Paper AD11

Exceptional Exception Reports

Gary McQuown
Data and Analytic Solutions Inc.
http://www.dasconsultants.com

Introduction

This paper presents an overview of exception reports for data quality control and describes how they can be assembled from pieces and parts of fairly common SAS® code. Topics include exceptional exception reports vs. simple exception reports, the reasons to use exception reports as frequently as possible, the application of logic in the reports and issues that may cause problems. An example of the general use exception report can be downloaded and used as is or modified to meet your requirements.

Exception Reports

Exception Reports document how well your data meets your pre-defined business rules. Knowing what you have in terms of data quality is essential to understanding the usability of the data and the validity of any processes performed with that data. Typically an exception report presents statistics describing non-conforming data, lists the non-conforming data or both. This information may then be used to guide to clean the data, manage how the data is used or to explain anomalies in downstream processes. While the simplest exception reports only call attention to anomalies in the data, exceptional exception reports provide detailed documentation and specific information with little additional effort from the user.

Why Exception Reports Are Important

The importance and necessity of exception reports has increased due to several trends. Historically the quantity of data being collected had increased dramatically and is now increasing exponentially. Simultaneously and for related reasons, the quality of the data collected has decreased to a slightly lesser degree. The most obvious reasons for these trends are that data is now collected automatically through many different processes, for different reasons and with little thought to its uses beyond the immediate. Much of what is now collected is obtained directly or indirectly through the internet (Amazon, eBay, online surveys, web logs, cookies, etc.), through automated processes tied to the use of credit cards, insurance cards, cell phones, etc. Additional data is obtained from public systems where values are manually entered, copied from other databases or scanned from hard copy documents (marriage, real estate, tax, criminal, etc.). While enormous quantities of data are obtained in this manner, the true value and quality of the data depends upon how useful the information obtained from the data actually is and how expensive this useful information is to obtain.

Therefore well-defined exception reports are a primary tool for reducing costs, increasing the accuracy of data related results and safeguarding against foreseeable errors. The use of “dirty data” increases the cost of obtaining meaningful information from the data, as the data must be “scrubbed”. Scrubbing data involves a cleaning process to identify non-conforming values, and either correcting them, removing them or substituting a less then ideal but more meaningful value. Even more costly is the possibility that dirty data will not be identified, cleaned and/or deleted, so it is incorrectly used to produce erroneous results deemed to be accurate.

Methodology

Programming methodology for the production of exception reports may differ depending upon the nature of the data or the preference of the user, but most will follow a similar progression:

1. Determine what you want
2. Determine what you have
3. Determine what the differences are
4. Document what was found

Tools and Techniques

Each step along the chosen path, various SAS tools can be used to accomplish the intended goal. The list provided is far from comprehensive. Its purpose is to introduce a short list of the easiest to use and useful tools and techniques for exception reporting. The tools and techniques could be used singularly, but are most often and best used in various combinations.

1. Parameter File
2. Procs
3. Formats
4. If/then/do/else and where statements
5. Macros

With the results of a well-defined exception report, the user may make informed decisions as to how the differences discovered will impact their results or a particular process. Without this information, the user would be blind to data related problems and decisions or actions based upon the data could be seriously flawed.

Methodology

1) Determine What You Have

This is a simple task easily accomplished with a Proc Contents or by querying the dictionary tables. Both will list all variables in a data set, their type (numeric or character), their length, their position in the data set, the informat used to read them and any labels associated with them. They will also provide the number of observations in the data set, it's physical location and inform the user if and how the data set has been indexed or sorted.

2) Determine What You Want

This is more challenging and may require significant effort and insight from the user. In a near perfect world, all data would be perfect and every data related application would come with a set of instructions that included pre-defined business rules for data acceptance. The rules provided would detail the minimum requirements for the input data and explain what should be done if the data did not conform to the rules. For every variable used in the process, every acceptable value would be listed, a range of acceptable values would be supplied or additional logic would be included to determine if the value is acceptable. The given list, range or logic could then be used to guide the exception report. As the world is far from perfect, we will have to create our own list of acceptable values, ranges and logic. However SAS makes this task easier as we can create a parameter file to automate the process.

While trying to determine what you want in terms of data quality, keep the following characteristics in mind.

1. Accuracy: Measure of agreement between data values ... assumed to be correct
2. Completeness: Degree to which values are present in the attributes that require them
3. Consistency: Measure of degree to which the data satisfies the constraints
4. Timeliness: Degree to which values are up to date
5. Uniqueness: Ability to establish the uniqueness of a data record (key fields).
6. Validity: Satisfies acceptance requirements

3) Determine What the Differences Are

Determining what you want may be further complicated by the application(s) that are to use the data. It is not uncommon for data to be acceptable for one process and unacceptable for another. This may also be true of different stages within the same process. For example, one condition may require a data field called age to range from 18 to 70. This field would commonly reflect the ages of what many consider to

be “working adults”. However a different process or later stage of the same process may set the condition for all possible ages, perhaps a range of 0 to 104. For this reason, it may be prudent and / or necessary to examine a variable from more than one perspective and to include multiple conditions in the analysis.

4) Document What You Found

The downstream processes in the application should dictate the appropriate format used to document non-conforming values. If the application is able to automatically clean and modify non-conforming values, the results should be stored in a SAS data set that can be used as an input into the cleaning process. If manual intervention, review and editing are involved, it may be best to produce meta report and output the data into generic “user friendly” format. Regardless of the format, the documentation should include information on the data set reviewed, the rules used to evaluate it and the comprehensive list of all non-conforming values.

An exceptional exception report should list the variable being reviewed, a determination of the quality for that observation (order), the value in question and the rule to which it was applied.

QUALITY	VARNAME	VALUE	FORMAT	ORDER	RULE
MISSING	age	.	AGE_A	20	4
MISSING	age	.	AGE_B	20	5
NON-CONFORMING	age	78	AGE_A	21	4
NON-CONFORMING	age	81	AGE_A	22	4
NON-CONFORMING	age	84	AGE_A	23	4
NON-CONFORMING	age	87	AGE_A	24	4
NON-CONFORMING	age	87	AGE_B	24	5

The output data may be further processed for reporting to produce percentages by variable or to designate two the variables meet the quality criteria. The data set may also be used as a parameter file for another process that performs the cleaning and transformation of non-conforming data.

EXCEPTIION AND PERCENTAGE REPORT (EAP)

Variable	Non Missing	Conforming	Scubable?	Accurate	Complete	Consistency	Timeliness	Uniqueness	Validity
NAME_LAST	99%	88%	11%	G	G	G	G	G	G
NAME_FIRST	86%	78%	8%	G	G	F	G	F	F
GENDER	63%	59%	4%	G	G	G	G	N	G
TELEPHONE	100%	6%	94%	B	B	G	B	N	B
AGE	57%	55%	2%	F	G	G	F	N	F
SPECIALTY	76%	72%	5%	G	F	G	G	N	G
EDUCATION	100%	100%	0%	G	G	G	G	B	B

G=Good, F=Fair, B=Bad, N=NA

Tools and Techniques:

1) Parameter File

The first step in the process of creating an exception report is to define what is acceptable for every variable in every step of the process and to place that information into a parameter file. Parameter files are used to drive application, directing the process by with the application will determine if a value is deemed acceptable. The parameter file can also be used as a form of documentation as it shows in detail how each variable was reviewed.

A value may be considered acceptable or conforming if it meets your business needs. Any value that does not meet your business need is unacceptable and non-conforming. To determine which is which, you must have a set of business rules that you can apply to your data. This should be a fairly simple task if the business process has already been defined or if an application has already been developed. If the process or application is in development, the task becomes more difficult as the quality of the data may drive the design of the process.

One method is to list every variable available in a spreadsheet. If you start with a proc contents enhances with ODS, you will already have the variables name, type, length, label, format and informat. Your results will also be in a format easily readable by most spreadsheets. Once you have the variables listed, you can add flags and additional information that will be required for the exception report. If rules can be created to describe the acceptable values or to describe values that are not acceptable, SAS code can be written to tell which is which.

```
/* example of Proc Content to XLS */
ods listing close;
ods html body="c:\das\sug\sesug2004\contents.xls";
  proc contents data=sesug04.fake_data;
  run;
ods html close;
quit;
```

-----Alphabetic List of Variables and Attributes-----

#	Variable	Type	Len	Pos	Format	Informat	Label
4	age	Num	8	8			age
10	dependents	Num	8	48			dependents
7	education	Num	8	24			education
12	end_dt	Num	8	64	DATETIME20.	DATETIME20.	end_dt
2	fname	Char	13	72	$13.00	$13.00	fname
5	gender	Char	1	97	$1.00	$1.00	gender
8	home	Num	8	32			home
3	lname	Char	12	85	$12.00	$12.00	lname
9	m_status	Num	8	40			m_status
15	office	Char	1	122	$1.00	$1.00	office
14	phone	Char	12	110	$12.00	$12.00	phone
6	salary	Num	8	16			salary
1	ssn	Num	8	0			ssn
11	start_dt	Num	8	56	DATETIME20.	DATETIME20.	start_dt
13	title	Char	12	98	$12.00	$12.00	title

As early as possible, flag any variables that will not be use, as this will reduce your workload. Now you can create a number of additional columns for your business rules. A variable may need to meet different criteria during different stages of an application, so it will have to be reviewed with more than one set of rules. If this is the case, you need only to copy a row as many times as the variable needs to be reviewed.

The easiest items to list for numeric variables are typically its minimum and maximum value, which define its range. Some categorical variables, both numeric and character may be defined by creating a short list of acceptable values. Examples are: Gender (M, F), Married (S, M, W, D), and bolenian expressions with values of (0, 1) or (Y, N). Other variables may be more difficult to define, or require definitions that are relative to other variables. Examples of the later would include Education to be less than the person's age, an end date to be later than a certain beginning date, or salaries to be appropriate to a certain title.

Other variables may be defined by their own characteristics. Social Security Numbers (SSN) always have a length of 9 and contain only numbers (0 to 9) after extraneous spacers are removed. Phone numbers will typically have 7, 10 or 11 numbers, depending on the use of area and national codes. Like SSN, they will include only numbers after extraneous spacers are removed. Acceptable values for names is typically the most difficult to define. A simple rule may require all names to be a certain length and to contain at least one vowel. More complicated rules may require the name to be included in a pre-defined list. The process will be greatly enhances if the list of acceptable values has been read into a format similar to the one previously discussed.

Variables such as SSN, Names, Dates, and Telephone numbers will require special processing and can be flagged by the value "S" under type and a special format to designate how they are to be treated. Dates should be handled differently as well and can be denoted by the letter "D" in the type field.

The resulting file should contain all of the business rules required to process the example data and look similar to the following. The following example states what variables are to be examined, provides that format, min, max values or list of acceptable values to apply.

VARNAME	TYPE	FORMAT	MIN	MAX	LIST	RULE
ssn	S	SSN				1
lname	S	NAME				2
fname	S	NAME				3
age	N	AGE_A				4
age	N	AGE_B				5
age	N		21	65		6
gender	C				"M" "F"	7
salary	N		30000	120000		8
education	N		0	18		9
home	N		0	60		10
m_status	C				"S" "M" "D" "W"	11
dependents	N		0	10		12
start_dt	D		1/1/1980	31-12-20		13
end_dt	D		1/1/1981	31-12-20		14

An easy way to transform the values in the created file into true parameters is to convert them into macro variables. Various parts of the exception report application can then access, scan and parse the information as needed. Proc SQL is an ideal method for performing the transformation. The following code reads the parm file (work._rules) and produces eight macro variables. Each macro variable consists of the values from the parm file, separated by a bar (|). The bar is used by the scan function to denote the separation of the values and is normally preferred over spaces or commas. Please note that it may be necessary to create a list with separating spaces so that the macro variable can be used in keep and drop statements.

```
proc sql noprint;
            select varname into  :_ervars   separated by "|" from work._rules;
            select varname into  :_ervlist  separated by " " from work._rules;
            select type    into  :_ertype   separated by "|" from work._rules;
            select format  into  :_erfmts   separated by "|" from work._rules;
            select min     into  :_ermin    separated by "|" from work._rules;
            select max     into  :_ermax    separated by "|" from work._rules;
            select list    into  :_erlist   separated by "|" from work._rules;
            select rule    into  :_errule   separated by "|" from work._rules;
 quit;
```

2) Procs

The simplest form of exception report might consist of a simple Proc Freq for categorical and/or character data and simple Proc Means/Summary or Proc Univariate for continuous and non-categorical numeric data. The flaws and shortcomings in this process are numerous but relatively easy to overcome. While a Proc Freq is an excellent tool for listing all occurrences of a value with related percentages, the results obtained do not automatically and clearly identify non-conforming values. In the output of a basic Proc Freq, conforming values are listed along with the non-conforming and the user must use “Proc Eye Ball” the results to differentiate between the two. As the number of categories grow, the results become unwieldy and more difficult to “eye ball”. This means that as the number of values grows, the opportunity for human error increases and the effectiveness and accuracy of the exception report decreases.

Proc Means/Summary and Proc Univariate have similar shortcomings for the identification of non-conforming values. The largest are that they only operate on numerical data and do not separate the good from the bad. If data type is character and the values are continuous, these tools become extremely unwieldy. Common fields of this nature are SSN and Ids of all types, phone numbers, financial figures, etc. However the procs listed can be very useful for identifying outliers of numeric fields. As with the Proc Freq, this process requires the user to make on the spot judgments as to what is constitutes a non-conforming value.

A clear exception is when a business rule dictates that acceptable numeric value must fall within a statistical measure. The following example can be used to identify and process values that are higher or lower than a desired percentage (%10 and %90). {Modified from Ron Cody’s , Cody’s Data Cleaning Techniques}.

```
%LET MIN=10;   %LET MAX =90;

PROC UNIVARIATE DATA=SESUG004.FAKE_DATA;
   VAR SALARY;
   ID SSN;
   OUTPUT OUT=TMP PCTLPTS=&MIN &MAX PCTLPRE = C_;
RUN;

DATA HILO;
   SET SESUG004.FAKE_DATA (KEEP=SSN SALARY);
   ***BRING IN UPPER AND LOWER CUTOFFS FOR VARIABLE;
   IF _N_ = 1 THEN SET TMP;
   IF SALARY LE C_&MIN THEN DO;
      RANGE = 'LOW ';
      OUTPUT;
   END;
   ELSE IF SALARY GE C_&MAX THEN DO;
      RANGE = 'HIGH';
      OUTPUT;
   END;
RUN;
```

The shortcomings of using many procs can easily be overcome by augmenting the procs with formats (proc format), various procedural options and/ or additional data step manipulation. Formats allow the user to pre-define values as either conforming or non-conforming. In a similar manner, procedural options may allow the user to focus on only those values that are of interest. Data step options (where) and statements (if/then) apply specific business rules to the task. All are discussed singularly and in combination in the following sections.

3) Formats

A simple solution that greatly enhances the Procs mentioned and many more is the use of formats. Formats associate a given value with another value. Their primary use in exception reports is to identify the non-conforming values and separate them from those business rules. One method is to use formats to identify all conforming values with the value of “CONFORM” and all others with either “MISSING” or “NON-CONFORMING”. Another method is to use formats to eliminate all conforming values so that only non-conforming values are processed.

In the following example, a format is used to assign conforming values the label “CONFORM” while all non-conforming values remain unchanged. This allows the user to immediately identify the fields that need to be dealt with. In this example, the non-conforming values are the seven values listed last and the three missing values listed first. The example following the output adds an expected value for missing values and assigns a single value to non-conforming values.

```
Proc format;

   Value age_a
         18 - 65 = "CONFORM”;
run;

Proc Freq data=sesug04.fake_data;
     Tables age;
     Format age age_a. ;
Run;
```

The FREQ Procedure

age

age	Frequency	Percent	Cumulative Frequency	Cumulative Percent
.	3	12.50	3	12.50
CONFORM	14	58.33	17	70.83
66	1	4.17	18	75.00
69	1	4.17	19	79.17
72	1	4.17	20	83.33
78	1	4.17	21	87.50
81	1	4.17	22	91.67
84	1	4.17	23	95.83
87	1	4.17	24	100.00

The following example expands upon the use of formats to categorize them as “MISSING”, “CONFORMING’ or “NON-CONFORMING”. In terms of exception reporting, this is an ideal application. The transformation was achieved by adding additional values to the Proc Format and the missing option to the Proc Freq.

```
Proc format;
   Value age_b
            .     = "   MISSING    "
         18 - 65 = "   CONFORM    "
         OTHER   = "NON-CONFORMING";
run;

Proc Freq data=sesug04.fake_data;
     Tables age / missing;
     Format age age_b. ;
Run;
```

The FREQ Procedure

AGE

AGE	FREQUENCY	PERCENT	CUMULATIVE FREQUENCY	CUMULATIVE PERCENT
MISSING	3	12.50	3	12.50
CONFORM	14	58.33	17	70.83
NON-CONFORMING	7	29.17	24	100.00

4) If /Then/Else and Where Statements

Data step statements that incorporate “If /Then/Do/Else” and “Where” statement allow for very specific culling. One may select or skip specific values. The values supplied may be in the forms of a minimal value, a maximal value, both that constitute a range, a short list or a related variable. They may also incorporate the use of formats to apply broader pre-defined business rules. The following example uses where statement enhanced with a format avoid processing values that conform to business rules. Only values that are non-conforming are selected which greatly increasing program efficiency and effectiveness. The output lists only the values that should be modified, cleaned or deleted.

```
Proc format;
   Value age_a
         18 - 65 = "CONFORM";
run;

Proc Freq data=sesug04.fake_data (where=(put(age, age_a.)^="CONFORM"
));
     Tables age / missing;
Run;
```

The FREQ Procedure

AGE

AGE	FREQUENCY	PERCENT	CUMULATIVE FREQUENCY	CUMULATIVE PERCENT
.	3	30.00	3	30.00
66	1	10.00	4	40.00
69	1	10.00	5	50.00
72	1	10.00	6	60.00
78	1	10.00	7	70.00
81	1	10.00	8	80.00
84	1	10.00	9	90.00
87	1	10.00	10	100.00

If/then/do/else statements may be used in a similar manner. While where statements are limited in a number of ways, If/then/else statements are extremely flexible. The shortcoming of if/then/else statements is that they are generally hard coded. This results in “wallpaper” programs that are difficult to modify as business rules change. An alternative is to utilize macro variables to increase flexibility.

By combining the if/then/do/else statements with macro variables and formats, the user has a powerful and versatile tool. Notice how the following code outputs both the name of the variable being reviewed and the value of the variable.

```
%if &_fmt ^= and &_typ ^= S %then
    %do;
          quality =  put( &_var , &_fmt.. )
          varname="&_var";
          value  = &_var ;
          format ="&_fmt";
          order  =_n_;
          rule   ="&_rule";

         if quality ^= 'GOOD' then
         do;
            _quality = quality;
            output exception;
         end;
              output all;
    %end;
```

5) Macros

Macros are what makes everything come together and turns a program into a data driven application. Macro variables can be used to pass parameters from the parameter file into the code avoiding the need to “hard code” variable names or the values derived from the business rules. Macro looping allows small but significant portions of code to be processed repeatedly, but only as often as is needed and always with the appropriate variables or values. Readers will need to seek other sources of information on macros, as this space is to limited to do more than list the examples previously provided.

Conclusion

SAS offers a wide variety of tools and techniques for the creation and automation of exception reports for quality control. Exception reports should be an integral part of any data related application. Their primary use is to identify non-conforming values that could adversely affect the results of a process. They can also be used to guide data cleaning and transformation efforts, or to provide insight into the origin of abnormalities found in downstream processes when data quality is not improved.

This paper and a modifiable example of an exception report may be downloaded from http://www.dasconsultants.com/pubs.html. The exception report is "as is" and may be modified and used as desired.

Acknowledgements

Ron Cody, Cody's Data Cleaning Techniques Using SAS Software, SAS Institute BBU 1999

Janet Stuelpner, "Mrs. Clean Tackles Dirty Data" SESUG Proceedings, 2002

Acknowledgements: Special thanks to Dr. Dawn Li for her assistance and support.

About the Author

Gary McQuown is a co-owner of Data and Analytic Solutions, Inc, a SAS Alliance Partner, located in Fairfax, Virginia. He has programmed in SAS long enough to learn the difference between a programmer and a developer. He provides SAS solutions for large and small firms in the US and Europe. He has previously presented at SUGI, NESUG, SESUG, SSU and MWSUG and can be contacted at mcquown@dasconsultants.com.