PharmaSUG 2022 - Paper AP-107

Macro for Working with Zip Transport Files from the Web: The Case of the Medical Expenditure Panel Survey

Pradip K. Muhuri, Agency for Healthcare Research and Quality

ABSTRACT

Downloading zip SAS transport files (one at a time) from the web, unzipping the files outside of SAS®, and then restoring them to SAS data sets can be time-consuming and cumbersome, mainly when a handful of files are of interest. This paper presents a SAS macro that automates downloading any number of zip SAS files from the Medical Expenditure Panel Survey website, unzipping them, and converting unzipped transport files to SAS data sets.

INTRODUCTION

The Medical Expenditure Panel Survey-Household Component (MEPS-HC, hereafter referred to as MEPS) is a nationally representative survey of families and individuals on health care use and spending in the United States. Starting in 1996, MEPS continued to collect detailed information from sample households and their members, providing annual national estimates of health care use, medical care expenditures, access to care, sources of payment, and insurance coverage for the US civilian noninstitutionalized population [1].

Over the years, the Agency for Healthcare Research and Quality (AHRQ) has made various MEPS data files available for public use. The AHRQ's MEPS website presents an exhaustive list of public use files (PUFs) with hyperlinks for multiple data file types, including documentation and codebooks by survey year[1]. Typical data files include person-level files[2] and files for medical conditions, prescribed medicines, office-based medical provider visits, outpatient visits, emergency room visits, inpatient hospital stays, home health visits, dental visits, and appendix to MEPS events for survey years 1996-2020[3]. In addition, using the data file-specific Uniform Resource Locator (URL), you can download any files in multiple data formats that vary by survey year. For example, for survey years 1996-2017, zip files are primarily available in two data formats (ASCII and SAS transport files)[4]. However, for survey years 2018 and later, zip PUFs are mainly available in five data formats: ASCII file, SAS transport file, SAS data set, Stata file, and XLSX file.

Now imagine that you need to download a single SAS transport data file (e.g., 2009 MEPS Full-Year Consolidated File) from the MEPS website and extract the SAS data set from it. How would you go about these data file operations? Here is the answer. First, you click the text `ZIP` for *Data File, SAS transport*

---

[1] See https://meps.ahrq.gov/data_stats/download_data_files.jsp for for all MEPS data sets' "PUF Data Details" pages through unrestricted search and for each "Data File" page by year and file type. However, direct access to file directories from the website is restricted due to the server's security settings.

[2] These include the full-year population characteristics file, the full-year consolidated data file, and the longitudinal data file by survey year. Use the URL in footnote 1 for additional details about these data files.

[3] See the MEPS website for the data release schedule for subsequent annual files.

[4] See https://meps.ahrq.gov/data_stats/download_data_files_detail.jsp?cboPufNumber=HC-129. This is a long URL for the 2009 MEPS Full Year Consolidated Data File (as an example). There also exists a short URL embedded in each format-specific data file. For example, hovering over the clickable text ZIP for *Data File, SAS transport format* displays a unique short URL at the bottom left-hand corner of the website page (e.g., https://meps.ahrq.gov/data_files/pufs/h129ssp.zip for the 2009 Full-Year Consolidated File in SAS transport file format). Furthermore, note that, for the 2017 MEPS, data files are available in two data formats (ASCII and SAS transport files) with one exception: the Full-Year Consolidated PUF is available in five data formats (ASCII file, SAS transport file, SAS data set, Stata file, and XLSX file).

format on that URL page and wait until the file download is completed and SecureZIP for Windows menu items are available. Then click Extract Files from the SecureZIP application and follow the instructions to save the unzipped file in the folder of your choice. Finally, you use an appropriate SAS procedure to convert the unzipped SAS transport file to a SAS data set. However, this three-step process (*download-unzip-convert*) can be time-consuming and cumbersome when you desire to download a handful of files (e.g., ten files) after navigating through the respective URL. Instead, you can use a SAS macro to automate those repetitive tasks for various data files of your interest. This paper presents a SAS macro that automates downloading any number of zip SAS transport files from the MEPS website using the respective short URL, unzipping them, and converting unzipped transport files to SAS data sets.

Two types of zip SAS transport files are available for download from the MEPS website. They are either (1) XPORT engine (with PROC COPY)-created files[5] (hereafter referred to as Type I files) or (2) PROC CPORT-created files[6] (hereafter referred to as Type II files). Each zip file contains a single Type I or Type II SAS transport file.

SAS PROGRAM

In APPENDIX A, the SAS program uses a macro that loops through ten zip SAS transport files' names (eight Type I files and two Type II files), generating standard SAS statements. The macro definition contains the following steps[7].

- Step 1: Download zip files from the website with PROC HTTP
- Step 2: Unzip SAS transport files and write them to external files in a DATA _NULL_ step using
  - CALL SYMPUTX routine
  - FILENAME ZIP access method
- Step 3: Restore SAS data sets from
  - Type I SAS transport files using PROC COPY with XPORT engine
  - Type II SAS transport files using PROC CIMPORT

CREATING MACRO VARIABLES

As shown below, first, the program creates four macro variables using a %LET macro statement outside the macro definition to specify physical paths for four file types: downloaded zip files, unzipped Type I SAS transport files, unzipped Type II SAS transport files, and restored SAS data sets.

```
%let zip_folder = c:\data\zip_folder;
%let xpt_folder = c:\data\xpt_folder;
%let cpt_folder = c:\data\cpt_folder;
%let sds_folder = c:\data\sds_folder;
```

Then, the program creates three horizontal macro variable arrays, with multiple data files' names stored in a single macro variable, using the %LET statement.

```
%let xpt_files = h192 h181 h171 h163 h155 h147 h138 h129;
%let cpt_files = h201 h209;
%let total_files = &xpt_files &cpt_files;
```

---

[5] In general, SAS transport files for survey years 1996-2017 are Type I files. However, one exception is the 2017 Full-Year Consolidated File, which is a Type II SAS transport file (created using the CPORT procedure instead of the XPORT engine).

[6] Starting in 2018, instead of the XPORT engine, the CPORT procedure is used to create MEPS Public Use Files (PUFS) (excluding the 2018 Point-in-Time file, HC-36, and HC-036BRR).

[7] Besides, step 4 (TABLES dictionary table with PROC SQL) and step 5 (X statement and PROC DATASETS to delete some of the files) that are outside the macro definition are discussed subsequently.

Note the following.

- `xpt_files`[8] lists Type I SAS transport files' names.
- `cpt_files`[5] specifies Type II SAS transport files' names.
- `total_files` contains references to the above two macro variable arrays (i.e., a combined list of both types of SAS transport files' names assigned to this macro variable array).

Also, note that files' names or prefixes across data formats referenced in the SAS program are identical, with a period followed by a different extension shown below.

- zip SAS transport files' names (e.g., `h129ssp.zip`)
- unzipped SAS transport files' names (e.g., `h129.ssp`)
- restored SAS data sets' names (e.g., `h129.SAS7bdat`)

MACRO

The macro has the following general form (See APPENDIX A for details):

```
%macro load_meps / minoperator;
   %local i;
    %do i=1 %to  %sysfunc(countw(&total_files));
      /* The other macro statements and standard SAS statements
        are skipped here but shown subsequently. */
     %end;
    %mend load_meps;
%load_meps;
```

- The `MINOPERATOR` option on the `%MACRO` statement causes the macro processor to treat the special character (`#`) as an operator in the `%EVAL` macro function (Steps 2B, 3A, and 3B below) within the macro definition. `#` is an alias for the `IN` operator.

- `%sysfunc(countw(&total_files));` counts the total number of data files in the `&total_files` macro variable array and supplies a stop value for the iterative macro `%DO` loop.

- When the macro `%load_meps` is executed, the `%DO` loop cycles through ten iterations (ten zip SAS transport files). During macro execution, the macro processor processes the macro language elements and resolves the macro variable references for the data files of interest. Next, SAS compiles and executes the standard SAS code generated by the macro processor. Finally, the process leads to downloading zip SAS transport files from the MEPS website, unzipping them, converting unzipped transport files to SAS data sets, and storing all files in the relevant pre-created computer folders.

The paper presents below the partial SAS log from the *first* and *ninth* iterations[9] (as examples) to primarily guide the discussion on the text substitution performed by the macro processor.

Step 1: Use PROC HTTP with iterative macro %DO loop

In the SAS program (APPENDIX A), the iterative macro `%DO` loop includes the following code snippets for this step:

---

[8] Failing to correctly assign Type I and II transport files to the appropriate macro variable array will result in transport file-to-SAS data set conversion errors. However, one of the two macro variable arrays (`xpt_files` and `cpt_files`) could be null.

[9] Note that the data file in the *first* iteration is a Type I SAS transport file, while the file in the *ninth* iteration is a Type II SAS transport file.

```
❶filename inzip1 "&zip_folder.\%scan(&total_files, &i)ssp.zip";
   ❷proc http
url=https://meps.ahrq.gov/data_files/pufs/%scan(&total_files,&i)ssp.zip
     out=inzip1;
   run;
```

❶ `&zip_folder` resolves to `c:\data\zip_folder` for each iteration.

❶, ❷The `%scan` function searches the `&total_files` (i.e., list of data files' names delimited by a blank) for the ith item and returns it as a part of the file's name on the `FILENAME` statement and the URL path. When concatenated with regular text (i.e., `ssp`), the returned ith value produces a zip file's name (e.g., `h129ssp.zip` in the first iteration and `h201ssp.zip` in the ninth iteration). In addition, the `PROC HTTP` request returns `200 OK` for each iteration in the log[10], suggesting the download attempt succeeded.

Partial SAS log from the *first* iteration (Example of downloading a Type I zip SAS transport file, `h129ssp.zip`)

```
MLOGIC(LOAD_MEPS):  Beginning execution.
MLOGIC(LOAD_MEPS):  %LOCAL  I
SYMBOLGEN:  Macro variable TOTAL_FILES resolves to h129 h138 h147 h155 h163 h171
h181 h192 h201 h209
MLOGIC(LOAD_MEPS):  %DO loop beginning; index variable I; start value is 1; stop value
is 10; by value is 1.
SYMBOLGEN:  Macro variable ZIP_FOLDER resolves to c:\data\zip_folder
SYMBOLGEN:  Macro variable TOTAL_FILES resolves to h129 h138 h147 h155 h163 h171 h181
h192 h201 h209
SYMBOLGEN:  Macro variable I resolves to 1
MPRINT(LOAD_MEPS):   filename inzip1 "c:\data\zip_folder\h129ssp.zip";
SYMBOLGEN:  Macro variable TOTAL_FILES resolves to h129 h138 h147 h155 h163 h171 h181
h192 h201 h209
SYMBOLGEN:  Macro variable I resolves to 1
MPRINT(LOAD_MEPS):   proc http url="https://meps.ahrq.gov/data_files/pufs/h129ssp.zip"
out=inzip1;
MPRINT(LOAD_MEPS):   run;

NOTE: PROCEDURE HTTP used (Total process time):
      real time           1.48 seconds
      cpu time            0.40 seconds

NOTE: 200 OK
```

Partial SAS log from the *ninth* iteration (Example of downloading a Type II zip SAS transport file, `h201ssp.zip`)

```
MLOGIC(LOAD_MEPS):  %DO loop index variable I is now 9; loop will iterate again.
SYMBOLGEN:  Macro variable ZIP_FOLDER resolves to c:\data\zip_folder
SYMBOLGEN:  Macro variable TOTAL_FILES resolves to h129 h138 h147 h155 h163 h171 h181
h192 h201 h209
SYMBOLGEN:  Macro variable I resolves to 9
MPRINT(LOAD_MEPS):   filename inzip1 "c:\data\zip_folder\h201ssp.zip";
SYMBOLGEN:  Macro variable TOTAL_FILES resolves to h129 h138 h147 h155 h163 h171 h181
h192 h201 h209
```

[10] The SAS log is due to `OPTIONS SYMBOLGEN MPRINT MLOGIC`; used in the program. However, only the partial SAS log from the *first* and *ninth* iterations representing two types of SAS Transport files is shown here.

```
SYMBOLGEN:  Macro variable I resolves to 9
MPRINT(LOAD_MEPS):   proc http url="https://meps.ahrq.gov/data_files/pufs/h201ssp.zip"
out=inzip1;
MPRINT(LOAD_MEPS):   run;

NOTE: PROCEDURE HTTP used (Total process time):
      real time           1.08 seconds
      cpu time            0.12 seconds

NOTE: 200 OK
```

Step 2A: Retrieve the member file name from the zip file in DATA _NULL _ step using iterative macro %DO loop

In the SAS program (APPENDIX A), the iterative macro `%DO` loop includes the following code snippets for this step:

```
❷filename inzip2 zip "&zip_folder.\%scan(&total_files, &i)ssp.zip";

  ❸data _null_;
   length memname $200 ;
   fid=dopen("inzip2");
   memname=dread(fid,1);
   rc=dclose(fid);
   call symputx('memname',fname);
   run;
  %put &=memname;
```

❷The resolution of `"&zip_folder.\%scan(&total_files, &i)ssp.zip"` is the same as in❶. For example, `&zip_folder` resolves to `c:\data\zip_folder` for each iteration. `"%scan(&total_files, &i)ssp.zip"` resolves to `h129ssp.zip` in the *first* iteration and `h201ssp.zip` in the *ninth* iteration.

❸`DATA _NULL_` step uses

- `DOPEN` function to open the directory file[11] identified by the `fileref inzip2` and return a directory file identifier value 0 (if not opened) or 1 (if opened)
- `DREAD` function to read the name of the directory file into a `DATA` step variable `fname`
- `DCLOSE` to close the directory file
- `CALL SYMPUTX` routine to assign the `DATA` step variable `fname` to the macro variable `memname`

The `FILENAME ZIP` access method permits unzipping and writing it to an external file [2]. `inzip2` is the file reference (`fileref`) for the directory and zip transport file of interest in each iteration. Note that the `CALL SYMPUTX` routine creates a macro variable containing the SAS transport file's name from the zip file in each iteration. Also, note that zip files' names and file entries are case-sensitive.

Partial SAS log from the *first* iteration (Example of retrieving the member file name from the zip file)

```
SYMBOLGEN:  Macro variable I resolves to 1

MPRINT(LOAD_MEPS):   filename inzip2 zip "c:\data\zip_folder\h129ssp.zip";
MPRINT(LOAD_MEPS):   data _null_ ;
```

[11] As stated earlier, the zip file contains only one SAS transport file, not multiple files. Hence, the data step does not use the DNUM function here to return the number of transport files in the zip file (Hemedinger, 2015).

```
MPRINT(LOAD_MEPS):   length fname $200 ;
MPRINT(LOAD_MEPS):   fid=dopen("inzip2");
MPRINT(LOAD_MEPS):   if fid=0 then stop;
MPRINT(LOAD_MEPS):   fname=dread(fid,1);
MPRINT(LOAD_MEPS):   rc=dclose(fid);
MPRINT(LOAD_MEPS):   call symputx('memname',fname);
MPRINT(LOAD_MEPS):   run;

NOTE: DATA statement used (Total process time):
      real time           0.03 seconds
      cpu time            0.01 seconds

MLOGIC(LOAD_MEPS):  %PUT &=memname
SYMBOLGEN:  Macro variable MEMNAME resolves to h129.ssp
MEMNAME=h129.ssp
```

Partial SAS log from the *ninth* iteration (Example of retrieving the member file name from the zip file)

```
SYMBOLGEN:  Macro variable I resolves to 9
MPRINT(LOAD_MEPS):   filename inzip2 zip "c:\data\zip_folder\h201ssp.zip";
MPRINT(LOAD_MEPS):   data _null_ ;
MPRINT(LOAD_MEPS):   length fname $200 ;
MPRINT(LOAD_MEPS):   fid=dopen("inzip2");
MPRINT(LOAD_MEPS):   if fid=0 then stop;
MPRINT(LOAD_MEPS):   fname=dread(fid,1);
MPRINT(LOAD_MEPS):   rc=dclose(fid);
MPRINT(LOAD_MEPS):   call symputx('memname',fname);
MPRINT(LOAD_MEPS):   run;

NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds


MLOGIC(LOAD_MEPS):  %PUT &=memname
SYMBOLGEN:  Macro variable MEMNAME resolves to H201.ssp
MEMNAME=H201.ssp
```

Notice the difference in the 'letter case' of the values (`h129.ssp` vs. `H201.ssp`) of `&memname` in the *first* and *ninth* iterations. This difference is due to the `CALL SYMPUTX` routine preserving the case of the value of `&memname`. So, for example, the value's starting character (`h`) remains lowercase in the *first* iteration. In contrast, the corresponding value-part (`H`) remains uppercase in the *ninth* iteration.

Step 2B: Unzip the SAS transport file and write it to an external file in DATA _NULL_ step using conditional macro logic within iterative macro %DO loop

In the SAS program (APPENDIX A), the iterative macro `%DO` loop includes the following code snippets for this step:

```
❹%if &xpt_files ne %then %do;
❺%if %eval(%scan(&total_files, &i) # &xpt_files) %then  %do;
    ❻filename sit "&xpt_folder.\&memname";
      %end;
   %end;
❼%if &cpt_files ne %then %do;
❽%if %eval(%scan(&total_files, &i)  # &cpt_files) %then %do;
    ❾filename sit "&cpt_folder.\&memname" ;
```

```
      %end;
    %end;
❿data _null_;
   infile inzip2(&memname.) lrecl=256 recfm=f length=length eof=eof
   unbuf;
   file sit lrecl=256 recfm=n;
   input;
    put _infile_ $varying256. length;
      return;
      eof:
   stop;
   run;
```

❹,❺[12] Only if the two `%if-%then-%do-%end` code blocks return a non-zero integer (TRUE), this statement❻ executes `filename sit "&xpt_folder.\&memname";`, generating the directory and the file's name on the `FILENAME` statement. It does so for Type I SAS transport files.

❼, ❽[13] Only if the two `%if-%then-%do-%end` code blocks return a non-zero integer (TRUE), this statemen❾ executes `filename sit "&cpt_folder.\&memname";`, generating the directory and the file's name on the `FILENAME` statements. It does so for Type II SAS transport files.

Note that the above `DATA _NULL_` step uses the `INFILE` statement and the member syntax `inzip2(&memname)` and a null `INPUT` statement with no argument to read a member of the zip file [2] in each iteration.

Partial SAS log from the *first* iteration (Example of unzipping a Type I SAS transport file, `h129.ssp`)

```
SYMBOLGEN:  Macro variable XPT_FILES resolves to h129 h138 h147 h155 h163 h171 h181
h192
MLOGIC(LOAD_MEPS):  %IF condition &xpt_files ne is TRUE
SYMBOLGEN:  Macro variable I resolves to 1
SYMBOLGEN:  Macro variable CPT_FILES resolves to h201 h209
MLOGIC(LOAD_MEPS):  %IF condition %eval(%scan(&total_files, &i)  # &cpt_files) is
FALSE
MPRINT(LOAD_MEPS):   data _null_;
SYMBOLGEN:  Macro variable MEMNAME resolves to h129.ssp
MPRINT(LOAD_MEPS):   infile inzip2(h129.ssp) lrecl=256 recfm=f length=length eof=eof
unbuf;
MPRINT(LOAD_MEPS):   file sit lrecl=256 recfm=n;
MPRINT(LOAD_MEPS):   input;
MPRINT(LOAD_MEPS):   put _infile_ $varying256. length;
MPRINT(LOAD_MEPS):   return;
MPRINT(LOAD_MEPS):   eof: stop;
MPRINT(LOAD_MEPS):   run;
```

---

[12]The first `%if-%then-%do-%end`❹ resolves to 1 (TRUE) if `&xpt` is not null, or 0 (FALSE) if `&xpt` is null. The second `%if-%then-%do-%end`❺ using the `%EVAL` macro function resolves to 1 (TRUE) for a match or 0 (FALSE) for a nonmatch. Here the macro processor evaluates if the values of the two macro variables (`&total_files` and `&xpt_files` [Type I files]) match for the ith item of the `&total_files`.

[13]The conditional macro logic for ❼, ❽ is similar to that for ❹,❺ in the previous footnote. However, the macro variable of interest here is `&cpt_files` (Type II files) instead of `&xpt_files` (Type I files).

```
NOTE: The infile library INZIP2 is:
      Directory=c:\data\zip_folder\h129ssp.zip

NOTE: The infile INZIP2(h129.ssp) is:
      Filename=c:\data\zip_folder\h129ssp.zip,
      Member Name=h129.ssp

NOTE: UNBUFFERED is the default with RECFM=N.
NOTE: The file SIT is:
      Filename=c:\data\xpt_folder\h129.ssp,
      RECFM=N,LRECL=256,File Size (bytes)=0,
      Last Modified=01Apr2022:21:12:22,
      Create Time=01Apr2022:19:36:49

NOTE: A total of 1275712 records were read from the infile library INZIP2.
NOTE: 1275712 records were read from the infile INZIP2(h129.ssp).
NOTE: DATA statement used (Total process time):
      real time           1.03 seconds
      cpu time            1.03 seconds
```

Partial SAS log from the *ninth* iteration(Example of unzipping a Type II SAS transport file, H201.ssp)

```
SYMBOLGEN:  Macro variable XPT_FILES resolves to h129 h138 h147 h155 h163 h171 h181
h192
MLOGIC(LOAD_MEPS):  %IF condition &xpt_files ne is TRUE
SYMBOLGEN:  Macro variable TOTAL_FILES resolves to h129 h138 h147 h155 h163 h171 h181
h192 h201   h209
SYMBOLGEN:  Macro variable I resolves to 9
SYMBOLGEN:  Macro variable XPT_FILES resolves to h129 h138 h147 h155 h163 h171 h181
h192
MLOGIC(LOAD_MEPS):  %IF condition %eval(%scan(&total_files, &i) # &xpt_files) is FALSE
SYMBOLGEN:  Macro variable CPT_FILES resolves to h201 h209
MLOGIC(LOAD_MEPS):  %IF condition &cpt_files ne is TRUE
SYMBOLGEN:  Macro variable TOTAL_FILES resolves to h129 h138 h147 h155 h163 h171 h181
h192 h201 h209
SYMBOLGEN:  Macro variable I resolves to 9
SYMBOLGEN:  Macro variable CPT_FILES resolves to h201 h209
MLOGIC(LOAD_MEPS):  %IF condition %eval(%scan(&total_files, &i)  # &cpt_files) is TRUE
SYMBOLGEN:  Macro variable CPT_FOLDER resolves to c:\data\cpt_folder
SYMBOLGEN:  Macro variable MEMNAME resolves to H201.ssp
MPRINT(LOAD_MEPS):   filename sit "c:\data\cpt_folder\H201.ssp" ;
MPRINT(LOAD_MEPS):   data _null_;
SYMBOLGEN:  Macro variable MEMNAME resolves to H201.ssp
MPRINT(LOAD_MEPS):   infile inzip2(H201.ssp) lrecl=256 recfm=f length=length eof=eof
unbuf;
MPRINT(LOAD_MEPS):   file sit lrecl=256 recfm=n;
MPRINT(LOAD_MEPS):   input;
MPRINT(LOAD_MEPS):   put _infile_ $varying256. length;
MPRINT(LOAD_MEPS):   return;
MPRINT(LOAD_MEPS):   eof: stop;
MPRINT(LOAD_MEPS):   run;

NOTE: The infile library INZIP2 is:
      Directory=c:\data\zip_folder\h201ssp.zip

NOTE: The infile INZIP2(H201.ssp) is:
      Filename=c:\data\zip_folder\h201ssp.zip,
      Member Name=H201.ssp

NOTE: UNBUFFERED is the default with RECFM=N.
NOTE: The file SIT is:
      Filename=c:\data\cpt_folder\H201.ssp,
      RECFM=N,LRECL=256,File Size (bytes)=0,
```

```
      Last Modified=17Apr2022:20:18:33,
      Create Time=17Apr2022:20:18:33

NOTE: A total of 483613 records were read from the infile library INZIP2.
NOTE: 483613 records were read from the infile INZIP2(H201.ssp).
NOTE: DATA statement used (Total process time):
      real time           0.29 seconds
      cpu time            0.29 seconds
```

Step 3A: Use PROC COPY with XPORT engine to restore SAS data sets from SAS transport files of Type I with conditional macro logic within iterative macro %DO loop

In the SAS program (APPENDIX A), the iterative macro `%DO` loop includes the following code snippets for this step:

```
⓫%if &xpt_files ne %then %do;
    ⓬%if %eval(%scan(&total_files, &i)  # &xpt_files) %then %do;
       ⓭libname xpt xport "&xpt_folder.\%scan(&total_files, &i).ssp";
       ⓮libname sds "&sds_folder";
       ⓯proc copy in=xpt out=sds;
       run;
    %end;
 %end;
```

⓫, ⓬Only if the two `%if-%then-%do-%end` code blocks return a non-zero integer (TRUE)[14], the statements⓭ ⓮ execute. Thus, the macro code conditionally generates the physical name of the SAS data library and the Type I transport file's name on the first `LIBNAME` statement⓭, and the physical location of the SAS data library (for the SAS data set restored from the transport file) on the second `LIBNAME` statement⓮.

Thus the `COPY` procedure⓯ converts Type I SAS transport files to SAS data sets in eight of the ten iterations in the context of the SAS program in APPENDIX A.

Partial SAS log from the *first* iteration(Example of restoring a SAS data set, `SDS.H129`, from a Type I SAS transport file)

```
SYMBOLGEN:  Macro variable XPT_FILES resolves to h129 h138 h147 h155 h163 h171 h181
h192
MLOGIC(LOAD_MEPS):  %IF condition &xpt_files ne is TRUE
SYMBOLGEN:  Macro variable TOTAL_FILES resolves to h129 h138 h147 h155 h163 h171 h181
h192 h201 h209
SYMBOLGEN:  Macro variable I resolves to 1
SYMBOLGEN:  Macro variable XPT_FILES resolves to h129 h138 h147 h155 h163 h171 h181
h192
MLOGIC(LOAD_MEPS):  %IF condition %eval(%scan(&total_files, &i)  # &xpt_files) is TRUE
SYMBOLGEN:  Macro variable XPT_FOLDER resolves to c:\data\xpt_folder
SYMBOLGEN:  Macro variable TOTAL_FILES resolves to h129 h138 h147 h155 h163 h171 h181
h192 h201 h209
SYMBOLGEN:  Macro variable I resolves to 1
MPRINT(LOAD_MEPS):   libname xpt xport "c:\data\xpt_folder\h129.ssp";
NOTE: Libref XPT was successfully assigned as follows:
```

[14] See footnote 12 for an explanation of this conditional macro logic, which involves Type I files.

```
      Engine:        XPORT
      Physical Name: c:\data\xpt_folder\h129.ssp
SYMBOLGEN:  Macro variable SDS_FOLDER resolves to c:\data\sds_folder
MPRINT(LOAD_MEPS):   libname sds "c:\data\sds_folder";
NOTE: Libref SDS was successfully assigned as follows:
      Engine:        V9
      Physical Name: c:\data\sds_folder
MPRINT(LOAD_MEPS):   proc copy in=xpt out=sds;
MPRINT(LOAD_MEPS):   run;
NOTE: Input library XPT is sequential.
NOTE: Copying XPT.H129 to SDS.H129 (memtype=DATA).
NOTE: BUFSIZE is not cloned when copying across different engines.  System Option for
BUFSIZE was used.
NOTE: There were 36855 observations read from the data set XPT.H129.
NOTE: The data set SDS.H129 has 36855 observations and 1908 variables.
NOTE: PROCEDURE COPY used (Total process time):
      real time           4.35 seconds
      cpu time            2.53 seconds
```

Step 3B: Use PROC CIMPORT to convert SAS transport files of Type II to SAS data sets with conditional macro logic within iterative macro %DO loop

In the SAS program (APPENDIX A), the iterative macro `%DO` loop includes the following code snippets for this step:

```
❻%if &cpt_files ne %then %do;
   ❼%if %eval(%scan(&total_files, &i) #  &cpt_files) %then %do;
      ❽filename cpt "&cpt_folder.\%scan(&total_files, &i).ssp";
      ❾libname sds base "&sds_folder";
      ❿proc cimport infile=cpt library=sds;
      run;
   %end;
%end;
```

❻, ❼Only if the two `%if-%then-%do-%end` code blocks return a non-zero integer (TRUE)[15], the statements❽ ❾ execute. Thus, the macro code conditionally generates the directory name and the Type II transport file's name❽ on the `FILENAME` statement and the physical location of the SAS data library (for the SAS data set restored from the transport file) on the `LIBNAME` statement❾.

Thus the `CIMPORT` procedure❿ converts SAS transport files (Type II) to SAS data sets in two of the ten iterations in the context of the SAS program in APPENDIX A.

Partial SAS log from the *ninth* iteration (Example of restoring a SAS data set, `SDS.H201`, from a Type II SAS transport file)

```
SYMBOLGEN:  Macro variable XPT_FILES resolves to h129 h138 h147 h155 h163 h171 h181
h192
MLOGIC(LOAD_MEPS):  %IF condition &xpt_files ne is TRUE
SYMBOLGEN:  Macro variable TOTAL_FILES resolves to h129 h138 h147 h155 h163 h171 h181
h192 h201 h209
SYMBOLGEN:  Macro variable I resolves to 9
```

[15]See footnote 13 for an explanation of this conditional macro logic, which involves Type II files.

```
SYMBOLGEN:  Macro variable XPT_FILES resolves to h129 h138 h147 h155 h163 h171 h181
h192
MLOGIC(LOAD_MEPS):  %IF condition %eval(%scan(&total_files, &i)  # &xpt_files) is
FALSE
SYMBOLGEN:  Macro variable CPT_FILES resolves to h201 h209
MLOGIC(LOAD_MEPS):  %IF condition &cpt_files ne is TRUE
SYMBOLGEN:  Macro variable TOTAL_FILES resolves to h129 h138 h147 h155 h163 h171 h181
h192 h201 h209
SYMBOLGEN:  Macro variable I resolves to 9
SYMBOLGEN:  Macro variable CPT_FILES resolves to h201 h209
MLOGIC(LOAD_MEPS):  %IF condition %eval(%scan(&total_files, &i) #  &cpt_files) is TRUE
SYMBOLGEN:  Macro variable CPT_FOLDER resolves to c:\data\cpt_folder
SYMBOLGEN:  Macro variable TOTAL_FILES resolves to h129 h138 h147 h155 h163 h171 h181
h192 h201 h209
SYMBOLGEN:  Macro variable I resolves to 9
MPRINT(LOAD_MEPS):   filename cpt "c:\data\cpt_folder\h201.ssp";
SYMBOLGEN:  Macro variable SDS_FOLDER resolves to c:\data\sds_folder
MPRINT(LOAD_MEPS):   libname sds base "c:\data\sds_folder";
NOTE: Libref SDS was successfully assigned as follows:
      Engine:        BASE
      Physical Name: c:\data\sds_folder
MPRINT(LOAD_MEPS):   proc cimport infile=cpt library=sds;
MPRINT(LOAD_MEPS):   run;
NOTE: PROC CIMPORT begins to create/update data set SDS.H201
NOTE: Data set contains 1564 variables and 31880 observations.
      Logical record length is 7704
```

Step 4: Use TABLES dictionary table with PROC SQL

I use the `TABLES` dictionary table with the `SQL` procedure's `SELECT` statement `FROM` and `WHERE` clauses to obtain restored SAS data sets' names, number of observations, and number of variables (See step 4 in the SAS program in APPENDIX A).

Member Name	Number of Physical Observations	Number of Variables
H129	36,855	1,908
H138	32,846	1,911
H147	35,313	2,052
H155	38,974	1,883
H163	36,940	1,790
H171	34,875	1,838
H181	35,427	1,831
H192	34,655	1,941
H201	31,880	1,564
H209	30,461	1,501

Output 1. Output from PROC SQL

Step 5 (Optional): Use X statement and PROC DATASETS to delete data files

After confirming the restoration of SAS data sets from the transport files of interest, I use the `X` statement to delete zip files from the relevant folder. I also use `PROC DATASETS` with `NOLIST` and `KILL` options to delete SAS transport files from the pertinent folders, suppressing the printing of the directory in the SAS log (See step 5 in the SAS program in APPENDIX A). However, I leave SAS data sets restored from zip transport files intact in the designated SAS data library.

CONCLUSION

The present macro is appropriate for working with zip SAS transport files from the MEPS website. Implementing the macro requires prior knowledge of the SAS procedure initially used to create the transport file. However, in general, the macro is portable with minimal programmer intervention.

REFERENCES

1. Cohen, J., S. Cohen, and J. Banthin. 2009. "The Medical Expenditure Panel Survey: A National Information Resource to Support Health Care Cost Research and Inform Policy and Practice." *Medical Care*, 47:S44-50.
2. Hemedinger, C. 2015. "Using FILENAME ZIP to unzip and read data files in SAS." SAS Blogs. https://blogs.sas.com/content/sasdummy/2015/05/11/using-filename-zip-to-unzip-and-read-data-files-in-sas/.

ACKNOWLEDGMENTS

The views expressed in this paper are those of the author, and no official endorsement by the Health and Human Services or the Agency for Healthcare Research and Quality is intended or should be inferred. However, the author would like to thank Gary Moore (Academic Chair, PharmaSUG 2022), Frank Canale, and Scott Burroughs (Section Chairs) for their support and review of the paper before publishing it in the PharmaSUG 2022 Conference Proceedings; Marc Zodet and Frederick Rohde for reviewing it earlier as part of the AHRQ's clearance process; SAS Technical Support for answering questions on the code optimization; and Emily Mitchell for testing the macro independently. Furthermore, the author is grateful to Scott Burroughs for his incredible support in formatting the manuscript. In addition, the author would like to express his sincere gratitude to Jayanth Iyengar for reviewing the pre-final version of the paper and providing valuable tips to fine-tune the code explanation. Finally, the author is responsible for any errors or omissions.

CONTACT INFORMATION

Your comments are valued and encouraged. Contact the author at:

Pradip K. Muhuri, PhD
AHRQ/CFACT
5600 Fishers Lane
Rockville, MD 20857
Pradip.Muhuri@ahrq.hhs.gov

APPENDIX A (COMPLETE SAS PROGRAM)

```
/*************************************************************
 The program automates downloading a handful of two types of zipped
 SAS transport files from the Medical Expenditure Panel Survey (MEPS)
 Website, unzipping them, and converting unzipped transport files
 to SAS data sets in Windowing mode.
*************************************************************/
options symbolgen mprint mlogic;
/* Create four global macro variables, assigning a folder name */
%let zip_folder = c:\data\zip_folder;
%let xpt_folder = c:\data\xpt_folder;
%let cpt_folder = c:\data\cpt_folder;
%let sds_folder = c:\data\sds_folder;
```

```
/*************************************************************************
Create three global pseudo-macro variable arrays
     - PROC XCOPY or XPORT engine (with PROC COPY)-created
        SAS transport file names (Type I files)
     - PROC CPORT-created SAS transport file names (Type II files)
       (Either of the two macro variable arrays could have a null value)
     - combined list of both types of SAS transport files.
*************************************************************************/
%let xpt_files = h129 h138 h147 h155 h163 h171 h181 h192 ;
%let cpt_files = h201 h209;
%let total_files = &xpt_files &cpt_files;

%macro load_meps / minoperator; /* begins the macro definition */
%local i;
%do i=1 %to %sysfunc(countw(&total_files));

/*****************************************************************
  Step 1: Download zipped SAS transport files to a pre-created folder.
 *****************************************************************/
filename inzip1 "&zip_folder.\%scan(&total_files, &i)ssp.zip";
proc http
url="https://meps.ahrq.gov/data_files/pufs/%scan(&total_files, &i)ssp.zip"
       out=inzip1;
run;

/*************************************************************************
Step 2A: Retrieve the member file's name from the zip file.
       Code adapted from Hemedinger (2015).
*************************************************************************/
filename inzip2 zip "&zip_folder.\%scan(&total_files, &i)ssp.zip";
      data _null_ ;
       length fname $200 ;
       fid=dopen("inzip2");
       if fid=0 then stop;
       fname=dread(fid,1);
       rc=dclose(fid);
       call symputx('memname',fname);
       run;
    %put &=memname;

/*************************************************************************
Step 2B: Unzip SAS transport files using the filename zip access method and
save them in desired folders.  Code adapted from Hemedinger (2015).
*************************************************************************/
      %if &xpt_files ne %then %do;
        %if %eval(%scan(&total_files, &i) # &xpt_files) %then %do;
           filename sit "&xpt_folder.\&memname" ;
         %end;
      %end;
      %if &cpt_files ne %then %do;
          %if %eval(%scan(&total_files, &i)  # &cpt_files) %then %do;
              filename sit "&cpt_folder.\&memname" ;
          %end;
      %end;

       /* hat tip: "data _null_" on sas-l */
       data _null_;
infile inzip2(&memname.) lrecl=256 recfm=f length=length eof=eof unbuf;
      file sit lrecl=256 recfm=n;
          input;
          put _infile_ $varying256. length;
          return;
        eof:
```

```
         stop;
       run;

/*****************************************************************
Step 3A: Restore SAS data sets from Type I SAS transport files.
*****************************************************************/
%if &xpt_files ne %then %do;
  %if %eval(%scan(&total_files, &i)  # &xpt_files) %then %do;
        libname xpt xport "&xpt_folder.\%scan(&total_files, &i).ssp";
        libname sds "&sds_folder";
              proc copy in=xpt out=sds;
              run;
   %end;
  %end;

/*****************************************************************
Step 3B: Restore SAS data sets from Type II SAS transport files.
*****************************************************************/
 %if &cpt_files ne %then %do;
   %if %eval(%scan(&total_files, &i) #  &cpt_files) %then %do;
     filename cpt "&cpt_folder.\%scan(&total_files, &i).ssp";
     libname sds base "&sds_folder";
        proc cimport infile=cpt library=sds;
        run;
   %end;
  %end;
 %end;
%mend load_meps; /* ends the macro definition */
%load_meps    /* macro call */

/*****************************************************************
Step 4: Use TABLES dictionary table to obtain information about restored SAS
data sets.
*****************************************************************/
proc sql;
 select memname, nobs format =comma9. ,nvar format=comma9.
 from dictionary.tables
 where libname='SDS' and memname like 'H%';
quit;

/*****************************************************************
Step 5 (Optional): The following X statement deletes the zip files and the
subsequent PROC DATASETS code block deletes the SAS transport files from the
respective folders programmatically, leaving the SAS data sets (restored from
SAS transport files) intact in the pertinent folder.
*****************************************************************/
options xsync noxwait;
x "del /q c:\data\zip_folder\*.zip";
libname myrefs ("&xpt_folder" "&cpt_folder");
proc datasets library= myrefs nolist kill;
run;
quit;
```