THE POWER TO KNOW.

# SAS/ETS® 15.1 User's Guide

# The TIMEDATA Procedure

This document is an individual chapter from *SAS/ETS® 15.1 User's Guide*.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2018. *SAS/ETS® 15.1 User's Guide*. Cary, NC: SAS Institute Inc.

**SAS/ETS® 15.1 User's Guide**



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

November 2018

# Chapter 36
# The TIMEDATA Procedure

**Contents**

# Overview: TIMEDATA Procedure

The TIMEDATA procedure analyzes time-stamped transactional data with respect to time and accumulates the data into a time series format.

After the transactional data are accumulated to form a time series and any missing values are interpreted, the accumulated time series can be functionally transformed using log, square root, logistic, or Box-Cox transformations. The time series can be further transformed using simple differencing, seasonal differencing, or both. After functional and difference transformations have been applied, the accumulated and transformed time series can be stored in an output data set. This working time series can then be analyzed further using various time series analysis techniques provided by this procedure or other SAS/ETS procedures.

The TIMEDATA procedure is very similar to the TIMESERIES procedure. However, unlike the TIMESERIES procedure (which enables you to perform a variety of standard time series analysis techniques), the TIMEDATA procedure enables you to define your own analyses using SAS programming statements.

By default, the TIMEDATA procedure provides no further analyses.

The TIMEDATA procedure forms time series vectors and then provides these vectors as SAS data arrays for subsequent processing by your SAS programming statements. Your programming statements are processed independently for each BY group. The TIMEDATA procedure is like the SAS DATA step for time series data. The SAS DATA step processes data by each row; the TIMEDATA procedure processes time series vectors.

As part of your SAS programming statements, you can include user-defined functions and subroutines created by the FCMP procedure. Additionally, you can use the RUN_MACRO subroutine provided by the FCMP procedure to submit SAS statements that use any SAS procedures.

All results of the transactional or time series analysis can be stored in output data sets or printed using the Output Delivery System (ODS).

# Getting Started: TIMEDATA Procedure

This section outlines the use of the TIMEDATA procedure and gives a cursory description of some of the analysis techniques that you can perform on time-stamped transactional data.

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

```
PROC TIMEDATA DATA=<input-data-set>
              OUT=<output-data-set>;
   BY <list-of-BY-variables>;
   ID <time-ID-variable> INTERVAL=<frequency>
                         ACCUMULATE=<statistic>;
   VAR <time-series-variables>;
   /* programming statements */
RUN;
```

The TIMEDATA procedure forms time series from the input time-stamped transactional data. It can provide results in output data sets or in other output formats by using the Output Delivery System (ODS).

Time-stamped transactional data are 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, such as daily, weekly, or monthly.

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 four variables that are related to these transactions: Customer, Date, Withdrawals, and Deposits. The following examples illustrate possible ways to analyze these transactions by using the TIMEDATA procedure.

The following TIMEDATA procedure statements accumulate the time-stamped transactional data to form a daily time series based on the accumulated daily totals of each type of transaction (Withdrawals and Deposits):

```
proc timedata data=transactions
              out=timeseries
              outarray=arrays;
   by customer;
   id date interval=day accumulate=total;
   var withdrawals deposits;
   outarrays balance;

   balance[1] = deposits[1] - withdrawals[1];
   do t = 2 to _LENGTH_;
      balance[t] = balance[t-1] + (deposits[t] - withdrawals[t]);
   end;

run;
```

The OUT=TIMESERIES option specifies that the resulting time series data for each customer are to be stored in the data set Work.Transactions. The OUTARRAY=ARRAYS option specifies that the resulting time series data along with a newly created variable, Balance, are to be stored in the data set Work.Arrays. 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 is to be calculated. After the transactional data are accumulated into a time series format, many of the procedures provided with SAS/ETS software can be used to analyze the resulting time series data.

For example, the following statements use the ARIMA procedure to model and forecast each customer's balance data by using an ARIMA(1,0,0)$(0,1,0)_s$ model (where the number of seasons is s=7 days in a week):

```
proc arima data=arrays;
   by customer;
   identify var=balance(7) noprint;
   estimate p=(1) outest=estimates noprint;
   forecast id=date interval=day out=forecasts;
quit;
```

The OUTEST=ESTIMATES data set contains the parameter estimates of the model specified. The OUT=FORECASTS data set contains forecasts based on the model specified. For more information, see Chapter 7, "The ARIMA Procedure."

By default, the TIMEDATA procedure produces no printed output.

# Syntax: TIMEDATA Procedure

The following statements are available in the TIMEDATA procedure:

**PROC TIMEDATA** *options* ;
  **BY** *variables* ;
  **ID** *variable* **INTERVAL=** *interval-option* ;
  **FCMPOPT** *options* ;
  **OUTARRAYS** *array-name-list* ;
  **OUTSCALARS** *scalar-name-list* ;
  **VAR** *variable-list / options* ;
  **REGISTER** *package* ;
  **Programming Statements** ;

## Functional Summary

Table 36.1 summarizes the statements and options that control the TIMEDATA procedure.

**Table 36.1** Functional Summary

| Description | Statement | Option |
|---|---|---|
| **Statements** | | |
| Specifies BY-group processing | BY | |
| Specifies variables to analyze | VAR | |
| Specifies the time ID variable | ID | |
| Specifies the FCMP options | FCMPOPT | |
| Specifies the arrays to output | OUTARRAYS | |
| Specifies the scalars to output | OUTSCALARS | |
| Specifies the packages to include | REGISTER | |
| **Data Set Options** | | |
| Specifies the auxiliary input data sets | PROC TIMEDATA | AUXDATA= |
| Specifies the input data set | PROC TIMEDATA | DATA= |
| Specifies the output data set | PROC TIMEDATA | OUT= |
| Specifies the array output data set | PROC TIMEDATA | OUTARRAY= |
| Specifies the run status data set | PROC TIMEDATA | OUTPROCINFO= |
| Specifies the scalar output data set | PROC TIMEDATA | OUTSCALAR= |
| Specifies the summary statistics output data set | PROC TIMEDATA | OUTSUM= |

**Table 36.1** *continued*

| Description | Statement | Option |
|---|---|---|
| **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 TIMEDATA | SEASONALITY= |
| Specifies the type of life-cycle indexing | PROC TIMEDATA | CYCLETYPE= |
| Specifies the interval alignment | ID | ALIGN= |
| Specifies that time ID variable values not be sorted | ID | NOTSORTED |
| Specifies the starting time ID value | ID | START= |
| Specifies the ending time ID value | ID | END= |
| Specifies the accumulation statistic | ID, VAR | ACCUMULATE= |
| Specifies missing value interpretation | ID, VAR | SETMISSING= |
| Specifies the zero value interpretation | ID, VAR | ZEROMISS= |
| | | |
| **Time Series Transformation Options** | | |
| Specifies simple differencing | VAR | DIF= |
| Specifies seasonal differencing | VAR | SDIF= |
| Specifies transformation | VAR | TRANSFORM= |
| | | |
| **Printing Control Options** | | |
| Specifies the time ID format | ID | FORMAT= |
| Specifies which output to print | PROC TIMEDATA | PRINT= |
| | | |
| **Miscellaneous Options** | | |
| Specifies the forecast horizon or lead used to extend the data set | PROC TIMEDATA | LEAD= |
| Limits error and warning messages | PROC TIMEDATA | MAXERROR= |
| | | |
| **ODS Graphics Options** | | |
| Specifies the variable and array graphical output | PROC TIMEDATA | PLOTS= |

## PROC TIMEDATA Statement

**PROC TIMEDATA** *options* **;**

The following *options* can be used in the PROC TIMEDATA statement:

**AUXDATA=***SAS-data-set*

names a SAS data set that contains auxiliary input data for the procedure to use for supplying time series variables. For more information, see the section "Auxiliary Data Sets" on page 2673.

**CYCLETYPE=***option*

specifies the indexing of each time series with respect to life-cycle. By default, CYCLETYPE=BOL.

The following CYCLETYPE= *options* are available:

**BOL** indexes the time series by the beginning of life. The first time value is 1. The following values are incremented by 1.

**MOL** indexes the time series by the middle of life. The middle time value is zero. The preceding values are decremented by 1. The following values are incremented by 1.

**EOL** indexes the time series by the end of life. The last time value is 1. The preceding values are incremented by 1.

The CYCLETYPE= option specifies the indexing of the _CYCLE_ variable contained in the OUTARRAY= data set and the predefined array _CYCLE_.

**DATA=***SAS-data-set*

names the SAS data set that contains the input data from which the procedure creates the time series. If the DATA= option is not specified, the most recently created SAS data set is used.

**LEAD=***n*

specifies the number of periods ahead to forecast (forecast lead or horizon) used to extend the data set. The default is LEAD=0.

The LEAD= value is relative to the last observation in the input data set and not to the last nonmissing observation of a particular series.

**MAXERROR=***number*

limits the number of warning and error messages that are produced during the execution of the procedure to the specified *number*. The default is MAXERRORS=50. This option is particularly useful in BY-group processing where it can be used to suppress recurring messages.

**OUT=***SAS-data-set*

names the output data set to contain the time series variables specified in the subsequent VAR statements. If BY variables are specified, they are also included in the OUT= data set. If an ID variable is specified, it is also included in the OUT= data set. The values are accumulated based on the INTERVAL= option or the ACCUMULATE= option or both in the ID statement. 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.

**OUTARRAY=***SAS-data-set*

names the output data set to contain the time series vectors listed in the VAR and OUTARRAYS statements.

The OUTARRAY= data set contains the variables specified in the BY, ID, and VAR statements in addition to the arrays that are specified in the OUTARRAYS statements.

**OUTSCALAR=***SAS-data-set*

names the output data set to contain the scalar names listed in the OUTSCALARS statements.

The OUTSCALAR= data set contains the variables specified in the BY statement and the scalars that are specified in the OUTSCALARS statements.

**OUTPROCINFO=***SAS-data-set*

names the output data set to summarize information in the SAS log, specifically the number of notes, errors, and warnings and the number of series processed, analyses requested, and analyses failed.

**OUTSUM=***SAS-data-set*

names the output data set to contain the descriptive statistics. The descriptive statistics are based on the accumulated time series when the ACCUMULATE= option, the SETMISSING= option, or both are specified in the ID or VAR statements. The OUTSUM= data set is particularly useful when analyzing large numbers of series and a summary of the results is needed.

**PLOTS=***option* **| (** *options* **)**

specifies the univariate graphical output desired. By default, the TIMEDATA procedure produces no graphical output. The PLOTS= option produces results that are similar to the data sets shown in parentheses next to the following *options*:

| | |
|---|---|
| **ARRAYS** | plots the time series (OUT= data set). |
| **ALL** | same as PLOTS=(ARRAYS). |

For example, PLOTS=ARRAYS plots the time series. The PLOTS= option produces graphical output for these results by using the Output Delivery System (ODS).

**PRINT=***option* **| (** *options* **)**

specifies the printed output desired. By default, the TIMEDATA procedure produces no printed output. The PRINT= option produces results that are similar to the data sets shown in parentheses next to the following *options*:

| | |
|---|---|
| **ARRAYS** | prints the arrays table (OUTARRAY= data set). |
| **SCALARS** | prints the scalars table (OUTSCALAR= data set). |
| **SUMMARY** | prints the descriptive statistics table for all time series (OUTSUM= data set). |
| **ALL** | same as PRINT=(ARRAYS SCALARS SUMMARY). |

For example, PRINT=SCALARS prints the scalars specified in the OUTSCALARS statement. The PRINT= option produces printed output for these results by using the Output Delivery System (ODS).

**SEASONALITY=***number*
specifies the length of the seasonal cycle. 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.

## BY Statement

You can include a BY statement with PROC TIMEDATA 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 TIMEDATA 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 statement, 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 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 information specified affects all variables listed in subsequent VAR statements. If the ID statement is specified, the INTERVAL= must also be used. If an ID statement is not specified, the observation number, with respect to the BY group, is used as the time ID.

You can specify the following *options* in the ID statement:

**ACCUMULATE=***option*

specifies how the data set observations are to be 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 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. Observations are accumulated based on the following: |
| **TOTAL** | total sum of their values |
| **AVERAGE \| AVG** | average of their values |
| **MINIMUM \| MIN** | minimum of their values |
| **MEDIAN \| MED** | median of their values |
| **MAXIMUM \| MAX** | maximum of their values |
| **N** | number of nonmissing observations |
| **NMISS** | number of missing observations |
| **NOBS** | number of observations |
| **FIRST** | first of their values |
| **LAST** | last of their values |
| **STDDEV \| STD** | standard deviation of their values |
| **CSS** | corrected sum of squares of their values |
| **USS** | 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 to be treated. If missing values should be interpreted as zero, then SETMISSING=0 should be used. For more information about accumulation, see the section “Details: TIMEDATA Procedure” on page 2669.

**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. BEGINNING is the default.

**END=***option*
specifies a SAS date or datetime 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. You can specify the START= and END= options to ensure that the 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 inferred from the INTERVAL= option.

**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 in the PROC TIMEDATA statement, 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. The INTERVAL= option is required and must be specified in the ID statement.

**NOTSORTED**
specifies that the time ID values not be in sorted order. The TIMEDATA procedure sorts the data with respect to the time ID prior to analysis.

**SETMISSING=***option* | *number*
specifies how missing values (either actual or accumulated) are to be interpreted in the accumulated time series. If a *number* is specified, missing values are set to the *number*. If a missing value indicates an unknown value, specify SETMISSING=MISSING. If a missing value indicates a zero value, specify SETMISSING=0. You would typically use SETMISSING=0 for transactional data because no recorded data usually implies no activity. You can use the following *options* to determine how missing values are assigned. Missing values are set as follows:

| | |
|---|---|
| **MISSING** | a missing value. This is the default. |
| **AVERAGE \| AVG** | the accumulated average value |
| **MINIMUM \| MIN** | the accumulated minimum value |
| **MEDIAN \| MED** | the accumulated median value |
| **MAXIMUM \| MAX** | the accumulated maximum value |

| | |
|---|---|
| **FIRST** | the accumulated first nonmissing value |
| **LAST** | the accumulated last nonmissing value |
| **PREVIOUS \| PREV** | the previous period's accumulated nonmissing value. Missing values at the beginning of the accumulated series remain missing. |
| **NEXT** | the next period's accumulated nonmissing value. Missing values at the end of the accumulated series remain missing. |

**START=***option*

specifies a SAS date or datetime 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 at the beginning of the series. If the first time ID variable value is less than the START= value, the series is truncated. You can specify the START= and END= options to ensure that data 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.

## OUTARRAYS Statements

**OUTARRAYS** *array-name-list* ;

Each array name listed in an OUTARRAYS statement specifies a numeric output array variable to be stored in the OUTARRAY= data set. You can include any number of OUTARRAYS statements.

Your programming statements can create and use any number of arrays. Only arrays that are listed in the OUTARRAYS statement are predefined and included in your output. The arrays are initialized to missing values.

## OUTSCALARS Statements

**OUTSCALARS** *scalar-name-list* ;

Each scalar name listed in an OUTSCALARS statement specifies a numeric output scalar variable to be stored in the OUTSCALAR= data set. You can include any number of OUTSCALARS statements.

Your programming statements can create and use any number of scalars. Only scalars that are listed in the OUTSCALARS statement are predefined and included in your output. The scalars are initialized to missing values.

## VAR Statements

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

The VAR statements list the numeric variables in the DATA= data set whose values are to be accumulated to form the time series.

An input data set variable can be specified in only one VAR statement. You can specify any number of VAR statements. You can also specify the following *options* in the VAR statements:

**ACCUMULATE=***option*
specifies how the data set observations are to be accumulated within each time period for the variables listed in the VAR statement. If the ACCUMULATE= option is not specified in the VAR statement, accumulation is determined by the ACCUMULATE= option in the ID statement. 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.

**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.

**SETMISS=***option* | *number*

**SETMISSING=** *option* | *number*
specifies how missing values (either actual or accumulated) are to be interpreted in the accumulated time series for variables listed in the VAR statement. If the SETMISSING= option is not specified in the VAR statement, missing values are set based on the SETMISSING= option in the ID statement. 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. You can specify the following transformation *options*:

| | |
|---|---|
| **NONE** | No transformation is applied. This option is the default. |
| **LOG** | Logarithmic transformation |
| **SQRT** | Square-root transformation |
| **LOGISTIC** | Logistic transformation |
| **BOXCOX(***n* ) | Box-Cox transformation with parameter number where *n* is between –5 and 5 |

When the TRANSFORM= option is specified, the time series must be strictly positive.

**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 VAR statement. If the ZEROMISS= option is not specified in the VAR statement, beginning and ending zero values are set based on the ZEROMISS= option of the ID statement. If the ZEROMISS= option is not specified in the ID statement or the VAR statement, no zero value interpretation is performed. For more information, see the ZEROMISS= option in the ID statement.

## REGISTER Statement

**REGISTER** *package* **;**

The REGISTER statement specifies which time series and time frequency analysis packages to make available for your user-defined program. These packages include functions that you can utilize from your program to perform sophisticated time series processing. These packages provide functionality that ranges from a simple function to count missing observations in an array to very sophisticated functions that perform time series statistical analysis.

The REGISTER statement enables you to specify package names that are available for use. You can only specify a single package in a REGISTER statement. However, you can specify multiple REGISTER statements.

All packages that are specified in REGISTER statements are loaded prior to parsing your program statements so that any references are defined at the time your code is parsed. If you specify an invalid package name, then an error is returned prior to parsing your program statements. For more information, see *SAS Forecast Server: Time Series Packages*.

## Program Statements

**Program Statements ;**

You can use most of the programming statements that are allowed in the SAS DATA step.

# Details: TIMEDATA Procedure

The TIMEDATA procedure forms time series data from transactional data. The accumulated time series can then be processed using SAS programming statements. The resulting time series can then be analyzed using time series techniques. The data are analyzed using the following steps (the relevant option is listed to the left):

| | | |
|---|---|---|
| 1. | accumulation | ACCUMULATE= option in the ID or VAR statement |
| 2. | missing value interpretation | SETMISSING= option in the ID or VAR statement |

3. time series transformation — TRANSFORM= option in the VAR statement
4. time series differencing — DIF= and SDIF= options in the VAR statement
5. program execution — SAS programming statements
6. descriptive statistics — OUTSUM= option

## Accumulation

If the ACCUMULATE= option in the ID or VAR statement is specified, 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 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 is specified, all of the preceding observations fall within a three-month period of time between March 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 resulting time series is

```
01MAR1999    40
01APR1999    .
01MAY1999    90
```

If the ACCUMULATE=AVERAGE option is specified, the resulting time series is

```
01MAR1999    20
01APR1999    .
01MAY1999    30
```

If the ACCUMULATE=MINIMUM option is specified, the resulting time series is

```
01MAR1999    10
01APR1999    .
01MAY1999    20
```

If the ACCUMULATE=MEDIAN option is specified, the resulting time series is

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

If the ACCUMULATE=MAXIMUM option is specified, the resulting time series is

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

If the ACCUMULATE=FIRST option is specified, the resulting time series is

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

If the ACCUMULATE=LAST option is specified, the resulting time series is

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

If the ACCUMULATE=STDDEV option is specified, the resulting time series is

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

As you can see from the preceding examples, the accumulated time series can have missing values even though the data set observations contain no 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 value—that is, zero. In the former case, the SETMISSING= option can be used to interpret how missing values are treated. Specify SETMISSING=0 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.

## Time Series Transformation

Four transformations are 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

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

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.

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

More complex time series transformations can be performed by using the EXPAND procedure in SAS/ETS.

## Time Series Differencing

After you optionally transform the series, you can simply or seasonally difference the accumulated series by using the DIF= and SDIF= options in the VAR statement. 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) sdif=(1)
dif=(1,12)
```

Additionally, when $y_t$ is strictly positive and the TRANSFORM=, DIF=, and SDIF= options are combined in the VAR statements, the transformation operation is performed before the differencing operations.

## Summary Statistics

You can compute summary statistics from the working series by specifying the OUTSUM= option or PRINT=SUMMARY.

## Programming Statements

You can typically use most of the SAS programming statements and SAS functions that you can use in a DATA step for defining the FCMP functions and subroutines. However, there are a few differences in the capabilities of the DATA step and the FCMP procedure. For more information, see the “FCMP Procedure” chapter in the *Base SAS Procedures Guide*.

All variables listed in the ID and VAR statements are assigned as predefined arrays for subsequent processing. Additionally, all of the array names listed in the OUTARRAYS statements and all of the scalars names listed in the OUTSCALARS statements are assigned as predefined symbols for subsequent processing.

## Predefined Symbols

In addition to both the predefined arrays listed in the OUTARRAYS statements and also the predefined scalars listed in the OUTSCALARS statements, the TIMEDATA procedure creates the following predefined symbols for use in the program statements:

### Predefined Scalar Values

| | |
|---|---|
| _FORMAT_ | time format either implied by the INTERVAL= option or specified by the FORMAT= option in the ID statement |
| _INTERVAL_ | time interval specified by the INTERVAL= option in the ID statement |
| _LEAD_ | forecast horizon or lead specified by the LEAD= option in the PROC TIMEDATA statement |
| _LENGTH_ | length of the time series associated with the current BY group |
| _SERIES_ | series index or BY-group counter |
| _SEASONALITY_ | length of the seasonal cycle specified by the SEASONALITY= option PROC TIMEDATA statement or implied by the INTERVAL= option in the ID statement |

### Predefined Array Values

| | |
|---|---|
| _TIMEID_ | time ID values |
| _SEASON_ | season index values |
| _CYCLE_ | life-cycle index values |

## Auxiliary Data Sets

Auxiliary data set support enables the TIMEDATA procedure to use auxiliary data sets to contribute input variables to the run of the procedure step. This functionality creates a virtual data source that enables some of the input variables to physically reside in different data sets with some defined in the primary data set defined by the DATA= option and others defined in the data sets that are specified by one or more AUXDATA= options. For example, this functionality enables sharing of common time series data across multiple projects.

Furthermore, auxiliary data set support enables more than the simple separation of shared data. It also facilitates the elimination of redundancy in these auxiliary data sources by performing partial matching on BY-group qualification. Duplication of time series for the full BY-group hierarchy is no longer required for the auxiliary data sets.

Finally, this functionality permits more than one auxiliary data source to be used concurrently to materialize the virtual time series vectors across a given BY-group hierarchy. So variables that have naturally different levels of BY-group qualification can be isolated into separate data sets and supplied with separate AUXDATA= options to optimize data management and performance.

## AUXDATA Functionality

When used, this option declares the presence of an auxiliary data set to optionally provide time series variables to satisfy various declaration statements in the respective procedure steps.

There are two classes of time series data set sources:

- a primary data set from the DATA=DataSet option
- auxiliary data sources from AUXDATA=DataSet options

You can specify zero or more AUXDATA= options in the PROC TIMEDATA statement. Each AUXDATA= option establishes an auxiliary data set source to supply variables declared in subsequent statements that comprise the procedure step.

Variables referenced in the PROC TIMEDATA invocation fall into three classes:

- those that must be physically present in the primary data set
- those that must be physically present in each auxiliary data set
- those that can reside in either the primary or an auxiliary data set

If you specify an ID variable for PROC TIMEDATA, it must be present in the primary data set and all of the auxiliary data sets that you specify. Variables that you specify in the BY statement must be present in the primary data set. A leftmost subset of those BY variables can be present in each of the auxiliary data sets that you specify, and it is not required that all auxiliary data sets contain the same subset. Partial BY-group matching is performed for each auxiliary data set independent of the others.

The time series variables that you specify in VAR statements can be resolved from either the primary data set or an auxiliary data set. Variable resolution proceeds in reverse order from the last AUXDATA= option in the PROC TIMEDATA statement to the first. If the variable in question is not found in any of those, the variable must be present in the primary data set for the procedure step to be successful.

## AUXDATA Alignment across BY Groups

All BY statement variables must be physically present in the primary data set. However, it is not necessary to have the BY variables present in any of the auxiliary data sets. All, some, or none of the BY variables can be present in any auxiliary data set, as your requirements dictate. Partial BY-group matching is performed between the primary data set and the auxiliary data sets based on the number of BY statement variables that are present in the respective auxiliary data sets.

For example, suppose you have a hierarchy of (REGION, PRODUCT) in the primary data set, which holds the time series variables for monthly sales metrics. Suppose you have an auxiliary data set with time series qualified by REGION for pertinent explanatory variables and another with time series for other explanatory variables to be applied across all (REGION, PRODUCT) groupings of the primary data set. In this scenario, each (REGION, PRODUCT) group in the primary data set seeks a match with a corresponding REGION from the first auxiliary data set to materialize the time series for its variables, but no matching is performed on the second auxiliary data set to materialize the time series for its variables. So if (‘SOUTH’,‘EDSEL’) is a BY group from the primary data set, the ‘SOUTH’ BY-group series from the first auxiliary data set are used, and the series from the second auxiliary data set are supplied without qualification. If the next primary BY group is (‘SOUTH’, ‘HUDSON’), then the ‘SOUTH’ BY group is again used to supply the time series from the first auxiliary data set, and the unqualified series are supplied from the second auxiliary data set. So on it goes, each auxiliary data set performing a partial match on the BY variables it holds within the BY group from the primary data set.

### AUXDATA Alignment over the Time Dimension

The series from each BY group of the primary data set defines a reference time span for the auxiliary data sets. Only the intersection of the time interval for each auxiliary series with the reference span is materialized. Head or tail missing values are inserted into the auxiliary series for start or stop times that lie inside the reference span. More generally, missing value semantics apply to the head and tail regions that require filling to materialize the full reference time span.

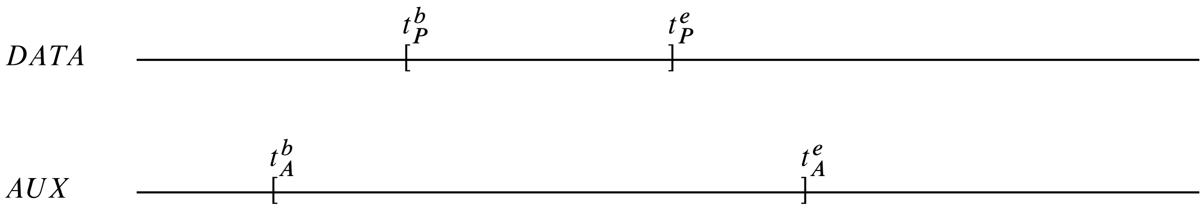
With time series materialized from a single primary data set, there is no latitude for different time ID ranges between the different variables because each observation read contains not only the time ID but also the associated values for all of the variables. With some series materialized from the primary data set and some materialized from auxiliary data sets, the possibility exists for the reference time span to have an arbitrary intersection with the time span of the corresponding series from the auxiliary sources. The intent is to materialize the portion of the auxiliary series time span that intersects with the reference time span and to handle head and tail shortages via missing value semantics as needed.

For the previous usage scenario with a primary data set and two auxiliary data sets, when data are read over a sequence of primary BY groups it might be necessary to materialize various spans of the auxiliary series with appropriate missing value semantics applied as needed to resolve head and tail shortages even though the actual time series contributed from the auxiliary data sets does not physically change. The following discussion breaks this down into several cases depending on intersection possibilities between the reference time span and the auxiliary time span.

Legend:

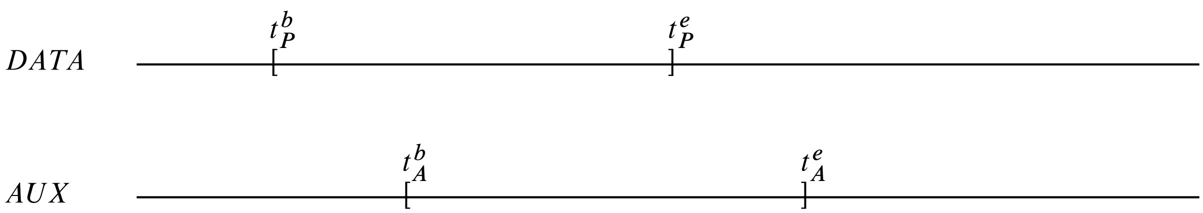
- $t_P^b$ denotes the begin time ID of the primary (DATA=) series.
- $t_P^e$ denotes the end time ID of the primary (DATA=) series.
- $t_A^b$ denotes the begin time ID of the AUXDATA series.
- $t_A^e$ denotes the end time ID of the AUXDATA series.
- $[t_P^b, t_P^e]$ denotes the time span for the primary (DATA=) series (also known as the reference time span).
- $[t_A^b, t_A^e]$ denotes the time span for the AUXDATA series.

### *Case 1:*



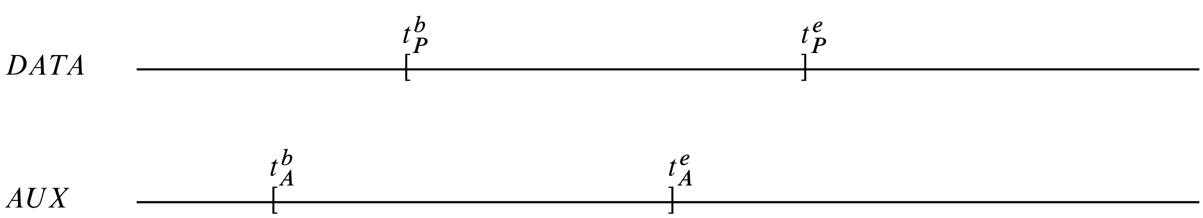

Here $[t_P^b, t_P^e] \subseteq [t_A^b, t_A^e]$. The auxiliary time span includes the reference span as a subset. Values in the AUXDATA series to the left of $t_P^b$ and values to the right of $t_P^e$ are truncated from the AUXDATA series that is materialized in connection with the primary series.

### *Case 2:*



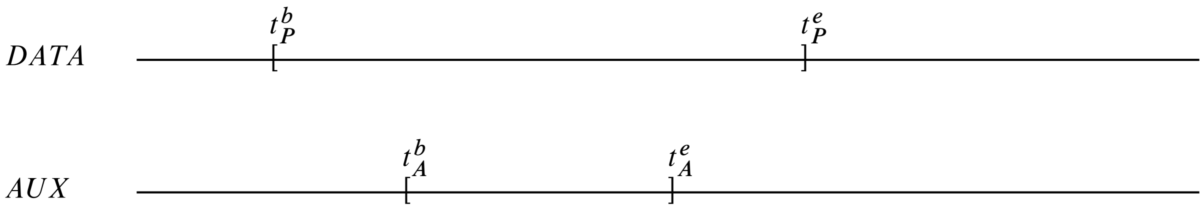

Here $[t_P^b, t_P^e] = [t_P^b, t_A^b) \cup [t_A^b, t_P^e]$. The reference time span leads the auxiliary time span with a non-empty intersection. AUXDATA series values in $[t_P^b, t_A^b)$ are materialized with missing value semantics. AUXDATA series values in $[t_A^b, t_P^e]$ are materialized as actual subject to missing value semantics.

### *Case 3:*



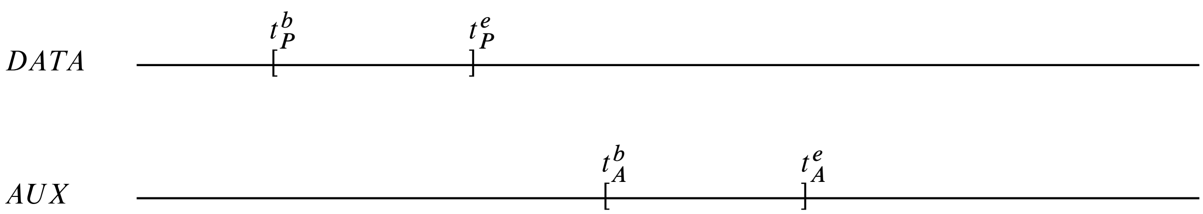

Here $[t_P^b, t_P^e] = [t_P^b, t_A^e] \cup (t_A^e, t_P^e]$. The reference time span lags the auxiliary time span with a non-empty intersection. AUXDATA series values in $[t_P^b, t_A^e]$ are materialized as actual subject to missing value semantics. AUXDATA series values in $(t_A^e, t_P^e]$ are materialized with missing value semantics.

### *Case 4:*




Here $[t_A^b, t_A^e] \subset [t_P^b, t_P^e]$. The auxiliary time span is a subset of the reference time span. AUXDATA series values in $[t_P^b, t_A^b)$ and values in $(t_A^e, t_P^e]$ are materialized with missing value semantics. AUXDATA series values in $[t_A^b, t_A^e]$ are materialized as actual subject to missing value semantics.

### *Case 5:*




Here $[t_P^b, t_P^e] \cap [t_A^b, t_A^e] = \oslash$. The auxiliary time span does not intersect the reference time span at all. In this case all AUXDATA series values are materialized with missing value semantics.

## Data Set Output

The TIMEDATA procedure can create the OUT=, OUTARRAY=, OUTPROCINFO=, OUTSCALAR=, and OUTSUM= data sets. In general, these data sets contain the variables listed in the BY statement. If an analysis step that is related to an output step fails, the values of this step are not recorded or are set to missing in the related output data set but appropriate error or warning messages (or both) are recorded in the log.

## OUT= Data Set

The OUT= data set contains the variables specified in the BY, ID, or VAR statements. If the ID statement is specified, the ID variable values are aligned and extended based on the ALIGN= and INTERVAL= options. The values of the variables specified in the VAR statements are accumulated based on the ACCUMULATE= option, and missing values are interpreted based on the SETMISSING= option.

## OUTARRAY= Data Set

The OUTARRAY= data set contains the variables specified in the BY, ID, or VAR statements. If the ID statement is specified, the ID variable values are aligned and extended based on the ALIGN= and INTERVAL= options. The values of the variables specified in the VAR statements are accumulated based on the ACCUMULATE= option, and missing values are interpreted based on the SETMISSING= option. Additionally, the OUTARRAY= data set contains the variables that are specified in the OUTARRAYS statements and the following variables:

| | |
|---|---|
| _STATUS_ | status flag that indicates whether the requested analyses were successful |
| _SERIES_ | series index or BY-group index |
| _TIMEID_ | time ID values |
| _SEASON_ | season index values |
| _CYCLE_ | life-cycle index values |
| *Array-Variable-Names* | variables listed in the OUTARRAYS statement |

The OUTARRAY= data set contains the arrays that are related to the (accumulated) time series.

## OUTPROCINFO= Data Set

The OUTPROCINFO= data set contains information about the run of the TIMEDATA procedure. The following variables are present:

| | |
|---|---|
| _SOURCE_ | name of the procedure, in this case TIMEDATA |
| _NAME_ | name of the item being reported |
| _LABEL_ | descriptive label for the item in _NAME_ |
| _STAGE_ | current stage of the procedure (for TIMEDATA this is set to ALL) |
| _VALUE_ | value of the item specified in _NAME_ |

## OUTSCALAR= Data Set

The OUTSCALAR= data set contains the variables specified in the BY statement. Additionally, the OUTSCALAR= data set contains the variables that are specified in the OUTSCALARS statements and following variables:

| | |
|---|---|
| _STATUS_ | status flag that indicates whether the requested analyses were successful |
| _SERIES_ | series index or BY-group counter |
| *Scalar-Variable-Names* | variables listed in the OUTSCALARS statement |

The OUTSCALAR= data set contains the scalars that are related to the (accumulated) time series.

## OUTSUM= Data Set

The OUTSUM= data set contains the variables that are specified in the BY statement as and the variables in the following list. The OUTSUM= data set records the descriptive statistics for each variable specified in a VAR statement. Variables related to descriptive statistics are based on the ACCUMULATE= and SETMISSING= options in the ID and VAR statements:

| | |
|---|---|
| _NAME_ | variable name |
| _STATUS_ | status flag that indicates whether the requested analyses were successful |
| _SERIES_ | count of the series processed in each BY group |
| START | the starting date of each series |
| END | the ending date of each series |
| STARTOBS | the beginning observation number of each series |
| ENDOBS | the ending observation number of each series |
| NOBS | number of observations |
| N | number of nonmissing observations |
| NMISS | number of missing observations |
| MINIMUM | minimum value |
| MAXIMUM | maximum value |
| AVG | average value |
| STDDEV | standard deviation |

The OUTSUM= data set contains the descriptive statistics of the (accumulated) time series.

## _STATUS_ Variable Values

The _STATUS_ variable that appears in the OUTSUM= data set contains a value that specifies whether the analysis has been successful or not. The _STATUS_ variable can take the following values:

| | |
|---|---|
| 0 | Analysis was successful. |
| 3000 | Accumulation failed. |
| 4000 | Missing value interpretation failed. |
| 6000 | Series is all missing. |
| 7000 | Transformation failed. |
| 8000 | Differencing failed. |
| 9000 | Descriptive statistics could not be computed. |

## Printed Output

The TIMEDATA 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= option associated with the PROC TIMEDATA statement. In general, if an analysis step related to printed output fails, the values of this step are not printed and appropriate error or warning messages or both are recorded in the log. The printed output is similar to the output data set as follows:

| | |
|---|---|
| PRINT=ARRAYS | prints the arrays similar to the OUTARRAY= data set. |
| PRINT=SCALARS | prints the scalars similar to the OUTSCALAR= data set. |
| PRINT=SUMMARY | prints the summary statistics similar to the OUTSUM= data set. |

## ODS Table Names

Table 36.2 relates the PRINT= options to ODS tables.

**Table 36.2** ODS Tables Produced in PROC TIMEDATA

| ODS Table Name | Description | Statement | Option |
|---|---|---|---|
| Arrays | Arrays table | PRINT | ARRAYS |
| Scalars | Scalars table | PRINT | SCALARS |
| StatisticsSummary | Statistics summary | PRINT | SUMMARY |

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

### Arrays Table

The arrays table (Arrays) illustrate the arrays in tabular form with respect to the Time ID values.

### Scalars Table

The scalars table (Scalars) illustrate the scalars in tabular form.

### Statistics Summary Table

The summary statistics table (StatisticsSummary) illustrate the summary statistics for each array in tabular form.

## ODS Graphics Names

This section describes the graphical output produced by the TIMEDATA procedure. PROC TIMEDATA assigns a name to each graph it creates. These names are listed in Table 36.3.

**Table 36.3** ODS Graphics Produced by PROC TIMEDATA

| ODS Graph Name | Plot Description | Statement | Option |
|---|---|---|---|
| ArrayPlot | Array plot | PLOTS | ARRAY |

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

### Array Plots

The array plots (ArrayPlot) illustrate time series associated with each array. The horizontal axis represents the time ID values, and the vertical axis represents the time series values.

# Examples: TIMEDATA Procedure

## Example 36.1: Accumulating Transactional Data into Time Series Data

This example uses the TIMEDATA procedure to accumulate 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, 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 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. Also, the accumulated time series values for time periods with no transactions are set to zero instead of to missing (SETMISS=0) and only transactions recorded between the first day of 1998 (START='01JAN1998'D ) and last day of 2000 (END='31DEC2000'D) are considered and, if needed, extended to include this range.

```
proc timedata data=retail out=mseries;
   by store;
   id timestamp interval=month
                accumulate=median
                setmiss=0
                start='01jan1998'd
                end  ='31dec2000'd;
   var item1-item8;
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 values Store, Timestamp, and each of the analysis variables in the input data set.

After each set of transactions has been accumulated to form a corresponding time series, accumulated time series can be analyzed 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 date;
   convert storeitem=smooth / transform=(ewma 0.1);
run;
```

The smoothed series are 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 must be changed accordingly and the following statements could be used:

```
proc timedata data=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;
   var _numeric_;
run;
```

The monthly time series data are stored in the data Work.Tseries, and the time ID values use a SAS datetime representation.

## Example 36.2: Using User-Defined Functions and Subroutines

This example uses the TIMEDATA procedure with a user-defined function and subroutine created by the FCMP procedure.

The following statements use the FCMP procedure to create a user-defined subroutine and a user-defined function. Mylog is a subroutine that log-transforms a time series. Mymean is a function that compute the mean of a time series. The subroutine and function definitions are stored in the data set Work.Timefnc. The OPTIONS statement loads the subroutine and function definitions.

```
proc fcmp outlib=work.timefnc.funcs;

   subroutine mylog(actual[*], transform[*]);
   outargs transform;
   actlen  = DIM(actual);
   do t = 1 to actlen;
      transform[t] = log(actual[t]);
   end;
   endsub;

   function mymean(actual[*]);
```

```
   actlen  = DIM(actual);
   sum = 0;
   do t = 1 to actlen;
      sum = sum + actual[t];
   end;
   return( sum / actlen );
   endsub;

run;
quit;

options cmplib = work.timefnc;
```

The input data set Sashelp.Air contains the variables Air and Date. The time series is recorded monthly.

The following statements form quarterly time series from the monthly series based on the median value (ACCUMULATE=TOTAL) of the transactions recorded with each time period and assign the SAS time format (FORMAT=YYMMDD.). The OUTARRAYS statement specifies the Logair and Myair arrays as output. The OUTSCALARS statement specifies the Mystats scalars as output. The other arrays and scalars are not part of the output. The subsequent programming statements create the output arrays and scalars. The PRINT=(ARRAYS SCALARS) prints the output arrays and scalars.

```
proc timedata data=sashelp.air out=work.air
              print=(scalars arrays);
              id date interval=qtr acc=t format=yymmdd.;
              vars air;
              outarrays logair myair;
              outscalars mystats;

       call mylog(air,logair);
       do t = 1 to dim(air);
       myair[t] = air[t] - logair[t];
       end;
       mystats= mymean(air);

run;
```

## Example 36.3: Using Auxiliary Data Sets with PROC TIMEDATA

This example demonstrates the use of the AUXDATA= option in PROC TIMEDATA. The data set Sashelp.Gulfoil contains oil and gas production data from the Gulf of Mexico. The variables RegionName and ProtractionName can be used to define a time series hierarchy of interest. Suppose you want to generate two new series that contain the protraction's share of oil and gas production for its associated region at each time index.

You first use PROC TIMESERIES to perform temporal aggregation (accumulation) of the time series for the RegionName level.

```
proc timeseries data=sashelp.gulfoil
                out=byregion(rename=(oil=roil gas=rgas));
   by regionname;
   id date interval=month accumulate=total notsorted;
   var oil gas;
run;
```

You can then use PROC TIMEDATA with the AUXDATA= option to compute the share of oil and gas production contributed by each protraction within its associated region. PROC TIMEDATA reads a monthly time series for each (RegionName, ProtractionName) group for the variables Oil and Gas from Sashelp.Gulfoil. Two new series are produced in the variables Oilshare and Gasshare that respectively contain the protraction's share of the oil and gas production at the region level of the hierarchy (given by variables Roil and Rgas). Those share variables are specified in the OUTARRAY statement for inclusion in the OUTARRAY= data set (Work.Shares). This example relies on the capability of the AUXDATA= feature to perform partial BY-group matching. The time series that are acquired for the variables Roil and Rgas are the result of matching on the RegionName BY variable from the data set Work.Byregion with the RegionName variable from the BY groups that are acquired from the Sashelp.Gulfoil data set.

```
proc timedata data=sashelp.gulfoil
              auxdata=byregion
              out=_null_
              outarray=shares;
   by regionname protractionname;
   outarray oilshare gasshare;
   var oil gas roil rgas;
   id date interval=month accumulate=total;
   do i=1 to _length_;
      oilshare[i] = oil[i] / roil[i];
      gasshare[i] = gas[i] / rgas[i];
   end;
run;
```

The following code demonstrates that the computed shares sum to 1 for each time index in the resulting Oilshare and Gasshare series. PROC TIMESERIES is used to accumulate the shares for these respective variables from the data set Work.Shares and the accumulated share series at the RegionName level are stored to the data set Work.Rshares with variable names Oilsum and Gassum, respectively. The summary from PROC MEANS for the distinct values of RegionName shows that per-time totals for both share series sums to 1.

```
proc timeseries data=shares
                out=rshares(rename=(oilshare=oilsum gasshare=gassum));
   by regionname;
   id date interval=month accumulate=total notsorted;
   var oilshare gasshare;
run;
proc means data=rshares;
   by regionname;
   var oilsum gassum;
run;
```

**Output 36.3.1** Validation of Oil and Gas Shares by Region

**The MEANS Procedure**

**Region within Gulf of Mexico=Central**

| Variable | N | Mean | Std Dev | Minimum | Maximum |
|---|---|---|---|---|---|
| oilsum | 123 | 1.0000000 | 0 | 1.0000000 | 1.0000000 |
| gassum | 123 | 1.0000000 | 0 | 1.0000000 | 1.0000000 |

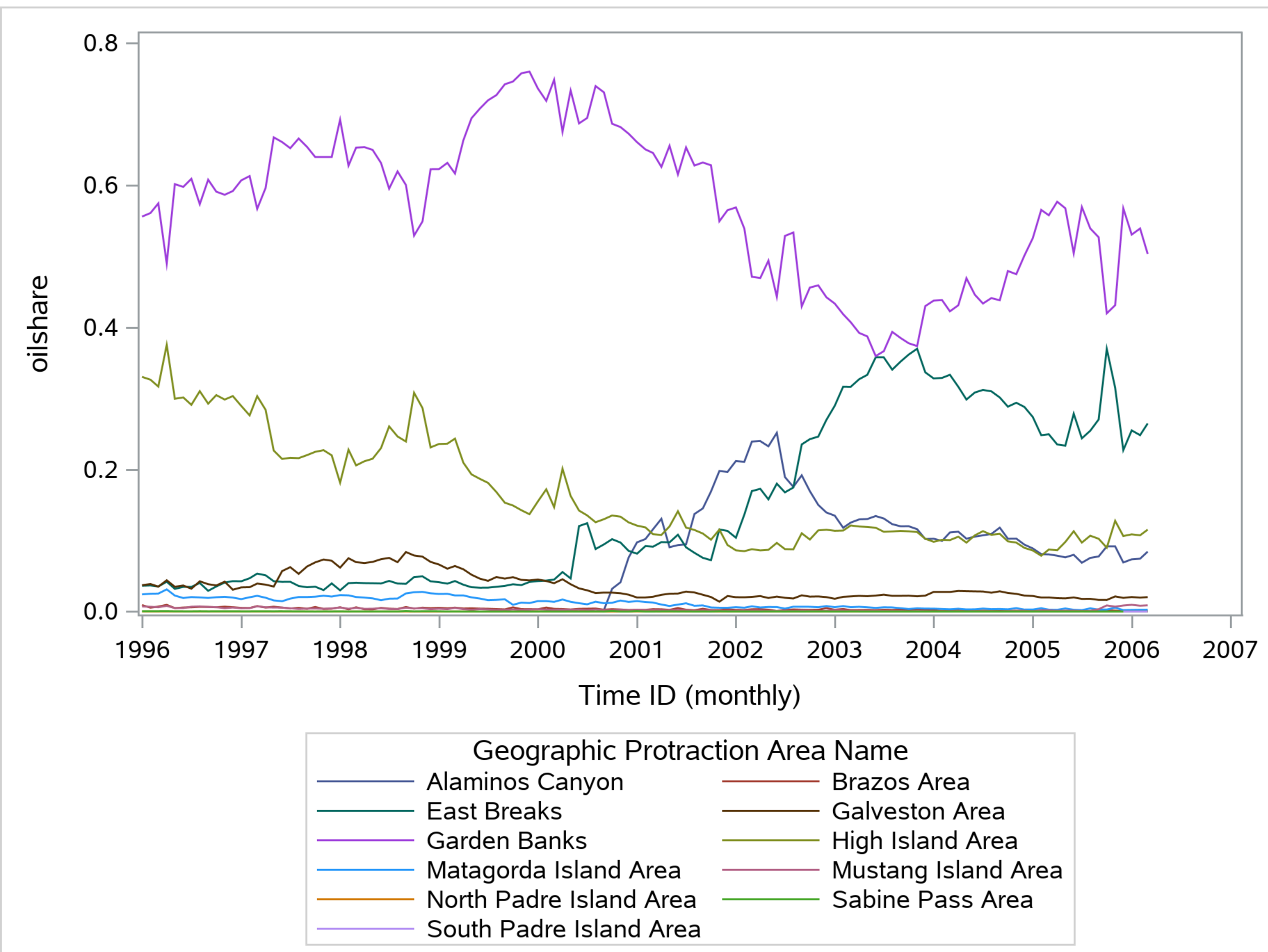
**Region within Gulf of Mexico=Western**

| Variable | N | Mean | Std Dev | Minimum | Maximum |
|---|---|---|---|---|---|
| oilsum | 123 | 1.0000000 | 0 | 1.0000000 | 1.0000000 |
| gassum | 123 | 1.0000000 | 0 | 1.0000000 | 1.0000000 |

You might also want to plot the share series. The following code produces a graph that overlays the protraction level share series for oil production for the Western region:

```
proc sgplot data=shares(where=(RegionName='Western'));
   series x=Date y=OilShare/group=ProtractionName;
run;
```

**Output 36.3.2** Protraction Share of Oil Production for Western Region



# References

Keogh, E., Chu, S., Hart, D., and Pazzani, M. (2004). "Segmenting Time Series: A Survey and Novel Approach." In *Data Mining in Time Series Databases*, edited by M. Last, A. Kandel, and H. Bunke, 1–22. London: World Scientific. https://www.worldscientific.com/worldscibooks/10.1142/5210.

# Subject Index

# Syntax Index