Extract SAS Programs from Log Files Using Regular Expressions

Hengwei Liu, Hengrui USA, Princeton, New Jersey

ABSTRACT

Programmers may accidentally delete a SAS® program when the work gets hectic. Don't panic. If the log file for the SAS program is available, don't grab the phone and bother the system administrator to restore the SAS program for you. The SAS program can be extracted from the log file. In this paper we will show how to use regular expressions in different languages to get this done.

INTRODUCTION

It is a real-life situation: the programmer deleted a SAS program by mistake, but the log file is still there. One can get the SAS program back by manually editing the log file, but manually editing a text file is an anathema to programmers. It is boring and prone to mistakes. Programmers should use the regular expressions to handle this task. Regular expression is about finding patterns in the text files and makes editing much easier in many scenarios.

We will use regular expression to extract the SAS program from the log fille. We'll show how to do this in SAS, R, Python and Linux shell script.

EXTRACT THE SAS PROGRAM

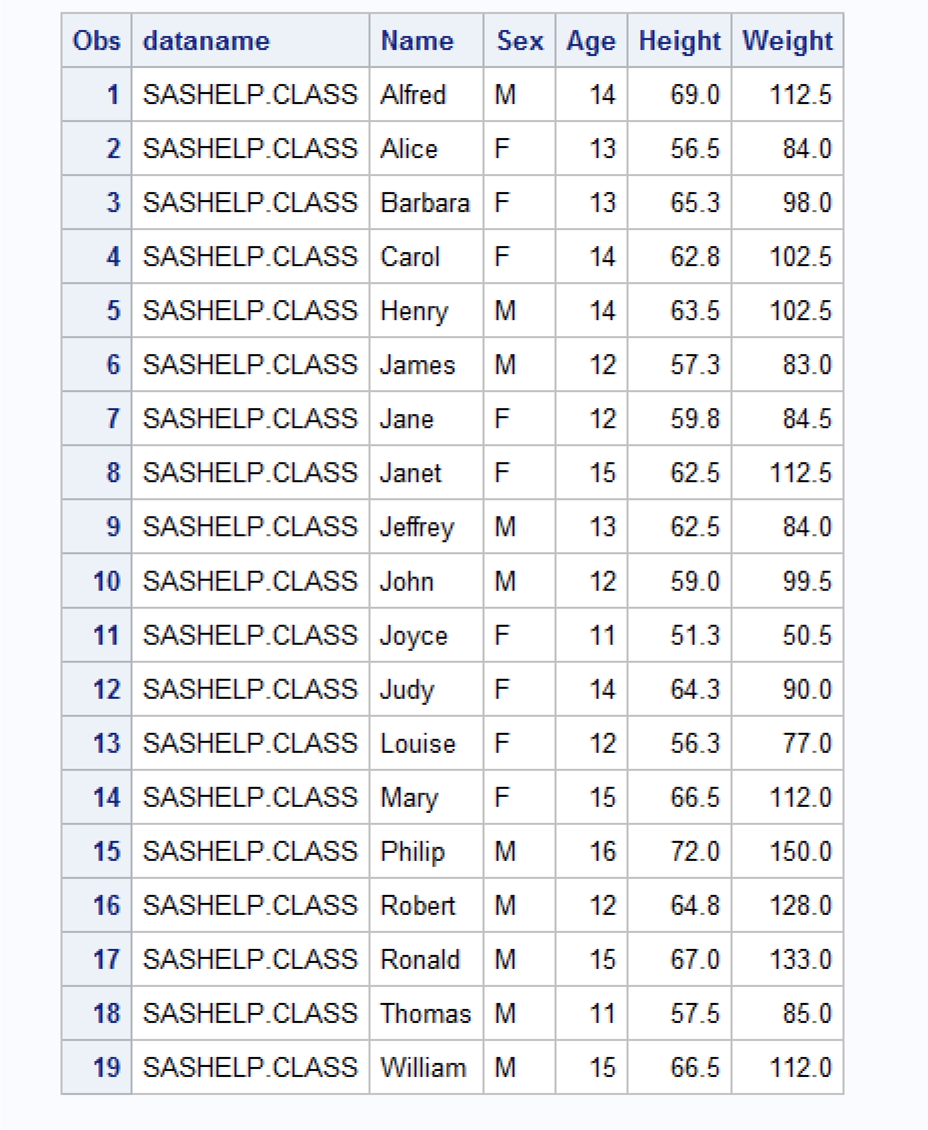
There is an interesting way of merging a dataset with many records and a dataset with one record. Let's look at a program merge.sas.

```
data dataname;
dataname="SASHELP.CLASS";
run;

data combine;
if _n_=1 then set dataname;
set sashelp.class;
run;

proc print data=combine;
run;
```

The display 1 shows the output.



Obs	dataname	Name	Sex	Age	Height	Weight
1	SASHELP.CLASS	Alfred	M	14	69.0	112.5
2	SASHELP.CLASS	Alice	F	13	56.5	84.0
3	SASHELP.CLASS	Barbara	F	13	65.3	98.0
4	SASHELP.CLASS	Carol	F	14	62.8	102.5
5	SASHELP.CLASS	Henry	M	14	63.5	102.5
6	SASHELP.CLASS	James	M	12	57.3	83.0
7	SASHELP.CLASS	Jane	F	12	59.8	84.5
8	SASHELP.CLASS	Janet	F	15	62.5	112.5
9	SASHELP.CLASS	Jeffrey	M	13	62.5	84.0
10	SASHELP.CLASS	John	M	12	59.0	99.5
11	SASHELP.CLASS	Joyce	F	11	51.3	50.5
12	SASHELP.CLASS	Judy	F	14	64.3	90.0
13	SASHELP.CLASS	Louise	F	12	56.3	77.0
14	SASHELP.CLASS	Mary	F	15	66.5	112.0
15	SASHELP.CLASS	Philip	M	16	72.0	150.0
16	SASHELP.CLASS	Robert	M	12	64.8	128.0
17	SASHELP.CLASS	Ronald	M	15	67.0	133.0
18	SASHELP.CLASS	Thomas	M	11	57.5	85.0
19	SASHELP.CLASS	William	M	15	66.5	112.0

Display 1. Output the Program Merge.sas

Display 2 shows the log file merge.log.

```
1                               The SAS System  10:07 Friday, February 18, 2022

NOTE: Copyright (c) 2016 by SAS Institute Inc., Cary, NC, USA.
NOTE: SAS (r) Proprietary Software 9.4 (TS1M7)
      Licensed to HENGRUI USA, Site 70285548.
NOTE: This session is executing on the X64_10PRO  platform.

NOTE: Analytical products:

      SAS/STAT 15.2

NOTE: Additional host information:

 X64_10PRO WIN 10.0.19041  Workstation
```

```
NOTE: SAS initialization used:
      real time           0.09 seconds
      cpu time            0.12 seconds

1          data dataname;
2          dataname="SASHELP.CLASS";
3          run;
2                                The SAS System  10:07 Friday, February 18, 2022

NOTE: The data set WORK.DATANAME has 1 observations and 1 variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds


4
5          data combine;
6            if _n_=1 then set dataname;
7            set sashelp.class;
8          run;

NOTE: There were 1 observations read from the data set WORK.DATANAME.
NOTE: There were 19 observations read from the data set SASHELP.CLASS.
NOTE: The data set WORK.COMBINE has 19 observations and 6 variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds


9
10         proc print data=combine;
3                                The SAS System  10:07 Friday, February 18, 2022

11         run;

NOTE: There were 19 observations read from the data set WORK.COMBINE.
NOTE: The PROCEDURE PRINT printed page 1.
NOTE: PROCEDURE PRINT used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds


NOTE: SAS Institute Inc., SAS Campus Drive, Cary, NC USA 27513-2414
NOTE: The SAS System used:
      real time           0.12 seconds
      cpu time            0.14 seconds
```

Display 2. Merge.log

Now let’s consider the situation where the program has been deleted and we need to get it back from the log file. Notice that the lines with SAS codes all start with a number. That’s the pattern we’ll use. But also note that the title line with the words “The SAS System” starts with a page number. We need to skip such lines when we extract the SAS codes.

This is how to do it in SAS. We use the PRXMATCH function to include the lines that start with a number and exclude those lines that contain the words “The SAS System”. And we use the function

PRXCHANGE to remove the leading numbers in a line. The regular expression /^[0-9]/+ means one or more digits at the start of a line. This is the SAS program log2sas.sas.

```
data log;
infile "C:\regular expression\merge.log" print lrecl=2000 pad missover;
input text $char2000.;
run;


data match; set log;
pos1=prxmatch("/^[0-9]+/", text);
pos2=prxmatch("/The SAS System/", text);
if pos1=1 and pos2=0;
code=prxchange("s/^[0-9]+//", 1, text);
run;

options nodate nonumber;
title;

ods listing file='C:\regular expression\merge.sas';
data _null_;
set match;
file print;
put @1 code;
run;

ods listing close;
```

To do this in R, we use the regexpr and sub functions. The regexpr is used to extract the lines with SAS codes. The sub function is used to remove the leading numbers in a line. The regular expression is the same as in the SAS program, i.e., ^[0-9]+. This is the R program log2sas.R.

```
data <- read.delim(file = "C:\\regular expression\\merge.log",
header=FALSE, quote="")

data.frame(data)


data$flag <- regexpr("^[0-9]+", data$V1)

data$flag2 <- regexpr("The SAS System", data$V1)


data2 <- data[(data$flag==1 & data$flag2==-1), ]

data2$V2 <- sub('^[0-9]+', '', data2$V1)

data2 <- data2[c("V2")]



sink('C:\\regular expression\\merge.sas')


print(unname(data2), row.names=FALSE)
```

```
sink()
```

To do this work in Python, we use the re module. We create two patterns with re.compile and use them to identify the lines to be extracted from the log file. The split function is used to remove the leading numbers in each line. This is the Python program log2sas.py done on Linux system.

```
#!/usr/bin/python3
import re
codes = [ ]
pattern = re.compile (r"^[0-9]")
pattern1 = re.compile (r"the sas system", re.I)
outfile=open('merge.sas','w')

with open ('merge.log','rt') as myfile:
    for line in myfile:
         if (pattern.search(line) != None) and (pattern1.search(line) == None) :
            codes.append(line)

for code in codes:
   print(code.split(' ',1)[1], file=outfile)
```

Finally, we show how to do this work in a BASH script log2sas.sh. We use the cut function to find out the log file name which is used to name the SAS program. We use the regular expression ^[[:digit:]] to identify the leading numbers in a line. The sed function is used to remove the lines containing the words "The SAS System" and to remove the leading numbers in a line. To run this script run the command ./log2sas.sh merge.log.

```
#!/usr/bin/bash
for file in $*
do
NAME=$(echo "$file" | cut -f 1 -d '.')
EXT=.sas
cat $file | grep ^[[:digit:]] \
| sed '/The SAS System/d' \
| sed 's/^[0-9]*//g' > $NAME$EXT
done
```

CONCLUSION

Regular expression is a powerful tool. It looks enigmatic and intimidating when one first looks at some long strings in regular expressions. But once you know how to construct regular expressions, you'll realize they are beautiful and they make perfect sense. Each programmer should spend some time to get a grasp of the mechanism and intricacy of the regular expressions. It is worth it.

CONTACT INFORMATION

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

Hengwei Liu
Hengrui USA
400 Alexander Park
Princeton, NJ 08540
Email: Hengwei_liu@yahoo.com