PharmaSUG 2019 - Paper AD-215

Use of SAS Merge in Adverse Events Reporting for DSUR

Andrew Wang, Celgene Corporation

ABSTRACT <HEADING 1>

In reporting adverse events for DSUR, statistical programmers often need to separate out AEs reported already in previous years from AEs present in the current database. The task of identifying the adverse events already reported could present a challenge to the programmer since there is no easy way to ID an adverse event using the variables in the datasets. All types of data issues could be present in the data which may cause the values of many variables to change over time in the data query and cleaning processes. For example, an AE could get deleted from the database or be split into multiple AEs. For an ongoing study, data might naturally change with time as well. For instance, un-coded AEs could get coded, ongoing AEs as of now will likely have an end date later. Many scenarios could happen. This paper will detail the above challenges and talk about possible solutions. The intended audience are statistical programmers and people responsible for DSUR reporting in drug safety group.

INTRODUCTION <HEADING 1>

In FDA's August 2011 "Guidance for Industry – E2F Development Safety Update Report", it is pointed out that "The main objective of a DSUR is to present a comprehensive, thoughtful annual review and evaluation of pertinent safety information collected during the reporting period related to a drug under investigation" and "Section 7.1-7.3 of the DSUR should present important clinical safety information through interval line listings of the SARs that were reported to the sponsor during the period covered by the DSUR…" As such, a statistical programmer is often tasked by the sponsor to separate the Adverse Events that have been reported in previous years from the ones that have just happened in the current reporting period. A seemingly simple task, however, could present significant challenges due to many reasons, such as the different CRF design and database structure, ongoing data entering and coding processes, various types of data issues etc. Uniquely identifying an adverse event is typically not easily achievable, which makes the later operation of match merge quite tricky and error-prone. In the following sections, the author will attempt to start from the simplest example and explain the different situation a statistical programmer might face in this process. Some thoughts on the best possible solutions are provided as well for the audience to form their own strategies. The hope of the author is to benefit the statistical programmers who directly work on DSUR datasets and tables/listings and the drug safety reporting group of sponsors, who may need to work with statisticians and programmers together to come up with the best strategy possible in adverse events reporting.

SIMPLEST SCENARIO <HEADING 1>

To illustrate the concept, a simplest scenario is presented below. Some mocked-up data will be used for current year's DSUR (dataset name: newae) and last year's DSUR (dataset name: oldae):

```
data oldae;
  length usubjid $5 aebodsys $20 aedecod $20 aestdtc $10 aeacn $20;
  infile datalines delimiter= ',';
  input usubjid $ aebodsys $ aedecod $ aestdtc $ aeacn $;
  datalines;
  101, INVESTIGATIONS, WEIGHT DECREASED, 10SEP2017, DRUG INTERRUPTED
  ;
run;
data newae;
  length usubjid $5 aebodsys $20 aedecod $20 aestdtc $10 aeacn $20;
  infile datalines delimiter= ',';
  input usubjid $ aebodsys $ aedecod $ aestdtc $ aeacn $;
  datalines;
  101, INVESTIGATIONS, WEIGHT DECREASED, 10SEP2017, DRUG INTERRUPTED
  101, VASCULAR DISORDERS, HYPERTENSION, 13JAN2018, DOSE REDUCED
  ;
```

```
run;
```

If you are not familiar with SAS programming language, the two mocked up datasets are also shown in Table 1 and Table 2 below.

USUBJID	AEBODSYS	AEDECOD	AESTDTC	AEACN
101	INVESTIGATIONS	WEIGHT DECREASED	10SEP2017	DRUG INTERRUPTED

Table 1. Mocked-up of Last Year's AE Dataset

USUBJID	AEBODSYS	AEDECOD	AESTDTC	AEACN
101	INVESTIGATIONS	WEIGHT DECREASED	10SEP2017	DRUG INTERRUPTED
101	VASCULAR DISORDERS	HYPERTENSION	13JAN2018	DOSE REDUCED

Table 2. Mocked-up of Current Year's AE Dataset

Suppose both datasets have been sorted properly, to pick out the adverse events that have already been reported in the previous year's DSUR and leave only the adverse events that we would like to report this year, we use a simple data step as follows to obtain the "final" dataset we want to work with. The resulted dataset is shown in Table 3.

```
data final;
  merge oldae(in=inold keep=usubjid aebodsys aedecod aestdtc)
        newae(in=innew) ;
  by usubjid aebodsys aedecod aestdtc;
  if innew and not inold;
run;
```

USUBJID	AEBODSYS	AEDECOD	AESTDTC	AEACN
101	VASCULAR DISORDERS	HYPERTENSION	13JAN2018	DOSE REDUCED

Table 3. Final AE Dataset Ready for Reporting in This Year's DSUR

Note that in the above data manipulation, we chose USUBJID, AEBODSYS, AEDECOD and AESTDTC as identifiers to determine which AE in one dataset has a corresponding identical AE in the other dataset. Considering the fact that there are many different variables in an AE dataset, we need to make a decision on which variables to select to act as the unique identifiers. This will be explained and discussed more in later sections.

WHAT CAN WE DO WHEN THERE ARE "REPEATS" IN DATA <HEADING 1>

The above section shows the simplest possible scenario. The reason we call it "the simplest" is due to the fact that we can easily tell which AE is from previous database and appeared again in the current database; and we can easily tell which AE is in the current database but not in previous database. Unfortunately, in real world programming practice, things almost always get more complicated. Let's start with the below mocked-up example dataset and let's pretend this is our AE dataset from current database.

USUBJID	AEBODSYS	AEDECOD	AESTDTC	AEACN
102	GENERAL DISORDERS AND ADMINISTRATION SITE CONDITIONS	OEDEMA PERIPHERAL	18AUG2015	DOSE NOT CHANGED
102	GENERAL DISORDERS AND ADMINISTRATION SITE CONDITIONS	OEDEMA PERIPHERAL	18AUG2015	DOSE NOT CHANGED

USUBJID	AEBODSYS	AEDECOD	AESTDTC	AEACN
102	GENERAL DISORDERS AND ADMINISTRATION SITE CONDITIONS	OEDEMA PERIPHERAL	18AUG2015	DOSE NOT CHANGED

Table 4. An Example AE Dataset Showing Seemingly Duplicate Adverse Events

A quick examination of the data reveals that the three AE records seem to be exactly the same! Or more precisely, for the variables we are showing, we cannot tell the difference between the three records. Knowing that there are many more variables in the AE domain than already shown here, a closer look at the data reveals that the three records are different in AETERM, as shown in Table 5.

USUBJID	AEBODSYS	AEDECOD	AESTDTC	AEACN	AETERM
102	GENERAL DISORDERS AND ADMINISTRATION SITE CONDITIONS	OEDEMA PERIPHERAL	18AUG2015	DOSE NOT CHANGED	INCREASED MILD PERIPHERAL EDEMA OF LOWER ARMS
102	GENERAL DISORDERS AND ADMINISTRATION SITE CONDITIONS	OEDEMA PERIPHERAL	18AUG2015	DOSE NOT CHANGED	INCREASED MILD PERIPHERAL EDEMA OF ANKLES
102	GENERAL DISORDERS AND ADMINISTRATION SITE CONDITIONS	OEDEMA PERIPHERAL	18AUG2015	DOSE NOT CHANGED	INCREASED MILD PERIPHERAL EDEMA OF HANDS

Table 5. An Example AE Dataset Showing Seemingly Duplicate Adverse Events Are Different

Now, suppose our AE dataset used for previous year’s DSUR looked as follows.

USUBJID	AEBODSYS	AEDECOD	AESTDTC	AEACN	AETERM
102	GENERAL DISORDERS AND ADMINISTRATION SITE CONDITIONS	OEDEMA PERIPHERAL	18AUG2015	DOSE NOT CHANGED	INCREASED MILD PERIPHERAL EDEMA OF LOWER ARMS
102	GENERAL DISORDERS AND ADMINISTRATION SITE CONDITIONS	OEDEMA PERIPHERAL	18AUG2015	DOSE NOT CHANGED	INCREASED MILD PERIPHERAL EDEMA OF ANKLES

Table 6. Example of Previous Year's AE Dataset

If we still use the SAS data step outlined in the “SIMPLEST SCENARIO” section to figure out which AEs to report this year.

```
data oldae;
  length usubjid $5 aebodsys $60 aedecod $20 aestdtc $10 aeacn $20
aeterm $60;
  infile datalines delimiter= ',';
  input usubjid $ aebodsys $ aedecod $ aestdtc $ aeacn $ aeterm $;
  datalines;
  102, GENERAL DISORDERS AND ADMINISTRATION SITE CONDITIONS, OEDEMA
PERIPHERAL, 18AUG2015, DOSE NOT CHANGED, INCREASED MILD PERIPHERAL EDEMA OF
LOWER ARMS
  102, GENERAL DISORDERS AND ADMINISTRATION SITE CONDITIONS, OEDEMA
PERIPHERAL, 18AUG2015, DOSE NOT CHANGED, INCREASED MILD PERIPHERAL EDEMA OF
ANKLES
run;

data newae;
  length usubjid $5 aebodsys $60 aedecod $20 aestdtc $10 aeacn $20 aeterm
$60;
  infile datalines delimiter= ',';
  input usubjid $ aebodsys $ aedecod $ aestdtc $ aeacn $ aeterm $;
  datalines;
  102, GENERAL DISORDERS AND ADMINISTRATION SITE CONDITIONS, OEDEMA
PERIPHERAL, 18AUG2015, DOSE NOT CHANGED, INCREASED MILD PERIPHERAL EDEMA OF
LOWER ARMS
  102, GENERAL DISORDERS AND ADMINISTRATION SITE CONDITIONS, OEDEMA
PERIPHERAL, 18AUG2015, DOSE NOT CHANGED, INCREASED MILD PERIPHERAL EDEMA OF
ANKLES
  102, GENERAL DISORDERS AND ADMINISTRATION SITE CONDITIONS, OEDEMA
PERIPHERAL, 18AUG2015, DOSE NOT CHANGED, INCREASED MILD PERIPHERAL EDEMA OF
HANDS
  ;
run;
data final;
  merge oldae(in=inold keep=usubjid aebodsys aedecod aestdtc)
        newae(in=innew) ;
  by usubjid aebodsys aedecod aestdtc;
  if innew and not inold;
run;
```

we would get the following SAS note in the log as shown in Figure 1 below:

```
NOTE: MERGE statement has more than one data set with repeats of BY
values.
NOTE: There were 2 observations read from the data set WORK.OLDAE.
NOTE: There were 3 observations read from the data set WORK.NEWAE.
NOTE: The data set WORK.FINAL has 0 observations and 6 variables.
```

Figure 1. SAS Log to Show a Many to Many Merge

This many-to-many merge situation should typically be avoided, and the results are wrong, because we are left with no AEs to report. To obtain the correct results, we could have used the following SAS code to avoid the many-to-many merge note:

```
proc sort data=oldae out=id(keep=usubjid aebodsys aedecod aestdtc)nodupkey;
  by usubjid aebodsys aedecod aestdtc;
run;
proc sort data =newae;
  by usubjid aebodsys aedecod aestdtc;
run;
data final;
  merge id(in=inid)
        newae(in=innew) ;
  by usubjid aebodsys aedecod aestdtc;
  if innew and not inid;
run;
```

This time, we will have a one to many merge case, and the SAS log is clean, but the results is still wrong since we still have 0 AEs to report, as shown in Figure 2:

```
NOTE: There were 1 observations read from the data set WORK.ID.
NOTE: There were 3 observations read from the data set WORK.NEWAE.
NOTE: The data set WORK.FINAL has 0 observations and 6 variables.
```

Figure 2. A SAS Log to Show a One-to-Many Merge Scenario

This result should not be a surprise. Since we know all three AEs have exactly the same values for all the identifier variable we are currently using. All three AEs are being identified as having been reported before, therefore, no AEs are left to be reported this year.

As one of the many ways to solve this problem, we could add AETERM into the list of identifier variables. In doing that, we will have the following new programming codes.

```
proc sort data=oldae out=id(keep=usubjid aebodsys aedecod aestdtc aeterm)
nodupkey;
  by usubjid aebodsys aedecod aestdtc aeterm;
run;
proc sort data =newae;
  by usubjid aebodsys aedecod aestdtc aeterm;
run;
data final;
  merge id(in=inid)
        newae(in=innew) ;
  by usubjid aebodsys aedecod aestdtc aeterm;
  if innew and not inid;
run;
```

The SAS log is shown below in Figure 3, and it can be seen that we now have the correct record we need to report in this year's DSUR (Table 7).

```
NOTE: There were 2 observations read from the data set WORK.ID.
NOTE: There were 3 observations read from the data set WORK.NEWAE.
NOTE: The data set WORK.FINAL has 1 observations and 6 variables.
```

Figure 3. A SAS Log to Show a Match Merge Without Repeats

USUBJID	AEBODSYS	AEDECOD	AESTDTC	AEACN	AETERM
102	GENERAL DISORDERS AND ADMINISTRATION SITE CONDITIONS	OEDEMA PERIPHERAL	18AUG2015	DOSE NOT CHANGED	INCREASED MILD PERIPHERAL EDEMA OF HANDS

Table 7. The Correct Dataset with AE to Report in This Year's DSUR

WHAT VARIABLES TO USE AS IDENTIFIERS<HEADING 1>

In the "SIMPLEST SCENARIO" section, we picked USUBJID, AEBODSYS, AEDECOD and AESTDTC as identifier variables. In the above section, we discussed one possible solution of adding more variables to the list of identifiers when repeats exist in the data. Since we might have hundreds of subjects and thousands of AEs in the dataset, the task of picking the right identifier variables to be on the list may not be as easy as we think.

We notice is that we could get duplicate data in many different ways, let's see a few more examples of that.

USUBJID	AEBODSYS	AEDECOD	AESTDTC	AEACN	AESEV
103	GASTROINTESTINAL DISORDERS	CROHN'S DISEASE	10SEP2017	DOSE NOT CHANGED	MILD
103	GASTROINTESTINAL DISORDERS	CROHN'S DISEASE	10SEP2017	DOSE NOT CHANGED	MODERATE

Table 8. Two AE Records with Difference Only for Variable AESEV

In the example above shown in Table 8, all variables including AETERM have the same values for the two AE records (not shown completely) except for AESEV.

USUBJID	AEBODSYS	AEDECOD	AESTDTC	AEACN	AECONTRT
104	GASTROINTESTINAL DISORDERS	ABDOMINAL PAIN	15FEB2017	DOSE NOT CHANGED	YES
104	GASTROINTESTINAL DISORDERS	ABDOMINAL PAIN	15FEB2017	DOSE NOT CHANGED	NO

Table 9. Two AE Records with Difference Only for Variable AECONTRT

In the example above shown in Table 9, all variables have the same values for the two AE records (not shown completely) except for AECONTRT.

USUBJID	AEBODSYS	AEDECOD	AESTDTC	AEACN	AELLT
105	GENERAL DISORDERS AND ADMINISTRATION SITE CONDITIONS	PYREXIA	21DEC2016	DOSE NOT CHANGED	FEVER
105	GENERAL DISORDERS AND ADMINISTRATION SITE CONDITIONS	PYREXIA	21DEC2016	DOSE NOT CHANGED	INTERMITTENT FEVER

Table 10. Two AE Records with Difference Only for Variable AELLT

In the example above shown in Table 10, all variables have the same values for the two AE records (not shown completely) except for AELLT.

USUBJID	AEBODSYS	AEDECOD	AESTDTC	AEACN	AESER
106	INFECTIONS AND INFESTATIONS	ANAL ABSCESS	18OCT2016	DOSE NOT CHANGED	YES
106	INFECTIONS AND INFESTATIONS	ANAL ABSCESS	18OCT2016	DOSE NOT CHANGED	NO

Table 11. Two AE Records with Difference Only for Variable AESER

In the example above shown in Table 11, all variables have the same values for the two AE records (not shown completely) except for AESER.

If every time we see repeats in the data and solve the problem by adding one variable into the list of identifiers as we do in the last section, this list tends to grow very long for a study with lots of adverse events. At a certain point, we will start asking ourselves the question: is this the best way to solve the problem?

DISCUSSIONS AND PROPOSED SOLUTIONS <HEADING 1>

Continuing from previous sections, we have come to a point that we start wondering whether growing the identifier list is the correct solution to deal with the “repeats” in the adverse events data.

We realize that these “repeats” are typically not real repeats, because if we consider all the relevant variables in the dataset as identifiers, then very rarely we should see any repeats. From the real-world experience, we also realize that almost any variable can change their values from last year to this year. In the author’s experience, AETERM, AESEV, AESER, AELLT, AEBODSYS, AEDECOD, AEAC(N), AESTDTC, AEONGO etc. can all change their values. The reasons for the changes fall into at least two categories. We may call the first category as “normal” changes because they are supposed to happen during the course of the study. For example, AEONGO can change from “ONGOING” to blank when an adverse event comes to an end. The second category could be called “changes due to data issues”, because many times, the data will change in the processes of data queries and cleaning. Therefore, if we keep growing the identifier list, we are taking an increasing risk to repeatedly report certain adverse events.

Some statistical programmers who are familiar with the adverse events data might at this moment think of a seemingly perfect variable to be used as an identifier (together with USUBJID), and that variable is AESEQ. Unfortunately, you are more than likely out of luck unless AESEQ is directly mapped from a sequence variable in the database. And even if it were, very often we see one single AE can become multiple AEs, and two more AEs can be combined into one AE, inheriting one of the old sequence numbers or getting a completely new number. Therefore, unless the audit trail can be output into our datasets, there is no perfect solution to identify the AEs that have been reported before.

To take this discussion one step further, we realize that all the problem-solving strategies we have been talking about so far assume that we need to figure out exactly which AEs have been reported before and we don’t need to report them again even if there are some data value changes in those AE records. This turns out to be not necessarily true. For example, if a certain AE was reported as “not leading to discontinuation”, but from the new dataset, it would have been reported as “leading to discontinuation”; in this scenario, don’t we want to report it one more time? This must be evaluated based on the sponsor’s overall reporting strategy and FDA’s guidance.

To suggest a solution, the author suggests that the statistical programmer needs to work with data manager, statistician and drug safety group colleagues closely to make the reporting objectives very clear and make all parties understand the complexity of the data, then work backwards to decide on the programming techniques to use, possibly including the practice of identifying the key variables to use when doing the match merge. All programming techniques will have to serve the reporting objectives and to that end, there is not one universal perfect solution since each sponsor could have similar but somewhat different goals to achieve.

CONCLUSION <HEADING 1>

In this paper, the author attempted to demonstrate the simple technique statistical programmers tend to use to separate out old AEs from new AEs in DSUR reporting and the challenges that this simple technique faces in real programming environment due to the natural changes of data from time to time and the additional challenges presented by data issues. Some thoughts on the solution are proposed for programmers’ reference and further discussion and explorations. Eventually, the programming strategy partly depends on the sponsor’s interpretation of FDA’s guidance and overall strategy of how the data should be reported. From a programmer’s perspective, we cannot simply use some SAS functionality without thinking deeply and understanding the data and the task at hand thoroughly. We may write perfectly valid SAS programs with no issues in the SAS log but deliver quite unsatisfactory results if they don’t fit with our purposes.

REFERENCES <HEADING 1>

FDA 2011. “Guidance for Industry. E2F Development Safety Update Report.” Accessed Dec 23, 2018. https://www.fda.gov/downloads/drugs/guidances/ucm073109.pdf.

CONTACT INFORMATION <HEADING 1>

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

Name: Andrew Wang
Email: xtwang@celgene.com