SAS/ETS® 14.3 User's Guide

The SIMILARITY Procedure

This document is an individual chapter from *SAS/ETS® 14.3 User's Guide*.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2017. *SAS/ETS® 14.3 User's Guide*. Cary, NC: SAS Institute Inc.

SAS/ETS® 14.3 User's Guide

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

September 2017

Chapter 30
The SIMILARITY Procedure

Contents

Overview: SIMILARITY Procedure

The SIMILARITY procedure computes similarity measures associated with time-stamped data, time series, and other sequentially ordered numeric data. PROC SIMILARITY computes similarity measures for time-stamped transactional data (transactions) with respect to time by accumulating the data into a time series format, and it computes similarity measures for sequentially ordered numeric data (sequences) by respecting the ordering of the data.

Given two ordered numeric sequences (input and target), a similarity measure is a metric that measures the distance between the input and target sequences while taking into account the ordering of the data. The SIMILARITY procedure computes similarity measures between an input sequence and a target sequence, in addition to similarity measures that "slide" the target sequence with respect to the input sequence. The "slides" can be by observation index (sliding-sequence similarity measures) or by seasonal index (seasonal-sliding-sequence similarity measures).

In order to compare the raw input and the raw target time-stamped data, the raw data must be accumulated to a time series format. After the input and target time series are formed, the two accumulated time series can be compared as two ordered numeric sequences.

For raw time-stamped data, after the transactional data are accumulated to form time series and any missing values are interpreted, each accumulated time series can be functionally transformed, if desired. Transformations are useful when you want to stabilize the time series before computing the similarity measures. Transformations performed by the SIMILARITY procedure include the following:

- log (LOG)
- square-root (SQRT)
- logistic (LOGISTIC)
- Box-Cox (BOXCOX)
- user-defined transformations

Each time series can be transformed further by using simple differencing or seasonal differencing or both. Additional time series transformations can be performed by using various time series transformation and analysis techniques provided by this procedure or other SAS/ETS procedures.

After optionally transforming each time series, the accumulated and transformed time series can be stored in an output data set (OUT= data set).

After optional accumulation and transformation, each of these time series are the "working series," which can now be analyzed as sequences of numeric data. Each of these sequences can be a target sequence, an input sequence, or both a target and an input sequence. Throughout the remainder of this chapter, the term "original sequence" applies to both the original input and target sequence. The term "working sequence" applies to a version of both the original input and target sequence under investigation.

Each original sequence can be normalized prior to similarity analysis. Normalizations are useful when you want to compare the "shape" or "profile" of the time series. Normalizations performed by the SIMILARITY procedure include the following:

- standard (STANDARD)
- absolute (ABSOLUTE)
- user-defined normalizations

After each original sequence is optionally normalized, each working input sequence can be scaled to the target sequence prior to similarity analysis. Scaling is useful when you want to compare the input sequence to the target sequence while discounting the variation of the target sequence. Input sequence scaling performed by the SIMILARITY procedure include the following:

- standard (STANDARD)
- absolute (ABSOLUTE)
- user-defined scaling

After the working input sequence is optionally scaled to the target sequence, similarity measures can be computed. Similarity measures computed by the SIMILARITY procedure include the following:

- squared deviation (SQRDEV)
- absolute deviation (ABSDEV)
- mean square deviation (MSQRDEV)
- mean absolute deviation (MABSDEV)
- user-defined similarity measures

In computing the similarity measure between two time series, tasks are needed for transforming time series, normalizing sequences, scaling sequences, and computing metrics or measures. The SIMILARITY procedure provides built-in routines to perform these tasks. The SIMILARITY procedure also enables you to extend the procedure with user-defined routines.

All results of the similarity analysis can be stored in output data sets, printed, or graphed using the Output Delivery System (ODS).

The SIMILARITY procedure can process large amounts of time-stamped transactional data, time series, or sequential data. Therefore, the analysis results are useful for large-scale time series analysis, analogous time series forecasting, new product forecasting, or time series (temporal) data mining.

The SAS/ETS EXPAND procedure can be used for frequency conversion and transformations of time series. The TIMESERIES procedure can be used for large-scale time series analysis. The SAS/STAT DISTANCE procedure can be used to compute various measures of distance, dissimilarity, or similarity between observations (rows) of a SAS data set.

Getting Started: SIMILARITY Procedure

This section outlines the use of the SIMILARITY procedure and gives a cursory description of some of the analysis techniques that can be performed on time-stamped transactional data, time series, or sequentially ordered numeric data.

Given an input data set that contains numerous transaction variables recorded over time at no specific frequency, the SIMILARITY procedure can form equally spaced input and target time series as follows:

```
PROC SIMILARITY DATA=<input-data-set>
                OUT=<output-data-set>
                OUTSUM=<summary-data-set>;
   ID <time-ID-variable> INTERVAL=<frequency>
                         ACCUMULATE=<statistic>;
   INPUT <input-time-stamp-variables>;
   TARGET <target-time-stamp-variables>;
RUN;
```

The SIMILARITY procedure forms time series from the input time-stamped transactional data. It can provide results in output data sets or in other output formats using the Output Delivery System (ODS). The examples in this section are more fully illustrated in the section “Examples: SIMILARITY Procedure” on page 2294.

Time-stamped transactional data are often recorded at no fixed interval. Analysts often want to use time series analysis techniques that require fixed-time intervals. Therefore, the transactional data must be accumulated to form a fixed-interval time series.

Suppose that a bank wants to analyze the transactions that are associated with each of its customers over time. Further, suppose that the data set WORK.TRANSACTIONS contains three variables that are related to the customer transactions (CUSTOMER, DATE, and WITHDRAWAL) and one variable that contains an example fraudulent behavior (FRAUD).

The following statements illustrate how to use the SIMILARITY procedure to accumulate time-stamped transactional data to form a daily time series based on the accumulated daily totals of each type of transaction (WITHDRAWALS and FRAUD):

```
proc similarity data=transactions out=timedata;
   by customer;
   id date interval=day accumulate=total;
   input withdrawals;
   target fraud;
run;
```

The OUT=TIMEDATA option specifies that the resulting time series data for each customer are to be stored in the data set WORK.TIMEDATA. The INTERVAL=DAY option specifies that the transactions are to be accumulated on a daily basis. The ACCUMULATE=TOTAL option specifies that the sum of the transactions are to be accumulated. After the transactional data are accumulated into a time series format, the time series data can be normalized so that the “shape” or “profile” is analyzed.

For example, the following statements build on the previous statements and demonstrate normalization of the accumulated time series:

```
proc similarity data=transactions out=timedata;
   by customer;
   id date interval=day accumulate=total;
   input withdrawals / NORMALIZE=STANDARD;
   target fraud      / NORMALIZE=STANDARD;
run;
```

The NORMALIZE=STANDARD option specifies that each accumulated time series observation is normalized by subtracting the mean and then dividing by the standard deviation of the accumulated time series. The WORK.TIMEDATA data set now contains the accumulated and normalized time series data for each customer.

After the transactional data are accumulated into a time series format and normalized to a mean of zero and standard deviation of one, similarity analysis can be performed on the accumulated and normalized time series.

For example, the following statements build on the previous statements and demonstrate similarity analysis of the accumulated and normalized time series:

```
proc similarity data=transactions
                out=timedata OUTSUM=SUMMARY;
   by customer;
   id date interval=day accumulate=total;
   input withdrawals / normalize=standard;
   target fraud      / normalize=standard MEASURE=MABSDEV;
run;
```

The MEASURE=MABSDEV option specifies the accumulated and normalized time series data that are associated with the variables WITHDRAWALS and FRAUD are to be compared by using mean absolute deviation. The OUTSUM=SUMMARY option specifies that the similarity analysis summary for each customer is to be stored in the data set WORK.SUMMARY.

Syntax: SIMILARITY Procedure

The following statements are used with the SIMILARITY procedure:

PROC SIMILARITY *options* **;**
  BY *variables* **;**
  ID *variable* **INTERVAL=** *interval options* **;**
  FCMPOPT *options* **;**
  INPUT *variable-list / options* **;**
  TARGET *variable-list / options* **;**

Functional Summary

The statements and options that control the SIMILARITY procedure are summarized in Table 30.1.

Table 30.1 Functional Summary

Description	Statement	Option
Statements		
Specifies BY-group processing	BY	
Specifies the time ID variable	ID	
Specifies the FCMP options	FCMPOPT	
Specifies input variables to analyze	INPUT	
Specifies target variables to analyze	TARGET	
Data Set Options		
Specifies the input data set	PROC SIMILARITY	DATA=
Specifies the time series output data set	PROC SIMILARITY	OUT=
Specifies the measure summary output data set	PROC SIMILARITY	OUTMEASURE=
Specifies the path output data set	PROC SIMILARITY	OUTPATH=
Specifies the sequence output data set	PROC SIMILARITY	OUTSEQUENCE=
Specifies the summary output data set	PROC SIMILARITY	OUTSUM=
User-Defined Functions and Subroutine Options		
Specifies FCMP quiet mode	FCMPOPT	QUIET=
Specifies FCMP trace mode	FCMPOPT	TRACE=
Accumulation and Seasonality Options		
Specifies the accumulation frequency	ID	INTERVAL=
Specifies the length of seasonal cycle	PROC SIMILARITY	SEASONALITY=
Specifies the interval alignment	ID	ALIGN=
Specifies that the time ID variable values are not sorted	ID	NOTSORTED
Specifies the starting time ID value	ID	START=
Specifies the ending time ID value	ID	END=

Description	Statement	Option
Specifies the accumulation statistic	ID, INPUT, TARGET	ACCUMULATE=
Specifies the missing value interpretation	ID, INPUT, TARGET	SETMISS=
Specifies the zero value interpretation	ID, INPUT, TARGET	ZEROMISS=
Specifies the type of missing value trimming	INPUT, TARGET	TRIMMISS=
Time Series Transformation Options		
Specifies simple differencing	INPUT, TARGET	DIF=
Specifies seasonal differencing	INPUT, TARGET	SDIF=
Specifies the transformation	INPUT, TARGET	TRANSFORM=
Input Sequence Options		
Specifies normalization	INPUT	NORMALIZE=
Specifies scaling	INPUT	SCALE=
Target Sequence Options		
Specifies normalization	TARGET	NORMALIZE=
Similarity Measure Options		
Specifies the compression limits	TARGET	COMPRESS=
Specifies the expansion limits	TARGET	EXPAND=
Specifies the similarity measure	TARGET	MEASURE=
Specifies the similarity measure and path	TARGET	PATH=
Specifies the sequence slide	TARGET	SLIDE=
Printing and Graphical Control Options		
Specifies the time ID format	ID	FORMAT=
Specifies printed output	PROC SIMILARITY	PRINT=
Specifies detailed printed output	PROC SIMILARITY	PRINTDETAILS
Specifies graphical output	PROC SIMILARITY	PLOTS=
Miscellaneous Options		
Specifies that analysis variables are processed in ascending order	PROC SIMILARITY	SORTNAMES
Specifies the ordering of the processing of the input and target variables	PROC SIMILARITY	ORDER=

PROC SIMILARITY Statement

PROC SIMILARITY *options* ;

The following options can be used in the PROC SIMILARITY statement.

DATA=*SAS-data-set*
names the SAS data set that contains the time series, transactional, or sequence input data for the procedure. If the DATA= option is not specified, the most recently created SAS data set is used.

ORDER=*order-option*
specifies the order in which the variables listed in the INPUT and TARGET statements are to be processed. This ordering affects the OUTSEQUENCE=, OUTPATH=, OUTMEASURE=, and OUTSUM= data sets, in addition to the printed and graphical output. The SORTNAMES option also affects the ordering of the analysis. You must specify one of the following *order-options*:

INPUT specifies that each INPUT variable be processed and then the TARGET variables be processed. The results are stored and printed based only on the INPUT variables.

INPUTTARGET specifies that each INPUT variable be processed and then the TARGET variables be processed. The results are stored and printed based on both the INPUT and TARGET variables. This is the default.

TARGET specifies that each TARGET variable be processed and then the INPUT variables be processed. The results are stored and printed based only on the TARGET variables.

TARGETINPUT specifies that each TARGET variable be processed and then the INPUT variables be processed. The results are stored and printed based on both the TARGET and INPUT variables.

OUT=*SAS-data-set*
names the output data set to contain the time series variables specified in the subsequent INPUT and TARGET statements. If an ID variable is specified in the ID statement, it is also included in the OUT= data set. The values are accumulated based on the ID statement INTERVAL= option or the ACCUMULATE= options or both. The values are transformed based on the INPUT or TARGET statement TRANSFORM=, DIF=, and SDIF= options in this order. The OUT= data set is particularly useful when you want to further analyze, model, or forecast the resulting time series with other SAS/ETS procedures.

OUTMEASURE=*SAS-data-set*
names the output data set to contain the detailed similarity measures by time ID value. The form of the OUTMEASURE= data set is determined by the PROC SIMILARITY statement SORTNAMES and ORDER= options.

OUTPATH=*SAS-data-set*
names the output data set to contain the path used to compute the similarity measures for each slide and warp. The form of the OUTPATH= data set is determined by the PROC SIMILARITY statement SORTNAMES and ORDER= options. If a user-defined similarity measure is specified, the path cannot be determined; therefore, the OUTPATH= data set does not contain information related to this measure.

OUTSEQUENCE=*SAS-data-set*
names the output data set to contain the sequences used to compute the similarity measures for each slide and warp. The form of the OUTSEQUENCE= data set is determined by the PROC SIMILARITY statement SORTNAMES and ORDER= options.

OUTSUM=*SAS-data-set*

names the output data set to contain the similarity measure summary. The OUTSUM= data set is particularly useful when analyzing large numbers of series and only the summary of the results are needed. The form of the OUTSUM= data set is determined by the PROC SIMILARITY statement SORTNAMES and ORDER= options.

PLOTS=*option*

PLOTS=(*options* **. . .)**

specifies the graphical output desired. To specify multiple *options*, separate them by spaces and enclose the group in parentheses. By default, the SIMILARITY procedure produces no graphical output. The following graphical *options* are available:

COSTS	plots graphics for time warp costs.
DISTANCES	plots graphics for similarity absolute and relative distances (OUTPATH= data set).
INPUTS	plots graphics for input variable time series (OUT= data set).
MAPS	plots graphics for time warp maps (OUTPATH= data set).
MEASURES	plots graphics for similarity measures (OUTMEASURE= data set).
NORMALIZED	plots graphics for both the input and target variable normalized sequence. These plots are displayed only when the INPUT or TARGET statement NORMALIZE= option is specified.
PATHS	plots time warp paths graphics (OUTPATH= data set).
SCALED	plots graphics for both the input variable scaled sequence. These plots are displayed only when the INPUT statement SCALE= option is specified.
SEQUENCES	plots graphics for both the input and target variable sequence (OUTSEQUENCE= data set).
TARGETS	plots graphics for the target variable time series (OUT= data set).
WARPS	plots graphics for time warps (OUTPATH= data set).
ALL	is the same as PLOTS=(INPUTS TARGETS SEQUENCES NORMALIZED SCALED DISTANCES PATHS MAPS WARPS COST MEASURES).

PRINT=*option*

PRINT=(*options* **. . .)**

specifies the printed output desired. To specify multiple *options*, separate them by spaces and enclose the group in parentheses. By default, the SIMILARITY procedure produces no printed output. The following printing *options* are available:

DESCSTATS	prints the descriptive statistics for the working time series.
PATHS	prints the path statistics table. If a user-defined similarity measure is specified, the path cannot be determined; therefore, the PRINT=PATHS table is not printed for this measure.
COSTS	prints the cost statistics table.
WARPS	prints the warp summary table.

SLIDES prints the slides summary table.

SUMMARY prints the similarity measure summary table.

ALL is the same as PRINT=(DESCSTATS PATHS COSTS WARPS SLIDES SUMMARY).

PRINTDETAILS
specifies that the output requested with the PRINT= option be printed in greater detail.

SEASONALITY=*integer*
specifies the length of the seasonal cycle where *integer* ranges from one to 10,000. For example, SEASONALITY=3 means that every group of three time periods forms a seasonal cycle. By default, the length of the seasonal cycle is 1 (no seasonality) or the length implied by the INTERVAL= option specified in the ID statement. For example, INTERVAL=MONTH implies that the length of the seasonal cycle is 12.

SORTNAMES
specifies that the variables specified in the INPUT and TARGET statements be processed in alphabetical order of the variable names. By default, the SIMILARITY procedure processes the variables in the order in which they are listed. The ORDER= option also affects the ordering in which the analysis is performed.

BY Statement

A BY statement can be used with PROC SIMILARITY to obtain separate dummy variable definitions for groups of observations defined by the BY variables.

When a BY statement appears, the procedure expects the input data set to be sorted in order of the BY variables.

If your input data set is not sorted in ascending order, use one of the following alternatives:

- Sort the data by using the SORT procedure with a similar BY statement.
- Specify the option NOTSORTED or DESCENDING in the BY statement for the SIMILARITY procedure. The NOTSORTED option does not mean that the data are unsorted, but rather that the data are arranged in groups (according to values of the BY variables) and that these groups are not necessarily in alphabetical or increasing numeric order.
- Create an index on the BY variables by using the DATASETS procedure.

For more information about the BY-group processing, see *SAS Language Reference: Concepts*. For more information about the DATASETS procedure, see the discussion in the *Base SAS Procedures Guide.*

FCMPOPT Statement

FCMPOPT *options* **;**

The FCMPOPT statement specifies the following options that are related to user-defined functions and subroutines:

QUIET=ON | OFF
specifies whether the nonfatal errors and warnings that are generated by the user-defined SAS language functions and subroutines are printed to the log. Nonfatal errors are usually associated with operations with missing values. The default is QUIET=ON.

TRACE=ON | OFF
specifies whether the user-defined SAS language functions and subroutines tracings are printed to the log. Tracings are the results of every operation executed. This option is generally used for debugging. The default is TRACE=OFF.

ID Statement

ID *variable* **INTERVAL=** *interval options* **;**

The ID statement names a numeric variable that identifies observations in the input and output data sets. The ID variable's values are assumed to be SAS date, time, or datetime values. In addition, the ID statement specifies the (desired) frequency associated with the time series. The ID statement options also specify how the observations are accumulated and how the time ID values are aligned to form the time series. The options specified affect all variables listed in subsequent INPUT and TARGET statements. If an ID statement is specified, the INTERVAL= option must also be specified. The other ID statement options are optional. If an ID statement is not specified, the observation number, with respect to the BY group, is used as the time ID.

The following options can be used with the ID statement:

ACCUMULATE=*option*
specifies how the data set observations are accumulated within each time period. The frequency (width of each time interval) is specified by the INTERVAL= option. The ID variable contains the time ID values. Each time ID variable value corresponds to a specific time period. The accumulated values form the time series, which is used in subsequent analysis.

The ACCUMULATE= option is particularly useful when there are zero or more than one input observations that coincide with a particular time period (for example, time-stamped transactional data). The EXPAND procedure offers additional frequency conversions and transformations that can also be useful in creating a time series.

The following *options* determine how the observations are accumulated within each time period based on the ID variable and the frequency specified by the INTERVAL= option:

NONE	No accumulation occurs; the ID variable values must be equally spaced with respect to the frequency. This is the default option.
TOTAL	Observations are accumulated based on the total sum of their values.

AVERAGE | AVG	Observations are accumulated based on the average of their values.
MINIMUM | MIN	Observations are accumulated based on the minimum of their values.
MEDIAN | MED	Observations are accumulated based on the median of their values.
MAXIMUM | MAX	Observations are accumulated based on the maximum of their values.
N	Observations are accumulated based on the number of nonmissing observations.
NMISS	Observations are accumulated based on the number of missing observations.
NOBS	Observations are accumulated based on the number of observations.
FIRST	Observations are accumulated based on the first of their values.
LAST	Observations are accumulated based on the last of their values.
STDDEV | STD	Observations are accumulated based on the standard deviation of their values.
CSS	Observations are accumulated based on the corrected sum of squares of their values.
USS	Observations are accumulated based on the uncorrected sum of squares of their values.

If the ACCUMULATE= option is specified, the SETMISSING= option is useful for specifying how accumulated missing values are treated. If missing values should be interpreted as zero, then SETMISSING=0 should be used. The section "Details: SIMILARITY Procedure" on page 2273 describes accumulation in greater detail.

ALIGN=*option*

controls the alignment of SAS dates that are used to identify output observations. The ALIGN= option accepts the following values: BEGINNING | BEG | B, MIDDLE | MID | M, and ENDING | END | E. ALIGN=BEGINNING is the default.

END=*option*

specifies a SAS date, datetime, or time value that represents the end of the data. If the last time ID variable value is less than the END= value, the series is extended with missing values. If the last time ID variable value is greater than the END= value, the series is truncated. For example, END="&sysdate"D uses the automatic macro variable SYSDATE to extend or truncate the series to the current date. The START= and END= options can be used to ensure that data that are associated within each BY group contain the same number of observations.

FORMAT=*format*

specifies the SAS format for the time ID values. If the FORMAT= option is not specified, the default format is implied by the INTERVAL= option. For example, FORMAT=DATE9. specifies that the DATE9. SAS format be used. Notice that the terminating "." is required when specifying a SAS format.

INTERVAL=*interval*

specifies the frequency of the accumulated time series. For example, if the input data set consists of quarterly observations, then INTERVAL=QTR should be used. If the SEASONALITY= option is not specified, the length of the seasonal cycle is implied from the INTERVAL= option. For example, INTERVAL=QTR implies a seasonal cycle of length 4. If the ACCUMULATE= option is also specified, the INTERVAL= option determines the time periods for the accumulation of observations.

NOTSORTED

specifies that the time ID values are not in sorted order. The SIMILARITY procedure sorts the data with respect to the time ID prior to analysis if the NOTSORTED option is specified.

SETMISSING=*option* | *number*

specifies how missing values (either actual or accumulated) are interpreted in the accumulated time series. If a *number* is specified, missing values are set to that number. If a missing value indicates an unknown value, the SETMISSING= option should not be used. If a missing value indicates no value, then SETMISSING=0 should be used. You typically use SETMISSING=0 for transactional data, because no recorded data usually implies no activity. The following *options* can also be used to determine how missing values are assigned:

MISSING	Missing values are set to missing. This is the default option.
AVERAGE | AVG	Missing values are set to the accumulated average value.
MINIMUM | MIN	Missing values are set to the accumulated minimum value.
MEDIAN | MED	Missing values are set to the accumulated median value.
MAXIMUM | MAX	Missing values are set to the accumulated maximum value.
FIRST	Missing values are set to the accumulated first nonmissing value.
LAST	Missing values are set to the accumulated last nonmissing value.
PREVIOUS | PREV	Missing values are set to the previous period's accumulated nonmissing value. Missing values at the beginning of the accumulated series remain missing.
NEXT	Missing values are set to the next period's accumulated nonmissing value. Missing values at the end of the accumulated series remain missing.

START=*option*

specifies a SAS date, datetime, or time value that represents the beginning of the data. If the first time ID variable value is greater than the START= value, missing values are added to the beginning of the series. If the first time ID variable value is less than the START= value, the series is truncated. The START= and END= options can be used to ensure that data that are associated with each BY group contain the same number of observations.

ZEROMISS=*option*

specifies how beginning and ending zero values (either actual or accumulated) are interpreted in the accumulated time series. The following *options* can also be used to determine how beginning and ending zero values are assigned:

NONE	Beginning and ending zeros are unchanged. This is the default.
LEFT	Beginning zeros are set to missing.
RIGHT	Ending zeros are set to missing.
BOTH	Both beginning and ending zeros are set to missing.

If the accumulated series is all missing or zero, the series is not changed.

INPUT Statement

INPUT *variable-list* < */ options*> **;**

The INPUT statement lists the input numeric variables in the DATA= data set whose values are to be accumulated to form the time series or represent ordered numeric sequences (when no ID statement is specified).

An input data set variable can be specified in only one INPUT or TARGET statement. Any number of INPUT statements can be used. The following *options* can be used with an INPUT statement:

ACCUMULATE=*option*
specifies how the data set observations are accumulated within each time period for the variables listed in the INPUT statement. If the ACCUMULATE= option is not specified in the INPUT statement, accumulation is determined by the ACCUMULATE= option of the ID statement. If the ACCUMULATE= option is not specified in the ID statement or the INPUT statement, no accumulation is performed. For more information, see the ACCUMULATE= option in the ID statement.

DIF=(*numlist***)**
specifies the differencing to be applied to the accumulated time series. The list of differencing orders must be separated by spaces or commas. For example, DIF=(1,3) specifies first, then third order, differencing. Differencing is applied after time series transformation. The TRANSFORM= option is applied before the DIF= option. Simple differencing is useful when you want to detrend the time series before computing the similarity measures.

NORMALIZE=*option*
specifies the sequence normalization to be applied to the working input sequence. The following normalization *options* are provided:

NONE	No normalization is applied. This option is the default.
ABSOLUTE	Absolute normalization is applied.
STANDARD	Standard normalization is applied.
User-Defined	Normalization is computed by a user-defined subroutine that is created using the FCMP procedure, where *User-Defined* is the subroutine name.

Normalization is applied to the working input sequence, which can be a subset of the working input time series if the SLIDE=INDEX or SLIDE=SEASON option is specified.

SCALE=*option*
specifies the scaling of the working input sequence with respect to the working target sequence. Scaling is performed after normalization. The following scaling *options* are provided:

NONE	No scaling is applied. This option is the default.
ABSOLUTE	Absolute scaling is applied.
STANDARD	Standard scaling is applied.
User-Defined	Scaling is computed by a user-defined subroutine that is created using the FCMP procedure, where *User-Defined* is the subroutine name.

Scaling is applied to the working input sequence, which can be a subset of the working input time series if the SLIDE=INDEX or SLIDE=SEASON option is specified.

SDIF=(*numlist*)
specifies the seasonal differencing to be applied to the accumulated time series. The list of seasonal differencing orders must be separated by spaces or commas. For example, SDIF=(1,3) specifies first, then third, order seasonal differencing. Differencing is applied after time series transformation. The TRANSFORM= option is applied before the SDIF= option. Seasonal differencing is useful when you want to deseasonalize the time series before computing the similarity measures.

SETMISSING=*option* | *number*

SETMISS=*option* | *number*
specifies how missing values (either actual or accumulated) are interpreted in the accumulated time series or ordered sequence for variables listed in the INPUT statement. If the SETMISSING= option is not specified in the INPUT statement, missing values are set based on the SETMISSING= option in the ID statement. If the SETMISSING= option is not specified in the ID statement or the INPUT statement, no missing value interpretation is performed. For more information, see the SETMISSING= option in the ID statement.

TRANSFORM=*option*
specifies the time series transformation to be applied to the accumulated time series. The following transformations are provided:

NONE	No transformation is applied. This option is the default.
LOG	Logarithmic transformation is applied.
SQRT	Square-root transformation is applied.
LOGISTIC	Logistic transformation is applied.
BOXCOX(*number*)	Box-Cox transformation with parameter is applied, where the real *number* is between –5 and 5.
User-Defined	Transformation is computed by a user-defined subroutine that is created using the FCMP procedure, where *User-Defined* is the subroutine name.

When the TRANSFORM= option is specified, the time series must be strictly positive unless a user-defined function is used.

TRIMMISSING=*option*

TRIMMISSING=*option*
specifies how missing values (either actual or accumulated) are trimmed from the accumulated time series or ordered sequence for variables that are listed in the INPUT statement. The following trimming options are provided:

NONE	No missing value trimming is applied.
LEFT	Beginning missing values are trimmed.
RIGHT	Ending missing values are trimmed.
BOTH	Both beginning and ending missing value are trimmed. This is the default.

ZEROMISS=*option*

specifies how beginning and ending zero values (either actual or accumulated) are interpreted in the accumulated time series or ordered sequence for variables listed in the INPUT statement. If the ZEROMISS= option is not specified in the INPUT statement, beginning and ending zero values are set based on the ZEROMISS= option of the ID statement. If the ZERO= option is not specified in the ID statement or the INPUT statement, no zero value interpretation is performed. For more information, see the ZEROMISS= option in the ID statement.

TARGET Statement

TARGET *variable-list < / options>* **;**

The TARGET statement lists the numeric target variables in the DATA= data set whose values are to be accumulated to form the time series or represent ordered numeric sequences (when no ID statement is specified).

An input data set variable can be specified in only one INPUT or TARGET statement. Any number of TARGET statements can be used. The following *options* can be used with a TARGET statement:

ACCUMULATE=*option*

specifies how the data set observations are accumulated within each time period for the variables listed in the TARGET statement. If the ACCUMULATE= option is not specified in the TARGET statement, accumulation is determined by the ACCUMULATE= option in the ID statement. If the ACCUMULATE= option is not specified in the ID statement or the TARGET statement, no accumulation is performed. For more information, see the ACCUMULATE= option in the ID statement.

COMPRESS=*option* **| (***options***)**

specifies the sliding sequence (global) and warping (local) compression range of the target sequence with respect to the input sequence. Compression of the target sequence is the same as expansion of the input sequence and vice versa. The compression limits are defined based on the length of the target sequence and are imposed on the target sequence. The following compression options are provided:

GLOBALABS=*integer* specifies the absolute global compression, where *integer* ranges from zero to 10,000. GLOBALABS=0 implies no global compression, which is the default unless the GLOBALPCT= option is specified.

GLOBALPCT=*number* specifies global compression as a percentage of the length of the target sequence, where *number* ranges from zero to 100. GLOBALPCT=0 implies no global compression, which is the default. GLOBALPCT=100 implies maximum allowable compression.

LOCALABS=*integer* specifies the absolute local compression, where *integer* ranges from zero to 10,000. The default is maximum allowable absolute local compression unless the LOCALPCT= option is specified.

LOCALPCT=*number* specifies local compression as a percentage of the length of the target sequence, where *number* ranges from zero to 100. The percentage specified by the LOCALPCT= option must be less than the GLOBALPCT= option. LOCALPCT=0 implies no local compression. LOCALPCT=100 implies maximum allowable local compression. The default is LOCALPCT=100.

If the SLIDE=NONE or SLIDE=SEASON option is specified in the TARGET statement, the global compression options are ignored. To disallow local compression, use the option COMPRESS=(LOCALPCT=0 LOCALABS=0).

If the SLIDE=INDEX option is specified, the global compression options are not ignored. To completely disallow both global and local compression, use the option COMPRESS=(GLOBALPCT=0 LOCALPCT=0) or COMPRESS=(GLOBALABS=0 LOCALABS=0). To allow only local compression, use the option COMPRESS=(GLOBALPCT=0 GLOBALABS=0). These are the default compression options.

The preceding options can be used in combination to specify the desired amount of global and local compression as the following examples illustrate, where L_c denotes the global compression limit and l_c denotes the local compression limit:

- COMPRESS=(GLOBALPCT=20) allows the global and local compression to range from zero to $L_c = \min\left(\lfloor 0.2N_y \rfloor, (N_y - 1)\right)$.
- COMPRESS=(GLOBALPCT=20 GLOBALABS=10) allows the global and local compression to range from zero to $L_c = \min\left(\lfloor 0.2N_y \rfloor, \min\left((N_y - 1), 10\right)\right)$.
- COMPRESS=(LOCALPCT=10) allows the local compression to range from zero to $l_c = \min\left(\lfloor 0.1N_y \rfloor, (N_y - 1)\right)$.
- COMPRESS=(LOCALPCT=20 LOCALABS=5) allows the local compression to range from zero to $l_c = \min\left(\lfloor 0.2N_y \rfloor, \min\left((N_y - 1), 5\right)\right)$.
- COMPRESS=(GLOBALPCT=20 LOCALPCT=20) allows the global compression to range from zero to $L_c = \min\left(\lfloor 0.2N_y \rfloor, (N_y - 1)\right)$ and allows the local compression to range from zero to $l_c = \min\left(\lfloor 0.2N_y \rfloor, (N_y - 1)\right)$.
- COMPRESS=(GLOBALPCT=20 GLOBALABS=10 LOCALPCT=10 LOCALABS=5) allows the global compression to range from zero to $L_c = \min\left(\lfloor 0.2N_y \rfloor, \min\left((N_y - 1), 10\right)\right)$ and allows the local compression to range from zero to $l_c = \min\left(\lfloor 0.1N_y \rfloor, \min\left((N_y - 1), 5\right)\right)$.

Suppose T_z is the length of the input time series and N_y is the length of the target sequence. The *valid* global compression limit, L_c, is always limited by the length of the target sequence: $0 \leq L_c < N_y$.

Suppose N_x is the length of the input sequence and N_y is the length of the target sequence. The *valid* local compression limit, l_c, is always limited by the lengths of the input and target sequence: $\max\left(0, (N_y - N_x)\right) \leq l_c < N_y$.

DIF=(*numlist*)

specifies the differencing to be applied to the accumulated time series. The list of differencing orders must be separated by spaces or commas. For example, DIF=(1,3) specifies first, then third, order differencing. Differencing is applied after time series transformation. The TRANSFORM= option is applied before the DIF= option. Simple differencing is useful when you want to detrend the time series before computing the similarity measures.

EXPAND=*option* | (*options*)

specifies the sliding sequence (global) and warping (local) expansion range of the target sequence with respect to the input sequence. Expansion of the target sequence is the same as compression of the input sequence and vice versa. The expansion limits are defined based on the length of the input sequence, but are imposed on the target sequence. The following expansion *options* are provided:

GLOBALABS=*integer* specifies the absolute global expansion, where *integer* ranges from zero to 10,000. GLOBALABS=0 implies no global expansion, which is the default unless the GLOBALPCT= option is specified.

GLOBALPCT=*number* specifies global expansion as a percentage of the length of the target sequence, where *number* ranges from zero to 100. GLOBALPCT=0 implies no global expansion, which is the default unless the GLOBALABS= option is specified. GLOBALPCT=100 implies maximum allowable global expansion.

LOCALABS=*integer* specifies the absolute local expansion, where *integer* ranges from zero to 10,000. The default is the maximum allowable absolute local expansion unless the LOCALPCT= option is specified.

LOCALPCT=*number* specifies local expansion as a percentage of the length of the target sequence, where *number* ranges from zero to 100. LOCALPCT=0 implies no local expansion. LOCALPCT=100 implies maximum allowable local expansion. The default is LOCALPCT=100.

If the SLIDE=NONE or SLIDE=SEASON option is specified in the TARGET statement, the global expansion options are ignored. To disallow local expansion, use the option EXPAND=(LOCALPCT=0 LOCALABS=0).

If the SLIDE=INDEX option is specified, the global expansion options are not ignored. To completely disallow both global and local expansion, use the option EXPAND=(GLOBALPCT=0 LOCALPCT=0) or EXPAND=(GLOBALABS=0 LOCALABS=0). To allow only local expansion, use the option EXPAND=(GLOBALPCT=0 GLOBALABS=0). These are the default expansion options.

The preceding options can be used in combination to specify the desired amount of global and local expansion as the following examples illustrate, where L_e denotes the global expansion limit and l_e denotes the local expansion limit:

- EXPAND=(GLOBALPCT=20) allows the global and local expansion to range from zero to $L_e = \min\left(\lfloor 0.2N_y \rfloor, (N_y - 1)\right)$.
- EXPAND=(GLOBALPCT=20 GLOBALABS=10) allows the global and local expansion to range from zero to $L_e = \min\left(\lfloor 0.2N_y \rfloor, \min\left((N_y - 1), 10\right)\right)$.
- EXPAND=(LOCALPCT=10) allows the local expansion to range from zero to $l_e = \min\left(\lfloor 0.1N_y \rfloor, (N_y - 1)\right)$.
- EXPAND=(LOCALPCT=10 LOCALABS=5) allows the local expansion to range from zero to $l_e = \min\left(\lfloor 0.1N_y \rfloor, \min\left((N_y - 1), 5\right)\right)$.
- EXPAND=(GLOBALPCT=20 LOCALPCT=10) allows the global expansion to range from zero to $L_e = \min\left(\lfloor 0.2N_y \rfloor, (N_y - 1)\right)$ and allows the local expansion to range from zero to $l_e = \min\left(\lfloor 0.1N_y \rfloor, (N_y - 1)\right)$.
- EXPAND=(GLOBALPCT=20 GLOBALABS=10 LOCALPCT=10 LOCALABS=5) allows the global expansion to range from zero to $L_e = \min\left(\lfloor 0.2N_y \rfloor, \min\left((N_y - 1), 10\right)\right)$ and allows the local expansion to range from zero to $l_e = \min\left(\lfloor 0.1N_y \rfloor, \min\left((N_y - 1), 5\right)\right)$.

Suppose T_z is the length of the input time series and N_y is the length of the target sequence. The *valid* global expansion limit, L_e, is always limited by the length of the input time series: $0 \le L_e < T_z$.

Suppose N_x is the length of the input sequence and N_y is the length of the target sequence. The *valid* local expansion limit, l_e, is always limited by the lengths of the input and target sequence: $\max\left(0, \left(N_x - N_y\right)\right) \leq l_e < N_x$.

MEASURE=*option*

specifies the similarity measure to be computed by using the working input and target sequences. The following similarity measures are provided:

SQRDEV	squared deviation. This option is the default.
ABSDEV	absolute deviation
MSQRDEV	mean squared deviation
MSQRDEVINP	mean squared deviation relative to the length of the input sequence
MSQRDEVTAR	mean squared deviation relative to the length of the target sequence
MSQRDEVMIN	mean squared deviation relative to the minimum valid path length
MSQRDEVMAX	mean squared deviation relative to the maximum valid path length
MABSDEV	mean absolute deviation
MABSDEVINP	mean absolute deviation relative to the length of the input sequence
MABSDEVTAR	mean absolute deviation relative to the length of the target sequence
MABSDEVMIN	mean absolute deviation relative to the minimum valid path length
MABSDEVMAX	mean absolute deviation relative to the maximum valid path length
User-Defined	The measure is computed by a user-defined function created by using the FCMP procedure, where *User-Defined* is the function name.

NORMALIZE=*option*

specifies the sequence normalization to be applied to the working target sequence. The following normalization *options* are provided:

NONE	No normalization is applied. This option is the default.
ABSOLUTE	Absolute normalization is applied.
STANDARD	Standard normalization is applied.
User-Defined	Normalization is computed by a user-defined subroutine that is created by using the FCMP procedure, where *User-Defined* is the subroutine name.

PATH=*option*

specifies the similarity measure and warping path information to be computed using the working input and target sequences. The following similarity measures and warping path are provided:

User-Defined	The measure and path are computed by a user-defined subroutine that is created by using the FCMP procedure, where *User-Defined* is the subroutine name.

For computational efficiency, the PATH= option should be only used when you want to compute both the similarity measure and the warping path information. If only the similarity measure is needed, use the MEASURE= option. If you specify both the MEASURE= and PATH= option in the TARGET statement, the PATH= option takes precedence.

SDIF=(*numlist*)

specifies the seasonal differencing to be applied to the accumulated time series. The list of seasonal differencing orders must be separated by spaces or commas. For example, SDIF=(1,3) specifies first, then third, order seasonal differencing. Differencing is applied after time series transformation. The TRANSFORM= option is applied before the SDIF= option. Seasonal differencing is useful when you want to deseasonalize the time series before computing the similarity measures.

SETMISSING=*option* | *number*

SETMISS=*option* | *number*

option specifies how missing values (either actual or accumulated) are interpreted in the accumulated time series for variables that are listed in the TARGET statement. If the SETMISSING= option is not specified in the TARGET statement, missing values are set based on the SETMISSING= option in the ID statement. If the SETMISSING= option is not specified in the ID statement or the TARGET statement, no missing value interpretation is performed. For more information, see the SETMISSING= option in the ID statement.

SLIDE=*option*

specifies the sliding of the target sequence with respect to the input sequence. The following slides are provided:

NONE	No sequence sliding. The input time series is compared with the target sequence directly with no sliding. This option is the default.
INDEX	Slide by time index. The input time series is compared with the target sequence by observation index.
SEASON	Slide by seasonal index. The input time series is compared with the target sequence by seasonal index.

The SLIDE= option takes precedence over the COMPRESS= and EXPAND= options.

TRANSFORM=*option*

specifies the time series transformation to be applied to the accumulated time series. The following transformations are provided:

NONE	No transformation is applied. This option is the default.
LOG	Logarithmic transformation is applied.
SQRT	Square-root transformation is applied.
LOGISTIC	Logistic transformation is applied.
BOXCOX(*number*)	Box-Cox transformation with parameter is applied, where the real *number* is between –5 and 5
User-Defined	Transformation is computed by a user-defined subroutine that is created by using the FCMP procedure, where *User-Defined* is the subroutine name.

When the TRANSFORM= option is specified, the time series must be strictly positive unless a user-defined function is used.

TRIMMISSING=*option*
TRIMMISS= *option*
specifies how missing values (either actual or accumulated) are trimmed from the accumulated time series or ordered sequence for variables that are listed in the TARGET statement. The following trimming options are provided:

NONE No missing value trimming is applied.

LEFT Beginning missing values are trimmed.

RIGHT Ending missing values are trimmed.

BOTH Both beginning and ending missing values are trimmed. This is the default.

ZEROMISS=*option*
specifies how beginning and ending zero values (either actual or accumulated) are interpreted in the accumulated time series or ordered sequence for variables listed in the TARGET statement. If the ZEROMISS= option is not specified in the TARGET statement, beginning and ending values are set based on the ZEROMISS= option in the ID statement. For more information, see the ZEROMISS= option in the ID statement.

Details: SIMILARITY Procedure

You can use the SIMILARITY procedure to do the following functions, which are done in the order shown. First, you can form time series data from transactional data with the options shown:

1. accumulation ACCUMULATE= option
2. missing value interpretation SETMISSING= option
3. zero value interpretation ZEROMISS= option

Next, you can transform the accumulated time series to form the working time series with the following options. Transformations are useful when you want to stabilize the time series before computing the similarity measures. Simple and seasonal differencing are useful when you want to detrend or deseasonalize the time series before computing the similarity measures. Often, but not always, the TRANSFORM=, DIF=, and SDIF= options should be specified in the same way for both the target and input variables.

4. time series transformation TRANSFORM= option
5. time series differencing DIF= and SDIF= options
6. time series missing value trimming TRIMMISSING= option
7. time series descriptive statistics PRINT=DESCSTATS option

After the working series is formed, you can treat it as an ordered sequence that can be normalized or scaled. Normalizations are useful when you want to compare the "shape" or "profile" of the time series. Scaling is useful when you want to compare the input sequence to the target sequence while discounting the variation of the target sequence.

8. normalization	NORMALIZE= option
9. scaling	SCALE= option

After the working sequences are formed, you can compute similarity measures between input and target sequences:

10. sliding	SLIDE= option
11. warping	COMPRESS= and EXPAND= options
12. similarity measure	MEASURE= and PATH= options

The SLIDE= option specifies observation-index sliding, seasonal-index sliding, or no sliding. The COMPRESS= and EXPAND= options specify the warping limits. The MEASURE= and PATH= options specify how the similarity measures are computed.

Accumulation

If the ACCUMULATE= option is specified in the ID, INPUT, or TARGET statement, data set observations are accumulated within each time period. The frequency (width of each time interval) is specified by the INTERVAL= option in the ID statement. The ID variable contains the time ID values. Each time ID value corresponds to a specific time period. Accumulation is particularly useful when the input data set contains transactional data, whose observations are not spaced with respect to any particular time interval. The accumulated values form the time series, which is used in subsequent analyses.

For example, suppose a data set contains the following observations:

```
19MAR1999    10
19MAR1999    30
11MAY1999    50
12MAY1999    20
23MAY1999    20
```

If the INTERVAL=MONTH option is specified, all of the preceding observations fall within three time periods of March 1999, April 1999, and May 1999. The observations are accumulated within each time period as follows:

If the ACCUMULATE=NONE option is specified, an error is generated because the ID variable values are not equally spaced with respect to the specified frequency (MONTH).

If the ACCUMULATE=TOTAL option is specified, the data are accumulated as follows:

```
O1MAR1999    40
O1APR1999     .
O1MAY1999    90
```

If the ACCUMULATE=AVERAGE option is specified, the data are accumulated as follows:

```
O1MAR1999    20
O1APR1999     .
O1MAY1999    30
```

If the ACCUMULATE=MINIMUM option is specified, the data are accumulated as follows:

```
O1MAR1999    10
O1APR1999     .
O1MAY1999    20
```

If the ACCUMULATE=MEDIAN option is specified, the data are accumulated as follows:

```
O1MAR1999    20
01APR1999     .
O1MAY1999    20
```

If the ACCUMULATE=MAXIMUM option is specified, the data are accumulated as follows:

```
O1MAR1999    30
O1APR1999     .
O1MAY1999    50
```

If the ACCUMULATE=FIRST option is specified, the data are accumulated as follows:

```
O1MAR1999    10
O1APR1999     .
O1MAY1999    50
```

If the ACCUMULATE=LAST option is specified, the data are accumulated as follows:

```
O1MAR1999    30
O1APR1999     .
O1MAY1999    20
```

If the ACCUMULATE=STDDEV option is specified, the data are accumulated as follows:

```
O1MAR1999    14.14
O1APR1999     .
O1MAY1999    17.32
```

As can be seen from the preceding examples, even though the data set observations contain no missing values, the accumulated time series can have missing values.

Missing Value Interpretation

Sometimes missing values should be interpreted as unknown values. But sometimes missing values are known, such as when missing values are created from accumulation and no observations should be interpreted as no (zero) value. In the former case, the SETMISSING= option in the ID, INPUT, or TARGET statement can be used to interpret how missing values are treated. The SETMISSING=0 option should be used when missing observations are to be treated as no (zero) values. In other cases, missing values should be interpreted as global values, such as minimum or maximum values of the accumulated series. The accumulated and interpreted time series is used in subsequent analyses.

The SETMISSING=0 option should be used with missing observations are to be treated as a zero value. In other cases, missing values should be interpreted as global values, such as minimum or maximum values of the accumulated series. The accumulated and interpreted time series is then used in subsequent analyses.

Zero Value Interpretation

When querying certain databases for time-stamped data based on a particular time range, time periods that contain no data are sometimes assigned zero values. For certain analyses, it is more desirable to assign these values to missing. Often, these beginning or ending zero values need to be interpreted as missing values. The ZEROMISS= option in the ID, INPUT, or TARGET statement specifies that the beginning, ending, or both the beginning and ending values are to be interpreted as zero values.

Time Series Transformation

Transformations are useful when you want to stabilize the time series before computing the similarity measures. There are four transformations available, for strictly positive series only. Let $y_t > 0$ be the original time series, and let w_t be the transformed series. The transformations are defined as follows:

Log
: is the logarithmic transformation,

$$w_t = \ln(y_t)$$

Logistic
: is the logistic transformation,

$$w_t = \ln(cy_t/(1 - cy_t))$$

where the scaling factor c is

$$c = (1 - e^{-6})10^{-\mathrm{ceil}(\log_{10}(\max(y_t)))}$$

and $\mathrm{ceil}(x)$ is the smallest integer greater than or equal to x.

Square root
: is the square root transformation,

$$w_t = \sqrt{y_t}$$

Box-Cox
: is the Box-Cox transformation,

$$w_t = \begin{cases} \frac{y_t^{\lambda}-1}{\lambda} & \lambda \neq 0 \\ \ln(y_t) & \lambda = 0 \end{cases}$$

User-Defined
: is the transformation computed by a user-defined subroutine that is created by using the FCMP procedure, where *User-Defined* is the subroutine name.

Other time series transformations can be performed prior to invoking the SIMILARITY procedure by using the SAS/ETS EXPAND procedure or the DATA step.

Time Series Differencing

After optionally transforming the series, the accumulated series can be simply or seasonally differenced using the INPUT or TARGET statement DIF= and SDIF= options. Simple and seasonal differencing are useful when you want to detrend or deseasonalize the time series before computing the similarity measures.

For example, suppose y_t is a monthly time series. The following examples of the DIF= and SDIF= options demonstrate how to simply and seasonally difference the time series: DIF=(1,3) specifies first, then third, order differencing; SDIF=(1,3) specifies first, then third, order seasonal differencing.

Additionally, assuming that y_t is strictly positive, the INPUT or TARGET statement TRANSFORM= option and the DIF= and SDIF= options can be combined.

Time Series Missing Value Trimming

In some instances, missing values should be interpreted as an unknown observation, but other times, missing values are known and should be interpreted as a zero value. This is the case when missing values are created from accumulation, and a missing observation should be interpreted as having no value (meaning a value of zero). In the former case, the SETMISSING=option in the ID, INPUT, or TARGET, statement can be used to interpret how missing observations should be treated. By default, missing values, at the beginning and ending of the data set, are trimmed from the data set prior to analysis. This can be performed using TRIMMISS=both.

Time Series Descriptive Statistics

After a series has been optionally accumulated and transformed with missing values interpreted, descriptive statistics can be computed for the resulting working series by specifying the PRINT=DESCSTATS option. This option produces an ODS table that contains the sum, mean, minimum, maximum, and standard deviation of the working series.

Input and Target Sequences

After the input and target working series are formed, they can be treated as two ordered sequences. Given an input time sequence, x_i, for $i = 1$ to N_x, where i is the input sequence index, and a target time sequence, y_j, for $j = 1$ to N_y, where j is the target sequence index, these sequences are analyzed for similarity.

Sliding Sequences

Similarity measures can be computed between the target sequence and any contiguous subsequences of the input time series.

There are three types of sequence sliding:

- no sliding
- slide by time index
- slide by season index

For more information, see Leonard et al. (2008).

Time Warping

Time warping allows for the comparison between target and input sequences of differing lengths by compressing or expanding the input sequence with respect the target sequence while respecting the order of the sequence elements.

For more information, see Leonard et al. (2008).

Sequence Normalization

The working (input or target) sequence can be normalized prior to further analysis. Let q_i be the original sequence with mean μ_q and standard deviation σ_q, and let r_t be the normalized sequence. The normalizations are defined as follows:

- Standard is the standard normalization

  $r_i = (q_i - \mu_q)/\sigma_q$

- Absolute is the absolute normalization

  $r_i = (q_i - \min(q_i))/(\max(q_i) - \min(q_i))$

- User-defined is a user-defined normalization created by the FCMP procedure.

Sequence Scaling

The working input sequence can be scaled to the working target sequence. Sequence scaling is applied after normalization. Let y_j be the working target sequence with mean μ_y and standard deviation σ_y. Let x_i be the working input sequence and let q_i be the scaled sequence. The scaling is defined as follows:

- Standard is the standard normalization

  $q_i = (x_i - \mu_y)/\sigma_y$

- Absolute is the absolute scaling

  $q_i = (x_i - \min(y_j))/(\max(y_j) - \min(y_j))$

- User-defined is a user-defined scaling created by the FCMP procedure.

Similarity Measures

The working input sequence can be compared to the working target sequence to create a similarity. For more information, see Leonard et al. (2008).

User-Defined Functions and Subroutines

A user-defined routine can be written in the SAS language by using the FCMP procedure or in the C language by using both the FCMP procedure and the PROTO procedure, respectively. The SIMILARITY procedure cannot use C language routines directly. The procedure can use only SAS language routines that might or might not call C language routines. Creating user-defined routines is more completely described in the FCMP procedure and the PROTO procedure documentation. The FCMP and PROTO procedures are part of Base SAS software.

The SAS language provides integrated memory management and exception handling such as operations on missing values. The C language provides flexibility and allows the integration of existing C language libraries. However, proper memory management and exception handling are solely the responsibility of the user. Additionally, the support for standard C libraries is restricted. If you have a choice, it is highly recommended that you write user-defined functions and subroutines in the SAS language using the FCMP procedure.

For each of the tasks previously described, the following sections describe the required subroutine or function signature and provide examples of using a user-defined routine with the SIMILARITY procedure.

Time Series Transformations

A user-defined transformation subroutine has the subroutine signature

```
SUBROUTINE <SUBROUTINE-NAME> ( <ARRAY-NAME>[*] );
```

where the *array-name* is the time series to be transformed.

For example, to duplicate the functionality of the built-in TRANSFORM=LOG option in the INPUT and TARGET statement, the following SAS statements create a user-defined version of this transformation called MYTRANSFORM and store this subroutine in the catalog SASUSER.MYSIMILAR:

```
proc fcmp outlib=sasuser.mysimilar.package;

   subroutine mytransform( series[*] );

      outargs series;

      length = DIM(series);

      do i = 1 to length;
         value = series[i];
         if value > 0 then do;
            series[i] = log( value );
         end;
         else do;
```

```
            series[i] = .;
         end;
      end;

   endsub;

run;
```

This user-defined subroutine can be specified in the TRANSFORM= option in the INPUT or TARGET statement as follows:

```
options cmplib = sasuser.mysimilar;

proc similarity ...;
...
input  myinput  / transform=mytransform;
target mytarget / transform=mytransform;
...
run;
```

Sequence Normalizations

A user-defined normalization subroutine has the signature

```
SUBROUTINE <SUBROUTINE-NAME> ( <ARRAY-NAME>[*] );
```

where the *array-name* is the sequence to be normalized.

For example, to duplicate the functionality of the built-in NORMALIZE=ABSOLUTE option in the INPUT and TARGET statement, the following SAS statements create a user-defined version of this normalization called MYNORMALIZE and store this subroutine in the catalog SASUSER.MYSIMILAR:

```
proc fcmp outlib=sasuser.mysimilar.package;

   subroutine mynormalize( sequence[*] );

      outargs sequence;

      length = DIM(sequence);
      minimum = .; maximum = .;

      do i = 1 to length;
         value = sequence[i];
         if nmiss(minimum) | nmiss(maximum) then do;
            minimum = value;
            maximum = value;
         end;
         if nmiss(value) = 0 then do;
            if value < minimum then minimum = value;
            if value > maximum then maximum = value;
         end;
      end;

      do i = 1 to length;
         value = sequence[i];
```

```
         if nmiss( value ) | minimum > maximum then do;
            sequence[i] = .;
         end;
         else do;
            sequence[i] = (value - minimum) / (maximum - minimum);
         end;
      end;

   endsub;

run;
```

This user-defined subroutine can be specified in the NORMALIZE= option in the INPUT or TARGET statement as follows:

```
options cmplib = sasuser.mysimilar;

proc similarity ...;
   ...
   input  myinput  / normalize=mynormalize;
   target mytarget / normalize=mynormalize;
   ...
run;
```

Sequence Scaling

A user-defined scaling subroutine has the signature

```
SUBROUTINE <SUBROUTINE-NAME> ( <ARRAY-NAME>[*], <ARRAY-NAME>[*] );
```

where the first *array-name* is the target sequence and the second *array-name* is the input sequence to be scaled.

For example, to duplicate the functionality of the built-in SCALE=ABSOLUTE option in the INPUT statement, the following SAS statements create a user-defined version of this scaling called MYSCALE and store this subroutine in the catalog SASUSER.MYSIMILAR:

```
proc fcmp outlib=sasuser.mysimilar.package;

   subroutine myscale( target[*], input[*] );

      outargs input;

      length = DIM(target);
      minimum = .; maximum = .;

      do i = 1 to length;
         value = target[i];
         if nmiss(minimum) | nmiss(maximum) then do;
            minimum = value;
            maximum = value;
         end;
         if nmiss(value) = 0 then do;
            if value < minimum then minimum = value;
            if value > maximum then maximum = value;
```

```
         end;
      end;

      do i = 1 to length;
         value = input[i];
         if nmiss( value ) | minimum > maximum then do;
            input[i] = .;
         end;
         else do;
            input[i] = (value - minimum) / (maximum - minimum);
         end;
      end;

   endsub;

run;
```

This user-defined subroutine can be specified in the SCALE= option in the INPUT statement as follows:

```
options cmplib=sasuser.mysimilar;

proc similarity ...;
   ...
   input  myinput  / scale=myscale;
   ...
run;
```

Similarity Measures

A user-defined similarity measure function has the signature

```
FUNCTION <FUNCTION-NAME> ( <ARRAY-NAME>[*], <ARRAY-NAME>[*] );
```

where the first *array-name* is the target sequence and the second *array-name* is the input sequence. The return value of the function is the similarity measure associated with the target sequence and the input sequence.

For example, to duplicate the functionality of the built-in MEASURE=ABSDEV option in the TARGET statement with no warping, the following SAS statements create a user-defined version of this measure called MYMEASURE and store this subroutine in the catalog SASUSER.MYSIMILAR:

```
proc fcmp outlib=sasuser.mysimilar.package;

   function mymeasure( target[*], input[*] );

      length = min(DIM(target), DIM(input));
      sum = 0; num = 0;

      do i = 1 to length;
         x = input[i];
         w = target[i];
         if nmiss(x) = 0 & nmiss(w) = 0 then do;
            d = x - w;
            sum = sum + abs(d);
            num = num + 1;
         end;
```

```
      end;

      if num <= 0 then return(.);

      return(sum);

   endsub;

run;
```

This user-defined function can be specified in the MEASURE= option in the TARGET statement as follows:

```
options cmplib=sasuser.mysimilar;

proc similarity ...;
   ...
   target  mytarget  / measure=mymeasure;
   ...
run;
```

For another example, to duplicate the functionality of the built-in MEASURE=SQRDEV and MEASURE=ABSDEV options by using the C language, the following SAS statements create a user-defined C language version of these measures called DTW_SQRDEV_C and DTW_ABSDEV_C and store these functions in the catalog SASUSER.CSIMIL.CFUNCS. DTW refers to dynamic time warping. These C language functions can then be called by SAS language functions and subroutines.

```
proc proto package=sasuser.csimil.cfuncs;

mapmiss double = 999999999;

double dtw_sqrdev_c( double * target / iotype=input,
                     int      targetLength,
                     double * input / iotype=input,
                     int      inputLength );

externc dtw_sqrdev_c;
 double dtw_sqrdev_c( double * target,
                      int      targetLength,
                      double * input,
                      int      inputLength )
   {
      int      i,j;
      double   x,w,d;
      double * prev = (double *)malloc( sizeof(double)*targetLength);
      double * curr = (double *)malloc( sizeof(double)*inputLength);
      if ( prev == 0 || curr == 0 ) return 999999999;

      x = input[0];
      for ( j=0; j<targetLength; j++ ) {
         w = target[j];
         d = x - w;
         d = d*d;
         if ( j == 0 ) prev[j] = d;
         else          prev[j] = d + prev[j-1];
      }
```

```
      for (i=1; i<inputLength; i++ ) {
         x = input[i];

         j = 0;
         w = target[j];
         d = x - w;
         d = d*d;
         curr[j] = d + prev[j];

         for (j=1; j<targetLength; j++ ) {
            w = target[j];
            d = x - w;
            d = d*d;
            curr[j] = d + fmin( prev[j],
                                fmin( prev[j-1], curr[j]));
         }

         if ( i < targetLength ) {
           for( j=0; j<inputLength; j++ )
             prev[j] = curr[j];
         }
       }

       d = curr[inputLength-1];
       free( (char*) prev);
       free( (char*) curr);
       return( d );
   }
externcend;

double dtw_absdev_c( double * target / iotype=input,
                     int      targetLength,
                     double * input / iotype=input,
                     int      inputLength );
externc dtw_absdev_c;
 double dtw_absdev_c( double * target,
                      int      targetLength,
                      double * input,
                      int      inputLength )
    {
      int      i,j;
      double   x,w,d;
      double * prev = (double *)malloc( sizeof(double)*targetLength);
      double * curr = (double *)malloc( sizeof(double)*inputLength);
      if ( prev == 0 || curr == 0 ) return 999999999;

      x = input[0];
      for ( j=0; j<targetLength; j++ ) {
        w = target[j];
        d = x - w;
        d = fabs(d);
        if (j == 0) prev[j] = d;
        else prev[j] = d + prev[j-1];
```

```
      }

      for (i=1; i<inputLength; i++ ) {
        x = input[i];

        j = 0;
        w = target[j];
        d = x - w;
        d = fabs(d);
        curr[j] = d + prev[j];

        for (j=1; j<targetLength; j++) {
          w = target[j];
          d = x - w;
          d = fabs(d);
          curr[j] = d + fmin( prev[j],
                              fmin( prev[j-1], curr[j] ));
        }

        if ( i < inputLength) {
          for ( j=0; j<targetLength; j++ )
             prev[j] = curr[j];
        }

      }

      d = curr[inputLength-1];
      free( (char*) prev);
      free( (char*) curr);
      return( d );
   }
 externcend;

run;
```

The preceding SAS statements create two C language functions that can then be used in SAS language functions or subroutines or both. However, these functions cannot be directly used by the SIMILARITY procedure. In order to use these C language functions in the SIMILARITY procedure, two SAS language functions must be created that call these two C language functions. The following SAS statements create two user-defined SAS language versions of these measures called DTW_SQRDEV and DTW_ABSDEV and stores these functions in the catalog SASUSER.MYSIMILAR.FUNCS. These SAS language functions use the previously created C language function; the SAS language functions can then be used by the SIMILARITY procedure.

```
proc fcmp outlib=sasuser.mysimilar.funcs
          inlib=sasuser.cfuncs;

   function dtw_sqrdev( target[*], input[*] );
      dev = dtw_sqrdev_c(target,DIM(target),input,DIM(input));
      return( dev );
   endsub;

   function dtw_absdev( target[*], input[*] );
```

```
      dev = dtw_absdev_c(target,DIM(target),input,DIM(input));
      return( dev );
   endsub;

run;
```

This user-defined function can be specified in the MEASURE= option in the TARGET statement as follows:

```
options cmplib=sasuser.mysimilar;

proc similarity ...;
   ...
   target  mytarget    / measure=dtw_sqrdev;
   target  yourtarget  / measure=dtw_absdev;
   ...
run;
```

Similarity Measures and Warping Path

A user-defined similarity measure and warping path information function has the signature

```
FUNCTION <FUNCTION-NAME> ( <ARRAY-NAME>[*], <ARRAY-NAME>[*],
                           <ARRAY-NAME>[*], <ARRAY-NAME>[*],
                           <ARRAY-NAME>[*] );
```

where the first *array-name* is the target sequence, the second *array-name* is the input sequence, the third *array-name* is the returned target sequence indices, the fourth *array-name* is the returned input sequence indices, and the fifth *array-name* is the returned path distances. The returned value of the function is the similarity measure. The last three returned arrays are used to compute the path and cost statistics.

The returned sequence indices must represent a valid warping path; that is, integers greater than zero and less than or equal to the sequence length and recorded in ascending order. The returned path distances must be nonnegative numbers.

Output Data Sets

The SIMILARITY procedure can create the OUT=, OUTMEASURE=, OUTPATH= , OUTSEQUENCE=, and OUTSUM= data sets. In general, these data sets contain the variables listed in the BY statement. The ID statement time ID variable is also included in the data sets when the time dimension is important. If an analysis step related to an output data step fails, then the values of this step are not recorded or are set to missing in the related output data set, and appropriate error and warning messages are recorded in the SAS log.

OUT= Data Set

The OUT= data set contains the variables that are specified in the BY, ID, INPUT, and TARGET statements. If the ID statement is specified, the ID variable values are aligned and extended based on the ALIGN=, INTERVAL=, START=, and END= options. The values of the variables specified in the INPUT and TARGET statements are accumulated based on the ACCUMULATE= option, missing values are interpreted based on the SETMISSING= option, and zero values are interpreted using the ZEROMISS= option. The accumulated time series is transformed based on the TRANSFORM=, DIF=, and SDIF= options.

OUTMEASURE= Data Set

The OUTMEASURE= data set records the similarity measures between each INPUT and TARGET statement variable with respect to each time ID value. The form of the OUTMEASURE= data set depends on the SORTNAMES and ORDER= options. The OUTMEASURE= data set contains the variables specified in the BY statement in addition to the variables listed below.

For ORDER=INPUTTARGET and ORDER=TARGETINPUT, the OUTMEASURE= data set has the following form:

INPUT	input variable name
TARGET	target variable name
TIMEID	time ID values
INPSEQ	input sequence values
TARSEQ	target sequence values
SIM	similarity measures

The OUTMEASURE= data set is ordered by the variables _INPUT_, then _TARGET_, then _TIMEID_ when ORDER=INPUTTARGET. The OUTMEASURE= data set is ordered by the variables _TARGET_, then _INPUT_, then _TIMEID_ when ORDER=TARGETINPUT.

For ORDER=INPUT, the OUTMEASURE= data set has the following form:

INPUT	input variable name
TIMEID	time ID values
INPSEQ	input sequence values
target-names	similarity measures that are associated with each TARGET statement variable name

The OUTMEASURE= data set is ordered by the variables _INPUT_, then _TIMEID_.

For ORDER=TARGET, the OUTMEASURE= data set has the following form:

TARGET	target variable name
TIMEID	time ID values
TARSEQ	target sequence values
input-names	similarity measures that are associated with each INPUT statement variable name

The OUTMEASURE= data set is ordered by the variables _TARGET_, then _TIMEID_.

OUTPATH= Data Set

The OUTPATH= data set records the path analysis between each INPUT and TARGET statement variable. This data set records the path sequences for each slide index and for each warp index associated with the slide index. The sequence values recorded are normalized and scaled based on the NORMALIZE= and SCALE= options.

The OUTPATH= data set contains the variables specified in the BY statement and the following variables:

INPUT	input variable name
TARGET	target variable name
TIMEID	time ID values
SLIDE	slide index
WARP	warp index
INPSEQ	input sequence values
TARSEQ	target sequence values
INPPTH	input path index
TARPTH	target path index
METRIC	distance metric values

The Warp Index indicates the total amount of warping for each slide. A negative number represents compression of the target sequence. A positive number represents expansion of the target sequence. The Warp Index is always zero for SLIDE=NONE and SLIDE=SEASON.

The sorting of the OUTPATH= data set depends on the SORTNAMES and ORDER= options.

The OUTPATH= data set is ordered by the variables _INPUT_, then _TARGET_, then _TIMEID_ when ORDER=INPUTTARGET or ORDER=INPUT. The OUTPATH= data set is ordered by the variables _TARGET_, then _INPUT_, then _TIMEID_ when ORDER=TARGETINPUT or ORDER=TARGET.

If there are a large number of slides or warps or both, this data set might be large.

OUTSEQUENCE= Data Set

The OUTSEQUENCE= data set records the input and target sequences that are associated with each INPUT and TARGET statement variable. This data set records the input and target sequence values for each slide index and for each warp index that is associated with the slide index. The sequence values that are recorded are normalized and scaled based on the NORMALIZE= and SCALE= options. This data set also contains the similarity measure associated with the two sequences.

The OUTSEQUENCE= data set contains the variables specified in the BY statement in addition to the following variables:

INPUT	input variable name
TARGET	target variable name

TIMEID	time ID values
SLIDE	slide index
WARP	warp index
INPSEQ	input sequence values
TARSEQ	target sequence values
SIM	similarity measure
STATUS	sequence status

The sorting of the OUTSEQUENCE= data set depends on the SORTNAMES and ORDER= options.

The OUTSEQUENCE= data set is ordered by the variables _INPUT_, then _TARGET_, then _TIMEID_ when ORDER=INPUTTARGET or ORDER=INPUT. The OUTSEQUENCE= data set is ordered by the variables _TARGET_, then _INPUT_, then _TIMEID_ when ORDER=TARGETINPUT or ORDER=TARGET.

If there are a large number of slides or warps or both, this data set might be large.

OUTSUM= Data Set

The OUTSUM= data set summarizes the similarity measures between each INPUT and TARGET statement variable. The form of the OUTSUM= data set depends on the SORTNAMES and ORDER= options. If the SORTNAMES option is specified, each variable (INPUT or TARGET) is analyzed in ascending order. The OUTSUM= data set contains the variables specified in the BY statement in addition to the variables listed below.

For ORDER=INPUTTARGET and ORDER=TARGETINPUT, the OUTSUM= data set has the following form:

INPUT	input variable name
TARGET	target variable name
STATUS	status flag that indicates whether the requested analyses were successful
TIMEID	time ID values
SIM	similarity measure summary

The OUTSUM= data set is ordered by the variables _INPUT_, then _TARGET_ when ORDER=INPUTTARGET. The OUTSUM= data set is ordered by the variables _TARGET_, then _INPUT_ when ORDER=TARGETINPUT.

For ORDER=INPUT, the OUTSUM= data set has the following form:

INPUT	input variable name
STATUS	status flag that indicates whether the requested analyses were successful
target-names	similarity measure summary that is associated with each TARGET statement variable name

The OUTSUM= data set is ordered by the variable _INPUT_.

For ORDER=TARGET, the OUTSUM= data set has the following form:

TARGET	target variable name
STATUS	status flag that indicates whether the requested analyses were successful
input-names	similarity measure summary that is associated with each INPUT statement variable name

The OUTSUM= data set is ordered by the variable _TARGET_.

STATUS Variable Values

The _STATUS_ variable contains a code that specifies whether the similarity analysis has been successful or not. The _STATUS_ variable can take the following values:

0	Success
3000	Accumulation failure
4000	Missing value interpretation failure
6000	Series is all missing
7000	Transformation failure
8000	Differencing failure
9000	Unable to compute descriptive statistics
10000	Normalization failure
11000	Input contains imbedded missing values
12000	Target contains imbedded missing values
13000	Scaling failure
14000	Measure failure
15000	Path failure
16000	Slide summarization failure

Printed Output

The SIMILARITY procedure optionally produces printed output by using the Output Delivery System (ODS). By default, the procedure produces no printed output. All output is controlled by the PRINT= and PRINTDETAILS options in the PROC SIMILARITY statement.

The sort, order, and form of the printed output depend on both the SORTNAMES option and the ORDER= option. If the SORTNAMES option is specified, each variable (INPUT or TARGET) is analyzed in ascending order. For ORDER=INPUTTARGET, the printed output is ordered by the INPUT statement variables (row) and then by the TARGET statement variables (row). For ORDER=TARGETINPUT, the printed output is ordered by the TARGET statement variables (row) and then by the INPUT statement variables (row). For ORDER=INPUT, the printed output is ordered by the INPUT statement variables (row) and then by the TARGET statement variables (column). For ORDER=TARGET, the printed output is ordered by the TARGET statement variables (row) and then by the INPUT statement variables (column).

In general, if an analysis step related to printed output fails, the values of that step are not printed and appropriate error and warning messages are recorded in the SAS log. The printed output is similar to the output data set; these similarities are described as follows:

PRINT=COSTS
prints the costs statistics.

PRINT=DESCSTATS
prints the descriptive statistics.

PRINT=PATHS
prints the path statistics.

PRINT=SLIDES
prints the sliding sequence summary.

PRINT=SUMMARY
prints the summary of similarity measures similar to the OUTSUM= data set.

PRINT=WARPS
prints the warp summary.

PRINTDETAILS
prints each table with greater detail.

ODS Table Names

Table 30.2 relates the PRINT= options to ODS tables.

Table 30.2 ODS Tables Produced in PROC SIMILARITY

ODS Table Name	Description	Option
CostStatistics	Cost statistics	PRINT=COSTS
DescStats	Descriptive statistics	PRINT=DESCSTATS
PathLimits	Path limits	PRINT=PATHS
PathStatistics	Path statistics	PRINT=PATHS
SlideMeasuresSummary	Summary of measure per slide	PRINT=SLIDES
MeasuresSummary	Measures summary	PRINT=SUMMARY
InputMeasuresSummary	Measures summary	PRINT=SUMMARY
TargetMeasuresSummary	Measures summary	PRINT=SUMMARY
WarpMeasuresSummary	Summary of measure per warp	PRINT=WARPS

The tables are related to a single series within a BY group.

ODS Graphics

Statistical procedures use ODS Graphics to create graphs as part of their output. ODS Graphics is described in detail in Chapter 21, “Statistical Graphics Using ODS” (*SAS/STAT User’s Guide*).

Before you create graphs, ODS Graphics must be enabled (for example, with the ODS GRAPHICS ON statement). For more information about enabling and disabling ODS Graphics, see the section "Enabling and Disabling ODS Graphics" in that chapter.

The overall appearance of graphs is controlled by ODS styles. Styles and other aspects of using ODS Graphics are discussed in the section "A Primer on ODS Statistical Graphics" in that chapter.

This section describes the use of ODS for creating graphics with the SIMILARITY procedure.

ODS Graph Names

PROC SIMILARITY assigns a name to each graph it creates by using ODS. You can use these names to selectively reference the graphs. The names are listed in Table 30.3.

Table 30.3 ODS Graphics Produced by PROC SIMILARITY

ODS Graph Name	Plot Description	Statement	PLOTS= Option
CostsPlot	Costs plot	SIMILARITY	COSTS
NormalizedSequencePlot	Normalized sequence plot	SIMILARITY	NORMALIZED
PathDistancePlot	Path distances plot	SIMILARITY	DISTANCES
PathDistanceHistogram	Path distances histogram	SIMILARITY	DISTANCES
PathRelativeDistancePlot	Path relative distances plot	SIMILARITY	DISTANCES
PathRelativeDistanceHistogram	Path relative distances histogram	SIMILARITY	DISTANCES
PathPlot	Path plot	SIMILARITY	PATHS
PathSequencesPlot	Path sequences plot	SIMILARITY	MAPS
PathSequencesScaledPlot	Scaled path sequences map plot	SIMILARITY	MAPS
ScaledSequencePlot	Scaled sequence plot	SIMILARITY	SCALED
SequencePlot	Sequence plot	SIMILARITY	SEQUENCES
SeriesPlot	Input time series plot	SIMILARITY	INPUTS
SimilarityPlot	Similarity measures plot	SIMILARITY	MEASURES
TargetSequencePlot	Target sequence plot	SIMILARITY	TARGETS
WarpPlot	Warping plot	SIMILARITY	WARPS
WarpScaledPlot	Scaled warping plot	SIMILARITY	WARPS

Time Series Plots

The time series plots (SeriesPlot) illustrate the input time series to be compared. The horizontal axis represents the input series time ID values, and the vertical axis represents the input series values.

Sequence Plots

The sequence plots (SequencePlot) illustrate the target and input sequences to be compared. The horizontal axis represents the (target or input) sequence index, and the vertical axis represents the (target or input) sequence values.

Path Plots

The path plot (PathPlot) and path limits plot (PathLimitsPlot) illustrate the path through the distance matrix. The horizontal axis represents the input sequence index, and the vertical axis represents the target sequence index. The dots represent the path coordinates. The upper parallel line represents the compression limit, and the lower parallel line represents the expansion limit. These plots visualize 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. These plots are useful for visualizing the amount of expansion and compression along the path.

Time Warp Plots

The time warp plot (WarpPlot) and scaled time warp plot (WarpScaledPlot) illustrate the time warping. The horizontal axis represents the (input and target) sequence index. The upper line plot represents the target sequence. The lower line plot 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 optimal path. These plots visualize 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. The plots are useful for visualizing the mapping between the input and target sequence values along the path. The plots are useful for comparing the path sequences or input and target sequence after time warping.

Path Sequence Plots

The path sequence plot (PathSequencesPlot) and scaled path sequence plot (PathSequencesScaledPlot) illustrate the sequence mapping along the optimal path. The horizontal axis represents the path index. The dashed line represents the time warped input sequence. The solid line represents the time warped target sequence. These plots visualize the mapping between the input and target sequence values with respect to the path index. The scaled plot with the input and target sequence values are scaled and evenly separated for visual convenience.

Path Distance Plots

The path distance plots (PathDistancePlot) and path relative distance plots (PathRelativeDistancePlot) illustrate the path (relative) distances. The horizontal axis represents the path index. The vertical needles represent the (relative) distances. The horizontal reference lines indicate one and two standard deviations.

The path distance histogram (PathDistanceHistogram) and path relative distance histogram (PathDistanceRelativeHistogram) illustrate the distribution of the path (relative) distances. The bars represent the histogram, and the solid line represents a normal distribution with the same mean and variance.

Cost Plots

The cost plot (CostPlot) and cost limits plot (CostPlot) illustrate the cost of traversing the distance matrix. The horizontal axis represents the input sequence index, and the vertical axis represents the target sequence index. The colors and shading within the plot illustrate the incremental cost of traversing the distance matrix. The upper parallel line represents the compression limit, and the lower parallel line represents the expansion limit.

Examples: SIMILARITY Procedure

Example 30.1: Accumulating Transactional Data into Time Series Data

This example uses the SIMILARITY procedure to illustrate the accumulation of time-stamped transactional data that has been recorded at no particular frequency into time series data at a specific frequency. After the time series is created, the various SAS/ETS procedures related to time series analysis, similarity analysis, seasonal adjustment and decomposition, modeling, and forecasting can be used to further analyze the time series data.

Suppose that the input data set WORK.RETAIL contains the variables STORE and TIMESTAMP and numerous other numeric transaction variables. The BY variable STORE contains values that break up the transactions into groups (BY groups). The time ID variable TIMESTAMP contains SAS date values recorded at no particular frequency. The other data set variables contain the numeric transaction values to be analyzed. It is further assumed that the input data set is sorted by the variables STORE and TIMESTAMP.

The following statements form monthly time series from the transactional data based on the median value (ACCUMULATE=MEDIAN) of the transactions recorded with each time period. The accumulated time series values for time periods with no transactions are set to zero instead of missing (SETMISS=0). Only transactions recorded between the first day of 1998 (START='01JAN1998'D) and last day of 2000 (END='31JAN2000'D) are considered and if needed are extended to include this range.

```
proc similarity data=work.retail out=mseries;
   by store;
   id timestamp interval=month
                accumulate=median
                setmiss=0
                start='01jan1998'd
                end  ='31dec2000'd;
   target _NUMERIC_;
run;
```

The monthly time series data are stored in the data set WORK.MSERIES. Each BY group associated with the BY variable STORE contains an observation for each of the 36 months associated with the years 1998, 1999, and 2000. Each observation contains the variables STORE and TIMESTAMP and each of the analysis variables in the input DATA= data set.

After each set of transactions has been accumulated to form the corresponding time series, the accumulated time series can be analyzed by using various time series analysis techniques. For example, exponentially weighted moving averages can be used to smooth each series. The following statements use the EXPAND procedure to smooth the analysis variable named STOREITEM:

```
proc expand data=mseries
            out=smoothed
            from=month;
   by store;
   id timestamp;
   convert storeitem=smooth / transform=(ewma 0.1);
run;
```

The smoothed series is stored in the data set WORK.SMOOTHED. The variable SMOOTH contains the smoothed series.

If the time ID variable TIMESTAMP contains SAS datetime values instead of SAS date values, the INTERVAL= , START=, and END= options in the SIMILARITY procedure must be changed accordingly, and the following statements could be used to accumulate the datetime transactions to a monthly interval:

```
proc similarity data=work.retail
                out=tseries;
   by store;
   id timestamp interval=dtmonth
                accumulate=median
                setmiss=0
                start='01jan1998:00:00:00'dt
                end  ='31dec2000:00:00:00'dt;
   target _NUMERIC_;
run;
```

The monthly time series data are stored in the data set WORK.TSERIES, and the time ID values use a SAS datetime representation.

Example 30.2: Similarity Analysis

This simple example illustrates how to use similarity analysis to compare two time sequences. The following statements create an example data set that 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 example data set:

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

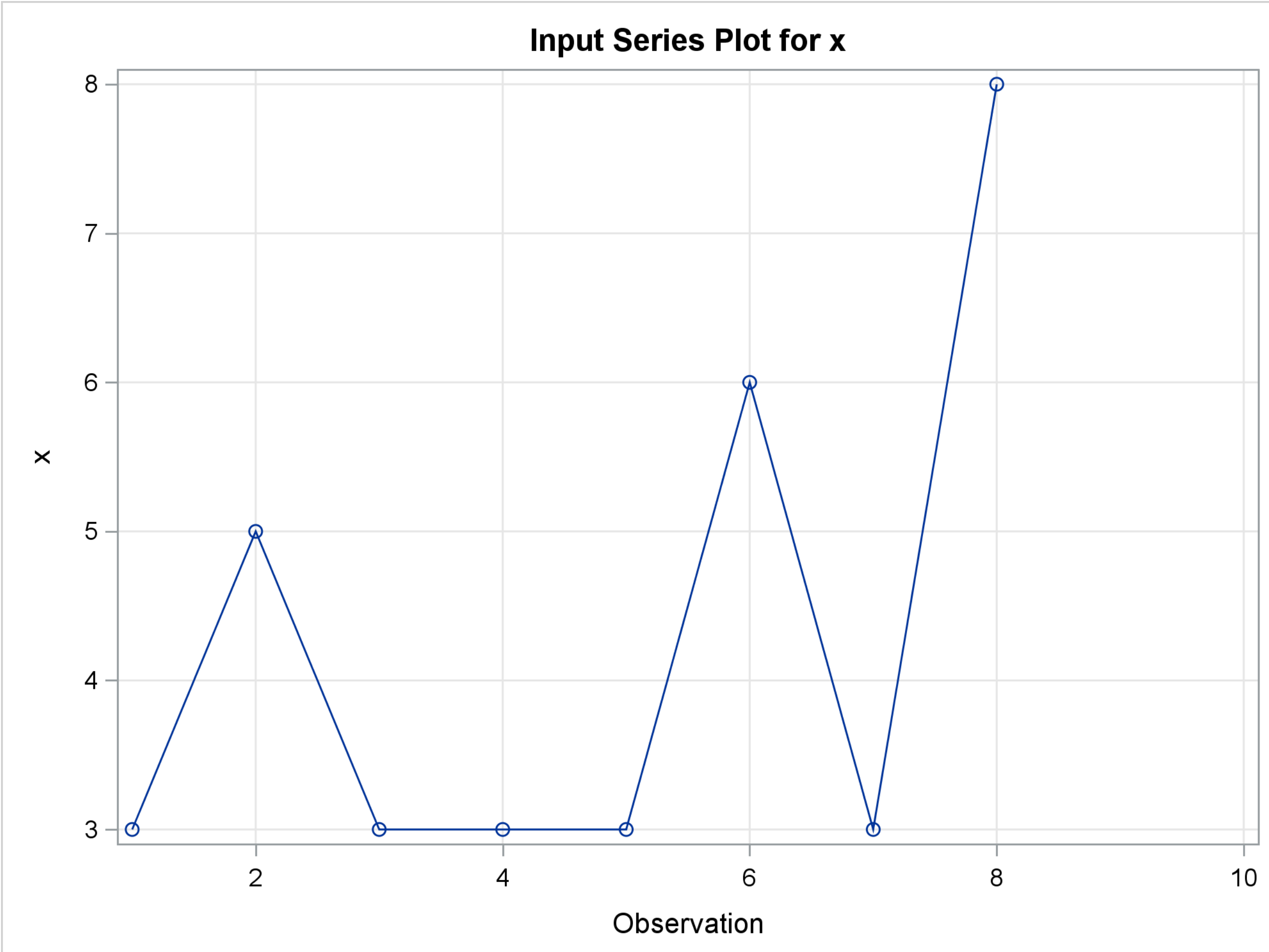
The DATA=TEST option specifies that the input data set WORK.TEST is to be used in the analysis. The OUT=_NULL_ option specifies that no output time series data set is to be created. The PRINT=ALL and PLOTS=ALL options specify that all ODS tables and graphs are to be produced. The INPUT statement specifies that the input variable is X. The TARGET statement specifies that the target variable is Y and that the similarity measure is computed using absolute deviation (MEASURE=ABSDEV).

Output 30.2.1 Description Statistics of the Input Variable, x

The SIMILARITY Procedure

Time Series Descriptive Statistics	
Variable	x
Number of Observations	10
Number of Missing Observations	2
Minimum	3
Maximum	8
Mean	4.25
Standard Deviation	1.908627

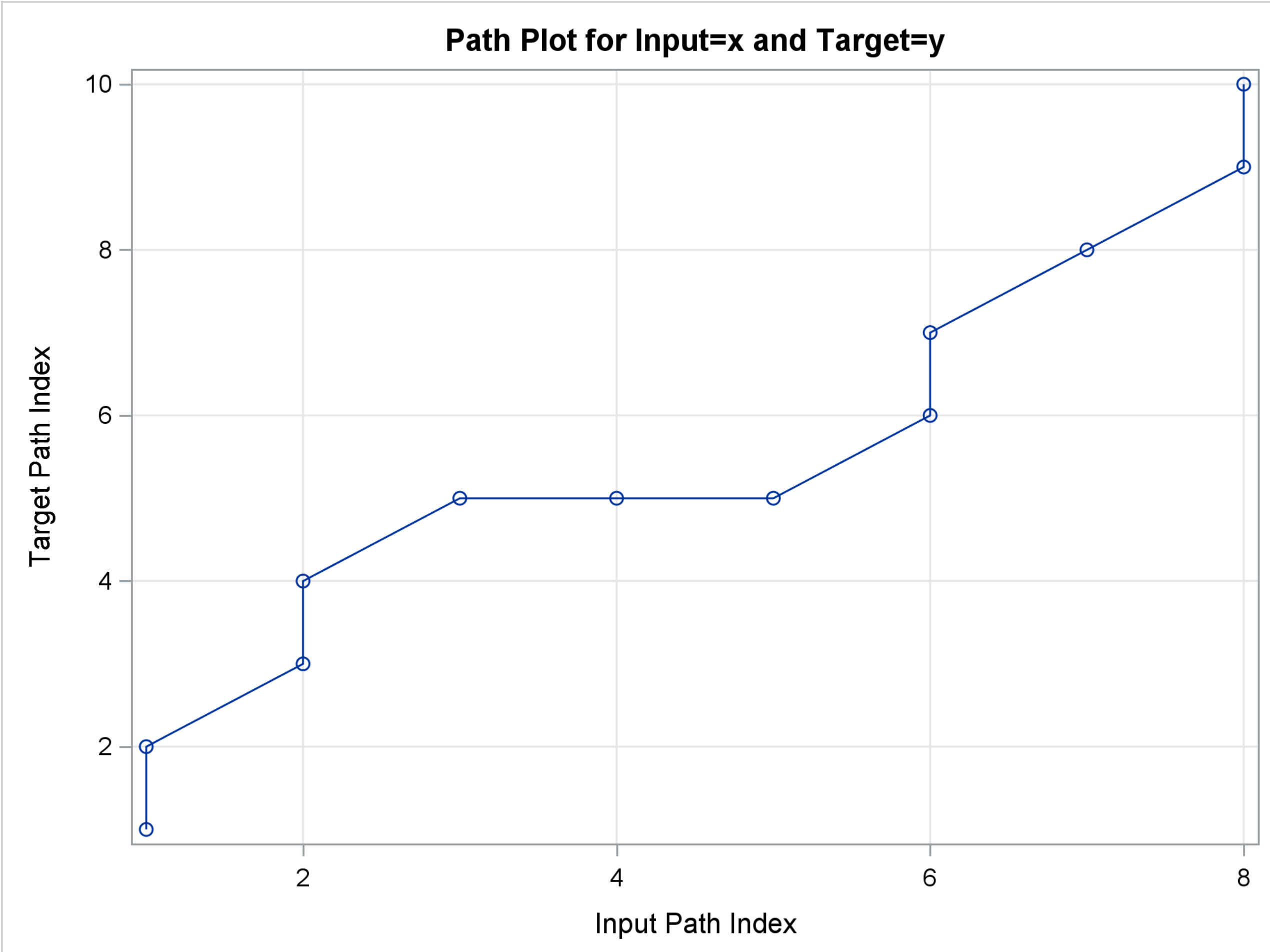
Output 30.2.2 Plot of Input Variable, x



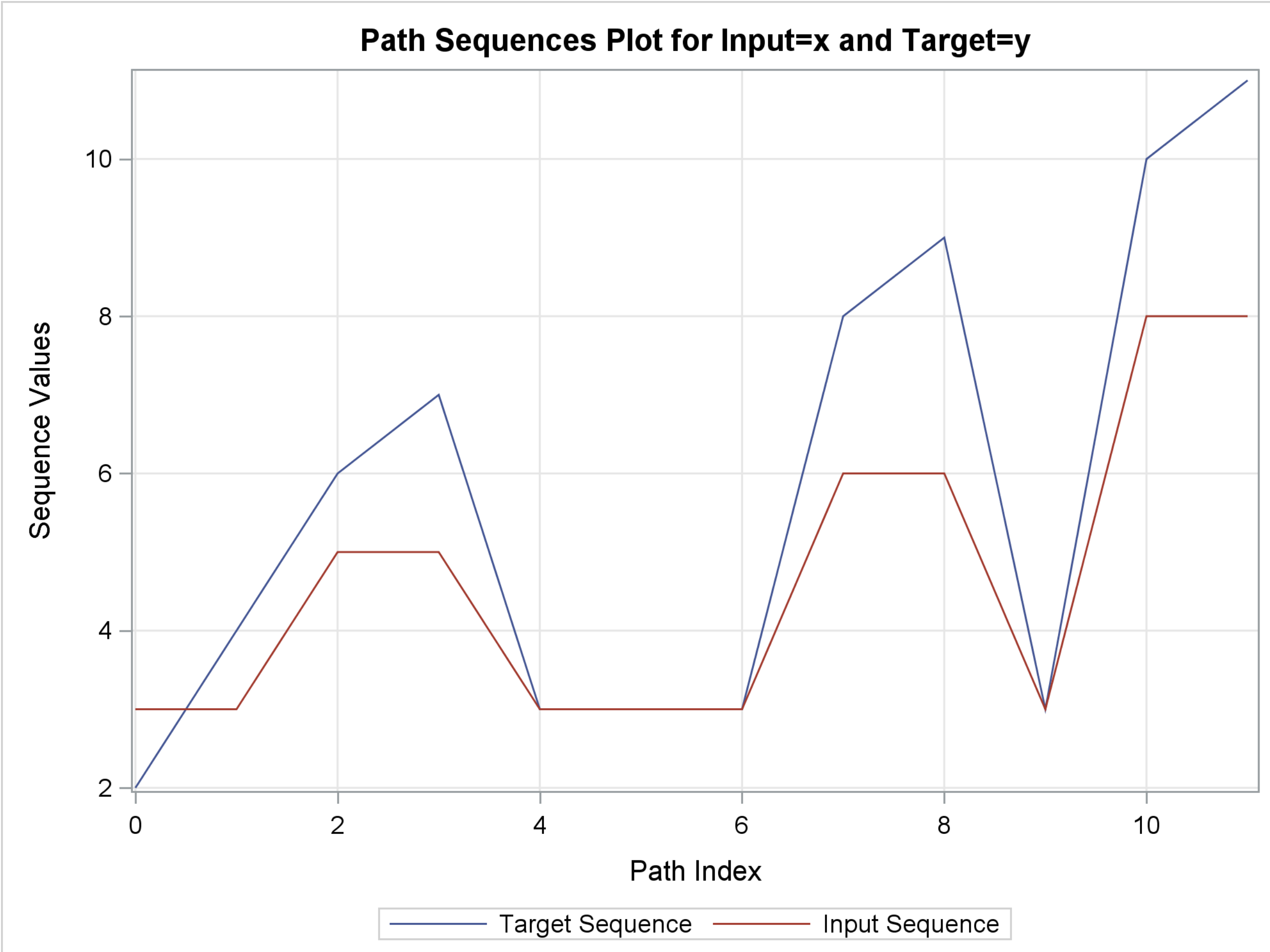
Output 30.2.3 Target Sequence Plot

Output 30.2.4 Sequence Plot

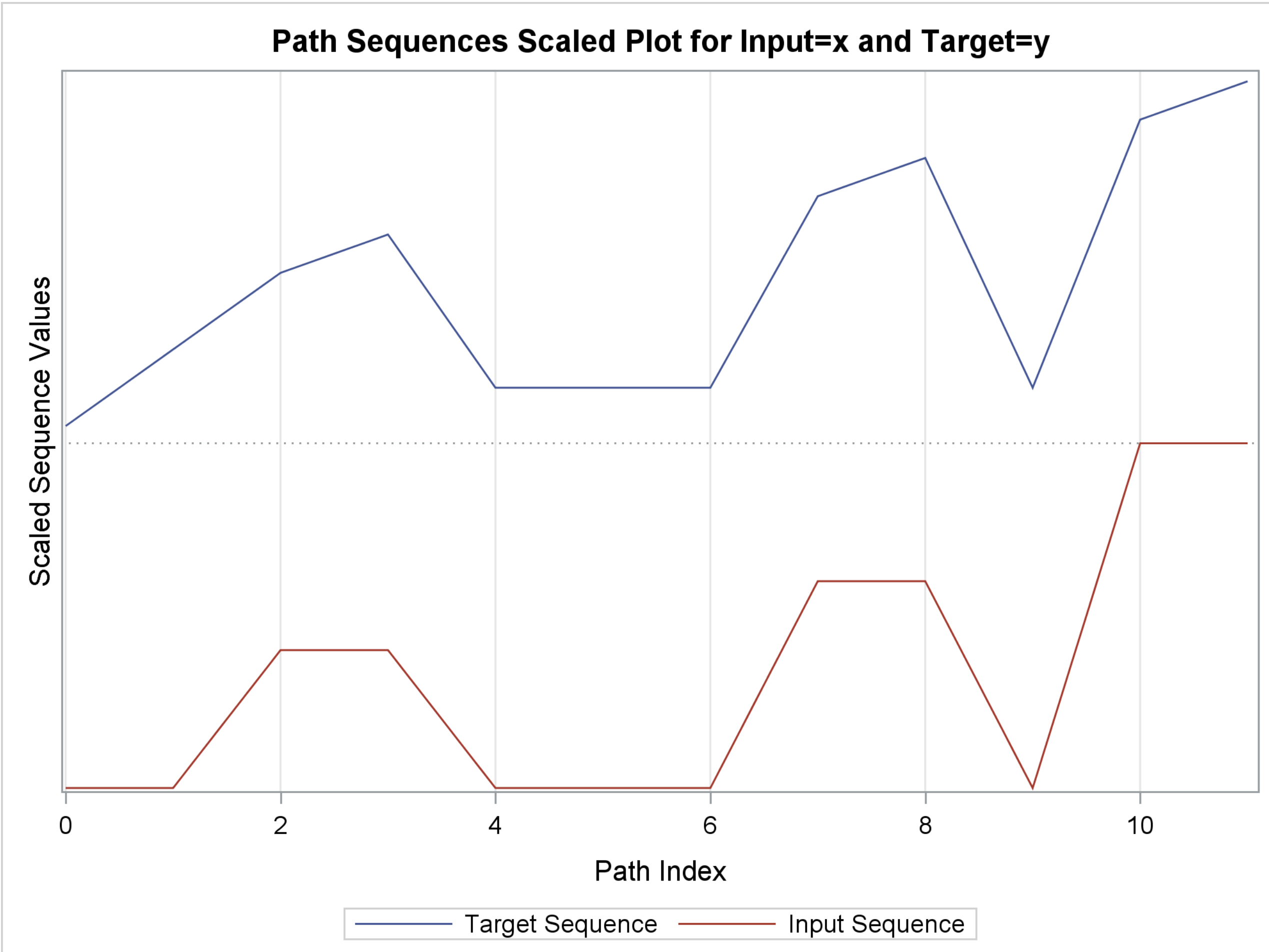
Output 30.2.5 Path Plot



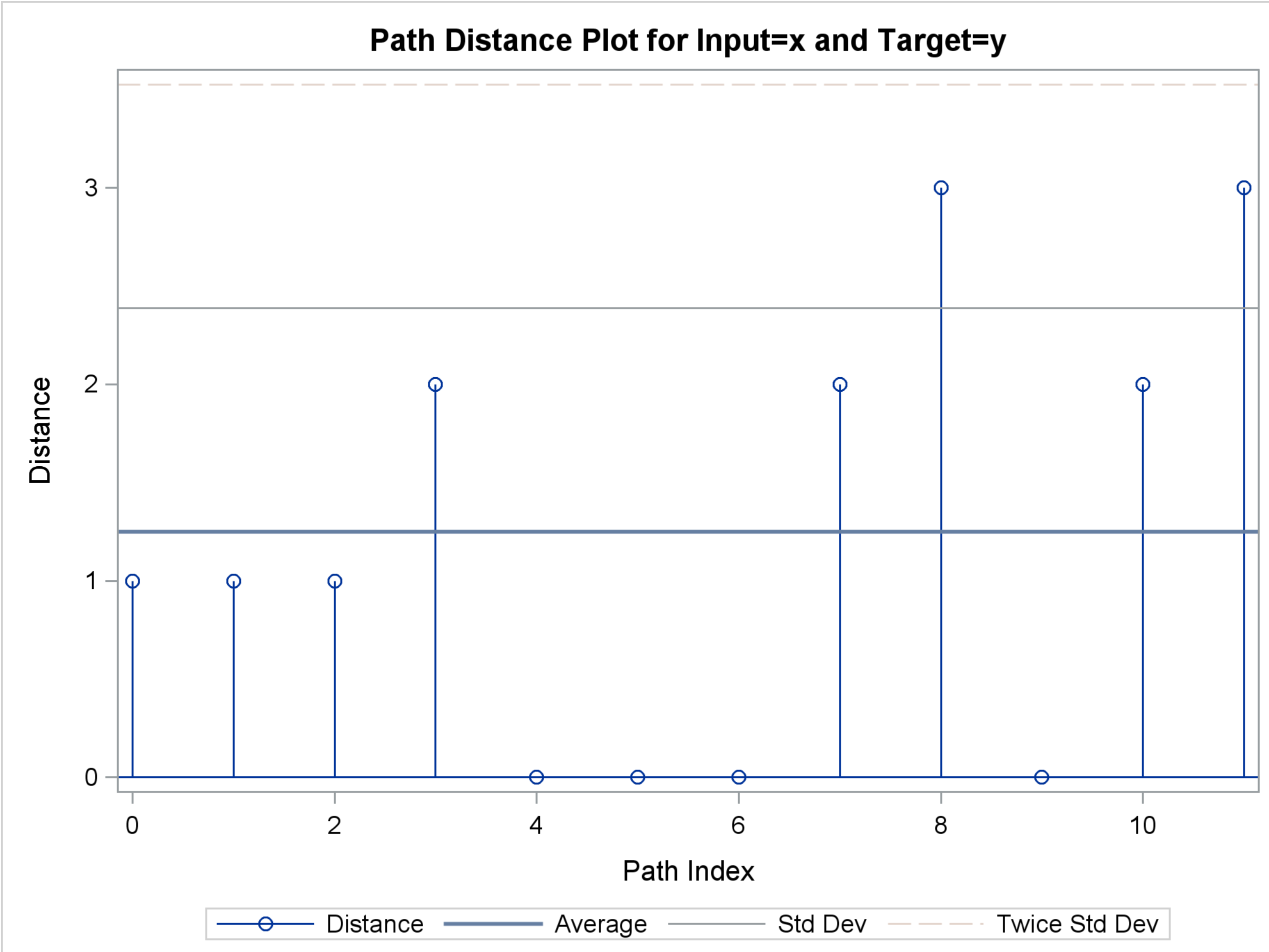
Output 30.2.6 Path Sequences Plot



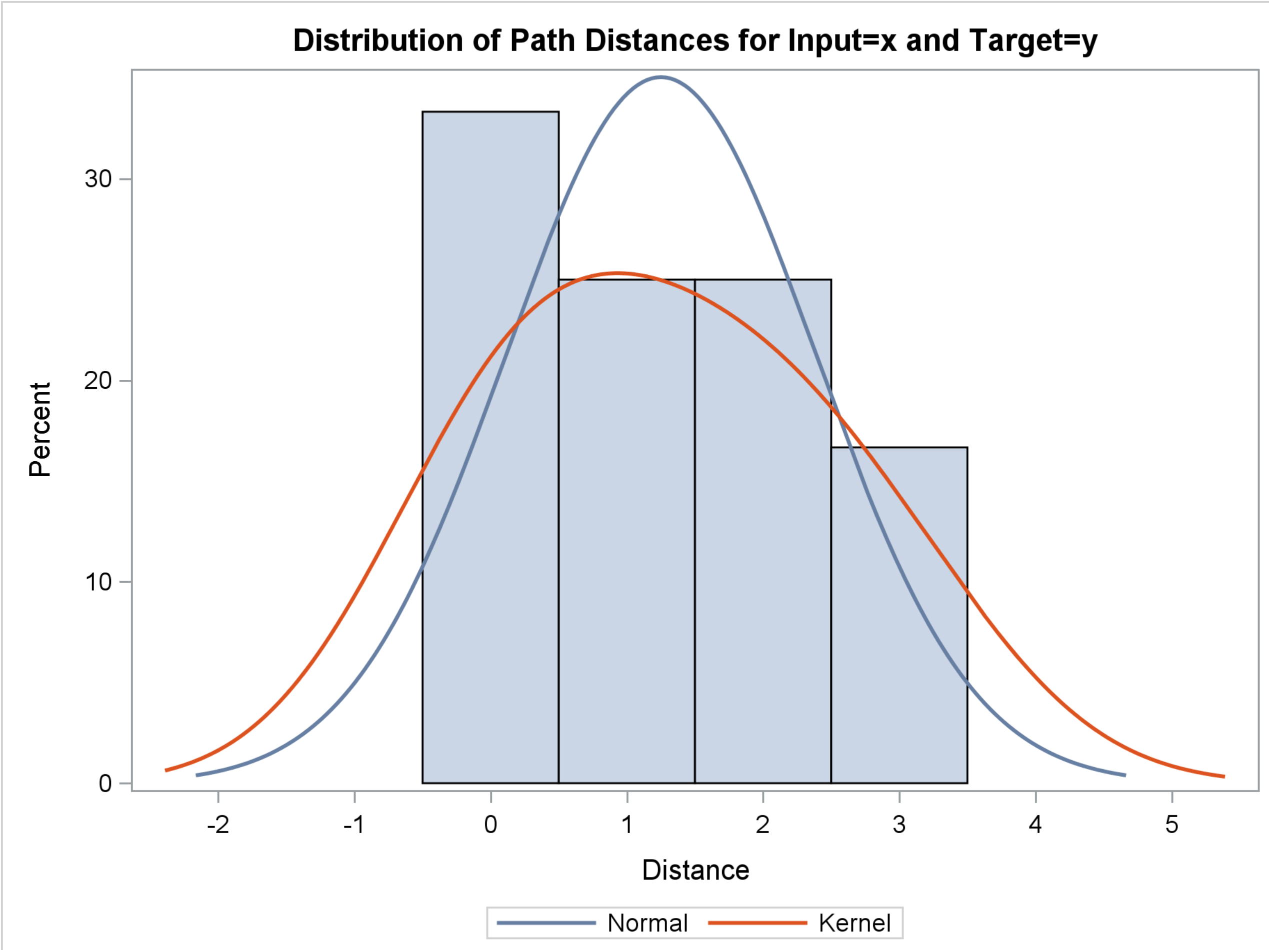
Output 30.2.7 Path Sequences Scaled Plot



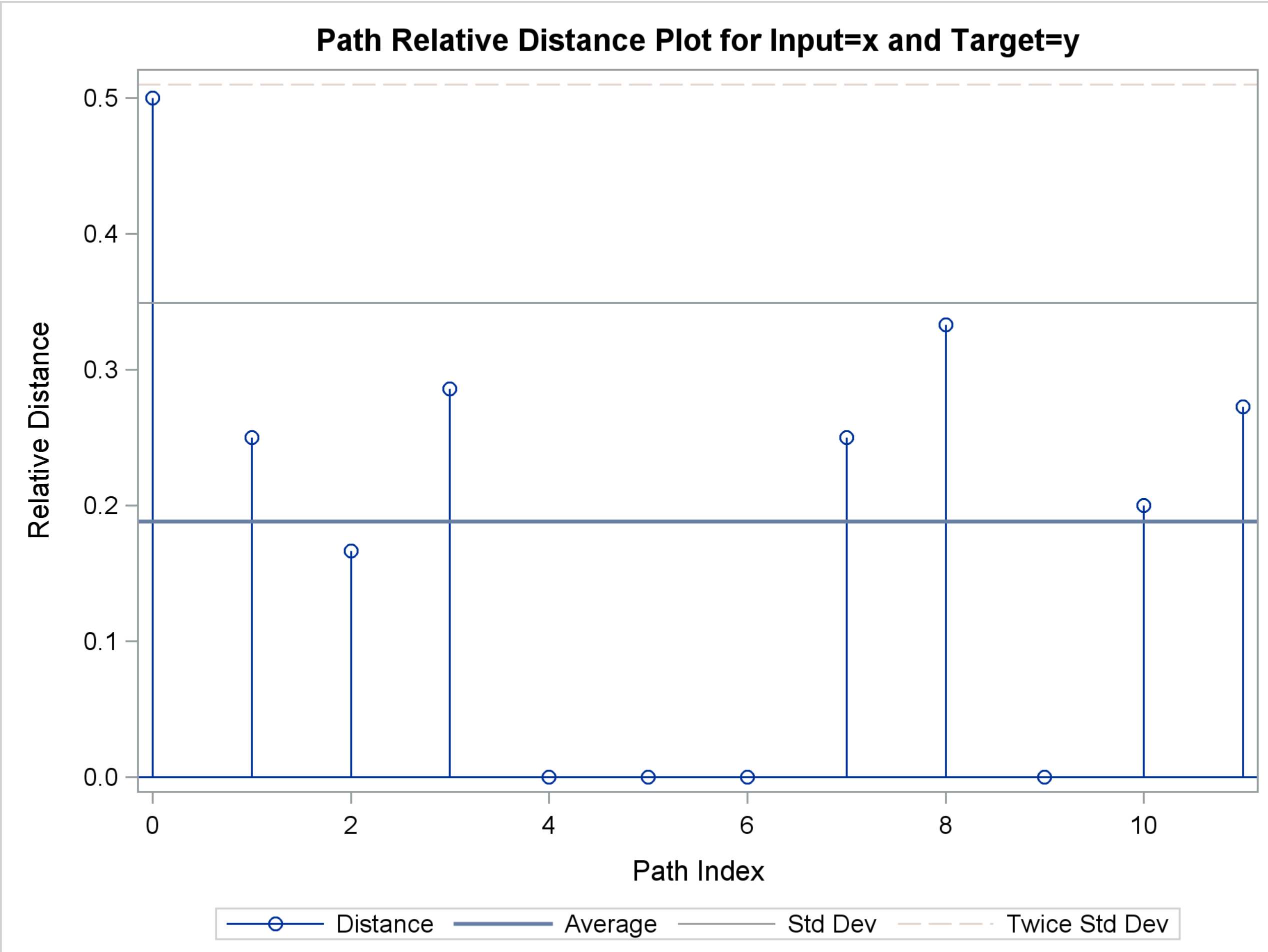
Output 30.2.8 Path Distance Plot



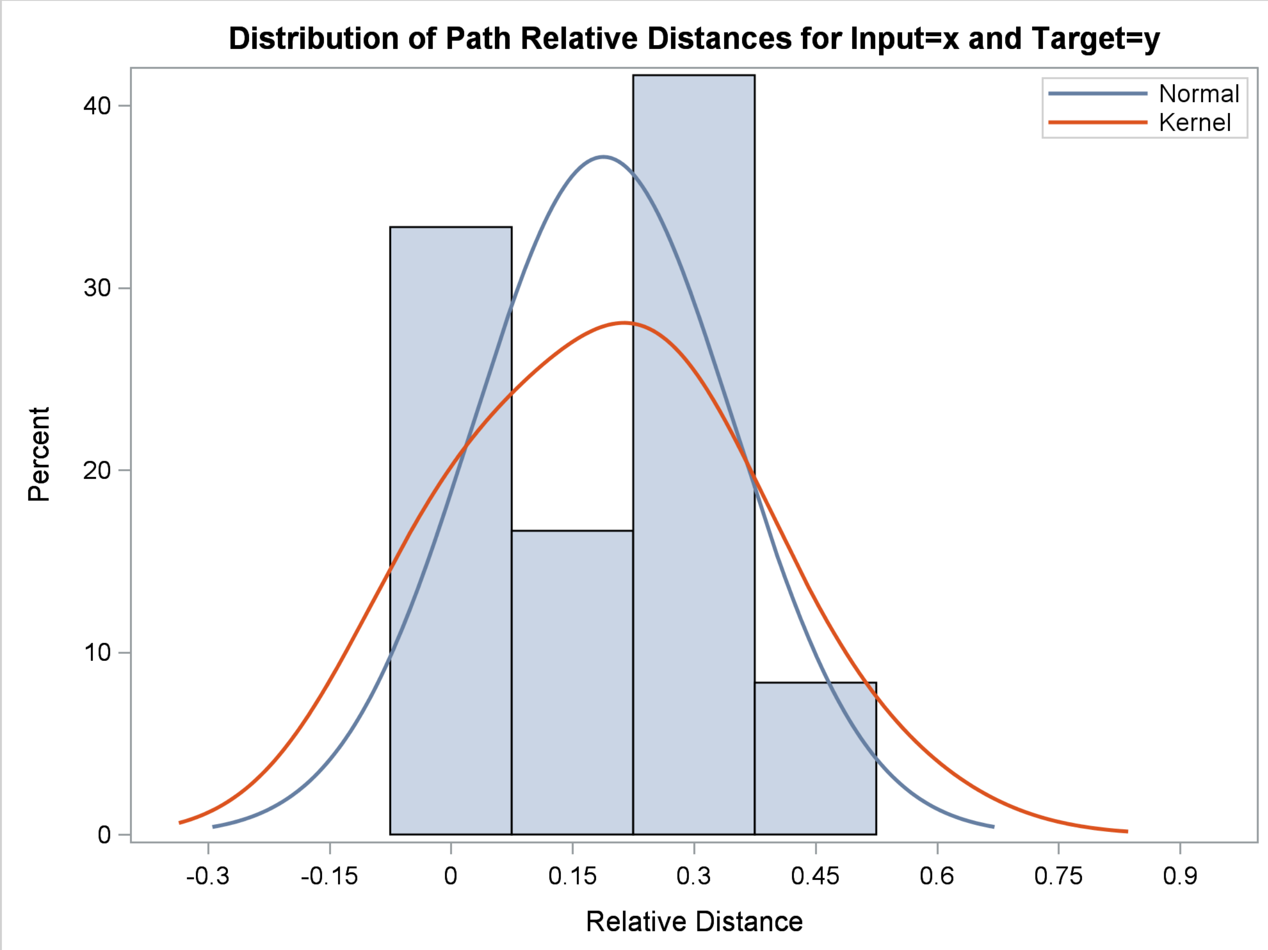
Output 30.2.9 Path Distance Histogram



Output 30.2.10 Path Relative Distance Plot



Output 30.2.11 Path Relative Distance Histogram



Output 30.2.12 Path Limits

Path Limits

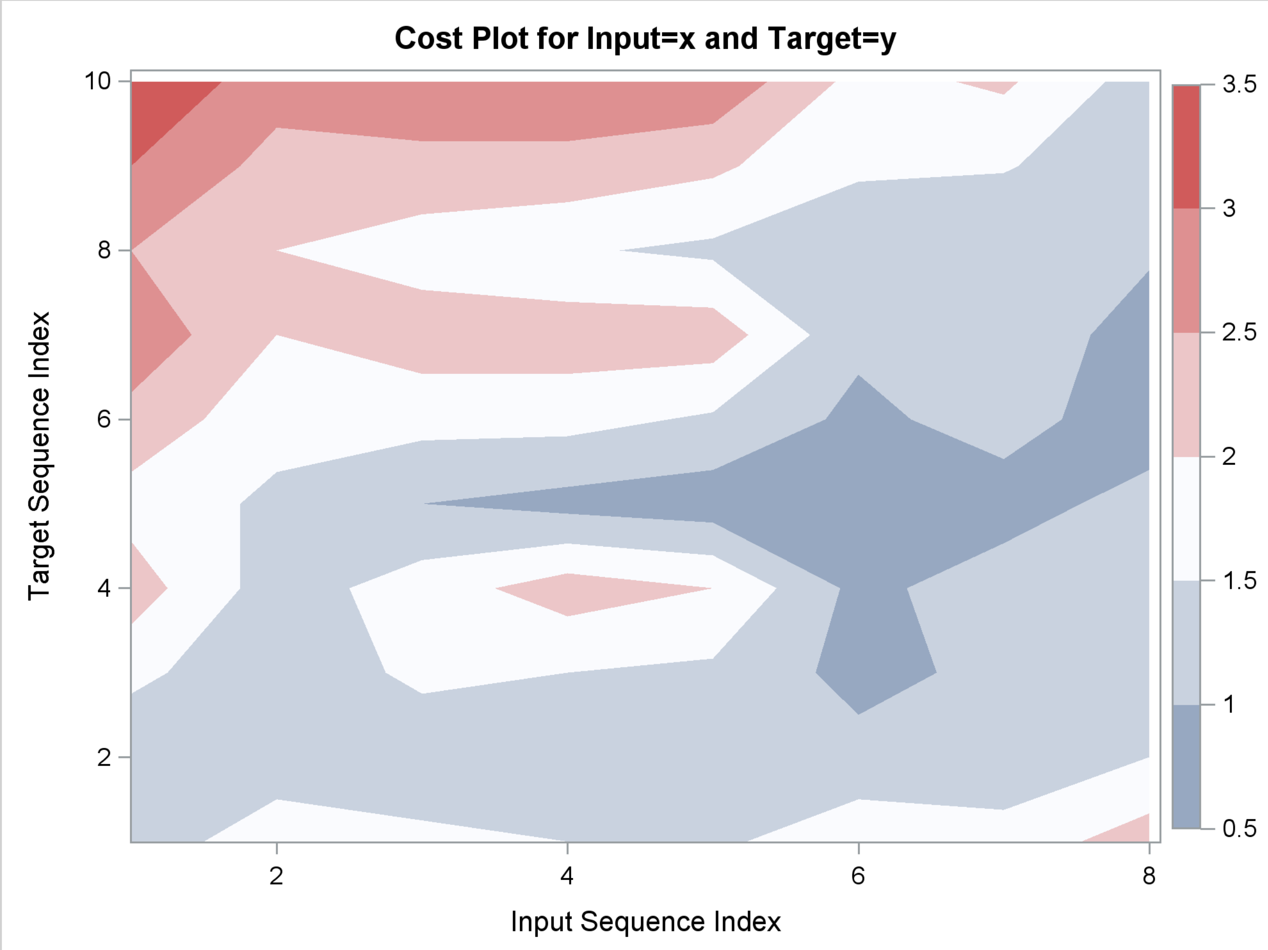
Limit	Specified Absolute	Specified Percentage	Minimum Allowed	Maximum Allowed	Applied
Compression	None	None	2	9	9
Expansion	None	None	0	7	7

Output 30.2.13 Path Statistics

Path Statistics

Path	Number	Path Percent	Input Percent	Target Percent	Maximum	Path Maximum Percent	Input Maximum Percent	Target Maximum Percent
Missing Map	0	0.000%	0.000%	0.000%	0	0.000%	0.000%	0.000%
Direct Maps	6	50.00%	75.00%	60.00%	2	16.67%	25.00%	20.00%
Compression	4	33.33%	50.00%	40.00%	1	8.333%	12.50%	10.00%
Expansion	2	16.67%	25.00%	20.00%	2	16.67%	25.00%	20.00%
Warps	6	50.00%	75.00%	60.00%	2	16.67%	25.00%	20.00%

Output 30.2.14 Cost Plot



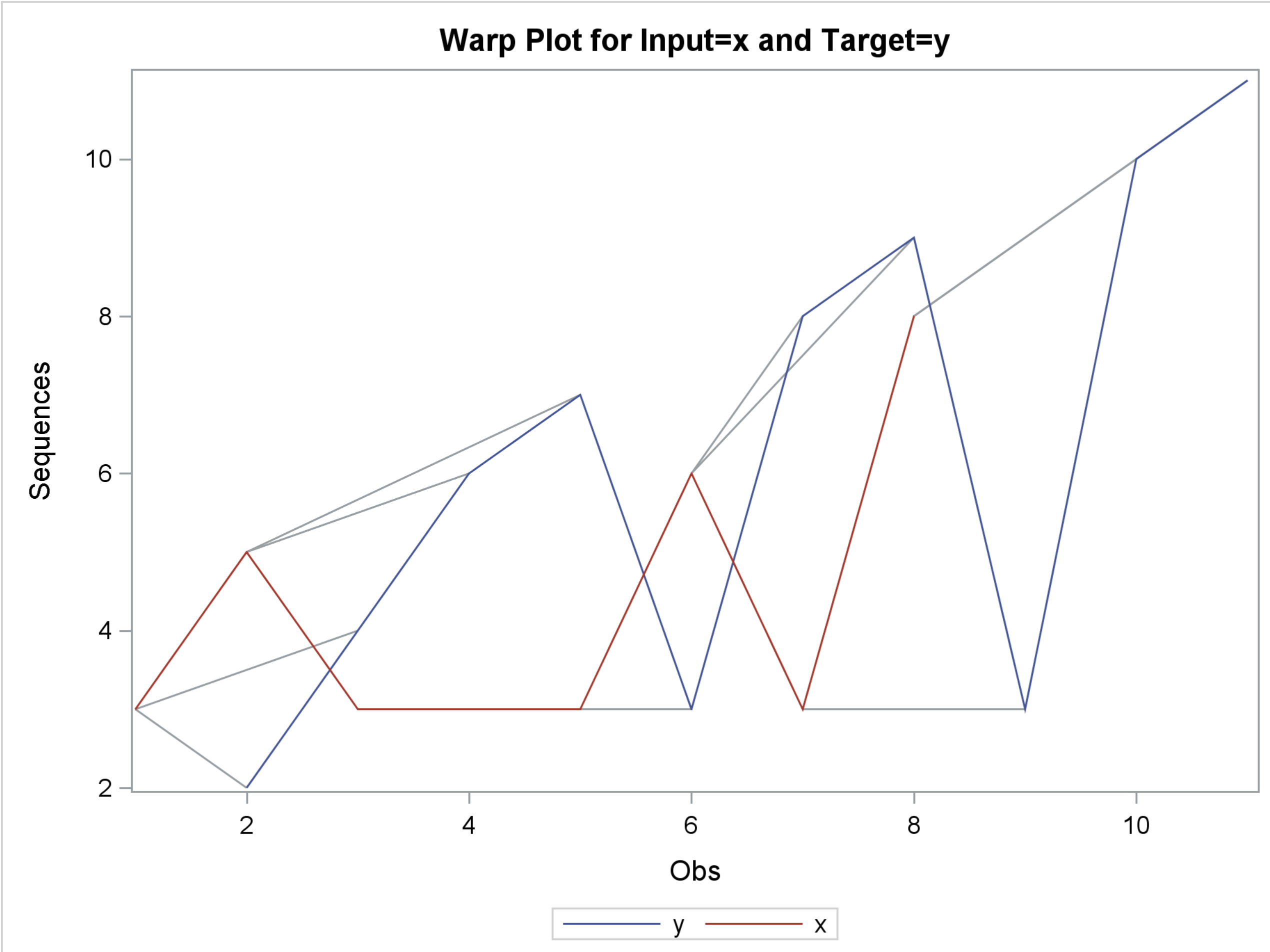
Output 30.2.15 Cost Statistics

Cost Statistics

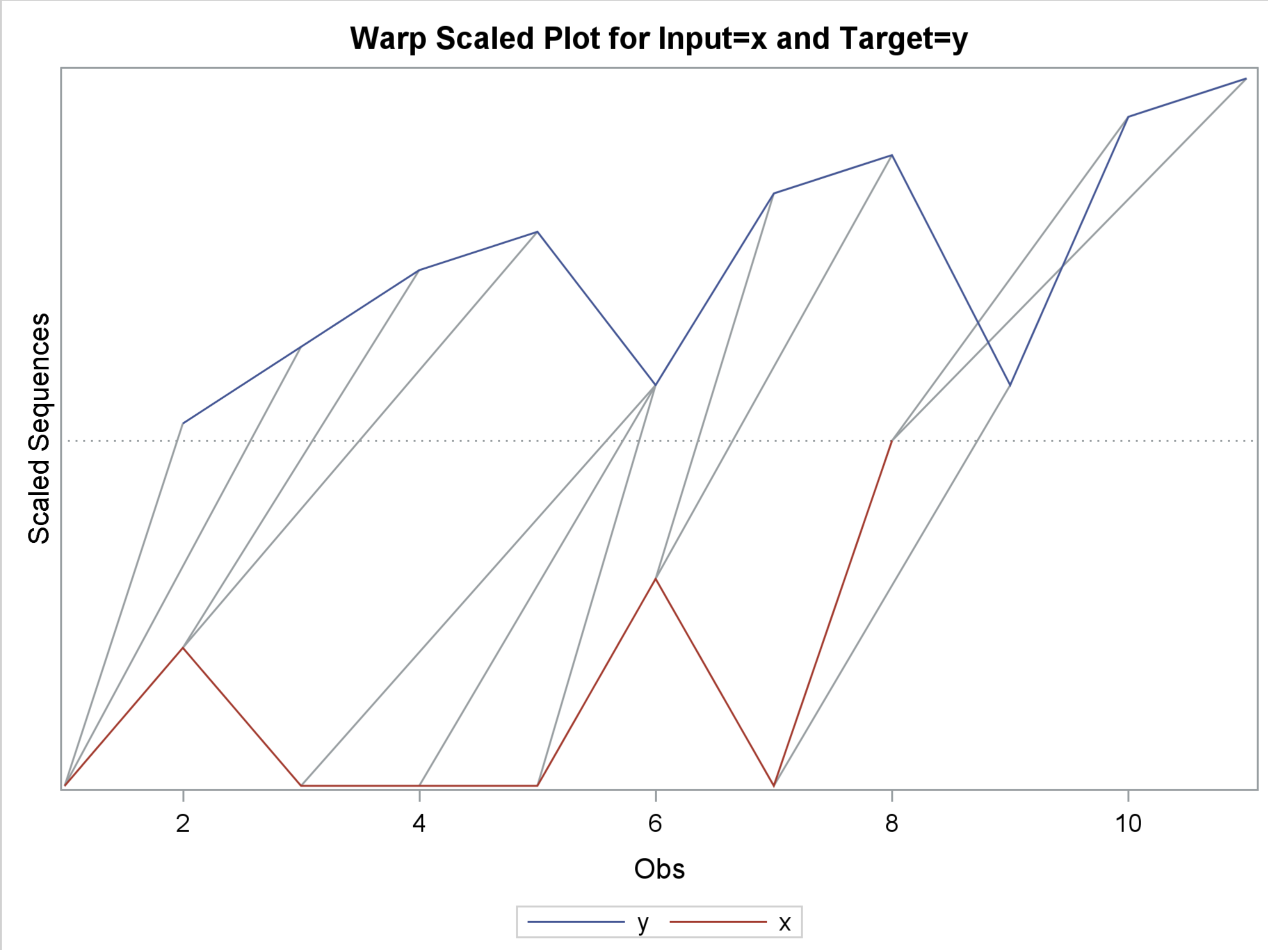
Cost	Number	Total	Average	Standard Deviation	Minimum	Maximum	Input Mean	Target Mean	Minimum Path Mean	Maximum Path Mean
Absolute	12	15.00000	1.250000	1.138180	0	3.000000	1.875000	1.500000	1.875000	0.8823529
Relative	12	2.25844	0.188203	0.160922	0	0.500000	0.282305	0.225844	0.282305	0.1328495

Relative Costs based on Target Sequence values

Output 30.2.16 Time Warp Plot



Output 30.2.17 Time Warp Scaled Plot

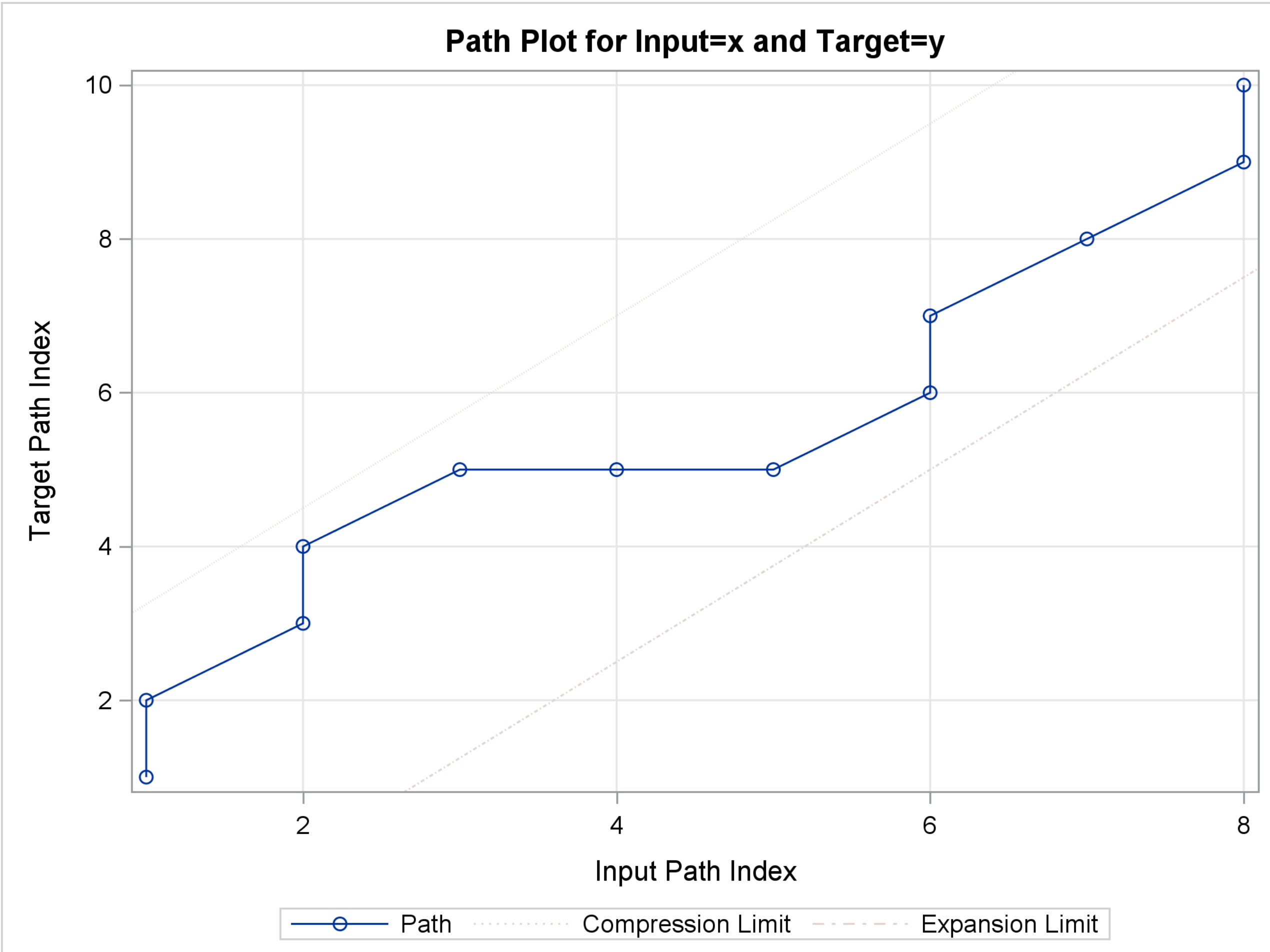


The following statements repeat the preceding similarity analysis on the example data set with warping limits:

```
proc similarity data=test out=_null_
   print=all plot=all;
   input x;
   target y / measure=absdev
              compress=(localabs=2)
              expand=(localabs=2);
run;
```

The COMPRESS=(LOCALABS=2) option limits local absolute compression to 2. The EXPAND=(LOCALABS=2) option limits local absolute expansion to 2.

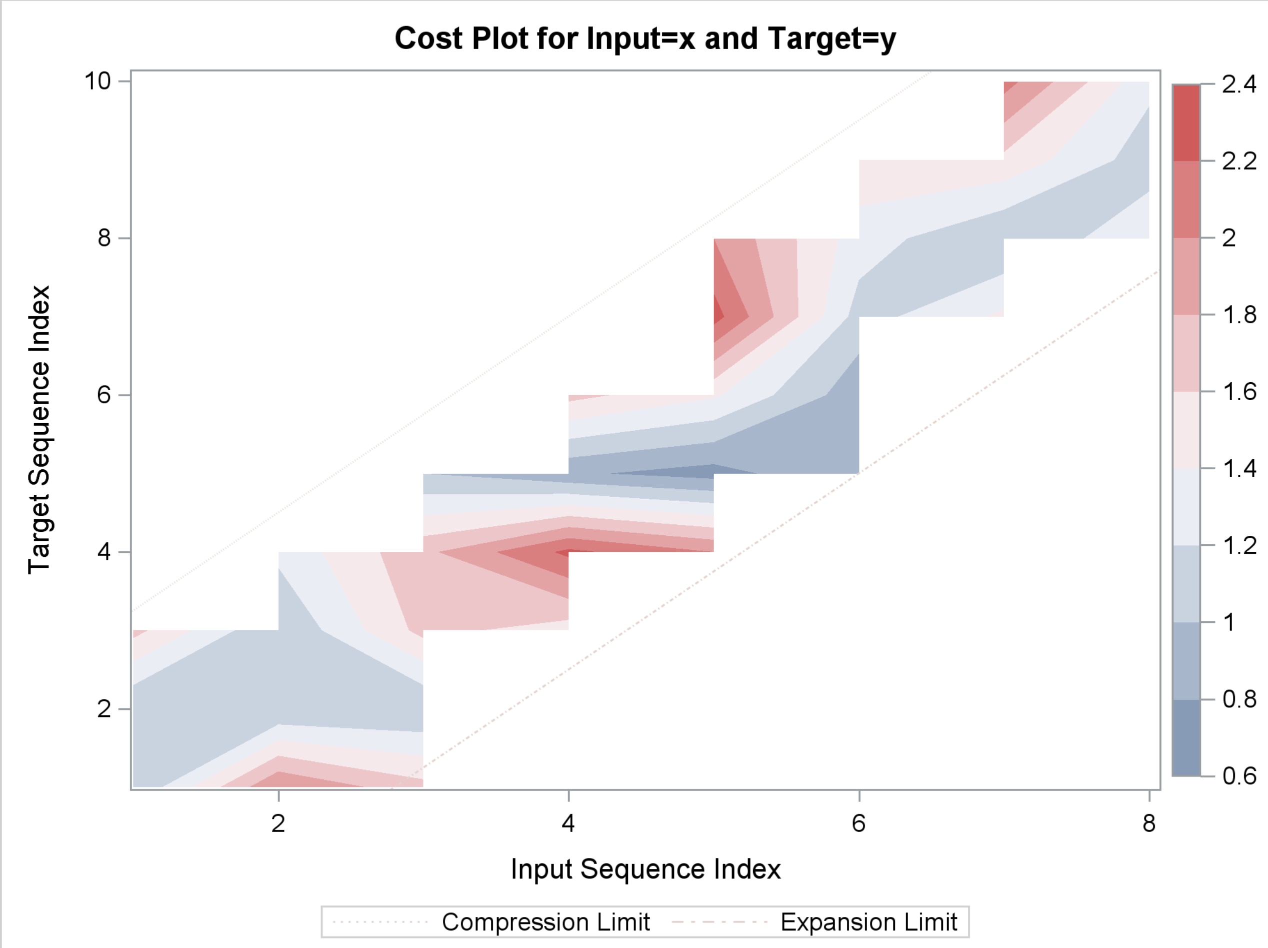
Output 30.2.18 Path Plot with Warping Limits



Output 30.2.19 Warped Path Limits

	Path Limits				
Limit	**Specified Absolute**	**Specified Percentage**	**Minimum Allowed**	**Maximum Allowed**	**Applied**
Compression	2	None	2	9	2
Expansion	2	None	0	7	2

Output 30.2.20 Cost Plot with Warping Limits



The following statements repeat the preceding similarity analysis on the example data set but store the results in output data sets:

```
proc similarity data=test out=series
   outsequence=sequences outpath=path outsum=summary;
   input x;
   target y / measure=absdev
              compress=(localabs=2)
              expand=(localabs=2);
run;
```

The OUT=SERIES, OUTSEQUENCE=SEQUENCES, OUTPATH=PATH, and OUTSUM=SUMMARY options specify that the output time series, time sequences, path analysis, and summary data sets be created, respectively.

Example 30.3: Sliding Similarity Analysis

This example illustrates how to use sliding similarity analysis to compare two time sequences. The SASHELP.WORKERS data set contains two similar time series variables (ELECTRIC and MASONRY), which represent employment over time. The following statements create an example data set that contains two time series of differing lengths, where the variable MASONRY has the first 12 and last 7 observations set to missing to simulate the lack of data associated with the target series:

```
data workers; set sashelp.workers;
   if '01JAN1978'D <= date < '01JAN1982'D then masonry = masonry;
   else masonry = .;
run;
```

The goal of sliding similarity measures analysis is find the slide index that corresponds to the most similar subsequence of the input series when compared to the target sequence. The following statements perform sliding similarity analysis on the example data set:

```
proc similarity data=workers out=_NULL_ print=(slides summary);
   id date interval=month;
   input  electric;
   target masonry / slide=index measure=msqrdev
                    expand=(localabs=3 globalabs=3)
                    compress=(localabs=3 globalabs=3);
run;
```

The DATA=WORKERS option specifies that the input data set WORK.WORKERS is to be used in the analysis. The OUT=_NULL_ option specifies that no output time series data set is to be created. The PRINT=(SLIDES SUMMARY) option specifies that the ODS tables related to the sliding similarity measures and their summary be produced. The INPUT statement specifies that the input variable is ELECTRIC. The TARGET statement specifies that the target variable is MASONRY and that the similarity measure be computed using mean squared deviation (MEASURE=MSQRDEV). The SLIDE=INDEX option specifies observation index sliding. The COMPRESS=(LOCALABS=3 GLOBALABS=3) option limits local and global absolute compression to 3. The EXPAND=(LOCALABS=3 GLOBALABS=3) option limits local and global absolute expansion to 3.

Output 30.3.1 Summary of the Slide Measures

The SIMILARITY Procedure

Slide Measures Summary for Input=ELECTRIC and Target=MASONRY

Slide Index	DATE	Slide Target Sequence Length	Slide Input Sequence Length	Slide Warping Amount	Slide Minimum Measure
0	JAN1977	48	51	3	497.6737
1	FEB1977	48	51	1	482.6777
2	MAR1977	48	51	0	474.1251
3	APR1977	48	51	0	490.7792
4	MAY1977	48	51	-2	533.0788
5	JUN1977	48	51	-3	605.8198
6	JUL1977	48	51	-3	701.7138
7	AUG1977	48	51	3	646.5918
8	SEP1977	48	51	3	616.3258
9	OCT1977	48	51	3	510.9836
10	NOV1977	48	51	3	382.1434
11	DEC1977	48	51	3	340.4702
12	JAN1978	48	51	2	327.0572
13	FEB1978	48	51	1	322.5460
14	MAR1978	48	51	0	325.2689
15	APR1978	48	51	-1	351.4161
16	MAY1978	48	51	-2	398.0490
17	JUN1978	48	50	-3	471.6931
18	JUL1978	48	49	-3	590.8089
19	AUG1978	48	48	0	595.2538
20	SEP1978	48	47	-1	689.2233
21	OCT1978	48	46	-2	745.8891
22	NOV1978	48	45	-3	679.1907

Output 30.3.2 Minimum Measure

Minimum Measure Summary

Input Variable	MASONRY
ELECTRIC	322.5460

This analysis results in 23 slides based on the observation index. The minimum measure (322.5460) occurs at slide index 13, which corresponds to the time value FEB1978. Note that the original data set SASHELP.WORKERS was modified beginning at the time value JAN1978. This similarity analysis justifies the belief that ELECTRIC lags MASONRY by one month based on the time series cross-correlation analysis despite the lack of target data (MASONRY).

The goal of seasonal sliding similarity measures is to find the seasonal slide index that corresponds to the most similar seasonal subsequence of the input series when compared to the target sequence. The following statements repeat the preceding similarity analysis on the example data set with seasonal sliding:

```
proc similarity data=workers out=_NULL_ print=(slides summary);
   id date interval=month;
   input  electric;
   target masonry / slide=season measure=msqrdev;
run;
```

Output 30.3.3 Summary of the Seasonal Slide Measures

The SIMILARITY Procedure

Slide Measures Summary for Input=ELECTRIC and Target=MASONRY

Slide Index	DATE	Slide Target Sequence Length	Slide Input Sequence Length	Slide Warping Amount	Slide Minimum Measure
0	JAN1977	48	48	0	1040.086
12	JAN1978	48	48	0	641.927

Output 30.3.4 Seasonal Minimum Measure

Minimum Measure Summary

Input Variable	MASONRY
ELECTRIC	641.9273

The analysis differs from the previous analysis in that the slides are performed based on the seasonal index (SLIDE=SEASON) with no warping. With a seasonality of 12, two seasonal slides are considered at slide indices 0 and 12 with the minimum measure (641.9273) occurring at slide index 12 which corresponds to the time value JAN1978. Note that the original data set SASHELP.WORKERS was modified beginning at the time value JAN1978. This similarity analysis justifies the belief that ELECTRIC and MASONRY have similar seasonal properties based on seasonal decomposition analysis despite the lack of target data (MASONRY).

Example 30.4: Searching for Historical Analogies

This example illustrates how to search for historical analogies by using seasonal sliding similarity analysis of transactional time-stamped data. The SASHELP.TIMEDATA data set contains the variable (VOLUME), which represents activity over time. The following statements create an example data set that contains two time series of differing lengths, where the variable HISTORY represents the historical activity and RECENT represents the more recent activity:

```
data timedata; set sashelp.timedata;
   drop volume;
   recent  = .;
   history = volume;
   if datetime >= '20AUG2000:00:00:00'DT then do;
      recent  = volume;
      history = .;
   end;
run;
```

The goal of seasonal sliding similarity measures is to find the seasonal slide index that corresponds to the most similar seasonal subsequence of the input series when compared to the target sequence. The following statements perform similarity analysis on the example data set with seasonal sliding:

```
proc similarity data=timedata out=_NULL_ outsequence=sequences
                outsum=summary;
   id datetime interval=dtday accumulate=total
               start='27JUL1997:00:00:00'dt
               end='21OCT2000:11:59:59'DT;
   input history / normalize=absolute;
   target recent / slide=season normalize=absolute measure=mabsdev;
run;
```

The DATA=TIMEDATA option specifies that the input data set WORK.TIMEDATA be used in the analysis. The OUT=_NULL_ option specifies that no output time series data set is to be created. The OUTSEQUENCE=SEQUENCES and OUTSUM=SUMMARY options specify the output sequences and summary data sets, respectively. The ID statement specifies that the time ID variable is DATETIME, which is to be accumulated on a daily basis (INTERVAL=DTDAY) by summing the transactions (ACCUMULATE=TOTAL). The ID statement also specifies that the data are accumulated on the weekly boundaries starting on the week of 27JUL1997 and ending on the week of 15OCT2000 (START='27JUL1997:00:00:00'DT END='21OCT2000:11:59:59'DT). The INPUT statement specifies that the input variable is HISTORY, which is to be normalized using absolute normalization (NORMALIZE=ABSOLUTE). The TARGET statement specifies that the target variable is RECENT, which is to be normalized by using absolute normalization (NORMALIZE=ABSOLUTE) and that the similarity measure be computed by using mean absolute deviation (MEASURE=MABSDEV). The SLIDE=SEASON options specifies season index sliding.

To illustrate the results of the similarity analysis, the output sequence data set must be subset by using the output summary data set.

```
data _NULL_; set summary;
   call symput('MEASURE', left(trim(putn(recent,'BEST20.'))));
run;

data result; set sequences;
```

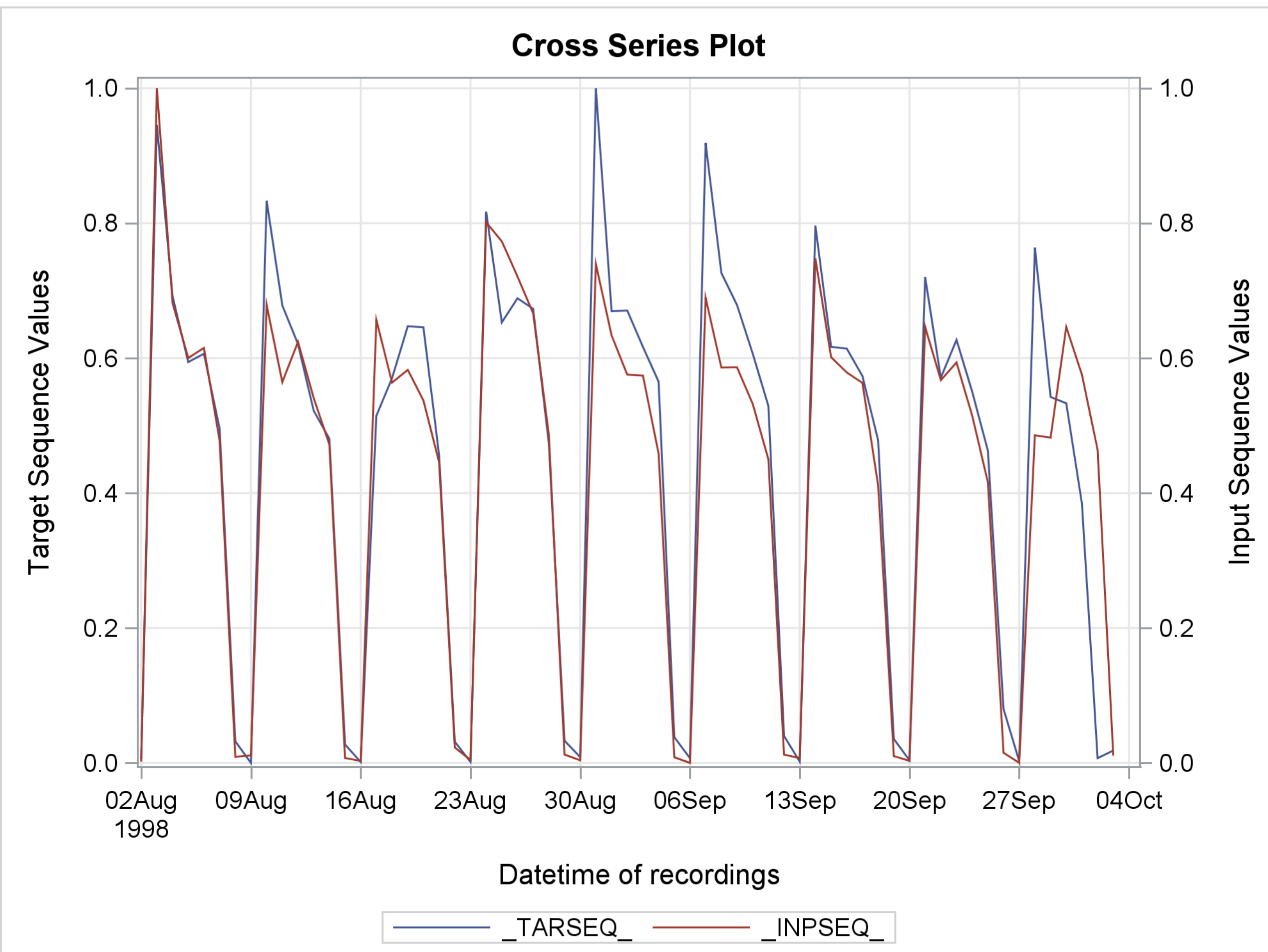
```
      by _SLIDE_;
      retain flag 0;
      if first._SLIDE_  then do;
         if (&measure - 0.00001 < _SIM_ < &measure + 0.00001)
         then flag = 1;
      end;
      if flag then output;
      if last._SLIDE_ then flag = 0;
   run;
```

The following statements generate a cross series plot of the results:

```
   proc timeseries data=result out=_NULL_ crossplot=series;
      id datetime interval=dtday;
      var _TARSEQ_;
      crossvar _INPSEQ_;
   run;
```

The cross series plot illustrates that the historical time series analogy most similar to the most recent time series data that started on 20AUG2000 occurred on 02AUG1998.

Output 30.4.1 Cross Series Plot of the Historical Time Series



Example 30.5: Clustering Time Series

This example illustrates how to cluster time series using a similarity matrix. The WORK.APPLIANCE data set contains 24 variables that record sales histories. The following statements create a similarity matrix and store the matrix in the WORK.SIMMATRIX data set:

```
proc similarity data=sashelp.applianc out=_null_ outsum=simmatrix;
   target units_1--units_24 / measure=mabsdev normalize=absolute;
run;
```

The following statements cluster the rows of the similarity matrix:

```
proc cluster data=simmatrix(drop=_status_) outtree=tree method=ward noprint;
   id _input_;
run;
```

The following statements plot the dendrogram:

```
proc tree data=tree horizontal;
run;
```

References

Barry, M. J., and Linoff, G. S. (1997). *Data Mining Techniques: For Marketing, Sales, and Customer Support*. New York: John Wiley & Sons.

Han, J., and Kamber, M. (2001). *Data Mining: Concepts and Techniques*. San Francisco: Morgan Kaufmann.

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

Leonard, M. J., and Wolfe, B. L. (2005). "Mining Transactional and Time Series Data." In *Proceedings of the Thirtieth Annual SAS Users Group International Conference*. Cary, NC: SAS Institute Inc. http://www2.sas.com/proceedings/sugi30/080-30.pdf.

Pyle, D. (1999). *Data Preparation for Data Mining*. San Francisco: Morgan Kaufmann.

Sankoff, D., and Kruskal, J. B. (2001). *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*. Stanford, CA: CSLI Publications.

Subject Index

Syntax Index