

A Dynamic Way to Manipulate Longitudinal Data with SAS®

Nora H. Ruel, Arthur X. Li,

City of Hope Comprehensive Cancer Center, Duarte, CA

ABSTRACT

This paper offers a solution to managing and manipulating a large longitudinal data set, making clever use of the CONTENTS procedure, macro variables created from the CALL SYMPUT routine and RETAIN statements.

INTRODUCTION

In the clinical world, we often encounter patient data with repeated measurements over time. This type of data often includes multiple records per subject, with dozens or even hundreds of variables collected at various time points. For example, a patient may have multiple measurements of weight, blood pressure, total cholesterol, or blood glucose level over several medical visits, but may not have all these values recorded every time. Researchers may be interested in a database depicting the most recent available information on their patients. Such a task of reducing a large dataset to a single observation per subject would involve selecting values from different time points. We generalize this problem and demonstrate how to solve it dynamically.

SMALL DATASET - HARDCODED SOLUTION

The following is a small dataset containing multiple observations for each of 4 patients, with a patient ID and date of visit, and several clinical indicators. It is common to have only some of the variables populated and to have lots of missing data.

PATID	VISIT	GLUC	TGL	HDL	LDL	HRT	MAMM	SMOKE
01	2005	88	.	32	99	Y		Y
01	2007	.	150	60	.		N	N
02	2004	110	.	.	120	N		
02	2005	.	200	65	165		Y	N
02	2006	90	210	.	150	Y		N
03	2005	88	.	32	210		Y	Y
04	2002	120	164	.	.	Y	Y	
04	2004	110	170	70	188			Y
04	2006	.	190	.	190	N	N	
04	2007	90	.	75	.		Y	N

The desired resulting dataset will have the most current nonmissing value of each clinical indicator for each individual. With this small dataset, we could easily hardcode this solution, by first sorting the dataset by the PatID variable in descending order of visit date. We then create an array for numeric and one for character variables, so we may retain the most recent nonempty value for each variable. At the end of the data step we keep only the last row of data for each ID, which will have a collection of all current values of all variables.

```
proc sort data=small;
  by PatID descending visit;
run;

data small_sol (keep=PatID visit GLUC_k TGL_k HDL_k LDL_k HRT_k MAMM_k SMOK_k);
  set small;
  by PatID descending visit;
  array keepchar[3] $ HRT_k MAMM_k SMOK_k;
  array keepnum[4] GLUC_k TGL_k HDL_k LDL_k;
```

```

array charvar[3] HRT MAMM SMOK;
array numvar[4] GLUC TGL HDL LDL;
retain HRT_k MAMM_k SMOK_k GLUC_k TGL_k HDL_k LDL_k;
do i = 1 to 3;
  *** assign values to character variables ***;
  if first.PatID then keepchar(i) = charvar(i); else
    if keepchar(i) eq '' then keepchar(i) = charvar(i);
end;
do i = 1 to 4;
  *** assign values to numeric variables ***;
  if first.PatID then keepnum(i) = numvar(i); else
    if keepnum(i) eq . then keepnum(i) = numvar(i);
end;
if last.PatID;
run;

```

Here, we have created a separate array for numeric and character variables, allowing us to loop through them in search of the nonmissing values. In addition, we have created a new variable with the suffix `_k`, which will be the final desired value that is kept of each variable.

DYNAMIC SOLUTION FOR LARGE DATASETS

To create a more dynamic solution we make use PROC CONTENTS in order to obtain the names and attributes of the variables. Variables that are used for unique ID and visit date are then deleted from the contents, and the variable attributes (character or numeric) are used to add individual variables to the list of character or numeric variables lists, respectively. CALL SYMPUT is used to create a macro variable containing these lists. The macro also accounts for datasets which may contain all numeric variables or all character variables, and checks for existence of each type within the contents.

```

%macro Dynamic_Sol(large, PatID, Visit, size=300);

proc contents data=&large noprint out=data_contents;
proc sort data=data_contents;
  by varnum;
run;
data _null_;
  set data_contents end=endfile;
  retain characters numerics characters_keep numerics_keep rename_statement;
  length characters numerics characters_keep numerics_keep rename_statement $
&size;

  *** remove our identifying variables id and sequence ***;
  Where name not in("&PatID", "&Visit");

  *** create 'keeper' variable names ***;
  name_k=trim(name)||'_k' ;

  *** create rename statement to return to original names later ***;
  rename_statement=trim(rename_statement)||' '||trim(name_k)||'='||trim(name);

  *** create lists of all numeric and character variables ***;
  if informat='$' then characters=trim(characters)||' '||name;
  else numerics=trim(numerics)||' '||name;

  if informat='$' then characters_keep=trim(characters_keep)||' '||name_k;
  else numerics_keep=trim(numerics_keep)||' '||name_k;

  if endfile then do;
    call symput('characters',characters);
    call symput('numerics',numerics);
    call symput('characters_keep',characters_keep);
    call symput('numerics_keep',numerics_keep);
    call symput('rename_statement',rename_statement);
  end;
run;

```

```

proc sort data=&large;
  by &PatID descending &Visit;
run;

data dynamic_sol (keep= &PatID &Visit &characters_keep &numerics_keep
                  rename=( &rename_statement ));
  set &large;
  by &PatID descending &Visit;
  retain LastVisit &numerics_keep &characters_keep;

  *** maintain the most recent observation date ***;
  if first.&PatID then LastVisit = &Visit;

  *** if there is ≥ 1 numeric variable then do this otherwise skip it ***;
  %if &numerics_keep gt '' %then %do;
    array keepnum[*] &numerics_keep ;
    array numvar[*] &numerics ;
    do i = 1 to dim(numvar);
      if first.&PatID then keepnum(i) = numvar(i); else
        if keepnum(i) eq . then keepnum(i) = numvar(i);
      end;
    %end;
  *** if there is ≥ 1 character variable then do this otherwise skip it ***;
  %if &characters_keep gt '' %then %do;
    array keepchar[*] $ &characters_keep ;
    array charvar[*] &characters ;
    do i = 1 to dim(charvar);
      if first.&PatID then keepchar(i) = charvar(i); else
        if keepchar(i) eq '' then keepchar(i) = charvar(i);
      end;
    %end;

  if last.&PatID;
  &Visit = LastVisit;

run;

%mend;

```

RESULTS FROM MACRO

The resulting dataset from this macro will be as follows, with the original variable names, and most recent values for all variables.

PATID	VISIT	GLUC	TGL	HDL	LDL	HRT	MAMM	SMOKE
01	2007	88	150	60	99	Y	N	N
02	2006	90	210	65	150	Y	Y	N
03	2005	88	.	32	210		Y	Y
04	2007	90	190	75	190	N	Y	N

ADDITIONAL USES

We can further use these tools to add summaries to our dataset. For instance, PROC MEANS can be used with the CLASS variable to group means of numeric variables by ID, and merged with the resulting dataset to not only have access to the most recent value of a variable, but other relevant statistics as well.

CONCLUSION

There are many different ways to deal with longitudinal data, and here we presented one way to handle a large number of variables and observations without manually typing difficult variable names. The methods presented here can be adjusted easily to use different datasets and methods of analysis.

CONTACT INFORMATION

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

Nora H. Ruel or Arthur X. Li
City of Hope
1500 East Duarte Rd.
Duarte, CA 91010
Work Phone: (626) 256-4673
Fax: (626) 471-7106
E-mail: nruel@coh.org, xueli@coh.org

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.