

Using PROC EXPAND to Easily Manipulate Longitudinal and Panel Data

Matthew J. Hoolsema, Carnegie Mellon University

ABSTRACT

Working with longitudinal and panel data comes with many data structure challenges. In order to merge data from multiple sources, analysts often have to manipulate a longitudinal data set to make the structure of the two data sets match. Other times, a particular analysis may be more conducive to a wide data set than a long data set, or vice versa. New SAS® users (or experienced SAS users without experience analyzing longitudinal data) can find writing code to manipulate their panel data and prepare an analysis data set particularly challenging.

This paper describes how PROC EXPAND can be used as a simple yet powerful tool for manipulating longitudinal and panel data. Examples explore calculating lags, leads, and moving averages of time varying-observations within a panel, as well as reshaping a data set from long to wide without the need to use arrays or macros. Also shown is how PROC EXPAND can be used in conjunction with other SAS procedures to easily calculate trends over time. Other examples demonstrate using PROC EXPAND to collapse time periods (i.e. converting monthly observations to quarterly observations), interpolate values between observed time periods, and perform data transformations on variables.

INTRODUCTION

PROC EXPAND can easily calculate lags, leads, and several different types of moving averages. Lags look backward in time and specify what the value for that panel member was X time periods ago. Similarly, leads look forward in time to what the value of the panel member is X time periods in the future. These lag and lead variables can be used to reshape longitudinal data from a long structure to a wide structure without needing to create arrays or macros. PROC EXPAND can also calculate a number of different types of time-series operations such as moving averages, minimum, and maximum, as well as cumulative functions.

The general syntax for PROC EXPAND uses the following format:

```
PROC EXPAND DATA=dataset OUT=out_dataset
              FROM=time_interval TO=time_interval METHOD=interpolation_method;
  CONVERT existing_variable = derived_variable / TRANSFORMOUT=(transformation);
  ID Time_id_variable;
  BY Panel_id_variable;
RUN;
```

The BY statement is not needed if you are only using time-series data without multiple subjects. If an OUT data set is not specified, PROC EXPAND will write over the existing data set. Likewise, if a derived variable is not specified, the CONVERT statement will write over the existing variable. The FROM= and TO= options can also be omitted if you do not want to aggregate or interpolate to a different time interval. Multiple CONVERT statements can be used in the same instance of PROC EXPAND. Details of how to use these different options, including types of transformations, are covered in detail in this paper.

USING LAGS AND LEADS TO RESHAPE PANEL DATA

Let's look at an example of a panel data set from a fictitious university's student enrollment database. The database has one record for each student for each year. We have queried the database to create a work data set that has all students that started with the Fall 2000 cohort. However, the data are in a "long" storage structure, where there are multiple records for each student. In order to calculate the percentage of students that are still enrolled in each subsequent year we need to convert the data to a "wide" structure with one record per student and different variables for each year. This could be done with PROC TRANSPOSE, but that requires fluency in creating arrays and can be

cumbersome when working with data with a large number of variables. Instead, we can use PROC EXPAND's lag and lead transformations to reshape our data.

It is important to specify METHOD=none, as we do not want PROC EXPAND to interpolate any missing observations. If we do not specify METHOD=none, SAS will default to interpolating missing values using a cubic spline function. Interpolating missing values is discussed in a later section of this paper.

```
PROC SORT DATA=cohort_2000; BY StudentID SAS_year; RUN; *Data must be sorted!;

PROC EXPAND DATA=cohort_2000 OUT=cohort_2000_lagleads METHOD=none;
  *Specify METHOD=none because we don't want to interpolate any missing data;
  CONVERT currently_enrolled = currently_enrolled_lead1 / TRANSFORMOUT = (lead 1);
  CONVERT currently_enrolled = currently_enrolled_lead2 / TRANSFORMOUT = (lead 2);
  CONVERT currently_enrolled = currently_enrolled_lead3 / TRANSFORMOUT = (lead 3);
  CONVERT currently_enrolled = currently_enrolled_lead4 / TRANSFORMOUT = (lead 4);
  CONVERT currently_enrolled = currently_enrolled_lead5 / TRANSFORMOUT = (lead 5);
  CONVERT currently_enrolled = currently_enrolled_lead6 / TRANSFORMOUT = (lead 6);
  ID SAS_year; *ID statement is the Time ID. Must have SAS date or datetime format;
  BY StudentID; *BY statement is the Panel ID.;
RUN;
```

What exactly is happening in this TRANSFORMOUT= option? For brevity, let's look at an example of one student from the data set. First, SAS creates a new variable we specified named currently_enrolled_lead1. SAS looks forward one time period, in our case one year, and records the value of the next time period into the new variable in the row for the current time period.

FIGURE 1: Calculating One-year Lead

StudentID	Cohort	Year	Currently_Enrolled	Currently_Enrolled_lead1
Doe, Jane	2000	2000	1	1
Doe, Jane	2000	2001	1	1
Doe, Jane	2000	2002	1	1
Doe, Jane	2000	2003	1	0
Doe, Jane	2000	2004	0	0
Doe, Jane	2000	2005	0	0
Doe, Jane	2000	2006	0	.

The same process happens with the TRANSFORMOUT=(lead2) option, except SAS looks forward two time periods, and records that value in the new variable currently_enrolled_lead2. This continues for each lead variable we specified in the CONVERT statements. Our final output has moved all of the values for future years into one record in the data set. Note that leads have been calculated for each year's observation. We can now use a DATA step to delete years not equal to 2000 and complete reshaping our data set from long to wide.

```
DATA cohort_2000_analysis;
  SET cohort_2000_lagleads;
  IF year ne 2000 THEN DELETE;
RUN;
```

FIGURE 2: Reshaping Data using Lead Variables

Student	Cohort	Year	Currently _Enrolled	Currently _Enrolled _lead1	Currently _Enrolled _lead2	Currently _Enrolled _lead3	Currently _Enrolled _lead4	Currently _Enrolled _lead5	Currently _Enrolled _lead6
Doe, Jane	2000	2000	1	1	1	1	0	0	0
Doe, Jane	2000	2001	1	1	1	0	0	0	0
Doe, Jane	2000	2002	1	1	0	0	0	0	0
Doe, Jane	2000	2003	1	0	0	0	0	0	0
Doe, Jane	2000	2004	0	0	0	0	0	0	0
Doe, Jane	2000	2005	0	0	0	0	0	0	0
Doe, Jane	2000	2006	0	0	0	0	0	0	0

The same lead variables have been created for every student in the data set. Now that the data are in wide form, it is easy to use PROC MEANS to calculate the percentage of students in the cohort that are still enrolled each year. Because all of our lead variable names start the same prefix, we can use a colon to include all the lead variables in the VAR statement.

```

TITLE "2000 Cohort Enrollment Rates 0-6 years after initial enrollment";
PROC MEANS DATA=cohort_2000_analysis;
    VAR currently_enrolled currently_enrolled_lead;
RUN;
TITLE;

```

We can see from the PROC MEANS output that 93% of students that enrolled with the 2000 cohort returned for their sophomore year and that 14% were still enrolled as 5th year seniors.

Figure 3: PROC MEANS Output

2000 Cohort Enrollment Rates 0-6 years after initial enrollment

The MEANS Procedure

Variable	N	Mean	Std Dev	Minimum	Maximum
currently_enrolled	500	1.0000000	0	1.0000000	1.0000000
currently_enrolled_lead1	500	0.9320000	0.2519980	0	1.0000000
currently_enrolled_lead2	500	0.8860000	0.3181296	0	1.0000000
currently_enrolled_lead3	500	0.8440000	0.3632187	0	1.0000000
currently_enrolled_lead4	500	0.1400000	0.3473345	0	1.0000000
currently_enrolled_lead5	500	0.0300000	0.1707581	0	1.0000000
currently_enrolled_lead6	500	0.0160000	0.1256008	0	1.0000000

CHANGING TIME PERIOD AGGREGATION

Sometimes data is collected from multiple sources and aggregated at different time intervals. Let's say when prospective students visit the fictitious university in our previous example they sign in at the welcome center in the student union, which reports the number of visitors monthly. However, the admissions office tracks the number of new applications daily. The director of admissions wants to know whether weeks with a larger number of visitors are

associated with a larger volume of new applications. In order to merge these two data sets we must first convert them to the same time frequency. This task is easily accomplished using PROC EXPAND.

AGGREGATING TIME PERIODS

To convert data to different time periods simply specify the FROM= and TO= options. These follow standard SAS date and time period interval conventions. Because our variable records the total number of applications submitted each day we will specify the OBSERVED=total option in the CONVERT statement. Otherwise, SAS will assume the variable is recording the beginning value for each day. Other projects may require capturing the beginning or ending value of a time period. Luckily, PROC EXPAND allows several methods that can be specified with the OBSERVED= option including FIRST, MIDDLE, LAST, AVERAGE, and TOTAL.

```
PROC EXPAND data=Applications OUT=Applications_Weekly FROM=day TO=week;
  CONVERT completed_Apps / OBSERVED=total;
  ID date;
RUN;
```

INTERPOLATING VALUES BETWEEN TIME PERIODS

Caution must be used when using interpolated values in an analysis. The results of any analysis will be based on the predicted values, not the actual values for those time periods. The best practice for analysis is to collapse the smaller time frames to match the largest observed intervals in a data set. However, there may be times where an analyst wants to use estimated values. For example, a company may want to make monthly sales projections based on last year's quarterly sales reports. However, if there were large variations between months, the monthly estimates may not be very accurate. The default is to fit interpolated values using a cubic spline function, however this can be changed using the METHOD= option.

In our example, it could be useful to make interpolated estimates of weekly student visits. We can accomplish this the same way as aggregating to larger time periods using the FROM= and TO= options. Once the data from our two data sets are aggregated at the same time interval, they can be easily merged using a DATA step.

```
PROC EXPAND DATA=Visitors OUT=Visitors_Weekly FROM=month TO=week;
  CONVERT HS_Student_Visits / OBSERVED=total;
  ID date;
RUN;

DATA combined;
  MERGE Application_Weekly Visitors_Weekly;
  BY date;
RUN;
```

FILLING IN TIME PERIODS WITH MISSING OBSERVATIONS

We can also interpolate missing data without changing the time period by omitting the FROM= and TO= options. SAS will fill in any time periods in the data set with missing observations using interpolated observations.

```
PROC EXPAND DATA=Visitors OUT=Visitors_NoMissing;
  CONVERT HS_Student_Visits / OBSERVED=total;
  ID date;
RUN;
```

OTHER FEATURES OF PROC EXPAND

MOVING AVERAGE, MINIMUM, MAXIMUM, AND OTHER TIME-SERIES OPERATIONS

Earlier we used the TRANSFORMIN= and TRANSFORMOUT= options to calculate lags or leads of time series variables. These options can also be used to calculate a number of time-series statistics, including several types of moving averages. A centered moving average is given using the CMOVAVE operator. If we want to calculate the moving average using only data recorded before a specified time-period we would use the MOVAVE operator to calculate a backward moving average. In both cases we must specify the window of time we wish to include in the

moving average. For our example, we will create centered and backward moving averages of application volume using a four week window.

```
PROC EXPAND DATA=Application_Weekly OUT=Visitors_MovingAvg METHOD=NONE;
  CONVERT Completed_Apps = Completed_BackMovAvg / TRANSFORMOUT=(MOVAVE 4);
  CONVERT Completed_Apps = Completed_CenterMovAvg / TRANSFORMOUT=(CMOAVE 4);
RUN;
```

We can also calculate other time-series operations such as moving minimum, median, and maximum, as well as cumulative statistics over a specified number of time periods. This is easily accomplished by changing the operator included in the parentheses of the TRANSFORMOUT= or TRANSFORMIN= option. For a full list of possible operators see:

<http://support.sas.com/documentation/cdl/en/etsug/60372/HTML/default/viewer.htm#etsug_expand_sect026.htm#etsug.expand.exptops>.

DATA TRANSFORMATIONS

When preparing an analysis data set it is common to use data transformations to account for large outliers, skewed or non-normal distributions, or instances where values must fall within a particular range such as probabilities and percentages. Transformations can be used to meet the assumptions of an analysis method, or to change the interpretation of the analysis results. For example, the beta coefficients of a linear regression model where variables have been transformed to the natural logarithm can be interpreted as percent changes rather than discrete changes in the units of the independent variable.

PROC EXPAND can easily perform these transformations at two different steps of analysis. Data transformations can be performed before calculating any interpolated values by using the TRANSFORMIN option. The transformed values are then used to calculate the interpolated values. Likewise, PROC EXPAND can transform values after performing an operation using the TRANSFORMOUT option. Both TRANSFORMIN and TRANSFORMOUT can be used in combination in a single CONVERT statement. This is useful when values must fall between a specific value, such as percentages or probabilities, the interpolating using a cubic spline function could create values outside of this range.

Suppose our visitor data have a variable for the proportion of students in a given month that brought a sibling with them during their campus visit, but for some reason we are missing observations for some months. We want to interpolate the observations, but we want the interpolated numbers to remain between 0 and 1. We can first transform the observed values using the logistic function. SAS will then interpolate missing values using the logged values of our observations, then perform the inverse logistic function to return the values to a proportion. Performing the interpolation on the logged values will prevent any values less than 0 or greater than 1.

```
proc expand DATA=Visitors OUT=Visitors_Imputed;
  CONVERT Sibling / TRANSFORMIN=(logit)
                        TRANSFORMOUT=(ilogit);
  ID date;
run;
```

CONCLUSION

This paper introduces some basic data manipulations that can be accomplished with PROC EXPAND. Panel data can easily be reshaped without needing to create arrays or macros by using the lag and lead transformations. Time series data can also be aggregated to larger time periods, or interpolated to more granular time periods. Time periods with missing observations can also be interpolated without changing the observation frequency. A number of different time-series operations can be calculated including moving averages, minimum, median, and maximum. These statistics can also be calculated cumulatively over time periods. Data transformations can also be performed both before and after calculating statistics. Because of the versatility of PROC EXPAND, it is an important tool for any analyst or SAS programmer working with time-series or panel data.

REFERENCES

SAS/ETS® 9.2 User's Guide, Copyright © 2008, SAS Institute Inc., Cary, NC, USA; All rights reserved. Produced in the United States of America.

ACKNOWLEDGMENTS

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

CONTACT INFORMATION

Matthew J. Hoolsema
Office of Institutional Research and Analysis
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
mhoolsema@cmu.edu