Paper 150-2023

Wrangling Excel Worksheets

Tom Kari, Tom Kari Consulting

ABSTRACT

Ah, Excel. Sadly, it's Excel's world and we only live in it. Anybody experienced with answering questions on the SAS Communities has seen the full gamut of problems that are encountered with worksheets: the data is in the wrong column; the data is in the wrong row; I'm expecting a number in a cell, but it's a character value; the cell should contain a date, but it doesn't; I have too many or too few rows or columns.

Yes, it's true that on occasion importing data into SAS from an Excel workbook can fail, but those are the GOOD results. Worse are the cases where it looks like the import worked, but you got the wrong data in the wrong place.

I was faced with the requirement to import multiple Excel worksheets monthly, in a production environment. Of course, I was ASSURED that "there wouldn't be any problems with the worksheets", and promptly ran into all of the problems described above.

To deal with these problems, I developed BASE SAS code to do the following...

- Import all of the data as character, and preprocess the records.
- After importing the data from Excel, transpose it into a one-column SAS dataset.
- Use a template file to determine whether the contents of a cell should be empty, a constant, or a variable, and the type of variable.

INTRODUCTION

Excel is a challenge. Unquestionably the most popular spreadsheet product ever created, users have embraced it with unflagging enthusiasm.

There was a day when spreadsheet software was used only to create spreadsheets, but those days are long gone. Almost any business process that deals with data will at some point have Excel in the process.

On one of my projects, I had to develop a process for dealing with data from Excel spreadsheets that I found worked quite nicely. In presenting it here, I hope that you find some worthwhile tips for your future Excel endeavors.

A PROBLEM WITH EXCEL

A few years ago, I had to import and process Excel data as a component of a system that I was working on.

I'm not going to present that data, but Statistics Canada is a wealth of socio-economic data about Canada. I grabbed Figure 1 from the Statistics Canada website, because it's reasonably similar to the data that I had to deal with, both in size and complexity.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1					Statistics Canada/ Statistique Canada										
2					Surveys and statistical programs										
3															
4	Marital status, age group and gender: Canada, provinces and territories, census metropolitan areas, census agglomerations and census subdivisions														
5															
6	Frequency: Occasional														
7															
8	Table: 98-10-0127-01														
9															
10	Release date: 2023-03-29														
11	Geography Canada														
12	Gender Total - Gender														
13															
14	Age (16)	Total - Age	15 to 19 years	20 to 24 years	25 to 29 years	30 to 34 years	35 to 39 years	40 to 44 years	45 to 49 years	50 to 54 years	55 to 59 years	60 to 64 years	65 to 69 years	70 to 74 years	75 to 79 years	80 to
15	Marital status (13)															
16	Total - Marital status	30,979,185	2,012,975	2,202,255	2,421,505	2,518,835	2,511,350	2,399,405	2,304,170	2,368,350	2,647,330	2,571,580	2,210,970	1,847,585	1,260,930	
17	Married or living common-law	17,626,005	18,970	289,370	973,220	1,571,875	1,782,995	1,741,050	1,667,450	1,681,435	1,848,845	1,769,870	1,521,735	1,232,255	780,775	
18	Married	13,725,625	2,285	71,265	404,850	995,445	1,314,925	1,340,820	1,332,425	1,361,520	1,514,875	1,515,565	1,339,340	1,118,275	722,820	
19	Living common-law	3,900,375	16,690	218,100	568,370	576,430	468,075	400,235	335,025	319,910	333,970	274,300	182,390	113,975	57,955	
20	Living common law - Never married	2,967,665	16,685	216,880	561,160	550,190	416,935	328,065	240,915	203,000	188,115	129,535	66,815	30,825	12,025	
21	Living common law - Separated	123,830	0	385	2,490	7,315	12,600	16,305	18,365	18,700	17,715	13,095	8,030	4,950	2,465	
22	Living common law - Divorced	721,715	5	340	4,215	18,210	37,480	54,080	72,730	93,200	119,025	119,230	93,540	63,720	31,345	
23	Living common law - Widowed	87,165	0	490	510	715	1,065	1,785	3,015	5,005	9,120	12,440	14,010	14,480	12,120	
24	Not married and not living common-law	13,353,180	1,994,000	1,912,890	1,448,285	946,955	728,350	658,355	636,715	686,920	798,485	781,710	689,240	615,330	480,160	
25	Not married and not living common law - Never married	9,025,865	1,993,990	1,903,710	1,416,870	864,765	578,290	444,390	360,165	343,710	354,275	288,475	192,040	121,315	66,765	
26	Not married and not living common law - Separated	741,135	10	3,715	16,015	40,315	68,070	86,695	95,025	97,255	97,620	82,190	61,210	44,150	25,225	
27	Not married and not living common law - Divorced	1,921,880	5	2,500	11,485	37,425	75,305	115,970	163,040	212,400	281,120	298,730	269,190	217,355	129,865	
28	Not married and not living common law - Widowed	1,664,295	5	2,965	3,915	4,445	6,685	11,295	18,485	33,560	65,465	112,315	166,800	232,510	258,310	
29																
30																
31	How to cite: Statistics Canada. Table 98-10-0127-01 Marital status, age group and gender: Canada, provinces and territories, census metropolitan areas, census agglomerations and census subdivisions															
32	DOI: https://doi.org/10.25318/9810012701-eng															
33																
34	Date modified: 2023-05-09															

Figure 1: A somewhat complex spreadsheet to import to SAS

The spreadsheet contains data about marital status, age, and gender for the population of Canada 15 years and older, from the 2021 Canadian Census of Population.

As with most spreadsheets, the cells contain information of varying types:

- The first thirteen rows contain a combination of blank cells, "boilerplate" (text that won't change from one edition of the spreadsheet to the next), and some identifying characteristics of the spreadsheet as a whole.
- Row 14 classifies the data by columns, starting with column B and continuing through column Q.
- Column A in rows 16 through 28 classifies the data by rows.
- Using the row and column classifiers, the data cells can be contextualized.
- Rows 16 through 28, columns B through Q, contain the actual data cells that are of interest to us.
- Finally, rows 29 through 34 again contain blank, "boilerplate", and identifying characteristics.

I see a LOT of spreadsheets that look like this, so I think a solution to this problem could be widely used.

REQUIREMENTS FOR PROCESSING

I think you'll agree that if we're only dealing with one spreadsheet, there are a number of easy ways to make the data usable in SAS, and there is no need to resort to software development. However, my challenge was that these spreadsheets were being produced by an upstream system, and I needed to handle one spreadsheet per hour, every hour of every day.

Essentially I had to capture the data cells, with the information needed to classify them, and complete or partial contents of some of the spreadsheet-oriented cells.

Using our example, I had to create a SAS table that looked like Figure 2:

	A	B	C	D	E	F	G
1	Frequency	TableID	ReleasedDate	ModifiedDate	MaritalStatus	AgeGroup	Population
2	Occasional	98-10-0127-01	29Mar2023	09May2023	1	1	2285
3	Occasional	98-10-0127-01	29Mar2023	09May2023	2	1	16685
4	Occasional	98-10-0127-01	29Mar2023	09May2023	3	1	0
5	Occasional	98-10-0127-01	29Mar2023	09May2023	4	1	5
6	Occasional	98-10-0127-01	29Mar2023	09May2023	5	1	0
7	Occasional	98-10-0127-01	29Mar2023	09May2023	6	1	1993990
8	Occasional	98-10-0127-01	29Mar2023	09May2023	7	1	10
9	Occasional	98-10-0127-01	29Mar2023	09May2023	8	1	5
10	Occasional	98-10-0127-01	29Mar2023	09May2023	9	1	5
11	Occasional	98-10-0127-01	29Mar2023	09May2023	1	2	71265
12	Occasional	98-10-0127-01	29Mar2023	09May2023	2	2	216880
13	Occasional	98-10-0127-01	29Mar2023	09May2023	3	2	385
14	Occasional	98-10-0127-01	29Mar2023	09May2023	4	2	340
15	Occasional	98-10-0127-01	29Mar2023	09May2023	5	2	490
16	Occasional	98-10-0127-01	29Mar2023	09May2023	6	2	1903710
17	Occasional	98-10-0127-01	29Mar2023	09May2023	7	2	3715
18	Occasional	98-10-0127-01	29Mar2023	09May2023	8	2	2500
19	Occasional	98-10-0127-01	29Mar2023	09May2023	9	2	2965
20	Occasional	98-10-0127-01	29Mar2023	09May2023	1	3	404850

Figure 2 Data Table created from Age Sex spreadsheet

With of course formats for MaritalStatus and AgeGroup along the lines of:

```
proc format;
    value marstf
          1="Married"
          2="Living common-law"
          3="Living common law - Never married"
…
    value agef
          1="15 to 19 years"
          2="20 to 24 years"
          3="25 to 29 years"
…
```

It's not immediately obvious, but there are redundant rows and columns in the data. Column B is a summary of columns C through Q, and rows 16, 17, and 23 are summaries of the rows beneath them. It is required to remove them.

To be honest, it looked pretty straightforward, so away I went.

YOU'LL NEVER GUESS WHAT HAPPENED NEXT...

Of course, I immediately inquired about what editing and verification would be required. "Absolutely none!" came back the answer, and I was assured that the spreadsheets were coming out of an automated system, so the cells would always be flawless.

Yup, the first few spreadsheets that I tested on went beautifully. And then, **CLANG!**, and the program failed. The spreadsheet had an extra column before the data columns that I was expecting. And it was explained to me that occasionally (I HATE that word!) the automated system glitched while creating the spreadsheet, and someone had to go in and manually clean it up. Then they saved it, and passed it on.

WHAT COULD POSSIBLY GO WRONG...

People are really, really good at seeing around format errors, without even realizing that they're doing it. Over the next little while, the following variations from expected format were found in the spreadsheets:

- Extra blank rows;
- A blank column in front of the first data column;
- Someone put a blank into one of the columns past the last data column, which led SAS to think that the cells contained data;
- In one case, one of the actual data columns was missing;
- Someone replaced one of the dates with an entry in a different format;

Well, this required a bit of a rethink. I thought it might be kind of cool if I could correct for some of the errors, but at a bare minimum I needed to be sure that I was finding all of them.

That led me to develop this solution, which I'm sharing because I think it's rather neat. I believe that it reliable identifies any deviance from what's expected, it allows a lot of flexibility with regards to capturing the data, and in a few cases, if desired, it can actually correct for cases where the spreadsheet isn't in the expected format.

A SIMPLIFIED EXAMPLE

So that I can include all of the code and artifacts for the solution, I'm going to introduce Figure 3, a smaller, simpler spreadsheet for the rest of the paper.

	A	B	C	D
1			Statistics Canada/ Statistique Canada	
2			Surveys and statistical programs	
3				
4	Broad age groups and sex: Canada, provinces and territories			
5				
6	Frequency: Occasional			
7				
8	Table: 98-10-0034-01			
9				
10	Release date: 2022-04-27			
11	Geography Canada			
12				
13	Sex (3)	Total - Sex	Male	Female
14	Broad age groups (5)			
15	Total - Age	36,991,980	18,219,520	18,772,465
16	0 to 14 years	6,012,795	3,083,520	2,929,275
17	15 to 64 years	23,957,755	11,908,295	12,049,465
18	65 years and over	7,021,430	3,227,700	3,793,725
19	85 years and over	861,395	319,605	541,785
20				
21				
22	How to cite: Statistics Canada. Table 98-10-0034-01 Broad age groups and sex: Canada, provinces and territories			
23	DOI: https://doi.org/10.25318/9810003401-eng			
24				

Figure 3 A Simpler Spreadsheet Example

Note that while the volume of data is much less, it still presents the same challenges as the preceding sections.

For our purposes, let's assume that the data from the spreadsheet need to be transformed into Figure 4 with the appropriate associated formats for Gender and AgeGroup:

	A	B	C	D	E	F	G
1	Frequency	Table	Release date	Gender	AgeGroup	Population	
2	Occasional	98-10-0034-01	27Apr2022	1	1	3083520	
3	Occasional	98-10-0034-01	27Apr2022	2	1	11908295	
4	Occasional	98-10-0034-01	27Apr2022	3	1	3227700	
5	Occasional	98-10-0034-01	27Apr2022	1	2	2929275	
6	Occasional	98-10-0034-01	27Apr2022	2	2	12049465	
7	Occasional	98-10-0034-01	27Apr2022	3	2	3793725	
8							

Figure 4 Simpler data table

STEP 1: TRANSFER THE DATA FROM EXCEL TO SAS

The first step is to get the data out of Excel, and into SAS. SAS knows that everybody uses Excel, so they are putting enormous resources into making it easy for SAS and Excel to interoperate.

In creating this paper, I did quite a bit of experimenting with some of the different options, and I discovered that they all have different strengths and weaknesses. There isn't any way to provide a "best" solution, I'm afraid you're going to have to go through the same trial-and-error work.

For my example, I found that the following code worked very nicely:

```
/* Import a spreadsheet */
proc import file="/home/cstkari/WUSS2023/CensusSexAge.xlsx"
          out=work.CensusSexAgeRaw dbms=xlsx replace;
    getnames=no;
run;
/* NOTE - all of the variables MUST be character. If not, do whatever
processing is needed to ensure that they are */
```

It created a SAS dataset, with all of the columns as character variables. Because I specified "getnames=no", the variable names are A, B, C, etc.

I can't emphasize enough that for this to work, all of your variables must be character. If you're creating some numeric variable in your SAS dataset, you'll need to convert them to character (the next section is the place to do this).

STEP 2: A BIT OF PREPROCESSING

We're not quite ready to go. Since I'm going to be transposing the data shortly, I need all of the character variables to be the same length, namely the length of the longest variable that's imported from Excel. In addition, I need to have a sequence number for each record in my SAS dataset (each row of the spreadsheet). These changes are made in the following code:

```
/* Put the length of the largest variable into a macro variable */
proc sql noprint;
    select max(length) into :maxlen from dictionary.columns where
           upcase(libname)="WORK" and upcase(memname)="CENSUSSEXAGERAW";
quit;

/* Get all of the variable names */
proc sql noprint;
    select name into :varlist separated by ' ' from dictionary.columns where
           upcase(libname)="WORK" and upcase(memname)="CENSUSSEXAGERAW" order by varnum;
quit;

/* Create a new dataset with all of the columns set at the maximum length */
data CensusSexAgeResized;
    length &varlist. $&maxlen.;
    set CensusSexAgeRaw;
run;

/* Add a sequence number to the dataset */
/* NOTE - it will be named SeqNo...VERY important that the incoming dataset doesn't have a
variable named SeqNo */
data CensusSexAgeResized;
    length SeqNo $5;
    set CensusSexAgeResized;
    SeqNo=put(_n_, z5.);
run;
```

Also, if any of your SAS dataset variables are numeric, this is the spot to convert them to character.

STEP 3: TRANSPOSE THE DATA

And now the fun begins! (Thank you for sticking with me up to here...)

In my opinion, the ability to transpose data, and the capabilities of PROC TRANSPOSE, are a vitally important functionality of SAS. A quick transposition of the data, to get it into wide versus short, or a narrow versus long, or something in the middle, can get you out of a lot of scrapes.

I use the following simple code

```
/* Transpose data, to create a dataset with a sequence number,
   with the column name from the imported spreadsheet, and with the contents of the cell. */
proc transpose data=CensusSexAgeResized out=CensusSexAgeTransposed(drop=_label_
            rename=(source=Column column1=CellValue)) prefix=column name=source;
    by SeqNo;
    var &varlist.;
run;
```

to convert my SAS dataset from this:

SeqNo	A	B	C	D
00001			Statistics Canada/ Statistique Canada	
00002			Surveys and statistical programs	
00003				
00004	Broad age groups and sex: Canada, provinces and territories			
00005				
00006	Frequency: Occasional			
00007				
00008	Table: 98-10-0034-01			
00009				
00010	Release date: 2022-04-27			
00011	Geography Canada			
00012				
00013	Sex (3)	Total - Sex	Male	Female
00014	Broad age groups (5)			
00015	Total - Age	36991980	18219520	18772465
00016	0 to 14 years	6012795	3083520	2929275
00017	15 to 64 years	23957755	11908295	12049465
00018	65 years and over	7021430	3227700	3793725
00019	85 years and over	861395	319605	541785
00020				
00021				
00022	How to cite: Statistics Canada. Table 98-10-0034-01 Broad age groups and sex: Canada, provinces and territories			
00023	DOI: https://doi.org/10.25318/9810003401-eng			

to this...

SeqNo	Column	CellValue
00001	A	
00001	B	
00001	C	Statistics Canada/ Statistique Canada
00001	D	
00002	A	
00002	B	
00002	C	Surveys and statistical programs
00002	D	
00003	A	
00003	B	
00003	C	
00003	D	
00004	A	Broad age groups and sex: Canada, provinces and territories
00004	B	
00004	C	
00004	D	
00005	A	
00005	B	
00005	C	
00005	D	
00006	A	Frequency: Occasional

You'll note instead of a variable number of rows and columns my SAS dataset now has three columns (Row of the original spreadsheet, Column of the original spreadsheet, and the contents of that cell), and many more rows. The next step is to prepare a template, or pattern, to identify what is expected in each cell.

STEP 4: CREATE A TEMPLATE

Or Step 0. This is created once, and used for every spreadsheet we want to process. Broadly, the way this software works is by matching the transposed spreadsheet to the "template", which is a pre-created pattern that the spreadsheet has to follow. I've delayed talking about it until now because it will make a bit more sense with the results from the last step in front of you.

What we need is an indication of what each cell in the spreadsheet is expected to be. At a high level, which I'll call “category”, we select one of:

- don't care;
- blank;
- boilerplate text;
- data;
- the cell requires special processing.

Reviewing our original spreadsheet:

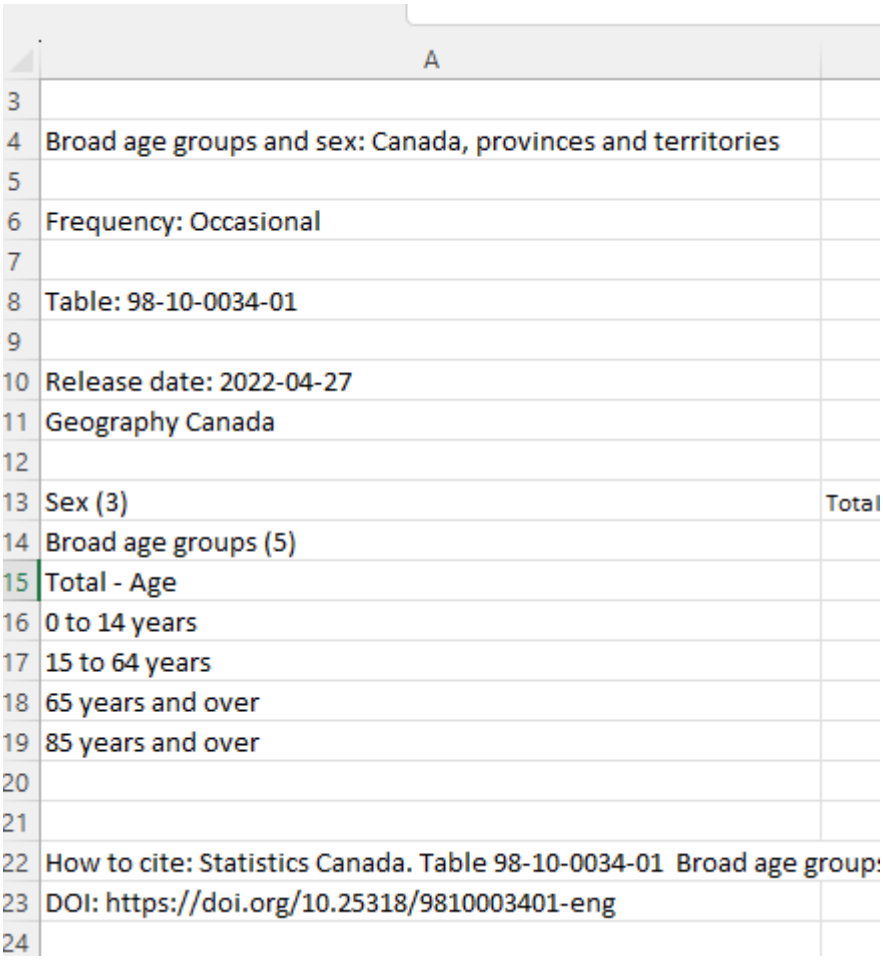
Cells C1, C2, A13, B13, C13, D13, A14 to A19 are boilerplate. We don't need to capture them, because we already know what they are. But we do need to ensure that they contain the expected values.

	A	B	C	D
1			Statistics Canada/ Statistique Canada	
2			Surveys and statistical programs	
3				
4	Broad age groups and sex: Canada, provinces and territories			
5				
6	Frequency: Occasional			
7				
8	Table: 98-10-0034-01			
9				
10	Release date: 2022-04-27			
11	Geography Canada			
12				
13	Sex (3)	Total - Sex	Male	Female
14	Broad age groups (5)			
15	Total - Age	36,991,980	18,219,520	18,772,465
16	0 to 14 years	6,012,795	3,083,520	2,929,275

Our data cells are B15 to B19, C15 to C19, and D15 to D19. (We don't actually need the data from row 15 or row 19, but for simplicity we'll grab it and dispose of it later). We expect to process each of these cells in the identical manner.

	A	B	C	D
9				
10	Release date: 2022-04-27			
11	Geography Canada			
12				
13	Sex (3)	Total - Sex	Male	Female
14	Broad age groups (5)			
15	Total - Age	36,991,980	18,219,520	18,772,465
16	0 to 14 years	6,012,795	3,083,520	2,929,275
17	15 to 64 years	23,957,755	11,908,295	12,049,465
18	65 years and over	7,021,430	3,227,700	3,793,725
19	85 years and over	861,395	319,605	541,785
20				
21				
22	How to cite: Statistics Canada. Table 98-10-0034-01 Broad age groups and sex: Canada, provinces and territories			
23	DOI: https://doi.org/10.25318/9810003401-eng			

Cells A4, A6, A8, A10, A11, A22, and A23 could require special processing, or not, depending on whether we expect them to change over time, or whether they are boilerplate. In our case, we want to process A6, A8, and A10 (they are populated in our resulting data table). Let's assume A4 and A11 are boilerplate, and that we don't care about A22 and A23, which can contain anything and won't be edited.



	A	
3		
4	Broad age groups and sex: Canada, provinces and territories	
5		
6	Frequency: Occasional	
7		
8	Table: 98-10-0034-01	
9		
10	Release date: 2022-04-27	
11	Geography Canada	
12		
13	Sex (3)	Total
14	Broad age groups (5)	
15	Total - Age	
16	0 to 14 years	
17	15 to 64 years	
18	65 years and over	
19	85 years and over	
20		
21		
22	How to cite: Statistics Canada. Table 98-10-0034-01 Broad age groups	
23	DOI: https://doi.org/10.25318/9810003401-eng	
24		

All of the rest of the cells, up to the last expected cell at D23, must be blank.

These details are specified in a template table, which looks like this (some rows excluded):

	A	B	C	D
1	SeqNo	Column	Category	CellData
2	00001	A	2	
3	00001	B	2	
4	00001	C	3	Statistics Canada/ Statistique Canada
5	00001	D	2	
6				...
7	00006	A	5	1
8	00006	B	2	
9	00006	C	2	
10				...
11	00013	A	3	Sex (3)
12	00013	B	3	Total - Sex
13	00013	C	3	Male
14	00013	D	3	Female
15	00014	A	3	Broad age groups (5)
16	00014	B	2	
17				...
18	00019	A	3	85 years and over
19	00019	B	4	
20	00019	C	4	
21	00019	D	4	

I'm repeating the first couple of rows of each category, with a description. Note that there is an additional column, "Celldata", which will also be discussed.

Category 1: Don't care: We only have two of these. The "Celldata" field is null, because we're not going to process these cells at all.

	A	B	C	D	E
1	SeqNo	Column	Category	CellData	
2				...	
3	00022	A	1		
4				...	
5	00023	A	1		
6				...	
7					

Category 2: Blank: It won't be unusual for this to be biggest category, as spreadsheets typically contain a lot of white space. Again "Celldata" is null, because we don't require any further information.

	A	B	C	D
1	SeqNo	Column	Category	CellData
2	00001	A	2	
3	00001	B	2	
4				...
5	00001	D	2	
6				
7				

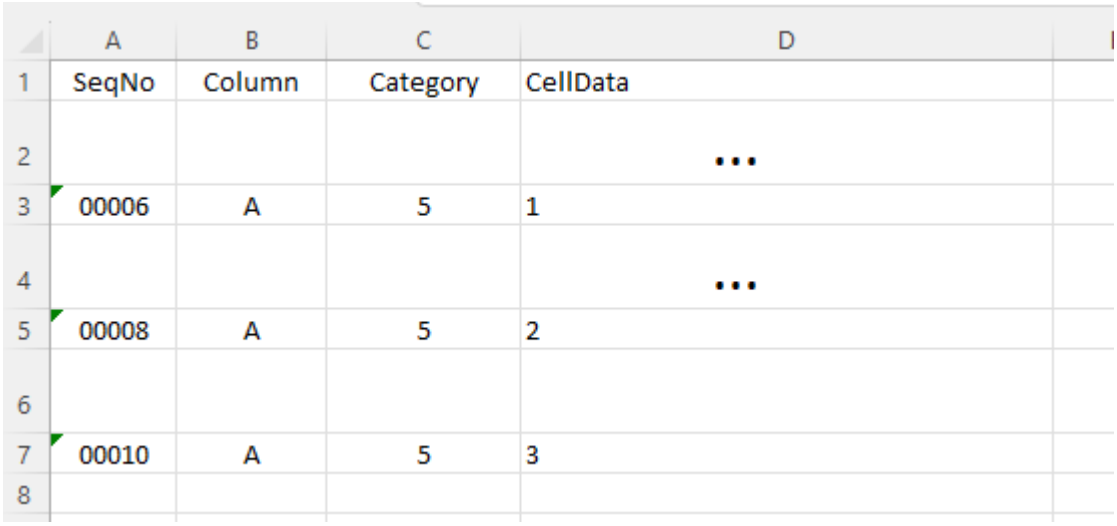
Category 3: Boilerplate: In this case, "Celldata" needs to contain the expected text. In the editing process, the contents of the cell will be compared to this.

	A	B	C	D	E	F
1	SeqNo	Column	Category	CellData		
2						
3	00001	C	3	Statistics Canada/ Statistique Canada		
4				...		
5	00002	C	3	Surveys and statistical programs		
6				...		
7	00004	A	3	Broad age groups and sex: Canada, provinces and territories		
8						

Category 4: Data: And finally, we begin to see the cells that we actually want to capture. Although "Celldata" is empty in this example, when I'm discussing how to capture the data there's a scenario where you might want to use it.

	A	B	C	D
1	SeqNo	Column	Category	CellData
2				...
3	00015	B	4	
4	00015	C	4	
5	00015	D	4	
6				...
7	00016	B	4	
8	00016	C	4	
9	00016	D	4	
10				

Category 5: Special processing: "Celldata" is a key to what kind of processing is required…more about this later.



	A	B	C	D
1	SeqNo	Column	Category	CellData
2				…
3	00006	A	5	1
4				…
5	00008	A	5	2
6				
7	00010	A	5	3
8				

One reason for leaving this discussion until now is that you can use Steps 1 through 3 to create a transposed version of your spreadsheet of interest, and that SAS dataset becomes an excellent beginning model of a template. Just edit it to meet your needs.

Once your template is finished, load it to SAS (I copied and pasted the datalines from an Excel spreadsheet, so I'm using tabs as the column delimiters):

```
/* Load the template */
data ExcelTemplate;
     length SeqNo $5 Column $3 Category 8 TemplateValue $&maxlen.;
     infile cards dlm='09'x missover;
     input SeqNo Column Category TemplateValue;
     cards;
00001       A       2
00001       B       2
00001       C       3       Statistics Canada/ Statistique Canada
                          . . .
00023       B       2
00023       C       2
00023       D       2
run;
```

STEP 5: MATCH THE DATA TO THE TEMPLATE

Now that our data is loaded, our template is completed, and the two are in the same format, we can match them using some very simple SQL:

```
/* Match the data from the spreadsheet with the template */
proc sql noprint;
     create table CombinedCellsForEditing as select c.SeqNo as SourceSeqNo,
            c.Column as SourceColumn, c.CellValue as CellValue, t.SeqNo as TemplateSeqNo,
            t.Column as TemplateColumn, t.Category, t.TemplateValue from
            CensusSexAgeTransposed c full join ExcelTemplate t on (c.SeqNo=t.SeqNo and
            c.Column=t.Column);
quit;
```

This will result in a table that looks like this.

	A	B	C	D	E	F	G
1	SourceSeqNo	SourceColumn	CellValue	TemplateSeqNo	TemplateColumn	Category	TemplateValue
2	1	A		1	A	2	
3	1	B		1	B	2	
4	1	C	Statistics Canada/ Statistique Canada	1	C	3	Statistics Canada/ Statistique Canada
5	1	D		1	D	2	
6	2	A		2	A	2	
7	2	B		2	B	2	
8	2	C	Surveys and statistical programs	2	C	3	Surveys and statistical programs
9	2	D		2	D	2	
10	3	A		3	A	2	
11	3	B		3	B	2	
12	3	C		3	C	2	
13	3	D		3	D	2	
14	4	A	Broad age groups and sex: Canada, province	4	A	3	Broad age groups and sex: Canada, provinces a
15	4	B		4	B	2	
16	4	C		4	C	2	
17	4	D		4	D	2	
18	5	A		5	A	2	
19	5	B		5	B	2	
20	5	C		5	C	2	
21	5	D		5	D	2	
22	6	A	Frequency: Occasional	6	A	5	1
23	6	B		6	B	2	
24	6	C		6	C	2	
25	6	D		6	D	2	
26	7	A		7	A	2	
27	7	B		7	B	2	
28	7	C		7	C	2	
29	7	D		7	D	2	
30	8	A	Table: 98-10-0034-01	8	A	5	2

Because my spreadsheet didn't contain any problems, the rows match perfectly.

STEP 6: PROCESS AS REQUIRED

This table can now be used to:

1. Check that the blank rows and boilerplate rows match exactly;
2. Extract the data from the data rows;
3. Process the "special processing" cells as required. A couple of examples will be provided.

While the details will depend on your requirements, here are some examples:

ENSURE THAT THE SPREADSHEET HAS THE CORRECT ROWS AND COLUMNS

If this query returns any rows…

```
/* Check for any spreadsheet cells beyond the template, or cells that are in the template but
missing from the spreadsheet */
proc sql;
     select * from CombinedCellsForEditing where SourceSeqNo is null or
            TemplateSeqNo is null;
quit;
```

it means that either there are rows and/or columns in the spreadsheet that are not in the template, or there are rows and/or columns in the template that are not in the spreadsheet. If no records are returned, the two have exactly the same rows and columns.

ENSURE THAT ALL THE CELLS THAT SHOULD BE BLANK ARE

If this query returns any rows…

```
/* Check for any spreadsheet cells that should be blank but are not */
proc sql;
     select * from CombinedCellsForEditing where Category=2 and CellValue is not
            null;
```

it means that a cell that is expected to be blank isn't.

ENSURE THAT THE BOILERPLATE TEXT IS CORRECT

If this query returns any rows…

```
/* Check for any spreadsheet cells that should be blank but are not */
proc sql;
     select * from CombinedCellsForEditing where Category=3 and CellValue NE
            TemplateValue;
quit;
```

it means that one or more of the boilerplate entries in the template doesn't match the spreadsheet.

EXTRACT THE DATA CELLS

Here's an example for rows. Columns are the same idea.

If your rows are dependably in the same place on your spreadsheet, you can use code like this:

```
proc format;
     invalue ageif
            "00016" = 1
            "00017" = 2
            "00018" = 3
            other = .;
     value agef
            1="0 to 14 years"
            2="15 to 64 years"
            3="65 years and over";
run;
data RetrievedAge;
     set CombinedCellsForEditing;
     Age=input(SourceSeqNo, ageif.);
     format Age agef.;
run;
```

If the labels for the rows can vary from one run to another, this should work:

```
proc format;
     invalue ageif
            "0 to 14 years" = 1
            "15 to 64 years" = 2
            "65 years and over" = 3
            other = .;
     value agef
            1="0 to 14 years"
            2="15 to 64 years"
            3="65 years and over";
run;
proc sort data=CombinedCellsForEditing out=DataForAge;
     by SourceSeqNo SourceColumn;
run;
data RetrievedAge;
     retain Age;
     set DataForAge;
     if Category=3 then
            Age=input(CellValue, ageif.);
     format Age agef.;
     if ^missing(Age) & Category=4 then
            output;
run;
```

This is where you might want to use "CellData" in the template. If for example you have columns that contain dates, and columns that contain numbers, and columns that contain text, you could use different

numbers in CellData to indicate what kind of processing is required.

DEAL WITH THE CELLS THAT REQUIRE SPECIAL PROCESSING

And finally, we have the "special" cells, that need custom processing. In our case, we have three:

CellData	CellValue
1	Frequency: Occasional
2	Table: 98-10-0034-01
3	Release date: 2022-04-27

My intention is to ensure that they are in the expected format, and to then extract the variable part. This code will handle that. Since they only occur once in the spreadsheet, I would be inclined to handle them as a special case, and drop them into macro variables.

```
data SpecialProcessing;
     length DataFrequency $20 TableID $20 ReleaseDate 8;
     keep DataFrequency TableID ReleaseDate;
     retain DataFrequency TableID ReleaseDate;
     format ReleaseDate date9.;
     retain FieldsFound 0;
     set CombinedCellsForEditing;
     if Category=5 & TemplateValue="1" then
            do;
                   /* Check if the spreadsheet contains "Frequency: xxx",
                      and capture the xxx */
                   prx1 = prxparse("/^Frequency:\s(\S+)\s*$/");
                   if prxmatch(prx1, CellValue)
                   then DataFrequency = prxposn(prx1, 1, CellValue);
                   FieldsFound=FieldsFound + 1;
            end;

     if Category=5 & TemplateValue="2" then
            do;
                   /* Check if the spreadsheet contains "Table: xxx",
                      and capture the xxx */
                   prx2 = prxparse("/^Table:\s(\S+)\s*$/");
                   if prxmatch(prx2, CellValue)
                   then TableID = prxposn(prx2, 1, CellValue);
                   FieldsFound=FieldsFound + 1;
            end;

     if Category=5 & TemplateValue="3" then
            do;
                   /* Check if the spreadsheet contains "Release date: 9999-99-99",
                      and capture the 9999-99-99, converting it to a date */
                   prx3 = prxparse("/^Release date:\s(\S+)\s*$/");
                   if prxmatch(prx3, CellValue)
                   then ReleaseDate = input(prxposn(prx3, 1, CellValue), yymmdd10.);
                   FieldsFound=FieldsFound + 1;
            end;

     if FieldsFound = 3 then
            do;
                   output;
                   FieldsFound=0;
            end;
run;
```

CORRECTING ERRORS

In some cases, it should be possible to use SAS to correct errors in the incoming spreadsheet. I'll illustrate the simplest one below. In this input spreadsheet, I "accidentally" applied a format to a cell in

column E. This causes SAS to think that the spreadsheet extends to column E. If you run the code from "Ensure that the spreadsheet has the correct rows and columns", you'll get the following result:

SourceSeqNo	NAME OF FORMER VARIABLE	CellValue	TemplateSeqNo	TemplateColumn	Category	TemplateValue
00001	E				.	
00002	E				.	
00003	E				.	
00004	E				.	
00005	E				.	
00006	E				.	
00007	E				.	
00008	E				.	
00009	E				.	
00010	E				.	
00011	E				.	
00012	E				.	
00013	E				.	
00014	E				.	
00015	E				.	
00016	E				.	
00017	E				.	
00018	E				.	
00019	E				.	
00020	E				.	
00021	E				.	
00022	E				.	
00023	E				.	

The following code will check that all columns past the rightmost expected one contain empty cells, and if the cells are all blank, they'll be removed.

```
/* Find the rightmost expected cell */
/* Note this is an example...will only work with one-character column names */
proc sql noprint;
     select max(Column) into :maxcol from ExcelTemplate;
quit;

%put &maxcol.;

/* Eliminate cells to the right, if it's safe to do so */
data RevisedData;
     set CombinedCellsForEditing;

     if SourceColumn > "&maxcol." /* Cell is further right than expected */
     then
            if missing(CellValue) /* The cell is empty */
            then
                   delete; /* Get rid of the cell */
            else
                   abort; /* put your error processing here */
     run;
```

Here's another example where I find rows composed completely of empty cells:

```
/* Tabulate row number by contents */
```

```
proc means data=CombinedCellsForEditing missing nway noprint;
     class SourceSeqNo CellValue;
     output out=RowCounts1(drop=_TYPE_ _FREQ_) n()=RowCount;
run;

/* Tabulate by row number, keep the max of CellValue. Should be one line per row number output
*/
proc means data=RowCounts1 missing nway noprint;
     class SourceSeqNo;
     output out=RowCounts2(drop=_TYPE_ _FREQ_) n()=RowCount idgroup (max(CellValue) obs
            out (CellValue)=CellValue);
run;

/* If there is only one row type per row number, and if the max cellvalue is blank, keep this
record */
data BlankRows(keep=SourceSeqNo);
     set RowCounts2;

     if RowCount=1 & missing(CellValue) then
            output;
run;
```

CONCLUSION

I hope you find this technique of some use. I believe it should be extensible to some very large spreadsheets, as the row count increases with increasing spreadsheet size.

ACKNOWLEDGMENTS

All of the code used for this paper was developed and tested using SAS On Demand for Academics. Over the years, SODA is evolving into a very capable platform for evaluating and learning SAS. I recommend it highly!

CONTACT INFORMATION

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

Tom Kari
Tom Kari Consulting Inc
Tom.kari.consulting@bell.net