The Art of Defensive Programming: Coping with Unseen Data

Philip R Holland, Holland Numerics Limited, UK

HOLLAND NUMERICS LIMITED

Agenda

- Introduction
- Read First, Convert Later
- Testing for Missing Values
- IF and SELECT
- Missing and Invalid Data Warnings
- Formats
- Never Hard-code Values
- Check for Unique Key Variables Used in Merge
- Conclusions

Introduction

Consider the following data scenario:

- The input data set is defined in so far as the variable names and lengths are fixed, but the content of each variable is in some way uncertain.
- How do you write a SAS program that can cope appropriately with data uncertainty?

"*Code doesn't care who wrote it, but it does matter how well it was written!*" -- Gavin Winpenny (2017)

Read First, Convert Later - 1

- External files can contain a mixture of character and numeric values in the same data column.
- PROC IMPORT statements MIXED=YES for Excel files, or GUESSINGROWS= for text files, can help to avoid unexpected missing values, but can still cause problems when new data is read.
- Prepending a header text file can help to force character values **and** variable lengths. To perform the same type of action with Excel files will require exporting a sheet to a text file first.

Read First, Convert Later - 2

```
PROC IMPORT FILE = 'a.txt' OUT = work.a DBMS = CSV REPLACE;
  GUESSINGROWS = 3;
  GETNAMES = NO;
RUN;
```

a.txt:

```
1, ABC, 123, DEFGHI, 456789
2, DEF, 456, JKLMNO, 789012
3, GHI, 789, PQRSTU, 345678
```

#	Variable	Type	Len	Format	Informat
1	VAR1	Num	8	BEST12.	BEST32.
2	VAR2	Char	3	$3.	$3.
3	VAR3	Num	8	BEST12.	BEST32.
4	VAR4	Char	6	$6.	$6.
5	VAR5	Num	8	BEST12.	BEST32.

work.a:

Obs	VAR1	VAR2	VAR3	VAR4	VAR5
1	1	ABC	123	DEFGHI	456789
2	2	DEF	456	JKLMNO	789012
3	3	GHI	789	PQRSTU	345678

Read First, Convert Later - 3

```
PROC IMPORT FILE = 'b.txt' OUT = work.b DBMS = CSV REPLACE;
  GUESSINGROWS = 3;
  GETNAMES = NO;
RUN;
```

b.txt:

```
1, ABC, 123, DEFGHI, 456789
2, DEF, 456, JKLMNO, 789012
3, GHI, 789, PQRSTU, 345678
4, JKLM, 2AB, VWXYZ01, 345678A
```

#	Variable	Type	Len	Format	Informat
1	VAR1	Num	8	BEST12.	BEST32.
2	VAR2	Char	3	$3.	$3.
3	VAR3	Num	8	BEST12.	BEST32.
4	VAR4	Char	6	$6.	$6.
5	VAR5	Num	8	BEST12.	BEST32.

work.b:

Obs	VAR1	VAR2	VAR3	VAR4	VAR5
1	1	ABC	123	DEFGHI	456789
2	2	DEF	456	JKLMNO	789012
3	3	GHI	789	PQRSTU	345678
4	4	JKL	.	VWXYZ0	.

Read First, Convert Later - 4

```
PROC IMPORT FILE = '*.txt' OUT = work.all DBMS = CSV REPLACE;
  GUESSINGROWS = 3;
  GETNAMES = NO;
RUN;
```

##header.txt:

```
ID, 3LETTERS, 3NUMBERS, 6LETTERS, 6NUMBERS
```

#	Variable	Type	Len	Format	Informat
1	VAR1	Char	2	$2.	$2.
2	VAR2	Char	8	$8.	$8.
3	VAR3	Char	8	$8.	$8.
4	VAR4	Char	8	$8.	$8.
5	VAR5	Char	8	$8.	$8.

work.all:

Obs	VAR1	VAR2	VAR3	VAR4	VAR5
1	ID	3LETTERS	3NUMBERS	6LETTERS	6NUMBERS
2	1	ABC	123	DEFGHI	456789
3	2	DEF	456	JKLMNO	789012
4	3	GHI	789	PQRSTU	345678
5	1	ABC	123	DEFGHI	456789
6	2	DEF	456	JKLMNO	789012
7	3	GHI	789	PQRSTU	345678
8	4	JKLM	2AB	VWXYZ01	345678A

Testing for Missing Values

```
DATA _NULL_;
  a = 49;
  b = .;
  c = 'text';
  d = ' ';
  e = '99';
  miss_a = MISSING(a);
  miss_b = MISSING(b);
  miss_c = MISSING(c);
  miss_d = MISSING(d);
  nmiss_atoe = NMISS(a, b, c, d, e);
  cmiss_atoe = CMISS(a, b, c, d, e);
  PUT a= b= c= d= e=;
  PUT miss_a= miss_b= miss_c= miss_d=;
  PUT nmiss_atoe= cmiss_atoe=;
RUN;
```

```
a=49 b=. c=text d= e=99
miss_a=0 miss_b=1 miss_c=0 miss_d=1
nmiss_atoe=3 cmiss_atoe=2

NOTE: Invalid numeric data, c='text'
```

IF and SELECT - 1

```
DATA testing1 (KEEP = name group);
  SET sashelp.class;
  IF name =: "A" THEN group = 1;
  ELSE IF name =: "J" THEN group = 2;
  ELSE IF name =: "P" THEN group = 3;
  ELSE IF name =: "R" THEN group = 4;
RUN;

DATA testing2 (KEEP = name group);
  SET sashelp.class;
  SELECT;
    WHEN (name =: "A") group = 1;
    WHEN (name =: "J") group = 2;
    WHEN (name =: "P") group = 3;
    WHEN (name =: "R") group = 4;
    OTHERWISE PUT 'WAR' 'NING: ' name
                          'has a missing group';
  END;
RUN;
```

```
PROC SQL;
  CREATE TABLE testing3 AS
    SELECT name
        ,(CASE
          WHEN SUBSTR(name, 1, 1) = "A" THEN 1
          WHEN SUBSTR(name, 1, 1) = "J" THEN 2
          WHEN SUBSTR(name, 1, 1) = "P" THEN 3
          WHEN SUBSTR(name, 1, 1) = "R" THEN 4
          ELSE .
          END) AS group
    FROM sashelp.class
  ;
QUIT;
```

IF and SELECT - 2

testing2.log:

```
WARNING: Barbara has a missing group
WARNING: Carol has a missing group
WARNING: Henry has a missing group
WARNING: Louise has a missing group
WARNING: Mary has a missing group
WARNING: Thomas has a missing group
WARNING: William has a missing group
```

Obs	Name	group
1	Alfred	1
2	Alice	1
3	Barbara	.
4	Carol	.
5	Henry	.
6	James	2
7	Jane	2
8	Janet	2
9	Jeffrey	2
10	John	2
11	Joyce	2
12	Judy	2
13	Louise	.
14	Mary	.
15	Philip	3
16	Robert	4
17	Ronald	4
18	Thomas	.
19	William	.

Missing and Invalid Data Warnings

```
DATA range1 (KEEP = name age agegroup);
  SET sashelp.class;
  SELECT;
    WHEN (age = .) PUT 'ER' 'ROR: ' name 'has a missing age';
    WHEN (age < 12) PUT 'WAR' 'NING: ' name 'is younger than 12';
    WHEN (12 <= age < 13) agegroup = '12';
    WHEN (13 <= age < 14) agegroup = '13';
    WHEN (14 <= age < 15) agegroup = '14';
    WHEN (15 <= age < 16) agegroup = '15';
    OTHERWISE PUT 'WAR' 'NING: ' name 'is older than 15';
  END;
RUN;
```

range1.log:

```
WARNING: Joyce is younger than 12
WARNING: Philip is older than 15
WARNING: Thomas is younger than 12
```

Obs	Name	Age	agegroup
1	Alfred	14	14
2	Alice	13	13
3	Barbara	13	13
4	Carol	14	14
5	Henry	14	14
6	James	12	12
7	Jane	12	12
8	Janet	15	15
9	Jeffrey	13	13
10	John	12	12
11	Joyce	11	
12	Judy	14	14
13	Louise	12	12
14	Mary	15	15
15	Philip	16	
16	Robert	12	12
17	Ronald	15	15
18	Thomas	11	
19	William	15	15

Formats

```
PROC FORMAT;
  VALUE testage
    12 = '12'
    13 = '13'
    14 = '14'
    15 = '15'
    OTHER = "value not in format testage"
    ;
RUN;

DATA range2 (KEEP = name age agegroup);
  SET sashelp.class;
  LENGTH agegroup $50;
  agegroup = PUT(age, testage.);
RUN;
```

Obs	Name	Age	agegroup
1	Alfred	14	14
2	Alice	13	13
3	Barbara	13	13
4	Carol	14	14
5	Henry	14	14
6	James	12	12
7	Jane	12	12
8	Janet	15	15
9	Jeffrey	13	13
10	John	12	12
11	Joyce	11	value not in format testage
12	Judy	14	14
13	Louise	12	12
14	Mary	15	15
15	Philip	16	value not in format testage
16	Robert	12	12
17	Ronald	15	15
18	Thomas	11	value not in format testage
19	William	15	15

Never Hard-code Values

Hard-coding values will create a significant program maintenance and reuse issue, and should be strongly discouraged.

Alternatives to hard-coding, where a single source of information can be maintained and shared with many programs, include:

- Formats and informats
- Look-up tables
- Macro libraries

Check for Unique Key Variables Used in Merge

```
PROC SQL;
  SELECT sex
        ,age
        ,SUBSTR(name, 1, 1) AS first_letter
        ,COUNT(*) AS count
  FROM   sashelp.class
  GROUP BY
         sex
        ,age
        ,CALCULATED first_letter
  HAVING COUNT(*) > 1
  ;
QUIT;

%PUT Duplicate count = &sqlobs.;
```

Sex	Age	first_letter	count
M	12	J	2

```
Duplicate count = 1
```

Conclusions

- When reading external data always read first into character variables, then convert into related numeric variables to avoid losing data.
- Test for any missing values with CMISS(), so you don't need to know whether the variables are character or numeric.
- Use a SELECT statements in Data Steps and CASE clauses in PROC SQL, because they both force you to code a fallback alternative when testing the data.
- Use lookup data sets or (in)formats instead of hard-coding values, so there is a common source of values for all related programs.
- If data sets are being merged by unique key variables, then test each data set for unique combinations of these keys variable first.

Contact Details

Philip R Holland, SAS Consultant

Holland Numerics Ltd

94 Green Drift, Royston, Herts SG8 5BT, UK

tel: +44-7714-279085

email: phil@hollandnumerics.org.uk

web: blog.hollandnumerics.org.uk