Using User Defined FORMATS and the INTNX Date Function to Extract LAGS and LEADS

Jonas V. Bilenas, Any Bank

PhilaSUG Spring 2019

June 5, 2019

Scope of the Presentation.

- I have been using the SAS® PROC FORMAT for decades to create efficient and fast table lookups. A USER created FORMAT (table lookup) generally uses a binary search method in memory to map unique keys (VALUES) to labels. Examples:
  - Mapping abbreviated codes to code descriptions. The mapping does not have to occur in a data set and can be applied when using other SAS procedures. For example, generating a report where medical diagnostic codes are expressed as labels. Saves space and time.
  - The format can be used to extract data from very large data set based on account keys. Example; you have a data set of 30,000 accounts you are doing analysis on. You have a total of 2 billion transactions and you want to extract only records that match to the sample of 30,000 accounts. The binary search of the 30,000 records will only require reading, at the worse, 15 records for each of the 2 billion transaction. Extract the records that match to the modeling data set. No need to sort the 2 billion records. Note: SQL will often sort both data sets. Max search from the format table is about **15 records at the worse [log(N)/log(2) + 1].** SQL or DATA merges often take 2 days or more. The FORMAT run typically takes less than 10 minutes. Similar to the speed of using HASH joins.
  - Adding new variables to a to the larger dataset in question based on segmentation or other keys.

A Simple Example of PROC FORMAT

- You can hard code the table look-up using PROC FORMAT. The table look-up is saved in the work directory and can be applied to other PROCS or DATA STEPS in the code after the FORMAT is processed.

```
proc format;
  value $sex
    'M','m'   = 'MALE'
    'F','f'   = 'FEMALE'
    ' '       = 'Missing'
    'O','o'   = 'OTHER'
    other     = [$2.]
  ;
  value $sex_a
    'M','m'   = 'MALE'
    'F','f'   = 'FEMALE'
    'O','o'   = 'OTHER'
    ' '       = 'Missing'
    other     = 'BAD DATA'
  ;
run;
proc freq data=test;
   table sex/missing;
   format sex $sex.;
 run;
proc freq data=test;
   table sex/missing;
   format sex $sex_a.;
 run;
```

Embedded FORMAT

The FREQ Procedure

sex	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Missing	24	5.21	24	5.21
?	55	11.93	79	17.14
FEMALE	87	18.87	166	36.01
MALE	154	33.41	320	69.41
OTHER	70	15.18	390	84.60
x	71	15.40	461	100.00

The FREQ Procedure

sex	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Missing	24	5.21	24	5.21
BAD DATA	126	27.33	150	32.54
FEMALE	87	18.87	237	51.41
MALE	154	33.41	391	84.82
OTHER	70	15.18	461	100.00

A Simple Example of PROC FORMAT

- This will work as well.

```
76          proc format;
77          value $sex
78          'M','m' = 'MALE'
79          'F','f' = 'FEMALE'
80          ' ' = 'Missing'
81          'O','o' = 'OTHER'
82           other = [$BEST.]

NOTE: The $BEST (in)format was specified on the right-hand side of an equal sign, but
      without a length specification. PROC FORMAT will assume a default length of at least
      40 for the format being generated. If this is an insufficient width, you can rerun
      PROC FORMAT with an explicit width for the $BEST (in)format, or provide a sufficient
      DEFAULT= option.
83 ;
NOTE: Format $SEX has been output.
```

An Example For This Presentation.

- Time series data sources typically have the macroeconomic data (historical actual data and forecasts into the future) by date.   They are typically stored in spreadsheets, SAS data sets, Oracle, Teradata, DB2, and other locations.
- In modelling, reporting, and/or generating plots we often need to capture information on macroeconomic data and calculate lags and/or leads of variables at a specific point in time.
  - Using the LAG function in a data step can be problematic. **NEVER** use the LAG function conditionally (Y. Tian, 2009).
  - Going Forward in time can be an issue as well. There is no LEAD function in the SAS data step.
  - Don’t worry, PROC FORMAT is HERE!

Why not use SQL?

- Colleagues tell me they like to use PROC SQL because you don’t have to SORT the data. Well, **you** are not coding a PROC SORT in your code but **SQL** decides how to do the join. Most of the time the SQL procedure will sort both data set and run times would be similar to a data step merge.
    - You can see how SQL is deciding on the join using the _METHOD SQL option and code translation papers (Shipp & Laffler, 2013 and Laffler, 2009).
    - **HASH joins and FORMAT beat out SQL and MERGE** statements in data step (Liang, Q. 2009).

Using DATES as a key to map labels that are tied to specific dates.

- The example we will work on here is from sample data from https://datahub.io/collections/economic-data using CPI data as an example. Monthly CPI data is from 01JAN1913 to 01JAN2014. **1,213** observations loaded to a SAS data set. No future forecast data provided.
- We don't want to hard code the FORMAT. We will generate a modified data set that contains the keys (aka values) and the labels that feed into PROC FROMAT and **that FORMAT will be saved in a permanent FORMAT CATALOG.**

First 5 Rows of Data; DATE, CPI, INFLATION

```
1913-01-01 9.8 .
1913-02-01 9.8 0.0
1913-03-01 9.8 0.0
1913-04-01 9.8 0.0
1913-05-01 9.7 -1.02
```

Using DATES as a Value to map labels that are tied to specific dates.

- Data was copied into a DATA step. We used an INFORMAT of **ANYDTDTE** to read in the date field and to confirm dates were read correctly. First 10 records were printed to the log, shown below to confirm the DATE variable was read in correctly by using a date9 FORMAT.

INFORMAT

```
data stuff.cpi;
  input date anydtdte10.
        index
        Inflation;
if _N_ < 10 then do;
  put date date9. +5 index +5 inflation ;
end;
datalines;


01JAN1913 9.8 .
01FEB1913 9.8 0
01MAR1913 9.8 0
01APR1913 9.8 0
01MAY1913 9.7 -1.02
01JUN1913 9.8 1.03
01JUL1913 9.9 1.02
01AUG1913 9.9 0
01SEP1913 10 1.01
```

FORMAT

Alphabetic List of Variables and Attributes

#	Variable	Type	Length
1	date	Num	8
2	index	Num	8
3	inflation	Num	8

Generating the FORMAT CATALOG using PROC FORMAT

- You can hard code the PROC FORMAT in SAS. However for larger format mappings you can create the format using a special CNTLIN data set.
- The CNTLIN data set must contain at least the following VARIABLES:
    - **START: name of the key variable. (optional END variable is for formats using a range of VALUES) We rename the variable DATE as START.**
    - **LABEL: name of the mapping variable. In this example we rename INDEX variable to variable name LABEL.**
    - **TYPE: 'N' for numeric keys, 'C' or character keys.**
    - **FMTNAME: name of the FORMAT.**

STELF: START, TYPE, END, LABEL, FMTNAME

Generating the FORMAT CATALOG using PROC FORMAT

```
options nocenter fullstimer;
libname  stuff '/somewhere/FORMATS';
libname library '/somewhere/FORMATS';

data fmt  / view=fmt;
  set stuff.cpi  (rename=(date=start
                  index=label)) end=eof;
  retain fmtname 'CPI'  type 'N';
  output;
  if eof then do;
    /*start=.*/;
    label=.;
    HLO='O';
    output;
  end;
run;
proc format cntlin=fmt library = library
/*fmtlib*/;
run;
```

- Looking at the LOG snippet below we see that it took about a 0.04 seconds to generate the catalog (reading the data and generating the permanent catalog).
- **Future code does not need to generate the catalog again since the catalog is stored in the `library` `libname`.** The code that generated the FORMAT does not need to be processed in future code unless the FORMAT is updated.

```
 NOTE: DATA STEP view saved on file WORK.FMT.
 NOTE: A stored DATA STEP view cannot run under a
different operating system.
 NOTE: DATA statement used (Total process time):
       real time           0.00 seconds
       user cpu time       0.00 seconds
       system cpu time     0.00 seconds
       memory              1053.71k
 NOTE: View WORK.FMT.VIEW used (Total process time):
       real time           0.02 seconds
       user cpu time       0.00 seconds
       system cpu time     0.01 seconds
       memory              1112.96k
 NOTE: PROCEDURE FORMAT used (Total process time):
       real time           0.02 seconds
       user cpu time       0.01 seconds
       system cpu time     0.01 seconds
       memory              1112.96k
 NOTE: There were 1213 observations read from the
data set STUFF.CPI.
 NOTE: There were 1214 observations read from the
data set WORK.FMT.
```

Generating the FORMAT CATALOG using PROC FORMAT

```
options nocenter fullstimer;
libname  stuff '/somewhere/FORMATS';
libname library '/somewhere/FORMATS';

proc format cntlin=fmt library = library;
run;
```

```
options nocenter fullstimer;
libname  stuff '/somewhere/FORMATS';
libname library '/somewhere/FORMATS';

proc format cntlin=fmt library = library.test;
run;
```

- **Generates a permanent file in the LIBNAME library**
  - **First example: formats.sas7bcat**
  - **Second example: test.sas7bcat**
- **Once the catalog file is created you don’t need the DATA FMT step nor the PROC FORMAT step.**
- **The format catalog can contain more than 1 format.**

How to use the FORMAT?

- As an example, say you have a data set with monthly dates anchored at the first of each month from **01JAN2000 to 01DEC2012**. You want to pull in the CPI for each of the months. Code is shown below. By default, the format search will, by default look in the WORK directory followed by the LIBRARY directory. You can use a FMTSEARCH option to specify other CATALOGS to seek the FORMAT or INFORMAT.

```
options nocenter fullstimer mprint symbolgen;
libname  stuff '/somewhere/FORMATS';
libname library '/ somewhere /FORMATS';
data test2;
  set test;  /* monthly data from 01JAN2000 to 01DEC2012 */
   cpi = input(put(date,CPI.),best.);
Run;
```

PUT always returns a character variable.

INPUT, in this example, will return a numeric variable.

Getting LEADS and LAGS with INTNX date/time DATA funtion

- What if you wanted to also calculate say a 12 month running average and a 48 month running average. You can use LAG functions in a data step.
  - But wait that won't work since you don't have the historical months in the data before `01JAN2000`.
  - Have no fear, we can use the FORMAT CATALOG to pull in the data from the FORMAT table and calculate the running averages using a macro.

```
%macro mv_avg (mnths) ;
  %let start = 1-&mnths.;
  roll&mnths= mean(
  %do i = &start %to 0;
    input(put(intnx('month',date,&i.),CPI.),best.)
    %if &i. ne 0 %then %do;
      ,
    %end;
  %end;
  )
%mend;

data test2;
  set test;
  cpi = input(put(date,CPI.),best.);
  %mv_avg(12);
  %mv_avg(48);
run;
```

See Next Page for Details of the INTNX Function

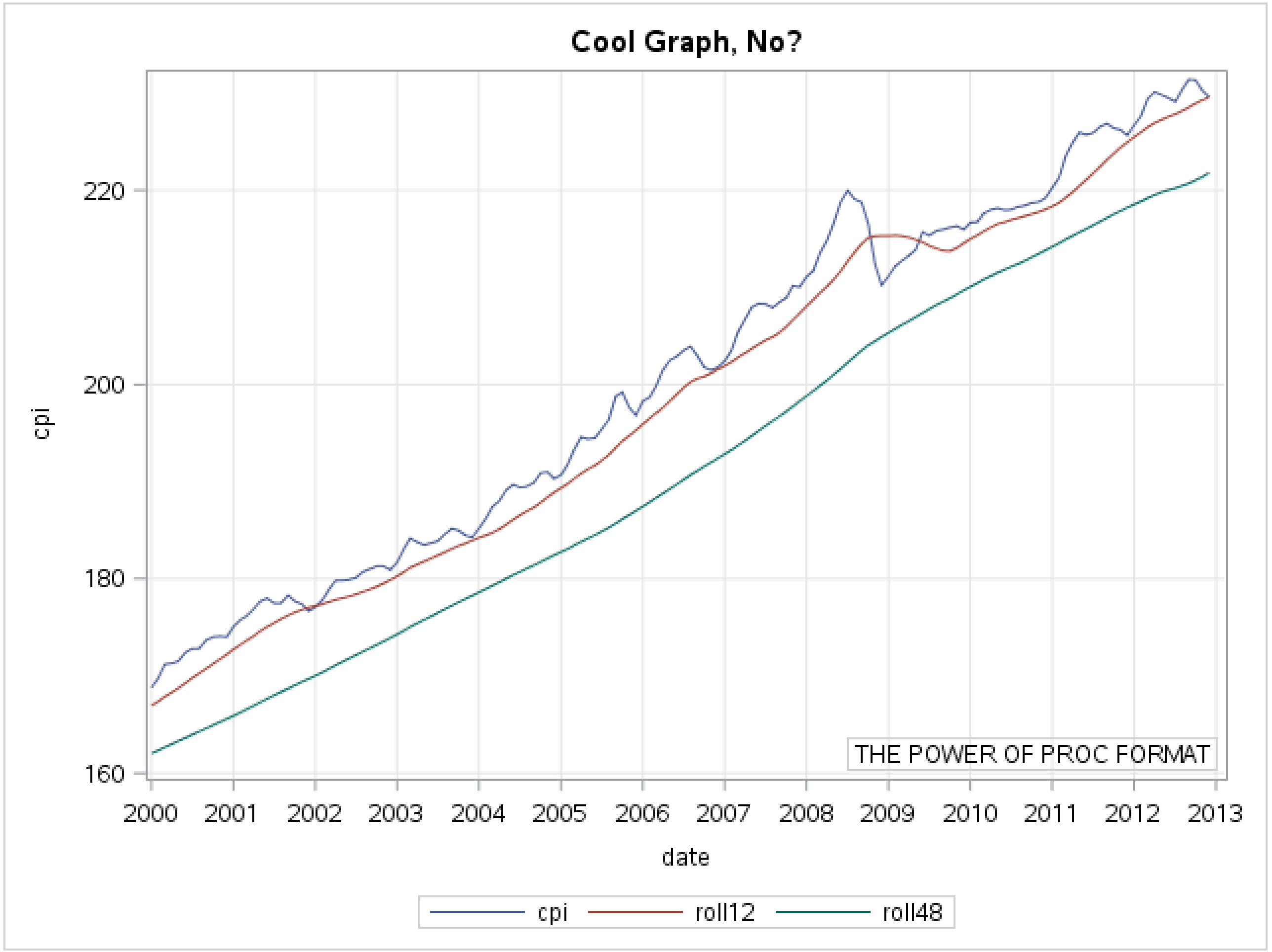
```
NOTE: There were 156 observations read from the
data set WORK.TEST.
NOTE: The data set WORK.TEST2 has 156
observations and 4 variables.
 NOTE: DATA statement used (Total process
time):
       real time           0.01 seconds
       user cpu time       0.01 seconds
       system cpu time     0.00 seconds
       memory              1674.87k
```

MORE INFO on INTNX

- The first argument in INTNX is the interval that you want to move by.
  - For date variables, the choices are; **DAY, WEEK, WEEKDAY, SEMIMONTH, MONTH, QTR, SEMIYEAR, YEAR.**
- The second argument is the start-from variable, here the variable name is date.
- The 3rd argument is the increment.
- There is a 4th argument.
  - By default, the anchor is the 4th argument; B (default) for beginning of the interval, M for middle, E for END, and S for SAME alignment as the input date.

```
input(put(intnx('month',date,&i.),CPI.),best.)
```

How to use the catalog?

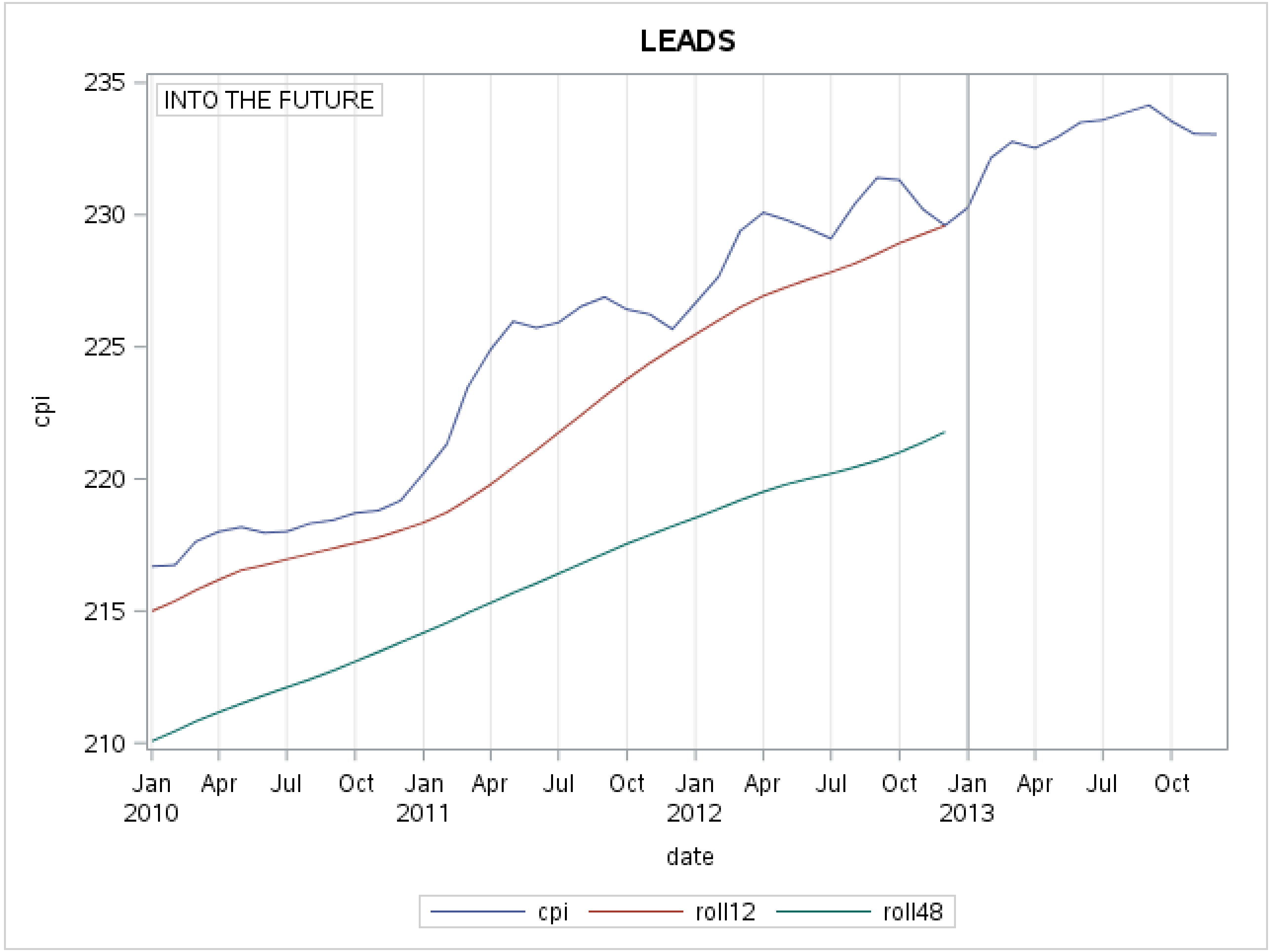


```
proc sgplot data=test2;
  series x=date y=cpi;
  series x=date y=roll12;
  series x=date y=roll48;
  xaxis discreteorder=formatted
        grid interval=YEAR;
  yaxis grid;
  format date year.;
  title Cool Graph, No?;
  inset "THE POWER OF PROC FORMAT" /
        Border position=bottomright;
 run;
```

Into the Future and Beyond

```
data test_leads;
  set test2 end=eof;
  retain date;
  output;
  if eof then do inc= 1 to 12;  /* ADD 12 MONTHs */
    date = intnx('month',date,1);
    cpi = input(put(date,CPI.),best.);
    roll12=.; roll48=.;
    output;
  end;
run;

proc sgplot data=test_leads;
  where date >= '01JAN2010'd;
  series x=date y=cpi;
  series x=date y=roll12;
  series x=date y=roll48;
  REFLINE '01JAN2013'd / axis=x;
  xaxis discreteorder=formatted grid interval=MONTH;
  yaxis grid;
  format date month. /*yymmp7. /*year.*/;
  title LEADS;
  inset "INTO THE FUTURE" /
        Border position=topleft;
run;
```

Into the Future and Beyond



Adding more than 1 FORMAT to a FORMAT Catalog

```
data stuff.cpi;
input date anydtdte10.
     index
     inflation;
  if _N_ < 10 then do;
    put date date9. +5 index +5 inflation ;
  end;
/*added 4 additional variables to the data set*/
  index80 = index*0.8;
  index90 = index*0.9;
  index110 = index*1.10;
  index120 = index*1.20;
  verybigindex34567890123456789012 = index*2;
datalines;
1913-01-01 9.8 .
1913-02-01 9.8 0.0
1913-03-01 9.8 0.0
1913-04-01 9.8 0.0
1913-05-01 9.7 -1.02
1913-06-01 9.8 1.03
```

Adding more than 1 FORMAT to a FORMAT Catalog

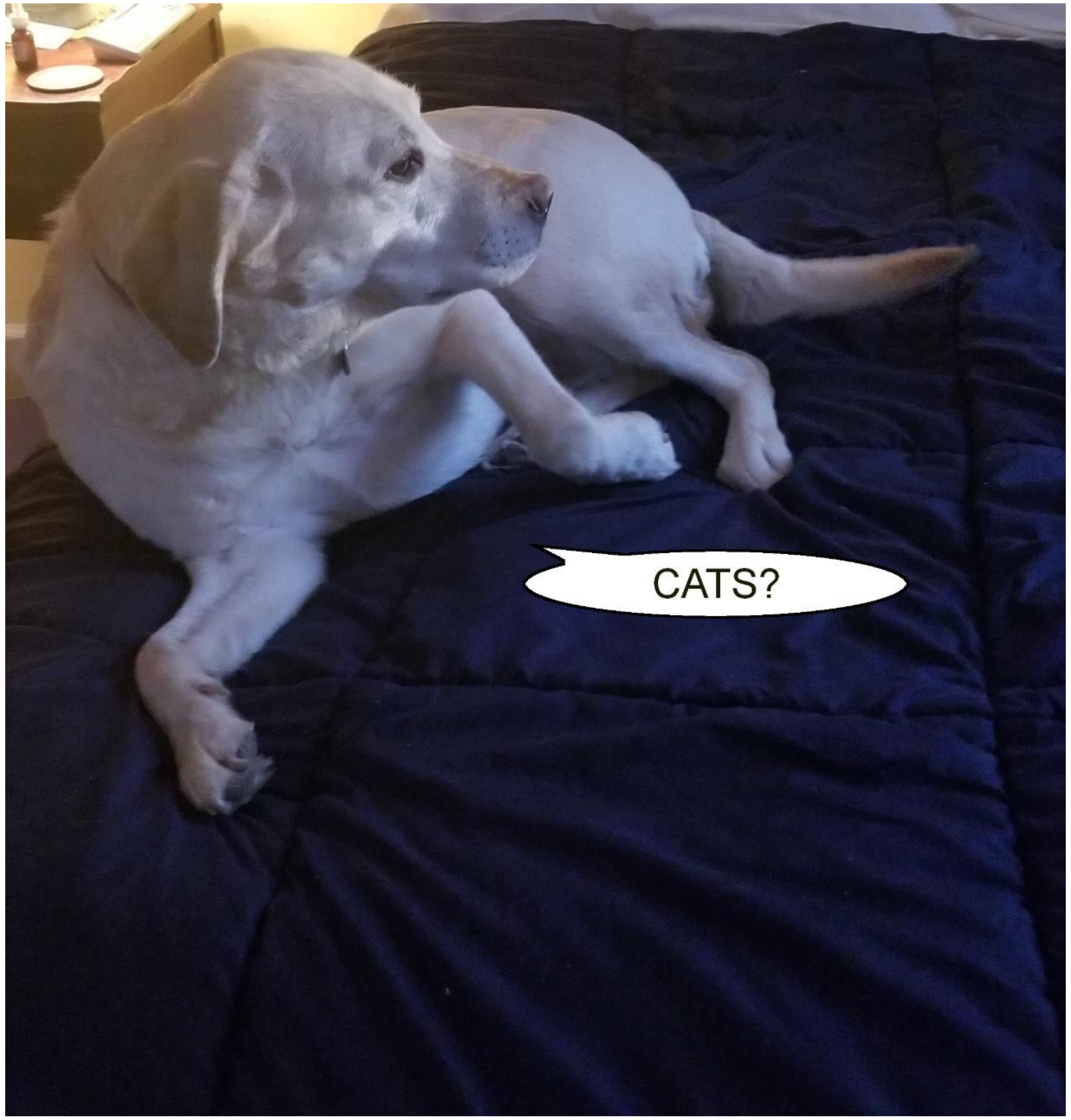
```
options nocenter fullstimer fmtsearch=(work library);
libname  stuff ‘/somewhere/FORMATS';
libname library ‘/somewhere/FORMATS';

proc contents data=stuff.cpi noprint out=cnts;
run;

proc sql noprint;
 select name into: vars separated by ' '
   from cnts
   where name not in('date' 'inflation’) and type=1
 ;
quit;
%put vars = &vars.;
```

```
NOTE: PROCEDURE CONTENTS used (Total process time):
real time 0.00 seconds
user cpu time 0.00 seconds
system cpu time 0.00 seconds
memory 1192.56k

vars = index index80 index90 index110 index120 verybigindex34567890123456789012

NOTE: PROCEDURE SQL used (Total process time):
real time 0.00 seconds
user cpu time 0.00 seconds
system cpu time 0.00 seconds
memory 5589.09k
```

Adding more than 1 FORMAT to a FORMAT Catalog

```
%macro multi_fm;
data fmt;
  length fmtname $32;
  retain type ‘N’;
  set stuff.cpi  (rename=(date=start)) end=eof;
  %let I=1;
  %do %until(%scan(&vars,&I.,%str( ) ) = %str( ) );
    %let var=%scan(&vars,&I.,%str( ) );
    %let v=&var.;
    %IF %LENGTH(&v.)>30 %THEN %DO;
      %let v=%SUBSTR(&v.,1,30);
      fmtname=cats("&v.",'_F');
    %END;
    %else %DO;
      fmtname=cats("&var.",'_F');
    %end;
    label=&var.;
    output;
   %let I = %eval(&I. + 1);
  %end;
```

Adding more than 1 FORMAT to a FORMAT Catalog



Adding more than 1 FORMAT to a FORMAT Catalog

```
if eof then do
  %let i=1;
  %do %until(%scan(&vars,&I.,%str( ) ) = %str( ) );
    %let var=%scan(&vars,&I.,%str( ) );
    %let v=&var.;
     %IF %LENGTH(&v.)>30 %THEN %DO;
      %let v=%SUBSTR(&v.,1,30);
      fmtname=cats("&v.",'_F');
    %END;
    %else %DO;
      fmtname=cats("&var.",'_F');
    %end;
    /*start=.;*/
    label=.;
    HLO='O';
    output;
    %let I = %eval(&I. + 1);
  %end;
end;
run;
%mend;
%multi_fm;
```

```
NOTE: There were 1213 observations read from the data set STUFF.CPI.
NOTE: The data set WORK.FMT has 7284 observations and 11 variables.
NOTE: DATA statement used (Total process time):
real time 0.00 seconds
user cpu time 0.01 seconds
system cpu time 0.00 seconds
memory 1908.56k
```

Adding more than 1 FORMAT to a FORMAT Catalog

```
proc sort data=fmt;
  by fmtname;
run;

proc format cntlin=fmt library=library.macroeco;
run;


proc catalog c=library.macroeco;
  contents;
run;

/*proc format fmtlib library=library.macroeco;
  select index_F;
run;*/
```

```
NOTE: PROCEDURE SORT used (Total process time):
      real time           0.00 seconds
      user cpu time       0.01 seconds
      system cpu time     0.00 seconds
      memory              2300.96k
```

```
NOTE: PROCEDURE FORMAT used (Total process time):
      real time           0.03 seconds
      user cpu time       0.01 seconds
      system cpu time     0.00 seconds
      memory              1037.87k
```

```
NOTE: PROCEDURE CATALOG used (Total process time):
      real time           0.02 seconds
      user cpu time       0.02 seconds
      system cpu time     0.01 seconds
      memory              3044.56k
```

Contents of Catalog LIBRARY.MACROECO

#	Name	Type	Create Date	Modified Date	Description
1	INDEX110_F	FORMAT	05/01/2019 17:08:19	05/01/2019 17:08:19	
2	INDEX120_F	FORMAT	05/01/2019 17:08:19	05/01/2019 17:08:19	
3	INDEX80_F	FORMAT	05/01/2019 17:08:19	05/01/2019 17:08:19	
4	INDEX90_F	FORMAT	05/01/2019 17:08:19	05/01/2019 17:08:19	
5	INDEX_F	FORMAT	05/01/2019 17:08:19	05/01/2019 17:08:19	
6	VERYBIGINDEX345678901234567890_F	FORMAT	05/01/2019 17:08:19	05/01/2019 17:08:19	

References & Recommended Reading

- Bilenas, J.V. “The Power of PROC FORMAT”, SAS Press, 2005. No longer in print.
- Bilenas, J.V., Tahiliani, K. (2016). “The Power of Proc Format, http://analytics.ncsu.edu/sesug/2016/BB-103_Final_PDF.pdf
- Bilenas, J.V., Tahiliani, K. (2019). “The Power of Proc Format, PharmaSUG 2019
- Bilenas, J.V. (2007) “Using SAS® Dates and Times – A Tutorial “, SAS GLOBAL FORUM 2007, https://support.sas.com/resources/papers/proceedings/proceedings/forum2007/226-2007.pdf
- Carpenter, A. “Looking for a Date? A Tutorial on Using SAS Dates and Times”, SUGI30, 2005.
- Laffler,K.P.(2009) “Exploring the Undocumented PROC SQL _METHOD Option”, SAS GLOBAL FORUM 2009, http://support.sas.com/resources/papers/proceedings09/063-2009.pdf
- Liang, Q. (2009) “Choosing the Right Technique to Merge Large Data Sets Efficiently”, SAS GLOBAL FORUM 2009, http://support.sas.com/resources/papers/proceedings09/071-2009.pdf
- Morgan, D. “The Essential Guide to SAS Dates and Times“, SAS Press. 2006
- Shipp, C.E., Laffler K.P. (2013) “Exploring the PROC SQL _METHOD Option”, SAS GLOBAL FORUM 2013, https://support.sas.com/resources/papers/proceedings13/200-2013.pdf
- Tian, Y. (2009), “LAG - the Very Powerful and Easily Misused SAS® Function”, SAS GLOBAL FORUM 2009, http://support.sas.com/resources/papers/proceedings09/055-2009.pdf

<u>May the “Power of PROC FORMAT” be with you!</u>

SAS® is a registered trademark of SAS Institute.

The contents of this paper are the work of the authors and do not necessarily represent the opinions, recommendations, or practices of current or previous employers.

The Power of

PROC FORMAT

ART CARPENTER'S SAS SOFTWARE SERIES

Jonas V. Bilenas

sas