ASK THE EXPERT

An A to Z Overview of Forecasting in SAS®

Gerhard Svolba
Advisory Presales Solutions Architect, SAS

Spiros Potamitis
Senior Product Marketing Manager, SAS

Gerhard Svolba

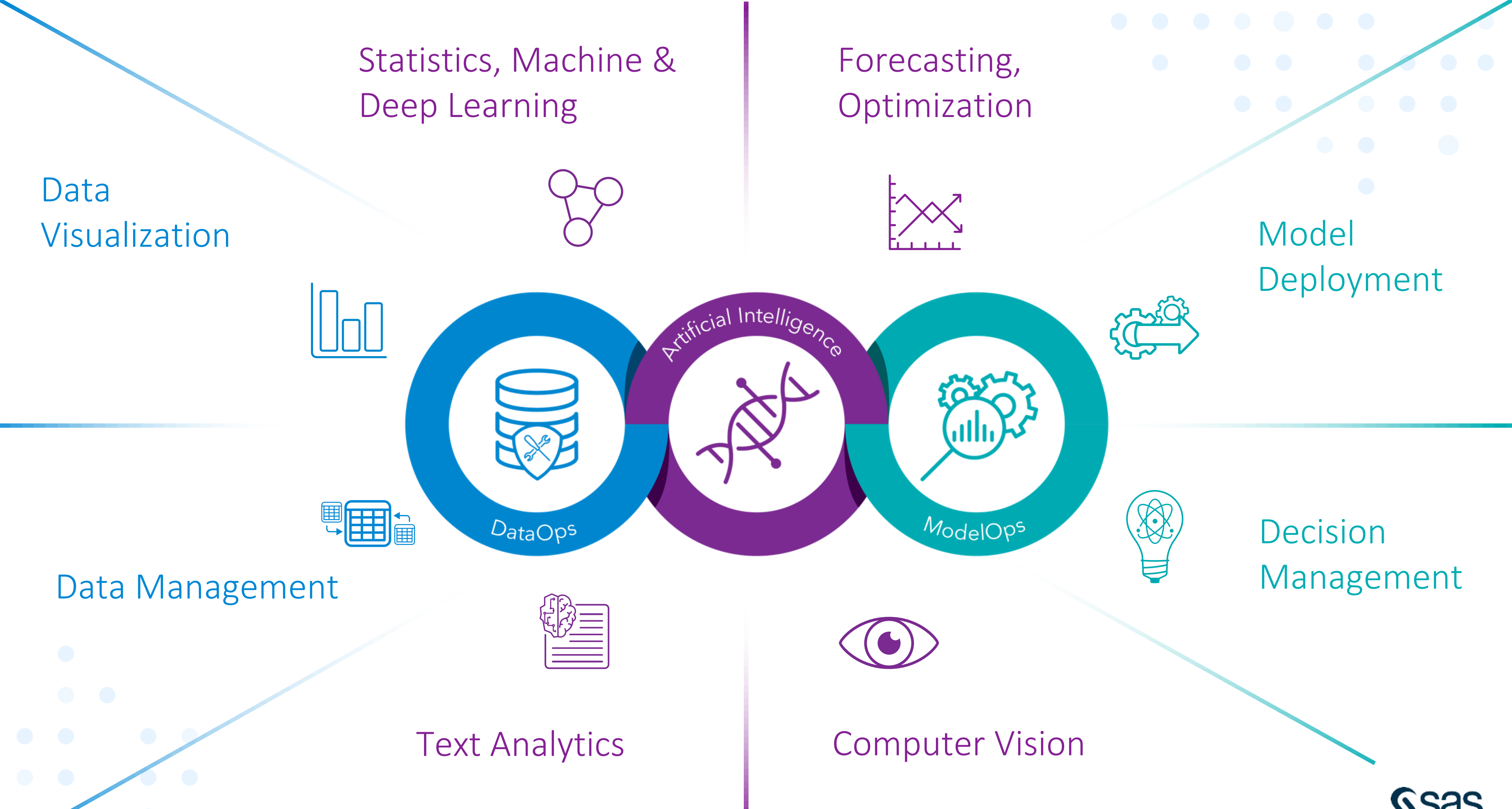
Advisory Presales Solutions Architect, SAS

Gerhard is involved in numerous analytic and data science projects in different business and research domains, including demand forecasting, analytical CRM, risk modeling, fraud prediction and production quality. His project experience ranges from business and technical conceptual considerations to data preparation and analytic modeling across industries. He is the author of the SAS Press books Data Preparation for Analytics Using SAS, Data Quality for Analytics Using SAS and Applying Data Science: Business Case Studies Using SAS. As a part-time lecturer, Svolba teaches data science methods at the University of Vienna, the Medical University of Vienna and for business schools.

Spiros Potamitis

Senior Product Marketing Manager, SAS

Spiros is a data scientist and a Global Product Marketing Manager of forecasting and optimization at SAS. He has extensive experience in the development and implementation of advanced analytics solutions across different industries and provides subject matter expertise in the areas of forecasting, machine learning and AI. Prior to joining SAS, Potamitis has worked on and led advanced analytics teams in various sectors such as credit risk, customer insights and CRM.


Statistics, Machine &
Deep Learning
Forecasting,
Optimization
Data
Visualization
Model
Deployment
Artificial Intelligence
DataOps
ModelOps
Decision
Management
Data Management
Text Analytics
Computer Vision

WHY Forecasting?

DER Touristik

SAS helps DER Touristik better plan for the future with fast and reliable forecasting.

LEVI STRAUSS & CO.

SAS helps Levi Strauss & Co develop smarter demand plans so consumers around the world can find the merchandise they want.

"We are incredibly grateful to have a flexible, experienced partner like SAS," says **Barilla**. "They've helped us start our digital transformation and face daunting challenges with confidence."

SAS helped Barilla meet extraordinary demand expectations during times when supply chain disruptions seem to be the norm. Read more on The New York Times.

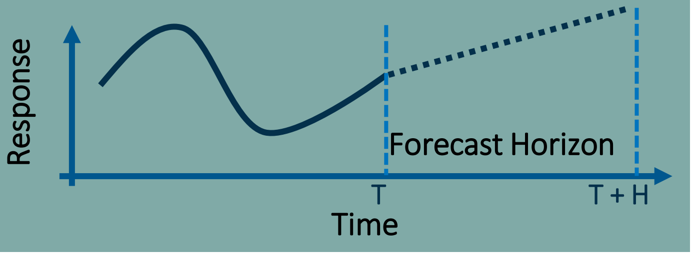
sas

Forecasting compared to Predictive Modeling

A critical distinction between forecasting and other predictive modeling is the **time** component. Forecasts at a future time (horizon) depend on responses in the historical periods.

Forecasting

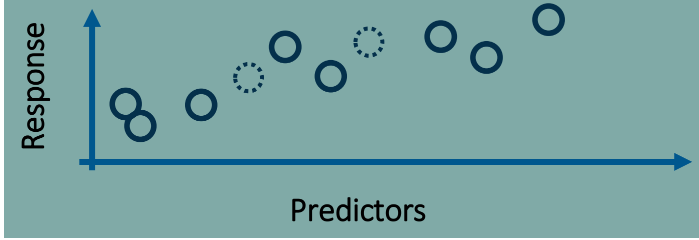
$$\hat{y}_{T+h} = f_h\,(y_T\,,\, y_{T-1}\,,\, \ldots,\, x_1,\, x_2,\, \ldots)$$



- Tourist visits next summer
- Product demand for the next 6 months
- Website visits in next weeks

Predictive Modeling

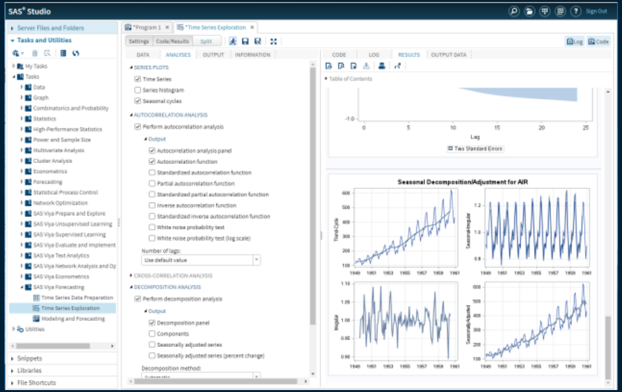
$$\hat{y} = f\,(x_1\,,\, x_2\,,\, \ldots)$$



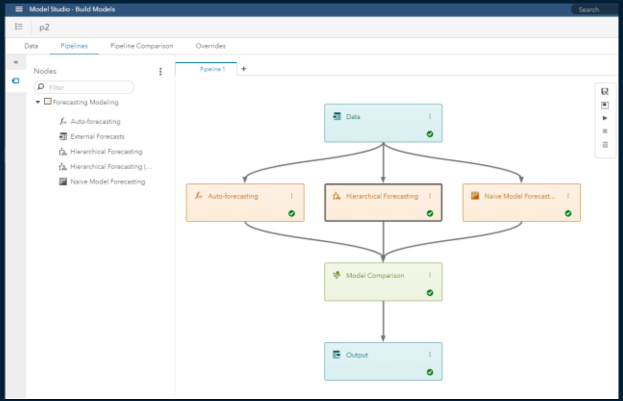
- MPG given car’s characteristics
- Wine quality score given lab test results
- Customers’ ratings given product information

Methods to Perform Forecasting in SAS

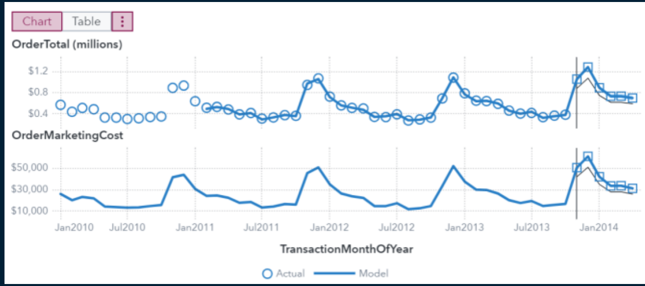
SAS Programming and SAS Studio Tasks



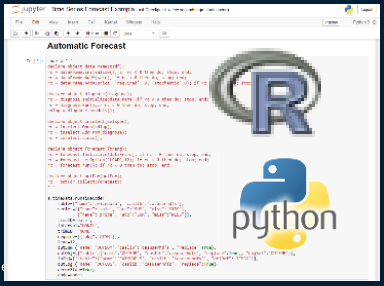
Visual Forecasting Pipelines/Automated Forecasting



Forecasting Object in SAS Visual Analytics



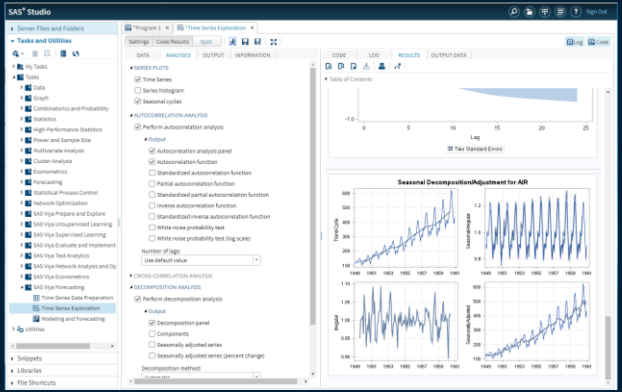
Open-Source Integration



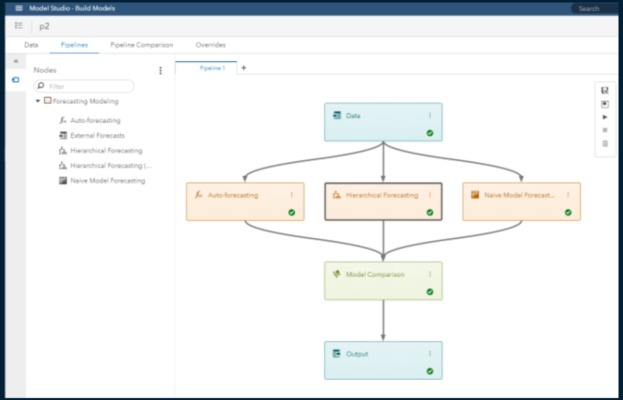
- Distribute open-source algorithms in Viya nodes
- Use APIs (like Python & R) to call SAS Viya algorithms

Methods to Perform Forecasting in SAS

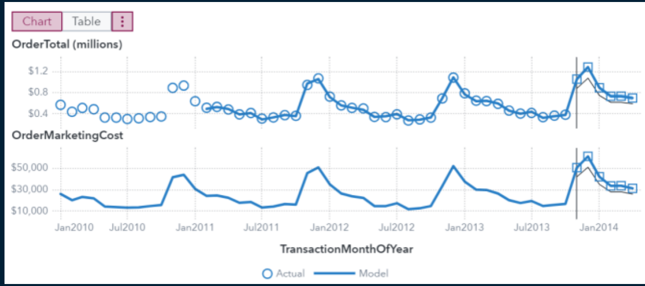
SAS Programming and SAS Studio Tasks



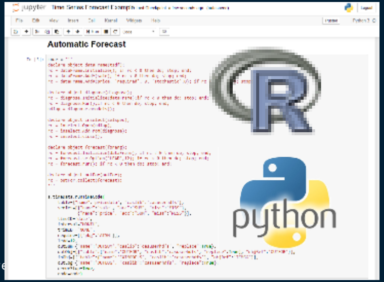
Visual Forecasting Pipelines/Automated Forecasting



Forecasting Object in SAS Visual Analytics



Open-Source Integration



- Distribute open-source algorithms in Viya nodes
- Use APIs (like Python & R) to call SAS Viya algorithms

Popular Forecasting Procedures in SAS/ETS and SAS Econometrics (selection)

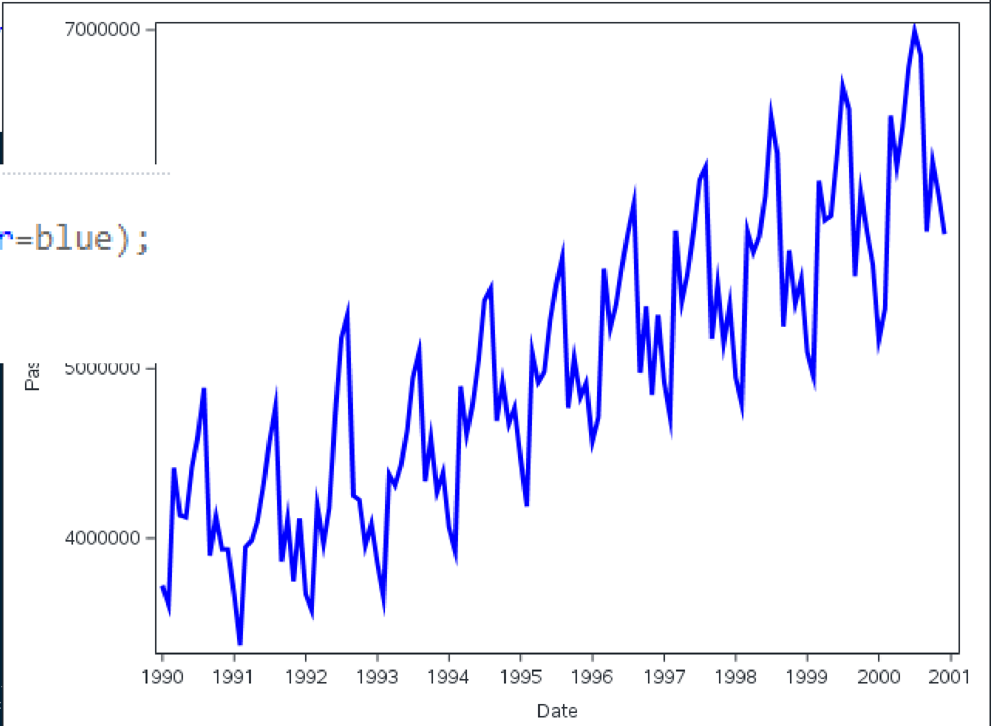
- PROC ESM: exponential smoothing models with optimized smoothing weights for many time series or transactional data
- PROC ARIMA: equally spaced univariate time series data, transfer function data, and intervention data by using the autoregressive integrated moving-average (ARIMA) or autoregressive moving-average (ARMA) model
- PROC AUTOREG: linear regression models for time series data when the errors are autocorrelated or heteroscedastic
- PROC UCM: equally spaced univariate time series data by using an unobserved components model (UCM)

- **Timeseries Data Management** with: PROC TIMESERIES, PROC EXPAND, PROC TIMEDATA

(Always) plot and review your Data

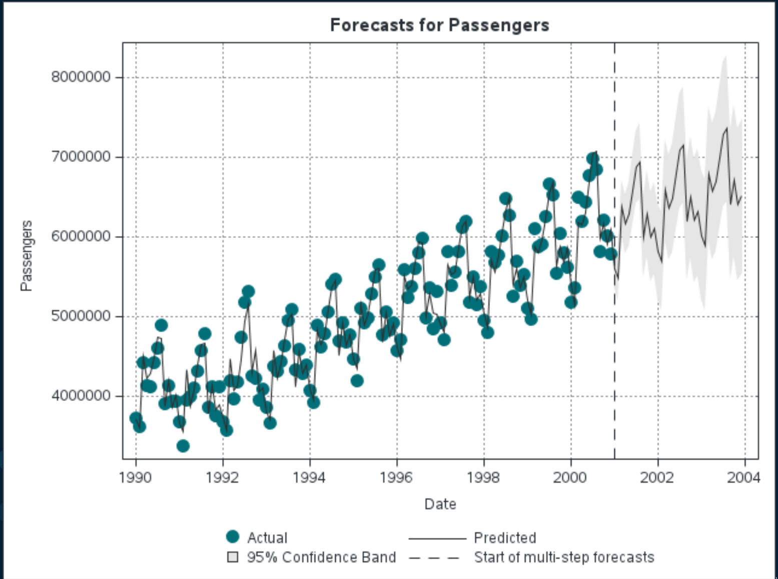
```
proc sgplot data=work.flights_911;
 series x=date y=passengers / lineattrs=(color
run;
```

```
proc sgplot data=work.flights_911;
 series x=date y=passengers / lineattrs=(color=blue);
 where 1990 <= year(date) <= 2000;
run;
```



Use the ESM procedures for exponential smoothing models

```
proc esm data=work.flights_911 plot=(Forecasts) lead=36 outfor=work.ESM_FC1 print=all ;
 where 1990 <= year(date) <= 2000;
 forecast passengers / model=addwinters ;
 id date  interval=month;
run;
```

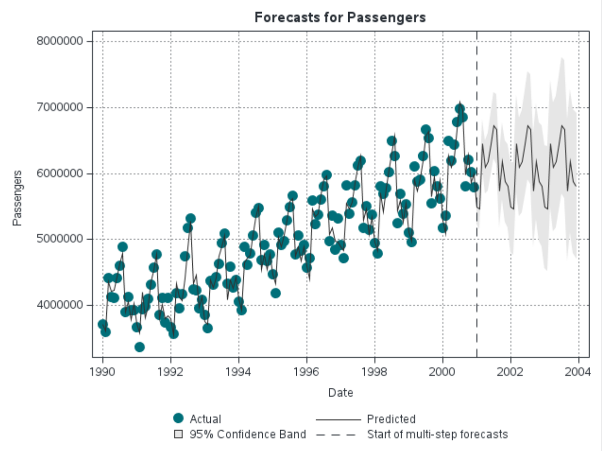


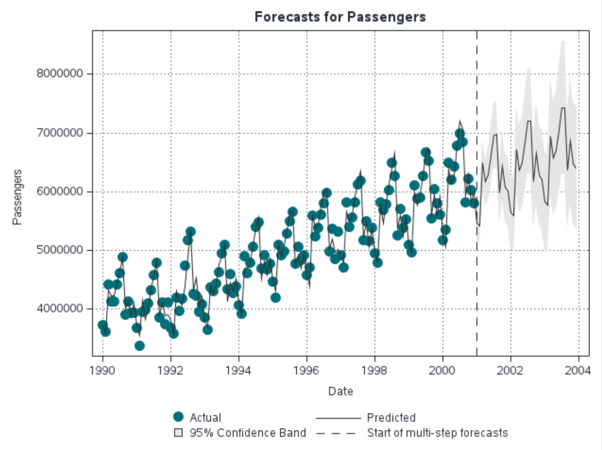
Total rows: 168 | Columns: 8 of 8 | Rows 1 to 168

Date	ACTUAL	PREDICT	LOWER	UPPER
SEP2000	5810787.8	5983266.9...	5731420....	6235113....
OCT2000	6215357.7	6174765.0...	5922918....	6426611....
NOV2000	6021347.9	5905018.3...	5653171....	6156865....
DEC2000	5794379.8	6086641.4...	5834794....	6338488....
JAN2001	.	5607069.3...	5355222....	5858916....
FEB2001	.	5487061.6...	5190572....	5783551....
MAR2001	.	6384545.9...	6049234....	6719857....
APR2001	.	6163654.084	5793503....	6533804....
MAY2001	.	6283725.3...	5881684....	6685766....

Evaluate model fit, select and compare different model types

Statistics of Fit for Variable Passengers	
Statistic	**Value**
Degrees of Freedom Error	130
Number of Observations	132
Number of Observations Used	132
Number of Missing Actuals	0
Number of Missing Predicted Values	0
Number of Model Parameters	2
Total Sum of Squares	3.40015E15
Corrected Total Sum of Squares	9.07022E13
Sum of Square Error	2.32503E12
Mean Square Error	1.76139E10
Root Mean Square Error	132717.317
Unbiased Mean Square Error	1.78849E10
Unbiased Root Mean Square Error	133734.323
Mean Absolute Percent Error	2.06163916
Mean Absolute Error	102384.498
R-Square	0.97436629
Adjusted R-Square	0.97416911
Amemiya's Adjusted R-Square	0.97357757
Random Walk R-Square	0.91538944
Akaike Information Criterion	3118.13785
Schwarz Bayesian Information Criterion	3123.90345
Amemiya's Prediction Criterion	1.81559E10
Maximum Error	354187.817
Minimum Error	-288559.88
Maximum Percent Error	7.23623991
Minimum Percent Error	-6.5895583
Mean Error	20971.6104






```
where 1990 <= year(date) <= 2000;
forecast passengers / model=seasonal ;
id date  interval=month;
```

SIMPLE
performs simple (single) exponential smoothing.

DOUBLE
performs double (Brown) exponential smoothing.

LINEAR
performs linear (Holt) exponential smoothing.

DAMPTREND
performs damped trend exponential smoothing.

ADDSEASONAL | SEASONAL
performs additive seasonal exponential smoothing.

MULTSEASONAL
performs multiplicative seasonal exponential smoothing.

WINTERS
uses the Winters multiplicative method.

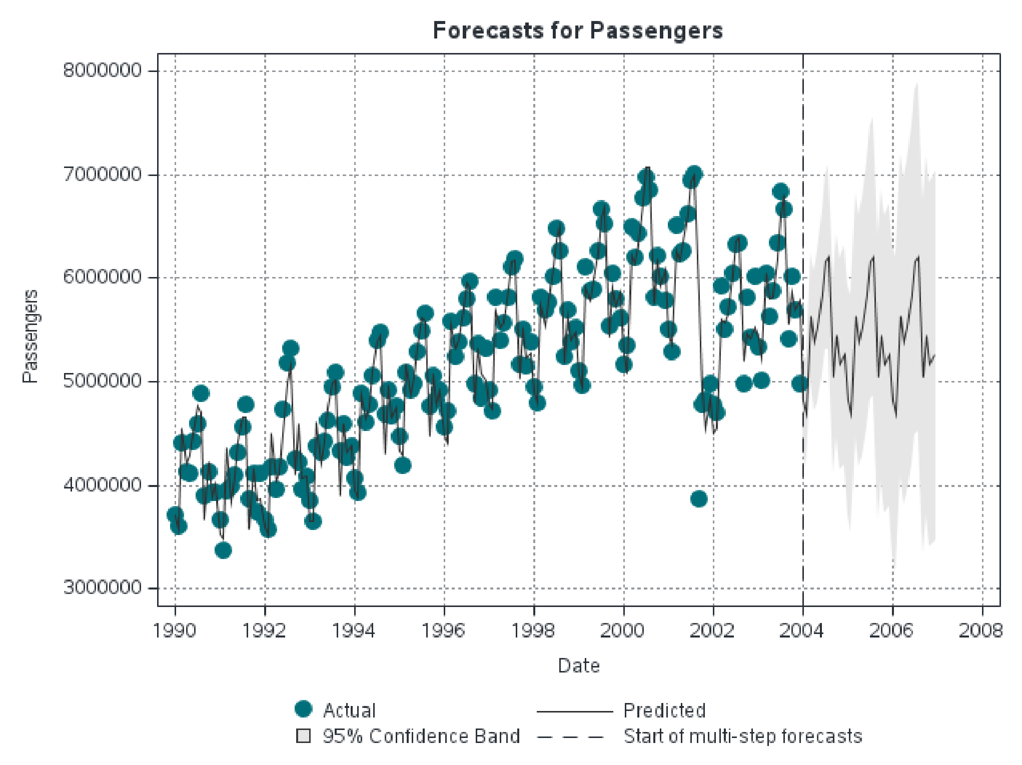
ADDWINTERS
uses the Winters additive method.

```
where 1990 <= year(date) <= 2000;
forecast passengers / model=addwinters ;
id date  interval=month;
```

Consider to use more advanced models for complicated problems

```
proc esm data=work.flights_911 plot=(Forecasts)
                    lead=36 outfor=work.ESM_FC1 ;
 where 1990 <= year(date) <= 2003;
 forecast passengers / model=seasonal ;
 id date  interval=month;
run;
```

PROC ESM cannot use input variables



Study the autocorrelation structure of your time series data with the IDENTIFY statement in PROC ARIMA

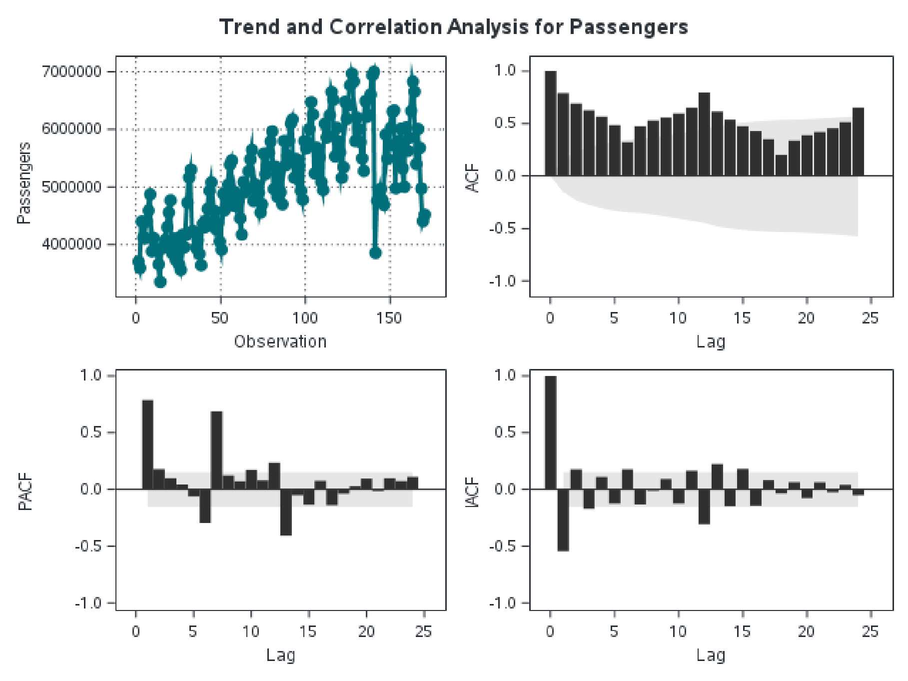
```
proc arima data=work.flights_911;
 identify var=passengers  ;
run;
```

The ARIMA Procedure

Name of Variable = Passengers	
Mean of Working Series	5154511
Standard Deviation	858150
Number of Observations	170

Autocorrelation Check for White Noise

To Lag	Chi-Square	DF	Pr > ChiSq	Autocorrelations					
6	376.97	6	<.0001	0.790	0.691	0.625	0.568	0.487	0.323
12	784.05	12	<.0001	0.473	0.529	0.558	0.596	0.651	0.795
18	1016.37	18	<.0001	0.614	0.536	0.473	0.428	0.350	0.200
24	1280.90	24	<.0001	0.337	0.391	0.420	0.454	0.514	0.651



Estimate model parameters and create forecastings with the ESTIMATE and the FORECAST statement in PROC ARIMA

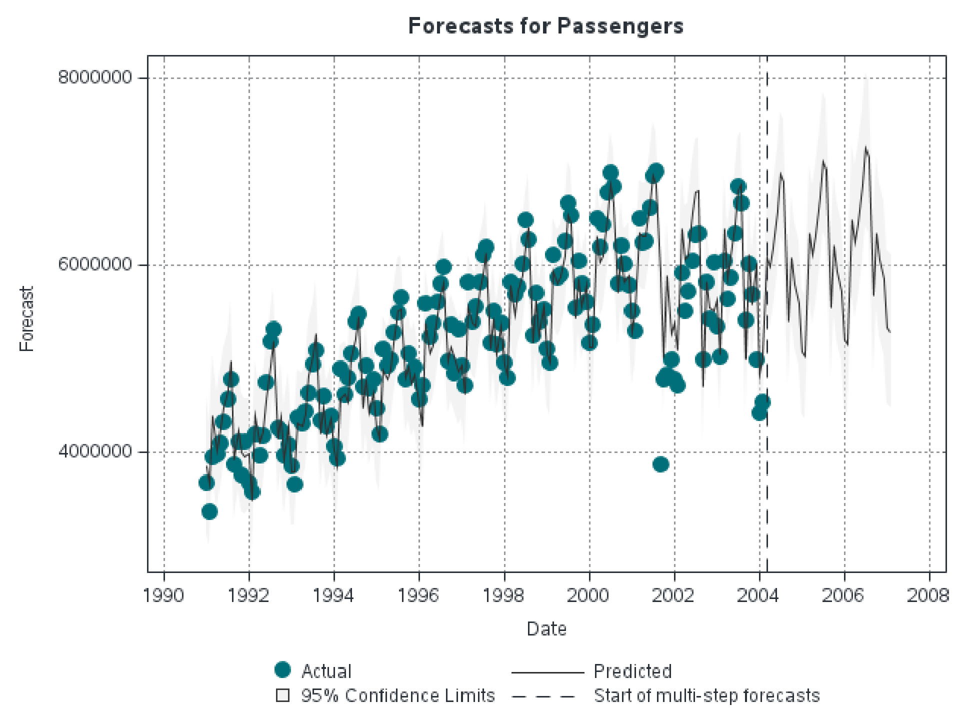
```
proc arima data=work.flights_911 plots=all;
 identify var=passengers(12)  ;
 estimate q=(1)(12)  method=ml;
 forecast id=date interval=month printal
          out=arima_fc1 lead=36;
run;
```

fit the airline model, ARIMA(0,1,1) x (0,1,1)

Maximum Likelihood Estimation

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|	Lag
MU	134099.0	20676.8	6.49	<.0001	0
MA1,1	-0.60915	0.06607	-9.22	<.0001	1
MA2,1	0.49213	0.08306	5.92	<.0001	12

Constant Estimate	134099
Variance Estimate	8.457E10
Std Error Estimate	290817.1
AIC	4430.572
SBC	4439.76
Number of Residuals	158



Use a SAS Datastep to create the intervention variable “MONTHS911”

Create a Dummy Variable with ‘1’ for SEP2001 till JAN2002

```
data work.flights_911_xt;
 set work.flights_911;
 if year(date) > 2003 then delete;
 if '01SEP2001'd <= date <= '01JAN2002'd then Months911 = 1;
 else Months911=0;
run;
```

Create records for future periods with values for that variable

```
data  work.flights_911_future;
 do lead = 1 to 36;
    Date = intnx('Month','01JAN2004'd,Lead-1);
    Months911 = 0;
    output;
 end;
 drop lead;
run;
```

Append these two datasets to be used in PROC ARIMA

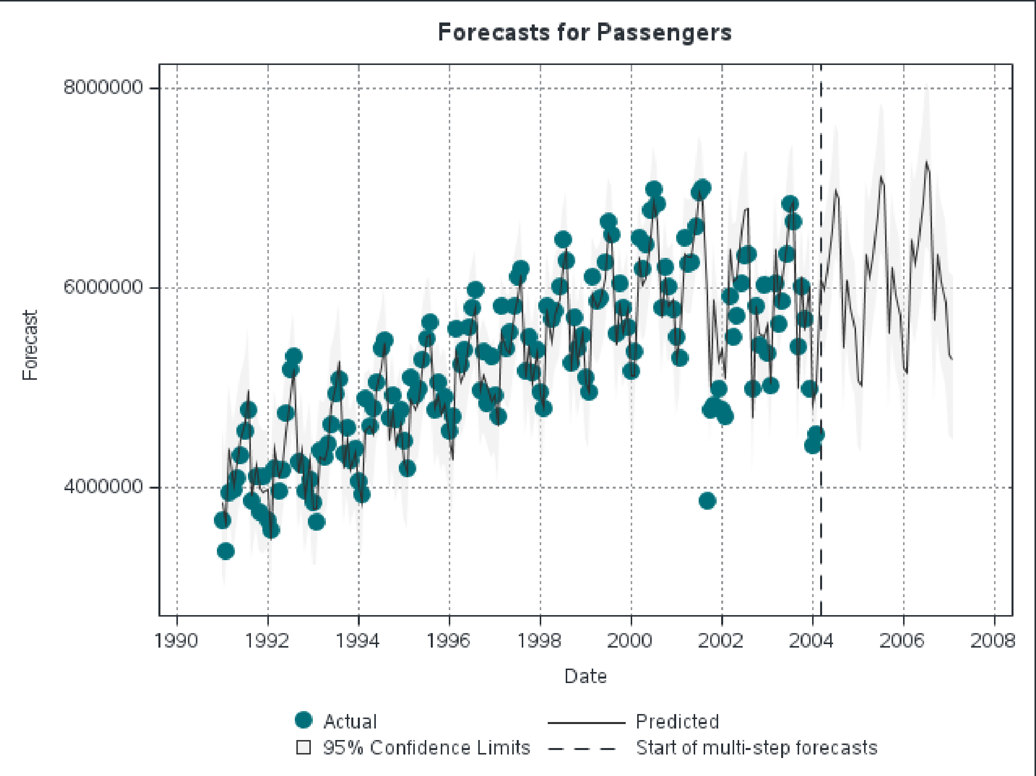
```
data work.flights_911_xt_plus;
 set work.flights_911_xt
     work.flights_911_future
;
run;
```

Use variable MONTHS911 in the IDENTIFY and ESTIMATE statement

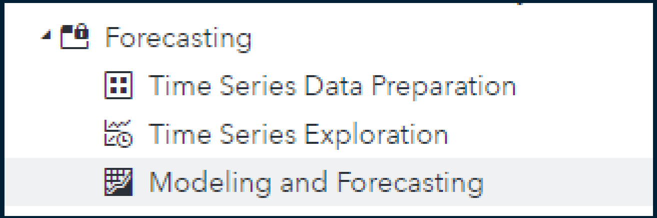
```
proc arima data=work.flights_911_xt_plus plots=all  ;
 where 1990 <= year(date) <= 2003;
 identify var=passengers(12) crosscorr=(months911) ;
 estimate q=(1)(12)  method=ml input=Months911;
 forecast id=date interval=month printall out=arima_fc2_
run;
```

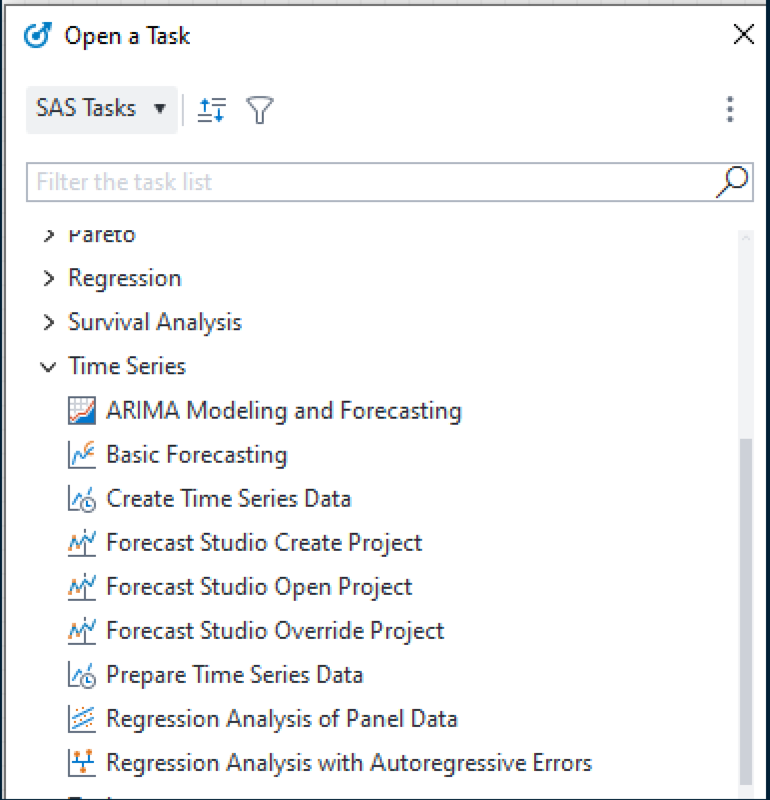
Maximum Likelihood Estimation

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|	Lag	Variable	Shift
MU	173311.6	16575.0	10.46	<.0001	0	Passengers	0
MA1,1	-0.57200	0.07379	-7.75	<.0001	1	Passengers	0
MA2,1	0.53677	0.09122	5.88	<.0001	12	Passengers	0
NUM1	-981861.9	130888.6	-7.50	<.0001	0	Months911	0



Use TASKS in SAS Studio and SAS Enterprise Guide to automatically write the procedure syntax



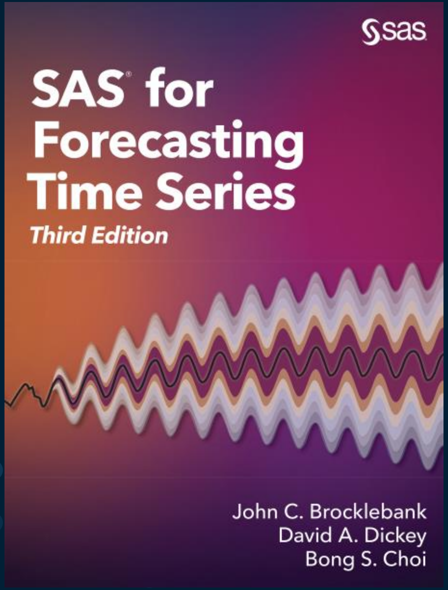


Get into timeseries foresting modeling details with books from SAS Press

SAS for Forecasting Time Series, Third Edition

John C. Brocklebank, Ph.D.; David A. Dickey, Ph.D.; Bong Choi

EISBN13: 9781629605449



https://sasinstitute.redshelf.com/app/ecom/book/1878352/sas-for-forecasting-time-series-third-edition-1878352-9781629605449-john-c-brocklebank-phd-david-a-dickey-phd-bong-choi

Transactional Data or Timeseries Data?

Transactional

one record per event/case/…

	Session Identifier	requested_file
1	43d0a4da826149b5 2002-02-17 08:38:12	/Home.jsp
2	43d0a4da826149b5 2002-02-17 08:38:12	/Cookie_Check.jsp
3	43d0a4da826149b5 2002-02-17 08:38:12	/Home.jsp
4	43d0a4da826149b5 2002-02-17 08:38:12	/Corporate_Relations.jsp
5	43d0a4da826149b5 2002-02-17 08:38:12	/Retail_Store.jsp
6	43d0a4da826149b5 2002-02-17 08:38:12	/Store/Store_Locations.jsp
7	43d639ebce6c73d8 2002-02-17 23:43:16	/Home.jsp
8	43d639ebce6c73d8 2002-02-17 23:43:16	/Cookie_Check.jsp
9	43d639ebce6c73d8 2002-02-17 23:43:16	/Home.jsp
10	43d639ebce6c73d8 2002-02-17 23:43:16	/Department.jsp
11	43d639ebce6c73d8 2002-02-17 23:43:16	/Department.jsp
12	43bb8704bb370e09 2002-02-17 13:44:04	/Home.jsp
13	43bb8704bb370e09 2002-02-17 13:44:04	/Home.jsp
14	43bb8704bb370e09 2002-02-17 13:44:04	/Subcategory.jsp
15	43bb8704bb370e09 2002-02-17 13:44:04	/Product.jsp
16	43bb8704bb370e09 2002-02-17 13:44:04	/Department.jsp
17	43bb8704bb370e09 2002-02-17 13:44:04	/Product.jsp
18	43bb8704bb370e09 2002-02-17 13:44:04	/Department.jsp

Timeseries Data

data accumulated to time intervals

	Time	NumberOfReqestedFiles
1	1:00:00	116
2	2:00:00	93
3	3:00:00	17
4	4:00:00	158
5	6:00:00	30
6	7:00:00	66
7	8:00:00	210
8	9:00:00	130
9	10:00:00	143
10	11:00:00	298
11	12:00:00	239
12	13:00:00	145

Aggregating data for time series analysis

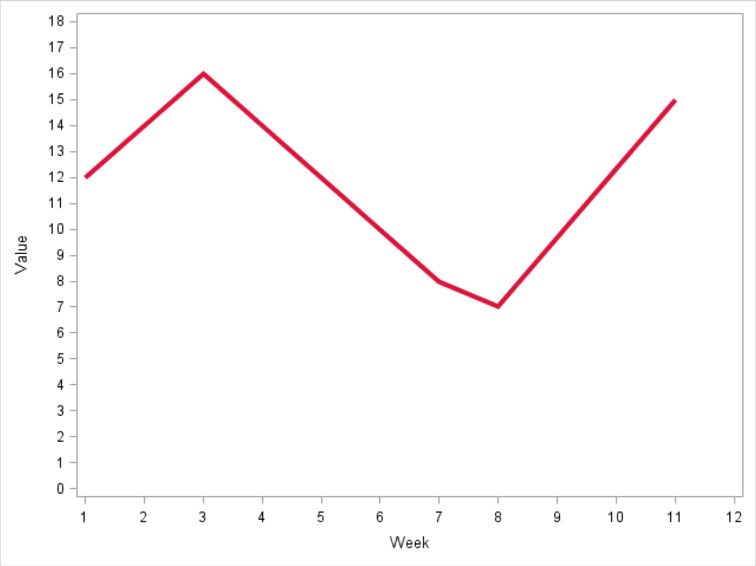
- PROC SQL
- PROC MEANS / PROC SUMMARY
- (SAS Datastep)

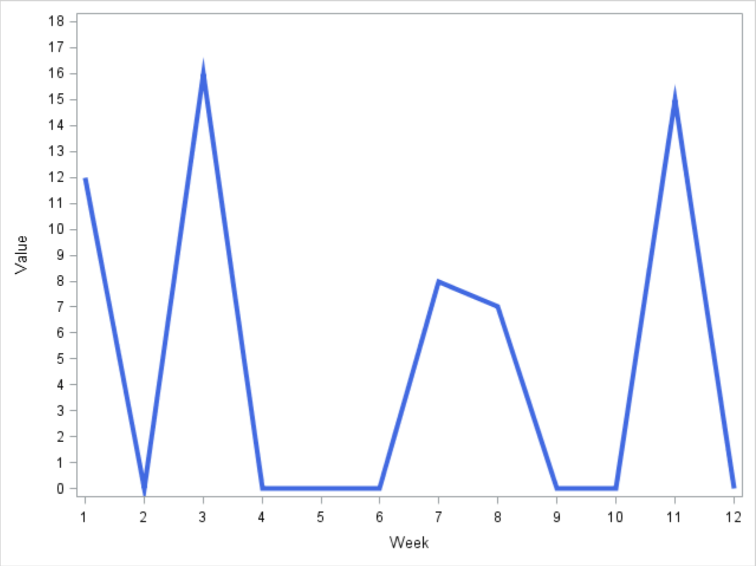
- PROC TIMESERIES

Overview: TIMESERIES Procedure

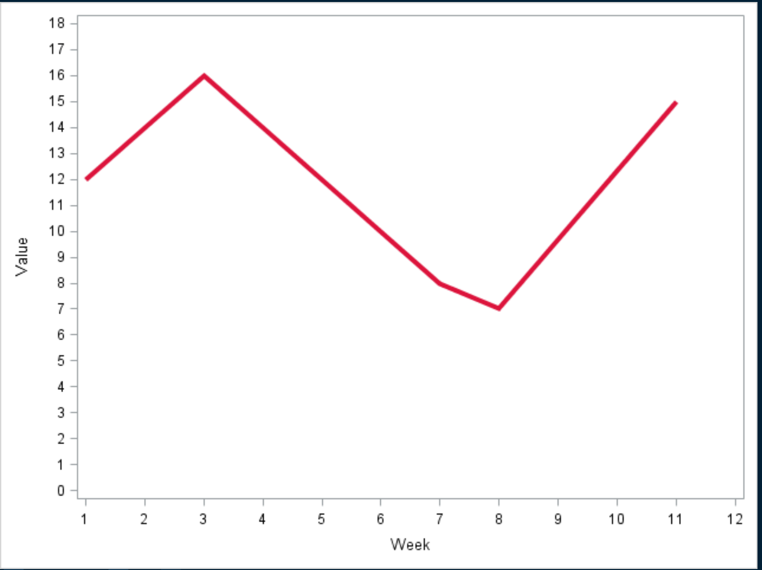
The TIMESERIES procedure analyzes time-stamped transactional data with respect to time and accumulates the data into a time series format. The procedure can perform trend and seasonal analysis on the transactions. After the transactional data are accumulated, time domain and frequency domain analysis can be performed on the accumulated time series.

Are these two graphs based on the same data?





For some measurements (inventory data) this might be the appropriate view

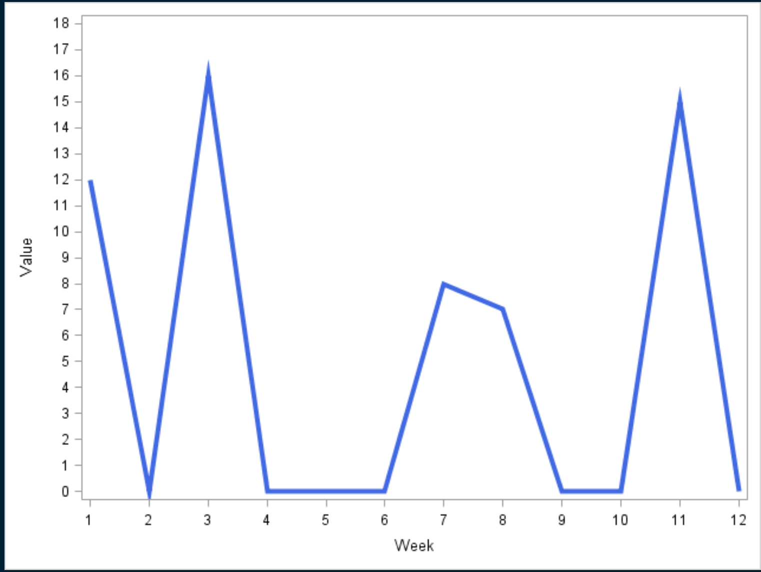


	Week	Value
1	1	12
2	3	16
3	7	8
4	8	7
5	11	15

For other measurements (movement data) this might be the appropriate view

Be careful with line-charts and missing values!

	Week	Value
1	1	12
2	2	.
3	3	16
4	4	.
5	5	.
6	6	.
7	7	8
8	8	7
9	9	.
10	10	.
11	11	15
12	12	.



Explicit or implicit missing values in longitudinal data

PNR	date	amount
56	2004-02-01	48
56	2004-03-01	51
56	2004-04-01	42
56	2004-05-01	36
56	2004-06-01	6
56	2004-07-01	.
56	2004-08-01	48
56	2004-09-01	36
56	2004-10-01	66
56	2004-11-01	15
56	2004-12-01	33
58	2005-06-01	39
58	2005-07-01	63
58	2005-08-01	84
58	2005-09-01	18
58	2005-12-01	69
58	2006-03-01	0
58	2006-07-01	90
58	2006-10-01	57
58	2007-01-01	48

Existing Record
Value Missing

Missing Record
No Continuity

Detect Missing Values and Zero Values in your Timeseries Data

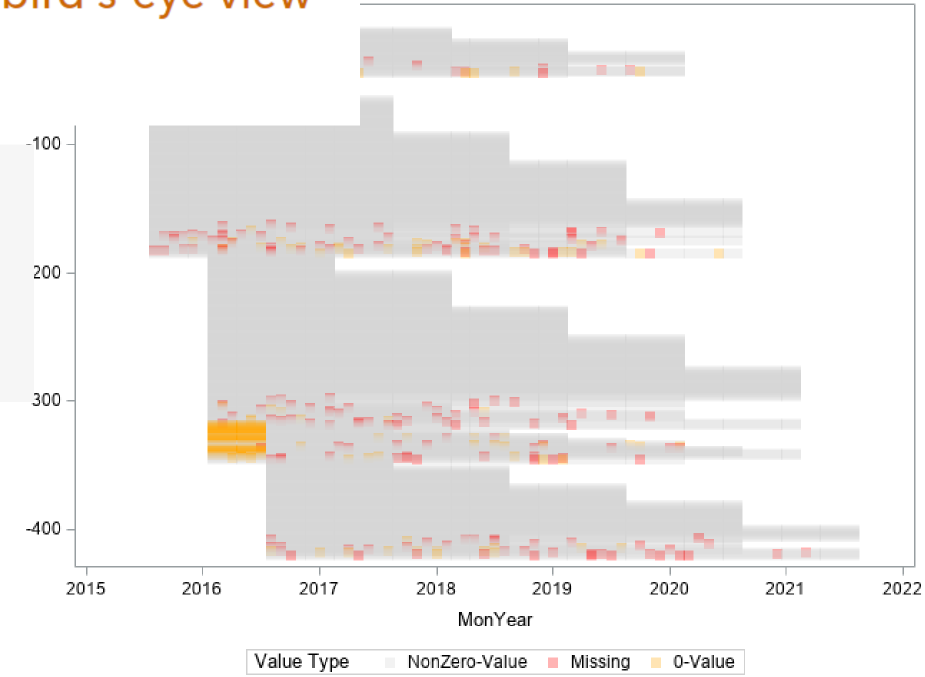
Home ▸ SAS Communities Library ▸ Have a look at your TIMESERIES data from a bird's-...

Article Options ▾ Bookmark Subscribe RSS Feed

Have a look at your TIMESERIES data from a bird's-eye view - Profile their missing value structure

Posted 02-25-2021 03:30 PM | by gsvolba (4205 views)

```
%Profile_TS_MV(data = timeseries_demodata,
               id    = tsid,
               date  = MonYear,
               value= Quantity,
               mv    = (.),
               zv    = (0), scatterplot=YES, NMAX_TS=1000);
```



https://communities.sas.com/t5/SAS-Communities-Library/Have-a-look-at-your-TIMESERIES-data-from-a-bird-s-eye-view/ta-p/717449

Replacing and interpolating missing values in longitudinal data with SAS

Insert missing records | Replace with 0 | Replace with last known value | Replace with mean | Interpolate based on splines

	DATE	air_mv	air_mv_zero	air_mv_previous	air_mv_mean	air_expand
1	JAN49	112	112	112	112	112
2	FEB49	118	118	118	118	118
3	MAR49	132	132	132	132	132
4	APR49	129	129	129	129	129
5	MAY49	.	0	129	284.54385965	128.29783049
6	JUN49	135	135	135	135	135
7	JUL49	.	0	135	284.54385965	144.73734152
8	AUG49	148	148	148	148	148
9	SEP49	136	136	136	136	136
10	OCT49	119	119	119	119	119
11	NOV49	.	0	119	284.54385965	116.19900978
12	DEC49	118	118	118	118	118
13	JAN50	115	115	115	115	115
14	FEB50	126	126	126	126	126
15	MAR50	141	141	141	141	141

Use PROC TIMESERIES and PROC EXPAND for these tasks!

Aggregation and Processing of Data in One Step with the TIMESERIES Procedure

```
proc timeseries data = air_missing
 out = air_setmissing_zero;
 id date interval =month setmiss=0;
 var air_MV;
run;
```

```
proc timeseries data = air_missing
 out = air_setmissing_mean;
 id date interval =month setmiss=MEAN;
 var air_MV;
run;
```

```
proc timeseries data = air_missing
 out = air_setmissing_previous;
 id date interval =month setmiss=PREVIOUS;
 var air_MV;
run;
```

Option value	Missing values are set to
<number>	Any number. (for example, 0 to replace missing values with zero)
MISSING	Missing
MINIMUM	Minimum value of the time series
FIRST	First non-missing value
NEXT	Next non-missing value

Convert Leading and Trailing Zeros to Missing Values

	DATE	sales
1	JAN49	0
2	FEB49	0
3	MAR49	0
4	APR49	0
5	MAY49	0
6	JUN49	0
7	JUL49	148
8	AUG49	148
9	SEP49	136
10	OCT49	119
11	NOV49	104
12	DEC49	118
13	JAN50	115

	DATE	sales
1	JAN1949	.
2	FEB1949	.
3	MAR1949	.
4	APR1949	.
5	MAY1949	.
6	JUN1949	.
7	JUL1949	148
8	AUG1949	148
9	SEP1949	136
10	OCT1949	119
11	NOV1949	104
12	DEC1949	118
13	JAN1950	115

```
proc timeseries
    data=sales_original
    out=sales_corrected;
id date interval=month
        zeromiss=both;
var sales;
run;
```

Two related Articles at Communities.sas.com

SAS COMMUNITIES

Using the TIMESERIES procedure to check the continuity of your timeseries data

Posted a week ago (562 views)

PROC_TIMESERIES_INSERT_RECORDS.sas CHECK_TIMEID_Macro.sas

This articles illustrates how you can use the TIMESERIES procedure to check whether your timeseries data contain a record for every time period and how to periods. The article illustrates the rationale for checking your timeseries data for missing records and introduces the %CHECK_TIMEID macro that automates time series data and inserting records.

Note that the TIMESERIES procedure is part of the SAS/ETS package, thus you only can run the code if you have SAS/ETS licensed. You could create a wor a SAS Datastep, however as soon as you have BY-groups in your data your SAS Datastep code gets complicated.

MISSING RECORDS or MISSING VALUES?

PNR	date	amount
56	2004-02-01	48

https://communities.sas.com/t5/SAS-Communities-Library/Using-the-TIMESERIES-procedure-to-check-the-continuity-of-your/ta-p/714678

Replace MISSING VALUES in TIMESERIES DATA using PROC EXPAND and PROC TIMESERIES

Posted yesterday (210 views)

REPLACE_MV_with_PROC_EXPAND_and_TIMESERIES.sas

This article illustrates how you can use the EXPAND and the TIMESERIES procedure to replace missing values in timeseries data. A separate SAS Communities article "U TIMESERIES procedure to check the continuity of your timeseries data" focuses on the problem of missing records in your analysis data.
Note that in order to run PROC TIMESERIES and PROC EXPAND you need SAS/ETS.

Replacing Missing Values with PROC TIMESERIES

This section discusses using the TIMESERIES procedure to replace missing values in time series data. Missing values in this context mean that the missing values occur time series data where the value for a certain time period is missing.

PROC TIMESERIES allows you to replace missing values by using one of the replacement methods listed in the table below. These methods are controlled with the optio SETMISS. For details, refer to the documentation of PROC TIMESERIES, section ID statement, SETMISS option.

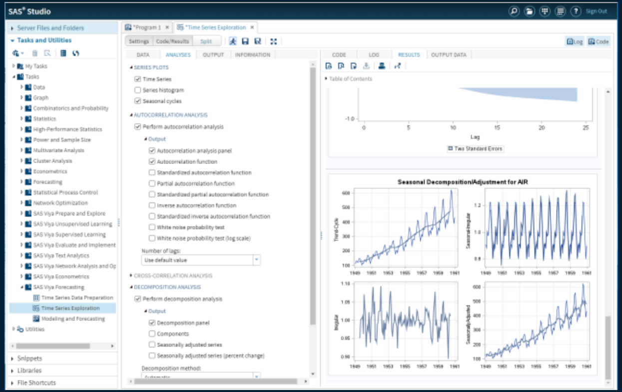
Option value	Missing values are set to
<number>	Any number. (for example, 0 to replace missing values with zero)

https://communities.sas.com/t5/SAS-Communities-Library/Replace-MISSING-VALUES-in-TIMESERIES-DATA-using-PROC-EXPAND-and/ta-p/714806

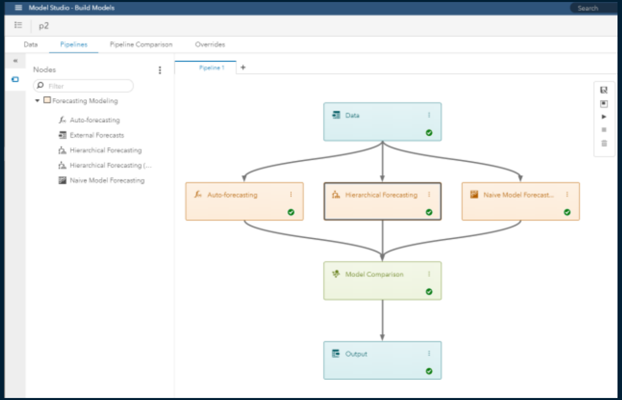
[SGF-Paper: Want an Early Picture of the Data Quality Status of Your Analysis Data? SAS® Visual Analytics Shows You How]()

Methods to Perform Forecasting in SAS

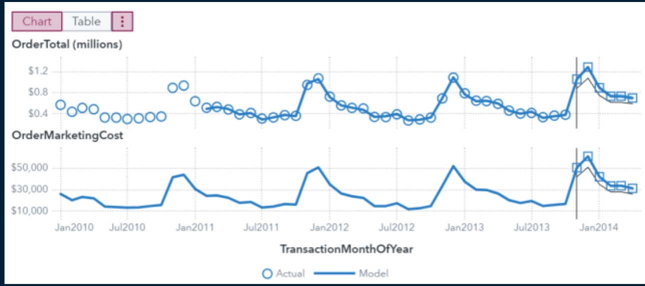
SAS Programming and SAS Studio Tasks



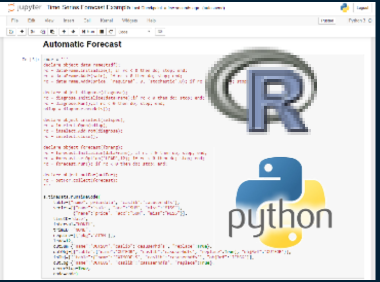
Visual Forecasting Pipelines/Automated Forecasting



Forecasting Object in SAS Visual Analytics



Open-Source Integration



- Distribute open-source algorithms in Viya nodes
- Use APIs (like Python & R) to call SAS Viya algorithms

SAS® Visual Forecasting

Key Capabilities

AUTOMATION

Automatically generate, manage, and deploy large numbers of trustworthy forecasts using a future-proof solution you can trust and configure.

CUTTING-EDGE TECHNIQUES

Employ time-series, machine-learning, hybrid (ML + time series), and deep learning techniques to improve your forecasting accuracy.

EXTERNAL DRIVERS

Incorporate events, holidays, and external drivers and let the system automatically choose which are important.

EMPOWER OPEN-SOURCE

Scale open-source algorithms to run in parallel in the cloud, and in a consistent framework.

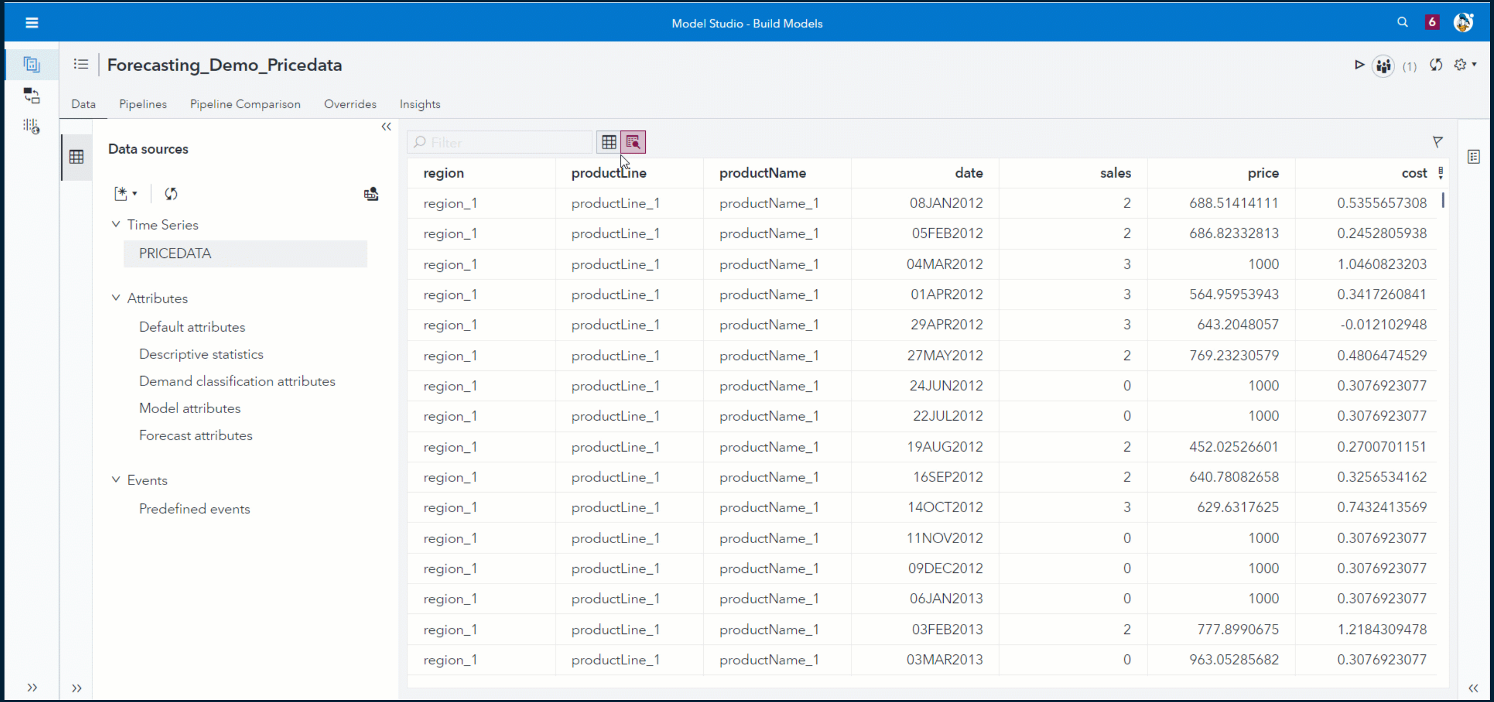
BUSINESS KNOWLEDGE

Using your business knowledge, apply overrides in a flexible manner.

CUSTOMER SUPPORT

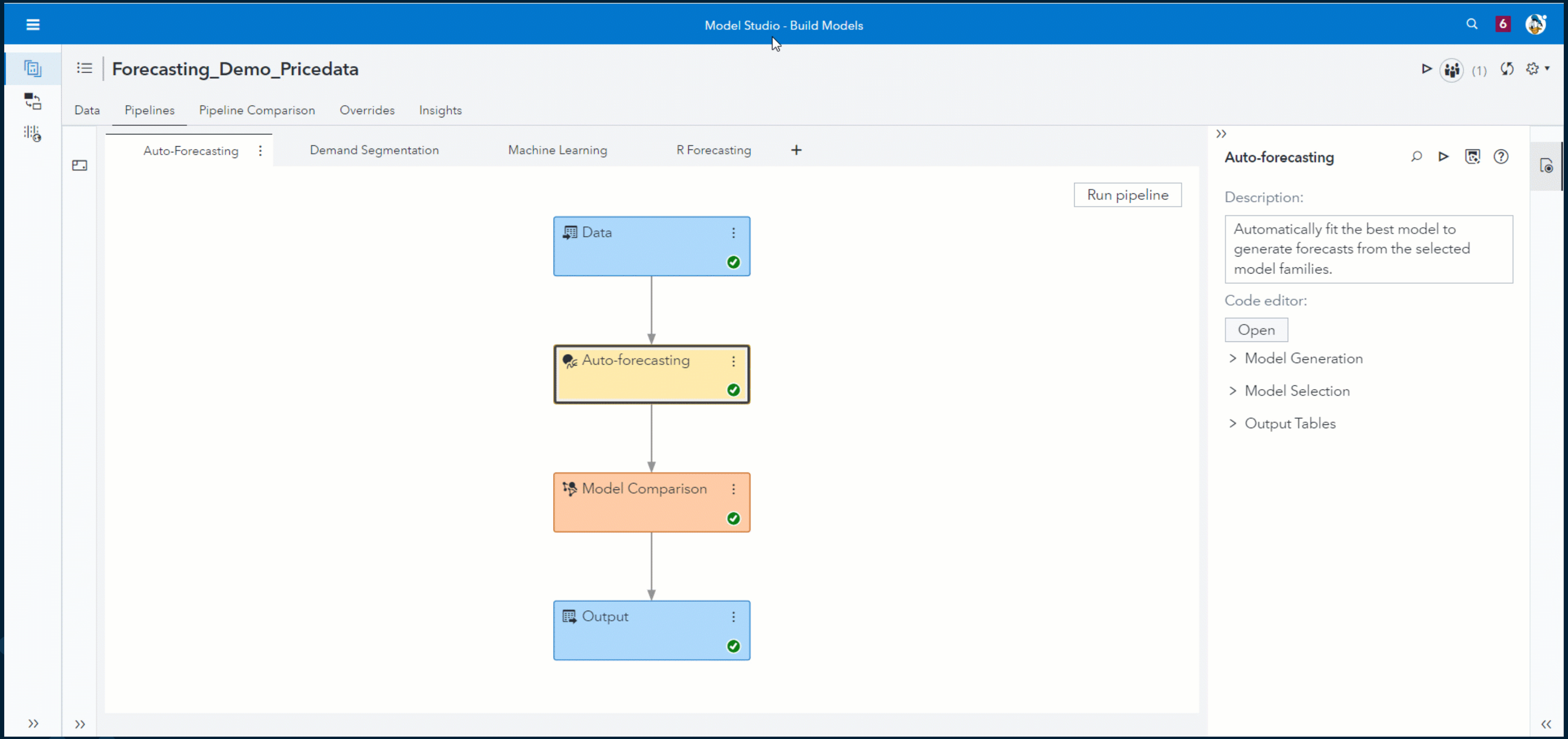
Get the support you need to succeed by taking SAS forecasting courses and joining a vibrant forecasting community.

Forecasting Project Set Up

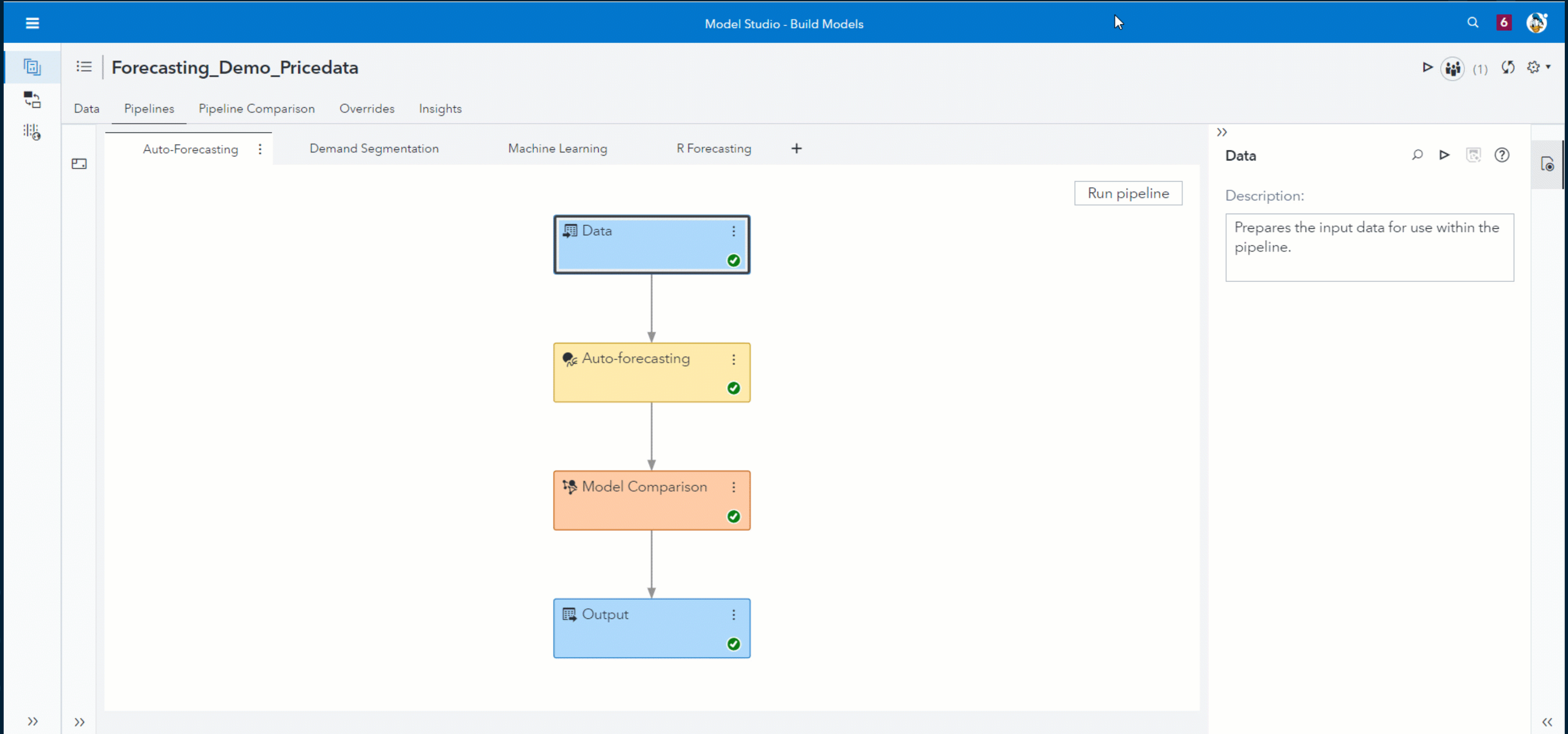


region	productLine	productName	date	sales	price	cost
region_1	productLine_1	productName_1	08JAN2012	2	688.51414111	0.5355657308
region_1	productLine_1	productName_1	05FEB2012	2	686.82332813	0.2452805938
region_1	productLine_1	productName_1	04MAR2012	3	1000	1.0460823203
region_1	productLine_1	productName_1	01APR2012	3	564.95953943	0.3417260841
region_1	productLine_1	productName_1	29APR2012	3	643.2048057	-0.012102948
region_1	productLine_1	productName_1	27MAY2012	2	769.23230579	0.4806474529
region_1	productLine_1	productName_1	24JUN2012	0	1000	0.3076923077
region_1	productLine_1	productName_1	22JUL2012	0	1000	0.3076923077
region_1	productLine_1	productName_1	19AUG2012	2	452.02526601	0.2700701151
region_1	productLine_1	productName_1	16SEP2012	2	640.78082658	0.3256534162
region_1	productLine_1	productName_1	14OCT2012	3	629.6317625	0.7432413569
region_1	productLine_1	productName_1	11NOV2012	0	1000	0.3076923077
region_1	productLine_1	productName_1	09DEC2012	0	1000	0.3076923077
region_1	productLine_1	productName_1	06JAN2013	0	1000	0.3076923077
region_1	productLine_1	productName_1	03FEB2013	2	777.8990675	1.2184309478
region_1	productLine_1	productName_1	03MAR2013	0	963.05285682	0.3076923077

Auto Forecasting



Advanced Forecasting Techniques



Pipeline Comparison, Overrides and Insights

Model Studio - Build Models

Forecasting_Demo_Pricedata

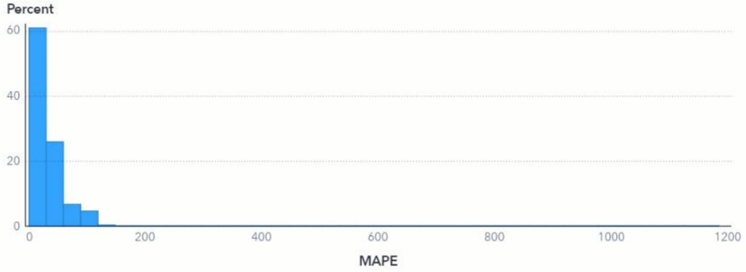
Data | Pipelines | Pipeline Comparison | Overrides | Insights

Filter

Compare

Champion	Pipeline	WMAE	WMAPE	WMASE	WASE	WRMSE	WAPE
★	Machine Learning	32.791	16.211	85,212.000	1.120	41.969	0.111
	Auto-Forecasting	39.677	19.736	2,445.894	1.153	50.671	0.125
	Demand Segmentation	37.712	15.556	0.480	1.267	47.217	0.128
	R Forecasting	35.239	15.306	0.449	1.147	44.608	0.117

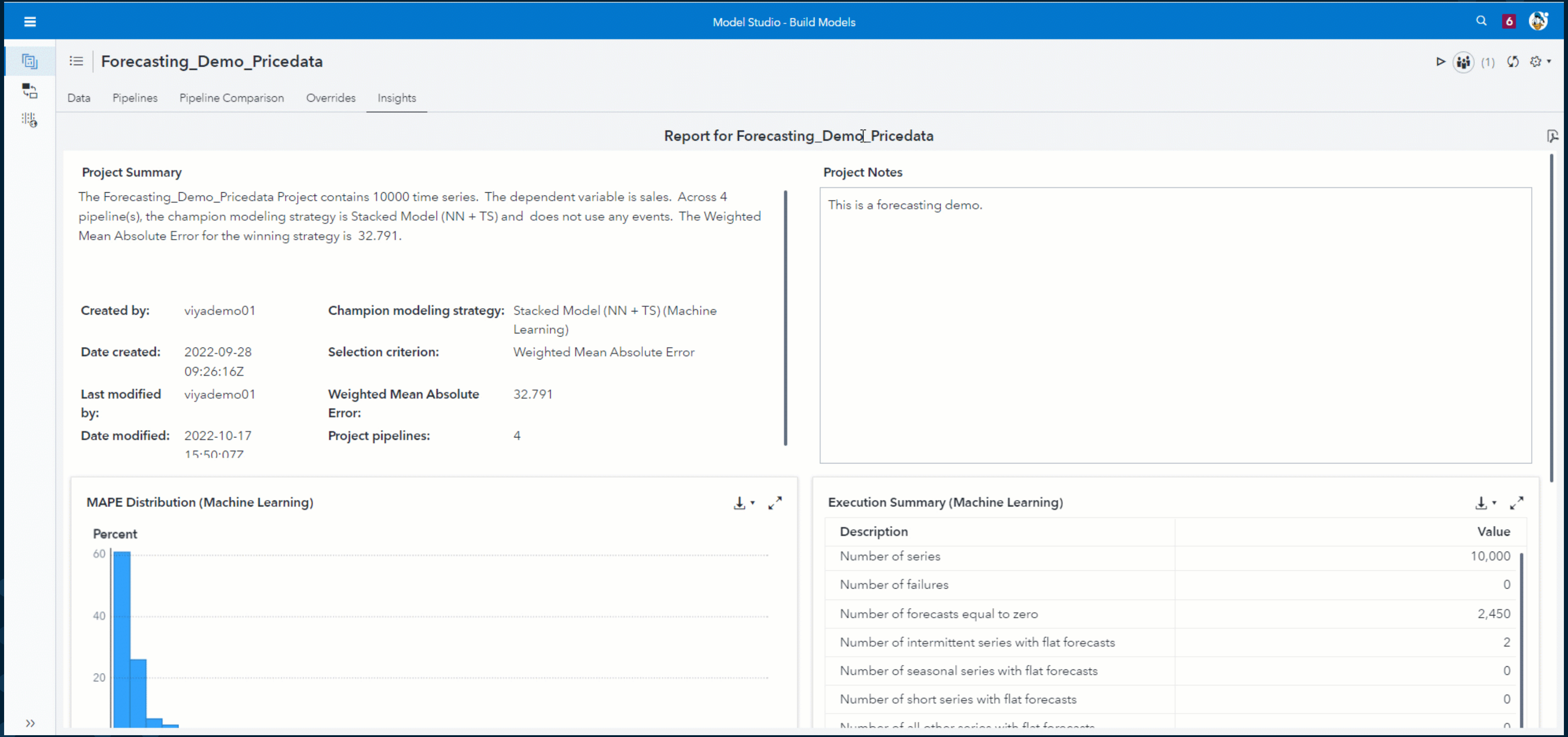
MAPE Distribution (Machine Learning)



Execution Summary (Machine Learning)

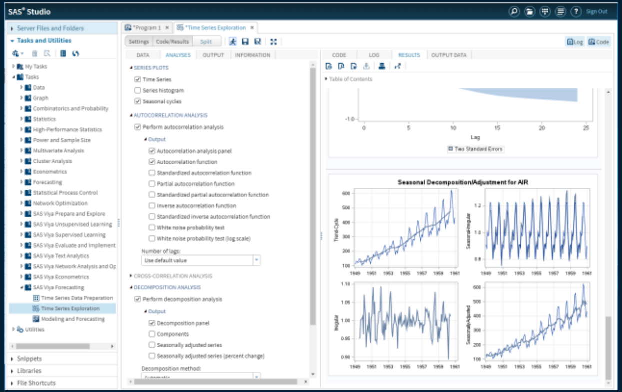
Description	Value
Number of series	10,000
Number of failures	0
Number of forecasts equal to zero	2,450
Number of intermittent series with flat forecasts	2
Number of seasonal series with flat forecasts	0
Number of short series with flat forecasts	0
Number of all other series with flat forecasts	0
Number of forecasts below 70%	861

Deployment to Production

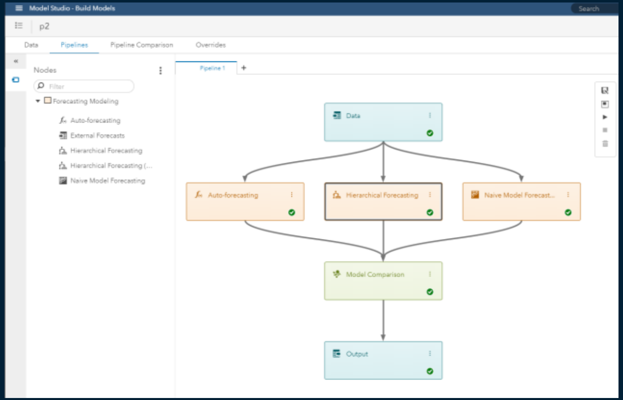


Methods to Perform Forecasting in SAS

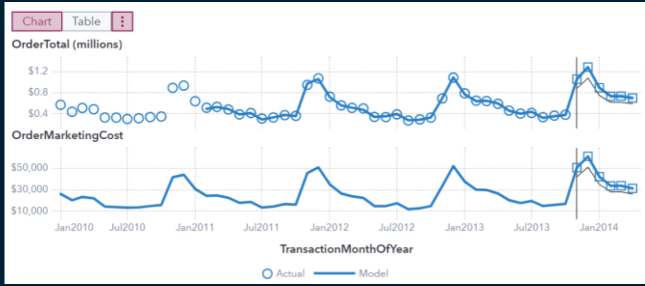
SAS Programming and SAS Studio Tasks



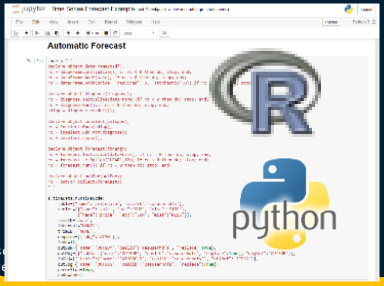
Visual Forecasting Pipelines/Automated Forecasting



Forecasting Object in SAS Visual Analytics

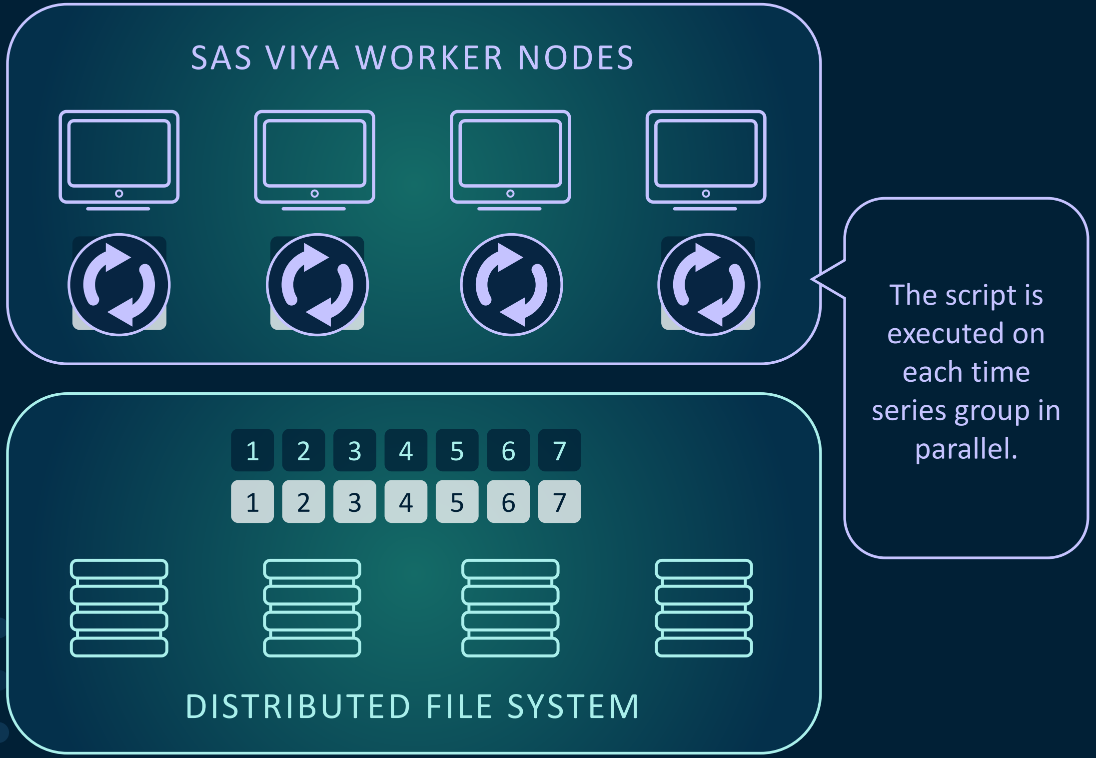


Open-Source Integration



- Distribute open-source algorithms in Viya nodes
- Use APIs (like Python & R) to call SAS Viya algorithms

Empower Open-Source Users



Distribute **native Python and R code, along with SAS code,** to run in parallel in the cloud

Scale **open-source algorithms** for large volumes of time series

Take advantage of all the **coding talent** in the organization

Apply, compare, and put into production the **latest forecasting algorithms** from the open-source world

Easily reuse open-source forecasting algorithms in all business areas by **creating custom nodes** which can be embedded in SAS pipelines

Empower Open-Source Users

Python Example

Initialize

```
PROC TSMODEL DATA=mycas.pricedata OUTARRAY=mycas.outarray
 OUTSCALAR=mycas.outscalar
 OUTOBJ=(pylog=mycas.pylog);
 ID date INTERVAL = MONTH;
 BY regionname productline productname;
 VAR SALE / ACCUMULATE = AVG;
 OUTSCALAR runtime exitCode rc1 rc2 rc3 rc4 rc5 rc6;
 OUTARRAY MAVG;

 REQUIRE EXTLANG;

SUBMIT;
 /*
 * Initialize the PYTHON2 object, which is the interface to the
 * Python interpreter.
 */
 declare object py(PYTHON3);
 rc = py.Initialize();
```

Python Code

```
/*
* 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")');
```

Exchange Vars and Collect Results

```
 /* 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;
```

Empower Open-Source Users

Initialize

Python Example

```
PROC TSMODEL DATA=mycas.pricedata OUTARRAY=mycas.outarray
 OUTSCALAR=mycas.outscalar
 OUTOBJ=(pylog=mycas.pylog);
 ID date INTERVAL = MONTH;
 BY regionname productline productname;
 VAR SALE / ACCUMULATE = AVG;
 OUTSCALAR runtime exitCode rc1 rc2 rc3 rc4 rc5 rc6;
 OUTARRAY MAVG;

 REQUIRE EXTLANG;

SUBMIT;
 /*
 * Initialize the PYTHON2 object, which is the interface to the
 * Python interpreter.
 */
 declare object py(PYTHON3);
 rc = py.Initialize();
```

Empower Open-Source Users

Python Example

Python Code

```
/*
* 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")');
```

Empower Open-Source Users

Python Example

Exchange Vars and Collect Results

```
/* 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;
```

Empower Open-Source Users

Python Example

Initialize

```
PROC TSMODEL DATA=mycas.pricedata OUTARRAY=mycas.outarray
 OUTSCALAR=mycas.outscalar
 OUTOBJ=(pylog=mycas.pylog);
 ID date INTERVAL = MONTH;
 BY regionname productline productname;
 VAR SALE / ACCUMULATE = AVG;
 OUTSCALAR runtime exitCode rc1 rc2 rc3 rc4 rc5 rc6;
 OUTARRAY MAVG;

 REQUIRE EXTLANG;

SUBMIT;
 /*
 * Initialize the PYTHON2 object, which is the interface to the
 * Python interpreter.
 */
 declare object py(PYTHON3);
 rc = py.Initialize();
```

Python Code

```
/*
* 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")');
```

Exchange Vars and Collect Results

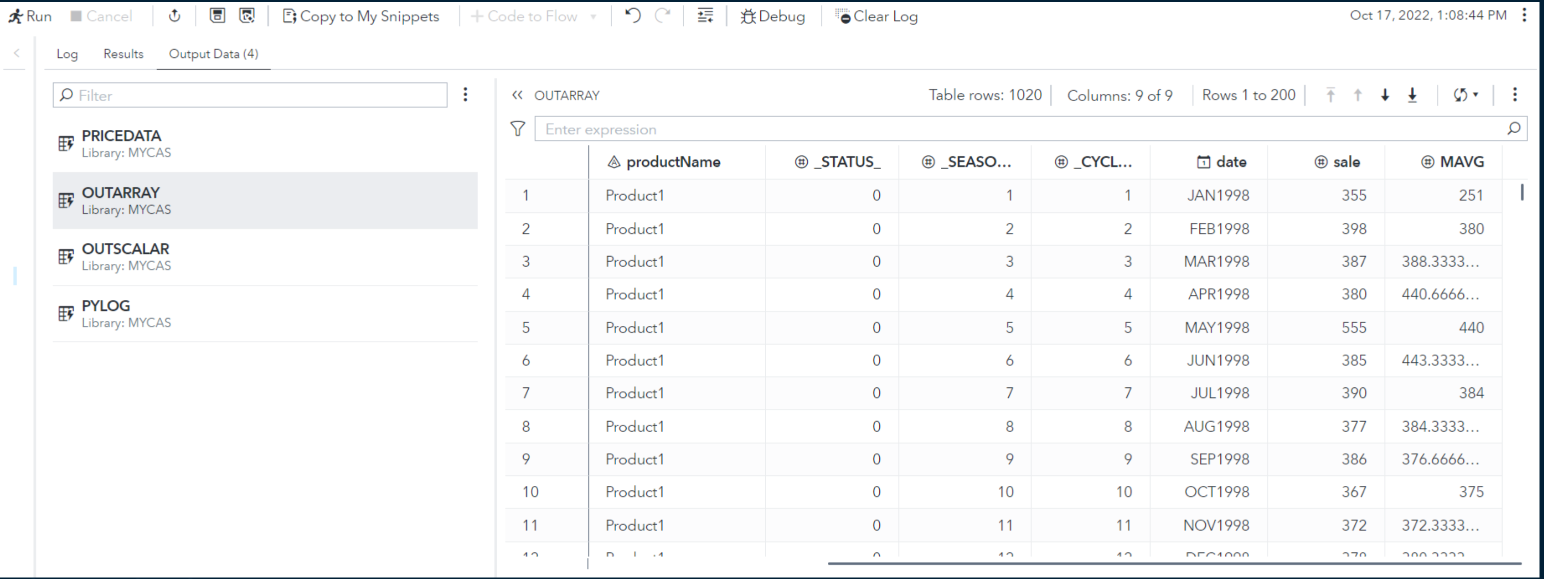
```
 /* Specify variables to share between SAS and Python.
 * The variable SALE is used only as input in the Python program;
 * the default value of READONLY is used to avoid propagating
 * its data back to SAS. MAVG is transferred back to SAS, where
 * it is stored in a CAS table for further analysis.*/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

Empower Open-Source Users

Python Example

Output



	productName	_STATUS_	_SEASO...	_CYCL...	date	sale	MAVG
1	Product1	0	1	1	JAN1998	355	251
2	Product1	0	2	2	FEB1998	398	380
3	Product1	0	3	3	MAR1998	387	388.3333...
4	Product1	0	4	4	APR1998	380	440.6666...
5	Product1	0	5	5	MAY1998	555	440
6	Product1	0	6	6	JUN1998	385	443.3333...
7	Product1	0	7	7	JUL1998	390	384
8	Product1	0	8	8	AUG1998	377	384.3333...
9	Product1	0	9	9	SEP1998	386	376.6666...
10	Product1	0	10	10	OCT1998	367	375
11	Product1	0	11	11	NOV1998	372	372.3333...

Empower Open-Source Users

R Example

Initialize

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

ID date INTERVAL = MONTH;
BY regionname productline productname;
VAR SALE / ACCUMULATE = AVG;
OUTSCALAR runtime exitCode rc1 rc2 rc3 rc4 rc5 rc6 rc7;
OUTARRAY rPred;
REQUIRE EXTLANG;

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() ;
```

R Code

```
 /* 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: */

 rc2 = robj.PushCodeFile('/shared-data/sasdata/r_arima_code.r') ;

/* 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) )
-----------------------------------------------------
```

Exchange Vars and Collect Results

```
 * 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;
```

Empower Open-Source Users

R Example

Initialize

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

ID date INTERVAL = MONTH;
BY regionname productline productname;
VAR SALE / ACCUMULATE = AVG;
OUTSCALAR runtime exitCode rc1 rc2 rc3 rc4 rc5 rc6 rc7;
OUTARRAY rPred;
REQUIRE EXTLANG;

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() ;
```

Empower Open-Source Users

R Example

R Code

```
 /* 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: */

 rc2 = robj.PushCodeFile('/shared-data/sasdata/r_arima_code.r') ;

/* 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) )
-----------------------------------------------------
```

Empower Open-Source Users

R Example

Exchange Vars and Collect Results

```
 * 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;
```

Empower Open-Source Users

R Example

Initialize

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

ID date INTERVAL = MONTH;
BY regionname productline productname;
VAR SALE / ACCUMULATE = AVG;
OUTSCALAR runtime exitCode rc1 rc2 rc3 rc4 rc5 rc6 rc7;
OUTARRAY rPred;
REQUIRE EXTLANG;

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() ;
```

R Code

```
 /* 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: */

 rc2 = robj.PushCodeFile('/shared-data/sasdata/r_arima_code.r') ;

/* 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) )
----------------------------------------------------
```

Exchange Vars and Collect Results

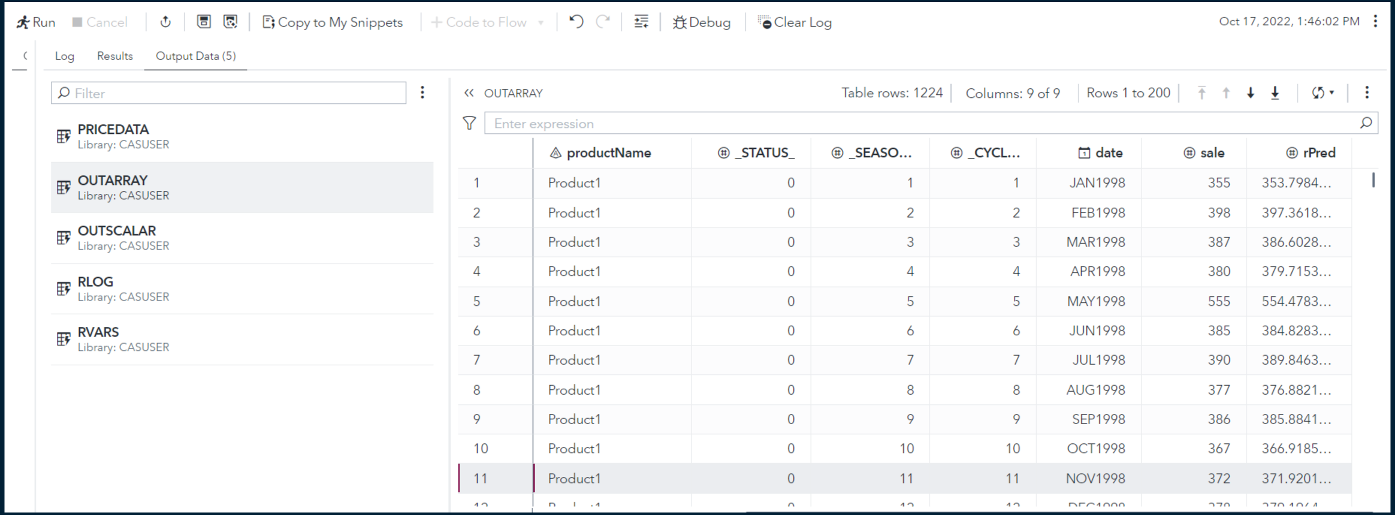
```
 * Specify variables to share between SAS and R.
 * SALE is the (READONLY) dependent variable. The ARIMA code
 * uses the generic name Y for the dependent variable, so
 * SALE is aliased to Y.
 * rPred will contain the predicted series, which is returned to the
 * SAS program. The R code that is 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;
```

Empower Open-Source Users

R Example

Output



	productName	_STATUS_	_SEASO...	_CYCL...	date	sale	rPred
1	Product1	0	1	1	JAN1998	355	353.7984...
2	Product1	0	2	2	FEB1998	398	397.3618...
3	Product1	0	3	3	MAR1998	387	386.6028...
4	Product1	0	4	4	APR1998	380	379.7153...
5	Product1	0	5	5	MAY1998	555	554.4783...
6	Product1	0	6	6	JUN1998	385	384.8283...
7	Product1	0	7	7	JUL1998	390	389.8463...
8	Product1	0	8	8	AUG1998	377	376.8821...
9	Product1	0	9	9	SEP1998	386	385.8841...
10	Product1	0	10	10	OCT1998	367	366.9185...
11	Product1	0	11	11	NOV1998	372	371.9201...

Selected SAS Training Courses

- https://support.sas.com/edu/schedules.html?crs=STSM&ctry=US - Time Series Modeling Essentials
- This course discusses the fundamentals of modeling time series data. The course focuses on the applied use of the three main model types used to analyze univariate time series: exponential smoothin...

- https://support.sas.com/edu/schedules.html?crs=FVVF&ctry=US - Forecasting Using Model Studio in SAS® Viya®
- This course provides a hands-on tour of the forecasting functionality in Model Studio, a component of SAS Viya. The course begins by showing how to load the data into memory and visualize the time ...

- https://support.sas.com/edu/schedules.html?crs=VFSP&ctry=US - Large-Scale Forecasting Using SAS® Viya®: A Programming Approach
- This course teaches students to develop and maintain a large-scale forecasting project using SAS Visual Forecasting tools. For the course project, students build and then refine a large-scale forec...

- https://support.sas.com/edu/schedules.html?crs=MTSS&ctry=US Models for Time Series and Sequential Data
- This course teaches students to build, refine, extrapolate, and, in some cases, interpret models designed for a single, sequential series. There are three modeling approaches presented. The traditi...

- https://support.sas.com/edu/schedules.html?crs=TSFM&ctry=US - Time Series Feature Mining and Creation
- In this course, you learn about data exploration, feature creation, and feature selection for time sequences. The topics discussed include binning, smoothing, transformations, and data set operatio...

Useful Resources

SAS Communities Library Articles

- Step by step guide for using open-source models in SAS VF
- How to incorporate RNNs in your SAS VF pipelines
- How to create a custom TensorFlow node in SAS VF with GUI parameters
- Modernizing Scenario Analysis with SAS Viya and SAS Visual Analytics
- Free SAS Sample Data Sets for Forecasting

SAS Papers

- Neural Network–Based Forecasting Strategies in SAS® Viya®
- Writing a Gradient Boosting Model Node for SAS® Visual Forecasting
- Scalable Cloud-Based Time Series Analysis and Forecasting Using Open-Source Software

SAS Forecasting E-Book

- Forecasting with SAS: Special Collection

Explore Helpful Resources

Ask the Expert
View other user webinars that provide insights into using SAS products to make your job easier.

FREE Training
Learn from home – free for 30 days. Get software labs to practice and online support if needed.

SAS Support Communities
Ask questions, get answers and share insights with SAS users.

SAS Analytics Explorers
An exclusive platform to collaborate, learn and share your expertise. Gain access to a diverse network to advance your career. Special rewards and recognition exclusively for SAS users.

SAS Users YouTube Channel
A plethora of videos on hundreds of topics, just for SAS users.

Newsletters
Get the latest SAS news plus tips, tricks and more.

Users Groups
Meet local SAS users, network and exchange ideas – virtually.

SAS Profile
If you haven't already done so, create your SAS Profile to access free training, SAS Support Communities, technical support, software downloads, newsletters and more.

Thank you

for joining us for
this SAS webinar