Chapter 1 Introduction to Time Series

1.1 Time Series Characteristics

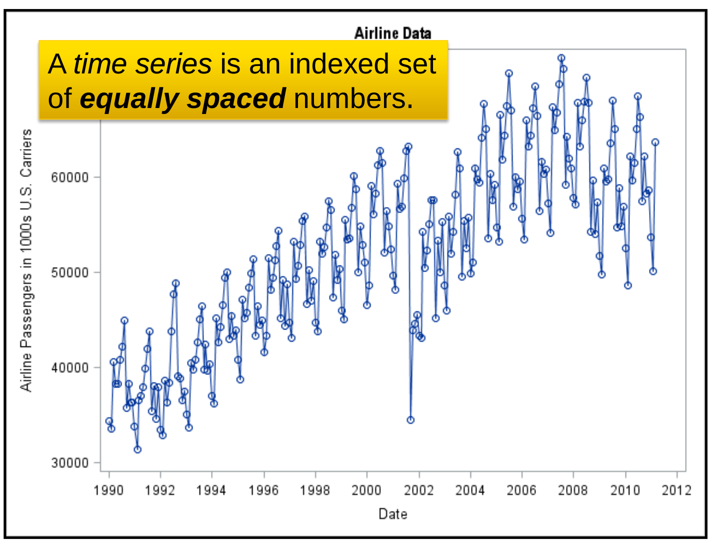
Objectives

- Define a time series.
- Describe the main ideas in time series data creation.
- Use the TIMESERIES procedure to transform transactional data into time series data (Accumulate).
- Define and explore the systematic components in a time series.

3

In business applications, time series usually start as transactional or timestamped data. An example of transactional data is a record of customer visits to a website over a period of a year. Each visit is recorded with a customer identifier and a timestamp. Transactional data are not organized with respect to a time interval. They must be made equally spaced, or indexed, before time series models can be used to quantify the systematic variation contained in it. *Accumulation* is the process of indexing or transforming transactional data into time series.

A Time Series



4

Time Series Creation

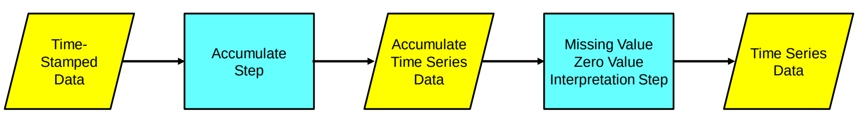
Data Accumulation:

Accumulates transactional data to a specified time interval of the data.

5

Transactional Data Preparation Steps

Before you can perform time series analysis and forecasting, the observation (raw) series must be accumulated and interpreted to form a time series. The following diagram summarizes the process of converting timestamped data to time series data:



6

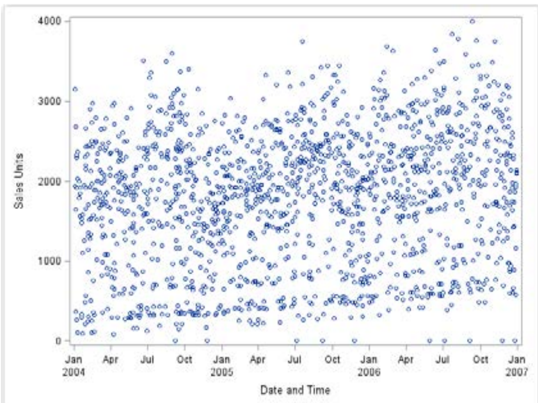
The selection of the interval and the accumulation method are critical considerations in applied time series modeling. When making these choices, the modeler *creates* the data for analysis. Different accumulation choices can emphasize different systematic characteristics of the data, and impact model usefulness and precision.

For example, consider timestamped calls into a call center. Accumulating the data to an hourly interval using a sum accumulation method gives analysts, assuming adequate volume, a good look at the hour-of-the-day cycle. It also provides direct information about what the peak and trough hours for calls are in a 24-hour cycle. This information is helpful for daily staffing decisions.

However, time series accumulated at an hourly interval might not be optimal for understanding and quantifying longer term trends and month-of-the-year cycles that might exist in the data. A better alternative for understanding and quantifying longer run patterns might be a monthly interval and an average accumulation method.

Transactional Data: Example

Historical Timestamped Data



No obvious patterns exist in the transactional data.

7

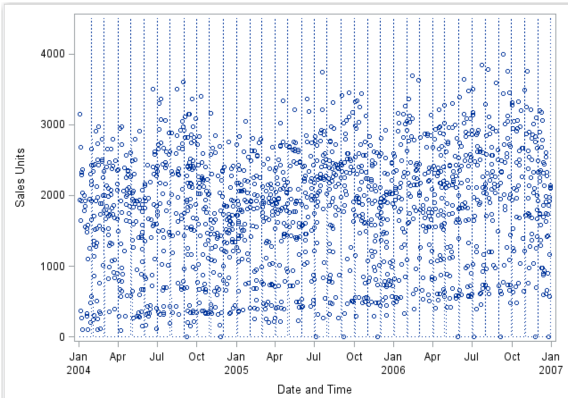
Transactional Analysis

- Given a timestamped data set, each observation of the data set can be assigned an observation (raw) index, a time index, and a season index.
- Count and frequency analysis can be applied to the timestamped data set based on these indices.
- Each of these indices does not depend on the data under analysis. These indices only structure the data for subsequent analysis.

8

Transactional Data Time Binning

Monthly Time Bins



A monthly interval time series has one observation per interval or bin.

9

Transactional Data Time Binning

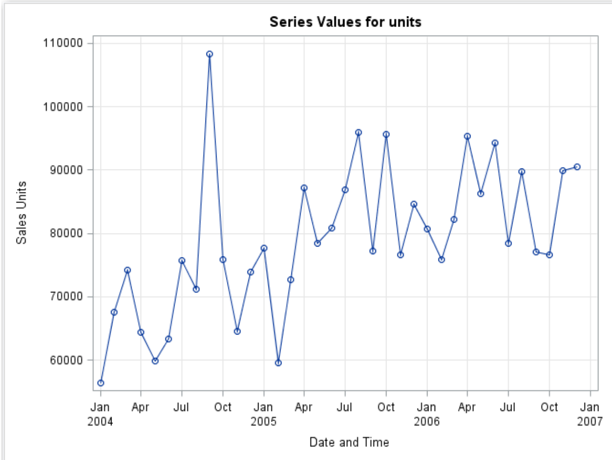
Perform time binning with the TIMESERIES procedure's INTERVAL= option. Use accumulated totals.

```
proc timeseries data=transactions out=outsum;
   id date interval=month
           accumulate=total;
   var units;
run;
```

10

Transactional Data Accumulation

Accumulated on a Monthly ***Total*** Basis



11

Transactional Data Time Binning

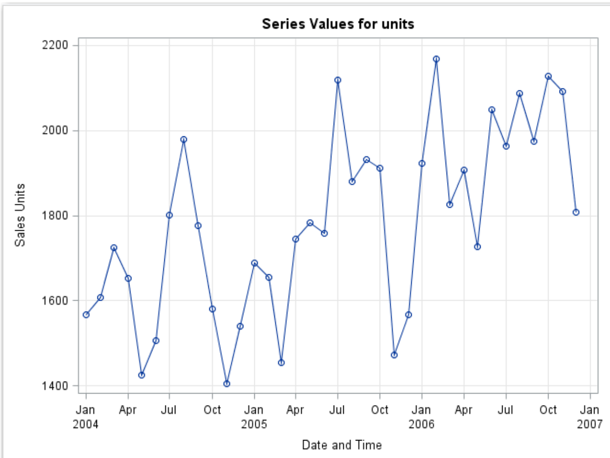
Perform time binning with the TIMESERIES procedure's INTERVAL= option. Use accumulated averages.

```
proc timeseries data=transactions out=outavg;
   id  interval=month
                accumulate=average;
   var units;
run;
```

12

Transactional Data Accumulation

Accumulated on a Monthly ***Average*** Basis



13

Transactional Data Time Binning

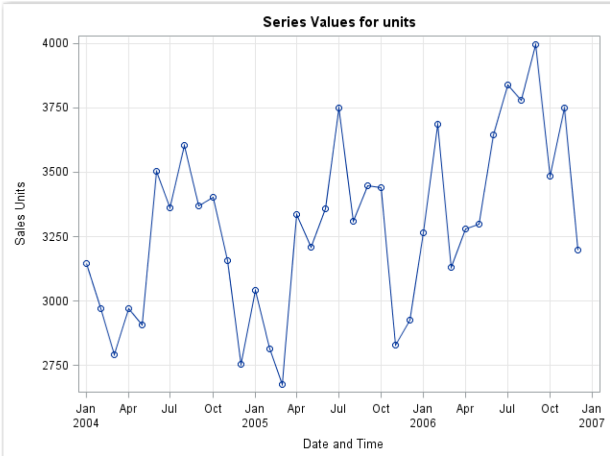
Perform time binning with the TIMESERIES procedure's INTERVAL= option. Use the maximum value in each interval.

```
proc timeseries data=transactions out=outmax;
   id  interval=month
                accumulate=maximum;
   var units;
run;
```

14

Transactional Data Accumulation

Accumulated on a Monthly **Maximum** Basis

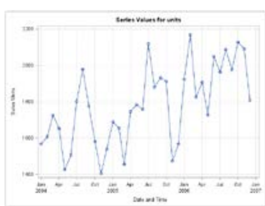


15

Transactional Data Accumulation

For this timestamped data, accumulating on a monthly ***average*** or ***maximum*** basis might be better than accumulating on a ***total*** basis.

ACCUMULATE=TOTAL



ACCUMULATE=AVERAGE

ACCUMULATE=MAXIMUM

16

Time Series Creation

This demonstration provides examples of creating time series through the process of accumulation.

The Time Series Exploration task in SAS Studio is used to illustrate this. The transactional data set for the analysis is **STSM.CH1_DEMODAT**. The dependent variable is **units** of sales. There are two timestamps: **date** is a date variable and **dtdate** is a datetime variable. A portion of the data is shown below. The transactional observations begin on 02January2004 and end on 29December2006 inclusive.

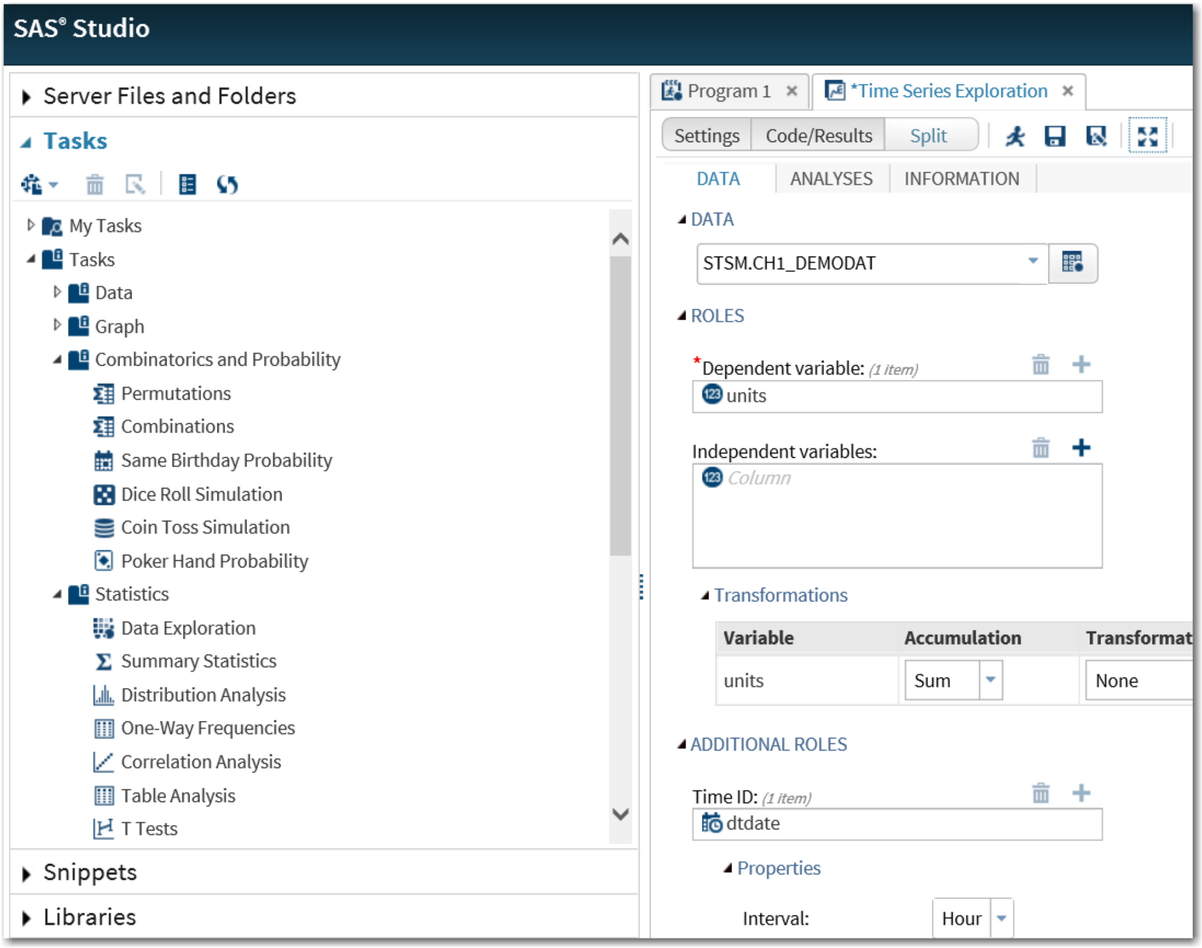
Obs	units	dtdate	date	mon	yr
1	.	02JAN2004:08:00:00	02JAN2004	1	2004
2	1940	02JAN2004:09:00:00	02JAN2004	1	2004
3	.	02JAN2004:10:00:00	02JAN2004	1	2004
4	3147	02JAN2004:11:00:00	02JAN2004	1	2004
5	.	02JAN2004:12:00:00	02JAN2004	1	2004
6	.	02JAN2004:13:00:00	02JAN2004	1	2004
7	.	02JAN2004:14:00:00	02JAN2004	1	2004
8	.	02JAN2004:15:00:00	02JAN2004	1	2004
9	.	02JAN2004:16:00:00	02JAN2004	1	2004
10	.	02JAN2004:17:00:00	02JAN2004	1	2004

1. Log on to SAS Studio. Use the credentials that are supplied by your instructor.
2. Expand **Tasks** and then expand **Forecasting Tasks**. Double-click the **Time Series Exploration** task to initiate it.
3. Click the **Select a Table** button under the Data property to navigate to the **STSM** library. Select the **CH1_DEMODAT** data table.

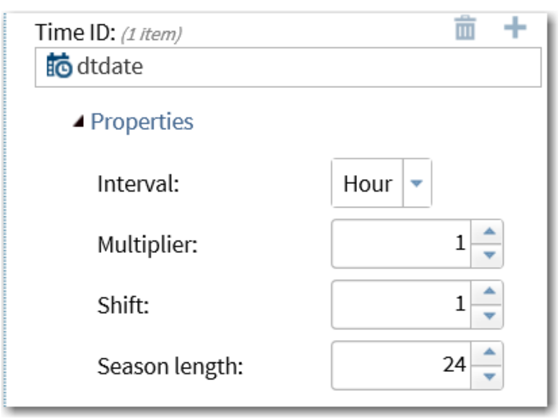
These steps are summarized in the display below.

Examining the Data on an Hourly Interval

4. Assign **units** as the dependent variable. Expand the **ADDITIONAL ROLES** property, and assign the **dtdate** column as the time ID.
5. Select **Sum** for the **Accumulation** field.



6. After the time ID variable is set, a one-hour interval that begins in the first hour of a 24-hour day is detected as the natural interval of the data. The options shown below can be changed to modify the detected interval.



The TIMEID procedure in SAS/ETS runs “under the hood” in SAS Studio to detect the interval of the data.

7. Click on the toolbar to maximize the view.

The TIMESERIES procedure in SAS/ETS provides the main functionality for the Time Series Exploration task. A subset of the contents of the CODE window is shown below.

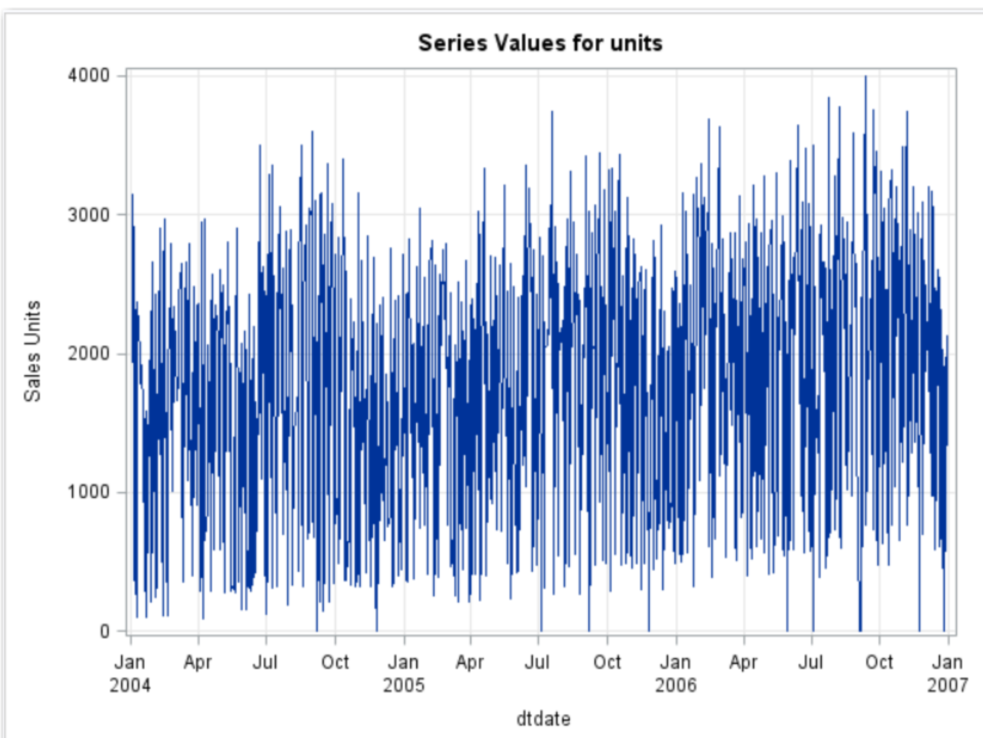
```
proc sort data=STSM.CH1_DEMODAT out=WORK.TempSorted;
   by dtdate;
run;

proc timeseries data=WORK.TempSorted seasonality=24
                plots=(series corr);
   id dtdate interval=hour;
   var units / accumulate=total transform=none dif=0 sdif=0;
run;
```

- The SORT procedure is used to create a working copy of the transactional data, and sort it from the earliest to the latest date.
- The TIMESERIES procedure is used to accumulate the transactional data to an hourly interval. The PLOTS option in the procedure statement creates the series plot and correlation panel. The ID statement defines the datetime ID variable, and the interval and accumulation options that correspond to the settings that are selected on the DATA tab.

If you produce the code by entering it directly in the editor, the code below produces the necessary output. This code assumes that the data in **stsm.ch1_demodat** are properly sorted.

```
/* accumulate to an hourly interval */
proc timeseries data=stsm.ch1_demodat out=out_totalhour
                plots=(series);
   id dtdate interval=dthour accumulate=total;
   var units;
run;
```

8. To submit the code and run the Time Series Exploration task, click the **Run** button on the toolbar. The series plot is shown below. There are many hourly intervals in a three-year time span.



9. To get a better look at the hourly intervaled time series, the data is truncated and then re-plotted. Because there is currently no option to do this in the DATA properties of SAS Studio, the PROC TIMESERIES syntax is edited directly. Click the **CODE** tab and then click **Edit**.

The syntax is now contained on a new tab.

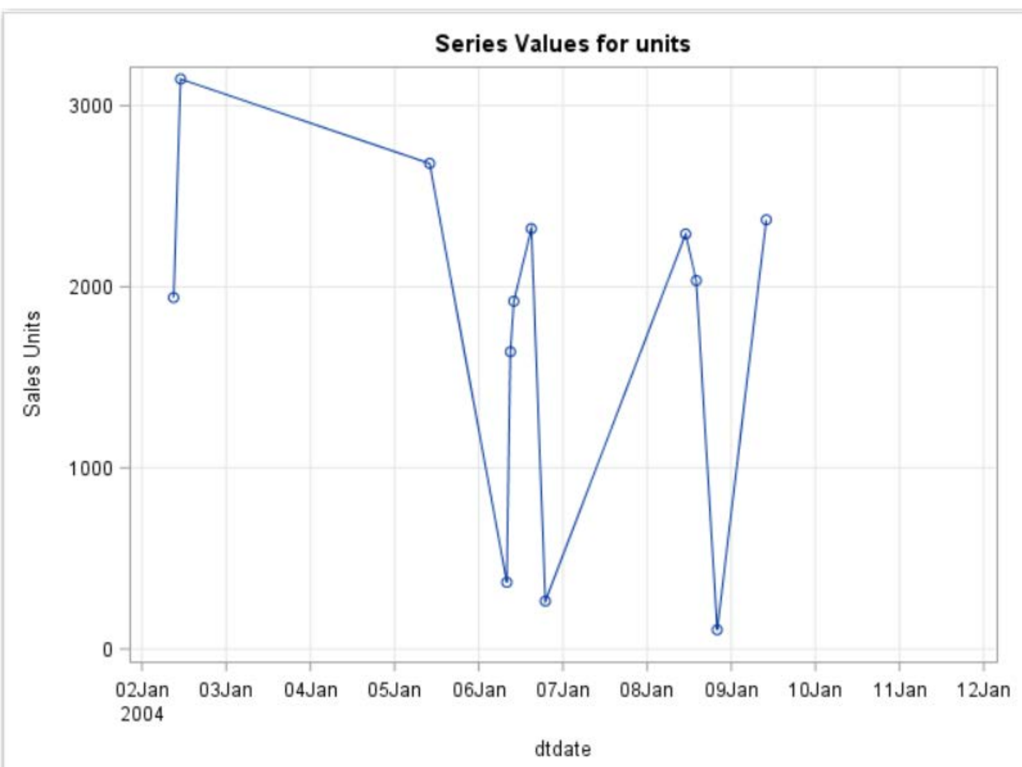
10. Add an END option to the ID statement in PROC TIMESERIES as shown below. The end value *12jan2004:00:00* truncates the data so that only the first 10 days of data are shown in the plots.

```
proc timeseries data=WORK.TempSorted seasonality=24
                plots=(series corr);
   id dtdate interval=hour end='12jan2004:00:00'dt;
   var units / accumulate=total transform=none dif=0 sdif=0;
run;
```

If you produce the code by entering it directly in the editor, the following code produces the necessary output:

```
/* subset the data using the END option */
proc timeseries data=out_totalhour plots=(series);
   id dtdate interval=dthour accumulate=total
             end='12jan2004:00:00'dt;
   var units;
run;
```

11. Click the **Run** button to submit the modified syntax.



One important point, illustrated in the plot above and the table below, is that issues with missing observations cannot be discerned until after the data is accumulated. If an hourly or daily interval is to be used for forecasting, a data imputation method might be necessary.

12. Click the **OUTPUT DATA** tab to view the hourly, intervaled, time series data.

	dtdate	units
1	02JAN04:08	.
2	02JAN04:09	1940
3	02JAN04:10	.
4	02JAN04:11	3147
5	02JAN04:12	.
6	02JAN04:13	.
7	02JAN04:14	.
8	02JAN04:15	.
9	02JAN04:16	.
10	02JAN04:17	.
11	02JAN04:18	.

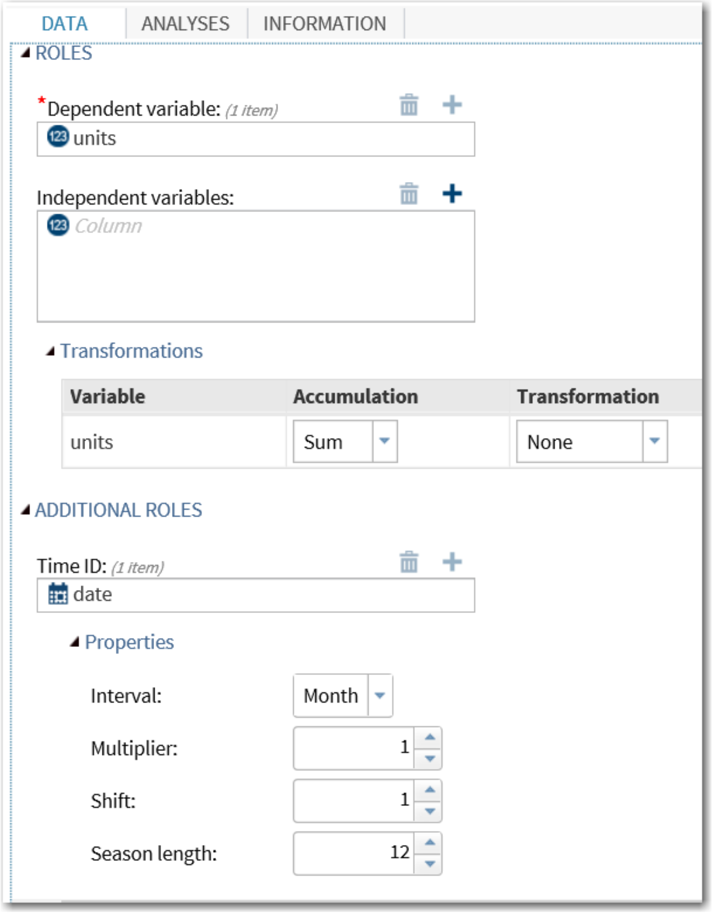
The SETMISSING option in the ID statement of the TIMESERIES procedure can be used to replace missing values in time series with imputed values.

A preferred method for missing value imputation in time series is replacing missing values with their one-step-ahead forecast. The ESM procedure in SAS/ETS can be used for this. See the REPLACEMISSING option in the FORECAST statement.

Creating the Time Series on a Monthly Interval

13. Click the **Time Series Exploration 1** tab to return to the original analysis.
14. Remove the assignment of the time ID variable. Highlight **dtdate** and click the **Trash Can** icon.
15. Assign **date** as the time ID variable, and change the interval property to **Month**.
16. Expand the **Transformations** properties, and set the **Accumulation** field to **Sum**.

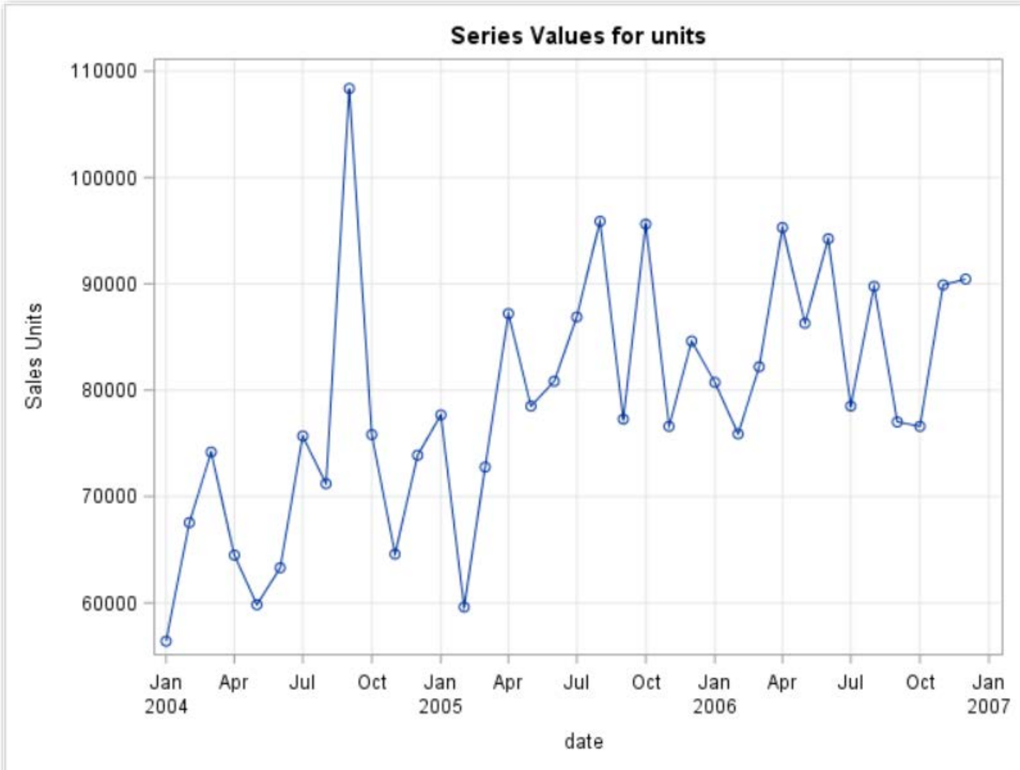
These steps are summarized in the display below.



If you produce the code by entering it directly in the editor, the following code produces the necessary output:

```
/* accumulate to a monthly interval using TOTAL */
proc timeseries data=stsm.ch1_demodat out=out_totalmonth
                plots=(series);
   id date interval=month accumulate=total;
   var units;
run;
```

17. Click the **Run** button to submit the modified Time Series Exploration task. A plot of the monthly intervaled **units** time series is shown below.



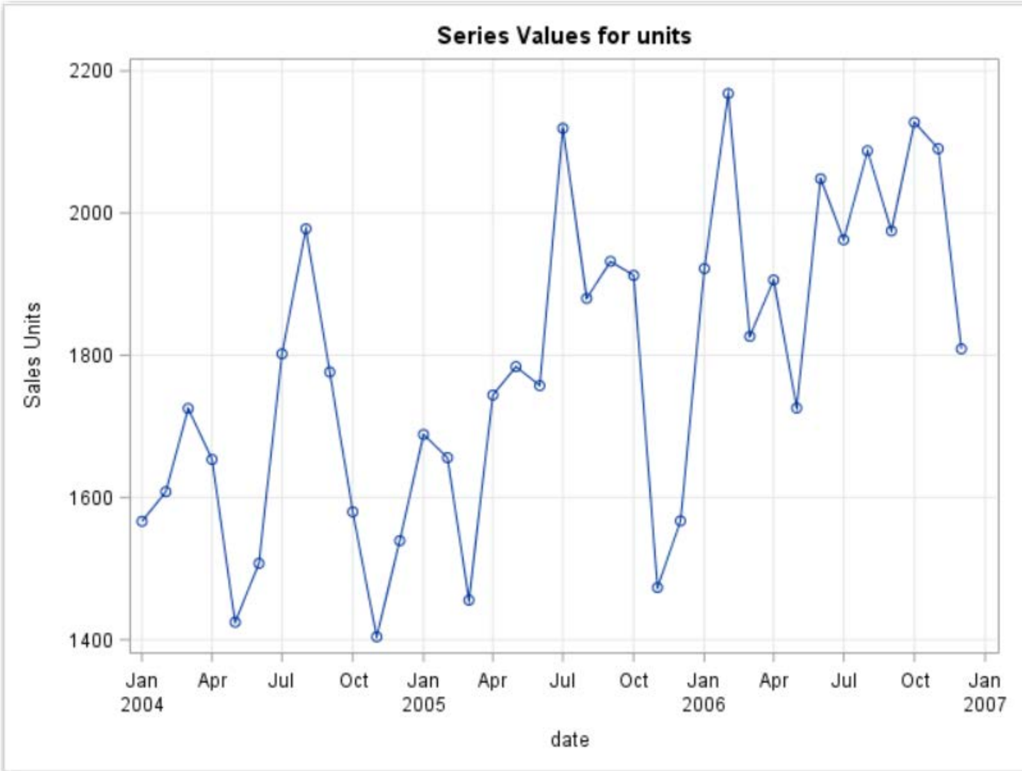
Using a Total or Sum accumulation method, the monthly intervaled data shows what might be a structural break near the middle of 2005. The spike in Sep2004 might be an anomaly that requires further investigation and consideration when you build a model to accommodate the variation in this series.

18. Change the **Accumulation** field value to **Average**.

19. If you produce the code by entering it directly in the editor, the following code produces the necessary output:

```
/* accumulate to a monthly interval using AVERAGE */
proc timeseries data=stsm.ch1_demodat out=out_totalmonth
                plots=(series);
   id date interval=month accumulate=average;
   var units;
run;
```

20. Click the **Run** button.



The plot of the time series that was created using an Average accumulation method reveals a trend and a cycle in the time series data.

End of Demonstration

Exercises

1. Creating a New Time Series Exploration Task

The **STSM.VISITS** table is used for this exercise. It contains approximately three years of transactional data on visits to a 24-hour, emergency clinic. The dependent variable is **visits**, and the time ID variable is **date**. Create a new Time Series Exploration task and choose appropriate interval and accumulation methods to answer the questions below.

Be sure to assign an output data set. Use an option in the procedure statement.

Hint: **out=xxx**

a. What is the average number of visits per year for each of the three years in the data?

b. What interval has the highest monthly total number of visits?

c. Are there any day intervals with zero visits?

End of Exercises

1.2 Time Series Components

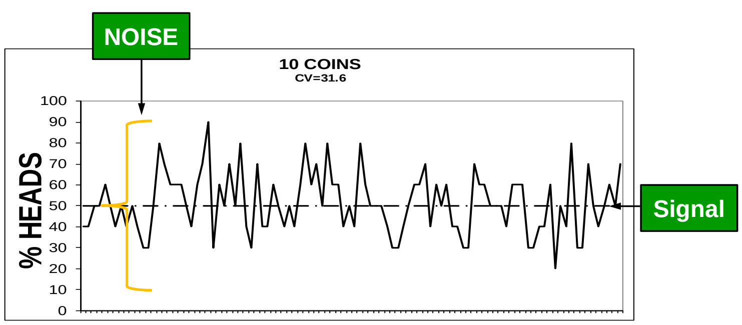
Objectives

- Describe the decomposition of time series variation.
- List the main components of systematic variation or signal that exist in time series.
- Perform a preliminary identification of a time series.

21

Variation in Time Series Data: Two Main Parts

- signal
- noise



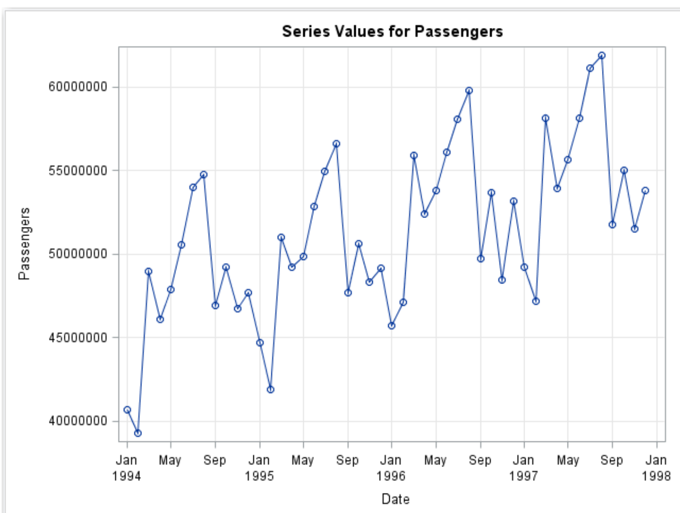
22

Signal Components

- level
- seasonality
- trend
- irregular
- exogenous
  (also known as *explanatory variable effects*)
- cycle

23

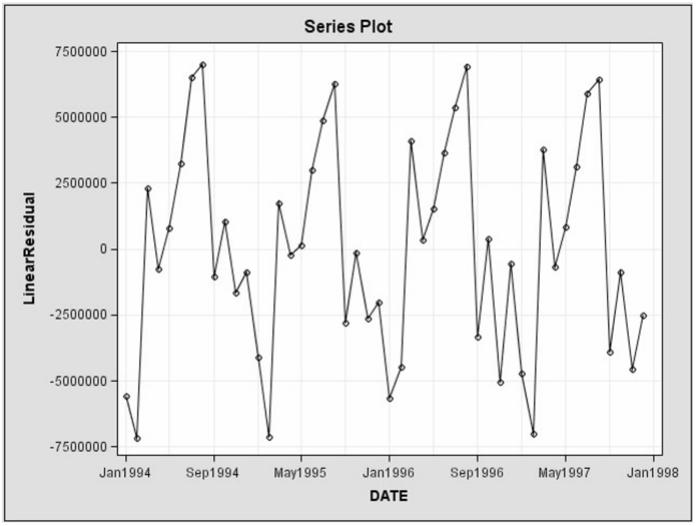
The Airline Data



24

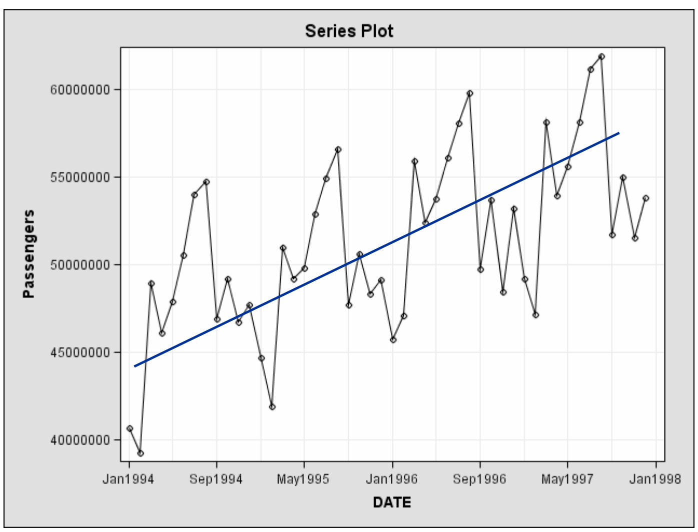
The airline data, **DOTAIR9497**, contain many of the signal components listed above. The data set contains a monthly intervaled time series of passengers who flew on commercial aircraft in the United States between January 1994 and December 1997. The slides below show a decomposition of the series into seasonal, trend, and irregular components.

Signal Components: Seasonality



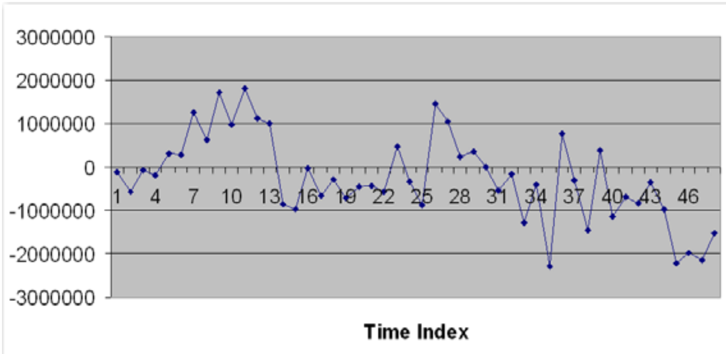
25

Signal Components: Trend



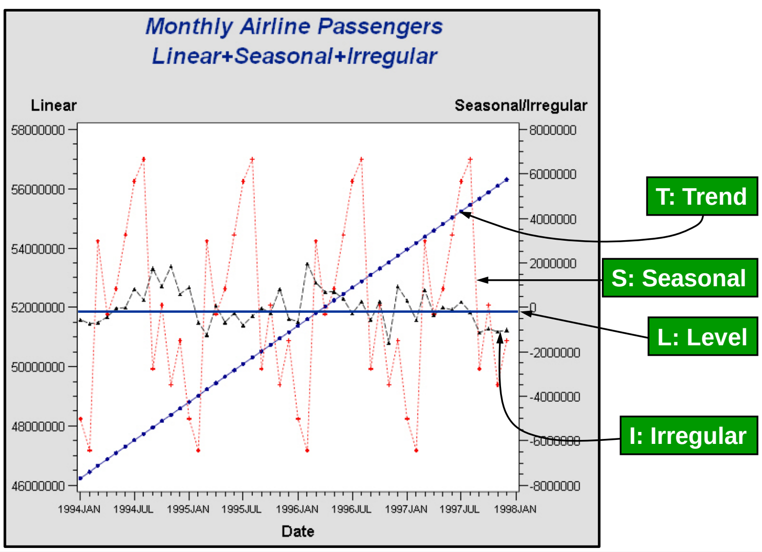
26

Signal Components: Irregular

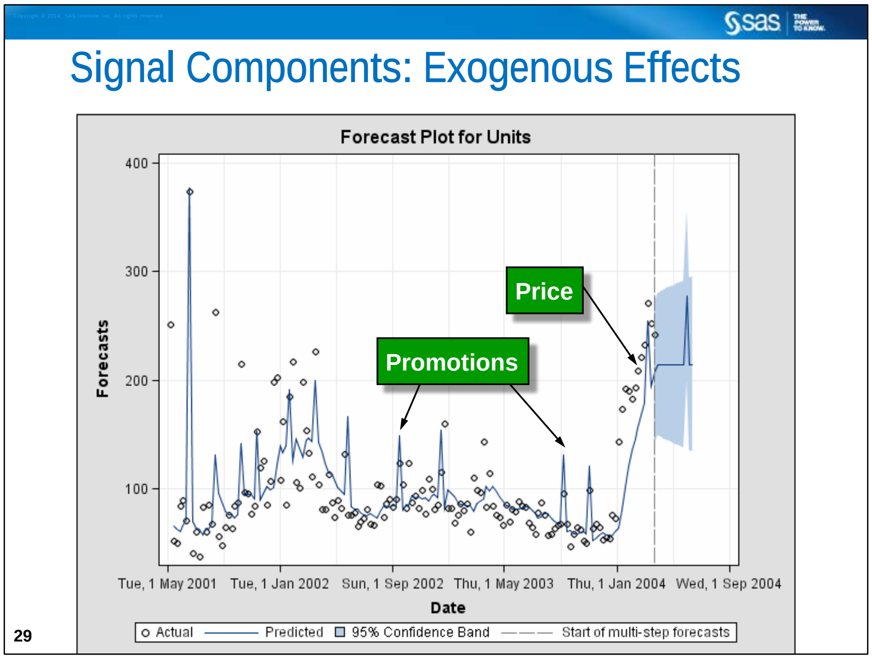


27

Component Decomposition



28



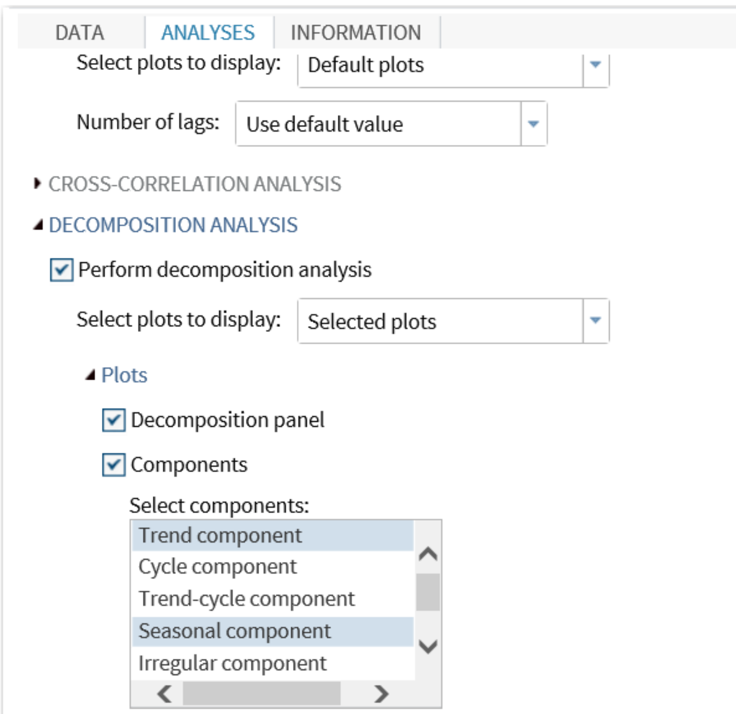
The Retail time series represents weekly unit sales for an item. The series shows evidence of promotion and price effects.

Time Series Identification

This demonstration begins where the previous demonstration ended. The monthly intervaled time series that was accumulated using the Average method is decomposed to assess the presence of trend, seasonal, and other components of systematic variation.

1. Click the **ANALYSES** tab in the Time Series Exploration task that you created in the previous demonstration.
2. Expand the **DECOMPOSITION ANALYSIS** properties.
3. Select the **Perform decomposition analysis** check box, and change the **Select plots to display** field to **Selected plots**.
4. Select the **Components** check box, and highlight the **Trend** and **Seasonal** component plots.

   These steps are summarized below.



5. Use SAS/ETS program code.

   Additional syntax includes the following decomposition plot options:

   - TC = Trend Component
   - SC = Seasonal Component
   - IC = Irregular Component
   - CORR plots correlation functions

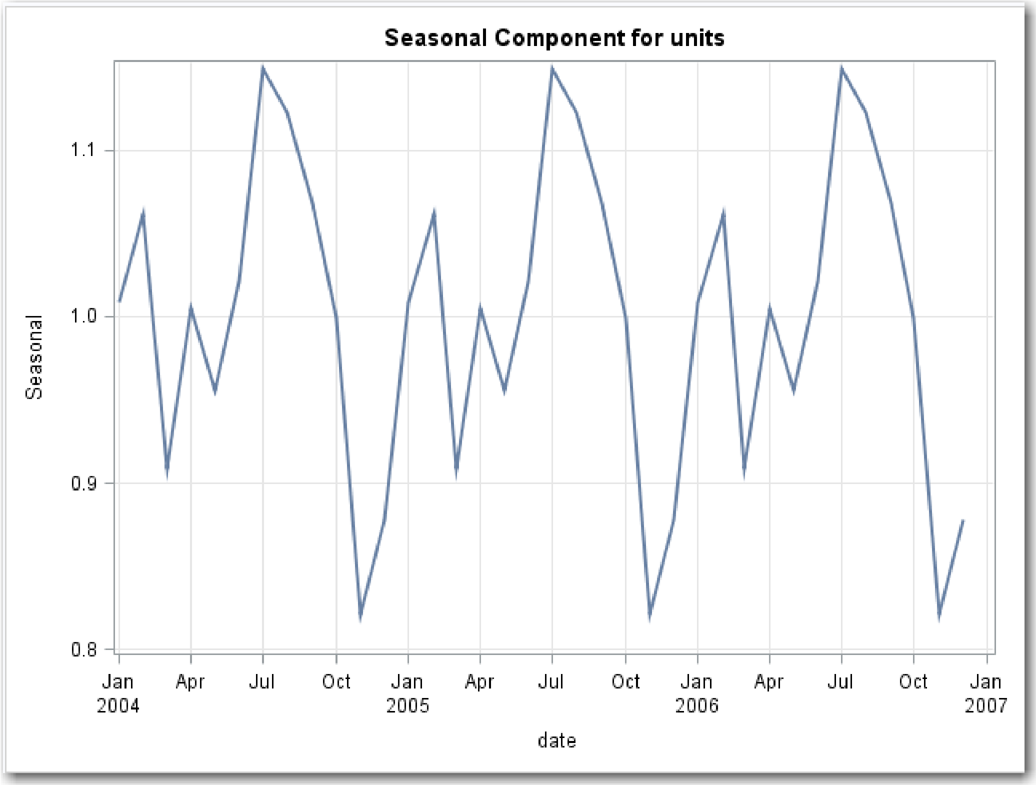
   The **decomp_total** data set contains the decomposition statistics and the *de-seasonalized series*. (The estimated seasonal component is subtracted from the original series to create the output series.)

```
proc timeseries data=stsm.ch1_demodat out=out_totalmonth
                outdecomp=decomp_total
                print=(descstats seasons decomp)
```

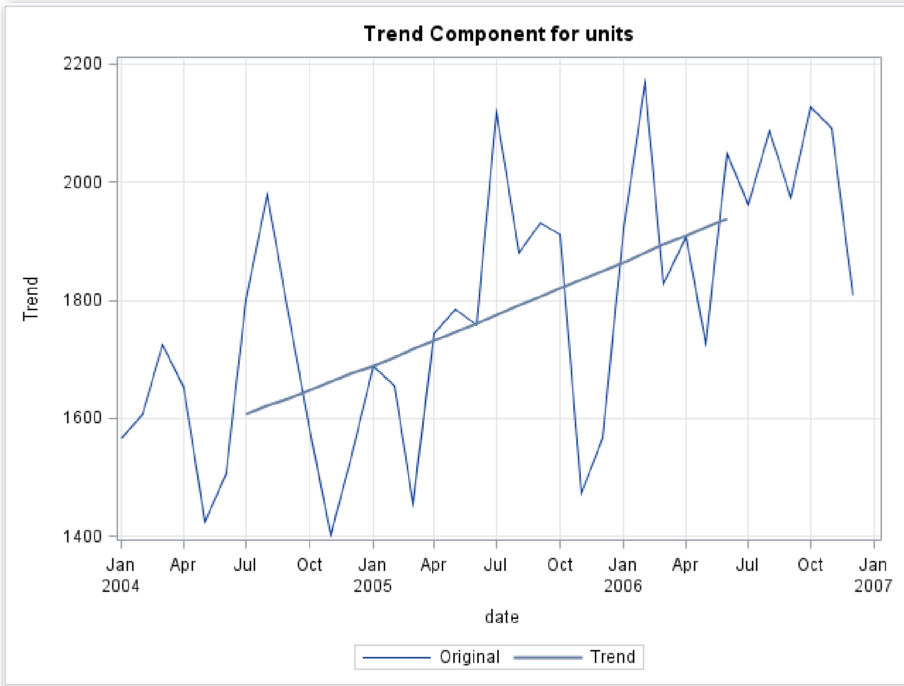
```
                 plot=(series corr tc sc ic);
   id date interval=month accumulate=average;
   var units;
run;
```

6. Click the **Run** button.

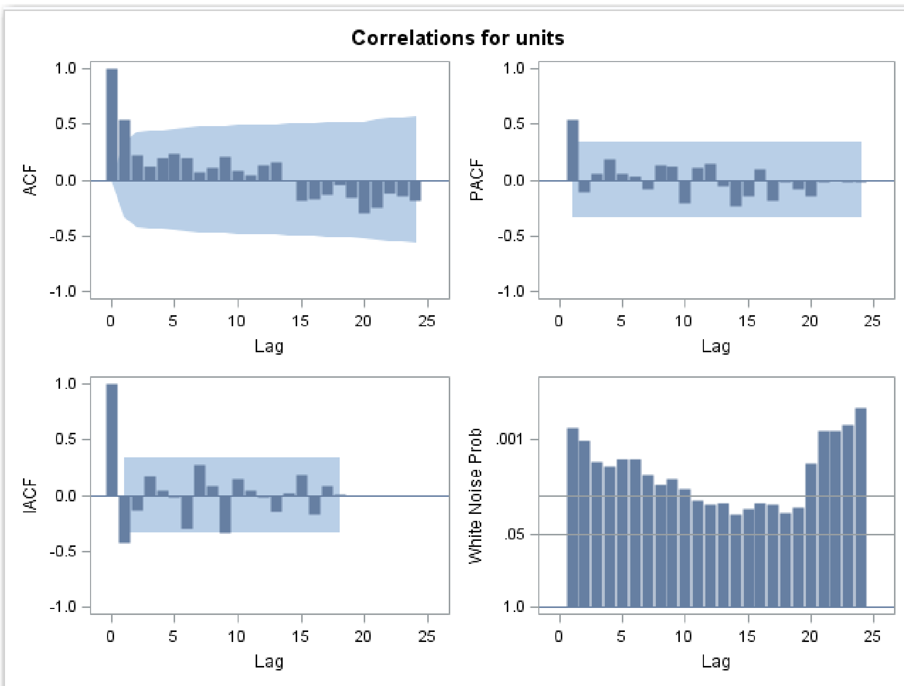
   The seasonality plot reveals a fairly strong seasonal cycle in the data that was accumulated to a monthly average. The seasonal peak month is July. Sales of **units** in July average more than 10% above average, annual sales. November is the seasonal low month. Sales in November average almost 20% below the annual average.



   The trend plot is based on a centered, moving-average representation of the series. A fairly strong and linear increase in average sales per month is depicted.



The correlation panel shows that the average **units** series is not white noise. The additional diagnostics provide information that can be used to select autoregressive and moving average orders for ARIMA models. (More information about these is provided in the "ARIMAX Models" chapter.)



A portion of the **decomp_total** data set is shown below. There are 12 unique values of the seasonal components. The peak month, July, is approximately 15% above the annual average. The Seasonally Adjusted series is derived by dividing the **units** (Original) series by the Seasonal Component series.

date	Seasonal Index	Original Series	Trend-Cycle Component	Seasonal Component	Irregular Component	Seasonally Adjusted Series
JAN2004	1	1566.44	.	1.00888	.	1552.65
FEB2004	2	1608.07	.	1.06163	.	1514.72
MAR2004	3	1725.09	.	0.90875	.	1898.30
APR2004	4	1653.51	.	1.00518	.	1644.99
MAY2004	5	1424.55	.	0.95608	.	1489.98
JUN2004	6	1507.31	.	1.02105	.	1476.23
JUL2004	7	1802.05	1635.47	1.14892	0.95903	1568.47
AUG2004	8	1978.14	1642.56	1.12292	1.07247	1761.60
SEP2004	9	1776.46	1633.32	1.06845	1.01796	1662.64
OCT2004	10	1579.73	1625.86	0.99908	0.97252	1581.19
NOV2004	11	1404.00	1644.62	0.82108	1.03972	1709.95
DEC2004	12	1539.23	1670.01	0.87796	1.04980	1753.18
JAN2005	1	1688.65	1693.65	1.00888	0.98827	1673.79
FEB2005	2	1655.89	1702.78	1.06163	0.91601	1559.76
MAR2005	3	1455.50	1705.19	0.90875	0.93928	1601.64

End of Demonstration

Exercises

2. Using the TIMESERIES Procedure and Creating Appropriate Decomposition Plots

This exercise uses the **STSM.VIOLENTCRIME** table. The dependent variable, **MurdersTX**, is the number of murders in Texas per month between JAN1989 and DEC1997. The time ID variable is **date**.

Create appropriate decomposition plots to answer the following questions:

a. Does the data have a seasonal cycle?

b. Is there a trend component in the data? If so, is it linear?

c. Assume that you are one of the Texas governors who were elected in the years 1991 or 1995. Is it reasonable for you to claim that your progressive, yet no-nonsense policies diminished the number of homicides in Texas during your term?

End of Exercises

1.3 Time Series Models

Objectives

- List the three families of time series models that are illustrated in this course.
- Describe the main features of each model family.
- Give examples of using an appropriate model from each family to fit the **DOTAIR9497** data.
- Discuss which model family is preferred for different types of time series analyses.

34

Necessary Conditions for Good Forecasts

- The identified signal continues into the future.
- Forecasting model complexity should be adequate to capture signal components.
- Forecasting models should not be overly complex.
- The best forecasting model is the one that captures and extrapolates the most signal, and that also ignores the noise.

35

Time Series Models in This Course

- exponential smoothing (ESM)
- autoregressive integrated moving average with exogenous variables (ARIMAX)
- unobserved components (UCM)

36

Exponential Smoothing Models

Exponential Smoothing Premise

- Weighted averages of past values can produce good forecasts of the future.
- The weights should emphasize the most recent data.
- Forecasting should require only a few parameters.
- Forecast equations should be simple and easy to implement.

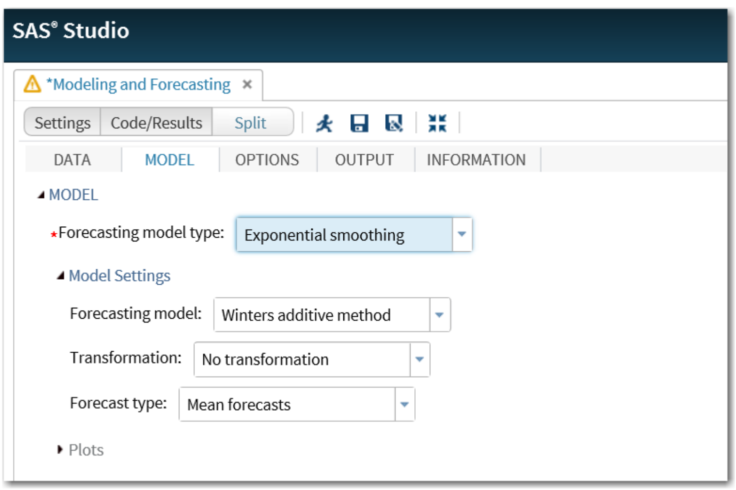
37

ESM Parameters and Keywords

ESM	Parameters	Option Name
Simple	ω	`simple`
Double	ω	`double`
Linear (Holt)	ω, γ	`linear`
Damped-Trend	ω, γ, ϕ	`damptrend`
Seasonal	ω, δ	`seasonal`
Additive Winters	ω, γ, δ	`addwinters`
Multiplicative Winters	ω, γ, δ	`winters`

38

Exponential Smoothing Model Implementation



39

Two of the seven exponential smoothing models accommodate trend, seasonal, and irregular variation (winters additive and winters multiplicative). The Modeling and Forecasting task in SAS Studio is used above to specify a winters additive method on the **DOTAIR9497** data.

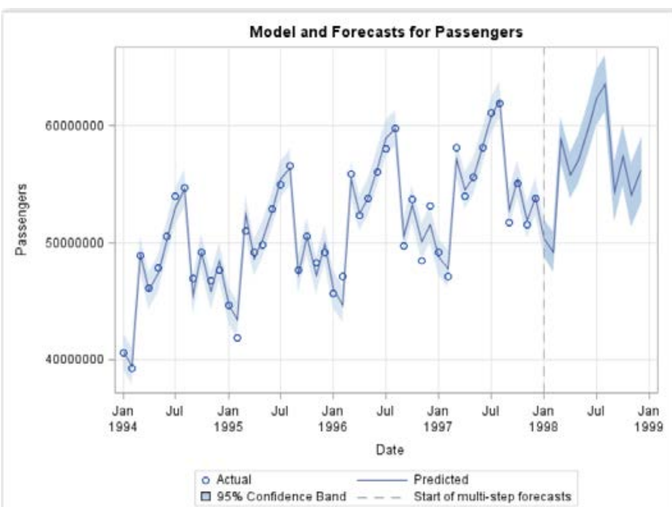
Exponential Smoothing Model Implementation

```
proc sort data=STSM.DOTAIR9497 out=WORK.TempSorted;
        by DATE;
run;

/*************************************************************************************/
/* Exponential Smoothing                                                            */
proc esm data=WORK.TempSorted back=0 lead=12 seasonality=12 plot=(corr errors
                modelforecasts);
        /* id statement */
        id DATE interval=month;
        forecast Passengers / alpha=0.05 model=addwinters transform=none;
run;
```

40

The ESM procedure (SAS/ETS) syntax, generated by SAS Studio, fits the winters additive (**addwinters**) specification and forecasts 12 months into the future.

Exponential Smoothing Model Forecasts



41

The selected ESM model seems to do a good job of capturing and extrapolating the trend and seasonal components of the data.

ARIMAX Models

Box-Jenkins ARIMAX Models

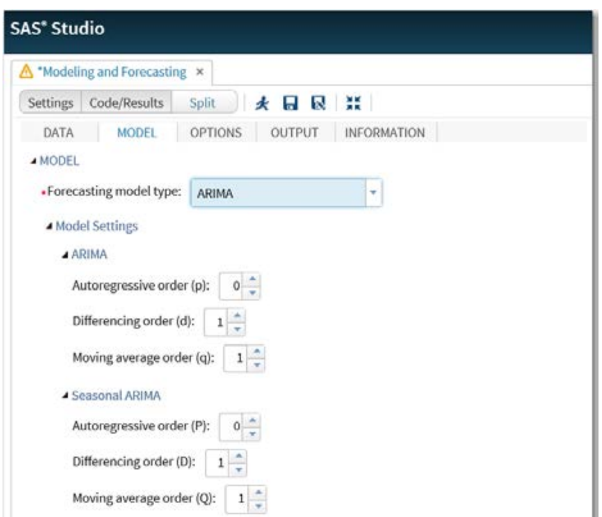
ARIMAX: AutoRegressive Integrated Moving Average with eXogenous variables

- AR: Autoregressive ⇒ Time series is a function of its own past.
- MA: Moving Average ⇒ Time series is a function of past shocks (deviations, innovations, errors, and so on).
- I: Integrated ⇒ Differencing provides stochastic trend and seasonal components, so forecasting requires *integration* (undifferencing).
- X: Exogenous ⇒ Time series is influenced by external factors.

42

Integration (the "I" in ARIMA) corresponds to the accommodation of nonstationary variation in an ARIMA model. Trend and seasonal components are examples of nonstationary variation. This is because, if a series has trend or seasonal components, its mean is a function of time. The mean of a stationary series is well defined, and is not a function of time. (A later chapter provides examples of stationary series.)

Box-Jenkins ARIMAX Implementation



43

An appropriate ARIMA model for the **DOTAIR9497** data set is the classic Box-Jenkins Airline model for series G, shown above. This model accommodates trend and seasonality with a first and seasonal (12) span differences. Differencing orders are offset by moving average orders at the same lags.

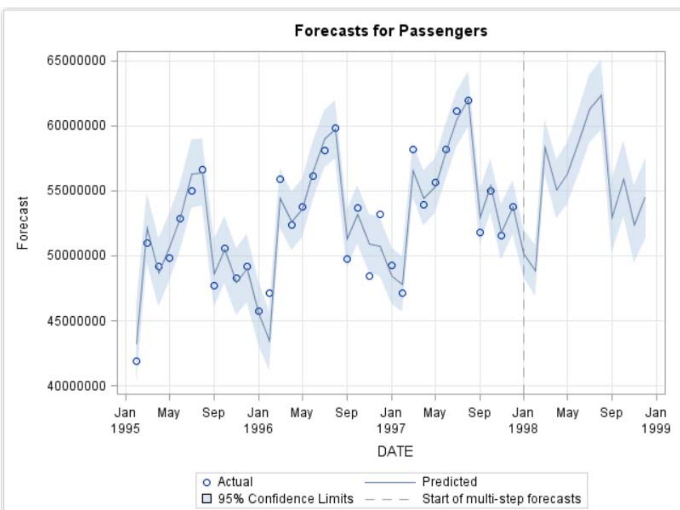
Box-Jenkins ARIMAX Implementation

```
proc sort data=STSM.DOTAIR9497 out=WORK.TempSorted;
        by DATE;
run;

/******************************************************************************/
/* ARIMA or ARIMAX                                                            */
proc arima data=WORK.TempSorted plots
        (only)=(series(corr crosscorr) residual(corr normal) forecast(forecastonly));
        identify var=Passengers(1 12);
        estimate q=(1) (12) method=ML;
        forecast lead=12 back=0 alpha=0.05 id=DATE interval=month;
        outlier;
quit;
```

44

In the generated, ARIMA procedure syntax, differencing orders are specified in the IDENTIFY statement. The Q option in the ESTIMATE statement specifies moving average orders. (More details about the ARIMA procedure syntax are provided in a later chapter.)

Box-Jenkins ARIMAX Forecasts



45

The ARIMA specification fitted above does a good job of capturing and extrapolating the trend, seasonal, and irregular components in the Passengers series.

Unobserved Components Models

Unobserved Components Models (UCMs)

- Also known as *structural time series models*
- Decompose time series into components:
  - trend
  - season
  - cycle
  - irregular
  - regressors
- General form:

$$Y_t = \text{Trend} + \text{Season} + \text{Cycle} + \text{Regressors}$$

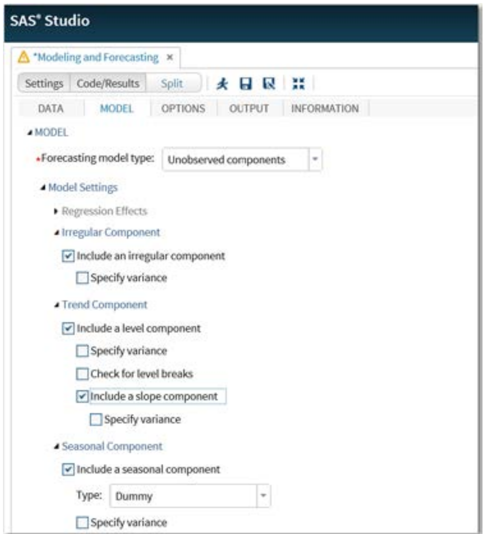
46

UCMs

- Each component captures some important feature of the series dynamics.
- Components in the model have their own models.
- Each component has its own source of error.
- The coefficients for trend, season, and cycle are dynamic.
- The coefficients are testable.
- Each component has its own forecasts.

47

UCM Implementation



48

A UCM model that accommodates the systematic components in the **DOTAIR9497** data is shown above. To create an appropriate UCM, a statement that corresponds to each component of hypothesized, systematic variation is specified.

The LEVEL and SLOPE components (statements) combine to fit the trend in the **DOTAIR9497** data.

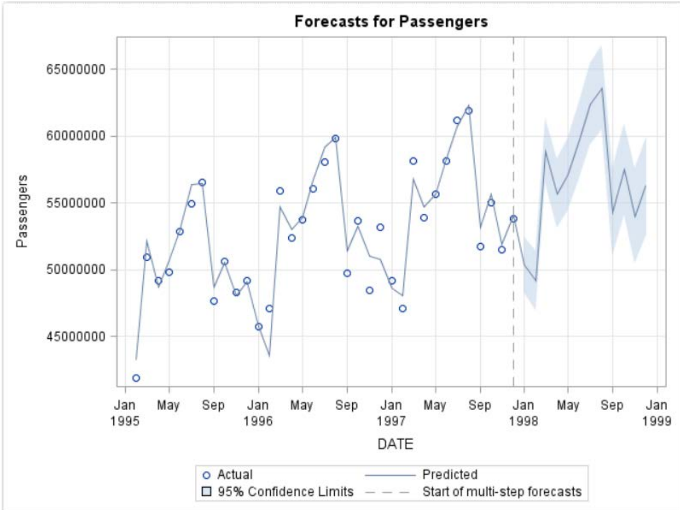
UCM Implementation

```
proc sort data=STSM.DOTAIR9497 out=WORK.TempSorted;
        by DATE;
run;

/*****************************************************************************************/
/* Unobserved Components                                                                */
proc ucm data=WORK.TempSorted;
        id DATE interval=month;
        model Passengers;
        irregular;
        level;
        slope;
        season length=12 type=dummy;
        forecast lead=12 back=0 alpha=0.05;
        outlier;
run;
```

49

A statement for each of the components of variation in the **DOTAIR9497** data is seen in the generated syntax above.

UCM Forecasts



50

The UCM specification does a good job of capturing and extrapolating the systematic variation in the **DOTAIR9497** data.

Choosing the Right Model for the Job

Usability

Best to worst:

1. ESM
2. UCM
3. ARIMAX

51

Complexity

Least to Most:

1. ESM
2. ARIMAX
3. UCM

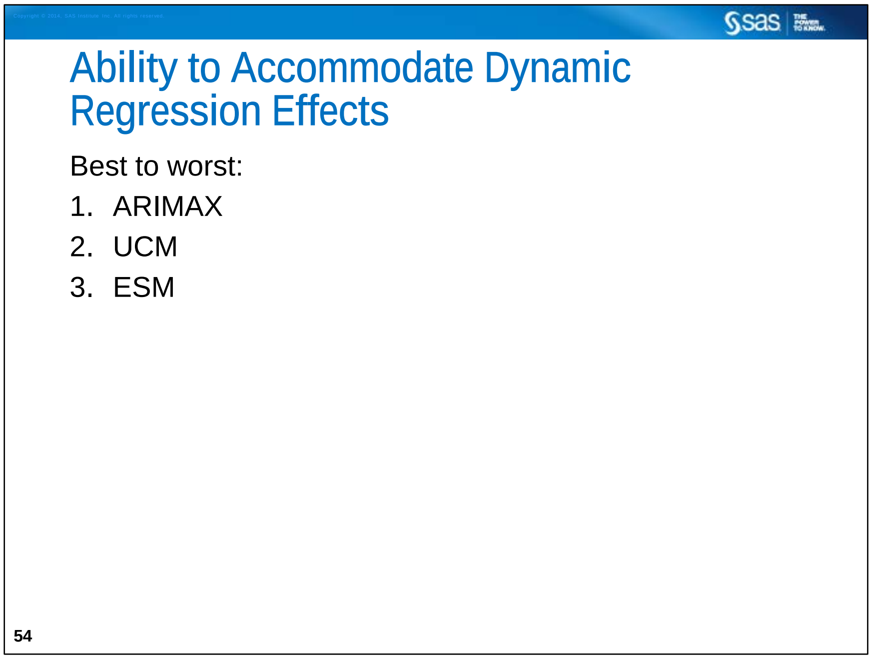
52

Robustness

Best to worst:

1. ESM
2. ARIMAX
3. UCM

53



Idea Exchange

List the types of forecasting models that you used in your analyses.

Give an example of an analysis that would be well suited to each of the following families of models:

- ESM
- ARIMAX
- UCM

1.4 SAS Studio Introduction

SAS Studio

SAS Studio is the new browser-based SAS programming environment.

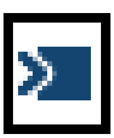
http://swww.sas.com/gobot/SASStudioTutorial

57

Interactive Mode

Some SAS procedures, such as PROC ARIMA, are interactive. That means that they remain active until you submit a QUIT statement, or until you submit a new PROC or DATA step.

In SAS Studio, you can use the Code Editor to run these procedures, as well as other SAS procedures, in interactive mode.



By default, SAS Studio does not run in interactive mode.
This icon in SAS Studio toggles interactive mode on and off.

58

Considerations for Running in Interactive Mode

- Interactive mode starts a new SAS session.
- Librefs and macro variables must be defined for each new SAS session.

SAS Studio Documentation:
http://sww.sas.com/gobot/SASStudioDoc

59