Protecting Patient Data Privacy

Creating data that protects patient data privacy and enables meaningful analyses for outside researchers

Gregory Steffens

Novartis, Associate Director, Programming

NOVARTIS

The Regulatory Background

Why we need to do this

- EMA requirement to provide clinical trial data to outside researchers to perform independent analyses
  - Supports EMA objective to make decision making transparent (Policy 0043)
  - Policy 0070 Publication and Access to Clinical Trial Data
- There are also objectives in the USA to be more transparent, expressed in the PhRMA Principles
- Other EU and USA regulations require the protection of personal data (e.g. Regulation (EC) No 45/2001 and Directive 95/46/EC and HIPPA) and to stay within the scope of informed consent

Types of Variables Requiring Protection

- a patient's name
- geography
- date, datetime
- telephone number, fax number
- email address
- social security number
- medical id, insurance id
- account numbers
- license numbers
- vehicle id
- device id
- URL, IP address
- biometric id
- image
- subject id
- investigator id
- site id
- other ids

The Objectives – Protection and Analyses

Protection of patient identity and enabling meaningful analyses

- OBJECTIVE 1: Data privacy protection
  - Protect all the data values that can identify a patient
- OBJECTIVE 2: To create a database that can be used for meaningful analyses and for the testing of internal systems.
  - Keep the data values that do not identify a patient – test results, etc.
  - Support the ability to merge data
  - Know that the same person with these lab tests had these AEs
  - Perform chronological analyses such as time to onset, duration, concurrency, etc.
  - Create derived variables

Incompatible Objectives?

Can we meet both objectives of protection and analyze-able data?

- It is easy to protect data privacy but to create a protected database that is still able to be used for meaningful analyses is more difficult
- The objective of protecting patient identify would lead us to dropping all patient identifying data variable values, but that would result in a database that cannot be analyzed. There would be no dates, times, patient identifiers, ages, etc.
- The objective of analyses would lead us to keeping identifying variables, but that would not protect data privacy.
- We need to provide data that can be meaningfully analyzed and that also protects the privacy of patient data
- To both drop and keep the same variables!

The Solution – Realistic but not Real Identifiers

- Data privacy protection that can also provide analyzable data can be created by breaking the link between patients' identities and their clinical data values.
- The identifying data variables must be **realistic but not real**.
- Some variables need to be translated to false values that behave in the same way as the real values when merging and analyzing the data. Translate consistently across data sets, so patient id X translates to Y in all data sets.
- Dates need to be changed but the intervals between any two dates need to be the same as in the real data.

Degrees of Protection

- The type and degree of protection can depend on the analyses that needs to be done with the data. Some types of analyses may be able to be done with more protection but other types may require less protection.
- A minimum level of protection is required for all data to meet legal and regulatory requirements.

The Privacy Protection Software

How to meet both objectives.

- Software must support the definition of several levels of protection and retain the data relationships to enable meaningful analyses, supporting both objectives.
- The levels and types of protection are specified in a definition data set, so the software does not need to be modified to support any of the types of protection nor for the selection of variables that require protection.
- The definition data set contains
  - A list of data set and variable names that require to be protected
  - A specification of the name of the type of protection to apply to each variable
  - The variables that require identical translations across data sets and libraries
- By the way, this same software can be used to protect the treatment blind, too. Protecting treatment assignment and lab results uses the same logoc as protecting patient identifiers.

Protection – Deidentification or Anonymization?

Two ways to interpret the regulations

- Two terms have come into the discussion
  - Deidentification – protect data privacy and allow the capability to re-identify the data
  - Anonymization – protect data privacy and prohibit the capability to re-identify the data
- Novartis has decided to anonymize data
- The regulations may be subject to interpretation, such that deidentification might be allowed.
  - Data privacy regulations apply to all industries, not just pharma. Some of these industries are *required to delete* identifying data as soon as they are finished with it. These non-pharma industries can retain the data only if they anonymize it. Pharma is *required to keep* the real data, so deidentification may be allowed.

Types of Protection

- Each variable can be assigned one of the following modes of protection
  - DROP – drop the variable
  - MISSING – keep the variable but set its value to missing
  - TRANSLATE – change the variable from one value to another one
  - TRANSCRAM – change the variable values and scramble the results
  - SCRAMBLE – retain the real variable values but scramble them
  - DATE – translates dates to a point in the future (to make it obvious they are protected) and retain the intervals between any two dates within and across data sets.
  - AGEINT – converts subjects' actual age values to age ranges

Details of Some of the Types

- TRANSLATE – change the real value to a protected value but doing so the same way for all instances of a variable in all of its data sets. An example is the site id. Analyses by site id requires that observations in different data sets have the same site id translation so that merges across data sets will result in the same matching observations.
- TRANSCRAM – The translated variables are also randomly scrambled to increase the level of protection, but the scramble is done the same way across data sets
- SCRAMBLE – the variable values are not translated to different values but they are scrambled to protect identity
- AGEINT – if the age is greater than 89 then group them to “90 or older”

Cross Data Set and Library Consistency

- Translations from real values to protected values must be done consistently across all data sets that will be analyzed together
  - Across data sets within e.g. SDTM
  - Across data sets between e.g. SDTM and ADaM
- Without this consistency, merging data sets would not be possible, identifying variables tend to be primary keys that are used in the BY statement when merging
- The definition data set enables this by allowing specification of shared translation tables used by any set of variables, across any number of domains.

Dates and Datetime Consistency

- The intervals between dates, datetimes and times in the protected data need to be the same as in the real data, but with different dates. Dates can identify a person, but analyses often requires temporal calculations, such as durations and time to event.
- So all dates, datetimes and times must be translated to protected values with complete consistency to retain intervals.
- Component variables must also be consistent, e.g. variables month, day, year and date must be consistent with each other

Row-level Protection

- It is very important to note that protection must recognize row-level metadata, e.g. it must deidentify ORRES, STRESN, STRESC in different ways for each value of TESTCD. Parameter-related variables require deidentification for each testcd row value as if they were separate variables
- For example, if xxTESTCD can identify a patient then ORRES, STRESN, etc. need to be deidentified in all and only those rows with that xxTESTCD value.

Creating the Definition File

- Another SAS macro initializes a definition file. It creates a proc contents output data set and transforms it into the definition data set structure.
- Date and datetime variables are automatically defined to be protected in this starting file. All other variables are left with no mode of protection, someone needs to decide which variables need to be protected and enter the mode into the definition file.
- This definition file can be defined at a global standards level and copied or used for each study. A definition standard can be defined for any set of databases, to implement a consistent protection across all studies.

The SAS Macro Logic

- Read the definition data set

	table	column	param	paramrel	mode	identifier_type	date_type	full_date
1	AE	ACCESSTS			date	datetime	sasdatetime	
2	CM	ACCESSTS			date	datetime	sasdatetime	
3	CO	ACCESSTS			date	datetime	sasdatetime	
4	DM	ACCESSTS			date	datetime	sasdatetime	
5	DS	ACCESSTS			date	datetime	sasdatetime	
6	DV	ACCESSTS			date	datetime	sasdatetime	
7	EG	ACCESSTS			date	datetime	sasdatetime	
8	IE	ACCESSTS			date	datetime	sasdatetime	
9	MH	ACCESSTS			date	datetime	sasdatetime	
10	PH	ACCESSTS			date	datetime	sasdatetime	
11	SC	ACCESSTS			date	datetime	sasdatetime	
12	SU	ACCESSTS			date	datetime	sasdatetime	
13	SV	ACCESSTS			date	datetime	sasdatetime	
14	VS	ACCESSTS			date	datetime	sasdatetime	
15	Y7	ACCESSTS			date	datetime	sasdatetime	
16	ZD	ACCESSTS			date	datetime	sasdatetime	
17	ZE	ACCESSTS			date	datetime	sasdatetime	
18	ZJ	ACCESSTS			date	datetime	sasdatetime	
19	ZN	ACCESSTS			date	datetime	sasdatetime	
20	ZQ	ACCESSTS			date	datetime	sasdatetime	
149	AE	AESTDTN			date	datetime	sasdatetime	
170	DM	AGECLUQ			missing			
194	DM	BRTHCL			drop			
195	DM	BRTHCLF			drop			
196	DM	BRTHCLL			drop			
197	DM	BRTHCLN			drop			
198	DM	BRTHCLQ			drop			
199	DM	BRTHCLS			drop			
200	DM	BRTHDAT			date	datec	yymmdd10.	
201	DM	BRTHDATF			date	datec	yymmdd10.	
202	DM	BRTHDATQ			drop			
203	DM	BRTHDDT			date	datec	yymmdd10.	
208	DM	BRTHYRQ			missing			
268	DM	CHDBERQ			missing			
378	DS	CONTEPQ			missing			
381	DM	COUNTRY			scramble			
382	DM	COUNTRYL			drop			
383	DM	COUNTRYN			drop			
384	DM	COUNTRYQ			drop			
385	DM	COUNTRYS			drop			

The SAS Macro Logic

- Randomly select a number of days between 200 and 274 years in the future from the run date
- Randomly select a number of seconds between 200 and 274 years in the future from the run datetime and that is consistent with the date offset
- These numbers will not be displayed in the SAS log file and are used to increment all date and datetime values such that they are not real but retain their durations if one were to compute durations from protected dates
- After the date and datetimes are incremented the date component variables are also incremented in the same way so as to be consistent (i.e. variables holding month, day, year, hour, minute and second)
- Consider seasonality, where the season of year might need to be retained in protected date and datetime values

The SAS Macro Logic

- Create translation tables for each variable assigned the translate or transcram modes
- The translation tables are implemented as SAS user defined formats for each indentifier_type in the definition file
- The translation tables take all the unique values of all the variables that are defined with the same identifier_type in the definition file. This across – data set processing enables the translation tables to translate variables values consistently so that data sets can be merged and analyzed with meaningful results.
- These translations also take row level into account
- A warning is generated about variables that have only one value, scrambling these will not protect

The SAS Macro Logic

- Character variables are translated to X, x and n
  - to_char = translate(from_char,
  - 'xxxxxxxxxxxxxxxxxxxxxxxxxxXXXXXXXXXXXXXXXXXXXXXXXXXX Xnnnnnnnnnn',
  - 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXY Z0123456789');
- Numeric values are translated so as to maintain the intervals between them.
- An optional output data set contains the from and to values of all the translations.

The SAS Macro Logic

- Scrambling values is based on the SAS ranuni function and ensures the same scramble across data sets for variables with the same identifier_type. i.e. a stratified scramble.
- The SAS log clearly shows how the protection is done. Optionally the re-identification keys can be generated too. It is possible to write a new SAS macro to re-identify a database, if that is useful to do.

The Steps to Protect Data

Initialize definition, populate definition, protect

libname confp ‘<study data>‘ access=readonly;

libname deident ‘<definition data set library>';

libname deiddata ‘<protected data to be created>’;

1. %deidentify_create_definition**(inlib**=*confp*,**out**=*deident.deiddef*)

2. Complete population of the definition data set or use standard

3. %deidentify(**inlib**=*confp*,**id_data**=*deident.deiddef*,**outlib**=*deiddata*)

- MPRINT(DT_DEIDENTIFY): *----------;
- MPRINT(DT_DEIDENTIFY): * number of variables by deidentification type in dm;
- MPRINT(DT_DEIDENTIFY): * 3 variables will be set to missing;
- MPRINT(DT_DEIDENTIFY): * 11 variables will be dropped;
- MPRINT(DT_DEIDENTIFY): * 1 non-date variable values will be translated;
- MPRINT(DT_DEIDENTIFY): * 0 non-date variable values will be transcramed;
- MPRINT(DT_DEIDENTIFY): * 1 non-date variable values will be scrambled;
- MPRINT(DT_DEIDENTIFY): * 3 date conversions;
- MPRINT(DT_DEIDENTIFY): * 3 date-time conversions;
- MPRINT(DT_DEIDENTIFY): * 1 ageint conversions;
- MPRINT(DT_DEIDENTIFY): *----------;

Details

- Demonstration
  - Show the definition file
  - Show the macro calls
  - Show the generated code that implements protection