Branch: master

Find file Copy path

sas / week_3_exploring_and_validating_data / week_3_exploring_validating_data.sas

kogilo week4

102c5b3 5 days ago

1 contributor

Raw Blame History

857 lines (586 sloc) 25.2 KB

```

**************************************
** Exploring Data with Procedures ****
***************************************

* After you access data, the next step is to make ssure that
* you understand it.
* You can use " PROC CONTENTS" to see the description portion of the table.
    * - PRINT
    * - MEANS
    * - UNIVARIATE
    * - FREQ
* ***************************
*** Activity 3.01 ***********
*****************************
* 1. Learning how to find answers in the SAS documentation is important for you as a programmer. Try it now.

        * 1. Go to support.sas.com/documentation. Click 9.4 after SAS Procedures by Name and Product.
        * 2. Look up the syntax for PROC PRINT (the PRINT Procedure).
        * 3. Which statement in PROC PRINT selects variables that appear in the report and determines their order?

    * BY
    * VAR ~
    * ID


**********************************************
*** Demo: Exploring Data with SAS Procedures **
***********************************************
* To print the first 10 rows;

proc print data=pg1.strom_summary (obs=10);
run;

* To limit the columns at print output;
proc print data=pg1.strom_summary (obs=10);
        var Season Name Basin MaxWindMPH MinPressure StartDate Enddate;
run;

* place the cursor after var and go to the table and select the columns name
* and drag and drop them.;

* we see that there are some missing values
* To compute the summary statistics;

proc means data=pg1.strom_summary;
        var MaxWindMPH MinPressure;  * what you want to analyize.
run;


* exanine extreme values;

proc UNIVARIATE data=pg1.strom_summary;
        var MaxWindMPH MinPressure;  * what you want to analyize.
run;


* list unique values and frequencies ;

proc FREQ data=pg1.strom_summary;
        tables Basin Type Season;
```

```
run;



*****************************************************************
*** Level 1 Practice: Exploring Data with Procedures *********
******************************************************************

* 1. If necessary, start SAS Studio. Open p103p01.sas from the practices folder and do the following:
    * 1. Complete the PROC PRINT statement to list the first 20 observations in pg1.np_summary.
    * 2. Add a VAR statement to include only the following variables: Reg, Type, ParkName, DayVisits, TentCampers, and R
    * 3. Highlight the step and run the selected code.
* Do you observe any possible inconsistencies in the data?

************************************************************;
*  LESSON 2, PRACTICE 1                                    *;
*    a) Complete the PROC PRINT statement to list the      *;
*       first 20 observations in PG1.NP_SUMMARY.           *;
*    b) Add a VAR statement to include only the following *;
*       variables: Reg, Type, ParkName, DayVisits,         *;
*       TentCampers, and RVCampers. Highlight the step     *;
*       and run the selected code.                         *;
*       Do you observe any possible inconsistencies in     *;
*       the data?                                          *;
*    c) Copy the PROC PRINT step and paste it at the end  *;
*       of the program. Change PRINT to MEANS and remove  *;
*       the OBS= data set option. Modify the VAR           *;
*       statement to calculate summary statistics for      *;
*       DayVisits, TentCampers, and RVCampers. Highlight  *;
*       the step and run the selected code.                *;
*       What is the minimum value for tent campers? Is     *;
*       that value unexpected?                             *;
*    d) Copy the PROC MEANS step and paste it at the end  *;
*       of the program. Change MEANS to UNIVARIATE.        *;
*       Highlight the step and run the selected code.      *;
*       Are there negative values for any of the columns? *;
*    e) Copy the PROC UNIVARIATE step and paste it at the *;
*       end of the program. Change UNIVARIATE to FREQ.     *;
*       Change the VAR statement to a TABLES statement to *;
*       produce frequency tables for Reg and Type.        *;
*       Highlight the step and run the selected code.     *;
*       Are there any lowercase codes? Are there any      *;
*       codes that occur only once in the table?          *;
*    f) Add comments before each step to document the     *;
*       program. Save the program as np_validate.sas in   *;
*       the output folder.                                *;
***********************************************************;

proc print data=PG1.np_summary (obs=20);
                var Reg Type ParkName DayVisits TentCampers RVCampers;
run;


proc means data=PG1.np_summary;
                var DayVisits TentCampers RVCampers;
run;


proc UNIVARIATE data=PG1.np_summary;
                var DayVisits TentCampers RVCampers;
run;


proc freq data=PG1.np_summary;
                tables DayVisits TentCampers RVCampers;
run;


*************************************************************************
*** Level 2 Practice: Using Procedures to Validate Data *****************
*************************************************************************

* run;
proc freq data=PG1.np_summary;
                tables Reg Type;
run;
```

```



************************************************
*** Filtering Rows with the WHERE Statement *****
*************************************************
* Use the WHERE statment;

proc procedure-name ....;
        WHERE expression ;
run;
  * Expression:
        * column
        * operator
                *  = or EQ
                *  ^= or ~= or NE
                * > or GT
                * < or LT
                * >= or GE
                * <= or LE
                * Example
                        * Type = "SUV"
                        * Type EQ "SUV"
                        * MSRP <= 30000
                        * MSRP LE 30000
        * value
                * Character values
                        * case sensitive
                        * enclosed in double or single quotation marks
                * Numeric values
                        * not enclosed in quotation marks
                        * standard values, no symbols
                * Numeric comparsion
                        * SAS data constant
                                * "ddmmmyyyy"d;
                                * Example
                                        * where date > "1jan15"d;
                                        * WHERE date > "01JAN2015"d;
```

```



**********************************************************
*** Combining Expressions in a WHERE Statement ***********
*********************************************************

* You can Combine expression using AND or OR;

proc print data=sashelp.cars;
        var Make Model Type MSRP MPG_City MPG_Highway;
        where Type = "SUV" and MSRP <= 30000;
run;



proc print data=sashelp.cars;
        var Make Model Type MSRP MPG_City MPG_Highway;
        where Type = "SUV" or Type="Truck" or Type="Wagon";
run;

* The following is more efficient;

WHERE col-name IN(value-1<..., value-n>);
WHERE col-name NOT IN(value-1<..,value-n>);


proc print data=sashelp.cars;
        var Make Model Type MSRP MPG_City MPG_Highway;
        where Type in ("SUV", "Truck", "Wagon");
run;



proc print data=sashelp.cars;
        var Make Model Type MSRP MPG_City MPG_Highway;
        where Type in ("SUV" "Truck" "Wagon");
run;
```

```


******************************************************
***** Demo: Filtering Rows with Basic Operators ****
******************************************************;

proc print data=pg1.strom_summary;
        where MaxWindMPH >= 156;
run;


proc print data=pg1.strom_summary;
        where Basin = "wp";
run;


proc print data=pg1.strom_summary;
        where Basin in ("SI" "NI");
run;



proc print data=pg1.strom_summary;
        where StartDate >= "01jan2010"d;
run;


proc print data=pg1.strom_summary;
        where Type="TS" and Hem_EW = "W";
run;


proc print data=pg1.strom_summary;
        where MaxWindMPH>156 or MinPressure<920;
run;

* The above result include missing values.;

```

```

proc print data=pg1.strom_summary;
        where MaxWindMPH>156 or 0<MinPressure<920;
run;

* Now missing values are excluded;



*****************************************************
**** Using Special WHERE Operators *****************
*****************************************************
* WHERE expression;

* Suppose you want to express your expression by missing values;

where Type =. or Type=" ";

* Or use the special operator;

WHERE col-name IS MISSING;
WHERE col-name IS NOT MISSING;

where Age is missing;
where Name is missing;

* For Data from DBMS;

where Item is null;


* Ranges;

WHERE col-name BETWEEN value-1 AND value-2;

where Age between 20 and 39;
* Inclusive;

* Pattern matching;

WHERE col-name LIKE "value";

* %   any number of characters
* _  single character;

* To return any string after NEW;

where City like "New%";

* to return single charater _ and %;

where City like "Sant_ %";
* Santa Clara, Santa Cruz, Santo Domingo, Santo Tomas

***********************************************************;
*  Filtering Rows with Basic Operators                    *;
***********************************************************;
*  Syntax and Example                                     *;
*                                                         *;
*    WHERE expression;                                    *;
*                                                         *;
*    Basic Operators:                                     *;
*         = , EQ                                          *;
*         ^= , ~= , NE                                    *;
*         > , GT                                          *;
*         < , LT                                          *;
*         >= , GE                                         *;
*         <= , LE                                         *;
*    SAS Date Constant                                    *;
*         "ddmmmyyyy"d ("01JAN2015"d)                     *;
***********************************************************;

proc print data=sashelp.cars;
        var Make Model Type MSRP MPG_City MPG_Highway;
        where Type="SUV" and MSRP <= 30000;
run;
```

```
***********************************************************;
*  Activity 3.03                                          *;
*    1) Uncomment each WHERE statement one at a time and  *;
*       run the step to observe the rows that are         *;
*       included in the results.                          *;
*    2) Comment all previous WHERE statements. Add a new  *;
*       WHERE statement to print storms that begin with   *;
*       Z. How many storms are included in the results?   *;
***********************************************************;

proc print data=pg1.storm_summary(obs=50);
        *where MinPressure is missing; /*same as MinPressure = .
        *where Type is not missing; /*same as Type ne " "*/
        *where MaxWindMPH between 150 and 155;
        *where Basin like "_I";


run;



*********************************************
*      Creating and Using Macro Variables    *
***********************************************
* macro variable -store strings   % &;
        * Step 1. Create the macro variable ;
        %LET macro-variable = value;

* Example;

%let CarType=Wagon;
```

```


*     &macro-var = &CarType

proc print data=sashelp.cars;
        where Type="&CarType";
        var Type Make Model MSRP;
run;


proc means data=sashelp.cars;
        where Type="&CarType";
        var MSRP MPG_Highway;
run;


proc freq data=sashelp.cars;
        where Type="&CarType";
        tables Origin Make;

run;


%let CarType=SUV;


*     &macro-var = &CarType

proc print data=sashelp.cars;
        where Type="&CarType";
        var Type Make Model MSRP;
run;


proc means data=sashelp.cars;
        where Type="&CarType";
        var MSRP MPG_Highway;
run;
```

```


proc freq data=sashelp.cars;
        where Type="&CarType";
        tables Origin Make;

run;



*******************************************
* Demo: Filtering Rows Using Macro Variables
*************************************************


************************************************************;
*  Filtering Rows Using Macro Variables                    *;
************************************************************;
*  Syntax and Example                                      *;
*                                                          *;
*    %LET macrovar=value;                                  *;
*                                                          *;
*    Usage:                                                *;
*    WHERE numvar=&macrovar;                               *;
*    WHERE charvar="&macrovar";                            *;
*    WHERE datevar="&macrovar"d;                           *;
************************************************************;

%let CarType=Wagon;

proc print data=sashelp.cars;
        where Type="&CarType";
        var Type Make Model MSRP;
run;

proc means data=sashelp.cars;
        where Type="&CarType";
        var MSRP MPG_Highway;
```

```
run;

proc freq data=sashelp.cars;
    where Type="&CarType";
        tables Origin Make;
run;

************************************************************;
*  Demo                                                    *;
*    1) Highlight the demo program and run the selected    *;
*       code.                                              *;
*    2) Write three %LET statements to create macro        *;
*       variables named WindSpeed, BasinCode, and Date.    *;
*       Set the initial values of the variables to match   *;
*       the WHERE statement.                               *;
*    3) Modify the WHERE statement to reference the macro *;
*       variables. Highlight the demo program and run the *;
*       selected code. Verify that the same results are   *;
*       produced.                                          *;
*    4) Change the values of the macro variables to        *;
*       values that you select. Possible values for Basin *;
*       include NA, WP, SP, WP, NI, and SI. Highlight the *;
*       demo program and run the selected code.            *;
************************************************************;

%let WindSpeed=156;
%let BasinCode=NA;
%let Date=01JAN2000;


proc print data=pg1.storm_summary;
        where MaxWindMPH>=&WindSpeed and Basin="&BasinCode" and StartDate>="&Date"d;
        var Basin Name StartDate EndDate MaxWindMPH;
run;

proc means data=pg1.storm_summary;
        where MaxWindMPH>=&WindSpeed and Basin="&BasinCode" and StartDate>="&Date"d;
        var MaxWindMPH MinPressure;
```

```
run;




*******************************************************
*** Formatting Data Values in Results        ***********
********************************************************

* To control how values appear in your reports;

proc print data=input-table;
        format col-name(s) format;
run;

* format; - affects display, not raw dat values;
        * specify as;

        <$>format-name<w>.<d>

* Example;
proc print data=pg1.class_birthdate;
        format Height Weight 3. Birthdate date9.;
run;


***************************************************
**  Common Formats for Numeric Values
***************************************************
* Format Name ****  Example Value *** Format Applied *** Formatted value;
     w.d              12345.67            5.               123456
     w.d              12345.67            8.1              12345.7
     COMMAw.d         12345.67           COMMA8.1          12,345.7
     DOLLARw.d        121345.67           DOLLAR10.2       $12,345.67
     DOLLARw.d        121345.67           DOLLAR10.        $12,346
     YENw.d           121345.67           YEN7.            Y12,346
     EUROXw.d         121345.67           EUROX10.2        €12,346

```

```

*******************************************
* Activity 3.05
***********************************

* Go to support.sas.com/documentation.
        * 1. Look up the Zw.d format.
        * 2. What does the format do?

Displays standard numeric data with leading zeroes.


Correct
Example: 1350 with the Z8. format applied would be displayed as 00001350

You can find this information by typing Zw.d in the search box and selecting link to Zw.d Format : : SAS 9.4 Formats and

You can also find this information by following these steps:

Under Popular Documentation, select Programming: SAS 9.4 and Viya.
Under Syntax - Quick Links, select Formats under Language Elements.
Select the link for Zw.d.


******************************************
** Common Formats for Date Values
************************************

Value              Format applied              Formatted value

21199                DATE7.                      15JAN18
21199                DATE9.                      15JAN2018
21199                MMDDYY10.                   01/15/2018
21199                DDMMYY8.                    15/01/18
21199                MONYY7.                     JAN2018
21199                MONNAME.                    January
21199                WEEKDATE                    Monday, January 15, 2018

```

```


**************************************************************
*** Demo: Formatting Data Values in Results
**************************************************************;

proc print data=pg1.strom_summary;
run;


* Now include format statemnt;

proc print data=pg1.strom_summary;
        format Date mmddyy10. Cost dollar16. Deaths comma5.;
run;

* With Change;

proc print data=pg1.strom_summary;
        format Date mmddyy8. Cost dollar14. Deaths comma5.;
run;

* mmddyy8.  width 8---the largest number will not be formatted;
proc print data=pg1.strom_summary;
        format Date mmddyy6. Cost dollar10. Deaths comma5.;
run;



***********************************************************;
*  Activity 3.06                                          *;
*    1) Highlight the PROC PRINT step and run the         *;
*       selected code. Notice how the values of Lat, Lon, *;
*       StartDate, and EndDate are displayed in the       *;
*       report.                                           *;
*    2) Change the width of the DATE format to 7 and run  *;
*       the PROC PRINT step. How does the display of      *;
*       StartDate and EndDate change?                     *;
*    3) Change the width of the DATE format to 11 and run *;
*       the PROC PRINT step. How does the display of      *;
*       StartDate and EndDate change?                     *;
*    4) Highlight the PROC FREQ step and run the selected *;
*       code. Notice that the report includes the number  *;
*       of storms for each StartDate.                     *;
*    5) Add a FORMAT statement to apply the MONNAME.      *;
*       format to StartDate and run the PROC FREQ step.   *;
*       How many rows are in the report?                  *;
***********************************************************;

proc print data=pg1.storm_summary(obs=20);
        format Lat Lon 4. StartDate EndDate date9.;
run;

proc freq data=pg1.storm_summary order=freq;
        tables StartDate;
        *Add a FORMAT statement;
run;


*****************************************
*** Sorting Data
******************************************
* Sorting
        * - improve visual arrangement of the data
        * - identify and remove duplicate rows
        * - prepare data for certain data processing steps;

PROC SORT
proc Sort data=input-table <out=output-table>;
        by <descending> col-name(s);
run;

* <descending>  --- overrides default ascending sort order;
* col-name(s) --- column(s) to sort by, or BY variables

* eg;
```

```

   by Name TestScore;

* ascending order by Name, then within Name by ascending TestSvore;
   by Subject decending TestScore;

* ascending order by Subject, then within Subject by descending TestScore;



*************************************
* Activity 3.07
*****************************************
* 1 Modify the OUT= option in the PROC SORT statement to create a temporary table named storm_sort.
* 2. Complete the WHERE and BY statements to answer the following question:
* Which storm in the North Atlantic Basin (NA or na) had the highest MaxWindMPH?;
* AN;   Allen

 proc sort data=pg1.storm_summary out=storm_sort;
     where Basin in("NA" "na");
     by descending MaxWindMPH;
 run;



***********************************************
* Identifying and Removing Duplicates
***********************************************;
proc sort data=input-table <out=output-table>
        NODUPRECS <DUPOUT=output-table>;
    BY_ALL_;
RUN;

* NODUPRECS <DUPOUT=output-table>  -- remove all adjacent duplocates

* _ALL_  -- sort by entire rows

* Example;
```

```

proc sort data=pg1.class_test3
        out=test_clean noduprecs dupout=test_dups;
    by _all_;
run;

* NODUPKEY;

proc sort data=input-table <out=output-table>
        NODUPKEY DUPOUT=output-table>;
      BY <descending> col-name(s);
run;

* NODUPKEY -- keeps only first occurrence of each unique value

* Example;

proc sort data=pg1.class_test2
        out=test_clean
        dupout=test_dups
        nodupkey;
      by Name;
run;

*****************************************************
* Demo: Identifying and Removing Duplicate Values
*****************************************************

************************************************************;
*  Identifying and Removing Duplicate Values              *;
************************************************************;
*  Syntax and Example                                     *;
*                                                         *;
*    Remove duplicate rows:                               *;
*    PROC SORT DATA=input-table <OUT=output-table>        *;
*        NODUPRECS <DUPOUT=output-table>;                 *;
*        BY _ALL_;                                        *;
*    RUN;                                                 *;
*                                                          *;
*    Remove duplicate key values:                          *;
*    PROC SORT DATA=input-table <OUT=output-table>         *;
*        NODUPKEY <DUPOUT=output-table>;                   *;
*        BY <DESCENDING> col-name (s);                     *;
*    RUN;                                                  *;
************************************************************;

************************************************************;
*  Demo                                                    *;
*    1) Modify the first PROC SORT step to sort by all     *;
*       columns and remove any duplicate rows. Write the   *;
*       removed rows to a table named STORM_DUPS.          *;
*       Highlight the step and run the selected code.      *;
*       Confirm that there are 107,821 rows in             *;
*       STORM_CLEAN and 214 rows in STORM_DUPS.            *;
*    2) Run the second PROC SORT step and confirm that     *;
*       the first row for each storm represents            *;
*       the minimum value of Pressure.                     *;
*       Note: Because storm names can be reused in         *;
*             multiple years and basins, unique storms     *;
*             are grouped by sorting by Season, Basin,     *;
*             and Name.                                    *;
*    3) Modify the third PROC SORT step to sort the        *;
*       MIN_PRESSURE table and keep the first row for      *;
*       each storm. You do not need to keep the removed    *;
*       duplicates. Highlight the step and run the         *;
*       selected code.                                     *;
************************************************************;

*Step 1;
proc sort data=pg1.storm_detail out=storm_clean noduprecs dupout=storm_dups;
        by _all_;
run;

*Step 2;
proc sort data=pg1.storm_detail out=min_pressure;
        where Pressure is not missing and Name is not missing;
        by descending Season Basin Name Pressure;
run;

*Step 3;
proc sort data=min_pressure nodupkey;
        by descending Season Basin Name;
run;

*************************************************************
* Level 1 Practice: Sorting Data and Creating an Output Table
************************************************************

************************************************************;
*  LESSON 3, PRACTICE 8                                    *;
*    a) Modify the PROC SORT step to read PG1.NP_SUMMARY  *;
*       and create a temporary sorted table named          *;
*       NP_SORT.                                           *;
*    b) Add a BY statement to order the data by Reg and   *;
*       descending DayVisits.                              *;
*    c) Add a WHERE statement to select Type equal to NP. *;
*       Submit the program.                                *;
************************************************************;

proc sort data=pg1.np_summary out=np_sort;
        where Type="NP";
        by Reg descending DayVisits;

run;

* AN 51;

proc sort data=pg1.np_summary out=np_sort;
    by Reg descending DayVisits;
    where Type="NP";
run;


**********************************************************
```

```
* Level 2 Practice: Sorting Data to Remove Duplicate Rows
************************************************************;

* The pg1.np_largeparks table contains gross acreage for large national parks. There are duplicate rows for some locatio

* Reminder: If you restarted your SAS session, you must run the libname.sas program in the EPG194 folder.

* Open and review the pg1.np_largeparks table. Notice that there are exact duplicate rows for some parks.
* Open a new program and write a PROC SORT step that creates two tables (park_clean and park_dups), and removes the dupl
* Submit the program and view the output data.
* How many rows are included in the park_dups table?

* 30;


proc sort data=PG1.NP_LARGEPARKS out=park_clean noduprecs dupout=park_dups;
        by _all_;
run;



* AN;

proc sort data=pg1.np_largeparks
                  out=park_clean
                  dupout=park_dups
                  noduprecs;
    by _all_;
run;
```