A02 - 2008

Data Set Investigator - Automated Exception Reporting from an electronic data dictionary with %DSI().

Authors:
Matthew T. Karafa, PhD (Paper and Main DSI macros)
Julie Thornton, MS (wheremissing, rangecheckcat rangecheckcont macros)

Abstract:

Data cleaning and data sleuthing can be the most tedious and time consuming part of any analysis. As part of the data elicitation process we often collect metadata about valid values, and other data rules that can be used to quickly check for such problems. By using a small amount of this metadata about the data set\u2019s variables the provided macro, %DSI(), can produce a data exception report both by rule and by record ID which can be quickly turned back to the client for data correction. %DSI() takes a comma separated data file defining data "rules" with fields including variable name, type, valid values and ranges. These are then applied to the data set using internal macros that report the records and values that violate the given set of rules. These are then organized into a fairly simple, MSWord compliant HTML document, which can be returned to the client for action.

Introduction

In a typical data analysis setting, one of the largest time consuming chores that must be dealt with is ensuring data accuracy and data integrity. Typically this involves a statistical programmer poring over data values and allowable values in each of the fields, sending information back and forth to the investigator and creating ad hoc reports to illustrate the problems and come to an acceptable solution.

Define the Rules "Metadata"

This first step can easily be done as part of the initial meeting with the client. As the types of data are discussed, valid ranges and an initial data dictionary naturally flow as part of the conversation. For example suppose you have a data set that will gather a gender, age, race, some hospital dates, and blood pressure information. The initial discussion with the investigator reveals that ages should be between 18 and 35 years; acceptable codes for gender are M and F; acceptable codes for race are W,B, H, and O; and that all of the surgeries need to have happened in either 2005 or 2006. Further, the investigator would like to look at the subset of males that are under age 21 in a separate listing. Thus we can create a data dictionary that looks something like:

Table 1: Example of Data Dictionary “Metadata” for %DSI()

Varname	VarType	VarFormat	AllowMissing	Min	Max	ValidValueList	Special	RuleCode	RuleText
Name	Type	Format	Missing OK	Min val	Max Val	Valid Values	Special	Rule Num	Rule Description
Id	i	ccfid	n					1	Valid CCFID
admit_dt	d	date	n	1/1/05	31Dec06			2	Check Valid Date
dcg_dt	d	date	n	1/1/05	31Dec06			3	Check Valid Date
Gender	c	cat	y			M F m f		4	Gender Male or Female
Race	c	cat	y			W B H O w b h o		5	Race White Black Hisp.Other
Age	n	cont	y	18	35			6	Age between 12 and 35
Dob	d	date	n	1/1/1970	1/1/1988			7	Check Valid DOB
Sbp	n	cont	n	50	400			8	Systolic between 50 and 400
Dbp	n	cont	n	25	200			9	Diastolic between 25 and 200
Age	r	spec					gt 21	10	All Men over 21
Gender	r	spec					eq "M"	10	All Men over 21
Age	r	spec					gt 25	11	All women over 25
gender	r	spec					eq "F"	11	All women over 25

Note that for all variables we provide a variable name, type, missing status a rule code number and some rule code text. This text is important, as the macro’s reports will use this for to tag to the report for those who violate a given rule. For variables of type cat [categorical], we define valid values in a space-delimited list. For dates and continuous values we give the minimum and maximum values. For our two level rule we define it as type “S,” and give the special conditions which are basically the right hand side of a comparison test. Further, since the routine to read in the rules dataset from csv comes from a modified version of Cram and Whitlock’s (REF) xls2sas macro, the first line is the SAS variable names, and line 2 is the corresponding SAS labels.

Sample Macro call and parameters

To call %DSI() we make a call that looks like the following:

```
%DSI( DS= MyData, Rules= “C:\data\datadict.csv”, RulesDlm=%str(','),
      ListingFile=T, HtmlFile=T, owrite_html=F, ofile_html=./Temp.html,
      PageBreakAtSplit=F, AppendToPrevRuns=F, ReportTitle=, DEBUG = 0);
```

The only two required parameters are DS and Rules. DS is the SAS data set you want to generate the report on and Rules is the path to the data dictionary dataset mentioned in the prior section. RulesDlm is an optional parameter to configure the dictionary data’s delimiter. Listingfile is a toggle to send information to the SAS list file or output window. HTMLFile is a toggle to create an MSWord compliant HTML report of the results. The parameters owrite_html and ofile_html tell the macro to overwrite the word doc or append to it and provide the HTML file’s name. PageBreakAtSplit puts each ID on its own page, and AppendtoPrevRuns allows one to use the same report for multiple calls of DSI. ReportTitle lets you pass your own title to the %DSI() report and DEBUG turns on several internal logging techniques used in the development of %DSI().

Check Values

Once called the macro reads in the dictionary data, and puts each field into some delimited lists to be used like a queue in conjunction with %scan() to “pop” the information off as needed. Then several sub-macros are called to check the given values against missing status and for valid values. Each of these macros return a dataset with the IDvars passed to %DSI(), the

name of the problem variable, the value of the problem variable, the rule code that it violated and the text description of that rule. If no errors are found, no data set is returned and %DSI() knows that there are no problems of that type.

Output “By ID”

Once the individual error datasets are made, they are all concatenated together and sorted by the IDvar set. This new data set is then passed to an HTML coding routine which generates the MSWOD file. Below is an exerpt of a sample of 3 id's based on our earlier data dictionary:

Table 2: Sample of 3 ID's with problems from %DSI()

11499473

Problem	Variable Involved	Value of Involved Variable
6 : Age between 18 and 35	age	55
2 : Check Valid Date	admit_dt	17DEC2001
3 : Check Valid Date	dcg_dt	26DEC2001
7 : Check Valid DOB	dob	25OCT1946

16107018

Problem	Variable Involved	Value of Involved Variable
0 : Missing Values	dbp	.
5 : Race White Black Hispanic Other	race	K
6 : Age between 12 and 35	age	63
2 : Check Valid Date	admit_dt	15AUG2001
3 : Check Valid Date	dcg_dt	24AUG2001
7 : Check Valid DOB	dob	13MAR1961

16133018

Problem	Variable Involved	Value of Involved Variable
1.1 : CCFID Invalid	id	16133018
0 : Missing Values	dob	.
0 : Missing Values	sbp	.
0 : Missing Values	dbp	.

Conclusion

By evaluating the metadata and using it to write SAS code for you, a nice report can be generated using very high level language rather than coding a series of if-then-else checks.