SAS/ETS® 14.2 User's Guide
The EXPAND Procedure

This document is an individual chapter from *SAS/ETS® 14.2 User's Guide*.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2016. *SAS/ETS® 14.2 User's Guide*. Cary, NC: SAS Institute Inc.

SAS/ETS® 14.2 User's Guide

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

November 2016

Chapter 15
The EXPAND Procedure

Contents

Overview: EXPAND Procedure

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 transformations 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 values can be aggregated to produce an annual series, or quarterly estimates can be interpolated from 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. (For a complete description of SAS interval names, see Chapter 4, "Date Intervals, Formats, and Functions.") When the FROM= and TO= options are used to specify *from* and *to* intervals, 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 881. 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: EXPAND Procedure

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 want to add up monthly values to produce annual totals, but want 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 / method=aggregate observed=total;
   id date;
run;
```

This example assumes that the variables X, Y, and Z represent point-in-time values observed at the beginning of each month, and that the desired results are point-in-time values observed at the beginning of each year. (The default value of the OBSERVED= option is OBSERVED=(BEGINNING,BEGINNING).) The variables A, B, and C are assumed to represent monthly totals, and that the desired results are annual totals; therefore the option OBSERVED=TOTAL is specified. For more information about the OBSERVED= option, see the section "Specifying Observation Characteristics" on page 868.

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 then 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. The quarterly GDP data are interpolated as the total GDP over each month (OBSERVED=TOTAL), while the weekly INTEREST data are converted to average rates over each month (OBSERVED=AVERAGE).

```
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;
```

For further discussion of time series periodicity, time series dating, and time series interpolation, see Chapter 3, "Working with Time Series Data." For more information about the OBSERVED= option, see the section "Specifying Observation Characteristics" on page 868.

Interpolating Missing Values

To interpolate missing values in time series without converting the observation frequency, omit the TO= option from the PROC EXPAND statement. 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;
```

This example assumes that the variables X, Y, and Z represent point-in-time values observed at the beginning of each year. (The default value of the OBSERVED= option is OBSERVED=BEGINNING.) The variables A, B, and C are assumed to represent annual totals.

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 using a cubic spline function, assuming that the observations are at equally spaced points in time:

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

For more information, see the section "Missing Values" on page 890.

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 in the CONVERT statement. The section "Conversion Methods" on page 881 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. For more information, see the section "ID Statement" on page 876.

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 the series are specified with the OBSERVED= option in 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= options for these five observation characteristics are as follows:

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.

For more information, see the section "OBSERVED= Option" on page 880.

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 in 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;
```

Note that the transformations specified by the TRANSFORMIN= option are applied before the data are interpolated; the cubic spline curve or other interpolation method is fitted to transformed input data. The transformations specified by the TRANSFORMOUT= option are applied to interpolated values computed from the curves fit to the transformed input data.

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;
```

When more than one transformation is specified in the TRANSFORMIN= or TRANSFORMOUT= option, the transformations are applied in the order in which they are listed. Thus in the preceding example the complete input transformation is *logit(mshare/100)* (and not *logit(mshare)/100*) because the division operation is listed first in the TIN= option.

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. For a list of the operations supported, see Table 15.2.

Syntax: EXPAND Procedure

The following statements are available in the EXPAND procedure:

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

Functional Summary

The statements and options controlling the EXPAND procedure are summarized in Table 15.1.

Table 15.1 Functional Summary

Description	Statement	Option
Statements		
Specify options	PROC EXPAND	
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=
Extrapolate values before or after input series	PROC EXPAND	EXTRAPOLATE
Specify the output data set	PROC EXPAND	OUT=
Write interpolating functions to a data set	PROC EXPAND	OUTEST=
Input and Output Frequencies		
Control the alignment of SAS date values	PROC EXPAND	ALIGN=
Specify frequency conversion factor	PROC EXPAND	FACTOR=
Specify input frequency	PROC EXPAND	FROM=
Specify output frequency	PROC EXPAND	TO=
Interpolation Control Options		
Specify interpolation method for all series	PROC EXPAND	METHOD=
Specify interpolation method for series	CONVERT	METHOD=
Specify observation characteristics for series	PROC EXPAND	OBSERVED=
Specify observation characteristics for series	CONVERT	OBSERVED=
Specify transformations of the input series	CONVERT	TRANSFORMIN=
specify transformations of the output series	CONVERT	TRANSFORMOUT=
Graphical Output Control Options		
Specify graphical output	PROC EXPAND	PLOTS=

The following sections describe the PROC EXPAND statement and then describe the other statements in alphabetical order.

PROC EXPAND Statement

PROC EXPAND *options* **;**

You can specify the following *options*:

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 resulting time series. If OUT= is not specified, the data set is named using the DATA*n* convention. For more information, see the section "OUT= Data Set" on page 897.

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. For more information, see the section "OUTEST= Data Set" on page 898.

Options That Define Input and Output Frequencies

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.

FACTOR=*n*

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

In the FACTOR=() option, a comma can be used instead of a colon or the delimiter can be omitted. Thus FACTOR=(*n*, *m*) or FACTOR=(*n m*) is the same as FACTOR=(*n* : *m*).

The FACTOR= option cannot be used if the TO= option is used. The default value is FACTOR=(1:1). For more information, see the section "Frequency Conversion" on page 876.

FROM=*interval*
specifies the time interval between observations in the input data set. Examples of FROM= values are YEAR, QTR, MONTH, DAY, and HOUR. For a complete description and examples of interval specifications, see Chapter 4, "Date Intervals, Formats, and Functions."

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. For a description of interval specifications, see Chapter 4, "Date Intervals, Formats, and Functions."

Options to Control the Interpolation

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. For more information, see the section "Extrapolation" on page 879.

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 rarely used.

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 in the PROC EXPAND statement can be overridden for particular series by the METHOD= option in 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*. For more information about these methods, see the section "Conversion Methods" on page 881.

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 in the PROC EXPAND statement sets the default OBSERVED= value for subsequent CONVERT statements. For more information, see the sections "CONVERT Statement" on page 875 and "OBSERVED= Option" on page 880. The default is OBSERVED=BEGINNING.

Options to Control Graphical Output

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

specifies the graphical output desired. If the PLOTS= option is used, the specified graphical output is produced for each output variable that is specified by a CONVERT statement. By default, the EXPAND procedure produces no graphical output. The following PLOTS= *options* are available:

INPUT	plots the input series.
TRANSFORMIN	plots the transformed input series. The TRANSFORMIN= option must also be specified in the CONVERT statement.
CROSSINPUT	plots both the input series and the transformed input series on one plot with two Y axes. The input and transformed series are shown on separate scales. The TRANSFORMIN= option must also be specified in the CONVERT statement.

JOINTINPUT plots both the input series and the transformed input series on one plot with one Y axis. The input and transformed series are shown on the same scale. The TRANSFORMIN= option must also be specified in the CONVERT statement.

CONVERTED plots the converted series after input transformations and interpolation, but before any TRANSFORMOUT= transformations are applied. The METHOD= option must also be specified in the PROC EXPAND or CONVERT statements.

TRANSFORMOUT plots the transformed output series. The TRANSFORMOUT= option must also be specified in the CONVERT statement.

CROSSOUTPUT plots both the converted series and the transformed output series on one plot with two Y axes. The converted and transformed output series are shown on separate scales. The TRANSFORMOUT= option must also be specified in the CONVERT statement.

JOINTOUTPUT plots both the converted series and the transformed output series on one plot with one Y axis. The converted and transformed output series are shown on the same scale. The TRANSFORMOUT= option must also be specified in the CONVERT statement.

OUTPUT plots the series stored in the OUT= data set. The OUTPUT option does not require any options to be specified in the CONVERT statement.

ALL produces all plots except the joint and cross plots. PLOTS=ALL is the same as PLOTS=(INPUT TRANFORMIN CONVERTED TRANSFORMOUT).

The PLOTS= option produces results associated with each CONVERT statement output variable and the options listed in the PLOTS= specification. For more information, see the section "PLOTS= Option Details" on page 900.

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. Variable lists may be used only when no name is given for the output series.

For example, variable lists can be specified as follows:

```
convert y1-y25 / observed=(beginning,end);
convert x--a / observed=average;
convert x-numeric-a / observed=average;
```

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 method specified by the METHOD= option is applied to the input data series after applying any transformations specified by the TRANSFORMIN= option.) The methods supported are SPLINE, JOIN, STEP, AGGREGATE, and NONE. The METHOD= option specified in the PROC EXPAND statement can be overridden for particular series by the METHOD= option in 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*. For more information about these methods, see the section "Conversion Methods" on page 881.

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.

When only one value is specified, that value specifies both the *from-value* and the *to-value*. (That is, OBSERVED=*value* is equivalent to OBSERVED=(*value*, *value*).) If the OBSERVED= option is omitted from both the PROC EXPAND and the CONVERT statements, the default is OBSERVED=(BEGINNING, BEGINNING). For more information, see the section "OBSERVED= Option" on page 880.

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. For the operations that can be specified, see the section "Transformation Operations" on page 884. 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. For the operations that can be specified, see the section "Transformation Operations" on page 884. 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 same logic as the DATA step 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: EXPAND Procedure

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 interval nests within the output interval 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 from 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= interval is not nested within the TO= interval (as when aggregating from weekly to monthly), the results depend on an interpolation. The METHOD=AGGREGATE option always produces exact results, never an interpolation. However, this method can only be used if 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= values 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 value is specified that assumes the observations refer to whole intervals and also requires interval widths (TOTAL or AVERAGE), 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 is 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.

The date or datetime values for the ID variable for output observations are the first date or datetime of the TO= interval, unless the ALIGN= option is used to specify a different alignment. (For example, if TO=WEEK is specified, then the output dates are Sundays. If TO=WEEK.2 is specified, then the output date are Mondays.) For more information about interval specifications, see Chapter 4, "Date Intervals, Formats, and Functions."

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 are specified, then 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 EXTRAPOLATE option should be used with caution.

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 that the input values represent totals over the time intervals corresponding to the input observations, and 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 is specified, 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 is specified, 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 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 De Boor (1978) 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:

NOTAKNOT 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. The value specified can be any positive or negative number, but extreme values may produce unreasonable results.

CURVATURE=*value* specifies the second derivative of the spline curve at the endpoint. The value specified can be any positive or negative number, but extreme values may produce unreasonable results. 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. 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 are specified. 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 are specified. 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.

Alternate methods for aggregating or accumulating time series data are supported by the TIMESERIES procedure. For more information, see Chapter 39, “The TIMESERIES Procedure.”

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 15.2. Operations are applied to each value of the series. Each value of the series is replaced by the result of the operation.

In Table 15.2, 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_{optional} indicates that the argument n_{optional} is an optional integer; 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 a number of periods *n* (where *n* is an integer) or a list of *n* weights in parentheses. The internal maximum value of the number of periods *n* is clipped at the number of observations in the series. The notation *sequence* is used as the argument for the sequence operators, and it indicates that you must specify a sequence of numbers. The notation *s* indicates the length of seasonality, and it is a required argument.

Table 15.2 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: $\lvert x \rvert$
ADJUST	Indicates that the following moving window summation or product operator should be adjusted for window width
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 : $\text{ceil}(x)$
CMOVAVE *window*	Centered moving average
CMOVCSS *window*	Centered moving corrected sum of squares
CMOVGMEAN *window*	Centered moving geometric mean for *window* = number of periods, *n*:

Table 15.2 *continued*

Syntax	Result
	$(\prod_{j=j_{\min}}^{j_{\max}} x_{t+j})^{1/n}$ $j_{\min} = -(n + n \mod 2)/2 + 1$ $j_{\max} = (n - n \mod 2)/2$ for *window* = weight list, *w*: $(\prod_{j=j_{\min}}^{j_{\max}} x_{t+j}^{w_{j-j_{\min}}})^{1/\sum_{j=0}^{n-1} w_j}$
CMOVMAX *n*	Centered moving maximum
CMOVMED *n*	Centered moving median
CMOVMIN *n*	Centered moving minimum
CMOVPROD *window*	Centered moving product for *window* = number of periods, *n*: $\prod_{j=j_{\min}}^{j_{\max}} x_{t+j}$ for *window* = weight list, *w*: $(\prod_{j=j_{\min}}^{j_{\max}} x_{t+j}^{w_{j-j_{\min}}})^{1/\sum_{j=0}^{n-1} w_j}$
CMOVRANGE *n*	Centered moving range
CMOVRANK *n*	Centered moving rank
CMOVSTD *window*	Centered moving standard deviation
CMOVSUM *n*	Centered moving sum
CMOVTVALUE *window*	Centered moving *t* value
CMOVUSS *window*	Centered moving uncorrected sum of squares
CMOVVAR *window*	Centered moving variance
CUAVE n_{optional}	Cumulative average
CUCSS n_{optional}	Cumulative corrected sum of squares
CUGMEAN n_{optional}	Cumulative geometric mean
CUMAX n_{optional}	Cumulative maximum
CUMED n_{optional}	Cumulative median
CUMIN n_{optional}	Cumulative minimum
CUPROD n_{optional}	Cumulative product
CURANK n_{optional}	Cumulative rank
CURANGE n_{optional}	Cumulative range
CUSTD n_{optional}	Cumulative standard deviation
CUSUM n_{optional}	Cumulative sum
CUTVALUE n_{optional}	Cumulative *t* value
CUUSS n_{optional}	Cumulative uncorrected sum of squares
CUVAR n_{optional}	Cumulative variance
DIF n_{optional}	Span *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: $\exp(x)$
FDIF *d*	Fractional difference with difference order *d* where $0 < d < 0.5$
FLOOR	Largest integer less than or equal to x : $\text{floor}(x)$

Table 15.2 *continued*

Syntax	Result
FSUM *d*	Fractional summation with summation order *d* where $0 < d < 0.5$
HP_T *lambda*	Hodrick-Prescott Filter trend component where *lambda* is the nonnegative filter parameter
HP_C *lambda*	Hodrick-Prescott Filter cycle component where *lambda* is the nonnegative filter parameter
ILOGIT	Inverse logistic function: $\frac{\exp(x)}{1+\exp(x)}$
LAG n_{optional}	Value of the series *n* periods earlier: x_{t-n}
LEAD n_{optional}	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*	Backward moving average of *n* neighboring values: $\frac{1}{n}\sum_{j=0}^{n-1} x_{t-j}$
MOVAVE *window*	Backward weighted moving average of neighboring values: $(\sum_{j=1}^{n} w_j x_{t-n+j})/(\sum_{j=1}^{n} w_j)$
MOVCSS *window*	Backward moving corrected sum of squares
MOVGMEAN *window*	Backward moving geometric mean for *window* = number of periods, *n*: $(\prod_{j=1}^{n} x_{t-n+j})^{1/n}$ for *window* = weight list, *w*: $(\prod_{j=1}^{n} x_{t-n+j}^{w_j})^{1/\sum_{j=1}^{n} w_j}$
MOVMAX *n*	Backward moving maximum
MOVMED *n*	Backward moving median
MOVMIN *n*	Backward moving minimum
MOVPROD *window*	Backward moving product for *window* = number of periods, *n*: $\prod_{j=1}^{n} x_{t-n+j}$ for *window* = weight list, *w*: $(\prod_{j=1}^{n} x_{t-n+j}^{w_j})^{1/\sum_{j=1}^{n} w_j}$
MOVRANGE *n*	Backward moving range
MOVRANK *n*	Backward moving rank
MOVSTD *window*	Backward moving weighted standard deviation: $\sqrt{\frac{1}{n-1}\sum_{j=1}^{n} w_i (x_j - \overline{x}_w)^2}$
MOVSUM *n*	Backward moving sum
MOVTVALUE *window*	Backward moving *t* value

Table 15.2 *continued*

Syntax	Result
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
PCTDIF n	Percent difference of the current value and lag n
PCTSUM n	Percent summation of the current value and cumulative sum n-lag periods
RATIO n	Ratio of current value to lag n
RECIPROCAL	Reciprocal: $1/x$
REVERSE	Reverses the series: x_{N-t}
SCALE n_1 n_2	Scales the series between n_1 and n_2
SEQADD *sequence*	Adds sequence values to series
SEQDIV *sequence*	Divides the series by sequence values
SEQMINUS *sequence*	Subtracts sequence values to series
SEQMULT *sequence*	Multiplies the series by sequence values
SET (n_1 n_2)	Sets all values of n_1 to n_2
SETEMBEDDED (n_1 n_2)	Sets embedded values of n_1 to n_2
SETLEFT (n_1 n_2)	Sets beginning values of n_1 to n_2
SETMISS *number*	Replaces missing values in the series with the number specified
SETRIGHT (n_1 n_2)	Sets ending values of n_1 to n_2
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 multiples of n-period lags: $x_t + x_{t-n} + x_{t-2n} + \cdots$
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 $t \leq n$
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 centered and backward versions of these operators.

The centered moving time window operators are CMOVAVE, CMOVCSS, CMOVGMEAN, CMOVMAX, CMOVMED, CMOVMIN, CMOVPROD, CMOVRANGE, CMOVRANK, CMOVSTD, CMOVSUM,

CMOVTVALUE, CMOVUSS, and CMOVVAR. These operators compute statistics of the n values x_i for observations $t - (n + n \mod 2)/2 + 1 \leq i \leq t + (n - n \mod 2)/2$

The backward moving time window operators are MOVAVE, MOVCSS, MOVGMEAN, MOVMAX, MOVMED, MOVMIN, MOVPROD, MOVRANGE, MOVRANK, MOVSTD, MOVSUM, MOVTVALUE, 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 resulting transformation 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 resulting transformation 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 lead value than lag value is included in the time window. For example, the result of the CMOVAVE 4 operator is

$$y_t = (x_{t-1} + x_t + x_{t+1} + x_{t+2})/4$$

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 resulting transformation 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, CMOVGMEAN, CMOVPROD, CMOVSTD, CMOVTVALUE, CMOVUSS, CMOVVAR, MOVAVE, MOVCSS, MOVGMEAN, MOVPROD, MOVSTD, MOVTVALUE, 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 resulting transformation 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.

A complete time window is not available at the beginning of the series. For the centered operators a complete window is also not available at the end of the series. 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

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

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

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

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.

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 $x_t, x_{t-1}, \ldots, x_1$. 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 specify a lag increment argument n for the cumulative statistics operators. In this case, the statistic is computed from the current and every nth previous value. When n is specified these operators compute statistics of the values $x_t, x_{t-n}, x_{t-2n}, \ldots, x_{t-in}$ for $t - in > 0$.

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=( cmovave 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 interpolated using the specified exponentially weighted moving average. (This is also called simple exponential smoothing.)

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

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 x_t is a seasonal time series with s observations per season, *classical decomposition* methods "break down" the 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 basic forms of classical decomposition: multiplicative and additive, which are show below.

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

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)

For multiplicative decomposition, all of the x_t values should be positive.

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:

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

For example, in the case where s=5, the CD_TC s operator

```
convert x=tc / transformout=( cd_tc 5 );
```

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:

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

For example, in the case where s=12, the CD_TC s operator

```
convert x=tc / transformout=( cd_tc 12 );
```

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 $SI_t = x_t / TC_t$ for the multiplicative model and by $SI_t = x_t - TC_t$ 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.

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

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 );
```

The X12 and X11 procedures provide other methods for seasonal decomposition. See Chapter 44, "The X12 Procedure," and Chapter 43, "The X11 Procedure."

Fractional Operators

For fractional operators, the parameter, d, represents the order of fractional differencing. Fractional summation is the inverse operation of fractional differencing.

Examples of Usage

Suppose that X is a fractionally integrated time series variable of order d=0.25. Fractionally differencing X forms a time series variable Y, which is not integrated.

```
convert x=y / transformout=(fdif 0.25);
```

Suppose that Z is a non-integrated time series variable. Fractionally summing Z forms a time series W, which is fractionally integrated of order $d = 0.25$.

```
convert z=w / transformout=(fsum 0.25);
```

Moving Rank Operators

For the rank operators, the ranks are computed based on the current value with respect to the cumulative, centered, or moving window values. If the current value is missing, the transformed current value is set to missing. If the NOMISS option was previously specified and if any missing values are present in the moving window, the transformed current value is set to missing. Otherwise, redundant values from the moving window are removed and the rank of the current value is computed among the unique values of the moving window.

Examples of Usage

The trades of a particular security are recorded for each weekday in a variable named PRICE. Given the historical daily trades, the ranking of the price of this security for each trading day, considering its entire past history, can be computed as follows:

```
convert price=history / transformout=( curank );
```

The ranking of the price of this security for each trading day considering the previous week's history can be computed as follows:

```
convert price=lastweek / transformout=( movrank 5 );
```

The ranking of the price of this security for each trading day considering the previous two week's history can be computed as follows:

```
convert price=twoweek / transformout=( movrank 10 );
```

Moving Product and Geometric Mean Operators

For the product and geometric mean operators, the current transformed value is computed based on the (weighted) product of the cumulative, centered, or moving window values. If missing values are present in the moving window and the NOMISS operator is previously specified, the current transformed value is set to missing. Otherwise, the current transformed value is set to the product of the nonmissing values within the moving window. If a geometric mean operator is specified for a window of size n, the nth root of the product is taken. In cases where weights are specified explicitly, both the product and geometric mean operators normalize these exponents so that they sum to one.

Examples of Usage

The interest rates for a savings account are recorded for each month in the data set variable RATES. The cumulative interest rate for each month considering the entire account past history can be computed as follows:

```
convert rates=history / transformout=( + 1 cuprod - 1);
```

The interest rate for each quarter considering the previous quarter's history can be computed as follows:

```
convert rates=lastqtr / transformout=( + 1 movprod 3 - 1);
```

The average interest rate for the previous quarter's history can be computed as follows:

```
convert rates=lastqtr / transformout=( + 1 movprod (1 1 1) - 1);
```

Sequence Operators

For the sequence operators, the sequence values are used to compute the transformed values from the original values in a sequential fashion. You can add to or subtract from the original series or you can multiply or divide by the sequence values. The first sequence value is applied to the first observation of the series, the second sequence value is applied to the second observation of the series, and so on until the end of the sequence is reached. At this point, the first sequence value is applied to the next observation of the series and the second sequence value on the next observation and so on.

Let $v_1, \ldots, v_m$ be the sequence values and let $x_t, t = 1, \ldots N$, be the original time series. The transformed

series, y_t, is computed as follows:

$$
\begin{aligned}
y_1 &= x_1 \; op \; v_1 \\
y_2 &= x_2 \; op \; v_2 \\
&\dots \\
y_m &= x_m \; op \; v_m \\
y_{m+1} &= x_{m+1} \; op \; v_1 \\
y_{m+2} &= x_{m+2} \; op \; v_2 \\
&\dots \\
y_{2m} &= x_{2m} \; op \; v_m \\
y_{2m+1} &= x_{2m+1} \; op \; v1 \\
y_{2m+2} &= x_{2m+2} \; op \; v2 \\
&\dots
\end{aligned}
$$

where $op = +, -, *,$ or $/$.

Examples of Usage

The multiplicative seasonal indices are 0.9, 1.2. 0.8, and 1.1 for the four quarters. Let SEASADJ be a quarterly time series variable that has been seasonally adjusted in a multiplicative fashion. To restore the seasonality to SEASADJ, use the following transformation:

```
convert seasadj=seasonal /
        transformout=(seqmult (0.9 1.2 0.8 1.1));
```

The additive seasonal indices are 4.4, –1.1, –2.1, and –1.2 for the four quarters. Let SEASADJ be a quarterly time series variable that has been seasonally adjusted in additive fashion. To restore the seasonality to SEASADJ, use the following transformation:

```
convert seasadj=seasonal /
        transformout=(seqadd (4.4 -1.1 -2.1 -1.2));
```

Set Operators

For the set operators, the first parameter, n_1, represents the value to be replaced and the second parameter, n_2, represents the replacement value. The replacement can be localized to the beginning, middle, or end of the series.

Examples of Usage

Suppose that a store opened recently and that the sales history is stored in a database that does not recognize missing values. Even though demand may have existed prior to the stores opening, this database assigns the value of zero. Modeling the sales history may be problematic because the sales history is mostly zero. To compensate for this deficiency, the leading zero values should be set to missing with the remaining zero values unchanged (representing no demand).

```
convert sales=demand / transformout=(setleft (0 .));
```

Likewise, suppose a store is closed recently. The demand might still be present; hence, a recorded value of zero does not accurately reflect actual demand.

```
convert sales=demand / transformout=(setright (0 .));
```

Scale Operator

For the scale operator, the first parameter, n_1, represents the value associated with the minimum value ($x_{\min}$) and the second parameter, n_2, represents the value associated with the maximum value ($x_{\max}$) of the original series (x_t). The scale operator rescales the original data to be between the parameters n_1 and n_2 as follows:

$$y_t = ((n_2 - n_1)/(x_{\max} - x_{\min}))(x_t - x_{\min}) + n_1$$

Examples of Usage

Suppose that two new product sales histories are stored in the variables X and Y and you want to determine their adoption rates. In order to compare their adoption histories, the variables must be scaled for comparison.

```
convert x=w / transformout=(scale 0 1);
convert y=z / transformout=(scale 0 1);
```

Adjust Operator

For the moving summation and product window operators, the window widths at the beginning and end of the series are smaller than those in the middle of the series. Likewise, if there are embedded missing values, the window width is smaller than specified. When preceded by the ADJUST operator, the moving summation (MOVSUM CMOVSUM) and moving product operators (MOVPROD CMOVPROD) are adjusted by the window width.

For example, suppose the variable X has 10 values, and the moving summation operator of width 3 is applied to X to create the variable Y with window width adjustment and the variable Z without adjustment.

```
convert x=y / transformout=(adjust movsum 3);
convert x=z / transformout=(movsum 3);
```

The preceding transformations result in the following relationship between Y and Z: $y_1 = 3z_1$, $y_2 = \frac{3}{2}z_2$, $y_t = z_t$ for $t > 2$ because the first two window widths are smaller than 3.

For example, suppose the variable X has 10 values and the moving multiplicative operator of width 3 is applied to X to create the variable Y with window width adjustment and the variable Z without adjustment.

```
convert x=y / transformout=(adjust movprod 3);
convert x=z / transformout=(movprod 3);
```

The preceding transformations result in the following: $y_1 = z_1^3$, $y_2 = z_2^{3/2}$, $y_t = z_t$ for $t > 2$ because the first two window widths are smaller than 3.

Moving T-Value Operators

The moving *t*-value operators (CUTVALUE, MOVTVALUE, CMOVTVALUE) compute the *t*-value of the cumulative series or moving window. They can be viewed as combinations of the moving average (CUAVE, MOVAVE, CMOVAVE) and the moving standard deviation (CUSTD, MOVSTD, CMOVSTD), respectively.

Percent Operators

The percentage operators compute the percent summation and the percent difference of the current value and the $\mathrm{lag}(n)$. The percent summation operator (PCTSUM) computes $y_t = 100x_t/\mathrm{cusum}(x_{t-n})$. If any of the values of the preceding equation are missing or the cumulative summation is zero, the result is set to missing. The percent difference operator (PCTDIF) computes $y_t = 100(x_t - x_{t-n})/x_{t-n}$. If any of the values of the preceding equation are missing or the lag value is zero, the result is set to missing.

For example, suppose the variable X contains the series. The percent summation of lag 4 is applied to X to create the variable Y. The percent difference of lag 4 is applied to X to create the variable Z.

```
convert x=y / transformout=(pctsum 4);
convert x=z / transformout=(pctdif 4);
```

Ratio Operators

The ratio operator computes the ratio of the current value and the $\mathrm{lag}(n)$ value. The ratio operator (RATIO) computes $y_t = x_t/x_{t-n}$. If any of the values of the preceding equation are missing or the lag value is zero, the result is set to missing.

For example, suppose the variable X contains the series. The ratio of the current value and the lag 4 value of X is assigned to the variable Y. The percent ratio of the current value and lag 4 value of X is assigned to the variable Z.

```
convert x=y / transformout=(ratio 4);
convert x=z / transformout=(ratio 4 * 100);
```

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 ith observation for the series. Let x_i be the value of the ID variable for the ith 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:

$$i = \begin{cases} i & x_i \le x < x_{i+1}, 1 \le i < n \\ 1 & x < x_1 \\ n & x \ge x_n \end{cases}$$

In other words, if x is between the first and last ID values ($x_1 \le x < x_n$), 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 ($x \ge x_n$), 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.

ODS Graphics

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

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

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

This section describes the use of ODS for creating graphics with the EXPAND procedure. To request these graphs, you must specify the PLOTS= option in the PROC EXPAND statement.

ODS Graph Names

PROC EXPAND assigns a name to each graph it creates using ODS. You can use these names to reference the graphs when using ODS. The names are listed in Table 15.3.

Table 15.3 ODS Graphics Produced by PROC EXPAND

ODS Graph Name	Plot Description	PLOTS= Options
ConvertedSeriesPlot	Converted Series Plot	CONVERTED OUTPUT ALL
CrossInputSeriesPlot	Cross Input Series Plot	CROSSINPUT
CrossOutputSeriesPlot	Cross Output Series Plot	CROSSOUTPUT
InputSeriesPlot	Input Series Plot	INPUT JOINTINPUT ALL
JointInputSeriesPlot	Joint Input Series Plot	JOINTINPUT
JointOutputSeriesPlot	Joint Output Series Plot	JOINTOUTPUT
OutputSeriesPlot	Output Series Plot	SERIES | OUTPUT
TransformedInputSeriesPlot	Transformed Input Series Plot	TRANSFORMIN OUTPUT ALL
TransformedOutputSeriesPlot	Transformed Output Series Plot	TRANSFORMOUT OUTPUT ALL

PLOTS= Option Details

Some plots are produced for a series only if the relevant options are also specified. For example, if PLOTS=TRANSFORMIN is specified, then the TRANSFORMIN plot is not produced for a variable unless the TRANSFORMIN= option is specified in a CONVERT statement for that variable. The PLOTS=TRANSFORMIN option plots the series after the input transformation (TRANSFORMIN= option) is applied.

The PLOTS=CONVERTED option plots the series after the input transformation (TRANSFORMIN= option) is applied and after frequency conversion (METHOD= option). If there is no frequency conversion for an output variable, the converted series plot is not produced.

The PLOTS=TRANSFORMOUT option plots the series after the output transformation (TRANSFORMOUT= option) is applied. If the TRANFORMOUT= option is not specified in the CONVERT statement for an output variable, the output transformation plot is not produced.

The PLOTS=OUTPUT option plots the series after it has undergone input transformation (TRANSFORMIN= option), frequency conversion (METHOD= option), and output transformation (TRANSFORMOUT= option) if these CONVERT statement options were specified.

Cross and Joint Plots

The PLOTS= option values CROSSINPUT and CROSSOUTPUT produce graphs that overlay plots of two series by using two Y axes and with each of the two plots shown at a separate scale. These plots are called cross plots.

The PLOTS= option values JOINTINPUT and JOINTOUTPUT produce graphs that overlay plots of two series by using a single Y axis and with both of the plots shown on the same scale. These plots are called joint plots. The joint graphics options (PLOTS=JOINTINPUT or PLOTS=JOINTOUTPUT) plot the (input or converted) series and the transformed series on the same scale; therefore if the transformation changes, the range of the series these plots might be hard to visualize.

The PLOTS=CROSSINPUT option plots both the input series and the series after the input transformation (TRANSFORMIN= option) is applied. The left vertical axis refers to the input series, while the right vertical axis refers to the series after the transformation. If the TRANFORMIN= option is not specified in the CONVERT statement for an output variable, then the cross input plot is not produced for that variable.

The PLOTS=JOINTINPUT option jointly plots both the input series and the series after the input transformation (TRANSFORMIN= option) is applied. If the TRANSFORMIN= option is not specified in the CONVERT statement for an output variable, then the joint input plot is not produced for that variable.

The PLOTS=CROSSOUTPUT option plots both the converted series and the converted series after the output transformation (TRANSFORMOUT= option) is applied. The left vertical axis refers to the input series, while the right vertical axis refers to the series after the transformation. If the TRANSFORMOUT= option is not specified in the CONVERT statement for an output variable, then the cross output plot is not produced for that variable.

The PLOTS=JOINTOUTPUT option jointly plots both the converted series and the converted series after the output transformation (TRANSFORMOUT= option) is applied. If the TRANSFORMOUT= option is not specified in the CONVERT statement for an output variable, then the joint output plot is not produced for that variable.

Requesting All Plots

The PLOTS=ALL option is a convenient way to specify all the plots except the OUTPUT plots and the joint and cross plots. The option PLOTS=(ALL OUTPUT JOINTINPUT JOINTOUTPUT CROSSINPUT CROSSOUTPUT) requests that all possible plots be produced.

Examples: EXPAND Procedure

Example 15.1: 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 15.1.1:

```
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 15.1.1 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 15.1.2:

```
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 15.1.2 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 15.1.3.

```
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 15.1.3 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 15.2: Illustration of ODS Graphics

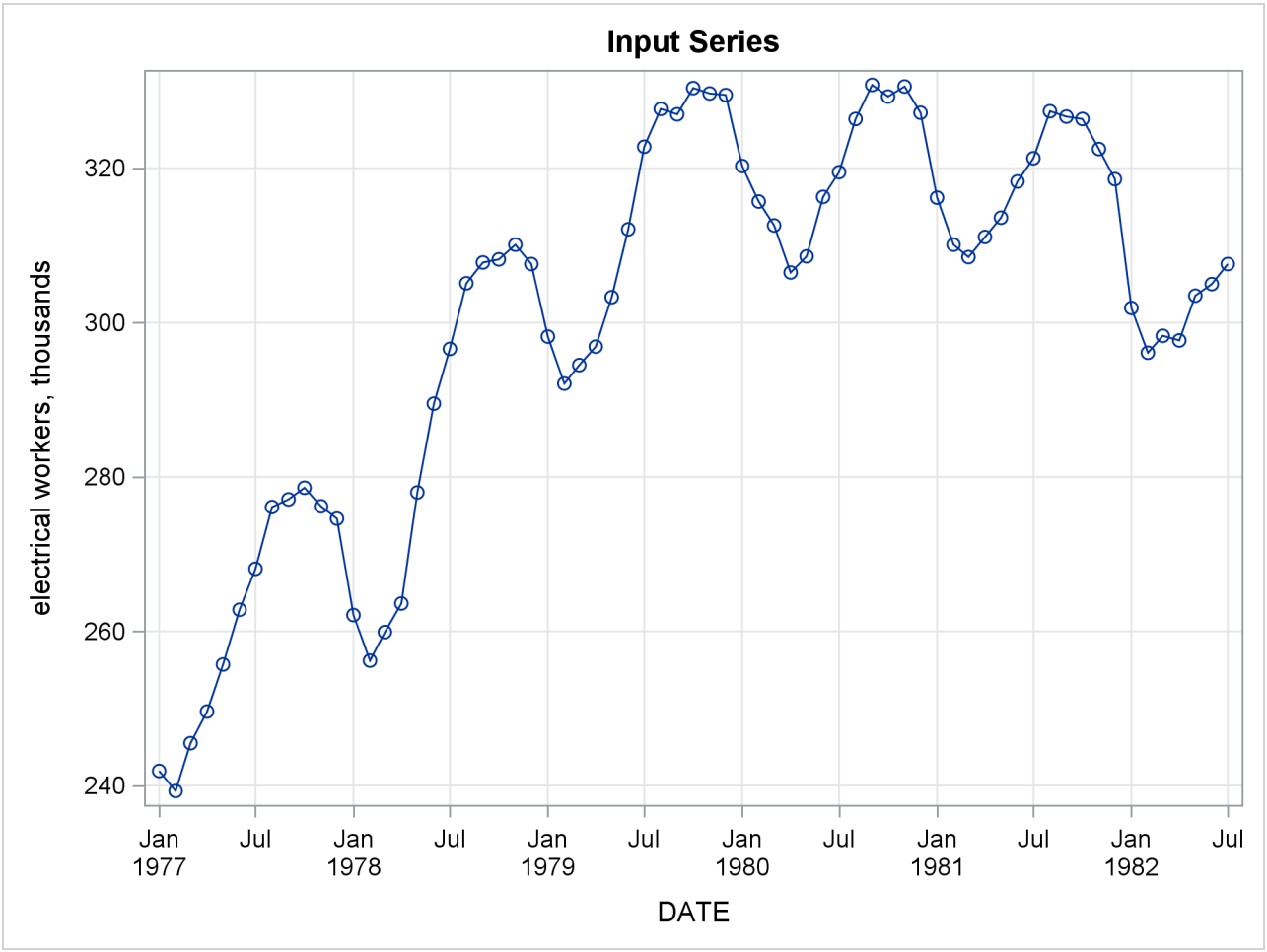
This example illustrates the use of ODS graphics with PROC EXPAND.

The graphical displays are requested by specifying the PLOTS= option in the PROC EXPAND statement. For information about the graphics available in the EXPAND procedure, see the section "ODS Graphics" on page 899.

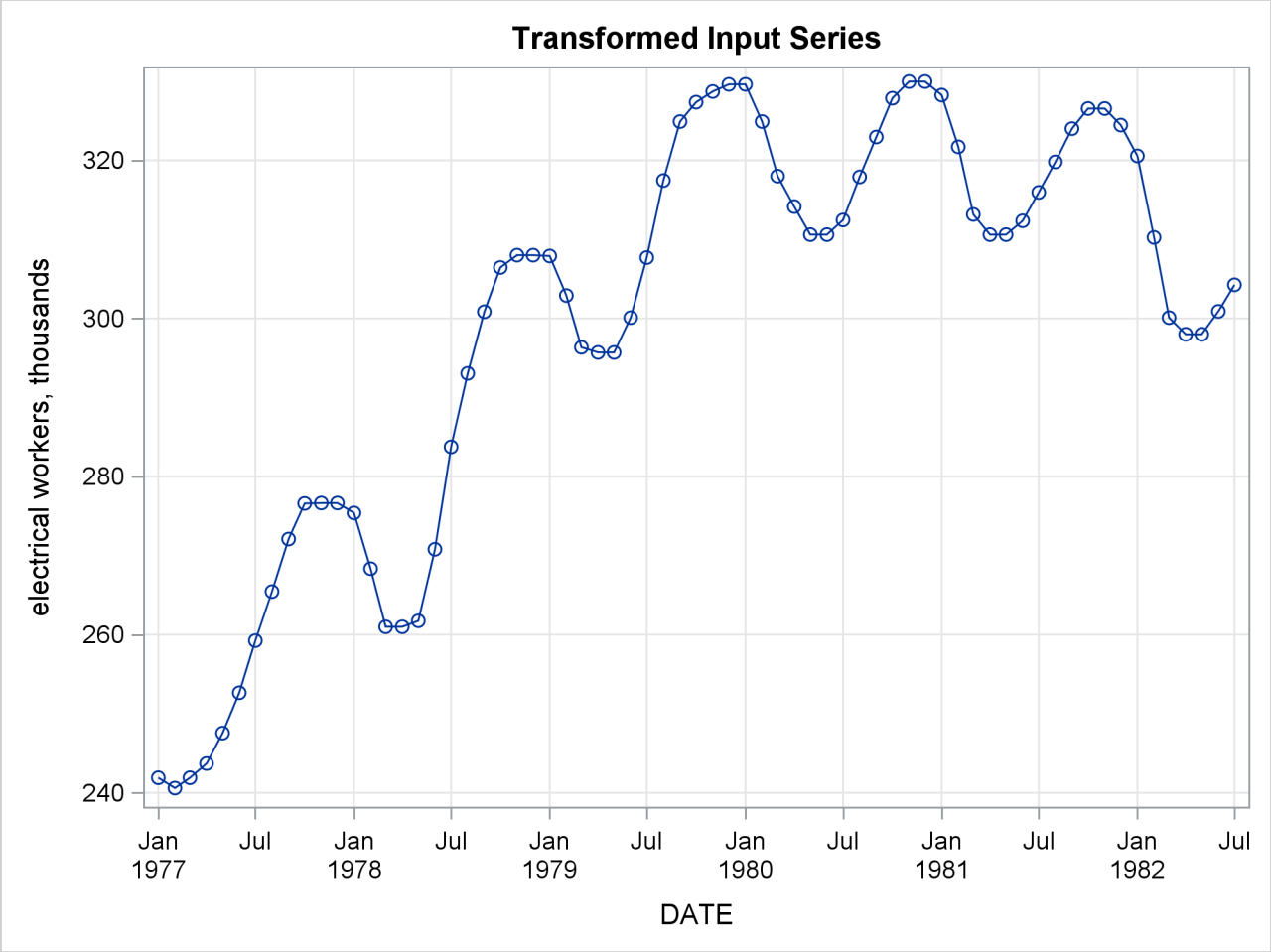
The following statements utilize the SASHELP.WORKERS data set to convert the time series of electrical workers from monthly to quarterly frequency and display ODS graphics plots. The PLOTS=ALL option is specified to request the plots of the input series, the transformed input series, the converted series, and the transformed output series. Figure 15.2.1 through Figure 15.2.4 show these plots.

```
proc expand data=sashelp.workers out=out
            from=month to=qtr
            plots=all;
   id date;
   convert electric=eout / method=spline
                           transformin=(movmed 4)
                           transformout=(movave 3);
run;
```

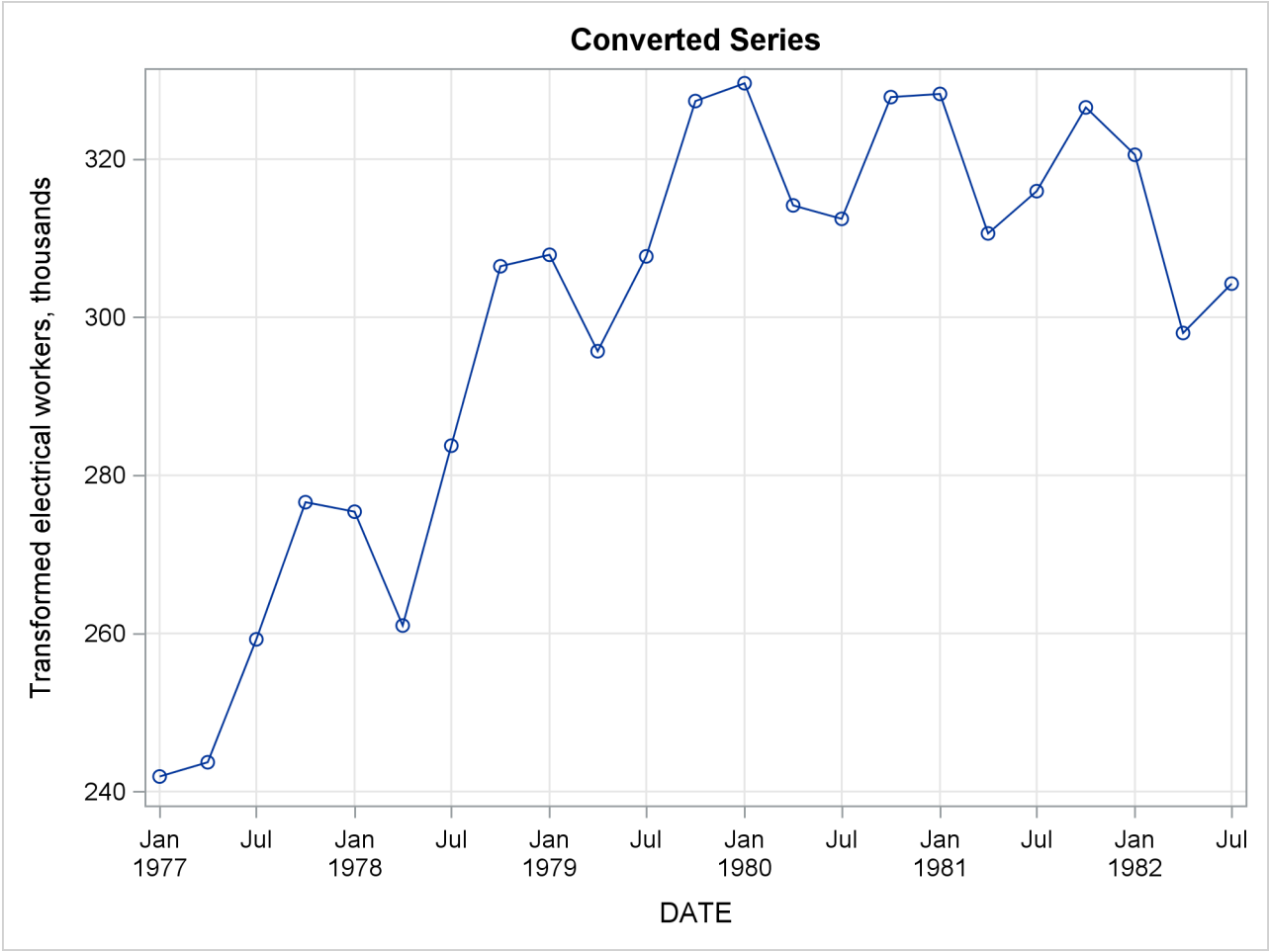
Output 15.2.1 Input Series Plot



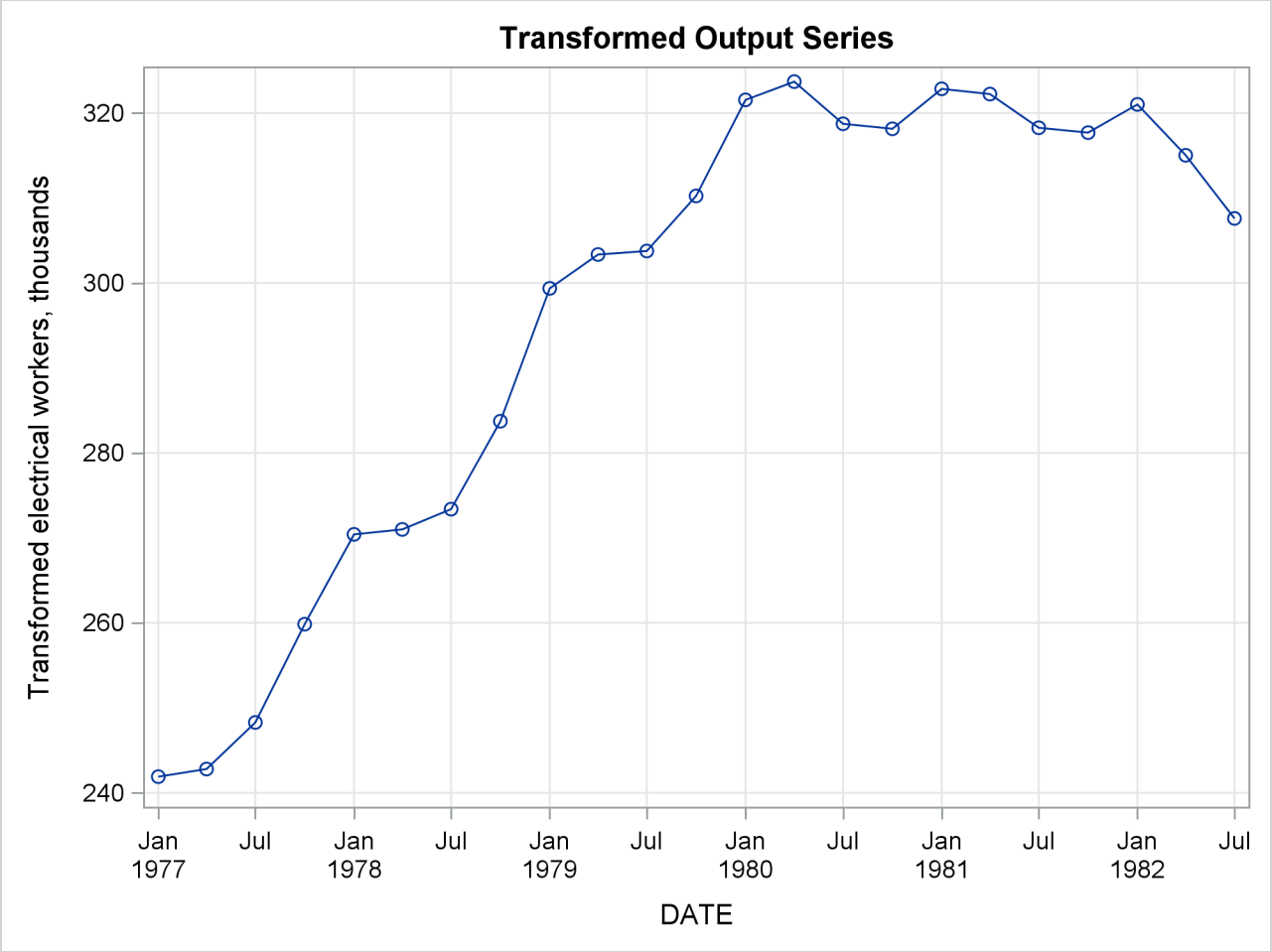
Output 15.2.2 Transformed Input Series Plot—Four-Period Moving Median



Output 15.2.3 Converted Plot of Transformed Input Series



Output 15.2.4 Transformed Output Series Plot—Three-Period Moving Average



Example 15.3: 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 15.3.1:

```
data samples;
  input date : date9. defects @@;
  label defects = "Defects per 1000 Units";
  format date date9.;
datalines;
13jan1992    55    27jan1992    73    19feb1992    84    8mar1992    69

   ... more lines ...
```

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

Output 15.3.1 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
            plots=(input output);
  id date;
  convert defects / observed=(beginning,average);
run;
```

The following PROC PRINT step prints the results, as shown in Output 15.3.2:

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

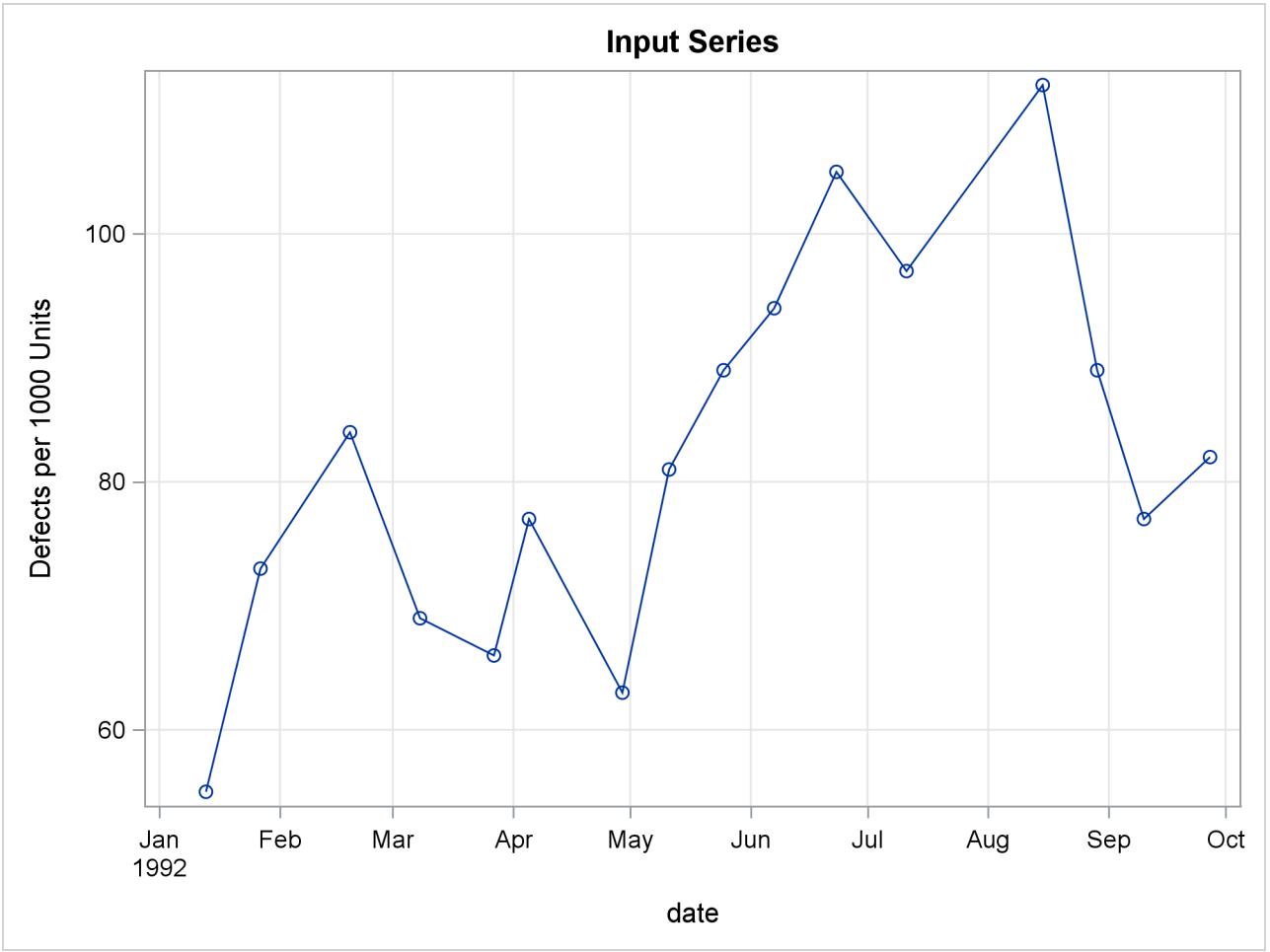
Output 15.3.2 Monthly Average Estimates

Estimated Monthly Average Defect Rates

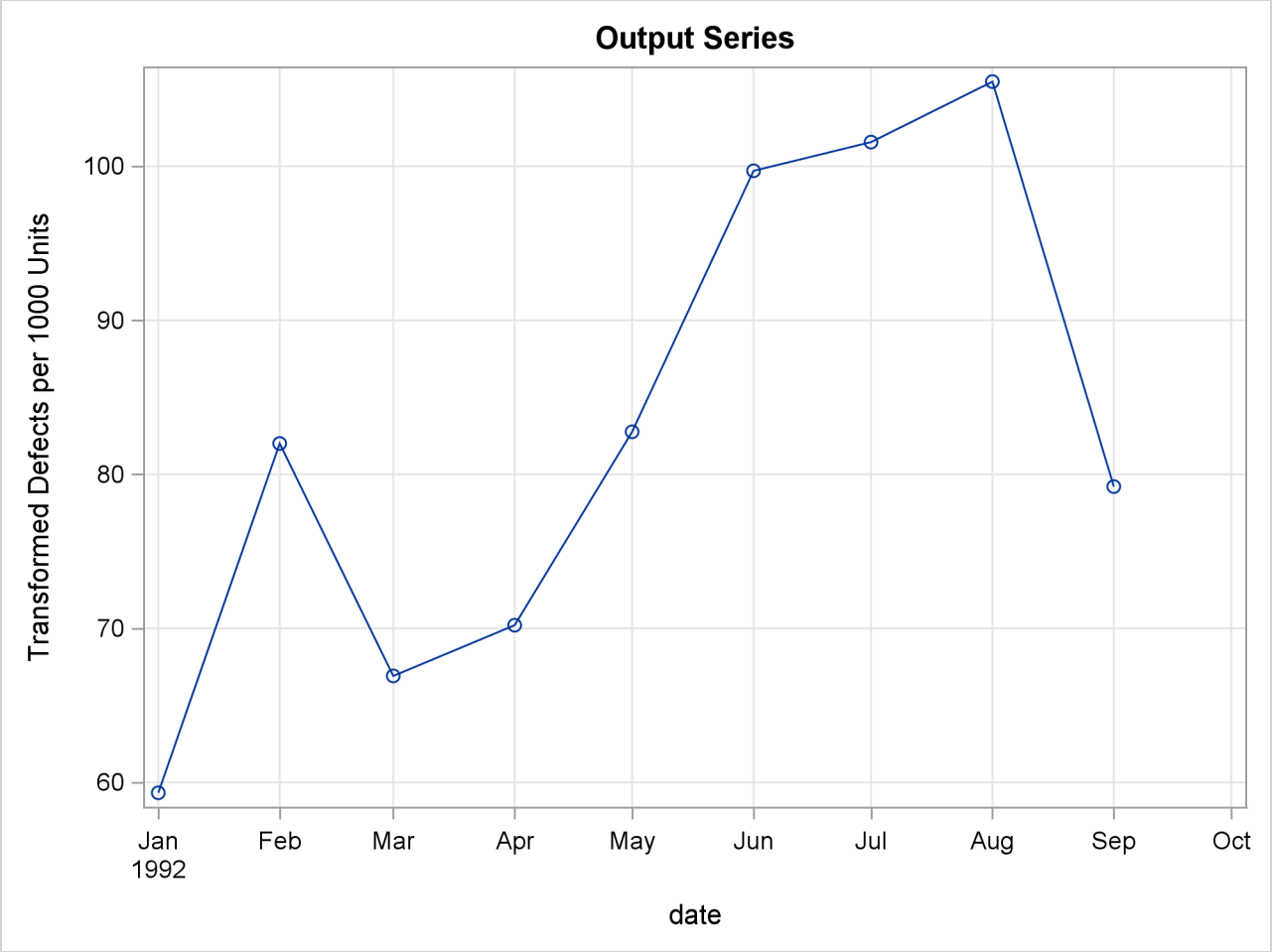
Obs	date	defects
1	JAN1992	59.323
2	FEB1992	82.000
3	MAR1992	66.909
4	APR1992	70.205
5	MAY1992	82.762
6	JUN1992	99.701
7	JUL1992	101.564
8	AUG1992	105.491
9	SEP1992	79.206

The plots produced by PROC EXPAND are shown in Output 15.3.3.

Output 15.3.3 Interpolated Defects Rate Curve



Output 15.3.3 *continued*



Example 15.4: 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   / transformout=(lag 2);
   convert x = x_lag1   / transformout=(lag 1);
   convert x;
   convert x = x_lead1  / transformout=(lead 1);
   convert x = x_lead2  / transformout=(lead 2);
   convert x = x_movave / transformout=(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 15.4.1.

Output 15.4.1 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

De Boor, C. (1978). *A Practical Guide to Splines*. New York: Springer-Verlag.

Hodrick, R. J., and Prescott, E. C. (1980). "Post-war U.S. Business Cycles: An Empirical Investigation." Discussion Paper 451, Carnegie Mellon University.

Levenbach, H., and Cleary, J. P. (1984). *The Modern Forecaster*. Belmont, CA: Lifetime Learning Publications.

Makridakis, S. G., and Wheelwright, S. C. (1978). *Interactive Forecasting: Univariate and Multivariate Methods*. 2nd ed. San Francisco: Holden-Day.

Wheelwright, S. C., and Makridakis, S. G. (1973). *Forecasting Methods for Management*. 3rd ed. New York: Wiley-Interscience.

Subject Index

Syntax Index