PharmaSUG 2024 - Paper PO-106

No LEAD Function? Let's Create It!

Xianhua Zeng, Elixir Clinical Research, Shanghai, China

ABSTRACT

In data analysis and statistical modeling, there are situations where the ability to access next observations (look-ahead) for specific variables is essential for making comparisons and calculations. Unfortunately, SAS does not provide a built-in LEAD function for this purpose. This paper presents a custom macro-based methodology to create a LEAD function using the PROC FCMP procedure. To access LEAD values in various scenarios, a versatile macro called "GetLeadValue" is also introduced. It supports obtaining lead values based on a specific by-group in dataset.

The macro code is currently hosted on my Github page:

https://raw.githubusercontent.com/XianhuaZeng/PharmaSUG/master/2024/GetLeadValue.sas

INTRODUCTION

Data analysis frequently involves the need to access future observations for specific variables, a task typically accomplished using a LEAD function. However, SAS lacks a built-in LEAD function, which can be a significant limitation for SAS users. This paper addresses this gap by presenting a custom macro-based approach to create a LEAD function using the PROC FCMP procedure. We also introduce a versatile macro ‘GetLeadValue’, which can be used to access LEAD values in various scenarios.

FUNCTION LEAD

In SAS, there is no direct function for calculating LEAD. So, it seemed appropriate to create a function LEAD with the opposite behavior of existing LAG function.

METHODOLOGY

The RUN_MACRO routine provides a convenient way to invoke a macro within an FCMP function. This powerful feature allows you to execute a macro and obtain a single value that can be returned to the DATA step. With the RUN_MACRO routine, you can seamlessly integrate the capabilities of a macro into your FCMP functions, enhancing the flexibility and functionality of your custom functions.

The code below is core part of function Lead. Please refer to the appendix I for the full code.

```
proc fcmp outlib = work.functions.demo;
    function lead(VAR $, SEQ);
        if SEQ = 1 then do;
            rec = run_macro('lead', VAR, SEQ);
        end;

        declare hash glv(dataset: "glv_temp");
        rc = glv.defineKey("SEQ");
        rc = glv.defineData("LEAD_VALUE");
        rc = glv.defineDone();

        rc = glv.find();
        return(LEAD_VALUE);
    endsub;
```

run;

USAGE EXAMPLE

LEAD function can be used in a similar way to other SAS functions.

```
options cmplib = work.functions;
data class;
    set sashelp.class;
run;

data demo;
    set class;
    AGE_LEAD = lead('AGE', _N_);
run;
```

MACRO GETLEADVALUE

MACRO PARAMETERS

Macro GetLeadValue has 5 input parameters as listed in Table 1 below:

Parameter	Description
dataIn	A dataset name which has variable needed to be calculated LEAD value. This is a REQUIRED parameter and does not have a default value.
dataOut	Output dataset name. This is a REQUIRED parameter and its default value is &dataIn.
varIn	A variable name for which LEAD value is calculated. Supports character variable and multiple variables. # Example values: NAME@AGE.
varOut	Output variable name. This is not a REQUIRED parameter and its default value is &GLV_&varIn.
byVar	Variable used to get lead value based on by group. This is not a REQUIRED parameter

Table 1. GetLeadValue Parameters

METHODOLOGY

Lead value can be calculated with the following logic:

Create a sequence of numbers and then sort the sequence by descending order. Then calculate lag of the variable for which we need to calculate lead. At last, sort the data by sequence ID.

```
data temp;
    set sashelp.class;
    seq + 1;
run;

proc sort data = temp;
    by descending seq;
```

```
run;

data want;
    set temp;
    LEAD_AGE = lag(AGE);
run;

proc sort data = want;
    by seq;
run;
```

The code above is not concise and is not effective. This macro uses Hash object. It has only one DATA step and is more effective.

```
data want;
    if _N_=1 then do;
        dcl hash h(ordered: "a") ;
        h.definekey("GLV_SEQ");
        h.definedata("GLV_SEQ", "LEAD_AGE");
        h.definedone();
        dcl hiter hi('h');
        do until(eof);
            set sashelp.class end=eof;
            LEAD_AGE=AGE;
            GLV_SEQ+1;
            h.add();
        end;
    end;
    set sashelp.class;
    hi.setcur(key: _N_);
    GLV_RC = hi.next();
    if GLV_RC ^= 0 then call missing(LEAD_AGE);
    drop GLV_SEQ GLV_RC;
run;.
```

USAGE EXAMPLE

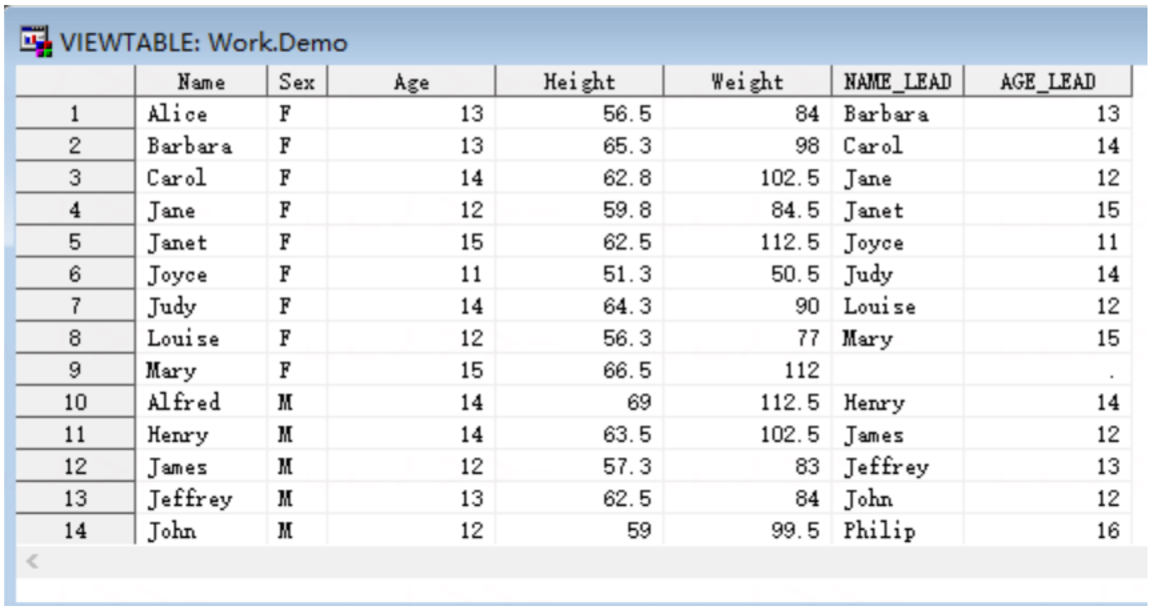
To include this macro in your SAS program, just run the following piece of code:

```
filename lead temp;
proc http
url="https://raw.githubusercontent.com/XianhuaZeng/PharmaSUG/master/2024/GetL
eadValue.sas"
    method = "GET"
    out = lead;
run;

filename code temp;
data _null_;
    file code;
    infile lead;
    input;
    put _infile_;
run;

%inc code;
```

Please refer to the appendix II for the full code. Let's illustrate how to use the "GetLeadValue" macro to obtain LEAD values for a specific dataset and columns:

```
%GetLeadValue(dataIn   = sashelp.class
             , dataOut = demo
             , varIn   = NAME @ AGE
             , varOut  = NAME_LEAD @ AGE_LEAD
             , byVar   = SEX
             );
```

Expected result is shown below:


VIEWTABLE: Work.Demo

	Name	Sex	Age	Height	Weight	NAME_LEAD	AGE_LEAD
1	Alice	F	13	56.5	84	Barbara	13
2	Barbara	F	13	65.3	98	Carol	14
3	Carol	F	14	62.8	102.5	Jane	12
4	Jane	F	12	59.8	84.5	Janet	15
5	Janet	F	15	62.5	112.5	Joyce	11
6	Joyce	F	11	51.3	50.5	Judy	14
7	Judy	F	14	64.3	90	Louise	12
8	Louise	F	12	56.3	77	Mary	15
9	Mary	F	15	66.5	112		.
10	Alfred	M	14	69	112.5	Henry	14
11	Henry	M	14	63.5	102.5	James	12
12	James	M	12	57.3	83	Jeffrey	13
13	Jeffrey	M	13	62.5	84	John	12
14	John	M	12	59	99.5	Philip	16

Figure 1. Expected Result

CONCLUSION

Creating a custom LEAD function in SAS using PROC FCMP fills a critical gap in situations where the built-in LEAD function is not available. Additionally, the "GetLeadValue" macro supports obtaining LEAD values based on specific groups or by-group variables, expanding the utility of the custom LEAD function.

REFERENCE

The FCMP Procedure. Available at https://support.sas.com/documentation/onlinedoc/base/91/fcmp.pdf

CONTACT INFORMATION

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

Name: Xianhua Zeng
Enterprise: Elixir Clinical Research
Address: Block 11, No. 889 Tianlin Road, Minhang District, Shanghai, 201103
E-mail: xianhua.zeng@ecr-global.com
Web: http://www.ecr-global.com/
http://www.xianhuazeng.com/

APPENDIX I

```
%macro lead;
%let dsn = _last_;
%let var = %sysfunc(dequote(&var));

data glv_temp;
    if _N_=1 then do;
        dcl hash h(ordered: "a") ;
        h.definekey("GLV_SEQ");
        h.definedata("GLV_SEQ", "LEAD_VALUE");
        h.definedone();
        dcl hiter hi('h');
        do until(eof);
            set &dsn end = eof;
            LEAD_VALUE = &var;
            GLV_SEQ + 1;
            h.add();
        end;
    end;
    set &dsn;
    SEQ = _N_;
    hi.setcur(key: _N_);
    GLV_RC = hi.next();
    if GLV_RC ^= 0 then call missing(LEAD_VALUE);
    drop GLV_SEQ GLV_RC;
run;
%mend lead;

proc fcmp outlib = work.functions.demo;
    function lead(VAR $, SEQ);
        if SEQ = 1 then do;
            rec = run_macro('lead', VAR, SEQ);
        end;

        declare hash glv(dataset: "glv_temp");
        rc = glv.defineKey("SEQ");
        rc = glv.defineData("LEAD_VALUE");
        rc = glv.defineDone();

        rc = glv.find();

        return(LEAD_VALUE);
    endsub;

    function leadc(VAR $, SEQ) $;
        if SEQ = 1 then do;
            rec = run_macro('lead', VAR, SEQ);
        end;

        length LEAD_VALUE $200;

        declare hash glv(dataset: "glv_temp");
        rc = glv.defineKey("SEQ");
        rc = glv.defineData("LEAD_VALUE");
        rc = glv.defineDone();

        rc = glv.find();

        return(LEAD_VALUE);
    endsub;
run;
```

APPENDIX II

```
%macro GetLeadValue(dataIn  =
                    , dataOut = &dataIn
                    , varIn   =
                    , varOut  =
                    , byVar   =
                    );

/* Parameter check */
/* Check if parameter is null */
%if "%superQ(dataIn)" = "" %then %do;
    %let glv_returnmsg = Source dataset (dataIn) is null.;
    %goto MacErr;
%end;

%if "%superQ(varIn)" = "" %then %do;
    %let glv_returnmsg = Source variable (varIn) is null.;
    %goto MacErr;
%end;

/* Check if parameter is valid */
/* Check if parameter dataIn is a valid library name and/or dataset name */
%if %qSysFunc(prxMatch(%qSysFunc(
        prxParse(/^%bQuote(([a-zA-Z_]{1}[a-zA-Z_0-9]{0,7}\.)?[a-zA-Z_]{1}[a-
zA-Z_0-9]{0,31})$/)), %superQ(dataIn))) ~= 1 %then %do;
    %let glv_returnmsg = %str(Parameter dataIn=&dataIn. has an invalid
library name and/or dataset name, please correct.);
    %goto MacErr;
%end;

/* Check if parameter dataOut is a valid library name and/or dataset name */
%if %qSysFunc(prxMatch(%qSysFunc(
        prxParse(/^%bQuote(([a-zA-Z_]{1}[a-zA-Z_0-9]{0,7}\.)?[a-zA-Z_]{1}[a-
zA-Z_0-9]{0,31})$/)), %quote(&dataOut))) ~= 1 %then %do;
    %let glv_returnmsg = %str(Parameter dataOut=&dataOut. has an invalid
library name and/or dataset name, please correct.);
    %goto MacErr;
%end;
```

```
%do i = 1 %to %sysfunc(countw(&varIn, @));
    %let varIn = &varIn;
    %let varIn = %sysfunc(prxchange(s/\@$//i, -1, %quote(&varIn)));
    %let varIn&i = %scan(&varIn, &i, @);
    /* Check if parameter varIn is a valid variable name */
    %if %qSysFunc(prxMatch(%qSysFunc(
            prxParse(/^%bQuote([a-zA-Z_]{1}[a-zA-Z_0-
9]{0,31})$/)), %quote(&&varIn&i))) ~= 1 %then %do;
        %let glv_returnmsg = %str(Parameter varIn=&&varIn&i. has an invalid
variable name, please correct.);
        %goto MacErr;
    %end;

    /* Check if parameter varIn exists in dataset dataIn */
    %if %index(&dataIn, .) %then %do;
        %let glv_lib = %sysfunc(prxchange(s/(.+)\.(.+)$/\U\1/,
1, %quote(&dataIn)));
        %let glv_dsn = %sysfunc(prxchange(s/(.+)\.(.+)$/\U\2/,
1, %quote(&dataIn)));
    %end;
    %else %do;
        %let glv_lib = WORK;
        %let glv_dsn = %upcase(&dataIn);
    %end;

    proc sql noprint;
        select NAME into :var_list separated by ' '
          from dictionary.columns
          where LIBNAME = "&glv_lib" and MEMNAME = "&glv_dsn"
          ;
    quit;

    %if not %sysfunc(prxmatch(/\b&&varIn&i\b/i, %quote(&var_list))) %then %do;
        %let glv_returnmsg = %str(Parameter varIn=&&varIn&i. does not exist
in dataset &dataIn, please check.);
        %goto MacErr;
    %end;
%end;
```

```
/* Check if parameter byVar exists in dataset dataIn */
%if "%superQ(byVar)" ^= "" %then %do;

    %let byVar = %sysfunc(compbl(%sysfunc(strip(&byVar))));

    %let gvl_byVar = %sysfunc(prxchange(s/\b(descending)\b//i, -
1, %quote(&byVar)));

    %do i=1 %to %sysfunc(countw(&gvl_byVar));
        %let glv_byVar_ind = %scan(&gvl_byVar, &i, %str( ));

        %if
not %sysfunc(prxmatch(/&glv_byVar_ind/i, %quote(&var_list))) %then %do;
            %let glv_returnmsg = %str(Parameter byVar &glv_byVar_ind. does
not exist in dataset &dataIn, please check.);
            %goto MacErr;
        %end;
    %end;
%end;

%if "%superQ(varOut)" ^= "" %then %do;
    /* Check if variable number of varOut is equal with number of varIn */
    %if  %sysfunc(countw(&varIn, @)) ^= %sysfunc(countw(&varOut,
@)) %then %do;
        %let glv_returnmsg = %str(Variable number of varOut is not equal with
variable number of varIn);
        %goto MacErr;
    %end;

    /* Check if parameter varOut is a valid variable name */
    %do i = 1 %to %sysfunc(countw(&varOut, @));
        %let varOut = &varOut;
        %let varOut = %sysfunc(prxchange(s/\@$//i, -1, %quote(&varOut)));
        %let varOut&i = %scan(&varOut, &i, @);
        %if %qSysFunc(prxMatch(%qSysFunc(
                prxParse(/^%bQuote([a-zA-Z_]{1}[a-zA-Z_0-
9]{0,31})$/)), %quote(&&varOut&i))) ~= 1 %then %do;
            %let glv_returnmsg = %str(Parameter varOut= &&varOut&i. has an
invalid variable name, please correct.);
```

```
            %goto MacErr;
        %end;
    %end;
%end;

%if "%superQ(varOut)" = "" and
not %sysfunc(prxmatch(/\@/, %quote(&varIn))) %then %let varOut=GLV_&varIn;
%else %if "%superQ(varOut)" = ""
and %sysfunc(prxmatch(/\@/, %quote(&varIn))) %then %let
varOut=GLV_%sysfunc(prxchange(s/\@/\@GLV_/, -1, %quote(&varIn)));;

/* Drive variable */
%if "%superQ(byVar)" ^= "" %then %do;
    proc sort data = &dataIn out = glv_temp;
        by &byvar;
    run;
%end;

%do i = 1 %to %sysfunc(countw(&varIn, @));
  data &dataOut;
      if _N_ = 1 then do;
          dcl hash h(ordered: "a") ;
          h.definekey("GLV_SEQ");
          h.definedata("GLV_SEQ", "%scan(&varOut, &i, @)");
          h.definedone();
          dcl hiter hi('h');
          do until(eof);
              set
              %if &i=1 %then %do;
                  %if "%superQ(byVar)" = "" %then &dataIn;
                  %else glv_temp;
              %end;
              %else &dataOut;
              end = eof;
              %scan(&varOut, &i, @) = %scan(&varIn, &i, @);
              GLV_SEQ + 1;
              h.add();
          end;
      end;
      set
      %if &i=1 %then %do;
          %if "%superQ(byVar)" = "" %then &dataIn;
          %else glv_temp;;
      %end;
      %else &dataOut;;
      hi.setcur(key: _N_);
      GLV_RC = hi.next();
      if GLV_RC ^= 0 then call missing(%scan(&varOut, &i, @));
      drop GLV_SEQ GLV_RC;
  run;
%end;

/* For BY group*/
%if "%superQ(byVar)" ^= "" %then %do;
    data &dataOut;
        set &dataOut;
        by &byVar;
        %do i = 1 %to %sysfunc(countw(&varOut, @));
            if last.&glv_byVar_ind then call missing(%scan(&varOut, &i, @));
        %end;
    run;

    /* Remove temp dataset */
    proc datasets library = work nolist;
        delete glv_temp;
    quit;

%end;

%goto macEnd;

/* ERR HANDLING */
%macErr:;

%if "%superQ(glv_returnmsg)" ^= "" %then %do;
```

```
    %put ERROR[ELR]: &glv_returnmsg;
    %ABORT CANCEL;
%end;

%macEnd:;

%mend GetLeadValue;
```