Paper SS04

Post-mortem Logs in R

Teckla Akinyi, Janssen Pharmaceutical Companies of Johnson & Johnson, Springhouse PA, USA

ABSTRACT

Outlined is code snippet in the R programming language that allows for a post-mortem analysis of the log files of program. The snippet will programmatically scan multiple logs generated by the {logrx} package[1] developed in R programming language as part of *pharmaverse*[2], or SAS (SAS Institute Inc). These will be parsed and information on errors, warnings and other pertinent messages used in validating programs will be extracted; the output is a "dump-file" in Excel with these pieces of information.

Given the log file from {logrx} is arranged in sections e.g., "User and File information", "Errors and Warning", "Messages", "Output" and "Results, different from SAS log where these messages are 'in-line', a highlight of this snippet is it extracts information from both systems.

Such an algorithm increases efficiency of validating logs, eliminating the need to manually scan multiple files for information pertinent in validating logs. Additionally, it will contribute to repository of R algorithms as the industry moves toward bilingualism.

INTRODUCTION

Log files generated from SAS and R ({logrx}) are both saved as text documents containing information on how the corresponding script was implemented. These files can be massive and require programmers to scan/search for the bits of information like errors, warnings, and other messages. A process that when left to manual checks is highly inefficient and tedious. An old age problem, solved in the industry by writing scripts that auto filter the logs in the form of an in-script SAS macro named as "scanLog", "checklog", "parseLog" etc. and spits a dump file of industry choice (excel, rtf, etc.) that is concise with information that need to be addressed by the programmer.

This paper builds on the known solution by expanding it to log files generated by R scripts using the {logrx} file. Other than adding another snippet for programmers to run depending on whether the log was generated by SAS or R, this snippet is language agnostic; however, it is designed to be run in-script within the R language and dumps the information into an excel sheet. Development in R allows for further enhancements including the potential of creating an add-in functionality in the R Studio integrated development environment (IDE), making parse logs a "point and click" task.

INSPECTING LOG FILES

Albeit both text files, the structure of SAS generated log files is different from that of ones by {logrx}. The first noticeable difference is the sections in {logrx} versus sequential implemented lines in SAS logs. Further, the lines in SAS contain the key words of relevant messages versus {logrx} where the key word is the section header. This makes it a challenge as the extraction of information may not be a simple search for keywords and pulling the lines in which they appear for {logrx}.

```
--------------------------------------------------------------------------------
-                          Used Package and Functions                          -
--------------------------------------------------------------------------------
{package:base} data.frame, c, print, cat, environment, library, message, warning, stop
{package:dplyr} select, arrange
{package:magrittr} %>%
{package:rlang} inform
--------------------------------------------------------------------------------
-                         Program Run Time Information                         -
--------------------------------------------------------------------------------
Start time: 2023-02-15 10:51:27 EST
End time: 2023-02-15 10:51:29 EST
Run time: 2 seconds
--------------------------------------------------------------------------------
-                              Errors and Warnings                             -
--------------------------------------------------------------------------------
Errors:
        log error

Warnings:
        log warning
        log warning2
--------------------------------------------------------------------------------
-                         Messages, Output, and Result                         -
--------------------------------------------------------------------------------
Messages:
        log inform
        log inform 2
        log message
        log message 2
Output:
        [1] "log print"
        [1] "log print 2"
        log catlog cat 2<environment: 0x00000250e0f96ea0>

Result:
        NULL
--------------------------------------------------------------------------------
```

```
297
298        data cmpr3(drop=col:);
299          set cmpr2x;
300          CMINDCPR=catx(';',of col:);
                         ____
                         71
ERROR 71-185: The CATX function call does not have enough arguments.

301        run;

WARNING: The variable 'col:'n in the DROP, KEEP, or RENAME list has never been referenced.
NOTE: The SAS System stopped processing this step because of errors.
WARNING: The data set WORK.CMPR3 may be incomplete.  When this step was stopped there were 0 observations and 3 variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      user cpu time       0.00 seconds
      system cpu time     0.00 seconds
      memory              889.59k
      OS Memory           39328.00k
      Timestamp           02/13/2023 04:48:21 PM
      Step Count                        197  Switch Count  2
      Page Faults                       0
      Page Reclaims                     12
      Page Swaps                        0
      Voluntary Context Switches        12
      Involuntary Context Switches      0
      Block Input Operations            0
      Block Output Operations           272


302        proc sort data=cmpr3; by usubjid cmseq; run;
```

Figure 1. Top – log file from {logrx} structured as section. Bottom – log file from SAS showing sequential messages based on implementation order and in-line keywords.

PARSELOGS() FUNCTION

The *parselogs*() function was developed under R version 4.2.2 (2022-10-31) as an in-script function to subset either and both SAS generated logs and {logrx} generated logs. The library dependencies of this function include: magrittr, dplyr, tidyr, data.table, stringi and openxlsx and should be installed before attempting to run the function.

parselogs() will read log files in a user specified directory and create an excel file that contains: the name of the log file, the category of the information (Error, Warning or Other) and the value – verbatim of the line. It expects 7 user specified input (arguments) of which 3 are optional. Table 1, describes these arguments:

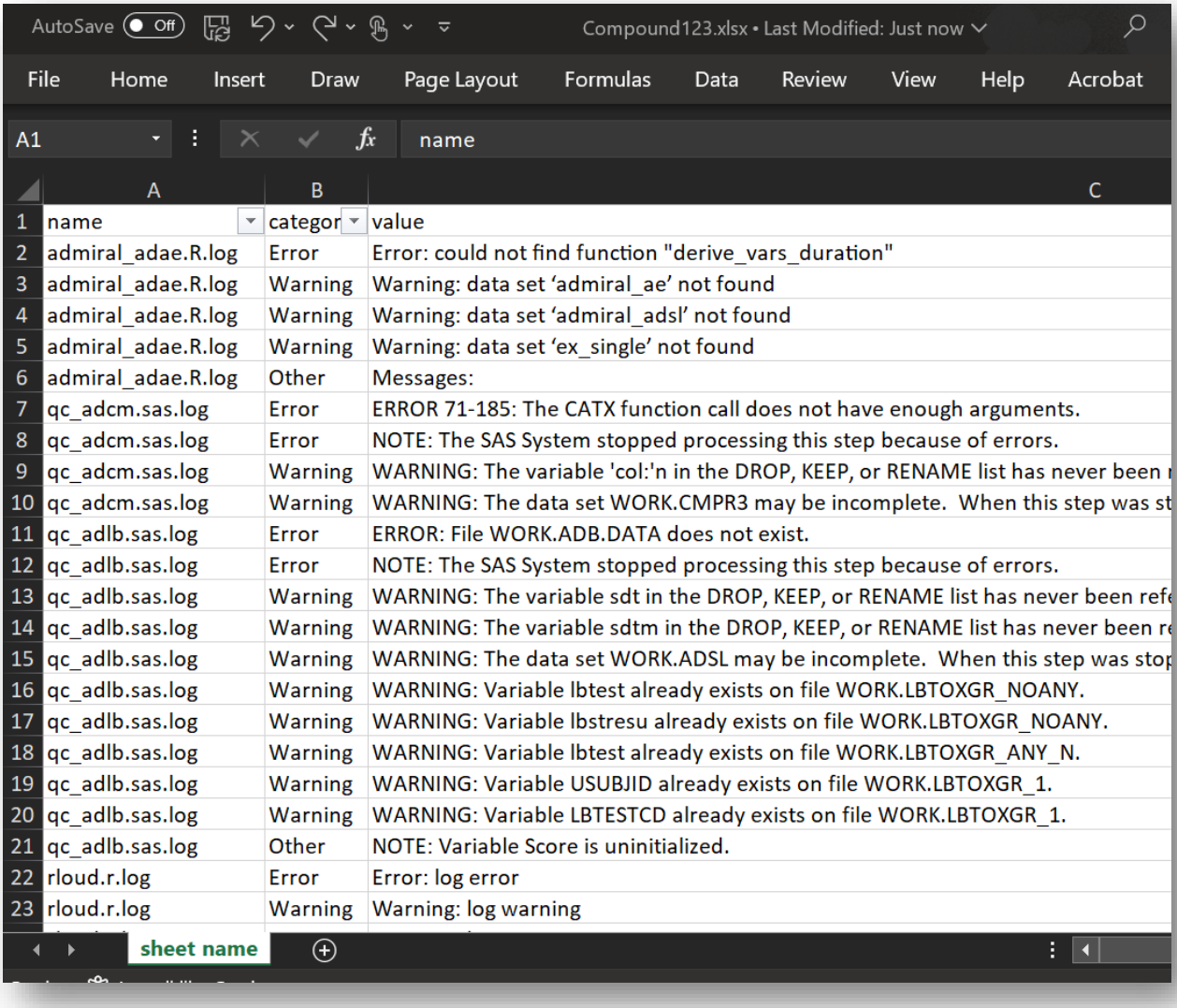
Argument	Requirement	Description
logdir	Required	Path to the input directory, defaults to the working directory
select_file	Optional	For user-selection of log file to be checked only one log at a time. Default to all logs in directory.
sas_file_ID	Required	Identifiers of a sas file, typically a string in header of program that also prints to log text file.
sas_key	Optional if no SAS logs are expected	Key words that user wants to be extracted from SAS files. The list is extensible for user to add other key words per company requirements.
add_r_sxctn	Optional	Used for R files to extract other sections beyond Errors, Warnings & Messages e.g section with masked function. If not specified or left at default then only errors, warnings & messages are extracted.
Sheet_nm	Optional	ID to be used as the excel sheet name can be file name or studyid if looking through different study dirs. Default is "sheet name"
outnm	Required	Name for the excel workbook. Can specify full path or just name. If only the desired file name is specified file will be stored in the path specified in the *logdir* argument.

BARE BONES CALL OF THE FUNCTION

This is the simplest call of the function:

```
parselogs(logdir = "C:/Users/TAkinyi/Desktop/Yangu2/PHUSEABC",
          select_file=,
          sas_file_ID = c("SAS Institute Inc| SAS Version"),
          sas_key = c("ERROR|WARNING"),
          add_r_sxtn =,
          Sheet_nm =,
          outnm ="Compound123.xlsx")
```

The *logdir* argument specifies the location of the log files of interest and the *sas_file_ID* contains information on how SAS files are identified. These identifiers are expected at the header of the log file and are versions of a statement with the word “SAS”. *sas_key* only searches for the key word “ERROR” or “WARNING” within SAS logs. For R logs, the sections “Errors and Warnings” and “Messages” are default searched and this need not be specified in the call. The *outnm* argument specifies the name of the output excel workbook, that will be saved in the same path specified in logdir. The arguments *select_file*, *add_r_sxtn* and *Sheet_nm* are left blank as these are optional. Note, they need not appear in the function call at all but are shown here for demonstration. Additionally, despite R being a case sensitive language, all the character strings specified by the user are not case sensitive. This is possible by use of functions within the source program that have options on case sensitivity to user input.
The output file, shown below, is a excel workbook named “Compound 123” with extracted errors and warnings from SAS logs, Errors, Warnings, and messages from {logrx} logs in a sheet named “sheet name” with filtering enabled. The log file names have the identifier infix of “. R” or “.sas” only for demonstrative purposes, these need not be in the name of the file.



	A	B	C
1	name	categor	value
2	admiral_adae.R.log	Error	Error: could not find function "derive_vars_duration"
3	admiral_adae.R.log	Warning	Warning: data set 'admiral_ae' not found
4	admiral_adae.R.log	Warning	Warning: data set 'admiral_adsl' not found
5	admiral_adae.R.log	Warning	Warning: data set 'ex_single' not found
6	admiral_adae.R.log	Other	Messages:
7	qc_adcm.sas.log	Error	ERROR 71-185: The CATX function call does not have enough arguments.
8	qc_adcm.sas.log	Error	NOTE: The SAS System stopped processing this step because of errors.
9	qc_adcm.sas.log	Warning	WARNING: The variable 'col:'n in the DROP, KEEP, or RENAME list has never been r
10	qc_adcm.sas.log	Warning	WARNING: The data set WORK.CMPR3 may be incomplete. When this step was st
11	qc_adlb.sas.log	Error	ERROR: File WORK.ADB.DATA does not exist.
12	qc_adlb.sas.log	Error	NOTE: The SAS System stopped processing this step because of errors.
13	qc_adlb.sas.log	Warning	WARNING: The variable sdt in the DROP, KEEP, or RENAME list has never been refe
14	qc_adlb.sas.log	Warning	WARNING: The variable sdtm in the DROP, KEEP, or RENAME list has never been re
15	qc_adlb.sas.log	Warning	WARNING: The data set WORK.ADSL may be incomplete. When this step was stop
16	qc_adlb.sas.log	Warning	WARNING: Variable lbtest already exists on file WORK.LBTOXGR_NOANY.
17	qc_adlb.sas.log	Warning	WARNING: Variable lbstresu already exists on file WORK.LBTOXGR_NOANY.
18	qc_adlb.sas.log	Warning	WARNING: Variable lbtest already exists on file WORK.LBTOXGR_ANY_N.
19	qc_adlb.sas.log	Warning	WARNING: Variable USUBJID already exists on file WORK.LBTOXGR_1.
20	qc_adlb.sas.log	Warning	WARNING: Variable LBTESTCD already exists on file WORK.LBTOXGR_1.
21	qc_adlb.sas.log	Other	NOTE: Variable Score is uninitialized.
22	rloud.r.log	Error	Error: log error
23	rloud.r.log	Warning	Warning: log warning

Figure 2. Sample output of simplest function call. With validation messages extracted from both SAS and {logrx} generated files as in the extension of the file names under the column "name".

OTHER EXAMPLE CALLS

This section provides example calls to show case the versatility of the *parselogs*() function. Each example is accompanied by a short descriptor of what it is trying to achieve.

1. The following call will search for all log files in the specified path. The argument *sas_file_ID* has been extended to include "SAS version", *sas_key* to include the key words "uninitialized" and "interruption". The output workbook, named "Compound123" will store the data in a sheet named "STUDY 123ABC".

```
parselogs(logdir = "C:/Users/TAkinyi/Desktop/Yangu2/PHUSEABC",
          sas_file_ID =  c("SAS Institute Inc|SAS program|SAS Version"),
          sas_key = c("WARNING|ERROR|UNINITIALIZED|INTERRUPTION"),
          Sheet_nm = "STUDY_123ABC",
          outnm ="Compound123.xlsx")
```

2. The following call is like example 1 above, however it reads logs from a different path e.g., a different study and creates a sheet named “STUDY 123ABC” in the same workbook. The workbook would now contain two aptly named sheets. This utility may be handy when validating logs from different studies of the same compound.

```
parselogs(logdir = "C:/Users/TAkinyi/Desktop/Yangu/phuse23",
          sas_file_ID =  c("SAS Institute Inc|SAS program|SAS Version"),
          sas_key = c("WARNING|ERROR|UNINITIALIZED|INTERRUPTION"),
          Sheet_nm = "STUDY_123ABC",
          outnm ="C:/Users/TAkinyi/Desktop/Yangu2/PHUSEABC/Compound123.xlsx")
```

3. This call will read only the SAS files in the specified path. A caveat of this call is that the files in the path have an extension that identifies the SAS logs or the user is certain that only these are present in the directory, then the input to *select_file* will take the string “*.log$”. The $ symbol anchors the string to search for files ending in .log. The output is another sheet added to the workbook with information extracted from only SAS generated logs, named “all sas”.

```
parselogs(logdir = "C:/Users/TAkinyi/Desktop/Yangu2/PHUSEABC",
          sas_file_ID =  c("SAS Institute Inc|SAS program|SAS Version"),
          sas_key = c("WARNING|ERROR|UNINITIALIZED|INTERRUPTION|REPEAT"),
          Sheet_nm = "all sas",
          select_file="*.sas.log$",
          outnm ="Compound123.xlsx")
```

4. This call will read only the {logrx} files in the specified path. A caveat of this call is that the files in the path have an extension that identifies the {logrx} log files or the user is certain that these logs are present in the directory, then the input to *select_file* will take the string “*.log$”. The $ symbol anchors the string to search for files ending in .log. The output is another sheet added to the workbook with information extracted from only SAS generated logs, named “all R”. For this call only errors, warnings and message are extracted.

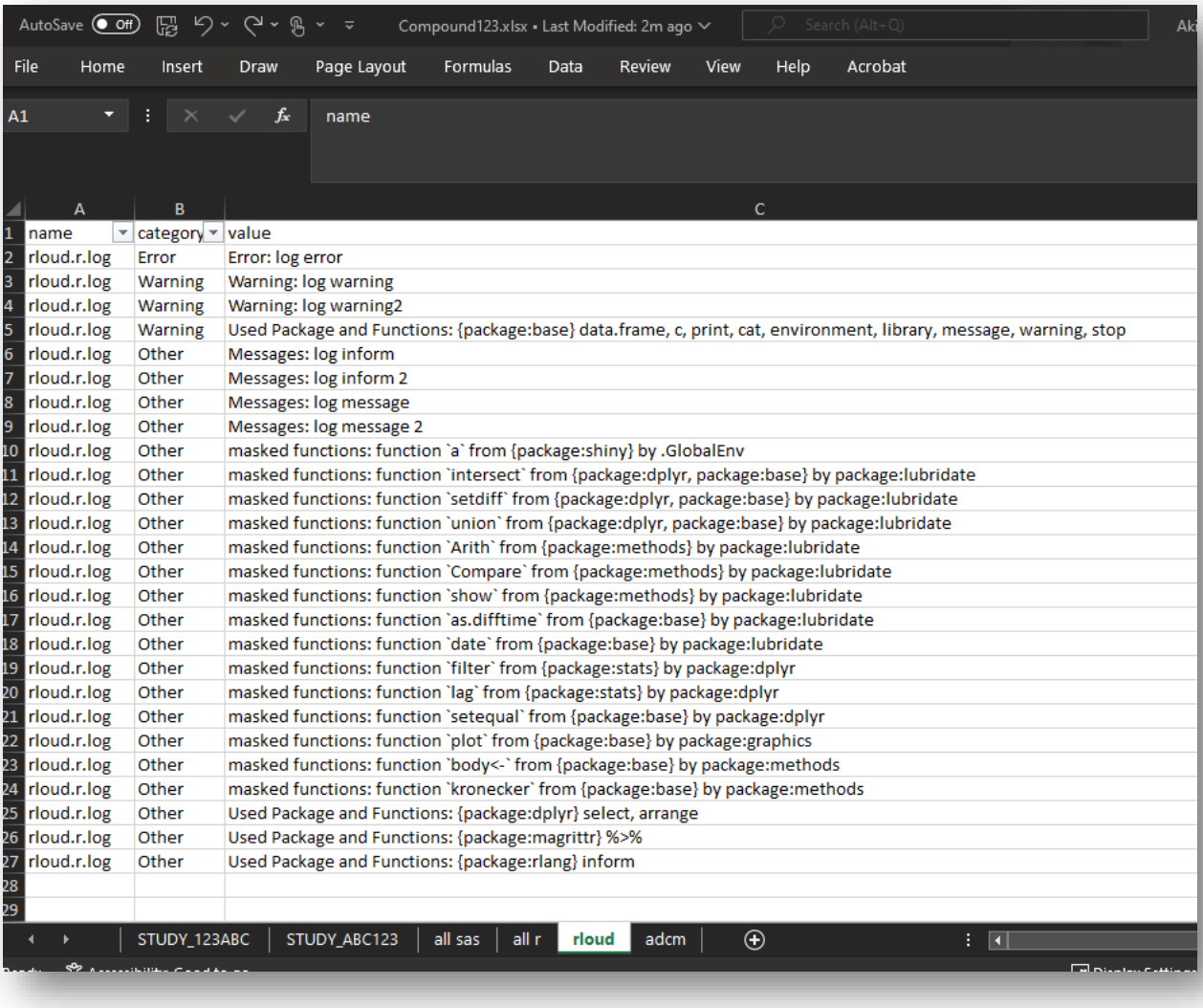
```
parselogs(logdir = "C:/Users/TAkinyi/Desktop/Yangu2/PHUSEABC",
          sas_file_ID =  c("SAS Institute Inc|SAS program|SAS Version"),
          Sheet_nm = "all r",
          select_file="*R.log$",
          outnm ="Compound123.xlsx")
```

5. The following two examples are for when the user is interested in a specific log file. The first specifies a {logrx} file and the latter a sas file. The output excel has the two respective named sheets added to it. If a different excel sheet is desired, then changing the excel sheet name using the *outnm* argument is necessary. Note for SAS generated files the *sas_key* argument is necessary, while the *add_r_sxtn* will extract the specified sections from the {logrx} file.

```
#call function 1- with one file selected - {logrx} file
parselogs(logdir = "C:/Users/TAkinyi/Desktop/Yangu2/PHUSEABC",
          sas_file_ID =  c("SAS Institute Inc|SAS program|SAS Version"),
          Sheet_nm = "rloud",
          select_file="rloud-r.log",
          add_r_sxtn = c("masked functions","Used Package and Functions"),
          outnm ="Compound123.xlsx")

#call function 2- with one file selected - SAS file
parselogs(logdir = "C:/Users/TAkinyi/Desktop/Yangu2/PHUSEABC",
          sas_file_ID =  c("SAS Institute Inc|SAS program|SAS Version"),
          sas_key = c("WARNING|ERROR|UNINITIALIZED|INTERRUPTION"),
          Sheet_nm = "adcm",
          select_file="qc_adcm.sas.log",
          outnm ="Compound123.xlsx")
```

If the above example calls are run sequentially, the output file is a single excel workbook with the 6 respective sheets as shown below.



	A	B	C
1	name	category	value
2	rloud.r.log	Error	Error: log error
3	rloud.r.log	Warning	Warning: log warning
4	rloud.r.log	Warning	Warning: log warning2
5	rloud.r.log	Warning	Used Package and Functions: {package:base} data.frame, c, print, cat, environment, library, message, warning, stop
6	rloud.r.log	Other	Messages: log inform
7	rloud.r.log	Other	Messages: log inform 2
8	rloud.r.log	Other	Messages: log message
9	rloud.r.log	Other	Messages: log message 2
10	rloud.r.log	Other	masked functions: function `a` from {package:shiny} by .GlobalEnv
11	rloud.r.log	Other	masked functions: function `intersect` from {package:dplyr, package:base} by package:lubridate
12	rloud.r.log	Other	masked functions: function `setdiff` from {package:dplyr, package:base} by package:lubridate
13	rloud.r.log	Other	masked functions: function `union` from {package:dplyr, package:base} by package:lubridate
14	rloud.r.log	Other	masked functions: function `Arith` from {package:methods} by package:lubridate
15	rloud.r.log	Other	masked functions: function `Compare` from {package:methods} by package:lubridate
16	rloud.r.log	Other	masked functions: function `show` from {package:methods} by package:lubridate
17	rloud.r.log	Other	masked functions: function `as.difftime` from {package:base} by package:lubridate
18	rloud.r.log	Other	masked functions: function `date` from {package:base} by package:lubridate
19	rloud.r.log	Other	masked functions: function `filter` from {package:stats} by package:dplyr
20	rloud.r.log	Other	masked functions: function `lag` from {package:stats} by package:dplyr
21	rloud.r.log	Other	masked functions: function `setequal` from {package:base} by package:dplyr
22	rloud.r.log	Other	masked functions: function `plot` from {package:base} by package:graphics
23	rloud.r.log	Other	masked functions: function `body<-` from {package:base} by package:methods
24	rloud.r.log	Other	masked functions: function `kronecker` from {package:base} by package:methods
25	rloud.r.log	Other	Used Package and Functions: {package:dplyr} select, arrange
26	rloud.r.log	Other	Used Package and Functions: {package:magrittr} %>%
27	rloud.r.log	Other	Used Package and Functions: {package:rlang} inform
28			
29			

Figure 3. Output file with multiple tabs containing extracted information from {logrx} and SAS generated files. The tab names correspond to the example calls in this paper.

OTHER FEATURES

The function is designed to print useful information in the console as it runs. Thus far, a message will print with a list of the log files read and the path they are read from. Additionally, if a sheet tab is already within the work book a warning message will be printed with the name of the sheet tab being over-written.

FURTHER DEVELOPMENT

Some known issues of this function include the following:

1. The *add_r_sxtn* can only be used when parsing an individual {logrx} file, as in example 5.
2. The source code for the function is currently a large function, which may be costly in system run time.
3. The argument *select_file* can only accommodate one file at a time.

Addressing the known issues would make for a more efficient function, however this presents a starting point for interested parties. Another potential enhancement is the addition of a sequential counter to the output file that informs on the order of messages.

REMARKS

This function relies on the source code "usrchklog3.R" included at the top of the script with the following syntax.

```
source("pathname/usrchklog3.R")
```

The source code is available in the following github repository on a read access:
https://github.com/tkakinyi/phuse2023/tree/main
Disclaimer: This function has not been peer tested and the author assumes no responsibility for mishaps caused by its use.

CONCLUSION

The function presented here offers an opportunity to harmonize tools as the industry moves toward bilingualism. Instead of creating a single tool for each programming language, some can be created to be language agnostic. Further opportunity is to include this function as a feature in the {logrx} package development, potentially as an add-in within the R Studio IDE, lessening the burden on each programmer storing the source code.

REFERENCES

1. Kosiba N, Bermudez T, Straub B, Rimler M, Masel N (2023). {logrx}: A Logging Utility Focus on Clinical Trial Programming Workflows. R package version 0.2.1, https://github.com/pharmaverse/{logrx}.
2. Pharmaverse. Retrieved February 16, 2023, from https://pharmaverse.org/

ACKNOWLEDGMENT

Appreciation to all industry mentors for the support in discussions as I explored the development of this function and much to Janssen leadership who afforded me the time in these efforts.

CONTACT INFORMATION

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

Author Name Teckla Akinyi
Company The Janssen Pharmaceutical Companies of Johnson & Johnson
Address 1400 McKean Rd
City / Postcode Lower Gwynedd Township /19002
Email: takinyi@its.jnj.com