Existence is Fruitful: How to Determine if Something Exists in Your SAS® Session

Erin O’Dea – NORC at the University of Chicago

Why would you want to know?

- There are lots of reasons to check existence in SAS!
  - Programmatically skip over code
  - Prevent processing problems due to lack of data
  - Testing things between programs and losing track of what is assigned
- SAS has some built in functions and routines that can help!
- We can also access dictionary views and properties that can help us find the rest

EXIST – Does your dataset exist?

- Does what it says on the tin!
- For Datasets, Views, and Catalogs
- Returns 1 if it exists and 0 if it does not
- Syntax: EXIST(*member-name* <, *member-type* <, *generation*>>)

```
%let dsname=sasuser.houses;
%macro opends(name); %if
%sysfunc(exist(&name)) %then %let
dsid=%sysfunc(open(&name, i));
%else %put Data set &name does
not exist.; %mend opends;
%opends(&dsname);
```

Dataset/Macro

```
data _null_;
dsname="test.myview";
   if (exist(dsname, "VIEW"))
then
      dsid=open(dsname, "i");
   else put dsname 'does not
exist.';
run;
```

View/Datastep

Checking Columns

DSID method

```
%macro VarExist(ds, var);
%local rc dsid;
%let dsid = %sysfunc(open(&ds));
%if %sysfunc(varnum(&dsid, &var)) > 0
%then %do; %put &var. exists; %end;
%else %do; %put &var does not exist; %end;
%let rc = %sysfunc(close(&dsid));
%mend VarExist;
```

Dictionary method

```
%macro VarExist(lib, ds, var);
proc sql noprint;
select * from dictionary.columns
where upcase(libname)=upcase("&lib.")
and upcase(memname)=upcase("&ds.")
and upcase(name)=upcase("&var.");
quit;
%if &sqlobs.=0
%then %do; %put &var does not exist; %end;
%else %do; %put &var. exists; %end;
%mend VarExist;
```

You can find out if datasets exist with similar methods too!

Checking Datasets

DSID method

```
%macro DataExist(ds);
%local rc dsid;
%let dsid = %sysfunc(open(&ds));
%let rc = %sysfunc(close(&dsid));
%if &dsid. NE 0 %then %do; %put &ds. exists;
%end;
%else %do; %put &ds does not exist; %end;
%mend DataExist;
```

Dictionary method

```
%macro DataExist(lib, ds);
proc sql noprint;
select * from dictionary.tables
where upcase(libname)=upcase("&lib.")
and upcase(memname)=upcase("&ds.");
quit;
%if &sqlobs.=0 %then %do; %put &ds. not
found; %end;
%else %do; %put found &ds.!; %end;
%mend DataExist;
%dataexist(sashelp, class);
```

Don’t forget EXIST exists!

FILEREF and FILEEXIST

FILEREF

- Syntax: FILEREF(*fileref*)
- Must be assigned to an external file via a filename statement or function
- Negative means *fileref* exists but the physical file associated with the *fileref* does not exist.
- Positive means that the *fileref* is not assigned.
- Zero means that the *fileref* and external file both exist.

FILEEXIST

- Syntax: FILEEXIST(*file-name*)
- Can also verify directories
- Accepts a character constant, variable name, or expression
- Does not need to be assigned to a filename
- Returns 1 if the file exists and 0 if it does not

Ever realize too late your library isn't valid?

```
1          OPTIONS NONOTES NOSTIMER NOSOURCE NOSYNTAXCHECK;
72
73         libname mylib "C:\Myfiles\PretendFile";
NOTE: Library MYLIB does not exist.
74
75         OPTIONS NONOTES NOSTIMER NOSOURCE NOSYNTAXCHECK;
87
```

SAS produces a NOTE when libnames are assigned to non-existent paths. This can be easily missed, creating ERRORs further down and hiding the real problem

LIBREF

- Verifies that a library has been assigned
- Returns 0 if assigned and non-zero if not assigned
- Syntax: LIBREF(*libref*)
- Verifies assignment, not existence
- Can accept a character expression, enclosed string, or variable name

```
%macro check(lib);
%if (%sysfunc(libref(&lib.)))
%then
  %put does not exist;
  %else %put exists;
%mend;
```

Non-zero response means we do not have it

Ever create a macro variable this way?

```
73          proc sql;
74          select distinct name into :name
75          from sashelp.class
76          where name like 'Erin';
NOTE: No rows were selected.
77          quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.02 seconds
      cpu time            0.01 seconds


78          data _null_;
79          set sashelp.class;
80          CALL SYMPUTX(name, name);
81          where name like 'Erin';
82          run;

NOTE: There were 0 observations read from the data set SASHELP.CLASS.
      WHERE name='Erin';
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

```
83          proc sql;
84          select * from sashelp.class
85          where name="&name.";
WARNING: Apparent symbolic reference NAME not resolved.
```

Creating macro variables via PROC SQL or call symput/x routines can result in a lack of existence if SAS does not find any rows to select from – this can cause failures down the line and mask the real issue!

SYMEXIST or %SYMEXIST

- Verifies existence of a macro variable
- Has an associated macro function
- Returns 1 if it exists and 0 if it does not exist
- Syntax: SYMEXIST (*argument*)
- Syntax: %SYMEXIST (*argument*)
  - Argument can be:
    - Macro variable name – in double quotes or unquoted, without the ampersand
    - A character variable – without quotes, which contains a macro variable name

```
%macro test;
 %if %symexist(name) %then %do;
proc sql;
select * from sashelp.class
where name="&name.";
quit; %end;
        %else %put MACRO NOT FOUND;
%mend test;
```

%SYSMACEXIST

- Verifies that a macro definition exists in your session
- Looks at the **Work.SASMacr** and the **Work.SASMac*n*** catalogs
- Syntax: %SYSMACEXIST(*macro-name*)
  - Macro-name here does not include percentage sign
- Returns 1 if the macro definition exists and 0 if it does not

```
%macro test;
        %if %SYSMACEXIST(alright)
%then %put FOUND;
        %else %put NOT FOUND;
%mend test;
```

Non-zero response means the macro definition exists in your session

Thank You for Listening!

- Erin O’Dea
- NORC at the University of Chicago
- 55 E Monroe St. Chicago, IL
- odea-erin@norc.org