**Paper AA05-2014**

# Creating Code writing algorithms for producing *n*-lagged variables

Matt Bates, J.P. Morgan Chase, Columbus, OH

## ABSTRACT

As a predictive modeler with time-series data there is a continuous search for new and “better” techniques to produce more precise forecasts. One of these techniques is to include lagged values as predictors in a model which leads to a technical coding question. How does one produce an *n*-number (large number of) of lagged variables without having to type every single lagged instance?

## INTRODUCTION

How can a series of 360 (or more) lagged columns be created in a data set without writing every single lag statement? If one has SAS/ETS then the solution may be easier but may still be prone to typing errors and retain a level of mind-numbing inhibition as one types: 1 2 3 4 ……180…. etc. If SAS/ETS is not available then such columns can be added within a data step using the lag*n*() function- but will require ever more mind numbing coding to produce 360+ lagged variable increasing the probability of syntax errors- and proofing becomes even more challenging. The proposed solution we demonstrate is similar in either case and can be leveraged in a similar way for other problems requiring a large number of iterative columns of transformations.

The data presented is based on the neural network forecasting competition of 2008 using altered ATM forecast volumes per day. This is a common scenario for banks to be able to forecast either the amount of withdrawals (total cash taken out) so that it may be know when or how often to service an ATM. The variable of to be forecasted using the lags in these examples is called “usage” which is a transformation of Withdrawals for the ATMs.

### ADDING N-LAGGED COLUMNS USING SAS/ETS (assuming no missing data)

The proc panel procedure is very useful for producing lagged columns with syntax like:

```
PROC PANEL DATA=TRAIN;
       ID  ATM TRANDATE;
       LAG USAGE(1 2 3)/OUT=IMPUTED_LAG;
RUN;
```

However if 360 lagged columns are desired all values 1- 360 would have to entered like:

```
PROC PANEL DATA=TRAIN;
       ID  ATM TRANDATE;
       LAG USAGE(1 2 3…180…360)/OUT=IMPUTED_LAG;
RUN;
```

The solution which is similar in nature for the non ETS method to be discussed is 3 steps:

1) Produce a data set of all values

```
DATA X;
       DO X=1 TO 360;
              OUTPUT;
       END;
RUN;
```

2) Create a global variable using the “separated” feature in proc sql of the entire data set

```
PROC SQL NOPRINT;
       SELECT X INTO:LG SEPARATED BY ' ' FROM X ;
QUIT;
```

3) Insert the global variable into the proc panel procedure.

```
PROC PANEL DATA=TRAIN;
       ID  ATM TRANDATE;
       LAG USAGE(&LG.)/OUT=IMPUTED_LAG;
RUN;
```

### Adding N-lagged columns without using SAS/ETS(assuming no missing data)

Lagged variables within a data step can be very easily achieved in a data step if the data is sorted correctly and the data step contains syntax like:

```
lag1=lag1(usage);
lag2=lag2(usage);
       .
       .
       .
lag<n>=lag<n>(usage);
```

This process adds n-lagged variables to the data step. A word of caution however- if the data is meant to be used with by processing, such as this example with multiple ATMs, then care needs to be taken so that the lag value of one entity (ATM) does not carry over to another entity (ATM). The solution for this scenario is to:

1) Create two temporary lag columns- one for the value the other for the entity (ATM)

2) Only create a lag value if the lag of the entity (ATM) matches the current record entity (ATM).

```
DATA LAG_NO_ETS;
   SET TRAIN;
   BY ATM;
   TMP_LAG1=LAG1(USAGE);
   TMP_LAG2=LAG2(USAGE);
   TMP_LAG3=LAG3(USAGE);

   ATM_LAG1=LAG1(ATM);
   ATM_LAG2=LAG2(ATM);
   ATM_LAG3=LAG3(ATM);

   IF ATM=ATM_LAG1 THEN LAG1=TMP_LAG1;
   IF ATM=ATM_LAG2 THEN LAG2=TMP_LAG2;
   IF ATM=ATM_LAG3 THEN LAG3=TMP_LAG3;
   DROP TMP_LAG1--ATM_LAG3;
RUN;
```

At this point it may be clear that creating 360+ lagged variables would require a significant amount of coding as 3 separate lines are needed for each lag.

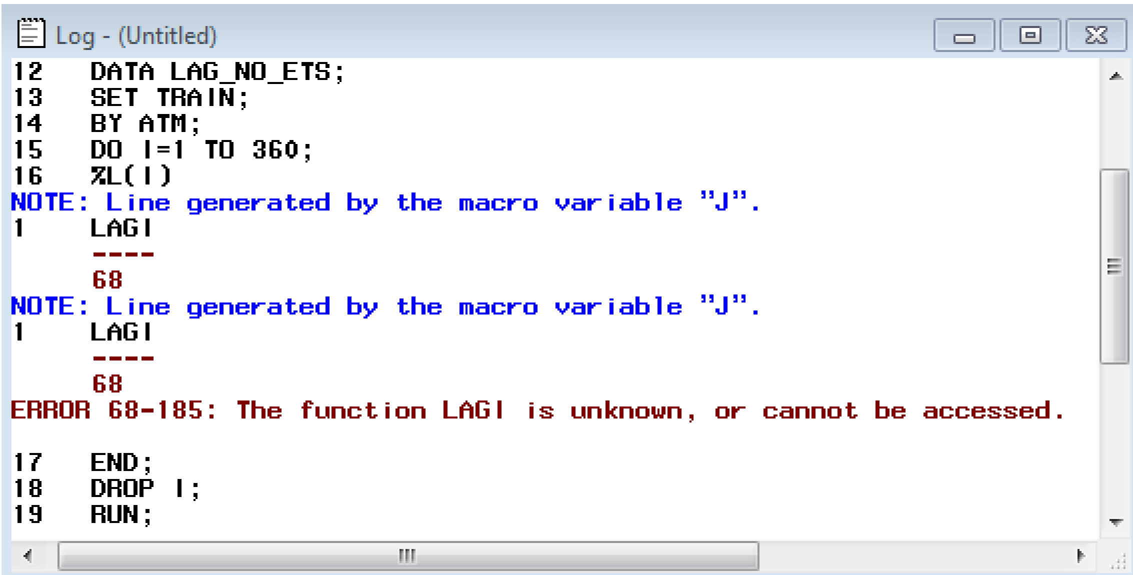
As demonstrated in the SAS/ETS example above the strategy of creating a data set of the actual line code and then using global variables to call appears to be a logical solution- but as shown below this solution is not sufficient without extra modification:

Function style macro:

```
*Function Style Macro;
%MACRO L(J);
L&J.=LAG&J.(USG_IMP);
ATM_LAG&J.=LAG&J.(ATM);
IF ATM=ATM_LAG&J. THEN LAG&J.=TMP_LAG&J.;
%MEND;
```

```
**Failed attempt to create 360 lagged values using the function style macro;
DATA LAG_NO_ETS;
       SET TRAIN;
       BY ATM;
       DO I=1 TO 360;
              %L(I)
       END;
       DROP I;
RUN;
```

Resulting in the following error message:



```
Log - (Untitled)
12     DATA LAG_NO_ETS;
13     SET TRAIN;
14     BY ATM;
15     DO I=1 TO 360;
16     %L(I)
NOTE: Line generated by the macro variable "J".
1      LAGI
       ----
       68
NOTE: Line generated by the macro variable "J".
1      LAGI
       ----
       68
ERROR 68-185: The function LAGI is unknown, or cannot be accessed.

17     END;
18     DROP I;
19     RUN;
```

**Figure 1. Log of Failed Attempt**

This problem results from the fact that the macro is written prior to processing the data step and "J" resolves to "I" rather than what "I" resolves to. The alternative to work around this issue is to imitate what was presented for the SAS/ETS solution presented with the following steps:

1) Create a syntax data set with a do loop that produce the SAS code desired to be processed (including the semicolons)

2) Create macro variables that resolve to the statements in the syntax data set

3) Insert the macro variable to resolve within the data step

```
*STEP 1) CREATING SYNTAX DATA SET;
%LET LAGS=360;

DATA TMP(DROP=I);
    FORMAT TMP_LAG  ATMLAG  LAG $100.;
    DO I=1 TO &LAGS.;
           TMP_LAG=COMPRESS("TMP_LAG"||I||"=LAG"||I||"(USAGE);");
           ATMLAG=COMPRESS("ATMLAG"||I||"=LAG"||I||"(ATM);");
           LAG="IF "||COMPRESS("ATMLAG"||I)||"=ATM THEN "
                  ||COMPRESS("LAG"||I||"=TMP_LAG"||I||";");
           OUTPUT;
    END;
RUN;
```

VIEWTABLE: Work.Tmp

| | TMP_LAG | ATMLAG | LAG |
|---|---|---|---|
| 1 | TMP_LAG1=LAG1(USAGE); | ATMLAG1=LAG1(ATM); | IF ATMLAG1=ATM THEN LAG1=TMP_LAG1; |
| 2 | TMP_LAG2=LAG2(USAGE); | ATMLAG2=LAG2(ATM); | IF ATMLAG2=ATM THEN LAG2=TMP_LAG2; |
| 3 | TMP_LAG3=LAG3(USAGE); | ATMLAG3=LAG3(ATM); | IF ATMLAG3=ATM THEN LAG3=TMP_LAG3; |
| 4 | TMP_LAG4=LAG4(USAGE); | ATMLAG4=LAG4(ATM); | IF ATMLAG4=ATM THEN LAG4=TMP_LAG4; |

**Figure 2. Tmp table to extract global variable reference**

```
*STEP 2) CREATE GLOBAL VARIABLES;
PROC SQL NOPRINT;
      SELECT TMP_LAG INTO:TMP_LAG SEPARATED BY ' ' FROM TMP ;
      SELECT ATMLAG INTO:ATMLAG SEPARATED BY ' ' FROM TMP ;
      SELECT LAG INTO:LAG SEPARATED BY ' ' FROM TMP ;
QUIT;

*STEP 3) INSERTING THE MACRO VARIABLES IN ORDER TO EXECUTE THE N-LAG FUNCTIONS;
DATA IMPUTED_LAG_NO_ETS;
      SET IMPUTED;
      BY ATM;
      &TMP_LAG.
      &ATMLAG.
      &LAG.
      ;
      DROP TMP_LAG1--ATMLAG&LAGS.;
RUN;
```

VIEWTABLE: Work.Lag360_no_ets

| | atm | trandate | usage | LAG1 | LAG2 | LAG3 | LAG4 | LAG5 | LAG6 | LA |
|---|---|---|---|---|---|---|---|---|---|---|
| 730 | NN5_001 | 17MAR1998 | 32.341269841 | 19.699546485 | 24.985827664 | 37.159863946 | 47.16553288 | 60.161564626 | 38.633786848 | 30.66 |
| 731 | NN5_001 | 18MAR1998 | 30.087868481 | 32.341269841 | 19.699546485 | 24.985827664 | 37.159863946 | 47.16553288 | 60.161564626 | 38.633 |
| 732 | NN5_001 | 19MAR1998 | 54.138321995 | 30.087868481 | 32.341269841 | 19.699546485 | 24.985827664 | 37.159863946 | 47.16553288 | 60.161 |
| 733 | NN5_001 | 20MAR1998 | 53.500566893 | 54.138321995 | 30.087868481 | 32.341269841 | 19.699546485 | 24.985827664 | 37.159863946 | 47.16 |
| 734 | NN5_001 | 21MAR1998 | 39.696712018 | 53.500566893 | 54.138321995 | 30.087868481 | 32.341269841 | 19.699546485 | 24.985827664 | 37.159 |
| 735 | NN5_001 | 22MAR1998 | 29.70521542 | 39.696712018 | 53.500566893 | 54.138321995 | 30.087868481 | 32.341269841 | 19.699546485 | 24.985 |
| 736 | NN5_002 | 18MAR1996 | 11.550453515 | . | . | . | . | . | . | |
| 737 | NN5_002 | 19MAR1996 | 13.591269841 | 11.550453515 | . | . | . | . | . | |
| 738 | NN5_002 | 20MAR1996 | 15.036848073 | 13.591269841 | 11.550453515 | . | . | . | . | |
| 739 | NN5_002 | 21MAR1996 | 21.570294785 | 15.036848073 | 13.591269841 | 11.550453515 | . | . | . | |
| 740 | NN5_002 | 22MAR1996 | 27.253401361 | 21.570294785 | 15.036848073 | 13.591269841 | 11.550453515 | . | . | |
| 741 | NN5_002 | 23MAR1996 | 11.706349206 | 27.253401361 | 21.570294785 | 15.036848073 | 13.591269841 | 11.550453515 | . | |
| 742 | NN5_002 | 24MAR1996 | 14.937641723 | 11.706349206 | 27.253401361 | 21.570294785 | 15.036848073 | 13.591269841 | 11.550453515 | |
| 743 | NN5_002 | 25MAR1996 | 12.244897959 | 14.937641723 | 11.706349206 | 27.253401361 | 21.570294785 | 15.036848073 | 13.591269841 | 11.550 |
| 744 | NN5_002 | 26MAR1996 | 15.504535147 | 12.244897959 | 14.937641723 | 11.706349206 | 27.253401361 | 21.570294785 | 15.036848073 | 13.591 |
| 745 | NN5_002 | 27MAR1996 | 18.934240363 | 15.504535147 | 12.244897959 | 14.937641723 | 11.706349206 | 27.253401361 | 21.570294785 | 15.036 |

**Figure 3. Imputed_lag_no_ets table results**

## Filling in for missing lags- SAS/ETS solution

An obvious consequence of adding a large degree of lag variables into any time series model is the increasing number of missing values. As seen in the screen shot in the previous page, when the number of lags increases so does the number of missing values at the beginning of the time series. This renders an increasing number of observations that will be thrown out in the model if these missing values are not imputed. This is one of the major advantages of utilizing proc panel as there are 4 different methods of instantly imputing these missing values when producing the lag columns. Replacing the Lag statement with any of the 4 following statements will cause the missing values to be imputed by the chosen method:

*CLAG, SLAG, XLAG, ZLAG.*

*CLAG*: replaces missing values with the cross section mean for that variable in that cross section. For example if an ATM had a missing value for of lag4 for 23Mar1996 then those values are filled in with the overall average for that ATM.

*SLAG:* replaces missing values with the average corresponding lag of other entities. For example if an ATM had a missing value for of lag4 for 23Mar1996 and there were 2 other ATMS that had lag4 volumes for the 23Mar1996 the missing value of the first ATM would be replace with the average of lag4 for the 2 other ATMs.

*XLAG*: replaces missing values with average of all volumes. For example if an ATM had a missing value for of lag4 for 23Mar1996 then the missing values would be replace by the average of all volumes of all days and ATMs

*ZLAG*: replaces missing values 0

### CLAG/SLAG/XLAG/ZLAG demonstrations:

Consider a data set of 3 ATM –NN5_001, NN5_002, NN5_003 with only one missing value for ATM NN5_001 on Mar24, 1996.

VIEWTABLE: Work.Train

| | atm | trandate | usage |
|---|---|---|---|
| 1 | NN5_001 | 18MAR1996 | 13.407029478 |
| 2 | NN5_001 | 19MAR1996 | 14.725056689 |
| 3 | NN5_001 | 20MAR1996 | 20.564058957 |
| 4 | NN5_001 | 21MAR1996 | 34.708049887 |
| 5 | NN5_001 | 22MAR1996 | 26.629818594 |
| 6 | NN5_001 | 23MAR1996 | 16.609977324 |
| 7 | NN5_001 | 24MAR1996 | . |
| 8 | NN5_001 | 25MAR1996 | 11.607142857 |
| 9 | NN5_001 | 26MAR1996 | 19.883786848 |
| 10 | NN5_001 | 27MAR1996 | 23.767006803 |

**Figure 4. Traintmp table for demonstrating proc panel results**

CLAG: fills in all missing values filled with 28.643468- This is the average of all usage of ATM "NN5_001".

```
PROC PANEL DATA=TRAIN;
       ID ATM TRANDATE;
       CLAG USAGE(0 1 2 3)/OUT=CLAG;
RUN;
```

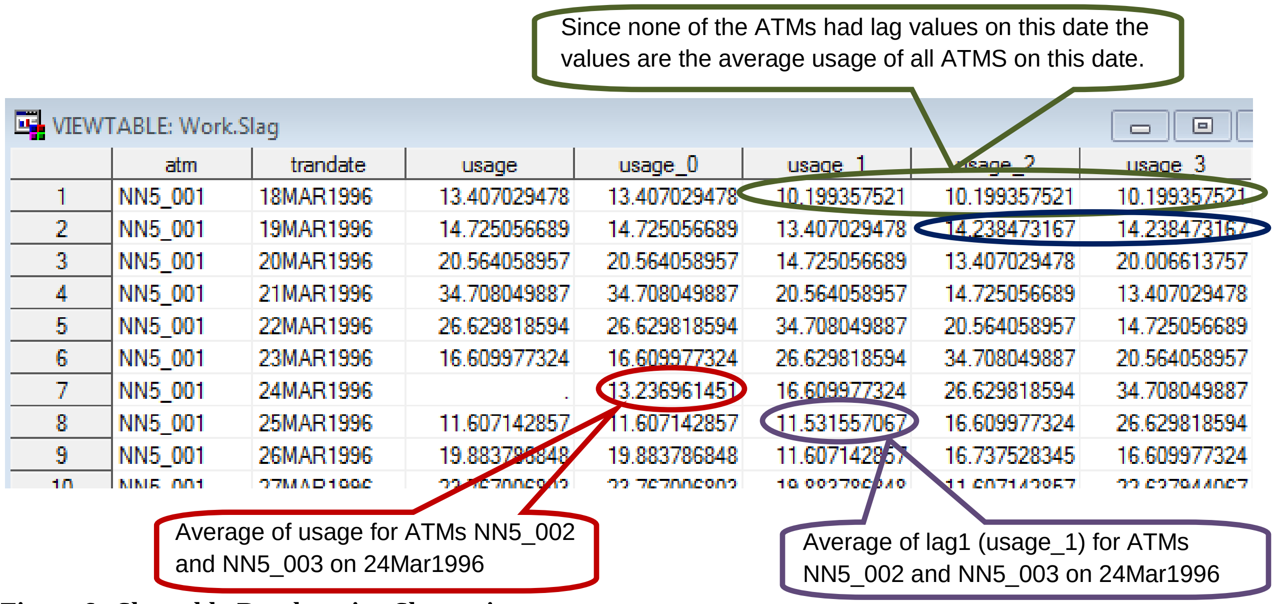
VIEWTABLE: Work.Clag

| | atm | trandate | usage | usage_0 | usage_1 | usage_2 | usage_3 |
|---|---|---|---|---|---|---|---|
| 1 | NN5_001 | 18MAR1996 | 13.407029 | 13.407029 | 28.643468 | 28.643468 | 28.6434680 |
| 2 | NN5_001 | 19MAR1996 | 14.725057 | 14.725057 | 13.407029 | 28.643468 | 28.6434680 |
| 3 | NN5_001 | 20MAR1996 | 20.564059 | 20.564059 | 14.725057 | 13.407029 | 28.6434680 |
| 4 | NN5_001 | 21MAR1996 | 34.70805 | 34.70805 | 20.564059 | 14.725057 | 13.4070294 |
| 5 | NN5_001 | 22MAR1996 | 26.629819 | 26.629819 | 34.70805 | 20.564059 | 14.7250566 |
| 6 | NN5_001 | 23MAR1996 | 16.609977 | 16.609977 | 26.629819 | 34.70805 | 20.5640589 |
| 7 | NN5_001 | 24MAR1996 | . | 28.643468 | 16.609977 | 26.629819 | 34.7080498 |
| 8 | NN5_001 | 25MAR1996 | 11.607143 | 11.607143 | 28.643468 | 16.609977 | 26.6298185 |
| 9 | NN5_001 | 26MAR1996 | 19.883787 | 19.883787 | 11.607143 | 28.643468 | 16.6099773 |
| 10 | NN5_001 | 27MAR1996 | 23.767007 | 23.767007 | 19.883787 | 11.607143 | 28.6434680 |
| 11 | NN5_001 | 28MAR1996 | 34.027778 | 34.027778 | 23.767007 | 19.883787 | 11.6071428 |
| 12 | NN5_001 | 29MAR1996 | 33.786848 | 33.786848 | 34.027778 | 23.767007 | 19.8837868 |
| 13 | NN5_001 | 30MAR1996 | 18.253968 | 18.253968 | 33.786848 | 34.027778 | 23.767006 |
| 14 | NN5_001 | 31MAR1996 | 19.387755 | 19.387755 | 18.253968 | 33.786848 | 34.0277777 |

**Figure 5. Clag table-Results using Clag option**

SLAG: fills in the missing values for the according date of Mar 24, 1996 with the average for the usage in ATMs NN5_002 and NN5_003 of 13.236961451. However as the future lags are calculated rather than carrying this value down the average of the respective Lag is the fill in value for that lag. For example lag1 the average of the usage on May 25, 1997 of ATMs NN5_002 and NN5_003 is 11.531557067 which is not the normal carry down of 13.236961451. Also notice that the average for the backfill dates is the average of all 3 ATM

```
PROC PANEL DATA=TRAIN;
        ID   ATM TRANDATE;
        SLAG USAGE(0 1 2 3)/OUT=SLAG;
RUN;
```

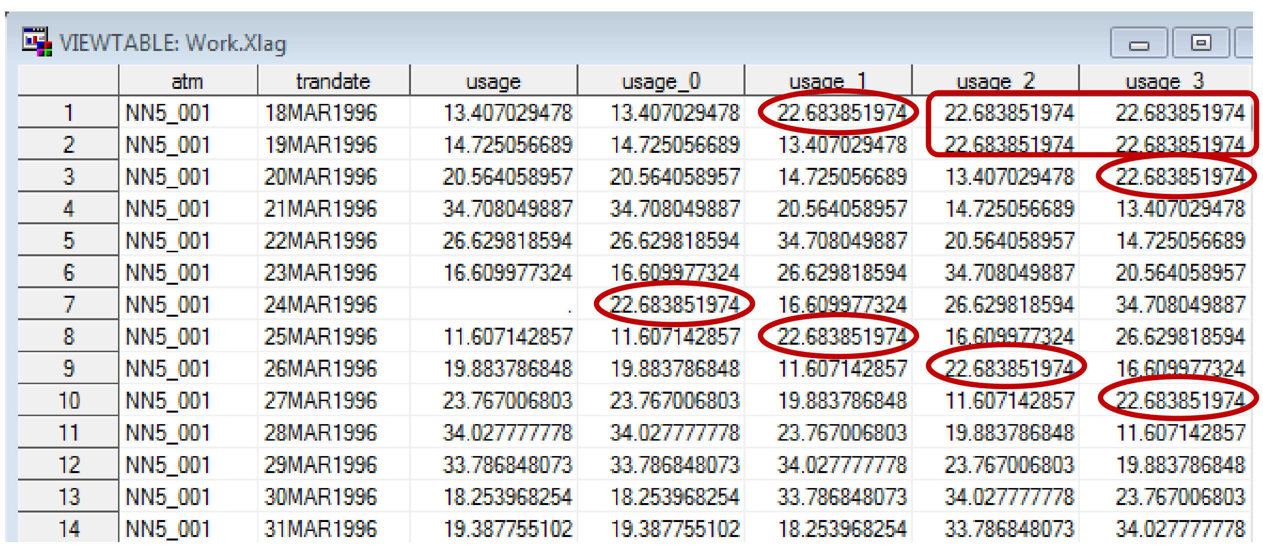



| | atm | trandate | usage | usage_0 | usage_1 | usage_2 | usage_3 |
|---|---|---|---|---|---|---|---|
| 1 | NN5_001 | 18MAR1996 | 13.407029478 | 13.407029478 | 10.199357521 | 10.199357521 | 10.199357521 |
| 2 | NN5_001 | 19MAR1996 | 14.725056689 | 14.725056689 | 13.407029478 | 14.238473167 | 14.238473167 |
| 3 | NN5_001 | 20MAR1996 | 20.564058957 | 20.564058957 | 14.725056689 | 13.407029478 | 20.006613757 |
| 4 | NN5_001 | 21MAR1996 | 34.708049887 | 34.708049887 | 20.564058957 | 14.725056689 | 13.407029478 |
| 5 | NN5_001 | 22MAR1996 | 26.629818594 | 26.629818594 | 34.708049887 | 20.564058957 | 14.725056689 |
| 6 | NN5_001 | 23MAR1996 | 16.609977324 | 16.609977324 | 26.629818594 | 34.708049887 | 20.564058957 |
| 7 | NN5_001 | 24MAR1996 | . | 13.236961451 | 16.609977324 | 26.629818594 | 34.708049887 |
| 8 | NN5_001 | 25MAR1996 | 11.607142857 | 11.607142857 | 11.531557067 | 16.609977324 | 26.629818594 |
| 9 | NN5_001 | 26MAR1996 | 19.883786848 | 19.883786848 | 11.607142857 | 16.737528345 | 16.609977324 |
| 10 | NN5_001 | 27MAR1996 | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] |

**Figure 6. Slag table-Results using Slag option**

XLAG: fills in all missing values with the overall average of all volumes. The Average for the 3 atms over all dates is 22.683851974.

```
PROC PANEL DATA=TRAIN;
        ID   ATM TRANDATE;
        XLAG USAGE(0 1 2 3)/OUT=XLAG;
RUN;
```




| | atm | trandate | usage | usage_0 | usage_1 | usage_2 | usage_3 |
|---|---|---|---|---|---|---|---|
| 1 | NN5_001 | 18MAR1996 | 13.407029478 | 13.407029478 | 22.683851974 | 22.683851974 | 22.683851974 |
| 2 | NN5_001 | 19MAR1996 | 14.725056689 | 14.725056689 | 13.407029478 | 22.683851974 | 22.683851974 |
| 3 | NN5_001 | 20MAR1996 | 20.564058957 | 20.564058957 | 14.725056689 | 13.407029478 | 22.683851974 |
| 4 | NN5_001 | 21MAR1996 | 34.708049887 | 34.708049887 | 20.564058957 | 14.725056689 | 13.407029478 |
| 5 | NN5_001 | 22MAR1996 | 26.629818594 | 26.629818594 | 34.708049887 | 20.564058957 | 14.725056689 |
| 6 | NN5_001 | 23MAR1996 | 16.609977324 | 16.609977324 | 26.629818594 | 34.708049887 | 20.564058957 |
| 7 | NN5_001 | 24MAR1996 | . | 22.683851974 | 16.609977324 | 26.629818594 | 34.708049887 |
| 8 | NN5_001 | 25MAR1996 | 11.607142857 | 11.607142857 | 22.683851974 | 16.609977324 | 26.629818594 |
| 9 | NN5_001 | 26MAR1996 | 19.883786848 | 19.883786848 | 11.607142857 | 22.683851974 | 16.609977324 |
| 10 | NN5_001 | 27MAR1996 | 23.767006803 | 23.767006803 | 19.883786848 | 11.607142857 | 22.683851974 |
| 11 | NN5_001 | 28MAR1996 | 34.027777778 | 34.027777778 | 23.767006803 | 19.883786848 | 11.607142857 |
| 12 | NN5_001 | 29MAR1996 | 33.786848073 | 33.786848073 | 34.027777778 | 23.767006803 | 19.883786848 |
| 13 | NN5_001 | 30MAR1996 | 18.253968254 | 18.253968254 | 33.786848073 | 34.027777778 | 23.767006803 |
| 14 | NN5_001 | 31MAR1996 | 19.387755102 | 19.387755102 | 18.253968254 | 33.786848073 | 34.027777778 |

**Figure 7. Xlag table-Results using Xlag option**

ZLAG Fills in all missing Values with 0.

```
PROC PANEL DATA=TRAINTMP;
        ID   ATM TRANDATE;
        ZLAG USAGE(0 1 2 3)/OUT=ZLAG;
RUN;
```

VIEWTABLE: Work.Zlag

| | atm | trandate | usage | usage_0 | usage_1 | usage_2 | usage_3 |
|---|---|---|---|---|---|---|---|
| 1 | NN5_001 | 18MAR1996 | 13.407029478 | 13.407029478 | 0 | 0 | 0 |
| 2 | NN5_001 | 19MAR1996 | 14.725056689 | 14.725056689 | 13.407029478 | 0 | 0 |
| 3 | NN5_001 | 20MAR1996 | 20.564058957 | 20.564058957 | 14.725056689 | 13.407029478 | 0 |
| 4 | NN5_001 | 21MAR1996 | 34.708049887 | 34.708049887 | 20.564058957 | 14.725056689 | 13.407029478 |
| 5 | NN5_001 | 22MAR1996 | 26.629818594 | 26.629818594 | 34.708049887 | 20.564058957 | 14.725056689 |
| 6 | NN5_001 | 23MAR1996 | 16.609977324 | 16.609977324 | 26.629818594 | 34.708049887 | 20.564058957 |
| 7 | NN5_001 | 24MAR1996 | . | 0 | 16.609977324 | 26.629818594 | 34.708049887 |
| 8 | NN5_001 | 25MAR1996 | 11.607142857 | 11.607142857 | 0 | 16.609977324 | 26.629818594 |
| 9 | NN5_001 | 26MAR1996 | 19.883786848 | 19.883786848 | 11.607142857 | 0 | 16.609977324 |
| 10 | NN5_001 | 27MAR1996 | 23.767006803 | 23.767006803 | 19.883786848 | 11.607142857 | 0 |
| 11 | NN5_001 | 28MAR1996 | 34.027777778 | 34.027777778 | 23.767006803 | 19.883786848 | 11.607142857 |
| 12 | NN5_001 | 29MAR1996 | 33.786848073 | 33.786848073 | 34.027777778 | 23.767006803 | 19.883786848 |
| 13 | NN5_001 | 30MAR1996 | 18.253968254 | 18.253968254 | 33.786848073 | 34.027777778 | 23.767006803 |
| 14 | NN5_001 | 31MAR1996 | 19.387755102 | 19.387755102 | 18.253968254 | 33.786848073 | 34.027777778 |

**Figure 8. Zlag table-Results using Zlag option**

### Filling in for missing lags- non-SAS/ETS solution

There are of course and infinite number of ways to deal with these missing values. The proposed method is to take the assumed most known significant cycle and use that number of forward lags to back fill. In the case of ATM Volumes Weekday volumes tend to be the most significant signal for ATMS. For this reason lag 7 is calculated for the reverse order of the ATM and then used to fill in for any missing values.

The steps to carry out this weekday closest lag solution are:

1) Sort the data by atm, weekday and descending trandate
2) Create a series of n-temporary variables to be referenced as a single global variable
3) Using array and retain statements reset each n-temporary variable to missing if either the first atm or first weekday. Using a do loop with the arrays to assign each lag variable to the last corresponding lag value.

VIEWTABLE: Work.Imputing_needed

| | atm | trandate | usage | LAG1 | LAG2 | LAG3 | LAG4 | LAG5 | LAG6 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | NN5_001 | 18MAR1996 | 13.407029478 | . | . | . | . | . | . |
| 2 | NN5_001 | 19MAR1996 | 14.725056689 | 13.407029478 | . | . | . | . | . |
| 3 | NN5_001 | 20MAR1996 | 20.564058957 | 14.725056689 | 13.407029478 | . | . | . | . |
| 4 | NN5_001 | 21MAR1996 | 34.708049887 | 20.564058957 | 14.725056689 | 13.407029478 | . | . | . |
| 5 | NN5_001 | 22MAR1996 | 26.629818594 | 34.708049887 | 20.564058957 | 14.725056689 | 13.407029478 | . | . |
| 6 | NN5_001 | 23MAR1996 | 16.609977324 | 26.629818594 | 34.708049887 | 20.564058957 | 14.725056689 | 13.407029478 | . |
| 7 | NN5_001 | 24MAR1996 | 15.320294785 | 16.609977324 | 26.629818594 | 34.708049887 | 20.564058957 | 14.725056689 | 13.407029478 |
| 8 | NN5_001 | 25MAR1996 | 11.607142857 | 15.320294785 | 16.609977324 | 26.629818594 | 34.708049887 | 20.564058957 | 14.725056689 |
| 9 | NN5_001 | 26MAR1996 | 19.883786848 | 11.607142857 | 15.320294785 | 16.609977324 | 26.629818594 | 34.708049887 | 20.564058957 |
| 10 | NN5_001 | 27MAR1996 | 23.767006803 | 19.883786848 | 11.607142857 | 15.320294785 | 16.609977324 | 26.629818594 | 34.708049887 |
| 11 | NN5_001 | 28MAR1996 | 34.027777778 | 23.767006803 | 19.883786848 | 11.607142857 | 15.320294785 | 16.609977324 | 26.629818594 |
| 12 | NN5_001 | 29MAR1996 | 33.786848073 | 34.027777778 | 23.767006803 | 19.883786848 | 11.607142857 | 15.320294785 | 16.609977324 |
| 13 | NN5_001 | 30MAR1996 | 18.253968254 | 33.786848073 | 34.027777778 | 23.767006803 | 19.883786848 | 11.607142857 | 15.320294785 |

**Figure 9. Imputing_needed table-demonstrating which values need to be shifted up**

Step 1)

```
*ADDING WEEKDAY IN ORDER TO SORT BY WEEKDAY;
DATA IMPUTING_NEEDED;
       SET LAG360_NO_ETS;
       WKDY=WEEKDAY(TRANDATE);
RUN;

PROC SORT DATA= IMPUTING_NEEDED;
       BY ATM WKDY DESCENDING TRANDATE;
RUN;
```

Step 2)
```
*CREATING A LIST OF N# OF TEMPORARY VARIABLES FOR RETAIN STATEMENT;
       DATA TMP;
              FORMAT TMP $8.;
              DO I=1 TO &LAGS.;
                     TMP="T"||LEFT(I);
                     OUTPUT;
              END;
       RUN;
       PROC SQL NOPRINT;
              SELECT TMP INTO:TMP SEPARATED BY ' ' FROM TMP ;
       QUIT;
```

Step 3)
```
*USING THE RETAIN AND ARRAY STATEMENTS TO FILL IN MISSING VALUES WITH CLOSEST
WEEKDAY LAG VALUE;
       DATA IMPUTED(DROP=WKDY);
              SET IMPUTING_NEEDED;
              BY ATM WKDY;
              RETAIN &TMP.;
              ARRAY L{*} LAG1--LAG&LAGS.;
              ARRAY T{*} T1--T&LAGS.;
       *ENSURING VALUE OF OTHER ATMS OR WEEKDAYS DO NOT CROSS OVER;
              IF FIRST.ATM OR FIRST.WKDY THEN DO I=1 TO &LAGS.;
                     T(I)=.;
              END;
              DO I=1 TO &LAGS.;
       *ONLY REPLACES LAG VARIABLE IF MISSING WITH LATEST NON MISSING WEEKDAY VOLUME;
                     IF L(I)=. THEN L(I)=T(I);
       *SET TEMPORARY LAG-N VARIABLES TO LATESTEST LAG-N VALUE;
                     T(I)=L(I);
              END;
       DROP I &TMP.;
       RUN;

       PROC SORT DATA=IMPUTED;
              BY ATM TRANDATE;
       RUN;
```

VIEWTABLE: Work.Imputed

| | atm | trandate | usage | LAG1 | LAG2 | LAG3 | LAG4 | LAG5 | LAG6 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | NN5_001 | 18MAR1996 | 13.407029478 | 15.320294785 | 16.609977324 | 26.629818594 | 34.708049887 | 20.564058957 | 14.725056689 |
| 2 | NN5_001 | 19MAR1996 | 14.725056689 | 13.407029478 | 15.320294785 | 16.609977324 | 26.629818594 | 34.708049887 | 20.564058957 |
| 3 | NN5_001 | 20MAR1996 | 20.564058957 | 14.725056689 | 13.407029478 | 15.320294785 | 16.609977324 | 26.629818594 | 34.708049887 |
| 4 | NN5_001 | 21MAR1996 | 34.708049887 | 20.564058957 | 14.725056689 | 13.407029478 | 15.320294785 | 16.609977324 | 26.629818594 |
| 5 | NN5_001 | 22MAR1996 | 26.629818594 | 34.708049887 | 20.564058957 | 14.725056689 | 13.407029478 | 15.320294785 | 16.609977324 |
| 6 | NN5_001 | 23MAR1996 | 16.609977324 | 26.629818594 | 34.708049887 | 20.564058957 | 14.725056689 | 13.407029478 | 15.320294785 |
| 7 | NN5_001 | 24MAR1996 | 15.320294785 | 16.609977324 | 26.629818594 | 34.708049887 | 20.564058957 | 14.725056689 | 13.407029478 |
| 8 | NN5_001 | 25MAR1996 | 11.607142857 | 15.320294785 | 16.609977324 | 26.629818594 | 34.708049887 | 20.564058957 | 14.725056689 |
| 9 | NN5_001 | 26MAR1996 | 19.883786848 | 11.607142857 | 15.320294785 | 16.609977324 | 26.629818594 | 34.708049887 | 20.564058957 |
| 10 | NN5_001 | 27MAR1996 | 23.767006803 | 19.883786848 | 11.607142857 | 15.320294785 | 16.609977324 | 26.629818594 | 34.708049887 |
| 11 | NN5_001 | 28MAR1996 | 34.027777778 | 23.767006803 | 19.883786848 | 11.607142857 | 15.320294785 | 16.609977324 | 26.629818594 |
| 12 | NN5_001 | 29MAR1996 | 33.786848073 | 34.027777778 | 23.767006803 | 19.883786848 | 11.607142857 | 15.320294785 | 16.609977324 |
| 13 | NN5_001 | 30MAR1996 | 18.253968254 | 33.786848073 | 34.027777778 | 23.767006803 | 19.883786848 | 11.607142857 | 15.320294785 |
| 14 | NN5_001 | 31MAR1996 | 19.387755102 | 18.253968254 | 33.786848073 | 34.027777778 | 23.767006803 | 19.883786848 | 11.607142857 |

**Figure 10. Imputed table results of shifting values up**

## CONCLUSION

Creating iterative variables using the proposed 3 step method to invoking command syntax stored as macro variables can save on coding time and is more robust to non detectable log type errors. This process is particular useful in that it can be applied to any other similar scenario as the demonstrated multiple lagged scenario in order to implement

## REFERENCES


- Source data used to demonstrate techniques originated from the NN5 Competition and was "pre-imputed" to fill in for missing dates/values prior to the execution of all techniques demonstrated. The source data of NN5 may be found at: http://www.neural-forecasting-competition.com/downloads/NN5/datasets/download.htm


## ACKNOWLEDGEMENTS


Special Thanks for their contribution to make this paper possible: Anoop Nair, Steve Winstead, Aslam Chaudhry, Debby Feurer, James Forrest, Jason Van Hulse, Emily Smith, James Gearheart, Donovan Gibson.


## RECOMMENDED READING

- Cody, R. (2008). *Cody's Data Cleaning Techniques Using SAS*. SAS Institute: Cary, NC.
- Tian,Yunchao, (2009). LAG- the Very Powerful and Easily Misused SAS® Function:Social & Scientific Systems, Inc. Silver Spring, MD

## CONTACT INFORMATION


Name:Matt Bates
Company:J.P.Morgan Chase
Email:matthew.bates@chase.com


## DISCLAIMER

The contents of this paper are the work of the author and do not necessarily represent the opinions, recommendations, or practices of J.P Morgan Chase.