How to Import a Text File with a Large Number of Variables But The Layout Stored in a Separate File?

Steven Yan, Mu Hu
Database Marketing, Zale Corporation

Abstract:

Proc import, Import wizard, and the Data Step are useful tools to convert text files into SAS[i] data sets. However, when it comes to read a text file with a large number of variables but the layout detailed in a separate file, all these tools have limitations. The Proc Import and the Import Wizard won't be able to read the variable names because they were stored in a separate file; while the data step has better control in this aspect, it is hard for SAS programmers to hard code and to update hundreds even thousands of variables, which oftentimes resulted in human errors. This paper presents a method that will effectively handle this challenge.

Introduction

Oftentimes, SAS programmers have to import a text file with a large number of variables. The text file contains data only, while the file layouts were detailed in a separate CSV or Excel file (see sample files below).

Sample Date File

```
100001Apple       Trees     AM12/01/78WM1123 AppleNot Exist Street…
100002Banana      Cake      BF10/21/82BS2555 Banana Avenue …
100003Cherry      Pie       CF02/14/45WM3MailStation 101…
100004Orange      Juice     OM07/04/51HM1999 Vita Lane …
100005Strawberry  Icecream  SF12/25/36OS3321 Walnut Hill…
```

Sample File Layout

VARIABLE	DESCRIPTION	FORMAT	LENGTH	START	END
CUST_ID	Customer ID	Number	6	1	6
FIRST_NAME	First name	Character	20	7	26
LAST_NAME	Last name	Character	20	27	46
INIT	Middle initial	Character	1	47	47
GENDER	Gender code	Character	1	48	48
	M: Male				
	F: Female				
BIRTHDATE	Date of Birth	MM/DD/YY	8	49	56
ETHNICITY	Ethnicity code	Character	1	57	57
	B: Black				
	H: Hispanic				
	W: White				
	O: OTHER				
	I: American Indian				

MARITAL	Marital status code M: Married S: Single	Character	1	58	58
EDUCATION	Level of education 1: Complete High School 2: Complete College 3: Complete Graduate School	Number	1	59	59
ADDRESS1	First line of the street address	Character	30	60	89
ADDRESS2	Second line of the street address	Character	30	90	119
CITY	City	Character	20	120	139
STATE	State Code	Character	2	140	141
ZIPCODE	Zip Code	Character	5	142	146
HOME_TYPE	Home owner/renter O: Owner R: Renter	Character	1	147	147
HOME_VALUE	Home market value A: $1 - $49,999 B: $50,000 - $99,999 C: $100,000 - $149,999 D: $150,000 - $199,999 E: $200,000 - $399,999 F: $400,000 - $499,999 G: $500,000 - $749,999 H: $750,000 - $999,999 R: $1,000,000 PLUS	Character	1	148	148

Proc Import and the Import Wizard will not be able to import the variable names directly because they were stored in a separate file. Proc Import will have to either use the first line as variable names or to create dummy variables for the file. Another problem is that the Proc Import may not properly assign attributes such as the variable length and format.

The Data Step with infile and input statement may provide better control in this aspect. However, hard coding hundreds or even thousands of variables, if not totally impossible, is very tedious and time consuming, increasing the possibility of human errors. Also the contents of the dataset may change from time to time, making it even more difficult for SAS programmers to update and to maintain the codes.

Fortunately, there is a way to solve the problem.

The Solution:

The data step plus the Select into: Clause of Proc SQL

If the conventional data step is used to import a text file, the code will appear like the following:

```
data sampleData;
```

```
infile "&projPath\SampleData.txt" missover lrecl=148 pad;
input Cust_ID $ 1-6 First_Name $ 7-26  Last_Name $ 27-46
Init $ 47-47 Gender $ 48-48 Birthday $  49-56  Ethnicity $ 57-57
Marital $ 58-58  Education $ 59-59 Address1 $ 60-89 Address2 $ 90-119 $
City $ 120-139 State $ 140-141 Zipcode $ 142-146 Home_type $ 147-147
Home_value $ 148-148…;
run;
```

To avoid as much keystroke as possible, the idea is to use the **Select into: Clause of Proc SQL** to host the macro variables for both variable attributes and also for variable labels. The revised code looks simple and clean:

```
data SampleData;
   infile "&projPath\SampleData.txt" lrecl=&lrecl pad missover;
   input  &varRead;
   format &varINFMT;
   label  &varLBL;
   run;
```

The beauty of this piece of code is that it will not only ease the pain of typing, but more importantly, will warrant the accuracy of the data. Also, literally speaking, it is maintenance free.

Below is the complete SAS code with comments.

```
*=====================================================================*
 * Program:
 * Purpose:
 * Note(s):
 * History:
 * DD/MM/YY
*=====================================================================*;

options ps=62 ls=145 FORMCHAR="|----|+|---+=|-/\<>*";

%let projPath=c:\SAS Paper;

*---------------------------------------------------------------------;
*Import the file layout                                               ;
*---------------------------------------------------------------------;
proc import datafile="&projPath\SampleDataLayout.xls" out=temp replace;
   run;

*---------------------------------------------------------------------;
*The original formats may vary from file to file.
*The goal of this data step is to convert them into SAS data informat
*---------------------------------------------------------------------;

data varINFMT;
   set temp(where=(variable ne ''));
   format=upcase(format);
if format=:'CH' then sasFMT='$'||compress(put(end-start+1,8.))||'.';
else if format=:'NUM' then sasFMT=compress(put(end-start+1,8.))||'.';
```

```
else if format='MM/DD/YY' then sasFMT='mmddyy'||compress(put(end-
start+1,8.))||'.';
   run;

*-------------------------------------------------------------------------;
* Use Select Into: of Proc SQL to host variable attributes, Formats,
* and Labels
*-------------------------------------------------------------------------;

proc sql;

  select '@'||compress(put(start,8.))||' '||trim(variable)||'
'||compress(sasFMT)
   into :varRead separated by ' '
   from varINFMT
   ;

   select trim(variable)||' '||compress(sasFMT)
   into :varINFMT separated by ' '
   from varINFMT
   ;

   select trim(variable)||'='||"'"||trim(description)||"'"
   into :varLBL separated by ' '
   from varINFMT
   ;

   select max(end)
   into :lrecl
   from varINFMT;

   quit;

*-------------------------------------------------------------------------;
* Import the data file
*-------------------------------------------------------------------------;

data SampleData;
   infile "&projPath\SampleData.txt" lrecl=&lrecl pad missover;
   input &varRead;
   format &varINFMT;
   label &varLBL;
   run;

*-------------------------------------------------------------------------;
* This part is for validation purpose, you may delete it.            ;
*-------------------------------------------------------------------------;
Proc contents data=sampleData; run;
```

Author Contact Information

Mu Hu
Database Marketing,
Zale Corporation, Irving, Texas
Email: mhu@zalecorp.com

Steven Yan
Database Marketing,
Zale Corporation, Irving, Texas
Email: syan@zalecorp.com

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.