

## 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 *seasonality*. 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  $t = 1, \dots, T$ .

## 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:

$$f(y_t) = TC_t + S_t + I_t \quad \text{for additive decomposition}$$

$$f(y_t) = TC_t S_t I_t \quad \text{for multiplicative decomposition}$$

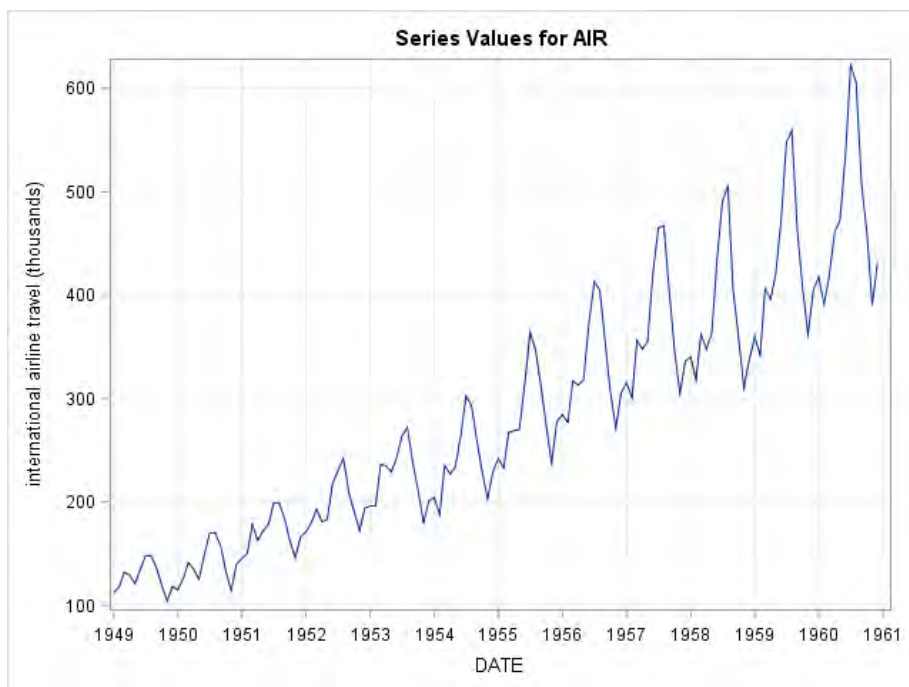
where  $f(y_t)$  represents a possible functional transformation for the dependent series, such as log, square-root, logistic, or Box-Cox transformation.

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.

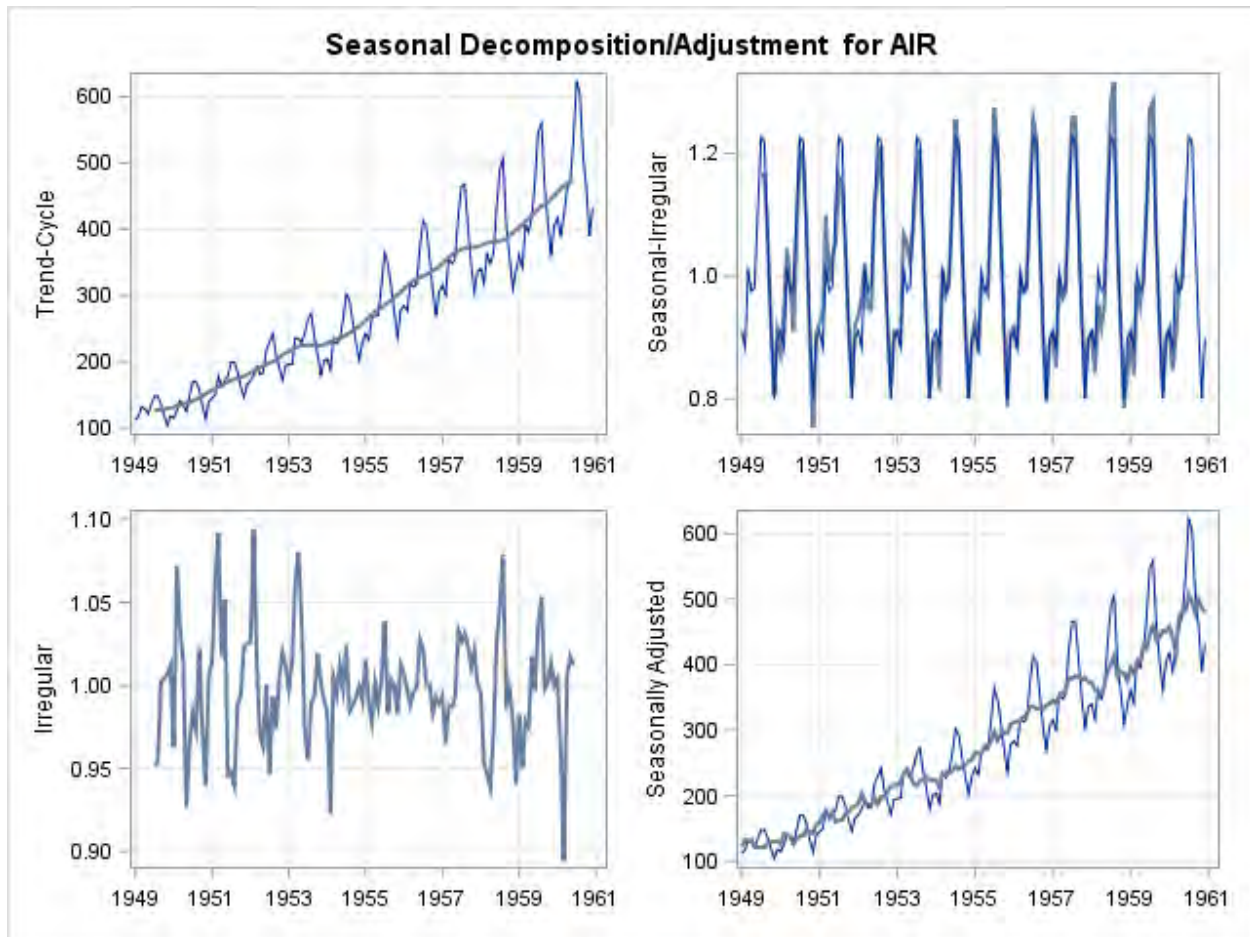


**Figure 1. Air Series**

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*.

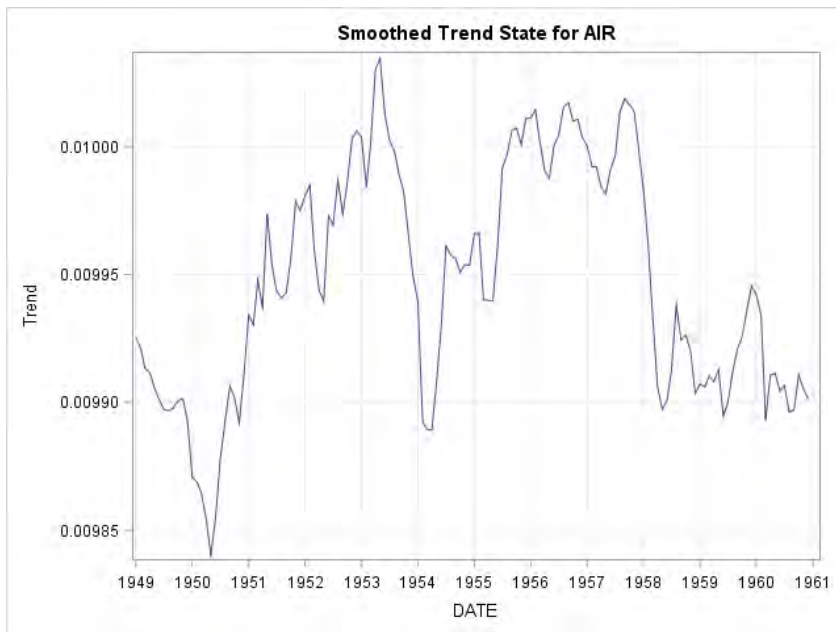
## Exponential Smoothing Decomposition

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:

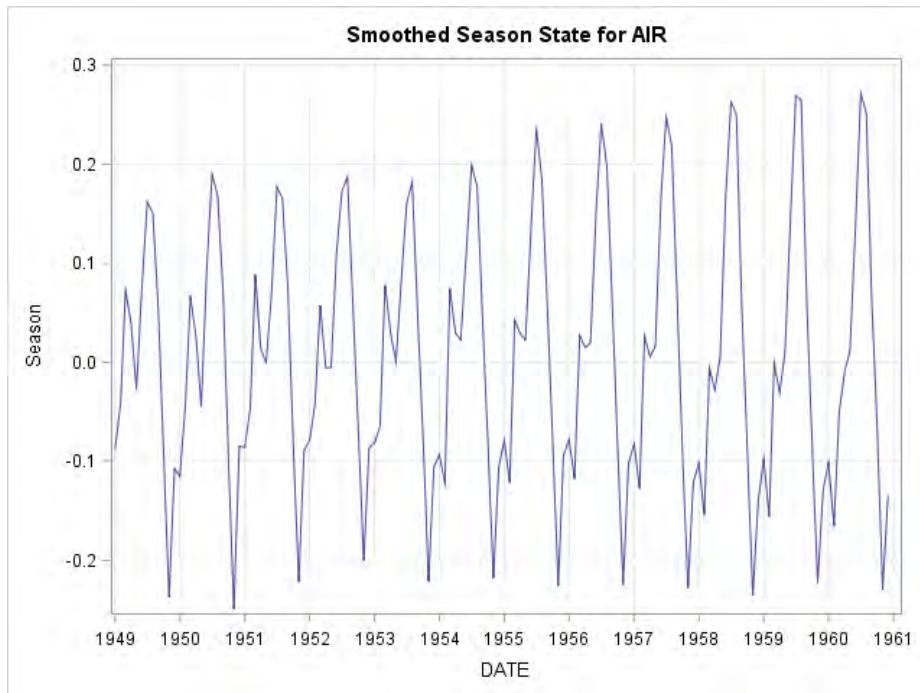
```
proc esm data=sashelp.air out=_null_
    lead=0
    back=0
    plot=(trend season);
    id date interval=month;
    forecast air / model=addwinters transform=log;
run;
```

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_	_PARAM_	_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 ESMspec 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  $\mathbf{X} = \{x_{k,l}\}_{k=1,l=1}^{K,L}$  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  $\mathbf{X} = \mathbf{U}\mathbf{Q}\mathbf{V}$ , where  $\mathbf{U}$  represents the  $K \times L$  matrix that contains the left-hand-side (LHS) eigenvectors,  $\mathbf{Q}$  represents the diagonal  $L \times L$  matrix that contains the singular values, and  $\mathbf{V}$  represents the  $L \times L$  matrix that contains the right-hand-side (RHS) eigenvectors.

Therefore,  $\mathbf{X} = \sum_{l=1}^L \mathbf{X}^{(l)} = \sum_{l=1}^L u_l q_l v_l'$ , where  $\mathbf{X}^{(l)}$  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,  $m = 1, \dots, M$ , define a group of window indices  $I_m \subset \{1, \dots, L\}$ . Let  $\mathbf{X}_{I_m} = \sum_{l \in I_m} \mathbf{X}^{(l)} = \sum_{l \in I_m} u_l q_l v_l'$  represent the grouped trajectory matrix for group  $I_m$ .

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

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

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

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,  $\cup_{m=1}^M I_m = \{1, \dots, L\}$ , 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  $m = 1, \dots, M$ .

5. **Forecasting (optional):** If the groupings represent a spectral partition, then each component series,  $\tilde{x}_t^{(m)}$  for  $m = 1, \dots, M$ , 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  $m = 1, \dots, M$  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.

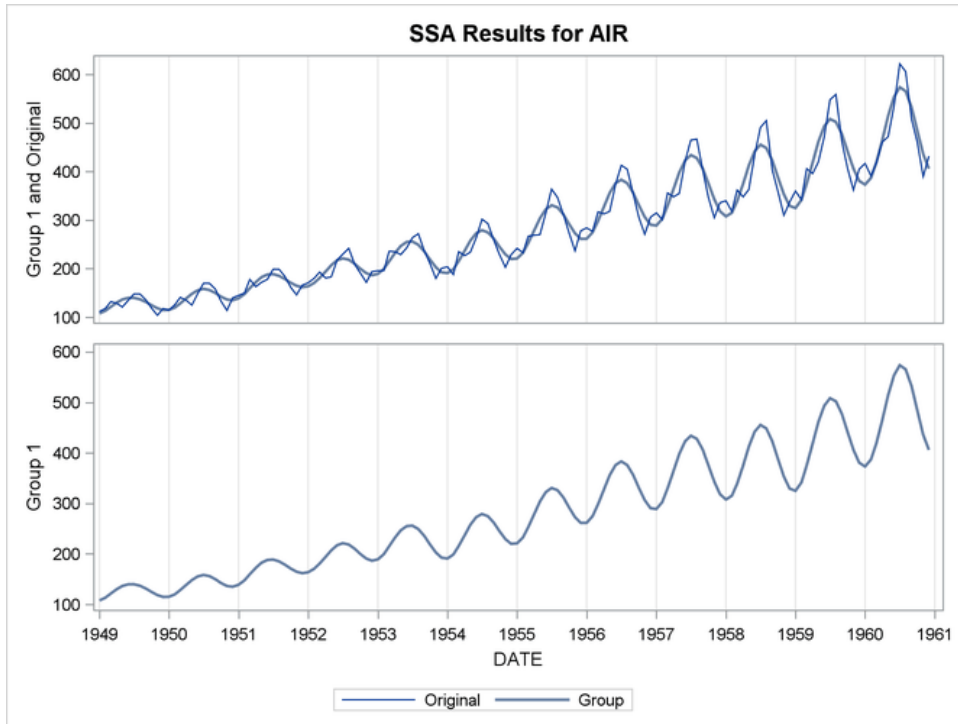


Figure 5. Plot for Singular Value Grouping 1

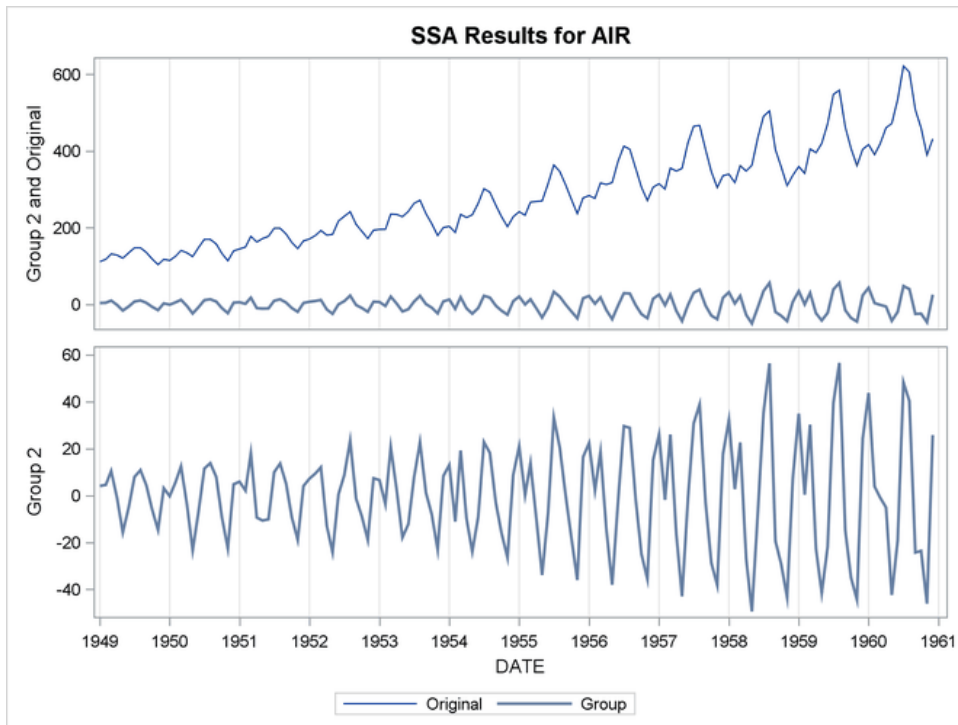
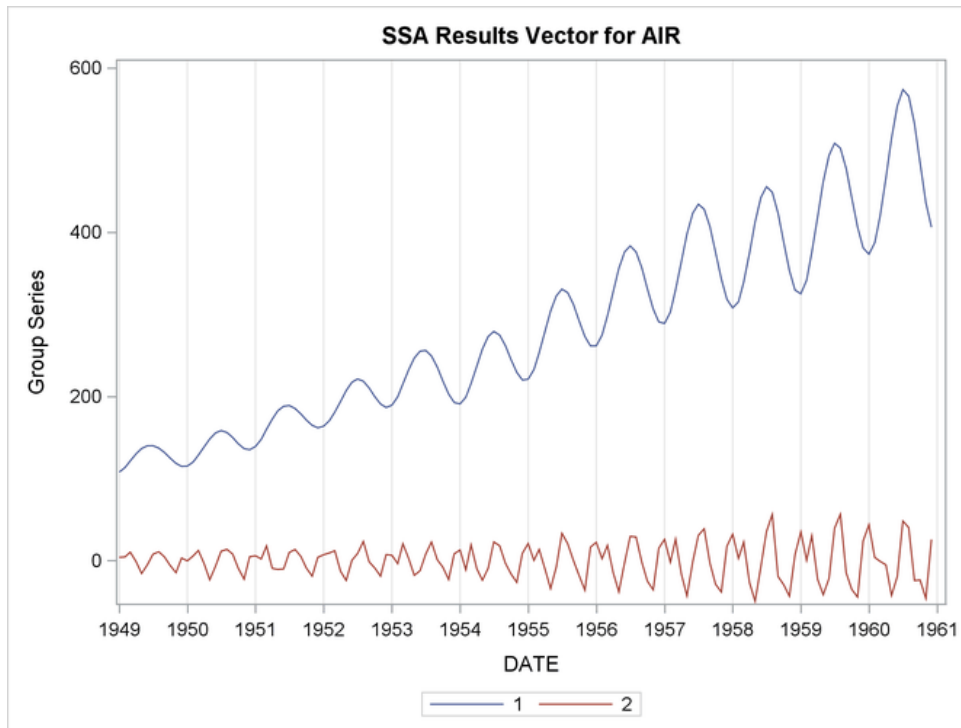


Figure 6. Plot for Singular Value Grouping 2



**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

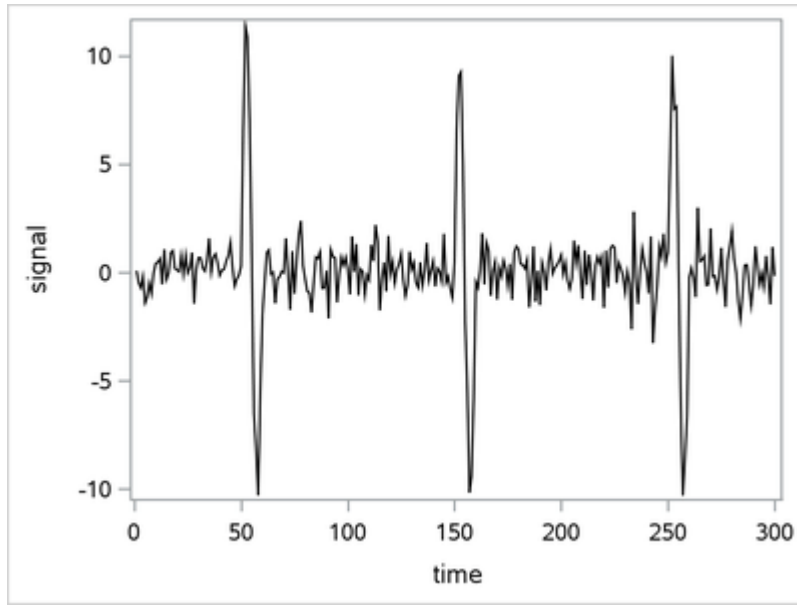
This paper demonstrates only the brute-force method of motif discovery. For more information about the other methods, see the *SAS Visual Forecasting: Time Series Packages*.

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:

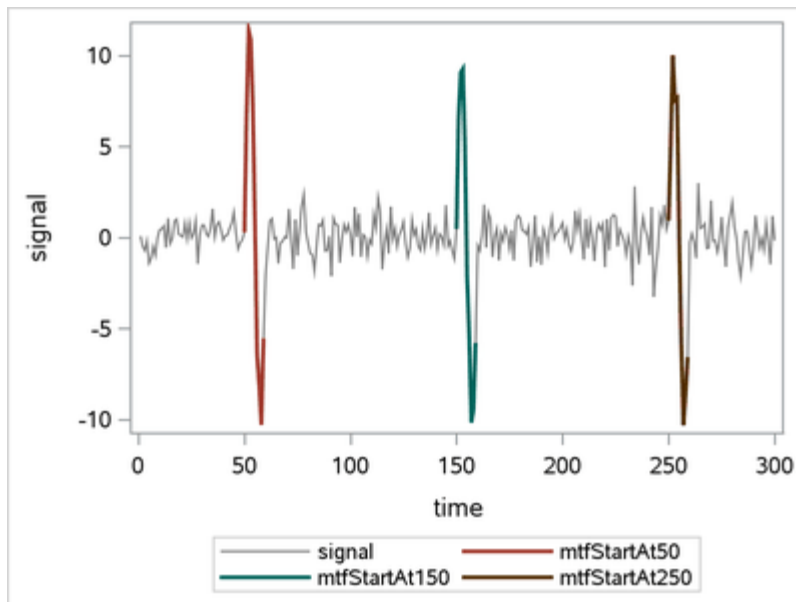
```
proc tsmodel data=mycas.SimuData
            outobj=(of=mycas.outmotif
                   ofms = mycas.outmotifseries);
var signal;
id time interval=day;
require mtf;
submit;
    declare object f(MTFBF);
    declare object of(OUTMTF);
    declare object ofms(OUTMTFSERIES);
    rc = f.Initialize();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

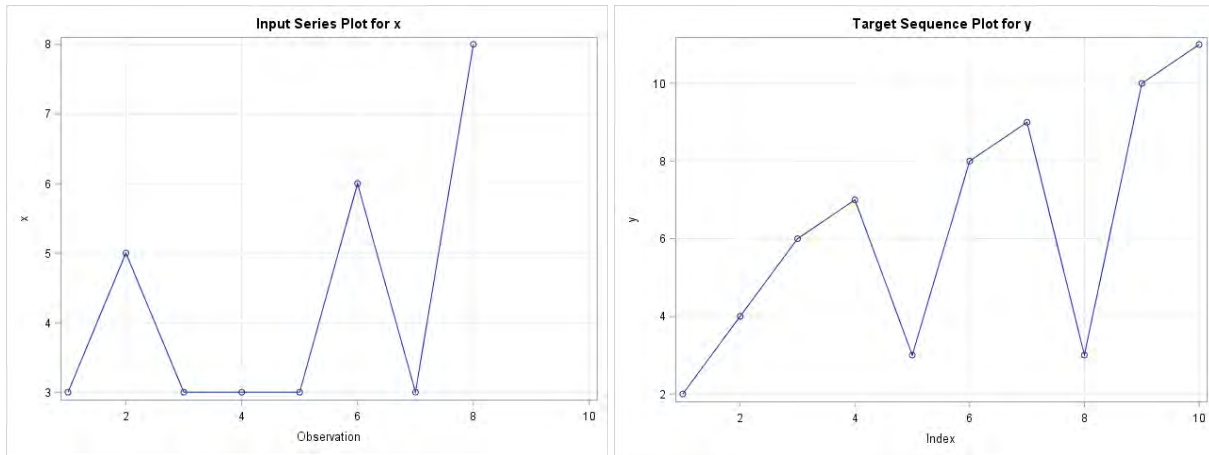
```
data test;
  input i y x;
  datalines;
  1 2 3
  2 4 5
  3 6 3
  4 7 3
  5 3 3
  6 8 6
  7 9 3
  8 3 8
  9 10 .
  10 11 .
  ;
run;
```

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.



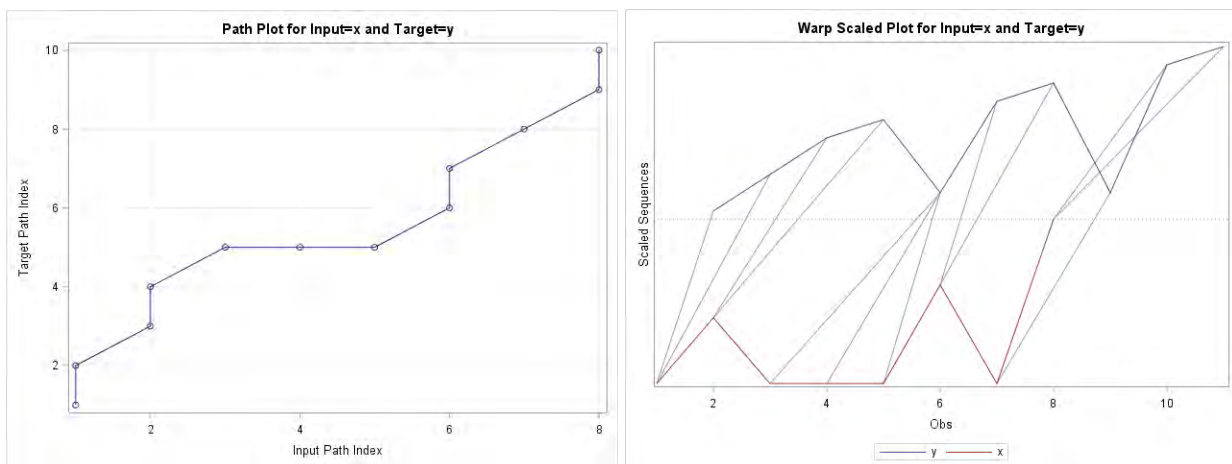
**Figure 10. Input and Target Series**

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](mailto:Michele.Trovero@sas.com)

Michael Leonard  
SAS Institute Inc.  
[Michael.Leonard@sas.com](mailto:Michael.Leonard@sas.com)

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.