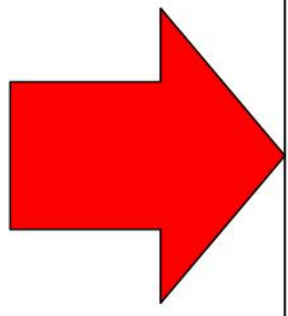


Using Microsoft Excel to write sas code

Andrew Boyd, Quanticate, Edinburgh, UK



When we write SAS code in the pharmaceutical industry, there is often a high level of repetition. This poster describes a method of writing SAS code that will reduce the overall time taken to write the code and make large scale amendments more easily and quickly.

Microsoft Excel - Programming Plan					
File Edit View Insert Format Tools Data Window Help					
A1 Name					
A	B	C	D	E	
1	Name	Length	Format	Label	Criteria
2	Field1	8	date9.	Date	Use xlib.date where datatype=1
3	Field2	8	datetime20.	Datetime	Use xlib.date where datatype=2
4	Field3	\$12		Big String	Use xlib.string where stype = 'Big'
5	Field4	\$4		Small String	Use xlib.string where stype = 'Small'
6	Field5	8		Number 1	Use xlib.num where ntype = 'A'
7	Field6	8		Number 2	Use xlib.num where ntype = 'B'
8	Field7	\$8		Text 1	Use xlib.text where ttype = 1
9	Field8	\$8		Text 2	Use xlib.text where ttype = 2
10	Field9	8		Number 3	Use xlib.num where ntype = 'C'
11	Field10	8		Number 4	Use xlib.num where ntype = 'D'
12	Field11	8		Number 5	Use xlib.num where ntype = 'E'

="%excel_example("&LOWER(A3)
& ", "&F3& ", %str("&G3&"))";

F	G
Source	Where
xlib.date	datatype=1
xlib.date	datatype=2
xlib.string	stype = 'Big'
xlib.string	stype = 'Small'
xlib.num	ntype = 'A'
xlib.num	ntype = 'B'
xlib.text	ttype = 1
xlib.text	ttype = 2
xlib.num	ntype = 'C'
xlib.num	ntype = 'D'
xlib.num	ntype = 'E'

=MID(E3,SEARCH("where",E3,1)
+6,LEN(E3)-SEARCH
("where",E3,1))

=MID(E2,SEARCH("
";E2,1)+1,SEARCH("
";E2,SEARCH("
";E2,1)+1)-SEARCH
("";E2,1))



H
Macro
%excel_example(field1, xlib.date, %str(datatype=1));
%excel_example(field2, xlib.date, %str(datatype=2));
%excel_example(field3, xlib.string, %str(stype = 'Big'));
%excel_example(field4, xlib.string, %str(stype = 'Small'));
%excel_example(field5, xlib.num, %str(ntype = 'A'));
%excel_example(field6, xlib.num, %str(ntype = 'B'));
%excel_example(field7, xlib.text, %str(ttype = 1));
%excel_example(field8, xlib.text, %str(ttype = 2));
%excel_example(field9, xlib.num, %str(ntype = 'C'));
%excel_example(field10, xlib.num, %str(ntype = 'D'));
%excel_example(field11, xlib.num, %str(ntype = 'E'));

Macro calls

One of the fastest ways to write SAS code that does a similar task many times over is to use the SAS macro language. Unfortunately once your macro has been written, you also need to write the macro invocations, each with specific parameters that will amend the SAS code that is executed during each invocation. Excel can be used to write these macro invocations in a similar way to the attrib statements described on the left. The programming plan for our example dataset contains the criteria to be used to derive each one of the fields (top left). As the criteria for each field is similar, we could use the following macro to derive every field:

```
%macro excel_example(field, source, where);  
  data &field;  
    set &source (keep=subjid &field);  
    where &where;  
  run;  
%mend;
```

The attrib statement

One way of specifying the attributes of a new or existing variable in SAS is to use the attrib statement. The attrib statement can be used to set the name, length, type, format and label of a field. Obviously, writing an attrib statement to set these attributes for every single field for a large output dataset would take a long time; this is where Excel can help.

Excel has the ability to concatenate strings using the ampersand (&) operator. It is also possible to perform operations during the concatenation.

To create the first attrib statement we can put the following formula in to cell F2:

= "attrib "&LOWER
(A2)&" length = "&IF
(C2="Char","\$","")
&B2&IF(D2<>"", "
format = "&D2,"")&"
label = "&E2&"

This formula will produce:

attrib field1 length = 8 format = date9. label = 'Date';

Use the colours to see which part of the formula produces each part of the string. Notice that the part in green does not produce anything for this attrib statement. This is because this part of the formula is the part that will include a "\$" to denote a character field, and the first field is numeric.

The cell F2 can now be filled down by selecting the cell and double clicking on the bottom right-hand corner of the selection box. This has the same effect as copying F2 and pasting it in to every other cell in column F where there is a non blank cell in column E for the same row. (This is how the fill down method knows where to stop).

Microsoft Excel - Programming Plan					
File Edit View Insert Format Tools Data Window Help					
A1 Name					
A	B	C	D	E	F
1	Name	Length	Type	Format	Label
2	Field1	8	Num	date9.	Date
3	Field2	8	Num	datetime20.	Datetime
4	Field3	12	Char		Big String
5	Field4	4	Char		Small String
6	Field5	8	Num	Number 1	attrib field5 length = 8 label = 'Number 1';
7	Field6	8	Num	Number 2	attrib field6 length = 8 label = 'Number 2';
8	Field7	8	Char	Text 1	attrib field7 length = \$8 label = 'Text 1';
9	Field8	8	Char	Text 2	attrib field8 length = \$8 label = 'Text 2';
10	Field9	8	Num	Number 3	attrib field9 length = 8 label = 'Number 3';
11	Field10	8	Num	Number 4	attrib field10 length = 8 label = 'Number 4';
12	Field11	8	Num	Number 5	attrib field11 length = 8 label = 'Number 5';

Now that the parameters required by our macro have been extracted, the macro invocations can be created by concatenating them together along with the necessary SAS code using the ampersand (&) operator:

The formula: ="%excel_example("&LOWER(A2)&" , "&C2&" , %str("&D2&"))";
Will now produce: %excel_example(field1, xlib.date, %str(datatype=1));

As the where criteria are likely to contain SAS special characters that might upset the macro process, the where criteria have been enclosed in the %str macro function that will suppress any special characters until the parameter is evaluated. This formula can now be filled down to provide the macro invocation for every field in the programming plan.

Comments

Entering comments into SAS code can often be one of the most tedious activities any programmer can undertake. However SAS code that is commented effectively is much easier to maintain and debug than plain SAS code. By making a small addition to the formula that produces the macro invocations we can put a comment above each macro invocation based on the information contained in the programming plan:

="/*"&A2&" - "&B2&"*/"%excel_example("&LOWER(A2)&" , "&C2&" , %str("&D2&"))";

This macro needs three parameters to derive the fields: field (field name), source (source dataset) and where (where clause).

In order to construct our macro invocation we first need to extract the parameters required from the programming plan. The best way to examine the programming plan and identify characters/strings that come before and after the sections of the criteria field we want to extract. The Excel MID() function can then be used to extract them.

Source

The source dataset and library appear after the first space and before the second space. The first space can be found using:

=SEARCH(" ",B2,1)

Using the result of this formula we can find the second space by incrementing its result by 1 and using this as the starting position for a second SEARCH():

=SEARCH(" ",B2,SEARCH(" ",B2,1)+1)

Now we have start and end positions for the source dataset name we can use the MID() function to extract the name from the criteria:

=MID(B2,SEARCH(" ",B2,1),SEARCH(" ",B2,SEARCH(" ",B2,1)+1)-SEARCH(" ",B2,1))

This formula can now be filled down by selecting the cell and double clicking on the bottom right-hand corner of the selection box.

Where

The where clause can be extracted from the criteria in a similar way using the word "where" to determine its starting position, and the length of the criteria field as the end:

=MID(B2,SEARCH("where",B2,1)+6,LEN(B2)-SEARCH("where",B2,1))

(Notice the +6 in the middle of the formula, this is used to advance the starting position of the MID function past the word "where".)



```
%macro excel_example(field, source, where);  
  
  data &field;  
    set &source (keep=subjid &field);  
    where &where;  
  run;  
  
%mend;  
  
/*Field1 - Use xlib.date where datatype=1*/  
%excel_example(field1, xlib.date, %str(datatype=1));  
/*Field2 - Use xlib.date where datatype=2*/  
%excel_example(field2, xlib.date, %str(datatype=2));  
/*Field3 - Use xlib.string where stype = 'Big'*/  
%excel_example(field3, xlib.string, %str(stype = 'Big'));  
/*Field4 - Use xlib.string where stype = 'Small'*/  
%excel_example(field4, xlib.string, %str(stype = 'Small'));  
/*Field5 - Use xlib.num where ntype = 'A'*/  
%excel_example(field5, xlib.num, %str(ntype = 'A'));  
/*Field6 - Use xlib.num where ntype = 'B'*/  
%excel_example(field6, xlib.num, %str(ntype = 'B'));  
/*Field7 - Use xlib.text where ttype = 1*/  
%excel_example(field7, xlib.text, %str(ttype = 1));  
/*Field8 - Use xlib.text where ttype = 2*/  
%excel_example(field8, xlib.text, %str(ttype = 2));  
/*Field9 - Use xlib.num where ntype = 'C'*/  
%excel_example(field9, xlib.num, %str(ntype = 'C'));  
/*Field10 - Use xlib.num where ntype = 'D'*/  
%excel_example(field10, xlib.num, %str(ntype = 'D'));  
/*Field11 - Use xlib.num where ntype = 'E'*/  
%excel_example(field11, xlib.num, %str(ntype = 'E'));
```