

Managing Data Issues Identified During Programming

Shafi Chowdhury, Shafi Consultancy Limited, London, U.K.
Aminul Islam, Shafi Consultancy Bangladesh, Sylhet, Bangladesh

ABSTRACT

Managing data issues identified during programming of analysis datasets, tables, listings or figures is a major headache for both Programmers and Data Managers. The issues have to be listed so the Programmer is aware of them, and they have to be sent to the Data Manager so that the issue can be resolved. The problem develops when the program is re-submitted over time. The Programmer is then no longer aware of what the new issues are and which have already been sent to the Data Manager. The Data Manager suffers equally from receiving the same data issues many times in various e-mails, and they then have to check what the new issues are and which have already been queried. Our solution is to simplify and automate the current process so that the issues are not sent to the Data Manager, but they are shared with the Data Manager, and to ensure that the same issues are not raised more than once. This centralization helps to manage the issues raised during programming. One can easily see which issues were raised, how many issues there were and how many are still unresolved.

INTRODUCTION

This paper will show a simple solution to manage data issues raised by programmers during the development of analysis datasets, tables, listings and figures. Checking data during reporting is a necessary part of the process, but one which can cause frustration to programmers as issues remain unresolved, and waste time of Data Managers as they are sent issues which they have already investigated and taken action to resolve.

There are always data issues when we start programming the analysis datasets, tables, listings and figures. This is because the database is still open and data is still being collected. Although the Data Managers do their own checks to clean the database, it is never enough. As programming starts before the database is clean, programmers always identify data issues. Some of these will be new issues which the Data Manager was perhaps not looking for, and some will be things they are already querying.

The problem for programmers is what they should do when they identify an issue. Typically they would have code in their programs to send data issues to the LOG window. These will then be sent to the Data Manager by e-mail for them to look at. When they submit the program again in the future with new data, they are no longer sure if the issues sent to the LOG have already been sent to the Data Manager or not. The usual response is to just send it again and let the Data Manager take care of it.

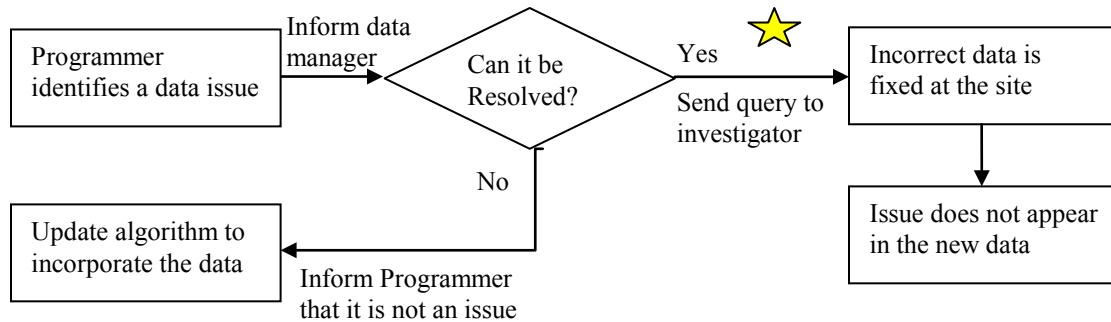
The problem for Data Managers is that they receive data issues by e-mail, sent to them from various Programmers working in different aspects of the trial reporting process. They need to save these e-mails and look at the issues identified, especially if they were not already searching for these issues. There is usually a constant stream of issues raised by different Programmers, and it is time consuming to see what has already been queried or looked into, and which issues are new.

STANDARD PROCESS FOR RAISING AN ISSUE

The main tasks and associated task manager involved in this process are:

- The Programmer identifies a data issue
- The Programmer communicates the issue to the Data Manager
- The Data Manager will resolve the issue, send query to the investigator or inform the Programmer that it will not be resolved
- The Programmer must use algorithms to resolve the issue if no action is taken by the Data Manager
- When the Programmer re-submits their program, the issue is either still there, or it has been resolved and therefore no longer present

Illustration of the process for raising an issue:



The diagram shows the flow of information from when an issue is identified to its final resolution. However, when the Programmer re-submits their program after a query has been raised, but before the data has been corrected, the issue is identified again. The Programmer sends the issues to the Data Manager, who then spends time checking this, only to find that they had already looked into it previously.

There are three main problems which need to be addressed in this process:

- How to communicate the data issues to the Data Manager without having to cut and paste data in an e-mail
- How to identify issues already sent so that they are not sent multiple times
- How to tell if an issue has been resolved or not

SOLUTION

The first problem of communicating the issue with the Data Manager is one which we need to look at the problem from a different perspective. Instead of sending the issues to the Data Manager, we can share the issue with the Data Manager. This simple change in terminology leads us to using a common central storage space where the issues are stored. This can then be accessed by both parties, and anyone else with the appropriate authorization, to see the status of the issues. Storing them in one location also allows us to have additional benefits. It is then possible to identify which types of issues are being raised, who raised them, how many issues there are, if they have been resolved and how long it took to resolve the issues.

There are many methods available for storing data in a central location, including different types of databases. There are advantages and disadvantages for using the various methods. For simplicity and ease of use, we have decided to **use Excel spreadsheet** to share the issues raised.

There are many issues which are checked in each program, and there are many programs. Creating a different file for each issue would be cumbersome to manage. However, a shared spreadsheet will allow multiple users to open and read or write to the file. The issues can all be in different sheets within the one spreadsheet.

If each issue is saved in a new sheet then there will be too many sheets, and providing each sheet with a meaningful name so that they are easy to identify was a major problem. The aim was to create a simple list of sheet names within the Excel file which the users can easily related to for finding the issues. It was decided to **place all issues from each program in one sheet**, and the sheet would have the same name as the program. This will make it easy to see where the issue was raised, and which Programmer was responsible. It will also eliminate the problems of too many sheets, and what they should all be called. One can simply look at the tabs below to find the program they wish to look at. Within the sheet there will be a title for the issue followed by all the patient data which should be investigated, then title of the next issue and all the related patient data underneath that title, and so on.

The second problem of resending the same issue multiple times can be resolved by adding a column to identify when the issue was first raised and a column containing the date and time the issue was most recently checked, and found to be still present. This would make it easier to see how much time has passed since the issue was first raised.

The final problem of identifying the status of an issue can be solved by a derivation rule. If an issue is no longer present when the program is re-submitted then it can be considered to have been resolved. This could be because

PhUSE 2011

the issue is no longer checked or because the data has been updated. In any case, an issue which was present in the excel file during a previous run of the program, and is no longer present is considered to have been resolved.

To make it easier to track an issue in the Excel file a certain structure of the sheet is required. In our case we used the first three columns to identify when the issue was first raised, when it was last updated, and what the current status of the issue is, i.e. is it resolved or not. This allows us to create a **summary sheet** which contains names of all the sheets in the spreadsheet, followed by the number of issues in that sheet, and how many are still un-resolved. This simple overview allows both Data Managers and Programmers to see the status of data issues very quickly. It also ensures that they do not have to go to each sheet every time to check that there are no new issues.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
	COL1	COL2	COL3	COL4	COL5	COL6	COL7	COL8	COL9								
4	TITLE: Age should not be over 60 in the <= 60 population																
6	ISSUE_CRDATE	ISSUE_LAST_UPDATED	ISSUE_RESOLVED	PATIENT	AGE	SEX	RACE										
7	01OCT11:18:52:01	01OCT11:19:00:36	Yes	10008	83	2											
8	01OCT11:18:52:01	01OCT11:19:00:36	Yes	10007	68	2											
9	01OCT11:18:52:01	01OCT11:19:00:36	Yes	10006	61	1											
10	01OCT11:18:58:04	01OCT11:18:58:04	No	10004			White										
11	01OCT11:18:58:04	01OCT11:18:58:04	No	10004	66	1											
12	01OCT11:18:52:01	01OCT11:18:58:04	No	10003			White										
13	01OCT11:18:52:01	01OCT11:18:58:04	No	10003	64	1											
14	01OCT11:18:52:01	01OCT11:18:58:04	No	10002	68	2											
15	01OCT11:18:52:01	01OCT11:18:58:04	No	10001	61	1											
18	TITLE: HbA1c values above 13																
20	ISSUE_CRDATE	ISSUE_LAST_UPDATED	ISSUE_RESOLVED	PATIENT	VISIT	SAMP_DT	SAMP_TM	RESULT	UNIT								
23	01OCT11:18:52:53	01OCT11:19:00:36	Yes	12191	10.00	15MAR2011	15:50	17.00 %									
24	01OCT11:18:52:53	01OCT11:19:00:36	Yes	11929	10.00	29JUN2011	10:58	14.40 %									
25	01OCT11:18:52:53	01OCT11:19:00:36	Yes	11733	10.00	02JUN2011	11:00	13.50 %									
26	01OCT11:18:52:53	01OCT11:19:00:36	Yes	11728	10.00	25FEB2011	8:00	13.20 %									
27	01OCT11:18:52:53	01OCT11:19:00:36	Yes	11579	10.00	26MAY2011	9:30	14.10 %									
28	01OCT11:18:52:53	01OCT11:19:00:36	Yes	11577	10.00	10JAN2011	9:30	13.40 %									
29	01OCT11:18:52:53	01OCT11:19:00:36	Yes	11507	10.00	25JAN2011	9:27	13.90 %									
30	01OCT11:18:52:53	01OCT11:19:00:36	Yes	10910	10.00	27MAY2011	9:50	13.10 %									

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
	COL1	COL2	COL3														
3	Sheet	Number of issues	Number of issues unresolved														
4	PROG1	37	18														
5	PROG2	18	6														

A SAS macro was developed for managing the issues raised during programming. It required the Programmer to create a new SAS dataset for each issue, and then calling the macro to add it to the spreadsheet. The label of the dataset is used as the title, and if a KEEP statement is provided in the label then only those variables specified after the KEEP statement will be printed. A patient data which is in the spreadsheet but is no longer appearing in the SAS dataset as an issue is considered to be resolved.

Of course some issues are just for the Programmer, and not for the Data Manager, therefore an option was provided to allow issues to be just printed in the OUTPUT window and not be sent to the Excel file.

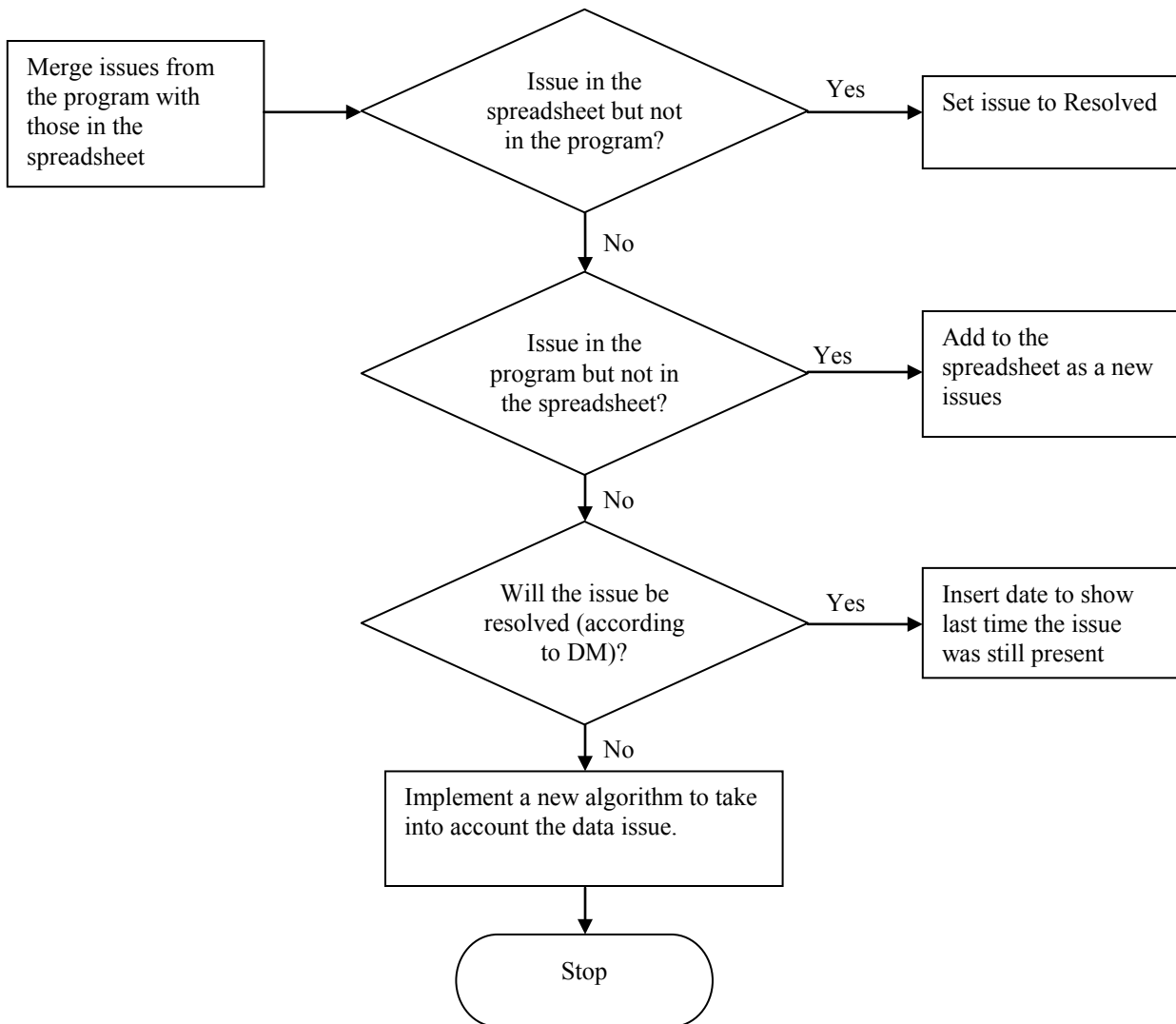
PhUSE 2011

A typical check would look like:

```
DATA hba1c_gt13(LABEL='HbA1c values above 13 KEEP=patient visit samp_dt  
samp_tm result unit');  
SET hba1c;  
WHERE sp GT 13;  
RUN;
```

```
%data_issue(pgmname=&progrname., indata=hba1c_gt13, send2xls=y);
```

Programmer's process for managing data issues will now become:



CONCLUSION

There were three main parts to this problem of managing data issues raised during the programming of analysis datasets, tables, listings and figures. By removing the sending of issues via e-mail and creating a shared Excel file we have eliminated one of the main causes of frustration, receiving issues more than once. The sharing of issues ensures there is **no extra work** for the Programmer when communicating the issues to the Data Manager. The addition of three extra columns in the Excel file to show the date of creation, the date the issue was last updated and whether it has been resolved or not allows the Programmer and the Data Manager to easily see if an issue is new, or one which already existed previously.

The use of a SAS macro in combination with an excel file allows the management of data issues to be taken to a new level. Programmers working on different parts of the trial reporting process can all call the macro to share data issues with the Data Manager using one central file. It can be used to assess the quality of the data, time taken to resolve issues and finally check if there are still any outstanding issues before the database is locked.

ACKNOWLEDGMENTS

We wish to thank Rafi Rahi for testing this macro and investigating different approaches to solving this problem.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Shafi Chowdhury
Shafi Consultancy Limited
7 Blossom Way
Uxbridge
Middlesex
UB10 9LL

Phone: +44 770 288 7219
Email: shafi@shaficonsultancy.com

Md. Aminul Islam
Shafi Consultancy Bangladesh
Parizath-1 (1st floor)
Housing Estate
Sylhet
Bangladesh

E-mail: aminul@shaficonsultancy.com

Web: www.shaficonsultancy.com

Brand and product names are trademarks of their respective companies.