PharmaSUG 2014 - Paper IB04

Cover the Basics, Tool for structuring data checking with SAS

Ole Zester, Novo Nordisk, Denmark

ABSTRACT

Data Cleaning and checking are essential parts of the Stat programmer's tasks. We therefore all develop some simple checks that we use for these tasks. This paper presents a program package for an easier way to structure and over viewing data checks. The program is easy to use and it is very easy to modify with new checks. If you have a common data structure (common between tasks) you can use this to cover some of the basic checks, and spend more time to search for the specials. If your data are changing i.e. not final, this program includes a way of remembering which findings you already have identified, to avoid double-querying. The program also includes a dynamic report which shows both the overview (number of findings) and more detailed information about the individual findings.

INTRODUCTION

Data checking is essential for all SAS programmers regardless of industry. As a programmer in the Pharma industry, working with clinical trials, a lot of time is spent with the data cleaning process. SAS programmers therefore do what they always do: make some SAS programs to handle this task. These programs are often made in such a way that they can be used from trial to trial. A way to get more usages of these programs is to combine them into one package with a common outcome reporting, a common monitoring tool.

This paper describes at a high level my version of such a tool. I will not present the total package since I judge it to be very company and data specific. But I hope that the reader will be inspired, and will have an idea of how to make a version of his/her own.

SET UP

In this paper the assumed setup is that I as Statistical Programmer get data delivered from another department (Data Management – DM) for a task. A task can be a Clinical Trial Data have already been exposed for cleaning cycles by others before delivered to me.

When describing this package for other skill areas I sometimes rephrase the term data cleaning to data checking. This is simply to acknowledge that the greater parts of data cleaning are done outside my own area of Biostatistics.

DATA CLEANING IN GENERAL TERMS

Data checks done in Statistics can be ordered into 4 main groups

1. Plots and graphs
2. Statistical Analyzes
3. Trial specific
4. Standard Checks

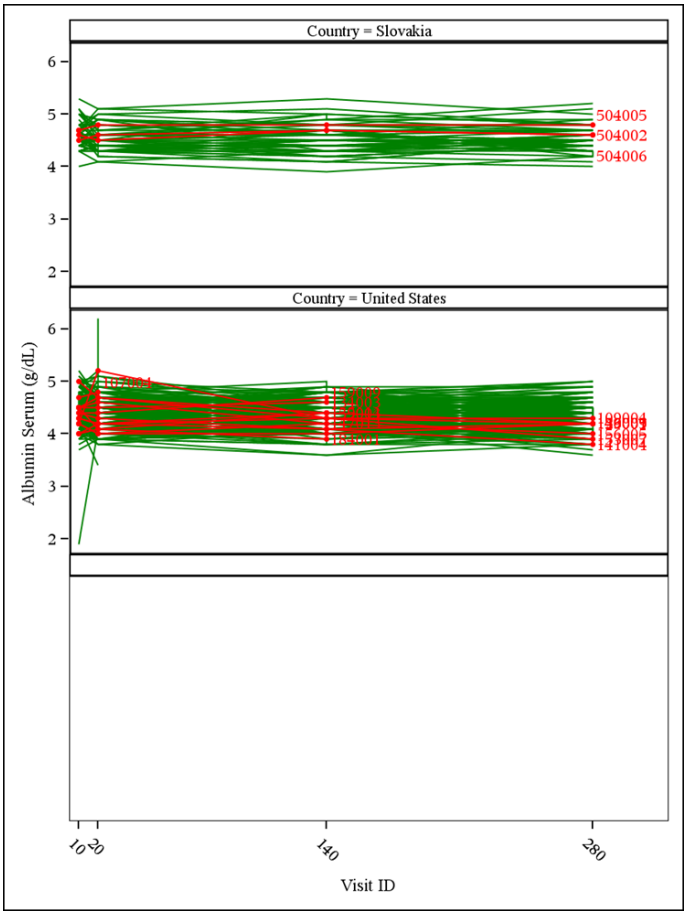
PLOTS and Graphs

This area is not the focus of this paper, so I will just give a brief example.

SAS GRAPHS is an excellent tool for making data checks and with the enhancements made by introduction of f.e. SGPANEL you can create nice plots that are very helpful in the process of data cleaning.-

Below is an example of a scatterplot combined with a simple table showing the lowest and highest values of a labsample. The plots are split in panels by country and the lines in reds indicating that the subject is lacking all values.

A quick glimpse indicates, that we should query about some of the values collected in US a visit 10 and 20.



Subject ID	Visit ID	Finding Value In Standard Unit	Finding Standard Unit	Des
143004	10	1.9	g/dL	Low 10
111001	20	3.4	g/dL	Low 10
107005	140	3.6	g/dL	Low 10
138005	140	3.6	g/dL	Low 10
177001	280	3.6	g/dL	Low 10
185004	280	3.7	g/dL	Low 10
138001	10	3.7	g/dL	Low 10
107005	280	3.8	g/dL	Low 10
107005	20	3.8	g/dL	Low 10
108002	140	3.8	g/dL	Low 10
601006	10	5.2	g/dL	Top 10
302012	140	5.2	g/dL	Top 10
302001	10	5.2	g/dL	Top 10
202007	280	5.2	g/dL	Top 10
308003	10	5.2	g/dL	Top 10
143005	20	5.2	g/dL	Top 10
113002	10	5.2	g/dL	Top 10
506001	10	5.3	g/dL	Top 10
506001	140	5.3	g/dL	Top 10
129002	20	6.2	g/dL	Top 10

Display 1. Example of scatterplot combined with table of high/low values

Statistical analyzes and trial specific checks

These areas are also not covered in this paper. I only mention them to acknowledge that they play a significant role in the data cleaning in clincal trials.

Standard Checks

This paper is dealing with the last group of checks. In the rest of the paper I will describe a program to handle these checks in an effective way.

Standard checks are checks that can be used for several tasks. This of course only makes sense if there is a strong degree of similarity in the data between the tasks. This is more and more the case in the industry with the strong demands for standardized data delivery.

The program consists of a master program (Shell), support macros, metadata and the data check modules themselves. The data checks are grouped together into bundles or small programs. The idea is, that the check programs should be simple programs. When developing such a program, you should be able to concentrate programming the check and not much else.

While executing, the program keeps track of which check programs need to be submitted (metadata driven), which data checks have been executed and which findings each check have revealed.

The SAS code is written in SAS v9.1.3 , and is working under 9.3.

CONCEPT OF THE STANDARD CHECK PROGRAM PACKAGE

The basic idea is that a engine is created that executes and reports a lot of datachecks. The engine is the same but datachecks should be easily added, deleted or modified. Besides reporting, the main shell also has a database for monitoring if findings are correted, which is the ultimate goal of the task of data cleaning.

Thereby you have a flexible program, which should cover you basic needs in datachecking.

CHECK PROGRAMS

Each check is simple to implement since all overhead are taken out and put into the shell program. Basically what it needs is to create a dataset with your findings. The dataset needs a descriptive text variable. The descriptions are, together with a subject identifier, part of a unique key.

Example

```
data miss_site;
    set subj;
    if trl_site_num = .;
    description = 'Subject missing trl_site_num';
    keep subj_id description;
run;

%prepare(xxx,xxx)
```

To make it as simple as possible, the only requirement is, that for each check a dataset is produced and that this dataset contains a variable ‘DESCRIPTION’. Other variables can be included and are at later stage added to a table displaying the findings. The dataset may be empty.

The prepare macro will see to that data regarding the check are added to an overall dataset. Each check is a small block of code, the blocks are then grouped into SAS programs. The benefit of having checks in multiple programs is, that using metadata you are able to control which checks you want to execute at a given execution. This is very efficient during development of checks but also at other times you may wish to look at a subset of checks only.

The prepare macro can look like this

```
%macro prepare(
          datasetname = ,
          id = ,
          desrp = description,
          vis_id = ,
          sobs= Y,
```

```
        subj_id = subj_id );

%if not %sysfunc(exist(&datasetname)) %then %goto error1;

%put Behandling af &datasetname. startet;

%let ii = %SYSFUNC(OPEN(&datasetname.)) ;
%let ik = %SYSFUNC(VARNUM(&ii.,subj_id));

%let ii= %sysfunc(close(&ii.));

proc sql;
    create table nondesp as
    select &desrp
    from &datasetname
    where &desrp = ''
    ;
quit;

%if &sqlobs. gt 0 %then %goto error5;

%numobs(ds=&datasetname.);

data &datasetname._find;
    length subj_id $20;
    retain vis_id . subj_id '' ;
    set &datasetname;
    id = "&id.";
    review ='STAT';
    keep  review &subj_id. &desrp. vis_id id ;
run;


data &datasetname._txt;
    headlink="&id.";
    nu=&numobs.;
    cmv="&datasetname.";
    sobs ="&sobs.";
run;

data headlines;
    set headlines &datasetname._txt;
run;

data findings;
    set findings &datasetname._find;
run;

proc datasets lib=work nolist;
    delete  &datasetname._txt &datasetname._find  nondesp
;
quit;
run;
%return;
```

METADATA

When I designed my own package I found it beneficial to analyze which kind of checks I had. Thereby I had some metadata defined.

Data Check Category	Description
Internal or external	Were the findings just for my own eyes or were they to be reported to others? F.e. I do not want to report errors in enrichments made by my department to be sent to another department (Data management).
Main event dependency	A check depending on a specific event. F.e. I do not want to report to data management that a last drug date is missingwhen the patient has not yet had the Last Visit.
Active	Some checks are part of the package, but not used in some trials. Example: Pregnancy checks on a male only population.
Category	For obvious reason some checks are easier to program together i.e. if you are checking if a value is too high you should probably also check if it is too low.
Sub Program	In which physical program (i.e.: code block) the check is placed. Often this will be the same as the Category.
Description	Small explanation of the check. The idea is that this together with the individual description will explain to others what has been found.
Id	Key

Table 1. Data Check categories.

Example of meta data for individual checks

ID	Description	For Data management report	Batch_id	DOMAIN	Active
AE1010	Either AE start date is missing or stop date appears before start date.	Yes	ADM_AE1	ADM	Yes
AE1020	AE linked to more than one diagnosis	Yes	ADM_AE1	ADM	Yes

Table 2. Example of metadata checks.

Example of meta data for programs

Batch_id	Included
ADM_ae1	no
ADM_ae2	no

Table 3. Example of metadata for programs.

STRUCTURE IN MAIN PROGRAM EXECUTING DATACHECKS

If you use metadata you can make a nice dynamic program. The main Loop in executing looks like this

```
 proc sql noprint;
    create table B_list as
    select distinct batch_id,included,monotonic () as n
    from batch_01 /* dataset containing the metadata regarding subprograms */
    order by n;

    select distinct batch_id into: b_var separated by ' '
    from batch_01;
quit;
run;

%let upper = &sqlobs;

%do i = 1 %to &upper;

    data _null_;
                set b_list (where= (n = &i));
                call symput("batch_now",trim(batch_id));
                call symput("included",trim(included));
    run;

           /* Execute a batch */
    %if &included = yes %then %do;
           %put &batch_now &included i &upper  starts execution;
           %include storage(&batch_now..sas);

/*Storage are the file liberary where the check programs are located.*/
    %end;
    %else  %put &batch_now not submittet due to included=&included;

%end;
%mend;
```

If this program is to be used by multiple users, checking that the sub program actually exists before executing it by the shell program is recommened.

FINDINGS DATABASE

Since each finding has a unique key you can monitor the data findings, using a timestamp. This is very usefull in an ongoing trial, that is, where data keep changing. The trick is to make a unique description variable. So use CATX a lot.

So

```
description = 'Numeric data wrong'
```

is bad,

and

```
description =
'Doses#differs#with#more#than#a#factor#2#(visit/dose#type/dose/prev.dose/prev.visit
)#:#'||trim(vis_id)||'/#'|trim(comp_d)||'''/'||trim(dose)||'#/'||trim(prev_val)||'#
/'||trim(old_vis)||')';

description = translate(compress(description),'','#');
```

is better. Though admittedly it is not easy to read the code.

	id	Sheet	BATCH_id	For_dm	Domain	Full_description	Headid	Status	subj_id	desrp	find_dt
1	AE1070	AE	ADM_ae1	Yes	ADM	AE linked to more t..	207	New	156002	Multiple diagnosi..	11DEC2013
2	AE1070	AE	ADM_ae1	Yes	ADM	AE linked to more t..	207	New	156002	Multiple diagnosi..	11DEC2013
3	AE1070	AE	ADM_ae1	Yes	ADM	AE linked to more t..	207	New	156002	Multiple diagnosi..	11DEC2013

Display 2. Example of entries in findings database

When the program is executed you will have a dataset of new findings plus a dataset with the old ones. Hence it is easy to keep track of the development in data quality.

The variable 'STATUS' can have these values.

Value	
New	New unique key not present in database from previous.
Old	New unique key not present in database.
Solved	Unique key in database not found in new findings.

Table 4. Example of metadata for programs.

RESULT (OUTPUT)

Running the program results in:

- Updated findings table
- Summary table updated, including tables of individual findings.
- Listings

SUMMARY REPORT

Check Report 14:33 Wednesday, December 11, 2013 1

Check Description	Status	Headlink	For DM	Number	Old	New	Solved
Repeats of identical AEs	OK	AE1010	Yes	0	.	.	.
Two or more AEs have identical AE number	OK	AE1020	Yes	0	.	.	.
Either AE start date is missing or stop date appears before start date	OK	AE1030	Yes	0	.	.	.
AE is lacking a low level term ID	Check	AE1050	Yes	9	.	9	.
Data for OUTCOME is missing on AE or missing AE stop date with outcome RECOVERED	Check	AE1060	Yes	6	.	6	.
AE linked to more than one diagnosis	Check	AE1070	Yes	3	.	3	.

Display 3. Summary report

Data for OUTCOME is missing on AE or missing AE stop date with outcome RECOVERED

Description	Subject ID
Missing outcome information (AE no.: 5)	126002
Missing outcome information (AE no.: 6)	126002
Missing outcome information (AE no.: 1)	152005
Missing outcome information (AE no.: 5)	176004
Missing outcome information (AE no.: 6)	185005
Missing outcome information (AE no.: 5)	185010

Back to top

Display 4. Detail report

The report is created as a PDF file, and hyperlinks are inserted between the overview table and the individual tables.

The essential code is given here

```
filename forsym temp ;
filename printsym temp ;

data _null_;
  date_sortable = put(today(),yymmddn8.);
  call symput ('Sdate',date_sortable);
run;

ods escapechar "^";
ods listing close;
ods pdf notoc file ="&folderforpdf./checkreport&Sdate..pdf" /*style=margins*/;
ods noproctitle noresults;
options orientation = landscape;

ODS pdf anchor ="TOP";

data _null_;
  set head1;
   by ind;
   file forsym;
     if first.ind then do;
            put  "PROC FORMAT ;";
         put  "VALUE $link";
     end;
     put  "'" Headlink "' = '#" Headlink "'";
     if last.ind then do;
          put "other = '#TOP'";
         put ";run;";
      end;
   ;
run;

data _null_;
   set head1;
   by ind;
   file printsym;
   put "%" "Print_tabel(…);";
run;

%include forsym;
```

The idea is, that SAS writes 2 temporary files containing SAS code that can later be included and executed.

```
title1'Check Report';


proc print data = head2 noobs label style={foreground=white};
 var  headtxt;
 var status /style= {background=$st. foreground=white};
 Var headlink /style ={URL=$link.};
 var for_dm;
 var nu  old new solved;
run;
ods pdf text="";
ods pdf text="Click on the cells under 'HEADLINK' to navigate to detailed Tables";

title1'Data Transfere Status';
%include printsym;
```

LISTINGS

With the help of metadata it is easy to make listings of findings tailored to the stakeholders needs.

The first problem here is which format to use. Many prefer receiving excel sheets, but as we know, SAS to EXCEL can be a challenge – at least when you are working across platforms (Unix and pc) . I have used CSV mostly because it was fast an easy to develop.

Next step is to determine what to present. Here I think it is a matter of taste and personal preferences of what to include. I recommend always including the following elements

- A row for each finding.
- A unique description for each finding
- A general description for the type of finding.

When designing the table remember that if the listing is a success it may be split into several lists e.g. site and country, so each listing row should have complete information level even if this means that there will be redundant data.

full_description	subj_id	status	Headid	description
AE is lacking a low level term ID	x1	New	205	missing LLT information (AE no.: 4) report. term: 'MISSING'
Subject has discontinue date but no last drug date	x4	New	508	Subject discontinued but no last drug date

Display 5. Example of listing

HOW TO USE THE RESULTS

Before passing the report

When fitting the package to a new trial or task, I recommend having a good look on the result before passing it on. In this business there are also some differences between tasks, and while we assume a common database structure , this differense can mean that some programmed checks do not works as supposed.

So recheck your checks: are these checks relevant in this trial?

Now is the time when you will appreciate that you have put in a feature to include exclude individual checks in your metadata. Or you will invent it.

Always have a look at the reports before sending them.

Just before passing the report

Imagine that you have developed the above system, and now you (hopefully) happily view the result. You have your first report and are eager to show to your data delivery department.

What to do.

I just email the result you think. Data Management will be so happy with how thorough I have been.

Well my guess will be, no, they are not as happy as you may imagine. In fact it may be that they are a little upset especially if you present a lot of findings to them. Nobody likes to receive a lot of findings in their work in the inbox.

My advice is to put a lot of effort in presenting the report beforehand.

When I start working with a new Data Manager, I usually invite in for a meeting for presenting my package and the concept. In connection with this I create a first report. This first report can be used as basis for a discussion . In the meeting, I always ask for feedback regarding the checks. This serves 2 purposes. I get the checks further enhanced and usually I get buy in from the Data Manager. In fact they often find the concept very good especially they idea that they will get many of my findings in a structured way and I can ensure that I will not present the same finding twice.

The essential part here is that the package report should be seen as a tool in your communication

OTHER POINTS

Data checks in a package like this are always retrospective. The point is that the checks are often based on problems you have previously experienced.

Collect good checks by inviting colleagues and other stakeholders to contribute.

CONCLUSION

It will take some time to set up a package like this and get it running. I think you will find the time investment worth it, though. When the package is up and running, maintenance and development is straightforward.

CONTACT INFORMATION

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

Name: Ole Zester
Enterprise: Novo Nordisk
Address: Vandtaarnsvej 83 , 2860 Soeborg
City, State ZIP: Denmark
Work Phone: +45 3079 4616
E-mail: Ozes@novonordisk.com