

Paper 4923-2020

Reading in a Comma Delimited File with a Data Dictionary

Kalyani Telu Henry Jackson Foundation

ABSTRACT

Comma-separated values files (CSV) are one of the most common files that SAS programmers import to create SAS datasets. When using PROC IMPORT, user defined formats and labels will be lost and the programmer needs to establish informats, formats and labels to each variable. This process is typically done manually, and it is highly time consuming if the dataset has many variables. This process can be automated through storing meta data in an excel file and writing a simple SAS program to combine the meta data with the raw data. A dynamic approach has been developed for reading large raw CSV files using simple data step statements understandable to beginner programmers.

INTRODUCTION

Data comes from a wide variety of sources with different format types and it is typical practice to use either PROC IMPORT or INFILE- INPUT statements to read such files. One main disadvantage for using the PROC IMPORT is that all the user defined formats and labels will be lost, which are useful when generating tables and graphs. In addition, SAS determines attributes of variables as either character or numeric and lengths of the fields depending on the first few observations. There is a possibility of incorrect assumptions by SAS when the first few observations have numeric data and later observations have non-numeric data. To overcome this problem, the programmer must assign attributes by writing a line listing of each single variable using INFILE-INPUT statements and later formats and labels. The task becomes problematic when there are hundreds of variables.

MANUAL PROGRAMMING TO READ CSV FILES

```
data demog_med;
infile 'location of the file- file name' dlm=',';
input @1 id $6.
      @7 gender $2.
      @9 age 3.
      @12 origin 3.
      . . . . . ;
label id='Studyid'
      gender='Gender'
      age='Age (in Years)'
      origin='Country of Origin'
      . . . . . ;
format gender gender.
      origin countryf.
      . . . . . ;
run;
```

One solution is to store meta data containing the list of variable names, lengths of the variables, labels, formats in a separate excel file, and use a simple code to combine the meta data with the raw data in the CSV file. Adding, removing or editing variables and updating meta data is easier when the entire meta data is stored in one single file. A step

by step approach to write a simple program that combines the external data structure with the related raw CSV file is discussed in this paper.

READ THE META DATA STORED IN THE EXTERNAL FILE

The first step is to create a meta data file that contains the variable names, labels, data types, lengths and formats and then import this data structure into SAS that creates the data set called 'Layout'.

VIEWTABLE: Work.Layout					
	Variable	Label	data_type	data_length	format
1	id	Study ID	Character	6	
2	gender	Gender	Character	2	\$Genderf.
3	age	Age in Years	Numeric	3	
4	origin	Country of Origin	Numeric	3	Countryf.
5	race	Race	Numeric	2	racef.
6	drug	Medication	Character	50	
7	start_date	Start date of the Medication	Date	10	date9.
8	end_date	End date of the Medication	Date	10	date9.
9	dose	Dose of the Medication (mg/day)	Numeric	8	
10	symptoms	Symptoms experienced	Character	100	
11	seek_care	Did patient seek hospital cate?	Numeric	2	yesno.
12	days_hospital	Number of days hospitalized	Numeric	8	

PROGRAM THAT GENERATES ANOTHER PROGRAM

Once the data structure is ready, the next step is to combine this meta data information with the raw CSV file. This is accomplished in three phases by creating three different _NULL_ data sets.

1. Generate inputs
2. Generate informats
3. Generate formats and labels

In these three phases, the put statement is used to write another program to store the meta data which is explained below. Three derived programs will be generated at the end of these three phases.

1) Generate Inputs

In this phase, infile statement and inputs are programmed. The raw data example is 'Demographics_medication.csv'.

Here is an example of a raw data file (made up test data).

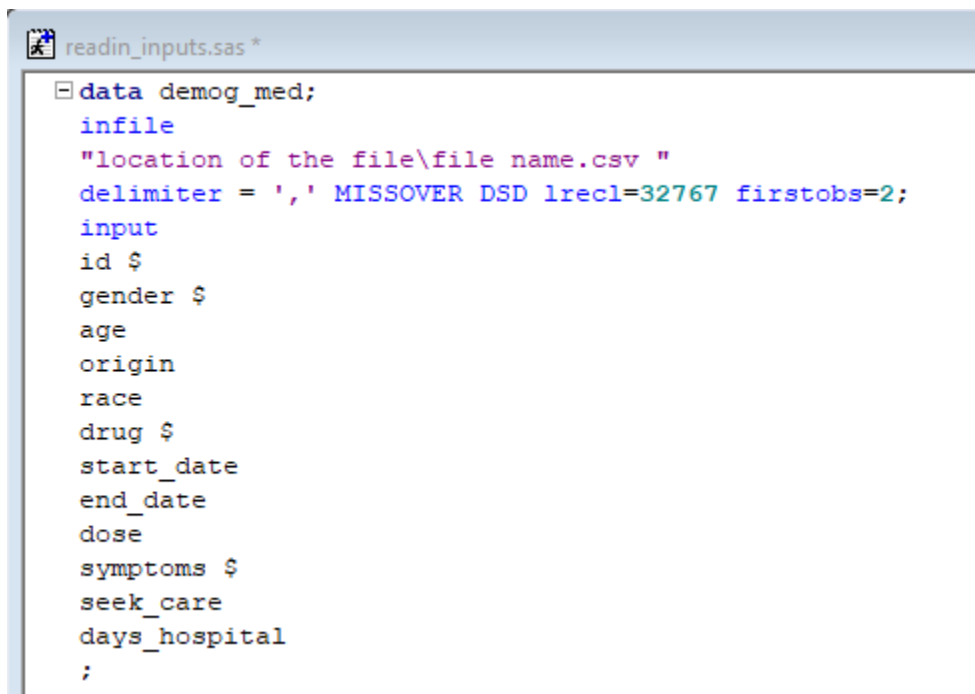
1	id	gender	age	origin	race	drug	start_date	end_date	dose	symptoms	seek_care	days_hospital
2	111111	F	20	1	1	Claritin	2/14/2019	3/14/2019	500	Headache	1	2
3	222222	M	25	4	2	Advil	1/2/2019	1/7/2019	100	Fever	0	0
4	333333	F	30	3	1	Advil	3/5/2019	4/5/2019	200	Fever	1	1

```

filename s_inputs "...\\readin_inputs.sas";
data _null_;
  set work.layout end=eof;
  file s_inputs;
  if _n_ = 1 then do;
    put "data demog_med;";
    put 'infile';
    put "'...'\\..\\Demogrpahics+Medication.csv'";
    put "delimiter = ',' MISSOVER DSD lrecl=32767 firstobs=2;";
    put "input";
    end;
  if upcase(data_type) = 'CHARACTER' then
    put variable '$';
  else put variable;
  if eof then do; put `';
  end;
run;

```

At the end of this first phase, a program is created: 'readin_inputs.sas'. Below is an example of this program. Note that the run statement is not coded here because the informat statement will be placed immediately after the input statement.



```

readin_inputs.sas *
data demog_med;
  infile
  "location of the file\\file name.csv "
  delimiter = ',' MISSOVER DSD lrecl=32767 firstobs=2;
  input
  id $
  gender $
  age
  origin
  race
  drug $
  start_date
  end_date
  dose
  symptoms $
  seek_care
  days_hospital
  ;

```

2) Generate informats

```

filename s_infmts "...\\readin_informats.sas";
data _null_;
  set work.layout end=eof;
  length = compress (catx ('', data_length, '.'));
  file s_infmts;
  if _n_ = 1 then do;
    put "infotmat";
    end;
  if upcase(data_type) = 'CHARACTER' then

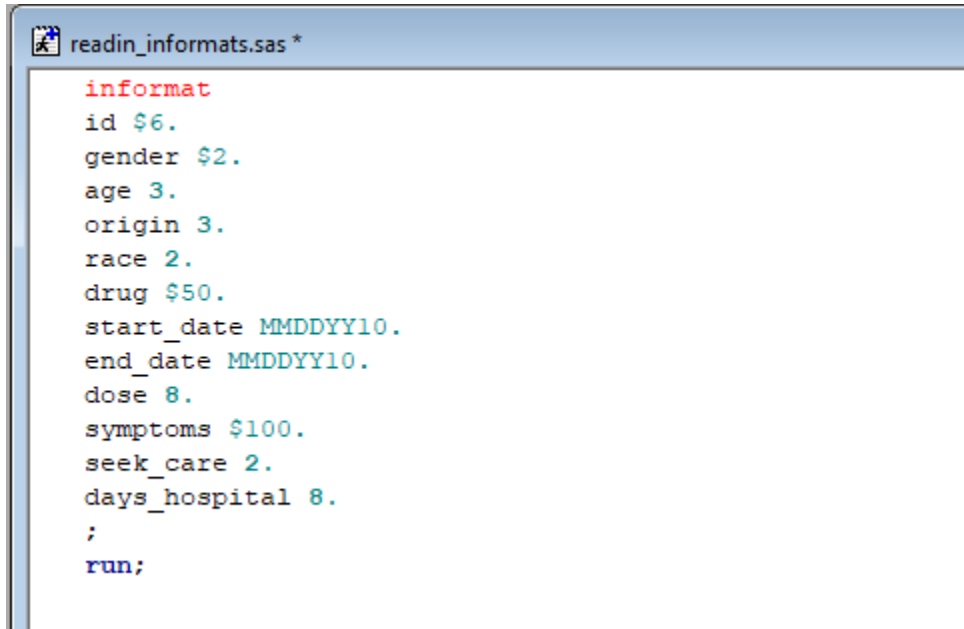
```

```

        put variable '$' length;
    else if upcase(data_type) = 'DATE' then
        put variable 'MMDDYY' length;
    else put variable length;
    if eof then do; put ';' ; put 'run;'; ebdl
run;

```

Below is an example of 'readin_informats.sas'.



```

informat
id $6.
gender $2.
age 3.
origin 3.
race 2.
drug $50.
start_date MMDDYY10.
end_date MMDDYY10.
dose 8.
symptoms $100.
seek_care 2.
days_hospital 8.
;
run;

```

3) Generate formats and labels

```

filename s_f_lbl "...\readin_formats_labels.sas";
data _null_;
    set work.layout end=eof;
    file s_f_lbl;
    if _n_ = 1 then do;
        put "proc datasets nolist;" /
            "modify demog_med;";
    end;
    *Generate format statement for each variable;
    Put " format " variable fomat ";";
    *Generate label statement for each variable;
    if indexc(lable,"'") = 0 then
        put " label " variable " = \: label "'";";
    else if index(label, '\') = 0 then
        put " label " variable \' = "' label \'';";
    if eof then put "quit;";
run;

```

Below is an example of 'readin_formats_labels.sas'.

```

readin_formats_labels.sas
proc datasets nolist;
  modify demog_med;
  format id ;
  label id = 'Study ID ';
  format gender $Genderf. ;
  label gender = 'Gender ';
  format age ;
  label age = 'Age in Years ';
  format origin Countryf. ;
  label origin = 'Country of Origin ';
  format race racef. ;
  label race = 'Race ';
  format drug ;
  label drug = 'Medication ';
  format start_date date9. ;
  label start_date = 'Start date of the Medication ';
  format end_date date9. ;
  label end_date = 'End date of the Medication ';
  format dose ;
  label dose = 'Dose of the Medication (mg/day) ';
  format symptoms ;
  label symptoms = 'Symptoms experienced ';
  format seek_care yesno. ;
  label seek_care = 'Did patient seek hospital care? ';
  format days_hospital ;
  label days_hospital = 'Number of days hospitalized ';
quit;

```

After completing the three phases, the final step is to include these programs for reading in the raw data.

```

%include "... \readin_inputs.sas";
%include "... \readin_informats.sas";
%include "... \readin_formats_labels.sas";

```

At the end of this program a data set called 'demog_med' is created, contents of this table show assignment of labels and fomrats.

	id	gender	age	origin	race	drug	start_date	end_date	dose	symptoms	seek_care	days_hospital
1	111111	Female	20	USA	Asian	Claritin	14FEB2019	14MAR2019	500	Headache	Yes	2
2	222222	Male	25	UK	White	Advil	02JAN2019	07JAN2019	100	Fever	No	0
3	333333	Female	30	Kenya	Asian	Advil	05MAR2019	05APR2019	200	Fever	Yes	1

VIEWTABLE: Work.Contents_demog_med			
	NAME	LABEL	FORMAT
1	age	Age in Years	
2	days_hospital	Number of days hospitalized	
3	dose	Dose of the Medication (mg/day)	
4	drug	Medication	
5	end_date	End date of the Medication	DATE
6	gender	Gender	\$GENDERF
7	id	Study ID	
8	origin	Country of Origin	COUNTRYF
9	race	Race	RACEF
10	seek_care	Did patient seek hospital care?	YESNO
11	start_date	Start date of the Medication	DATE
12	symptoms	Symptoms experienced	

CONCLUSION

This paper demonstrates one of the best approaches to overcome the tedious process of manually listing numerous variables, their attributes, formats and labels within a SAS program. It provides a simple and easy to understand data step procedure for automatically combining meta data stored in an external file with the raw CSV file.

RECOMMENDED READING

- *Base SAS® Procedures Guide*
- *SAS® For Dummies®*

CONTACT INFORMATION

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

Kalyani Telu
 Henry M. Jackson Foundation in support of
The Infectious Disease Clinical Research Program
 Department of Preventive Medicine and Biostatistics
 Uniformed Services University of the Health Sciences
 ktelu@idcrp.org