Paper CT08

Improving your SAS code without SAS: Using Notepad++ to super-charge your coding

Simon Purkess, Phastar, London, UK

ABSTRACT

While the SAS® text editor has limited functionality for smart code-writing, dedicated text editors such as Notepad++ come packed with features that can make coding faster and more accurate. This paper explores a few of the tools available in Notepad++; the use of simultaneous editing to create similar lines of code at once, how to use regular expressions to re-shape text from outside sources (e.g. SAS variable attributes from a dataset specification) into SAS code, and how to record macro routines to rerun common tasks at the click of a mouse.

INTRODUCTION

When writing SAS code, the obvious place to start is the SAS Enhanced editor. While it does come with a few productivity features such as a basic Find & Replace and Abbreviations, other dedicated text editors provide a host of features that can streamline the code-writing process. In this paper I will be sharing some examples of how I have used Notepad++ in my work, illustrating some of functionality that is available and to encourage readers to explore further and find use cases for their programming.

ABOUT Notepad++

Notepad++ is a source code editor for MS Windows, designed by developer Don Ho in 2003 as an upgrade to Microsoft's Notepad text editor. It is free to download and use with an extensive feature list, as well as the ability to install plugins which provide even more tools. For example the "Compare" plugin allows you to compare two text files (such as two versions of a SAS program) and will highlight differences between them.

Notepad++ can colour-code syntax in a similar way to how it is done in the SAS Enhanced editor; colouring keywords and comments to improve readability. While SAS is not included as a language option at install, it is simple enough to import a SAS language file (see references for guide).

Figure 1: Example snippet of code before and after applying the SAS language file.

Before

```
**------------------------------------------------------------------------------------------------------------**;
** VARS FROM RAW.DS                                                                                           **;
**------------------------------------------------------------------------------------------------------------**;

data ds1;
  attrib rfpendtc length=$19 label="Date/Time of End of Participation";
  set raw.ds (keep = subject dsstdat dsdecod);
  if dsstdat ne "" then rfpendtc=dsstdat;
run;
```

After

```
**------------------------------------------------------------------------------------------------------------**;
** VARS FROM RAW.DS                                                                                           **;
**------------------------------------------------------------------------------------------------------------**;

data ds1;
  attrib rfpendtc length=$19 label="Date/Time of End of Participation";
  set raw.ds (keep = subject dsstdat dsdecod);
  if dsstdat ne "" then rfpendtc=dsstdat;
run;
```

Some basic features of Notepad++ include:

- Bracket-matching – when the cursor is next to a bracket it will highlight the matching one.
- Keyword matching – when a word is selected, other instances of that word will be highlighted. This is often useful when looking for instances where a dataset name or variable name have been used.
- Tabs – multiple tabs can be opened at once, and Notepad++ will remember open tabs from the previous session. If an open file is updated elsewhere, Notepad++ will notice and give the option to load the updated version.

COLUMN EDITOR

A key to writing clear SAS code is to minimise repetition with the use of macros. However, sometimes you may need to write or edit code where the same text needs to be written on each line. Rather than using copy & paste on each individual line, Notepad++ allows you to set multiple cursors by holding down the Alt key (either dragging down a column or clicking each cursor location) and then whatever you type will appear in each selected location.

Figure 2: Example of typing across multiple lines

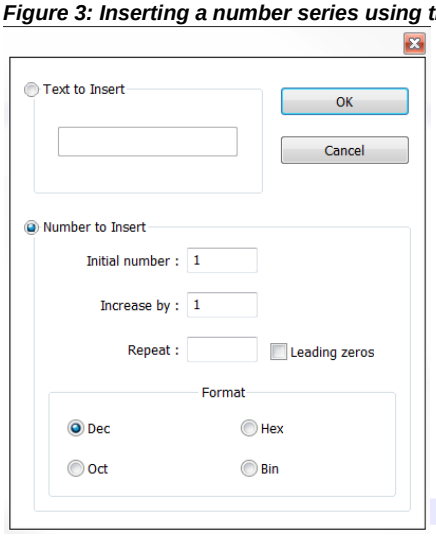
```
if qstestcd='QSTN01' | 'Q-Mobility';
if qstestcd='QSTN02' | 'Q-Self-Care';
if qstestcd='QSTN03' | 'Q-Usual Activities';
if qstestcd='QSTN04' | 'Q-Pain/Discomfort';
if qstestcd='QSTN05' | 'Q-Anxiety/Depression';
if qstestcd='QSTN06' | 'Q-EQ VAS Score';
```

```
if qstestcd='QSTN01' then qstes 'Q-Mobility';
if qstestcd='QSTN02' then qstes 'Q-Self-Care';
if qstestcd='QSTN03' then qstes 'Q-Usual Activities';
if qstestcd='QSTN04' then qstes 'Q-Pain/Discomfort';
if qstestcd='QSTN05' then qstes 'Q-Anxiety/Depression';
if qstestcd='QSTN06' then qstes 'Q-EQ VAS Score';
```

```
if qstestcd='QSTN01' then qstest = 'Q-Mobility';
if qstestcd='QSTN02' then qstest = 'Q-Self-Care';
if qstestcd='QSTN03' then qstest = 'Q-Usual Activities';
if qstestcd='QSTN04' then qstest = 'Q-Pain/Discomfort';
if qstestcd='QSTN05' then qstest = 'Q-Anxiety/Depression';
if qstestcd='QSTN06' then qstest = 'Q-EQ VAS Score';
```

Sometimes you may wish to create a numbered series of variables, and the Column Editor can do this too! After setting your cursors, go to “Edit” > “Column Editor” and use ‘Number to Insert’.

Figure 3: Inserting a number series using the Column Editor menu




```
if qstestcd='QSTN01' then qstest = 'Q1-Mobility';
if qstestcd='QSTN02' then qstest = 'Q2-Self-Care';
if qstestcd='QSTN03' then qstest = 'Q3-Usual Activities';
if qstestcd='QSTN04' then qstest = 'Q4-Pain/Discomfort';
if qstestcd='QSTN05' then qstest = 'Q5-Anxiety/Depression';
if qstestcd='QSTN06' then qstest = 'Q6-EQ VAS Score';
```

FIND & REPLACE

One of Notepad++'s most useful tools is the Find & Replace functionality. As with the Enhanced Editor, Notepad++ allows you to search through a program for a string or a regular expression. However, there are several extra features available in Notepad++:

- "Extended" search mode allows you to match special characters including newlines and tabs.
- Counting the number of matches in the document.
- Search with a highlighted section, or across every open document.

The real power in this tool comes in the 'Find in Files' tab. Using this you can search through a whole directory and create a list of matches, or even perform a directory-wide replace. **WARNING: 'Replace in Files' should be done with great care!** Replacing a string across a directory can be very useful as it is faster and more accurate than a manual replace, but there is no undo option if it goes wrong. With this in mind, I would recommend the following safeguards:

- 'Replace in Files' should only be used by (or with the supervision of) the programmer in charge of a project area.
- Always run a 'Find All' before running 'Replace in Files' and review the list of matches to check that there are no false positives or false negatives. You can save this list as a log of what has been changed.
- Use the 'Filters' box to restrict your search to the file type(s) you are interested in – e.g. "*.sas" for SAS files or "*.bat" for batch files.
- Check which options are selected – 'Match whole word only', 'Match case' and 'In all sub-folders' may or may not be what you want in a given situation.
- Regex with care – 'Replace in Files' can be used in conjunction with regular expressions for more flexible replacing, but this comes with a greater risk of unexpected results.

NOTEPAD++ MACROS

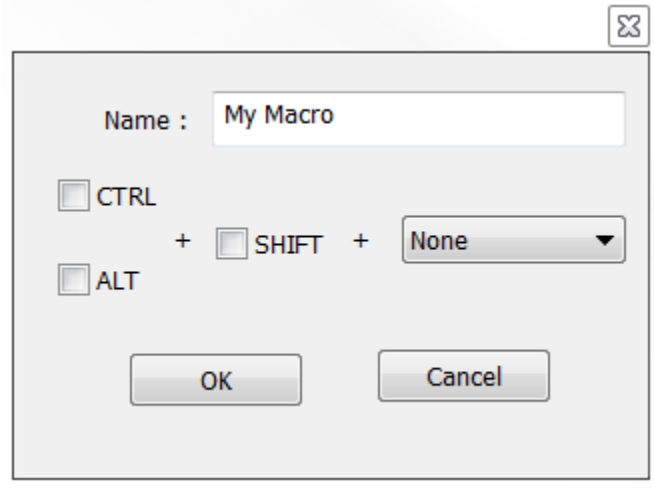
Once you've got to grips with using Notepad++ in your day-to-day SAS programming, you may discover applications which you want to regularly use. In the same way that you might write a SAS macro for a repetitive SAS task, you can record a Notepad++ macro to repeat any set of steps at the push of a button.

RECORDING A MACRO

Rather than using code, Notepad++ macros are created by recording user actions (e.g. keystrokes, mouse clicks, Find & Replace steps). The key steps are:

1. Start the recording using "Macro" > "Start Recording", or by clicking the red dot 'record' icon.
2. Perform the steps that you want to record.
3. Stop the recording using "Macro" > "Stop Recording", or by clicking the black square 'stop' icon.
4. Save the macro using "Macro" > "Save Current Recorded Macro…". From here you can name your macro, and optionally assign a keyboard shortcut to it.

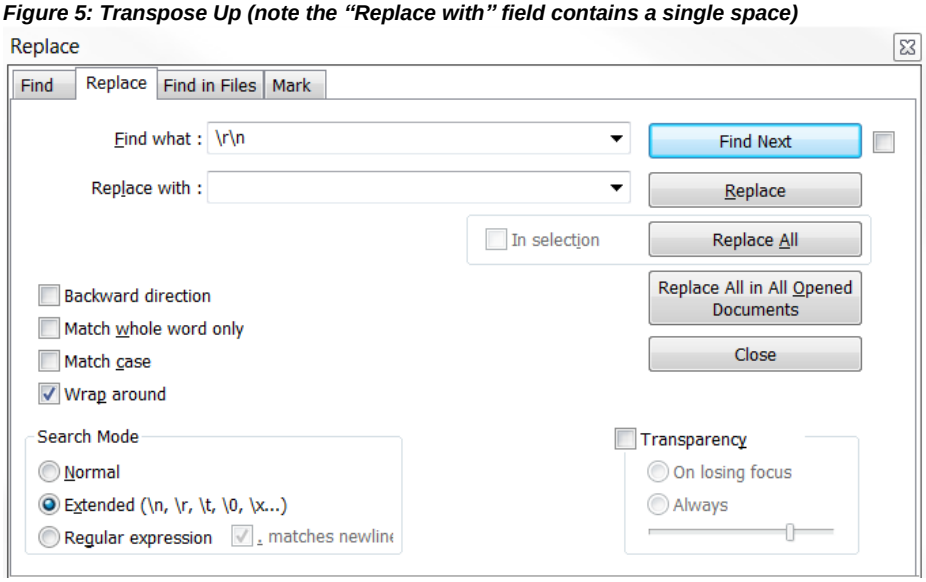
Figure 4: Naming a recorded macro



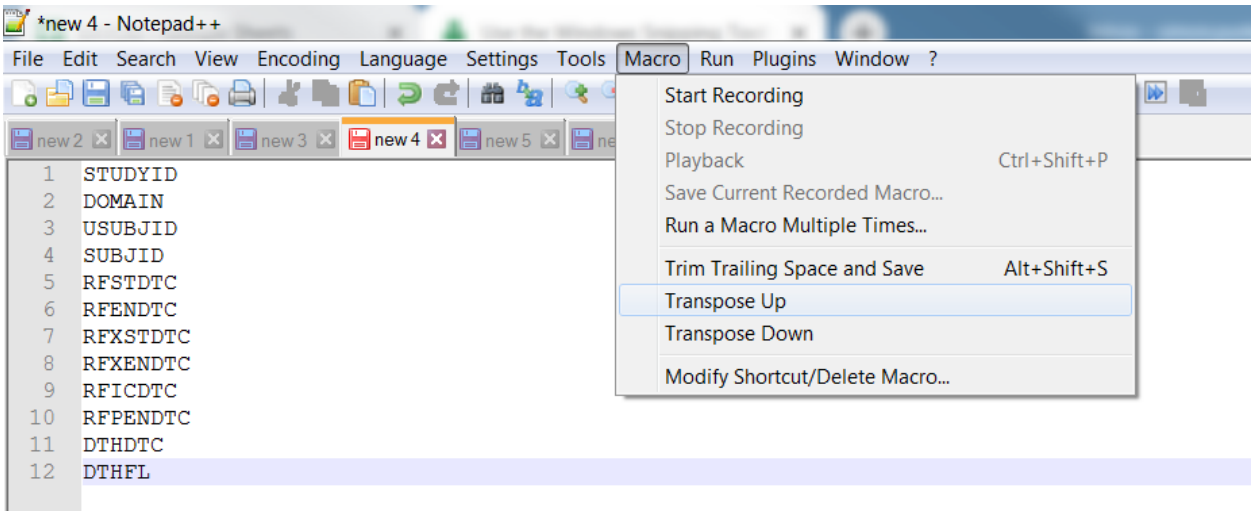
Example – Transpose text up or down

It is often useful to switch a list of variables between vertical format (e.g. copied in from a dataset specification) and horizontal format (e.g. in a keep statement). This can be done using simple Find & Replace steps, which can then be saved as macros:

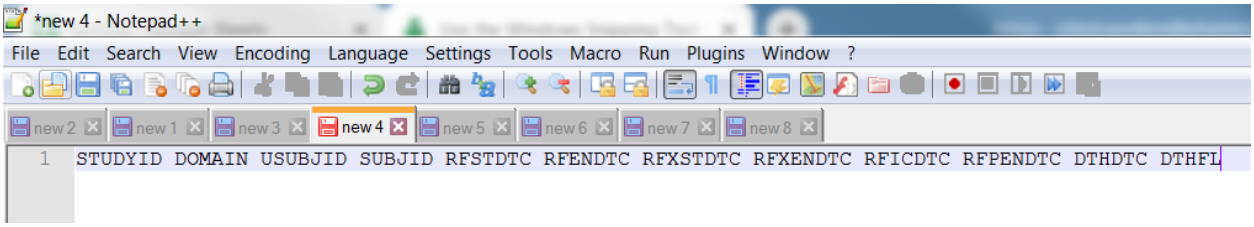
Figure 5: Transpose Up (note the "Replace with" field contains a single space)



Before



After



Example – Convert a dataset specification into a SAS attrib statement

When creating a dataset from a specification, the attributes of each variable (name, label, type and length) need to be set In the SAS dataset; one way to do this is using an attrib statement. Manually typing out the statement can be time-consuming for a large dataset, but you can create a macro which can convert lines from a specification into the required SAS format:

Find what:	Replace with:	Comment
^(\w*)\s*	\1 label='	Insert `label='` after variable name
\s*text\s*	' length=$	Insert `' length=$` for character variables
\s*(integer|float)\s*	' length=	Insert `' length=` for numeric variables

Figure 6: Example of the Attrib macro in action

Before (Text copied from a dataset specification into Notepad++)

```
STUDYID Study Identifier    text    21
DOMAIN  Domain Abbreviation text    8
USUBJID Unique Subject Identifier   text    30
DSSEQ   Sequence Number integer 8
DSTERM  Reported Term for the Disposition Event text    200
DSDECOD Standardized Disposition Term   text    200
DSCAT   Category for Disposition Event  text    40
VISITNUM    Visit Number    integer 8
VISIT   Visit Name  text    40
VISITDY Planned Study Day of Visit  integer 8
EPOCH   Epoch   text    40
DSDY    Study Day of Collection integer 8
```

After

```
STUDYID label='Study Identifier' length=$21
DOMAIN label='Domain Abbreviation' length=$8
USUBJID label='Unique Subject Identifier' length=$30
DSSEQ label='Sequence Number' length=8
DSTERM label='Reported Term for the Disposition Event' length=$200
DSDECOD label='Standardized Disposition Term' length=$200
DSCAT label='Category for Disposition Event' length=$40
VISITNUM label='Visit Number' length=8
VISIT label='Visit Name' length=$40
VISITDY label='Planned Study Day of Visit' length=8
EPOCH label='Epoch' length=$40
DSDY label='Study Day of Collection' length=8
```

Now the code is ready to be pasted into SAS:

```
data sdtm.ds (keep=STUDYID DOMAIN USUBJID DSSEQ DSTERM DSDECOD DSCAT VISITNUM VISIT VISITDY EPOCH DSDY label="Disposition");
  set ds;
  attrib
  STUDYID label='Study Identifier' length=$21
  DOMAIN label='Domain Abbreviation' length=$8
  USUBJID label='Unique Subject Identifier' length=$30
  DSSEQ label='Sequence Number' length=8
  DSTERM label='Reported Term for the Disposition Event' length=$200
  DSDECOD label='Standardized Disposition Term' length=$200
  DSCAT label='Category for Disposition Event' length=$40
  VISITNUM label='Visit Number' length=8
  VISIT label='Visit Name' length=$40
  VISITDY label='Planned Study Day of Visit' length=8
  EPOCH label='Epoch' length=$40
  DSDY label='Study Day of Collection' length=8
  ;
run;
```

CONCLUSION

Notepad++ opens up a whole range of possibilities to help you save time in your SAS programming. I hope that the examples presented above will provide a jumping-off point for you to discover your own use cases.

REFERENCES

- Notepad++ homepage https://notepad-plus-plus.org/
- Guide to applying SAS colour scheme https://blogs.sas.com/content/sasdummy/2017/08/25/npp-with-sas/

CONTACT INFORMATION

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

Simon Purkess
PHASTAR
Unit 2, 2A Bollo Lane
London, W4 5LE
Email: simon.purkess@phastar.com
Web: https://phastar.com/