Fundamentals of Programming in SAS®

A Case Studies Approach

James Blum
Jonathan Duggins

Student Solutions

sas

sas.com/books

This set of Solutions to Exercises is a companion piece to the following SAS Press book: Blum, James and Jonathan Duggins. 2019. *Fundamentals of Programming in SAS®: A Case Studies Approach*. Cary, NC: SAS Institute Inc.

Fundamentals of Programming in SAS®: A Case Studies Approach

978-1-64295-228-5 (Hardcover)
978-1-63526-672-6 (Paperback)
978-1-63526-671-9 (Web PDF)
978-1-63526-669-6 (epub)
978-1-63526-670-2 (kindle)

SAS Institute Inc., SAS Campus Drive, Cary, NC 27513-2414

July 2019

Student Solutions

Chapter 1

Concepts: Multiple Choice

2. B
4. C

Concepts: Short-Answer

2.
    a. Sometimes true. See Figure 1.7.1 for an example of an error message with valid syntax.
    b. Sometimes true. See Figure 1.4.8B for an example of a syntax error that results in a warning message.
    c. Sometimes true. See Figure 1.4.8A for a note that indicates a logic error.
    d. Never true. See the previous answer.
4.
    a. Never true. The invocation of a PROC or DATA step is seen as a step boundary for the prior step.
    b. Sometimes true. Though it is good programming practice to use explicit step boundaries at the end of each DATA or PROC step, the SAS windowing environment requires an explicit step boundary only after the last step (and statements other than RUN can serve this purpose), and SAS University Edition requires no explicit step boundaries.
    c. Always true. See Program 1.4.2 as an example; also, submitting sections of code in the SAS windowing environment requires a step boundary at the end of the selected section, which is guaranteed if all steps have explicit step boundaries.
    d. Always true. Global statements compile and take effect immediately and are not included as part of the compilation or execution of DATA or PROC steps.

Chapter 2

Concepts: Multiple Choice

2. C
4. D
6. C
8. B
10. D

Concepts: Short Answer

1.

a. Bins2 is an invalid name for a format – format names cannot end with a digit.
b. No semicolon is permitted between the format name ($codes) and the beginning of the format definition.
c. The Boolean AND cannot be followed by the comparison operator LE.

5. PROC COMPARE only compares variables with common names and types. It is important to determine if any additional variables were present in one or both data sets that PROC COMPARE did not validate. Also, if attribute differences are of concern, they are displayed in the PROC COMPARE output.

7.

a. Input Buffer

1	1	4	,	1	2	/	1	5	/	2	0	0	0	,	L	A	X	,	,	1	8	7

PDV

FlightNum	Date	Destination	FirstClass	EconClass
114	12/15/20	LAX	.	187

b. Input Buffer

4	3	9	,	1	2	/	1	1	/	2	0	0	0	,	L	A	X	,	2	0	,	1	3	7

PDV

FlightNum
439,12/1

Programming Basics

1.
```
proc means data=sashelp.cars min q1 median q3 max maxdec=1;
  class origin;
  var mpg_city;
run;

proc freq data=sashelp.cars;
  table type*origin/nocol nopercent;
  where type not in ('Truck','Hybrid');
run;
```

5.
```
proc sort data = sashelp.mdv out = sorted;
  by type code country;
run;

proc print data = sorted;
  by type code country;
    sumby code;
  id type code country;
  var;
  sum sales95 _4cast96;
  format sales95 _4cast96 dollar11.2;
run;
```

Chapter 3

Concepts: Multiple Choice

2. D
4. C
6. A
8. B
10. A

Concepts: Short-Answer

2.
    a. Family, Color, Size, Weight, Style
    b. Color, Transparency
    c. Pattern, Thickness, Color
5. The answers below assume all raw records contain complete information.
    a. The INPUT statement begins reading at the current location of the column pointer and uses the width provided with the format (or the default width, if none is provided) to define how many columns to read. It reads until the specified number of columns are read or until it encounters the end-of-line marker.
    b. Beginning with the current location of the column pointer, the INPUT statement scans for the next non-delimiter and advances the column pointer to that position. (If the current column is a non-delimiter, then the pointer is not advanced.) The INPUT statement reads until it encounters the next delimiter or reaches the end-of-line marker.
    c. Beginning with the current location of the column pointer, the INPUT statement scans for the next non-delimiter and advances the column pointer to that position. (If the current column is a non-delimiter, then the pointer is not advanced.) The INPUT statement reads until it encounters the next delimiter followed by a space or reaches the end-of-file marker.
6. *Partial Solution*
    f. Input Buffer

1	1	4	,	1	2	/	1	5	/	2	0	0	0	,	L	A	X	,	,	1	8	7

PDV

FlightNum	Date	Destination	FirstClass	EconClass
114	12/15/20	LAX	187	.

7. *Partial Solution*
    f. Input Buffer

1	1	4	,	1	2	/	1	5	/	2	0	0	0	,	L	A	X

PDV

FlightNum	Date	Destination	FirstClass	EconClass
114	12/15/20	LAX	.	.

8. No, TRUNCOVER and MISSOVER only differ in how they handle partial information on a variable in column or formatted input.

Programming Basics

1.

a. Patient X: is your ID 17
b. It provides a list of characters each of which is removed from the string in the first argument.
c. Patient X: is your ID# ?
d. It removes all digits from the string in the first argument.
e. 17
f. The default behavior of compress is to remove characters, either default or specified through arguments. In the third argument, K modifies the behavior to keep characters, in this case only digits as is also specified by D in the third argument.
g. #17
h. The first argument provides the target string, the second argument provides a list of characters, and the third argument can augment this list (such as adding all digits to the list) and/or modify how the list affects the target string (such as keeping all values in the list or removing all values in the list).
i. There are several answers. The brute force option of COMPRESS('Patient X: is your ID# 17?', 'PX:ID#17','k') is valid, but COMPRESS('Patient X: is your ID# 17?',,'LS'); is a more robust solution that would work for other patient/ID number combinations. While no explicit list of characters is included in the second argument, the L and S modifiers in the third argument add lowercase letters and spaces, respectively, to the list of characters for removal from the string in the first argument.

2.

a.
```
ods listing image_dpi = 300;
ods graphics / reset imagename = 'Ch3PB2a' width = 4in
               imagefmt = tiff;
proc sgplot data=sashelp.cars noautolegend noborder;
  hbar origin / response=mpg_city stat=mean
        dataskin=gloss fillattrs=(color=cx66FF66)
        limits=upper limitattrs=(color=blue);
  yaxis display=(nolabel);
  xaxis label='Avg. MPG, City';
run;
```

b.
```
ods listing image_dpi = 300;
ods graphics / reset imagename = 'Ch3PB2b' width = 4in
               imagefmt = tiff;
proc sgplot data=sashelp.cars noborder;
  hbar origin / response=mpg_city stat=mean
        group=type groupdisplay=cluster
        limits=upper limitattrs=(color=blue)
        limitstat=stderr dataskin=pressed;
  keylegend / position=bottomright location=inside
        across=1 noborder title='';
  yaxis display=(nolabel);
  xaxis label='Avg. MPG, City';
  where type not in ('Hybrid','Truck');
run;
```

6. *Partial Solution*

c.
```
data Formatted2;
  infile RawData('Ipums2005Formatted.txt');
  input      Serial              1-7
             State             $ 8-27
        @135 MortgagePayment :   dollar6.
        @155 HomeValue       :   dollar10.
             City              $ 33-72
```

```
                CityPop             comma6.
                Metro               79
                CountFIPS           80-82
  ;
  format MortgagePayment dollar6. HomeValue dollar10.
         CityPop comma6.;
run;
```

Chapter 4

Concepts: Multiple Choice

3. A
4. C
6. D
8. B
10. D

Concepts: Short-Answer

3. The conditions in the WHERE statement are applied to incoming records and may only use variables in the data set(s) being read. The subsetting IF applies during execution to all records coming through the PDV and may use any variables in the PDV regardless of origin— e.g. from the data set(s) being read, derived or computed in the current DATA step, automatic, etc.
5. *Partial Solution*
    a. Assistant—even though the condition is true for two WHEN conditions, the first one encountered is applied.
    c. 3—Much like a) the first true condition encountered determines the result.
7. *Partial Solution*
    a. Yes, the subsetting IF is always available for selecting records in the DATA step (and the condition is correct).
8. *Partial Solution*
    a. In the first data step, the implicit conversion of the value of X to store in Y is limited to 6 characters of length, so SAS represents the 9 digit value of X in scientific notation as 1.23E8 and converts. In the second data step, the implicit conversion still occurs, but is not subject to the same length restriction, so the character version of the 9 digit value of X is compared to the character representation in scientific notation and fails to match.

Programming Basics

1.

a.
```
data PB1A;
  set BookData.Ratings2016 BookData.Ratings2017;
run;
```

b.
```
proc sort data=BookData.Ratings2016 out=Ratings2016;
  by Str2;
run;

proc sort data=BookData.Ratings2017 out=Ratings2017;
  by Str2;
run;
```

```
data PB1B;
  set Ratings2016 Ratings2017;
  by Str2;
run;
```

c.
```
proc sort data=BookData.Ratings2016 out=Ratings2016B;
  by Str2 Str1;
run;
proc sort data=BookData.Ratings2017 out=Ratings2017B;
  by Str2 Str1;
run;
data PB1C;
  set Ratings2016B Ratings2017B;
  by Str2 Str1;
run;
```

2.
```
data ratings2016;
   set bookdata.ratings2016;
   stars=input(scan(str1,1,' /'),best12.);
run;
data ratings2017;
   set bookdata.ratings2017;
   stars=input(scan(str1,1,' /'),best12.);
run;
proc sort data=ratings2016;
   by descending stars id;
run;
proc sort data=ratings2017;
   by descending stars id;
run;
data final;
   set ratings2016(in=in2016) ratings2017;
   by descending stars id;
   if in2016 then year=2016;
     else year=2017;
run;
```

3. *Partial Solution*

a.
```
data stocks;
  set sashelp.stocks;
    select(month(date));
    when(10,11,12) FiscalSeason='Federal1';
    when(1,2,3) FiscalSeason='Federal2';
    when(4,5,6) FiscalSeason='Federal3';
    when(7,8,9) FiscalSeason='Federal4';
    otherwise FiscalSeason='';
  end;
  year=year(date);
run;
```

b.
```
proc means data=stocks min max;
  class year FiscalSeason stock;
  var close low high;
  ods output summary=hlstock;
  where year ge 2000;
run;
```

Chapter 5

Concepts: Multiple Choice

4. C
4. A
6. D
8. A
10. B
12. B
14. B

Concepts: Short-Answer

2. Because row and column information have been exchanged, the data set resulting from PROC TRANSPOSE has a fundamentally different structure than the incoming data set. E.g., in general, it does not have the same set of columns so replacement is not possible.
4. The CORR procedure is designed to produce measures of association for continuous variables while the FREQ procedure produces measures of association for categorical variables. Because user-defined formats are generally applied to a continuous variable to produce groups, such applications of a format are better suited to PROC FREQ than to PROC CORR and, in fact, the CORR Procedure ignores them for variables in the VAR or WITH statements.
5. *Partial Solution*
    a. PDV for each requested iteration are below:

_N_	CustomerID	Contact	QTR	Sales	Date
2	1	Fulanah AlFulaniyyah	2	23000	01APR2019
6	2	John Doe	2	10000	.

6. The full outer join of A with B contains exactly the rows of the union of the other three joins because the left antijoin contains rows in A but not B, the right antijoin contains rows in B but not A, and the inner join contains rows in both A and B.

Programming Basics

4.

a.
```
proc sgplot data = sashelp.stocks;
  scatter x = date y = high / group = stock;
run;
```

b.
```
proc sgplot data = stockTr;
  scatter x = date y = IBM;
  scatter x = date y = Intel;
  scatter x = date y = Microsoft;
run;
```

c. In terms of coding efficiency, the method used in part (a) clearly scales to larger scenarios better. The GROUP= option automatically creates the three separate scatter plots, and so does not require an update for a change in the number of unique values of Stock, while the second method requires each to be manually created.

3.
```
ods select PearsonCorr;
proc corr data = sashelp.baseball;
  var nhome nruns nrbi;
  with nhits nbb;
run;
```

6.
```
data OneToManyMM;
  merge cost(in=inCost) means(in=inMeans);
  by state mortgagestatus;
  if inCost eq 1 and inMeans eq 1;
```

```
  if not missing(homevalue) and not missing(hvmean)
    then HVdiff=homevalue-HVmean;
  if not missing(homevalue) and not missing(hvmean) and hvmean ne 0
    then HVratio=homevalue/HVmean;
  if not missing(hhincome) and not missing(HHImean)
    then HHIdiff=hhincome-HHImean;
  if not missing(hhincome) and not missing(HHImean) and hhimean ne 0
    then HHIratio=hhincome/HHImean;
  if not missing(mortgagepayment) and not missing(MPmean)
    then MPdiff=mortgagepayment-MPmean;
  if not missing(hhincome) and not missing(HHImean) and mpmean ne 0
    then MPratio=mortgagepayment/MPmean;
run;
```

Chapter 6

Concepts: Multiple Choice

2. D
4. D
6. C
8. D
10. C

Concepts: Short Answer

2. *Partial Solution*
    a. Conditional because the number of iterations is not known in advance.
3. *Partial Solution*
    a. No. Arrays cannot be used in a LABEL statement.
4. *Partial Solution*
    e. False, the SUMMARIZE option provides summaries on all analysis variables at the BREAK line; however, the BREAK statement is applied to the GROUP or ORDER variable, which can be either type (and is often character).
5. *Partial Solution*
    a.
```
do i = 1 to 4;
  diff(i) = value(i) - means(i);
  end;
keep diff:;
```

Programming Basics

1.
```
proc report data = sashelp.baseball;
  columns league division team natbat nhits nhome salary;
  define league / group;
  define division / group;
  define team / group;
  define natbat / analysis mean format = 5.1;
  define nhits / analysis mean format = 5.1;
  define nhome / analysis mean format = 5.1;
  define salary / analysis median format = 5.1;
  rbreak before / summarize;
  break after division / summarize;
run;
```

3.

a.
```
data CoinFlip01;
  call streaminit(5);
  *Used so the solutions generate a consistent answer;
  X = 0;
  do trial = 1 to 50;
    Outcome = rand('binomial',0.5,1);
    X = X + outcome;
    Estimate = X/Trial;
    output;
  end;
run;
```

b.
```
data CoinFlip02;
   call streaminit(5);
   *Used so the solutions generate a consistent answer;
   do TrueProp = 0.10, 0.25, 0.40;
      X = 0;
      do trial = 1 to 50;
         if trial eq 1 then call missing(X);
         Outcome = rand('binomial',TrueProp,1);
         X = X + outcome;
         Estimate = X/Trial;
         output;
      end;
   end;
run;
```

c.
```
proc sgplot data = CoinFlip02;
   series x=Trial y=Estimate / group=TrueProp markers
          markerattrs=(size=2mm symbol=circlefilled);
   refline TrueProp / axis = y lineattrs = (thickness = 2);
   yaxis grid label = 'Estimated Proportion of Tails';
   xaxis label = 'Flip #';
run;
```

7.

a.
```
proc sort data = sashelp.leutest(drop = x7129) out = leutest;
  by y;
run;

proc means data = leutest;
  class y;
  var x1 - x7128;
  output out=statz(where=(y ne .)) mean=mean1-mean7128;
run;

proc format;
  value leuk -1 = 'Myeloid'
              1 = 'Lymphoblastic'
  ;
run;

data leuk(drop = mean1-mean7128);
  merge leutest statz;
  by y;

  array orig[*] x1-x7128;
  array means[*] mean1-mean7128;
  array diffs[7128];
```

```
  do i = 1 to 7128;
    diffs[i] = orig[i] - means[i];
  end;

  format y leuk. x: diff: 6.3;
run;

proc report data = leuk;
  columns y x1 diffs1 x100 diffs100 x7128 diffs7128;
  define y / display 'Acute Leukemia Type';
  define x1 / display 'Gene 1';
  define diffs1 / display 'Gene 1 Deviation';
  define x100 / display 'Gene 100';
  define diffs100 / display 'Gene 100 Deviation';
  define x7128 / display 'Gene 7128';
  define diffs7128 / display 'Gene 7128 Deviation';
run;
```

b.
```
data leuk2(drop = mean1-mean7128);
  merge leutest statz;
  by y;

  array orig[9,24,33] x1-x7128;
  array means[9,24,33] mean1-mean7128;
  array diffs[9,24,33];

  do i = 1 to dim(orig,1);
    do j = 1 to dim(orig,2);
      do k = 1 to dim(orig,3);
        diffs[i,j,k] = orig[i,j,k] - means[i,j,k];
      end;
    end;
  end;

  format y leuk. x: diffs: 6.3;
run;

proc report data = leuk2;
  columns y x1 diffs1 x100 diffs100 x7128 diffs7128;
  define y / display 'Acute Leukemia Type';
  define x1 / display 'Gene 1';
  define diffs1 / display 'Gene 1 Deviation';
  define x100 / display 'Gene 100';
  define diffs100 / display 'Gene 100 Deviation';
  define x7128 / display 'Gene 7128';
  define diffs7128 / display 'Gene 7128 Deviation';
run;
```

c. The DO loop in part (a) uses simpler logic since the use of a one-dimensional array allows for easy traversal of the array elements with a single indexing variable. This is expected since the multidimensional array is typically associated with nested DO loops where each dimension of the array is associated with a specific DO loop. As a result, unless there is good reason to arrange the elements using multiple dimensions, the simplicity of using one-dimensional arrays is typically preferable.

Chapter 7

Concepts: Multiple Choice

3. A
4. D
6. B
8. A
10. B
12. B
14. B

Concepts: Short Answer

2.
    a. Yes, this is a record corresponding to the start of a new group for A that contains more than one record.
5. Both are generated and the text from the LINE statement is placed after the summary information from the BREAK statement.
6.
    a. *variable.stat-keyword*
    b. *alias*
    c. *_c#_*, where # is the column number (count includes columns not displayed)

Programming Basics

2.
```
data Vert(drop = utility1-utility4 cost1-cost4 i);
  set bookdata.Utility2005ComplexE;

  array ut[*] utility:;
  array co[*] cost:;

  do i = 1 to dim(ut);
    Utility = ut[i];
    Cost = co[i];
    if not missing(utility) and not missing(cost) then output;
  end;
run;

data UtilityE2005v1;
  infile rawdata('Utility2005ComplexE.txt') dsd missover;
  input serial Utility : $11. cost : dollar7. @;
  do while(not missing(cost));
      output;
      input Utility : $11. cost : dollar7. @;
  end;
run;

proc compare base = utilitye2005v1 compare = vert out = diff
             outbase outcomp outdiff outnoequal noprint
             method = absolute criteria = 1E-9;
run;
```

4.

```
libname Sales xlsx '--insert path-- \Sales.xlsx';

ods output position = compDesc;
proc contents data = sales.'First Sheet'n varnum;
run;

ods output position = baseDesc;
proc contents data = sashelp.prdsale varnum;
run;

proc compare base = baseDesc compare = compDesc out = diff
             outbase outcompare outdiff outnoequal noprint
             method = absolute criterion = 1E-12;
run;

libname sales clear;
```

Labels and formats differ in the two data sources, see data set produced by the PROC COMPARE for a full set of differences.

8.

```
proc report data = sashelp.baseball;
  columns league division team natbat nhits nhome
          salary fullSal HRcost;
  define league / group 'League';
  define division / group 'Division';
  define team / group 'Team';
  define natbat / sum format = comma10. 'At Bats';
  define nhits / sum format = comma10. 'Hits';
  define nhome / sum format = comma10. 'Home Runs';
  define salary / sum format = comma10. noprint;
  define fullSal / computed 'Salary' format=dollar15.;
  define HRCost / computed '$ per HR' format=dollar15.2;
  break after division / summarize suppress
            style=[fontweight=bold backgroundcolor=grayEE];

  compute fullSal;
    fullsal=salary.sum*1000;
  endcomp;
  compute HRcost;
    HRcost=fullSal/nhome.sum;
  endcomp;
run;
```