PharmaSUG 2012 – Paper DS03

SAS File Design: Guidelines for Statisticians and Data Managers

Douglas Zirbel, Independent Consultant, St Louis, Missouri

ABSTRACT

Although it might at first seem harmless, poor or careless design of clinical trials data files can result in big problems later. Misunderstandings, delays, and even erroneous data are potential consequences to the clinical research organization, the pharmaceutical company, and ultimately to patients. This paper reviews a number of mistakes made in analysis/submission file specifications, then provides guidelines and a checklist for intelligent file design, whether CDISC or other formats.

INTRODUCTION

The primary purpose of clinical trials datasets is to serve both the pharmaceutical company's statisticians as well as reviewers at the U.S. Food and Drug Administration and other regulatory bodies. However, because data is the foundation of clinical research, any missteps at this stage impacts everything that follows. The problems may be caught in time, but the effort spent to deal with them may prove to be an additional and unexpected burden on the staff. As the saying goes, "an ounce of prevention is worth a pound of cure". Can anything be done to prevent file-level problems?

Let us first examine the broader clinical trials environment. The process of developing and submitting molecular products for approval is one of focused custom work aimed at releasing a product with a limited lifespan to the public. What is done in this phase of the business must be both reliable and flexible, two challenging requirements which do not always readily coexist.

SAS is well-suited to the dynamic, time-is-money-driven pace of these efforts. It has broad capabilities. It could be thought of as a Relational Database Management System with Structured Query Language (SQL) functionality. Moreover, it is augmented with a powerful data manipulation language (Base SAS, SAS/STAT, SAS/GRAPH, etc) that goes far beyond that of SQL. It runs on mainframes, Linux/Unix servers, and PCs. SAS is flexible.

Furthermore, its performance is not limited by a computer's memory in the way that many other statistical packages are. It has a simple, character / numeric data foundation, many simple-to-use "Procedures" which automate much of the programming, and its basic "DATA Step" programming language statements have a short learning curve. As a platform it has been validated. It is welcomed by the FDA. And it is widely familiar to statisticians, researchers, and programmers both in and outside of the pharmaceutical industry. Thus we have reliability.

In many companies, constructing a database or a set of data files using SAS may not require going through a formal IT bureaucracy of Database Administrators, Security procedures, and Change Control paperwork. Technically, SAS makes creation of databases and files very simple. Yet the informality and simplicity can also pose weaknesses. Hastily-designed files may include subtle flaws in their structure which may not be corrected by others prior to going into production. Even in formal environments where data and programming peer reviews are required by policy, they are not always rigorously enforced.

BASICS OF FILE DESIGN

Fortunately, file and database design is an established discipline with its own body of knowledge. There are standard database frameworks – for example, hierarchical databases, object-oriented databases, relational databases, and others, all with generally-accepted rules for their architectures. For many reasons, the relational database model is well-suited to clinical trials data, not the least of which is its accessibility by SAS and the widely-used SQL programming language.

In practice, there are a number of file structures in use today, and it is worth a few moments to review them before going further.

- "**Flat files**" – these are simple stand-alone files in which all data is stored in one set of rows. They can be text, comma-delimited, Excel, SAS, or other storage formats.

- **Multiple record-type files** – these are flat files with an added dimension: different rows have different columns. In other words, there is a column (often the first) that indicates the row type. For example, a row_type of "H" might mean that the row is a header row with columns for Patient ID, Visit ID, and a clinical test's start and stop times, while row_type of "T" might contain columns for each of the tests performed during those start and stop times.

- **Relational tables** – better known as "databases". These too, are essentially simple flat files, but designed ("normalized") with thought given to relationships between the data that the files represent. They are typically stored in a Relational Database Management System (RMDS) such as SQLServer, Oracle, DB2, and others, but can also be easily stored in SAS.

- **XML files** – Extended Markup Language files are often similar to multiple record-type files, but in these files each "column" is given a "tag" naming the "column". If you have ever seen a raw HTML file you have a good idea of what XML files look like.

- **CDISC files** – relatively new to submissions data, but now accepted for FDA submissions, these are actually a set of file layout standards. They are essentially denormalized relational tables designed specifically for clinical trials.

Study data should be relational data. Whether it's a company's legacy dataset standards, a statistician's own design, or a more widely-adopted standard, clinical trials data should be easily handled by computer programs and easily understood by those viewing the data.

In this discussion we will cover both the simplest level of data – the single "flat file" design, in which all data for a topic is stored in one file with various columns (or "fields", or "variables") – and also present some basic rules for creating "normalized" database files.

EXAMPLES OF REAL-LIFE PROBLEMS IN SUBMISSION FILE DESIGN

We will begin with an easy scenario. Suppose a trial consists of a series of visits in which patients are first given an electrocardiogram (ECG), then administered a treatment drug or placebo, and later given one or more ECGs. The single-submission-file design contains columns for protocol, site number, patient number, visit, ECG date, ECG time, pre-or-post-dose indicator, followed by Heart Rate, QRS interval, QT interval, QTCB, and so on. To keep the illustration uncomplicated, we will limit our ECG measurement to Heart Rate.

PROTOCOL	SITE_NUM	PAT_NUM	VIS_NUM	ECG_DATE	ECG_TIME	DOSE_IND	HEARTRT
ABC01	15	1	2	15Mar2012	10:13	PRE	67

FDA SAS Transport file requirements. Although this is not the focus of the file design problems in this paper, some of the FDA's SAS submission file guidelines are summarized here. In the examples that follow, we will usually restrict variable names to the old SAS Version 5 limit of 8 characters.

- Files must be in SAS Version 5 or 6 XPORT storage format prior to sending to the FDA
- Variable names 8 characters or less
- Variable labels 32 characters or less
- File names 8 characters or less
- Note – working SAS files not sent for regulatory review can be less restrictive, taking advantage of longer labels, file names, and variable names.

HOW (NOT) TO NORMALIZE YOUR DATA

No unique "primary key". In a trial in which the main endpoint is an electrocardiogram measurement, it is expected that a patient has only one record in the file for each ECG measurement. Nevertheless, one or more patients must repeat the ECG test because of misplaced ECG leads. Furthermore, the ECG_TIME column happens to record only hours and minutes. The risk is an inflated number of patient visits, as well as the introduction of bad data. The solution is to plan for this kind of possibility in the file design stage by ensuring that the "seconds" component of ECG_TIME is captured. A normalized file or data table should have a column or set of columns for which there are no possible duplicate records – a primary key.

Repeating columns or variables in the file (First Normal Form). A study is planned to have up to two 250-byte Comment fields for each ECG, to be filled in by a cardiologist during review. The ECG file thus contains the ECG file's columns followed by COMMENT1 and COMMENT2. The problem appears when a cardiologist determines that a particular ECG merits more comments than 500 bytes will allow. Faced with this limitation, the cardiologist must then decide whether to abbreviate words, to leave out some comments, or to create a dummy ECG row with identical values to the prior row in order to take advantage of two more Comment variables. The better solution is to remove the Comment fields entirely from the file and create a new file for them. The new file will contain the same primary key columns: PROTOCOL, SITE_NUM, PAT_NUM, VISIT, VIS_DATE, ECG_DATE, ECG_TIME, DOSE_IND, followed by COMMENT and COMNTNUM. In this way the cardiologist can write as many comment rows as appropriate, which can always be joined or merged to the ECG file by PROTOCOL, SITE_NUM, PAT_NUM, VISIT, VIS_DATE, ECG_DATE, ECG_TIME. This is the "First Rule of Normalization": remove repeating columns, and be sure that each row has the same number of variables.

A variable is tied to only part of the primary key of the record (Second Normal Form). In an effort to be clear to others looking at the file, the designer includes not only SITE_NUM, but SITENAME. The record will show SITE_NUM of "001" and SITENAME of "Anchorage"; SITE_NUM "002" and SITENAME of "Seattle", and so on. However, the primary key variables in the earlier illustration are PROTOCOL, SITE_NUM, PAT_NUM, VISIT, ECG_DATE, and ECG_TIME. SITENAME is tied only to SITE_NUM.

The problem results in additional data that someone must enter, introducing more risk of data discrepancies. The solution is to create a separate lookup table containing SITE_NUM and SITENAME. These can always be joined or merged to the ECG file by SITE_NUM. Once the file is in First Normal Form, this is the "Second Rule of Normalization": there should be no variables that are tied to a subset of the primary key.

A variable or variables are not directly tied to whole primary key (Third Normal Form). This ECG file has primary key columns and the Heart Rate column shown above, but has an additional column, HRTRTCAT. This is "Heart Rate Category", which classifies the Heart Rate speed as slow, medium, or fast (bradycardia, normal, tachycardia) as follows: < 60 "Bradycardia"; 60 <= 99 "Normal"; >= 100 Tachycardia.

The problem is that HRTRTCAT is not really about the whole primary key; it is about the non-key variable HEARTRT. The problem and the solution are very similar to the issue found in the Second Normal Form example – create a separate lookup table for Heart Rate and its category HRTRTCAT. This is the "Third Rule of Normalization": every variable that is not one of the primary key variables should be directly related to that entire primary key.

Please note that with the three or four normalization rules above (and there are others beyond this) it is quite common to normalize the data during design, then to "denormalize" it if somewhat-denormalized files offer less complicated programming or better performance.

OTHER DATA DESIGN ISSUES

Ambiguous variable or file names. The FDA requires SAS submission files sent for review to be in the older SAS Version 5 transport format, which limits variable names to 8 characters or less. In this example, the statistician who designed the SAS submission file layout needed variables to capture summary values for a one-hour continuous ECG monitor: Minimum Heart Rate, Mean Heart Rate, and Maximum Heart Rate. He chose MNHR, MHR, and MXHR. What's the problem? An outside observer, such as a SAS programmer or an FDA reviewer, is confused. The solution is to think in terms of the outside observer, and use enough characters to intelligently abbreviate. Use names such as MINHRATE, MEANHRTE, MAXHRATE.

Where two or more files are involved, inconsistent variable names for same data. A study includes hourly ECG measurements and a once-per-visit drug dose. The ECG data is in one file and the Dose data is in another, later to be merged into a single SAS file. The ECG file has a variable, TMPT ("time point", storing a SAS date-time value), while the Dose file has a variable called DATETIME. The problem is that it raises questions in the outside observer's mind. If variables are really capturing the same data, the solution is to use the same variable names.

Inconsistent formats for similar variables. In one study there is a Visit timepoint as well as a Dose timepoint; that is, on a particular visit day the patient arrives at the site and stays for 6 hours, having an ECG every hour as well as the treatment drug at hour 2. The variables ECG_DATE, ECG_TIME, DSE_DATE, DSE_TIME are used, but the SAS formats are DATE., TIME., DATE9., and TIME8., respectively, resulting in values of "15MAR12", while "15MAR2012", "10:13" and "10:13:06". Again, needless questions are raised in the mind of the reviewer. The solution is to provide consistent formats for all dates, times and other pairs of variables that are similar.

Limiting SAS labels to 8 char or less. Because of the FDA SAS variable name limit, the longer SAS variable labels are used to clarify the meaning of the variables. In this case, the statistician specifies labels such as “Min HR”, “Mean HR”, Max HR”, - or worse – simply re-uses the variable names themselves in the labels: “MNHR”, MHR”, “MXHR”. There is no virtue in saving either keystrokes or disk space in this situation. The principle to follow is, “How would an outside observer see this?” Or, said another way, “Would you have to explain it to them?” A better label for the MIN_HRT might be “Minimum Heart Rate in 1 hour”.

No SAS file label. A programmer creates a SAS file from an ECG device text file and saves it, naming it “ecg15mar”. A week later he extracts the latest ECG file and saves it as “ecg22mar”. Later that day he makes a change to the SAS extract program and re-runs, using “ecg22ma2”. The next week he makes another change to the SAS program and runs that week’s extract, and so on. Two months later a question arises about differences between the data delivered on March 15 and the data delivered on March 22. This problem is actually two problems in one. The first is that no source code control system was used, and the second is that he does not know why he has two files from March 22. The solution is twofold – first, use source code control (CVS, TFS, GIT, among others) – but this is beyond the scope and deserves its own paper. Second, use a SAS file label to your advantage. For example,

```
%let file_label = Used explicit date and heart rate infmts;

data ecglib.ecg22ma2 (label=”&file_label”);
    infile . . .;
run;
```

When the question arises as to why the two files are different, the answer (the label text) will appear in the Proc CONTENTS.

No expected-visit-sequence number. A trial’s visit schedule lists visits in a character variable in this order: “SCREEN”, “DAY2”, “DAY5”, “DAY10”, “DAY20”. A patient is not supposed to receive the drug at the third visit, DAY5, but the data says that he or she does receive it at DAY5. So questions arise. Is this really the DAY5 visit? Was it indeed the DAY5 visit but the drug was administered incorrectly? Or was it perhaps an unscheduled visit after the DAY5 visit? A programmer subsequently validating the data will need to compare the sequence of visits to visit date and time, but without an expected visit sequence number this will not be possible. These kinds of issues must be resolved, and a Visit Schedule including the expected sequence number should be established. For reasons shown in the next point, an expected visit sequence number should be a SAS numeric variable.

Sequence or quantity numbers in character format. In this problem, drug dosage amounts are stored in a character variable. Later, the statistician will want to show the effects of increasing dosage in a chart. However, the lowest dose level is “5mg”, the next dose is “10mg”, and the last dose is “15mg”. Because “5” comes later in file sorting than “1”, when the file is sorted, the records will appear in this order – “10mg”, “15mg”, **“5mg”**, which is clearly not intended. Or the visit sequence variable contains “1”, “2”, “3”, “4”, “5”, “6”, “7”, “8”, “9”, “10”, “11” etc. This will sort to “1”, **“10”, “11”,** “2”, “3”, “4”, “5”, “6”, “7”, “8”, “9” – the same kind of problem. The solution is to make all sequences and quantities numeric.

No Cohort identifier variable. For a variety of reasons, a trial may extend beyond its plan, and it may make sense for the company to add another dosing regimen. New patients or even previous patients are recruited, but because the file has no provision for this, the Data Management team is instructed to treat all visits at sites 10, 17, and 23 which occur after February 23 as part of a second group. More sites and dates are added, and the programming team begins to make errors because of the complexity. The solution, again, is to plan ahead. Include a Cohort or Treatment Group ID column if there is any possibility that this could occur.

File layout without reference to Case Report Form, device fields or laboratory results. The file layout may be specified in a table of variables in the Statistical Analysis Plan or some other specification document but there is no column indicating the source of the variables. Or there is a column for the source of the variables, but it has been left blank. The problem here is a break in the audit trail, which is a necessary component of Good Clinical Practices, and the solution is to provide that source in the document.

CDISC SDTM DATA

The Clinical Data Interchange Standards Consortium (CDISC) data models for file design (such as the Study Data Tabulation Model and the Analysis Data Model) do not strictly follow the rules of relational data. They may have initially normalized their design but have subsequently denormalized them. Their strength lies in their standardization, and they are accepted for FDA submissions. They thus simplify some of the effort in designing datasets. The models are not comprehensive and often need slight modifications or supplementation. The principles above still apply.

THE CHECKLIST

SUBMISSION FILE DESIGN CHECKLIST

FDA SAS SUBMISSION FILE RULES

- ☐ Is the SAS file in SAS Version 5 or 6 XPORT format?
- ☐ Is the SAS file name 8 characters or less?
- ☐ Are the SAS variable names 8 characters or less?
- ☐ Are SAS variable Labels 32 characters or less?

NORMALIZATION

- ☐ Does the file have a set of variables that always uniquely identifies each observation?
- ☐ Is that “primary key” clearly stated in the file’s documentation?
- ☐ Have all “repeating” columns (e.g., COMMENT1, COMMENT2, etc) been removed?
- ☐ Are there any variables (which are not part of the primary key) tied to only part of the primary key?
- ☐ Are there any variables (which are not part of the primary key) that are not tied to the whole primary key, i.e., tied to other non-primary-key variables?

OTHER ISSUES

- ☐ Is the SAS file name unambiguous?
- ☐ Are variable names unambiguous?
- ☐ Are identical-data variable names, data types, and formats consistent from one file to another?
- ☐ Are formats consistent for similar variables?
- ☐ Do variable labels clarify variable name meaning?
- ☐ Do the SAS files have labels that clarify their contents?
- ☐ Does the file contain a (numeric) expected visit sequence number?
- ☐ Are all sequence and quantity variables available as SAS numeric variables?
- ☐ If there is a possibility of additions to the visit schedule, or additional treatment groups or cohorts, does the file contain a cohort or treatment group variable?
- ☐ Does the file layout specification include each variable’s source?

CONCLUSION

SAS clinical trial file design can be informal and easily accomplished. To ensure a problem-free design, follow known requirements (such as the FDA guidelines). Anticipate changes during the trial and accommodate them in the file design. Ask, “how would an outside observer see this?” Ask others, such as the clinical trials programmers, to review and even to test the design. Maintain an audit trail between the source (device, lab test, questionnaire, CRF, etc) and the submission file variables. Make use of the checklist.

REFERENCES

Chapple, Mike, Database Normalization Basics, http://databases.about.com/od/specificproducts/a/normalization.htm, retrieved 31mar2012

Codd, E.F., A Relational Model of Data for Large Shared Data Banks, in *Communications of the ACM*, June 1970; http://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf, retrieved 31mar2012.

Kent, William, A Simple Guide to Five Normal Forms in Relational Database Theory, in *Communications of the ACM 26(2)*, Feb 1983; http://bkent.net/Doc/simple5.htm, retrieved 17mar2012.

SAS Institute, FDA and SAS Technology, http://www.sas.com/industry/government/fda/index.html, retrieved 31mar2012.

U.S. Food and Drug Administration, Guidance for Industry: Providing Regulatory Submissions in Electronic Format - General Considerations, 1999, http://www.fda.gov/downloads/Drugs/GuidanceComplianceRegulatoryInformation/Guidances/ucm072390.pdf, retrieved 31mar2012.

U.S. Food and Drug Administration, Providing Regulatory Submissions in Electronic Format – NDAs, http://www.fda.gov/downloads/Drugs/DevelopmentApprovalProcess/FormsSubmissionRequirements/ElectronicSubmissions/UCM163187.pdf, 1999, retrieved 17mar2011.

U.S. Food and Drug Administration, Study Data Specifications. Current version: 1.5.1, 2010, http://www.fda.gov/downloads/ForIndustry/DataStandards/StudyDataStandards/UCM199599.pdf retrieved 31mar2012.

CONTACT INFORMATION

Your comments, questions, feedback, and other examples of file design problems are appreciated. They may be incorporated into future versions of this paper. Contact the author at:

Name: Douglas Zirbel, MBA, SAS Certified Advanced Programmer
Enterprise: Independent Consultant
City, State ZIP: St Louis, MO, USA
E-mail: doug_zirbel@hotmail.com

sas Certified Advanced Programmer for SAS® 9