Reading and Writing RTF Documents as Data: Automatic Completion of CONSORT Flow Diagrams

Art Carpenter, California Occidental Consultants, Anchorage, AK
Dennis G. Fisher, Ph.D., CSULB, Long Beach, CA

ABSTRACT

Whenever the results of a randomized clinical trial are reported in scientific journals, the published paper must adhere to the CONSORT (CONsolidated Standards Of Reporting Trials) statement. The statement includes a flow diagram, and the generation of these CONSORT flow diagrams is always problematic, especially when the trial is not the typical two-arm parallel design. Templates of the typical two-arm design flow diagram are generally available as RTF documents, however the completion of the individual fields within the diagram is both time consuming and prone to error. The SAS Macro language was used to read a RTF template file for the CONSORT flow diagram of choice, fill in the fields using information available to the SAS program, and then rewrite the table as a completed RTF CONSORT flow diagram. This paper describes the process of reading and writing RTF files.

KEYWORDS

RTF, CONSORT flow diagram, macro language, INFILE, _INFILE_

INTRODUCTION

The CONSORT flow diagram is a graphical representation of the progress through the phases of a clinical trial and typically includes: enrollment, allocation, follow-up, and analysis. While the overall complexity of the diagram depends on the study design, and can vary radically in appearance from study to study, the diagram itself is made-up of a series of interlinked boxes. These boxes contain specific information about that phase of the study. Figure 1 to the left tracks the number of subjects allocated to intervention for a particular ARM of a study. Generally there will be many such boxes in the CONSORT table and the manual completion of the individual values can be both tedious and error prone.

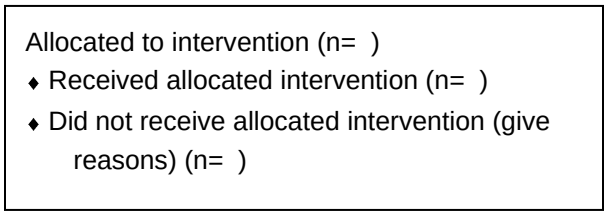

Figure 1

The tables are usually created using a word processor such as MS Word®, which of course, saves the table in a file. One of the more flexible file forms, and therefore the one most commonly used, is Rich Text Format, RTF (which is assumed for this paper).

RTF is a proprietary document file format developed by Microsoft Corporation in the late 1980s. Unlike a MS Word .DOC binary file, a RTF file can be read by text editors. This means that if we treat a RTF file as text, we can use SAS to read and write the RTF file as data, and this opens the door for the power and flexibility associated with the use of the SAS DATA step and the SAS macro language.

The layout of the CONSORT table depends on the study design. This includes the number of ARMS and the phases of the study. The techniques discussed in this paper, however, are completely independent of the study design. The first step in its construction is to create a template form of the CONSORT table. This RTF table will contain all the needed information with blank fields. Figure 2 shows the "Enrollment" portion of a CONSORT table, which will show the number of subjects and their status relative to the study. Typically the N= values would be filled in by hand once they had been determined.

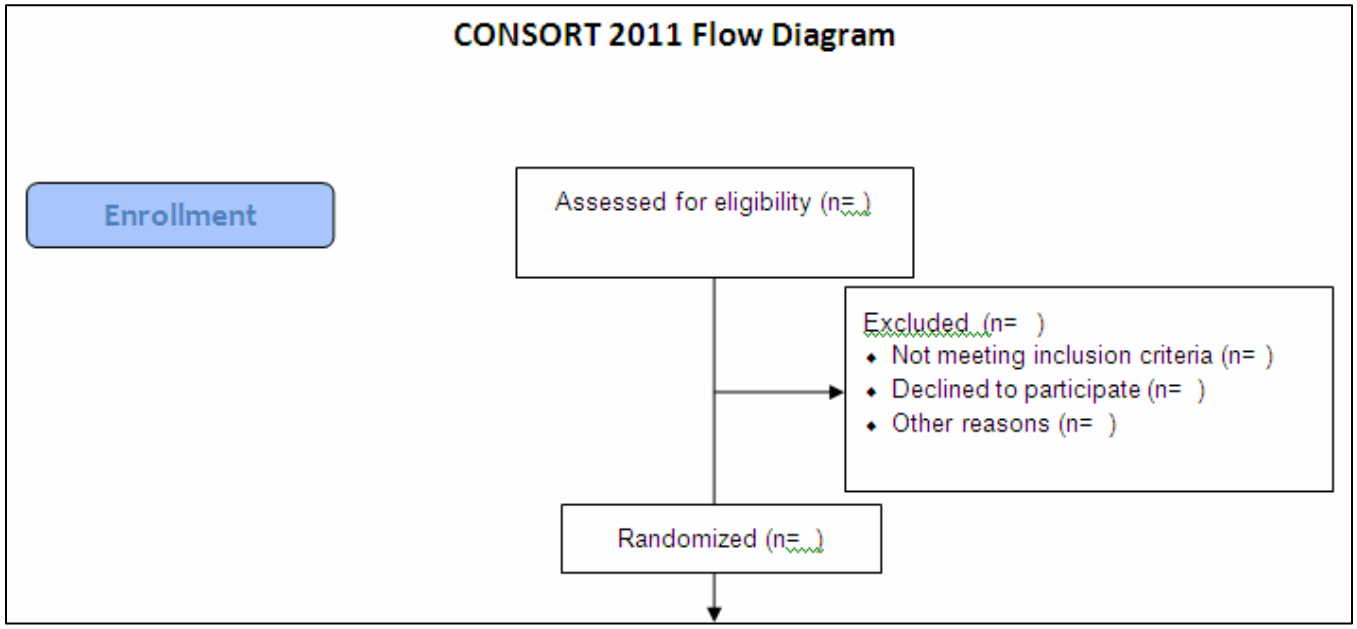

Figure 2

The RTF CONSORT table can easily have over a dozen fields that require completion. In the process described below, each field will be assigned a code unique to the table. The entire table (RTF file) will then be read as data and the codes will be translated into the final values through the use of DATA step functions. The resulting modified table will be rewritten, again as an RTF file, where it will then be available for use by a word processor.

TEMPLATE PREPARATION

The template is prepared for use by SAS by filling in each of the individual fields using unique codes. In Figure 3 the unique codes for the first six fields are TOTASSESSD, TOTEXCL, INELIG, DECLIN, EXCLOTH, and NRAN. For our purposes we are assuming that these names never occur otherwise in the table. Other than being unique, the code that you choose is unimportant, but for a more complicated table the field code names can be used to help make sure that the values are inserted in the correct location.

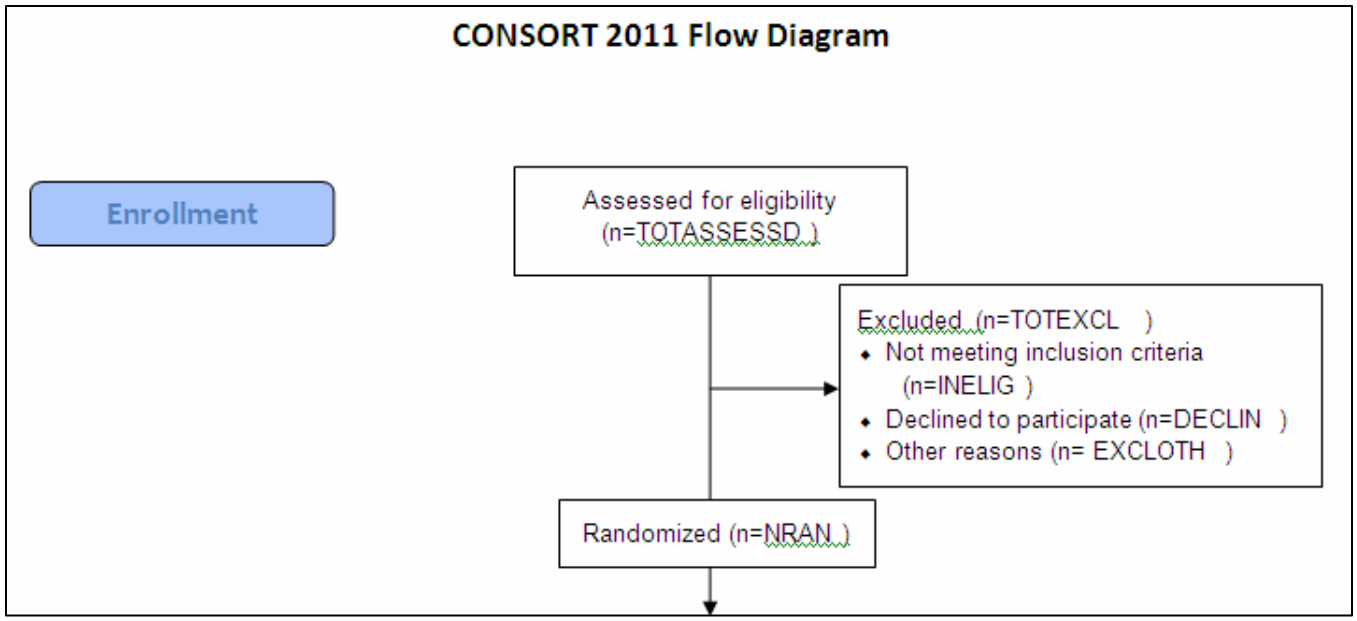

Figure 3

RTF AS DATA

Fortunately we need to know very little about RTF code in order to work with it using SAS. A quick look at a portion of the RTF code that generated Figure 3, shows a text language which is mostly not human readable. However a closer inspection shows one of our designated keywords (DECLIN).

```
\par }{\rtlch\fcs1 \af0\afs16 \ltrch\fcs0 \f3\fs16\lang0\langfe1033\langnp0
\fs16\lang4105\langfe1033\langnp4105\insrsid4260155\charrsid1516310   }{\rt
\f1\fs20\lang4105\langfe1033\langnp4105\insrsid1909421 DECLIN}{\rtlch\fcs1
\par }{\rtlch\fcs1 \af0\afs16 \ltrch\fcs0 \f3\fs16\lang0\langfe1033\langnp0
```

Our approach will be to have SAS read the RTF text strings, find the appropriate codes, replace the codes with the values of interest, and then replace the modified RTF text strings. The search and replace operations will be handled by using the TRANSTRN function, which replaces all occurrences of the second argument with the third argument. For our purposes there should only be one occurrence for each of our codes.

The DATA step used to read and write the RTF CONSORT table is fairly straight forward. RTF does not have a fixed maximum record length; however the length is generally under 500 characters. Here the LRECL is set to 3000 – just in case. The incoming RTF file is designated by the *fileref* CONFFILE1. The new version of the CONSORT table is written to the file named in the CONFILE2 *fileref*. Through the use of the automatic variable _INFILE_ we read each RTF line as an entire entity. This string is then searched and the appropriate codes are replaced. In this example the TRANSTRN function replaces the text 'TOTASSESSED' with the appropriate number which we have provided (345). Figure 5 shows that the placeholder codes that we used in the template version of the table have been replaced with the values supplied in the SAS program.

```
filename confile1 "C:\temp\CONSORT_Diagram1.rtf";
filename confile2 "C:\temp\CONSORT_Diagram2.rtf";

data _null_;
infile confile1 lrecl=3000;
input;
_infile_ = transtrn(_infile_,'TOTASSESSED','345');
_infile_ = transtrn(_infile_,'TOTEXCL',    '56');
_infile_ = transtrn(_infile_,'INELIG',     '35');
_infile_ = transtrn(_infile_,'DECLIN',     '17');
_infile_ = transtrn(_infile_,'EXCLOTH',    '4');
_infile_ = transtrn(_infile_,'NRAN',       '289');
file confile2  lrecl=3000;
put _infile_;
run;
```

Figure 4

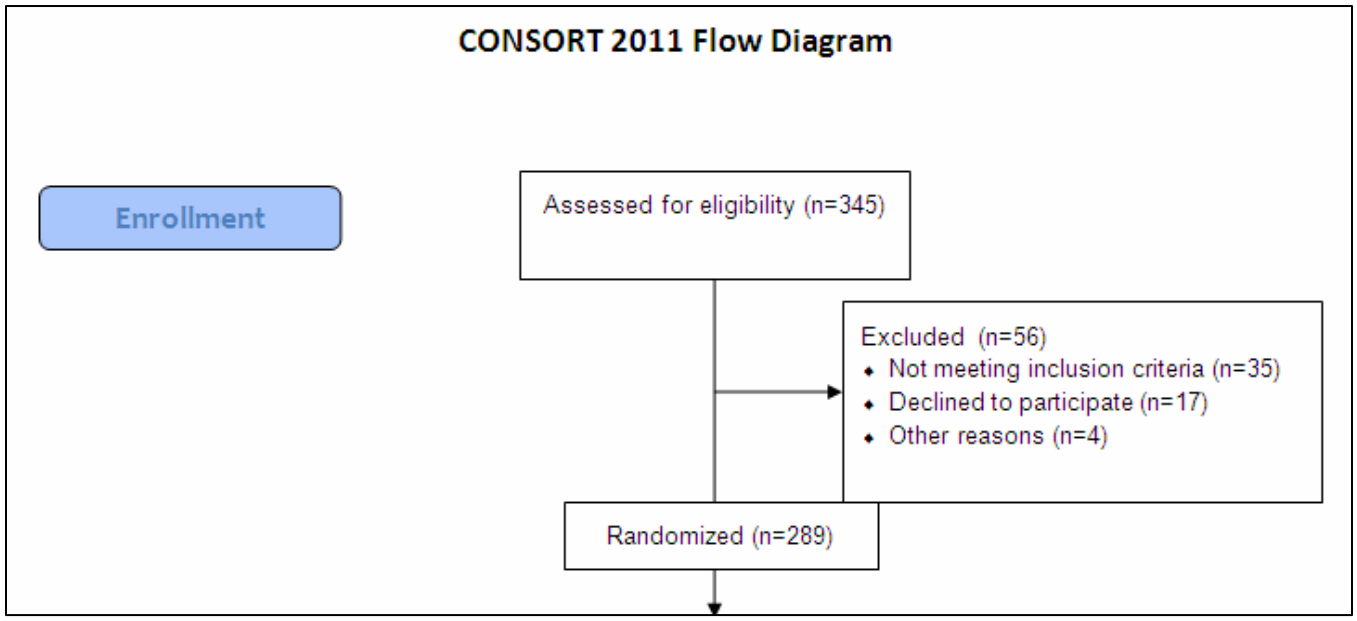

Figure 5

USING DATA STEP STATEMENTS

In the previous example the field values have been hard coded (Figure 4). It is more likely that the values will be contained in a data set or perhaps in a spread sheet. The DATA step used to fill in the values for Figure 5 can be slightly modified to accept non-hardcoded values.

We create a single observation data set with one variable for each field that is to be replaced. For simplicity we have named each field using the code that we placed in the RTF template file. The value of each variable is then the value that is to be substituted into the template. This data set is then read into the DATA step that will be reading and writing the RTF files, and the values are then available for substitution.

In this DATA step we are assuming that the data set COUNTS contains a single observation and a variable for

```
* Create data to simulate the numbers coming from data;
data counts;
   TOTASSESSED= 345;
   TOTEXCL    = 56;
   INELIG     = 35;
   DECLIN     = 17;
   EXCLOTH    = 4;
   NRAN       = 289;
   output counts;
   run;

data _null_;
if _n_=1 then set counts;
infile confile1 lrecl=3000;
input;
_infile_ = transtrn(_infile_,'TOTASSESSED',trim(left(put(totassessed,6.))));
_infile_ = transtrn(_infile_,'TOTEXCL',    trim(left(put(totexcl,6.))));
_infile_ = transtrn(_infile_,'INELIG',     trim(left(put(inelig,6.))));
_infile_ = transtrn(_infile_,'DECLIN',     trim(left(put(declin,6.))));
_infile_ = transtrn(_infile_,'EXCLOTH',    trim(left(put(excloth,6.))));
_infile_ = transtrn(_infile_,'NRAN',       trim(left(put(nran,6.))));

file confile2  lrecl=3000;
put _infile_;
run;
```

Figure 6

each of the fields. The COUNTS data set is read using a SET statement, however because of the restriction that _N_=1, it is read only once. Since the TRANSTRN function expects the third argument to be character, the numeric variables are converted using the PUT function. It is of course, only convenient and certainly not necessary that the data set variable names and the field codes are the same.

Since we are working with a DATA step there is no reason why we cannot add some data checks as well. For instance for these fields the number of subjects randomized must equal the total number of assess subjects less those excluded. IF-THEN/ELSE processing can be used when reading in the data set containing the field values.

```
data _null_;
if _n_=1 then do;
   set counts;
   if nran ne (totassessed-totexcl)
     or totexcl ne (inelig + declin + excloth)
          then put 'WARNING: Counts incorrect';
end;
infile confile1 lrecl=3000;
input;
_infile_ = transtrn(_infile ,'TOTASSESSED',trim(left(put(totassessed,6.))));
                        . . . . remainder of the DATA step is not shown . . . .
```

Figure 7

GENERALIZING WITH THE MACRO LANGUAGE

In the previous example (Figure 6) the COUNT data set was constructed with one observation containing a series of variables. It could have also been built with one observation for each variable / value pair. A data set in this form maximizes its flexibility as it can be used to build a CONSORT table with any number of fields, boxes, and ARMs.

```
data counts;
   fldname='TOTASSESSED'; fldvalue= 345; output counts;
   fldname='TOTEXCL';     fldvalue= 56;  output counts;
   fldname='INELIG';      fldvalue= 35;  output counts;
   fldname='DECLIN';      fldvalue= 17;  output counts;
   fldname='EXCLOTH';     fldvalue= 4;   output counts;
   fldname='NRAN';        fldvalue= 289; output counts;
   run;
```

Figure 8

These data values are then loaded into macro variable lists, and the values from these lists will then be used in the TRANSTRN function. The data values can be written to the macro symbol table in either a DATA step (using CALL SYMPUTX) or in a SQL step. The SQL step is easiest for building a macro list and is shown in Figure 9. The macro variable lists, shown in Figure 10, will necessarily be synchronized so that, for instance, the second name (TOTEXCL) will be associated with the second number (56). This allows us to use a %DO loop to step through these lists and generate a TRANSTRN function call for each pair of values individually.

```
proc sql noprint;
select fldname, fldvalue
   into :fnamelist separated by ',',
        :fvaluelist separated by ','
      from &dsn;
%let fldcnt = &sqlobs;
quit;
```

Figure 9

```
%put &fnamelist;
TOTASSESSED,TOTEXCL,INELIG,DECLIN,EXCLOTH,NRAN

%put &fvaluelist;
345,56,35,17,4,289
```

Figure 10

Because we are using %DO loops to construct the TRANSTRN functions, we must therefore also create a macro to control the overall process. Most of the actual DATA step that does the translations remains unchanged. We no longer need to import the COUNTS data set as we did in Figures 6 and 7, however we do need to parse the macro lists, and this is done with the %SCAN macro function.

A macro %DO loop is now used in what will become the DATA step. The %DO loop is used to control the word counter, which in turn is used to parse the word list using the %SCAN function. The &ith word is selected by the %SCAN function and written to the appropriate macro variable (&NAME and &VALUE). These macro variables are then used in the assignment statement that modifies the _INFILE_ value. We could use the %SCAN

```
data _null_;
infile confile1 lrecl=3000;
input;
%do i = 1 %to &fldcnt;
   %let name =%scan(%bquote(&fnamelist),&i,%str(,));
   %let value =%scan(%bquote(&fvaluelist),&i,%str(,));
   _infile_ = transtrn(_infile_,"&name",trim(left(put(&value,6.))));
%end;

file confile5  lrecl=3000;
put _infile_;
run;
```

Figure 11

function calls within the assignment statement, and could have saved the generation of the &NAME and &VALUE macro variables, however that would have also made the code more complex.

The resulting macro (the full macro code for the %FILLFLDS macro can be found in the appendix at the end of this paper) can be used to replace as many text character strings as needed. The macro is independent of the number of fields, the number of ARMs, and the number of text boxes in the table.

SUMMARY

The manual completion of a study's CONSORT flow diagram can be prone to error. Since the data are already being analyzed in SAS and the numbers are already available in SAS, why not let SAS do the work for us. Using the techniques shown here allows us to eliminate the necessity of manually editing the table and to easily auto-complete the CONSORT flow diagram as a part of the processing of the data.

ABOUT THE AUTHORS

Art Carpenter's publications list includes four books, and numerous papers and posters presented at SUGI, SAS Global Forum, and other user group conferences. Art has been using SAS® since 1977 and has served in various leadership positions in local, regional, national, and international user groups. He is a SAS Certified Advanced Professional programmer, and through California Occidental Consultants he teaches SAS courses and provides contract SAS programming support nationwide.

Dennis Fisher, PhD is the current director of the Center for Behavioral Research and Services at California State University, Long Beach. He has been the Principal Investigator on two randomized clinical trials funded by the National Institute on Drug Abuse. He is the co-author (with Scott Hershberger and Brian Wiens) of a book titled *Multivariate Clinical Trials for Randomized Experiments in the Behavioral Sciences*. He has published over 200 peer-reviewed journal articles.

AUTHOR CONTACT

Dennis G. Fisher, Ph.D.
Professor and Director
Center for Behavioral Research and Services
1090 Atlantic Avenue
Long Beach, CA 90813

562-495-2330 x121
dfisher@csulb.edu

Arthur L. Carpenter
California Occidental Consultants
10606 Ketch Circle
Anchorage, AK 99515

(907) 865-9167
art@caloxy.com
www.caloxy.com

ACKNOWLEDEMNENTS

We would like to thank at least someone. Probably that is the one person who actually read the paper to the end, and that would be you.

REFERENCES

Fairfield-Carter, Brian and Suzanne Humphreys, 2011, "Alternative Approaches to Creating Disposition Flow Diagrams", proceedings of the Pharmaceutical SAS Users Group Conference (PharmaSUG), 2011, Cary, NC: SAS Institute Inc. http://www.pharmasug.org/proceedings/2011/TT/PharmaSUG-2011-TT08.pdf

Tran, Duong, 2008, “A Novel Approach to Patient Profiling”,
http://www.tranz.co.uk/PSI2008_WD_Patient_Profiling.pdf
This paper uses a similar approach to fill cells in an EXCEL table.

Xu, Michelle and Jay Zhou, 2007, “%DIFF: A SAS Macro to Compare Documents in Word or ASCII Format”, proceedings of the Pharmaceutical SAS Users Group Conference (PharmaSUG), 2007, Cary, NC: SAS Institute Inc.
http://www.lexjansen.com/pharmasug/2007/cc/cc09.pdf

A discussion of the CONSORT standards can be found at:
http://www.consort-statement.org/index.aspx?o=1413, which includes an example of a downloadable flow diagram at: http://www.consort-statement.org/consort-statement/flow-diagram0/

TRADEMARK INFORMATION

APPENDIX – SAMPLE CODE

```
%let path= c:\temp;

filename confile1 "&path\CONSORT_2011_Flow_Diagram1.rtf";
filename confile2 "&path\CONSORT_2011_Flow_Diagram4.rtf";

* Use a data table to provide input to a DATA step;
* Create data to simulate the numbers coming from data;
data counts;
   TOTASSESSED= 345;
   TOTEXCL    = 56;
   INELIG     = 35;
   DECLIN     = 17;
   EXCLOTH    = 4;
   NRAN       = 289;
   output counts;
   run;

data _null_;
if _n_=1 then do;
   set counts;
   if nran ne (totassessed-totexcl)
     or totexcl ne (inelig + declin + excloth) then put 'WARNING: Counts
incorrect';
end;
infile confile1 lrecl=3000;
input;
_infile_ = transtrn(_infile_,'TOTASSESSED',trim(left(put(totassessed,6.))));
_infile_ = transtrn(_infile_,'TOTEXCL',    trim(left(put(totexcl,6.))));
_infile_ = transtrn(_infile_,'INELIG',     trim(left(put(inelig,6.))));
_infile_ = transtrn(_infile_,'DECLIN',     trim(left(put(declin,6.))));
_infile_ = transtrn(_infile_,'EXCLOTH',    trim(left(put(excloth,6.))));
_infile_ = transtrn(_infile_,'NRAN',       trim(left(put(nran,6.))));
```

```
file confile2  lrecl=3000;
put _infile_;
run;


************************************************************;
filename confile5 "&path\CONSORT_2011_Flow_Diagram5.rtf";
*  Use a data set to provide input to a macro;
data counts;
   fldname='TOTASSESSED'; fldvalue= 345; output counts;
   fldname='TOTEXCL';     fldvalue= 56;  output counts;
   fldname='INELIG';      fldvalue= 35;  output counts;
   fldname='DECLIN';      fldvalue= 17;  output counts;
   fldname='EXCLOTH';     fldvalue= 4;   output counts;
   fldname='NRAN';        fldvalue= 289; output counts;
   run;

%macro fillflds(dsn=counts);
%local fnamelist fvaluelist fldcount i name value;
* Load the values into macro variable lists;
proc sql noprint;
select fldname, fldvalue
   into :fnamelist separated by ',',
        :fvaluelist separated by ','
      from &dsn;
%let fldcnt = &sqlobs;
quit;

%* For illustration: Show the lists;
%put &fnamelist;
%put &fvaluelist;

data _null_;
infile confile1 lrecl=3000;
input;
%do i = 1 %to &fldcnt;
   %let name =%scan(%bquote(&fnamelist),&i,%str(,));
   %let value =%scan(%bquote(&fvaluelist),&i,%str(,));
   _infile_ = transtrn(_infile_,"&name",trim(left(put(&value,6.))));
%end;

file confile5  lrecl=3000;
put _infile_;
run;
%mend fillflds;
%fillflds(dsn=counts)
```