

CT10 - PHUSE EU Connect 2020

**Don't UnderMine the value of your CRF
with Perl Regular Expression**

John McDade



Introduction

The SDTM annotated CRF (aCRF) is a key regulatory submission document that provides essential traceability between the database and SDTM datasets.

Traditionally this has been a predominantly manual task but we will explore use of perl regular expression to expedite production of the aCRF.

Annotated Case Report Form (aCRF)

DM = Demographics

D933GC00001_Version 2.00_28JAN2019: Unique

Form: Demographics (DM)

Generated On: 2019 Jan 29 20:18

(Version:AZ14:09 Date:2018-02-08)

Birth date	BRTDTC	Fixed Unit: yyyy mmm dd
Age	AGE	
Age unit	AGEU	Years ☒
Sex	SEX	Male ☐ Female ☐
Ethnicity	ETHNIC	Hispanic or Latino ☐ Not Hispanic or Latino ☐
Race	RACE	Black or African American ☐ Native Hawaiian or other Pacific Islander ☐ American Indian or Alaska Native ☐ Asian ☐ White ☐ Other ☐
Specify Other	SUPPDM.QVAL when QNAM="RACEOTH"	

SDTM annotated CRF

The annotated CRF is required as part of the SDTM Electronic Case Report Tabulation (eCRT) package. It is provided as a file called acrf.pdf and is hyperlinked from the define.xml.

The SDTM eCRT package is a collection of files and documents that are sent to regulators to allow them to understand the study from data collection to SDTM mapping. The package consists of the aCRF, SDTM XPT datasets, define.xml and Clinical Study Data Reviewer's Guide (cSDRG).

The contents must comply with the FDA Study Data Technical Conformance Guide. <https://www.fda.gov/industry/fda-resources-data-standards/study-data-standards-resources>

Database Annotated Case Report Form

Field Name	Data Type	Field Label	SAS Format	SAS Label	Values
② BIRTHDAT	yyyy MMM-dd-	Birth date	\$10	Birth Date	
③ AGE	3	Age	8	Age	
④ AGEU	\$6	Age unit	\$6	Age Unit	C29848 = Years
⑤ SEX	\$6	Sex	\$6	Sex	C20197 = Male C16576 = Female
⑥ ETHNIC	\$6	Ethnicity	\$6	Ethnicity	C17459 = Hispanic or Latino C41222 = Not Hispanic or Latino
⑦ RACE	\$6	Race	\$6	Race	C16352 = Black or African American C41219 = Native Hawaiian or other Pacific Islander C41259 = American Indian or Alaska Native C41260 = Asian C41261 = White 99 = Other
⑧ RACEOTH	\$200	Specify Other	\$200	Specify Other	

Extracting database CRF metadata

Form: Demographics (DM)

Generated On: 2010 Jun 20 12:12

Field Name	Data Type	Field Label	SAS Format	SAS Label	Values
BIRTHDAT	yyyy MM-dd-	Birth date	\$10	Birth Date	
AGE	3	Age	8	Age	
AGEU	\$6	Age unit	\$6	Age Unit	C29848 = Years
SEX	\$6	Sex	\$6	Sex	C20197 = Male C16576 = Female
ETHNIC	\$6	Ethnicity	\$6	Ethnicity	C17459 = Hispanic or Latino C41222 = Not Hispanic or Latino
RACE	\$6	Race	\$6	Race	C16352 = Black or African

memname	formname	name	code_value	code_decode
DM	Demographics	AGEU	C29848	Years
DM	Demographics	ETHNIC	C17459	Hispanic or Latino
DM	Demographics	ETHNIC	C41222	Not Hispanic or Latino
DM	Demographics	RACE	99	Other
DM	Demographics	RACE	C16352	Black or African American
DM	Demographics	RACE	C41219	Native Hawaiian or other Pacific Islander
DM	Demographics	RACE	C41259	American Indian or Alaska Native
DM	Demographics	RACE	C41260	Asian
DM	Demographics	RACE	C41261	White
DM	Demographics	SEX	C16576	Female
DM	Demographics	SEX	C20197	Male

Perl Regular Expression

- Available in SAS since version 6.12
- Not SAS specific
- Can be complex to read and write
- PRX functions provide powerful pattern matching more concisely

Perl Regular Expression Syntax

- `regex-id = prxparse(perl-regex)`
 - Compile Perl regular expression `perl-regex` and return `regex-id` to be used by other PRX functions.
- `pos = prxmatch(regex-id | perl-regex, source)`
 - Search in `source` and return position of match or zero if no match is found.

```
data _null_;  
  regex = prxparse("/ache/");  
  text = "headache";  
  position = prxmatch(regex, text);  
  if position > 0 then put text= position=;  
run;  
  
text=headache position=5
```

Perl Regular Expression Syntax

Wildcards/Character Class Shorthands

Character	Behavior
.	Match any character
\w	Match a word character (alphanumeric plus " _")
\W	Match a non-word character
\s	Match a whitespace character
\S	Match a non-whitespace character
\d	Match a digit character
\D	Match a non-digit character

Repetition Factors

(greedy, match as many times as possible)

Character	Behavior
*	Match 0 or more times
+	Match 1 or more times
?	Match 1 or 0 times
{n}	Match exactly n times
{n,}	Match at least n times
{n,m}	Match at least n but not more than m times

Character Classes

Character	Behavior
[...]	Match a character in the brackets
[^...]	Match a character not in the brackets
[a-z]	Match a character in the range a to z

Position Matching

Character	Behavior
^	Match beginning of line
\$	Match end of line
\b	Match word boundary
\B	Match non-word boundary

Character	Behavior
<i>non-meta character</i>	Match character
{ } [] () ^ \$. * + ? \	Metacharacters, to match these characters, override (escape) with \
\	Override (escape) next metacharacter
\n	Match capture buffer n
(?:...)	Non-capturing group

https://support.sas.com/rnd/base/datastep/perl_regexp/regexp-tip-sheet.pdf



Regular Expression Examples

Simple text

Cat	
Matches	Doesn't Match
Cat	cat
Cats	concatenate
My Cat is furry	Cart

OR – single characters

c[oa]t	
Matches	Doesn't Match
cat	coat
cot	act
cats	

OR – multiple characters

(one two three)	
Matches	Doesn't Match
three	four
two doses	on
bone	net

Importing database CRF metadata

A tablet is shown at an angle, displaying a dashboard. The dashboard includes a pie chart labeled 'Traffic Sources' and a line graph labeled 'Visitors Overview'. The line graph shows data points over time, with labels for 'Apr 15', 'Apr 25', and 'May 3'. Below the graph, the text 'Visitors 2,958' is visible. The entire image has a reddish-purple tint.

As a CRO has multiple clients, we face different database types and varying CRF design. Every database vendor though should be able to provide a database annotated CRF and a blank CRF.

Ideally we can extract useful information from the database annotated CRF including raw variable names and page numbers.

The process – import CRF

First step is to save the database annotated CRF as a text file and import into SAS.

```
*--- read in annotated CRF saved as txt file;  
data book_anno;  
  infile "&specpath.\aCRF\&anno_crf..txt" delimiter='09'x MISOVER DSD lrecl=13106 ;  
  informat VAR1 $500. ;  
  format VAR1 $500. ;  
  input VAR1 $ ;  
run;
```

	Field Name	Data Type	Field Label	SAS Format	SAS Label	Values
2	BRTHDAT	yyyy MMM- dd-	Birth date	\$10	Birth Date	
3	AGE	3	Age	8	Age	
4	AGEU	\$6	Age unit	\$6	Age Unit	C29848 = Years
5	SEX	\$6	Sex	\$6	Sex	C20197 = Male C16576 = Female
6	ETHNIC	\$6	Ethnicity	\$6	Ethnicity	C17459 = Hispanic or Latino C41222 = Not Hispanic or Latino
7	RACE	\$6	Race	\$6	Race	C16352 = Black or African

2	BRTHDAT	yyyy MMM- dd-	Birth date	\$10	Birth Date	
3	AGE	3	Age	8	Age	
4	AGEU	\$6	Age unit	\$6	Age Unit	C29848 = Years
5	SEX	\$6	Sex	\$6	Sex	C20197 = Male C16576 = Female
6	ETHNIC	\$6	Ethnicity	\$6	Ethnicity	C17459 = Hispanic or Latino C41222 = Not Hispanic or Latino

The process – build expression

- Decide unique features of variable names to build regular expression
 - Upper case letters
 - Prefixed by a number of up to 3 digits
 - Prefixed number at beginning of string
 - 1 to 4 spaces between number and variable name
 - Variable name can include underscore but not at start or end
 - Variable name can include digit but not at start
- The more criteria you can apply the more accurate and cleaner the data extraction will be

The process – build expression

- Decide unique features of variable names to build regular expression
 - Upper case letters **[A-Z]**
 - Prefixed by a number of up to 3 digits **d{1,3}**
 - Prefixed number at beginning of string **^**
 - 1 to 4 spaces between number and variable name **\s{1,4}**
 - Variable name can include underscore but not at start or end **__***
 - Variable name can include 1 digit in middle of name **\d***
 - Variable name can include up to 2 digits at the end of name **\d{0,2}**
- `regex = prxparse("/^d{1,3}\s{1,4}[A-Z]+\d*__*[A-Z]{1,7}\d{0,2}/");`

The process – CRF extraction

Applying the expression and a simple page counter we can extract the variable names and respective page. Be careful of using the values from Page X of Y on the PDF, these are not always the same as the actual PDF pages which is what we need for annotating and define.xml hyperlinking.

```
data crf_vars;
  set book_anno;
  *--- compile regular expression;
  regex = prxparse("/^\d{1,3}\s{1,4}[A-Z]+\d*_*[A-Z]{1,7}\d{0,2}/");
  *--- find variables;
  varmatch = prxmatch(regex, var1);
  *--- simple scan to extract var names;
  if varmatch>0 then name=scan(var1,2);

  *--- increment page numbering;
  retain page .;
  if prxmatch("/^Form: /",var1) then page+1;
run;
```

memname	fomname	ord	name	page
DM	Demographics	1	BIRTHDAT	10
DM	Demographics	2	AGE	10
DM	Demographics	3	AGEU	10
DM	Demographics	4	SEX	10
DM	Demographics	5	ETHNIC	10
DM	Demographics	6	RACE	10
DM	Demographics	7	RACEOTH	10
DM	Demographics	8	Z_SITE	10

The process – merge mapping file and positions

Most pharmaceutical companies will have a single set of database to SDTM mapping standards. By reusing the same standards we can streamline a number of processes. For this type of CRF we can also extract hyperlink positions and merge by page which will allow us to write SDTM annotations to the required position.

Form: Demographics
Generated On: 15 Nov 2013 00:54:29

Birth Date (dd-mon-yyyy) _____

Age _____

Diagram illustrating the process of extracting hyperlink positions (x1, y1 and x2, y2) from a form field (Birth Date) for SDTM mapping.

RAW					SDTM	
memname	fomname	ord	name	page	annot	position
DM	Demographics	1	BRTHDAT	10	BRTHDTC	522 537 644 629
DM	Demographics	2	AGE	10	AGE	522 537 611 596
DM	Demographics	3	AGEU	10	AGEU	522 537 596 581
DM	Demographics	4	SEX	10	SEX	522 537 577 562
DM	Demographics	5	ETHNIC	10	ETHNIC	522 537 542 527
DM	Demographics	6	RACE	10	RACE	522 537 507 492
DM	Demographics	7	RACEOTH	10	SUPPDM.QVAL when QNAM="RACEOTH"	522 537 392 377



The process – write annotations

With all the information we can write an FDF (Forms Data Format) file which contains all the attributes for the annotations.

```
45 0 obj
<</Contents (RACE)/Page 9/Rect [302 492 337 505]/C [0.75 1 1]/BS 21 0 R/DS(font: Arial 10.0pt; text-align:left; color:#000000 )/S
endobj

46 0 obj
<</Contents (ETHNIC)/Page 9/Rect [302 527 346 540]/C [0.75 1 1]/BS 21 0 R/DS(font: Arial 10.0pt; text-align:left; color:#000000 )
endobj
```

DM = Demographics		
D933GC00001_Version 2.00_28JAN2019: Unique		
Form: Demographics (DM)		
Generated On: 2019 Jan 29 20:18		
(Version:AZ14:09 Date:2018-02-08)		
Birth date	BIRTHDC	Fixed Unit: yyyy mmm dd
Age	AGE	
Age unit	AGEU	Years ☒
Sex	SEX	Male ☐ Female ☐
Ethnicity	ETHNIC	Hispanic or Latino ☐ Not Hispanic or Latino ☐
Race	RACE	Black or African American ☐ Native Hawaiian or other Pacific Islander ☐ American Indian or Alaska Native ☐ Asian ☐ White ☐ Other ☐
Specify Other	SUPPDM.QVAL when QNAM="RACEOTH"	

The process – lab pages

The CRF lab pages contain a code which relates to a lab dictionary and specific LBTESTCD / LBSPEC combinations. The matching pattern is more finite here so the regular expression can be more straightforward:

- String preceded by space and followed by underscore
- 5 character string with 5 digits OR
- 4 digits with 1 upper case letter in the middle

```
data lab_vars;  
  set book_anno (where=(substr(memname,1,2)='LB'));  
  *--- compile regular expression;  
  regex = prxparse("/ \d{2}[A-Z]{1}\d{2}_| \d{5}_/");  
  *--- find lab codes;  
  varmatch = prxmatch(regex, var1);  
  *--- simple scan to extract lab codes;  
  if varmatch>0 then name = scan(var1,2,' _');  
run;
```

	Field Name	Data Type	Field Label	SAS Format	SAS Label
20	05C28_BLOOD_OCCULT_QUANT_URI_NE	BLOO9.3	Blood, Occult, Quant (mg/dL) (Urine)		U-Glucose, Strip, Semiquant (mmol/L) (Urine)
var1:					
5	LBTIM	HH:nn	What was the lab specimen collection time?	\$8	Collection Time
6	04004_HAEMOGLOBIN_BLOOD	HAEM9.3	Haemoglobin (Blood)		Haemoglobin (Blood)
7	04005_LEUCOCYTES_WBC_BLOOD	LEUC9.3	Leucocytes, WBC (Blood)		Leucocytes, WBC (Blood)
8	04051_LYMPHOCYTES_ABSOLUTE_COUNT_BLOOD	LYMP9.3	Lymphocytes, Absolute Count (Blood)		Lymphocytes, Absolute Count (Blood)
9	04006_PLATELETS_BLOOD	PLAT9.3	Platelets (Blood)		Platelets (Blood)

The process – lab pages

This allows us to write specific value level annotations for these pages. Note labs are collected 'vertically' here instead of in distinct variables which means we cannot use a mapping file to link raw and SDTM variables.

memname	name	ord	lbspec	lbtestcd	lbtest
LB1	LBCAT	1			
LB1	LBPERF	2			
LB1	LBDAT	3			
LB1	LBTIM	4			
LB1	04004	5	BLOOD	HGB	Hemoglobin
LB1	04005	6	BLOOD	WBC	Leukocytes
LB1	04051	7	BLOOD	LYM	Lymphocytes
LB1	04006	8	BLOOD	PLAT	Platelets
LB1	04009	9	BLOOD	APTT	Activated Partial Thromboplastin Time
LB1	04032	10	BLOOD	NEUT	Neutrophils
LB1	04049	11	BLOOD	NEUTLE	Neutrophils/Leukocytes
LB1	04035	12	BLOOD	LYMLE	Lymphocytes/Leukocytes
LB1	04404	13	BLOOD	INR	Prothrombin Intl. Normalized Ratio

What was the lab specimen collection date?

Fixed Unit: yyyy mmm dd

LBDTC

What was the lab specimen collection time?

Fixed Unit: 24-hour clock

LBDTC

Haemoglobin (Blood)	LBORRES when LBTESTCD="HGB" and LBSPEC="BLOOD"
Leucocytes, WBC (Blood)	LBORRES when LBTESTCD="WBC" and LBSPEC="BLOOD"
Lymphocytes, Absolute Count (Blood)	LBORRES when LBTESTCD="LYM" and LBSPEC="BLOOD"
Platelets (Blood)	LBORRES when LBTESTCD="PLAT" and LBSPEC="BLOOD"
Activated Partial Thromboplastin Time (Blood)	LBORRES when LBTESTCD="APTT" and LBSPEC="BLOOD"
Neutrophils, Absolute Count (Blood)	LBORRES when LBTESTCD="NEUT" and LBSPEC="BLOOD"
Neutrophils, %/Ratio (Blood)	LBORRES when LBTESTCD="NEUTLE" and LBSPEC="BLOOD"
Lymphocytes, %/Ratio (Blood)	LBORRES when LBTESTCD="LYMLE" and LBSPEC="BLOOD"
Prothrombin Complex INR (Blood)	LBORRES when LBTESTCD="INR" and LBSPEC="BLOOD"

The process – other CRF types

We can apply the same approach to other CRF types, as below we can quickly see raw variables have a set format (RAWVAR / TYPE) which will allow is to easily extract. We might not be able to position exactly on the page this time but at least placing annotations on the page with autogenerated domain headers and having page numbers for define.xml will save much time.

```
data crf_vars;  
  set book_anno (where=(substr(memname,1,2)='LB'));  
  *--- compile regular expression;  
  regex = prxparse("/\[A-Z]{3,8} \\/ (NUM|CHAR)/");  
  *--- find variables;  
  varmatch = prxmatch(regex, var1);  
  *--- simple scan to extract vars;  
  if varmatch>0 then name = scan(var1,2,' _');  
run;
```

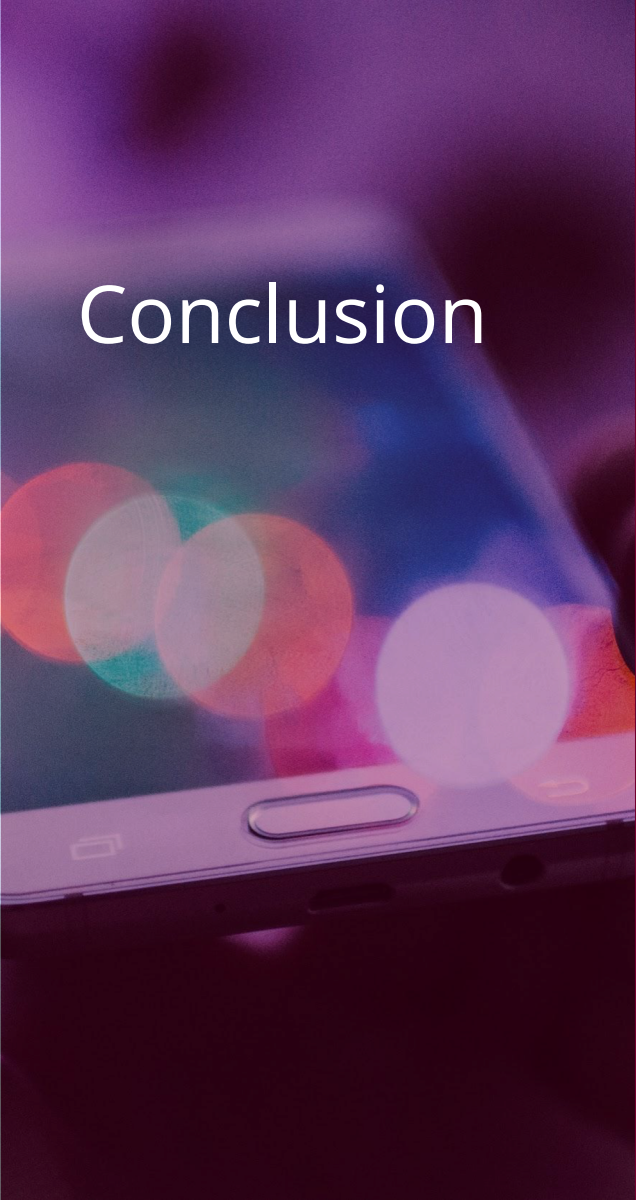
```
*--- compile regular expression;  
regex = prxparse("#\[A-Z]{3,8} / (NUM|CHAR)#");
```

2. Gender * (SEX / NUM) ☐ Male (=1) ☐ Female (=2) message :
====> Date of Visit s.
to Date of Informed

3. Date of Birth * (DOB / NUM – MM/DD/YY10.) Date of Bi
message :
====> Date
future, ple

Date: / /

ReadOnly field – Calculated Age (Date of Visit - Date of Birth) (AGE / NUM)



Conclusion

We've had an overview of Perl Regular Expressions and their application in extracting information from a number of CRF types.

We've also seen how this can be utilised to streamline some tasks required for producing an SDTM annotated CRF and providing page numbers for the define.xml

Another benefit of using this information during study set-up is we can identify any gaps in our mapping before programming begins. A report is generated for any raw values which don't have a match with the database to SDTM mapping standards or for the lab pages, a match with the lab dictionary.

Any Questions?

John McDade

PHASTAR

Unit 2, 2a Bollo Lane

Chiswick, London, W4 5LE

Email: john.mcdade@PHASTAR.com

Web: www.PHASTAR.com

