Paper 150-2019

Spec to Code in Seconds:

Use Microsoft Excel for Repetitive Code Writing

Pinky Anandani Dutta, Axio Research

ABSTRACT

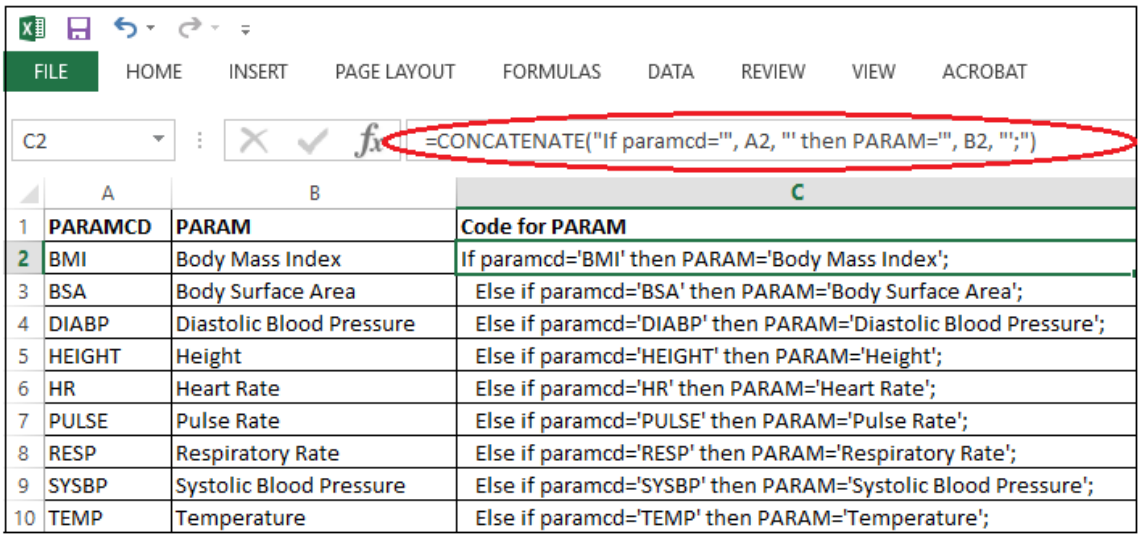
The role of a SAS® programmer in the Pharmaceutical industry has become very versatile over the years. SAS® programmers are no longer just coding using SAS® but also performing many other functions including writing programming specifications, annotating Case Report Forms (CRF) and mocks shells, preparing documents for a New Drug Application (NDA), etc. As such, knowledge of robotic features in other tools and editors can come in very handy. One such tool is Microsoft Excel, which is relatively easy to learn and widely used already. This paper will discuss how programming specifications that require repetitive code can be transformed into SAS® code in seconds by taking advantage of different functions in Excel. This paper assumes that readers have a working knowledge of Microsoft Excel and Clinical programming in SAS®.

INTRODUCTION

With the increasing number of clinical studies, SAS® programmers are always looking for new ways to reduce coding time and increase efficiency. Repetitive coding such as if-else statements for example, often seem simple to write yet consumes a lot of time when typed up manually. Manual typing of code also introduces the risk of human errors such as typos, missing semicolon, missing quotation marks, etc. This paper will discuss in detail three specific examples of the use of Excel to generate repetitive code commonly used in clinical programming.

GENERATING IF – ELSE STATEMENTS

If – Else Statements are very commonly used in a SAS® program to perform different actions or assign results based on different conditions. If the conditions to these statements are based on set values, those conditions and their desired results or actions can be placed in different columns of an Excel spreadsheet, and the resulting SAS® code can be generated in a new column using the CONCATENATE function in Excel as shown in Display 1, where Column A denotes the first condition's value, and column B denotes the second condition's values.



	A	B	C
1	**PARAMCD**	**PARAM**	**Code for PARAM**
2	BMI	Body Mass Index	If paramcd='BMI' then PARAM='Body Mass Index';
3	BSA	Body Surface Area	Else if paramcd='BSA' then PARAM='Body Surface Area';
4	DIABP	Diastolic Blood Pressure	Else if paramcd='DIABP' then PARAM='Diastolic Blood Pressure';
5	HEIGHT	Height	Else if paramcd='HEIGHT' then PARAM='Height';
6	HR	Heart Rate	Else if paramcd='HR' then PARAM='Heart Rate';
7	PULSE	Pulse Rate	Else if paramcd='PULSE' then PARAM='Pulse Rate';
8	RESP	Respiratory Rate	Else if paramcd='RESP' then PARAM='Respiratory Rate';
9	SYSBP	Systolic Blood Pressure	Else if paramcd='SYSBP' then PARAM='Systolic Blood Pressure';
10	TEMP	Temperature	Else if paramcd='TEMP' then PARAM='Temperature';

Display 1. Microsoft Excel Spreadsheet Example for Generating If – Else Statements

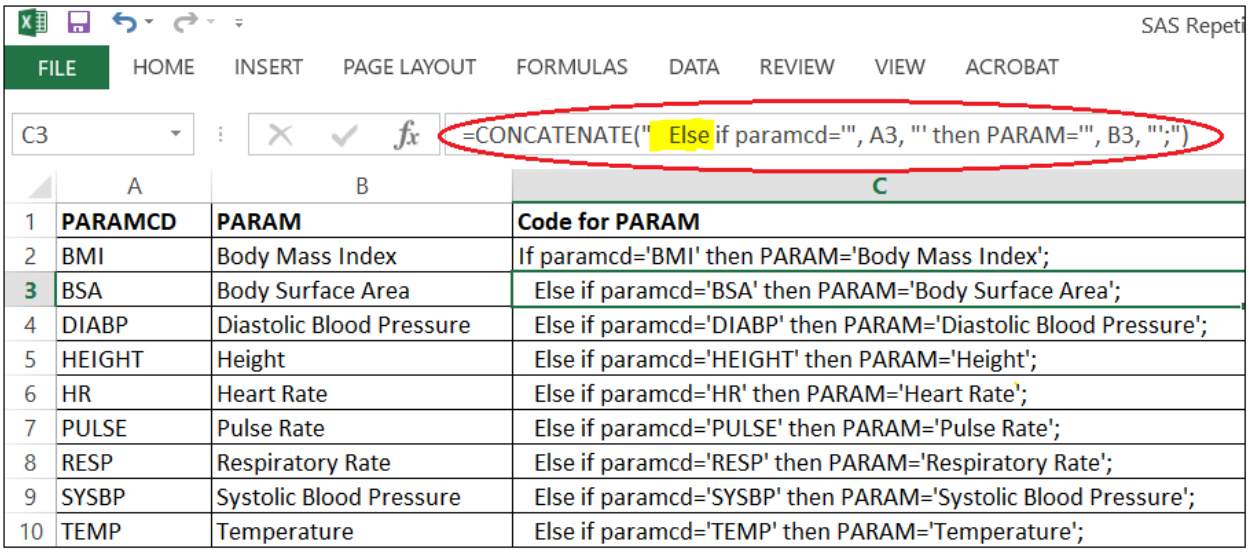
Display 1 makes use of the CONCATENATE function in Excel to join information from two different columns appropriately to create an If – Else code for the PARAM variable.

The CONCATENATE function in Microsoft Excel is used to join two or more text strings into one string. This function concatenates (joins) up to 30 items together and returns the result as a text. Arguments to this function can be a text value, a number or an Excel cell reference. The text items, also known as arguments to the function are listed in the order in which they need to be concatenated, separated by a comma, and quoted by quotation marks whenever the argument is not an Excel cell reference[1]. Note that it is however not required to quote numbers as an argument, but it is recommended.

In order to write a function in Excel, first, double click on the cell where the result of the function is desired. Next, start by typing an equal sign. This indicates to Excel that the user is starting to type a function. Thereafter, type in the name of the desired function, CONCATENATE in this case, followed by a set of parenthesis that contains the arguments to the function, separated by a comma. Press the ENTER key on your keyboard when done to execute the function.

The CONCATENATE function shown in Display 1 consists of a total of 5 arguments, of which the second and fourth arguments, A2 and B2 are Excel cell references. Note that the text arguments listed here, arguments one, three and five, are quoted using double quotation marks. Since some of the text in these arguments contain single quotation marks, it is crucial to quote the argument with double quotation marks to allow the single quotations to be treated as text. Text arguments can be quoted using single quotation marks in the absence of any apostrophes or single quotation marks that are part of the text.

The CONCATENATE function shown in Display 1 only creates the first line of code, for Cell C2. In order to generate Cells C3 to C10, the concatenation function is slightly modified as shown in Display 2.



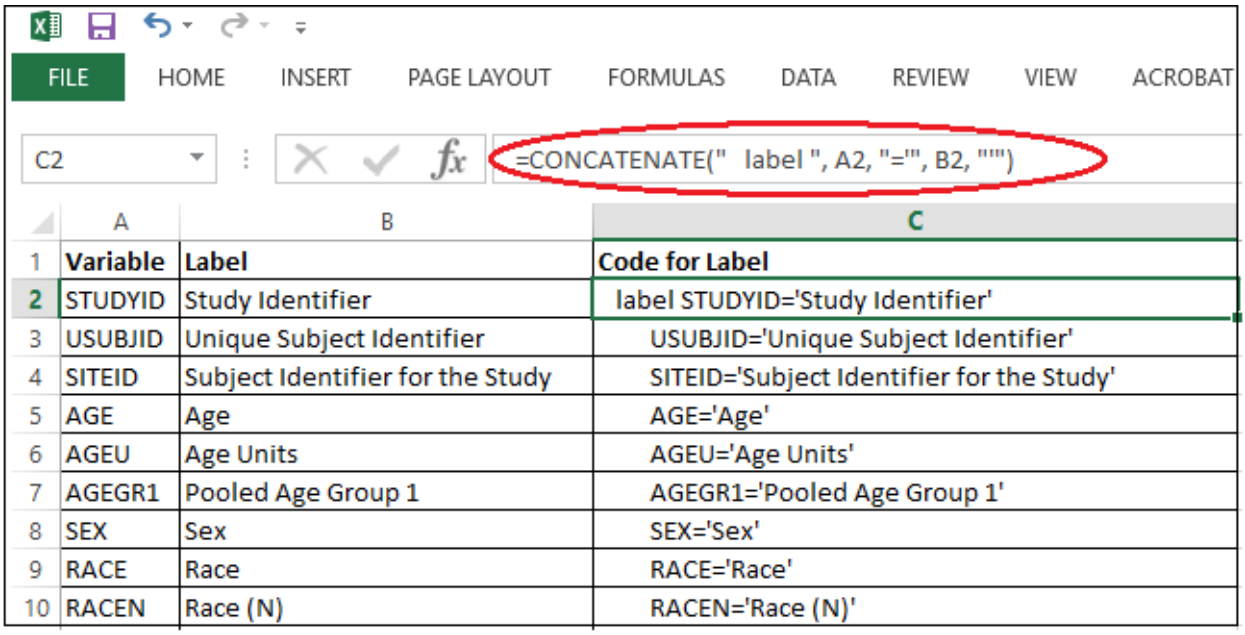
	A	B	C
1	**PARAMCD**	**PARAM**	**Code for PARAM**
2	BMI	Body Mass Index	If paramcd='BMI' then PARAM='Body Mass Index';
3	BSA	Body Surface Area	Else if paramcd='BSA' then PARAM='Body Surface Area';
4	DIABP	Diastolic Blood Pressure	Else if paramcd='DIABP' then PARAM='Diastolic Blood Pressure';
5	HEIGHT	Height	Else if paramcd='HEIGHT' then PARAM='Height';
6	HR	Heart Rate	Else if paramcd='HR' then PARAM='Heart Rate';
7	PULSE	Pulse Rate	Else if paramcd='PULSE' then PARAM='Pulse Rate';
8	RESP	Respiratory Rate	Else if paramcd='RESP' then PARAM='Respiratory Rate';
9	SYSBP	Systolic Blood Pressure	Else if paramcd='SYSBP' then PARAM='Systolic Blood Pressure';
10	TEMP	Temperature	Else if paramcd='TEMP' then PARAM='Temperature';

Display 2. Microsoft Excel Spreadsheet Example for Generating If – Else Statements

Similar to the example shown in Display 1, Display 2 shows how the CONCATENATE function can be modified (as highlighted) for subsequent lines starting at cell C3. In order to do so, set up the revised formula in cell C3, position the mouse in the lower right-hand corner of the cell until a plus sign appears, and double-click the plus sign. This will copy the formula down as far as Excel finds data to the left. Cells C2 to C10 can then be copied from the above table display and pasted directly into a SAS® program DATA step. In the future, the above code generator setup can be re-used for a different set of PARAM/PARAMCD values by replacing the values of PARAM and PARAMCD in Columns A and B.

GENERATING LABEL ASSIGNMENT STATEMENT

Label statement is used in a SAS® program to assign variable labels to variables in a SAS® dataset. Similar to the If – Else statements, the label statement is also based on a fixed number of input conditions, the variable name and its desired label. The variable names and its respective labels can be placed into two separate columns of an Excel spreadsheet, and the label statement code can be generated in a third column using the CONCATENATE function as shown in Display 3.



	A	B	C
1	**Variable**	**Label**	**Code for Label**
2	STUDYID	Study Identifier	label STUDYID='Study Identifier'
3	USUBJID	Unique Subject Identifier	USUBJID='Unique Subject Identifier'
4	SITEID	Subject Identifier for the Study	SITEID='Subject Identifier for the Study'
5	AGE	Age	AGE='Age'
6	AGEU	Age Units	AGEU='Age Units'
7	AGEGR1	Pooled Age Group 1	AGEGR1='Pooled Age Group 1'
8	SEX	Sex	SEX='Sex'
9	RACE	Race	RACE='Race'
10	RACEN	Race (N)	RACEN='Race (N)'

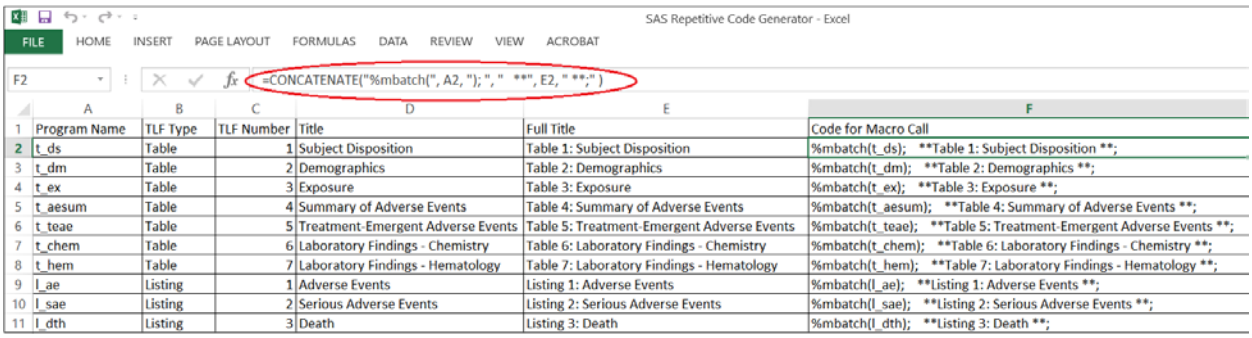
Display 3. Microsoft Excel Spreadsheet Example for Generating Label Assignment Statement

Display 3 shows another example of the use of Microsoft Excel for generating repetitive code for a common clinical programming task. The CONCATENATE function displayed here only creates the first line of code, for Cell C2. In order to generate Cells C3 to C10, the function will need to be slightly modified as demonstrated in Display 2, to remove the keyword label and add or remove spaces as desired. Thereafter, cells C2 to C10 can easily be copied from the above table display and pasted directly into a SAS® program DATA step. Please note that the label statement will need to be closed in the SAS® program with a semi-colon. Considering the importance and need to add variable labels to almost every SAS® dataset that is created, the above code generator setup can come in very handy for different sets of Variables/Labels by replacing the values in Columns A and B, which automatically updates Column C each time there's a change in Column A or B.

GENERATING MACRO CALL STATEMENTS

During the course of clinical programming, there's often a need to use macros by passing different macro parameters and calling a macro several times. A very common situation is during the creation of a batch program for running all of the Table, Listing and Figure (TLF) programs in a clinical study sequentially.

Display 4 is a sample display of the Tables, Listings and Figures needed for a clinical study project along with the name of the SAS® program used to create the respective report.



	A	B	C	D	E	F
1	Program Name	TLF Type	TLF Number	Title	Full Title	Code for Macro Call
2	t_ds	Table	1	Subject Disposition	Table 1: Subject Disposition	%mbatch(t_ds); **Table 1: Subject Disposition **;
3	t_dm	Table	2	Demographics	Table 2: Demographics	%mbatch(t_dm); **Table 2: Demographics **;
4	t_ex	Table	3	Exposure	Table 3: Exposure	%mbatch(t_ex); **Table 3: Exposure **;
5	t_aesum	Table	4	Summary of Adverse Events	Table 4: Summary of Adverse Events	%mbatch(t_aesum); **Table 4: Summary of Adverse Events **;
6	t_teae	Table	5	Treatment-Emergent Adverse Events	Table 5: Treatment-Emergent Adverse Events	%mbatch(t_teae); **Table 5: Treatment-Emergent Adverse Events **;
7	t_chem	Table	6	Laboratory Findings - Chemistry	Table 6: Laboratory Findings - Chemistry	%mbatch(t_chem); **Table 6: Laboratory Findings - Chemistry **;
8	t_hem	Table	7	Laboratory Findings - Hematology	Table 7: Laboratory Findings - Hematology	%mbatch(t_hem); **Table 7: Laboratory Findings - Hematology **;
9	l_ae	Listing	1	Adverse Events	Listing 1: Adverse Events	%mbatch(l_ae); **Listing 1: Adverse Events **;
10	l_sae	Listing	2	Serious Adverse Events	Listing 2: Serious Adverse Events	%mbatch(l_sae); **Listing 2: Serious Adverse Events **;
11	l_dth	Listing	3	Death	Listing 3: Death	%mbatch(l_dth); **Listing 3: Death **;

Display 4. Microsoft Excel Spreadsheet Example for Generating Macro Call Statements

Display 4 shows a bit more of a complex example of the use of Microsoft Excel for generating macro calls for a TLF batch program. Column F of this display shows the generated code for the macro calls where the macro mbatch is called by passing the name of the SAS® program of the individual report from Column A, as a parameter. In addition to the macro call, Column F also shows the concatenation of a commented SAS® code which displays the full title of the individual report from Column E. As before, the CONCATENATE function in Excel works by simply entering into the function all of the different texts (arguments of the function) to be joined together and separating them by a comma. In this case, the function has a total of 6 arguments that are joined together. These texts are as follows:

1. “%mbatch(”
2. Value from Column A, starting at cell A2
3. “);”
4. “ **”
5. Value from Column E, starting at cell E2
6. “ **;”

Note that all texts are bound in quotation marks, while the value from other columns are referenced using the Column name and cell number without any quotation marks.

Setting up macro call statements as described in this example in a Tracker or Table of Contents spreadsheet of a study, has proven very helpful during the programming phase. It was noted that once all of the SAS® program names and TLF titles were entered correctly into the Excel file, it was relatively easy for anyone working on the project to copy the macro calls into a batch program for continuous use. This method is found to be much easier than manually typing in macro calls into a batch program, which is tedious, time consuming and also introduces the risk of typos and human errors.

COMMON PROBLEMS

While writing a function in Excel to generate repetitive code is fairly simple, it is quite easy to make mistakes while typing up the arguments to the function and run into issues. Sticking to the rules is key to avoiding such issues. Microsoft has provided some documentation[1] on its website on common problems a user may encounter when using the CONCATENATE function incorrectly. Review of those problems and its solutions is highly recommended.

CONCLUSION

It is evident that the use of Microsoft Excel in writing repetitive code can aid in reduced programming time and increased efficiency. This technique can be used by programmers of any skill level. Higher efficiency can lead to better quality and unused programming time could in turn benefit an organization in other areas of process development. Given how common such repetitive tasks are in clinical programming, a code generator setup in Excel can come in very handy for different sets of input parameters. Replacing the values of the input columns will automatically update the generated code column each time there’s a

change in the input columns. This code generator setup can then be used conveniently like a calculator from one program to another, one study after the other, for a wide range of repetitive programming tasks in addition to those demonstrated in this paper.

REFERENCES

1. "CONCATENATE function." *Office Support*, Microsoft. Available at https://support.office.com/en-us/article/concatenate-function-8f8ae884-2ca8-4f7a-b093-75d702bea31d.

ACKNOWLEDGMENTS

The author would like to thank Lyma Faroz for her valuable feedback and suggestions.

RECOMMENDED READING

"CONCATENATE function." *Office Support*, Microsoft. Available at https://support.office.com/en-us/article/concatenate-function-8f8ae884-2ca8-4f7a-b093-75d702bea31d.

CONTACT INFORMATION

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

Pinky Anandani Dutta
Axio Research
2601 4th Avenue, Suite 200
Seattle, WA 98121
pnanandani@gmail.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.