PharmaSUG 2023 - Paper SD-221

SAS® System Macros to Summarize the COMPARE Procedure Results and SAS Logs for a Directory or Single File

Kevin R. Viel, Ph.D., Navitas Data Sciences; Histonis, Incorporated

ABSTRACT

A moderate project in clinical trial programming might have 30-50 ADaM data sets and 200-500 tables, listing, and figures. The convention is 100% independent programming with a comparison of the data sets. When the SAS® System is used, this may be referred to as DP-PC: Double-Programming, Proc COMPARE. Further, a check of the SAS logs for certain words or phrases indicative of unacceptable NOTES, ERROR messages, or WARNING is also appropriate. At times, a lead programmer or biostatistician may want to verify progress or confirm the attestations that Validation is complete yet reviewing so many SAS .lst and .log files manually is cumbersome and a sample may not suffice. The **goal** of this paper is to introduce five SAS macros and ancillary macros that they require, that submit programs with batch submission, read the contents of a list of directories and optionally provide file metadata, and summarize the results of COMPARE procedures and log checks of all files in directories or single files. Such activities are essential to a readiness audit for delivery or submission and for routine programming.

INTRODUCTION

Validation is an essential component of clinical trial programming. Typically, Validation includes independent (blinded) programming in which specifications and shells are shared, but the code and logs of the Main programmer cannot be viewed by the peer Validation programmer and vice versa. A central task of Validation, besides assuring the *accuracy* of the Main Programs, includes a verification that the Main programs are of acceptable fidelity as determined by "clean" (SAS® System) logs, that is, the lack of any notes, warnings, or errors that the Standard Operating Procedures (SOPs), Working Instructions (WIs), or the programmers deem unacceptable since these Main programs are part of a submission or delivery and can be subject to audits; they are one of the "faces" of the programmer teams, a point of *team* pride. While the Main and Validation programmers attest to the successful completion of tasks according to the SOPs/WIs, other audit duties to confirm this must be performed by such people as the lead programmer, the biostatistician, the project managers, or the executives. In fact, some audit tasks, such as confirming that the datetime stamps of the data sets and files are appropriate occurs at the program level at every production execution.

The language can be confusing. A **run**, **submission**, or an **execution** and their respective verbs are synonyms that refer to compiling and executing code. The term **submission** also refers to body of files composing an application or filing to a regulatory agency (RA, like the FDA) or the act of transferring these files to an RA. A delivery is an instance of a submission of an application or a filing or transferring the package of files (results) outside of the team (group, for instance, transferring Validated and reviewed files to a vice-president or DSMC). This paper refers to program **development** as the process of drafting or updating code, to **QC** (Quality Control) as the process of (blinded) peer review and/or programming with Validation programs or code review as the SOPs/WIs require, and to **production runs** (execution) as the formal execution of code in the appropriate computing environment following the applicable SOPs/WIs. Production runs can be the final step prior to promotion or to produce the final output (see details later in this paragraph), i.e., production runs can occur in the development environment, if one exists. "Formal" or "official" might be better adjectives for this concept, avoiding confusion with the stages below. The **Main programmer**, sometimes (confusingly) called the Production programmer, has responsibility for creating the output, i.e., the data sets (SDTM or ADaM, for instance) or the PDF and/or RTF files for **TFL**s (Tables, Figures, and Listings), that will be part of the **CSR** (Clinical Study Report) and submission. The **Validation programmer** is the *peer* programmer who replicates the task typically with 100% independence, but usually does not produce TFL output, and assures that both sets or programs

and output meet the requirements established in the SOPs/WIs. The Main programmer often works in a folder called “PROD” or “SAS”, whereas the Validation programmer often works in a folder called “VAL”. Sometimes, a hierarchy of folders that are nearly replications of each other distinguish between the stages of development, QC, and final runs and the folders are often named “DEV”, “QC”, and “PROD”, respectively. The Main and Validation programmers both develop in their respective DEV folders, promote acceptable programs to QC for evaluation, and promote programs that pass Validation in QC to PROD for a final run and where other members, like a biostats, may review them. Being in PROD may mean that the product is ready for higher review, but not necessarily delivery; indeed, although individual programs pass Validation, the entire set may not (for instance, a discrepancy in a column total across ten tables may match in Validation, but is not acceptable). Good hygiene might dictate that programmers delete files from lower hierarchical folders upon satisfying requirements for the next level, i.e., a clean run with an acceptable log check and an acceptable match of data sets, which is typically a 100% exact match of values, if not labels, lengths, and attributes. Such a structure and process may not be a requirement, but lack of something similar requires good hygiene and, usually, more details (macro parameter values or post-processing) to use these macros.

At every stage of programming, from development to production runs, the programmers should be checking the datetime stamp of the data sets, files (.sas, .log, .lst) and, if any, raw data files. For instance, the datetime stamp of ADSL should be older than that of any other ADaM data set, but younger than DM (and other required SDTM domains). The datetime stamps of the log and output files should be younger than the (.sas) program that created them. The datetime stamp of the log of the Validation program should be younger than the datetime stamps of the Main log and output. One way to insure this is to have the Validation programmer execute the Main program before executing the Validation program, *if* the SOPs/WIs allow this and since the Validation programmer should not be opening the Main program (and vice versa), batch submission is a salient and easy solution that also provides an audit trail. A contention might be that ADSL was sent back to development and re-Validated, but ADAE in PROD had not, yet, been re-run and so would any TFL in PROD that used ADAE (and ADSL). Depending on the SOPs/WIs, after completion of programming, i.e., obtaining the attestations of the Main and Validation programmers and passing review by their superiors for each program in the delivery, batch submission of the entire collection of programs, in appropriate hierarchical order, may be required. To “pre-program” a project, then execute every program and deliver within three business days of a database lock is not quite an unusual requirement. After each production execution, every programmer should be checking the log of her or his program prior; no promoted program should generate a log that contains unacceptable phrases that are not documented in the program header (a failure to converge error may be unavoidable, for instance, but the header should alert the programmer and review a priori). If one is a Validator, one should verify that the log of the Main program is acceptable. While this is necessary for each program, the persons responsible for delivery need to confirm that this is true across the *entire delivery*. While this can be accomplished by spot checking a sample of the programs, programmatically approaching this task is reasonable. Moreover, programmatic approaches may be more accurate and complete, highly efficient, and easily and dependably repeatable.

The **goal** of this paper is to present SAS System macros **1)** to batch submit files using SAS, not the usual .bat approach or another programming languages such as Python, **2)** to recursively read the contents of a directory, similar to *dir* command in DOS or the *ls* command in Unix/Linux, including, importantly, the reporting of the Last Modified datetime stamps, **3)** to summarize a log file from one program with respect to the presence of certain phrases indicating potential or certain unacceptable issues, **4)** to summarize log check findings of the logs files in directories, and **5)** to summarize the results from COMPARE procedures in single files or every file in a directory. An ancillary macro to delete a list of SAS data sets is included, but like the other macros, can be substituted or modified as the reader sees fit.

TEST CODE AND FILES

Appendix 1 presents a macro for testing and demonstration purposes only. For brevity and clarity, this macro may not follow best practices in a regulated programming environment. For instance, the Validation program needs access to SDTM, but that would be readonly access. The Main programmer should not have to access V_SDTM. In fact, these LIBREFs would not contain the Validation Data Sets (VALDS’s) for TFL programs, but SDTM domains. Lines 11-41 “clean up” the directories, they delete

SAS data sets and other files. The %DO loop in lines 50-163 creates SAS, LOG, LST, and RTF files to demonstrate the macros in this paper. Again, to emphasize they are *contrived* for demonstration purposes. The sections below will discuss their respective macros and calls of them, if necessary.

For simplicity (and some safety), consider the following code to create folders in the WORK directory that should be deleted at the end of an interactive SAS session:

```
options dlcreatedir ;

libname sdtm
        "%sysfunc( pathname( work ))/sdtm"
        ;

libname v_sdtm
        "%sysfunc( pathname( work ))/v_sdtm"
        ;
```

MAC_U_BATCH_SUBMIT

To introduce this suite of utility and reporting macros with the log check macro might seem reasonable, since one should generally always check one's log before any other action, this is an appropriate place to start. To be assured that no "artifact", such as a LIBREF, FILEREF, or global macro variable had the wrong value, each production run of a program should be executed in a fresh SAS session. The log will reflect whether that occurred, but a few lines could slip by even a very astute eye. Batch submission of each program assures a fresh SAS session. Note that developing in PC SAS, i.e., interactive SAS, does not preclude, for example, submitting the program of interest for batch submission within that same session. **Appendix 2** presents MAC_U_BATCH_SUBMIT, a SAS macro to batch submit either a single program or a list of programs contained in a data set.

The author adopted the convention of starting the name every macro with "MAC_" so that it is easier to find macro in code, especially when reading every SAS program in a drive or directory. A manager or lead can perform such a task to assist with decisions like when should a macro be considered for promotion from study-level to compound-level or from compound-level to enterprise-level or what would be the impact of updating or retiring a macro, i.e., how widespread is the use of the macro and when was it last used? The second component indicates classification: U-utility, R-reporting, DEBUG-debugging, PGM-defined in the current program, et cetera. The third component is a brief description (what, perhaps, was the name of the macro before adoption of this convention). The author adopts the convention that a debugging macro, for instance, should never appear in a production run, and "mac_debug_" readily identifies such macros. Finally, a warning: when obtaining the files for batch submission, one needs to be sure *not* to include the batch submission program or, potentially a call to MAC_U_BATCH_SUBMIT (or something similar). Otherwise, one may generate an endless loop: a batch submission program batch submitting itself.

While developing in an interactive session, one can submit snippets of code or the full program, but the final test should be a batch submission to ensure a fresh SAS session. The following "one-liner" in the same interactive session suffices (note an interactive SAS session can have multiple editors):

```
%mac_u_batch_submit
   ( pathfile = C:\Users\vielk\Documents\My SAS Files\9.4\AD221\ad221_batch.sas ) ;
```

Lines 27-32 of Appendix 1 shows one way to create a data set of paths and files names of files that can be used as the input data set for the macro parameter DS (line 2 of Appendix 2). Using Windows Explorer, another way to create a list of path and file names is to select the files of interest and Shift-right click and select "Copy as path" (**Figure 1**).

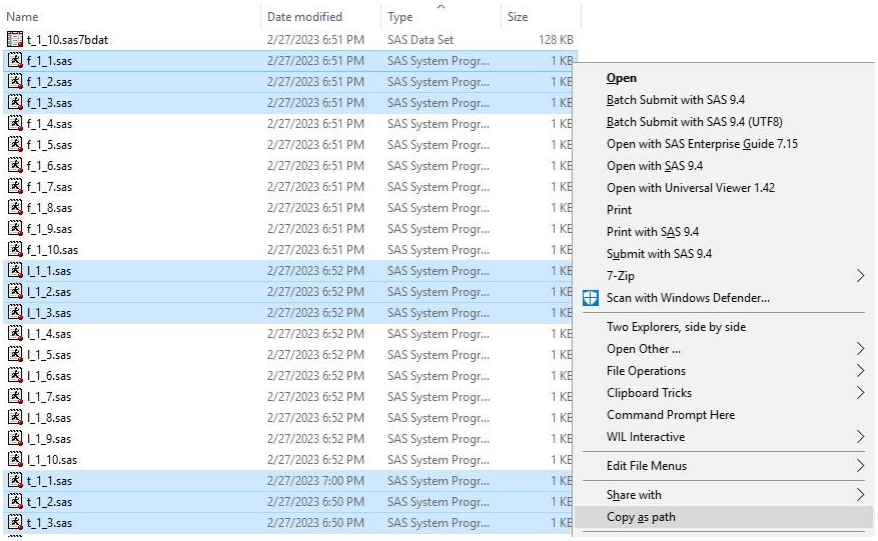

Figure 1. Selecting files in Windows Explorer to "Copy as path".

One can then paste the results into the DATALINES of a data step for instance:

```
"C:\Users\vielk\Documents\My SAS Files\9.4\sdtm\t_1_3.sas"
"C:\Users\vielk\Documents\My SAS Files\9.4\sdtm\f_1_1.sas"
"C:\Users\vielk\Documents\My SAS Files\9.4\sdtm\f_1_2.sas"
"C:\Users\vielk\Documents\My SAS Files\9.4\sdtm\f_1_3.sas"
"C:\Users\vielk\Documents\My SAS Files\9.4\sdtm\l_1_1.sas"
"C:\Users\vielk\Documents\My SAS Files\9.4\sdtm\l_1_2.sas"
"C:\Users\vielk\Documents\My SAS Files\9.4\sdtm\l_1_3.sas"
"C:\Users\vielk\Documents\My SAS Files\9.4\sdtm\t_1_1.sas"
"C:\Users\vielk\Documents\My SAS Files\9.4\sdtm\t_1_2.sas"
```

Note the order or, rather, lack thereof in the pasted files. If this is the technique used to create a batch submit program, then being aware of the order and changing it is essential. For instance, ADSL.sas should be the first program run, followed by V_ADSL.sas. The same ordering issue is true for the directory read approach to create the list of files.

The macro ultimately generates the command line that is submitted to the operating system via the X statement:

```
1   + x '"C:\Program Files\SAS94\SASFoundation\9.4\sas.exe" -CONFIG "C:\Program
Files\SAS94\SASFoundation\9.4\nls\u8\sasv9.cfg" -NOSPLASH -NOSTATUSWIN -SASUSER !userprofile -SYSIN
"C:\Users\vielk\Documents\My SAS Files\9.4\PharmaSUG\2023\AD221\ad221_batch.sas"
2   + -NOLOG -ALTLOG "C:\Users\vielk\Documents\My SAS Files\9.4\PharmaSUG\2023\AD221\ad221_batch.log" -PRINT
"C:\Users\vielk\Documents\My SAS
Files\9.4\PharmaSUG\2023\AD221\ad221_batch.lst" ' ;
```

The macro performs some "housecleaning", like checking if the files and paths exists and generating the LOG and LST paths and files names by substitution of the PATHFILE of the program, the patterns of which can be replaced with a macro parameter with slight programming, if desired. Note the use of -NOLOG and -ALTLOG. The purpose of this is to provide a "shadow" log, assuming that the log will be redirected in the program or macro that initializes the programming environment, including generating the paths and file names of the log and lst and redirecting the log via the PRINTTO procedure. The impetus of this was to be able to close the log, releasing the lock, to perform a log check the results of which would be sent to the lst file, all within the current program execution, i.e., no post-processing required. The macro also provides the opportunity to display the paths as a check before the run to be sure that they are correct. While not, yet, common in clinical trial programming, some analysis programs, such as Bayesian or Mixed models can take hours or longer to run. When genomics is required, data sets that can make LB/ADLB looks small, we might see increased CPU times. Currently, the submissions are in sequence, not parallel, but that can be easily updated for every program at the same hierarchy, i.e., each

program in a given level, those that do not depend on other programs at that level, can be sent to separate nodes or for asynchronous execution via the SAS System Option “NOXSYNC”.

MAC_U_DIRECTORY_READ

Appendix 3 presents the macro MAC_U_DIRECTORY_READ. This macro reads the directories and files in a path (directory), optionally recursively, and optionally, obtains file information using the SAS FINFO() function (mileage may vary, but bytes and last modified datetimes seem consistently reported in Windows, in the experience of the author). This can be thought of as using the DOS *DIR* command or the Unix *ls* command. The data set in **Figure 2** is generated by the following macro call:

```
%mac_u_directory_read
  ( path  =  C:\Users\vielk\Documents\My SAS Files\9.4\PharmaSUG )
```

	pathfile	_level	_dopen
1	C:\Users\vielk\Documents\My SAS Files\9.4\PharmaSUG\2023	1	2
2	C:\Users\vielk\Documents\My SAS Files\9.4\PharmaSUG\2023\AD221	2	2
3	C:\Users\vielk\Documents\My SAS Files\9.4\PharmaSUG\2023\QT222	2	2
4	C:\Users\vielk\Documents\My SAS Files\9.4\PharmaSUG\2023\AD221\ad221.sas	3	0
5	C:\Users\vielk\Documents\My SAS Files\9.4\PharmaSUG\2023\AD221\ad221.zip	3	0
6	C:\Users\vielk\Documents\My SAS Files\9.4\PharmaSUG\2023\AD221\ad221_batch.log	3	0
7	C:\Users\vielk\Documents\My SAS Files\9.4\PharmaSUG\2023\AD221\ad221_batch.lst	3	0
8	C:\Users\vielk\Documents\My SAS Files\9.4\PharmaSUG\2023\AD221\ad221_batch.sas	3	0
9	C:\Users\vielk\Documents\My SAS Files\9.4\PharmaSUG\2023\AD221\batch_submit.sas	3	0
10	C:\Users\vielk\Documents\My SAS Files\9.4\PharmaSUG\2023\AD221\compare.sas	3	0
11	C:\Users\vielk\Documents\My SAS Files\9.4\PharmaSUG\2023\QT222\QT222.sas	3	0

Figure 2. The data set created by a call of MAC_U_DIRECTORY_READ.

Examples of calls of MAC_U_DIRECTORY_READ with alternate values of the macro parameters appear below. For instance, a recursive read of a root drive may take hours or only .sas files are of interest (though, hygiene might suggest not to mix file types in certain folders, we find it still occurs).

MAC_U_LOG_CHECK

Appendix 4 presents the macro MAC_U_LOG_CHECK. This macro hard codes patterns (phases or words) to search for in (log) files. The macro parameter, PRINT_ALL (Line 4), is a verbose mode, so that the reviewer can see all of the patterns and the number of occurrences, including 0. If a pattern is found, its frequency is reported and one log line with its line number is reported. The data set is deleted by default (Line 8). The macro reports the path, filename, Last Modified datetime stamp, and the number of log lines read:

```
%mac_u_log_check
   ( log_filename         = %sysfunc( getoption( SASUSER ))\sdtm\f_1_1.log
   , print_file           = %str()
   )

C:\Users\vielk\Documents\My SAS Files\9.4\sdtm\f_1_1.log

Path                                                File            Last Modified
C:\Users\vielk\Documents\My SAS Files\9.4\sdtm    f_1_1.log          27FEB2023:18:51:07

Records read from the log:              29

***** No messages found  *****
```

The Validation programmer may not be able to judge the number of appropriate log lines, but a truncated or empty log will generate the target of zero messages:

```
data _null_ ;
  file "%sysfunc( getoption( SASUSER ))\sdtm\empty.log" ;
  put " " ;
run ;
```

```
%mac_u_log_check
   ( log_filename         = %sysfunc( getoption( SASUSER ))\sdtm\empty.log
   , print_all            = N
   , print_file           = %str()
   )

C:\Users\vielk\Documents\My SAS Files\9.4\sdtm\empty.log

Path                                          File          Last Modified
C:\Users\vielk\Documents\My SAS Files\9.4\sdtm   empty.log     19MAR2023:12:33:14

Records read from the log:                1

***** No messages found  *****
```

An example of a log generated by code that MUST be corrected is:

```
proc printto
   log = "%sysfunc( getoption( SASUSER ))\sdtm\positive_control.log" ;
run ;

data _null_ ;
  x = scan( "Positive control" , y , " " ) ;
run ;

proc printto ;
run ;

%mac_u_log_check
   ( log_filename         = %sysfunc( getoption( SASUSER ))\sdtm\positive_control.log
   , print_all            = N
   , print_file           = %str()
   )
C:\Users\vielk\Documents\My SAS Files\9.4\sdtm\positive_control.log


Path                                          File                 Last Modified
C:\Users\vielk\Documents\My SAS Files\9.4\sdtm  positive_control.log  19MAR2023:12:44:46


message                                                                    frequency
-------------------------------------------------------------------------------------
UNINITIALIZED                                                                      1
_ERROR_=1                                                                          1


Records read from the log:              22

Line = 11
NOTE: Variable y is uninitialized.

Line = 13
x=  y=. _ERROR_=1 _N_=1
```

Certainly, the lines above (11 and 13) are not enough to decipher the origins of the issues, but they are also, typically, not enough to violate requirement of 100% independent peer programming. For absolute compliance, the macro can be modified to not display the lines from the peer program's log.

MAC_U_PATH_LOG_CHECK

Appendix 5 presents the MAC_U_PATH_LOG_CHECK macro. This macro calls MAC_U_LOG_CHECK, creating a "rolling" data set of findings, for presentation or processing. Multiple paths (Line 5) separated by a delimiter (Line 6) can be provided:

```
%mac_u_path_log_check
   ( path                 =  %sysfunc( getoption( SASUSER ))\sdtm
                             #%sysfunc( getoption( SASUSER ))\v_sdtm
   , print_messages_all   = N
   , print_messages_gt_0  = N
   ) ;
```

The results are summarized by file within a path, by path, and in total:

```
                                                                    log_has_    log_has_    log_has_    log_has_
file                          last_modified    records    messages    message     warning     error       stops

empty.log                 19MAR2023:12:33:14         1           0           0           0           0           0
f_1_1.log                 27FEB2023:18:51:07        29           0           0           0           0           0
<SNIPPED>
positive_control.log      19MAR2023:12:44:46        22           2           1           0           1           0
<SNIPPED>
t_1_9.log                 27FEB2023:18:50:57        29           0           0           0           0           0
--------------------                                      --------    --------    --------    --------    --------
Total in path:                                                 2           1           0           1           0

Logs in path: 32

path = C:\Users\vielk\Documents\My SAS Files\9.4\v_sdtm

                                                                    log_has_    log_has_    log_has_    log_has_
file                          last_modified    records    messages    message     warning     error       stops

v_f_1_1.log               27FEB2023:18:51:13        31           0           0           0           0           0
<SNIPPED>
v_t_1_9.log               27FEB2023:18:51:02        31           0           0           0           0           0
-------------------------                                 --------    --------    --------    --------    --------
Total in path:                                                 0           0           0           0           0
=========================                                 ========    ========    ========    ========    ========
Grand total in all paths:                                      2           1           0           1           0

          Logs in path: 30
Total Logs in all paths: 62
```

Clearly, from the perspective of someone who must audit the files in preparation for a delivery, clean logs (and, as we will see below, clean COMPARE's) greatly aid the accuracy and efficiency of the review. Even with attestations by the Main and Validation programmers and the biostats, one may want to visit the logs with issues to verify that they are not critical or fatal. Setting the macro parameter PRINT_MESSAGES_GT_0, for instance, will provide details, but the reading the offending log may be required by a neutral (authorized) person (non-peer programmer). A list of specific logs can also be provided to the macro, for instance, by obtaining a list of every SAS file in a directory and deriving the log paths and names from them (substitution, for instance):

```
%mac_u_directory_read
  ( path                 = %sysfunc( getoption( SASUSER ))\sdtm
  , recursive            = N
  , finfo                = Y
  , code                 = if prxmatch( "/\.sas$/i" , trim( pathfile )) then
  ) ;

data dir_read_logs
        ( keep   = log )
    ;
  set dir_read ;
  log = prxchange( "s/(?<=\.)sas/log/i" , 1 , trim( pathfile )) ;
run ;

%mac_u_path_log_check
   ( in_ds                =  dir_read_logs ) ;
```

For brevity, the results are not shown, but these data can be used to check if any SAS file was (inadvertently) updated since the logs were created:

```
proc sql ;
  select coalescec( scan( scan( a.pathfile , -1 , "\" ) , 1 , "." )
                  , scan( b.file                       , 1 , "." )
                  )
         as base
              length = 50
       , a.last_modified
       , b.last_modified as last_modified_log
  from      dir_read as a
       full join log_check_all as b
          on scan( scan( a.pathfile , -1 , "\" ) , 1 , "." ) = scan( b.file , 1 , "." )
  having a.last_modified > b.last_modified
  order by base
  ;
quit ;
```

```
base                                                 last_modified     last_modified_log
-------------------------------------------------------------------------------------------
t_1_1                                          27FEB2023:19:00:10    27FEB2023:18:50:15
```

MAC_R_COMPARE

Appendix 6 presents the MAC_R_COMPARE macro. This macro parses any COMPARE output in the lst file provided or in all of the lst files in a directory. For brevity, neither the COMPARE results will be displayed, but using the HELP = Y macro parameter (Line 24) displays an annotated example in the log. The example call below creates the output in **Figure 3**:

```
%mac_r_compare_report
   ( path          = %sysfunc( pathname( v_sdtm ))
   , file          = *.lst
   ) ;
```

C:\Users\vielk\Documents\My SAS Files\9.4\v_sdtm

Obs	file	data	comp	data_last_modified	comp_last_modified	datetime_ issue	data_ vars	vars_in_ common	vars_ issue	data_obs	obs_in_ common	obs_ issue	obs_w_ unequal	obs_all_ equal
1	v_f_1_1.lst	SDTM.F_1_1	V_SDTM.V_F_1_1	19MAR2023:16:09:38	19MAR2023:16:09:43		7	7		6	6		0	6
2	v_f_1_10.lst	SDTM.F_1_10	V_SDTM.V_F_1_10	19MAR2023:16:10:25	19MAR2023:16:10:30		7	7		4	4		0	4
3	v_f_1_2.lst	SDTM.F_1_2	V_SDTM.V_F_1_2	19MAR2023:16:09:43	19MAR2023:16:09:48		7	7		4	4		0	4
4	v_f_1_3.lst	SDTM.F_1_3	V_SDTM.V_F_1_3	19MAR2023:16:09:48	19MAR2023:16:09:53		7	7		1	1		0	1
5	v_f_1_4.lst	SDTM.F_1_4	V_SDTM.V_F_1_4	19MAR2023:16:09:53	19MAR2023:16:09:59		7	7		3	3		0	3
6	v_f_1_5.lst	SDTM.F_1_5	V_SDTM.V_F_1_5	19MAR2023:16:09:59	19MAR2023:16:10:04		7	7		2	2		0	2
7	v_f_1_6.lst	SDTM.F_1_6	V_SDTM.V_F_1_6	19MAR2023:16:10:04	19MAR2023:16:10:09		7	7		5	5		0	5
8	v_f_1_7.lst	SDTM.F_1_7	V_SDTM.V_F_1_7	19MAR2023:16:10:09	19MAR2023:16:10:14		7	7		9	9		0	9
9	v_f_1_8.lst	SDTM.F_1_8	V_SDTM.V_F_1_8	19MAR2023:16:10:14	19MAR2023:16:10:20		7	7		8	8		0	8
10	v_f_1_9.lst	SDTM.F_1_9	V_SDTM.V_F_1_9	19MAR2023:16:10:20	19MAR2023:16:10:25		7	7		2	2		0	2
11	v_l_1_1.lst	SDTM.L_1_1	V_SDTM.V_L_1_1	19MAR2023:16:10:30	19MAR2023:16:10:35		7	7		1	1		0	1
12	v_l_1_10.lst	SDTM.L_1_10	V_SDTM.V_L_1_10	19MAR2023:16:11:18	19MAR2023:16:11:23		7	7		8	8		0	8
13	v_l_1_2.lst	SDTM.L_1_2	V_SDTM.V_L_1_2	19MAR2023:16:10:35	19MAR2023:16:10:41		7	7		3	3		0	3
14	v_l_1_3.lst	SDTM.L_1_3	V_SDTM.V_L_1_3	19MAR2023:16:10:41	19MAR2023:16:10:46		7	7		9	9		0	9
15	v_l_1_4.lst	SDTM.L_1_4	V_SDTM.V_L_1_4	19MAR2023:16:10:46	19MAR2023:16:10:51		7	7		5	5		0	5
16	v_l_1_5.lst	SDTM.L_1_5	V_SDTM.V_L_1_5	19MAR2023:16:10:51	19MAR2023:16:10:56		7	7		10	10		0	10
17	v_l_1_6.lst	SDTM.L_1_6	V_SDTM.V_L_1_6	19MAR2023:16:10:57	19MAR2023:16:11:02		7	7		10	10		0	10
18	v_l_1_7.lst	SDTM.L_1_7	V_SDTM.V_L_1_7	19MAR2023:16:11:02	19MAR2023:16:11:07		7	7		7	7		0	7
19	v_l_1_8.lst	SDTM.L_1_8	V_SDTM.V_L_1_8	19MAR2023:16:11:07	19MAR2023:16:11:12		7	7		2	2		0	2
20	v_l_1_9.lst	SDTM.L_1_9	V_SDTM.V_L_1_9	19MAR2023:16:11:12	19MAR2023:16:11:17		7	7		6	6		0	6
21	v_t_1_1.lst	SDTM.T_1_1	V_SDTM.V_T_1_1	19MAR2023:16:08:45	19MAR2023:16:08:50		7	7		9	9		0	9
22	v_t_1_10.lst	SDTM.T_1_10	V_SDTM.V_T_1_10	19MAR2023:16:09:32	19MAR2023:16:09:37		7	7		9	9		0	9
23	v_t_1_2.lst	SDTM.T_1_2	V_SDTM.V_T_1_2	19MAR2023:16:08:50	19MAR2023:16:08:55		7	7		10	10		0	10
24	v_t_1_3.lst	SDTM.T_1_3	V_SDTM.V_T_1_3	19MAR2023:16:08:55	19MAR2023:16:09:00		7	7		4	4		0	4
25	v_t_1_4.lst	SDTM.T_1_4	V_SDTM.V_T_1_4	19MAR2023:16:09:01	19MAR2023:16:09:06		7	7		6	6		0	6
26	v_t_1_5.lst	SDTM.T_1_5	V_SDTM.V_T_1_5	19MAR2023:16:09:06	19MAR2023:16:09:11		7	7		7	7		0	7
27	v_t_1_6.lst	SDTM.T_1_6	V_SDTM.V_T_1_6	19MAR2023:16:09:11	19MAR2023:16:09:16		7	7		7	7		0	7
28	v_t_1_7.lst	SDTM.T_1_7	V_SDTM.V_T_1_7	19MAR2023:16:09:16	19MAR2023:16:09:22		7	7		1	1		0	1
29	v_t_1_8.lst	SDTM.T_1_8	V_SDTM.V_T_1_8	19MAR2023:16:09:22	19MAR2023:16:09:27		7	7		7	7		0	7
30	v_t_1_9.lst	SDTM.T_1_9	V_SDTM.V_T_1_9	19MAR2023:16:09:27	19MAR2023:16:09:32		7	7		7	7		0	7
													=======	
													0	

C:\Users\vielk\Documents\My SAS Files\9.4\v_sdtm
COMPARE results with matching variable and observations numbers and no unequal values

30

Figure 3. The results of MAC_R_COMPARE_REPORT.

MAC_U_COMPARE_REPORT generates a global macro variable (Line 23) that indicates whether any issue was found in any COMPARE RESULTS of the given lst file. This macro variable can be used in a batch submission program to end it without further batch submitting any other program. For instance, if ADSL does not “pass” Validation by virtue of having a different number of observations, variables, or having mismatching values, the processing depending on ADSL, including every subsequent ADaM data set and TFL, should be stopped. SAS can email, if IT allows certain things, so the team can be alerted as soon as fatal issue is encountered.

Allowing issues in a single log may expedite programming at that moment, but it creates problems for audits at the delivery level. For instance, a Validation programmer may reason that her or his program and data sets are not part of the delivery so failing to drop certain “transactional” variables may be acceptable, but when viewing the report in Figure 3, one cannot tell if the Main data set omitted

required variables. Further, variables (and observations) not common to both data sets are not COMPARE'd and one should not assume that they are correct or not required. The macro creates a data set that can be archived or used for metrics a manager or stakeholder might use to judge the quality and efficiency of programming to create more accurate budgets and timelines or to allocate resources (programmers) differently.

MAC_U_DELETE

Appendix 7 presents the MAC_U_DELETE macro. The essence of this macro is the DATASETS procedure and its DELETE statement. The macro checks for the existence of the file before deleting it to avoid messages in the log when it may not exist. The DS macro parameter (Line 5) can take a mixed list of one- or two-level SAS data sets; the data sets can be in different libraries (directories). Several macros call this macro. This is an "uncontrolled" delete, no prompt occurs. In certain environments, the author may use a PW= data set option to avoid inadvertent or unintended deletions (or changes), but that is not a common approach in clinical trial programming.

DEMONSTRATIONS USING %MAC_PGM_TFL

A few contrived examples of "mismatch" can be generated, by design, with the %MAC_PGM_TFL macro. This is just one way to change the files. Positive and negative control test are usually a wise addition to DUT (Device Under Testing).

TEST_1

The first demonstration of an example of these programs used shows how to end batch submission if Validation fails and the abbreviated results of the MAC_R_COMPARE_REPORT macro:

```
%mac_prg_tfl
   ( test_1      = drop = col_1 ) ;
```

The log, among other things, will show:

```
ALERT:  v_t_1_1.lst has mismatches
```

The dropping of COL_1 not only exits MAC_PGM_TFL prematurely, but also shows in the summary of the compare results:

```
%mac_r_compare_report
   ( path         = %sysfunc( pathname( v_sdtm ))
   , file         = *.lst
   ) ;
```

Obs	file	data	comp	datetime_ issue	data_ vars	vars_in_ common	vars_ issue
1	v_t_1_1.lst	SDTM.T_1_1	V_SDTM.V_T_1_1		7	6	Y

```
COMPARE results with matching variable and observations numbers and no unequal values

--------
       0
```

The criteria (Lines 396-399) for the concluding statement can be updated to suite the needs of the team or satisfy the requirements set by the SOPs/Wis. Using a macro parameter might be a good option. Lines 411-417 define criteria to populate the macro parameter MV_FAIL (Line 23, default value COMPARE_FAIL).

TEST_2

TEST_2 demonstrates the situation in which the number of observations differ. When using the ID statement for the COMPARE procedure, the number of observations in both data sets can match, but they may not match on the ID variable. The LISTALL option to the COMPARE statement is useful.

```
%tfl
   ( test_2      = obs = 1 ) ;

%mac_r_compare_report
   ( path         = %sysfunc( pathname( v_sdtm ))
   , file         = *.lst
   ) ;
```

```
                                                    obs_in_    obs_    obs_w_    obs_all_
file         data          comp           data_obs  common  issue    unequal      equal
v_t_1_1.lst SDTM.T_1_1 V_SDTM.V_T_1_1       8         1       Y          0           1
                                                                     =======
                                                                        0
```

Note that although these two data sets have one observation in common, the value of variables match and OBS_W_UNEQUAL is 0 (zero).

TEST_3

TEST_3 demonstrates mismatching (discrepant values). Although this is the final demonstration test, matching values (accuracy) is the primary criterion of Validation. In this type of programming, the values have to match exactly, including formatting, such as leading spaces or SAS inline styles, potentially complicating Validation. Sometimes, additional items cannot be avoided, such as superscripts. Leading spaces in SAS, for instance, to indent a row label under its section header label, is avoidable in SAS using a COMPUTE block in the REPORT procedure and the STYLE INDENT. The author demonstrates this and how to quickly display mismatches that are too long to be displayed by COMPARE results in another paper in this conference[1].

```
%tfl
   ( test_3      = call missing( col_1 ) %str(;))

%mac_r_compare_report
   ( path         = %sysfunc( pathname( v_sdtm ))
   , file         = *.lst
   ) ;
```

```
                                                      obs_w_    obs_all_
Obs       file           data            comp        unequal      equal
 1     v_t_1_1.lst    SDTM.T_1_1    V_SDTM.V_T_1_1      10          0
                                                     =======
                                                        10
```

CONCLUSION

This paper contributes to the essential process of Validation in clinical trial programming by providing programmatic approaches to audit an entire delivery (which may comprise multiple directories). Not only

do we owe a duty to the subjects who volunteer to enter our trial, potentially with meaningful risk, and their loved ones and healthcare providers, to be certain of our work, we program in one of the most regulated fields on trials that can cost millions of dollars and last years, indeed the cumulative costs can run into the tens of millions of dollars and beyond. A second programmatic approach that executes in minutes, both displays the results concisely and creates data sets of them, and itself leaves an audit trail should be a welcomed addition to, but not a substitution for any manual review.

The macros provided can be used by individual programs in production run. For instance, a log check can be incorporated into a macro used at the very end of a program (to close and clean up, for instance). The author creates a batch_submission.sas file for most folders in which he works. In addition to a call to MAC_U_BATCH_SUBMIT, even while developing, the author then calls MAC_U_LOG_CHECK, followed by two calls to MAC_R_COMPARE_REPORT when Validating. The call to the MAC_U_LOG_CHECK in this situation display the results in the output window of interactive SAS. The first call to MAC_R_COMPARE_RESULTS examines the result from the files only in the batch_submisison.sas and the second reports the COMPARE results for the entire directory. This assures the author that the status, for instance, of his current ADaM program, such as V_ADAE.sas, is not affected by issues in the COMPARE results of V_ADSL.sas, on which it relies.

Programming in clinical trials can have strict deadlines and strict rules. In the case of a delivery in a set number of business days after a data lock, the time to just review the Validation results may be nearly prohibitive: viewing the COMPARE results of just 30-50 ADaM data sets could take hours; TFL, which can easily require 300-500 programs, have the added complexity that the output must be viewed for formatting issues, even if the VALDS match 100%, and their titles and footnotes also have to be verified. Programmatically approaching some issues, like batch submission in appropriate order and summarizing logs and COMPARE results at the delivery level may increase the accuracy, completeness of the review, and efficiency, but the SOPs/WIs must allow such approaches. The ability to audit directories over time can also provide metrics to a manager that indicate which programs might cause issues over several deliverables or how to create realistic timelines or allocate resources and budgets. The ability to view a log from a batch submission program and the datetime stamps to assure a manager or lead that the programs were run correctly and that the files have appropriate (relative) ages increases confidence in the quality of the delivery.

The macros and code in this paper are provided "**as is**". The author and his employers assume no responsibility for their use, but do appreciate notification of errors, bugs, corrections, or suggestions.

REFERENCES

[1] Viel, K. 2023. " A SAS® System Macro to Quickly Display Discrepant Values that are too Long for the COMPARE Procedure Output." Proceedings of the PharmaSUG 2023 Conference, San Francisco, CA: PharmaSUG. In press.

CONTACT INFORMATION

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

Kevin R. Viel, Ph.D.

Navitas Data Sciences
kevin.viel@navitaslifesciences.com
www.navitasdatasciences.com

Histonis, Incorporated
kviel@histonis.org

APPENDIX

Appendix 1. A suggested macro to provide test programs, logs, lsts, and output.

```
%macro mac_pgm_tfl
   ( list        = t f l
   , number      = 10
   , end_on_fail = Y
   , test_1      = %str( )
   , test_2      = %str( )
   , test_3      = %str( )
   ) ;

  /****/
  proc datasets
     library = SDTM
     nolist
     memtype = data
     kill
     ;
  quit ;

  proc datasets
     library = V_SDTM
     nolist
     memtype = data
     kill
     ;
  quit ;

  %mac_u_directory_read
    ( path                 =  %sysfunc( pathname( sdtm   ))
                            #%sysfunc( pathname( v_sdtm ))
    , recursive            = N
    , code                 = if prxmatch( "/(?:lst|log|sas|rtf)$/i" , trim( pathfile )) then
    ) ;

  data _null_ ;
    set dir_read ;
    __rc = filename( "temp"
                   , pathfile
                   ) ;
    __rc = fdelete( "temp" ) ;
    __rc = filename( "temp" ) ;
  run ;

  /****/
  %let __i  = 1 ;
  %let l    = %sysfunc( scan( &list. , &__i. , %str( ))) ;

  ods listing close ;
  ods noresults ;

  %do %while ( &l. ne %str( )) ;

     %do __j = 1 %to &number. ;

        proc printto
           log   = "%sysfunc( pathname( sdtm ))\&l._1_&__j..log"
           new
           ;
        run ;

        data sdtm.&l._1_&__j. ;

            file "%sysfunc( pathname( sdtm ))\&l._1_&__j..sas" ;
            put "placeholder" ;

            length page_1          $ 1
                   page_order_1    8
                   section_1       $ 1
                   section_order_1 8
                   row_1           $ 6
                   row_order_1     8
                   col_1           $ 10
                   ;

            retain page_1          "Safety Population"
                   page_order_1    1
                   section_1       "Sex n(%)"
                   section_order_1 1
                   ;

            do row_order_1 = 1 to ceil( ranuni( 0 ) * 10 ) ;

               row_1 = cat( "Row " , strip( put( row_order_1 , 8. ))) ;
               col_1 = cat( strip( put( row_order_1 , 8. ))
```

```
                          , " ("
                          , strip( put( row_order_1 / 50 * 100 , 8.1 ))
                          , ")"
                          ) ;
                output ;

            end ;

        run ;

        ods rtf
            file = "%sysfunc( pathname( sdtm ))\&l._1_&__j..rtf"
            ;

        proc print
           data = sdtm.&l._1_&__j. ;
        run ;

        ods rtf close ;

        proc printto ;
        run ;

        /************/
        data _null_ ;
          call sleep( 5 , 1 ) ;
        run ;

        proc printto
           log   = "%sysfunc( pathname( v_sdtm ))\v_&l._1_&__j..log"
           print = "%sysfunc( pathname( v_sdtm ))\v_&l._1_&__j..lst"
           new
           ;
        run ;

        data v_sdtm.v_&l._1_&__j.
                ( &test_1. )
            ;
          file "%sysfunc( pathname( v_sdtm ))\v_&l._1_&__j..sas" ;
          put "placeholder" ;
          set sdtm.&l._1_&__j.
                 ( &test_2. )
              ;
          &test_3.
        run ;

        /***********/
        proc compare
           data = sdtm.&l._1_&__j.
           comp = v_sdtm.v_&l._1_&__j.
           listall
           ;
          id page_order_1
             section_order_1
             row_order_1
             ;
        run ;

        proc printto ;
        run ;

        %mac_r_compare_report
           ( path         = %sysfunc( pathname( v_sdtm ))
           , file         = v_&l._1_&__j..lst
           , print        = N
           ) ;

        %if &compare_fail. = 1
        %then
           %do ;
               %put A%str(LERT: ) v_&l._1_&__j..lst has mismatches ;
               %if &end_on_fail. = Y %then %goto __END ;
           %end ;

     %end ;

     %let __i  = %eval( &__i. + 1 ) ;
     %let l    = %sysfunc( scan( &list. , &__i. , %str( ))) ;

  %end ;

  %__END:

    ods listing ;
    ods results ;

%mend  mac_pgm_tfl ;
```

Appendix 2. The MAC_U_BATCH_SUBMIT macro.

```
%macro mac_u_batch_submit
   ( ds             = %str()
   , pathfile       = %str()
   , derive_log_lst = Y
   , log            = %str()
   , lst            = %str()
   , sas_exe        = ~\SASHome\SASFoundation\9.4\sas.exe
   , sas_cfg        = ~\SASHome\SASFoundation\9.4\nls\u8\sasv9.cfg
   , sasuser        = !userprofile
   , nosplash       = Y
   , nostatuswin    = Y
   , display_paths  = N
   , help           = N
   ) ;

  %let xwait   = %sysfunc( getoption ( xwait )) ;
  %let xsync   = %sysfunc( getoption ( xsync )) ;
  %let xmin    = %sysfunc( getoption ( xmin  )) ;

  %if &help. = Y
  %then
     %do ;
        %let mprint_orig = %sysfunc(getoption(mprint)) ;
        options nomprint ;

        skip ;
        skip ;
        %put ____________________________________________________________________________________________________________________ ;
        %put Purpose of program:  This utility macro submits SAS programs in the input data set or a single program provided by ;
        %put %str(                    )the user in batch. Each program is run by a separate SAS session sequentially, not in   ;
        %put %str(                    ) parallel.                                                                               ;
        skip ;
        %put Macro Parameter    Description                                                                                      ;
        %put _______________    ____________________________________________________________________________________________     ;
        %put ds                 = Name of the data set with the variable PATHFILE, the path and filename with .sas extension of ;
        %put %str(                    )the program to run. PATHFILE is the argument to -SYSIN                                  ;
        %put pathfile           = The path and file name of the program to run individually by batch submission.                  ;
        %put %str(                    )PATHFILE is the argument to -SYSIN                                                      ;
        %put %str(                       )Default: %nrstr(%%)str%str(())                                                       ;
        %put Provide only one of DS or PATHFILE.                                                                                ;
        skip ;
        %put log                = The log path and filename. LOG is the argument to -ALTLOG                                      ;
        %put %str(                       )Default: %nrstr(%%)str%str(())                                                       ;
        %put lst                = The lst path and filename. LST is the argument to -LST                                         ;
        %put %str(                       )Default: %nrstr(%%)str%str(())                                                       ;
        %put sas_exe            = The path and filename of the SAS executable.                                                   ;
        %put %str(                       )Default: E:\SASHome\SASFoundation\9.4\sas.exe                                         ;
        %put sas_cfg               = The path and filename of the SAS .cfg file.                                                 ;
        %put %str(                       )Default: ~\SASFoundation\9.4\nls\u8\sasv9.cfg                                         ;
        %put sasuser               = The argument to the -SASUSER option                                                        ;
        %put %str(                          )Default: !userprofile                                                             ;
        %put nosplash              = Whether to use the option -NOSPLASH                                                        ;
        %put %str(                          )Default: Y                                                                        ;
        %put nostatuswin           = Whether to use the option -NOSTATUSWIN                                                     ;
        %put %str(                          )Default: Y                                                                        ;
        %put ____________________________________________________________________________________________________________________ ;
        %put End of help  ;

        options &mprint_orig. ;

        %goto __END ;
     %end ;

  %let delete_ds = N ;

  options noxwait
          xsync
          xmin
          ;

  %if     &ds.                  = %str()
      and %nrbquote(&pathfile.) = %str()
  %then
     %do ;
        %put ER%str(ROR:) you must provide a data set name to DS or a path and filename to PATHFILE ;
        %goto __END ;
     %end ;

  %if     %nrbquote(&pathfile.)       ne %str()
      and %nrbquote(&derive_log_lst.) ne Y
      and (    %nrbquote(&log.) ne %str()
           or %nrbquote(&lst.) ne %str()
          )
  %then
      %do ;
         %put W%str(ARNING: ) LOG and LST must both have values if PATHFILE is populated and DERIVE_LOG_LST ~= Y. ;
         %goto __END ;
      %end ;

  %if     &ds.                  ne %str()
```

```
      and %nrbquote(&pathfile.) ne %str()
   %then
      %do ;
          %put ER%str(ROR:) you can only provide a data set name to DS or a path and filename to PATHFILE ;
          %goto __END ;
      %end ;
    %else %if     &ds.                  =  %str()
              and %nrbquote(&pathfile.) ne %str()
   %then
      %do ;
          data __ds ;
            length pathfile
                   log
                   lst       $ 1000
                   ;
            pathfile = "&pathfile." ;

            %if %nrbquote(&derive_log_lst.) ne Y
            %then
               %do ;
                   log      = "&log." ;
                   lst      = "&lst." ;
               %end ;

             %else
                %do ;
                    log = prxchange( "s/\.sas$/.log/i"
                                   , 1
                                   , trim( prxchange( "s/\\PGM\\/\\LOG\\/i"
                                                    , -1
                                                    , pathfile
                                                    )
                                         )
                                   ) ;

                    lst = prxchange( "s/\.sas$/.lst/i"
                                   , 1
                                   , trim( prxchange( "s/\\PGM\\/\\LST\\/i"
                                                    , -1
                                                    , pathfile
                                                    )
                                         )
                                   ) ;
                %end ;

          run ;

          %let ds = __ds ;
          %let delete_ds = Y ;

      %end ;

   /* Verify that the input data set exists, that the files it contains exists, and that they are .sas files.
      The macro stops once one of these criteria fail.
   */
   %if %sysfunc( exist( &ds. ))
   %then
      %do ;

          %let __flag = 0 ;

          data _null_ ;
            set &ds. ;
            __fileexist = fileexist( pathfile ) ;
            __sas       = prxmatch( "/\.sas$/i" , strip( pathfile )) ;
            if     __fileexist = 0
               or __sas       = 0
            then
              do ;
                 if __fileexist = 0 then put "ER" "ROR: the file " pathfile "does not exist. Processing will stop" ;
                 if __sas       = 0 then put "ER" "ROR: the file " pathfile "is not a .sas program. Processing will stop" ;
                 call symput( "__flag" , "1" ) ;
                 stop ;
              end ;

            /* Check for the existence of log and lst paths */
            logpath = prxchange( "s/^(.*)(?:\\[^\\]+\.log)$/$1/i"
                               , 1
                               , strip( log )
                               ) ;
            __logexist = fileexist( logpath ) ;
            if     __logexist = 0
            then
              do ;
                 put "ER" "ROR: the path " logpath "does not exist. Processing will stop" ;
                 call symput( "__flag" , "1" ) ;
                 stop ;
              end ;

            lstpath = prxchange( "s/^(.*)(?:\\[^\\]+\.lst)$/$1/i"
                               , 1
                               , strip( lst )
                               ) ;
            __lstexist = fileexist( lstpath ) ;
            if     __lstexist = 0
```

```
186             then
187               do ;
188                  put "ER" "ROR: the path " lstpath "does not exist. Processing will stop" ;
189                  call symput( "__flag" , "1" ) ;
190                  stop ;
191               end ;
192
193           run ;
194
195           %if &__flag. = 1 %then %goto __END ;
196
197        %end ;
198
199     %else
200        %do ;
201            %put ER%str(ROR:) the data set &ds. does not exist. ;
202            %goto __END ;
203        %end ;
204
205     data _null_ ;
206       set &ds. ;
207
208       %if &display_paths. ne Y
209       %then
210          %do ;
211              call execute( cat( "x '"
212                                 , %unquote(%str(%'%")&sas_exe.%str(%" %'))
213                                 , %unquote(%str(%'-CONFIG %")&sas_cfg.%str(%" %'))
214                                 %if &nosplash.    = Y %then , '-NOSPLASH ' ;
215                                 %if &nostatuswin. = Y %then , "-NOSTATUSWIN " ;
216                                 %if %nrbquote(&sasuser.) ne %str() %then , "-SASUSER &sasuser. " ;
217                                 , "-SYSIN "
218                                 , '"'
219                                 , strip( pathfile )
220                                 , '" '
221                                 , "-NOLOG "
222                                 , "-ALTLOG "
223                                 , '"'
224                                 , strip( log )
225                                 , '" '
226                                 , "-PRINT "
227                                 , '"'
228                                 , strip( lst )
229                                 , '" '
230                                 , "' ;"
231                                 )
232                           ) ;
233          %end ;
234       %else
235          %do ;
236              length file_path
237                     file_log
238                     file_lst $ 100
239                     ;
240              file_path = scan( pathfile , -1 , "\" ) ;
241              file_log  = scan( log      , -1 , "\" ) ;
242              file_lst  = scan( lst      , -1 , "\" ) ;
243              put  @1   file_path
244                   @100 pathfile
245                 / @1   file_log
246                   @100 log
247                 / @1   file_lst
248                   @100 lst
249                 /
250                 ;
251          %end ;
252       ;
253
254     run ;
255
256     %__END:
257
258     options &xwait.
259             &xsync.
260             &xmin.
261             ;
262
263  %mend  mac_u_batch_submit ;
264
```

Appendix 3. The MAC_U_DIRECTORY_READ macro.

```
%macro mac_u_directory_read
    ( path                =
    , path_delim          = #
    , keep                = pathfile
    , recursive           = Y
    , levels              = 20
    , output_dir          = Y
    , output_files        = Y
    , out                 = dir_read
    , code                = %str()
    , os_path_delim       = \
    , finfo               = N
    , foptname            = Last Modified
    , foptname_delim      = #
    , help                = N
    ) ;

  %if &help. = Y
  %then
     %do ;
         %let mprint_orig = %sysfunc(getoption(mprint)) ;
         options nomprint ;

         skip ;
         skip ;
         %put ____________________________________________________________________________________________________________________________ ;
         %put Purpose of program:     This utility macro                                                                                  ;
         %put %str(                           )1) Reads the contents of a list of directories (paths).                                    ;
         %put %str(                           )2) Optionally recursively reads the child directories.                                     ;
         %put %str(                           )3) Optionally limites the level of recursion.                                              ;
         %put %str(                           )4) Optionally includes free-text code prior to the OUTPUT statement.                       ;
         %put %str(                           )5) Optionally provides FINFO.                                                              ;
         skip ;
         %put Macro Parameter         Description                                                                                         ;
         %put ___________________     ____________________________________________________________________________________________________ ;
         %put path                  = List of paths, separated by PATH_DELIM, of the directories whose contents will be read              ;
         %put path_delim            = Delimiter of the separate paths in PATH                                                           ;
         %put %str(                           )Default: #                                                                                 ;
         %put keep                  = Variables to keep                                                                                 ;
         %put %str(                           )Default: pathfile                                                                          ;
         %put recursive             = Whether to recursively read the child directories.                                                 ;
         %put %str(                           )Default: Y                                                                                 ;
         %put levels                = How many levels to recursively read.                                                              ;
         %put %str(                           )Default: 20                                                                                ;
         %put output_dir            = Whether to output directories.                                                                    ;
         %put %str(                           )Default: Y                                                                                 ;
         %put output_files          = Whether to output files.                                                                          ;
         %put %str(                           )Default: Y                                                                                 ;
         %put out                   = LIBREF.DATASET name of the SAS data set created                                                   ;
         %put %str(                           )Default: dir_read                                                                          ;
         %put code                  = Free-text SAS code. Note the need for %nrstr(%%)str%str((;)).                                     ;
         %put %str(                           )Default: %nrstr(%%)str%str(())                                                             ;
         %put os_path_delim         = The delimiter of the path in the operating system.                                                ;
         %put %str(                           )Default:  \                                                                                ;
         %put finfo                 = Whether to obtain file information.                                                               ;
         %put %str(                           )Default:  N                                                                                ;
         %put foptname              = The names of the options to obtain.                                                               ;
         %put %str(                          )Filename                                                                                    ;
         %put %str(                          )Owner Name         (Unix)                                                                   ;
         %put %str(                          )Group Name         (Unix)                                                                   ;
         %put %str(                          )Access Permission  (Unix)                                                                   ;
         %put %str(                          )File Size (bytes)  (Windows)                                                                ;
         %put %str(                          )Last modified                                                                               ;
         %put %str(                            )Default:    Last Modified                                                                ;
         %put %str(                                     )# Owner Name                                                                     ;
         %put foptname_delim        = Delimiter of the separate file options in FOPTNAME.                                               ;
         %put %str(                           )Default:  #                                                                                ;
         %put ____________________________________________________________________________________________________________________________ ;
         %put End of help                                                                                                                 ;

         options &mprint_orig. ;
         %goto __END ;
     %end ;


  %if     &output_dir.   ne Y
      and &output_files. ne Y
  %then
     %do ;
         %put ER%str(ROR:) OUTPUT_DIR or OUTPUT_FILES or both must be Y ;
         %put ER%str(ROR:) OUTPUT_DIR   = &output_dir. ;
         %put ER%str(ROR:) OUTPUT_FILES = &output_files. ;
         %goto __END ;
     %end ;

  %mac_u_delete
      ( ds = &out.
             __dir
      ) ;

  data &out.
          ( keep = &keep.
```

```
                    %if %sysfunc( prxmatch( /pathfile/ , &keep. )) = 0 %then pathfile ;
                    __dopen
                    __level
                    %if &finfo. = Y
                    %then
                       %do ;
                           /* i=1 foptname=Filename
                              i=2 foptname=RECFM
                              i=3 foptname=LRECL
                              i=4 foptname=File Size (bytes)
                              i=5 foptname=Last Modified
                              i=6 foptname=Create Time
                           */
                           %if %sysfunc( prxmatch( /Filename/i            , &foptname. )) %then filename           ;
                           %if %sysfunc( prxmatch( /Owner Name/i          , &foptname. )) %then owner              ;
                           %if %sysfunc( prxmatch( /Group Name/i          , &foptname. )) %then goup               ;
                           %if %sysfunc( prxmatch( /Access Permission/i   , &foptname. )) %then access_permission  ;
                           %if %sysfunc( prxmatch( /Last modified/i       , &foptname. )) %then last_modified      ;
                           %if %sysfunc( prxmatch( /File Size \(bytes\)/i , &foptname. )) %then file_size_bytes    ;
                       %end ;
                )
             ;

      length __path   $ 512
             object   $ 256
             pathfile $ 512

             %if %sysfunc( prxmatch( /\bfile/ , &keep. )) %then file $ 256 ;

             %if &finfo. = Y
             %then
                %do ;
                    foptname $ 100
                    finfo    $ 512
                    %if %sysfunc( prxmatch( /Filename/i          , &foptname. )) %then filename           $ 512 ;
                    %if %sysfunc( prxmatch( /Owner Name/i        , &foptname. )) %then owner              $  30 ;
                    %if %sysfunc( prxmatch( /Group Name/i        , &foptname. )) %then goup               $  50 ;
                    %if %sysfunc( prxmatch( /Access Permission/i , &foptname. )) %then access_permission  $  30 ;
                %end ;
             ;

      do __p = 1 to countc( "&path." , "&path_delim." ) + 1 ;

                             /* Remove the trailing OS path delimiter, if it is present */
        __path   = prxchange( "s/(.*)(?:\&os_path_delim.)$/$1/"
                            , 1
                            , strip( scan( "&path." , __p , "&path_delim." ))
                            ) ;

        __level  = 1 ;

        __rc     = filename( "dir" , __path ) ;
        dopen    = dopen( "dir" ) ;

        /**************/
        if dopen > 0
        then
          do ;

             dnum = dnum( dopen ) ;

             if dnum > 0
             then
               do ;
                  do objectnum = 1 to dnum ;

                    object   = dread( dopen , objectnum ) ;
                    pathfile = catx( "&os_path_delim."
                                   , __path
                                   , object
                                   ) ;

                    call missing( finfo
                                %if %sysfunc( prxmatch( /Filename/i            , &foptname. )) %then , filename           ;
                                %if %sysfunc( prxmatch( /Owner Name/i          , &foptname. )) %then , owner              ;
                                %if %sysfunc( prxmatch( /Group Name/i          , &foptname. )) %then , goup               ;
                                %if %sysfunc( prxmatch( /Access Permission/i   , &foptname. )) %then , access_permission  ;
                                %if %sysfunc( prxmatch( /Last modified/i       , &foptname. )) %then , last_modified      ;
                                %if %sysfunc( prxmatch( /File Size \(bytes\)/i , &foptname. )) %then , file_size_bytes    ;
                                ) ;

                    /* Attempt to open the object as a directory */
                    __rc       = filename( "dirchild" , pathfile ) ;
                    __dopen    = dopen( "dirchild" ) ;
                    __rc       = dclose( __dopen ) ;
                    __rc       = filename( "dirchild" ) ;

                    %if %sysfunc( prxmatch( /path\b/ , &keep. ))
                    %then
                       %do ;
                           if __dopen = 0
                           then
                             do ;
                                path = __path ;
                                %if %sysfunc( prxmatch( /\bfile/ , &keep. )) %then file = object %str(;) ;
```

```
188                            end ;
189                           else
190                             do ;
191                                path = pathfile ;
192                                %if %sysfunc( prxmatch( /\bfile/ , &keep. )) %then file = " " %str(;) ;
193                             end ;
194                      %end ;
195
196                   %if &finfo. = Y
197                   %then
198                      %do ;
199
200                          rc  = filename( "fn" , pathfile ) ;
201                          fid = fopen( "fn" ) ;
202                          if fid ne 0
203                          then
204                            do ;
205
206                               call missing( finfo
207                                           %if %sysfunc( prxmatch( /Filename/i            , &foptname. ))
208                                           %then , filename           ;
209                                           %if %sysfunc( prxmatch( /Owner Name/i          , &foptname. ))
210                                           %then , owner              ;
211                                           %if %sysfunc( prxmatch( /Group Name/i          , &foptname. ))
212                                           %then , goup               ;
213                                           %if %sysfunc( prxmatch( /Access Permission/i   , &foptname. ))
214                                           %then , access_permission  ;
215                                           %if %sysfunc( prxmatch( /Last modified/i       , &foptname. ))
216                                           %then , last_modified      ;
217                                           %if %sysfunc( prxmatch( /File Size \(bytes\)/i , &foptname. ))
218                                           %then , file_size_bytes    ;
219                                           ) ;
220
221                               do i = 1 to countc( "&foptname." , "&foptname_delim." ) + 1 ;
222
223                                 foptname = strip( scan( "&foptname." , i , "&foptname_delim." )) ;
224
225                                 finfo    = finfo( fid , foptname ) ;
226
227                                 select ( upcase( foptname )) ;
228                                   when ( "FILENAME"          ) filename          = finfo ;
229                                   when ( "OWNER NAME"        ) owner             = finfo ;
230                                   when ( "GROUP NAME"        ) group             = finfo ;
231                                   when ( "ACCESS PERMISSION" ) access_permission = finfo ;
232                                   when ( "LAST MODIFIED"     ) last_modified     = input( finfo , datetime20. ) ;
233                                   when ( "FILE SIZE (BYTES)" ) file_size_bytes   = input( finfo , best.       ) ;
234
235                                   otherwise put "WAR" "NING: " foptname= finfo= ;
236
237                                 end ;
238
239                               end ;
240
241                               close = fclose( fid ) ;
242                               rc    = filename( "fn" ) ;
243
244                            end ;
245
246                      %end ; /* END OF &finfo. = Y */
247
248                   %if &output_files. = N
249                   %then
250                      %do ;
251                          if __dopen > 0
252                          then
253                            do ;
254                                &code.
255                                output ;
256                             end ;
257                      %end ;
258                    %else
259                       %do ;
260                           &code.
261                           output ;
262                        %end ;
263
264                end ; /* CYCLED THROUGH objectnum = 1 to dnum */
265
266             end ; /* END OF dnum > 0 */
267
268           __rc   = dclose( dopen ) ;
269
270         end ; /* END OF dopen > 0 */
271
272       __rc = dclose( dopen ) ;
273       __rc = filename( "dir" ) ;
274
275     end ; /* END OF cycled through __p */
276
277     stop ;
278
279     %if     &finfo. = Y
280         and %sysfunc( prxmatch( /Last Modified/i , &foptname. ))
281     %then format last_modified datetime20. %str(;) ;
282
```

```
283     run ;
284
285     %if &recursive. = Y
286     %then
287        %do ;
288
289            proc sql noprint ;
290              select count( * ) into : dirs
291              from &out.
292              where __dopen > 0
293              ;
294            quit ;
295
296            %if &sqlobs. > 0
297            %then
298               %do ;
299                   data __dirs
300                          ( keep = __path
301                                   __level
302                          )
303                         ;
304                     set &out.
305                            ( rename = ( pathfile = __path )
306                              where  = ( __dopen > 0 )
307                            )
308                          ;
309                   run ;
310               %end ;
311
312            %if &output_dir. = N
313            %then
314               %do ;
315                   data &out. ;
316                      set &out.
317                             ( where = ( __dopen = 0 ))
318                           ;
319                   run ;
320               %end ;
321
322            %let level = 1 ;
323
324            %do %while (     &dirs.  > 0
325                        and &level. < &levels.
326                       ) ;
327
328               %let level = %eval( &level. + 1 ) ;
329
330               data __dirs
331                      ( keep = &keep.
332                               %if %sysfunc( prxmatch( /pathfile/ , &keep. )) = 0 %then pathfile ;
333                               __dopen
334                               __level
335                               %if &finfo. = Y
336                               %then
337                                  %do ;
338                                      %if %sysfunc( prxmatch( /Filename/i            , &foptname. )) %then filename           ;
339                                      %if %sysfunc( prxmatch( /Owner Name/i          , &foptname. )) %then owner              ;
340                                      %if %sysfunc( prxmatch( /Group Name/i          , &foptname. )) %then goup               ;
341                                      %if %sysfunc( prxmatch( /Access Permission/i   , &foptname. )) %then access_permission  ;
342                                      %if %sysfunc( prxmatch( /Last modified/i       , &foptname. )) %then last_modified      ;
343                                      %if %sysfunc( prxmatch( /File Size \(bytes\)/i , &foptname. )) %then file_size_bytes    ;
344                                  %end ;
345                      )
346                     ;
347
348                  length __path   $ 512
349                         object   $ 256
350                         pathfile $ 512
351
352                         %if %sysfunc( prxmatch( /\bfile/ , &keep. )) %then file $ 256 ;
353
354                         %if &finfo. = Y
355                         %then
356                            %do ;
357                                foptname $ 100
358                                finfo    $ 512
359                                %if %sysfunc( prxmatch( /Filename/i          , &foptname. )) %then filename           $ 512 ;
360                                %if %sysfunc( prxmatch( /Owner Name/i        , &foptname. )) %then owner              $  30 ;
361                                %if %sysfunc( prxmatch( /Group Name/i        , &foptname. )) %then goup               $  50 ;
362                                %if %sysfunc( prxmatch( /Access Permission/i , &foptname. )) %then access_permission  $  30 ;
363                            %end ;
364
365                         ;
366
367                  set __dirs ;
368
369                  __path = prxchange( "s/(.*)(?:\&os_path_delim.)$/$1/"
370                                    , 1
371                                    , strip( __path )
372                                    ) ;
373
374                  __level = __level + 1 ;
375
376                  __rc    = filename( "dir" , __path ) ;
377                  dopen   = dopen( "dir" ) ;
```

```
378
379                 /**************/
380                 if dopen > 0
381                 then
382                   do ;
383
384                       dnum = dnum( dopen ) ;
385
386                       if dnum > 0
387                       then
388                         do ;
389                            do objectnum = 1 to dnum ;
390
391                              object   = dread( dopen , objectnum ) ;
392                              pathfile = catx( "&os_path_delim."
393                                             , __path
394                                             , object
395                                             ) ;
396
397                              call missing( finfo
398                                          %if %sysfunc( prxmatch( /Filename/i            , &foptname. )) %then , filename          ;
399                                          %if %sysfunc( prxmatch( /Owner Name/i          , &foptname. )) %then , owner             ;
400                                          %if %sysfunc( prxmatch( /Group Name/i          , &foptname. )) %then , goup              ;
401                                          %if %sysfunc( prxmatch( /Access Permission/i   , &foptname. )) %then , access_permission ;
402                                          %if %sysfunc( prxmatch( /Last modified/i       , &foptname. )) %then , last_modified     ;
403                                          %if %sysfunc( prxmatch( /File Size \(bytes\)/i , &foptname. )) %then , file_size_bytes   ;
404                                          ) ;
405
406                              /* Attempt to open the object as a directory */
407                              __rc       = filename( "dirchild" , pathfile ) ;
408                              __dopen    = dopen( "dirchild" ) ;
409                              __rc       = dclose( __dopen ) ;
410                              __rc       = filename( "dirchild" ) ;
411
412                              %if %sysfunc( prxmatch( /path\b/ , &keep. ))
413                              %then
414                                 %do ;
415                                     if __dopen = 0
416                                     then
417                                       do ;
418                                          path = __path ;
419                                          %if %sysfunc( prxmatch( /\bfile/ , &keep. )) %then file = object %str(;) ;
420                                       end ;
421                                      else
422                                        do ;
423                                           path = pathfile ;
424                                           %if %sysfunc( prxmatch( /\bfile/ , &keep. )) %then file = " " %str(;) ;
425                                       end ;
426                                 %end ;
427
428                              %if &finfo. = Y
429                              %then
430                                 %do ;
431
432                                     rc  = filename( "fn" , pathfile ) ;
433                                     fid = fopen( "fn" ) ;
434                                     if fid ne 0
435                                     then
436                                       do ;
437
438                                         call missing( finfo
439                                                     %if %sysfunc( prxmatch( /Filename/i            , &foptname. ))
440                                                     %then , filename          ;
441                                                     %if %sysfunc( prxmatch( /Owner Name/i          , &foptname. ))
442                                                     %then , owner             ;
443                                                     %if %sysfunc( prxmatch( /Group Name/i          , &foptname. ))
444                                                     %then , goup              ;
445                                                     %if %sysfunc( prxmatch( /Access Permission/i   , &foptname. ))
446                                                     %then , access_permission ;
447                                                     %if %sysfunc( prxmatch( /Last modified/i       , &foptname. ))
448                                                     %then , last_modified     ;
449                                                     %if %sysfunc( prxmatch( /File Size \(bytes\)/i , &foptname. ))
450                                                     %then , file_size_bytes   ;
451                                                     ) ;
452
453                                         do i = 1 to countc( "&foptname." , "&foptname_delim." ) + 1 ;
454
455                                           foptname = strip( scan( "&foptname." , i , "&foptname_delim." )) ;
456
457                                           finfo    = finfo( fid , foptname ) ;
458
459                                           select ( upcase( foptname )) ;
460                                             when ( "FILENAME"          ) filename          = finfo ;
461                                             when ( "OWNER NAME"        ) owner             = finfo ;
462                                             when ( "GROUP NAME"        ) group             = finfo ;
463                                             when ( "ACCESS PERMISSION" ) access_permission = finfo ;
464                                             when ( "LAST MODIFIED"     ) last_modified     = input( finfo , datetime20. ) ;
465                                             when ( "FILE SIZE (BYTES)" ) file_size_bytes   = input( finfo , best.       ) ;
466
467                                              otherwise put "WAR" "NING: " foptname= finfo= ;
468                                           end ;
469
470                                         end ;
471
472                                         close = fclose( fid ) ;
```

```
                                    end ; /* END OF fid ne 0 */

                                  rc  = filename( "fn" ) ;

                               %end ;

                            %if &output_files. = N
                            %then
                               %do ;
                                   if __dopen > 0
                                   then
                                     do ;
                                         &code.
                                         output ;
                                     end ;
                               %end ;
                             %else
                                %do ;
                                    &code.
                                    output ;
                                %end ;

                         end ; /* CYCLED THROUGH objectnum = 1 to dnum */

                     end ; /* END OF dnum > 0 */

                 end ; /* END OF dopen > 0 */

               __rc = dclose( dopen ) ;
               __rc = filename( "dir" ) ;

            run ;

            proc append
               base = &out.
               data =  __dirs
                         %if &output_dir.   = N %then ( where = ( __dopen = 0 )) ;
               ;
            run ;

            proc sql noprint ;
              select count( * ) into : dirs
              from __dirs
              where __dopen > 0
              ;
            quit ;

            %if &sqlobs. > 0
            %then
               %do ;
                   data __dirs
                           ( keep = __path
                                    __level
                           )
                       ;
                     set __dirs
                            ( rename = ( pathfile = __path )
                              where  = ( __dopen  > 0 )
                            )
                     ;
                   run ;
               %end ;

         %end ; /* END OF ( &dirs. > 0 and &level. < &levels. ) */

         %mac_u_delete
            ( ds = __dirs ) ;

      %end ; /* END OF recursive = Y */

    %else %if     &recursive.  ne Y
              and &output_dir. =  N
    %then
       %do ;
           data &out. ;
             set &out.
                    ( where = ( __dopen = 0 ))
               ;
           run ;
        %end ;

      %if %sysfunc( prxmatch( /pathfile/ , &keep. )) = 0
      %then
         %do ;
             proc sql ;
               alter table &out.
               drop pathfile
               ;
             quit ;
         %end ;

   %__END:

%mend mac_u_directory_read ;
```

Appendix 4. The MAC_U_LOG_CHECK macro.

```
%macro mac_u_log_check
   ( log_filename         = &logpathname.
   , print                = Y
   , print_all            = Y
   , print_file           = &lstpathname.
   , print_append         = Y
   , report               = Y
   , delete               = 1
   , warn_unbalance_quote = N
   , help                 = N
   ) ;

  %if &help. = Y
  %then
     %do ;
         %let mprint_orig = %sysfunc(getoption(mprint)) ;
         options nomprint ;

         skip ;
         skip ;
         %put ____________________________________________________________________________________________________________________ ;
         %put Purpose of program:        This utility macro should be run at the end of a program or sub-program. It reports the ;
         %put %str(                           )whether ERRORs, WARNINGs, or other NOTEs of interest appear in the log, how many  ;
         %put %str(                           )times, and on what line(s).                                                       ;
         %put                                                                                                                    ;
         %put Macro Parameter         Description                                                                                ;
         %put _________________       ___________________________________________________________________________________________ ;
         %put log_filename         =  Path\Filename of the SAS log file to review                                                ;
         %put %str(                           )Default: %nrstr(&)logpathname.                                                    ;
         %put print_all            =  Print the entire list of messages even those count = 0                                     ;
         %put %str(                           )Default: Y                                                                        ;
         %put print_file           =  Path\Filename of the file to write the results via PRINTTO procedure                       ;
         %put %str(                           )Default: %nrstr(&)lstpathname.                                                    ;
         %put print_append         =  If writing to PRINT_FILE, should the macro append or write a new file,                     ;
         %put %str(                        )i.e. NEW option to the PRINTO statement                                              ;
         %put %str(                           )Default: Y                                                                        ;
         %put delete               =  1 results in the deletion of the temporary datasets used or generated by the               ;
         %put %str(                        )macro                                                                               ;
         %put %str(                           )Default: 1                                                                        ;
         %put warn_unbalance_quote =  Whether to include the warning %str(%")WARNING: The quoted string currently being         ;
         %put %str(                        )processed has become more than 262 characters long.  You might have unbalanced      ;
         %put %str(                        )quotation marks.%str(%") in the search.                                             ;
         %put %str(                           )Default: N                                                                        ;
         %put ____________________________________________________________________________________________________________________ ;
         %put End of help                                                                                                        ;

         options &mprint_orig. ;

         %goto __END ;
     %end ;


  %if %nrbquote( &log_filename. ) = %str()
  %then
     %do ;
         %put ER%str(ROR: ) log_filename cannot be missing. ;
         %goto __END ;
     %end ;

  %let __rc_fr = %str() ;

  %if     (     %sysfunc( prxmatch( %str(/^[a-z_][a-z0-9_]{0,7}$/i) , %nrbquote(&log_filename.))) = 0
            and %sysfunc( fileexist( &log_filename. )) =  0
          )
      or  %sysfunc( prxmatch( %str(/^[a-z_][a-z0-9_]{0,7}$/i) , %nrbquote(&log_filename.)))
  %then
     %do ;
         %if %sysfunc( prxmatch( %str(/^[a-z_][a-z0-9_]{0,7}$/i) , %nrbquote(&log_filename.)))
         %then %let __rc_fr = %sysfunc( fileref( &log_filename. )) ;

         %if &__rc_fr ne 0
         %then
            %do ;
                %if %sysfunc( prxmatch( %str(/^[a-z_][a-z0-9_]{0,7}$/i) , %nrbquote(&log_filename.))) = 0
                %then %put ER%str(ROR: ) log_filename = &log_filename. does not  exist ;
                 %else %put ER%str(ROR: ) The FILEREF log_filename = &log_filename. does not  exist ;
                %goto __END ;
            %end ;
     %end ;

  %let nc   = %sysfunc( getoption ( center )) ;
  %let fd   = %sysfunc( getoption ( formdlim )) ;
  %let ls   = %sysfunc( getoption ( ls )) ;

  options nocenter
          formdlim = " "
          ls       = 256
          ;

  title    " " ;
  footnote ;

```

```
 93    %if     %nrbquote(&print_file.) ne %str()
 94        and %sysfunc( prxmatch( /Y|YES/i , &print. ))
 95    %then
 96       %do ;
 97            proc printto
 98               print = "&print_file."
 99               %if &print_append. = N %then New ;
100               ;
101           run ;
102        %end ;
103
104    /* Conditionally provide the last modification data if the log_filename is not a SAS fileref, i.e. not path
105       or file extension (. in .log)
106    */
107    %if %sysfunc( prxmatch( /\/|\\|\./ , %bquote(&log_filename.))) ne 0
108    %then
109       %do ;
110           %if %sysfunc( prxmatch( /Y|YES/i , &print. ))
111           %then
112              %do ;
113                  data _null_ ;
114                  file print ;
115                  put "&log_filename." ;
116                  run ;
117              %end ;
118
119           /** Last modification date */
120           %mac_u_finfo
121              ( in_ds    = %str()
122              , pathfile = &log_filename.
123              , delete   = 0
124              %if %sysfunc( prxmatch( /Y|YES/i , &print. )) = 0 %then , print               = N ;
125              , put_header           = @1   "Path"
126                                       @150 "File"
127                                       @200 "Last Modified"
128              , put_variables        = @1   path
129                                       @150 file
130                                       @200 last_modified
131              )
132
133           %if &delete. = 1
134           %then
135              %do ;
136                  proc datasets
137                     library = WORK
138                     nolist
139                     ;
140                    delete finfo ;
141                  quit ;
142              %end ;
143
144       %end ; /* log_filename is a path/log, not a fileref */
145
146     %else %if %sysfunc( prxmatch( /Y|YES/i , &print. ))
147     %then
148        %do ;
149            data _null_ ;
150              file print ;
151              put "&log_filename." ;
152            run ;
153        %end ;
154
155    data __log_messages
156           ( keep = message
157                    line
158                    log_line
159           )
160          __records
161              ( keep = records )
162          ;
163      length message $ 100 ;
164      infile %if %sysfunc( prxmatch( /\/|\\|\./ , %bquote(&log_filename.))) ne 0
165             %then "&log_filename." ;
166              %else &log_filename. ;
167             length = len
168             end    = end
169             ;
170      input line $varying256. len ;
171
172      records = _n_ ;
173
174      if prxmatch( "/SAS SYSTEM STOPPED PROCESSING THIS STEP/i" , line )
175      then
176        do ;
177           message  = "SAS SYSTEM STOPPED PROCESSING THIS STEP" ;
178           log_line = _n_ ;
179           output __log_messages ;
180           goto __end ;
181        end ;
182
183      if     prxmatch( "/ERROR:/"  , line )
184      then
185        do ;
186           message  = "ERROR:" ;
187           log_line = _n_ ;
```

```
188           output __log_messages ;
189           goto __end ;
190         end ;
191
192       if     prxmatch( "/ERROR \d+-\d+/"  , line )
193       then
194         do ;
195           message  = "ERROR \d+-\d+" ;
196           log_line = _n_ ;
197           output __log_messages ;
198           goto __end ;
199         end ;
200
201       if prxmatch( "/_ERROR_=1/" , line )
202       then
203         do ;
204           message  = "_ERROR_=1" ;
205           log_line = _n_ ;
206           output __log_messages ;
207           goto __end ;
208         end ;
209
210       if     prxmatch( "/WARNING/"  , line )
211          and prxmatch( "/UWARNING/" , line ) = 0
212          and prxmatch( "/WARNING: Engine XPORT does not support SORTEDBY operations.  SORTEDBY information cannot be copied./"
213                      , line ) = 0
214          and prxmatch( "/WARNING: Some character data was lost during transcoding in column/" , line ) = 0
215          /* WARNING: The quoted string currently being processed has become more than 262 characters long.
216             You might have unbalanced quotation marks.
217          */
218          %if &warn_unbalance_quote. = N
219          %then and prxmatch( "/WARNING: The quoted string currently being processed has become more than 262 characters long\./"
220                            , line ) = 0
221          ;
222       then
223         do ;
224           message  = "WARNING" ;
225           log_line = _n_ ;
226           output __log_messages ;
227           goto __end ;
228         end ;
229
230       if prxmatch( "/ABNORMALLY TERMINATED/i" , line )
231       then
232         do ;
233           message  = "ABNORMALLY TERMINATED" ;
234           log_line = _n_ ;
235           output __log_messages ;
236           goto __end ;
237         end ;
238
239       if prxmatch( "/ALREADY EXISTS/i" , line )
240       then
241         do ;
242           message  = "ALREADY EXISTS" ;
243           log_line = _n_ ;
244           output __log_messages ;
245           goto __end ;
246         end ;
247
248       if prxmatch( "/ARGUMENT TO FUNCTION/i" , line )
249       then
250         do ;
251           message  = "ARGUMENT TO FUNCTION" ;
252           log_line = _n_ ;
253           output __log_messages ;
254           goto __end ;
255         end ;
256
257       if prxmatch( "/COULD NOT BE WRITTEN/i" , line )
258       then
259         do ;
260           message  = "COULD NOT BE WRITTEN" ;
261           log_line = _n_ ;
262           output __log_messages ;
263           goto __end ;
264         end ;
265
266       if prxmatch( "/DIVISION BY ZERO DETECTED/i" , line )
267       then
268         do ;
269           message  = "DIVISION BY ZERO DETECTED" ;
270           log_line = _n_ ;
271           output __log_messages ;
272           goto __end ;
273         end ;
274
275       if     prxmatch( "/DOES NOT EXIST/i" , line )
276          and prxmatch( "/NOTE: BASE data set does not exist\. DATA file is being copied to BASE file\./" , line ) = 0
277       then
278         do ;
279           message  = "DOES NOT EXIST" ;
280           log_line = _n_ ;
281           output __log_messages ;
282           goto __end ;
```

```
283       end ;
284
285     if prxmatch( "/ENDSAS/i" , line )
286     then
287       do ;
288          message  = "ENDSAS" ;
289          log_line = _n_ ;
290          output __log_messages ;
291          goto __end ;
292       end ;
293
294     if prxmatch( "/EXPERIMENTAL IN RELEASE/i" , line )
295     then
296       do ;
297          message  = "EXPERIMENTAL IN RELEASE" ;
298          log_line = _n_ ;
299          output __log_messages ;
300          goto __end ;
301       end ;
302
303     if prxmatch( "/FORMAT WAS TOO SMALL FOR THE NUMBER TO BE PRINTED/i" , line )
304     then
305       do ;
306          message  = "FORMAT WAS TOO SMALL FOR THE NUMBER TO BE PRINTED" ;
307          log_line = _n_ ;
308          output __log_messages ;
309          goto __end ;
310       end ;
311
312     if prxmatch( "/INVALID ARGUMENT/i" , line )
313     then
314       do ;
315          message  = "INVALID ARGUMENT" ;
316          log_line = _n_ ;
317          output __log_messages ;
318          goto __end ;
319       end ;
320
321     if prxmatch( "/INVALID NUMERIC DATA/i" , line )
322     then
323       do ;
324          message  = "INVALID NUMERIC DATA" ;
325          log_line = _n_ ;
326          output __log_messages ;
327          goto __end ;
328       end ;
329
330     if prxmatch( "/LOST CARD/i" , line )
331     then
332       do ;
333          message  = "LOST CARD" ;
334          log_line = _n_ ;
335          output __log_messages ;
336          goto __end ;
337       end ;
338
339     if prxmatch( "/MATHEMATICAL OPERATIONS COULD NOT BE PERFORMED/i" , line )
340     then
341       do ;
342          message  = "MATHEMATICAL OPERATIONS COULD NOT BE PERFORMED" ;
343          log_line = _n_ ;
344          output __log_messages ;
345          goto __end ;
346       end ;
347
348     if prxmatch( "/MISSING VALUES WERE GENERATED/i" , line )
349     then
350       do ;
351          message  = "MISSING VALUES WERE GENERATED" ;
352          log_line = _n_ ;
353          output __log_messages ;
354          goto __end ;
355       end ;
356
357     if prxmatch( "/MORE THAN ONE DATA SET WITH REPEATS OF BY VALUES/i" , line )
358     then
359       do ;
360          message  = "MORE THAN ONE DATA SET WITH REPEATS OF BY VALUES" ;
361          log_line = _n_ ;
362          output __log_messages ;
363          goto __end ;
364       end ;
365
366     if     prxmatch( "/NOT FOUND/i" , line )
367        and prxmatch( "/Pinnacle Finding/" , line ) = 0
368     then
369       do ;
370          message  = "NOT FOUND" ;
371          log_line = _n_ ;
372          output __log_messages ;
373          goto __end ;
374       end ;
375
376     if prxmatch( "/NOT PREVIOUSLY/i" , line )
377     then
```

```
      do ;
         message  = "NOT PREVIOUSLY" ;
         log_line = _n_ ;
         output __log_messages ;
         goto __end ;
      end ;

   if prxmatch( "/NOTE: FORMATTED VALUES OF/i" , line )
   then
      do ;
         message  = "NOTE: FORMATTED VALUES OF" ;
         log_line = _n_ ;
         output __log_messages ;
         goto __end ;
      end ;

   if prxmatch( "/ONE OR MORE LINES WERE TRUNCATED/i" , line )
   then
      do ;
         message  = "ONE OR MORE LINES WERE TRUNCATED" ;
         log_line = _n_ ;
         output __log_messages ;
         goto __end ;
      end ;

   if prxmatch( "/OUTSIDE THE AXIS RANGE/i" , line )
   then
      do ;
         message  = "OUTSIDE THE AXIS RANGE" ;
         log_line = _n_ ;
         output __log_messages ;
         goto __end ;
      end ;

   if prxmatch( "/SAS WENT TO A NEW LINE/i" , line )
   then
      do ;
         message  = "SAS WENT TO A NEW LINE" ;
         log_line = _n_ ;
         output __log_messages ;
         goto __end ;
      end ;

   if prxmatch( "/SEGMENTATION VIOLATION/i" , line )
   then
      do ;
         message  = "SEGMENTATION VIOLATION" ;
         log_line = _n_ ;
         output __log_messages ;
         goto __end ;
      end ;

   if prxmatch( "/THE MEANING OF AN IDENTIFIER AFTER A QUOTED STRING MAY CHANGE/i" , line )
   then
      do ;
         message  = "THE MEANING OF AN IDENTIFIER AFTER A QUOTED STRING MAY CHANGE" ;
         log_line = _n_ ;
         output __log_messages ;
         goto __end ;
      end ;

   if prxmatch( "/UERROR/i" , line )
   then
      do ;
         message  = "UERROR" ;
         log_line = _n_ ;
         output __log_messages ;
         goto __end ;
      end ;

   if prxmatch( "/UNINITIALIZED/i" , line )
   then
      do ;
         message  = "UNINITIALIZED" ;
         log_line = _n_ ;
         output __log_messages ;
         goto __end ;
      end ;

   if prxmatch( "/UWARNING/i" , line )
   then
      do ;
         message  = "UWARNING" ;
         log_line = _n_ ;
         output __log_messages ;
         goto __end ;
      end ;

   if prxmatch( "/VALUES HAVE BEEN CONVERTED TO/i" , line )
   then
      do ;
         message  = "VALUES HAVE BEEN CONVERTED TO" ;
         log_line = _n_ ;
         output __log_messages ;
         goto __end ;
```

```
473         end ;
474
475       if prxmatch( "/ILLEGAL/i" , line )
476       then
477         do ;
478            message  = "ILLEGAL" ;
479            log_line = _n_ ;
480            output __log_messages ;
481            goto __end ;
482         end ;
483
484       if prxmatch( '/SHIFTED BY THE "BEST" FORMAT/i' , line )
485       then
486         do ;
487            message  = 'SHIFTED BY THE "BEST" FORMAT' ;
488            log_line = _n_ ;
489            output __log_messages ;
490            goto __end ;
491         end ;
492
493       if prxmatch( '/NOTE: The variable label/i' , line )
494       then
495         do ;
496            message  = 'NOTE: The variable label' ;
497            log_line = _n_ ;
498            output __log_messages ;
499            goto __end ;
500         end ;
501
502       if prxmatch( '/Pinnacle Finding:/' , line )
503       then
504         do ;
505            message  = 'PINNACLE FINDING:' ;
506            log_line = _n_ ;
507            output __log_messages ;
508            goto __end ;
509         end ;
510
511       __end:
512
513       if end then output __records ;
514
515     run ;
516
517     %if &print_all. = Y
518     %then
519        %do ;
520            data __messages ;
521              length message $ 100 ;
522
523              message  = "WARNING" ;
524              output ;
525
526              message  = "ABNORMALLY TERMINATED" ;
527              output ;
528
529              message  = "ALREADY EXISTS" ;
530              output ;
531
532              message  = "ARGUMENT TO FUNCTION" ;
533              output ;
534
535              message  = "COULD NOT BE WRITTEN" ;
536              output ;
537
538              message  = "DIVISION BY ZERO DETECTED" ;
539              output ;
540
541              message  = "DOES NOT EXIST" ;
542              output ;
543
544              message  = "ENDSAS" ;
545              output ;
546
547              message  = "ERROR:" ;
548              output ;
549
550              message  = "ERROR \d+-\d+" ;
551              output ;
552
553              message  = "EXPERIMENTAL IN RELEASE" ;
554              output ;
555
556              message  = "FORMAT WAS TOO SMALL FOR THE NUMBER TO BE PRINTED" ;
557              output ;
558
559              message  = "INVALID ARGUMENT" ;
560              output ;
561
562              message  = "INVALID NUMERIC DATA" ;
563              output ;
564
565              message  = "LOST CARD" ;
566              output ;
567
```

```
568             message  = "MATHEMATICAL OPERATIONS COULD NOT BE PERFORMED" ;
569             output ;
570
571             message  = "MISSING VALUES WERE GENERATED" ;
572             output ;
573
574             message  = "MORE THAN ONE DATA SET WITH REPEATS OF BY VALUES" ;
575             output ;
576
577             message  = "NOT FOUND" ;
578             output ;
579
580             message  = "NOT PREVIOUSLY" ;
581             output ;
582
583             message  = "NOTE: FORMATTED VALUES OF" ;
584             output ;
585
586             message  = "ONE OR MORE LINES WERE TRUNCATED" ;
587             output ;
588
589             message  = "OUTSIDE THE AXIS RANGE" ;
590             output ;
591
592             message  = "SAS SYSTEM STOPPED PROCESSING THIS STEP" ;
593             output ;
594
595             message  = "SAS WENT TO A NEW LINE" ;
596             output ;
597
598             message  = "SEGMENTATION VIOLATION" ;
599             output ;
600
601             message  = 'SHIFTED BY THE "BEST" FORMAT' ;
602             output ;
603
604             message  = "THE MEANING OF AN IDENTIFIER AFTER A QUOTED STRING MAY CHANGE" ;
605             output ;
606
607             message  = "UERROR" ;
608             output ;
609
610             message  = "UNINITIALIZED" ;
611             output ;
612
613             message  = "UWARNING" ;
614             output ;
615
616             message  = "VALUES HAVE BEEN CONVERTED TO" ;
617             output ;
618
619             message  = "_ERROR_=1" ;
620             output ;
621
622             message  = "ILLEGAL" ;
623             output ;
624
625             message  = 'NOTE: The variable label' ;
626             output ;
627
628             message  = 'PINNACLE FINDING:' ;
629             output ;
630
631           run ;
632
633       %end ;
634
635     %if %sysfunc( prxmatch( /Y|YES/i , &report. ))
636     %then
637       %do ;
638           proc sql ;
639             create table __report as
640             %if &print_all. = Y
641             %then
642                select a.message
643                     , b.log_line
644                     , b.line
645                     , case when b.log_line = . then 0
646                            else 1
647                       end as found
648                from __messages              as a
649                     left join __log_messages as b
650                       on a.message = b.message
651                order by a.message
652                       , b.log_line
653             ; /* end of %if */
654               %else
655                 select message
656                      , log_line
657                      , line
658                      , 1 as found
659                 from __log_messages
660                 order by message
661                        , log_line
662               ; /* end of %else */
```

```
663             ; /* end of create */
664
665              select message
666                   , sum( found ) as frequency
667              from __report
668              group by message
669              ;
670           quit ;
671
672           data __report ;
673             set __report
674                  ( where = ( found = 1 ))
675                 ;
676           run ;
677
678           proc sort
679              data = __report ;
680             by log_line
681                message
682                ;
683           run ;
684
685       %end ;
686
687     %if %sysfunc( prxmatch( /Y|YES/i , &print. ))
688     %then
689        %do ;
690           data _null_ ;
691             file print ;
692             set __records ;
693             put "Records read from the log: "
694                 records comma15.
695                 ;
696           run ;
697
698           %if %sysfunc( prxmatch( /Y|YES/i , &report. ))
699           %then
700              %do ;
701                 data _null_ ;
702                   file print ;
703                   if nobs = 0 then put "***** No messages found  *****" ;
704                   set __report
705                        nobs = nobs
706                        ;
707                   put "Line = "
708                       log_line
709                       ;
710                    put line $160. / ;
711                 run ;
712              %end ;
713       %end ;
714
715     options &nc.
716             formdlim = "&fd."
717             ls       = &ls.
718             ;
719
720     %if &delete. = 1
721     %then
722        %do ;
723           proc datasets
724              library = WORK
725              nolist
726              memtype = data
727              ;
728             delete __log_messages
729                    __records
730                    %if &print_all. = Y %then __messages ;
731                    %if %sysfunc( prxmatch( /Y|YES/i , &report. )) %then __report ;
732                      ;
733           quit ;
734       %end ;
735
736     %if     %nrbquote(&print_file.) ne %str()
737         and %sysfunc( prxmatch( /Y|YES/i , &print. ))
738     %then
739        %do ;
740             proc printto ;
741             run ;
742         %end ;
743
744     %__END:
745
746   %mend mac_u_log_check ;
```

Appendix 5. The MAC_U_PATH_LOG_CHECK macro.

```
%macro mac_u_path_log_check
   ( in_ds                = %str()
   , in_ds_keep           = log
   , in_ds_rename         = log = pathfile
   , path                 = %str()
   , path_delim           = #
   , print_messages_all   = N
   , print_messages_gt_0  = N
   , delete               = Y
   , help                 = N
   ) ;

  %if &help. = Y
  %then
     %do ;
         %let mprint_orig = %sysfunc( getoption( mprint )) ;
         options nomprint ;

         skip ;
         skip ;
         %put ____________________________________________________________________________________________________________________ ;
         %put Purpose of program:     This macro performs a log check on the .log files in the directories provided by the user. ;
         %put                                                                                                                       ;
         %put Macro Parameter         Description                                                                                   ;
         %put ____________________    ____________________________________________________________________________________________ ;
         %put in_ds                 = Data set contain path and file or pathfile.                                                 ;
         %put %str(                          )Default: %nrstr(%%)str%str(())                                                     ;
         %put path                  = The paths to the log files delimited by the value in PATH_DELIM.                            ;
         %put %str(                          )Default: %nrstr(%%)str%str(())                                                     ;
         %put path_delim            = The delimiter of the paths in PATH.                                                         ;
         %put %str(                          )Default: #                                                                         ;
         %put print_messages_all    = Whether to print each LINE for the log check for each log file.                             ;
         %put %str(                          )Default: N                                                                         ;
         %put print_messages_gt_0   = Whether to print each LINE for the log check for each log file that has at least one       ;
         %put %str(                        )message.                                                                             ;
         %put %str(                          )Default: N                                                                         ;
         %put delete                = Whether to delete the data set holding the lists of log files.                              ;
         %put %str(                          )Default: Y                                                                         ;
         %put ____________________________________________________________________________________________________________________ ;
         %put End of help                                                                                                           ;
         skip ;

         options &mprint_orig. ;

         %goto __END ;
     %end ;

  %if     &in_ds.           = %str()
      and %nrbquote(&path.) = %str()
  %then
     %do ;
         %put W%str(ARNING: ) IN_DS and PATH cannot both be missing (null) ;
         %goto __END ;
     %end ;

  %if     &in_ds.           ne %str()
      and %nrbquote(&path.) ne %str()
  %then
     %do ;
         %put W%str(ARNING: ) IN_DS and PATH both are non-missing.  Please provide a value for only one. ;
         %goto __END ;
     %end ;

  %if %nrbquote(&path.) ne %str()
  %then %let __ds = dir_read ;
   %else %let __ds = &in_ds. ;

  /***/
  %let byline_orig = %sysfunc( getoption( byline )) ;
  options nobyline ;

  %if %nrbquote(&path.) ne %str()
  %then
     %do ;
         %mac_u_directory_read
            ( path                 = &path.
            , path_delim           = &path_delim.
            , recursive            = N
            , output_dir           = N
            , output_files         = Y
            ) ;
     %end ;

  %mac_u_delete
     ( ds = log_check_all
            log_messages_all
     ) ;

  proc sql
     noprint ;
    select pathfile
              into : pf1 -
```

```
 93      from &__ds.
 94           %if     &in_ds. ne %str()
 95              and %nrbquote(&in_ds_rename.) ne %str()
 96           %then
 97              %do ;
 98                  ( %if %nrbquote(&in_ds_keep.) ne %str() %then keep   = log  ;
 99                    rename = ( &in_ds_rename. )
100                  )
101              %end ;
102        where prxmatch( "/\.log$/i" , trim( pathfile ))
103        ;
104     quit ;
105
106     %if %nrbquote(&path.) ne %str()
107     %then
108        %do ;
109           %mac_u_delete
110              ( ds = &__ds. ) ;
111        %end ;
112
113     %do __i = 1 %to &sqlobs. ;
114
115        %mac_u_log_check
116           ( log_filename         = &&pf&__i.
117           , report               = N
118           , print                = N
119           , print_file           = %str()
120           , delete               = 0
121           ) ;
122
123        proc sql
124           noprint ;
125          select count( * )
126                   into : messages
127          from __log_messages
128          ;
129        quit ;
130
131        proc sql
132           noprint ;
133          select count( * ) > 0
134                   into : errs
135          from __log_messages
136          where     prxmatch( "/ERROR:/"         , line )
137                or  prxmatch( "/ERROR \d+-\d+/"  , line )
138                or  prxmatch( "/_ERROR_=1/"      , line )
139                or  prxmatch( "/UERROR/i"        , line )
140          ;
141        quit ;
142
143        proc sql
144           noprint ;
145          select count( * ) > 0
146                   into : warns
147          from __log_messages
148          where     prxmatch( "/WARNING/"  , line )
149                and prxmatch( "/UWARNING/" , line ) = 0
150                and prxmatch( "/WARNING: Engine XPORT does not support SORTEDBY operations.  SORTEDBY information cannot be copied./"
151                            , line ) = 0
152                and prxmatch( "/WARNING: Some character data was lost during transcoding in column/" , line ) = 0
153                and prxmatch( "/WARNING: The quoted string currently being processed has become more than 262 characters long\./"
154                            , line ) = 0
155          ;
156        quit ;
157
158        proc sql
159           noprint ;
160          select count( * ) > 0
161                   into : stops
162          from __log_messages
163          where     prxmatch( "/ABNORMALLY TERMINATED/i"                  , line )
164                or  prxmatch( "/ENDSAS/i"                                 , line )
165                or  prxmatch( "/SAS SYSTEM STOPPED PROCESSING THIS STEP/i" , line )
166                or  prxmatch( "/SEGMENTATION VIOLATION/i"                 , line )
167          ;
168        quit ;
169
170        /*************/
171        data __log_check ;
172
173          merge finfo
174                __records
175                ;
176
177          messages = &messages. ;
178          log_has_message = messages > 0 ;
179
180          log_has_warning = &warns. ;
181          log_has_error   = &errs.  ;
182          log_has_stops   = &stops. ;
183
184        run ;
185
186        %mac_u_delete
187           ( ds = __records
```

```
188                 __messages
189         ) ;
190
191      %if &messages. = 0
192      %then
193         %do ;
194             data __log_messages ;
195               if 0 then set __log_messages ;
196               message = "No messages found" ;
197               line    = "No messages found" ;
198               output ;
199               stop ;
200             run ;
201         %end ;
202
203      proc sql
204         undo_policy = none ;
205        create table __log_messages as
206        select a.path
207             , a.file
208             , b.*
209        from finfo          as a
210           , __log_messages as b
211        order by a.path
212               , a.file
213               , b.message
214               , b.log_line
215        ;
216      quit ;
217
218      %mac_u_delete
219         ( ds = finfo )
220
221      /**********/
222      %if %sysfunc( exist( log_messages_all ))
223      %then
224         %do ;
225             proc append
226                base = log_messages_all
227                data = __log_messages
228                ;
229             run ;
230
231             proc datasets
232                library = WORK
233                nolist
234                ;
235               delete __log_messages ;
236             quit ;
237         %end ;
238        %else
239           %do ;
240               proc datasets
241                  library = WORK
242                  nolist
243                  ;
244                 change __log_messages = log_messages_all ;
245               quit ;
246           %end ;
247
248      /****/
249      %if %sysfunc( exist( log_check_all ))
250      %then
251         %do ;
252             proc append
253                base = log_check_all
254                data = __log_check
255                ;
256             run ;
257
258             proc datasets
259                library = WORK
260                nolist
261                ;
262               delete __log_check ;
263             quit ;
264         %end ;
265        %else
266           %do ;
267               proc datasets
268                  library = WORK
269                  nolist
270                  ;
271                 change __log_check = log_check_all ;
272               quit ;
273           %end ;
274
275
276   %end ; /* CYCLED THROUGH __i */
277
278   proc sort
279      data = log_check_all ;
280     by path
281        file
282        ;
```

```
283    run ;
284
285    proc sort
286       data = log_messages_all ;
287      by path
288         file
289         log_line
290         ;
291    run ;
292
293    /**********/
294    footnote ;
295
296    title "path = #byval1" ;
297
298    proc print
299       data = log_check_all
300       noobs
301       sumlabel         = "Total in path:"
302       grandtotal_label = "Grand total in all paths: "
303       n = "Logs in path: "
304           "Total Logs in all paths: "
305       ;
306      by path ;
307      sum messages
308          log_has_message
309          log_has_warning
310          log_has_error
311          log_has_stops
312          ;
313    run ;
314
315    title " " ;
316
317    %if &print_messages_all. = Y
318    %then
319       %do ;
320
321           title1 "path = #byval1" ;
322           title2 "file = #byval2" ;
323
324           proc print
325              data = log_messages_all
326              noobs
327              ;
328             by path
329                file
330                ;
331             var log_line
332                 line
333                 ;
334             format line $240. ;
335           run ;
336
337           title " " ;
338
339       %end ;
340
341    %if     &print_messages_gt_0. = Y
342        and &print_messages_all.  = N
343    %then
344       %do ;
345           proc sql
346              noprint ;
347             select distinct quote( strip( catx( "/"
348                                                , path
349                                                , file
350                                                )
351                                         )
352                                  )
353                     into : list separated by " , "
354             from log_check_all
355             where messages > 0
356             ;
357           quit ;
358
359           title1 "path = #byval1" ;
360           title2 "file = #byval2" ;
361
362           proc print
363              data = log_messages_all
364                        ( where  = ( catx( "/"
365                                         , path
366                                         , file
367                                         ) in
368                                        ( &list. )
369                                   )
370                        )
371               noobs
372               ;
373             by path
374                file
375                ;
376             var log_line
377                 line
```

```
                ;
              format line $240. ;
          run ;

          title " " ;

       %end ;

   options &byline_orig. ;

   %if %sysfunc( upcase( &delete. )) = Y
   %then
      %do ;
          %mac_u_delete
             ( ds = &__ds. )
       %end ;

   %__END:

%mend  mac_u_path_log_check ;

```

Appendix 6. The MAC_R_COMPARE macro.

```
%macro mac_r_compare_report
   ( path         = %str()
   , file         = *.lst
   , filename     = %str()
   , out          = __compare
   , print        = Y
   , print_vars   = file
                    data
                    comp
                    data_last_modified
                    comp_last_modified
                    datetime_issue
                    data_vars
                    vars_in_common
                    vars_issue
                    data_obs
                    obs_in_common
                    obs_issue
                    obs_w_unequal
                    obs_all_equal
   , print_where = %str()
   , print_sum   = obs_w_unequal
   , mv_fail     = compare_fail
   , help         = N
   ) ;

  %if &help. = Y
  %then
     %do ;
         %put ____________________________________________________________________________________________________ ;
         %put Purpose of program:        This report macro parses .lst files extracting pertinent COMPARE data.  ;
         skip ;
         %put The following shows what data are extracted with their SAS variable names in parentheses:          ;
         skip ;
         %put The COMPARE Procedure                                                                               ;
         skip ;
         %put %str(             )(DATA)                 (COMP)                                                    ;
         %put Comparison of VALDS.T14_1_9_62_1 with WORK.V_T14_1_9_62_1                                           ;
         %put (Method=EXACT)                                                                                      ;
         skip ;
         %put Data Set Summary                                                                                    ;
         skip ;
         %put Dataset                      Created          Modified  NVar    NObs                              ;
         skip ;
         %put %str(                                        )(DATA_LM)     (DATA_VARS)   (DATA_OBS)             ;
         %put VALDS.T14_1_9_62_1   06OCT16:12:24:34  06OCT16:12:24:34    14      72                            ;
         %put WORK.V_T14_1_9_62_1  28OCT16:13:39:02  28OCT16:13:39:02    14      72                            ;
         %put %str(                                         )(COMP_LM)     (COMP_VARS)   (COMP_OBS)           ;
         skip ;
         %put Variables Summary                                                                                   ;
         skip ;
         %put Number of Variables in Common: 14. (VARS_IN_COMMON)                                                 ;
         %put Number of ID Variables: 6.                                                                          ;
         skip ;
         %put Observation Summary                                                                                 ;
         skip ;
         %put Observation      Base  Compare  ID                                                                  ;
         skip ;
         %put First Obs           1        1  PARAM=Alanine Aminotransferase (U/L) ORD0=1 TRTN=1 AVISITN=2 DAY=Baseline ;
         %put Last  Obs          72       72  PARAM=Urea Nitrogen (mmol/L) ORD0=4 TRTN=4 AVISITN=4 DAY=Day 15    ;
         skip ;
         %put Number of Observations in Common: 72.                      (OBS_IN_COMMON)                          ;
         %put Total Number of Observations Read from VALDS.T14-1-9: 72.  (OBS_DATA)                               ;
         %put Total Number of Observations Read from WORK.V_T14-1-9: 72. (OBS_COMP)                               ;
         skip ;
         %put Number of Observations with Some Compared Variables Unequal: 0. (OBS_W_UNEQUAL)                     ;
         %put Number of Observations with All Compared Variables Equal: 72.   (OBS_ALL_EQUAL)                     ;
         skip ;
         %put NOTE: No unequal values were found. All values compared are exactly equal.                          ;
         skip ;
         %put Macro Parameter           Description                                                             ;
         %put _________________         ____________________________________________________________________    ;
         %put path                    = The directory of interest.                                                ;
         %put %str(                             )Default: %nrstr(%%)str()                                       ;
         %put file                    = The file of interest.  %str(*).lst reads every .lst file in the directory.;
         %put %str(                             )Default: %str(*).lst                                           ;
         %put filename                = A FILEREF of interest.                                                    ;
         %put %str(                             )Default: %nrstr(%%)str()                                       ;
         %put out                     = The name of a SAS output data set.                                        ;
         %put %str(                             )__compare                                                      ;
         %put print                   = Whether to send the results to the output window (interactive mode).      ;
         %put %str(                             )Y                                                              ;
         %put print_vars              = The variables to include in the output                                    ;
         %put %str(                             )Default: file                                                  ;
         %put %str(                                      )data                                                  ;
         %put %str(                                      )comp                                                  ;
         %put %str(                                      )data_last_modified                                    ;
         %put %str(                                      )comp_last_modified                                    ;
         %put %str(                                      )datetime_issue                                        ;
         %put %str(                                      )data_vars                                             ;
         %put %str(                                      )vars_in_common                                        ;
         %put %str(                                      )vars_issue                                            ;
```

```
          %put %str(                                      )data_obs                                                   ;
          %put %str(                                      )obs_in_common                                              ;
          %put %str(                                      )obs_issue                                                  ;
          %put %str(                                      )obs_w_unequal                                              ;
          %put %str(                                      )obs_all_equal                                              ;
          %put print_where             = The clause to the data set option WHERE= to the PRINT statement.             ;
          %put %str(                             )Default: %nrstr(%%)str()                                            ;
          %put print_sum               = The variables for the SUM statement of the PRINT procedure.                  ;
          %put %str(                             )Default: obs_w_unequal                                              ;
          %put mv_fail                 = The name of the GLOBAL marco variable that indicates if any of criteria failed. ;
          %put %str(                                  )data_vars     ne vars_in_common                                ;
          %put %str(                             )or  comp_vars     ne vars_in_common                                 ;
          %put %str(                             )or  data_obs      ne obs_in_common                                  ;
          %put %str(                             )or  comp_obs      ne obs_in_common                                  ;
          %put %str(                             )or  obs_w_unequal >  0                                              ;
          %put %str(                               )Default: compare_fail                                             ;
          skip ;
          %put ______________________________________________________________________________________________________ ;
          %put End of help                                                                                            ;

          %goto __END ;
      %end ;

   %if %nrbquote(&path.) ne %str()
   %then %let path = %sysfunc( prxchange( s/(^*.)\\$/$1/ , 1 , %nrbquote(&path.))) ;

   %if     %nrbquote(&path.)                   ne %str()
       and %nrbquote(&file.)                   ne %str()
   %then
      %do ;
          %if %sysfunc( fileexist(&path.\&file.)) =  0
          %then
             %do ;
                 %goto __END ;
             %end ;
           %else %let filename = "&path.\&file." ;
      %end ;
    %else
       %do ;
           %if     %nrbquote(&filename.)               ne %str()
               and %sysfunc( fileref( &filename. ))    > 0
           %then
              %do ;
                  %goto __END ;
              %end ;
       %end ;

   %if %sysfunc( exist( &out. ))
   %then
      %do ;
          proc datasets
             library = %if %sysfunc( index( &out. , . )) %then %scan( &out. , 1 , . ) ;
                        %else WORK ;
             nolist
             ;
            delete %if %sysfunc( index( &out. , . )) %then %scan( &out. , 2 , . ) ;
                        %else &out. ;
                  ;
            ;
          quit ;
      %end ;

   %global &mv_fail. ;


   /***********/
   data &out.
           ( drop   = __: )
        ;

     if _n_ = 1
     then
       do ;
          __rc1 = prxparse( "/Comparison of ([_A-Z][_A-Z0-9]{0,7}\.[_A-Z][_A-Z0-9]{0,31}) with ([_A-Z][_A-Z0-9]{0,7}\.[_A-Z][_A-Z0-
9]{0,31})/i" ) ;
          __rc2 = prxparse( "/([_A-Z][_A-Z0-9]{0,7}\.[_A-Z][_A-Z0-9]{0,31})\s+\d{2}[A-Z]{3}\d{2}:\d{2}:\d{2}:\d{2}\s+(\d{2}[A-
Z]{3}\d{2}:\d{2}:\d{2}:\d{2})\s+(\d+)\s+(\d+)/i" ) ;
          __rc3 = prxparse( "/Number of Variables in Common: (\d+)\./" ) ;
          __rc4 = prxparse( "/Number of Observations in Common: (\d+)\./" ) ;
          __rc7 = prxparse( "/Number of Observations with Some Compared Variables Unequal: (\d+)\./" ) ;
          __rc8 = prxparse( "/Number of Observations with All Compared Variables Equal: (\d+)\./" ) ;
       end ;

     length file
            pathfile        $ 200
            data
            comp
            __data
            __comp          $  41
            data_last_modified
            data_vars
            data_obs
            comp_last_modified
            comp_vars
            comp_obs
```

```
188             vars_in_common
189             obs_in_common
190             obs_data
191             obs_comp
192             obs_w_unequal
193             obs_all_equal   8
194             __fn
195             __fn1           $ 200
196             ;
197
198         retain __flag1 " "
199                __flag2 "1"
200                __flag3 " "
201                __rc1
202                __rc2
203                __rc3
204                __rc4
205                __rc5
206                __rc6
207                __rc7
208                __rc8  .
209                data
210                comp
211                __data
212                __comp
213                data_last_modified
214                data_vars
215                data_obs
216                comp_last_modified
217                comp_vars
218                comp_obs
219                vars_in_common
220                obs_in_common
221                obs_data
222                obs_comp
223                obs_w_unequal
224                obs_all_equal
225                __fn1
226                ;
227
228         infile &filename.
229                length   = len
230                filename = __fn
231                end      = __end
232                ;
233
234         if __fn1 = " " then __fn1 = __fn ;
235          else if __fn1 ne __fn
236          then
237            do ;
238               if __flag3 = " "
239               then
240                 do ;
241                    pathfile = __fn1 ;
242                    file = scan( pathfile , -1 , "\" ) ;
243                    output ;
244                 end ;
245                else __flag3 = " " ;
246               __fn1 = __fn ;
247            end ;
248
249         input __line $varying256. len ;
250
251         if prxmatch( "/The COMPARE Procedure/" , __line ) then __flag1 = "1" ;
252
253         if __flag1 = "1"
254         then
255           do ;
256              if prxmatch( __rc1 , __line )
257              then
258                do ;
259                   data = prxposn( __rc1 , 1 , __line ) ;
260                   comp = prxposn( __rc1 , 2 , __line ) ;
261
262                  __rc5 = prxparse( cat( "/Total Number of Observations Read from "
263                                       , strip( data )
264                                       , ": (\d+)\./"
265                                       )
266                                  ) ;
267                  __rc6 = prxparse( cat( "/Total Number of Observations Read from "
268                                       , strip( comp )
269                                       , ": (\d+)\./"
270                                       )
271                                  ) ;
272                end ;
273              if prxmatch( __rc2 , __line )
274              then
275                do ;
276                   __data             = prxposn( __rc2 , 1 , __line ) ;
277                   data_last_modified = input( prxposn( __rc2 , 2 , __line ) , datetime. ) ;
278                   data_vars          = input( prxposn( __rc2 , 3 , __line ) , 8. ) ;
279                   data_obs           = input( prxposn( __rc2 , 4 , __line ) , 8. ) ;
280
281                   if data ne __data then put "ER" "ROR: data mismatch: " data= __data= ;
282
```

```
283              input __line $varying256. len ;
284              if prxmatch( __rc2 , __line )
285              then
286                do ;
287                   __comp             = prxposn( __rc2 , 1 , __line ) ;
288                   comp_last_modified = input( prxposn( __rc2 , 2 , __line ) , datetime. ) ;
289                   comp_vars          = input( prxposn( __rc2 , 3 , __line ) , 8. ) ;
290                   comp_obs           = input( prxposn( __rc2 , 4 , __line ) , 8. ) ;
291
292                   if comp ne __comp then put "ER" "ROR: comp mismatch: " comp= __comp= ;
293                end ;
294            end ;
295
296          if prxmatch( __rc3 , __line ) then vars_in_common = input( prxposn( __rc3 , 1 , __line ) , 8. ) ;
297          if prxmatch( __rc4 , __line ) then obs_in_common  = input( prxposn( __rc4 , 1 , __line ) , 8. ) ;
298          if      __rc5 ne .
299             and prxmatch( __rc5 , __line )
300          then obs_data       = input( prxposn( __rc5 , 1 , __line ) , 8. ) ;
301          if      __rc6 ne .
302             and prxmatch( __rc6 , __line )
303          then obs_comp       = input( prxposn( __rc6 , 1 , __line ) , 8. ) ;
304          if prxmatch( __rc7 , __line ) then obs_w_unequal  = input( prxposn( __rc7 , 1 , __line ) , 8. ) ;
305          if     prxmatch( __rc8 , __line )
306             or  prxmatch( "/Comparisons of data values not performed./o" , __line )
307          then
308            do ;
309               if prxmatch( __rc8 , __line ) then obs_all_equal  = input( prxposn( __rc8 , 1 , __line ) , 8. ) ;
310               pathfile = __fn ;
311               file     = scan( pathfile , -1 , "\" ) ;
312
313               if     data_last_modified ne .
314                  and comp_last_modified ne .
315                  and comp_last_modified < data_last_modified
316               then datetime_issue = "Y" ;
317
318               if     data_vars      ne .
319                  and vars_in_common ne .
320                  and data_vars      ne vars_in_common
321               then vars_issue = "Y" ;
322
323               if     data_obs      ne .
324                  and obs_in_common ne .
325                  and data_obs      ne obs_in_common
326               then obs_issue = "Y" ;
327
328               output ;
329
330               call missing( file
331                           , data
332                           , comp
333                           , __data
334                           , data_last_modified
335                           , data_vars
336                           , data_obs
337                           , __comp
338                           , comp_last_modified
339                           , comp_vars
340                           , comp_obs
341                           , vars_in_common
342                           , obs_in_common
343                           , obs_data
344                           , obs_comp
345                           , obs_w_unequal
346                           , obs_all_equal
347                           , datetime_issue
348                           , __flag2
349                           ) ;
350               __flag3 = "1" ;
351            end ;
352
353        end ; /* END OF __flag1 = "1" */
354
355      if      __end   = 1
356         and __flag3 = " "
357      then
358        do ;
359           pathfile = __fn ;
360           file     = scan( pathfile , -1 , "\" ) ;
361           output ;
362        end ;
363
364      format data_last_modified
365             comp_last_modified datetime20.
366             ;
367
368    run ;
369
370    proc sort
371       data = &out. ;
372      by file
373         data
374         ;
375    run ;
376
377    %if &print. = Y
```

```
378    %then
379       %do ;
380           %if %nrbquote(&path.) ne %str() %then title "&path." %str(;) ;
381            %else title " " %str(;) ;
382           proc print
383              data = &out.
384                        %if %nrbquote(&print_where.) ne %str() %then ( where  = ( &print_where. )) ;
385              ;
386             %if &print_vars. ne %str() %then var &print_vars. %str(;) ;
387             %if &print_sum.  ne %str() %then sum &print_sum. %str(;) ;
388           run ;
389
390           %if %nrbquote(&path.) ne %str() %then title1 "&path." %str(;) ;
391            %else title1 " " %str(;) ;
392           title2 "COMPARE results with matching variable and observations numbers and no unequal values" ;
393           proc sql ;
394             select count( * )
395             from &out.
396             where     data ne ""
397                   and data_vars     = vars_in_common
398                   and data_obs      = obs_in_common
399                   and obs_w_unequal = 0
400             ;
401           quit ;
402
403           title " " ;
404
405       %end ;
406
407    %let &mv_fail. = 0 ;
408
409    data _null_ ;
410      set __compare ;
411      if     data ne " "
412         and (     data_vars     ne vars_in_common
413               or  comp_vars     ne vars_in_common
414               or  data_obs      ne obs_in_common
415               or  comp_obs      ne obs_in_common
416               or  obs_w_unequal >  0
417             )
418      then
419        do ;
420           call symput( "&mv_fail."
421                      , "1"
422                      ) ;
423           stop ;
424        end ;
425
426    run ;
427
428
429    %__END:
430
431 %mend  mac_r_compare_report ;
```

Appendix 7. The MAC_U_DELETE macro.

```
%macro mac_u_delete
   ( library   = WORK
   , ds        =
   , help      = N
   ) ;

  %if &help. = Y
  %then
     %do ;
         %let mprint_orig = %sysfunc(getoption(mprint)) ;
         options nomprint ;
         skip ;
         skip ;
         %put ____________________________________________________________________________________________________________ ;
         %put Purpose of program:     This utility macro uses the DATASETS procedure to delete (a list of) SAS data sets, if     ;
         %put %str(                        )they exist.                                                                          ;
         %put                                                                                                                    ;
         %put Macro Parameter         Description                                                                               ;
         %put _________________       ____________________________________________________                                      ;
         %put library               = The libref of the data(s) to delete                                                        ;
         %put %str(                         )Default: WORK                                                                       ;
         %put ds                    = The name or the list of (one- or two-level) names of data set(s) to delete.                ;
         %put ____________________________________________________________________________________________                       ;
         %put End of help                                                                                                        ;
         options &mprint_orig. ;
         %goto __END ;
     %end ;


  %local i
         d
         library
         ds
         ;

  %if %index( &ds. , . ) = 0
  %then
     %do ;
         proc datasets
            library = &library.
            NoList
            ;

           %let i = 1 ;
           %let d = %scan( &ds. , &i. , %str( )) ;

           %do %while ( &d. ne ) ;

              %if %sysfunc( exist( &library..&d. )) %then delete &d. %str(;) ;

              %let i = %eval( &i. + 1 ) ;
              %let d = %scan( &ds. , &i. , %str( )) ;

           %end ;

        quit ;
     %end ;

   %else
      %do ;
          %let i = 1 ;
          %let d = %scan( &ds. , &i. , %str( )) ;

          %do %while ( &d. ne ) ;

             %if %sysfunc( index( &d. , . )) = 0 %then %let d = &library..&d. ;

             %if %sysfunc( exist( &d. ))
             %then
                %do ;
                    proc datasets
                       library = %if %index( &d. , . ) %then %scan( &d. , 1 , . ) ;
                                 %else &library. ;
                       NoList
                       ;

                      delete %if %index( &d. , . ) %then %scan( &d. , 2 , . ) ;
                             %else &d. ;
                            ;
                    quit ;
                %end ;

             %let i = %eval( &i. + 1 ) ;
             %let d = %scan( &ds. , &i. , %str( )) ;
          %end ;
      %end ;
  %__END:
%mend  mac_u_delete ;
```