An Efficient Report Checking Method

Xuejing (Susan) Mao, Eli Lilly and Company, Indianapolis, IN
Mario Widel, Eli Lilly and Company, Indianapolis, IN

ABSTRACT
Ensuring the quality of the work developed by third party organizations (TPOs) and in-house works is of paramount importance for pharmaceutical companies since outsourcing projects has become an increasingly popular practice. A common report checking approach is to reproduce the report and check the result manually. This is time consuming and can be inaccurate. This paper introduces a more efficient method to validate reports. This methodology is very helpful, especially for long reports and reports with similar attributes. By following a standard process to reproduce the report information and store results into a SAS® dataset (benchmark dataset) with predefined standards, duplicate coding for the same kind of reports is avoided and efficiency is improved; by using a macro to read and convert the source report (.txt file) to a SAS dataset with predefined standards, then compare this dataset with the benchmark dataset, visual checking is replaced by computer checking therefore efficiency is improved and risk of human errors is minimized. In this paper, we show how to implement this methodology in overall adverse event report; however, it can be generalized to other categories of clinical study reports.

INTRODUCTION
While outsourcing projects becomes a trend, validating the quality of the reports generated by third party organizations (TPOs) becomes a daily task for statisticians in pharmaceutical companies. Usually, statisticians in pharmaceutical companies will randomly select some reports, reproduce some or all the results in the selected reports and check the results manually. This method can be very time consuming and inaccurate, especially when the reports are relatively large such as listings, AE reports and LABs reports.

In this paper we introduce a more efficient report checking method. This new method can be summarized in the following steps:

- Modify and run the code prewritten for a particular kind of report to generate standard SAS analysis datasets;
- Call the macro prewritten for the particular kind of report to conduct statistical analyses using the newly generated standard datasets and store the reproduced results in a SAS dataset with predefined standards, subsequently we refer to this dataset as the “benchmark”;
- Once the benchmark dataset is available, another macro prewritten for this kind of report (the main macro) will be called to convert the source report (.txt file) generated by TPOs to another SAS dataset with predefined standards, which is compared with the benchmark dataset and print out all discrepancies.

Note:
- This method is very helpful for long reports and reports that can be categorized into common “families” of reports (e.g. overall AE report, LABs report and patient demographic report, etc.). For reports that don’t belong to any families, the advantage of using this method may not be very significant.
- In SAS dataset with predefined standards, each row of the dataset represents one number appeared in the source/reproduced report. The dataset has one column for numbers and other columns for information related to each number (e.g. page, row, column, description).

By using this method, time for report checking will be greatly reduced by avoiding visual checking and duplicate coding for the same kind of report; additionally, human errors can be eliminated by using computer checking to replace the visual checking.

At the beginning of this paper, we will show how the new method works by comparing it with the traditional method, and then we will describe this method in detail using the overall AE report as an example; at last a brief conclusion will be given.

OVERVIEW OF THE METHOD
In order to better understand the proposed method, here we describe briefly the traditional report checking method (see the following flowchart): A TPO generates a report according to the report requirements in SAP and sends the report to the pharmaceutical company. After receiving the report, the report checkers in the company will write a
program independently according to the report requirements to reproduce some or all the information in the report and compare with the source report visually.

The checking program can usually be divided into two parts: code used to generate the standard analysis datasets and code used to conduct statistical analyses (e.g., generate counts, means and p-values).

Two major shortcomings for this approach are:
- Visual comparing is time consuming and can be inaccurate.
- Writing a new program for each report is a non-reusable process which causes double coding for the same kind of report.

To solve the above problems, we proposed a more efficient method for report validating. In this new method, a TPO generates a report and sends it to the pharmaceutical company. After receiving the report, the report checkers in the company will use the following standard process to generate a SAS dataset that contains all reproduced information:
- Modify and run the standard code prewritten for this particular kind of report to generate the standard analysis datasets.
- Call a macro prewritten for this kind of report to generate the analysis results (reproduced results) using the newly created analysis datasets and store the results in a SAS dataset with predefined standards (the benchmark).

After obtaining the benchmark dataset that contains all reproduced results, the report checkers will call another macro (the main macro). This macro reads the data from the source report (.txt file) into a SAS dataset with predefined standards; this macro also compares the dataset with the benchmark dataset and prints out all differences. The flowchart of this method is presented below.
By using prewritten code and macro, this new method successfully avoids double coding for the same kind of report; by using the main macro that reads the source report into a SAS dataset and does the comparison, this method avoids the errors caused by visual checking. Efficiency and quality are significantly improved for report checking.

IMPLEMENTATION IN OVERALL AE REPORT
To implement the new method to a particular kind of report, the following four steps are needed.

- Understand the table shell for this particular kind of report.
- Generate prewritten code for this kind of report. This code will be used to generate the standard analysis datasets for this kind of report.
- Generate the prewritten macro for this kind of report. This macro will be used to conduct the required analyses using the newly created analysis datasets and store reproduced analysis results in the benchmark dataset with predefined standards.
- Generate the main macro prewritten for this kind of report. This macro will read data from the source report generated by TPOs into another SAS dataset, then compare this dataset with the benchmark dataset and print out all the differences.

We will use the overall AE report as an example to show how this method works in details.

TABLE SHELL AND EXAMPLE TABLE FOR OVERALL AE REPORT
Overall AE tables are frequently used in clinical study report. In our company, we use the following standard table shell for overall AE tables. The first column of the table shell is "event term". There are at least 3 spaces between this column and the other columns. Following the first column are several columns that contain counts or percentages for each therapy group. The p-values used in this report may be chi-square p-values, Fisher exact p-values or trend test p-values. The three dashed lines divide the report horizontally into four parts: titles, table headers, table contents and footnotes.

<table>
<thead>
<tr>
<th>Overall Occurrence of Adverse Events - Table Shell</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event Term</td>
</tr>
<tr>
<td>------------</td>
</tr>
<tr>
<td>PATIENTS WITH &gt;= 1 TEAE</td>
</tr>
<tr>
<td>PATIENTS WITH NO TEAE</td>
</tr>
<tr>
<td>Oedema peripheral</td>
</tr>
<tr>
<td>Congestive heart failure</td>
</tr>
<tr>
<td>Fluid retention</td>
</tr>
</tbody>
</table>

* This is footnote

The following table is a simulated overall AE table. We will use this as an example in this paper to illustrate the new method.

<table>
<thead>
<tr>
<th>Serious Adverse Events Developing After Treatment (High Level Term MedDRA)</th>
<th>All Randomized Patients</th>
</tr>
</thead>
<tbody>
<tr>
<td>High Level Term</td>
<td>Placebo N=400</td>
</tr>
<tr>
<td>PATIENTS WITH &gt;= 1 SAE</td>
<td>185 (46.2%)</td>
</tr>
<tr>
<td>PATIENTS WITH NO SAE</td>
<td>215 (53.8%)</td>
</tr>
<tr>
<td>Coronary artery disorders NEC</td>
<td>41 (10.3%)</td>
</tr>
<tr>
<td>Heart failures NEC</td>
<td>22 (5.6%)</td>
</tr>
<tr>
<td>Central nervous system hemorrhages and cerebrovascular disorders NEC</td>
<td>14 (3.5%)</td>
</tr>
<tr>
<td>Supraventricular arrhythmias</td>
<td>16 (4.1%)</td>
</tr>
</tbody>
</table>

Abbreviations: N = number of patients, SAE = serious adverse event.
*p-value will not be reported if the total number of events is <= 3.

PREWRITTEN CODE
The prewritten code is used to generate the standard analysis datasets. The standard analysis datasets are datasets that contain only required records and variables for the analysis, so that statistical analyses can be conducted on these datasets directly. The prewritten code is used to perform the following tasks: select all required records, select or create variables needed for the analysis, define parameters that will be used in later analysis, etc. For each
specific report of the same kind, the report checkers only need to modify this code slightly according to the report requirements, so that the standard analysis datasets for the report will be created.

The following program is the prewritten code for the simulated table. Necessary comments are added to explain the code. The highlighted parts are what the report checkers need to modify correspondingly if another overall AE table needs to be checked.

***Part 1. Define the dataset location, create necessary macro variables ***;
/*dose(n)" denotes the count for the nth therapy in the table. For example, "dose3" stands for the column that contains count for the third therapy (total) appeared in the table.*/
libname CORN "D:\MBCM\FINAL\DATA"; *declare dataset location;
%let path=d:\areview\b3spaces.txt; *declare source report location and name;
%let limitation=dose3>3; *report p-values if total number of events > 3;
%let order=dose3; *the output is sorted by total number of events and event term;
%let AEkind=SAE; *this is a SAE report. If it’s TEAE report, change it to TEAE;

*** Part 2. Create analysis dataset PATINFO ***;
/* Select all patients for this analysis from “patinfo” dataset and create a new variable “dose” according to the order in which each therapy appears in the report*/
data patinfo;
  set corn.patinfo;
  if therapy^='NONE'; *select all patients for the analysis;
  if therapy in ('Placebo') then dose=1; *dose=1 for the 1st therapy in report;
  else if therapy in ('30mg QD') then dose=2; *dose=2 for the 2nd therapy in report;
  run;

  /* create records for columns in report with combined therapy (e.g, total), and assign value to variable dose for these records*/
data patinfo;
  set patinfo;
  output;
  if therapy in ('Placebo' '30mg QD') then do;
    dose=3; *dose=3 for the "total"(placebo + 30m QD) therapy in report;
    output;
  end;
  run;

*** Part 3. Create analysis dataset events1 ***;
data events;
  set corn.events;
  where therapy ne 'NONE' and serevnt='Y' and tessevnt = 'Y' and eventtyp = 'EV' and 4<=visit<=16; *Select patients, required AE records and select visits;
  patnew=protocol||patient; *Patnew=study + patient id;
  if therapy in ('Placebo') then dose=1; *dose=1 for the 1st therapy in report;
  else if therapy in ('30mg QD') then dose=2; *dose=2 for the 2nd therapy in report;
  term=HLtclast; *High-level term is the event term used for this report;
  run;

  /*create an extra record for each patient in the previous dataset, so we can calculate the total number of patients who have AE*/
  proc sort data=events out=events;
  by patnew term;
  run;

data events1;
  set events;
  by patnew term;
  output;
  if first.patnew then do;
    term="PATIENTS WITH >= 1 &AEkind.";
    output;
  end;
  run;
/* create records for the column in report with combined therapy (e.g. total), and assign value to variable dose for these records*/
data events1;
  set events1;
  output;
  if therapy in ('Placebo' '30mg QD') then do;
    dose=3; /*dose=3 for the “total”(placebo + 30m QD) therapy in report;*/
    output;
  end;
run;

/*If a patient have the same event more than once, keep one record only*/
proc sort data=events1 nodupkey;
  by dose patnew term;
run;

With two newly created analysis datasets “events1” and “patinfo” ready, it’s time to call the prewritten macro to conduct analysis directly on these datasets.

PREWRITTEN MACRO
The prewritten macro is used to conduct the required statistical analyses (calculate counts, means and p-values) using the newly created analysis datasets and store the reproduced results in a benchmark dataset according to the predefined standards.

The following program is the macro calls (for prewritten macro) for the simulated table. Necessary comments are added. The highlighted parts are what report checker need to modify if another overall AE table need to be checked. We omit the code for the prewritten macro here because it only contains codes for some simple analyses (e.g. counts and p-value calculations). Full code for this macro will be available upon request.

/* We have macros "fisher", "trend" and "pvalue" for fisher exact p-value, trend test p-value and chi-square p-value respectively. They are nested in the prewritten macro AEcheck. So here we need to define the macro calls according to which p-values, if any, are needed for the report. For our sample table, there is only one fisher exact p-value needed and it's for the test of the difference between placebo and 30mg group, so only one row is filled. The parameters "order" and "select" are the same parameters for these three macros. Parameter "order" declares the order of the p-value appeared in the report among all the p-values in the report. For example, the fisher p-value in the sample table is the first p-value among all p-values(for this case, only one p-value), so we put 1 for parameter "order". Parameter "select" identifies all therapies that are involved in the p-value calculation. We use “dose(n) nodose(n)" to indicates the nth therapy in the table. For example, the fisher p-value in the table is for placebo (the fist therapy) vs. 30 mg (the second therapy), so we will put "dose1 nodose1 dose2 nodose2" to parameter select. */
%let call1=%fisher(order=1,select=dose1 nodose1 dose2 nodose2);
%let call2=;
%let call3=;
%let call4=;
%let call5=;

/*Prewritten macro AEchecker: parameter "group" defines the number of therapies in the report and parameter "nopvalue" defines the number of p-values in the report. Because the sample table contains 3 therapy groups(placebo, 30mg QD and total) and one p-value column, we use "group=3" and "nopvalue=1" in the macro call. */
%AEcheck(group=3,nopvalue=1);

THE MAIN MACRO
Assume that the benchmark dataset is available after the above macro calls. The last step of this method is to call the main macro READIN (no parameters are needed for this macro).

The macro READIN works in two steps:
- Read in the TPO source report and save the information in a SAS dataset.
- Compare this dataset with the benchmark and print out discrepancies, if any.
The following description illustrates how the macro reads the source report into a SAS dataset and compares the results with the benchmark dataset.

- Read the source report (a txt file) line by line into a SAS dataset; keep data in the table contents only by discarding titles, table header and footnote; and assign row numbers.
- Separate column1 from the other columns in the table.
- Split composite columns and remove symbols like “%”.
- Split each row such that after split each row contains one number only — the predefined standards.
- Compare with the benchmark dataset and print out the inconsistent records.

In order to illustrate this macro, we present below some important code segments of the main macro for overall AE report (slight changes are needed for other report families). Full code for this macro will be available upon request.

1. Read the txt file into SAS and keep table contents only
First of all, we need to read the source report (.txt file) line by line into a SAS dataset. When symbol “_” which indicates the beginning of a new page is reached, variable “page” will increase by 1 and variable “dashline” will be reset to 0.

```sas
data rawtfl;
  infile "/path" pad;
  input @1 line $136.;
  if index(line,'_') then do;
    page+1;
    dashline=0;
  end;
```

Then we will use the dashed lines to help us identify the table titles, column headers, table contents and footnotes. Whenever a new line with all “-” is reached, the variable “dashline” will increase by 1 and variable “lnum” will be reset to 0. Then we keep those records with dashline=2 only. Because those records with dashline=2 are all records between the second dashed line and the third dashed line. That is table contents. Variable “lnum” represents the row number.

```sas
  if line ne '' and compress(line,'-')=' ' then do;
    lnum=0;
    dashline+1;
    delete;
  end;
  if dashline=2 then do;
    col_value=line;
    lnum+1;
  end;
  else delete;
```

2. Separate column1 from the other columns.
To separate column1 from the other columns, we need to first determine where this column ends using the following code. Because there are at least 3 spaces between column 1 and column 2, the starting point of 3 spaces is the ending place of column 1.

```sas
  do i=1 to 130 ;
    if substr(line,i,3)='   ' then do;
      separator=i;
      leave;
    end;
  end;
```

After having variable “separator” which indicates the ending place of column 1, we use it to separate column1 from the other columns using the following code.

```sas
  lhs=substr(col_value,1,separator-1);
  rhs=substr(col_value,separator+1);
```

3. Split composite columns and remove symbols such as “%”.
All the left columns are numbers which are separated by spaces. So we can select one number at a time whenever spaces are reached. After we get a number, we will use translate function to remove symbol “(%)”.

```sas
```
array col(*) $12 col1-col&colno ;
do i=1 by 1 while (scan(rhs,i,' ')>'');
col{i}=translate(scan(rhs,i,' '),'','(%)');
end;

Note: Macro variable "colon" contains the total number columns in the report except column 1. It’s easy to create and the code is omitted.

4. Transfer the dataset using predefined standards.
The last step is to standardize this dataset using predefined standards. That means to split each row such that after split each row contains one number only.

array col{0:&colno} $80 col0-col&colno ;
do i=0 to dim(col)-1;
  column=i;
  TEAMvalue= col{i};
  output;
end;

Now this dataset is ready to merge with the benchmark dataset. Proc print will be used to print out the inconsistent records.

OUTPUT
The output is as follows:

<table>
<thead>
<tr>
<th>Page</th>
<th>lnum</th>
<th>column</th>
<th>TP0value</th>
<th>TEAMvalue</th>
<th>TEAMvalue</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>17</td>
<td>1</td>
<td>Central nervous system haemorrhages and cereb</td>
<td>Central nervous system haemorrhages and cerebrovascular accidents</td>
<td></td>
</tr>
</tbody>
</table>

This tells us that one discrepancy has been detected in this report. It’s in page 1, row 17 and column 1 of the source report. The reason for this discrepancy is that the high level term is only partially printed out in the source report.

CONCLUSION
In this paper, we present a more efficient report checking method and demonstrated that this method can be implemented on overall AE tables. The authors have also successfully extended this method to other kinds of reports. With this method, substantial gain in time and energy can be achieved by eliminating unnecessary code writing. This method also reduces the possibility of human errors in the validating process by eliminating visual checking.

REFERENCES

ACKNOWLEDGMENTS
We would like to thank Dr. Jyoti Rayamajhi for his valuable suggestions on the new method and his great support.

CONTACT INFORMATION
Your comments and questions are valuable and encouraged. The SAS code can be sent upon request. Contact the author at:

Xuejing (Susan) Mao
Eli Lilly and Company, Lilly Corporate Center,
Indianapolis, Indiana 46285
(317) 433-3626
Email: maoxu@lilly.com

Mario Widel
Eli Lilly and Company, Lilly Corporate Center,
Indianapolis, Indiana 46285
(317) 433-3556
Email: mwidel@lilly.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. © Indicates USA registration. Other brand and product names are trademarks of their respective companies.