PharmaSUG 2020 - Paper QT-249

Text Wrangling with Regular Expressions: A Short Practical Introduction

Noory Kim, SDC

ABSTRACT

Regular expressions provide a powerful way to find and replace patterns in text, but their syntax can seem intimidating at first. This paper presents a few simple practical examples of using a text editor to adapt text from SDTM or ADaM data set specifications for insertion into SAS® programs.

This paper is intended for SAS users of any skill level. No prior knowledge of regular expressions is needed.

INTRODUCTION

As SAS programmers, we often need to take text from specifications or shells and put it into our code. From data set specifications we can get variable names and variable labels. From table and listing shells we can get row and column labels.

PROC IMPORT may help in this regard, but in some cases may not work as expected (e.g. sometimes when the Excel file is already opened by another user). Copying and pasting by hand is also an option; however, this may require the tedious and time-consuming process of cleaning and adapting the text for SAS programs.

Using regular expressions (a.k.a. *regex*) can help cut down on the tedium and save time. Essentially, using regular expressions is an enhanced way of finding and replacing text.

Even knowing just a little bit of regex can go a long way. The tasks I present in this paper are only those that I carry out frequently; hence my claim that this introduction is a practical one.

OUTLINE

This paper will touch on how to do the following:

- Add a quotation mark to the beginning of a line
- Add a quotation mark to the end of a line (after trimming blank spaces)
- Add quotation marks around a line
- Select a substring of a delimited line

These tasks can be combined to do the following:

- Prepare lines for a LABEL statement
- Prepare lines for a FORMAT statement for mapping variables (e.g. from –TESTCD to –TEST in SDTM data sets, or from PARAMCD to PARAM in ADaM data sets)

The workflow for the examples in this paper is shown in Figure 1. All tasks below occur in the middle stage of this workflow (shown in green).

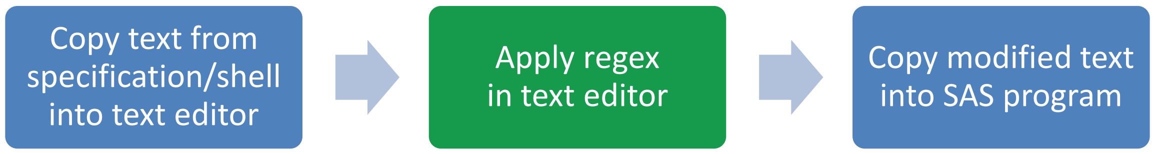

Copy text from specification/shell into text editor
Apply regex in text editor
Copy modified text into SAS program

Figure 1: Workflow from specification/shell to SAS program. All tasks demonstrated in this paper are to be done in the middle stage (shown in green).

SOME GENERAL GUIDELINES AND COMMENTS

- As you go through these examples, don't be afraid to make mistakes as you learn. Thankfully we can undo our mistakes as needed. (In many Windows programs you can use the keyboard shortcut CTRL-Z.)
- To be extra careful, avoid practicing on text in original copies of files. In this paper, all regex will be applied in the text editor. (SAS interactive does allow the use of regex, but I discourage replacing text within the original program.)
- In this paper I use the text editor Notepad++, an open source text editor (available at https://notepad-plus-plus.org). Regex is available in other text editors, but the dialect may differ. Some tweaks may be needed to get the regular expressions used here to work in other contexts.
- To get the dialog box in Notepad++ to find and replace text, you can do either of the following: (1) In the menu bar, select Search, then select Replace, OR (2) Use the keyboard shortcut CTRL-H.
- Check that the option for regular expressions is selected in your text editor. (For example, see the figure in Task 1 below at the bottom of this page.)

Let's get started!

TASK 1: ADD A QUOTATION MARK TO THE BEGINNING OF A LINE

WHAT

Figure 2 illustrates this task: We would like to add quotation marks to the beginning of each variable label copied from an SDTM or ADaM data set specification. Each variable label is on a separate line.

```
Study Identifier
Domain Abbreviation
Unique Subject Identifier
```

```
"Study Identifier
"Domain Abbreviation
"Unique Subject Identifier
```

Figure 2: Adding quotation marks to the beginning of each line

HOW

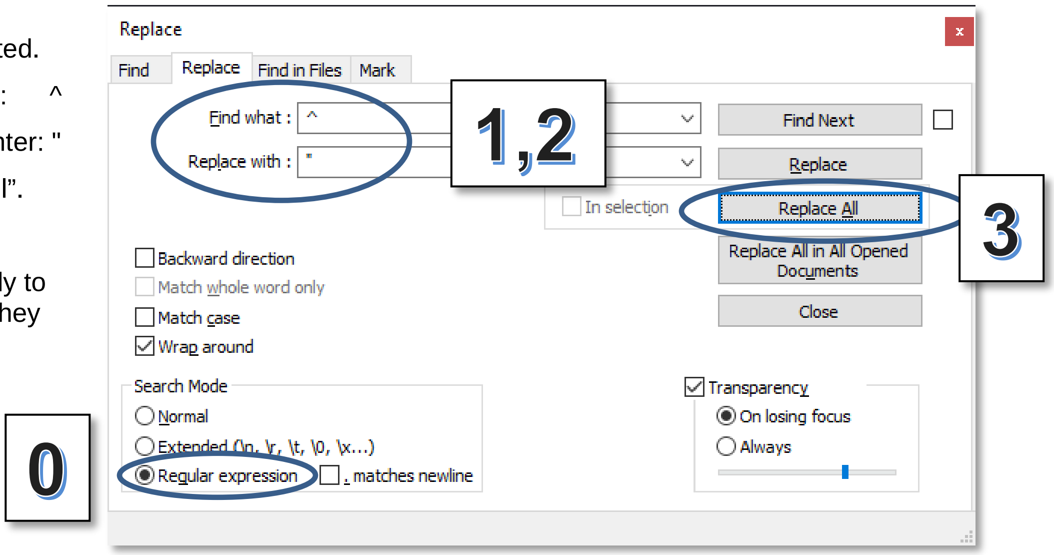
We can employ the following regex syntax:

- ^ - This refers to the beginning of a line.

You can insert a quotation mark at the beginning of all lines as follows:

0. Check that "Regular expression" is selected.
1. In "Find what", enter: ^
2. In "Replace with", enter: "
3. Click on "Replace All".

Note: Steps 0 and 3 will apply to all examples in this paper. They will be implicit in all tasks presented hereafter.



Find what : ^
Replace with : "
[✓ Wrap around]
Search Mode: [✓ Regular expression]
[✓ Transparency] [✓ On losing focus]

TASK 2: ADD A QUOTATION MARK TO THE END OF A LINE

WHAT

Figure 3 illustrates this task: We would now like to take the result from Figure 2 and add quotation marks to the end of each line. The resulting text will be used as part of a LABEL statement.

```
"Study Identifier
"Domain Abbreviation
"Unique Subject Identifier
```

```
"Study Identifier"
"Domain Abbreviation"
"Unique Subject Identifier"
```

Figure 3: Adding quotation marks to the end of each line

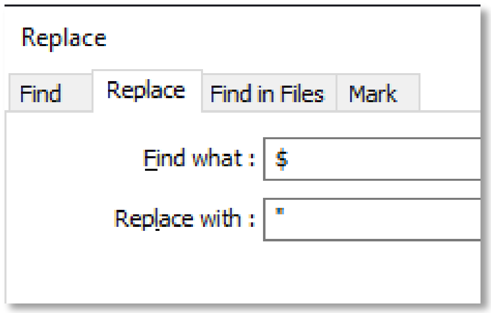
HOW

We can employ the following regex syntax:

- $ - This refers to the end of a line.

You can insert a quotation mark at the end of all lines as follows:

1. Find what: $
2. Replace with: "



Find what : $

Replace with : "

TASK 3: TRIM THE END OF A LINE AND ADD A QUOTATION MARK

Sometimes in carrying out Task 2, you may find that there are blank spaces at the end of the line.

```
"Study Identifier
"Domain Abbreviation
"Unique Subject Identifier
```

```
"Study Identifier         "
"Domain Abbreviation      "
"Unique Subject Identifier"
```

Figure 4: Having unwanted whitespace before the closing quotation marks

WHAT

We would like to remove any blank spaces before the closing quotation marks.

```
"Study Identifier         "
"Domain Abbreviation      "
"Unique Subject Identifier"
```

```
"Study Identifier"
"Domain Abbreviation"
"Unique Subject Identifier"
```

Figure 5: Removing blank spaces before the closing quotation marks

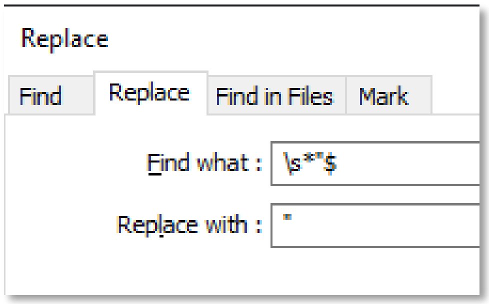
HOW

We can employ the following regex syntax:

- $ - This refers to the end of a line (as seen in Task 2).
- \s - This refers to a blank space.
- \s* - This refers to any (non-negative) number of consecutive blank spaces.

Steps:

1. Find what: \s*"$
2. Replace with: "



Find what : \s*"$
Replace with : "

TASK 4: ENCLOSE EACH LINE IN QUOTATION MARKS

WHAT

As an alternate to combining Tasks 1 and 2, we can enclose each line in quotes in one step.

```
Study Identifier
Domain Abbreviation
Unique Subject Identifier
```

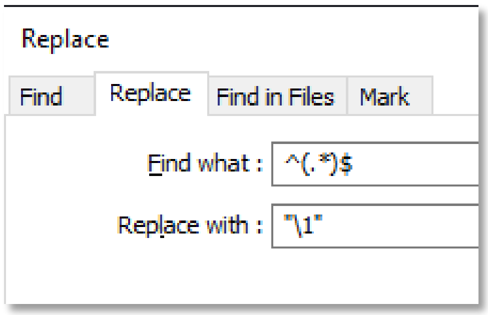
```
"Study Identifier"
"Domain Abbreviation"
"Unique Subject Identifier"
```

Figure 6: Enclosing each line in quotation marks (in one step)

HOW

We can use the following syntax:

- (.*) - This is a wildcard referring to any non-negative number of characters.
  The dot refers to any character, including a blank space.
  The asterisk is a *quantifier* denoting a non-negative integer.
  The parenthesis defines what is termed a *capturing group*.
- \1 - This refers to the first capturing group in a line. It is called a *backreference*.



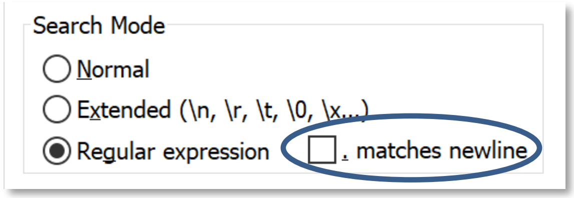
Find what : ^(.*)$
Replace with : "\1"

Steps:

1. Find what: ^(.*)$
2. Replace with: "\1"

Note: The caret and dollar sign are optional in this case, but only if the box "_._ matches newline" remains unchecked. (See screenshot detail on the lower right of this page.) We can then simplify the steps as follows:

1. Find what: (.*)
2. Replace with: "\1"



Search Mode
○ Normal
○ Extended (\n, \r, \t, \0, \x...)
[✓ ● Regular expression]
☐ . matches newline

TASK 5: PREPARE LINES FOR A LABEL STATEMENT

WHAT

Using backreferences, we can generate lines for a LABEL statement. The starting text here is from an Excel spreadsheet containing an SDTM or ADaM data set specification. There is a tab delimiter between the variable names (i.e. “STUDYID”) and the variable labels (i.e. “Study Identifier”).

```
STUDYID      Study Identifier
DOMAIN       Domain Abbreviation
USUBJID      Unique Subject Identifier
```

```
STUDYID = "Study Identifier"
DOMAIN = "Domain Abbreviation"
USUBJID = "Unique Subject Identifier"
```

Figure 7: Generating lines for a LABEL statement, starting from tab-delimited text. Only the text after the tab delimiter is to be enclosed in quotation marks.

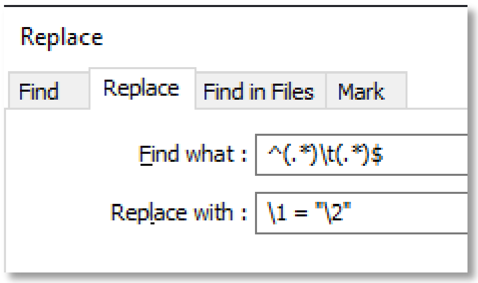
HOW

We can use the following syntax:

- (.*) - A capturing group (as seen in Task 4).
- \t - This refers to a tab delimiter.
- \1 - This backreference refers to the first capturing group in a line.
- \2 - This backreference refers to the second capturing group in a line.

Steps:

1. Find what: ^(.*)\t(.*)$
2. Replace with: \1 = "\2"



Find what : ^(.*)\t(.*)$

Replace with : \1 = "\2"

Notes:

1. As with Task 4, the caret and dollar sign are optional in this case so long as the box “. matches newline” remains unchecked, whereby we can simplify the steps as follows:

   Find what: (.*)\t(.*)

   Replace with: \1 = "\2"

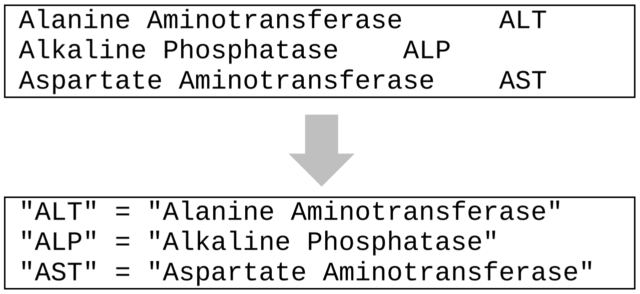
2. Note that because the tab delimiter is expressly stated in the “Find what” field, it is not included in either capturing group. In fact, it acts as the boundary between the two capturing groups.

3. If these steps were applied to text that had more than one tab delimiter per line, then a single \t (within “Find what”) would refer only to the last tab delimiter in each line. (This reflects the ‘greediness’ of the asterisk quantifier in the first capturing group. See Friedl (2006), Chapter 4.)

TASK 6: PREPARE LINES FOR A FORMAT STATEMENT TO MAP VARIABLES

WHAT

We can also use backreferences to generate lines for a FORMAT statement to map variables. The starting text here are values of LBTEST and LBTESTCD, with a tab delimiter in between. In this case we will also switch the order of appearance of these two.



```
Alanine Aminotransferase       ALT
Alkaline Phosphatase     ALP
Aspartate Aminotransferase     AST
```

```
"ALT" = "Alanine Aminotransferase"
"ALP" = "Alkaline Phosphatase"
"AST" = "Aspartate Aminotransferase"
```

Figure 8: Generating lines for a FORMAT statement, starting from tab-delimited text with values of LBTEST and LBTESTCD. The values of each variable are to be enclosed in quotation marks, and their order of appearance is to be switched.

After defining this format, we could then generate values of LBTEST from values of LBTESTCD using a PUT statement:

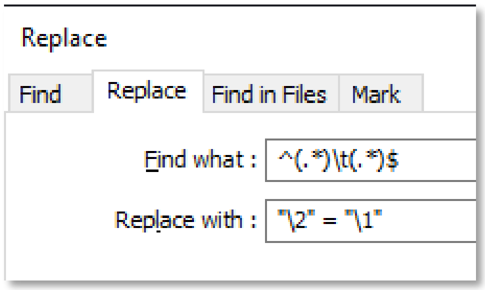
```
LBTEST = PUT(LBTESTCD, $formatname.);
```

HOW

We can use the same syntax as we did for Task 5.

Steps:

1. Find what: ^(.*)\t(.*)$
2. Replace with: "\2" = "\1"



Find what : ^(.*)\t(.*)$

Replace with : "\2" = "\1"

Notes:

1. The notes for Task 5 apply here as well.
2. The backreference \1 refers to the first capturing group and corresponds to values of LBTEST in the starting text.
3. The backreference \2 refers to the second capturing group and corresponds to values of LBTESTCD in the starting text.
4. The backreferences can be cited in any order. This allows us to easily switch the order of appearance of LBTESTCD and LBTEST.

SUMMARY

Regular expressions, a.k.a. *regex*, are useful for modifying text from a specification or shell for insertion into a SAS program, especially in cases where PROC IMPORT is not a reliable option. The expressions can be used with the find and replace function of a text editor.

The examples in this paper illustrate how to generate lines for a LABEL or FORMAT statement for use in an SDTM or ADaM data set program. The same methods can be used to adapt text from table or listing shells for use in their respective SAS programs.

Here is a list of the regex syntax used in this paper:

- ^ - the beginning of a line
- $ - the end of a line
- \s - a blank space
- \s* - any (non-negative) number of consecutive blank spaces
- \t - a tab delimiter
- (.*) - a wildcard referring to any non-negative number of characters

  The dot refers to any character, including a blank space.

  The parenthesis defines what is termed a *capturing group*.
- \1 - This *backreference* refers to the first capturing group in a line.
- \2 - This backreference refers to the second capturing group in a line.

REFERENCES

Friedl, Jeffrey E. F. 2006. Mastering Regular Expressions. 3rd ed. Sebastopol, CA: O'Reilly and Associates, Inc.

Goyvaerts, Jan. "Regular Expression Reference: Capturing Groups and Backreferences." 22 November 2019. Available at www.regular-expressions.info/refcapture.html.

Ho, Don. Notepad++. Available at https://notepad-plus-plus.org.

ACKNOWLEDGMENTS

Thanks to Danielle Rupper, Kevin Uchimara, and Bharath Donthi for reviewing a draft of this paper and providing helpful suggestions and corrections.

CONTACT INFORMATION

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

Noory Kim
Statistics & Data Corporation
nkim@sdcclinical.com
www.sdcclinical.com