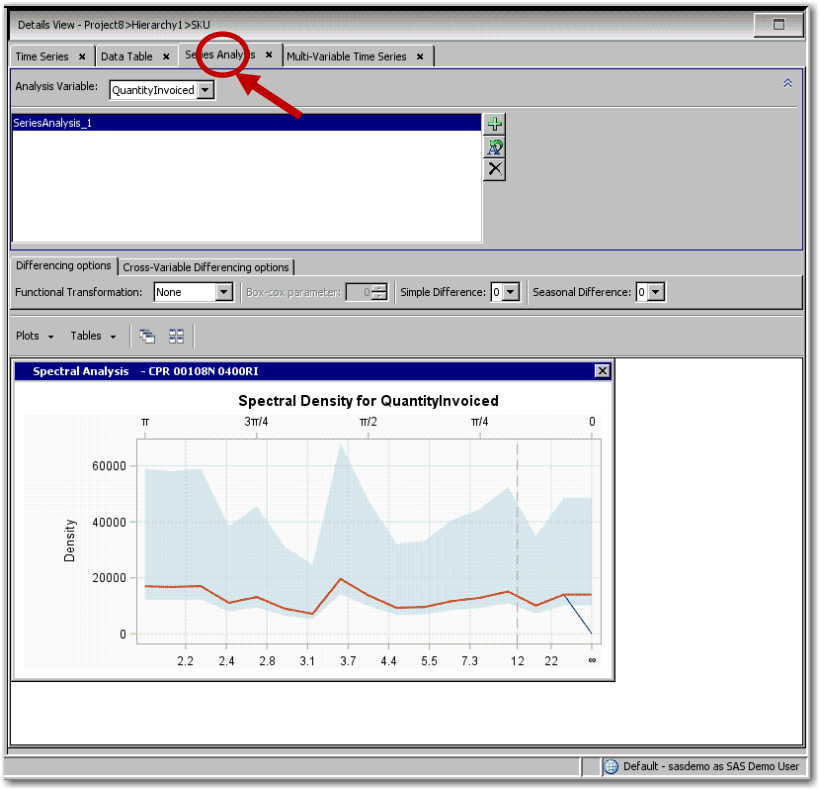



**Figure 1.9: Series Analysis Plots**

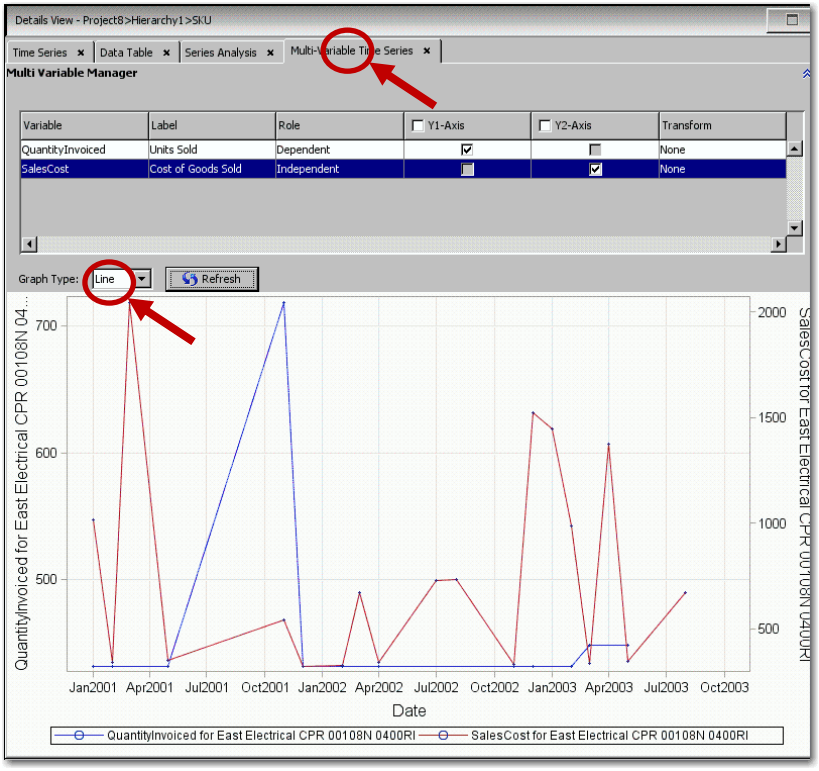



**Figure 1.10: Multi-Variable Series, Line Plots**

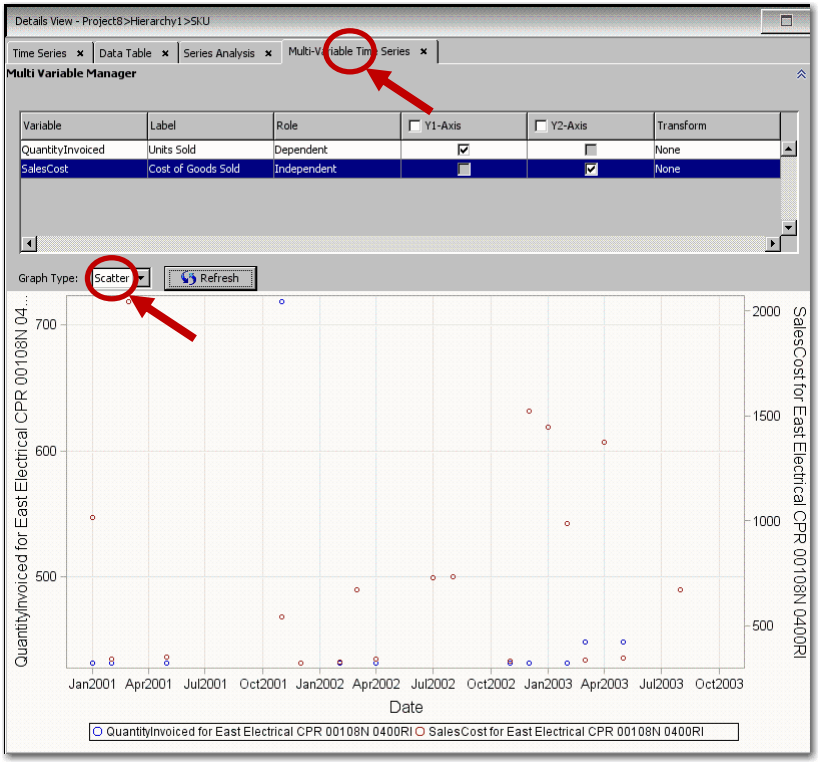



**Figure 1.11: Multi-Variable Series, Scatter Plots**

One unique feature of SAS Time Series Studio is its ability to structure the data in multiple ways within a project so that you can compare the results within a single view. You set this up by adding one or more

new hierarchies to the **Flow Manager**, selecting BY variables in a particular order, selecting a time interval, and then specifying aggregation and accumulation options. Once you have created the hierarchies you want to explore, the selection view enables you select nodes in any of your hierarchies. The results appear in the **Details View.**

Another unique feature is segmentation, which enables you to identify subsets of the data that behave differently and lend themselves to different forecasting strategies. You can use a graphical query (by drawing a bounding box around a section of a graph) or a parameter query (by entering bounding values on selected variable). Either way, the segments appear in the **Flow Manager**, where you can select them for viewing in the **Details View**. Selected segments can be exported for use in SAS Forecast Studio and other applications.

Once you are happy with your SAS Time Series Studio project, you can use the data to create a Forecast Studio project. You do this by selecting elements in the **Flow Manager** and right-clicking **Export** (which save the results as a SAS data set), and then selecting this data set as input to a new Forecast Studio project. The hierarchy and variable assignments you set up in SAS Time Series Studio will be in effect.

For more information about time series exploration, see *SAS Forecast Studio: User's Guide*.

## FEATURE 2: CATCH DATA PROBLEMS THROUGH WARNINGS ABOUT THE TIME ID VARIABLE

Statistical forecasting models assume that the data points in a series occur at evenly spaced points in time. SAS Forecast Server identifies these points in time by the values of the time ID variable that you select. When you select this variable in SAS Forecast Studio, the time values are analyzed to identify the time interval and checked for possible problems in the data. In many cases, SAS Forecast Studio is quite forgiving, but it gives you warnings in some cases so that you can check the data and correct problems if needed. The following warnings are common:

- **`Gaps were detected in the values of the Time ID variable.`** This warning is issued if the values of the time ID are not evenly spaced. To see more information about this warning, click **Diagnostic graphs** in the warning dialog box. The diagnostics include an offset histogram, which indicates the relative time distance between observations. Any bars that differ from zero indicate an abnormal time ID value. The X axis indicates the position in the data where the problem occurs.
- **`The Time ID variable has duplicate values.`** This warning is issued if the values of the time ID are not unique. To see more information about this warning, click **Diagnostic graphs** in the warning dialog box. In the diagnostic graphs, one or more bars in the Interval Count histogram will be greater than one.

In most cases, SAS Forecast Studio will produce forecasts even when the time ID is in bad shape. For example, if you use the Sashelp.Workers data and you mistakenly select the variable Electric as the time ID and the variable Date as the dependent series, you will get a forecast, albeit an odd-looking one. But the diagnostic histograms for the time ID will be off-kilter. They should all be flat, and in this case, none are. Figure 2.2 illustrates the **TIMEID Analysis Graphs** dialog box.

One case in which the application is not forgiving is when the time ID values are missing values. The message is "ERROR: Missing or invalid values found." There are no diagnostic graphs and you cannot proceed.

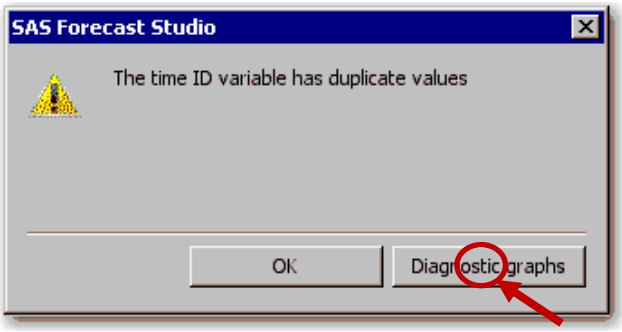



**Figure 2.1: Warning Dialog Box**

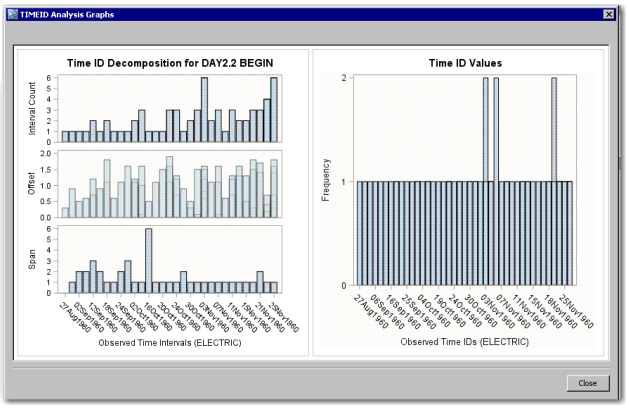



**Figure 2.2: TIMEID Analysis Graphs Dialog Box**

## FEATURE 3: LOOK UNDER THE COVERS BY USING THE SAS LOG

You can view the SAS code that was used to execute the project by selecting **Project ► SAS Code** from the main menu. The **SAS Code** dialog box appears as shown in Figure 3.1. You can save the SAS code to a file from this dialog box.

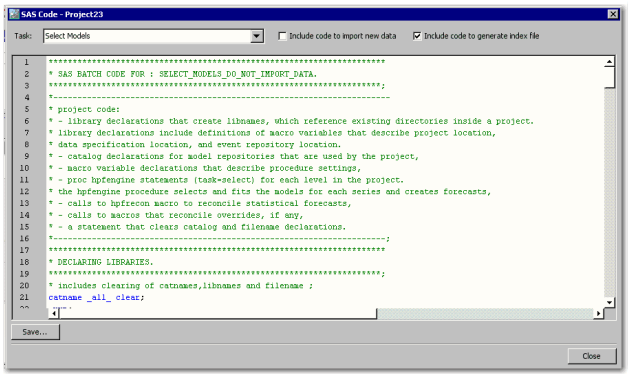



**Figure 3.1: SAS Code Dialog Box**

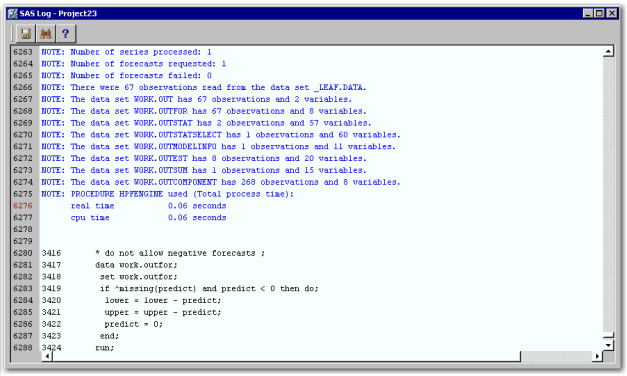



**Figure 3.2: SAS Log Dialog Box**

You can view the SAS log for an opened project by selecting **Tools ► SAS Log** from the main menu. The **SAS Log** dialog box contains information about the SAS execution, including the resources that were used, notes, and warnings, as illustrated in Figure 3.2.

If an error occurs during the SAS execution, the **[!]** icon usually appears on the status bar. Clicking this icon opens the log, which is searchable. For example, search for "`ERROR:`" to find error messages.

## FEATURE 4: USE ADJUSTMENT VARIABLES AND START-UP CODE TO FIX DATA ISSUES

Sometimes it is necessary to forecast a time series that is not expressed as desired in the input data. Adjusting the time series for known systematic variations or deterministic components enables you to more readily identify and model the underlying stochastic (unknown) time series process. Examples of systematic adjustments are currency-unit conversions, exchange rates, trading days, and other known systematic variations. Examples of deterministic adjustments are advanced bookings and reservations, contractual agreements, and other known contributions or deterministic components.

By providing suitable adjustment variables in the same data set, you can transform the time series values without changing the original series. In the **New Project** dialog box, assign your dependent variable and then click **Adjustments,** as illustrated in Figure 4.1. Alternatively, for an existing project, select **Project ► Hierarchy and Variable Settings** from the main menu.

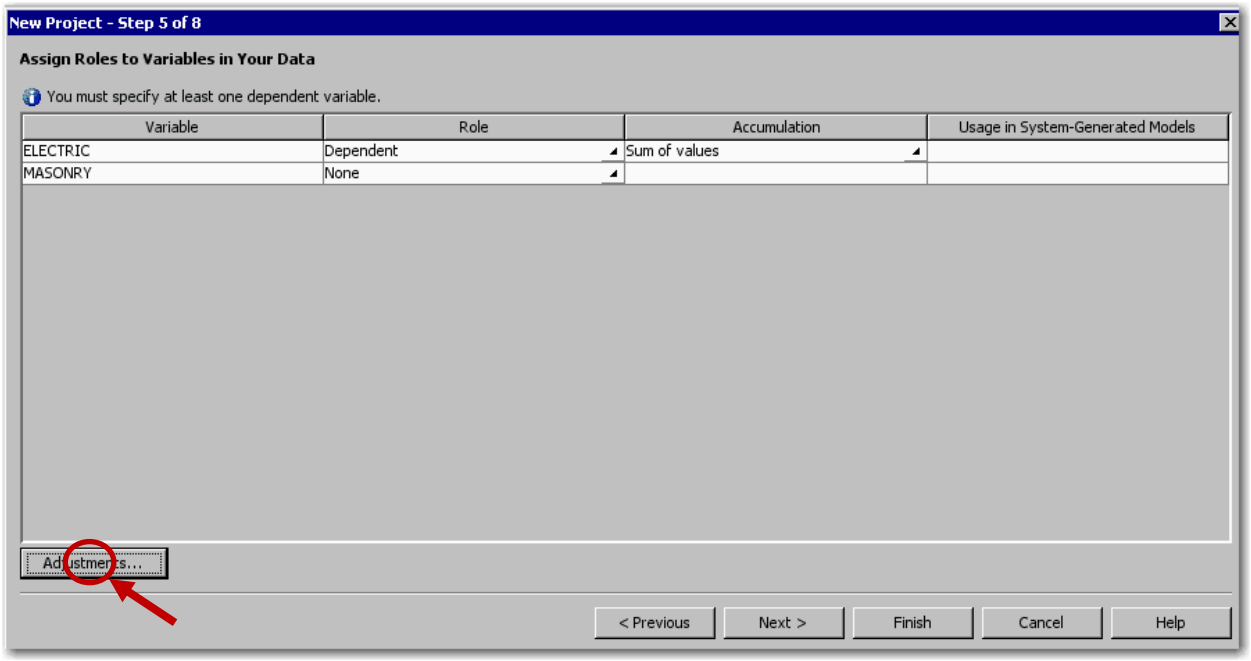



**Figure 4.1: Assign Roles to Variables in Your Data: Adjustments Button**

After you click **Adjustments**, the **Adjustments** dialog box appears, as shown in Figure 4.2; it lists the existing adjustments that are being applied to the project. Click **New** to open the **Adjustment Properties** dialog box, which is shown in Figure 4.3. In the **Adjustment Properties** dialog box, you can specify how to accumulate the adjustment variable values, the pre-operation (add, subtract, multiply, or divide), and the post-operation (add, subtract, multiply, or divide). Typically, the post-operation is the inverse of the pre-operation. So, if the pre-operation is add, the post-operation is subtract.

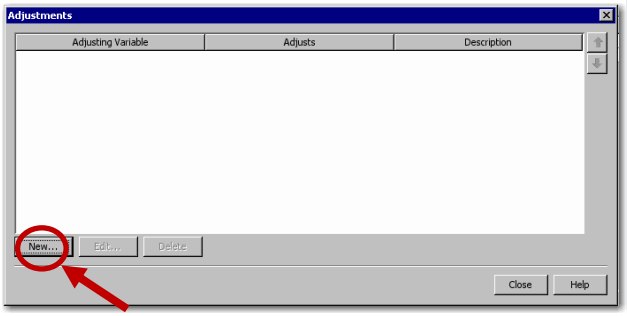



**Figure 4.2: Adjustments Dialog Box**

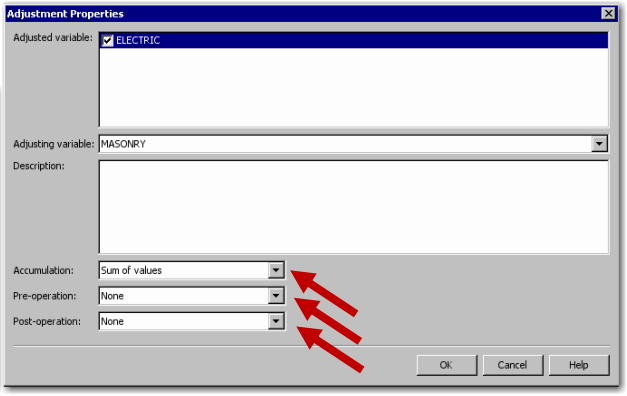
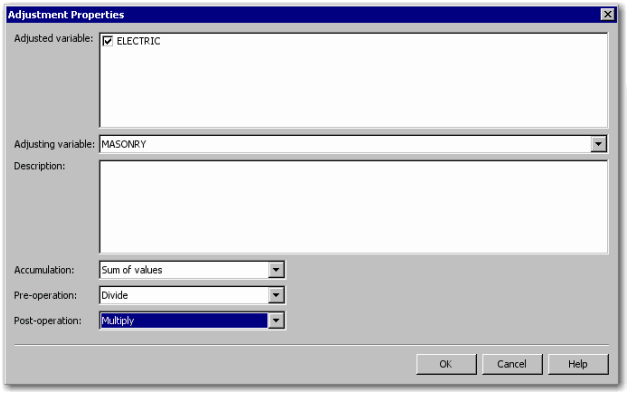



**Figure 4.3: Adjustment Properties Dialog Box**



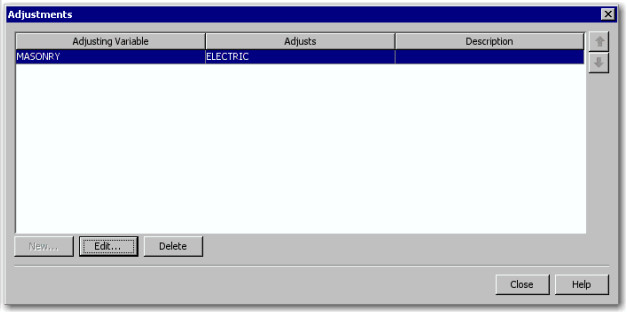




**Figure 4.4: Adjustment Properties Dialog Box** **Figure 4.5: Adjustments Dialog Box**

You can apply multiple adjustments. For example, you can select a variable whose values contain adjustments for estimated loss of data that occurred on certain dates by selecting **Add** from the **Pre-operation** list. The actual series will be modified according to the first adjustment added, and the results will be modified according to the second adjustment.

You can use pre-adjustment variables to adjust the dependent series prior to model parameter estimation, evaluation, and forecasting. After the predictions of the adjusted dependent series are obtained from the forecasting mode, you can use the post-adjustment variables to adjust these forecasts to obtain predictions that more closely match the original dependent series.

A typical example is adjusting a monetary-valued series for exchange rates. Suppose that the dependent series is product sales in Europe, expressed in Euros, and you want to convert currency to US dollars. Select the exchange rate as the adjustment variable and select **Multiply** from the **Pre-operation** list. The forecast graph will show the original data values, but the forecast will use the values that have been adjusted for the exchange rate. If you want to convert these values back to the original metric for display, select **Divide** from the **Post-operation** list.

As another example, suppose that your dependent series has missing data in some cases and you want to use values of a proxy series in these cases. Select the proxy series as the adjustment variable, and select **Max** from the **Pre-operation** list. Any nonmissing value of the proxy series will be greater than a missing value. The forecast graph will show the original actual values, with missing values, but the forecast will use the adjusted series with no missing values.

## FEATURE 5: DEFINE RECURRING EVENTS THAT INFLUENCE THE TIME SERIES

An event repository stores information about calendar events with a brief description of each event. Calendar events can be represented by indicator variables that could be stored in the time series data. However, because the influential calendar events can vary from series to series, there might be too many to store efficiently and many calendar events will be redundant, making updates difficult. Therefore, to allow the reuse and update of the calendar events, it is better to store a brief description of the calendar event, reproduce the indicator variable in the computer's memory when needed, and store the calendar events independently of the time series data.

To create a new event:

1. You can create new events in the **New Project – Step 7 of 8** dialog box (as shown in Figure 5.1), or you can create new events for an existing project by selecting **Project ► Event Repository** from the main menu. The **Event Repository** dialog box appears, as shown in Figure 5.2.

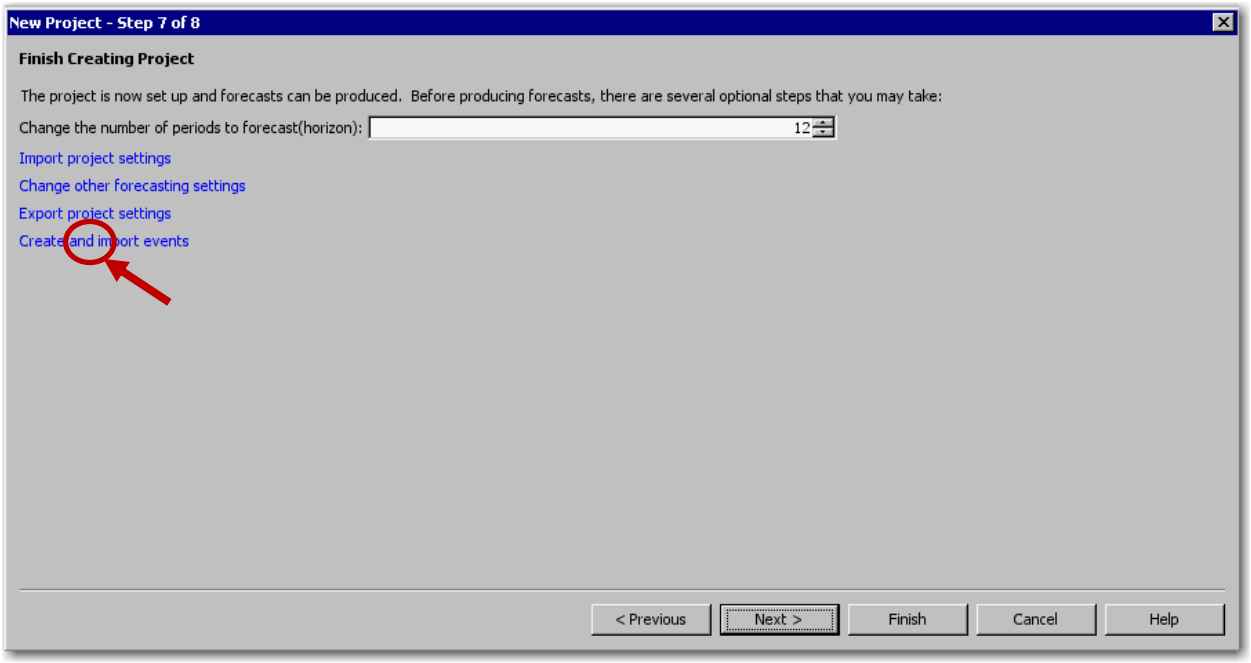



**Figure 5.1: New Project Window**

2. Click **New**. The **Event Properties** dialog box appears, as shown in Figure 5.3.
3. In the **Name** box, specify a valid SAS name for the event. You can also specify a description for the event in the **Description** box.
4. Click **Change** to specify an event type, and select from **Pulse**, **Level Shift**, **Ramp**, or **Temporary Change**.
5. Click **Add** to add an occurrence. The **Select Occurrences** dialog box appears, as shown in Figure 5.4. Select the time period for the occurrence and click **OK**. The **Occurrence** box in the **Event Properties** dialog box now displays a list of the occurrences for the event, as illustrated in Figure 5.4.
6. To specify the options for the occurrence, click **Edit** next to the **Options** box. The **Event Options** dialog box appears (not shown). In this dialog box, you can specify the duration of the occurrences and the time of the occurrences.
7. To specify recurrence, click **Edit** next to the **Recurrence** box. The **Event Recurrence** dialog box appears (not shown). In this dialog box, you can specify how frequently the event recurs and how long the recurrence will last.

   **Note:** Before you can specify a recurrence, you must specify an occurrence. However, you cannot specify a recurrence if you have selected two or more occurrences or if you have selected a holiday for the event occurrence.
8. Click **OK** to save the event. The new event appears in the **Event Repository** dialog box.

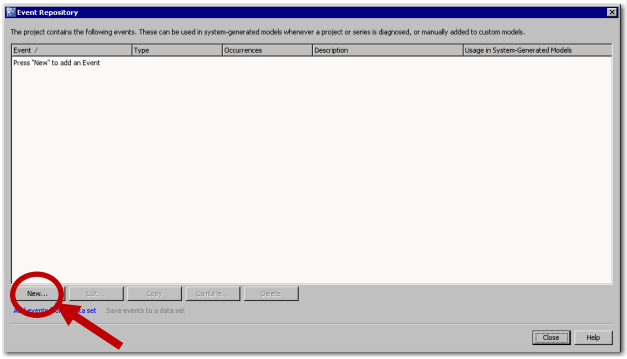



**Figure 5.2: Event Repository Dialog Box**

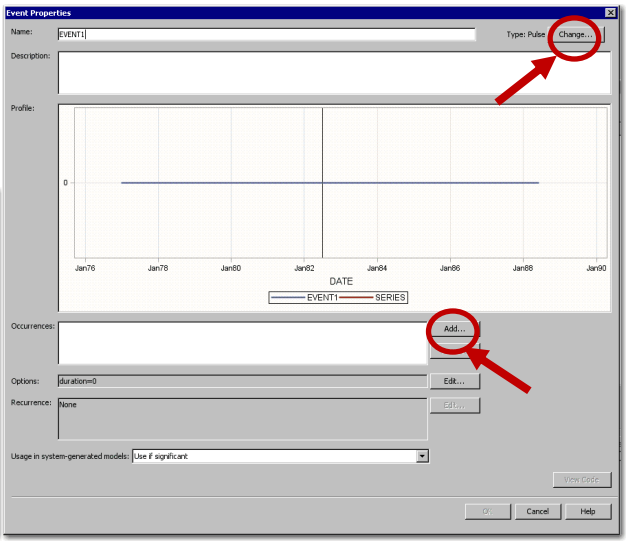



**Figure 5.3: Event Properties Dialog Box**

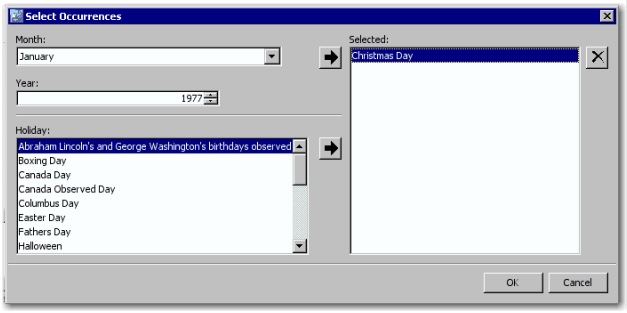



**Figure 5.4: Select Occurrences Dialog Box**

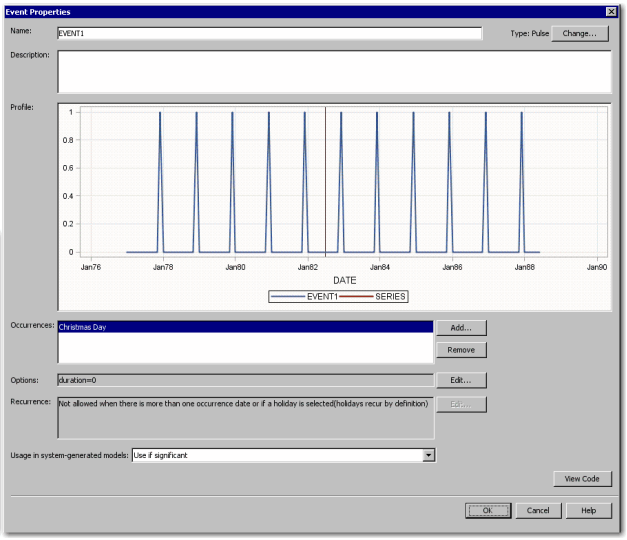



**Figure 5.5: Event Properties Dialog Box**

## FEATURE 6: ADD MODELS BY IMPORTING FROM AN EXTERNAL LIST

SAS Forecast Studio supports the following types of models:

- *Default models* are included in SAS Forecast Studio. You cannot edit or delete these models. However, you can copy a default model and customize it to meet your needs.
- *System-generated models* are automatically generated by the diagnostics in SAS Forecast Studio. You cannot edit these models, but you can delete them and you can copy a system-generated model and customize it to meet your needs.
- *Custom models* are created in SAS Forecast Studio. When you create a model for a particular time series, that model is automatically added to the project model repository. After the model is added, you can apply it to other series in the project. Because these custom models are user-defined, you can add, edit, copy, or delete them.

- *Models from an external list* originate from the external list that is specified in the **Model Generation** panel of the **Forecasting Setting** dialog box. In the model repository, models from an external list are listed as having type **Custom** (**Read-Only**). You cannot edit or delete these models. However, you can copy a model from an external list and customize it to meet your needs.

To select models to fit to each series from an external model selection list:

1. In the **New Project - Step 7 of 8** dialog box, click **Change other forecast settings**, as shown in Figure 6.1. For a previously created project, you can select **Project ► Forecasting Settings** from the main menu. The **Forecasting Settings** dialog box appears, as shown in Figure 6.2.
2. Select **Model Generation**, as highlighted in Figure 6.2.
3. Select the **Models from an external list** check box, and click **Browse**. The **Select Model Selection List** dialog box appears (not shown here), and you can select the model selection list that you want to use.
4. Click **OK**.

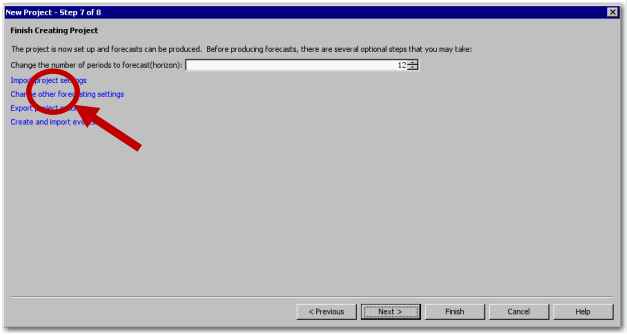



**Figure 6.1: New Project – Step 7 of 8**

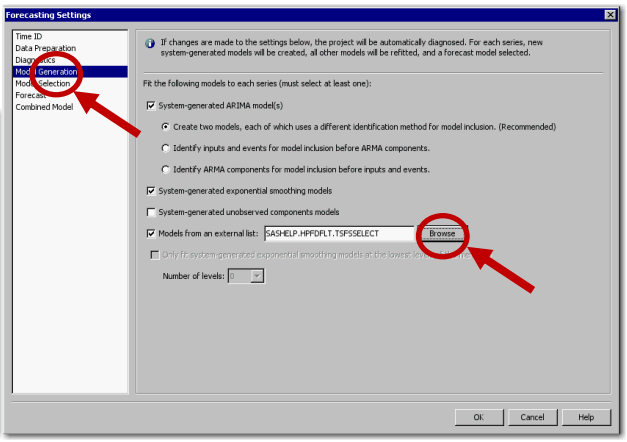



**Figure 6.2: Forecasting Settings**

## FEATURE 7: USE ROLLING SIMULATIONS TO EVALUATE FORECAST ACCURACY OVER THE NUMBER OF PERIODS YOU NEED TO FORECAST

For a particular time series data set, you can use automatic time series modeling software to select an appropriate time series model. You can use various statistics to judge how well each candidate model fits the data (in-sample). Likewise, you can use various statistics to select an appropriate model from a list of candidate models (in-sample or out-of-sample or both). Finally, you can use rolling simulations to evaluate ex-ante forecast performance over several forecast origins. Leonard et al. (2014) demonstrate how you can use SAS® Forecast Server Procedures and SAS Forecast Studio software to perform the statistical analyses that are related to rolling simulations.

On the **Model View** tab, click the rolling simulation button (circled in red in Figure 7.1) or select **Series ► Rolling Simulation** from the main menu.

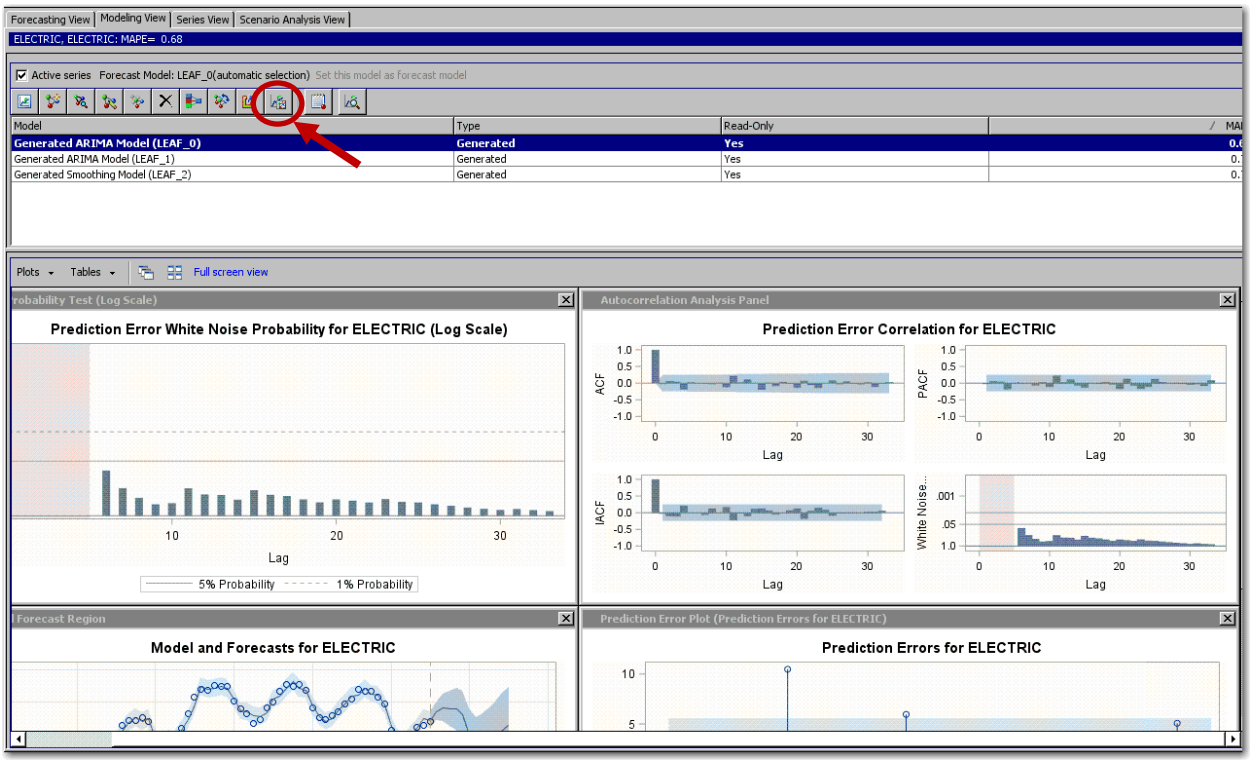



**Figure 7.1: Modeling View: Rolling Simulations Button**

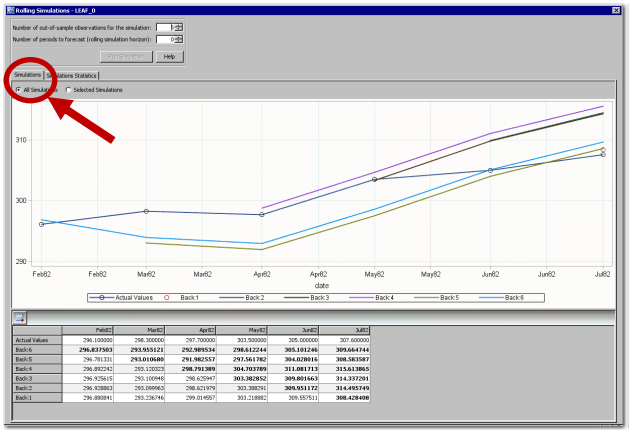


**Figure 7.2: Rolling Simulation: Prediction Plots**

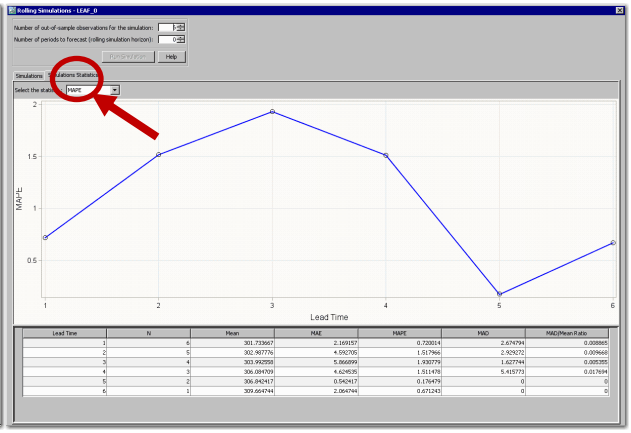


**Figure 7.3: Rolling Simulation: Statistics of Fit**

Rolling simulations (also called rolling horizon) mimic how the forecast model performs over time. Figure 7.2 illustrates a rolling horizon of six periods. The actual time series values are the black circles, which are connected by the black lines. The colored lines represent the six predictions, each beginning at a different origin. Figure 7.3 illustrates a rolling statistic of fit for six periods.

## FEATURE 8: EVALUATE THE EFFECTS OF INDEPENDENT VARIABLES

Organizations often want to make decisions on the basis of time series forecasts that depend on future, controllable causal factors. Examples of future causal factors are pricing and advertising expenditures for products and services. To help your organization make better decisions, you can vary the future values of the controllable causal factors to help determine the best decision (a what-if analysis). For example, if you are forecasting the profit of your company and material cost is an underlying factor, then you could use scenario analysis to determine how the forecasted profit would change if the material cost increased by 10%. Leonard (2000) thoroughly demonstrates how you can use SAS/ETS® software for this purpose and

describes the detailed statistical analyses that are related to scenario analysis. To perform scenario analyses, click the **Scenario Analysis View** tab on the **Modeling View** tab**,** as shown in Figure 8.1.

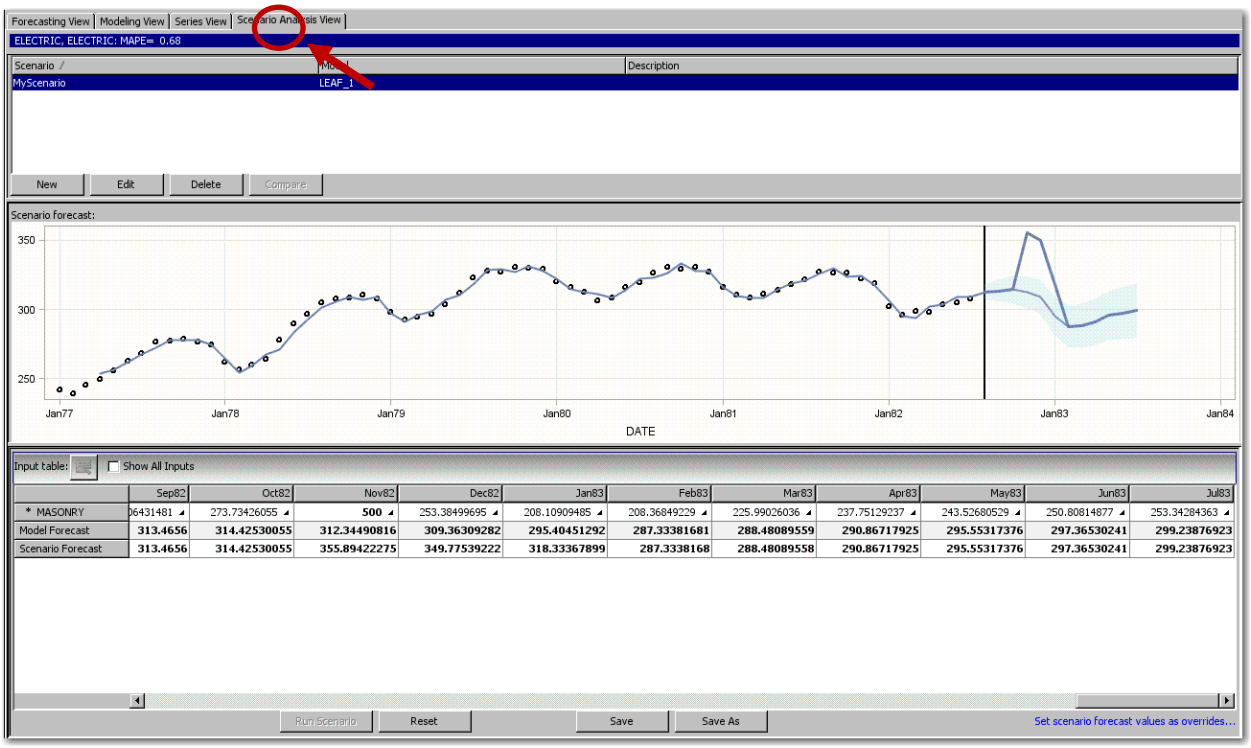



Figure 8.1: Scenario Analysis View

## FEATURE 9: GAIN INSIGHT INTO RESULTS BY COMPARING MODELS

If you have created several models for a time series, then you might want to compare the statistics of fit for the models that were fitted to that series. This comparison can help you determine which model you want to use to generate forecasts.

From the **Modeling View**, click the compare models button (circled in red in Figure 9.1) or select **Series ► Compare Model**s from the main menu. The **Compare Models** dialog box appears.

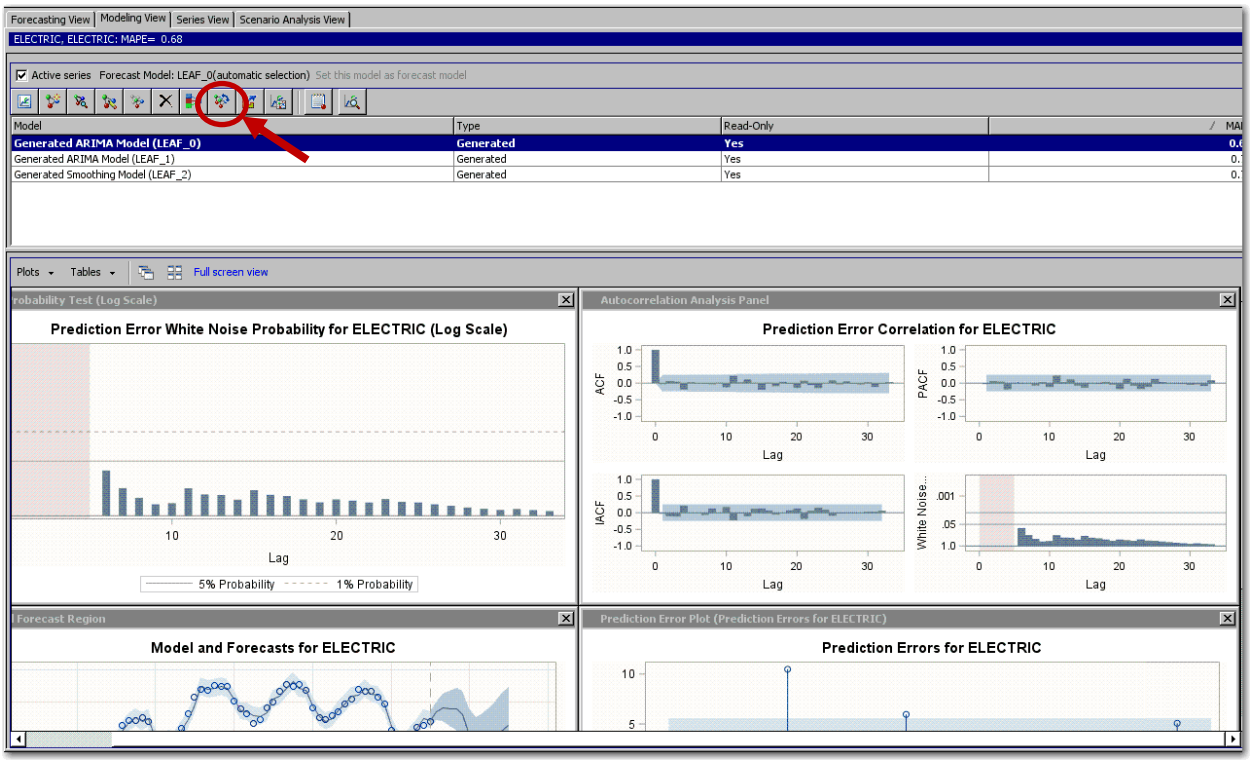



**Figure 9.1: Modeling View: Compare Models Button**

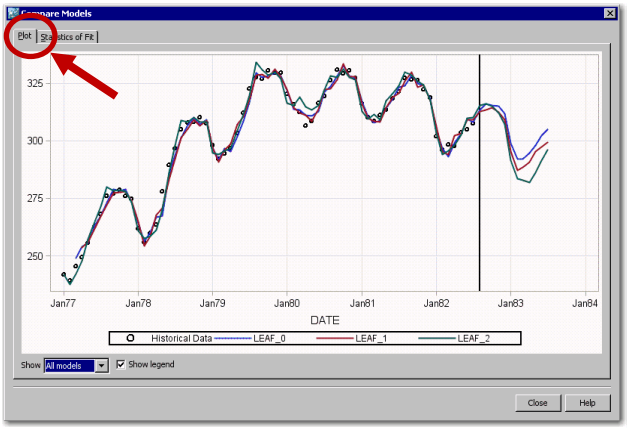



**Figure 9.2: Compare Models: Prediction Plots**

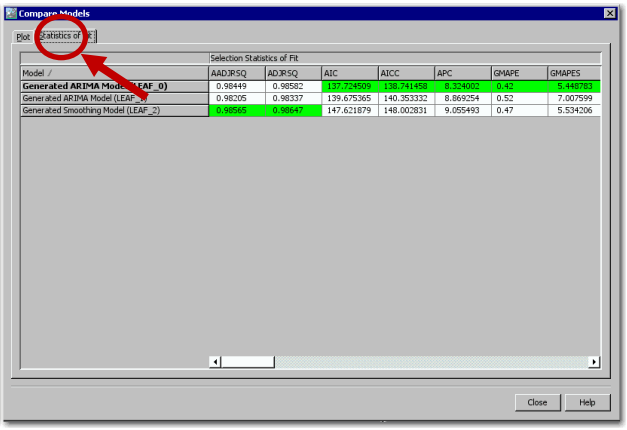



**Figure 9.3: Compare Models: Statistics of Fit**

Figure 9.2 illustrates the model comparison plot for three models that are associated with a single time series. The actual time series values are represented by the black circles. The colored lines represent the model predictions from three different time series models. For this time series, the models are in close agreement. This is a good situation.

To view the model comparison statistics for the three time series models, click the **Statistics of Fit** tab, as shown in Figure 9.3. The champion model for each statistic is highlighted in green. For this time series, the models are in close agreement. This is a good situation.

## FEATURE 10: RUN THE CODE IN BATCH

Forecasts can be generated either interactively or in batch. When you interact with SAS Forecast Studio, it generates SAS code for execution. This code is stored in the system directory of your Forecast Studio

projects. These SAS code files implement the forecasting process steps that are associated with SAS Forecast Server. Table 1 lists the files and describes their contents.

| File Names | Contains SAS Code to: |
|---|---|
| CREATE_PROJECT_DO_NOT_IMPORT_DATA and CREATE_PROJECT_IMPORT_DATA | Re-create the project. This code includes all the data preparation and forecasting steps. |
| DIAGNOSE_DESTRUCTIVE_DO_NOT_IMPORT_DATA and DIAGNOSE_DESTRUCTIVE_IMPORT_DATA | Re-diagnose the project. This code does not include the data preparation steps. |
| SELECT_MODELS_DO_NOT_IMPORT_DATA and SELECT_MODELS_IMPORT_DATA | Perform model selection, estimate the parameters of the selected model, and produce forecasts. This code does not include the data preparation, diagnose, fit, and forecasting steps. |
| FIT_MODELS_DO_NOT_IMPORT_DATA and FIT_MODELS_IMPORT_DATA | Estimate parameters by using the model that you specified and then create a forecast. No model selection is performed. This code does not include the data preparation and diagnose steps. |
| FORECAST_MODELS_DO_NOT_IMPORT_DATA and FORECAST_MODELS_IMPORT_DATA | Re-forecast the model and parameters. When refitting the model parameters, SAS Forecast Studio uses the estimate of the previous parameter as a starting point for re-estimation. This code does not include the data preparation, diagnose, and fit model steps. |
| RECONCILE_FORECASTS_AND_OVERRIDES_DO_ NOT_IMPORT_DATA | Reconcile model forecasts and overrides. This code does not include the data preparation, diagnose, select, fit, and forecast steps. This code can be run only if those steps have been executed at least once. |
| RECONCILE_FORECASTS_DO_NOT_IMPORT_DATA | Reconcile model forecasts. This code does not include the data preparation, diagnose, select, fit, forecast, and override reconciliation steps. This code can be run only if those steps have been executed at least once. |
| RECONCILE_OVERRIDES_DO_NOT_IMPORT_DATA | Reconcile overrides. This code does not include the data preparation, diagnose, select, fit, forecast, and reconciliation of the forecast steps. This code can be run only if those steps have been executed at least once. |

**Table 1: Batch Execution Files**

## CONCLUSION

SAS Forecast Server is one of the most feature-rich forecasting products on the market. This paper described 10 underused features that can improve your workflow.

## REFERENCES


Leonard, Michael. 2000. “Promotional Analysis and Forecasting for Demand Planning: A Practical Time Series Approach.” Presented at the 2000 International Symposium of Forecasting. Lisbon, Portugal.

Available https://support.sas.com/rnd/app/ets/papers/PromotionalAnalysis.pdf

Leonard, Michael. 2002. “Large-Scale Automatic Forecasting: Millions of Forecasts.” Presented at the 2002 International Symposium of Forecasting. Dublin, Ireland.

Leonard, Michael. 2004. “Large-Scale Automatic Forecasting with Calendar Events and Inputs.” Presented at the 2004 International Symposium of Forecasting. Sydney, Australia.

Leonard, Michael, Ashwini Dixit, and Udo Sglavo. 2014. “Ex-Ante Forecast Model Performance with Rolling Simulations.” *Proceedings of the SAS Global Forum 2014 Conference*. Cary, NC: SAS Institute Inc.

Available https://support.sas.com/resources/papers/proceedings14/SAS213-2014.pdf


## ACKNOWLDGMENTS


The authors would like to thank Anne Baxter from SAS for her editing.


## CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:


Michael Leonard
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
919-531-6967
Michael.Leonard@sas.com

Evan Anderson
evanlewisanderson@gmail.com

Paper SAS 3471-2019

# Using SAS® Forecast Server and the SASEFRED Engine to Enhance Your Forecast

Catherine LaChapelle, SAS Institute

## ABSTRACT

The SASEFRED interface engine is the best-kept secret in SAS/ETS® software. It dramatically reduces the amount of time and effort required to include economic indicator variables in your time series analysis. Using the SASEFRED engine in SAS® Enterprise Guide®, I can directly query the economic database of the Federal Reserve Bank of St. Louis. This public database contains over 529,000 economic time series aggregated from 86 sources. In this paper, I forecast wine demand and enrich my predictions via the inclusion of economic variables such as "Retail Sales: Beer, Wine, and Liquor Stores" and "Producer Price Index by Industry: Beer, Wine, and Liquor Stores." Although I use a retail example, this technology is relevant to all industries. The diversity of economic variables provided in this database ensures that it is useful to virtually every time series analysis and industry. This specific example leverages SAS Enterprise Guide and SAS® Forecast Server as interfaces. However, this functionality works on SAS® 9.4 as well as on SAS® Viya® technology.

## INTRODUCTION

The SASEFRED interface engine is a powerful tool that allows you to seamlessly integrate external economic variables into your forecasting project. In this paper I introduce the engine, discuss the FRED data source, and show an example of how to pull all this data together in a SAS project. By including external macroeconomic variables, I can enhance the accuracy of my forecast and answer additional questions about my models and data.

## WHY ENRICH TIME SERIES DATA?

As a forecaster, there are many factors I must consider when I am trying to build an accurate model to predict future values of my dependent variable. My goal is to accurately identify the signal, while ignoring the noise in the data. Although the signal of my data might be represented strongly by my dependent variable, in most cases there are independent variables which also add relevant information. For example, if I am trying to predict product sales in a given month, it is likely that the promotion I am running on that product is relevant. By including these independent variables in my time series model, I can create more accurate forecasts.

Another benefit of including independent variables is the ability to perform scenario analysis. Scenario analysis allows me to test the "what if" of a situation. For example, if I am predicting wine sales and I have a promotion variable, I can test different scenarios. Using the previous example, I could examine what would happen if I discounted a particular bottle of wine by 20% this month instead of 10%. I might want to look at whether my sales increase proportionally and what is the impact on my profit. This example uses a business variable I probably already have contained in my database. If we expand from this type of scenario, we can look at macroeconomic factors at play. In this paper's example, I include

several wine-specific variables contained within the FRED database. These variables give me the ability to look at the larger context surrounding my business problem, not just the ones contained within my business.

Typically, when I build a model as a statistician, I am wary of adding too many independent variables for fear of introducing too much extraneous noise to my model. One of the great benefits of using SAS® Forecast Server and SAS® Visual Forecasting as forecasting tools is the automated variable selection feature within the software. In both products, when I am creating a forecasting project I am prompted to assign each independent variable to a classification for use. The option **Use if Significant** provides a lot of value and automation for the forecaster. This option automatically tests each independent variable for significance against each time series evaluated in the project. For example, if you have a hierarchical forecasting project with 1,000 individual series, SAS Forecast Server and SAS Visual Forecasting will test each of these series individually for significance with each independent variable. This automated feature selection makes it more practical to pull in all of the possible independent variables, such as economic information from FRED, and test them at scale.

## HOW SASEFRED WORKS

The SASEFRED interface engine allows you to directly query the Federal Reserve Economic Database of St. Louis. The Federal Reserve Bank of St. Louis is in the Eighth District of the Federal Reserve System in the United States. The charter of the Research Division of the Bank is to advise the Bank president on relevant factors that influence economic policy. These areas of research include money and banking, macroeconomics, and international and regional economics. Currently there are over 529,000 individual time series published by FRED at varying levels of time series aggregation. These time series are aggregated from 87 distinct data sources. Due to the breadth of time series in this database, there is relevant economic information for virtually every industry.

This database is published and maintained by the United States government and thus is free for anyone to access. It does not require a paid subscription to access the data, but requires only the generation of a free API key for the query. More information about the creation of an API key can be found in the SASEFRED documentation on the FRED website.

## CODE EXAMPLE- WINE SALES FORECASTING

In this paper I apply the SASEFRED engine to capture wine-related macroeconomic variables and enhance my existing wine sales forecasting data set. I use SAS Enterprise Guide as an interface to run the program, prepare my data, and then join my tables together into a final data set for forecasting. Figure 1 depicts an overview of the process flow in SAS Enterprise Guide.

Figure 1 is an overview of the data manipulation process in SAS® Enterprise Guide.

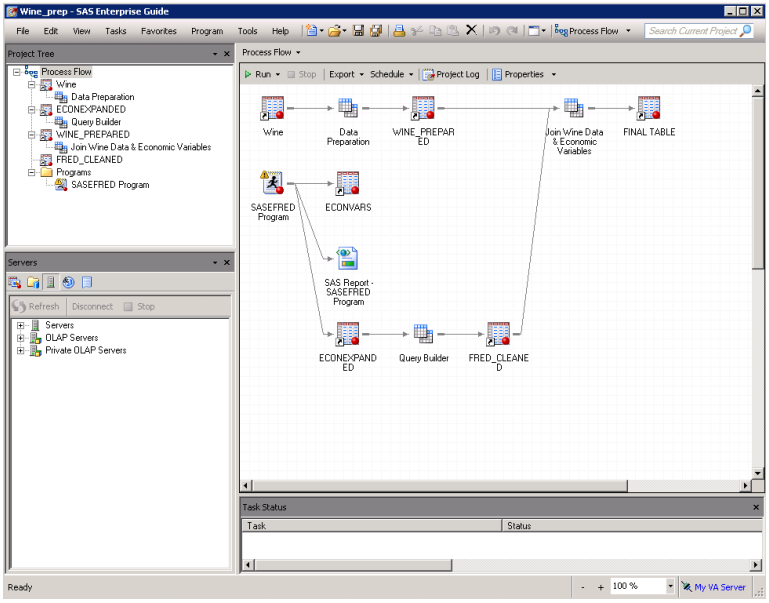



**Figure 1. SAS Enterprise Guide Project Overview**

The SASEFRED engine runs in a SAS LIBNAME statement. The sample code that I use for this example is included below. It is easy to copy and paste this code into your own program. You can then simply change out the relevant fields such as your file directory where you would like to save the generated table and your own individual API key in place of the x's I have here as a placeholder.

To determine which time series I would like to request for my analysis, I go to the FRED website and search for a relevant term for the data I am trying to model. Each series is tagged, which makes it easier to narrow my search. I can filter the results by concept, geography, geography type, frequency, source, release, and seasonal adjustment.

In this example, I searched for "wine," and chose three time series that seemed like they could add relevant additional information to my model. Because the series names are a string of letters and numbers and are often difficult to decipher, I recommend adding a key in a commented section of your code. Every time I add a series to my code, I add the description of the series to the comments so that I can easily reference it later.

```
title 'Retrieve Economic Indicator Variables';
libname _all_ clear;
libname fred sasefred "D:\FRED"
   OUTXML=fredex01
   AUTOMAP=replace
   MAPREF=MyMap
   XMLMAP="D:\FRED\fredex01.map"
   APIKEY='xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'
   IDLIST='PCU31213031213009, MRTSSM4453USN, PCU4453104453101'
   START='2014-02-23'
   END='2016-10-23'
   FREQ='m'
   OUTPUT=1
```

```
   AGG='avg'
   FORMAT=xml;

data econvars;
   set fred.fredex01 ;
run;
   /* Retail Sales: Beer, Wine, and Liquor Stores (MRTSSM4453USN)
   Producer Price Index by Industry: Wineries: Wines, Dessert,
   Effervescent, and Wine Coolers(PCU31213031213009)
   Producer Price Index by Industry: Beer, Wine, and Liquor Stores:
   Retailing of Beer, Wine, and Liquor(PCU445310445310l1) */
```

Figure 2 is a sample of the first 30 observations from the SASEFRED code example.

| | date | realtime_start | realtime_end | PCU31213031213009 | MRTSSM4453USN | PCU445310445310l1 |
|---|---|---|---|---|---|---|
| 1 | 2014-01-01 | 2019-03-21 | 2019-03-21 | 100.7 | 3381 | 122.9 |
| 2 | 2014-02-01 | 2019-03-21 | 2019-03-21 | 100.7 | 3310 | 124.7 |
| 3 | 2014-03-01 | 2019-03-21 | 2019-03-21 | 101.5 | 3652 | 121.5 |
| 4 | 2014-04-01 | 2019-03-21 | 2019-03-21 | 101.5 | 3702 | 122.2 |
| 5 | 2014-05-01 | 2019-03-21 | 2019-03-21 | 101.5 | 4165 | 122.1 |
| 6 | 2014-06-01 | 2019-03-21 | 2019-03-21 | 101.5 | 4036 | 124.3 |
| 7 | 2014-07-01 | 2019-03-21 | 2019-03-21 | 101.5 | 4242 | 125.2 |
| 8 | 2014-08-01 | 2019-03-21 | 2019-03-21 | 101.5 | 4219 | 125.9 |
| 9 | 2014-09-01 | 2019-03-21 | 2019-03-21 | 101.5 | 3814 | 124.1 |
| 10 | 2014-10-01 | 2019-03-21 | 2019-03-21 | 101.6 | 4090 | 127.3 |
| 11 | 2014-11-01 | 2019-03-21 | 2019-03-21 | 101.6 | 4103 | 126.2 |
| 12 | 2014-12-01 | 2019-03-21 | 2019-03-21 | 101.7 | 5572 | 125.9 |
| 13 | 2015-01-01 | 2019-03-21 | 2019-03-21 | 101.6 | 3572 | 128.3 |
| 14 | 2015-02-01 | 2019-03-21 | 2019-03-21 | 101.7 | 3482 | 127.7 |
| 15 | 2015-03-01 | 2019-03-21 | 2019-03-21 | 101.7 | 3857 | 126.2 |
| 16 | 2015-04-01 | 2019-03-21 | 2019-03-21 | 101.7 | 3867 | 127 |
| 17 | 2015-05-01 | 2019-03-21 | 2019-03-21 | 101.7 | 4335 | 125.2 |
| 18 | 2015-06-01 | 2019-03-21 | 2019-03-21 | 101.7 | 4217 | 125.4 |
| 19 | 2015-07-01 | 2019-03-21 | 2019-03-21 | 101.7 | 4532 | 126.8 |
| 20 | 2015-08-01 | 2019-03-21 | 2019-03-21 | 101.6 | 4260 | 127.9 |
| 21 | 2015-09-01 | 2019-03-21 | 2019-03-21 | 101.6 | 4073 | 127.1 |
| 22 | 2015-10-01 | 2019-03-21 | 2019-03-21 | 101.7 | 4273 | 127 |
| 23 | 2015-11-01 | 2019-03-21 | 2019-03-21 | 101.8 | 4266 | 126 |
| 24 | 2015-12-01 | 2019-03-21 | 2019-03-21 | 101.8 | 5816 | 125.4 |
| 25 | 2016-01-01 | 2019-03-21 | 2019-03-21 | 103.3 | 3610 | 124.5 |
| 26 | 2016-02-01 | 2019-03-21 | 2019-03-21 | 103.3 | 3773 | 125 |
| 27 | 2016-03-01 | 2019-03-21 | 2019-03-21 | 103.3 | 4051 | 123.6 |
| 28 | 2016-04-01 | 2019-03-21 | 2019-03-21 | 103.3 | 4114 | 123.5 |
| 29 | 2016-05-01 | 2019-03-21 | 2019-03-21 | 104.2 | 4403 | 124.7 |
| 30 | 2016-06-01 | 2019-03-21 | 2019-03-21 | 104.2 | 4474 | 124.8 |

**Figure 2. Output Table from SASEFRED Code**

After I have identified the series I would like to query, the next step is to identify the temporal component of the request. The three components of the temporal component are the start date, the end date, and the frequency of the desired output. Because the same level of aggregation is not available for every series, it is important to review the frequency options for each time series on the FRED website. In this example, there are three series whose most granular frequency level is monthly, but the wine sales data I am trying to predict uses a weekly frequency. Figure 2 contains a sample of the output from the SASEFRED code.

The FREQ option in the SASEFRED engine allows you to convert a higher frequency series to a lower frequency series. For example, if it is published in a monthly format and I would like quarterly data I can request quarterly data in the FREQ statement and the interface engine will automatically do the conversion for me.

```
proc expand data=econvars out=econexpanded from=month to=week;
    id date;
    convert PCU31213031213009;
    convert MRTSSM4453USN;
    convert PCU4453104453101;
run;
```

Because I want to change the level of aggregation, I follow my query with PROC EXPAND code. The EXPAND procedure allows me to change the temporal aggregation of my input data set. Because I am going from a lower frequency (monthly) to a higher frequency (weekly), the EXPAND procedure uses a cubic spline of the input data to determine the new values of my series.

Figure 3 is a sample of the first 30 observations of the final merged table with all variables.

| | Date | Region | Varietal | Sales | Promotion | PPI_Wineries | Retail_Sales_Wine | PPI_Wine_Stores |
|---|---|---|---|---|---|---|---|---|
| 1 | 23FEB2014 | Mid West | Bordeaux | 144 | 0 | 101.37546337 | 3605.7617208 | 121.9511894 |
| 2 | 02MAR2014 | Mid West | Bordeaux | 165 | 0 | 101.51508202 | 3656.5664728 | 121.4550645 |
| 3 | 09MAR2014 | Mid West | Bordeaux | 312 | 0 | 101.57676297 | 3667.0951475 | 121.36877747 |
| 4 | 16MAR2014 | Mid West | Bordeaux | 243 | 0 | 101.57959509 | 3658.2830483 | 121.56527095 |
| 5 | 23MAR2014 | Mid West | Bordeaux | 269 | 0 | 101.54872827 | 3655.5728475 | 121.88119704 |
| 6 | 30MAR2014 | Mid West | Bordeaux | 347 | 0 | 101.5093124 | 3684.4072173 | 122.15320787 |
| 7 | 06APR2014 | Mid West | Bordeaux | 262 | 0 | 101.48454451 | 3766.6980135 | 122.23563683 |
| 8 | 13APR2014 | Mid West | Bordeaux | 197 | 0 | 101.47843697 | 3889.6703495 | 122.15651828 |
| 9 | 20APR2014 | Mid West | Bordeaux | 165 | 0 | 101.48411313 | 4020.8615052 | 122.04247746 |
| 10 | 27APR2014 | Mid West | Bordeaux | 197 | 0 | 101.49457134 | 4127.5827881 | 122.02127117 |
| 11 | 04MAY2014 | Mid West | Bordeaux | 248 | 0 | 101.50292459 | 4177.9484668 | 122.21750296 |
| 12 | 11MAY2014 | Mid West | Bordeaux | 254 | 0 | 101.50607242 | 4166.6002542 | 122.65160101 |
| 13 | 18MAY2014 | Mid West | Bordeaux | 261 | 0 | 101.50547782 | 4120.1495977 | 123.21844582 |
| 14 | 25MAY2014 | Mid West | Bordeaux | 216 | 0 | 101.50287546 | 4067.1112591 | 123.80544347 |
| 15 | 01JUN2014 | Mid West | Bordeaux | 352 | 0 | 101.5 | 4036 | 124.3 |
| 16 | 08JUN2014 | Mid West | Bordeaux | 222 | 0 | 101.49825898 | 4047.8896447 | 124.62127518 |
| 17 | 15JUN2014 | Mid West | Bordeaux | 177 | 0 | 101.49775147 | 4094.0902671 | 124.81544366 |
| 18 | 22JUN2014 | Mid West | Bordeaux | 241 | 0 | 101.49824941 | 4158.4710038 | 124.96043379 |
| 19 | 29JUN2014 | Mid West | Bordeaux | 225 | 0 | 101.49952475 | 4224.900991 | 125.13417392 |
| 20 | 06JUL2014 | Mid West | Bordeaux | 383 | 0 | 101.5013111 | 4277.8351305 | 125.40288716 |
| 21 | 13JUL2014 | Mid West | Bordeaux | 292 | 0 | 101.50296562 | 4307.482882 | 125.71780418 |
| 22 | 20JUL2014 | Mid West | Bordeaux | 341 | 0 | 101.50363178 | 4307.319932 | 125.96488716 |
| 23 | 27JUL2014 | Mid West | Bordeaux | 287 | 0 | 101.50245058 | 4270.8594564 | 126.02934913 |
| 24 | 03AUG2014 | Mid West | Bordeaux | 268 | 0 | 101.49858051 | 4191.7617811 | 125.79772994 |
| 25 | 10AUG2014 | Mid West | Bordeaux | 237 | 0 | 101.49270344 | 4076.5077044 | 125.2721674 |
| 26 | 17AUG2014 | Mid West | Bordeaux | 213 | 0 | 101.48818511 | 3954.1656 | 124.65846402 |
| 27 | 24AUG2014 | Mid West | Bordeaux | 248 | 0 | 101.48866448 | 3856.1030653 | 124.18315365 |
| 28 | 31AUG2014 | Mid West | Bordeaux | 278 | 0 | 101.4977805 | 3813.6876979 | 124.07277015 |
| 29 | 07SEP2014 | Mid West | Bordeaux | 352 | 0 | 101.51808652 | 3850.1853592 | 124.50180288 |
| 30 | 14SEP2014 | Mid West | Bordeaux | 315 | 0 | 101.54543636 | 3938.8636972 | 125.3235592 |

**Figure 3. Final Merged Table with Dependent and Independent Variables**

The final step in the process, as seen in Figure 1, is to merge the table of economic indicator variables, with the wine sales data. I use the Query Builder in SAS Enterprise Guide to join the tables and do the final data preparation. Figure 3 contains the first 30 observations of the final data preparation step in this process. When my final table is prepared for forecasting, I am prepared to use my forecasting engine of choice. With SAS Forecast Server and SAS Visual Forecasting, I can leverage the automatic independent variable selection feature. This feature tests the statistical significance of my independent variables and includes only those variables that are significant, ensuring that I will not overfit my time series model by including these independent variables.

## ADDITIONAL ETS INTERFACE ENGINES

In addition to the SASEFRED interface engine, there are 10 other ETS interface engines with similar functionality to SASEFRED. Each of these additional interface engines connects to a different externally published database.

| Interface Engine | Database |
|---|---|
| SASECRSP | Center for Research in Security Prices |
| SASEFRED | Federal Reserve Economic Database of St. Louis (FRED) |
| SASEFAME | Fame |
| SASEHAVR | Haver Analytics DLX (Data Link Express) |
| SASENOAA | National Oceanic and Atmospheric Administration (NOAA) |
| SASEOECD | Organisation for Economic Co-Operation and Development (OECD) |
| SASEQUAN | Quandl |
| SASERAIN | World Weather Online |
| SASEWBGO | World Bank Group Open (WBGO) data website |
| SASEXCCM | CRSP/Compustat Merged (CCM) Database |
| SASEXFSD | FactSet OnDemand data service |

**Table 1. Full List of SAS/ETS Interface Engines**

Some of these databases require a subscription, but others, such as the FRED data, are free to the public. Development work on these interface engines is ongoing, and SAS continues to add support for new engines. Table 1 lists of all the current SAS/ETS interface engines and the databases to which they are connected.

## CONCLUSION

The SASEFRED interface engine is a powerful tool in every forecaster's arsenal. With a few simple lines of code in a LIBNAME statement, you can quickly and easily integrate econometric time series data into a SAS table. Because of the powerful SAS forecasting engines, I can pull many times series variables from FRED that might be relevant, and let SAS statistical models test their significance for me.

## REFERENCES

"Economic Research." Federal Reserve Bank of St. Louis. http://research.stlouisfed.org. Accessed March 20, 2019.

"API Keys." Federal Reserve Bank of St. Louis. https://research.stlouisfed.org/docs/api/api_key.html. Accessed March 25, 2019.

SAS Institute Inc. 2018. *SAS/ETS 15.1 User's Guide*. Cary, NC: SAS Institute Inc. Available https://support.sas.com/documentation/onlinedoc/ets/151/etsug.pdf. Accessed March 20, 2019.

## ACKNOWLEDGMENTS

Thank you to my manager, Vinicius Vivaldi, for your unending support. This paper would not have been possible without Kelly Fellingham, the developer in SAS R&D responsible for all the ETS interface engines. A big thanks to the technical editor of this paper, David Bass.

## RECOMMENDED READING

SAS Institute Inc. 2018. *SAS/ETS 15.1 User's Guide*. Cary, NC: SAS Institute Inc. Available https://support.sas.com/documentation/onlinedoc/ets/151/etsug.pdf. Accessed March 20, 2019.

## CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Catherine LaChapelle
SAS Institute
212-413-2573
Catherine.LaChapelle@sas.com
www.linkedin.com/in/CatherineLachapelle

**Paper SAS1879-2018**

# Regime-Switching Models: Capturing Structural Changes in Time Series

Xilong Chen and Ji Shen, SAS Institute Inc.

## ABSTRACT

Stock market conditions, government policy changes, or even weather patterns can be regarded as stochastic processes that are driven by unobserved regimes. A powerful tool to explore these behavioral patterns is the regime-switching model (RSM) that is offered in the HMM procedure and the associated action in SAS® Econometrics software. This model, which is widely used in finance, economics, science, and engineering, has two characteristics: it allows different parameter values for different regimes, and it models the transition probabilities between regimes. These characteristics enable it to fully capture the structural changes in the time series. This paper uses two examples to illustrate how you can use RSMs to better understand the regime patterns in your data and improve your economic analysis. The first example demonstrates how regime-switching autoregression (RS-AR) models help you characterize the volatility and dynamics of stock returns. The second example examines the relationship and movement between the Japanese yen and the Thai baht by using regime-switching regression (RS-REG) models.

## INTRODUCTION

Many time series data, such as stock market conditions, government policy changes, weather patterns, and so on, follow different dynamics in different time periods; this behavior is called structural change or regime switching. One type of model for this kind of behavior is the regime-switching model (RSM). RSMs enable you to assign different sets of parameter values to different regimes and model the transition probabilities between regimes. They have been powerful tools for sequential data analysis (including time series analysis) in finance, economics, science, and engineering for several decades. The HMM procedure and the associated action in SAS Econometrics software support RSMs.

The two examples in this paper demonstrate how you can easily specify RSMs in the HMM procedure, perform model selection, and evaluate the predictability performance. The first example shows how regime-switching autoregression models enable you to characterize the dynamics of stock returns, identify the market states, and forecast the value at risk (VaR). The second example uses regime-switching regression to explore the relationship and movement between the Japanese yen and another East Asian currency, the Thai baht.

## REGIME-SWITCHING AUTOREGRESSION MODELS

Although bull market and bear market are well-known terms, these market states cannot be directly observed; they can be interpreted only from the observed stock prices and other financial data. In this example, regime-switching autoregression (RS-AR) models are applied to the S&P 500 index weekly returns to analyze the market states and dynamics and to forecast the VaR. The forecasting performance is also assessed.

In the RS-AR models, the observed variables (weekly returns) follow different autoregressive (AR) models in different regimes (market states), and the regimes follow a Markov chain: that is, the transition probability from the current regime to the next regime does not depend on previous regimes.

The S&P 500 index weekly returns from January 10, 1950, to December 15, 2017, are considered (S&P Dow Jones Indices LLC 2018). The original daily data are retrieved from the FRED database at the Federal Reserve Bank of St. Louis and stored in the data set **sp500Original**. The sample is divided into two periods: the in-sample period includes the 2,665 weekly returns before January 1, 2003, and the out-of-sample period includes the remaining 754 weekly returns. The following statements generate the weekly returns from the daily close price and save the client-side data to the server:

```
%let cutDate = '31DEC2002'd;
data sp500w sp500wIn;
   set sp500Original;
   format date MMDDYY10.;
   retain cumReturn 0;
   return = (log(price)-log(lag(price)))*100;
   if(return~=.) then cumReturn + return;
   if(mod(_N_,5)=1 and _N_>1) then do;
      returnw = cumReturn;
      w + 1;
      output sp500w;
      if (date<=&cutDate.) then output sp500wIn;
      cumReturn = 0;
   end;
   keep w date returnw;
run;
data cashmm.sp500wIn; set sp500wIn; run;
data cashmm.sp500w; set sp500w; run;
```

To model the weekly returns, how many market states should be considered? How many AR lags should a regime include? These are common questions in the model selection process. In this paper, the best model is selected using Akaike's information criterion (AIC): the smaller the AIC, the better the model.

An RS-AR($p$) model estimates the likelihood of $(T-p)$ observations conditional on the first $p$ observations, where $T$ is the sample size. To compare the AICs for the same number of observations among different RS-AR($p$) models with different $p$ values, you need to adjust the sample start dates. In this example, the AICs based on the 2,663 weeks of returns before January 1, 2003, are compared. The following macro variables specify how many weeks should be skipped for each RS-AR($p$) model:

```
%let w0 = '31Jan1950'd; * for AR(0), skip first 2 weeks;
%let w1 = '24Jan1950'd; * for AR(1), skip first week;
%let w2 = '17Jan1950'd; * for AR(2), skip no week;
```

In the HMM procedure, when you specify TYPE=AR, NSTATE=$k$, and YLAG=$p$ in the MODEL statement, you specify the $k$-state RS-AR($p$) model. The estimation of RS-AR models is nontrivial. It is a nonlinear optimization problem. For an RS-AR model, there might be many local optima. To increase the chance of finding the global optimum, the following measures are applied:

1. The initial parameter values are very important. In the estimation, for $k$-state RS-AR(0) models, where $k = 2, \ldots, 10$, the initial values are obtained randomly by the HMM procedure, and there is no need to use the INITIAL statement. However, for $k$-state RS-AR($p$) models, where $k = 2, \ldots, 10, p = 1, 2$, using the random initial values often leads to bad solutions; an effective approach is to use the INITIAL statement to set the initial values as the final parameter estimates from the corresponding $k$-state RS-AR($p-1$) models.
2. Although the maximum likelihood (ML) method is commonly applied to estimating RS-AR models, in theory the likelihood of an RS-AR model is unbounded and "the ML estimator as a global maximizer of the likelihood function does not exist" (Frühwirth-Schnatter 2006). The introduction of the proper prior distribution of the parameters in the MAP method can solve the unboundedness problem. You specify METHOD=MAP in the MODEL statement to apply the MAP method. In this example, several flat priors for parameters are applied.
3. A global optimization mechanism, multistart, can be used. When you specify MULTISTART=1 in the OPTIMIZE statement, multistart mode is turned on. This mechanism checks thousands of initial values and finds the best solution among dozens of local optima.

The following statements estimate 2- to 10-state RS-AR(0) models and save the information criterion for model selection. The OUTMODEL= option in the SCORE statement stores model information and parameter estimates from the in-sample data, and later they are applied to score the out-of-sample data (for example, to forecast VaRs). For the best results, multistart mode is strongly recommended.

```
* estimate k-state RS-AR(0) models, k from kStart to kEnd;
* when qMultiStart=0, multistart mode is off;
* when qMultiStart=1, multistart mode is on;
%macro estimateRSAR0(kStart, kEnd, qMultiStart);
   %let p = 0;
   %do k = &kStart. %to &kEnd.;
      proc hmm data=cashmm.sp500wIn(where=(date>=&&w&p.))
               outstat=cashmm.sp500StatIn_k&k._p&p.;
         id time=date;
         model returnw / type=ar nstate=&k. ylag=&p. method=map;
         optimize printLevel=3 printIterFreq=1 algorithm=interiorpoint
            multistart=&qMultiStart.;
         score outmodel=cashmm.sp500ModelIn&k._&p.;
         prior tpm~dir(J(&k.,&k.,1)),
            musigma~niw(J(&k.,1+&p.,0),J(&k.,1,10),
                        J(&k.,1,0.00001)@I(1+&p.),J(&k.,1,4.00001));
      run;
      data sp500StatIn_p&p._k&k.;
         set cashmm.sp500StatIn_k&k._p&p.;
         nStates=&k.; lag=&p.;
         keep nStates lag logLikelihood AIC AICC BIC HQC;
      run;
   %end;
   data sp500SelectModelIn_p&p.;
      set sp500StatIn_p&p._k&kStart. - sp500StatIn_p&p._k&kEnd.;
   run;
%mend estimateRSAR0;

* estimate k-state RS-AR(0), k from 2 to 10, with multistart mode on;
* be aware that the following macro might take tens of hours to finish;
* uncomment it to run;
* even if you do not run this macro here, later you still have a chance to get
* estimates of RS-AR(0) models;
* %estimateRSAR0(kStart=2, kEnd=10, qMultiStart=1);
```

For each of the 18 RS-AR(1) and RS-AR(2) models, the INITIAL statement with the corresponding initial parameter values is specified. In the following code, only one example, a 7-state RS-AR(1) model, is listed to illustrate how to estimate these RS-AR(1) and RS-AR(2) models. The SAS® code for other model estimations is omitted here to save space; you can find it online. Multistart mode is turned off, because it is not necessary in this example for RS-AR(1) and RS-AR(2) models.

```
* for 7-state RS-AR(1) model;
* using parameter estimates of 7-state RS-AR(0) as initial values;
%macro estimateRSAR(k, p, qMultiStart);
   ods output FinalParameterEstimates=myParmEst TPM=myTPM ISPV=myISPV;
   proc hmm data=cashmm.sp500wIn(where=(date>= &&w&p.))
            outstat=cashmm.sp500StatIn_&k._&p.;
      id time=date;
      model returnw / type=ar ylag=&p. nstate=&k. method=map;
      optimize printLevel=3 printIterFreq=1 algorithm=interiorpoint
         Multistart=&qMultiStart.;
      score outmodel=cashmm.sp500ModelIn&k._&p.;
      prior tpm~dir(J(&k.,&k.,1)),
         musigma~niw(J(&k.,1+&p.,0),J(&k.,1,10),
                     J(&k.,1,0.00001)@I(1+&p.),J(&k.,1,4.00001));
      initial tpm={0.67872 0.00000 0.00000 0.00000 0.00000 0.02177 0.29951,
                   0.00000 0.10799 0.00000 0.05413 0.00000 0.83787 0.00000,
                   0.00000 0.09589 0.90411 0.00000 0.00000 0.00000 0.00000,
                   0.59099 0.00000 0.00000 0.16756 0.02889 0.00000 0.21257,
                   0.00000 0.42665 0.48353 0.00759 0.00000 0.00000 0.08223,
```

```
                    0.00000 0.30024 0.00000 0.00000 0.03013 0.66964 0.00000,
                    0.02400 0.00000 0.00000 0.24036 0.00000 0.00380 0.73184},
            const={0.61433, 2.47350, -0.18333, 1.73559, -5.19095, -0.61715, -0.47292},
            cov={0.55251, 2.04663, 20.20570, 0.74352, 1.54215, 3.11860, 1.59261};
    run;
    data sp500StatIn_&k._&p.;
       set cashmm.sp500StatIn_&k._&p.;
       nStates=&k.; lag=&p.;
       keep nStates lag logLikelihood AIC AICC BIC HQC;
    run;
%mend estimateRSAR;
%estimateRSAR(k=7, p=1, qMultiStart=0);
```

After running the estimation of all 27 of the 2- to 10-state RS-AR(0) to AR(2) models, you get Table 1, which displays the AICs (the printing code is omitted here). As the table shows, the smallest AIC corresponds to the 7-state RS-AR(1) model.

| k | p = 0 | p = 1 | p = 2 |
|---|---|---|---|
| 2 | 11064.28 | 11055.14 | 11058.14 |
| 3 | 10982.10 | 10975.28 | 10981.72 |
| 4 | 10961.74 | 10954.12 | 10950.02 |
| 5 | 10943.19 | 10943.54 | 10949.26 |
| 6 | 10940.67 | 10940.03 | 10943.92 |
| 7 | 10939.72 | 10935.71 | 10942.71 |
| 8 | 10948.64 | 10950.38 | 10955.33 |
| 9 | 10966.16 | 10971.05 | 10973.85 |
| 10 | 10983.23 | 10991.19 | 10987.96 |

**Table 1. AICs for 27 RS-AR Models**

The 7-state RS-AR(1) model contains 70 parameters. The following statements print the observation parameters and calculate the unconditional mean and variance of weekly returns for each regime (which are displayed in the columns “mean” and “variance”), as shown in Table 2. The values of the AR coefficients show that the process in each regime is stationary. The standard errors of the AR parameter estimates (which are omitted here) show that the AR parameters are significant at the 10% significance level in four out of seven regimes.

```
%Let k=7;
%Let p=1;
%Let nTPM = %SYSEVALF(&k.*&k.);
%Let nObsParms = %SYSEVALF(3*&k.);
%Let nParms = %SYSEVALF(&nObsParms.+&nTPM.);
data obsParms;
   set myParmEst;
   array myest(&nObsParms.) _temporary_;
   retain myest:;
   if _N_ > (&nTPM.) then do;
      myest[_N_-(&nTPM.)] = Estimate;
   end;
   if _N_ = &nParms. then
      do regime=1 to &k.;
```

```
         constant = myest[regime];
         ar = myest[&k.+regime];
         cov = myest[2*&k.+regime];
         mean = constant/(1-ar);
         variance = cov/(1-ar**2);
         output;
      end;
   keep regime constant ar cov mean variance;
run;
proc print data=obsParms noobs; run;
```

| regime | constant | ar | cov | mean | variance |
|---|---|---|---|---|---|
| 1 | 0.64751 | 0.02249 | 0.5317 | 0.66241 | 0.5320 |
| 2 | 1.29306 | -0.19557 | 2.6660 | 1.08154 | 2.7721 |
| 3 | -0.07326 | -0.12035 | 17.6076 | -0.06539 | 17.8664 |
| 4 | 1.89273 | 0.13222 | 0.6766 | 2.18112 | 0.6886 |
| 5 | -5.63288 | -0.16052 | 0.8664 | -4.85373 | 0.8893 |
| 6 | -1.85121 | -0.10825 | 1.5858 | -1.67039 | 1.6046 |
| 7 | -0.66543 | -0.04465 | 1.3219 | -0.63698 | 1.3245 |

**Table 2. Observation Parameters and Unconditional Means and Variances**

According to the unconditional mean and variance of weekly returns for each regime, you can draw the Gaussian kernel for each regime and compare them with the histogram of weekly returns by using the following statements. As shown in Figure 1, roughly speaking, three regimes (regimes 1, 2, and 4) could be considered bull market states, where the mean of weekly returns is significantly positive and the risk is relatively low, and four regimes (regimes 3, 5, 6, and 7) could be considered bear market states, where the mean of weekly returns is significantly negative (regimes 5, 6, and 7) or the risk (measured by the unconditional variance) is extremely high (regime 3).

```
data muSigma;
   set obsParms end=eof;
   array mu(&k.); array sd(&k.);
   retain mu: sd:;
   mu(_N_) = mean;
   sd(_N_) = sqrt(variance);
   if eof then output;
run;

%macro plotLearning(myParm,k,myData,myColumn);
   data _NULL_;
      set &myParm.;
      %do i = 1 %to &k.;
         call symputx("mu&i.",mu&i.,'G');
         call symputx("sigma&i.",sd&i.,'G');
      %end;
   run;
   proc sgplot data=&myData.;
      refline 0 / axis=x lineattrs=(thickness=3);
      histogram &myColumn. / nbins=200;
      %do i = 1 %to &k.;
         density &myColumn. / type=normal(mu=&&mu&i. sigma=&&sigma&i.)
            name="regime&i."
            legendlabel="Expected Gaussian Dist. for Regime &i.";
```

```
      %end;
      keylegend %do i = 1 %to &k.; "regime&i." %end;;
   run;
%mend plotLearning;

%plotLearning(muSigma,&k.,sp500wIn,returnw);
```

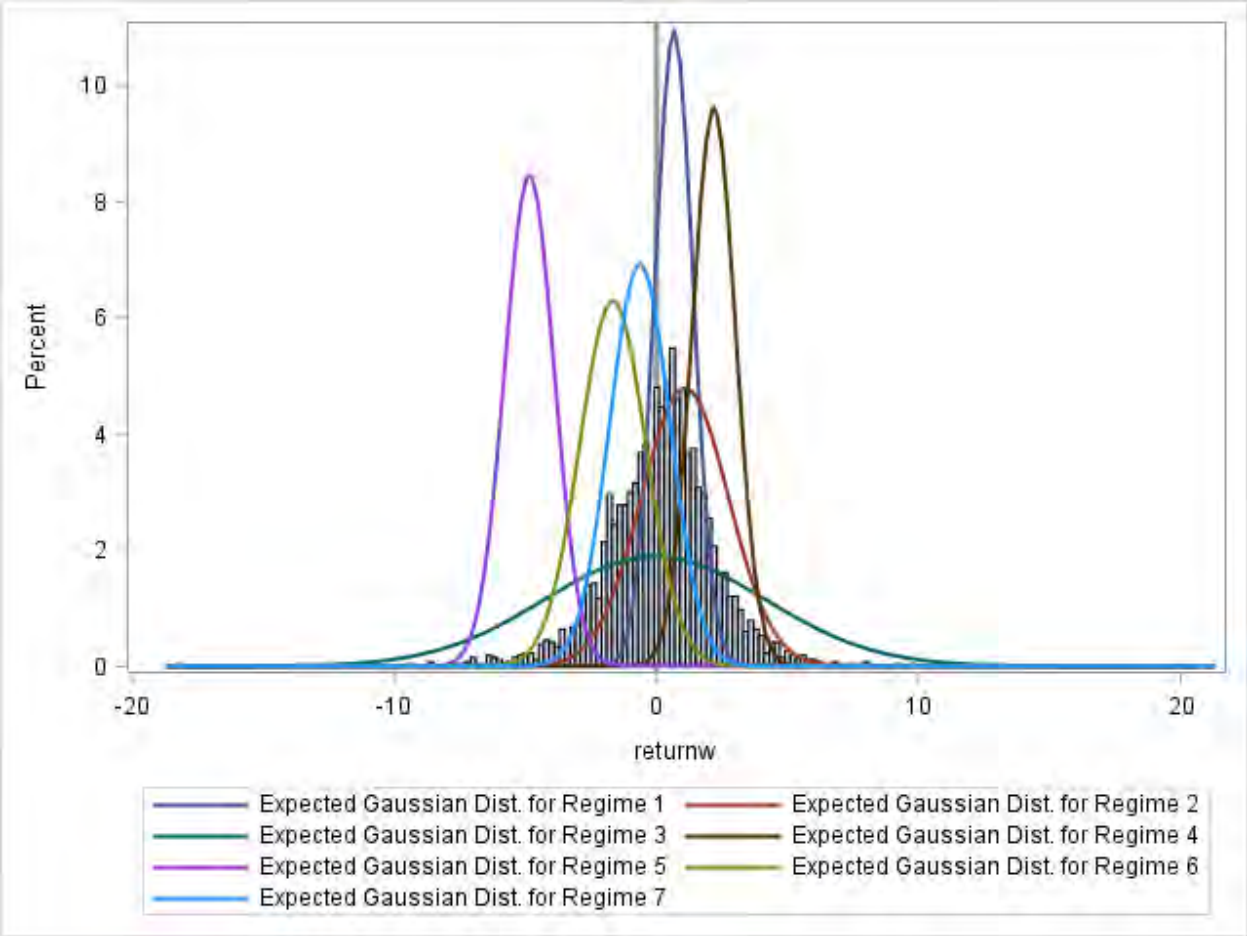



**Figure 1. Gaussian Kernels for Seven Regimes**

The following statements display the transition probabilities, the steady-state probability distribution (in the column “STPD”), and the expected duration of each regime (in the column “duration”) in Table 3. Regime 5 is an exceptional bear market state: there is only a 1% chance that the market falls in that regime; and even when the market falls in that regime, it quickly recovers. A very bullish market (regime 4) or a very bearish market (regime 6) has about a 10% chance, and neither market lasts long. The market has a 50% chance to be somewhat bullish (regime 1 and 2) or a 20% chance to be somewhat bearish (regime 7). Regime 3 might be highly related to the financial crisis (as discussed later): the risk is extremely high; it has a 7% chance of occurring; and when it does occur, it takes a long time to recover (the expected duration is about 12 weeks).

```
data tp;
   set myTPM;
   array tpm(&k.) state1-state&k.;
   duration = 1/(1-tpm(_N_));
   keep state: duration;
run;
proc print data=tp noobs label; label state='regime'; run;
```

| regime | 1 | 2 | 3 | 4 | 5 | 6 | 7 | STPD | duration |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.69303 | 0.00000 | 0.00000 | 0.00000 | 0.00000 | 0.02434 | 0.28263 | 0.20418 | 3.2577 |
| 2 | 0.00000 | 0.71175 | 0.00000 | 0.02256 | 0.00000 | 0.26568 | 0.00000 | 0.27468 | 3.4693 |
| 3 | 0.00000 | 0.08387 | 0.91613 | 0.00000 | 0.00000 | 0.00000 | 0.00000 | 0.06982 | 11.9238 |
| 4 | 0.61677 | 0.00000 | 0.00000 | 0.10621 | 0.03129 | 0.00000 | 0.24574 | 0.07502 | 1.1188 |
| 5 | 0.00000 | 0.41006 | 0.47943 | 0.05448 | 0.00000 | 0.00000 | 0.05602 | 0.01013 | 1.0000 |
| 6 | 0.00000 | 0.49339 | 0.00713 | 0.00000 | 0.05552 | 0.44396 | 0.00000 | 0.14019 | 1.7984 |
| 7 | 0.07260 | 0.00000 | 0.00000 | 0.26685 | 0.00000 | 0.00000 | 0.66055 | 0.22598 | 2.9460 |

**Table 3. Transition Probabilities, Steady-State Probability, and Expected Duration**

Next, the INMODEL= option in the SCORE statement is used to forecast the VaRs at different levels, such as 1%, 5%, and 10%, as shown in Table 4. The following macros can predict the VaRs for the out-of-sample period and evaluate the predictive performance by the likelihood ratio (LR) test for the unconditional coverage of the VaR forecast (Kuester, Mittnik, and Paolella 2006).

```
%let alpha1 = 0.80;
%let alpha2 = 0.90;
%let alpha3 = 0.98;
%macro VaREstimation(dsModel,k,p,iStart,iEnd,oosStart,oosEnd,dsForecastPrefix);
   %do i = &iStart. %to &iEnd.;
      proc hmm data=cashmm.sp500w(where=(date>=&&w&p.))
               outstat=cashmm.sp500Stat&k._&p.;
         score inmodel=&dsModel.;
         forecast out=cashmm.&dsForecastPrefix.&i. alpha=&&alpha&i. online;
         decode out=cashmm.sp500Decode&k._&p.;
      run;
      data &dsForecastPrefix.&i.;
         set cashmm.&dsForecastPrefix.&i.;
      run;
      proc sort data=&dsForecastPrefix.&i.; by date; run;
      data &dsForecastPrefix.&i.;
         set &dsForecastPrefix.&i. (FIRSTOBS=&oosStart. OBS=&oosEnd.
                                    keep=date returnw_Q1
                                    rename=(returnw_Q1=returnw_Q1_&k.&p.&i.));
         time=_N_; nStates = &k; Lag = &p;
      run;
   %end;
%mend VaREstimation;

%macro VaREvaluate(dsForecastPrefix,k,p,iStart,iEnd,n,dsOut);
   data sp500wout;
      set sp500w;
      if (date>&cutDate.) then output;
   run;
   data forecastData;
      set sp500wout;
      time=_N_;
   run;
   data forecastData;
      merge &dsForecastPrefix.: forecastData;
      by time;
```

```
    run;
    data &dsOut.;
       set forecastData;
       retain %do i = &iStart. %to &iEnd.; cviol&k.&p.&i. 0 %end; ;
       retain %do i = &iStart. %to &iEnd.; creturnw&k.&p.&i. 0 %end; ;
       %do i = &iStart. %to &iEnd.;
          if returnw le returnw_Q1_&k.&p.&i. then do;
             cviol&k.&p.&i.=cviol&k.&p.&i.+1;
          end;
          creturnw&k.&p.&i. = creturnw&k.&p.&i. + returnw_Q1_&k.&p.&i.;
       %end;
       if _N_ = &n. then do;
          %do i = &iStart. %to &iEnd.;
             Norminal = (1-&&alpha&i.)/2;
             Viol = cviol&k.&p.&i. / &n.;
             LR = 2*(cviol&k.&p.&i.*log(Viol)+(&n.-cviol&k.&p.&i.)*log(1-Viol)
                -(cviol&k.&p.&i.*log(Norminal)+(&n.-cviol&k.&p.&i.)
                 *log(1-Norminal)));
             pValue = 1 - cdf("CHISQUARE", LR, 1);
             meanVaR = creturnw&k.&p.&i. / &n.;
             output;
          %end;
       end;
       label Norminal='Target Prob.' Viol='Violation Ratio' LR='LR Stat.'
             pValue='Pr > ChiSq' meanVaR='Avg. of VaR' nStates='Number of States'
             lag='Lag';
       keep nStates Lag Norminal Viol LR pValue meanVaR;
    run;
    proc print data=&dsOut. noobs label;  format Viol LR pValue meanVaR 6.4; run;
%mend VaREvaluate;

%macro VaR(k,p);
   %VaREstimation(cashmm.sp500ModelIn&k._&p.,&k,&p,1,3,
      %eval(2665-(2-&p.)),%eval(3418-(2-&p.)),sp500Forecastk&k._p&p.);
   %VaREvaluate(sp500Forecastk&k._p&p.,&k,&p,1,3,754,VaR_outputk&k._p&p.);
%mend VaR;

%VaR(7,1);
```

| Number of States | Lag | Target Prob. | Violation Ratio | LR Stat. | Pr > ChiSq | Avg. of VaR |
|---|---|---|---|---|---|---|
| 7 | 1 | 0.10 | 0.0968 | 0.0857 | 0.7697 | -2.307 |
| 7 | 1 | 0.05 | 0.0610 | 1.8028 | 0.1794 | -3.110 |
| 7 | 1 | 0.01 | 0.0186 | 4.4635 | 0.0346 | -4.928 |

**Table 4. Predictive Performance of VaR Forecasts**

According to the *p*-values in the “Pr > ChiSq” column in Table 4, at the 1% significance level, no tests can reject the null hypothesis that the number of violations is correct. Hence, the 7-state RS-AR(1) model has the correct unconditional coverage for the 1%, 5%, and 10% VaR forecasts.

In fact, besides the good predictability of the tail of the distribution of the weekly returns as shown in the VaR forecast analysis, the 7-state RS-AR(1) model also provides a very good prediction of the whole distribution of the weekly returns, which can be shown by comparing the average weekly log likelihoods of the in-sample period and the out-of-sample period. The SAS code is omitted here. As shown in Table 5, for the 7-state RS-AR(1) model, the average weekly log likelihood in the out-of-sample period is even better than in the in-sample period. Compared to both the simplest model, the 2-state RS-AR(0) model, which has the fewest parameters and the worst in-sample fit, and the most complex model, the 10-state RS-AR(2) model, which has the most parameters and the best in-sample fit, the 7-state RS-AR(1) model has the best out-of-sample forecast ability (that is, the largest average weekly log likelihood in the out-of-sample period).

| k | p | In-sample | Out-of-sample |
|---|---|---|---|
| 2 | 0 | -2.07515 | -2.05037 |
| 7 | 1 | -2.02961 | -2.01583 |
| 10 | 2 | -2.01426 | -2.02827 |

**Table 5. Comparison of Average Weekly Log Likelihoods**

Finally, the decoded regimes provide a historical view of what happened, given all the available data. The data set **cashmm.sp500Decode7_1**, as a by-product, is generated by the DECODE statement in the VaR forecast. The following statements plot the decoded regimes, as shown in Figure 2. Regimes 1, 4, and 7 seem to belong to one group, the bullish market: although there are very upward-moving (regime 4) and somewhat downward-moving (regime 7) days, the main trend is upward (regime 1). Regimes 2, 3, 5, and 6 seem to belong to another group, the bearish market: although there are some upward-moving days (regime 2), the main trend is downward (regime 6), even very downward (regime 5) or very volatile (regime 3). Regimes 3 and 5 seem to be indicators of a financial crisis; notice their appearance around the 1987 financial crisis, the 1997 Asian financial crisis, the Y2K crash, and the 2008 financial crisis. When the market is in the bullish state, it lasts for a long time before switching to the bearish state, or vice versa.

```
proc sgplot data=cashmm.sp500Decode7_1(where=(state~=.));
   scatter x=date y=state / group=state;
run;
```


**Figure 2. Decoded Regime for Each Week**

## REGIME-SWITCHING REGRESSION MODELS

A paper by Kim, Min, McDonald, and Hwang (2012) uses the regime-switching regression (RS-REG) models to find that there are synchronization periods (one regime) and desynchronization periods (the other regime) between the Swiss franc exchange rates of floating East Asian currencies and the exchange rate between the Swiss franc and the Japanese yen. This example follows ideas of this paper, using different data and focusing on only one East Asian currency, the Thai baht, and shows how you can use the HMM procedure to estimate the RS-REG models and interpret the results.

The daily exchange rate data (Board of Governors of the Federal Reserve System (US) 2018a, 2018b, 2018c, 2018d), stored in the data set **ero**, are retrieved from the FRED database, including US dollar (USD) exchange rates from January 1999 to January 2018 for the Australian dollar (AUD), the euro (EUR), the Japanese yen (JPY), and the Thai baht (THB). The following statements prepare the weekly returns of exchange rates for the RS-REG model. The variable DEXTHUSw is the return of the THB-USD exchange rate; DEXJPUSw is the return of the JPY-USD exchange rate; DEXEUUSw is the return of the EUR-USD exchange rate; and DEXALUSw is the return of the AUD-USD exchange rate.

```
data er;
   set ero(where=(DEXUSEU~=.));
   array xr [4] DEXTHUS  DEXJPUS  DEXEUUS  DEXALUS;
   array xrr[4] DEXTHUSr DEXJPUSr DEXEUUSr DEXALUSr;
   array xrc[4] DEXTHUSc DEXJPUSc DEXEUUSc DEXALUSc;
   array xrw[4] DEXTHUSw DEXJPUSw DEXEUUSw DEXALUSw;
   DEXEUUS = 1 / DEXUSEU;
   DEXALUS = 1 / DEXUSAL;
   do i = 1 to 4;
      xrr[i] = (log(xr[i])-log(lag(xr[i])))*100;
      if(xrr[i]~=.) then xrc[i] + xrr[i];
   end;
   if(mod(_N_,5)=1 and _N_>1) then do;
      do i = 1 to 4; xrw[i] = xrc[i]; end;
      w + 1;
      output er;
      do i = 1 to 4; xrc[i] = 0; end;
   end;
   keep date DEXTHUSw DEXJPUSw DEXEUUSw DEXALUSw;
run;
data mycas.er; set er; run;
```

The following statements estimate the bi-state RS-REG model. You specify the regression in the MODEL statement: on the left-hand side is the dependent variable, DEXTHUSw, and on the right-hand side are the regressors, DEXJPUSw, DEXEUUSw, and DEXALUSw. You specify TYPE=REG in the MODEL statement for the RS-REG model. NSTATE=2 in the MODEL statement indicates that there are two regimes. The SMOOTH statement outputs the smoothed probabilities of the two regimes.

```
proc hmm data=mycas.er;
   id time=date;
   model DEXTHUSw = DEXJPUSw DEXEUUSw DEXALUSw / type=reg nstate=2;
   smooth out=mycas.erSmooth;
run;
```

The parameter estimates are shown in Table 6. XL$k$_$l$_$i$_$j$ is the parameter for the $j$th regressor at lag $l$ in the $i$th equation for the $k$th regime. Because this model has two regimes, $k$ can take the value 1 or 2. This is a univariate model, so there is only one equation, and $i$ is always 1. The regression projects only the current regressor; hence lag $l$ is always 0. There are three regressors; in sequence, $j = 1$ for the first regressor, DEXJPUSw, then 2 for DEXEUUSw, and 3 for DEXALUSw. The main interest is in XL$k$_0_1_1, the relationship between THB-USD and JPY-USD. In the first regime, XL1_0_1_1 is significant at the 5% significance level, indicating that regime 1 is the synchronization regime, where the THB-USD exchange rate is influenced by the JPY-USD exchange rate; and XL2_0_1_1 is not significant at the 5% significance level, indicating that regime 2 is the desynchronization regime, where the THB-USD exchange rate is not influenced by the JPY-USD exchange rate. This result confirms what has been found by Kim et al. (2012). It is also worth mentioning that the covariance of innovations in the synchronization regime is much lower than in the desynchronization regime, which is opposite to another finding of Kim et al. (2012): there is greater volatility during the synchronization period than during the desynchronization period. It might be because the model in this paper has different regressors and uses different data. Further discussion of this topic is beyond the scope of this paper.

| Parameter Estimates | | | | |
|---|---|---|---|---|
| Parameter | Estimate | Standard Error | t Value | Pr > \|t\| |
| TPM1_1 | 0.950479 | 0.022524 | 42.20 | <.0001 |
| TPM1_2 | 0.049521 | 0.022524 | 2.20 | 0.0282 |
| TPM2_1 | 0.169027 | 0.063339 | 2.67 | 0.0077 |
| TPM2_2 | 0.830973 | 0.063339 | 13.12 | <.0001 |
| CONST1_1 | -0.033719 | 0.021445 | -1.57 | 0.1162 |
| CONST2_1 | 0.062503 | 0.100694 | 0.62 | 0.5349 |
| XL1_0_1_1 | 0.125724 | 0.016067 | 7.83 | <.0001 |
| XL1_0_1_2 | 0.087433 | 0.019151 | 4.57 | <.0001 |
| XL1_0_1_3 | 0.100358 | 0.014144 | 7.10 | <.0001 |
| XL2_0_1_1 | 0.063758 | 0.076146 | 0.84 | 0.4026 |
| XL2_0_1_2 | 0.028165 | 0.085654 | 0.33 | 0.7424 |
| XL2_0_1_3 | 0.231856 | 0.071930 | 3.22 | 0.0013 |
| COV1_1_1 | 0.230889 | 0.034013 | 6.79 | <.0001 |
| COV2_1_1 | 1.745741 | 0.316710 | 5.51 | <.0001 |

**Table 6. Parameter Estimates of RS-REG Model on CHF-PHP**

In this model, the Markov chain is assumed to be stationary, and the initial state probability vector (ISPV) is the same as the steady-state probability distribution. The estimates of ISPV are shown in Table 7. The synchronization regime has about a 77% chance. This is confirmed by the smoothed probabilities plot in Figure 3.

| Initial State Probability Vector | |
|---|---|
| State | Estimation |
| 1 | 0.77341 |
| 2 | 0.22659 |

**Table 7. Initial State Probability Vector (Steady-State Probability Distribution)**

The following statements plot the smoothed probabilities of the synchronization regime. Figure 3, which can be compared to Figure 2 in Kim et al. (2012), shows that the degree of synchronization is high for the Thai baht.

```
proc sgplot data=mycas.erSmooth;
   series x=date y=state1;
run;
```

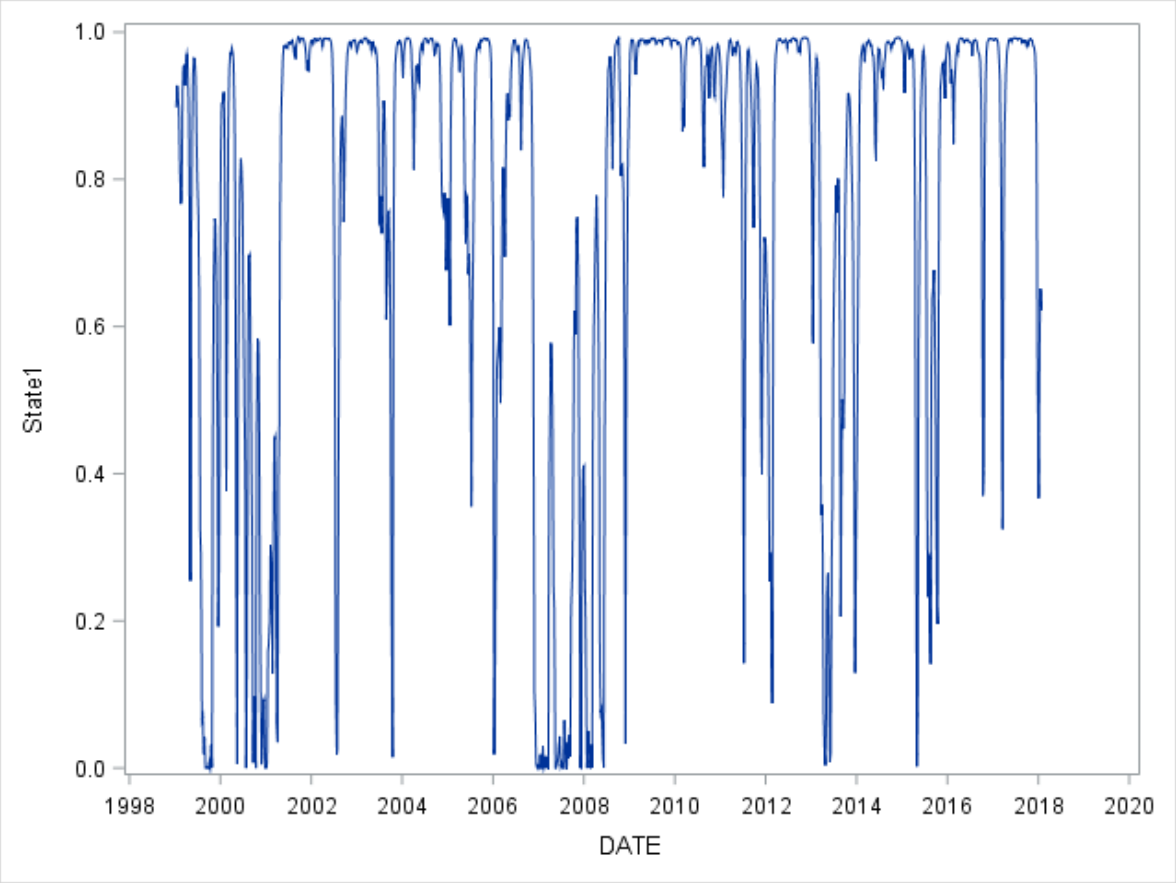



**Figure 3. Smoothed Probabilities of Synchronization Regime**

## CONCLUSION

By using the HMM procedure, you can exploit regime-switching models to better understand the regime patterns in your data and improve your economic analysis. Beyond the topics discussed in this paper, PROC HMM also provides support for Gaussian HMMs and Gaussian mixture HMMs, which are two other powerful tools for time series analysis. You can also use the HMM procedure to analyze cross-sectional time series data (also known as panel data).

## REFERENCES

Board of Governors of the Federal Reserve System (US) (2018a). Japan/U.S. Foreign Exchange Rate [DEXJPUS]. Retrieved from FRED database, Federal Reserve Bank of St. Louis. https://fred.stlouisfed.org/series/DEXJPUS.

Board of Governors of the Federal Reserve System (US) (2018b). Thailand/U.S. Foreign Exchange Rate [DEXTHUS]. Retrieved from FRED database, Federal Reserve Bank of St. Louis. https://fred.stlouisfed.org/series/DEXTHUS.

Board of Governors of the Federal Reserve System (US) (2018c). U.S./Australia Foreign Exchange Rate [DEXUSAL]. Retrieved from FRED database, Federal Reserve Bank of St. Louis. https://fred.stlouisfed.org/series/DEXUSAL.

Board of Governors of the Federal Reserve System (US) (2018d). U.S./Euro Foreign Exchange Rate [DEXUSEU]. Retrieved from FRED database, Federal Reserve Bank of St. Louis. https://fred.stlouisfed.org/series/DEXUSEU.

Frühwirth-Schnatter, S. (2006). *Finite Mixture and Markov Switching Models*. New York: Springer.

Kim, B., Min, H., McDonald, J., and Hwang, Y. (2012). "Yen-Synchronization of Floating East Asian Currencies: A Regime-Switching Regression Model and Micro-structural Analysis." *Journal of the Japanese and International Economies* 26:221–232.

Kuester, K., Mittnik, S., and Paolella, M. S. (2006). "Value-at-Risk Prediction: A Comparison of Alternative Strategies." *Journal of Financial Econometrics* 4:53–89.

S&P Dow Jones Indices LLC (2018). S&P 500 [SP500]. Retrieved from FRED database, Federal Reserve Bank of St. Louis. https://fred.stlouisfed.org/series/SP500.

## ACKNOWLEDGMENTS

The authors would like to thank Jan Chvosta and Mark Little for their valuable comments and suggestions and Ed Huddleston for his thorough editorial review.

## CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors:

Xilong Chen and Ji Shen
SAS Institute Inc.
Xilong.Chen@sas.com and Ji.Shen@sas.com

Paper SAS1673-2018

# Getting More Insight into Your Forecast Errors with the GLMSELECT and QUANTSELECT Procedures

Gerhard Svolba, SAS Institute Inc. Austria

## ABSTRACT

Is it sufficient just to monitor the quality of your forecast models over time? Can data science methods identify the drivers for large forecast errors and provide more insights than descriptive statistics? Do demand planners really improve forecast accuracy with their manual overwrites? Using a real-life case study, this paper answers these questions. It shows how you can study the impact of factors like product group, forecast horizons, seasonality, or the forecast model type on forecast accuracy and convert them into actionable results. You learn how univariate methods provide first insights into the structure and relationships of your forecast data. You gain insight into how manual overwrites of the statistical forecast change forecast accuracy in both directions and how you use analytical and graphical methods to illustrate these findings. You see how multivariate analytical methods like linear and quantile regression provide additional relevant insight. You learn how to use the GLMSELECT, QUANTSELECT, and QUANTREG procedures to identify the most important influential factors on the forecast error. You see how you can enhance and interpret the output of these procedures to quantify the effects of the influential factors. You learn how to convert the results from the SAS® procedures into actions to improve your forecasting process. The paper shows an outline of how to use the REGSELECT and QTRSELECT procedures to apply these methods in SAS® Viya®.

## INTRODUCTION

### APPLY ANALYTICAL METHODS ACROSS DIFFERENT BUSINESS DOMAINS

Analytical methods can leverage the analysis outcome for various business questions. Going one level deeper than simple descriptive methods provides insights in the relationship between influential variables. Analytical methods also help you spot multivariate relationships and enable you to receive an objective and data driven answer to your business questions.

The book *Applying Data Science: Business Case Studies Using SAS®* (Svolba 2017) is dedicated to the application of analytical methods to different types of practical questions. It shows how analytical methods that have been successfully used in certain business domains can and should be applied also to other business areas. For example, you can apply survival analysis techniques to analyze the retention time of employees, or you can use ARIMA methods and multivariate adaptive regression splines to automatically detect breakpoints in your time series data.

### CASE STUDY: ANALYZING THE FORECAST ERROR

This paper deals with a case study from the demand forecasting area. The focus is to investigate the forecast error, which is measured as the deviation between the forecasted demand and the actual demand. It shows how analytical methods like regression analysis can be used to identify factors that have an impact on the magnitude of the forecast error.

The case study does not deal with the creation of the statistical forecast itself but with the evaluation of the forecast quality. Typical business questions in forecast quality are discussed and this paper shows how they can be solved with analytical methods, like descriptive analyses or general linear models.

### USING REGRESSION ANALYSIS

The statistical tools that are shown here include boxplots, histograms, and descriptive measure like mean, median, and the quartiles, as well as linear regression and quantile regression methods.

The analysis provides insight about the drivers for different levels of forecast quality. It shows that general linear models are perfectly suited to answer business questions related to forecast quality.

- General linear models enable you to automatically select the most important variable for the analysis.
- They provide an answer about the importance of different influential factors.
- They express the mathematical relationship between forecast error and analysis variable.

### STUDYING THE EFFECT OF MANUAL OVERRIDES

In the forecasting process, statistical forecasts are often overridden with judgmental forecasts by the forecaster or demand planner. The analysis shows whether the overall forecast quality is improved with manual overrides. A detailed analysis of the characteristics of the manual overrides shows their effect on the forecast quality.

The case study also explains the main assumptions and deliverables of regression analyses and illustrates the main features with results from the business questions.

### BUSINESS QUESTIONS FOR THE ANALYSIS

From a business point of the view, the following questions are of interest and are analyzed in this case study.

- What is the distribution of the forecast error over all products?
    - What is the average forecast error?
    - What is the forecast error that is not exceeded by the top 25%, 50%, and 75% of the forecasts?
- Which factors influence the forecast error?
    - Is the forecast error different between product groups or price categories?
    - Does the launch month or the age of the product influence the forecast error?
    - Do different forecast models generate different forecast quality?
    - Do forecasts get better if the target months get closer?
    - Is there a difference in forecast quality between the calendar months or between years?
- How do the manual forecasts compare to statistical forecasts?
    - What is the average improvement of applying judgmental corrections?
    - Are there areas where judgmental corrections have a larger benefit?
    - Are there cases where judgmental corrections decrease forecast quality?
- Are there trends over time in the forecast errors that can be detected?

## BUSINESS BACKGROUND OF THE CASE STUDY

### NEED FOR DIFFERENT TYPES OF FORECASTS AND ARTICLE SEGMENTATION

In this case study the business department is the operational planning department of an international retail and manufacturing company. For their sales and demand planning, demand forecasts on a monthly basis are needed. These forecasts are generated automatically with analytical models in SAS® Forecast Server and SAS® Enterprise Miner™.

Some of the articles that are sold by the company have been in the assortment already for some years, and other articles remain in the product offering only for a limited period of time, like 6 or 12 months. With part of its product range, the company operates in the fashion business. Here, articles are retired when a new collection comes on the market or when articles do not sell as expected.

Note that in some industries, the term SKU is often used instead of “articles”. SKU is the abbreviation for Stock Keeping Unit. In this case study, the term “article” is used.

### Article Segmentation

For the forecasting process, the articles are segmented into LONG and SHORT articles, based their available data history.

- LONG articles have a history of 15 months or more and are forecast with time series forecasting methods like exponential smoothing and ARIMA models.
- SHORT articles have a history up to 14 months and are forecast with a predictive model based on attributes of the product itself.

The future forecast horizon for which forecasts shall be created ranges from 4 to 18 months. These forecasts are used for different purposes:

- The rolling monthly forecasts for sales and demand planning are created for 4-6 months in the future.
- Forecasts of up to 18 months are used for budget planning for the next business year.

### Target Months and Create Month

A forecast for a particular month, TARGET_MONTH, is usually created in more than one period of time (CREATE_MONTH). The forecast for the target month July might be created in the create months February, March, April, May, and June. Table 1 illustrates this case.

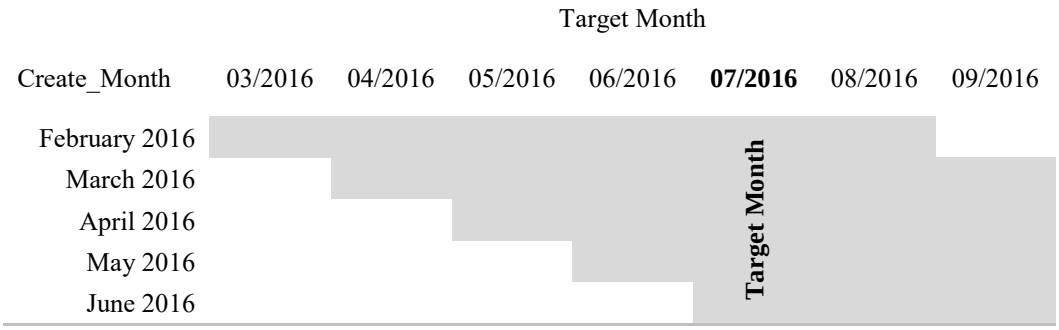



**Table 1. Target Month and Create Month**

For a particular target month, forecasts from different create months are available in the data. These records have the same TARGET_MONTH but different values of the CREATE_MONTH.

| ArticleNum | CreateMonth | TargetMonth | LeadTime | Forecast |
|---|---|---|---|---|
| 15942 | 2016.02 | 2016.07 | 5 | 1609 |
| 15942 | 2016.03 | 2016.07 | 4 | 1555 |
| 15942 | 2016.04 | 2016.07 | 3 | 1635 |
| 15942 | 2016.05 | 2016.07 | 2 | 1578 |
| 15942 | 2016.06 | 2016.07 | 1 | 1571 |

**Output 1. Forecasts from Different Create Months**

Forecasts can also be differentiated by the LEADTIME, which is the interval in the future for which they are created. The lead time of a forecast for July that is created in February, is 5.

## MEASURING THE FORECAST ERROR OF STATISTICAL AND JUDGMENTAL FORECAST

### Statistical and Judgmental Forecast

The forecasts that are created by analytical methods are called statistical forecasts. For articles in the segment LONG, three different model types are available to create the statistical forecast; for articles in

the segment SHORT, two different models are available. For each article, one of these models is defined as the champion model, and the other models are treated as challenger models for model quality performance monitoring over time.

The forecast can either directly be used for the demand planning, or it can be overwritten by the demand planner and is then called a judgmental forecast.

Usually, the demand planners use information about short-term market trends, promotion activities, or other factors that are not considered in the analytical model for the judgmental forecast.

There is a lot of discussion in forecasting practice about whether manual overrides truly improve forecast quality. See also Goodwin 2009 or Gilliland 2010 for this topic.

The data that are shown here are based on real world data. However, they have been amended for data privacy reasons.

### Variety of Forecast Measures

There are many different measures to quantify the forecast error. Depending on the industry, the nature of the forecasting problem, and personal preferences, different methods are applied. The methods range in complexity of the calculation method, and some of them are combinations of other basic measures. In the forecasting community, there is no general agreed-upon “best measure” for the forecast error. Gilliland, 2010, for example, discusses the forecast value added (FVA). The FVA is the added value of the forecast in accuracy, compared to a naïve or baseline forecast.

### Using the MAPE

In the example shown here, the forecast error is measured with the MAPE, the mean absolute percentage error. There are many critics for using the MAPE.

- The MAPE is asymmetric; a perfect fit results in a MAPE of 0. However, there is no restriction to the upper limit.
- For an observed demand of 0, the MAPE formula causes a division by zero.
- A forecast of 0 leads to a MAPE of 100. Thus, a forecasting model could learn this feature and limit its forecast error by forecasting 0 for all time points.

The advantage of the MAPE, however, is its interpretability, and it is thus very broadly used in business forecasting. The MAPE is calculated with the following steps:

1. Calculate the absolute value of the difference between the forecasted value and the actual value (that’s where the A in MAPE comes from).
2. Convert the absolute difference into a relative difference by dividing it by the actual value. This expresses the forecast error as a percentage of the actual value (that is, the origin of the P and the E in the abbreviation MAPE).
3. Finally, you average these absolute percentage errors over all available time points and receive a Mean Absolute Percentage Error (this is where the M comes from).

### Calculating the APE for the Analysis

For the task of analyzing the forecast error per month, only the APE and not the MAPE is calculated. This means that the last step of averaging the forecast errors per article is not performed.

In forecast error analysis, you want to see the deviation for each individual point in time. This provides more detailed insight and also allows analyzing potential seasonal effects in the forecast error. It also provides insight into the change of forecast quality from different forecast create months for a particular target month.

For this case study, the forecast error is the absolute percentage error between the statistical forecast and the actual demand. The abbreviation APE_STAT is used for it.

# AVAILABLE DATA AND DATA PREPARATION

## OVERVIEW OVER THE AVAILABLE DATA SOURCES

The data table for the analysis is built based on three tables:

| Name | Description | Primary Key Columns |
|---|---|---|
| STATFC | Contains the statistical forecast. This table is filled from the statistical forecasting process, and it is also used as basis for the judgmental forecasting by the demand planners. | ID, CREATE_MONTH, TARGET_MONTH |
| MANFC | Contains the forecasts that are finally committed by the demand planner. | ID, CREATE_MONTH, TARGET_MONTH |
| MATERIAL | Contains the product base data. | ID |

**Table 2. Data Sources for the Analysis Data Mart**

The full process of data preparation and more details about the available data is explained in Svolba 2017, Chapter 10.

| FC_ID | ID | Target_Month | Create_Month | Model | Product_Gro... | Price_Index | Launch_Mon... | Target_CalM... | Target_Year | Lead_Time | Product_Age |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 3335539 | 2009.12 | 2009.10 | LONG S5 Dow... | 8 | 180 | 1 | 12 | 2009 | 2 | 120 |
| 2 | 3335539 | 2009.12 | 2009.11 | LONG S5 Dow... | 8 | 180 | 1 | 12 | 2009 | 1 | 120 |
| 3 | 3335539 | 2010.01 | 2009.10 | LONG S5 Dow... | 8 | 180 | 1 | 1 | 2010 | 3 | 120 |
| 4 | 3335539 | 2010.01 | 2009.11 | LONG S5 Dow... | 8 | 180 | 1 | 1 | 2010 | 2 | 120 |
| 5 | 3335539 | 2010.01 | 2009.12 | LONG S5 Dow... | 8 | 180 | 1 | 1 | 2010 | 1 | 120 |
| 6 | 3335539 | 2010.02 | 2009.10 | LONG S5 Dow... | 8 | 180 | 1 | 2 | 2010 | 4 | 120 |
| 7 | 3335539 | 2010.02 | 2009.11 | LONG S5 Dow... | 8 | 180 | 1 | 2 | 2010 | 3 | 120 |
| 8 | 3335539 | 2010.02 | 2009.12 | LONG S5 Dow... | 8 | 180 | 1 | 2 | 2010 | 2 | 120 |
| 9 | 3335539 | 2010.02 | 2010.01 | LONG S5 Dow... | 8 | 180 | 1 | 2 | 2010 | 1 | 120 |
| 10 | 3335539 | 2010.03 | 2009.11 | LONG S5 Dow... | 8 | 180 | 1 | 3 | 2010 | 4 | 120 |

**Output 2. Important Input Variables for the Analysis**

Table 3 lists these variables with a short description and the measurement type. The measurement type determines the type of descriptive analysis and graph that can be used and defines how this variable is treated in the regression analysis.

| Variable Name | Description | Measurement Type |
|---|---|---|
| PRODUCT_GROUP | Product group | Category |
| PRICE_INDEX | Price index | Interval |
| LAUNCH_MONTH | Calendar month of product launch | Category |
| PRODUCT_AGE | Number of months since the article was launched | Interval |
| MODEL | Model that was used for statistical forecasting | Category |
| LEAD_TIME | Number of months in the future for which the forecast is created | Interval |
| TARGET_CALMONTH | Calendar month for which the forecast is created | Category |
| TARGET_YEAR | Year for which the forecast is created | Interval |

**Table 3. Important Input Variables for the Analysis**

## CALCULATING DERIVED VARIABLES

The creation of selected derived variables is illustrated in this section. You see selected code lines from a larger DATA step that prepares the data.

### Product and Forecast Process-Related variables

Derived variables from the date variables are created with the YEAR function and the MONTH function.

```
/*** FC-derived variables ***/
Create_CalMonth = month(create_month);
Create_Year     = year(create_month);
Target_CalMonth = month(target_month);
Target_Year     = year(target_month);
```

The LEAD_TIME and the PRODUCT_AGE are calculated as the difference between the two respective date values using the INTCK function. The INTCK function is very convenient to calculate the number of intervals (in this case, months) that lie between two date variables. Compare also Svolba 2006.

```
/*** Lead Times ***/
 Lead_Time = intck('MONTH',create_month,target_month);
 Product_Age = intck('MONTH',launch_date,target_month);
 if Product_Age > 120 then Product_Age=120;
```

### Calculating Forecast Error

You create the average percentage error, APE, by calculating the absolute difference between the forecast value and the observed value. Divide this value through the observed value to receive a percentage error.

```
/*** MAPE-Block ***/
format APE_Stat APE_Man APE_Stat_Shift APE_Man_Shift 8.1;
APE_Stat = abs(statfc - actual)/actual * 100;
APE_Man  = abs(JudgmFC - actual)/actual * 100;
```

A variable with shifted APE values is created where extreme large outliers are shifted to a lower value. Otherwise, the graphs and the regression analyses might be dominated by these outliers.

```
ape_stat_shift = min(ape_stat,300);
ape_man_shift  = min(ape_man,300);
```

### Calculating the Difference between the Manual and the Statistical Forecast

Two variables are created that describe the difference between the manual and the statistical forecast: APE_DIF and FC_DIF.

**APE_DIF** contains the difference between the average percentage error of the statistical and the manual forecast. Positive values mean that the APE of the judgmental forecast is larger.

```
APE_DIF = ape_judgm - ape_stat;
```

Extreme values beyond -500 and 500 are shifted toward -500 and 500, respectively.

```
if APE_DIF ne . and APE_DIF < -500 then APE_DIF = - 500;
else if APE_DIF > 500 then APE_DIF = 500;
```

**FC_DIF** contains the difference between the judgmental forecast and the statistical forecast.

- Positive values mean that the judgmental override increased the forecast.
- Negative values represent a decrease of the forecast through the override.

```
FC_DIF  = JudgmFC-statfc;
```

Extreme values beyond -5,000 and 5,000 are shifted toward -5,000 and 5,000, respectively.

```
if FC_DIF ne . and FC_DIF < -5000 then FC_DIF = - 5000;
else if FC_DIF > 5000 then FC_DIF = 5000;
```

## DESCRIPTIVE ANALYSIS OF THE FORECAST ERROR

This section shows that you can gain initial insight into the relationships of your data just by using descriptive methods. Only selected results are shown here. For more insight refer to Svolba 2017, Chapter 10.

### CHECKING THE DISTRIBUTION OF THE FORECAST ERROR

The mean of APE_STAT is 86.5 with a standard deviation of 585.8. The median is 40.6. Looking just at the mean and the median, it seems that the forecast quality of this company is quite bad. On average the forecast is 86.5% away from the true demand. And 50% of the forecasts have a forecast error of larger than 40.6%.

However, bear in mind that many products with a very short data history are forecasted in this business example. Products that were just put on the market do not provide a lot of insight in their demand pattern. In many of these cases, only a rough estimate can be created. Rough estimates based on fewer data points have a higher forecast error. The influence of the available history can also be seen later on when the model type or the product age is analyzed.

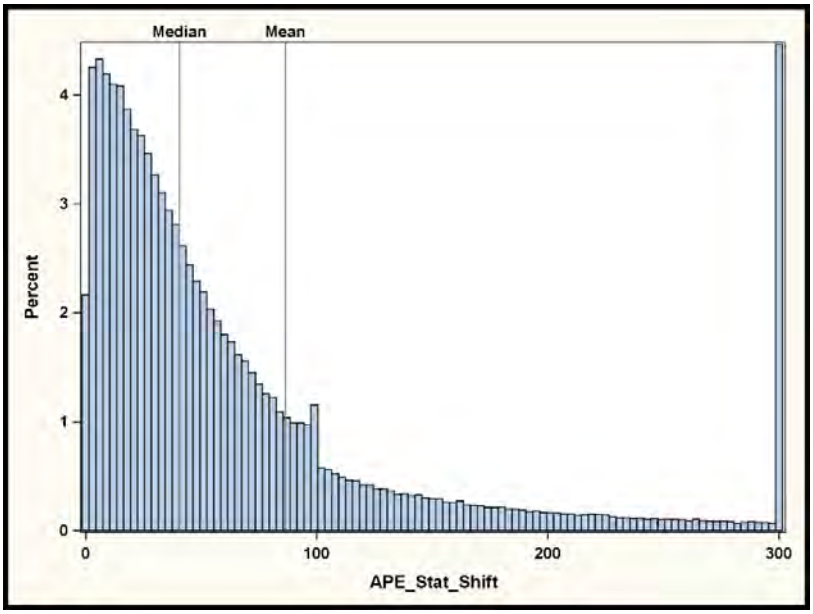
You also see that the distribution of the forecast error is heavily skewed to the right. The mean is twice as large as the median and the standard error is extremely high, due to a few outliers with extreme values. Table 4 also reveals that the maximum forecast error is higher than 230,000.

| Quantile | Value |
|---|---|
| 100% Max | 238,954.6 |
| 95% | 276.6 |
| 90% | 169.5 |
| 75% Q3 | 81.7 |
| 50% Median | 40.6 |
| 25% Q1 | 18.0 |
| 10% | 7.0 |
| 0% Min | 0 |

**Table 4. Quantiles of APE_STAT**

The statistics in this table have been created with the following SAS code.

```
proc means data=fc_mart mean std min p10 q1 median q3 p90 p95 max maxdec=1;
 var APE_Stat;
run;
```

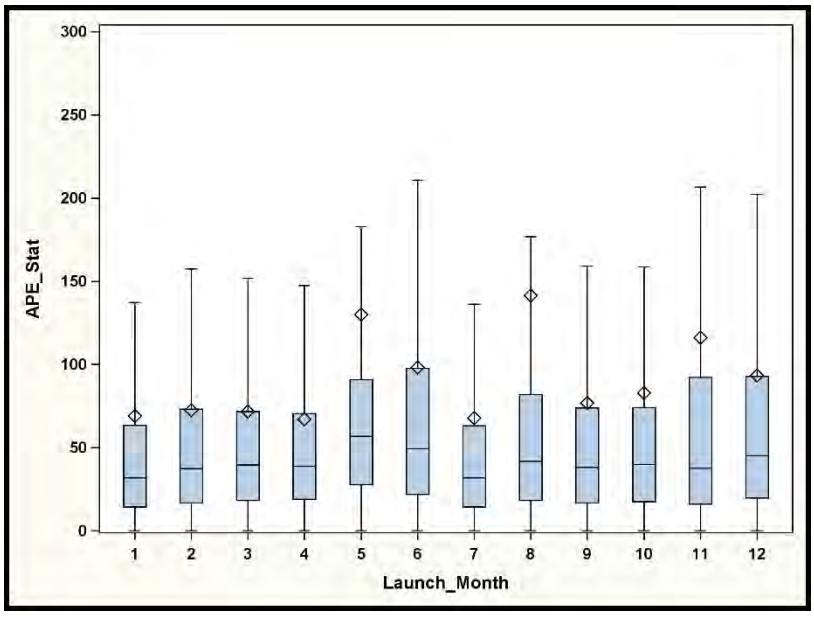



**Output 3. Histogram for APE_STAT_SHIFT**

## Calendar Month of Product Launch

Output 4 shows that the forecast quality also differs by calendar month. You see that products launched in May or June have larger forecast errors compared to products launched in July.

- This might be due to an association between the launch of product groups that are easier to forecast in certain months of the year.
- Another reason might be the interaction between the launch month and the seasonal demand pattern, like the larger demand around Christmas. Products that are launched in July might directly move from a demand peak in the launch phase to a demand peak in the pre-Christmas season.




**Output 4. Histogram for APE_STAT_SHIFT by Launch Month**

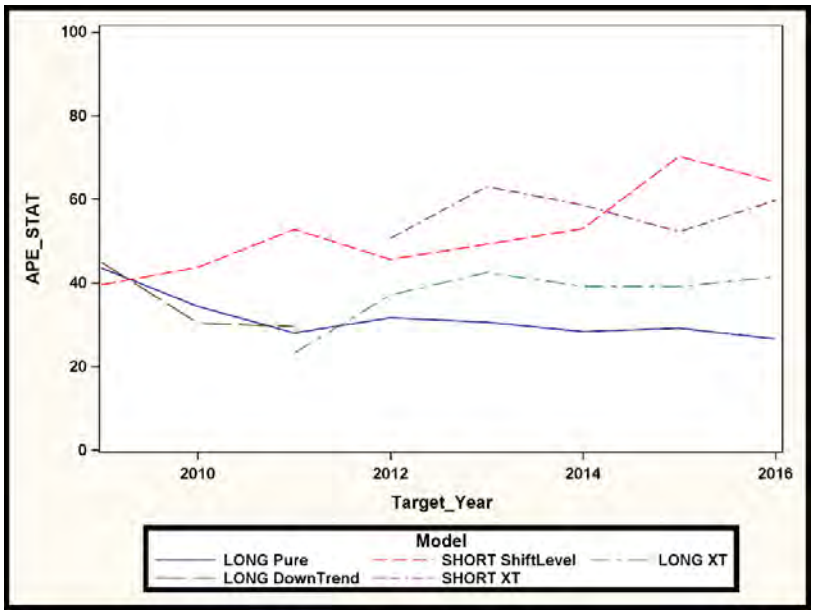
A multivariate analysis with interaction might give more insight into this question. This is shown in the subsequent sections.

## DIFFERENT FORECAST MODELS PRODUCE DIFFERENT FORECAST ERRORS OVER TIME

Before interpreting the forecast error of the different model types, you have to bear in mind that for each article, the model that generates the best forecast is selected.

- This also implies that Model A might be selected for the easy-to-forecast articles, while Model B, which is more robust, is selected for articles with a complicated demand pattern.
- In terms of average forecast quality, Model B might look bad compared to Model A, as it is mostly used to forecast complicated articles.

The result is shown in Output 5.




**Output 5. Median Course over TARGET_YEAR of Different MODEL Types**

### Interpretation of the Results

You see that there are different courses of the forecast error over time for different model types. The red dashed line and the violet double-dashed dotted line represent the median course of the forecast error of the models with short demand history.

- You see that there is a slight increase in the forecast error of the SHORT ShiftLevel model over time.
- The SHORT XT model has some variation over time, but stays stable on average.

You see that model LONG DownTrend was discontinued in 2011 and replaced by model LONG XT. It is interesting to see that only in 2011, model LONG XT has a better forecast quality. The forecast error however increases in the later years.

- This might be an indication that model LONG XT is overfitted and only fit well when it was first introduced.
- It might also be the case that the articles with a complicated demand pattern were forecast with model LONG XT. If these articles were forecast with the LONG Pure model, the average forecast

error for this model might have increases as well.

### Creating the Line Chart of the Medians

In order to display the context in a single chart, a line plot of the median forecast errors per target year and model type can be created. First you calculate the median forecast error per subgroup with the MEANS procedures and store the results in a data set APE_MEDIAN.

```
PROC MEANS DATA=fc_mart NWAY NOPRINT;
  VAR ape_stat;
  CLASS model Target_year;
  OUTPUT OUT=ape_median mean= median= /autoname;
RUN;
```

Note that the NOPRINT option suppresses printed output in the results window. NWAY specifies that only the lowest level of the subgroup hierarchy, MODEL x TARGET_YEAR, is stored in the output data set.

Next you use the SGPLOT procedure to plot the median course over target year, by model type.

```
PROC SGPLOT DATA=ape_means;
  SERIES X=Target_year Y=ape_stat_median / group=model;
  YAXIS LABEL ="APE_STAT" min=0 max=100;
RUN;
```

## THE EFFECT OF MANUAL OVERWRITES

### ORIGIN OF MANUAL OVERWRITES

In the operational forecasting process, the statistical forecast is often not used as the final forecast. Demand planners perform a judgmental correction to the statistical forecast.

- This correction is based on their personal experience with the business context.
- They might have additional information available that should influence the forecast value, like a regionally isolated marketing campaign for a certain product that is planned to run next month.
- It might also be the gut feeling of the demand planner.
- The judgmental correction might also have a political reason.
- Sometimes the values of the statistical forecast are just rounded to add a judgmental flavor to it.

### AWARENESS OF THE DEFINITION OF THE DERIVED VARIABLES

For a clear interpretation of the results, it is a best practice to show and repeat the definition of the derived variables in the comments or in the results file. This makes sure that the value and the sign of the difference can immediately be interpreted by the analyst.

In the case of the difference in percentage error, the following definition is used:

### APE_DIF = APE_MAN – APE_STAT

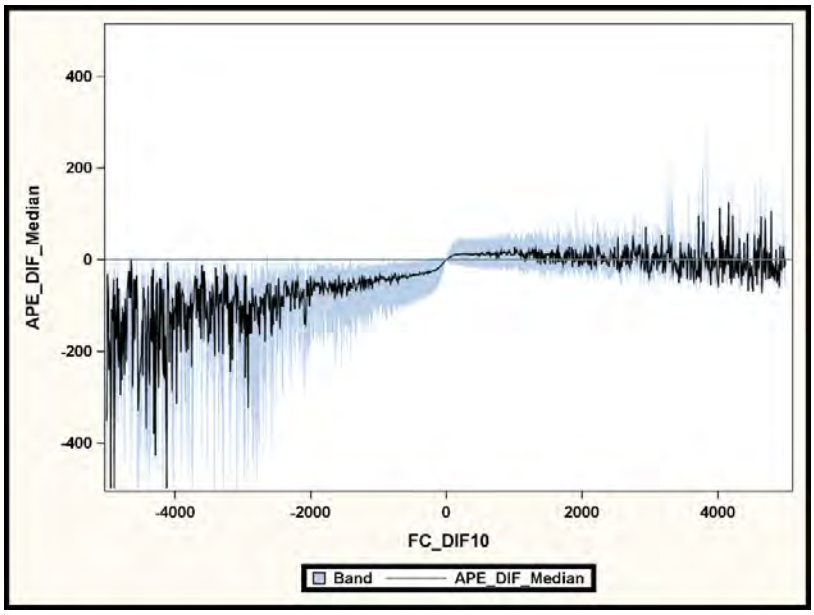
- Thus, a positive value of APE_DIF means that the forecast error got larger because of the manual correction. The manual correction did not improve the accuracy of the forecast.
- A negative value of APE_DIF means that the forecast error got smaller by the manual correction and it improved the accuracy of the forecast.

### FC_DIF = JUDGMFC – STATFC

- A positive value indicates that the manual override increased the forecast; a negative value decreased the forecast.

### THE BENEFIT OF LARGE OVERWRITES

Output 6 shows the manual override on the X axis and the change in the average percentage forecast error at the Y axis.




**Output 6. Relationship between FC_DIF and APE_DIF**

Note that the override values at the X axis are rounded to 10 in order to avoid a too busy graph. For each override value at the X axis, you can have several observations. Thus, the median and the first and third quartile are calculated.

- The median is shown by a solid black line.
- The first and third quartile are displayed by a band.

You see that larger changes of the forecast value have on average a much larger effect in decreasing the forecast error. This finding corresponds with the work of Goodwin 2009, who states that small changes to the forecast value usually do not improve forecast quality, while larger changes have a positive effect. Small changes to the forecast are made by the demand planner with less care than large changes. If a large change to the statistical forecast has to be applied, the planner investigates, in much more detail, whether the adjustment shall be made or not.

The consequence should not be to recommend large changes instead of small changes. Rather, the goal is to eliminate the small changes that do not add any benefit and save the time for analyzing whether large changes shall be made.

Again you see that a decrease of the forecast through the manual overwrite had on average a positive effect. You however also see that even large positive changes have a positive effect on the forecast quality. Demand planners obviously put more thought into large changes and apply them only if they are really convinced about it.

## Creating the Line Chart and the Band Chart with SGPLOT Procedure

The above graph has been created in two steps. First the median and the first and third quartile have been calculated with the MEANS procedure. Note that the AUTONAME option has been used to automatically create the respective variable names that contain the name of the descriptive statistic:

```
proc means data=fc_mart noprint nway;
 class fc_dif10;
 var ape_dif;
 output out= dif10_mean median= q3= q1= / autoname;
run;
```

Next the data is plotted with the SGPLOT procedure. Note that the BAND statement should precede the SERIES statement for the mean value. Otherwise, the line hides behind the band.

```
proc sgplot data=dif10_mean;
 band x=fc_dif10 lower=ape_dif_q1 upper=ape_dif_q3;
 series x=fc_dif10 y=ape_dif_median;
 refline 0 / axis=y;
run;
```

## QUANTIFYING THE EFFECT OF DRIVERS FOR THE FORECAST ERROR WITH THE GLMSELECT PROCEDURE

### OVERVIEW

OLS regression enables you to quantify the effect of each explanatory factor, like PRODUCT_AGE or PRODUCT_GROUP on the forecast error. You can run a univariate regression with only one input variable for each influential factor.

This provides insight into the explanatory power of the respective variable on the forecast error. It also allows quantifying this relationship using the regression coefficients.

- If you use an interval input variable, you receive one coefficient.
- If you use a categorical input variable, you receive a coefficient for all categories expect the reference category.

### Interpretability versus Statistical Correctness

As in many business analyses, the decision between interpretability and applicability of the results and the statistical correctness needs to be made. From a statistical point of view, the target variable APE_STAT should be definitely log transformed before being used in the regression model.

The price that is paid in this case is that the regression coefficients cannot then be interpreted in units of the target variable. Svolba 2016 shows in more detail that the model fit of the model using the log-transformed variable is not better than those of the untransformed variable. In this case, to be on the safe side, it is better to leave the variable untransformed for better interpretability.

It always makes sense, however, to check both models for their model fit. This enables you to see how much the fit between the two modeling approaches differs.

### UNIVARIATE ANALYSIS USING THE GLMSELECT PROCEDURE

The following code example shows how you can perform this analysis for using the GLMSELECT procedure for an interval variable and a categorical variable.

```
PROC GLMSELECT DATA=fc_mart;
 MODEL ape_stat_shift =  product_Age  /  SHOWPVALUES;
RUN;

PROC GLMSELECT DATA=fc_mart;
 CLASS product_group / PARAM=effect;
 MODEL ape_stat_shift =  product_group / SHOWPVALUES;
RUN;
```

Table 5 shows the available input variables ordered by descending $R^2$. You see that variable MODEL, PRODUCT_AGE, and PRODUCT GROUP are the most influential variables.

| Ranking | Input Variable | R-squared linear | Beta linear |
|---|---|---|---|
| 1 | MODEL | 0.0554 | |
| 2 | PRODUCT_AGE | 0.0433 | -0.51 |
| 3 | PRODUCT_GROUP | 0.0224 | |
| 4 | LAUNCH_MONTH | 0.0172 | |
| 5 | TARGET_YEAR | 0.0102 | 4.16 |
| 6 | TARGET_CALMONTH | 0.0084 | |
| 7 | LEAD_TIME | 0.0046 | 1.68 |
| 8 | PRICE_INDEX | 0.0016 | -0.02 |

**Table 5. Input Variables Sorted by Descending Adjusted R-Square**

The coefficient of PRODUCT_AGE of -0.51, for example, can be interpreted as the average decrease in forecast error for each additional month of demand history. You can conclude that an additional year of forecast history results on average in a decrease of around 6 percentage points (0.51 times 12 months).

### Variables Model Type and Product Age

You see that the two top variables are model type and product age with an $R^2$ in the linear model of 5.54% and 4.33%, respectively. Variable model type implicitly also contains information about the product age, as the models are separated by short and long data history.

The fact that the explanatory power of the variable model type is higher than those of variable product age indicates that the model type contains more information than just the length of the available data history.

In a multivariate regression model, it is interesting to see whether both variables are still selected or whether the additional explanatory power of the second variable is not high enough to cause the second variable to be added to the model. Using a multivariate regression model enables you to investigate the relative importance of a variable compared to the fact that other variables are already in the model.

## MULTIVARIATE ANALYSIS OF THE INFLUENCE ON THE FORECAST ERROR

### Code of GLMSELECT Procedure

The following code has been used to perform a multivariate regression analysis with stepwise selection of the input variables:

```
PROC GLMSELECT DATA=fc_mart;
 CLASS product_group launch_month model target_calmonth / PARAM=effect ;
 MODEL ape_stat_shift =
                 product_group|price_index|launch_month|product_age|
                 model|lead_time|target_calmonth|target_year_shift     @1
                     /DETAILS=steps
                      SELECTION=stepwise (SELECT=sl)
                      ORDERSELECT
                      SHOWPVALUES;
RUN;
```

Note the following from the code:

- The CLASS statement, which lists all four categorical variables, is used and the EFFECT coding is requested with the PARAM option.
- The MODEL statement contains the list of input variables. Note that the list of variables could also be specified with blanks between the variables.
- Using the “pipe” | has the advantage that the MODEL statement can be used in a flexible way if, for

example, quadratic terms shall be requested.

- @1 indicates that you want to use these variables only to the power of 1.
- @2 would cause the normal effect and the quadratic effect for each variable.
- This feature is not limited to the GLMSELECT procedure; it can be applied for all regression procedures with MODEL statements.
- A stepwise regression is requested with the SELECTION= stepwise option.
- The SL option specifies that the significance level of each variable to enter or leave the model shall be checked.
- Information about each forecasting step is requested with the DETAILS= steps option.
- The option ORDERSELECT causes the parameters in the final parameter estimates table to be sorted in the order of their inclusion into the model, instead of alphabetic order.
- The SHOWPVALUES options requests that p-values are shown in the parameter estimates table.

## Results of the Multivariate Regression

Table 6 shows the list of input variables in their selection order for the linear regression for the non-transformed target error. You see that all available eight variables are selected, even if the last variable only marginally contributes to the improvement of the model fit. This is also due to the large number of observations (> 400,000 records).

| Ranking | Input Variable | Adjusted R-square |
|---|---|---|
| 0 | INTERCEPT | 0% |
| 1 | MODEL | 5.46% |
| 2 | TARGET_CALMONTH | 6.58% |
| 3 | PRODUCT_GROUP | 7.59% |
| 4 | TARGET_YEAR_SHIFT | 8.50% |
| 5 | PRODUCT_AGE | 9.04% |
| 6 | LEAD_TIME | 9.76% |
| 7 | LAUNCH_MONTH | 9.90% |
| 8 | PRICE_INDEX | 9.90% |

**Table 6. Input Variables Sorted by Adjusted R-Square of the Multivariate Model**

The first variable MODEL has been selected, which adds 5.46% of the explanation of the values in variable APE_STAT_SHIFT. Note that in Table 5 you have also seen variable MODEL on top of the list ordered by their univariate contribution, so its selection is intuitive.

## Additional Information Is Prioritized

In step 2, however, variable TARGET_CALMONTH has been selected, although it was only at rank 6 of the ordered list in Table 5. It can be assumed that it “overtook” the other variables, because after variable MODEL was selected, the additional explanatory power of the variable TARGET_CALMONTH was higher than that of the others.

At rank 2 of the univariate analysis in Table 5, you saw PRODUCT_AGE. In the multivariate model it is selected only in the fifth step. In the multivariate regression model, the variables are considered in a combined or simultaneous way.

As variable MODEL is already in the regression equation, the additional explanatory power of variable PRODUCT_AGE is not that high anymore. Variable MODEL has already “told” part of its information, for example, that older products, forecasted with “LONG-models” have a lower forecast error than younger products, forecasted with “SHORT-models”.

### True Increase of Model Fit

Thus, the relative benefit of variable PRODUCT_AGE is not 4.75% as in the univariate model but only 0.54% (9.04 – 8.50). Variables TARGET_CALMONTH, PRODUCT_GROUP, and TARGET_YEAR are selected first as they obviously can “tell new details”.

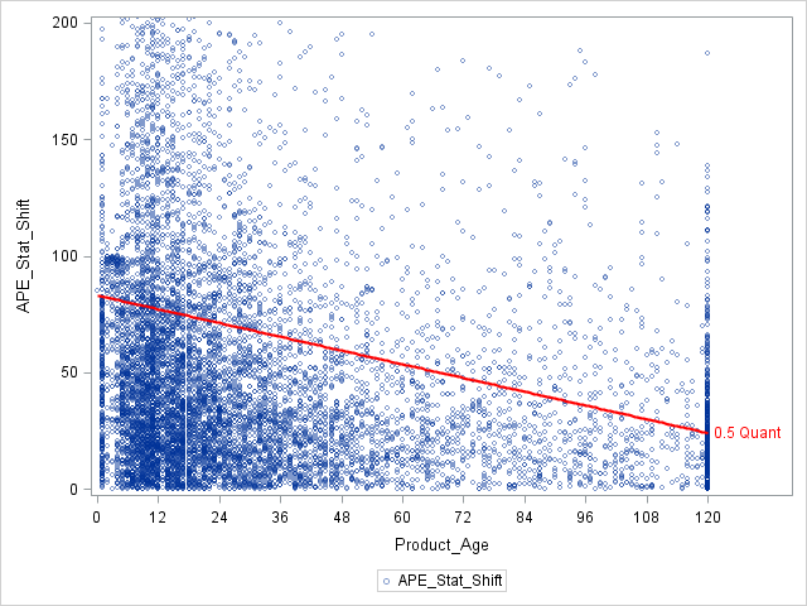
You also see that the additional explanatory power of PRODUCT_GROUP is not 2.24% as shown in Table 5 for the univariate results. It is only around 1%. This indicates that variables MODEL and TARGET_CALMONTH have already contributed more than half of what variable PRODUCT_GROUP could contribute in a univariate model.

## STUDYING THE REGRESSION RESULTS VISUALLY

### Univariate Analysis of PRODUCT AGE

Output 7 shows the plot of the predicted APE_STAT values from a univariate OLS regression with variable PRODUCT_AGE. The actual values are plotted as blue circles. The predicted values are plotted as a solid red line. You see a decreasing trend of the forecast error over the increasing values of product age.

This result corresponds with the findings shown Table 3. A larger data history for a product decreased the forecast error.




**Output 7. Plot of the Predicted APE_STAT Values from the Univariate Regression Model**

### Multivariate Analysis of Product Age Provide More Insight

Output 8 shows the same plot, however, based on the predicted values of a more detailed regression model. In this model all selected variables have been included. (Compare Table 6.)

You still see a downward trend of the forecast error over product age. However, the relationship is no longer a straight line as the effect of product age is not only measured on its own. It is corrected for the effect of the other available variables.

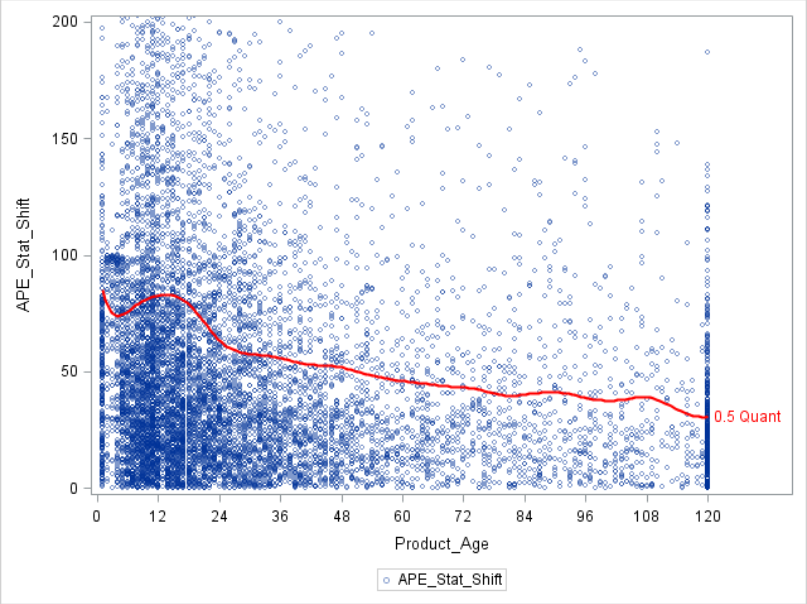
Thus, Output 8 shows the effect of the variable PRODUCT_AGE after correcting for the influence of other co-variables.

This provides much more detailed insight in the effect of variable product age. All other co-influences have already been filtered and this enables you to view only the remaining effect of variable product age.

- You see an interesting drop of the forecast error around month 3, which is hard to explain from a

business point of view.

- You see that additional months of data history have the strongest effect between months 18 and 24, this is when a second full year of data history is achieved.
- You see a rather linear decreasing trend after month 24.




**Output 8. Plot of the Predicted APE_STAT Values from the Multivariate Regression Model**

## SAS Code

The code for the multivariate analysis is shown here. Note that an OUTPUT statement is added to the GLMSELECT procedure to output the predicted values from the regression model. These predicted values are then used in the SGPLOT procedure.

```
PROC glmselect DATA=FC_Mart_10smp ;
  partition rolevar=_ROLE_ (train = 'TRN' validate='VAL');
  CLASS product_group launch_month model target_calmonth / PARAM=effect ;
  MODEL ape_stat_shift = product_group|price_index|launch_month|
                         product_age|model|lead_time|
                         target_calmonth|target_year @1
                    /selection=stepwise(choose=validate slentry=0.001);;
    output out=LinReg1Pred p=APE_STAT_PRED
RUN;
```

The SGPLOT procedure is used to combine a SCATTER plot for the actual data value with a SPLINE plot for the predicted values.

```
proc sgplot data=LinReg1Pred;
 scatter y=ape_stat_shift x=product_age/
             markerattrs=(size=5) transparency=0.5
             filledoutlinedmarkers;
 pbspline y=APE_STAT_PRED x=product_age/
             lineattrs=(thickness=2 color=red) nknots=20 nolegfit
             curvelabel="0.5 Quant" curvelabelattrs=(color=red) nomarkers;
   xaxis values= (0 to 120 by 12) ;
   yaxis max=200;
   where product_age ne 0;
run;
```

# GETTING ADDITIONAL INSIGHT WITH QUANTILE REGRESSION

## BASIC IDEA OF QUANTILE REGRESSION

### Idea of Linear Regression

With a linear regression model, as presented in the previous section, an important implicit assumption is made: The conditional mean of the dependent variable is modeled.

In linear regression, the model equation is:

$$Y_i = x_i'\beta + \varepsilon_i$$

And the vector β is determined by minimizing the errors:

$$\min \sum_{i=1}^{n} \varepsilon_i^2$$

In many cases the conditional mean is what should be modeled and predicted and not a lot of thought is put into that fact. There are, however, some cases where you are not interested in a model that explains or predicts the conditional mean of the distribution of the dependent variable, but you are rather interested in specific quantiles.

Ordinary least squares regression models the relationship between one or more covariates X and the conditional mean of the response variable Y given X=x. Quantile regression extends the idea of regression models to conditional quantiles of the response variable, such as the 90th percentile (0.9 quantile).

### Quantile Regression

Here the quantile regression comes into play. It allows you to model selected conditional quantiles. You receive a model that predicts the value of a certain quantile instead of the mean. The model equation for a quantile $\tau$ is the following:

$$Q(\tau|X = x) = x'\beta(\tau) + \varepsilon_i$$

Here the following expression is minimized,

$$\min \sum_{i=1}^{n} \rho_\tau|\varepsilon_i| + \sum_{i=1}^{n} (1 - \rho_\tau)|\varepsilon_i|$$

where $\rho_\tau|\varepsilon_i|$ and $(1 - \rho_\tau)|\varepsilon_i|$ are the penalty terms for over and under estimation.

In the case of the forecast errors analysis, this might answer the following business questions: “What influences the 1st quartile of the target variable?”

- Do you want to see the predictors for a better quarter of the forecast errors?
- Quantile regression shows you the list of variables and their parameters that are related to the forecast error value, which is not exceeded by the 25% of time series with the lowest forecast error.

You can perform the same procedure for the 3rd quartile to get insight on those variables that relate to the upper range of the forecast errors.

- You also might want to know whether the list of influential factors for the upper and the lower quarter of forecast error differs.

## QUANTILE REGRESSION FOR THE STATISTICAL FORECAST ERROR

## Rationale of Using Quantile Regression

Quantile regression allows you to better understand the influence of independent variables for different quantiles of the statistical forecast error. The 0.75 quantile regression enables you to identify and parameterize those factors that influence whether a certain forecast error is not exceeded by 75% of the time series.

Linear regression only deals with the analysis of the average location of the forecast error and the variables that influence this average error.

Quantile regression enables you to make assumptions about the extreme areas. “Which forecast error is not exceeded, if I have data history of more than 24 months?” From a business point of view, the information that “75% of the time series have a forecast error smaller than x” is more important than the average forecast error.

The same applies to the 0.25 quantile: You receive information about the forecast error that is not exceeded by your best 25% of the time series, if you increase the available time history.

## Quantile Regression for Selected Quartiles

In this example, the quantile regression for the 0.1, 0.25, 0.5, 0.75 and 0.9 quantile of the statistical forecast error is performed using the following SAS code:

```
PROC QUANTSELECT DATA=fc_mart_10smp;
  CLASS product_group launch_month model target_calmonth / PARAM=effect ;
  MODEL ape_stat_shift = product_group|price_index|launch_month|
                         product_age|model|lead_time|
                         target_calmonth|target_year @1
                    /quantile= 0.1 0.25 0.5 0.75 0.9
                     selection=stepwise(choose=validate slentry=0.001);
  ods select SelectionSummary;
RUN;
```

Note that the QUANTILE option in the MODEL statement is used to specify the quantiles of interest.

## Variables Selected for Different Quantiles

Tables 7–9 show the variables that have been selected by different quantile regressions. You see that these sets differ. This indicates that for different quantiles, different combinations of influential factors are relevant.

- You see that variables model type, product age, product group, and lead time are selected in every model.
- You also see that the model for the 0.25 quantile uses a smaller set of variables than the model for the 0.5 and the 0.75 quantiles.
- You could also study the coefficients of the parameter for each model. In that case, you would see that the coefficients differ for each model, even if the same set of variables is selected.

| Step | EffectEntered | SBC |
|---|---|---|
| 1 | Model | 36293.1995 |
| 2 | Product_Group | 36356.7482 |
| 3 | Lead_Time | 36356.6815 |
| 4 | Product_Age | 36330.2975 |

**Table 7. Variables Selected for the 0.25 Quantile**

| Step | EffectEntered | SBC |
|---|---|---|
| 1 | Model | 43049.0936 |
| 2 | Launch_Month | 43118.2647 |
| 3 | Lead_Time | 43087.3543 |
| 4 | Product_Group | 43094.7320 |
| 5 | Product_Age | 43077.2793 |
| 6 | Target_CalMonth | 43123.6095 |
| 7 | Target_Year | 43125.2832 |

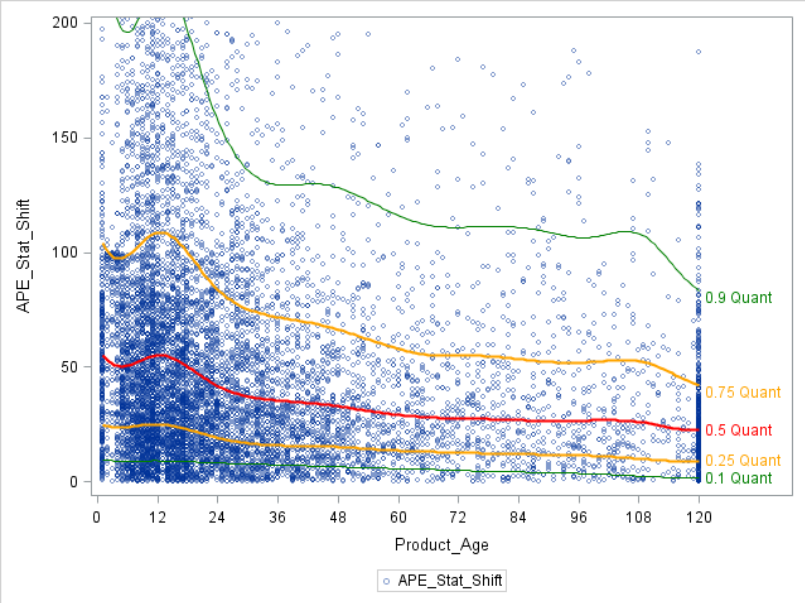
**Table 8. Variables Selected for the 0.5 Quantile**

| Step | EffectEntered | SBC |
|---|---|---|
| 1 | Model | 44077.9745 |
| 2 | Target_CalMonth | 44027.8505 |
| 3 | Launch_Month | 44054.7556 |
| 4 | Lead_Time | 43978.5909 |
| 5 | Product_Age | 43929.8234 |
| 6 | Product_Group | 43931.3299 |
| 7 | Target_Year | 43882.6948 |

**Table 9. Variables Selected for the 0.75 Quantile**

## Displaying the Results Visually

Output 9 displays the results of the multivariate quantile regression in the same way as shown in Output 8 for the OLS regression.




**Output 9. Plot of the Predicted APE_STAT Values from the Multivariate Quantile Regression Model**

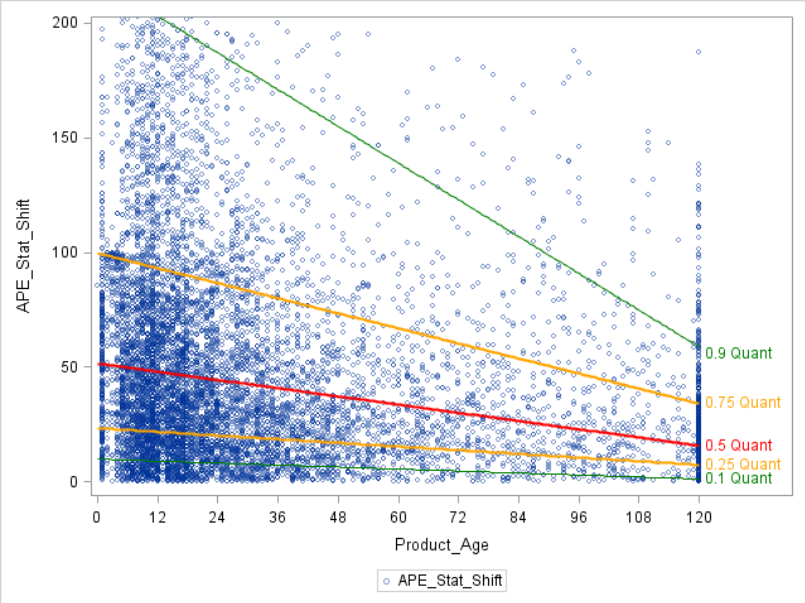
The blue circles represent a scatter plot for PRODUCT_AGE and APE_STAT of the actual data. You see solid lines in different colors for the predicted values of APE_STAT from the multivariate quantile regression for different quantiles. The relationship of product age, however, is not only measured on its own. It is corrected for the effect of the other available variables as multivariate regression model is used.

- It is remarkable to see that for the 0.1 and the 0.25 quantile of the forecast error (the value that is not exceeded by your top time series) the relationship between the product age and forecast error is very flat.
- You see that the increase in forecast errors around months 12 (which is hard to explain from a business point of view) increases with larger quantiles. It is almost not present for quantile 0.25 and 0.5.
- You also see that additional data history after month 36 has only a marginal effect, except for quantile 0.9, where additional time history does matter for higher product age values.

### Results from Univariate Quantile Regression for Variable PRODUCT_AGE

Output 10 shows the results from a univariate quantile regression model that uses only variable PRODUCT_AGE. Similar to Output 7, you see only linear trends, as the influence is not corrected for other available variables. You clearly see that the slope for the trend curve is different for the different quantiles. Larger quantiles have a steeper curve than the lower quantiles.

This leads to the interpretation that additional available data history has a much stronger positive effect on the higher quantiles than the lower quantiles. The times series with a smaller forecast error do not benefit as much from additional data history as those that are in general harder to predict.




**Output 10. Plot of the Predicted APE_STAT Values from the Univariate Quantile Regression Model**

### Create the SCATTER and SPLINE Plot

The plot is generated in a similar way as shown above. Here you use a separate PBSPLINE statement for each quantile.

```
proc sgplot data=QuRegPred;
 scatter y=ape_stat_shift x=product_age/
         markerattrs=(size=5)   transparency=0.5 filledoutlinedmarkers;
 pbspline y=APE_STAT_PRED1 x=product_age/
        lineattrs=(thickness=1 color=green ) nolegfit
        curvelabel="0.1 Quant" curvelabelattrs=(color=green) nomarkers;
 pbspline y=APE_STAT_PRED2 x=product_age/
       lineattrs=(thickness=2 color=orange) nolegfit
      curvelabel="0.25 Quant" curvelabelattrs=(color=orange)  nomarkers;
 pbspline y=APE_STAT_PRED3 x=product_age/
```

```
       lineattrs=(thickness=2 color=red) nolegfit
       curvelabel="0.5 Quant" curvelabelattrs=(color=red) nomarkers ;
 pbspline y=APE_STAT_PRED4 x=product_age/
       lineattrs=(thickness=2 color=orange) nolegfit
       curvelabel="0.75 Quant" curvelabelattrs=(color=orange) nomarkers;
 pbspline y=APE_STAT_PRED5 x=product_age/
       lineattrs=(thickness=1 color=green ) nolegfit
       curvelabel="0.9 Quant" curvelabelattrs=(color=green) nomarkers;
   xaxis values= (0 to 120 by 12) ;
   yaxis max=200;
run;
```

## CREATING A PROCESS PLOT FOR THE PARAMETER ESTIMATES

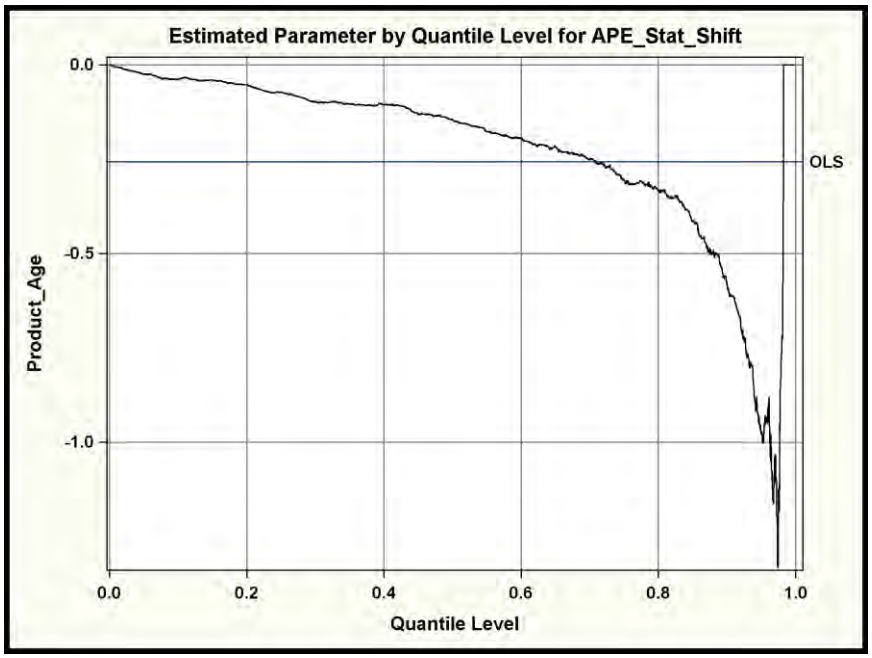
### General Idea

Output 10 shows that with increasing quantiles, the coefficient of variable PRODUCT_AGE changes. This "quantile process" can be easily analyzed with the QUANTSELECT and the QUANTREG procedure. However, only the QUANTREG procedure enables you to create the process plot that enables you to visually study the effects.

A quantile process means that a quantile regression for all quantiles from 0 to 1 is performed. The results enable you to study the relationship of an independent variable on a target variable across the quantiles. This provides insight into the influence of the independent variable on different quantile levels.

### Process Plot and Interpretation

The process plot for variable product age is shown in Output 11.




**Output 11. Process Plot for PRODUCT_AGE**

- From the plot, you see the range of quantiles on the horizontal axis and the parameter estimate for product age on the vertical axis.
- The value for the ordinary least squares estimator is shown as a horizontal line and is labeled as OLS. Note that the OLS estimate for the (conditional) mean is constant around -0.3. It does not

depend on the quantile level.

You see that the estimate in the quantile regression for parameter product age decreases over the quantile process.

- For larger quantiles of the forecast error, the estimate of product age decreases to -1. Here an additional month decreases the forecast error by 1 percentage point.
- One additional month of product age decreases the conditional 0.2 quantile of the forecast error by around 0.1 percentage points.
- One additional month of product age decreases the conditional 0.8 quantile of the forecast error by around 0.35 percentage points.

You learn that the influence of product age is not constant over the quantile process. Higher quantiles of the forecast error benefit much more (by getting smaller) from an additional month of product age available in the data.

### Using the QUANTREG Procedure

The QUANTREG procedure can be used to create a quantile process and the process plot with the following code:

```
PROC QUANTREG data=fc_mart_smp10000 algorithm=simplex ;
 CLASS Product_Group  model  ;
 MODEL ape_stat_shift =  product_group Product_Age model target_year_shift
                        /   QUANTILE=process
                            PLOT=quantplot(Product_Age) /UNPACK OLS
                             ;
RUN;
```

Note that the shifted average percentage error is used here. Otherwise, the outliers in the 0.99 and 1.00 percentile would also cause extreme values in the estimate of product age for these quantiles.

- You specify the value PROCESS with the QUANTILE = option to request the analysis of the quantile process for quantiles from 0 to 1.
- The PLOT = QUANTPLOT option requests the quantile plot for PRODUCT_AGE.
    - The UNPACK option creates an individual process plots.
    - It enables you to specify the OLS option that shows the ordinary least squares estimate as a horizontal line in the plot for comparison.

## RUN TIMES, SAMPLING, AND DATA PARTITIONING

### SAMPLING THE DATA FOR QUANTILE REGRESSION

The following run times are measured on 4 core Intel i7 processor with 2.3 GHz. Running stepwise linear regression on the entire data set of 411743 observations takes 4 seconds. Running stepwise quantile regression for the median on a sample of 10.000 observations takes 29 seconds real time and 1 minute 16 seconds CPU time. Note that the real time differs from the total CPU time as the procedure distributes computing across the available nodes.

This result shows that quantile regression is very compute intense. It thus makes sense to sample the available data. SAS®9 provides the SURVEYSELECT procedure to sample the data.

```
proc surveyselect data=FC_MART method=srs smpsize=10000
                  seed=60502    out=FC_Mart_10smp;
run;
```

You specify method SRS for simple random sampling. The SEED option enables you to fix the seed for the random sampling to generate reproducible results with every run.

## PARTITIONING THE DATA

In order to partition the data into training and validation data, you can use the PARTITION statement in the GLMSELECT or QUANTSELECT procedure. Here you can either split the data randomly or split the data according to a predefined ROLE variable.

Using a predefined ROLE variable makes sense if you want to use a fix seed for the portioning. As in the case so sampling, fixing the seed makes sense if you want to generate reproducible results with every run.

The PARTITION statement in the GLMSELECT or QUANTSELECT procedure does not enable you to specify the seed. Using a SAS DATA step you can use a fixed seed to partition the data.

```
data FC_Mart_10smp;
 set FC_Mart_10smp;
 format _ROLE_ $3.;
 call streaminit(seed=2311);
 if rand('Uniform') > 0.3 then _ROLE_ = 'TRN';
 else _ROLE_ = 'VAL';
run;
```

You initialize the random number generation with the CALL STREAMINIT statement, where you specify the seed. You assign the TRAIN or VALID role depending on the values of the random number generated with the RAND function.

SAS Viya provides the REGSELECT procedure for linear regression and the QTRSELECT procedure for quantile regression.

## LINEAR AND QUANTILE REGRESSION WITH SAS VIYA

### Sampling and Partitioning the Data

SAS Viya allows distributed high performance computing for large-scale data. SAS Viya provides the REGSELECT procedure for ordinary least squares regression and the QTRSELECT procedure for quantile regression.

SAS Viya provides very good performance. If you still want to sample your data in SAS Viya, you can use the PARTITION procedure in a similar way as shown above with the SURVEYSELECT procedure.

```
proc partition data=FC_MART samppct=10 seed=60502;
   output out=FC_Mart_10smp;
run;
```

Different to SAS9 the PARTITION statement in the SAS Viya QTRSELECT procedure enables you to specify a seed and thus fix the partitioning for reproducible results.

### The QTRSELECT procedure

The following code shows how the QTRSELECT procedure can be called in SAS Viya to run the same quantile regression as presented in the previous chapter. You see that the code is very similar to the code used for the QUANTSELECT procedure.

```
PROC qtrselect DATA=FC_Mart_10smp ;
 partition fraction(validate=0.3 seed=2311);
 CLASS product_group launch_month model target_calmonth / PARAM=effect ;
 MODEL ape_stat_shift =
                  product_group|price_index|launch_month|product_age|
                  model|lead_time|target_calmonth|target_year_shift @1
                    /quantile= 0.1 0.25 0.5 0.75 0.9;
    selection method=stepwise(choose=validate slentry=0.001)  ;
    output out=cas1.QuRegPred
```

```
            copyvars=(price_index product_age ape_stat_shift)
            p=APE_STAT_PRED
            role=Role;
    ods output selectionsummary=work.selectionsummary;
RUN;
```

Note that when using the OUTPUT statement, you have to explicitly specify the variables that shall be copied to the output data set.

## CONCLUSION

You have seen that the application of analytical methods provides many relevant insights to help you make better business decisions. This is not only the case for the analysis of the forecast error as presented in this paper, but also for many other business questions. Svolba 2016 presents a collection of examples and SAS code where relevant business questions are analyzed with analytical methods.

In the example presented here, you have seen that the descriptive method also provides a lot of insight. Using linear regression enables you to better quantify the importance of different influential factors and to assess the strength and the direction of different categories. You see that the multivariate analysis provides a more comprehensive picture than the isolated univariate analysis of influential factors.

Quantile regression enables you get a clearer picture about the extremes of your distribution. You learn which influential factors trigger the fact that forecast errors do not exceed a certain limit. In the above example you have seen that some variables are important to explain the higher quantiles of the outcome but not the lower quantiles of the outcome.

The SAS platform with SAS9 and SAS Viya procedures provides a comprehensive set of analytical methods that enable you gain more insight in the relationships between your data and your business processes.

## REFERENCES


Svolba, Gerhard. 2017. *Applying Data Science: Business Case Studies Using SAS®*. Cary, NC: SAS Institute Inc.

Fildes R., P. Goodwin, M. Lawrence, and K. Nikolopoulos K. 2009. “Effective Forecasting and Judgmental Adjustments: An Empirical Evaluation and Strategies for Improvement in Supply-Chain Planning.” *International Journal of Forecasting* 25(1): 3–23 (DOI: 10.1016/j.ijforecast.2008.11.010).

Gilliland, M. 2010. *The Business Forecasting Deal*. Hoboken, NJ: Wiley.

Svolba, Gerhard. 2006. *Data Preparation for Analytics Using SAS®*. Cary, NC: SAS Institute Inc. Available http://www.sascommunity.org/wiki/Data_Preparation_for_Analytics.

Svolba, Gerhard. 2012. *Data Quality for Analytics Using SAS®*. Cary, NC: SAS Institute Inc. Available http://www.sascommunity.org/wiki/Data_Quality_for_Analytics.


## APPENDIX

### THE %CALC_REFERENCE_CATEGORY MACRO

#### Introduction

The %CALC_REFERENCE_CATEGORY macro enables you to calculate the “hidden” coefficients of the reference category in dummy coding when the EFFECT parameterization has been used.

When using the EFFECT parameterization, the coefficient of the reference group (the “missing coefficient”) can be calculated by summing the coefficients of the other categories and changing the sign. This is also referred to as the negative sum of the coefficients of the other categories.

Although it is technically possible to perform these calculations by hand, it is more convenient and efficient to use a program to do this automatically.

### Prerequisites for the Macro

The macro has the following prerequisites.

- In the model, the EFFECT coding has been used for the creation of the dummy variables.
- The dummy variables have been automatically created using the CLASS statement. This is, for example, possible in the GLMSELECT and the QUANTSELECT procedure.
- Note that the DMREG procedure in SAS Enterprise Miner also creates dummy variables based on EFFECT coding. However, the macro has not been tested for the output of the DMREG procedure.
- The parameter estimates file contains the p-value for each parameter. This can be requested with the option SHOWPVALUES in the MODEL statement.
- The macro assumes that the input table that is used in the macro call has been created using the ODS OUTPUT statement in the respective regression procedure.
- The ODS objects PARAMETERESTIMATES and CLASSLEVELINFO can be created with the following ODS OUTPUT statement in the respective regression procedure:

```
ODS OUTPUT ParameterEstimates= ParameterEstimates
           ClassLevelInfo    = ClassLevelInfo;
```

### Limitations of the Macro

The current version of the macro has the following limitations:

- The categories of the CLASS variables must not contain blanks. For example, a category value “Model 1” is invalid. It needs to be transformed, for example, to “Model1” or “Model_1” before the regression analysis is run.
- Note that the macro ignores interaction terms in the output table. The reason for this is that the effect names that are created from the interaction terms are often abbreviated and cannot be reproduced by the macro from the ClassLevelsList.

### Macro Parameters

The following parameters can be specified with the macro.

- **ParmEst**: The name of the data set that contains the ParameterEstimates, created with the ODS OUTPUT statement. Default = ParameterEstimates.
- **ClassLevels**: The name of the data set that contains the ClassLevelInfo, created with the ODS OUTPUT statement. Default = ClassLevelInfo.
- **OutputDS**: The name of the data set that shall contain the output data set. Default = _ParmEst_XT_.

Refer to Svolba 2016, Chapter 12 for a comprehensive explanation of the macro and its functionality. The macros can be downloaded here: http://www.sascommunity.org/wiki/Data_Quality_for_Analytics_--_Download_Page.

## ACKNOWLEDGMENTS

Many people have helped and inspired me to write and to complete this paper: Bob Rodriguez, Paul Goodwin, Mike Gilliland, Mihai Paunescu, Albert Tösch, and Robin Langford.

## RECOMMENDED READING

Svolba, Gerhard. 2006. “Efficient ‘One-Row-per-Subject’ Data Mart Construction for Data Mining.” *Proceedings of the Thirty-First Annual SAS Users Group International Conference*. Cary, NC: SAS Institute Inc. Available http://www2.sas.com/proceedings/sugi31/078-31.pdf.

Svolba, Gerhard. 2015. “Want an Early Picture of the Data Quality Status of Your Analysis Data? SAS® Visual Analytics Shows You How.” *Proceedings of the SAS Global Forum 2015 Conference.* Cary, NC: SAS Institute Inc. Available http://support.sas.com/resources/papers/proceedings15/SAS1440-2015.pdf.

## CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Gerhard Svolba
SAS Institute Inc. Austria
Mariahilfer Strasse 116, A-1070 Wien
Email: mailto: Sastools.by.gerhard@gmx.net
Web: http://www.sascommunity.org/wiki/Gerhard_Svolba

# Monitoring Forecast Models Using Control Charts

JOSEPH H. KATZ

**PREVIEW** *In this article, Joe Katz presents a new application of control charts for automatic monitoring of forecast errors. He reviews the traditional rules for statistical process control and then applies these along with custom rules to determine whether a forecasting model should be maintained, adjusted/refit, or discarded and replaced.*

## INTRODUCTION

When should a forecasting model be adjusted or replaced? This is an important consideration, because an inappropriate model results in avoidable forecast bias and error and can consume management resources making manual forecast adjustments.

Until now, there has not been an automated capability for monitoring forecast-model fitness for each time series to determine when models need to be discarded, adjusted, or left unchanged. Automating this process becomes even more critical when the number of time series is very large. The two main questions that need to be answered are

- Is it possible to evaluate forecast-model suitability and viability for each individual time series as time passes, and as new data points are added to the historical data?
- Can this forecast monitoring methodology be automated?

The answer to both questions is yes! The approach described in this article demonstrates that it is possible to evaluate and automate the monitoring of forecast-model fitness using control charts. Please note that this methodology is patented and owned by SAS Institute (Katz, 2015).

## BACKGROUND

Today automatic forecasting software is available. These software tools will develop candidate models, incorporate events and independent variables, select the "best" model with optimized parameters, and generate forecasts—all without the need for human intervention. Yet the monitoring of model performance—determining when a model must be adjusted or discarded—has not reached a similar level of sophistication.

There are two common practices for monitoring model performance:

- Assess all time series based on a common criterion, such as "MAPE > 50%." In this approach, every time a forecast error exceeds the criterion, the time series is flagged for review.
- Replace models for all time series at fixed intervals (e.g., weekly, monthly, quarterly, biannually, or annually).

Note that both methodologies use a "one size fits all" approach, where *all time series* are treated in the same manner. This is problematic and inefficient. Regarding the first approach, a single forecast may exceed the error criterion just by chance, with nothing fundamentally wrong with the model. Such a model would not need to be reviewed. Or for a stable, easy-to-forecast time series, the error may always fall below the criterion, yet still be much higher than should be achievable with a more appropriate model. In this case, such a model should be replaced even though it may never be flagged for review.

Regarding the second approach, a suitably performing model does not need to be replaced on a routine basis. Doing so is just a waste of time and computational resources. Also, frequent model replacement is neither practical nor advisable as

it can result in a lack of forecast-model stability and cause forecasts to fluctuate, in some cases dramatically.

Once forecast models are selected and forecasts are generated, the resultant models need to be monitored for quality and continued suitability, as time passes and new data points are added to the historical data. This ability to automatically evaluate each individual time series would represent a paradigm shift from the "one size fits all" approach that currently exists in many forecasting tools. The approach described below demonstrates that it is possible to automate the monitoring of forecast-model fitness using control charts.

## RESIDUAL ANALYSIS METHODOLOGY

For each individual time series, the new approach begins with an examination of model *residuals*—the difference between the forecast value and the actual value. The focus on residuals makes it different from the discussion of process behavior charts in forecasting (Joseph and Finney, 2013), which is based on time series of *product sales*. For a more general discussion on statistical process control and control charts, refer to Wheeler and Chambers (1992) and Montgomery (2009).

When the forecast model is well specified, the residuals will be centered on a mean of zero, appear random with no specific patterns evident, and thus represent a process that is *in control*. Since the method uses only model residuals, it is model agnostic. The models used could be traditional time-series based, machine-learning based, ensemble models, or hybrid models.

The method analyzes residuals using Shewhart Control Charts for individual measurements, since the data is time-series based and there is one observation for each time period. The control chart, as shown in **Figure 1**, is constructed around the mean, $\overline{X}$ ,of the residuals, with UCL/LCL representing the Upper/Lower Control Limits. Zone C, in green, represents the 1-sigma range above and below the mean. Zone B, in yellow, represents

## Key Points

- While automatic forecasting algorithms are adept at simplifying the model-selection process and generating forecasts without manual input, the monitoring of model performance—determining when a model must be adjusted or discarded—has not reached a similar level of sophistication.
- Once forecast models are selected and forecasts are generated, the resultant models need to be monitored for quality and continued suitability as time passes and new data points are added to the historical data. Automating this process would be a valuable advance, and one even more critical when the number of time series is very large.
- An automated monitoring method based on the application of control charts to forecast model residuals is demonstrated. The method flags residuals for classification into severity zones. User rules then determine the action to be taken: Low Severity—No model changes required; Medium Severity—Adjust/refit existing model parameters; High Severity—Discard current model and develop new model.
- The process is illustrated in detail using time series of product sales at two distribution centers.

**Figure 1. Control Chart Zones**

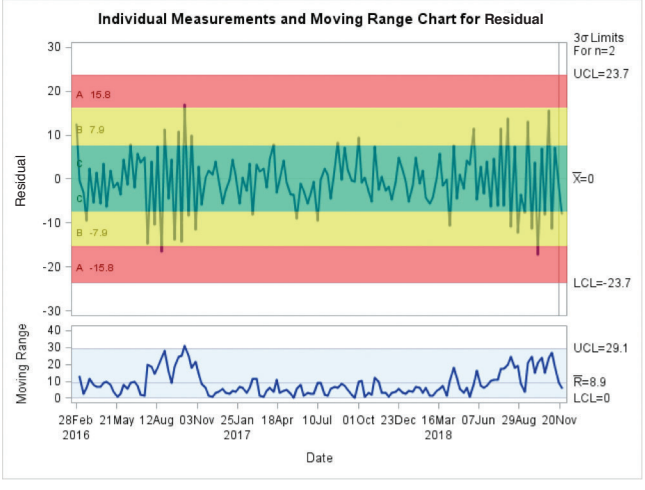


**Table 1. Western Electric Rules or Tests (Source: SAS®/QC Documentation)**

| Rule/Test | Pattern Description |
|---|---|
| 1 | One point beyond Zone A (outside the control limits) |
| 2 | Nine points in a row in Zone C or beyond on one side of the central line (The number of points can be specified as 7, 8, 9, 11, 14, or 20) |
| 3 | Six points in a row steadily increasing or steadily decreasing (The number of points can be specified as 6, 7, or 8) |
| 4 | Fourteen points in a row alternating up and down |
| 5 | Two out of three points in a row in wZone A or beyond |
| 6 | Four out of five points in a row in Zone B or beyond |
| 7 | Fifteen points in a row in Zone C on either or both sides of the central line |
| 8 | Eight points in a row on either or both sides of the central line with no points in Zone C |

**Table 2. Shewhart Residual Analysis Custom Control Chart Rules/Tests**

| Rule/Test | Pattern Description |
|---|---|
| Discard/Redevelop | 2 out of 2 points beyond 3-sigma limits |
| Discard/Redevelop | 7 out of 7 points between 1-sigma and 3-sigma limits |
| Adjust/Refit | 3 out of 3 points beyond 2-sigma limit |
| Adjust/Refit | 5 points in a row increasing |
| Adjust/Refit | 5 points in a row decreasing |

**Figure 2. Historical Data for SKU Code=603-560613 at Distribution Center DC1**

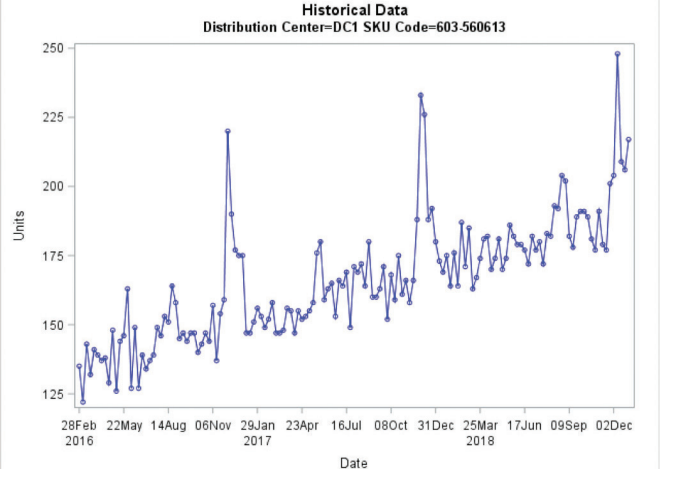
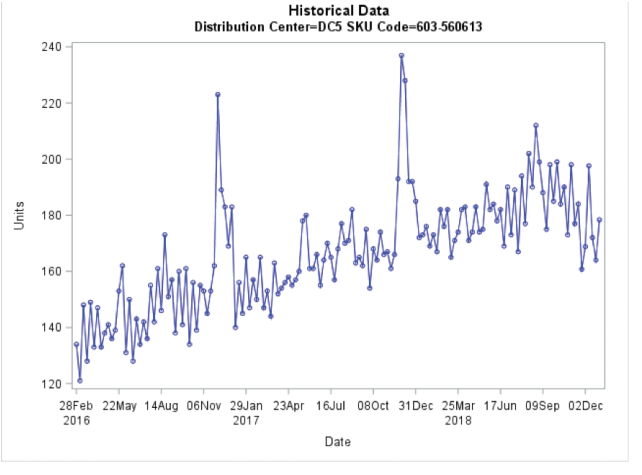



**Figure 3. Historical Data for SKU Code=603-560613 at Distribution Center DC5**




the 1-sigma to 2-sigma range above and below the mean. Zone A, in red, represents the 2-sigma to 3-sigma range above and below the mean.

Once the control limits and zones have been specified, exceptions are identified in the residual patterns using a combination of standard Western Electric rules and additional rules that are customizable by the user to best suit their business needs. As such, these rules can be made to be very sensitive to changes in residual patterns or structured to take a more conservative approach. The rules below try to strike a balance and are provided as guidelines.

The standard Western Electric Rules or Tests are shown in **Table 1**.

The custom rules/tests that we have employed in this example are shown in **Table 2**.

These custom rules/tests provide an example of how a user might set the trigger conditions for when to discard and redevelop a model, when to adjust and refit a model, or otherwise when to just leave the current model alone. They represent a judgment call on how sensitive or conservative the user wishes the method to be.

We use the term *model vintage* to represent the date of the last model update or adjustment. You can think of it as the date of the most recent actual value included in the latest model update.

Once the exceptions are identified for dates greater than or equal to the model vintage date, residual process anomalies are classified by applying additional logic/business rules to the exceptions flagged from the Shewhart Control Charts. The anomalies are then sorted by level of severity and appropriate action is taken:

- Low Severity—No model changes required
- Medium Severity—Adjust/refit existing model parameters
- High Severity—Discard current model and develop new model

## ILLUSTRATIVE EXAMPLES

For illustrative purposes, we'll execute the method for the two time series shown for an item sold from Distribution Centers DC1 in **Figure 2** and DC5 in **Figure 3**. The data are weekly, spanning February 28, 2016 through December 30, 2018.

The patterns for these time series look somewhat similar except near the end of the series. The difference in their behaviors becomes apparent, however, when analyzing the residuals over time.

Shown in **Table 3** are the initial models created via automated model selection, using actuals posted through November 18, 2018 (which becomes the model vintage date).

Using a forecast horizon of six periods, the plot of the initial model for DC1 is shown in **Figure 4** with a fit MAPE = 2.63%. The model for DC1 is a traditional exponential-smoothing model with linear trend and additive seasonality. For DC5, a seasonal ARIMA model was chosen. (For *Foresight* tutorials on exponential smoothing and ARIMA models, see Stellwagen, 2012 and 2013.)

Using a forecast horizon of six periods, the plot of the initial model for DC5 is shown in **Figure 5** with a fit MAPE = 4.46%.

As noted, our monitoring methodology begins with the calculation of the residuals from a forecasting model. Control limits are then determined based on the residuals. Residuals are, in effect, the "fit error" of the model over history.

Once the control chart has been created, we begin evaluating each new observation starting on the model vintage date. How the current model has fit in the past is no longer relevant since it isn't actionable: the model developed on November 18, 2018 can't be used to assess what happened in history—even though there may be several historical points outside the control limits or patterns detected. We react only to new observations and associated patterns that are actionable.

**Table 3. Initial Models by Location**

| Location | Model Type | Model Specification |
|---|---|---|
| DC1 | Winters Method (Additive) | α=0.0327137184, β=0.0265873762, and γ=0.001 |
| DC5 | ARIMA | P = ((1)(52)) D = (1) Q = (1,2,3) |

**Figure 4. Winters Model Fit and Forecast SKU Code=603-560613 at Distribution Center DC1**

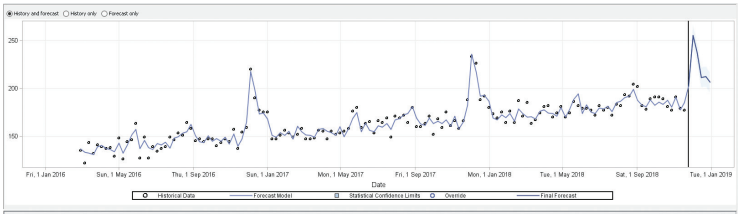


**Figure 5. ARIMA Model Fit and Forecast SKU Code=603-560613 at Distribution Center=DC5**

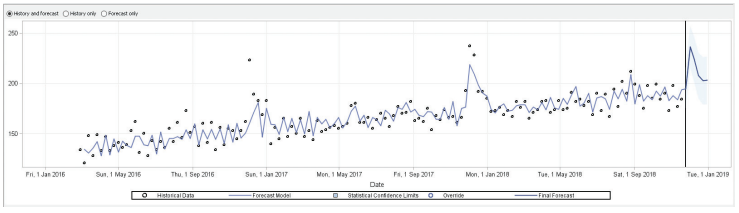
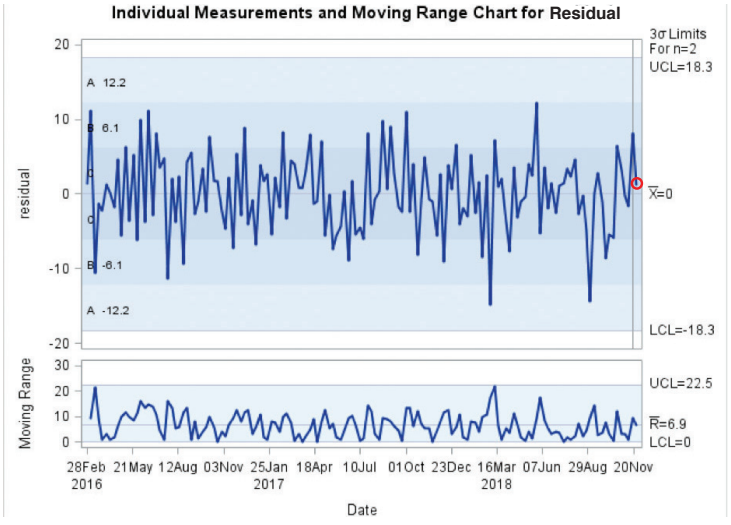


**Figure 6. IR Chart Distribution Center=DC1 SKU Code=603-560613 Posted History through: 25NOV18 Model Vintage: 18NOV18**




### *Example for DC1*

**Figure 6** illustrates the Shewhart Control Plot when the next week of data (November 25) becomes available.

The residual of 1.34 for November 25, highlighted above, is within the control limits with no patterns detected so no additional action is taken. This residual

**Figure 7. IR Chart Distribution Center=DC1 SKU Code=603-560613 Posted History through: 23DEC18 Model Vintage: 18NOV18**

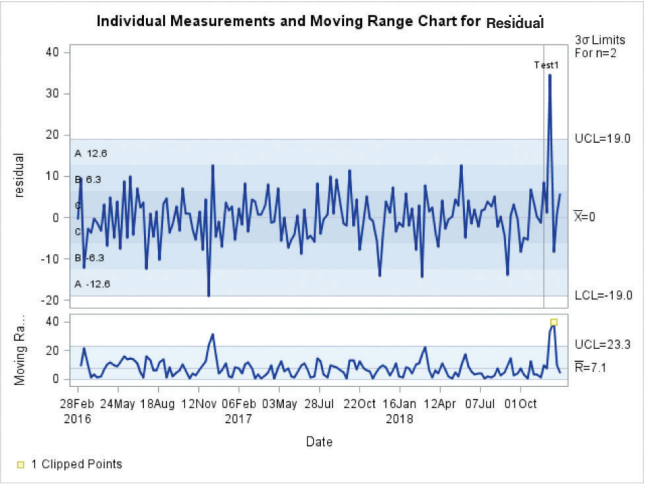



**Figure 8. Winters Model Fit and Out-of-Sample Forecast SKU Code=603-560613 at Distribution Center=DC1**

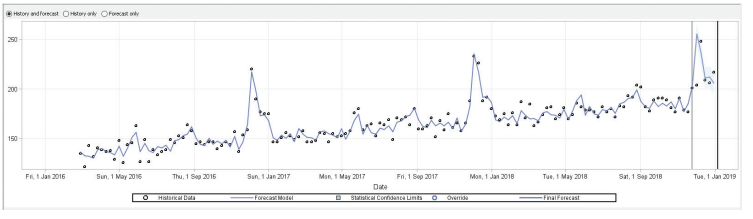
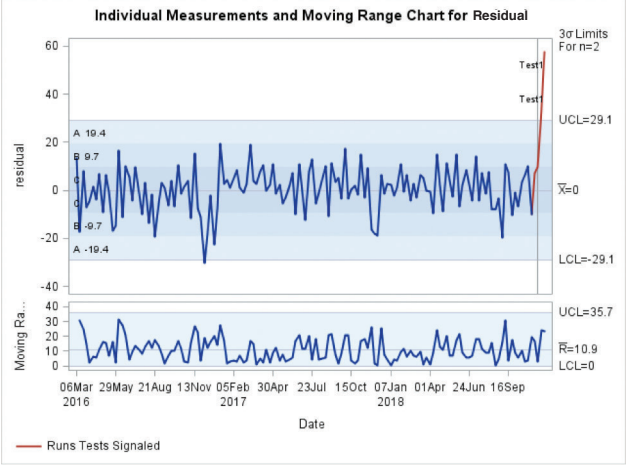


**Figure 9. IR Chart Distribution Center=DC5 SKU Code=603-560613 Posted History through: 02DEC18 Model Vintage: 18NOV18**




evaluation process is conducted each week as the new actual is observed and posted. Fast-forwarding four weeks, with actuals posted through December 23, 2018, the updated control chart is shown in **Figure 7**.

Except for December 2, where the residual is outside the 3-sigma limit, we see that the residuals posted each week from November 25 to December 23 all fell within the control limits, so no action was required. We do not want to overreact to a single point being outside the 3-sigma limits. The original model developed on November 18 has remained in place without the need for any updates. (Note: A "Clipped Point" indicates an extreme value beyond the scale of the axis to make the chart more readable.)

Using an out-of-sample range of six periods, the plots of the Winters model with a November 18 vintage date and data through December 30 is shown in **Figure 8** with a fit MAPE = 2.63% and an Out-of-Sample MAPE = 6.57%.

### *Example for DC5*

**Figure 9** shows the Shewhart control chart plots with actuals posted through December 2, two weeks following the vintage.

We now see five consecutive points increasing in value, which indicates (per Table 2) that the model needs an adjustment/refit. In this case, the pattern started prior to the model vintage date but continued after it. Since the test is not signaled for these five points until after the model vintage date, it is a legitimate pattern that has been flagged. But we also observe that a Test 1 exception is thrown for both November 25 and December 2—two consecutive points beyond the 3-sigma limits indicates (per Table 2) that the model should be discarded.

Since the discard flag is of higher priority than the refit flag, the model will be discarded and redeveloped with a new model vintage date of December 2, 2018. The new model is as follows:

ARIMA: P = 1 D = (1) Q = ((1,2,3)(52)) + AO02DEC2018D + AO25NOV2018D

The plot of the new model is shown in **Figure 10** with a fit MAPE = 4.38%.

This residual evaluation process is conducted each week when the new actual is observed and posted. Fast-forwarding four weeks, with actuals posted through December 23, 2018, the updated control chart is shown in **Figure 11**.

The residuals posted each week from December 9 to December 23 all fell within the control limits with no patterns detected, so no action was required.

The plot of the new ARIMA model with a December 2 vintage date and data through December 30 is shown in **Figure 12** with a fit MAPE = 4.38% and an Out-of-Sample MAPE = 13.51%.

For DC5, the original model developed on November 18, 2018 was discarded and redeveloped on December 2.

## SUMMARY

The methodology demonstrates an innovative use of control charts for monitoring forecasts and determining forecast-model suitability. Each time series is managed and modeled (i.e., discard/redevelop, adjust/refit, or reforecast) based on its individual residual pattern and characteristics.

The methodology can be extended for use with each level in hierarchical forecasting and other modeling techniques. The methodology is efficient, robust, customizable/extendable, and scalable. It also avoids the "one size fits all" limitation that exists in many forecasting tools today.

**Figure 10. Updated ARIMA Model Fit and Forecast SKU Code=603-560613 at Distribution Center=DC5**

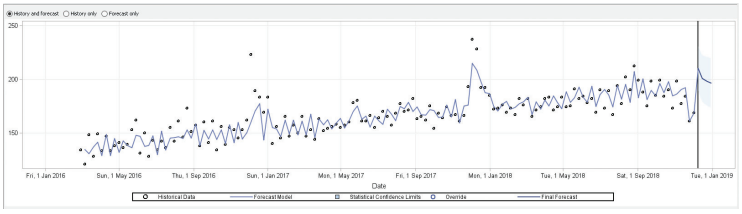


**Figure 11. IR Chart Distribution Center=DC5 SKU Code=603-560613 Posted History through: 23DEC18 Model Vintage: 02DEC18**

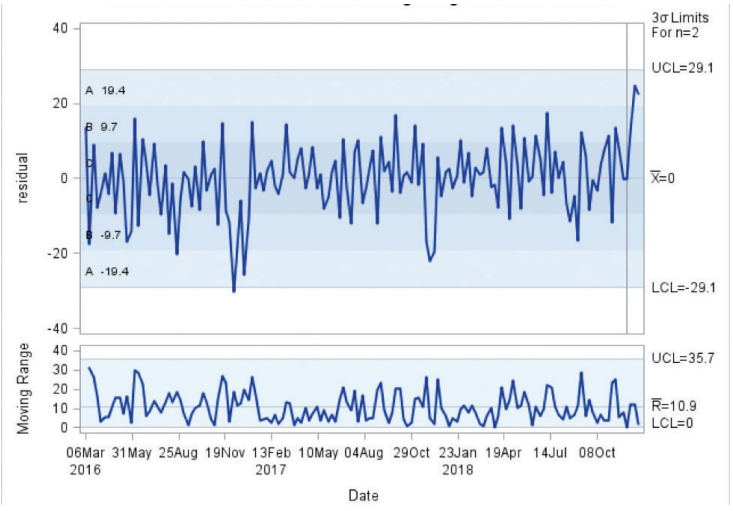
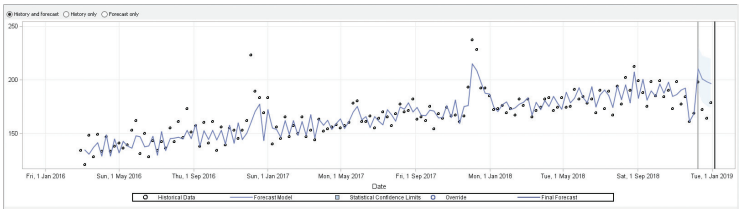



**Figure 12. Updated ARIMA Model Fit and Out-of-Sample Forecast SKU Code=603-560613 at Distribution Center=DC5**



### REFERENCES


Joseph, M. & Finney, A. (2013). Using Process Behavior Charts to Improve Forecasting and Decision Making, *Foresight*, Issue 31 (Fall 2013), 41-48.

Katz, J. H. (2015). *United States of America Patent No. 9,208,209*.

Montgomery, D. C. (2009). *Introduction to Statistical Quality Control*, 6th ed. Hoboken: Wiley.

*SAS®/QC 15.1 User's Guide*. (2019). Retrieved from ***https://go.documentation.sas.com/?cdcId=pgmsascdc&cdcVersion=9.4_3.4&docsetId=qcug&docsetTarget=titlepage.htm&locale=en***

Stellwagen, E. (2012). Exponential Smoothing: The Workhorse of Business Forecasting, *Foresight*, Issue 27 (Fall), 23-28.

Stellwagen, E. & Tashman, L. (2013). ARIMA: The Models of Box and Jenkins, *Foresight*, Issue 29 (Spring), 28-33.

Wheeler, D. J. & Chambers, D. S. (1992). *Understanding Statistical Process Control*, 2nd ed. Knoxville: SPC Press.



**Joseph H. Katz** is the Forecasting Product Manager at SAS. Prior to moving into the product management role, Joe had over 25 years of experience as an advanced analytics and process improvement professional in both consulting and operations roles. Previously, he served as forecasting manager in the telecommunications and airline industries as well as a process improvement consultant with Bell Labs. Joe has a PhD in mathematics (operations research) from the Colorado School of Mines.

**joe.katz@sas.com**

WHITE PAPER

# Assisted Demand Planning Using Machine Learning for CPG and Retail

sas
THE POWER TO KNOW.

# Contents

We are entering the second Machine Age. This first began with the Industrial Revolution, which was driven primarily by technology innovation – the ability to generate mechanical power to make humans more productive. Rather than the steam engine that started the Industrial Revolution, the second Machine Age uses computers and other digital advancements to help our brains understand and shape our environments.

Impressive advances in artificial intelligence (AI) and machine learning (ML) during the past decade are supported by supervised deep learning to train ML algorithms to perform narrow, single-domain tasks. The learning is supervised because you're telling the algorithm the correct answer as it is exposed to many examples using big data sets. But we're now seeing unsupervised learning systems that learn faster, require less data and achieve impressive performance. These supervised and unsupervised intelligent automation techniques can help humans automate tasks. That doesn't eliminate the need for humans, it just allows them to do their work more effectively.

Intelligent automation techniques can be applied to all kinds of activities across your organization to reduce the everyday repetitive work while uncovering key insights to improve the effectiveness of your processes, as well as your workforce.

## How intelligent automation enhances existing processes

Intelligent automation driven by AI and ML are disrupting the way companies do business. The rapid deployment of automation is helping us set new standards of efficiency, speed and functionality.

Instead of being replaced, humans will see unprecedented job creation and new opportunities to add more value. Intelligent automation techniques can be applied to all kinds of activities across your organization to reduce the daily repetitive work while uncovering key insights to improve the effectiveness of your processes and your workforce.

Applications can range from routine to groundbreaking, such as collecting, analyzing and making decisions about textual information to guiding demand planners to anticipating consumer purchasing behavior. It is already helping companies surmount conventional performance tradeoffs to achieve unprecedented levels of efficiency that reduce costs while increasing profitability.

The variety of business challenges to which intelligent automation can be applied is expanding as technologies for voice recognition, natural language processing and ML improve. These technologies are becoming increasingly available as IoT devices capture streaming information and open-source cloud-based services become more widespread.

> "The challenge is that people have not developed the level of trust in artificial intelligence and machine learning that they have in other technologies that automate tasks. People sometimes confuse automation with autonomy."
>
> Oliver Schabenberger, COO and CTO, SAS

Leading organizations are driving more of their processes into smarter machines. They are rethinking what they do across every area of the enterprise – from their business processes to the customer experience. Some activities where intelligent automation is helping companies are:

- Data collection.
- Security and systems monitoring.
- Transaction management with ERP systems.
- Scheduling and staffing.
- Accounting and finance.
- Business planning.
- Customer experience.
- Marketing and communications.

The levels of automation include:

- Basic automation of frequent repetitive simple tasks.
- Advance automation that orchestrates workflows across departments and applications.
- Intelligent automation that mimics complex research and expert decision making.

The foundational elements for intelligent automation are:

- Institutionalized business process.
- Reducing effort and increasing accuracy.
- Centralized application with auditing and instrumentation.
- Quantifiable metrics to measure and inform model improvements.
- Analytics infrastructure to support machine learning.
- APIs to allow software agents to mimic and drive tasks.
- Continuous monitoring and automation governance.

Intelligent automation is empowering humans with advanced smart technologies and agile processes for faster, more intelligent decisions. The key benefits of intelligent automation in business include:

- Improved productivity.
- Increased process efficiency.
- Improved customer experience.
- Unprecedented value (ROI).

We hope you have enjoyed this excerpt of “Assisted Demand Planning Using Machine Learning for CPG and Retail.”

To access the full whitepaper, please visit https://www.sas.com/en/whitepapers/assisted-demand-planning-109971.html.

WHITE PAPER

# What Management Must Know About Forecasting

sas

# Contents

# The Forecasting Dilemma

Forecasts never seem to be as accurate as we would like, or need, them to be. As a result, there's a temptation to throw money at the problem in hopes of making it go away. There are plenty of consultants and software vendors willing to take that money in exchange for lots of promises, but are these promises ever fulfilled? How many organizations are you aware of – perhaps even your own – that have thrown thousands or even millions of dollars at the forecasting problem, only to end up with the same lousy forecasts?

The questions boil down to:

- Why do forecasts always seem to be wrong and sometimes terribly wrong?
- Is there anything you can do about it?

This white paper explores why forecasting is often poorly done and offers suggestions for improving it. Because forecasting is a core capability of AI, it also examines the emergence of AI and machine learning to enhance traditional time-series forecasting methods.

# Why Are Forecasts Wrong?

There are at least four reasons why your forecasts are not as accurate as you would like them to be.

The first reason is unsuitable software – software that doesn't have the necessary capabilities, has mathematical errors or uses inappropriate methods. It is also possible that the software is perfectly sound but due to untrained or inexperienced forecasters, it is misused.

The second reason is untrained, unskilled or inexperienced forecasters who exhibit behaviors that affect forecast accuracy. One example is overadjustment, or as W. Edwards Deming put it, "fiddling" with the process. This happens when a forecaster constantly adjusts the forecast based on new information. Research has shown that much of this fiddling makes no improvement in forecast accuracy and is simply wasted effort.[1] Forecasting should be objective and scientific.

The third reason for forecasting inaccuracy is contamination by the biases and personal agendas of the process participants. Instead of presenting an unbiased best guess at what is going to happen, the forecast comes to represent what management wants to have happen – no matter what the marketplace is saying. Forecast value added (FVA) analysis, described later, can identify these sorts of biases and help streamline the forecasting process.

Finally, bad forecasting can occur because the desired level of accuracy is unachievable for the behavior you are trying to forecast. Consider the example of calling heads or tails in the tossing of a fair coin. It doesn't matter that you may want to achieve 60, 70 or 90 percent accuracy. The reality is that over a large number of tosses, you will only be right half of the time and nothing can change that. The nature of the behavior determines how well you can forecast it – and this applies to demand for products and services just as it does to tossing coins.

## Inadequate, Unsound or Misused Software

A common mistake in bad or misused software is choosing a forecasting model based solely on the model's "fit to history" (often referred to as "best fit" or "pick best" functionality). The software provides (or the forecaster builds) several models so you can evaluate them against recent history. The model that has the best fit to history is selected to create forecasts of the future.

In Figure 1, the history consists of four weeks of actual sales: 5, 6, 4 and 7 units. You can see these as the four dots in each graph. Let's consider four models for forecasting future sales.

Model 1 is simply the average of the four points of history, and it forecasts 5.5 units for week 7. Model fit over the four points of history has a mean absolute percent error (or MAPE) of 18 percent.

Model 2 is a least squares regression line that shows an upward trend, and it forecasts 7.2 units for week 7. It has a fit error of 15 percent over the four points of history.

Model 3 is a quadratic equation with a fit error of only 8 percent, and it forecasts 16.5 units in week 7.

Finally, Model 4 is a cubic equation that fits the history perfectly with a fit error of 0 percent. It forecasts about 125 units in week 7.

Remember, the objective is not to fit a model to history – it is to find an appropriate model for forecasting future weekly sales. Fitting a model to history is easy. Anyone can do it, and it is always possible to find a model that has a perfect fit. But having perfect fit to history is no guarantee that the model will generate accurate forecasts. In this example, bad software (or misguided forecasters) using fit to history as the sole criterion for selecting the forecasting model would have chosen Model 4. Unless you have good reason to believe that sales of this product are about to explode, Model 4 actually appears to be the worst choice of the forecasting models. Models 1 or 2 (the ones with the worst fit) are probably the most appropriate models for forecasting the future, given the limited historical information.

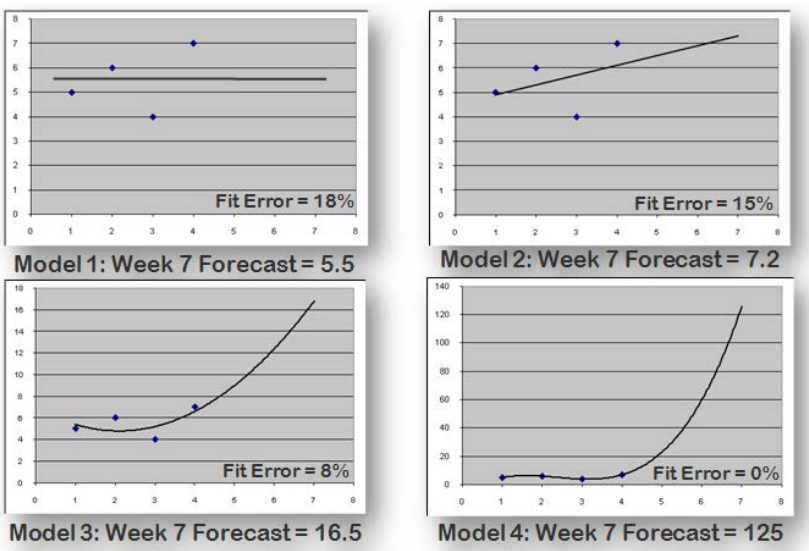



Figure 1: Confusing fit to history with appropriateness for forecasting.

We hope you have enjoyed this excerpt of “What Management Must Know About Forecasting.”

To access the full whitepaper, please visit https://www.sas.com/en_us/whitepapers/management-forecasting-104529.html.

# APPENDIX

For further reading, we suggest the following resources.

## Books and Journals


Brocklebank, J., Dickey, D., and Choi, B. (2018). *SAS for Forecasting Time Series* (3rd edition). SAS Institute.

Chase, C. (2016). *Next Generation Demand Management: People, Process, Analytics and Technology*. Wiley.

Gilliland, M. (2010). *The Business Forecasting Deal*. Wiley.

Gilliland, M., Tashman, L., and Sglavo, U. (eds.) (2015). *Business Forecasting: Practical Problems and Solutions*. Wiley.

Kolassa, S. and Siemsen, E. (2016). *Demand Forecasting for Managers*. Business Expert Press.

Makridakis, S., Wheelwright, S., and Hyndman, R. (1998). *Forecasting: Methods and Applications* (3rd edition). Wiley.

Makridakis, S. and Petropoulos, F. (eds.) (2020). *International Journal of Forecasting*, Vol. 36, Issue 1. Special issue containing 35 articles with results, analysis, and commentary on the M4 Forecasting Competition.

Morlidge, S. (2018). *The Little Illustrated Book of Operational Forecasting*. Troubador.


## Whitepapers and SGF Papers (downloadable)


Chase, C. (2014) Using Multitiered Causal Analysis to Improve Demand Forecasts and Optimize Marketing Strategy.

Gilliland, M. (2015). Forecast Value Added Analysis: Step by Step.

Joshi, M. (2018). Economic Capital Modeling with SAS® Econometrics.

Leonard, M. and Elsheimer, B. (2015). Count Series Forecasting.

Leonard, M. and Elsheimer, B. (2017). Automatic Singular Spectrum Analysis and Forecasting.

Quirino, T., Leonard, M., and Blair E. (2018). Scalable Cloud-Based Time Series Modeling and Forecasting.

Selukar, R. (2017). Detecting and Adjusting Structural Breaks in Time Series and Panel Data Using the SSM Procedure.