

Interfering with Data Management by the Statistician

Jim Groeneveld, OCS Consulting, Rosmalen, the Netherlands.

A. ABSTRACT

Within clinical research studies a statistician generally is part of the project team and responsible for the statistical analysis and derived conclusions. In that position he is generally involved with the purely statistical and closely related aspects of the study. This of course is the most important part for both (the course of) the study and the statistician, but it may be very worthwhile, also for both (the course of) the study and the statistician, to have him involved in many other, if not all parts of a study.

There are various reasons to stress the need for a statistician in all phases of clinical research, from protocol development to final reporting. It is the main task of a statistician to perform statistical analysis and to draw sound conclusions from the analysis results, which are to be reported finally. In order to perform the analysis the statistician has to use statistical software either interactively or via a command language (prepare the analysis in the form of batch programs). Data to be analyzed often is stored in a database system with combinations of more or less complex data structures, often logically organized from the point of view of data management or IT, not necessarily from the viewpoint of statistical analysis. It is very much recommended to let the statistician also play a leading role during the development of (the structure of) the database in order to avoid accessibility problems at a later stage. And to stay ahead of coding problems, ambiguous question (variable) names and illogical answers (data) it is very desirable to also cooperate in the CRF development phase. His involvement in the whole clinical process should preferably be documented in the (statistical) SOP.

Pitfalls and drawbacks of data management issues, like bad CRF and database designs are presented. Advantages of the involvement of a statistician at all stages of clinical research studies to avoid potential problems and to create better structured CRFs and databases to improve alignment between data management and statistical analysis are stressed.

B. INTRODUCTION

1. OVERVIEW

In clinical research the statistician's task obviously is to perform statistical analysis, to draw conclusions from the results of the analysis and to justify those in a report. While preparing for all that he writes a statistical analysis plan (SAP), writes or reviews the statistical part of the study protocol, performs sample size calculations and randomizes patients across experimental conditions. He receives data to perform the analysis on from clinical researchers or data managers.

A sometimes underestimated or even neglected task of the statistician is to be involved in the process of building and structuring the database and to participate in designing the CRF. The goal of such concern is to guard data structure and consistency, and to prevent possible additional and unnecessary data manipulation later to correct inconsistencies and to restructure data for analysis. This way the statistician can positively influence the quality of the data he will get and the amount of preparatory work during the statistical analysis.

It is the purpose of this paper to emphasize the necessity for the involvement of the statistician in all phases of research studies, in particular database design.

2. MAIN STATISTICIAN'S TASK

In clinical research studies a statistician generally performs the statistical analysis. He justifies the findings in a statistical report and/or a statistical section in the final clinical report or at least reviews the statistical section in the clinical report during the report writing process. The statistical analysis is planned and described by the statistician in a statistical analysis plan (SAP) and/or the statistical section in the study protocol.

While performing statistical analysis the statistician uses statistical software, like SAS®. The data is available in electronic form, obtained from clinical researchers or data managers. The statistician (or a SAS programmer) also writes preceding programs for the necessary data manipulation like dataset linking, class construction and other derivations.

3. PRE-STATISTICAL TASKS, QUALITY OF RESEARCH

Next to the statistical process a statistician also should participate in other, preceding processes within a clinical study. This is to ensure the justification, consistent description and course of the study. For example, while reviewing the protocol and possibly writing the statistical section of it, the statistician guards the quality of the objectives and hypotheses stated, which form the basis for the analysis to be done at a later stage as outlined in the SAP. He should also support the researchers in randomizing patients in experimental (and control) groups and calculating the minimal sample size for the study, which is needed to obtain sufficient power for the (primary) statistical tests. In order to facilitate correspondence between the protocol and the SAP it may be recommended to write, review and finalize them simultaneously.

4. DATA MANAGEMENT

While being involved in earlier stages of a study the statistician actively participates in the study team. He is the one who will finally receive data to analyse and instead of passively awaiting the data and its structure and observing what he depends on, he rather should actively guard the data quality during the early stages of the study: database building

PhUSE 2006

(structure) and CRF design. In doing so he can avoid unnecessary complex data structures, limit inconsistencies within the data and thus later limit complex data manipulation (in SAS). The minimum he should do is to audit the data management process by reviewing the data management plan (DMP) and the data validation plan (DVP), especially checking the logic of the validation algorithms. Special attention should be paid to coding and code lists.

C. DATABASE STRUCTURE

Databases, their structures, if built without the intervention of a statistician, may be built in a less optimal way than actually would be preferred. E.g. if a database is being structured by a data manager or clinical researcher the structure may be based too much on the physical structure of the CRFs, the simplicity of the database building process and the ease of data entry. If the database is being built by an IT specialist or database administrator the way the data is stored may play a large role. All this may result in structures and data fields which are not directly usable to a statistician, not most optimally designed. To a statistician it is very important that the data are logically and unambiguously stored, structured and as such extractable according to the relations between the data elements. Restructuring the data (with SAS) requires extra ad hoc work and time for the statistician or SAS programmer. This can be avoided.

Practical examples of unnecessary complex data structures are firstly the so called “empty cells” (consequential standard holes), secondly the two “question and answer” variables (containing variable indications and their values or scores) and thirdly multiple answers per cell.

1. EMPTY CELLS

Empty cells look like to be stemming from interleaved data sub files (CRF pages) with different variables. The records only contain data from a single sub file (e.g. CRF page). So where the variables of one page are filled the other ones are empty and vice versa. Example 1a shows two CRF pages where one of them contains spontaneous answers to a question and the other one the asked (presented) answers in case there was no spontaneous answer.

Example 1a. “Interleaved” data with consequential empty cells (holes)

| Sub-ject | Visit nr | Page nr | Form | Instr man. | Oral instr | Trial proc. | Sub-ject | Other | Spec. other | De-vice | Spont Y/N | Spont compl | Asked compl | Asked Y/N |
|----------|----------|---------|-----------|------------|------------|-------------|----------|-------|-------------|---------|-----------|-------------|-------------|-----------|
| 1 | 2 | 14 | SPONTAN_2 | . | . | . | . | . | . | . | N | . | . | . |
| 1 | 2 | 14 | ASKING_2 | . | . | . | . | . | . | . | . | . | . | N |
| 1 | 3 | 21 | SPONTAN_2 | . | . | . | . | . | . | . | N | . | . | . |
| 1 | 3 | 21 | ASKING_2 | N | N | N | N | N | . | Y | . | . | text1 | Y |
| 2 | 2 | 14 | SPONTAN_2 | . | . | . | . | . | . | . | Y | textA | . | . |
| 2 | 2 | 14 | ASKING_2 | N | N | N | N | N | . | Y | . | . | text2 | Y |
| 2 | 3 | 21 | SPONTAN_2 | . | . | . | . | . | . | . | Y | textB | . | . |
| 2 | 3 | 21 | ASKING_2 | N | N | N | N | N | . | Y | . | . | . | N |
| 3 | 2 | 14 | SPONTAN_2 | . | . | . | . | . | . | . | Y | textC | . | . |
| 3 | 2 | 14 | ASKING_2 | N | N | N | N | N | . | Y | . | . | . | N |
| 3 | 3 | 21 | SPONTAN_2 | . | . | . | . | . | . | . | N | . | . | . |
| 3 | 3 | 21 | ASKING_2 | . | . | . | . | . | . | . | . | . | . | N |

They should be split into two (or more) different data files (one per CRF page) (Example 1b) or they could be compressed into a single file with single records, where the variables of all original pages are filled (Example 1c). Example 1b shows the two different files with different variables (except for the ID variables) of which one shows the spontaneous answers and the other one the asked ones. Example 1c shows the merged result of both files, which has all information stored in all variables without unnecessary empty cells. Remaining empty cells are either really missing or not applicable.

Example 1b. “Interleaved” data solution 1: 2 data files

Data file 1

| Sub-ject | Visit nr | Page nr | Form | Spont Y/N | Spont compl |
|----------|----------|---------|-----------|-----------|-------------|
| 1 | 2 | 14 | SPONTAN_2 | N | . |
| 1 | 3 | 21 | SPONTAN_2 | N | . |
| 2 | 2 | 14 | SPONTAN_2 | Y | textA |
| 2 | 3 | 21 | SPONTAN_2 | Y | textB |
| 3 | 2 | 14 | SPONTAN_2 | Y | textC |
| 3 | 3 | 21 | SPONTAN_2 | N | . |

Data file 2

| Sub-ject | Visit nr | Page nr | Form | Instr man. | Oral instr | Trial proc. | Sub-ject | Other | Spec. other | De-vice | Asked compl | Asked Y/N |
|----------|----------|---------|----------|------------|------------|-------------|----------|-------|-------------|---------|-------------|-----------|
| 1 | 2 | 14 | ASKING_2 | . | . | . | . | . | . | . | . | N |
| 1 | 3 | 21 | ASKING_2 | N | N | N | N | N | . | Y | text1 | Y |
| 2 | 2 | 14 | ASKING_2 | N | N | N | N | N | . | Y | text2 | Y |
| 2 | 3 | 21 | ASKING_2 | N | N | N | N | N | . | Y | . | N |
| 3 | 2 | 14 | ASKING_2 | N | N | N | N | N | . | Y | . | N |
| 3 | 3 | 21 | ASKING_2 | . | . | . | . | . | . | . | . | N |

PhUSE 2006

Example 1c. “Interleaved” data solution 2: 1 data file

| Sub-ject | Visit nr | Page nr | Form | Instr man. | Oral instr | Trial proc. | Sub-ject | Other | Spec. other | De-vice | Spont Y/N | Spont compl | Asked compl | Asked Y/N |
|----------|----------|---------|------|------------|------------|-------------|----------|-------|-------------|---------|-----------|-------------|-------------|-----------|
| 1 | 2 | 14 | both | . | . | . | . | . | . | . | N | . | . | N |
| 1 | 3 | 21 | both | N | N | N | N | N | . | Y | N | . | text1 | Y |
| 2 | 2 | 14 | both | N | N | N | N | N | . | Y | Y | textA | text2 | Y |
| 2 | 3 | 21 | both | N | N | N | N | N | . | Y | Y | textB | . | N |
| 3 | 2 | 14 | both | N | N | N | N | N | . | Y | Y | textC | . | N |
| 3 | 3 | 21 | both | . | . | . | . | . | . | . | N | . | . | N |

To solve the above issue ad hoc using SAS a SAS macro **MPR2SR** is available on request. This restructures the data from Multiple Partial Records To Single Records, thus in agreement with the example 1c.

2. QUESTION AND ANSWER

The second example of unnecessary complex data structures, the two “question and answer” variables, is shown in Example 2. Amidst the usual columns (variables) with rows of values there are two columns, here called Question and Answer, which actually represents a 90° rotated data sub matrix. The column Question contains the questions, variables actually, while the column Answer contains the answers, the values to those questions. This structure has emerged because a data manager did not have sufficient time to extend the structure of a database. He did not realize that his choice would later cause even more time loss and frustration for the statistician. Though all the information is there it is quite a job to process such a sub matrix into a normal one, to be merged with the rest of the data file (using SAS).

Example 2. “Question and Answer” data mixed with normally structured data

| ID | Question | Answer | Visitdate | Visitnumber |
|----|------------|-----------|-----------|-------------|
| 1 | Sex | female | . | . |
| 1 | DoB | 01jan1960 | . | . |
| 1 | Length | 1.80 m | 28oct2005 | 3 |
| 1 | Weight | 75 kg | . | . |
| 1 | Medication | Yes | . | 3 |
| 2 | DoB | 31dec1959 | 29oct2005 | 5 |
| 2 | Weight | 77 | 29oct2005 | 5 |
| 2 | Sex | male | 29oct2005 | 5 |
| 2 | Length | 1.88 | 29oct2005 | 4 |
| 3 | Gender | F | 01oct2005 | 2 |

Also note the missing values or the repetition of the values for the “normal” columns, and the sometimes even discrepant information in the multiple records. This was due to entering each record manually. Besides, both the Question values, the actual variable names, and the Answer values were not coded consistently. Further note the arbitrary order and sometimes the incompleteness of the Question values. A case like this may need additional data checking. This is a really ugly structure.

To restructure such data in SAS can not be programmed generally using a macro. It needs dedicated code, dependent on the contents of the data, to solve this issue.

3. MULTIPLE ANSWERS

The third example of unnecessary complex data structures, the multiple answers per cell, is shown in Example 3. Each cell, expected to contain only one value, sometimes appears to contain more than one value, separated by newline characters. Actually these multiple values should be stored either in multiple records or multiple, additional, repeated variables. The number and the order of the multiple values within a record should be consistent to interpret corresponding values unambiguously. However, it does not always appear to be consistent.

Note the 5-digit numbers in the data columns. These are SAS dates as the number of days since 1 Jan 1960 and originally were dates specified as text and converted somewhere in the conversion process. The multiple dates in the cells were not recognized as such by the conversion process and left as they were. Trying to solve such a structure needs quite some (SAS macro) programming both in case the multiple values per cell already are present as such in a SAS dataset or whether they are still only available in ascii data to read. Then it depends very much on what kind of ascii data it concerns: varying length records with comma delimited values, whether quoted or not, or fixed length records (with newlines as record delimiters).

Another cause of such a structure can often be observed with (long) comment fields and is due to the data entry typist, entering data with Excel or data management software. Normally when the typist has to enter long comments (or values) he should enter them without breaking the text. The result may be that the text scrolls off the text window to the left and may partly not be visible anymore. The typist then may wrap the data by one or more newlines (the Enter key) in order to continue on a new line in the cell while keeping the previous one visible. Excel allows entering a newline in a cell; at least some data management software allows that too, while it might be better not to allow that.

Anyway, after the data has been saved to an ascii file with either comma delimited varying length records or fixed format records it seems that extra records, extra lines have emerged due to the extra newline characters. It is very difficult to read

PhUSE 2006

such a structure correctly into a SAS dataset. The best solution is to try to initially correct the raw ascii data file. For both ascii file types algorithms can be developed to recognize and remove the unwanted newline characters. For the fixed format ascii file type a SAS macro **RemBugNL** is available on request, that automatically replaces redundant newlines with one or more spaces in order to preserve the constant fixed record length.

Example 3. Multiple Answers in cells, separated by newline characters

| | | | | | | | |
|--|---|---|---|---|-------------|----------------------------|---|
| higher temperatures febrilias | 2 | 2 | 2 | 2 | 37797 | 37821 | 2 |
| NYHA indicated for BIV; still ongoing | 2 | 1 | 2 | 2 | 02 Dec 2004 | | 2 |
| failed BIV implant endovaz. | 2 | 2 | 1 | 2 | 02 Dec 2004 | | 2 |
| failed BIV implant II. transeptal | 2 | 2 | 1 | 2 | 06 Jan 2005 | 03 Dec 2004 10 Jan 2005 | 2 |
| a lead dislodgement and reposition | 1 | 2 | 2 | 1 | 15 Mar 2004 | 30 Apr 2004 | 2 |
| hospitalisation due to holter monitoring | 1 | 2 | 4 | 1 | 15 Feb 2004 | 18 Feb 2004 | 2 |
| RESKG -> nothing | 2 | 2 | 4 | 2 | 38292 | 38293 | 2 |
| a lead | 1 | 2 | 2 | 1 | 12 Nov 2003 | 21 Nov 2003 | 2 |
| TIA with left sided sym. | 1 | 2 | 2 | 2 | 20 Dec 2003 | 20 Dec 2003 | 2 |
| | 2 | 2 | 2 | 2 | 03 Jun 2004 | 03 Jun 2004 | 2 |
| pericardial effusion | 2 | 1 | 2 | 1 | 38162 | | 2 |

D. CRF DESIGN

1. SIMPLICITY

Inconsistencies in data, like those presented, often are induced by a complex CRF design, a complex data structure and inconsistent naming, labelling and coding. A CRF design can be improved by avoiding too many dependencies: skipping unnecessary questions and answers, limiting conditional routing and level of sub questions and coding only final answers, no intermediate ones.

Example 4a. Inefficient CRF design

10. Medications

10.1 Is the patient currently on any cardiovascular medications?

- No, skip to section 11
- Yes, please specify below:
 - Anti-coagulation
 - Anti-arrhythmics Class I
 - Anti-arrhythmics Class II
 - Beta-blockers
 - ACE inhibitors or AT antagonist
 - Calcium Antagonist
 - Other medication

Example 4b. More efficient CRF design

10. Medications

10.1 Is the patient currently on any cardiovascular medications?

- No, skip to section 11
- Anti-coagulation (Warfarin)
- Platelet aggregation inhibitor (Aspirin, ASA)
- Anti-arrhythmics Class I
- Anti-arrhythmics Class II (Beta-blockers)
- Anti-arrhythmics Class III (Amiodarone)
- Anti-arrhythmics Class III (Sotalol)
- Anti-arrhythmics Class IV (Calcium Antagonist)
- Digitalis
- ACE inhibitors or AT antagonist
- Other medication, please specify: _____

Example 4a shows a somewhat inefficient part of a CRF. The answer on question 10.1 initially can be either "Yes" or "No". If it is "No" then the question is done, if it is "Yes" then one or more of the medications have to be checked. This is a rather simple example; sometimes nesting of sub questions goes deeper and more frequent. The disadvantage is that data management has to check and validate the answers and that a statistician also has to deal with those answer levels. Chances on discrepancies are real. If "No" has been answered the medication items should not be checked, but may nevertheless appear to be so. Contrary to that "Yes" may have been answered without any medication specification, which is as invalid.

PhUSE 2006

Example 4b shows the more efficient alternative of this part of the CRF. Note the absence of the answer “Yes” and “Other” coding; only its specification can be coded. It is clear that at least one (and possibly more) answer should be given and coded. The only possible discrepancy is both an answer “No” and one or more of the medications. The advantage is that all answers are coded in one variable, if only one alternative can be checked, or in a set of related multiple response variables if more answers can apply at a time. Coding errors are thus less likely.

2. NAMING, LABELING AND CODING

More inconsistencies regarding CRFs and database structures occur with naming, labelling and coding. There still often appear to be unnecessary and confusing differences both between and within studies. Standardization is very much needed. Even between and within CRFs there may be coding discrepancies. And of course there are the inevitable, but unwanted differences as demanded by different EDC software systems and their SAS export facilities.

Inconsistent naming between CRFs or CRF pages (and related data files) may emerge as either different names for the same (ID) variables (of sometimes even different type) or the reverse, i.e. equal variable names for different variables, different entities actually. This is especially annoying if it is initially unknown and invisible because one, as a statistician, was not involved in the database building process. During a SAS merge of two or more of such datasets in the last case the variable's values of the last specified data set are being kept, but its values in the other data sets are lost. Renaming variables in such a case is a must.

These equal variable names in different datasets may have different labels. Even if they indicate the same entity and thus should be equal, during a merge or other combination in SAS all but one label will be lost. The reverse, different variable names with same labels, also occurs, even within the same dataset. This obscures the meaning of the variables.

Partially depending on the database structure and partially on the EDC system and its SAS export facility it may appear that first of all quite some multiple equally named code lists (SAS formats) are being exported. E.g. while “Yes” and “No” may have been coded as 1 and 2 they have an associated code list, which is being exported for every variable needing it, while only one would be sufficient. Though SAS notes the multiple format specification it luckily doesn't do any harm. Happily too, those multiple code lists appear to be identical; it would cause a large problem if they wouldn't be identical.

Secondly a variation of that, also very much depending on the EDC system and its SAS export, is the occurrence of many (conceptually) identical code lists with different names, where only one would be sufficient as well. No problem of course (for SAS), but overdone and sometimes confusing.

And thirdly another variation, strictly speaking not erroneous, but very confusing (for the statistician) is the occurrence of similar code lists with identical categories, but different codes. An example is an occasion where at one instance Yes/No has been coded as e.g. 1/0 and in another instance as 1/2 or even 2/1. Such occurrences should be avoided, but if occurring should be recoded.

E. REDUNDANT DATA

Quite often superfluous information has been stored in the data files. Data may have been included more than once in a dataset (or the datasets belonging to a study). Apart from wasting disk space and increasing processing time this also induces uncertainty about the data quality if the different duplicates appear not to be entirely equal. Duplicate information should be avoided as much as possible. For example subject classification (and demographic) information should only be kept in one panel or SAS dataset, that can be linked to, merged with other data using the minimal key variable Subject_ID. Categories should preferably not be coded as their texts, but with numeric values with an associated format containing the concerning texts. This also rules out spelling differences (that often is a problem with comment fields). Text fields should be avoided as much as possible unless inevitable, like names (if any), comments, descriptions.

MedDRA codes, such as Lowest Level Term (LLT) codes sometimes appear to be available in quadruple, each with its different associated SAS format: one to the Preferred Term (PT), one to the High Level Term (HLT), one (with a multilabel SAS format) to the High Level Group Term (HLGT) and one (also with a multilabel SAS format) to the (Primary) System Organ Class ((P)SOC). Just one LLT code would satisfy, with which the four various formats can be associated at wish. Coding the MedDRA field descriptions as text variables in study data is very much redundant. If really needed as values they can be PUT from a SAS format to a SAS VIEW.

F. SAS PROGRAMMING

Unnecessary complex data manipulation by the statistician or SAS programmer should be limited as much as possible. This is one of the purposes of the active concern of the statistician with the whole process of gathering data. He should avoid solving problems caused by/in other departments which can be solved much better within the concerning department.

During SAS programming, while investigating the obtained data, some additional annoying problems are:

- numeric values coded as character strings, possibly in order to support non-digit characters, like 'N/A';
- (or the reverse:) large numeric values, but without any real value meaning, like IDs and telephone numbers, coded as numerical values instead of character strings;
- routing indications may unnecessarily be coded and thus also require validation. Exclude them from the database.

G. REQUIREMENTS TO SAS EXPORT

EDC systems, Data Management software generally support exporting data to SAS. The most common way is by generating a generally useful ascii data file, for instance comma delimited, and additionally a specific SAS program to read that data file and to generate one or more SAS datasets of it. Based on experience with a few of those systems, none of which appeared to be perfect, I would like to propose improving requirements for both the (EDC) software system itself and its use by Data Management (DM). Minimal requirements for the SAS export contents and quality should be:

- one SAS dataset per CRF or CRF page (or other source) to avoid "interleaving" as in example 1 (DM issue);
- a **consistent** within-dataset **structure** (with multiple records or repeated variables) (DM issue);
- without **bugs**, no SAS (syntax) errors, no unexpected issues like outlined below (EDC issue).

The SAS export should have a **consistent structure** within each dataset:

- either all single records per subject (with single variables);
- possibly with repeated measurements (at different times), as additional variables within the same record;
- or multiple records per subject, repetitions for the same variable in additional records.

Avoid a mix of these structures within single datasets (see example 1).

SAS export (severe) **bug** examples are:

- CT: in certain circumstances exported field lengths above 197 might cause severe errors in the export generated SAS program reading the data. Solution: limit exported text lengths to 197 (more than enough and most of the time empty);
- DF: the exported SAS program overwrites same named DF fields in data files (by page), which are being merged by the DF generated SAS program, causing to loose valuable information.

Solution: adapt the generated SAS program manually in advance to correct these problems before reading the data.

Further requirements to SAS exports are:

- required **adaptation** of the generated program should be kept to a **minimum** (see below) (EDC issue);
- prevention of non-unique variable names in dataset, also if caused by the data manager (DM issue);
- prevention of export of illegal dates with date informats resulting in missing values (EDC and DM issue).

Minimizing the **adaptation** of the generated SAS program from the SAS export concerns local specification of:

- drives, directories, library names for data and formats (possibly as macro variables, etc.);
- permanent dataset labels and their locations (libnames);
- permanent SAS (in)formats catalog to create at a specific location.

Though it can not be expected from EDC systems to support all this automatically there certainly is room for improvement.

H. KNOW YOUR DATA

If the statistician receives rather unknown data it is very much recommended to try to get insight into the data, about what is in there, how structured, coded and related, before doing the actual statistical analysis. He should make tables of the datasets and the variables therein; common or equally named variables should be evidently shown. Next, and with the aid of the CRF or other data sheet, he should make several elementary, simple tables (cross tabulations) of the main variables, like sex, age, groups, visits, and the like. Emerging frequencies should be used as reference checks for later analysis. Illegal and unexpected values would show too and provide information about the weaknesses of the underlying database.

In this way the statistician will become acquainted with the data and its structure. The data structure should be logical, not so much related to the original CRF structure, edges and layout. Intermediate answers (like in example 4) are irrelevant. The information should have been converted from a physical CRF structure into a logical data structure.

I. CONCLUSIONS

While minimizing the problems that can occur during the process of data collection, the statistician can focus on the statistical analysis without bothering about the data structure and its consistency. There is still enough preparatory data manipulation to do, like dataset linking, class construction and other (complex) derivations.

Next to his purely statistical task of writing the SAP or the statistical part of the protocol, determining the sample size, analysis and reporting, it is very much worthwhile for the statistician (and the study) to be involved in the whole path of the data collection. His involvement in the whole clinical process should preferably be documented in the statistical SOP.

CONTACT INFORMATION

Y. (Jim) Groeneveld
OCS Consulting Benelux
PO BOX 490
5240 AL ROSMALEN
THE NETHERLANDS
Office: +31 (0)73 523 6000
Fax.: +31 (0)73 523 6600
JimG@OCS-Consulting.com
www.ocs-consulting.com
home.hccnet.nl/jim.groeneveld