

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

```
1
2 *****
3 ** Exploring Data with Procedures **
4 *****
5
6 * After you access data, the next step is to make ssure that
7 * you understand it.
8 * You can use " PROC CONTENTS" to see the description portion of the table.
9     * - PRINT
10    * - MEANS
11    * - UNIVARIATE
12    * - FREQ
13 * *****
14 *** Activity 3.01 *****
15 *****
16 * 1. Learning how to find answers in the SAS documentation is important for you as a programmer. Try it now.
17
18     * 1. Go to support.sas.com/documentation. Click 9.4 after SAS Procedures by Name and Product.
19     * 2. Look up the syntax for PROC PRINT (the PRINT Procedure).
20     * 3. Which statement in PROC PRINT selects variables that appear in the report and determines their order?
21
22     * BY
23     * VAR ~
```

```
24      * ID
25
26
27      *****
28      *** Demo: Exploring Data with SAS Procedures **
29      *****
30      * To print the first 10 rows;
31
32      proc print data=pg1.strom_summary (obs=10);
33      run;
34
35      * To limit the columns at print output;
36      proc print data=pg1.strom_summary (obs=10);
37          var Season Name Basin MaxWindMPH MinPressure StartDate Enddate;
38      run;
39
40      * place the cursor after var and go to the table and select the columns name
41      * and drag and drop them.;
42
43      * we see that there are some missing values
44      * To compute the summary statistics;
45
46      proc means data=pg1.strom_summary;
47          var MaxWindMPH MinPressure; * what you want to analyze.
48      run;
49
50
51      * exanine extreme values;
52
53      proc UNIVARIATE data=pg1.strom_summary;
54          var MaxWindMPH MinPressure; * what you want to analyze.
55      run;
56
57
58      * list unique values and frequencies ;
59
60      proc FREQ data=pg1.strom_summary;
61          tables Basin Type Season;
```

```
62 run;
63
64
65
66 *****
67 *** Level 1 Practice: Exploring Data with Procedures *****
68 *****
69
70 * 1. If necessary, start SAS Studio. Open p103p01.sas from the practices folder and do the following:
71     * 1. Complete the PROC PRINT statement to list the first 20 observations in pg1.np_summary.
72     * 2. Add a VAR statement to include only the following variables: Reg, Type, ParkName, DayVisits, TentCampers, and RV
73     * 3. Highlight the step and run the selected code.
74 * Do you observe any possible inconsistencies in the data?
75
76 *****;
77 * LESSON 2, PRACTICE 1 *;
78 * a) Complete the PROC PRINT statement to list the *;
79 * first 20 observations in PG1.NP_SUMMARY. *;
80 * b) Add a VAR statement to include only the following *;
81 * variables: Reg, Type, ParkName, DayVisits, *;
82 * TentCampers, and RVCampers. Highlight the step *;
83 * and run the selected code. *;
84 * Do you observe any possible inconsistencies in *;
85 * the data? *;
86 * c) Copy the PROC PRINT step and paste it at the end *;
87 * of the program. Change PRINT to MEANS and remove *;
88 * the OBS= data set option. Modify the VAR *;
89 * statement to calculate summary statistics for *;
90 * DayVisits, TentCampers, and RVCampers. Highlight *;
91 * the step and run the selected code. *;
92 * What is the minimum value for tent campers? Is *;
93 * that value unexpected? *;
94 * d) Copy the PROC MEANS step and paste it at the end *;
95 * of the program. Change MEANS to UNIVARIATE. *;
96 * Highlight the step and run the selected code. *;
97 * Are there negative values for any of the columns? *;
98 * e) Copy the PROC UNIVARIATE step and paste it at the *;
99 * end of the program. Change UNIVARIATE to FREQ. *;
```

```
100 *      Change the VAR statement to a TABLES statement to *;
101 *      produce frequency tables for Reg and Type.          *;
102 *      Highlight the step and run the selected code.        *;
103 *      Are there any lowercase codes? Are there any         *;
104 *      codes that occur only once in the table?             *;
105 *      f) Add comments before each step to document the     *;
106 *      program. Save the program as np_validate.sas in      *;
107 *      the output folder.                                   *;
108 *****;
109
110 proc print data=PG1.np_summary (obs=20);
111         var Reg Type ParkName DayVisits TentCampers RVCampers;
112 run;
113
114
115 proc means data=PG1.np_summary;
116         var DayVisits TentCampers RVCampers;
117 run;
118
119
120 proc UNIVARIATE data=PG1.np_summary;
121         var DayVisits TentCampers RVCampers;
122 run;
123
124
125 proc freq data=PG1.np_summary;
126         tables DayVisits TentCampers RVCampers;
127 run;
128
129
130 *****
131 *** Level 2 Practice: Using Procedures to Validate Data *****
132 *****
133
134 * run;
135 proc freq data=PG1.np_summary;
136         tables Reg Type;
137 run;
```

```
138
139
140
141 *****
142 *** Filtering Rows with the WHERE Statement *****
143 *****
144 * Use the WHERE statment;
145
146 proc procedure-name ....;
147     WHERE expression ;
148 run;
149     * Expression:
150         * column
151         * operator
152             * = or EQ
153             * ^= or ~= or NE
154             * > or GT
155             * < or LT
156             * >= or GE
157             * <= or LE
158             * Example
159                 * Type = "SUV"
160                 * Type EQ "SUV"
161                 * MSRP <= 30000
162                 * MSRP LE 30000
163     * value
164         * Character values
165             * case sensitive
166             * enclosed in double or single quotation marks
167         * Numeric values
168             * not enclosed in quotation marks
169             * standard values, no symbols
170     * Numeric comparsion
171         * SAS data constant
172             * "ddmmmyyy"d;
173         * Example
174             * where date > "1jan15"d;
175             * WHERE date > "01JAN2015"d;
```

```
176
177
178
179 *****
180 *** Combining Expressions in a WHERE Statement *****
181 *****
182
183 * You can Combine expression using AND or OR;
184
185 proc print data=sashelp.cars;
186     var Make Model Type MSRP MPG_City MPG_Highway;
187     where Type = "SUV" and MSRP <= 30000;
188 run;
189
190
191
192 proc print data=sashelp.cars;
193     var Make Model Type MSRP MPG_City MPG_Highway;
194     where Type = "SUV" or Type="Truck" or Type="Wagon";
195 run;
196
197 * The following is more efficient;
198
199 WHERE col-name IN(value-1<..., value-n>);
200 WHERE col-name NOT IN(value-1<...,value-n>);
201
202
203 proc print data=sashelp.cars;
204     var Make Model Type MSRP MPG_City MPG_Highway;
205     where Type in ("SUV", "Truck", "Wagon");
206 run;
207
208
209
210 proc print data=sashelp.cars;
211     var Make Model Type MSRP MPG_City MPG_Highway;
212     where Type in ("SUV" "Truck" "Wagon");
213 run;
```

```
214
215
216 *****
217 ***** Demo: Filtering Rows with Basic Operators *****
218 *****;
219
220 proc print data=pg1.strom_summary;
221     where MaxWindMPH >= 156;
222 run;
223
224
225 proc print data=pg1.strom_summary;
226     where Basin = "wp";
227 run;
228
229
230 proc print data=pg1.strom_summary;
231     where Basin in ("SI" "NI");
232 run;
233
234
235
236 proc print data=pg1.strom_summary;
237     where StartDate >= "01jan2010"d;
238 run;
239
240
241 proc print data=pg1.strom_summary;
242     where Type="TS" and Hem_EW = "W";
243 run;
244
245
246 proc print data=pg1.strom_summary;
247     where MaxWindMPH>156 or MinPressure<920;
248 run;
249
250 * The above result include missing values.;
251
```

```
252
253 proc print data=pg1.strom_summary;
254     where MaxWindMPH>156 or 0<MinPressure<920;
255 run;
256
257 * Now missing values are excluded;
258
259
260
261 *****
262 **** Using Special WHERE Operators *****
263 *****
264 * WHERE expression;
265
266 * Suppose you want to express your expression by missing values;
267
268 where Type =. or Type=" ";
269
270 * Or use the special operator;
271
272 WHERE col-name IS MISSING;
273 WHERE col-name IS NOT MISSING;
274
275 where Age is missing;
276 where Name is missing;
277
278 * For Data from DBMS;
279
280 where Item is null;
281
282
283 * Ranges;
284
285 WHERE col-name BETWEEN value-1 AND value-2;
286
287 where Age between 20 and 39;
288 * Inclusive;
289
```



```

290 * Pattern matching;
291
292 WHERE col-name LIKE "value";
293
294 * % any number of characters
295 * _ single character;
296
297 * To return any string after NEW;
298
299 where City like "New%";
300
301 * to return single charater _ and %;
302
303 where City like "Sant_ %";
304 * Santa Clara, Santa Cruz, Santo Domingo, Santo Tomas
305
306 *****;
307 * Filtering Rows with Basic Operators *;
308 *****;
309 * Syntax and Example *;
310 * *;
311 * WHERE expression; *;
312 * *;
313 * Basic Operators: *;
314 * = , EQ *;
315 * ^= , ~= , NE *;
316 * > , GT *;
317 * < , LT *;
318 * >= , GE *;
319 * <= , LE *;
320 * SAS Date Constant *;
321 * "ddmmyyyy"d ("01JAN2015"d) *;
322 *****;
323
324 proc print data=sashelp.cars;
325     var Make Model Type MSRP MPG_City MPG_Highway;
326     where Type="SUV" and MSRP <= 30000;
327 run;

```

```
328
329
330
331
332
333
334
335 *****;
336 *   Activity 3.03                                     *;
337 *   1) Uncomment each WHERE statement one at a time and *;
338 *       run the step to observe the rows that are      *;
339 *       included in the results.                        *;
340 *   2) Comment all previous WHERE statements. Add a new *;
341 *       WHERE statement to print storms that begin with *;
342 *       Z. How many storms are included in the results? *;
343 *****;
344
345 proc print data=pg1.storm_summary(obs=50);
346     *where MinPressure is missing; /*same as MinPressure = .
347     *where Type is not missing; /*same as Type ne " "*/
348     *where MaxWindMPH between 150 and 155;
349     *where Basin like "_I";
350
351
352 run;
353
354
355
356 *****
357 *       Creating and Using Macro Variables             *
358 *****
359 * macro variable -store strings    % &;
360     * Step 1. Create the macro variable ;
361     %LET macro-variable = value;
362
363 * Example;
364
365 %let CarType=Wagon;
```

```
366
367
368 *      &macro-var = &CarType
369
370 proc print data=sashelp.cars;
371     where Type="&CarType";
372     var Type Make Model MSRP;
373 run;
374
375
376 proc means data=sashelp.cars;
377     where Type="&CarType";
378     var MSRP MPG_Highway;
379 run;
380
381
382 proc freq data=sashelp.cars;
383     where Type="&CarType";
384     tables Origin Make;
385
386 run;
387
388
389 %let CarType=SUV;
390
391
392 *      &macro-var = &CarType
393
394 proc print data=sashelp.cars;
395     where Type="&CarType";
396     var Type Make Model MSRP;
397 run;
398
399
400 proc means data=sashelp.cars;
401     where Type="&CarType";
402     var MSRP MPG_Highway;
403 run;
```

```
404
405
406 proc freq data=sashelp.cars;
407     where Type="&CarType";
408     tables Origin Make;
409
410 run;
411
412
413
414 *****
415 * Demo: Filtering Rows Using Macro Variables
416 *****
417
418
419 *****;
420 *   Filtering Rows Using Macro Variables                               *;
421 *****;
422 *   Syntax and Example                                                *;
423 *                                                                    *;
424 *   %LET macrovar=value;                                             *;
425 *                                                                    *;
426 *   Usage:                                                            *;
427 *   WHERE numvar=&macrovar;                                           *;
428 *   WHERE charvar="&macrovar";                                       *;
429 *   WHERE datevar="&macrovar"d;                                       *;
430 *****;
431
432 %let CarType=Wagon;
433
434 proc print data=sashelp.cars;
435     where Type="&CarType";
436     var Type Make Model MSRP;
437 run;
438
439 proc means data=sashelp.cars;
440     where Type="&CarType";
441     var MSRP MPG_Highway;
```

```
442 run;
443
444 proc freq data=sashelp.cars;
445     where Type="&CarType";
446     tables Origin Make;
447 run;
448
449 *****;
450 * Demo *;
451 * 1) Highlight the demo program and run the selected *;
452 * code. *;
453 * 2) Write three %LET statements to create macro *;
454 * variables named WindSpeed, BasinCode, and Date. *;
455 * Set the initial values of the variables to match *;
456 * the WHERE statement. *;
457 * 3) Modify the WHERE statement to reference the macro *;
458 * variables. Highlight the demo program and run the *;
459 * selected code. Verify that the same results are *;
460 * produced. *;
461 * 4) Change the values of the macro variables to *;
462 * values that you select. Possible values for Basin *;
463 * include NA, WP, SP, WP, NI, and SI. Highlight the *;
464 * demo program and run the selected code. *;
465 *****;
466
467 %let WindSpeed=156;
468 %let BasinCode=NA;
469 %let Date=01JAN2000;
470
471
472 proc print data=pg1.storm_summary;
473     where MaxWindMPH>=&WindSpeed and Basin="&BasinCode" and StartDate>="&Date"d;
474     var Basin Name StartDate EndDate MaxWindMPH;
475 run;
476
477 proc means data=pg1.storm_summary;
478     where MaxWindMPH>=&WindSpeed and Basin="&BasinCode" and StartDate>="&Date"d;
479     var MaxWindMPH MinPressure;
```

```

480 run;
481
482
483
484
485 *****
486 *** Formatting Data Values in Results *****
487 *****
488
489 * To control how values appear in your reports;
490
491 proc print data=input-table;
492     format col-name(s) format;
493 run;
494
495 * format; - affects display, not raw dat values;
496     * specify as;
497
498     <$>format-name<w>.<d>
499
500 * Example;
501 proc print data=pg1.class_birthdate;
502     format Height Weight 3. Birthdate date9.;
503 run;
504
505
506 *****
507 ** Common Formats for Numeric Values
508 *****
509 * Format Name **** Example Value *** Format Applied *** Formatted value;
510     w.d           12345.67           5.           123456
511     w.d           12345.67           8.1          12345.7
512     COMMAw.d      12345.67           COMMA8.1      12,345.7
513     DOLLARw.d     121345.67          DOLLAR10.2    $12,345.67
514     DOLLARw.d     121345.67          DOLLAR10.     $12,346
515     YENw.d        121345.67          YEN7.         Y12,346
516     EUROXw.d      121345.67          EUROX10.2     €12,346
517

```

```

518
519 *****
520 * Activity 3.05
521 *****
522
523 * Go to support.sas.com/documentation.
524     * 1. Look up the Zw.d format.
525     * 2. What does the format do?
526
527 Displays standard numeric data with leading zeroes.
528
529
530 Correct
531 Example: 1350 with the Z8. format applied would be displayed as 00001350
532
533 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
534
535 You can also find this information by following these steps:
536
537 Under Popular Documentation, select Programming: SAS 9.4 and Viya.
538 Under Syntax – Quick Links, select Formats under Language Elements.
539 Select the link for Zw.d.
540
541
542 *****
543 ** Common Formats for Date Values
544 *****
545
546 Value                Format applied          Formatted value
547
548 21199                DATE7.                15JAN18
549 21199                DATE9.                15JAN2018
550 21199                MMDDYY10.            01/15/2018
551 21199                DDMMYY8.              15/01/18
552 21199                MONYY7.               JAN2018
553 21199                MONNAME.              January
554 21199                WEEKDATE              Monday, January 15, 2018
555

```

```
556
557
558 *****
559 *** Demo: Formatting Data Values in Results
560 *****;
561
562 proc print data=pg1.strom_summary;
563 run;
564
565
566 * Now include format statemnt;
567
568 proc print data=pg1.strom_summary;
569     format Date mmddyy10. Cost dollar16. Deaths comma5.;
570 run;
571
572 * With Change;
573
574 proc print data=pg1.strom_summary;
575     format Date mmddyy8. Cost dollar14. Deaths comma5.;
576 run;
577
578 * mmddyy8. width 8---the largest number will not be formatted;
579 proc print data=pg1.strom_summary;
580     format Date mmddyy6. Cost dollar10. Deaths comma5.;
581 run;
582
583
584
585 *****;
586 * Activity 3.06 *;
587 * 1) Highlight the PROC PRINT step and run the *;
588 * selected code. Notice how the values of Lat, Lon, *;
589 * StartDate, and EndDate are displayed in the *;
590 * report. *;
591 * 2) Change the width of the DATE format to 7 and run *;
592 * the PROC PRINT step. How does the display of *;
593 * StartDate and EndDate change? *;
```



```
594 * 3) Change the width of the DATE format to 11 and run *;
595 * the PROC PRINT step. How does the display of *;
596 * StartDate and EndDate change? *;
597 * 4) Highlight the PROC FREQ step and run the selected *;
598 * code. Notice that the report includes the number *;
599 * of storms for each StartDate. *;
600 * 5) Add a FORMAT statement to apply the MONNAME. *;
601 * format to StartDate and run the PROC FREQ step. *;
602 * How many rows are in the report? *;
603 *****;
604
605 proc print data=pg1.storm_summary(obs=20);
606     format Lat Lon 4. StartDate EndDate date9.;
607 run;
608
609 proc freq data=pg1.storm_summary order=freq;
610     tables StartDate;
611     *Add a FORMAT statement;
612 run;
613
614
615 *****
616 *** Sorting Data
617 *****
618 * Sorting
619     * - improve visual arrangement of the data
620     * - identify and remove duplicate rows
621     * - prepare data for certain data processing steps;
622
623 PROC SORT
624 proc Sort data=input-table <out=output-table>;
625     by <descending> col-name(s);
626 run;
627
628 * <descending> --- overrides default ascending sort order;
629 * col-name(s) --- column(s) to sort by, or BY variables
630
631 * eg;
```

```
632
633     by Name TestScore;
634
635 * ascending order by Name, then within Name by ascending TestScore;
636     by Subject descending TestScore;
637
638 * ascending order by Subject, then within Subject by descending TestScore;
639
640
641
642 *****
643 * Activity 3.07
644 *****
645 * 1 Modify the OUT= option in the PROC SORT statement to create a temporary table named storm_sort.
646 * 2. Complete the WHERE and BY statements to answer the following question:
647 * Which storm in the North Atlantic Basin (NA or na) had the highest MaxWindMPH?;
648 * AN;    Allen
649
650 proc sort data=pg1.storm_summary out=storm_sort;
651     where Basin in("NA" "na");
652     by descending MaxWindMPH;
653 run;
654
655
656
657 *****
658 * Identifying and Removing Duplicates
659 *****;
660 proc sort data=input-table <out=output-table>
661     NODUPRECS <DUPOUT=output-table>;
662     BY_ALL_;
663 RUN;
664
665 * NODUPRECS <DUPOUT=output-table> -- remove all adjacent duplicates
666
667 * _ALL_ -- sort by entire rows
668
669 * Example;
```

```

670
671 proc sort data=pg1.class_test3
672     out=test_clean noduprecs dupout=test_dups;
673     by _all_;
674 run;
675
676 * NODUPKEY;
677
678 proc sort data=input-table <out=output-table>
679     NODUPKEY DUPOUT=output-table>;
680     BY <descending> col-name(s);
681 run;
682
683 * NODUPKEY -- keeps only first occurrence of each unique value
684
685 * Example;
686
687 proc sort data=pg1.class_test2
688     out=test_clean
689     dupout=test_dups
690     nodupkey;
691     by Name;
692 run;
693
694 *****
695 * Demo: Identifying and Removing Duplicate Values
696 *****
697
698 *****;
699 * Identifying and Removing Duplicate Values *;
700 *****;
701 * Syntax and Example *;
702 * *;
703 * Remove duplicate rows: *;
704 * PROC SORT DATA=input-table <OUT=output-table> *;
705 *     NODUPRECS <DUPOUT=output-table>; *;
706 *     BY _ALL_; *;
707 * RUN; *;

```

```

708 *                                     *;
709 *   Remove duplicate key values:       *;
710 *   PROC SORT DATA=input-table <OUT=output-table> *;
711 *       NODUPKEY <DUPOUT=output-table>; *;
712 *       BY <DESCENDING> col-name (s); *;
713 *   RUN;                               *;
714 *****;
715
716 *****;
717 *   Demo                               *;
718 *   1) Modify the first PROC SORT step to sort by all *;
719 *       columns and remove any duplicate rows. Write the *;
720 *       removed rows to a table named STORM_DUPS. *;
721 *       Highlight the step and run the selected code. *;
722 *       Confirm that there are 107,821 rows in *;
723 *       STORM_CLEAN and 214 rows in STORM_DUPS. *;
724 *   2) Run the second PROC SORT step and confirm that *;
725 *       the first row for each storm represents *;
726 *       the minimum value of Pressure. *;
727 *       Note: Because storm names can be reused in *;
728 *       multiple years and basins, unique storms *;
729 *       are grouped by sorting by Season, Basin, *;
730 *       and Name. *;
731 *   3) Modify the third PROC SORT step to sort the *;
732 *       MIN_PRESSURE table and keep the first row for *;
733 *       each storm. You do not need to keep the removed *;
734 *       duplicates. Highlight the step and run the *;
735 *       selected code. *;
736 *****;
737
738 *Step 1;
739 proc sort data=pg1.storm_detail out=storm_clean noduprecs dupout=storm_dups;
740     by _all_;
741 run;
742
743 *Step 2;
744 proc sort data=pg1.storm_detail out=min_pressure;
745     where Pressure is not missing and Name is not missing;

```

```
746         by descending Season Basin Name Pressure;
747     run;
748
749     *Step 3;
750     proc sort data=min_pressure nodupkey;
751         by descending Season Basin Name;
752     run;
753
754     *****
755     * Level 1 Practice: Sorting Data and Creating an Output Table
756     *****
757
758     *****;
759     * LESSON 3, PRACTICE 8 *;
760     * a) Modify the PROC SORT step to read PG1.NP_SUMMARY *;
761     * and create a temporary sorted table named *;
762     * NP_SORT. *;
763     * b) Add a BY statement to order the data by Reg and *;
764     * descending DayVisits. *;
765     * c) Add a WHERE statement to select Type equal to NP. *;
766     * Submit the program. *;
767     *****;
768
769     proc sort data=pg1.np_summary out=np_sort;
770         where Type="NP";
771         by Reg descending DayVisits;
772
773     run;
774
775     * AN 51;
776
777     proc sort data=pg1.np_summary out=np_sort;
778         by Reg descending DayVisits;
779         where Type="NP";
780     run;
781
782
783     *****
```

```
784 * Level 2 Practice: Sorting Data to Remove Duplicate Rows
785 *****;
786
787 * The pg1.np_largeparks table contains gross acreage for large national parks. There are duplicate rows for some location
788
789 * Reminder: If you restarted your SAS session, you must run the libname.sas program in the EPG194 folder.
790
791 * Open and review the pg1.np_largeparks table. Notice that there are exact duplicate rows for some parks.
792 * Open a new program and write a PROC SORT step that creates two tables (park_clean and park_dups), and removes the dupl
793 * Submit the program and view the output data.
794 * How many rows are included in the park_dups table?
795
796 * 30;
797
798
799 proc sort data=PG1.NP_LARGE PARKS out=park_clean noduprecs dupout=park_dups;
800     by _all_;
801 run;
802
803
804
805 * AN;
806
807 proc sort data=pg1.np_largeparks
808     out=park_clean
809     dupout=park_dups
810     noduprecs;
811     by _all_;
812 run;
813
814
815
816
817
818
819
820
821
```

822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857