

The Lazy Rsubmit

Stephanie R. Thompson, University of Memphis, Memphis, TN

ABSTRACT

Have you ever written a long program in sections in order to debug it more easily only to find some stray rsubmits messing up your final check? Rsubmit is a necessity for remote processing but it only needs to be used once in the final version of the program. However, using rsubmit along the way when writing your code saves you from re-submitting sections that have already been error checked. The "Lazy Rsubmit" is a way you can execute various sections of code without having to remove existing rsubmits.

INTRODUCTION

SAS programs submitted on a remote machine for execution require the use of the rsubmit statement in your client session. Without this statement your program will not run on the remote system. There are many reasons to use a remote system to run your SAS code, but they are not part of this paper. Let's just assume that you need to use remote processing for the project you are working on. If your project is very complex or has a lot of parts, you may want to follow good programming practice and debug your code in sections rather than all at once. Debugging an entire program at once can be daunting since many of the errors and warnings build on each other. Also, you will use much more remote processor time this way since your program may continue to attempt execution even though it has encountered an error.

Debugging in sections allows you to test each section of code on the remote system to make sure it is working. Once all of the pieces are debugged, you have a high probability that your entire program will be bug free. As you develop your program, you will be inserting a rsubmit statement before each section of code you wish to test. You can just highlight the section of code you want to submit remotely (starting at the rsubmit statement) and hit F8 or the running man icon. In this manner only the errors, warning, and notes relevant to this particular section of code are shown in the Log Window. Plus, since the amount of code submitted is smaller (usually one data step or one procedure) finding errors is much simpler than looking through a hundred lines of code.

The following sections will walk you through the general syntax for the rsubmit and introduce you to the Lazy Rsubmit.

GENERAL RSUBMIT SYNTAX

The rsubmit statement has a very simple syntax. The general form is shown below:

```
rsubmit;
```

The data step or procedure you wish to run on the remote system follows the rsubmit statement and can optionally end with an endrsubmit statement as seen in the following example:

```
rsubmit;
  data two;
    set one;
    where j_class in ('Full Time', 'Flex Time');
  run;
endrsubmit;
```

If all this code is submitted or if just the rsubmit and data step are submitted, you will see the notes, warnings, and errors associated with the submission. The use of the endrsubmit is not necessary.

EXAMPLE OF LOG

The code from above was submitted and the log file is below.

```
20  rsubmit;
NOTE: Remote submit to MYNODE commencing.
756  data two;
757  set one;
758  where j_class in ('Full Time', 'Flex Time');
759  run;
```

NOTE: The dataset WORK.TWO has 136 observations and 4 variables.

NOTE: Remote submit to MYNODE complete.

The log file is the same whether or not the endrsubmit; line is part of the submission or not.

LAZY RSUBMIT

Making a small change to the code used previously turns the rsubmit into a Lazy Rsubmit. The change is shown below:

```
*rsubmit;  
  data two;  
  set one;  
  where j_class in ('Full Time', 'Flex Time');  
  run;  
endrsubmit;
```

The addition of the asterisk in front of the rsubmit allows the user to select part of the code for submission or will allow the continuation of an existing rsubmit since this line will be viewed as a comment if part of a larger program. The section of the code to highlight and submit is shown below:

```
*rsubmit;  
  data two;  
  set one;  
  where j_class in ('Full Time', 'Flex Time');  
  run;  
endrsubmit;
```

Submitting the code above produces the same results and log as the first examples.

REPEATED USE IN A PROGRAM

Using the *rsubmit; allows you to select all of a program or only a part of a program to remotely submit to SAS. The concept is illustrated below:

```
rsubmit;  
  data two;  
  set one;  
  where j_class in ('Full Time', 'Flex Time');  
  run;  
  
*rsubmit;  
  proc sort data = two;  
  by id;  
  run;  
  
*rsubmit;  
  proc sort data = three;  
  by id;  
  run;  
  
*rsubmit;  
  data compensation;  
  merge two three;  
  by id;  
  run;  
  
*rsubmit;  
  proc print data = compensation;  
  run;
```

Submitting the entire program will result in the execution of the first data step, two subsequent sorts, one merge, and a final print. The commented rsubmit commands do not interfere with the execution of the program. They look like comments to SAS. However, if you had a syntax error in your merge, you would not want to re-run the entire program. You would only need to highlight part of the program. This is illustrated below:

```
rsubmit;
  data two;
  set one;
  where j_class in ('Full Time', 'Flex Time');
run;
```

```
*rsubmit;
  proc sort data = two;
  by id;
run;
```

```
*rsubmit;
  proc sort data = three;
  by id;
run;
```

```
*rsubmit;
  data compensation;
  merge two three;
  by id;
run;
```

```
*rsubmit;
  proc print data = compensation;
run;
```

This approach allows you to execute sections of code as you build a program for testing and to make changes to sections of code without the need to re-run all of the program or to insert and then remove the rsubmit statement.

If the second *rsubmit; for the print procedure was not commented, you would see the following errors and notes in the log:

```
ERROR: A link must be established by executing the SIGNON command before you can
communicate with UNKNOWN.
NOTE: Subsequent lines will be ignored until ENDRSUBMIT;
ERROR: Remote submit to UNKNOWN canceled.
NOTE: Remote submit to MYNODE complete.
```

Now an endrsubmit statement must be submitted in order to clear this error. Then you can continue to remotely submit code.

CONCLUSION

Using a Lazy Rsubmit makes developing and debugging code easier. There is no need to remember to take out rsubmit statements from your code. This approach saves you from typing and removing text from your program. Flexibility is enhanced by allowing you to execute whatever sections of code you need. It has saved me time over the years and has not generated any processing errors on both UNIX and DEC/ALPHA systems. If you discover any glitches with this approach, please let me know.

ACKNOWLEDGMENTS

Thank you to Rajeeve Kaul who has given me many fine SAS tips.

CONTACT INFORMATION

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

Stephanie R. Thompson
The University of Memphis
211 Administration Bldg.
Memphis, TN 38152-3370
Work Phone: (901)678-5529
Fax: (901) 678-5138
Email: Stephanie.Thompson@memphis.edu
Web: <http://oir.memphis.edu>

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.