PharmaSUG 2025 - Paper AP-192

A Macro to Automatically Check SAS Logs for Common Issues

Hailin Yu, Clinical Outcomes Solutions

ABSTRACT

Manually reviewing SAS logs can be time-consuming and error-prone, especially in complex clinical trial programming. This paper presents a macro that automatically scans SAS logs for common issues like errors, warnings, unresolved macro variables, and merge statements with missing BY variables. By automating log checking, programmers can save time and ensure high-quality outputs.

INTRODUCTION

The LOGCHK macro was developed to streamline the detection of critical messages in SAS®log files, such as ERROR, WARNING, and NOTE messages related to uninitialized variables. In large-scale or batch processing environments, where multiple programs are executed in a shared directory, manually reviewing each log file becomes time-consuming and error-prone.

This macro provides an automated and standardized approach to monitor log file health, making it easier for programmers and quality control teams to identify problems early in the workflow. By excluding its own log file from the scan, the macro avoids self-reporting and ensures more relevant results.

PROGRAM PATH IDENTIFICATION

To ensure the macro works regardless of the execution context (interactive, batch, or scheduled), it tries to detect the current program path using:

```
%if %SYMEXIST(_SASPROGRAMFILE) %then

	%let file = %sysfunc(DEQUOTE(&_SASPROGRAMFILE));

%else

	%do;

		%if %sysevalf(%superq(file)=,boolean) %then

			%let file = %sysfunc(getoption(sysin));


		* If still not found, try getting it from the environment;

		%if %sysevalf(%superq(file)=,boolean) %then

			%let file = %sysget(SAS_EXECFILEPATH);

	%end;
```

LOG FILE LISTING

The macro then extracts all .log files in the current directory using the Windows dir command through a PIPE:

```
filename loglst pipe "dir ""&path.\*.log"" /b";
```

LOG CONTENT PARSING

For each identified log file, the macro opens and reads each line. It checks for:

- Lines starting with ERROR: or WARNING:

- NOTE: messages that contain the word uninitialized

These are common indicators of code issues, and are flagged as potential problems.

OUTPUT AND REPORTING

All issues are appended into a dataset called logchk. The macro counts the number of records and:

- If issues are found, prints them in a clean report.
- If no issues are found, prints a clear message indicating that all logs are clean.

This helps streamline audits and code validation processes.

CUSTOMIZATION TIPS

All issues are appended into a dataset called logchk. The macro counts the number of records and:

- To change what gets flagged, modify the condition inside the data iter step.
- To run this on a different operating system, adjust the PIPE command accordingly.
- This macro can be embedded into nightly batch jobs or automated QC workflows.

CONCLUSION

In summary, the logchk macro is a simple and practical tool to help review SAS log files quickly. It automates the process of checking for common issues like errors, warnings, and uninitialized variables, saving time during programming and validation. While it has some limitations, it's still useful for catching key problems across multiple logs and improving overall log review efficiency.

LIMITATIONS

This macro works well for scanning .log files on Windows systems, but it has some limitations. It won't run on Unix without changes, since it uses Windows dir commands. It only checks for ERROR, WARNING, and uninitialized NOTE messages, so other issues won't be flagged. It skips its own log by hardcoding the name logchk, which needs to be updated if reused under a different name. It also doesn't sort files by date or group the messages by type, and performance might drop if there are too many large logs in the folder.

APPENDIX

```
/*************************************************************************

Get a list of all log files in the current folder,

but skip the log file from this logchk program itself.

*************************************************************************/

%macro logchk();

	%global file;


	* Try to figure out the full path of the current program;

	%if %SYMEXIST(_SASPROGRAMFILE) %then

		%let file = %sysfunc(DEQUOTE(&_SASPROGRAMFILE));

	%else

		%do;

			%if %sysevalf(%superq(file)=,boolean) %then

				%let file = %sysfunc(getoption(sysin));
```

```
		* If still not found, try getting it from the environment;
		%if %sysevalf(%superq(file)=,boolean) %then
			%let file = %sysget(SAS_EXECFILEPATH);
	%end;

data _null_;
	* Get program name without extension;
	pgm = tranwrd(scan(lowcase("&file."), -1, "\"), '.sas', '');
	call symputx("pgm", pgm);

	* Get the directory path of the program;
	path = tranwrd("&file.", scan("&file.", -1, "\"), '');
	call symputx("path", path);
run;

%put pgm is &pgm. path is &path;

* Get all .log files in the same folder;
filename loglst pipe "dir ""&path.\*.log"" /b";

data lognames;
	length fname $50;
	infile loglst;
	input fname $;

	* Remove file extension;
	fname = upcase(scan(fname, 1, '.'));

	* Skip this macro's log file;
	if fname ^= 'logchk';
run;

/*************************************************************************
Store all log file names (no extensions) into a macro variable
```

```
      ************************************************************************/
      proc sql noprint;
             select fname into :pgmlogs separated by ' ' from lognames;
             select count(*) into: nlogs from lognames;
      quit;

      /************************************************************************
      Read each log file, pick up ERRORs, WARNINGs, and uninitialized variable messages
      ************************************************************************/
      %do i = 1 %to &nlogs;
             %let logfile = %scan(&pgmlogs, &i);

             data iter;
                    length line $200 fname $50;
                    infile "&path.\&logfile..log" missover pad;
                    input line $1-200;

                    * Look for ERROR, WARNING, or "uninitialized" NOTE messages;
                    if scan(line, 1, ': ') in ('WARNING', 'ERROR') or (scan(line, 1, ':
') = 'NOTE' and index(lowcase(line), 'uninitialized') > 0);
                    fname = "&logfile";
             run;

             proc append base=logchk data=iter force;
             run;

      %end;

      /************************************************************************
      Count how many issues (errors/warnings/uninit vars) were found
      ************************************************************************/
      proc sql noprint;
             select count(*) into :nobs from logchk;
      quit;
```

```
	/*************************************************************************
	Print a report of the log issues (or show a clean message if none)
	*************************************************************************/
	title "Log Issues in &path.";
	footnote j=l"System User ID: &sysuserid" j=r"Runtime: &pgmrundt.";

	%if &nobs > 0 %then
		%do;

			proc print data=logchk;
				var fname line;
			run;

		%end;
	%else
		%do;

			data _null_;
				put @1 '***** No Errors or Warnings Found in Log Files *****';
				put @1 "System User ID: &sysuserid Runtime: &pgmrundt.";
			run;

		%end;
%mend;

%logchk;

* End of program;
```

CONTACT INFORMATION

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

Hailin Yu
Clinical Outcomes Solutions
hailin.yu@clinoutsolutions.com