Chapter 11
The EXPAND Procedure

Chapter Table of Contents

Chapter 11

The EXPAND Procedure

Overview

The EXPAND procedure converts time series from one sampling interval or frequency to another and interpolates missing values in time series. A wide array of data transformation is also supported. Using PROC EXPAND, you can collapse time series data from higher frequency intervals to lower frequency intervals, or expand data from lower frequency intervals to higher frequency intervals. For example, quarterly estimates can be interpolated from an annual series, or quarterly values can be aggregated to produce an annual series.

Time series frequency conversion is useful when you need to combine series with different sampling intervals into a single data set. For example, if you need as input to a monthly model a series that is only available quarterly, you might use PROC EXPAND to interpolate the needed monthly values.

You can also interpolate missing values in time series, either without changing series frequency or in conjunction with expanding or collapsing the series.

You can convert between any combination of input and output frequencies that can be specified by SAS time interval names. (See Chapter 3, "Date Intervals, Formats, and Functions,", for a complete description of SAS interval names.) When the "from" and "to" intervals are specified, PROC EXPAND automatically accounts for calendar effects such as the differing number of days in each month and leap years.

The EXPAND procedure also handles conversions of frequencies that cannot be defined by standard interval names. Using the FACTOR= option, you can interpolate any number of output observations for each group of a specified number of input observations. For example, if you specify the option FACTOR=(13:2), 13 equally spaced output observations are interpolated from each pair of input observations.

You can also convert aperiodic series, observed at arbitrary points in time, into periodic estimates. For example, a series of randomly timed quality control spot-check results might be interpolated to form estimates of monthly average defect rates.

The EXPAND procedure can also change the observation characteristics of time series. Time series observations can measure beginning-of-period values, end-of-period values, midpoint values, or period averages or totals. PROC EXPAND can convert between these cases. You can construct estimates of interval averages from end-of-period values of a variable, estimate beginning-of-period or midpoint values from interval averages, or compute averages from interval totals, and so forth.

By default, the EXPAND procedure fits cubic spline curves to the nonmissing values of variables to form continuous-time approximations of the input series. Output series are then generated from the spline approximations. Several alternate conversion methods are described in the section “Conversion Methods” on page 557. You can also interpolate estimates of the rate of change of time series by differentiating the interpolating spline curve.

Various transformations can be applied to the input series prior to interpolation and to the interpolated output series. For example, the interpolation process can be modified by transforming the input series, interpolating the transformed series, and applying the inverse of the input transformation to the output series. PROC EXPAND can also be used to apply transformations to time series without interpolation or frequency conversion.

The results of the EXPAND procedure are stored in a SAS data set. No printed output is produced.

Getting Started

Converting to Higher Frequency Series

To create higher frequency estimates, specify the input and output intervals with the FROM= and TO= options, and list the variables to be converted in a CONVERT statement. For example, suppose variables X, Y, and Z in the data set ANNUAL are annual time series, and you want monthly estimates. You can interpolate monthly estimates by using the following statements:

```
proc expand data=annual out=monthly from=year to=month;
   convert x y z;
run;
```

Note that interpolating values of a time series does not add any real information to the data as the interpolation process is not the same process that generated the other (nonmissing) values in the series. While time series interpolation can sometimes be useful, great care is needed in analyzing time series containing interpolated values.

Aggregating to Lower Frequency Series

PROC EXPAND provides two ways to convert from a higher frequency to a lower frequency. When a curve fitting method is used, converting to a lower frequency is no different than converting to a higher frequency–you just specify the desired output frequency with the TO= option. This provides for interpolation of missing values and allows conversion from non-nested intervals, such as converting from weekly to monthly values.

Alternatively, you can specify simple aggregation or selection without interpolation of missing values. This might be useful, for example, if you wanted to add up monthly values to produce annual totals but wanted the annual output data set to contain values only for complete years.

To perform simple aggregation, use the METHOD=AGGREGATE option in the CONVERT statement. For example, the following statements aggregate monthly values to yearly values:

```
proc expand data=monthly out=annual from=month to=year;
   convert x y z / method=aggregate;
   convert a b c / observed=total method=aggregate;
   id date;
run;
```

Note that the AGGREGATE method can be used only if the input intervals are nested within the output intervals, as when converting from daily to monthly or from monthly to yearly frequency.

Combining Time Series with Different Frequencies

One important use of PROC EXPAND is to combine time series measured at different sampling frequencies. For example, suppose you have data on monthly money stocks (M1), quarterly gross domestic product (GDP), and weekly interest rates (INTEREST), and you want to perform an analysis of a model that uses all these variables. To perform the analysis, you first need to convert the series to a common frequency and combine the variables into one data set.

The following statements illustrate this process for the three data sets QUARTER, MONTHLY, and WEEKLY. The data sets QUARTER and WEEKLY are converted to monthly frequency using two PROC EXPAND steps, and the three data sets are then merged using a DATA step MERGE statement to produce the data set COMBINED.

```
proc expand data=quarter out=temp1 from=qtr to=month;
   id date;
   convert gdp / observed=total;
run;

proc expand data=weekly out=temp2 from=week to=month;
   id date;
   convert interest / observed=average;
run;

data combined;
   merge monthly temp1 temp2;
   by date;
run;
```

See Chapter 2, “Working with Time Series Data,”, for further discussion of time series periodicity, time series dating, and time series interpolation.

Interpolating Missing Values

To interpolate missing values in time series without converting the observation frequency, leave off the TO= option. For example, the following statements interpolate any missing values in the time series in the data set ANNUAL.

```
proc expand data=annual out=new from=year;
   id date;
   convert x y z;
   convert a b c / observed=total;
run;
```

To interpolate missing values in variables observed at specific points in time, omit both the FROM= and TO= options and use the ID statement to supply time values for the observations. The observations do not need to be periodic or form regular time series, but the data set must be sorted by the ID variable. For example, the following statements interpolate any missing values in the numeric variables in the data set A.

```
   proc expand data=a out=b;
      id date;
   run;
```

If the observations are equally spaced in time, and all the series are observed as beginning-of-period values, only the input and output data sets need to be specified. For example, the following statements interpolate any missing values in the numeric variables in the data set A, assuming that the observations are at equally spaced points in time.

```
   proc expand data=a out=b;
   run;
```

Refer to the section "Missing Values" on page 564 for further information.

Requesting Different Interpolation Methods

By default, a cubic spline curve is fit to the input series, and the output is computed from this interpolating curve. Other interpolation methods can be specified with the METHOD= option on the CONVERT statement. The section "Conversion Methods" on page 557 explains the available methods.

For example, the following statements convert annual series to monthly series using linear interpolation instead of cubic spline interpolation.

```
   proc expand data=annual out=monthly from=year to=month;
      id date;
      convert x y z / method=join;
   run;
```

Using the ID Statement

An ID statement is normally used with PROC EXPAND to specify a SAS date or datetime variable to identify the time of each input observation. An ID variable allows PROC EXPAND to do the following:

- identify the observations in the output data set
- determine the time span between observations and detect gaps in the input series caused by omitted observations
- account for calendar effects such as the number of days in each month and leap years

If you do not specify an ID variable with SAS date or datetime values, PROC EXPAND makes default assumptions that may not be what you want. See the section "ID Statement" for details.

Specifying Observation Characteristics

It is important to distinguish between variables that are measured at points in time and variables that represent totals or averages over an interval. Point-in-time values are often called *stocks* or *levels*. Variables that represent totals or averages over an interval are often called *flows* or *rates*.

For example, the annual series “U.S. Gross Domestic Product” represents the total value of production over the year and also the yearly average rate of production in dollars per year. However, a monthly variable *inventory* may represent the cost of a stock of goods as of the end of the month.

When the data represent periodic totals or averages, the process of interpolation to a higher frequency is sometimes called *distribution*, and the total values of the larger intervals are said to be *distributed* to the smaller intervals. The process of interpolating periodic total or average values to lower frequency estimates is sometimes called *aggregation*.

By default, PROC EXPAND assumes that all time series represent beginning-of-period point-in-time values. If a series does not measure beginning of period point-in-time values, interpolation of the data values using this assumption is not appropriate, and you should specify the correct observation characteristics of the series. The observation characteristics of series are specified with the OBSERVED= option on the CONVERT statement.

For example, suppose that the data set ANNUAL contains variables A, B, and C that measure yearly totals, while the variables X, Y, and Z measure first-of-year values. The following statements estimate the contribution of each month to the annual totals in A, B, and C, and interpolate first-of-month estimates of X, Y, and Z.

```
proc expand data=annual out=monthly from=year to=month;
   id date;
   convert x y z;
   convert a b c / observed=total;
run;
```

The EXPAND procedure supports five different observation characteristics. The OBSERVED= option values for these five observation characteristics are:

BEGINNING	beginning-of-period values
MIDDLE	period midpoint values
END	end-of-period values
TOTAL	period totals
AVERAGE	period averages

The interpolation of each series is adjusted appropriately for its observation characteristics. When OBSERVED=TOTAL or AVERAGE is specified, the interpolating curve is fit to the data values so that the area under the curve within each input interval equals the value of the series. For OBSERVED=MIDDLE or END, the curve is fit through the data points, with the time position of each data value placed at the specified offset from the start of the interval.

See the section “The OBSERVED= Option” on page 549 for details.

Converting Observation Characteristics

The EXPAND procedure can be used to interpolate values for output series with different observation characteristics than the input series. To change observation characteristics, specify two values in the OBSERVED= option. The first value specifies the observation characteristics of the input series; the second value specifies the observation characteristics of the output series.

For example, the following statements convert the period total variable A in the data set ANNUAL to yearly midpoint estimates. This example does not change the series frequency, and the other variables in the data set are copied to the output data set unchanged.

```
proc expand data=annual out=new from=year;
   id date;
   convert a / observed=(total,middle);
run;
```

Creating New Variables

You can use the CONVERT statement to name a new variable to contain the results of the conversion. Using this feature, you can create several different versions of a series in a single PROC EXPAND step. Specify the new name after the input variable name and an equal sign:

```
convert variable=newname ... ;
```

For example, suppose you are converting quarterly data to monthly and you want both first-of-month and midmonth estimates for a beginning-of-period variable X. The following statements perform this task:

```
proc expand data=a out=b from=qtr to=month;
   id date;
   convert x=x_begin  / observed=beginning;
   convert x=x_mid    / observed=(beginning,middle);
run;
```

Transforming Series

The interpolation methods used by PROC EXPAND assume that there are no restrictions on the range of values that series can have. This assumption can sometimes cause problems if the series must be within a certain range.

For example, suppose you are converting monthly sales figures to weekly estimates. Sales estimates should never be less than zero, but since the spline curve ignores this restriction some interpolated values may be negative. One way to deal with this problem is to transform the input series before fitting the interpolating spline and then reverse transform the output series.

You can apply various transformations to the input series using the TRANSFORMIN= option on the CONVERT statement. (The TRANSFORMIN= option can be abbreviated as TRANSFORM= or TIN=.) You can apply transformations to the output series using the TRANSFORMOUT= option. (The TRANSFORMOUT= option can be abbreviated as TOUT=.)

For example, you might use a logarithmic transformation of the input sales series and exponentiate the interpolated output series. The following statements fit a spline curve to the log of SALES and then exponentiate the output series.

```
proc expand data=a out=b from=month to=week;
   id date;
   convert sales / observed=total
                   transformin=(log) transformout=(exp);
run;
```

As another example, suppose you are interpolating missing values in a series of market share estimates. Market shares must be between 0% and 100%, but applying a spline interpolation to the raw series can produce estimates outside of this range.

The following statements use the logistic transformation to transform proportions in the range 0 to 1 to values in the range $-\infty$ to $+\infty$. The TIN= option first divides the market shares by 100 to rescale percent values to proportions and then applies the LOGIT function. The TOUT= option applies the inverse logistic function ILOGIT to the interpolated values to convert back to proportions and then multiplies by 100 to rescale back to percentages.

```
proc expand data=a out=b;
   id date;
   convert mshare / tin=( / 100 logit ) tout=( ilogit * 100 );
run;
```

You can also use the TRANSFORM= (or TRANSFORMOUT=) option as a convenient way to do calculations normally performed with the SAS DATA step. For example, the following statements add the lead of X to the data set A. The METHOD=NONE option is used to suppress interpolation.

```
proc expand data=a method=none;
   id date;
   convert x=xlead / transform=(lead);
run;
```

Any number of operations can be listed in the TRANSFORMIN= and TRANSFORMOUT= options. See Table 11.1 for a list of the operations supported.

Syntax

The EXPAND procedure uses the following statements:

PROC EXPAND *options* **;**
BY *variables* **;**
CONVERT *variables / options* **;**
ID *variable* **;**

Functional Summary

The statements and options controlling the EXPAND procedure are summarized in the following table.

Description	Statement	Option
Statements		
specify BY-group processing	BY	
specify conversion options	CONVERT	
specify the ID variable	ID	
Data Set Options		
specify the input data set	PROC EXPAND	DATA=
specify the output data set	PROC EXPAND	OUT=
write interpolating functions to a data set	PROC EXPAND	OUTEST=
extrapolate values before or after input series	PROC EXPAND	EXTRAPOLATE
Input and Output Frequencies		
specify input frequency	PROC EXPAND	FROM=
specify output frequency	PROC EXPAND	TO=
specify frequency conversion factor	PROC EXPAND	FACTOR=
control the alignment of SAS Date values	PROC EXPAND	ALIGN=
Interpolation Control Options		
specify interpolation method	PROC EXPAND, CONVERT	METHOD=
specify observation characteristics	CONVERT	OBSERVED=
specify transformations of the input series	CONVERT	TRANSIN=
specify transformations of the output series	CONVERT	TRANSOUT=

PROC EXPAND Statement

PROC EXPAND *options;*

The following options can be used with the PROC EXPAND statement.

Data Set Options

DATA=*SAS-data-set*

names the input data set. If the DATA= option is omitted, the most recently created SAS data set is used.

OUT=*SAS-data-set*

names the output data set containing the result time series. If OUT= is not specified, the data set is named using the DATA*n* convention. See the section "OUT= Data Set" on page 567 for details.

OUTEST=*SAS-data-set*

names an output data set containing the coefficients of the spline curves fit to the input series. If the OUTEST= option is not specified, the spline coefficients are not output. See the section "OUTEST= Data Set" on page 568 for details.

Options That Define Input and Output Frequencies

FACTOR=*n*
FACTOR=*(n:m)*
FACTOR=*(n,m)*

specifies the number of output observations to be created from the input observations. FACTOR=(*n*:*m*) specifies that *n* output observations are to be produced for each group of *m* input observations. FACTOR=*n* is the same as FACTOR=(*n*:1).

The FACTOR= option cannot be used if the TO= option is used. The default value is FACTOR=(1:1). For more information, see the "Frequency Conversion" section (page 552).

FROM=*interval*

specifies the time interval between observations in the input data set. Examples of FROM= values are YEAR, QTR, MONTH, DAY, and HOUR. See Chapter 3, "Date Intervals, Formats, and Functions," for a complete description and examples of interval specification.

TO=*interval*

specifies the time interval between observations in the output data set. By default, the TO= interval is generated from the combination of the FROM= and the FACTOR= values or is set to be the same as the FROM= value if FACTOR= is not specified. See Chapter 3, "Date Intervals, Formats, and Functions," for a description of interval specifications.

ALIGN=*option*

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

Options to Control the Interpolation

METHOD=*option*

METHOD=SPLINE(*constraint* **[,** *constraint***])**

specifies the method used to convert the data series. The methods supported are SPLINE, JOIN, STEP, AGGREGATE, and NONE. The METHOD= option specified on the PROC EXPAND statement can be overridden for particular series by the METHOD= option on the CONVERT statement. The default is METHOD=SPLINE. The *constraint* specifications for METHOD=SPLINE can have the values NOTAKNOT (the default), NATURAL, SLOPE=*value*, and/or CURVATURE=*value*. See the "Conversion Methods" section on page 557 for more information about these methods.

OBSERVED=*value*

OBSERVED=*(from-value, to-value)*

indicates the observation characteristics of the input time series and of the output series. Specifying the OBSERVED= option on the PROC EXPAND statement sets the default OBSERVED= value for subsequent CONVERT statements. See the sections "CONVERT Statement" and "The OBSERVED= Option" later in this chapter for details. The default is OBSERVED=BEGINNING.

EXTRAPOLATE

specifies that missing values at the beginning or end of input series be replaced with values produced by a linear extrapolation of the interpolating curve fit to the input series. See the section "Extrapolation" later in this chapter for details.

By default, PROC EXPAND avoids extrapolating values beyond the first or last input value for a series and only interpolates values within the range of the nonmissing input values. Note that the extrapolated values are often not very accurate, and for the SPLINE method the EXTRAPOLATE option results may be very unreasonable. The EXTRAPOLATE option is not normally used.

BY Statement

BY *variables;*

A BY statement can be used with PROC EXPAND to obtain separate analyses on observations in groups defined by the BY variables. The input data set must be sorted by the BY variables and be sorted by the ID variable within each BY group.

Use a BY statement when you want to interpolate or convert time series within levels of a cross-sectional variable. For example, suppose you have a data set STATE containing annual estimates of average disposable personal income per capita (DPI) by state and you want quarterly estimates by state. These statements convert the DPI series within each state:

```
proc sort data=state;
   by state date;
run;
```

```
proc expand data=state out=stateqtr from=year to=qtr;
   convert dpi;
   by state;
   id date;
run;
```

CONVERT Statement

CONVERT *variable=newname ... / options;*

The CONVERT statement lists the variables to be processed. Only numeric variables can be processed.

For each of the variables listed, a new variable name can be specified after an equal sign to name the variable in the output data set that contains the converted values. If a name for the output series is not given, the variable in the output data set has the same name as the input variable.

Any number of CONVERT statements can be used. If no CONVERT statement is used, all the numeric variables in the input data set except those appearing in the BY and ID statements are processed.

The following options can be used with the CONVERT statement.

METHOD=*option*
METHOD=SPLINE(*constraint* **[,** *constraint* **])**
specifies the method used to convert the data series. The methods supported are SPLINE, JOIN, STEP, AGGREGATE, and NONE. The METHOD= option specified on the PROC EXPAND statement can be overridden for particular series by the METHOD= option on the CONVERT statement. The default is METHOD=SPLINE. The *constraint* specifications for METHOD=SPLINE can have the values NOTAKNOT (the default), NATURAL, SLOPE=*value*, and/or CURVATURE=*value*. See the “Conversion Methods” section on page 557 for more information about these methods.

OBSERVED=*value*
OBSERVED=*(from-value, to-value)*
indicates the observation characteristics of the input time series and of the output series. The values supported are TOTAL, AVERAGE, BEGINNING, MIDDLE, and END. In addition, DERIVATIVE can be specified as the *to-value* when the SPLINE method is used. See the section “The OBSERVED= Option” later in this chapter for details.

The default is the value specified for the OBSERVED= option on the PROC EXPAND statement, if any, or else the default value is OBSERVED=BEGINNING.

TRANSFORMIN=*(operation ...)*
specifies a list of transformations to be applied to the input series before the interpolating function is fit. The operations are applied in the order listed. See the section “Transformation Operations” later in this chapter for the operations that can be spec-

ified. The TRANSFORMIN= option can be abbreviated as TRANSIN=, TIN=, or TRANSFORM=.

TRANSFORMOUT=*(operation ...)*

specifies a list of transformations to be applied to the output series. The operations are applied in the order listed. See the section "Transformation Operations" later in this chapter for the operations that can be specified. The TRANSFORMOUT= option can be abbreviated as TRANSOUT=, or TOUT=.

ID Statement

ID *variable;*

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.

The input data must form time series. This means that the observations in the input data set must be sorted by the ID variable (within the BY variables, if any). Moreover, there should be no duplicate observations, and no two observations should have ID values within the same time interval as defined by the FROM= option.

If the ID statement is omitted, SAS date or datetime values are generated to label the input observations. These ID values are generated by assuming that the input data set starts at a SAS date value of 0, that is, 1 January 1960. This default starting date is then incremented for each observation by the FROM= interval (using the INTNX function). If the FROM= option is not specified, the ID values are generated as the observation count minus 1. When the ID statement is not used, an ID variable is added to the output data set named either DATE or DATETIME, depending on the value specified in the TO= option. If neither the TO= option nor the FROM= option is given, the ID variable in the output data set is named TIME.

Details

Frequency Conversion

Frequency conversion is controlled by the FROM=, TO=, and FACTOR= options. The possible combinations of these options are explained in the following:

None Used
If FROM=, TO=, and FACTOR= are not specified, no frequency conversion is done. The data are processed to interpolate any missing values and perform any specified transformations. Each input observation produces one output observation.

FACTOR=(n:m)
FACTOR=(*n*:*m*) specifies that *n* output observations are produced for each group of *m* input observations. The fraction *m*/*n* is reduced first: thus FACTOR=(10:6) is equivalent to FACTOR=(5:3). Note that if *m*/*n*=1, the result is the same as the case given previously under “None Used”.

FROM=interval
The FROM= option used alone establishes the frequency and interval widths of the input observations. Missing values are interpolated, and any specified transformations are performed, but no frequency conversion is done.

TO=interval
When the TO= option is used without the FROM= option, output observations with the TO= frequency are generated over the range of input ID values. The first output observation is for the TO= interval containing the ID value of the first input observation; the last output observation is for the TO= interval containing the ID value of the last input observation. The input observations are not assumed to form regular time series and may represent aperiodic points in time. An ID variable is required to give the date or datetime of the input observations.

FROM=interval TO=interval
When both the FROM= and TO= options are used, the input observations have the frequency given by the FROM= interval, and the output observations have the frequency given by the TO= interval.

FROM=interval FACTOR=(n:m)
When both the FROM= and FACTOR= options are used, a TO= interval is inferred from the combination of the FROM=*interval* and the FACTOR=(*n:m*) values specified. For example, FROM=YEAR FACTOR=4 is the same as FROM=YEAR TO=QTR. Also, FROM=YEAR FACTOR=(3:2) is the same as FROM=YEAR used with TO=MONTH8. Once the implied TO= interval is determined, this combination operates the same as if FROM= and TO= had been specified. If no valid TO= interval can be constructed from the combination of the FROM= and FACTOR= options, an error is produced.

TO=interval FACTOR=(n:m)
The combination of the TO= option and the FACTOR= option is not allowed and produces an error.

ALIGN= option

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

Converting to a Lower Frequency

When converting to a lower frequency, the results are either exact or approximate, depending on whether or not the input intervals nest within the output intervals and depending on the need to interpolate missing values within the series. If the TO= interval is nested within the FROM= interval (as when converting monthly to yearly), and if there are no missing input values or partial periods, the results are exact.

When values are missing or the FROM= intervals are not nested within the TO= intervals (as when aggregating weekly to monthly), the results depend on an interpolation. The METHOD=AGGREGATE option always produces exact results, never an interpolation. However, this method cannot be used unless the FROM= interval is nested within the TO= interval.

Identifying Observations

The variable specified in the ID statement is used to identify the observations. Usually, SAS date or datetime values are used for this variable. PROC EXPAND uses the ID variable to do the following:

- identify the time interval of the input values
- validate the input data set observations
- compute the ID values for the observations in the output data set

Identifying the Input Time Intervals

When the FROM= option is specified, observations are understood to refer to the whole time interval and not to a single time point. The ID values are interpreted as identifying the FROM= time interval containing the value. In addition, the widths of these input intervals are used by the OBSERVED= cases TOTAL, AVERAGE, MIDDLE, and END.

For example, if FROM=MONTH is specified, then each observation is for the whole calendar month containing the ID value for the observation, and the width of the time interval covered by the observation is the number of days in that month. Therefore, if FROM=MONTH, the ID value ’31MAR92’D is equivalent to the ID value ’1MAR92’D–both of these ID values identify the same interval, March of 1992.

Widths of Input Time Intervals

When the FROM= option is not specified, the ID variable values are usually interpreted as referring to points in time. However, if an OBSERVED= option is specified that assumes the observations refer to whole intervals and also requires interval widths, then, in the absence of the FROM= specification, interval widths are assumed to be the time span between ID values. For the last observation, the interval width is assumed to be the same as for the next to last observation. (If neither the FROM= option nor the ID statement are specified, interval widths are assumed to be 1.0.) A note is printed in the SAS log warning that this assumption is made.

Validating the Input Data Set Observations

The ID variable is used to verify that successive observations read from the input data set correspond to sequential FROM= intervals. When the FROM= option is not used, PROC EXPAND verifies that the ID values are nonmissing and in ascending order. An error message is produced and the observation is ignored when an invalid ID value is found in the input data set.

ID values for Observations in the Output Data Set

The time unit used for the ID variable in the output data set is controlled by the interval value specified by the TO= option. If you specify a date interval for the TO= value, the ID variable values in the output data set are SAS date values. If you specify a datetime interval for the TO= value, the ID variable values in the output data set are SAS datetime values.

Range of Output Observations

If no frequency conversion is done, the range of output observations is the same as in the input data set.

When frequency conversion is done, the observations in the output data set range from the earliest start of any result series to the latest end of any result series. Observations at the beginning or end of the input range for which all result values are missing are not written to the OUT= data set.

When the EXTRAPOLATE option is not used, the range of the nonmissing output results for each series is as follows. The first result value is for the TO= interval that contains the ID value of the start of the FROM= interval containing the ID value of the first nonmissing input observation for the series. The last result value is for the TO= interval that contains the end of the FROM= interval containing the ID value of the last nonmissing input observation for the series.

When the EXTRAPOLATE option is used, result values for all series are computed for the full time range covered by the input data set.

Extrapolation

The spline functions fit by the EXPAND procedure are very good at approximating continuous curves within the time range of the input data but poor at extrapolating beyond the range of the data. The accuracy of the results produced by PROC EXPAND may be somewhat less at the ends of the output series than at time periods for which there are several input values at both earlier and later times. The curves fit by PROC EXPAND should not be used for forecasting.

PROC EXPAND normally avoids extrapolation of values beyond the time range of the nonmissing input data for a series, unless the EXTRAPOLATE option is used. However, if the start or end of the input series does not correspond to the start or end of an output interval, some output values may depend in part on an extrapolation.

For example, if FROM=YEAR, TO=WEEK, and OBSERVED=BEGINNING, the first observation output for a series is for the week of 1 January of the first nonmissing input year. If 1 January of that year is not a Sunday, the beginning of this week falls before the date of the first input value, and therefore a beginning-of-period output value for this week is extrapolated.

This extrapolation is made only to the extent needed to complete the terminal output intervals that overlap the endpoints of the input series and is limited to no more than the width of one FROM= interval or one TO= interval, whichever is less. This restriction of the extrapolation to complete terminal output intervals is applied to each series separately, and it takes into account the OBSERVED= option for the input and output series.

When the EXTRAPOLATE option is used, the normal restriction on extrapolation is overridden. Output values are computed for the full time range covered by the input data set.

For the SPLINE method, extrapolation is performed by a linear projection of the trend of the cubic spline curve fit to the input data, not by extrapolation of the first and last cubic segments.

The OBSERVED= Option

The values of the CONVERT statement OBSERVED= option are as follows:

BEGINNING indicates that the data are beginning-of-period values. OBSERVED=BEGINNING is the default.

MIDDLE indicates that the data are period midpoint values.

ENDING indicates that the data represent end-of-period values.

TOTAL indicates that the data values represent period totals for the time interval corresponding to the observation.

AVERAGE indicates that the data values represent period averages.

DERIVATIVE requests that the output series be the derivatives of the cubic spline curve fit to the input data by the SPLINE method.

If only one value is specified in the OBSERVED= option, that value applies to both the input and the output series. For example, OBSERVED=TOTAL is the same as OBSERVED=(TOTAL,TOTAL), which indicates both that the input values represent totals over the time intervals corresponding to the input observations and that the converted output values also represent period totals. The value DERIVATIVE can be used only as the second OBSERVED= option value, and it can be used only when METHOD=SPLINE is specified or is the default method.

Since the TOTAL, AVERAGE, MIDDLE, and END cases require that the width of each input interval be known, both the FROM= option and an ID statement are normally required if one of these observation characteristics is specified for any series. However, if the FROM= option is not specified, each input interval is assumed to extend from the ID value for the observation to the ID value of the next observation, and the width of the interval for the last observation is assumed to be the same as the width for the next to last observation.

Scale of OBSERVED=AVERAGE Values

The average values are assumed to be expressed in the time units defined by the FROM= or TO= option. That is, the product of the average value for an interval and the width of the interval is assumed to equal the total value for the interval. For purposes of interpolation, OBSERVED=AVERAGE values are first converted to OBSERVED=TOTAL values using this assumption, and then the interpolated totals are converted back to averages by dividing by the widths of the output intervals. For example, suppose the options FROM=MONTH, TO=HOUR, and OBSERVED=AVERAGE are specified.

Since FROM=MONTH in this example, each input value is assumed to represent an average rate per day such that the product of the value and the number of days in the month is equal to the total for the month. The input values are assumed to represent a per-day rate because FROM=MONTH implies SAS date ID values that measure time in days, and therefore the widths of MONTH intervals are measured in days. If FROM=DTMONTH is used instead, the values are assumed to represent a per-second rate, because the widths of DTMONTH intervals are measured in seconds.

Since TO=HOUR in this example, the output values are scaled as an average rate per second such that the product of each output value and the number of seconds in an hour (3600) is equal to the interpolated hourly total. A per-second rate is used because TO=HOUR implies SAS datetime ID values that measure time in seconds, and therefore the widths of HOUR intervals are measured in seconds.

Note that the scale assumed for OBSERVED=AVERAGE data is important only when converting between AVERAGE and another OBSERVED= option, or when converting between SAS date and SAS datetime ID values. When both the input and the output series are AVERAGE values, and the units for the ID values are not changed, the scale assumed does not matter.

For example, suppose you are converting a series gross domestic product (GDP) from quarterly to monthly. The GDP values are quarterly averages measured at annual rates. If you want the interpolated monthly values to also be measured at annual rates, then the option OBSERVED=AVERAGE works fine. Since there is no change of scale involved in this problem, it makes no difference that PROC EXPAND assumes daily rates instead of annual rates.

However, suppose you want to convert GDP from quarterly to monthly and also convert from annual rates to monthly rates, so that the result is total gross domestic product for the month. Using the option OBSERVED=(AVERAGE,TOTAL) would fail, because PROC EXPAND assumes the average is scaled to daily, not annual, rates.

One solution is to rescale to quarterly totals and treat the data as totals. You could use the options TRANSFORMIN=(/ 4) OBSERVED=TOTAL. Alternatively, you could treat the data as averages but first convert to daily rates. In this case you would use the options TRANSFORMIN=(/ 365.25) OBSERVED=AVERAGE.

Results of the OBSERVED=DERIVATIVE Option

If the first value of the OBSERVED= option is BEGINNING, TOTAL, or AVERAGE, the result is the derivative of the spline curve evaluated at first-of-period ID values for the output observation. For OBSERVED=(MIDDLE,DERIVATIVE), the derivative of the function is evaluated at output interval midpoints. For OBSERVED=(END,DERIVATIVE), the derivative is evaluated at end-of-period ID values.

Conversion Methods

The SPLINE Method

The SPLINE method fits a cubic spline curve to the input values. A cubic spline is a segmented function consisting of third-degree (cubic) polynomial functions joined together so that the whole curve and its first and second derivatives are continuous.

For point-in-time input data, the spline curve is constrained to pass through the given data points. For interval total or average data, the definite integrals of the spline over the input intervals are constrained to equal the given interval totals.

For boundary constraints, the *not-a-knot* condition is used by default. This means that the first two spline pieces are constrained to be part of the same cubic curve, as are the last two pieces. Thus the spline used by PROC EXPAND by default is not the same as the commonly used natural spline, which uses zero second-derivative endpoint constraints. While DeBoor (1981) recommends the *not-a-knot* constraint for cubic spline interpolation, using this constraint can sometimes produce anomalous results at the ends of the interpolated series. PROC EXPAND provides options to specify other endpoint constraints for spline curves.

To specify endpoint constraints, use the following form of the METHOD= option.

METHOD=SPLINE(*constraint* **[,** *constraint***])**

The first constraint specification applies to the lower endpoint, and the second constraint specification applies to the upper endpoint. If only one constraint is specified, it applies to both the lower and upper endpoints.

The *constraint* specifications can have the following values:

NOTANOT
specifies the not-a-knot constraint. This is the default.

NATURAL
specifies the *natural spline* constraint. The second derivative of the spline curve is constrained to be zero at the endpoint.

SLOPE= value
specifies the first derivative of the spline curve at the endpoint.

CURVATURE= value
specifies the second derivative of the spline curve at the endpoint. Specifying CURVATURE=0 is equivalent to specifying the NATURAL option.

For example, to specify natural spline interpolation, use the following option in the CONVERT or PROC EXPAND statement:

```
   method=spline(natural)
```

For OBSERVED=BEGINNING, MIDDLE, and END series, the spline knots are placed at the beginning, middle, and end of each input interval, respectively. For total or averaged series, the spline knots are set at the start of the first interval, at the end of the last interval, and at the interval midpoints, except that there are no knots for the first two and last two midpoints.

Once the cubic spline curve is fit to the data, the spline is extended by adding linear segments at the beginning and end. These linear segments are used for extrapolating values beyond the range of the input data.

For point-in-time output series, the spline function is evaluated at the appropriate points. For interval total or average output series, the spline function is integrated over the output intervals.

The JOIN Method

The JOIN method fits a continuous curve to the data by connecting successive straight line segments. (This produces a linear spline.) For point-in-time data, the JOIN method connects successive nonmissing input values with straight lines. For interval total or average data, interval midpoints are used as the break points, and ordinates are chosen so that the integrals of the piecewise linear curve agree with the input totals.

For point-in-time output series, the JOIN function is evaluated at the appropriate points. For interval total or average output series, the JOIN function is integrated over the output intervals.

The STEP Method

The STEP method fits a discontinuous piecewise constant curve. For point-in-time input data, the resulting step function is equal to the most recent input value. For interval total or average data, the step function is equal to the average value for the interval.

For point-in-time output series, the step function is evaluated at the appropriate points. For interval total or average output series, the step function is integrated over the output intervals.

The AGGREGATE Method

The AGGREGATE method performs simple aggregation of time series without interpolation of missing values.

If the input data are totals or averages, the results are the sums or averages, respectively, of the input values for observations corresponding to the output observations. That is, if either TOTAL or AVERAGE is specified for the OBSERVED= option, the METHOD=AGGREGATE result is the sum or mean of the input values corresponding to the output observation. For example, suppose METHOD=AGGREGATE, FROM=MONTH, and TO=YEAR. For OBSERVED=TOTAL series, the result for each output year is the sum of the input values over the months of that year. If any input value is missing, the corresponding sum or mean is also a missing value.

If the input data are point-in-time values, the result value of each output observation equals the input value for a selected input observation determined by the OBSERVED= attribute. For example, suppose METHOD=AGGREGATE, FROM=MONTH, and TO=YEAR. For OBSERVED=BEGINNING series, January observations are selected as the annual values. For OBSERVED=MIDDLE series, July observations are selected as the annual values. For OBSERVED=END series, December observations are selected as the annual values. If the selected value is missing, the output annual value is missing.

The AGGREGATE method can be used only when the FROM= intervals are nested within the TO= intervals. For example, you can use METHOD=AGGREGATE when FROM=MONTH and TO=QTR because months are nested within quarters. You cannot use METHOD=AGGREGATE when FROM=WEEK and TO=QTR because weeks are not nested within quarters.

In addition, the AGGREGATE method cannot convert between point-in-time data and interval total or average data. Conversions between TOTAL and AVERAGE data are allowed, but conversions between BEGINNING, MIDDLE, and END are not.

Missing input values produce missing result values for METHOD=AGGREGATE. However, gaps in the sequence of input observations are not allowed. For example, if FROM=MONTH, you may have a missing value for a variable in an observation for a given February. But if an observation for January is followed by an observation for March, there is a gap in the data, and METHOD=AGGREGATE cannot be used.

When the AGGREGATE method is used, there is no interpolating curve, and therefore the EXTRAPOLATE option is not allowed.

METHOD=NONE

The option METHOD=NONE specifies that no interpolation be performed. This option is normally used in conjunction with the TRANSFORMIN= or TRANSFORMOUT= option.

When METHOD=NONE is specified, there is no difference between the TRANSFORMIN= and TRANSFORMOUT= options; if both are specified, the TRANSFORMIN= operations are performed first, followed by the TRANSFORMOUT= operations. TRANSFORM= can be used as an abbreviation for TRANSFORMIN=.

METHOD=NONE cannot be used when frequency conversion is specified.

Transformation Operations

The operations that can be used in the TRANSFORMIN= and TRANSFORMOUT= options are shown in Table 11.1. Operations are applied to each value of the series. Each value of the series is replaced by the result of the operation.

In Table 11.1, x_t or *x* represents the value of the series at a particular time period *t* before the transformation is applied, y_t represents the value of the result series, and N represents the total number of observations.

The notation [*n*] indicates that the argument *n* is optional; the default is 1. The notation *window* is used as the argument for the moving statistics operators, and it indicates that you can specify either an integer number of periods *n* or a list of *n* weights in parentheses. The notation *s* indicates the length of seasonality, and it is a required argument.

Table 11.1. Transformation Operations

Syntax	**Result**
+ *number*	adds the specified *number*: $x + number$
- *number*	subtracts the specified *number*: $x - number$
* *number*	multiplies by the specified *number*: $x * number$
& *number*	divides by the specified *number*: $x \& number$
ABS	absolute value: $\lvert x \rvert$
[] CD_I *s*	classical decomposition irregular component
CD_S *s*	classical decomposition seasonal component
CD_SA *s*	classical decomposition seasonally adjusted series
CD_TC *s*	classical decomposition trend-cycle component
CDA_I *s*	classical decomposition (additive) irregular component
CDA_S *s*	classical decomposition (additive) seasonal component
CDA_SA *s*	classical decomposition (additive) seasonally adjusted series
CEIL	smallest integer greater than or equal to *x*: $\mathrm{ceil(x)}$
CMOVAVE *window*	centered moving average
CMOVCSS *window*	centered moving corrected sum of squares
CMOVMAX *n*	centered moving maximum
CMOVMED *n*	centered moving median
CMOVMIN *n*	centered moving minimum
CMOVRANGE *n*	centered moving range
CMOVSTD *window*	centered moving standard deviation
CMOVSUM *n*	centered moving sum
CMOVUSS *window*	centered moving uncorrected sum of squares
CMOVVAR *window*	centered moving variance
CUAVE [*n*]	cumulative average
CUCSS [*n*]	cumulative corrected sum of squares
CUMAX [*n*]	cumulative maximum
CUMED [*n*]	cumulative median
CUMIN [*n*]	cumulative minimum
CURANGE [*n*]	cumulative range
CUSTD [*n*]	cumulative standard deviation
CUSUM [*n*]	moving sum
CUUSS [*n*]	cumulative uncorrected sum of squares
CUVAR [*n*]	cumulative variance
DIF [*n*]	lag *n* difference: $x_t - x_{t-n}$
EWMA *number*	exponentially weighted moving average of x with smoothing weight *number*, where $0 < number < 1$: $y_t = number\ \ x_t + (1 - number)y_{t-1}$. This operation is also called simple exponential smoothing.
EXP	exponential function: $\exp(x)$
FLOOR	largest integer less than or equal to *x*: $\mathrm{floor(x)}$
ILOGIT	inverse logistic function: $\frac{\exp(x)}{1+\exp(x)}$
LAG [*n*]	value of the series *n* periods earlier: x_{t-n}

Table 11.1. (continued)

Syntax	Result
LEAD [*n*]	value of the series *n* periods later: x_{t+n}
LOG	natural logarithm: $\log(x)$
LOGIT	logistic function: $\log(\frac{x}{1-x})$
MAX *number*	maximum of *x* and *number*: $\max(x, number)$
MIN *number*	minimum of *x* and *number*: $\min(x, number)$
> *number*	missing value if $x <= number$, else *x*
>= *number*	missing value if $x < number$, else *x*
= *number*	missing value if $x \neq number$, else *x*
∧= *number*	missing value if $x = number$, else *x*
< *number*	missing value if $x >= number$, else *x*
<= *number*	missing value if $x > number$, else *x*
MOVAVE *n*	moving average of *n* neighboring values: $\frac{1}{n}\sum_{j=0}^{n-1} x_{t-j}$
MOVAVE($w_1 \ldots w_n$)	weighted moving average of neighboring values: $(\sum_{j=1}^{n} w_j x_{t-j+1})/(\sum_{j=1}^{n} w_j)$
MOVAVE *window*	backward moving average
MOVCSS *window*	backward moving corrected sum of squares
MOVMAX *n*	backward moving maximum
MOVMED *n*	backward moving median
MOVMIN *n*	backward moving minimum
MOVRANGE *n*	backward moving range
MOVSTD *window*	backward moving standard deviation
MOVSUM *n*	backward moving sum
MOVUSS *window*	backward moving uncorrected sum of squares
MOVVAR *window*	backward moving variance
MISSONLY <MEAN>	indicates that the following moving time window statistic operator should replace only missing values with the moving statistic and should leave nonmissing values unchanged. If the option MEAN is specified, then missing values are replaced by the overall mean of the series.
NEG	changes the sign: $-x$
NOMISS	indicates that the following moving time window statistic operator should not allow missing values.
RECIPROCAL	reciprocal: $1/x$
REVERSE	reverse the series: x_{N-t}
SETMISS *number*	replaces missing values in the series with the number specified.
SIGN	-1, 0, or 1 as *x* is < 0, equals 0, or > 0 respectively
SQRT	square root: $\sqrt{x}$
SQUARE	square: x^2
SUM	cumulative sum: $\sum_{j=1}^{t} x_j$
SUM *n*	cumulative sum of *n*-period lags: $x_t + x_{t-n} + x_{t-2n} + \ldots$
TRIM *n*	sets x_t to missing a value if $t \leq n$ or $t \geq N - n + 1$.
TRIMLEFT *n*	sets x_t to missing a value if $t \leq n$.

Table 11.1. (continued)

Syntax	Result
TRIMRIGHT *n*	sets x_t to missing a value if $t \geq N - n + 1$.

Moving Time Window Operators

Some operators compute statistics for a set of values within a moving time window; these are called *moving time window operators*. There are backward and centered versions of these operators.

The centered moving time window operators are CMOVAVE, CMOVCSS, CMOVMAX, CMOVMED, CMOVMIN, CMOVRANGE, CMOVSTD, CMOVSUM, CMOVUSS, and CMOVVAR. These operators compute statistics of the n values x_i for observations $t - (n-1)/2 \leq i \leq t + (n-1)/2$.

The backward moving time window operators are MOVAVE, MOVCSS, MOVMAX, MOVMED, MOVMIN, MOVRANGE, MOVSTD, MOVSUM, MOVUSS, and MOVVAR. These operators compute statistics of the n values $x_t, x_{t-1}, \ldots, x_{t-n+1}$.

All the moving time window operators accept an argument n specifying the number of periods to include in the time window. For example, the following statement computes a five-period backward moving average of X.

```
convert x=y / transformout=( movave 5 );
```

In this example, the final result is $y_t = (x_t + x_{t-1} + x_{t-2} + x_{t-3} + x_{t-4})/5$.

The following statement computes a five-period centered moving average of X.

```
convert x=y / transformout=( cmovave 5 );
```

In this example, the final result is $y_t = (x_{t-2} + x_{t-1} + x_t + x_{t+1} + x_{t+2}/5$.

If the window with a centered moving time window operator is not an odd number, one more lagged value than lead value is included in the time window. For example, the result of the CMOVAVE 4 operator is $y_t = (x_{t-2} + x_{t-1} + x_t + x_{t+1})/4$.

You can compute a forward moving time window operation by combining a backward moving time window operator with the REVERSE operator. For example, the following statement computes a five-period forward moving average of X.

```
convert x=y / transformout=( reverse movave 5 reverse );
```

In this example, the final result is $y_t = (x_t + x_{t+1} + x_{t+2} + x_{t+3} + x_{t+4})/5$.

Some of the moving time window operators enable you to specify a list of weight values to compute weighted statistics. These are CMOVAVE, CMOVCSS, CMOVSTD, CMOVUSS, CMOVVAR, MOVAVE, MOVCSS, MOVSTD, MOVUSS, and MOVVAR.

To specify a weighted moving time window operator, enter the weight values in parentheses after the operator name. The window width n is equal to the number of weights that you specify; do not specify n.

For example, the following statement computes a weighted five-period centered moving average of X.

```
      convert x=y / transformout=( cmovave( .1 .2 .4 .2 .1 ) );
```

In this example, the final result is $y_t = .1x_{t-2} + .2x_{t-1} + .4x_t + .2x_{t+1} + .1x_{t+2}$.

The weight values must be greater than zero. If the weights do not sum to 1, the weights specified are divided by their sum to produce the weights used to compute the statistic.

At the beginning of the series, and at the end of the series for the centered operators, a complete time window is not available. The computation of the moving time window operators is adjusted for these boundary conditions as follows.

For backward moving window operators, the width of the time window is shortened at the beginning of the series. For example, the results of the MOVSUM 3 operator are

$$\begin{aligned}
y_1 &= x_1 \\
y_2 &= x_1 + x_2 \\
y_3 &= x_1 + x_2 + x_3 \\
y_4 &= x_2 + x_3 + x_4 \\
y_5 &= x_3 + x_4 + x_5 \\
&\ldots
\end{aligned}$$

For centered moving window operators, the width of the time window is shortened at the beginning and the end of the series due to unavailable observations. For example, the results of the CMOVSUM 5 operator are

$$\begin{aligned}
y_1 &= x_1 + x_2 + x_3 \\
y_2 &= x_1 + x_2 + x_3 + x_4 \\
y_3 &= x_1 + x_2 + x_3 + x_4 + x_5 \\
y_4 &= x_2 + x_3 + x_4 + x_5 + x_6 \\
&\ldots \\
y_{N-2} &= x_{N-4} + x_{N-3} + x_{N-2} + x_{N-1} + x_N \\
y_{N-1} &= x_{N-3} + x_{N-2} + x_{N-1} + x_N \\
y_N &= x_{N-2} + x_{N-1} + x_N
\end{aligned}$$

For weighted moving time window operators, the weights for the unavailable or unused observations are ignored and the remaining weights renormalized to sum to 1.

Cumulative Statistics Operators

Some operators compute cumulative statistics for a set of current and previous values of the series. The cumulative statistics operators are CUAVE, CUCSS, CUMAX, CUMED, CUMIN, CURANGE, CUSTD, CUSUM, CUUSS, and CUVAR. These operators compute statistics of the values $x_t, x_{t-n}, x_{t-2n}, \ldots, x_{t-in}$ for $t - in > 0$.

By default, the cumulative statistics operators compute the statistics from all previous values of the series, so that y_t is based on the set of values $x_1, x_2, \ldots, x_t$. For example, the following statement computes y_t as the cumulative sum of nonmissing x_i values for $i \leq t$.

```
convert x=y / transformout=( cusum );
```

You can also specify a lag increment argument n for the cumulative statistics operators. In this case, the statistic is computed from the current and every n^{th} previous value. For example, the following statement computes y_t as the cumulative sum of nonmissing x_i values for odd i when t is odd and for even i when t is even.

```
convert x=y / transformout=( cusum 2 );
```

The results of this example are

$$\begin{aligned}
y_1 &= x_1 \\
y_2 &= x_2 \\
y_3 &= x_1 + x_3 \\
y_4 &= x_2 + x_4 \\
y_5 &= x_1 + x_3 + x_5 \\
y_6 &= x_2 + x_4 + x_6 \\
&\ldots
\end{aligned}$$

Missing Values

You can truncate the length of the result series by using the TRIM, TRIMLEFT, and TRIMRIGHT operators to set values to missing at the beginning or end of the series.

You can use these functions to trim the results of moving time window operators so that the result series contains only values computed from a full width time window. For example, the following statements compute a centered five-period moving average of X, and they set to missing values at the ends of the series that are averages of fewer than five values.

```
convert x=y / transformout=( movave 5 trim 2 );
```

Normally, the moving time window and cumulative statistics operators ignore missing values and compute their results for the nonmissing values. When preceded by the NOMISS operator, these functions produce a missing result if any value within the time window is missing.

The NOMISS operator does not perform any calculations, but serves to modify the operation of the moving time window operator that follows it. The NOMISS operator has no effect unless it is followed by a moving time window operator.

For example, the following statement computes a five-period moving average of the variable X but produces a missing value when any of the five values are missing.

```
convert x=y / transformout=( nomiss movave 5 );
```

The following statement computes the cumulative sum of the variable X but produces a missing value for all periods after the first missing X value.

```
convert x=y / transformout=( nomiss cusum );
```

Similar to the NOMISS operator, the MISSONLY operator does not perform any calculations (unless followed by the MEAN option), but it serves to modify the operation of the moving time window operator that follows it. When preceded by the MISSONLY operator, these moving time window operators replace any missing values with the moving statistic and leave nonmissing values unchanged.

For example, the following statement replaces any missing values of the variable X with an exponentially weighted moving average of the past values of X and leaves nonmissing values unchanged. The missing values are then interpolated using an exponentially weighted moving average or simple exponential smoothing.

```
convert x=y / transformout=( missonly ewma 0.3 );
```

For example, the following statement replaces any missing values of the variable X with the overall mean of X.

```
convert x=y / transformout=( missonly mean );
```

You can use the SETMISS operator to replace missing values with a specified number. For example, the following statement replaces any missing values of the variable X with the number 8.77.

```
convert x=y / transformout=( setmiss 8.77 );
```

Classical Decomposition Operators

If y_t is a seasonal time series with s observations per season, *classical decomposition* methods "break down" a time series into four components: trend, cycle, seasonal, and irregular components. The trend and cycle components are often combined to form the trend-cycle component. There are two forms of decomposition: multiplicative and additive.

$$
\begin{aligned}
y_t &= TC_t S_t I_t \\
y_t &= TC_t + S_t + I_t
\end{aligned}
$$

where

TC_t is the trend-cycle component.

S_t is the seasonal component or seasonal factors that are periodic with period s and with mean one (multiplicative) or zero (additive).

I_t is the irregular or random component that is assumed to have mean one (multiplicative) or zero (additive).

The CD_TC operator computes the trend-cycle component for both the multiplicative and additive models. When s is odd, this operator computes an s-period centered moving average as follows:

$$TC_t = \sum_{k=-\lfloor s/2 \rfloor}^{\lfloor s/2 \rfloor} y_{t+k}/s$$

In the case $s = 5$, the CD_TC s operator is equivalent to the following CMOVAVE operator:

```
convert x=tc / transformout=( cmovave 5 trim 2 );
```

When s is even, the CD_TC s operator computes the average of two adjacent s-period centered moving averages as follows:

$$TC_t = \sum_{k=-\lfloor s/2 \rfloor}^{\lfloor s/2 \rfloor -1} (y_{t+k} + y_{t+1+k})/2s$$

In the case $s = 12$, the CD_TC s operator is equivalent to the following CMOVAVE operator:

```
convert x=tc / transformout=(cmovave 12 movave 2 trim 6);
```

The CD_S and CDA_S operators compute the seasonal components for the multiplicative and additive models, respectively. First, the trend-cycle component is computed as shown previously. Second, the seasonal-irregular component is computed by $SI_t = y_t / TC_t$ for the multiplicative model and by $SI_t = y_t - TC_t$ for the additive model. The seasonal component is obtained by averaging the seasonal-irregular component for each season.

$$S_{k+js} = \sum_{t=k \bmod s} \frac{SI_t}{n/s}$$

where $0 \leq j \leq n/s$ and $1 \leq k \leq s$. The seasonal components are normalized to sum to one (multiplicative) or zero (additive).

The CD_I and CDA_I operators compute the irregular component for the multiplicative and additive models respectively. First, the seasonal component is computed as shown previously. Next, the irregular component is determined from the seasonal-irregular and seasonal components as appropriate.

$$\begin{aligned} I_t &= SI_t / S_t \\ I_t &= SI_t - S_t \end{aligned}$$

The CD_SA and CDA_SA operators compute the seasonally adjusted time series for the multiplicative and additive models, respectively. After decomposition, the original time series can be seasonally adjusted as appropriate.

$$\begin{aligned} \tilde{y}_t &= y_t / S_t = TC_t I_t \\ \tilde{y}_t &= y_t - S_t = TC_t + I_t \end{aligned}$$

The following statements compute all the multiplicative classical decomposition components for the variable X for s=12.

```
convert x=tc / transformout=( cd_tc 12 );
convert x=s  / transformout=( cd_s  12 );
convert x=i  / transformout=( cd_i  12 );
convert x=sa / transformout=( cd_sa 12 );
```

The following statements compute all the additive classical decomposition components for the variable X for s=4.

```
convert x=tc / transformout=( cd_tc  4 );
convert x=s  / transformout=( cda_s  4 );
convert x=i  / transformout=( cda_i  4 );
convert x=sa / transformout=( cda_sa 4 );
```

OUT= Data Set

The OUT= output data set contains the following variables:

- the BY variables, if any
- an ID variable that identifies the time period for each output observation

- the result variables
- if no frequency conversion is performed (so that there is one output observation corresponding to each input observation), all the other variables in the input data set are copied to the output data set

The ID variable in the output data set is named as follows:

- If an ID statement is used, the new ID variable has the same name as the variable used in the ID statement.
- If no ID statement is used, but the FROM= option is used, then the name of the ID variable is either DATE or DATETIME, depending on whether the TO= option indicates SAS date or SAS datetime values.
- If neither an ID statement nor the TO= option is used, the ID variable is named TIME.

OUTEST= Data Set

The OUTEST= data set contains the coefficients of the spline curves fit to the input series. The OUTEST= data set is of interest if you want to verify the interpolating curve PROC EXPAND uses, or if you want to use this function in another context, (for example, in a SAS/IML program).

The OUTEST= data set contains the following variables:

- the BY variables, if any
- VARNAME, a character variable containing the name of the input variable to which the coefficients apply
- METHOD, a character variable containing the value of the METHOD= option used to fit the series
- OBSERVED, a character variable containing the first letter of the OBSERVED= option name for the input series
- the ID variable that contains the lower breakpoint (or “knot”) of the spline segment to which the coefficients apply. The ID variable has the same name as the variable used in the ID statement. If an ID statement is not used, but the FROM= option is used, then the name of the ID variable is DATE or DATETIME, depending on whether the FROM= option indicates SAS date or SAS datetime values. If neither an ID statement nor the FROM= option is used, the ID variable is named TIME.
- CONSTANT, the constant coefficient for the spline segment
- LINEAR, the linear coefficient for the spline segment
- QUAD, the quadratic coefficient for the spline segment
- CUBIC, the cubic coefficient for the spline segment

For each BY group, the OUTEST= data set contains observations for each polynomial segment of the spline curve fit to each input series. To obtain the observations defining the spline curve used for a series, select the observations where the value of VARNAME equals the name of the series.

The observations for a series in the OUTEST= data set encode the spline function fit to the series as follows. Let a_i, b_i, c_i, and d_i be the values of the variables CUBIC, QUAD, LINEAR, and CONSTANT, respectively, for the *i*th observation for the series. Let x_i be the value of the ID variable for the *i*th observation for the series. Let *n* be the number of observations in the OUTEST= data set for the series. The value of the spline function evaluated at a point *x* is

$$f(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i$$

where the segment number *i* is selected as follows:

$$i = \begin{cases} i & \text{such that } x_i \leq x \ < \ x_{i+1}, 1 \leq i \ < \ n \\ 1 & \text{if } x \ < \ x_1 \\ n & \text{if } x \geq x_n \end{cases}$$

In other words, if *x* is between the first and last ID values ($x_1 \leq x < x_n$), use the observation from the OUTEST= data set with the largest ID value less than or equal to *x*. If *x* is less than the first ID value x_1, then $i = 1$. If *x* is greater than or equal to the last ID value ($x \geq x_n$), then $i = n$.

For METHOD=JOIN, the curve is a linear spline, and the values of CUBIC and QUAD are 0. For METHOD=STEP, the curve is a constant spline, and the values of CUBIC, QUAD, and LINEAR are 0. For METHOD=AGGREGATE, no coefficients are output.

Examples

Example 11.1. Combining Monthly and Quarterly Data

This example combines monthly and quarterly data sets by interpolating monthly values for the quarterly series. The series are extracted from two small sample data sets stored in the SASHELP library. These data sets were contributed by Citicorp Data Base services and contain selected U.S. macro economic series.

The quarterly series gross domestic product (GDP) and implicit price deflator (GD) are extracted from SASHELP.CITIQTR. The monthly series industrial production index (IP) and unemployment rate (LHUR) are extracted from SASHELP.CITIMON. Only observations for the years 1990 and 1991 are selected. PROC EXPAND is then used to interpolate monthly estimates for the quarterly series, and the interpolated series are merged with the monthly data.

The following statements extract and print the quarterly data, shown in Output 11.1.1.

```
data qtrly;
   set sashelp.citiqtr;
   where date >= ’1jan1990’d &
         date <  ’1jan1992’d ;
   keep date gdp gd;
run;

title "Quarterly Data";
proc print data=qtrly;
run;
```

Output 11.1.1. Quarterly Data Set

```
                           Quarterly Data

                  Obs      DATE       GD         GDP

                   1     1990:1    111.100    5422.40
                   2     1990:2    112.300    5504.70
                   3     1990:3    113.600    5570.50
                   4     1990:4    114.500    5557.50
                   5     1991:1    115.900    5589.00
                   6     1991:2    116.800    5652.60
                   7     1991:3    117.400    5709.20
                   8     1991:4       .       5736.60
```

The following statements extract and print the monthly data, shown in Output 11.1.2.

```
data monthly;
   set sashelp.citimon;
   where date >= ’1jan1990’d &
         date <  ’1jan1992’d ;
   keep date ip lhur;
run;
```

```
title "Monthly Data";
proc print data=monthly;
run;
```

Output 11.1.2. Monthly Data Set

```
                          Monthly Data

                  Obs       DATE        IP         LHUR

                    1    JAN1990    107.500    5.30000
                    2    FEB1990    108.500    5.30000
                    3    MAR1990    108.900    5.20000
                    4    APR1990    108.800    5.40000
                    5    MAY1990    109.400    5.30000
                    6    JUN1990    110.100    5.20000
                    7    JUL1990    110.400    5.40000
                    8    AUG1990    110.500    5.60000
                    9    SEP1990    110.600    5.70000
                   10    OCT1990    109.900    5.80000
                   11    NOV1990    108.300    6.00000
                   12    DEC1990    107.200    6.10000
                   13    JAN1991    106.600    6.20000
                   14    FEB1991    105.700    6.50000
                   15    MAR1991    105.000    6.70000
                   16    APR1991    105.500    6.60000
                   17    MAY1991    106.400    6.80000
                   18    JUN1991    107.300    6.90000
                   19    JUL1991    108.100    6.80000
                   20    AUG1991    108.000    6.80000
                   21    SEP1991    108.400    6.80000
                   22    OCT1991    108.200    6.90000
                   23    NOV1991    108.000    6.90000
                   24    DEC1991    107.800    7.10000
```

The following statements interpolate monthly estimates for the quarterly series and merge the interpolated series with the monthly data. The resulting combined data set is then printed, as shown in Output 11.1.3.

```
proc expand data=qtrly out=temp from=qtr to=month;
   convert gdp gd / observed=average;
   id date;
run;

data combined;
   merge monthly temp;
   by date;
run;

title "Combined Data Set";
proc print data=combined;
run;
```

Output 11.1.3. Combined Data Set

```
                          Combined Data Set

         Obs       DATE        IP        LHUR       GDP         GD

           1    JAN1990    107.500    5.30000    5409.69    110.879
           2    FEB1990    108.500    5.30000    5417.67    111.048
           3    MAR1990    108.900    5.20000    5439.39    111.367
           4    APR1990    108.800    5.40000    5470.58    111.802
           5    MAY1990    109.400    5.30000    5505.35    112.297
           6    JUN1990    110.100    5.20000    5538.14    112.801
           7    JUL1990    110.400    5.40000    5563.38    113.264
           8    AUG1990    110.500    5.60000    5575.69    113.641
           9    SEP1990    110.600    5.70000    5572.49    113.905
          10    OCT1990    109.900    5.80000    5561.64    114.139
          11    NOV1990    108.300    6.00000    5553.83    114.451
          12    DEC1990    107.200    6.10000    5556.92    114.909
          13    JAN1991    106.600    6.20000    5570.06    115.452
          14    FEB1991    105.700    6.50000    5588.18    115.937
          15    MAR1991    105.000    6.70000    5608.68    116.314
          16    APR1991    105.500    6.60000    5630.81    116.600
          17    MAY1991    106.400    6.80000    5652.92    116.812
          18    JUN1991    107.300    6.90000    5674.06    116.988
          19    JUL1991    108.100    6.80000    5693.43    117.164
          20    AUG1991    108.000    6.80000    5710.54    117.380
          21    SEP1991    108.400    6.80000    5724.11    117.665
          22    OCT1991    108.200    6.90000    5733.65       .
          23    NOV1991    108.000    6.90000    5738.46       .
          24    DEC1991    107.800    7.10000    5737.75       .
```

Example 11.2. Interpolating Irregular Observations

This example shows the interpolation of a series of values measured at irregular points in time. The data are hypothetical. Assume that a series of randomly timed quality control inspections are made and defect rates for a process are measured. The problem is to produce two reports: estimates of monthly average defect rates for the months within the period covered by the samples, and a plot of the interpolated defect rate curve over time.

The following statements read and print the input data, as shown in Output 11.2.1.

```
data samples;
  input date : date9. defects @@;
  label defects = "Defects per 1000 units";
  format date date9.;
datalines;
13jan1992    55    27jan1992   73    19feb1992   84    8mar1992   69
27mar1992    66     5apr1992   77    29apr1992   63   11may1992   81
25may1992    89     7jun1992   94    23jun1992  105   11jul1992   97
15aug1992   112    29aug1992   89    10sep1992   77   27sep1992   82
;

title "Sampled Defect Rates";
proc print data=samples;
run;
```

Output 11.2.1. Measured Defect Rates

```
                          Sampled Defect Rates

                      Obs         date    defects

                        1    13JAN1992       55
                        2    27JAN1992       73
                        3    19FEB1992       84
                        4    08MAR1992       69
                        5    27MAR1992       66
                        6    05APR1992       77
                        7    29APR1992       63
                        8    11MAY1992       81
                        9    25MAY1992       89
                       10    07JUN1992       94
                       11    23JUN1992      105
                       12    11JUL1992       97
                       13    15AUG1992      112
                       14    29AUG1992       89
                       15    10SEP1992       77
                       16    27SEP1992       82
```

To compute the monthly estimates, use PROC EXPAND with the TO=MONTH option and specify OBSERVED=(BEGINNING,AVERAGE). The following statements interpolate the monthly estimates.

```
proc expand data=samples out=monthly to=month;
  id date;
  convert defects / observed=(beginning,average);
run;

title "Estimated Monthly Average Defect Rates";
proc print data=monthly;
run;
```

The results are printed in Output 11.2.2.

Output 11.2.2. Monthly Average Estimates

```
                 Estimated Monthly Average Defect Rates

                       Obs       date    defects

                        1     JAN1992     59.323
                        2     FEB1992     82.000
                        3     MAR1992     66.909
                        4     APR1992     70.205
                        5     MAY1992     82.762
                        6     JUN1992     99.701
                        7     JUL1992    101.564
                        8     AUG1992    105.491
                        9     SEP1992     79.206
```

To produce the plot, first use PROC EXPAND with TO=DAY to interpolate a full set of daily values, naming the interpolated series INTERPOL. Then merge this data set with the samples so you can plot both the measured and the interpolated values on the same graph. PROC GPLOT is used to plot the curve. The actual sample points are plotted with asterisks. The following statements interpolate and plot the defects rate curve.

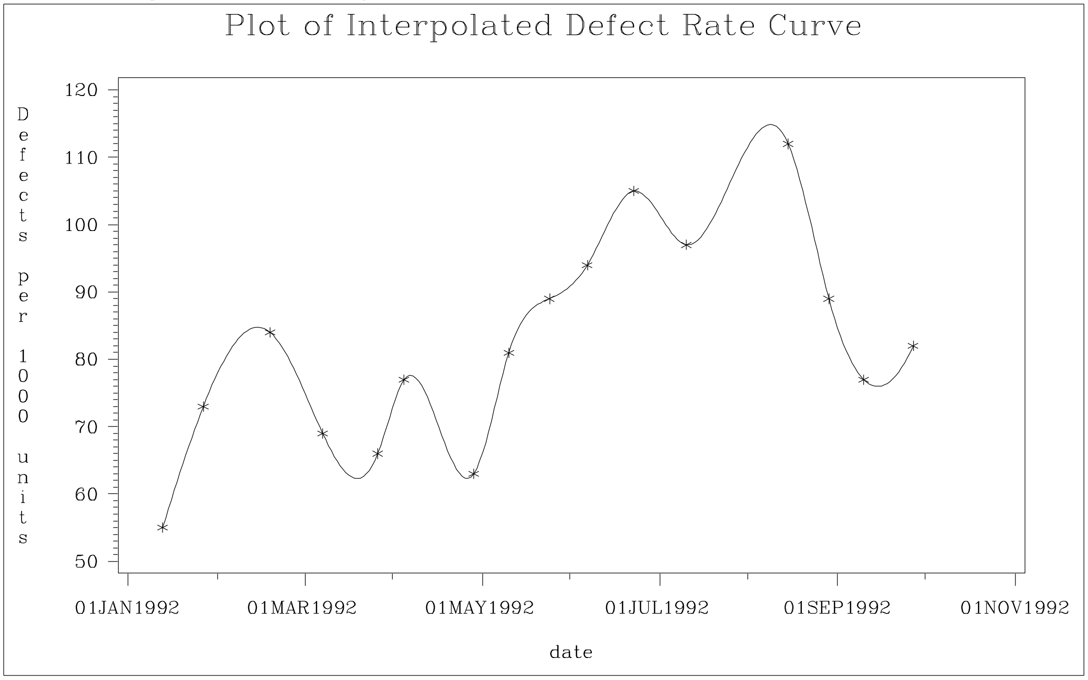
```
proc expand data=samples out=daily to=day;
  id date;
  convert defects = interpol;
run;

data daily;
  merge daily samples;
  by date;
run;

title "Plot of Interpolated Defect Rate Curve";
proc gplot data=daily;
   axis2 label=(a=-90 r=90 );
   symbol1 v=none i=join;
   symbol2 v=star i=none;
   plot interpol * date = 1 defects * date = 2  /
        vaxis=axis2 overlay;
run;
quit;
```

The plot is shown in Output 11.2.3.

Output 11.2.3. Interpolated Defects Rate Curve



Example 11.3. Using Transformations

This example shows the use of PROC EXPAND to perform various transformations of time series. The following statements read in monthly values for a variable X.

```
data test;
   input year qtr x;
   date = yyq( year, qtr );
   format date yyqc.;
datalines;
1989 3 5238
1989 4 5289
1990 1 5375
1990 2 5443
1990 3 5514
1990 4 5527
1991 1 5557
1991 2 5615
;
```

The following statements use PROC EXPAND to compute lags and leads and a 3-period moving average of the X series.

```
proc expand data=test out=out method=none;
   id date;
   convert x = x_lag2   / transform=(lag 2);
   convert x = x_lag1   / transform=(lag 1);
   convert x;
   convert x = x_lead1  / transform=(lead 1);
   convert x = x_lead2  / transform=(lead 2);
   convert x = x_movave / transform=(movave 3);
run;

title "Transformed Series";
proc print data=out;
run;
```

Because there are no missing values to interpolate and no frequency conversion, the METHOD=NONE option is used to prevent PROC EXPAND from performing unnecessary computations. Because no frequency conversion is done, all variables in the input data set are copied to the output data set. The CONVERT X; statement is included to control the position of X in the output data set. This statement can be omitted, in which case X is copied to the output data set following the new variables computed by PROC EXPAND.

The results are shown in Output 11.3.1.

Output 11.3.1. Output Data Set with Transformed Variables

```
                                   Transformed Series

    Obs    date    x_lag2  x_lag1    x    x_lead1  x_lead2  x_movave  year  qtr

     1   1989:3       .       .    5238    5289     5375    5238.00  1989   3
     2   1989:4       .    5238    5289    5375     5443    5263.50  1989   4
     3   1990:1    5238    5289    5375    5443     5514    5300.67  1990   1
     4   1990:2    5289    5375    5443    5514     5527    5369.00  1990   2
     5   1990:3    5375    5443    5514    5527     5557    5444.00  1990   3
     6   1990:4    5443    5514    5527    5557     5615    5494.67  1990   4
     7   1991:1    5514    5527    5557    5615        .    5532.67  1991   1
     8   1991:2    5527    5557    5615       .        .    5566.33  1991   2
```

References

DeBoor, Carl (1981), *A Practical Guide to Splines*, New York: Springer-Verlag.

Levenbach, H. and Cleary, J.P. (1984), *The Modern Forecaster*, Belmont, CA: Lifetime Learning Publications (a division of Wadsworth, Inc.), 129-133.

Makridakis, S. and Wheelwright, S.C. (1978), *Interactive Forecasting: Univariate and Multivariate Methods*, Second Edition, San Francisco: Holden-Day, 198-201.

Wheelwright, S.C. and Makridakis, S. (1973), *Forecasting Methods for Management*, Third Edition, New York: Whiley-Interscience, 123-133.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS/ ETS User's Guide, Version 8*, Cary, NC: SAS Institute Inc., 1999. 1546 pp.

SAS/ETS User's Guide, Version 8

ISBN 1–58025–489–6

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, October 1999

The Institute is a private company devoted to the support and further development of its software and related services.