Define Your Own SAS Functions for Date and Numeric Validation Using PROC FCMP

Jason A Smith, Argo Analytics Ltd, Cambridge, UK

ABSTRACT
Every clinical trials programmer is at some point going to encounter invalid numeric or date values collected in a character field. There are many different methods of checking for valid numbers or dates in a character variable; from writing a validation macro to just using an input statement and hoping for the best. Wouldn’t it be useful if there were an easy to use SAS® function that would check if a character variable holds a valid numeric value, or a valid date?

With the introduction of the SAS Function Compiler (FCMP) procedure in SAS version 9.2 we can now define these SAS functions that we always wanted to have. This paper will present some simple validation functions and some more complex date functions as examples of what can be achieved with PROC FCMP.

INTRODUCTION
SAS 9.2 introduced the FCMP procedure, which allows the user to define their own SAS functions. These functions can be stored in a work area, in the SASUSER area or in a shared library that the whole team can access.

Example usage of the functions in this paper will be:

```sas
if numval(lbscresc) then lbscresn = input(lbscresc, best.);
if dateval(visitdtc, 'yymmd10') then visitdt = input(visitdtc, yymmd10.);
if visitdtc ne '' then visitdti = dateimp(visitdtc);
```

The location of the function must be specified before the function can be accessed. This can be a dataset in the WORK library, the SASUSER library or in any assigned library. Any number of datasets may be specified:

```sas
options cmplib = work.funcs;
```

NUMVAL FUNCTION
This simple function will check if a character variable holds a valid numeric value. This function will return a Boolean 0 (value is not numeric) or 1 (value is numeric) value, depending on the data that is input. (It will not allow the scientific E notation, but will otherwise permit anything that SAS would accept in an input statement.) The function definition is:

```sas
proc fcmp outlib=work.funcs.myfuncs;
function numval(var $);
return (not (notdigit(strip(translate(var,'000','+.+''))) or count(var,'.')>1 or countc(var,'+-')>1 or indexc(left(var), '+-')>1 or left(var) in ('+','+.','-','-.')));
endsub;
run;
```

After submitting this in SAS, the log confirms that the function has been saved:

NOTE: Function numval saved to work.funcs.myfuncs.
NOTE: PROCEDURE FCMP used (Total process time):
          real time         0.01 seconds
          cpu time         0.02 seconds

NUMVAL FUNCTION EXAMPLE USAGE
data one;
    input lbscresc $
    if numval(lbscresc) then lbscresn = input(lbscresc, best.);
    else put "USER WARNING: invalid value " lbscresc;

datalines;
1.1
<5
-3
The log outputs the specified User Warning for any invalid value. Note that there are no longer any untidy "Invalid argument to function INPUT" messages:

```
USER WARNING: invalid value lbscresc=<5
NOTE: The data set WORK.ONE has 3 observations and 2 variables.
NOTE: DATA statement used (Total process time):
    real time     0.02 seconds
    cpu time      0.03 seconds
```

The output dataset shows that the valid numeric values have been input as expected:

```
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>&lt;5</td>
</tr>
<tr>
<td>3</td>
<td>-3</td>
</tr>
</tbody>
</table>
```

**DATEVAL FUNCTION**

This is a more complex example and will validate a date in any of a specified DATE, DDMMYY, MMDDYY or YYMMDD format. Again, the function returns a Boolean 0 (value is invalid) or 1 (value is a valid date) result. The function has a second argument to specify the date format required.

This function mimics the rules that are used by SAS when using an input statement with the corresponding format, with the following exceptions built in:
- the year must be 4 characters (i.e. must include the century);
- except for the date, no other characters are permitted in the string (e.g. "13-10-2014 to 15-10-2014" would be considered as an invalid date, although SAS would permit it to be input as ‘13 Oct 2014’ using the DDMMYYx informat).

In the following example, a character date is first checked in the YYMMDD10 format and then in the DATE11 format. If neither format is valid then a User Warning is output:

```
data two;
  length visitdtc $20;
  format visitdt date9.;
  input visitdtc $;
    if dateval (visitdtc, 'yymmdd10') then visitdt = input (visitdtc, yymmdd10.);
    else if dateval (visitdtc, 'date11') then visitdt = input (visitdtc, date11.);
    else if visitdtc ne '' then put "USER WAR" "NING: invalid date value " visitdtc=;
datalines;
13-OCT-2014
2014/13/14
2014/10/15
; run;
```

The log outputs the specified User Warning for the invalid date, and again there are no longer any untidy "Invalid argument to function INPUT" messages:

```
USER WARNING: invalid date value visitdtc=2014/13/14
NOTE: The data set WORK.TWO has 3 observations and 2 variables.
NOTE: DATA statement used (Total process time):
    real time     0.06 seconds
    cpu time      0.07 seconds
```

The output dataset shows that the date values have been input as expected:

```
<table>
<thead>
<tr>
<th>visitdtc</th>
<th>visitdt</th>
</tr>
</thead>
<tbody>
<tr>
<td>13-OCT-2014</td>
<td>13OCT2014</td>
</tr>
<tr>
<td>2014/13/14</td>
<td></td>
</tr>
<tr>
<td>2014/10/15</td>
<td>15OCT2014</td>
</tr>
</tbody>
</table>
```
Please see the Appendix for the complete DATEVAL Function Definition.

NESTING FUNCTIONS
Once a function has been saved it can be used in further function definitions. In this example, the DATEIMP function uses the DATEVAL function to check if a partial date is valid (YYYY-MM-DD) and then imputes the date to the start of the month for a missing day and to the start of the year for a missing month/day:

```sas
proc fcmp outlib=work.funcs.myfuncs;
function dateimp (dvar $);
  if dateval (dvar, 'yymmdd10') then
    impdt = input (dvar, yymmdd10.);
  else if dateval (cats (dvar, '-01'), 'yymmdd10') then
    impdt = input (cats (dvar, '-01'), yymmdd10.);
  else if dateval (cats (dvar, '-01-01'), 'yymmdd10') then
    impdt = input (cats (dvar, '-01-01'), yymmdd10.);
  else put "%str(USER WARNING: invalid date" dvar; return (impdt);
endsub;
run;
```

DATEIMP FUNCTION EXAMPLE USAGE
This example will set VISITDT if the date is valid and complete, and will set VISITDTI using the DATEIMP function for any valid complete or partial date:

```sas
data three;
  length visitdtc $20;
  format visitdt visitdti date11.;
  input visitdtc $;
  if dateval (visitdtc, 'yymmdd10') then visitdt = input (visitdtc, yymmdd10.);
  visitdti = dateimp (visitdtc);
datalines;
  2014-10-11
  2014-10
  2014-101
  2014
; run;
```

The log outputs the specified User Warning for the invalid date and allows the partial dates to be imputed. Again, there are no longer any untidy “Invalid argument to function INPUT” messages:

```
USER WARNING: invalid date 2014-101
```

**NOTE:** The data set WORK.THREE has 4 observations and 3 variables.

**NOTE:** DATA statement used (Total process time):
  real time 0.07 seconds
  cpu time 0.08 seconds

The output dataset shows that the date values have been input as expected:

<table>
<thead>
<tr>
<th>visitdtc</th>
<th>visitdt</th>
<th>visitdti</th>
</tr>
</thead>
<tbody>
<tr>
<td>2014-10</td>
<td>.</td>
<td>01-OCT-2014</td>
</tr>
<tr>
<td>2014</td>
<td>.</td>
<td>01-JAN-2014</td>
</tr>
</tbody>
</table>

CONCLUSION
The SAS Function Compiler is a powerful and flexible tool that allows us to define and save our own functions. This paper has shown some examples of how using this functionality can simplify programming and data checking.

I hope that this paper will encourage SAS users to try defining their own functions. Next time that you find yourself wishing for a function that does not exist, think, "I can define this function for myself!".


APPENDIX: DATEVAL Function Definition code

* DEATEVAL Function Definition

* Parameters: DVAR - Input Date Variable in character format (required)
  * DFMT - Date Format - must be DATEx, DDMMYYx, MMDDYYx or YYMMDDx (required)

proc fcmp outlib=work.funcs.myfuncs;
function dateval (dvar $, dfmt $);
  n = 0;
  fmtc = upcase (translate (dfmt, ' ', '.'));
  alpha = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
  num = '0123456789';
  sep = ' !@#$%^&*()+-/,:;?\{\}~';
  leapyr = '04,08,12,16,20,24,28,32,36,40,44,48,52,56,60,64,68,72,76,80,84,88,92,96';

  * validate input parameters

  if fmtc in :('DDMMYY','MMDDYY','YYMMDD') then do;
    fmt = substr(fmtc,1,6);
    lenc = substr(fmtc,7);
  end;
  else if fmtc =: 'DATE' then do;
    fmt = substr(fmtc,1,4);
    lenc = substr(fmtc,5);
  end;
  if notdigit(trim(lenc))=0 then len = input(lenc,best.);
  else if lenc = '' then len = 32;
  if fmt='' or len=. then do;
    put "%str(ERR)OR:[DATEVAL] date format must be DATEx, DDMMYYx, MMDDYYx or YYMMDDx";
    goto uexit;
  end;
  else if fmt='DATE' & (len < 9 or len > 32) then do;
    put "%str(ERR)OR:[DATEVAL] format length must be 9-32";
    goto uexit;
  end;
  else if len < 8 or len > 32 then do;
    put "%str(ERR)OR:[DATEVAL] format length must be 8-32";
    goto uexit;
  end;
  else if length(dvar) > len then goto uexit;

  * parse character date into YY, MM and DD macro variables

  if fmt='DATE' then do;
    dd = put(scan(upcase(dvar),1,sep||alpha),$2.);
    mm = put(scan(upcase(dvar),1,sep||num),  $3.);
    yy = put(scan(upcase(dvar),2,sep||alpha),$4.);
    if compress(upcase(dvar),sep) ne cats(dd,mm,yy) then goto uexit;
  end;
  else if fmt='DDMMYY' then do;
    dd = strip(put(left(put(scan(right(put(strip(dvar),$11.)),1,num)||scan(dvar,1,sep), $5.)),$2.));
    mm = strip(substr(put(scan(dvar,1,sep)||scan(dvar,2,sep),$4.),length(dd)+1,2));
    yy = substr(right(put(strip(dvar),$10.)),7,4);
    ss = put(scan(strip(dvar),1,num),$1.);
dd = strip(substr(put(scan(dvar,1,sep)||scan(dvar,2,sep),$4.),length(mm)+1,2));

yy = substr(right(put(strip(dvar),$10.)),7,4);

ss = put(scan(strip(dvar),1,num),$1.);

if strip(dvar) ne cat(strip(mm),ss,strip(dd),ss,strip(yy)) and strip(dvar) ne cats(mm,dd,yy) then goto uexit;

end;
else if fmt='YYMMD' then do;

yy = put(strip(dvar),$4.);
mm = strip(substr(put(scan(strip(dvar),1,sep)||scan(strip(dvar),2,sep),$6.),5));

dd = strip(substr(right(put(strip(dvar),$10.)),9));

ss = put(scan(strip(dvar),1,num),$1.);

if strip(dvar) ne cat(strip(yy),ss,strip(mm),ss,strip(dd)) and strip(dvar) ne cats(yy,mm,dd) then goto uexit;
end

* convert DD, MM and YY to numeric values

if notdigit(trim(dd))=0 then ddn = input(dd,2.);
if notdigit(trim(mm))=0 then mmn = input(mm,2.);
if notdigit(trim(yy))=0 then yyn = input(yy,4.);

if ddn=. or (mmn=. & fmt ne 'DATE') or yyn < 1582 or yyn > 2999 then goto uexit;

* check that DD, MM and YY is a valid date

if fmt='DATE' then

n = ((mm in ('JAN','MAR','MAY','JUL','AUG','OCT','DEC') and 1 <= ddn <= 31) or

(mm in ('APR','JUN','SEP','NOV') and 1 <= ddn <= 30) or

(mm = 'FEB' and 1 <= ddn <= 28) or

(mm = 'FEB' and ddn = 29 and (index (leapyr, substr (yy, 3, 2)) or

yyn in (1600,2000,2400,2800))));

else n = ((mmn in (1,3,5,7,8,10,12) and 1 <= ddn <= 31) or

(mmn in (4,6,9,11) and 1 <= ddn <= 30) or

(mmn = 2 and 1 <= ddn <= 28) or

(mmn = 2 and ddn = 29 and (index (leapyr, substr (yy, 3, 2)) or

yyn in (1600,2000,2400,2800))));

uexit:

return (n);
endsub;
run;

RECOMMENDED READING

SAS Online FCMP Guide:

FCMP Procedure User Guide:

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:
Jason A Smith
Argo Analytics Ltd
32 Woodlark Road
Cambridge CB3 0HS, UK
Work Phone: +44 7792 046599
Email: jason@argoanalytics.co.uk
LinkedIn: http://uk.linkedin.com/pub/jason-smith/46/baa/61
Web: http://www.argoanalytics.co.uk

Brand and product names are trademarks of their respective companies.