Paper 128-2023

Automatically Creating Dated Archives with SAS®

Derek Morgan, Bristol Myers Squibb

ABSTRACT

When creating patient profiles, it can be useful for clinical scientists to compare current data with previous data in real time without having to request those data from an Information Technology (IT) source. This is a method for using SAS® to perform the archiving via a daily scheduled job. The primary advantage of SAS over an operating script is its date handling ability, removing many difficult calculations in favor of intervals and functions. This paper details an application that creates dated archive folders and copies SAS data sets into those dated archives, with automated aging and deletion of old data and folders. The application allows clinical scientists to customize their archive frequency (within certain limits.) It also keeps storage requirements to a minimum as defined by IT. This replaced a manual process that required study programmers to create the archives, eliminating the possibility of missed or incorrectly dated archives. The flexibility required for this project and the conditions under which it ran required using SAS date and time intervals, their functions, and manipulating of files and directories with SAS.

INTRODUCTION

A SAS solution was requested after failed attempts to build this as a bash script. SAS has advanced capabilities for handling dates, which is important because this process is driven by timing. SAS can handle the creation of folders, copying of SAS datasets, removing datasets, and removing folders without having to invoke LINUX commands, so it satisfied the base requirements as a tool.

The solution involves SAS date intervals, the ability to multiply and shift intervals, and the INTCK() interval counting function. File manipulation is handled with PROC DATASETS, PROC COPY, and SAS directory and file functions. It is all tied together by the SAS macro facility. The solution also uses a couple of handy little macros: %ISBLANK determines if a macro variable is blank. %OBSCOUNT determines how many records are in a SAS dataset and returns -1 if the dataset doesn't exist. The code for %OBSCOUNT is included in Appendix B. See the references for more information about %ISBLANK.

REQUIREMENTS

The real-world use case has several requirements:

- The archive process runs every day as a scheduled job.
- Archives are only created on the appropriate day, and under the appropriate conditions.
- Archives can be created any day of the week. It can't be assumed that archives will be done only on Mondays.
- Programmers are not restricted to implementing the schedule on a Monday. They can set the schedule on any day of the week.
- The most recent data refresh within a snapshot period must be archived. Clinical science can request a data refresh at any time. It does not have to be on a set schedule.
- Data snapshots can be taken weekly, semimonthly, or monthly, as defined by clinical science's needs, therefore, can't hard code the process to run every 7 days.
- Each individual study must be able to set its own frequency of archiving and retention. Users must also be able to change them based on their needs.
- Archives must be retained on a weekly, semimonthly, monthly, and/or quarterly basis according to a frequency set by clinical science, and old archives must be removed from the system.
- Any dated manual archives created prior to the implementation of this automated process must be incorporated into the process.

- Special archives must be excluded from the process.

The datasets to be archived are refreshed by a separate system job on an irregular basis, and the archiving must take place after the data are refreshed. Even if the data were refreshed every Monday, that schedule would have to include company holidays. Otherwise, it would have to be delayed until the next business day. In addition, technical issues such as power loss or natural disaster can delay that expected data refresh to Wednesday or Thursday. Ultimately, if the archiving only runs on Mondays, any data refresh that doesn't happen on Monday won't have archives. In short, scheduling to run every seven days starting on a Monday won't work.

USER INPUT

This runs as a timed job, so an interactive front end is not possible. Studies fill out an Excel spreadsheet template (.XLTX) and store it as an Excel spreadsheet. All values except Study Name and Notification List are selected from drop down lists. The template is shown in Figure 1:

Figure 1: Excel Template for Auto-Archive Set-up

Name	Value	Instructions
Study Name		The study name
Notification List		Email address(es) of people who should receive the activity report by email. You may provide up to 5 addresses, separated by semicolons.
Snapshot Frequency	Monthly	How often you want the archive created. Your choices are: Weekly, Semimonthly (every 2 weeks), and Monthly (every 4 weeks). The default is Weekly.
Weekly archives to retain	0	The number of weekly archives to retain if you've selected a weekly archive frequency. Must be a value between 0 and 3. The default is 3. This will be ignored if you aren't archiving weekly.
Semimonthly archives to retain	0	The number of semimonthly (every two weeks) archives to retain if you've selected a semimonthly snapshot frequency. Must be a value between 0 and 7. The default is 0, and this only takes effect if you're archiving semimonthly.
Monthly archives to retain	3	The number of monthly (every 4 weeks) archives to retain. Must be a value between 0 and 3. The default is 3. This will be ignored if you are archiving semimonthly.
Quarterly Archives to retain	3	The number of quarterly (every 12 weeks) archives to retain. The default is 3. Must be a value between 0 and 3.

This spreadsheet allows studies to specify the internal study name and add an email notification list for any issues that may occur during the process. Users also specify the snapshot frequency (weekly, semimonthly, or monthly), and how many of each type of archive (weekly, semimonthly, monthly, quarterly) they wish to retain, up to the IT-specified limit of nine, or twelve if they only wish to retain monthly archives. Either way, clinical science can have on-line access up to a year's worth of dated archives from the most recent to the oldest.

This only applies to dated archives. If a study wants to preserve a manually created archive (e.g., aligning with a study milestone), they must use something other than a date for the folder name.

TIMING AND ARCHIVE CONTROL

Since each study can have different timing, the code cannot refer to absolute timings, specific dates, or days of the week. The user team chose timing in weeks instead of actual months and quarters, to reflect the clinical data process more accurately. This also made timing easier. Defining timing based on weeks makes our SAS interval definitions WEEK.2, WEEK2.2 (semimonthly), WEEK4.2 (monthly), and WEEK12.2 (quarterly). The number after the interval name ("WEEK") multiplies the interval, and ".2" tells SAS that our week starts on Monday instead of the default, Sunday.

The program uses an archive control dataset to track the archives. This dataset has two variables: archive date and type. There is one record per retained archive. Since a maximum of 12 archives can be retained, this dataset can only have 12 records at most. When an archive is rolled off, its archive control record is deleted. When an archive is "promoted" (e.g., a weekly archive becomes a monthly archive,) the archive control record is updated with the new archive type.

MODULAR DESIGN

A simplified process flow diagram is in Appendix A. The program was constructed as a series of macros designed to perform one or two specific tasks. This made it easy to debug each function separately and run unit tests. Although there is a little extra overhead associated with this approach, it also made it easy to create a simulation program to test the entire process. We will discuss each macro in-depth. The modules are:

- **%DVARC**: The main macro and entry point into the process. It reads and validates user parameters. The first time it runs, it creates the archive control and activity log datasets, then incorporates any existing dated archives into the process. The determination on whether to make an archive takes place in %DVARC, and the archive is created in this macro. If an archive is to be created, it makes the directory using the current date in extended ISO8601 format (YYYY-MM-DD) for the directory name, and copies the files being archived into that directory. It then calls %AGESNAPS, the aging macro. Finally, it removes any records from the activity log that are more than a year old.
- **%AGESNAPS**: This macro does the heavy lifting. It determines whether an existing archive should be promoted to the next level (e.g., weekly to monthly), or removed. The rules are different for each specified archive frequency. The weekly and semimonthly frequencies are mutually exclusive, so if weekly or semimonthly, we need to process the existing weekly/semimonthly archives as well as any monthly and quarterly archives. If archiving is done monthly, then we only process existing monthly and quarterly archives.
- **%GETMAXMINDATES**: Calculates the maximum and minimum dates for each type of archive (weekly, semimonthly, monthly, and monthly) and places them in global macro variables.
- **%AGEARCHIVE**: Updates an archive control record for a specific dated archive with a new archive type.
- **%RMVARCHIVE**: Deletes expired archive files and folders. Once the dated archive directory is empty, it can then be removed.

DVARC

First, the program processes the user's Excel file and sets up macro variables that control archive frequency and retention. Code Fragment 1 demonstrates how the user input is pulled into the process' global macro space. Line 16 uses the $dvsetparam informat to create the macro variable names based on the text in the first column from the standard Excel spreadsheet.

Code Fragment 1

```
/* set-up name/value pairs */
1.   PROC FORMAT;
2.   INVALUE $dvsetparam
3.   'Study Name'='studynam'
4.   'Notification List'='emailList'
5.   'Snapshot Frequency'='snapfreq'
6.   'Weekly archives to retain'='wk'
7.   'Semimonthly archives to retain'='semimth'
8.   'Monthly archives to retain'='mth'
9.   'Quarterly Archives to retain'='quarter'
10.  OTHER = _ERROR_
11.  ;
12.  RUN;

13.  /* put user-defined parameters into program space */
14.  DATA _NULL_;
15.  SET import; /* Dataset from imported Excel file */
16.  CALL SYMPUTX(INPUT(name,$dvsetparam.),STRIP(value));

17.  /* process snapshot frequency parameter */
18.  IF INPUT(name,$dvsetparam.) EQ 'snapfreq' THEN DO;
19.     snapabbr = SUBSTR(value,1,1);
20.     CALL SYMPUTX('snapfreq',snapabbr);
21.     CALL SYMPUTX('snaptext',lowcase(value));
22.     SELECT(snapabbr); /* define interval based on snapshot frequency */
23.       WHEN('W') CALL SYMPUTX('intvl','week.2');
24.       WHEN('S') CALL SYMPUTX('intvl','week2.2');
25.       WHEN('M') CALL SYMPUTX('intvl','week4.2');
26.       WHEN('Q') CALL SYMPUTX('intvl','week12.2');
27.       OTHERWISE DO;
28.          ABORT RETURN;
29.       END;
30.     END;
31.  END;
32.  RUN;
```

Lines 18 through 31 create macro variables for the snapshot key (‘W’,’S’,’M’ or ‘Q’), the interval to use (&INTVL), and the full text of the snapshot frequency, which is used in the activity record dataset. After these have been set up, the user parameters are checked to ensure users haven’t asked for retention options incompatible with their chosen archive frequency, such as requesting weekly archive retention when only taking monthly snapshots. It also keeps them from retaining more than the maximum number of archives allowed.

Next, the program gets a list of dated archive folders to determine the date of the most recent archive. Because there are many ways to do it, it isn’t included in this paper.

If the archive control dataset does not exist, then this is the first time the automated process has run, and it must create empty archive control and activity log datasets. The first time the automated process runs, any existing dated archives also must be incorporated into the automated process. To do this, each existing archive is classified as “weekly”, “semimonthly”, “monthly”, or “quarterly” based on the date of the archive (obtained from the folder name). This code was designed to work with intervals, not hardcoded days, just in case the user team changes their mind about basing the archiving solely on weeks, as shown in Code Fragment 2:

Code Fragment 2

```
1.   DATA pre_xist;
2.   SET alldirs (KEEP=dirdate RENAME=(dirdate=date));
3.   LENGTH archiveType $ 1 intvl $ 10;
4.   intvl = STRIP(SYMGET('intvl'));
5.   approxage = INTCK(intvl,date,&rundt);
6.   SELECT(SYMGET('snapfreq'));
7.     WHEN('W') DO;
8.        IF approxage GE 12 THEN
9.           archiveType = 'Q';
10.       ELSE IF approxage GT 4 THEN
11.          archiveType = 'M';
12.       ELSE
13.          archiveType = 'W';
14.   END;
15.   WHEN('S') DO;
16.      IF approxage GE 6 THEN
17.         archiveType = 'Q';
18.      ELSE
19.         archiveType = 'S';
20.   END;
21.   WHEN('M') DO;
22.      IF approxage GE 3 THEN
23.         archiveType = 'Q';
24.      ELSE
25.         archiveType = 'M';
26.   END;
27.   OTHERWISE archiveType = 'Q';
28.  END;
29.  DROP intvl approxage;
30.  RUN;
```

Line 5 calculates how many intervals have elapsed based on the interval of the user's snapshot frequency (&INTVL), the date of the archived folder, and the current date (&RUNDT). From there, it just a matter of counting. If the snapshots are being done weekly, then an archive twelve or more weeks old is classified as a quarterly archive; if the archive is four to eleven weeks old, it's a monthly archive; otherwise, it's a weekly archive. Each snapshot frequency has corresponding classification code. These records are then added to the empty archive control dataset. If there are no dated archive folders, then there is nothing to add. This ends the segment of code that only executes the first time the process is run.

The standard part of the process begins with calculating the date of the most recent archive from the list of existing archives using an SQL query. When referring to dates in the code, the suffix "_H" indicates it is the formatted version of the corresponding variable holding the SAS date to. Therefore, if RUNDT is the current date as a SAS date value (e.g., 23217), RUNDT_H is the current date in extended ISO format, (e.g., 2023-07-26). "H" stands for "human readable." Line 2 in Code Fragment 3 puts the most recent date (as a SAS date value and as a formatted ISO date) into left-justified and trimmed macro variables:

Code Fragment 3

```
1. PROC SQL NOPRINT;
2. SELECT STRIP(PUT(MAX(dirdate),12.)), STRIP(PUT(MAX(dirdate),E8601DA.))
   INTO :lastarcdt, lastarcdt_h
3. FROM alldirs
4. ;
5. QUIT;
```

Finally, it's time to determine if enough time has elapsed to create a new archive. The next version will also test to see if any data has changed. No archive will be created unless data has changed AND enough time has elapsed. For now, however, the code is:

```
%IF %SYSFUNC(INTCK(&intvl,&lastarcdt,&rundt)) %THEN %DO;
```

The INTCK() function returns the number of &INTVL (WEEK.2, WEEK2.2, etc.) intervals between the date of the last archive (&LASTARCDT) and the current date (&RUNDT). If it's greater than zero, then at least one interval has elapsed, so it's time to create the archive. The first step is to create the dated folder:

```
%LET NEWDIR=%SYSFUNC(DCREATE(&rundt_h,/data/&project));
```

Only the data files are to be copied, not the files that the archive process uses, so the program builds an exclusion list based on the existence of these files. DV_ARCHV_CTRL will always exist at this point, so there's always at least one file to be excluded. Code Fragment 4 accomplishes this in a simple way with the FILEEXIST() function and creating an exclusion list of dataset names in &EXCLDSTR.

Code Fragment 4

```
%IF %SYSFUNC(FILEEXIST(data/&project/dv_archv_ctrl.sas7bdat)) %THEN
    %LET excldstr = dv_archv_ctrl;
%IF %SYSFUNC(FILEEXIST(data/&project/action_record.sas7bdat)) %THEN
    %LET excldstr = &excldstr action_record;
%IF %SYSFUNC(FILEEXIST(data/&project/dv_dataspecs.sas7bdat)) %THEN
    %LET excldstr = &excldstr dv_dataspecs;
```

The macro variable &EXCLDSTR is used with PROC COPY in line 3 of Code Fragment 5:

Code Fragment 5

```
1. LIBNAME curarchv /data/&project/&rundt_h";
2. PROC COPY IN=dmodel out=curarchv datecopy;
3. EXCLUDE &excldstr;
4. RUN;
5. LIBNAME curarchv CLEAR;
```

After the copy is made, it calls %AGESNAPS. The rest is housekeeping: creating new records for the archive control and activity tracking datasets and then appending them, shown in Code Fragment 6:

Code Fragment 6

```
1.   /***** Archive Control *****/
2.   DATA weekinfo;
3.   LENGTH date 4 archiveType $ 1;
4.   FORMAT date E8601DA.;
5.   INFORMAT date E8601DA.;
6.   LABEL archiveType="Archive type (W,S,M,Q)"
7.         date = 'Archive Date'
8.   ;
9.   archiveType = SUBSTR(SYMGET('snapfreq'),1,1);
10.  date = &rundt;
11.  RUN;

12.  PROC APPEND BASE=dmodel.&archv_ctrl DATA=weekinfo;
13.  RUN;

14.  /***** Activity Tracker *****/
15.  DATA daily_record;
16.  LENGTH date 4 action $ 128;
17.  date = &rundt;
18.  action = CATX(' ',SYMGET('rundt_h'),SYMGET('snaptext'),
19.                 "archive created on",PUT(date,WEEKDATE32.));
20.  RUN;

21.  PROC APPEND BASE=dmodel.action_record DATA=daily_record;
22.  RUN;
```

Since the process is designed to run every day, there's still housekeeping to be done if not enough time has elapsed to create an archive. No archive control record is created, but a record is created for the activity log stating that nothing was done. The last task in the main module is to remove any records over a year old in the activity tracker, shown in Code Fragment 7:

Code Fragment 7

```
1. DATA dmodel.action_record;
2. SET dmodel.action_record;
3. IF INTCK('year',date,&rundt,'CONTINUOUS') EQ 1 THEN
4.    DELETE;
5. RUN;
```

The INTCK() function with the CONTINUOUS parameter (line 3) calculates one calendar year from the current day. Without this parameter, it would calculate intervals the way SAS has always calculated intervals, from interval starting point of the start date to the interval starting point of the end date. Here, if an activity log record has a date at least one year older than the current date, it's deleted.

%DVARC SUMMARY

Most of this macro is dedicated to processing user input and process set-up. Creating folders and copying files is the easy part, using the SAS file functions FILEEXIST() and DCREATE(), then PROC COPY. Adding dates and enforcing periods of time between archives is simply a matter of using dates, formatted dates, and SAS intervals. The dates are used to determine how much time has elapsed, the formatted dates for folder naming (and debugging), and the intervals count for you instead of relying on a specific number of days.

AGESNAPS

This macro determines whether an archive folder is “aged” or removed. Archives are aged by changing an archive’s type in the archive control dataset. Each archive type (weekly, semimonthly, monthly, quarterly) must be processed separately. The process is triggered by counting the number of existing archives of each type and checking how much time has elapsed between the oldest archive at the lowest frequency and the most recent archive of the next highest frequency. If there are weekly or semimonthly archives, then you compare those dates against the monthly archive dates. Similarly, you compare monthly archive dates against quarterly ones. Quarterly archives are simple: if there are more quarterly archives than should be retained, the oldest is removed.

Table 1 shows a partial example of how this works, using weekly snapshots and a starting date of January 16, 2023. In this example, three weekly, monthly, and quarterly archives each are to be retained. There isn’t enough room to show the retention of the three quarterly archives here, but a full example for one year is in Appendix C. It also shows the removal of the oldest quarterly archive after a year.

Table 1: Promoting Weekly Archives

Week #	Archive Date	Weekly Archive 1	Weekly Archive 2	Weekly Archive 3	Monthly Archive 1	Monthly Archive 2	Monthly Archive 3	Quarterly Archive 1
1	1/16/2023							
2	1/23/2023	2023-01-16						
3	1/30/2023	2023-01-16	2023-01-23					
4	2/6/2023	2023-01-16	2023-01-23	2023-01-30				
5	2/13/2023	2023-01-23	2023-01-30	2023-02-06	2023-01-16			
6	2/20/2023	2023-01-30	2023-02-06	2023-02-13	2023-01-16			
7	2/27/2023	2023-02-06	2023-02-13	2023-02-20	2023-01-16			
8	3/6/2023	2023-02-13	2023-02-20	2023-02-27	2023-01-16			
9	3/13/2023	2023-02-20	2023-02-27	2023-03-06	2023-01-16	2023-02-13		
10	3/20/2023	2023-02-27	2023-03-06	2023-03-13	2023-01-16	2023-02-13		
11	3/27/2023	2023-03-06	2023-03-13	2023-03-20	2023-01-16	2023-02-13		
12	4/3/2023	2023-03-13	2023-03-20	2023-03-27	2023-01-16	2023-02-13		
13	4/10/2023	2023-03-20	2023-03-27	2023-04-03	2023-01-16	2023-02-13	2023-03-13	
14	4/17/2023	2023-03-27	2023-04-03	2023-04-10	2023-01-16	2023-02-13	2023-03-13	
15	4/24/2023	2023-04-03	2023-04-10	2023-04-17	2023-01-16	2023-02-13	2023-03-13	
16	5/1/2023	2023-04-10	2023-04-17	2023-04-24	2023-01-16	2023-02-13	2023-03-13	
17	5/8/2023	2023-04-17	2023-04-24	2023-05-01	2023-02-13	2023-03-13	2023-04-03	2023-01-16

No archive needs to be created for the first week. Each previous week’s snapshot is retained as a weekly archive in weeks 2-4. Week 5 would give us four weekly archives, exceeding the limit of three. Instead of removing the oldest weekly archive, January 16, it is promoted to a monthly archive, and the previous three weeks’ archives are still retained as weekly ones.

In week 6, there are again more than three weekly archives: January 23, January 30, February 6, and the newest one, February 13. The oldest weekly archive, January 23, is removed to make room for the newest. The next week, the January 30 archive is removed, and so on until week 9. The February 13 archive is promoted to monthly in week 9, providing two monthly archives and three weekly archives. The week 6 process is repeated in week 10, and the February 20 weekly archive is removed. Finally, in week 17, the April 3 archive becomes a monthly one. Now there are four monthly archives from January, February, March, and April, so the January 16 archive is promoted again, this time to a quarterly archive. This repeats until the fourth quarterly archive is created, and the oldest quarterly archive (from January 16, 2023) will be removed.

For this to work, the most recent and the latest dates for each type of archive are stored in macro variables. When working with SAS date (or datetime) values, it’s handy to remember that the earliest one is the maximum value and the oldest is the minimum value. I used PROC MEANS and a DATA step to fill

the macro variables instead of a SQL query because each macro variable suffix must correspond to a specific archive type as shown in Table 2:

Table 2: Numeric Values for Archive Types

1	Weekly (W)
2	Semimonthly (S)
3	Monthly (M)
4	Quarterly (Q)

The SQL query in Code Fragment 8 will only fill the macro variables &MINGRPDT1-&MINGRPDT4 with the results it returns:

Code Fragment 8

```
1.   PROC SQL NOPRINT;
2.   SELECT MIN(date) INTO :mingrpdt1-:mingrpdt4
3.   FROM dmodel.&archv_ctrl
4.   GROUP BY type;
```

If all four levels exist, this will work as expected. However, if any level is missing, the macro variables &MINGRPDT1-&MINGRPDT3 will be filled, and &MINGRPDT4 will be missing. If that level is an intermediate one (such as semimonthly), then the correspondence would be as follows:

1	Weekly (W)
2	Monthly (M)
3	Quarterly (Q)
4	(missing)

At least one archive type is always missing because weekly and semimonthly are mutually exclusive. Therefore, &MINGRPDT2 will correspond with “monthly”, not “semimonthly”. This trap can be avoided by using the method in Figure 2:

Figure 2: Getting Maximum and Minimum Dates for Each Archive Type

```
1.   PROC SORT DATA=dmodel.&archv_ctrl OUT=getdates;
2.   BY archiveType date;
3.   RUN;

4.   PROC MEANS DATA=getdates NOPRINT MIN MAX;
5.   BY archiveType;
6.   VAR date;
7.   OUTPUT OUT=grpdates (DROP=_TYPE_ _FREQ_) MIN=mingdt MAX=maxgdt;
8.   RUN;

9.   DATA _NULL_;
10.  SET grpdates;
11.  indx = INPUT(archiveType,retprio.);
12.  CALL SYMPUTX(CATS('mingrpdt',PUT(indx,1.)),STRIP(PUT(mingdt,8.)));
13.  CALL SYMPUTX(CATS('maxgrpdt',PUT(indx,1.)),STRIP(PUT(maxgdt,8.)));
14.  RUN;
```

Line 11 gets the numeric index for each archive type from an informat. Line 12 assigns the minimum date for each archive type to the macro variables &MINGRPDT1-&MINGRPDT4, and line 13 assigns the maximum dates the same way. The CATS() function creates the macro variable name, and the STRIP() function left justifies the number, which is a SAS date value.

AGING AND ARCHIVE REMOVAL

Now for the actual aging/deletion process. Each type of archive must be processed individually. If weekly snapshots are taken, then weekly, monthly, and quarterly archives must also be processed. Semimonthly archives aren't processed for weekly snapshots because weekly and semimonthly archiving is mutually exclusive. Figure 3 shows the full code that processes weekly snapshots:

Figure 3: Aging/Deletion Algorithm for Weekly Snapshot Frequencies

```
%IF &snapfreq EQ W %THEN %DO;
/* How many weekly archives exist? */
  PROC SQL NOPRINT;
  SELECT COUNT(date) INTO :wkArchv
  FROM dmodel.&archv_ctrl
  WHERE archiveType EQ "W";
  QUIT;

/* Runs after data refresh, so add 1 to ignore current
   week's archive */

  %IF &wkArchv GT %EVAL(&wk+1) %THEN %DO;
    %getmaxmindates; /* get maximum and minimum dates */

    %IF %isblank(&maxgrpdt3) OR NOT %SYMEXIST(maxgrpdt3) %THEN %DO;
      %LET elapwks = 0; * no monthly archives - skip weekly deletion;
    %END;
    %ELSE %DO; /* CALCULATE ELAPSED WEEKS */
      %LET elapwks = %SYSFUNC(INTCK(week.2,&maxgrpdt3,&mingrpdt1));
    %END;

 /* Is oldest wk archive >4 weeks after newest monthly archive? */
  %IF &elapwks LT %EVAL(&wk+1) AND &elapwks NE 0 %THEN %DO;
      /* delete oldest week archive */
    %rmvArchive(type=W,rmvdate=&mingrpdt1);
  %END;
  %ELSE %DO; /* promote oldest week archive to monthly */
    %ageArchive(type=M,agedate=&mingrpdt1); *update archive control
                                             record;
      /* save action to activity log */
    DATA daily_record;
    LENGTH date 4 action $ 128;
    date = &rundt;
    action = CATX(' ',PUT(&mingrpdt1,e8601da.),
                  "weekly archive retained as monthly archive on",
                   PUT(date,weekdate32.));
    RUN;

    PROC APPEND BASE=dmodel.action_record DATA=daily_record;
    RUN;
  %END;
```

```
37.  /* Now process existing monthly archives. Same as weekly, just using
38.     monthly/quarterly parameters */

39.    /* how many monthly archives? */
40.     PROC SQL NOPRINT;
41.     SELECT COUNT(date) AS mthArchv INTO :mthArchv
42.     FROM dmodel.&archv_ctrl
43.     WHERE archiveType EQ 'M';
44.     QUIT;

45.  /* more than maximum # monthly archives? Don't need to add 1. Current
     snapshot frequency is weekly */
46.     %IF &mthArchv GT &mth %THEN %DO;
47.       %getmaxmindates; /* get maximum and minimum dates (redundant?) */
48.       %IF %isblank(&maxgrpdt4) OR NOT %SYMEXIST(maxgrpdt4) %THEN %DO;
49.         %LET elapmo = 0; * skip month deletion if no qtrly archives;
50.       %END;
51.       %ELSE %DO; /* calc elapsed months (month = 4 weeks) */
52.         %LET elapmo = %SYSFUNC(INTCK(week4.2,&maxgrpdt4,&mingrpdt3));
53.       %END;

54.  /* Oldest monthly archive >4 months after newest quarterly archive? */
55.       %IF &elapmo LT %eval(&mth) AND &elapmo ne 0 %THEN %DO;
56.         %rmvArchive(type=M,rmvdate=&mingrpdt3); /* delete oldest
57.                                                    monthly archive */
58.       %END;
59.       %ELSE %DO;
60.         %ageArchive(type=Q,agedate=&mingrpdt3);/* promote oldest month
61.                                                 archive to quarterly */
62.         DATA daily_record;
63.         LENGTH date 4 action $ 128;
64.         date = &rundt;
65.         action = CATX(' ',PUT(&mingrpdt3,e8601da.),
66.                       "monthly archive retained and promoted to
67.                        quarterly archive on", PUT(date,weekdate32.));
68.         RUN;

69.         PROC APPEND BASE=dmodel.action_record DATA=daily_record;
70.         RUN;
71.       %END;
72.     %END; /*  %IF &mthArchv ge &mth */
```

```
73.  /* Do quarterly - only roll off because there's no yearly) */
74.  /* how many quarterly archives? */
75.     PROC SQL NOPRINT;
76.     SELECT COUNT(date) AS qtrArchv INTO :qtrArchv
77.     FROM dmodel.&archv_ctrl
78.     WHERE archiveType EQ 'Q';
79.     QUIT;

80.     %IF &qtrArchv gt &quarter %THEN %DO; * > max # of qtr archives?;
81.       %getmaxmindates; /* get maximum/minimum dates (redundant?) */
82.       %rmvArchive(type=Q,rmvdate=&mingrpdt4); /* remove oldest
83.                                                  quarterly archive */
84.       DATA daily_record;
85.       LENGTH date 4 action $ 128;
86.       date = &rundt;
87.       action = catx(' ',put(&mingrpdt4,e8601da.),
88.                     "quarterly archive removed on",
89.                     PUT(date,weekdate32.));
90.       RUN;

91.       PROC APPEND BASE=dmodel.action_record DATA=daily_record;
92.       RUN;
93.     %END;
94.    %END;  /* %IF &wkArchv ge %eval(&wk+1)  */
95.  %END; /* %IF &snapfrEQ EQ W  */
```

First, figure out if there are too many weekly archives. That comes from the archive control dataset (lines 2-7). The current snapshot archive doesn't count toward the number of archives to retain, so the aging/deletion process isn't triggered for these until there are more than the number of weekly archives the study team wants to retain plus one (line 10). If the study team wants to retain three weekly archives, the aging vs. deletion decision occurs for that oldest archive when a fifth weekly archive is added. See week 5 in Table 1 for an example. Archives have been created for January 16, January 30, February 6, February 13, and the current date, February 20, so a decision must be made for the January 16 archive. The following week, weekly archives exist for January 30, February 6, February 13, February 20, and February 27, and now the decision must be made for the January 30 archive.

Lines 12-17 determine how many weeks have elapsed between the most recent monthly archive (stored in &MAXGRPDT3), and the oldest weekly archive (in &MINGRPDT1.) If no monthly archives exist, then there's no worry about aging or deleting the weekly archives, and the count is set to zero, ensuring we retain all existing weekly archives. If we have monthly archives, we use the INTCK() function via %SYSFUNC to calculate the elapsed weeks (line 16). If the time between those two archives is less than 4 weeks (a "month" as currently defined by the project team), and we have the maximum number of weekly archives (plus 1), then the oldest weekly one should be deleted. If it's more than that, then it's time to promote the oldest weekly archive to a monthly archive (lines 19-36).

Now it's time to process the existing monthly archives, but we're still in the overall case of archiving weekly snapshots. The same process described above is repeated in lines 39-72 for existing monthly archives, with a couple of changes. There's no need to add 1 for an extra monthly archive (line 46), and test the number of four-week periods (beginning on Mondays) between the most recent quarterly archive and the oldest monthly archive (line 52).

Finally, any existing quarterly archives are processed (lines 75-93). Quarterly archives are much easier because they don't need to be promoted. Are there more quarterly archives than should be kept? If so, just delete the oldest one.

The same process is followed for semimonthly snapshots. Determine whether the oldest one should be deleted or aged to monthly, then work on existing monthly snapshots and end with the existing quarterly snapshots.

Monthly snapshots work the same way. Age or delete the oldest, and then check to see if there are too many quarterly archives.

%AGESNAPS SUMMARY

The beauty of this method is that it doesn't depend on a specific number of days; all that matters is whether enough time has elapsed since the last archive of a lesser frequency (e.g., weekly vs. monthly) to create another archive at the lesser frequency. In the case of weekly snapshots, a minimum of four weeks must have passed since the most recent monthly archive. It doesn't matter if five weeks have elapsed, or even ten weeks. A monthly archive will be created if enough time has passed since the last monthly archive. Yes, this can create unevenly dated archives. However, if you consider the case of data that doesn't change every week, you won't end up with a lot of identical archives just because it's working every four weeks. We also save system resources. Promotion works by changing an internal reference in a SAS data set, so we don't need to copy a folder full of files to a new location.

AGEARCHIVE

%AGEARCHIVE updates the archive control file when a given archive is promoted. It takes two parameters: an archive type and the date of the archive to be promoted. This is always the oldest date of the archives to be promoted. The only thing to note is the use of UPDATE instead of MERGE in line 12. While a merge would have worked just fine in this case as there are only two variables, technically, this is an update operation. The code is shown in its entirety in Figure 4:

Figure 4: AGEARCHIVE.SAS

```
1.   %MACRO ageArchive(type=,agedate=);

2.   /* create archive control record and update action log */

3.   DATA ageup;
4.   LENGTH date 4 archiveType $ 1;
5.   archivetype = STRIP(SYMGET('type'));
6.   date = SYMGET('agedate');
7.   RUN;

8.   PROC SORT DATA=dmodel.&archv_ctrl PRESORTED;
9.   BY date archivetype;
10.  RUN;

11.  DATA dmodel.&archv_ctrl;
12.  UPDATE dmodel.&archv_ctrl ageup;
13.  BY date;
14.  RUN;
15.  %MEND ageArchive;
```

RMVARCHIVE

This deletes an archive. On our system, you must remove any files within a folder before you can delete the folder itself. We can do this easily in SAS with PROC DATASETS and the FDELETE() function. The complete code is in Figure 5:

Figure 5: RMVARCHIVE code

```
1.  %MACRO rmvArchive(type=,rmvdate=,_debug=1);
2.  %LOCAL rc;
3.  %LET rmvdate_h = %SYSFUNC(PUTN(&rmvdate,e8601da.));

4.  /*remove files in folder */
5.  LIBNAME remov "data/&project/&rmvdate_h";
6.  PROC DATASETS LIB=remov KILL MT=DATA NOLIST NOWARN NODETAILS;
7.  RUN;
8.  QUIT;
9.  LIBNAME remov clear;

10. /* delete archive directory */
11. FILENAME remvdir "data/&project/&rmvdate_h";
12. %LET rc=%SYSFUNC(FDELETE(remvdir));
13. %put delete status=&rc; * For log;
14. FILENAME remvdir clear;

15. /**** log this archive's removal ***/
16. DATA daily_record;
17. LENGTH date 4 action $ 128 period $ 11;
18. success = INPUT(SYMGET('rc'),12.); * SYMGET returns character values;
19. type = STRIP(SYMGET('type'));
20. SELECT(type);
21.    WHEN('W') period='weekly';
22.    WHEN('S') period='semimonthly';
23.    WHEN('M') period='monthly';
24.    WHEN('Q') period='quarterly';
25.    OTHERWISE period="ERROR";
26. END;
27. date = &rundt;
28. IF success EQ 0 THEN
29.    action = CATX(' ',SYMGET('rmvdate_h'),period,"archive removed on",
30.                  PUT(date,weekdate32.));
31. ELSE
32.    action = CATX(' ',SYMGET('rmvdate_h'),period,
33.                  "archive removal failed on", PUT(date,weekdate32.));
34. DROP success type period;
35. RUN;

36. PROC APPEND BASE=dmodel.action_record DATA=daily_record;
37. RUN;
38.
39. /* Remove this archive's record from archive control dataset */
40. DATA dmodel.&archv_ctrl;
41. SET dmodel.&archv_ctrl;
42. IF date EQ &rmvdate THEN
43.    DELETE;
44. RUN;
45. %MEND rmvArchive;
```

Removal is based on the date parameter passed to the macro. It is passed as a SAS date value to find the correct records in the archive control and action record datasets. However, the ISO-formatted date is necessary because that is the directory name, which is used in lines 3, 5, and 11. PROC DATASETS removes the files (lines 5-9), with the NOLIST, NOWARN, NODETAILS options making it work silently. I added MT=DATA to limit the KILL operation to datasets out of habit.

Lines 10-14 delete the archive directory itself. The rest is housekeeping. The removal is logged in lines 15-37 by creating a record of the action based on whether the FDELETE() was successful and adding it to the activity log. Finally, the record for that archive is removed from the archive control dataset in lines 39-44.

FUTURE WORK AND ENHANCEMENTS

There is code clean-up to be done. For example, %GETMAXMINDATES doesn't have to be a macro. Since it places information into global macro variables that can be used by all the sub-macros, it probably only needs to be run once, and can be hardcoded into %DVARC. I'm sure there are other efficiencies to be had as well.

Currently, the archives are created based solely on timing. That means if the clinical data hasn't changed since the last scheduled snapshot, we will have an exact copy of older data in one or more archive folders. Although the original requirements didn't include the condition that the data must have changed, the team has since agreed that is the next logical revision of the archive code. We need to develop an efficient mechanism for detecting changes between the current data snapshot and the most recent archive. We don't need the detailed comparison information provided by PROC COMPARE; we just need to detect any change in any of the datasets. Some of the datasets are large and contain a lot of character data. PROC COMPARE may be too resource-intensive for this.

CONCLUSION

The capabilities of the SAS date and time facility allow for a great deal of creativity in solving timing problems. This is a practical demonstration of the power of SAS intervals. It has allowed for the creation of an archiving process without needing to refer to any absolute dates such as the date the process was initiated. It is self-contained, self-maintaining, and requires very little overhead with two small datasets that clean themselves to limit their size.

REFERENCES

Morgan, Derek P. 2014. *The Essential Guide to SAS® Dates and Times, Second Edition*. Cary, NC: SAS Institute Inc.

Chung, Chang Y, and John King. "Is This Macro Parameter Blank?" SAS Global Forum 2009 Proceedings, 2009, http://support.sas.com/resources/papers/proceedings09/022-2009.pdf, August 4, 2023.

ACKNOWLEDGMENTS

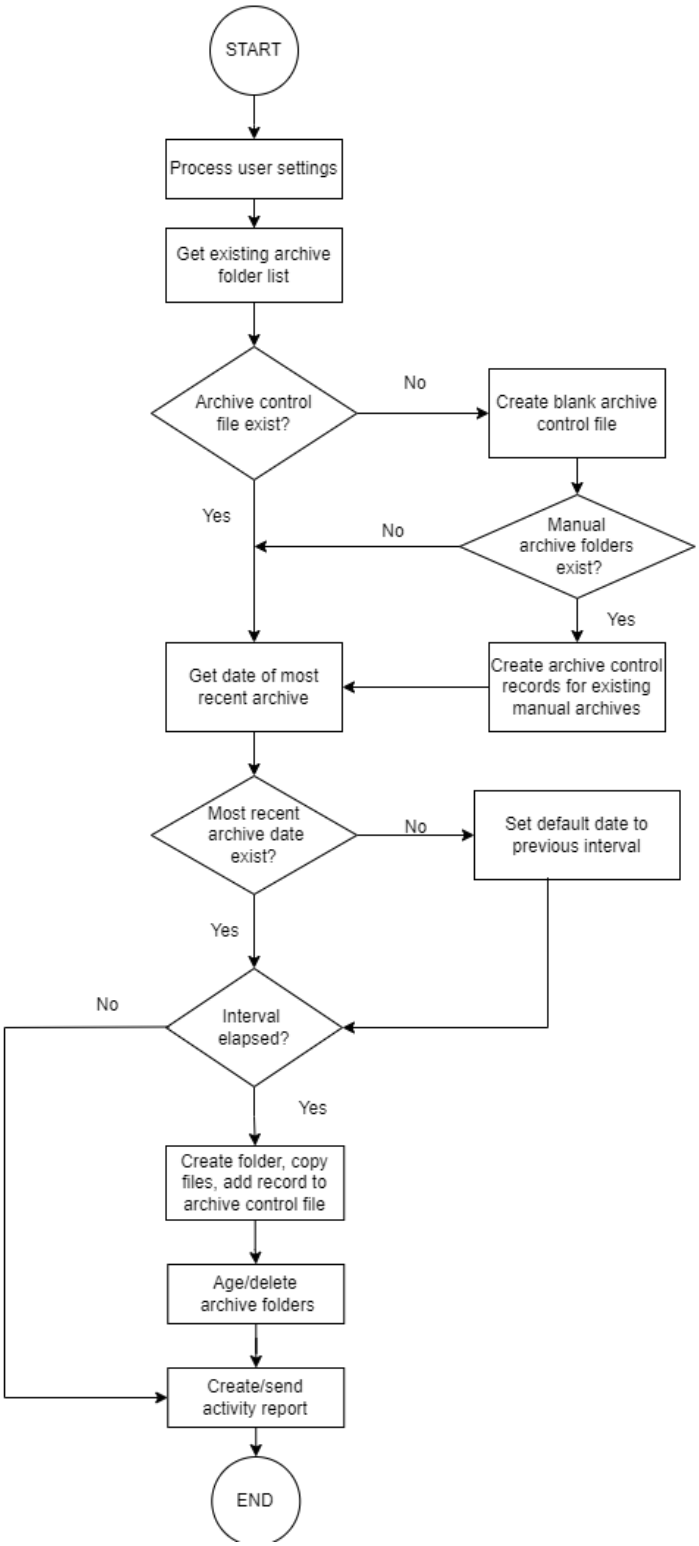
Many thanks to SAS Technical Support in general for their assistance over the past 35 (!) years.

CONTACT INFORMATION:

Comments and questions welcome to:

Derek Morgan
E-mail: mrdatesandtimes@gmail.com

APPENDIX A: PROCESS FLOW DIAGRAM



APPENDIX B: %OBSCOUNT UTILITY MACRO

```
1.   %macro obscount(dsn=last,count=nobs);
2.   OPTIONS NOMPRINT NOMLOGIC NOSYMBOLGEN NOMACROGEN;
3.   %GLOBAL &count;
4.   %LOCAL dsname dsid;
5.   %LET dsname=&dsn;
6.   %IF %SYSFUNC(EXIST(&dsname)) %THEN %DO;
7.      %LET dsid=%SYSFUNC(OPEN(&dsname, i));
8.      %LET &count=%SYSFUNC(ATTRN(&dsid,NOBS));
9.      %LET rc=%SYSFUNC(CLOSE(&dsid));
10.  %END;
11.  %ELSE
12.     %LET &count=-1;
13.  %MEND obscount;
```

APPENDIX C: ARCHIVING OVER THE COURSE OF ONE YEAR

Week #	Archive Date	Weekly Archive 1	Weekly Archive 2	Weekly Archive 3	Monthly archive 1	Monthly Archive 2	Monthly Archive 3	Quarterly Archive 1	Quarterly Archive 2	Quarterly Archive 3
1	1/16/2023									
2	1/23/2023	2023-01-16								
3	1/30/2023	2023-01-16	2023-01-23							
4	2/6/2023	2023-01-16	2023-01-23	2023-01-30						
5	2/13/2023	2023-01-23	2023-01-30	2023-02-06	2023-01-16					
6	2/20/2023	2023-01-30	2023-02-06	2023-02-13	2023-01-16					
7	2/27/2023	2023-02-06	2023-02-13	2023-02-20	2023-01-16					
8	3/6/2023	2023-02-13	2023-02-20	2023-02-27	2023-01-16					
9	3/13/2023	2023-02-20	2023-02-27	2023-03-06	2023-01-16	2023-02-13				
10	3/20/2023	2023-02-27	2023-03-06	2023-03-13	2023-01-16	2023-02-13				
11	3/27/2023	2023-03-06	2023-03-13	2023-03-20	2023-01-16	2023-02-13				
12	4/3/2023	2023-03-13	2023-03-20	2023-03-27	2023-01-16	2023-02-13				
13	4/10/2023	2023-03-20	2023-03-27	2023-04-03	2023-01-16	2023-02-13	2023-03-13			
14	4/17/2023	2023-03-27	2023-04-03	2023-04-10	2023-01-16	2023-02-13	2023-03-13			
15	4/24/2023	2023-04-03	2023-04-10	2023-04-17	2023-01-16	2023-02-13	2023-03-13			
16	5/1/2023	2023-04-10	2023-04-17	2023-04-24	2023-01-16	2023-02-13	2023-03-13			
17	5/8/2023	2023-04-17	2023-04-24	2023-05-01	2023-02-13	2023-03-13	2023-04-03	2023-01-16		
18	5/15/2023	2023-04-24	2023-05-01	2023-05-08	2023-02-13	2023-03-13	2023-04-03	2023-01-16		
19	5/22/2023	2023-05-01	2023-05-08	2023-05-15	2023-02-13	2023-03-13	2023-04-03	2023-01-16		
20	5/29/2023	2023-05-08	2023-05-15	2023-05-22	2023-02-13	2023-03-13	2023-04-03	2023-01-16		
21	6/5/2023	2023-05-15	2023-05-22	2023-05-29	2023-03-13	2023-04-03	2023-05-08	2023-01-16		
22	6/12/2023	2023-05-22	2023-05-29	2023-06-05	2023-03-13	2023-04-03	2023-05-08	2023-01-16		
23	6/19/2023	2023-05-29	2023-06-05	2023-06-12	2023-03-13	2023-04-03	2023-05-08	2023-01-16		
24	6/26/2023	2023-06-05	2023-06-12	2023-02-19	2023-03-13	2023-04-03	2023-05-08	2023-01-16		
25	7/3/2023	2023-06-12	2023-06-19	2023-06-26	2023-04-03	2023-05-08	2023-06-05	2023-01-16		
26	7/10/2023	2023-06-19	2023-06-26	2023-07-03	2023-04-03	2023-05-08	2023-06-05	2023-01-16		
27	7/17/2023	2023-06-26	2023-07-03	2023-07-10	2023-04-03	2023-05-08	2023-06-05	2023-01-16		
28	7/24/2023	2023-07-03	2023-07-10	2023-07-17	2023-04-03	2023-05-08	2023-06-05	2023-01-16		
29	7/31/2023	2023-07-10	2023-07-17	2023-07-24	2023-05-08	2023-06-05	2023-07-03	2023-01-16	2023-04-03	
30	8/7/2023	2023-07-17	2023-07-24	2023-07-31	2023-05-08	2023-06-05	2023-07-03	2023-01-16	2023-04-03	
31	8/14/2023	2023-07-24	2023-07-31	2023-08-07	2023-05-08	2023-06-05	2023-07-03	2023-01-16	2023-04-03	
32	8/21/2023	2023-07-31	2023-08-07	2023-08-14	2023-05-08	2023-06-05	2023-07-03	2023-01-16	2023-04-03	
33	8/28/2023	2023-08-07	2023-08-14	2023-08-21	2023-06-05	2023-07-03	2023-07-31	2023-01-16	2023-04-03	
34	9/4/2023	2023-08-14	2023-08-21	2023-08-28	2023-06-05	2023-07-03	2023-07-31	2023-01-16	2023-04-03	
35	9/11/2023	2023-08-21	2023-08-28	2023-09-04	2023-06-05	2023-07-03	2023-07-31	2023-01-16	2023-04-03	
36	9/18/2023	2023-08-28	2023-09-04	2023-09-11	2023-06-05	2023-07-03	2023-07-31	2023-01-16	2023-04-03	
37	9/25/2023	2023-09-04	2023-09-11	2023-09-18	2023-07-03	2023-07-31	2023-08-28	2023-01-16	2023-04-03	
38	10/2/2023	2023-09-11	2023-09-18	2023-09-25	2023-07-03	2023-07-31	2023-08-28	2023-01-16	2023-04-03	
39	10/9/2023	2023-09-18	2023-09-25	2023-10-02	2023-07-03	2023-07-31	2023-08-28	2023-01-16	2023-04-03	
40	10/16/2023	2023-09-25	2023-10-02	2023-10-09	2023-07-03	2023-07-31	2023-08-28	2023-01-16	2023-04-03	
41	10/23/2023	2023-10-02	2023-10-09	2023-10-16	2023-07-31	2023-08-28	2023-09-25	2023-01-16	2023-04-03	2023-07-03
42	10/30/2023	2023-10-09	2023-10-16	2023-10-23	2023-07-31	2023-08-28	2023-09-25	2023-01-16	2023-04-03	2023-07-03
43	11/6/2023	2023-10-16	2023-10-23	2023-10-30	2023-07-31	2023-08-28	2023-09-25	2023-01-16	2023-04-03	2023-07-03
44	11/13/2023	2023-10-23	2023-10-30	2023-11-06	2023-07-31	2023-08-28	2023-09-25	2023-01-16	2023-04-03	2023-07-03

Week #	Archive Date	Weekly Archive 1	Weekly Archive 2	Weekly Archive 3	Monthly archive 1	Monthly Archive 2	Monthly Archive 3	Quarterly Archive 1	Quarterly Archive 2	Quarterly Archive 3
45	11/20/2023	2023-10-30	2023-11-06	2023-11-13	2023-08-28	2023-09-25	2023-10-23	2023-01-16	2023-04-03	2023-07-03
46	11/27/2023	2023-11-06	2023-11-13	2023-11-20	2023-08-28	2023-09-25	2023-10-23	2023-01-16	2023-04-03	2023-07-03
47	12/4/2023	2023-11-13	2023-11-20	2023-11-27	2023-08-28	2023-09-25	2023-10-23	2023-01-16	2023-04-03	2023-07-03
48	12/11/2023	2023-11-20	2023-11-27	2023-12-04	2023-08-28	2023-09-25	2023-10-23	2023-01-16	2023-04-03	2023-07-03
49	12/18/2023	2023-11-27	2023-12-04	2023-12-11	2023-09-25	2023-10-23	2023-11-20	2023-01-16	2023-04-03	2023-07-03
50	12/25/2023	2023-12-04	2023-12-11	2023-12-18	2023-09-25	2023-10-23	2023-11-20	2023-01-16	2023-04-03	2023-07-03
51	1/1/2024	2023-12-11	2023-12-18	2023-12-25	2023-09-25	2023-10-23	2023-11-20	2023-01-16	2023-04-03	2023-07-03
52	1/8/2024	2023-12-18	2023-12-25	2023-01-01	2023-09-25	2023-10-23	2023-11-20	2023-01-16	2023-04-03	2023-07-03
53	1/15/2024	2023-12-25	2023-01-01	2023-01-08	2023-10-23	2023-11-20	2023-12-18	2023-04-03	2023-07-03	2023-09-25