

Custom Reports - Examples For a Table of Contents and Index

Deb Cassidy, Cardinal Health, Dublin, OH

ABSTRACT

Have you ever needed a report where none of the PROC's gave you what you wanted? Then "DATA _NULL_" could be your answer. While some people will complain it takes too long or it is too complex, this method does give you great flexibility. In this case, I needed some "unusual" reports - a table of contents and a couple indices.

In these examples, new variables are also being calculated. These examples not only show you how to use DATA _NULL_ to create a custom report but also show the use of RETAIN and ARRAY statements, LAG, TRIM, LEFT, SUBSTR, MOD and PUT functions, concatenation, and "first-dot/last-dot" processing.

BACKGROUND & REQUIREMENTS

The MWSUG Conference Proceedings required a table of contents, an author index and a keyword index. Since some of the papers don't arrive until the deadline, there was limited time to produce these pages. If another paper arrived later, the paper would either be excluded from the proceedings or everything would need to be redone. An easy method which allowed for last minute arrivals was to use SAS® to create the table of contents and indices as text files. These text files were then opened in a word-processor which has an option to convert text to a table. Once the information was in table form, you could quickly right and left-justify the appropriate columns and make other spacing adjustments such as wrapping information in a particular cell. There were also several macros created to quickly apply the necessary formatting (font, size, bold, etc.). In less than 30 minutes, everything was created. Samples of the table of contents and the indices are shown at the end of the paper.

The variables used in these reports were:

<u>Description</u>	<u>Name</u>
author first name	firstname
author last name	lastname
affiliation	affil
paper title	title
7 keyword fields	key1-key7
number of pages in the paper	page_cnt
section paper was assigned	section

The requirements were as follows. The proceedings needed a separator page between each section although the separator page did not require page numbers. This meant the first paper in the section had to start on the right-hand side (odd numbers). The Table of Contents, Foreword and other information found before the first paper were in roman numerals so they did not have to be counted.

In the Table of Contents the title, author and affiliation were on separate lines with the page number also on the first line of the title. This meant that extremely long titles needed to wrap while maintaining the location of the page number. There needed to be two blank lines before and after the section name and a single line after the author affiliation. Author names were shown as you normally read names. Papers were listed alphabetically by author and paper title within each section.

The author index needed names in the form "last name, first name" and had to be sorted by last name. Some authors had multiple papers so these paper numbers needed to be combined. In addition, there needed to be a single letter starting each section of names to make it easy to find names.

The keyword index was similar to the author index with the use of the first letter as a header to make it easy to find keywords. Both indices would be printed in two columns once they were brought into the word processor. However, this meant the keyword index had an additional problem which the author index did not have. Some keywords were found in so many papers that not all page numbers would fit on one line.

The keyword index also had a sorting issue that did not cause problems in the author index although it might have. The sorting algorithm in SAS doesn't work the way the keywords needed to be sorted. For example, consider the following keywords:

Covariance
CPU utilization
Cross-platform

These keywords need to appear in the index as shown above. Although different operation systems can have different results, my sort ended up with "CPU utilization" listed first because capitals are sorted

before lower case. Converting everything to uppercase was not at option since some keywords were case-sensitive.

Another problem was the fact that some keywords start with symbols rather than letters. For example, %ANNOMAC or *= are keywords. I did not want a header on this type of keyword and I wanted them treated as one group. Given my operating system, these keywords would sort before the regular A-Z keywords.

SAS CODE - TABLE OF CONTENTS

The actual code is shown later in this paper. The first dataset eliminated any papers which would not be in the proceedings by deleting any paper with a page count less than one. Names were created in two forms - "first-name last-name" and "last-name, first-name". The TRIM function eliminates trailing blanks while the LEFT function left-justifies the values. Both functions were used since there was no way to know how many leading or trailing blanks might have been added in the data-entry process as well as those that appeared because the value was padded to be the entire length of the variable. The || is the concatenate symbol to put the pieces back together. The blank that you really did want to have between the names had to be included in the concatenation process.

The next step is to sort the papers in alphabetical order by the author's last name within the assigned section. Title is also need as part of the sort since there were authors with multiple papers in the same section. The OUT= option is used on the sort so the original dataset is not overwritten. New datasets were created throughout the program for verification purposes. If you were working with large datasets or running the program on a regular basis, you would want to delete old datasets when you were done with each one to avoid running out of space.

Now comes the fun of assigning page numbers. The first paper will start on page 1. If that paper is 6 pages long, the next paper will start on page 7. The third paper then starts on page 12. Your dataset would look something like this.

Paper#	# of pages	Page # in Proceedings
1	6	1
2	5	7
3	1	12

However, if that third paper was the start of a new section it would have to be adjusted to page 13 since all sections must begin on the right-hand side.

RETAIN is used to carry the assigned page number along to the next observation. The LAG function picks up the number of pages in the previous paper since that is the number which must be added to the page assignment to get the page assignment for the observation you are processing. A word of caution on using the LAG function - never use it as part of an IF statement or you will get very unexpected results.

N is an automatic variable which represents the number of times the DATA step has iterated.****You need to have a starting point so the IF statement uses _N_ to identify the first observation and assign page number 1. If it isn't the first paper, you just add the page count of the previous paper to get the page number for the this paper.

Other variables used in this program are commonly referred to "first-dot" and "last-dot". When your read a dataset with a BY statement, these variables will automatically be created. First-dot identifies if the observation was the first one with the particular value and last.dot identifies if the observation was the last one with the particular value. The following shows the values created with a statement like BY SECTION AUTHOR.

Section	Author	First. Section	Last. Section	First. Author	Last. Author
A	Doe	1	0	1	1
A	Smith	0	0	1	0
A	Smith	0	1	0	1
B	Metar	1	1	1	1

The last statement in the dataset adjusts for the fact that the first paper in each new section must be on the right-hand-side and must be an odd number. The MOD function calculates the remainder when you divide the first argument by the second argument. Any even number divided by 2 will have a remainder of 0 and any odd number divided by 2 will have a remainder of 1. If the start of the first paper in a section is not on the odd page, you must add 1 to make it odd.

Now to create the Table of Contents page. While it is possible to do all the fonts and some of the other formatting from within the SAS code, you will need to know printer codes to do so. This could also require changes to the code if you needed to use a

different printer. Therefore, it was much simpler to create a macros in the word processor to take care of the fonts.

To convert text to a table, the word processor needs to know what will be used to identify the individual columns. Commas are often used but since I have commas in my data it wouldn't work here. Another alternative is tab - just don't try to use the "TAB" key on your computer. To put the value of a "TAB" into your code, it needs to be written in hexadecimal notation which happens to be '09'x. In the Table of Contents, I'll need to have three columns. The first one is for the section header and the last one is for the page number. The middle column will be used for the paper title, author's name, and author's affiliation with each field on a separate line. The code may appear to have lots of extra tabs but you'll discover that each row has two tabs to be able to create the three columns.

DATA _NULL_ is used since I don't need to create a new dataset. The FILE statement will redirect the output to the specified file. Otherwise, PUT statements will write to the LOG by default. I create a variable called TB and set it equal to '09'x simply so I don't have to type the hexadecimal value all the time. It is also easier to read the code.

Once again, a BY statement is used to create the first-dot and last-dot automatic variables to identify each section change. If the record is the first for a section, I need to write out the section name. After each section heading, there are two blank lines. The paper title and page number will be written followed by the author's name on the next line and the author's affiliation on the following line. There will be another line separating each paper. If the paper is the last in the section, another blank line will be used to separate the sections.

SAS CODE - AUTHOR INDEX

This next section of code creates the author index. The same dataset is used except it is sorted by the author's last name. The page number must also be included in the sort since some authors had multiple papers. The final table will need to have two columns so each line will have one tab.

The code starts out similar to the previous code. I also need to separate the author's names based on the first letter of their last name. A variable LETTER is RETAINED for this purpose. The SUBSTR function is used to create the value of LETTER from the

author's name. The function indicates to start at the first character of LCOMMAF and continue for one character. If the observation is the first one, create LETTER and write it out followed by a blank line. If the observation is not the first, then you need to check to see if the first character of the name is the same as the current value of LETTER. If it is not, then you must reassign LETTER to the first character of the current name. You'll need a blank line to separate the previous name from the letter heading and also a blank line after you write the letter heading.

Now if the observation is the first one for the author, you'll write the author's name (last name first) and the page number. However, since some authors have multiple papers, you need to make adjustments. The @ symbol is called a "trailing at". In this instance, it means to hold the line you are currently writing out because more information may be coming to be written on that line. If the observation is not the first one for the author, you'll need to put a comma and then put the page number for the next paper for that author. To make things line up a little neater, a format of 3. is used so the page numbers always occupy 3 characters even if the number is under 100. Whether you use the format depends upon how you want the text to look in the final version. You must tell SAS to go ahead and release the line so it will move to the next line in the report you are creating. The PUT followed only by a semicolon will take care of this for the last observation for that author. In cases where there is only one paper for an author, first.lcommaf and last.lcommaf refer to the same observation.

SAS CODE-KEYWORD INDEX

The last part of this process creates the keyword index. In this case, we do need to create a new dataset before it can be printed. Each paper is a single observation and could have values for anywhere between 1 and 7 keywords. However, the index needed to be printed with each keyword being treated as an observation and a single variable representing the keywords. That means taking a dataset that looked like this

Page #	Key1	Key2	Key3	Key4	Key5	Key6	Key7
1	Formats	Cntlin	Cntlout				
8	Tabulate	Formats					

and turning it into a dataset that looked like this:

Page #	Key
1	Cntlin
1	Cntlout
1	Formats
8	Formats
8	Tabulate

The `datastep` is written to cycle through each of the 7 keyword variables. If the variable has a value, that value is assigned to the new "KEY" variable and the output statement is used to write the observation to the new dataset. This example uses an implicit array although the same thing can be done with an explicit array. (Some sites only want you to use explicit arrays since implicit arrays are the older style.) The array statement basically assigns a single alternate name to the specified list of variables. You can then refer to the alternate name and the same actions will be taken on all the variables you have listed in the array. In this case, the alternate name is `KEYS`. The `DO OVER` statement is used to cycle through the variables. There are two `END` statements. The first is needed to end the `IF-THEN-DO` statement. The second corresponds with the `DO OVER` statement.

In addition to assigning the value to the new `KEY` variable, this `datastep` also creates two other variables. Since the sorting must be done as though everything was in uppercase but the actual case must be maintained, a variable `KEYSORT` is created with the capitalized version of the value. The `KEYLTR` variable containing the first letter of the keyword is also created here. The latter could have been processed just like the author index but since the `datastep` was being created anyway, it was just as simple to do it here. The dataset is then sorted.

The step to write out the index could have been done just like the author index. However, I opted for a different method. An earlier version of the code actually created a dataset instead of using `DATA _NULL_` so I could verify my results. In this case, a new variable is created to hold the page numbers of all the papers with the same keyword specified. Since there will be commas in the page number list, the new variable must be character. The `LENGTH` statement is specified to set the length to 100 characters since I don't know how many papers I will have for each keyword. The `RETAIN` is used to carry the value created to the next observation. I use

the `PUT` function to apply the format of 3. to ensure all numbers are printed the same width. If the observation is the first for the keyword, I assign the page number to the page number list. Otherwise, I trim the trailing blanks from the existing list, add a comma and a blank, and then add the current page number. The trim is necessary because the page list variable actually contains blanks at the end to finish filling up the length of 100.

Some of the keywords start with symbols rather than letters. For example, `%ANNOMAC` or `=*` are keywords. I did not want a header on this type of keyword and I wanted them treated as one group. This is the reason for the `KEYLTR>='A'` part of the `IF` statement that creates the header lines. Once again I needed a blank line before and after each header. Since I knew there were keywords before the first header, I didn't need to worry about any special processing for the first header like I did for the author index. If the observation is the last one for the keyword, then the final list of page numbers is written out.

FINAL TOUCHES

Once all the files were created, they were opened in the word-processor. Fonts and other formatting were applied as appropriate using macros which had already been written. The appropriate columns were right or left-justified and other spacing adjustments were done. Both the author index and keyword index were set to print in a two-column format.

SUMMARY

You have now seen some of the flexibility SAS provides to create a report in any form you might need.

REFERENCE

SAS Institute Inc., SAS Language: Reference, Version 6, First Edition Cary, NC: SAS Institute Inc., 1990. 1042 pp.

CONTACT INFORMATION

Deb Cassidy
Cardinal Health
5555 Glendon Court
Dublin, OH 43016
614-717-7136
dcassidy@cardhealth.com
(Note: Mailing address will change late 1998.)

SAS CODE

**** STEP 1 - KEEP PAPERS TO BE IN PROCEEDINGS AND CREATE NAME VARIABLES;
data papers1;

```
    set papers;
    if page_cnt<1 then delete;
    lcommaf=trim(left(lastname)) || ' , ' || trim(left(firstname));
    frstlast=trim(left(firstname)) || ' ' || trim(left(lastname));
run;
```

**** STEP 2 - SORT DATA AND CALCULATE PAGE NUMBERS;

```
proc sort data=papers1 out=paper;
    by section lcommaf title;
run;
```

```
data paper2;
    set paper;
    by section ;
    retain newpage ;
    z=lag(page_cnt);
    if _n_=1 then newpage=1;
    else newpage=newpage+z;
    if first.section and _n_ ne 1 and mod(newpage,2) ne 1 then newpage+1;
run;
```

**** STEP 3 - PRINT TABLE OF CONTENTS;

```
data _null_;
    file 'c:\myloc\title.txt';
    tb='09'x;
    set bysec;
    by section;
    if first.section then put @1 section tb tb / tb tb / tb tb;
    put  tb title tb newpage / tb frstlast tb  / tb affil tb / tb tb ;
    if last.section then put tb tb;
run;
```

**** STEP 4 - SORT DATA AND PRINT AUTHOR INDEX;

```
proc sort data=paper2 out=byauthor;
    by lcommaf newpage;
run;
```

```
data _null_;
    file 'c:\myloc \authr.txt';
    tb='09'x;
    set byauthor;
    by lcommaf;
    retain letter;
    if _n_=1 then do;
        letter=substr(lcommaf,1,1);
        put @1 letter tb / tb;
    end;
    else if substr(lcommaf,1,1) ne letter then do;
        letter=substr(lcommaf,1,1);
```

```
        put tb / letter tb / tb;
    end;
    if first.lcommaf then put @1 lcommaf tb newpage @;
    else put ', ' newpage 3.. @;
    if last.lcommaf then put;
run;

**** STEP 5 - CREATE KEYWORD DATASET
data keywords (keep=paper_ key newpage keysort keyltr);
    set paper2;
    array keys key1-key7;
    do over keys;
        if keys ne ' ' then do;
            key=keys;
            keysort=upcase(key);
            keyltr=substr(keysort,1,1);
            output;
        end;
    end;
run;

**** STEP 6 - SORT DATA AND PRINT KEYWORD INDEX;
proc sort data=keywords out=bykey;
    by keyltr keysort newpage;
run;

data _null_;
    file ' c:\myloc\keyword.txt';;
    set bykey;
    by keyltr keysort;
    tb='09'x;
    length pagelist $100;
    retain pagelist* ' ';
    if first.keysort then pagelist=(put(newpage,3.));
    else pagelist=trim(pagelist)||', '|| (put(newpage,3.));
    if first.keyltr and keyltr>='A' then put @1 tb / keyltr tb / tb;
    if last.keysort then put @1 key tb pagelist ;
run;
```

SAMPLE OF FINAL TABLE OF CONTENTS PAGE

Applications Development

A Helpful Guide to SAS Applications	1
<i>John Doe</i>	
J. D. Consulting Services	
Interfacing SAS to Other Data Sources Such as Microsoft Excel, Microsoft Word or Microsoft Access	8
<i>Jane Smith</i>	
Whatshare Distributing, Inc.	

SAMPLE OF FINAL AUTHOR INDEX PAGE

Note: There are 2 columns per page

D

Diverge, Mo	45
Doe, John	1,56

M

Mastermind, Thomas	347
Mythor, Amy	425

SAMPLE OF FINAL KEYWORD INDEX PAGE

Note: There are 2 columns per page

D

Data _NULL_	8, 21, 75, 174, 224
Documentation	121

M

Merge	347
Multicollinearity	173

SAMPLE OF TABLE OF CONTENTS PAGE FROM SAS

Applications Development		
	A Helpful guide to SAS Applications	1
	John Doe	
	J.D. Consulting Services	
	Interfacting SAS to Other Data Sources Such as Microsoft Excel, Microsoft Word or Microsoft Access	3
	Jane Smith	
	Whatshare Distributing, Inc.	
Tutorials		
	Custom Reports - Examples For a Table of Contents and Index	73
	Deb Cassidy	
	Cardinal Health, Inc.	

SAMPLE OF AUTHOR INDEX PAGE FROM SAS

C	
Canvil, Sandra	303 , 310
Cassidy, Deb	73
D	
Diverge, Mo	45
Doc, John	1 , 56, 314
Dyne Peter	32

SAMPLE OF KEYWORD INDEX PAGE FROM SAS

%Annomac	167
%Include	147
%Put	47
-2 Log L statistic	180
=*	381
=:	197
A	
Accelerated life testing	215
Adverse events	113
Analytical tool	208
Animal health	232
Annotate	167