Representation of Data Created with Proprietary Missing Value Schemes

Larry Hoyle, IPSR, University of Kansas

Most current data manipulation and analysis software packages have at least some way of representing missing values, but each of them may have different ways of doing so. This makes representing data in a non-proprietary format compatible across packages quite complicated.

- Some (like SPSS) tag otherwise valid values or ranges of values as missing.
- Others (like R, SAS and Stata) use special values for missing which are not ever within the set of valid values
  - For some software packages (like R) any operation involving a missing value produces a missing value
  - For others (like SAS and Stata) missing values are ignored for numerical calculations but have meaning in logical comparisons (e.g. .a<-9999 is true in SAS and .a>9999 is true in Stata)
  - Some may have multiple built-in missing values with special meanings (R has NA, NaN, -Inf, and Inf)
- Some allow missing values to be labeled by users (e.g. SPSS, SAS, Stata)
- Others do not allow missing values to be labeled (R does, however, have implicit meanings for its 4 types)
- R has special values for positive and negative infinity Inf and –Inf compatible with the *float* and *double* XML Schema: Datatypes used by DDI.

In the example at the right the variable "i" has the value Inf (infinite). Inf is a valid value in numeric calculations and in comparisons: 0 divided by Inf is 0, Inf/2 is Inf, and 1000000 is less than Inf as shown here. The value Inf then, is not missing but is not a normal value.

```
> I <- 1/0
> i
[1] Inf
> 0 / i
[1] 0
> 1000000 < i
[1] TRUE
> i/2
[1] Inf

> nan <- 0/0
> nan
[1] NaN
> nan < 10
[1] NA
> nan/2
[1] NaN

> n <- NA
> 0 / n
[1] NA
```

```
1   data _null_;
2   i = 1 / 0;
NOTE: Division by zero detected
3   put i=;
4   j = 0 / i;
5   put j=;
6   run;
i=.
j=.     . is the result of 0 over missing
```

In many other packages division by 0 produces a missing value as seen in the SAS code to the right. Excel has special missing codes for different invalid operations log(-1)= #NUM!; 1/0 = #DIV/0!; #DIV/0! / 2 = #DIV/0!; 0 /#DIV/0! = #DIV/0!.

```
> v<-c(1,Inf)
> summary(v)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      1     Inf     Inf     Inf     Inf     Inf
```

The R value NaN (not a number) is produced when a numeric result of a calculation is not defined. NaN is a type of missing value. R also has a value that corresponds to the system missing in other packages: NA.

With all of these differences, how can data produced with one scheme be best represented in a non-proprietary format conveying the maximum of the original information?

The following SAS program creates three variables, two of which share the same format. In addition to the numerical values of the three variables the dataset contains missing codes which represent catagories (concepts). The code .d represents “Don’t Know” and the code .i represents an intentionally missing value. In addition, variables `MissingAsSASStata1` and `MissingAsSASStata3` have missing represented by the same code scheme.

```
Proc format cntlout=alt.control;
/* two categorizations of missing */
value UI
.u = "unintentional"
.i = "intententional"
;
value DR
.d = "Don't Know"
.r = "Refused"
;
run;

data Alt.testdata;
input MissingAsSASStata1 MissingAsSASStata2 MissingAsSASStata3;
format MissingAsSASStata1 MissingAsSASStata3 DR.;
format MissingAsSASStata2 UI.;
datalines;
1 1 1
2 .u .d
3 2 .r
.d 3 2
4 .i 3
5 4 4
.r 5 5
6 6 6
;
run;
```

	MissingAsSASStata1	MissingAsSASStata2	MissingAsSASStata3
1	1	1	1
2	2	U	D
3	3	2	R
4	D	3	2
5	4	I	3
6	5	4	4
7	R	5	5
8	6	6	6

Figure 1 SAS Data Unformatted – (note: a letter like “D” is an internal missing value .D)

	MissingAsSASStata1	MissingAsSASStata2	MissingAsSASStata3
1	1	1	1
2	2	unintentional	Don't Know
3	3	2	Refused
4	Don't Know	3	2
5	4	intententional	3
6	5	4	4
7	Refused	5	5
8	6	6	6

Figure 2 SAS Data Formatted

The following R code:

- reads the variables as character,
- computes a numeric column where missing is NA, and
- creates an additional column indicating the category of missing.

Note that the code schemes and associated category schemes are represented as separate list objects. In this way R can create reusable categorizations, although they are not used by reference as in SAS and Stata.

```
# Missing Values Most Compatible With SAS and Stata

# CodeList from SAS Format DR
CodeSchemeListDR <- list(codes=c(".D", ".R"), categories=c("Don't Know","Refused"))
# CodeList from SAS Format UI
CodeSchemeListUI <- list(codes=c(".I", ".U"), categories=c("intentional","unintentional"))

FromSASStata <- read.table("SASStata3.csv", header=TRUE,
sep=",",quote="",row.names=NULL,as.is=TRUE)             # reads the data as character

FromSASStata$MissingAsSASStata1Valid <- as.numeric(FromSASStata$MissingAsSASStata1)
FromSASStata$WhyMissing1 <-
factor(ifelse(is.finite(FromSASStata$MissingAsSASStata1),0,FromSASStata$MissingAsSASStata1 ),
                                  levels=CodeSchemeListDR$codes, CodeSchemeListDR$categories )

FromSASStata$MissingAsSASStata2Valid <- as.numeric(FromSASStata$MissingAsSASStata2)
FromSASStata$WhyMissing2 <-
factor(ifelse(is.finite(FromSASStata$MissingAsSASStata2),0,FromSASStata$MissingAsSASStata2 ),
                                  levels=CodeSchemeListUI$codes, CodeSchemeListUI$categories )

FromSASStata$MissingAsSASStata3Valid <- as.numeric(FromSASStata$MissingAsSASStata3)
FromSASStata$WhyMissing3 <-
factor(ifelse(is.finite(FromSASStata$MissingAsSASStata3),0,FromSASStata$MissingAsSASStata3 ),
                                  levels=CodeSchemeListDR$codes, CodeSchemeListDR$categories )
```

```
> FromSASStata
  MissingAsSASStata1 MissingAsSASStata2 MissingAsSASStata3 MissingAsSASStata1Valid WhyMissing1 MissingAsSASStata2Valid   WhyMissing2 MissingAsSASStata3Valid WhyMissing3
1                  1                  1                  1                      1       <NA>                      1          <NA>                      1        <NA>
2                  2                 .U                 .D                      2       <NA>                     NA unintentional                     NA  Don't Know
3                  3                  2                 .R                      3       <NA>                      2          <NA>                     NA     Refused
4                 .D                  3                  2                     NA  Don't Know                      3          <NA>                      2        <NA>
5                  4                 .I                  3                      4       <NA>                     NA   intentional                      3        <NA>
6                  5                  4                  4                      5       <NA>                      4          <NA>                      4        <NA>
7                 .R                  5                  5                     NA     Refused                      5          <NA>                      5        <NA>
8                  6                  6                  6                      6       <NA>                      6          <NA>                      6        <NA>
```

Figure 3 Imported SAS Data with Additional Variables

DDI3.2 could describe this by defining 3 VariableGroups
(Group1 would be: MissingAsSASStata1, MissingAsSASStata1Valid, WhyMissing1)

MissingAsSASStata1Valid would have a SourceVariableReference to MissingAsSASStata1 and its VariableRepresentation would have a ProcessingInstructionReference to a GenerationInstruction describing how to transform from the text representation to a variable with system missing values (or named missing values). The GenerationInstruction could have Command code for multiple languages describing how to process the missing values.

WhyMissing1 would have a SourceVariableReference to MissingAsSASStata1 and its VariableRepresentation would have a ProcessingInstructionReference to a GenerationInstruction describing how to transform from the text representation to a variable labels for the missing values. The GenerationInstruction could have Command code for multiple languages describing how to process the missing values.

Other Missing Value Representations

There are multiple other ways missing values are represented in other software packages and non-proprietary files. The examples below describe some of them.

Missing data represented as single characters or character strings

MissingAsChars1	MissingAsChars2
1	1
2	u
3	2
d	3
4	i
5	4
r	5
6	6

This form is quite common in non-proprietary data representations like CSV files, where there is special meaning to different missing values. The letters may also represent numeric categories though like, for instance, top coded values, or where exact values are restricted by confidentiality.

Reading in R

Letters make the column default to character (then factor)

```
> AsChars <- read.table("Letters.csv", header=TRUE, sep=",",row.names=NULL)
> AsChars
  MissingAsChars1 MissingAsChars2
1               1               1
2               2               u
3               3               2
4               d               3
5               4               i
6               5               4
7               r               5
8               6               6
> is(AsChars$MissingAsChars1)
[1] "factor"   "integer"  "oldClass" "numeric"  "vector"
> is(AsChars2$MissingAsChars2)
[1] "factor"   "integer"  "oldClass" "numeric"  "vector"
```

Na.strings specifies letters as missing codes

```
> AsChars2 <- read.table("Letters.csv", header=TRUE, sep=",",row.names=NULL,
na.strings=c("d","r","u","i"))
> AsChars2
  MissingAsChars1 MissingAsChars2
1               1               1
2               2              NA
3               3               2
4              NA               3
5               4              NA
6               5               4
7              NA               5
8               6               6
> is(AsChars2$MissingAsChars1)
[1] "integer"             "numeric"             "vector"
[4] "data.frameRowLabels"
> is(AsChars2$MissingAsChars2)
[1] "integer"             "numeric"             "vector"
[4] "data.frameRowLabels"
```

Missing Data represented by values just Out of Valid Range

MissingAsOutOfRange1	MissingAsOutOfRange2
1	1
2	-3
3	2
-1	3
4	-4
5	4
-2	5
6	6

This is a risky approach. If statistics are computed treating the missing values as valid the results may not appear obviously wrong.

Missing Data represented by empty values (successive delimiters)

MissingAsEmpty1	WhyMissing1	MissingAsEmpty2	WhyMissing2
1		1	
2			u
3		2	
	d	3	
4			i
5		4	
	r	5	
6		6	

```
MissingAsEmpty1,WhyMissing1,MissingAsEmpty2,WhyM
issing2
1,,1,
2,,,u
3,,2,
,d,3,
4,,,i
5,,4,
,r,5,
6,,6,
```

Missing

```
> FromEmpty <- read.table("Empty.csv", header=TRUE, sep=",",row.names=NULL,as.is=TRUE )
> FromEmpty$WhyMissing1F <- factor(FromEmpty$WhyMissing1, levels=c("","d","r"),
labels=c("valid","Don't Know", "Refused"))
> FromEmpty$WhyMissing2F <- factor(FromEmpty$WhyMissing2, levels=c("","i","u"),
labels=c("valid","intended", "unintended"))
> FromEmpty
  MissingAsEmpty1 WhyMissing1 MissingAsEmpty2 WhyMissing2 WhyMissing1F WhyMissing2F
1               1                           1                    valid        valid
2               2                          NA           u        valid   unintended
3               3                           2                    valid        valid
4              NA           d               3               Don't Know        valid
5               4                          NA           i        valid     intended
6               5                           4                    valid        valid
7              NA           r               5                  Refused        valid
8               6                           6                    valid        valid
> is(FromEmpty$MissingAsEmpty1)
[1] "integer"             "numeric"             "vector"              "data.frameRowLabels"
> is(FromEmpty$WhyMissing1)
[1] "character"           "vector"              "data.frameRowLabels" "SuperClassMethod"
```

Missing Values Represented by Values Way out of Valid Range

MissingWayOut1	MissingWayOut2
1	1
2	-666666
3	2
-999999	3
4	-777777
5	4
-888888	5
6	6

At least with this approach statistics computed treating the missing values as valid will appear obviously wrong.

```
> # Missing Values Represented by Values Way out of Valid Range
> FromWayOut <- read.table("WayOut.csv", header=TRUE, sep=",",row.names=NULL, )
> FromWayOut
  MissingWayOut1 MissingWayOut2
1              1              1
2              2        -666666
3              3              2
4        -999999              3
5              4        -777777
6              5              4
7        -888888              5
8              6              6
> is(FromWayOut$MissingAsWayOut1)
[1] "NULL"             "OptionalFunction" "optionalMethod"
> summary(FromWayOut)
 MissingWayOut1       MissingWayOut2
 Min.   : -999999.0   Min.   :-777777.0
 1st Qu.: -222221.2   1st Qu.:-166665.8
 Median :       2.5   Median :      2.5
 Mean   : -236108.2   Mean   :-180552.8
 3rd Qu.:       4.2   3rd Qu.:      4.2
 Max.   :       6.0   Max.   :      6.0
> FromWayOut2 <- FromWayOut
> FromWayOut2$WhyMissing1 <- factor(ifelse(FromWayOut2$MissingWayOut1 < -
600000,FromWayOut2$MissingWayOut1,0),levels=c(0,-888888,-999999), labels=c("valid","Don't Know",
"Refused"))
> FromWayOut2$MissingWayOut1 <- ifelse(FromWayOut2$MissingWayOut1 < -
600000,NA,FromWayOut2$MissingWayOut1)
> FromWayOut2$WhyMissing2 <- factor(ifelse(FromWayOut2$MissingWayOut2 < -
600000,FromWayOut2$MissingWayOut2,0),levels=c(0,-666666,-777777), labels=c("valid","unintended",
"intended"))
> FromWayOut2$MissingWayOut2 <- ifelse(FromWayOut2$MissingWayOut2 < -
600000,NA,FromWayOut2$MissingWayOut2)
>
> FromWayOut2
  MissingWayOut1 MissingWayOut2 WhyMissing1 WhyMissing2
1              1              1       valid       valid
2              2             NA       valid  unintended
3              3              2       valid       valid
4             NA              3     Refused       valid
5              4             NA       valid    intended
6              5              4       valid       valid
7             NA              5  Don't Know       valid
8              6              6       valid       valid
```

Missing Values Most Compatible With SAS and Stata

MissingAsSASStata1	MissingAsSASStata2
1	1
2	.u
3	2
.d	3
4	.i
5	4
.r	5
6	6

The point-and-click tools provided by SAS are inconsistent here. The SAs export to CSV tool writes the missing values without the leading ".". The import tool will see something like ".d" as a special missing value and would import the table above as numeric variables with special missing values. Without the leading dots, the variables would be imported as text by default. If the import type was forced to numeric the missing values in the table above would be read as system missing and their special meanings would be lost. To preserve these the "Missing" command would need to be used to list the special missing values included in the data.

Missing Values as Empty Field With Secondary Variable

MissingAsEmpty2	WhyMissing1	MissingAsEmpty2	WhyMissing2
1		1	
2			u
3		2	
	d	3	
4			i
5		4	
	r	5	
6		6	

This form is perhaps less directly readable by packages which have the capability of coding named missing values, but does have the advantage of clearly showing that there are missing values in the numeric variables.

Missing Values As R NA with Secondary Variable

MissingAsNA1	WhyMissing1	MissingAsNA2	WhyMissing2
1		1	
2		NA	u
3		2	
NA	d	3	
4		NA	i
5		4	
NA	r	5	
6		6	