

Make Your Match and Capture the Data

Robert Rosofsky

Health Information Systems Consulting

Boston Area SAS Users Group

March 13, 2002

Statement of the Problem

Create a flat analytical SAS file from text data

Goal: File that meets this description

Variable	Type/ Length	Label
StudyID	C, 8	ID Number for child
VACDATE	N, 4	Date of vaccine administration
VAC	C, 3	Code for vaccine type
SITE	C, 2	Code for body site of injection
MFR	C, 3	Code for manufacturer
LOT	C, 15	Lot number for vaccine

Statement of the Problem

Sample Target Data:

StudyID	VACDATE	VAC	SITE	MFR	LOT
12345678	1/25/1992	30	LA	SKB	X9876542
12345678	1/25/1992	45	LA	LED	588005
12345678	4/1/1992	30	RA	SKB	X9876544
12345678	4/1/1992	45	RA	LED	588005
55884466	10/10/1994	49	RL	MSD	D58554458
55884466	12/15/1996	45	RL	MA	5802
87654321	8/23/1993	45	RL	MSD	PR12585
87654321	10/25/1993	30			
87654321	10/25/1993	45			
87654321	10/25/1993	49			
87654321	1/10/1994	30	RL	SKB	X9876545
87654321	1/10/1994	45	RA	LED	588006
87654321	1/10/1994	49	RL	MSD	D58554458
87654321	1/10/1994	03	RL	MA	1456

Statement of the Problem

Sample source data

PIP-987-65-43-Q

BUSH, GEORGE W

CONTINUED...

IMMUNIZATIONS

AGE	DATE	TYPE	COMMENT
8 YRS. 6 MOS.	6/2/95	MUMPS-MEASLES-RUBELLA	COMBINED VACCINE (MMR) #2 MFG/LOT#: MS/0180A ADMIN BY: FLORENCE NIGHTINGALE, RN 1/25/96, 0.5CC SC LT ARM
14 YRS.	11/1/90	DPT-1	DATE BY HX: 11/26/76
		DPT-2	DATE BY HX: 01/25/77
		DPT-3	DATE BY HX: 04/13/77
		POLIO - 4 (TOPV)	DATE BY HX: 01/03/79

Statement of the Problem

Sample source data continued:

8 WKS	6/15/88	DPT - 1	6/17/88 LOT 260 EXP 3/3/89 CONSENT 6/17/88
		POLIO - 1 (TOPV)	
15 MOS.	8/10/89	DPT - 4	MFG/LOT#: MA/266 ADMIN BY: DICK CHENEY, MD
18 YRS. 9 MOS.	8/1/94	HEPATITIS B VACCINE (RECOMBINANT) (RECOMBIVAX-HB)	MFTR: MERCK. LOT#: 0475A. SITE: LEFT ARM. BY B. DYLAN.

Process: Summary

- Locate one of many possible text strings
- “Parse out” text strings
- Convert strings to data

Process: Reorganize Data

This set of text data:

15 MOS.	8/10/89	DPT - 4	MFG/LOT#: MA/266 ADMIN BY: DICK CHENEY, MD
14 YRS.	11/1/90	DPT-1 T-D (ADULT)	DATE BY HX: 11/26/76 EXP 10/23/97 SITE RT ARM

Becomes reorganized with 3 variables as:

VACDATE	VACTEXT	IMMDATA
8/10/89	DPT - 4	MFG/LOT#: MA/266 ADMIN BY: DICK CHENEY, MD
11/1/90	DPT-1	DATE BY HX: 11/26/76
11/1/90	T-D (ADULT)	EXP 10/23/97 SITE RT ARM

Process: Simple Example

Locate the patient ID (PATID) when it occurs

ABC-345-67-89-W

ALOU,FELIPE

CONTINUED...

...CONTINUED ABC-345-67-89-W

7:04 PM 4/18/2001 (EDP) PAGE 2

Process: Simple Example

```
PATID_PAT= RXPARSE(  
    '$A$A$A'-'$D$D$D'-'$D$D'-'$D$D'-'$A');
```

1. 3 letters

2. hyphen

3. 3 digits

4. hyphen

5. 2 digits

6. hyphen

7. 2 digits

8. 1 letter

Will match: **ABC-345-67-89-W**

RXPARSE function

Notes on the RXPARSE function:

- The function returns a *numeric* value, when it is executed
- In cases like this, it is executed only once during a DATA step as follows:

```
RETAIN PAT_ID;  
IF _N_=1 THEN DO;  
    PAT_ID= RXPARSE( - - - );  
END;
```

RXPARSE function

Pattern Wildcards: a few examples

\$a or \$A: any letter (*not* case significant)

\$d or \$D: any digit ('\$0-9')

\$l or \$L: lowercase letter ('\$a-z').

\$u or \$U: any uppercase letter ('\$A-Z')

\$w or \$W: match any white space character (e.g. blank, tab, backspace, CR, etc.)

\$f or \$F: match a floating point number.

\$n or \$N: match a SAS name.

\$q or \$Q: match a string in quotation marks.

RXPARSE function

User-defined patterns: a few examples

String in quotation-marks: match the quoted string exactly

e.g. 'XYZ' matches only the single upper-case "XYZ"

String in quotation marks, preceded by a dollar sign: matches *any* of the single characters in quotes

e.g. '\$3PCO' matches any occurrence of "3" or "P" or "C" or "O"

String in quote marks, preceded by either "^" or "~": matches any of the characters *not* in quotes

e.g. '^3PCO' matches occurrence of anything *other* than a "3" or "P" or "C" or "O"

RXMATCH function

Syntax: RXMATCH(*rxvalue*, *string expr*);

Similar to INDEX() function:

- Identifies the starting position of pattern from first argument in string of second argument
- Returns zero if not found

```
IF RXMATCH(PATID_PAT, (SUBSTR(_INFILE_,1,15)))  
> 0 THEN DO;
```

Process: Multiple Pattern Search

Goal: Locate *date* data of different layouts

Full date strings (MM/DD/YY, M/DD/YY,
MM/D/YY, M/D/YY

```
FDATE_PAT1= RXPARSE ("
    '$'0-1'$D'/'$'0-3'$D'/'$D$D/* MM/DD/YY */
|$'1-9''/'$'1-3'$D'/'$D$D      /* M/DD/YY */
|$'0-1'$D'/'$'1-9''/'$D$D      /* MM/D/YY */
|$'1-9''/'$'1-9''/'$D$D")      /* M/D/YY */;
```

Process: Multiple Pattern Search

Determine if a date pattern is present, using the
RXMATCH function

```
WHEN (RXMATCH(FDATE_PAT1,IMMDATA)> 0) DO;
```

Process: Multiple Pattern Search

If present, use the CALL RXSUBSTR() function to extract the string

```
WHEN (RXMATCH(FDATE_PAT1, IMMDATA) > 0) DO;  
CALL RXSUBSTR(FDATE_PAT1, IMMDATA, POSX, LENX);
```

The CALL RXSUBSTR() *assigns information* about a substring.

CALL RXSUBSTR function

Syntax: CALL RXSUBSTR(*rxvalue*,
stringexpr, *startpos*[, *length*] [, *score*]);

Similar to RXMATCH() function:

- Assigns the *starting position* of *rxpattern* as 3rd argument in *string* of 2nd argument
- Assigns the *length* of the substring to the variable specified in the 4rd argument
- Provides a count of # of patterns found to the variable in the 5th argument

Process: Multiple Pattern Search

```
CALL RXSUBSTR(FDATE_PAT1, IMMDATA, POSX, LENX);
```

With information from CALL RXSUBSTR, we can
“extract” or parse out the string:

```
VACDATCHAR= SUBSTR(IMMDATA, POSX, LENX);
```

And then we can assign the value to a date
variable:

```
VACDATE= INPUT(VACDATCHAR, MMDDYY8.);
```

Process: Multiple Word Search (1)

Look for one of many words to exist: Either they're there or they're not. Example:

```
IF INDEX(source, 'word1' ) > 0 OR  
   INDEX(source, 'word2' ) > 0 OR  
   INDEX(source, 'word3' ) > 0 OR  
   INDEX(source, 'word4' ) > 0 OR  
   INDEX(source, 'word5' ) > 0 etc. THEN  
   do_whatever ;
```

Very tedious to code when you have more than a few words or strings to search for.

Process: Multiple Word Search (1)

Much easier to simply insert a series of strings as part of a pattern

```
pattern= RXPARSE(" 'word1' | 'word2' | 'word3' |  
'word4' | 'word5' ... ");
```

```
NOIMM_PAT= RXPARSE(" 'CONTRAINDICATED'  
| 'DECLINED' | 'DEFERRED' | 'REFUSE' | 'REFUSED'  
| 'NOT GIVEN' | 'NOT INDICATED' | 'OMITTED'  
| 'POSTPONED' | 'NOT DONE' | 'NOT ESSENTIAL' ");
```

Process: Multiple Word Search (1)

And then search for the pattern.

```
IF RXMATCH(NOIMM_PAT1,IMMDATA) > 0 THEN DO;  
    multiple statements...  
END;
```

Benefit is that data is separate from code

- Easier to read
- Fewer places to modify
- Less prone to error

Process: Multiple Word Search (2)

The goal is to locate data that appears *after* any of a set of words: Example:

18 YRS. 9 MOS. 8/1/94 HEPATITIS B VACCINE

MFTR: Merck LOT#
0475A. SITE: LEFT
ARM. BY J. SWIFT

Process: Multiple Word Search (2)

Set this up as a pattern match

```
MFR_PAT= RXPARSE (" 'MFTR' | 'MF' | 'MFG' | 'MF:' |  
    'MFD' | 'MF' | 'MFR' | 'MANF' | 'MFTD' | 'MANUF' |  
    'MANF BY' | 'MANUFACTURER' | 'MFD BY' |  
    'MFTD BY' " );
```

Process: Multiple Word Search (2)

Extract data appearing *after* the manufacturer string. Where is the end of the string?

```
WHEN (RXMATCH(MFR_PAT, IMMDATA) > 0 ) DO;
```

This one statement will locate *any* of a set of words within a string expression/variable as shown earlier

Process: Multiple Word Search (2)

Extract data appearing after "any one of many" string search (Manufacturer)

```
CALL RXSUBSTR(MFR_PAT, IMMDATA, POSX, LENX);
```

Using both `POSX` and `LENX`, we can locate where the manufacturer string ends.

Process: Multiple Word Search (2)

The actual manufacturer is then the next string "piece" found

```
MFRSTR= SCAN ( SUBSTR ( IMMDATA , POSX+LENX ) , 1 , ' : ' ) ;
```

Within the SCAN function, by adding together POSX and LENX, the SUBSTR function tells us where to start scanning

Other Pattern Match Functions

CALL RXCHANGE: Changes one or more substrings that match a pattern.

CALL RXFREE: Frees memory allocated by other regular expression (RX) functions and CALL routines.

Other Pattern Match Functions

Syntax: CALL RXCHANGE(*rx*, *times*,
old-string[, *new-string*]) ;

Rx: is a pattern match expression.

Times: # of times that a change can take place.

Old-string: source string searched/ changed.

New-string is optional for writing change to a new string.

Other Pattern Match Functions: CALL RXCHANGE()

```
data _null_;  
  length dat $80 newdat $80;  
  if _n_=1 then do;  
    FDATE_PAT1= RXPARSE("$'0-1'$D to 'change'");  
  end;  
  
  dat="...light bulb want to 05 23 12 58";  
  do i=0 to 3;  
    call rxchange(fdate_pat1,i,dat,newdat);  
    put i= dat= / newdat= /;  
  end;  
run;
```

Other Pattern Match Functions:

CALL RXCHANGE()

```
i=0 dat=...light bulb want to 05 23 12 58  
newdat=...light bulb want to 05 23 12 58
```

```
i=1 dat=...light bulb want to 05 23 12 58  
newdat=...light bulb want to change 23 12 58
```

```
i=2 dat=...light bulb want to 05 23 12 58  
newdat=...light bulb want to change 23 change 58
```

```
i=3 dat=...light bulb want to 05 23 12 58  
newdat=...light bulb want to change 23 change 58
```

Other Pattern Match Functions:

Syntax: `CALL RXFREE(rx) ;`

Rx: is a numeric value returned by `RXPARSE()` functions

The function is used to free up memory

Consider using a statement like this at the end of a DATA step:

`CALL RXFREE(rx);`

```
End_Pat=RXPARSE(  
    "'The End' | 'Fin' | 'Finished' | 'Done'")
```

Robert Rosofsky
Health Information Systems Consulting
www.HealthInfoSys.net