The Analysis System:
How a research team can organize and analyze multiple datasets with multiple outcomes measured multiple times

Kathryn Benton, University of Colorado at Denver and Health Sciences Center, Colorado Health Outcomes Program, Aurora, CO
Diane Fairclough, University of Colorado at Denver and Health Sciences Center, Colorado Health Outcomes Program, Aurora, CO
Brenda Beaty, University of Colorado at Denver and Health Sciences Center, Colorado Health Outcomes Program, Aurora, CO

ABSTRACT

Consider a randomized controlled trial, with approximately 400 subjects, spread across three years and multiple sites. During the course of the study, each patient was intended to receive six treatments and four assessments. A number of questions were asked about the patients' physical symptoms and quality of life at each of their visits, and if they survived to the end of the study, 43 surveys on 15 forms should have been completed. A study of this complexity required a large number of investigators, doctors, nurses, research assistants, and analysts. The richness of the datasets provided for a vast spectrum of hypothesis generation, which, in turn, required a significant amount of planning both at implementation and throughout the duration of the study. How can a research team create a system to allow multiple users to access and analyze the data, and at the same time maintain organization and uniformity in data manipulation, variable creation, and SAS™ program use and generation? Moreover, how do multiple users share data across a secure drive without having to alter root directories? Furthermore, how can we conduct analyses of multiple outcomes with repeated measures without having to recreate the same code over and over?

We have created a system for addressing all of these aforementioned problems which relies heavily on SAS macros. Tasks ranging from reading in the raw data, identifying the user, and running repeated measures for twenty outcomes performed with one simple click of the "run" button. This system allowed multiple analysts to use a copious and elucidative data set to its full potential.

INTRODUCTION

This system was created for the REST study (Reducing End-of-Life Symptoms with Touch). It was a multi-site, three year, randomized controlled trial of touch therapy, either "moving touch" (massage therapy) or "non-moving touch" (active control group) for pain relief in terminal cancer patients. The study was funded by the National Center for Complementary and Alternative Medicine at the National Institutes of Health.

READING IN THE RAW DATA

For the REST study, data entry was conducted by an outside source. Data files were then sent to us in batches, typically via compact disc, about every three months. The data for all fifteen forms were submitted to us in text (ASCII) format throughout the duration of the three year study. The ASCII file names indicated both the form number and the batch number which ranged from 1 to 12. Each file was formatted and converted to a permanent SAS data set using a separate macro program for each of the 15 forms. These files were then combined using one conversion macro. The conversion macro prompted the user to identify him/herself (using the root macro, which will be discussed later) and how many batches of ASCII files were currently available. Therefore, one would only need to run the single conversion program (Convert_ALL.sas) and indicate the most recent iteration of the data, in order to update the SAS library with the most recent submission.

Convert_ALL.sas performed the following steps:

- Runs a program that assigns appropriate libnames, depending on the user
- Requests the user to provide the latest ‘batch’ number of all the data forms
- Runs another SAS program, Macro.sas to compile macros used later
- Runs an input program for each of the 15 data forms. We'll be looking at Form 1, so the program we'll look at is IN_Form1.sas. All the other forms have analogous programs.

Let's look at it piece by piece.

- Runs a program that assigns the appropriate libnames, depending on the user:
The first thing Convert_ALL.sas does is call a SAS program, Root_REST.sas. This allows different users to have different drive mappings for the same data.

%include 'C:\SAS Macros\Root_REST.sas'; *Identifies the root directory for each user*
libname library "&root\REST\SASlib";
libname sasdata "&root\REST\SASlib";
libname source "&root\REST\Source"

When accessing data on a secure server from different computers, each computer might have the same data source mapped differently. For example (below), user ABC’s drive is mapped to Z:\ and user DEF’s computer is mapped to G:\. The root macro has been saved in each individual users’ C:\SAS Macros’ folder (since this will always indicate the hard drive).

Let’s look at the Root_REST.sas program:

%global ABC DEF Root;

%window interface1 color=yellow

#5 @25 "USER IDENTITY"
#7 @25 "Please type in 'Y' after your identity"
#10 @25 'Person ABC' ABC 1 ATTR=UNDERLINE REQUIRED=NO
#11 @25 'Person DEF' DEF 1 ATTR=UNDERLINE REQUIRED=NO
#24 @25 "Please Press ENTER";

%display interface1;

First, SAS displays a screen that prompts the user to identify him/herself by typing a “Y” next to his/her name.

Then, the macro ‘Root’ assigns the drive letter value to the macro variable ‘root’ depending on who is accessing the data.

%macro Root:
  %if %upcase(&ABC)=Y %then %do; *Root for person ABC*;
   %let root=Z:;
   %end;
  %if %upcase(&DEF)=Y %then %do; *Root for person DEF*;
   %let root=G:;
   %end;
%mend root;

%root;

The following statements then automatically map the libname statements to the correct drive.

libname library "&root\REST\SASlib";
libname sasdata "&root\REST\SASlib";
libname source "&root\REST\Source;"

• Requests the user to provide the 'batch' number of all the data forms

%global last;
%window interface color=yellow
#9 @20 "REST "
#10 @25 "Generates SAS Permanent Dataset from ASCII Files"
#13 @20 '# ASCII Files: ' last 2 ATTR=UNDERLINE REQUIRED=YES
#14 @25 "If less than 10, please use one digit instead of two."
#15 @25 "For example : 5 "
#16 @25 "Please do not use 05"
#25 @20 "Please Press ENTER";
%display interface;

This generates an interface that requests the user to provide the latest batch of the data source into a macro variable called "last." Now all of the other programs called will be run iteratively up to the number represented by "last." Note that "last" is created as a global macro variable so it can be used across programs and macros.

• Runs another SAS program, Macro.sas to compile macros used later:

%include "&root\REST\SAS programs\Convert ASCII\Macro.sas";
* Macros %Source, %RunInTxt, and %ListDSN                  *
* %Source generates infile statement                *
* %RunInTxt repeatedly runs %Intxt for each batch file*
* %ListDSN lists dataset names                      *

Macro.sas compiles the macros 'Source', 'RunInTxt', and 'ListDSN.' Note that this program contains no calls to these macros, but simply compiles them to be ready for use. These macros could have been included in each of the text conversion programs, but it was more efficient to have them stored separately and simply called from each of the other programs. Let's look at each of these macros separately.

The 'source' macro creates the infile statements for the 15 forms. The only difference in the infile statements is how the macro variable is coded. If the batch number is less than ten, then the placeholder will have two zeros in front of it (this reflects how the raw data were saved in the text files), and if the number is greater than ten, then the placeholder only has one zero.

%macro source;
%if &no < 10 %then %do;
  infile "&root\REST\Source\ASCII Files\R&NumDSN._00&no..txt" missover lrecl=500;
%end;
%if &no >= 10 and &no < 100 %then %do;
  infile "&root\REST\Source\ASCII Files\R&NumDSN._0&no..txt" missover lrecl=500;
%end;
%mend source;

Next, the RunInTxt macro is compiled. This macro will call the 'Intxt' macro (located in another SAS program we'll see a bit later) repeatedly in order to generate a SAS datasets for each batch of the text files (ultimately, 12 times).

*%RunInTxt repeatedly runs %Intxt which generates dataset for each TXT file*;

%macro runIntxt;
  %do i=1 %to &last;
    %Intxt(\i);
  %end;
%mend runIntxt;

The 'ListDSN' macro simply creates a list of datasets that will be used in a SET statement later.

*%ListDSN lists dataset names*;
%macro listdsn(DSN);
  %do i=1 %to &last;
    REST_&DSN._&i
  %end;
%mend listdsn;
**Runs an input program specific to each of the 15 data forms.** We'll be looking at Form 1, so the program we'll look at is IN_Form1.sas.

Here, we are ultimately creating a dataset called sasdata.REST_F1. Note that the only variable 'fed' to this macro is the number of batches, 'no.'

```sas
%let dsn=F1; *Defines the dataset name*
%let NumDSN=01; *Defines the order of the form*
%let first=1;
%macro Intxt (no);
data rest_&_dsn._&no;
  *Generates Infile statement*;
  %SOURCE is stored in MACRO.sas*;
  %source;
  informat DATE MMDDYY8.;
  input ID 1-5 INIT $ 6-8 VISIT 9 FORM 10-11 DateMM 12-13 DateDD 14-15 DateCC 16-17
  DateYY 18-19 F1Q2a 20 F1Q2b 21 F1Q2c 22 F1Q2d 23 F1Q3a 24 F1Q3b 25 F1Q4a 26 F1Q4b 27
  F1Q4c 28 F1Q4d 29 F1Q4d1...;

  ***Indicator for batch file***;
  Batch=&no;
  label ID='Patient ID'
  INIT='Patients Initials'
  VISIT='Visit No.'
  FORM='Form No.'...

  format DATE Enrolldt date7. F1Q2a F1q2b F1Q2c F1Q2d yesno. F1Q3a F1Q3b F1Q3c
  logicalU.;
run;

proc sort data=rest_&_dsn._&no;
  by id visit;
run;
%mend Intxt;

*---------------------------------------------*;
*Generate a series of datasets for batch files*;
*---------------------------------------------*;

* %RunInTxt repeatedly runs %Intxt which generates dataset for each batch file*;
%runInTxt;

*-------------------------------------------*;
*Generates SAS dataset by merging all       *;
*datasets of batch files                    *;
*-------------------------------------------*;

data sasdata.REST_&_dsn;
  set %listdsn(&dsn); *%ListDSN lists dataset names*;
  by id visit;
run;

REPEATED MEASURES MACRO

In the REST study, patients were followed until all six treatments and four assessments were received, or until their time of death or dropout. There were twenty outcomes of interest ranging from specific bodily pain, mood, respiratory rate, and perceived quality of life. To prevent repetition of SAS code, we created a macro that would run a repeated measures mixed model analysis for all of our outcomes sequentially. The procedures are broken down below.

First we created two datasets called “REST_Covar1” with our centered covariates, which were included in all of our mixed models by way of a macro variable, &covars.
proc sql;  * Reads Covariates from Form 1 *
create table REST_Covar as
select ID,
    (age - mean(age)) as CAge label='Age Centered',
    (comorbid - mean(comorbid)) as CComorbid label='# Comorbid Cndtn Cntrd',
    (F1Q7 - mean(F1Q7)) as CSex label='Sex Centered',
    (F1Q18a - mean(min(F1Q18a,2))) as CPrevM label='No PrevMassage Centered',
    (F1Q19e - mean(F1Q19e)) as CWPainWk label='Worst Pain Last Week Centered'
from work.REST_F1;
quit;

proc sql;  * Reads Covariates from Form 2 *
create table REST_Covar2 as
select ID,
    (F5AR - mean(F5AR)) as CKarnofsky label='Karnofsky Functional Status'
from work.rest_f5 where visit eq 1;
quit;

%let covars=CAge CComorbid CSex CPrevM CWPainWk CKarnofsky;

We then combined the covariates with repeated observations.

*** Merge repeated assessment data by visit ***;
data work.assess;
    merge work.rest_f6(Keep=id visit BPIWorst BPIMean)
        work.rest_f8(Keep=id visit MQOLPhys MQOLPsych)
        work.rest_f7(Keep=id visit MSAS1-MSAS5)
        work.rest_f9( Keep=id visit analgesic);
    by id visit;
run;

*** Merge Treatment Indicator and Covariates ***;
data work.subject;
    merge  work.REST_F2(keep=id treat)
          work.REST_Covar
          work.REST_Covar2;
    by id;
run;

*** Merge Covariates with repeated assessments ***;
data work.assessment;
    merge work.subject work.assess;
run;

Next, we called the repeated measures macro. Two separate macros were created for treatments and assessments, because there were four assessments and six treatments, so the estimate statements had to be specified differently. Only the assessments program will be discussed (the syntax was similar for both). In the mixed macro, each outcome was fit to a repeated measures mixed model with treatment group (treat) and visit being the primary variables of interest, and '&covars' being variables we wanted to control for based on a priori knowledge.

%include "&ROOT\REST\SASMacros\RM_macros.sas"; *Includes macro*;

Mixed Model Macro:

%macro mixed4(no,outcome,label,covars);
proc mixed data=work.assessment;
class Treat visit id;
model &outcome = Treat*visit &covars/ noint solution;
repeated visit / subject=id type=un r=3 rcorr=3;
estimate 'Visit 1, MT vs NMT' Treat*visit 1 0 0 0 -1 0 0 0;
estimate 'Visit 2, MT vs NMT' Treat*visit 0 1 0 0 0 -1 0 0 0;
estimate 'Visit 3, MT vs NMT' Treat*visit 0 0 1 0 0 0 -1 0 0 0;
run;
%mend mixed4;
estimate 'Visit 4, MT vs NMT' Treat*visit 0 0 0 1
                                      0 0 0 -1;
estimate 'BL vs FU, MT ' Treat*visit -3 1 1 1
                                      0 0 0 0 / divisor=3;
estimate 'BL vs FU, NMT' Treat*visit 0 0 0 0
                                      -3 1 1 1 / divisor=3;
estimate 'BL vs FU, MT vs NMT' Treat*visit -3 1 1 1
                                      3 -1 -1 -1/divisor=3;
format visit 2.;
ods output Estimates=work.Est&no SolutionF=work.Beta&no  
        Covparms=work.cov&no;
runc
%mend

Returning to the analysis program, we then ran the mixed macro on each of our outcomes by indicating the (arbitrary) number
of the variable, the variable name, its label, and the &covars macro variable we created previously. So, by creating this macro,
we saved ourselves hundreds of lines of code in our SAS program.

%mixed4(1, var1, Variable One,&covars)
%mixed4(2, var2, Variable Two,&covars)
%mixed4(3, var3, Variable Three,&covars)
.
.
%mixed4(20, var20, Variable Twenty,&covars);
quit;

All of the output was then written to a rich text document and saved to the share drive.

ods rtf file="&root\Rest\Reports\Repeated Measures\RMAssess.rtf"

CONCLUSION

Although this system catered specifically to the needs of the REST study, it is malleable, and it could be useful for any study
that requires the accommodation of multiple users at multiple sites, repeated data collection times, and numerous outcomes of
interest. Although timely at the initiation of data management and analysis, this system saves everyone a lot of time at the end
of the study, when the data are needed most. It also maintains impressive organization of the data, programs, and reports.

ACKNOWLEDGMENTS

We would like to acknowledge Dongmei Pan for creating the root macro.

CONTACT INFORMATION

Kathryn Benton
University of Colorado at Denver and Health Sciences Center
Colorado Health Outcomes Program
12477 East 19th Avenue, MS F443
P.O. Box 6508
Aurora, CO 80045
Work Phone: 303-724-1371
Fax: 303-724-1839
E-mail: Kathryn.Benton@UCHSC.edu
Web: www.uchsc.edu/coho

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.