

## **Adapt to Change of Table Numbers in Deliverable with One Click**

Tse-Hua Shih, Ph. D., inVentiv Clinical Solutions, LLC., Baltimore, MD

Cliff Meng, Ph. D., inVentiv Clinical Solutions, LLC., Baltimore, MD

Xitao Fan, Ph. D., University of Virginia, Charlottesville, VA

Yuan-Chi Chang, M.A., inVentiv Clinical Solutions, LLC., Baltimore, MD

Eugenia Henry, Ph. D., inVentiv Clinical Solutions, LLC., Baltimore, MD

### **ABSTRACT**

In pharmaceutical industry, a Contract Research Organization (CRO) employs numerous programmers to work on individual tables that will eventually be concatenated into a master file of deliverable. In order to generate tables with table numbers in order, a Lead Programmer needs maintain a list of table numbers along with names of their source programs and thus specifies each SAS<sup>®</sup> program name correctly in a batch run. Table numbers most likely do not match with names of their source programs, because people often name SAS programs after various modules or programmers' IDs. Therefore, manually preparing for the sequential order of program names to be run in a batch is time-consuming. Utilizing introduction comments at the beginning of each program, the current paper introduces a macro that adaptively runs a batch job with correct sequence of table numbers. This macro helps the Lead Programmer adapt to clients' requests (e.g., changes of table numbers, inclusion or exclusions of certain tables, etc) in a timely manner. Most importantly, an idea of two-way or interactive programming is demonstrated here.

## MOTIVATION

It is common to run individual SAS programs by using a run-all macro. Because table numbers and SAS program names have no logical correlation, it is time-consuming for a project programmer to manually spell out SAS program names in a run-all SAS macro so that it can generate tables in a sequential order of table numbers. The current paper introduces an alternative run-all macro that saves such a manual matching work for the project programmer.

## RATIONALE

A regular run-all macro initiates individual SAS programs according to listed program names. Any CRO that implements a good programming policy will ask its programmers to spell out table numbers as introduction comments in the beginning of a SAS program. In the current paper, a novel run-all macro first retrieves table numbers from SAS programs, sort them in sequential order, and thus call SAS program names with the sorted order of table numbers.

### Sample SAS Program Appearance

```
*****
Program:      47_chem.sas
Programmer:   Jeremy Shih
Validator:    Dewey Wilson
Client:       XXX
Protocol:     XXX
Date:         23JUN2009
Production Path: C:\47_chem.sas
Purpose:      Table 13.4.1
*****;
```

## Code

```
%macro run_all (code);
/* Step 1: Retrieve Table & Listing Numbers */
FILENAME foo PIPE "dir &code\*.sas /B /S";
DATA rawpgm;
INFILE foo TRUNCOVER;
INPUT @1 FILENAME $256. ;
prgrnm= fileName;
INFILE BAR FILEVAR=FILENAME END=END TRUNCOVER;
DO WHILE (NOT END);
INPUT LINE $ 1-256 ;
OUTPUT;
END;
retain prgrnm ' ';
RUN;

data a;
  set rawpgm;
  if index (line, 'Purpose: ') > 0 ;
  table = tranwrd(line, "Purpose: ", ""); put table;
  table = trim(left(table));
  p=scan(prgrnm, -1, '\');
  proc sort;
  by line;
  run;

/* Step 2: Prepare the old list of unique table & listing numbers */
data list;
  set a;
  cat= SCAN(table,1, ' '); /* get the text before the first space */
  lst3=tranwrd(table, "Listing", ""); put lst3;
  lst4=tranwrd(lst3, "Table", ""); put lst4;
  lst4 = trim(left(lst4));
  lst5=tranwrd(upcase(lst4), "A", ".0001"); put lst5;
  lst6=tranwrd(upcase(lst5), "B", ".0002"); put lst6;
  lst7=tranwrd(upcase(lst6), "C", ".0003"); put lst7;
  lst8=tranwrd(upcase(lst7), "D", ".0004"); put lst8;
  lst9=tranwrd(upcase(lst8), "E", ".0005"); put lst9;
  lst10=tranwrd(upcase(lst9), "F", ".0006"); put lst10;
  lst11=tranwrd(upcase(lst10), "G", ".0007"); put lst11;
  lst12=tranwrd(upcase(lst11), "H", ".0008"); put lst12;
  /* add as many as you need */

  array as[10] 8 a1-a10;
  do i = 1 to 10;
  as[i] = scan(lst12,i, '.|-');
  end;
  proc sort nodupkey;
  by cat a1-a10;
  run;

data _null_;
  set list;
```

```

call symput('val' || trim(left(_n_)), trim(left(p)));
call symput('lst' || trim(left(_n_)), trim(left(table)));
call symput('total', trim(left(_n_)));
run;

%do k=1 %to &total;
options NOCENTER pageno=1;
%include "&code\&&val&k";
%end;

%mend;
%run_all (%str(C:\code) );

/*location of SAS programs to be run*/

```

### Mechanism

The following table is truncated from dataset “list”. Table numbers have been sorted by A1 to A5, and the variable “PRGRMNM” let SAS to be run in a sequential order of sorted table numbers.

| PRGRMNM                  | TABLE              | CAT     | A1 | A2 | A3 | A4 | A5 |
|--------------------------|--------------------|---------|----|----|----|----|----|
| C:\code\47_l_conc.sas    | Listing 16.2.5-1   | Listing | 16 | 2  | 5  | 1  |    |
| C:\code\47_l_conc_e.sas  | Listing 16.2.5-2   | Listing | 16 | 2  | 5  | 2  |    |
| C:\code\47_l_pk_lead.sas | Listing 16.2.6.2-1 | Listing | 16 | 2  | 6  | 2  | 1  |
| C:\code\47_l_pk.sas      | Listing 16.2.6.2-2 | Listing | 16 | 2  | 6  | 2  | 2  |
| C:\code\47_l_pk_e.sas    | Listing 16.2.6.2-4 | Listing | 16 | 2  | 6  | 2  | 4  |
| C:\code\47_t_conc.sas    | Table 14.2.2-1     | Table   | 14 | 2  | 2  | 1  |    |
| C:\code\47_t_pk.sas      | Table 14.2.2-2     | Table   | 14 | 2  | 2  | 2  |    |
| C:\code\47_t_dose.sas    | Table 14.2.2-4     | Table   | 14 | 2  | 2  | 4  |    |
| C:\code\47_t_conc_e.sas  | Table 14.2.2-5     | Table   | 14 | 2  | 2  | 5  |    |

## **LIMITATION**

Currently, this run-all macro in the current paper cannot be applied to the situation where any run SAS program is written to generate tables with no sequential order of table numbers. In other words, each SAS program to be run in a batch should be designed to generate one table or multiple tables that are already in sequential order of table numbers.

## **CONCLUSION**

Talents are many. Ideas are few. The current paper emphasizes the idea of a two-way or interactive programming, instead of a one-way programming. Conventionally, SAS does exactly what has been spelled out in a program. In a one-way avenue, a command goes out without returning. However, in a two-way avenue, a command goes out and returns with pre-arranged evidences from SAS programs and then goes out again to function according to the reasoning of those retrieved evidences. Just like humans use their perceptive power to reason based on evidences they see, the idea of two-way programming helps SAS proceeds as an artificial intelligence, while the conventional one-way programming makes SAS proceeds like a robot. Of course, being human is a personal choice.

## **CONTACT INFORMATION**

Your comments and questions can be directed to:

Shih Tse-Hua (Jeremy)

Email: [jshih@inventivclinical.com](mailto:jshih@inventivclinical.com)

## **TRADEMARK INFORMATION**

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.