PharmaSUG 2015 - Paper IB07

The Disposition Table

Make it Easy

Endri Endri, ProXpress Clinical Research GmbH, Berlin, Germany
Benedikt Trenggono, ProXpress Clinical Research GmbH, Berlin, Germany

ABSTRACT

The disposition table is one of the most complex tables in statistical programming, because it provides an overview of clinical data and events from numerous sources. Disposition tables include (but are not limited to): overviews of patients who completed the various study epochs, and the number of adverse events and serious adverse events related to study medication. Some disposition tables require the merging of several datasets, and there are often other complexities, such as the requirement to apply imputation rules, which must be taken into account. These complexities lead to an increased workload and more time spent on programming the table. This presentation explains the steps required to create a disposition table using SAS as a labor and time-saving tool not just to produce the table but to generate the table-creation code automatically based on the data specifications as well as the data itself.. The idea behind it is the retrieval of content and structures of SAS datasets using simple text comparison algorithms. For a better understanding of the algorithms, the presentation will include simple examples with figures and explanations, as well as step-by-step instructions for automated programming with practical examples of disposition tables. The purpose of the paper is to present ideas on how the manual programming process can be simplified by the use of artificial intelligence, the aim being to make the life of the statistical analyst easier, and to provide more time for validation and review with less time spent on writing programs.

INTRODUCTION

A disposition table (or a so-called “overview table“) gives an oversight of a clinical study. In certain cases the programming process (using SAS) of these kinds of tables belongs to one of the most complicated ones and therefore leads to an increasing amount of work and time to understand the clinical data first and generate the disposition tables afterwards.

Even though CDISC SDTM is defined as the data standard within clinical research the clinical data (especially structures) between many studies are very different. It mostly depends on the clinical study design and the definition of SDTM structures.

1. DISPOSITION TABLE

The effort to generate these tables therefore depends on the simplicity or complexity of the disposition table’s content and may include data collection of various sources, datasets or even data transformation / manipulation. The following is a classic example of a disposition table (incl. the description of how many patients are involved within the study (screened), randomized, drop-outs (incl. drop-out reasons):

Table xx.x.xxxx Disposition of patients

	Treat A	Treat B	Total
Number of patients	X (xxx.x)	X (xxx.x)	X (xxx.x)
Enrolled / signed informed consent [N (%)]			
N	X (xxx.x)	X (xxx.x)	X (xxx.x)
No	X (xxx.x)	X (xxx.x)	X (xxx.x)
Yes	X (xxx.x)	X (xxx.x)	X (xxx.x)
Entered / randomised [N (%)]			
N	X (xxx.x)	X (xxx.x)	X (xxx.x)
No	X (xxx.x)	X (xxx.x)	X (xxx.x)
Yes	X (xxx.x)	X (xxx.x)	X (xxx.x)
Termination of trial medication [N (%)]			
N	X (xxx.x)	X (xxx.x)	X (xxx.x)
Progressive disease	X (xxx.x)	X (xxx.x)	X (xxx.x)
Adverse Event	X (xxx.x)	X (xxx.x)	X (xxx.x)
Protocol Deviation	X (xxx.x)	X (xxx.x)	X (xxx.x)
Lost to follow-up	X (xxx.x)	X (xxx.x)	X (xxx.x)
Refused cont. medicat	X (xxx.x)	X (xxx.x)	X (xxx.x)
Other	X (xxx.x)	X (xxx.x)	X (xxx.x)
Missing	X (xxx.x)	X (xxx.x)	X (xxx.x)

Figure 1. Disposition of patients

Table xx.x.xxxx Analysis Sets (All Randomized Subjects)

Analysis Set	Treat A (N=xx)	Treat B (N=xx)	Total (N=xx)
Randomized	xx (xx.x%)	xx (xx.x%)	xx (xx.x%)
Safety Analysis Set[1]	xx (xx.x%)	xx (xx.x%)	xx (xx.x%)
Full Analysis Set[2]	xx (xx.x%)	xx (xx.x%)	xx (xx.x%)
Per Protocol Set[3]	xx (xx.x%)	xx (xx.x%)	xx (xx.x%)
Pharmacokinetics Analysis Set[4]	xx (xx.x%)	xx (xx.x%)	xx (xx.x%)
Pharmacodynamic Analysis Set[5]	xx (xx.x%)	xx (xx.x%)	xx (xx.x%)

[1] All randomized subjects who took at least one dose of study drug.
[2] All randomized subjects who took at least one dose of study drug and had at least one efficacy endpoint evaluation.
[3] xxxx
[4] xxxx
[5] xxxx

Figure 2. Analysis Sets

Another classical example is an oversight of Adverse Events (AE), as presented below:

Table xx.x.xxxx Overall Summary of Treatment-Emergent Adverse Events (Safety Population)

	Treat A (N = x) n(%)
Treatment emergent adverse event (TEAEs)	x (xx.x%)
TEAEs Related to Treat A	x (xx.x%)
TEAEs, Grade >= 3	x (xx.x%)
TEAEs Related to Treat A, Grade >= 3	x (xx.x%)
Serious TEAEs	x (xx.x%)
Serious TEAEs with fatal outcome	x (xx.x%)
TEAEs leading to an action regarding the dose (dose reduced/increased/interrupted)	x (xx.x%)
TEAEs Related to Treat A leading to an action regarding the dose (dose reduced/increased/interrupted)	x (xx.x%)
TEAEs leading to drug withdrawn/study termination	x (xx.x%)

Figure 3. Overall Summary of Adverse Events

2. CURRENT PROGRAMMING PROCESSES

The quality of clinical data, illustrated statistical tables and time and effort within the clinical research are very important. Therefore different programming methods have been used up to now. A classical method which is often used within clinical research is the “Code Template” method to accelerate programming processes.

The usage of these methods is a common practice and provides the advantage to work swiftly and therefore save time, but the usage also provides a substantial disadvantage: it’s susceptibility to errors. A programmer has to understand exactly how to use the program code and if necessary how to modify it.

3. ARTIFICIAL INTELLIGENCE

Another possible solution is to develop a type of artificial intelligence, which should support the coding process. Artificial intelligence in this regard means that the system should have the intelligence to analyze text (e.g. Mock-up Tables, Statistical Analysis Plan etc.) and therefore writes SAS programs based on the clinical data for evaluation purpose.

The complexity of this system depends on the challenges it has to cope with. The two following requirements are being defined.

- Independent of structures and content of clinical data
- Free text in Mock-Up Tables / Statistical Analysis Plan

These increase the complexity of the system with artificial intelligence. Therefore the system will be categorized into 3 process groups:

- Extract Information (EI)
  This process extracts clinical data independently from its structure and its content and saves it in a metadata dataset to enable the later search function
- Text Analysis (TA)
  Here the Statistical Analysis Plan text as well as Mock-Up Tables will be scanned and compared with the metadata dataset from the previous EI step. The details of this comparison method will be explained more in the next chapter.
- Code Generator
  The result of this process are SAS-programs, which were generated by the system. Therefore programmers can continue working with these programs for further processing (e.g. Validation).

3.1 EXTRACT INFORMATION (EI)

As illustrated before the system should be able to extract the clinical data independently. At first this might seem complicated, but its realization is in fact trivial, because SAS provides functions / procedures which can be used to solve this problem.

The following are basic approaches to help solving this problem:

SASHELP.VCOLUMN / SASHELP.VTABLE

The usage of SASHELP.VCOLUMN helps to extract the following information:

- All datasets within a SAS Library
- All variable-names of each dataset within a SAS Library
- Attributes of a variable: variable-length, -type, -format and –label

PROC FORMAT

Formats based on specific studies can be saved in a dataset by using the procedure FORMAT.

PROC SORT NODUPKEY

For a better and more complete capture of data it is possible to save “unique” text variables. This procedure is highly recommendable for big datasets to provide a more swiftly search of certain information in later processes.

PROC FREQ

The final step of the Extract Information (EI) process includes the creation of the Entity Relationship Model (ERM) which will be done automatically by PROC FREQ. The Entity Relationship Model (ERM) gives information about how datasets are connected to each other. The trick is to find the unique variable within datasets.

In the end of the Extract Information (EI) process all information will be saved in a “metadata“ dataset.

	Member Name	Column Name	Column Label	Column Type	Column Length	Column Format	Column Number in Table
102	CONC	LLQ	LLQ of conc1 or conc2	num	8	-	12
103	CONC	LLQ_DPM	LLQ in dpm	num	8	-	10
104	CONC	MATRIX	matrix	char	20	-	2
105	CONC	PCTPTNUM	Planned Time Point Number	num	8	-	6
106	CONC	SAMCOM		char	42	-	14
107	CONC	SAMPNO	sample number	num	8	-	5
108	CONC	STUDYID	Study Identifier	char	40	-	23
109	CONC	SUBJECT	subject (N)	num	8	-	4
110	CONC	TRTMNT		char	2	-	1
111	CONC	UNIT	unit (cold)	char	8	-	9
113	CONC	WEIGHT	sample weight (g)	num	8	-	17
114	CONC	WEIGHT_C	sample weight (g)	char	8	-	16
115	CY	CYDTC	Date/Time of Collection	char	19	-	7
116	CY	CYSEQ	Sequence Number	num	8	-	4
117	CY	CYSPECIFY	Specification of Reason	char	40	-	8
119	CY	STUDYID	Study Identifier	char	40	-	1
121	CY	VISIT	Visit Name	char	20	-	6
122	CY	VISITNUM	Visit Number	num	8		5

Figure 4. Modification of SASHELP.VCOLUMN

	Memb Name	Column Name	value	Column Label
1	AE	AEACN	DOSE INTERRUPTED	Action Taken with Study Treatment
2	AE	AEACN	DOSE NOT CHANGED	Action Taken with Study Treatment
3	AE	AEACN	DOSE REDUCED	Action Taken with Study Treatment
4	AE	AEACN	DRUG WITHDRAWN	Action Taken with Study Treatment
5	AE	AEACN	NOT APPLICABLE	Action Taken with Study Treatment
6	AE	AEACNOTH	CONCOMITANT MEDICATION	Other Action Taken
7	AE	AEACNOTH	CONCOMITANT PROCEDURE	Other Action Taken
8	AE	AEACNOTH	NONE	Other Action Taken
9	AE	AECONTRT	N	Concomitant or Additional Trtmnt Given
10	AE	AECONTRT	Y	Concomitant or Additional Trtmnt Given
11	AE	AEENDTC		End Date/Time of Adverse Event
12	AE	AEENDTC	2012-12-02	End Date/Time of Adverse Event
13	AE	AEENDTC	2012-12-05	End Date/Time of Adverse Event

Figure 5: Output of PROC SORT NODUPKEY

Comparing the Figure 3, 4 and 5, it is possible to find any variable and any text within the clinical database!

There are also other methods to identify clinical data regardless of their structures, but the previous methods are supposed to indicate that an identification of clinical datasets and their contents regardless of their database structures should not pose a big problem.

3.2 TEXT-ANALYSIS (TA)

3.2.1 Similarity - Evaluation

In order to find the texts from the given mock-up tables in the clinical database (as extracted in previous process), the similarity calculation process need to be done.

In the following a common practice to similarity evaluation of two words will be described:

- Fuzzy string method
  In computer science, the approximate string matching, also called fuzzy string searching, includes a class of string-matching-algorithms, which are supposed to search or find a certain string within a longer string or a text.
- Levenshtein distance Method
  In computer science, the Levenshtein distance (also called edit distance) between two strings is the minimum number of single-character edits (i.e. insertions, deletions or substitutions) required to change one word into the other. It is named after Vladimir Levenshtein, who introduced this distance in 1965.
- Cosine Similarity Method
  Cosine similarity is a measure of similarity between two vectors of an inner product space that measures the cosine of the angle between them. The cosine of 0° is 1, and it is less than 1 for any other angle. Therefore it is a measurement to indicate if two vectors have approximately the same orientation or not.

The three methods mentioned above are a common practice to similarity-evaluations of two words. The similarity of words in these methods will be given in percentage terms. These classical methods have been developed for common cases only and therefore are not suitable for a concrete comparison within the area of clinical research. For example the terms "**Normal**" and "**Abnormal**" have a similarity of more than 70%!

Based on the results an own calculation-algorithm for the evaluation of similarities of words with the help of SAS macros has been developed. Thereby, merely a composition of loops, a single-letter comparison of strings and a certain weight on similarity had been installed within the calculation. The calculation is a self-development and therefore it is possible to vary / proportion the emphasis of the same string. Hence, the above mentioned comparison between "**Normal**" and "**Abnormal**" will now be evaluated with a similarity of **less than 50%**.

3.2.2 Internal Dictionary

An internal dictionary has been developed in the system to identify synonyms and to translate study specific definitions. For example, "subject", "patient" could have the same meaning.

This internal dictionary has two types of keywords which will be differentiated as followed:

- System-Keywords
  These include general SAS procedures, functions, syntax and definitions of standard macros (division/company specific), which will be embedded into the system. Other keywords and synonyms, which are often used in the SAP (e.g. "only", "at least") can also be defined as keywords
- Content-Keywords
  All keywords which are not categorized as system-keywords are content-keywords (e.g. study specific content of clinical data)

An internal dictionary is a method to quicken the determination of definitions and keywords. The more studies the system has evaluated the more intelligent and faster it processes the next time, since a memory method has been implemented. This memory functions saves all words and definitions as well as the solution method into the system.

3.2.3 Layout Analysis

To correctly write the programs for calculating the disposition table or other statistical tables the following aspects with the support of the text analysis have to be considered.

- Limitation of data selection
  Some tables need only specific data for the evaluation process, e.g. limitation of population
- PAGE variable / BY variable
  Statistical tables can be displayed on several pages (regarding variables), as described in the SAP, e.g. analysis of specific subgroups
- Frequency/ Incidence / Descriptive Statistics
  Depending on the tables, these will be differentiated in 3 different groups.

- Frequency: describes the frequency of events
- Incidence: describes the number of patients with specific events
- Descriptive Statistics calculates N, MIN, MEDIAN, MAX of variables with numeric formats.

3.2.4 Examples of Text Analysis

A “safety population” for example is defined in the SAP as followed:

“. . . if he/she is *randomized* to a treatment group and has taken at least one unit of the study medication and has post-treatment safety data available”

In this example CDISC SDTM datasets will be used. The following shows the solution:

- “randomized” : DS dataset
- “treatment group” : DM dataset(ARMCD variable)
- “at least” : System-keyword
  defined as „HAVING MIN (##var##) >= ##“.
- “study medication” : EX dataset
- “post-treatment safety data” : VS dataset

Analogously the disposition tables as displayed in figure 1 – 3 will be analyzed and translated in SAS programs. Below is an example of written SAS program (using SQL) of the overall summary of adverse events, as displayed in figure 3 (first 5 overview events only), using the text analysis process.

```
/*************************************************************************
 * Project           : xxxxxxxxxxxxxxxx
 * Program name      :
 */     %progstart(program = t_15030101_adae_sum.sas)
/* Author            : Endri Endri (EE) - ProXpress Clinical Research GmbH
 * Date/version      : 2015-03-26 18:44:02 /v 1.0
 * Enviroment        : SAS 9.2 Windows
 * Purpose           : Table 15.3.1.1 Overall Summary of Treatment-Emergent Adverse
Events (Safety population)
 * Note              :
 *************************************************************************/


DATA temp_select;
  SET test.ae;
RUN;


/*************************************************************************
 * Counting
 *************************************************************************/
PROC SQL;
  CREATE TABLE adae_100_calc (WHERE =(NOT MISSING(trt01a))) AS

   /* 1. Treatment emergent adverse event (TEAEs) */
   SELECT DISTINCT 1 AS var1_id
        , 'Treatment emergent adverse event (TEAEs)' AS var1
        , SUM(temp) AS count
        , trt01a
   FROM (SELECT DISTINCT usubjid, trt01a, MAX(IFN(aetrtfl = 'Y', 1, 0)) AS temp
          FROM  temp_select
          GROUP BY usubjid
        )
   GROUP BY trt01a
```

```
   /* 2. TEAEs Related to Treat A*/
   OUTER UNION CORR
   SELECT DISTINCT 2 AS var1_id
        , 'TEAEs Related to Treat A' AS var1
        , SUM(temp) AS count
        , trt01a
   FROM (SELECT DISTINCT usubjid, trt01a, MAX(IFN(aetrtfl = 'Y' AND aerel IN
('RELATED' ''), 1, 0)) AS temp
          FROM temp_select
          GROUP BY usubjid
        )
   GROUP BY trt01a

   /* 3. TEAEs, Grade >= 3*/
   OUTER UNION CORR
   SELECT DISTINCT 3 AS var1_id
        , 'TEAEs, Grade >= 3' AS var1
        , SUM(temp) AS count
        , trt01a
   FROM (SELECT DISTINCT usubjid, trt01a, MAX(IFN(aetrtfl = 'Y' AND INPUT(aetoxgr,
best.) >= 3, 1, 0)) AS temp
          FROM temp_select
          GROUP BY usubjid
        )
   GROUP BY trt01a

   /* 4. TEAEs Related to Treat A, Grade >= 3 */
   OUTER UNION CORR
   SELECT DISTINCT 4 AS var1_id
        , 'TEAEs Related to Treat A, Grade >= 3' AS var1
        , SUM(temp) AS count
        , trt01a
   FROM (SELECT DISTINCT usubjid, trt01a, MAX(IFN(aetrtfl = 'Y' AND INPUT(aetoxgr,
best.) >= 3 AND aerel IN ('RELATED' ''), 1, 0)) AS temp
          FROM temp_select
          GROUP BY usubjid
        )
   GROUP BY trt01a

   /* 5. Serious TEAEs */
   OUTER UNION CORR
   SELECT DISTINCT 5 AS var1_id
        , 'Serious TEAEs' AS var1
        , SUM(temp) AS count
        , trt01a
   FROM (SELECT DISTINCT usubjid, trt01a, MAX(IFN(aetrtfl = 'Y' AND aeser = 'Y', 1,
0)) AS temp
          FROM temp_select
          GROUP BY usubjid
        )
   GROUP BY trt01a
  ;
QUIT;
PROC SORT; BY trt01a; RUN;
```

3.3 CODE GENERATOR (CG)

SAS code generated by the process can be defined to adhere to standards and SOPs, make appropriate use of SAS procedures and functions (as these procedures and functions are already included in the internal dictionary) as well as available standard macros which carry out the necessary calculations and derivations and generate output such as frequencies, summaries, incidences, be well structured and clearly commented, contain the required header information, etc., and at the end be as close to validation-ready as possible.

Using the artificial intelligence processing, the SQL program code for displayed example (overview of adverse events) was produced within few minutes.

The code generation method was presented at the 2011 PhUSE Conference held in Brighton, UK (DH07 – Endri Endri; Rowland Hale: Much ADaM about Nothing - a PROC Away in a Day). Further information about the SAS program generator engine can be found in this paper, which describes in detail how ADaM data sets can be programmed using Microsoft Excel.

With auto-text functionality (##text##) a code generator engine is able to generate the pre-defined program header, use existing standard macros, utilise SAS functions and procedures, etc.

4. CONCLUSION

Since we have new standards using CDISC Structure, the tasks of programming in clinical research becomes more and more. There is a lot of new requirement from the authorities. On the other side, the performance and the quality of programming must be constantly improved.

The aim of this exercise was to achieve, as far as possible, a fully automated study analysis – from SAP, TFL specifications and mock tables to final analysis – by automatically capturing the required information from these documents and applying it to an SDTM clinical database through the use of complex algorithms.

The advantages by using such artificial intelligence programming are:

- More Capacity of “Programmers“
- Less Human Errors
- Time Benefits
- “Self-learning”
- Standard Layout of Programs
- Several Methods in Text Analysis and Several Technique in Output SAS Programs
- Pre-written programs are editable, in case the programs are not correct.

Furthermore, the fact that the system comprises of a number of individual tasks means that this is not an “all or nothing” process and it can be used to automate one or more of the following tasks, as required:

- Simple assignment
- Mapping and data derivation
- Query and data validation
- Table, Figure and Listing programming

So the fundamental question is whether it is possible to effect a comprehensive “detection” of data and requirements and automatically gain an understanding of the clinical data regardless of data structure to such an extent that a proper study analysis in full accordance with information provided in the SAP can be carried out.

And the answer now is “**yes, it can!”**

CONTACT INFORMATION <HEADING 1>

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

Name:	Endri Endri	Benedikt Trenggono
Enterprise:	ProXpress Clinical Research GmbH	ProXpress Clinical Research GmbH
Address:	Berlin	Berlin
City, State ZIP:	Germany	Germany
Work Phone:	+49 30 120 59581	+49 30 120 59581
E-mail:	endri@proxpress-clinical.com	benedikt.trenggono@proxpress-clinical.com
Web:	http://www.proxpress-clinical.com	http://www.proxpress-clinical.com