Branch: master

Find file | Copy path

sas / Week_2_Accessing Data / Week2AccessingData.sas

kogilo more content

a6a3f4d 12 days ago

1 contributor

Raw | Blame | History

550 lines (381 sloc) 16.2 KB

```
* Week 2 - Accessing Data
* Types of Data
* 2 types : Structured and unstractured


* Structured Data
    * - defined rows and columns
    * - include - SAS, Microsoft Access, Hadoop, and other
    * - engines enables SAS to read structures data

* Unstructured Data
    * - no definied colomns
    * - text, delimited, JSON, webblogs and other
    * - must be imported into SAS

* What is a SAS Table?
* - is structured data file
* - define rows and columns
* - has file extension ` .sas7bdat `
* Has two parts:
    * - descriptor
        * contain the metadata
            * - table name
            * - number of rows
            * - column names
            * - column attributes
    * - data
        * - data values
*  Column or variable
* row or observation


* **************************
* Required Column Attributes for SAS Tables
* *****************************
* What does it mean for column to be defined?
    * - columns has three attribute
        * Name:
            * can be 1 - 32 characters long
            * start with letter or underscore
            * continues with letters numbers, or underscrores
            * uppercase, lowercase, or mixed case

        * Type
            * Two types:
                * Numeric
                    * digits 0 - 9
                    * minus sign  - 20568
                    * decimal point  - -25.43
                    * scientific notation (E) - 20E5
                * Character
                    * letters - CA
                    * numbers - 555-1212
                    * special character 20568
                    * blanks #Love this product!
                * SAS Dates
                    * 01 Jan1960   - 0 -

        * Length
            * related with Numeeric and Character
                * Numeeric
```

```
                     * 8 bytes ( ~ 16 significant digits )
                 * Character
                     * 1 - 32,767 bytes (1byte = on character)
                     * eg FR - has length 2, FRANCE hase 6 lenght
             *


* ************************************
* Listing Table and Column Attributes
*************************************

* But another way to view the table attributes is to write a Proc Contents step.
* The syntax:

 *  PROC CONTENTS DATA = data-set;
 *  RUN;

proc contents data="filepath/class_birthdate.sas7bdat";
run;

* The first 2 sections of the report give general information about the table.
* Including wwhere the table is stored, when it created and modified, the number of row and columns.
* Next show the aphabetic list of varaibles and attributes.
* For eg. Birthdate is a numeric column and missing numeric
*  values are stored as a period. Missing character values are stored as a space.

* **************
* Activity 2.03
**************
*  1. In a new program window, write a PROC CONTENTS step to generate
* a report of the storm_summary.sas7bdat table. Be sure to specify the path
* to your EPG194/data folder and the full name of the table.

    * Run the program.

    * How many observations (rows) are in the table? Note: Type a number for your answer.


```

```
proc contents data="EPG194/data/storm_summary.sas7bdat";
run;


* *********************************
*  Accessing Data in a Program
**********************************

* So far we have been using the hardcoded path
* w/c need 2 info - Location and name and type of data
 * Probelm may arise If
    * we have: long program, change data location, change to other data types
    * All of these issues can be solved by using a Library
    * SAS library
**********************************
* Using a Library to Read SAS Data
***********************************
* SAS library required you to specify
    * - Location
    * - type of data
* You create a SAS Library as :

LIBNAME libref engine "path";
* - LIBNAME - is a keyword
* - libref
    * - is library name
        * - eight-character maximum
        * - starts with letter or underscore
        * - continues with letters, numbers or underscores
* - engine
    * set of instructions
    * includes:
        * - Base
        * - Excel
        * - Teeradata
        * - Hadoop
        * - etc
* - "path"
```

```
    * - Location

* The LIBNAME is a global statment. It doesn't need a Run statment at the end
* Example

libname mylib base "s:/workshop/data";

* Library name - mylib
* Base engine - base
* location - s:/worksop/data

* base is the default engine, so you could write without it as follow:

 libname mylib "s:/workshop/data";

    * you use the library to access data:
  libref.table-name


proc contents data=mylib.class;
run;


proc contents data=mylib.class;
run;


* if your data move to another location, you have to only edit one statement

* delete libray refrence

libname mylib clear;


* *************************************************************************
* Activity 2.04: Create a Library for This Course (Required)
************************************************

```

```
* Open a new program window in SAS Studio.
* Write a LIBNAME statement to create a library named PG1 that reads SAS tables in the
* EPG194/data folder. If you are not sure of the path to your data folder, right-click the data folder in the navigation
* You can copy the path shown there.

*libname mylib base "s:/workshop/data";

run;

libname PG1 base "s:/home/u48576857/EPG194/data";



* 2. Run the code. After the code runs, you should see a note in the log that the library was successfully assigned.

* 3. Select the Code tab. Save your program as libname.sas in the EPG194 folder. You can replace the file if it already
* 4. Select Libraries in the navigation pane and expand My Libraries.
* 5. Expand the PG1 library and view the list of SAS tables.
* Why are the Excel and text files in the data folder not included in the library?
 * ==== The PG1 library uses the BASE engine, so it reads only SAS tables. In your LIBNAME statement the path should be



************************************************************
**********   Automatic SAS Libraries     *******************
************************************************************

* Work Library
    * - is a temporary library that automatically defined by SAS
    * - contents deleted at end of SAS session
    * - default library
    * Eg.

data=work.test
data=test

* Sashelp library
    * - includes sample data
    data=sashelp.cars


*****************************************************
********* Demo: Exploring Automatic SAS Libraries ***
*****************************************************



***********************************************************;
*  Exploring Automatic SAS Libraries                      *;
***********************************************************;
*  Syntax                                                 *;
*                                                         *;
*    Work library - personal temporary tables             *;
*    Sashelp library - sample tables                      *;
*                                                         *;
*    WORK is the default library                          *;
*    **equivalent statements**                            *;
*    proc contents data=work.class;                       *;
*    proc contents data=class;                            *;
***********************************************************;

***********************************************************;
*  Demo                                                   *;
*  1) Run the demo program and use the navigation pane to *;
*     examine the contents of the Work and Out libraries. *;
*  2) Which table is in the Work library? Which table is  *;
*     in the Out library?                                 *;
*  3) Restart SAS.                                        *;
*     * Enterprise Guide: In the Servers list, select     *;
*       Local and click Disconnect. Click Yes in the      *;
*       confirmation window. Expand Local to start SAS    *;
*       again, and then expand Libraries.                 *;
*     * SAS Studio: Select More application options ->    *;
*       Reset SAS Session.                                *;
*  4) Discuss the following questions:                    *;
*     a) What is in the Work library?                     *;
*     b) Why are the out and pg1 libraries not available? *;
*     c) Is class_copy2 saved permanently?                *;
*     d) What must be done to re-establish the out        *;
*        library?                                         *;
*  5) To re-establish the pg1 library, open and run the   *;
*     libname.sas program saved previously in the main    *;
*     course files folder.                                *;
***********************************************************;

*Modify the path if necessary;
libname out "s:/workshop/output";

data class_copy1 out.class_copy2;
        set sashelp.class;
run;

* It ruturn error b/c the default library is WORK

Reset the sas session  -- at More application options
class_copy1 will be delete from WORK



* ***************************************************************************
** Using a Library to Read Other File Types   *****************************
*****************************************************************************

* You can use XLSX engine to read data directly from excel
* requires license for SAS/ACCESS to PC Files
* Now the create library statmenet will look like:

LIBNAME libref XLSX "path/file-name.xlsx"

run;
libname xlclass xlsx "s:/workshop/data/class.xlsx";

* There are two extra statements that you often use when you read Excel data.
* The first is the OPTIONS statement, a global statement for specifying system options.
```

```

run;
LIBNAME libref XLSX "path/file-name.xlsx"
OPTIONS option(s);
* Eg
* run;

OPTIONS VALIDVARNAME=v7;
* In this case, SAS replace the space between name with under_score
* When you define a connection to a data source such as Excel or other databases, it's a good practice to clear, or dele

* run;
LIBNAME libref CLEAR;

* In this example, we use the OPTIONS statement to enforce SAS naming conventions for the columns.
* Then, we create the xlclass library with the XLSX engine to read data from the class Excel workbook located in s:/work
* The PROC CONTENTS step is reading the class_birthdate worksheet in the class workbook. At the end, we clear the xlclas

* run;

options validvarname=v7;
libname xlclass xlsx "s:/workshop/data/class.xlsx";

proc contents data = xlclass.class_birthdate;
run;

libname xlclass clear;



* ****************************************************************
**** Demo: Using a Library to Read Excel Files ******************
*****************************************************************

* run;

options validvarname=v7;

```

```

libname xlstorm xlsx "s:/workshop/data/strom.xlsx";


* run the above 2 statments first
* run;

proc contents data=xlstorm.storm_summary;
run;

libname xlstorm clear;


* Now run the whole program


******************************************
***** Activity 2.05 *********************
******************************************

* 1 . In a new program window, write a LIBNAME statement to create a library named NP that reads np_info.xlsx in the data
* Be sure to specify the full path to your EPG194/data folder and the complete file name.
* 2. Run the code.
* 3. Navigate to the Libraries panel and open the NP library.

* How many tables are there in the NP library?
* run;

libname NP xlsx "s:/home/u48576857/EPG194/data/np_info.xlsx";
proc contents data=NP.Parks;
run;

libname NP clear;

* Write an OPTIONS statement to ensure that column names follow SAS naming conventions.
* Write a PROC CONTENTS step to read the Parks table in the NP library.
* Add a LIBNAME statement after PROC CONTENTS to clear the NP library.
* Run the program and examine the log. What changes to column names are noted in the log?
```

```


**************************************
***** Importing Unstructured Data ****
**************************************

* Import Wizards -- offer an just click and browse to impprt the file
* But learn the programming option

PROC IMPORT DATAFILE="path/filename" DBMS=filetype
            OUT=output-table;
RUN;


* Some options

PROC IMPORT DATAFILE="path/filename" DBMS=filetype
            OUT=output-table<REPLACE>
    <GUESSINGROWS=n|MAX;>
RUN;



*******************************************************
**** Demo: Importing a Comma-Delimited (CSV) File *****
*******************************************************

* run;

proc import DATAFILE="s:/workshop/data/storm_damage.csv" dbms=csv
              out=strom_damage_import replace ;

run;





```

```

proc contents data=strom_damage_import;

run;


* run the program;

*****************************************************
************ Activity 2.06 *************************
*****************************************************

* 1. In the PROC IMPORT statement, change the path to your EPG194/data folder. This program imports a tab-delimited file
* run;

proc import datafile="s:/home/u48576857/EPG194/data/storm_damage.tab"
                        dbms=tab out=storm_damage_tab replace;
run;



* 2. Run the program to import the data.
* 3. Suppose the original file changes and you want to refresh the SAS table. Run the code again.
* Did the import run on the second submission




******************************************
**** Importing an Excel File *************
******************************************

PROC IMPORT DATAFILE="path/file-name.xlsx" DBMS=XLSX
                    OUT=output-table <REPLACE>;
            SHEET=sheet-name
RUN;

proc import datafile="s:/workshop/data/class.xlsx"
```

```
            dbms=xlsx
            out=work.class_test_import replace;
run;

* *** XLSX engine ***
    * reads directly from Excel file
    * data is always current

* *** PROC IMPORT ***
    * creates copy of Excel file
    * data must be reimported if it changes



* ******************************************
**** Level 1 Practice: Importing Excel   ****
***** Data from a Single Worksheet       ****
*********************************************


* 1. In this practice, you create a table that contains a copy of the data that is in an Excel workbook.
* The Excel workbook contains a single worksheet.
* If necessary, start SAS Studio before you begin.


    * 1. Open p102p01.sas from the practices folder. Complete the PROC IMPORT step to read eu_sport_trade.xlsx.
    *  Be sure the replace FILEPATH with the path to your EPG194/data folder. Create a SAS table named
    * eu_sport_trade and replace the table if it exists.
* run;


proc import datafile="/home/u48576857/EPG194/data/eu_sport_trade.xlsx" DBMS=XLSX
                    out=eu_sport_trade replace;
            SHEET=sheet-name
run;


    * 2. Modify the PROC CONTENTS code to display the descriptor portion of the eu_sport_trade table.
    * 3. Submit the program, and then view the output data and the results.
    * How many variables are in the eu_sport_trade table?

* run;
proc contents data=eu_sport_trade;
run;



* SOLUTION;

proc import datafile="FILEPATH/eu_sport_trade.xlsx"
            dbms=xlsx
            out=eu_sport_trade
            replace;
run;

proc contents data=eu_sport_trade;
run;



** *******************************************************
*** Level 2 Practice: Importing Data from a CSV File
********************************************************

* 1. Open a new program window and write a PROC IMPORT step to read the np_traffic.csv file
* and create the traffic SAS table.
* run;


proc import datafile="/home/u48576857/EPG194/data/np_traffic.csv"
            dbms=csv
            out=traffic replace;
run;

* 2 Add a PROC CONTENTS step to view the descriptor portion of the newly created table.
* run;
```

```
proc contents data=traffic;
run;

* 4.Examine the data interactively. Scroll down to row 37.
* Notice that the values for ParkName and TrafficCounter seem to be truncated.

* 5. Modify the program to resolve this issue. Submit the program and verify that ParkName and
* TrafficCounter are no longer truncated;


PROC IMPORT DATAFILE="path/filename" DBMS=filetype
            OUT=output-table<REPLACE>
    GUESSINGROWS=n|MAX;
RUN;




************************************************
*** Solution **********************************
*********************************************

* run;

proc import datafile="FILEPATH/np_traffic.csv"
            dbms=csv
            out=traffic
            replace;
    guessingrows=max;
run;

proc contents data=traffic;
run;
```