phuse·connect
connect·share·advance
5-8 November
ICC Birmingham
EU 2023
The Clinical Data Science Conference

Paper DH06

Transparent is the New Black – Improving Data Findability Through Better Clinical Metadata

Lukasz Kniola, Biogen, Maidenhead, UK

ABSTRACT

The secondary use of clinical trial data can significantly increase its value beyond the original analysis and help find new insights, signals, and ultimately, treatments. A major hurdle to data reuse is the lack of transparency and discoverability.

SDTM and ADaM datasets have reliable, strict naming conventions and consistent structure within and across studies. These qualities make it possible to create generalized programs which stack and process datasets in an automated manner, regardless of their contents. The output is descriptive metadata that is useful, searchable, and not constrained by privacy concerns. Such metadata can be made open and searchable while access to data remains restricted. It allows for finding tests, events, visits, and characteristics across datasets and studies before requesting access to relevant data assets.

This paper describes the technique for leveraging CDISC standards and simple statistics to unlock the value of clinical data and increase its transparency and discoverability for researchers and data scientists. It also discusses the benefits and challenges of this approach and provides practical examples of how it can be implemented.

INTRODUCTION: "F" IN F.A.I.R. STANDS FOR FINDABLE

The FAIR data principles, standing for Findable, Accessible, Interoperable, and Reusable, emerged as a framework aimed at enhancing the organization, sharing, and utilization of digital data, particularly within the realm of scientific investigations, including clinical research. These principles were formulated in 2014 to tackle the challenges presented by the increasing volume of data and the need to optimize data sharing, encourage collaborative discoveries, and ultimately advance medical knowledge and patient care through the meaningful integration of data.

CDISC standards, such as SDTM and ADaM, go a long way in addressing the challenges of interoperability of tabular clinical data by providing unambiguous and well-defined standards for data structure and format, as well as, to a great extent, its contents, through controlled terminology and standardized vocabularies.

Accessibility and Reusability are topics for another paper; however, they would be meaningless if data were not Findable in the first place.

It could be argued that anything that exists can be found, which would make any dataset findable to a degree. But this statement assumes that whoever is searching knows what it is they are looking for. In the context of secondary research and data re-use, this is often not the case. The following example should illustrate the point:

Adam is a researcher who intends to run an analysis of HbA1c levels using pooled data from historical studies he previously came across. He requests access to studies ALPHA, GAMMA, and SIGMA since he knows those studies collected HbA1c levels. What Adam is not aware of is that studies LAMBDA and KAPPA also tested HbA1c, and if included, they would more than double the population for his analysis. Since he doesn't know about them, he doesn't request them. FINDABILITY is not the problem in this example. It is DISCOVERABILITY, or the lack thereof, that lets the researcher down.

Clinical data has sensitivities, which means it's usually not open and free to view, query, and use by everyone at will. There are privacy concerns, commercial confidentiality issues, unblinding risks, and others. This necessitates a request and approval process to allow access to data only as needed and/or justified. But it also creates a

conundrum. You can't request data if you don't know what's inside. You can't know what's inside if you don't query the data. You can't query the data if you don't request it.

But there is a way to get a good idea of what's inside the data, without opening each dataset. Enter METADATA. This paper offers a solution that takes advantage of the brilliant design characteristics of CDISC datasets to build a simple, reusable, scalable, study and dataset-independent system, which can make the discovery of datasets and the data they contain not only possible but also relatively easy.

DIFFERENT FLAVOURS OF METADATA

Metadata is structured information that provides context and details about other data or, in short, data about data. It includes various attributes, properties, or characteristics that describe the content, quality, format, location, and other aspects of a dataset or information resource. Metadata plays a crucial role in organizing, managing, discovering, and using data effectively, as well as automating the use of the data it describes. It can help users not only locate relevant resources but also understand how to interact with them in meaningful ways.

There are different categories of metadata – administrative, technical, structural, provenance, semantic, temporal, to name a few. Metadata can also be categorized as prescriptive – offering instructions for how the resource should be processed or used to achieve certain outcomes, and descriptive – describing the resource's content, making it easier for users to find and understand the resource.

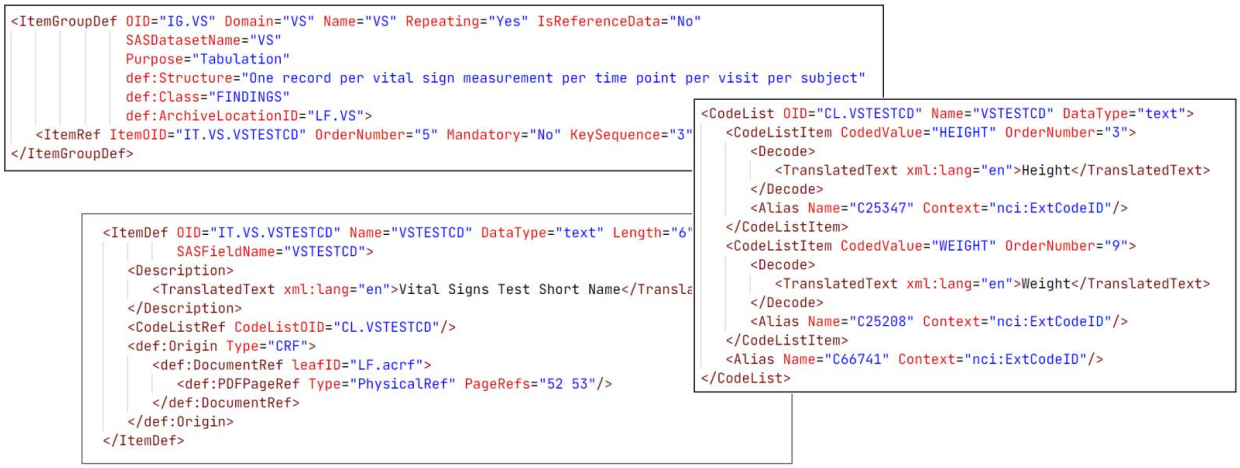

Figure1. Define.xml file fragments

Dataset specifications are a form of metadata. For example, Define.xml (see Figure 1) not only describes and informs the structure of datasets, but it also prescribes the exact way of sourcing, processing, and deriving values, and in some instances, limits the possible entries based on dictionaries and controlled terminology values lists. However, it isn't a description of the contents of the final product.

Even if the specifications tell us that dataset DM has a variable COUNTRY that uses ISO3 country codes as values (so we can't just enter any random value), they don't specify which countries are specifically present in the dataset or the frequencies for those countries.

Even if they define that dataset VS has values for parameters (VSTESTCD) HEIGHT, WEIGHT, DIABP, SYSBP, PULSE, they do little to describe how many records for each of them there are, what the ranges are, or what visits they were collected at.

Some metadata is created without our intervention by the systems we use. The most obvious example is a listing of the directory like the one shown on Figure 2 where datasets are stored.

```
-bash-4.2$ ls -l --full-time
total 5256
-rwxr-xr-x 1 lkniola lkniola  176597 2018-09-10 15:10:37.000000000 -0400 ae.csv
-rwxr-xr-x 1 lkniola lkniola  958464 2019-12-02 11:08:18.175240500 -0500 ae.sas7bdat
-rwxr-xr-x 1 lkniola lkniola   16227 2020-07-06 15:32:44.156412200 -0400 dm.csv
-rwxr-xr-x 1 lkniola lkniola  131072 2019-12-02 11:08:02.013118500 -0500 dm.sas7bdat
-rwxr-xr-x 1 lkniola lkniola  354866 2018-09-10 15:10:38.000000000 -0400 ex.csv
-rwxr-xr-x 1 lkniola lkniola 1179648 2019-12-02 11:08:34.733648300 -0500 ex.sas7bdat
-rwxr-xr-x 1 lkniola lkniola 1046280 2018-09-10 15:10:41.000000000 -0400 vs.csv
-rwxr-xr-x 1 lkniola lkniola 1572864 2023-04-06 07:32:33.454165018 -0400 vs.sas7bdat
```

Figure 2. Directory listing in Unix

Metadata is automatically available for existing datasets. For example, in SAS® we can find all attributes of any dataset, as illustrated on Figure 3.

Library Name	Member Name	Member Type	Column Name	Column Type	Column Length	Column Position	Column Number in Table	Column Label
DATA	DM	DATA	STUDYID	char	8	8	1	Study Identifier
DATA	DM	DATA	DOMAIN	char	2	16	2	Domain Abbreviation
DATA	DM	DATA	USUBJID	char	19	18	3	Unique Subject Identifier
DATA	DM	DATA	SUBJID	char	10	37	4	Subject Identifier for the Study
DATA	DM	DATA	RFSTDTC	char	19	47	5	Subject Reference Start Date/Time
DATA	DM	DATA	RFENDTC	char	19	66	6	Subject Reference End Date/Time
DATA	DM	DATA	SITEID	char	10	85	7	Study Site Identifier
DATA	DM	DATA	BRTHDTC	char	19	95	8	Date/Time of Birth
DATA	DM	DATA	AGE	num	8	0	9	Age in AGEU at RFSTDTC
DATA	DM	DATA	AGEU	char	6	114	10	Age Units
DATA	DM	DATA	SEX	char	8	120	11	Sex
DATA	DM	DATA	RACE	char	47	128	12	Race
DATA	DM	DATA	ARMCD	char	20	175	13	Planned Arm Code
DATA	DM	DATA	ARM	char	30	195	14	Description of Planned Arm
DATA	DM	DATA	COUNTRY	char	3	225	15	Country

Figure 3. Dataset attributes table

It's easy to notice that none of those contain any information about the contents, only the location, time, format, and structure. To learn about the contents, it is necessary to process the datasets individually.

This can be almost as complex as creating report tables out of those datasets, and it could provide detailed content descriptions. However, it would almost certainly require creating a separate script for each SDTM and ADaM dataset based on its structure and purpose, which would likely cause more problems than it would solve.

On the other hand, extracting details about contents can be as simple as running descriptive statistics (n, mean, min, max, etc.) on all numeric variables and frequencies of values of all character variables. This option is quick and scalable. With a simple function or macro, we can take any SDTM or ADaM dataset with any number and set of columns and get an idea of what each character and numeric variable looks like.

Dataset	Variable	Desc	Value
LB	LBTESTCD	BCALB	711
LB	LBTESTCD	BCALK	712
LB	LBTESTCD	BCALT	712
LB	LBTESTCD	BCAST	705
LB	…	…	…
LB	LBSTRESN	N	104804
LB	LBSTRESN	Mean	7321.78
LB	LBSTRESN	Std Dev	40413.92
LB	LBSTRESN	Minimum	0
LB	LBSTRESN	Maximum	973600

Table 1.

However, there are significant issues with this approach.

Firstly, not all character variables are categorical (therefore best described by frequencies). For example, dates are expressed as character variables. What would be the use of having birth dates frequencies? Equally, not all numeric variables are continuous, and descriptive statistics are pointless.

Secondly, the vertical structure of SDTM and ADaM datasets means that the results may be meaningless. In Table 2, the frequencies of LBTESTCD variable may be helpful, but the mean, standard deviation, and others for the LBSTRESN are calculated over all values of all lab parameters, making it somewhat pointless.

Fortunately, CDISC formats like SDTM and ADaM, along with their well-thought-out properties and structure, provide another way to describe these datasets. This approach is both valuable and highly scalable.

RESULT LEVEL METADATA

The solution is to focus on the core information, along with just the necessary context, rather than providing a full description of all datasets, every variable, and the values they contain.

For this information to be useful and enable the discoverability of data contained in the datasets, it is necessary to describe not only the questions asked in the study and the tests performed but also the answers and results of these tests. It is not enough to know that country information was collected; it should be possible to find which countries and

how many participants were reported in each country. It may be valuable to learn that HbA1c levels were reported, but it would be much more useful to know some aggregate characteristics of the results as well.

It is also essential that the descriptions of all datasets have a consistent structure, enabling them to be pooled and queried efficiently as the foundation for a search engine.

Such information can be referred to as *Result Level Metadata*. Unlike study-level metadata or dataset metadata, which describe the setup, assumptions, and inputs of studies, result-level metadata describes the outputs at the level, which is informative but observes the limitations of access.

THE STACKABILITY OF SDTM AND ADAM DATASETS

Furthermore, what we need is a method for extracting result level metadata in an automated manner, ideally one that can handle any dataset as input. This might initially seem daunting, but a closer examination of the SDTM and ADaM standards, including their definitions and structure, provides a promising solution.

In a general sense, each dataset (with a few exceptions we will address separately) should have variables that capture:

- Category or some form of internal record grouping
- Question, test, assessment
- Answer, result, finding, observation
- Timing information in the form of dates or study days
- Timing information in the form of study periods, stages, or visits

The SDTM Implementation Guide defines several domain models, which include:

- Special-purpose domains (DM, CO, SE, SV)
- Trial design domains (TA, TE, TV, TI, TS)
- Interventions (CM, EX, SU)
- Events (AE, DS, MH, DV, CE)
- Findings (EG, IE, LB, PE, QS, SC, VS, etc.)

SDTM FINDINGS DATASETS

Let's examine a few examples of FINDINGS domains and variables containing this information.

SDTM Domain	Grouping vars	Question vars	Answer vars	Timing vars (date/day)	Timing vars (period/visit)
LB	LBCAT, LBSCAT	LBTESTCD, LBTEST	LBORRES, LBORRESU, LBSTRESC, LBSTRESN, LBSTRESU	LBDTC, LBDY	VISITNUM, VISIT
QS	QSCAT, QSSCAT	QSTESTCD, QSTEST	QSORRES, QSORRESU, QSSTRESC, QSSTRESN, QSSTRESU	QSDTC, QSDY	VISITNUM, VISIT
SC	SCCAT, SCSCAT	SCTESTCD, SCTEST	SCORRES, SCORRESU, SCSTRESC, SCSTRESN, SCSTRESU	SCDTC, SCDY	
VS	VSCAT, VSSCAT	VSTESTCD, VSTEST	VSORRES, VSORRESU, VSSTRESC, VSSTRESN, VSSTRESU	QSDTC, QSDY	VISITNUM, VISIT

Table 2.

We quickly notice a clear pattern. If we remove the prefixes of variables, which are also the domain names, we end up with a uniform structure.

SDTM Domain	Grouping vars	Question vars	Answer vars	Timing vars (date/day)	Timing vars (period/visit)
__Findings	__CAT, __SCAT	__TESTCD, __TEST	__ORRES, __ORRESU, __STRESC, __STRESN, __STRESU	__DTC, __DY	VISITNUM, VISIT

Table 3.

By renaming those variables in each FINDINGS dataset and keeping only the variables common to all, we can stack these datasets. By retaining STUDYID, USUBJID, and DOMAIN variables in each of them, we preserve the details of which study, subject, and domain each record comes from.

Of course, there are other variables in different domains relevant to each of them (LBSPEC, LBFAST, LBTOX, QSEVLINT, SCREASND, VSLOC, and many more). Not using them will remove a level of detail and precision from the metadata we will create, but it will make the process of stacking and bulk processing more flexible and easier to scale and maintain.

SDTM INTERVENTIONS AND EVENTS DATASETS

Similar patterns can be found in INTERVENTIONS and EVENTS domains.

SDTM Domain	Grouping vars	Question vars	Answer vars	Timing vars (date/day)	Timing vars (period/visit)
CM	CMCAT, CMSCAT	Implied from DOMAIN: "Concomitant medications reported"	CMTRT, CMDECOD, CMOCCUR	CMSTDTC, CMENDTC, CMSTDY, CMENDY	CMENRF
SU	SUCAT, SUSCAT	Implied from DOMAIN: "Substance use reported"	SUTRT, SUDECOD, SUOCCUR	SUSTDTC, SUENDTC, SUSTDY, SUENDY	SUENRF
AE	AECAT, AESCAT	Implied from DOMAIN: "Adverse events reported"	AETERM, AEDECOD, AEBODSYS	AESTDTC, AEENDTC, AESTDY, AEENDY	AEENRF
MH	MHCAT, MHSCAT	Implied from DOMAIN: "Medical history reported"	MHTERM, MHDECOD, MHBODSYS	MHSTDTC, MHENDTC, MHSTDY, MHENDY	MHENRF
CE	CECAT, CESCAT	Implied from DOMAIN: "Clinical events reported"	CETERM, CEDECOD, CEBODSYS	CESTDTC, CEENDTC, CESTDY, CEENDY	CEENRF

Table 4.

Here, too, we can take advantage of those characteristics.

SDTM Domain	Grouping vars	Question vars	Answer vars	Timing vars (date/day)	Timing vars (period/visit)
__ Interventions	__CAT, __SCAT	DOMAIN	__TRT, __DECOD, __OCCUR	__STDTC, __ENDTC, __STDY, __ENDY	__ENRF
__ Events	__CAT, __SCAT	DOMAIN	__TERM, __DECOD, __BODSYS	__STDTC, __ENDTC, __STDY, __ENDY	__ENRF

Table 5.

Interventions and Events domains can be kept separate, or, with simple logic, processed together. This will depend on whether any additional variables are relevant to the search engine that uses this information (__SEV, __SER, etc. may or may not be useful, although they could be treated as grouping variables).

SDTM DEMOGRAPHIC/BASELINE DATASET

Finally, let's examine the special-purpose domains. DM (Demographics) will be an obvious domain to stack. It is also a rare SDTM dataset with a horizontal structure, where AGE, SEX, RACE, COUNTRY, are not stored as individual question-answer records, but each detail is stored in a dedicated variable. Being a unique case, DM can be processed as is.

SDTM Domain	Grouping vars	Question vars	Answer vars	Timing vars
DM	"Total Count"	distinct USUBJID	count(*)	-
DM	"Age" / "Age Group"	AGE / AGEGRP=floor(AGE/10)*10	count(*) group by AGE / count(*) group by AGEGRP	-
DM	"Sex"	SEX	count(*) group by SEX	-
DM	"Race"	RACE	count(*) group by RACE	-
DM	"Country"	COUNTRY	count(*) group by COUNTRY	-
DM	"Treatment Group"	ARM	count(*) group by ARM	-
DM	"Treatment Start Year"	STARTYEAR=year(RFSTDTC)	count(*) group by year(STARTYEAR)	-

Table 6.

It may be worth considering using AGE groups instead of individual AGE values (the example below uses 10-year groups). The year part of RFSTDTC can be used to gather information about the time of the start of treatment. Other details can be summarized as required by the project.

OTHER SDTM DATASETS

It makes little sense to inventory the contents of the CO (Comments) domain as it mainly contains verbatim text, with no categorization qualities. SE and SV, as well as the trial design domains, do not contain subject-level information and could arguably be stacked and stored along with the metadata extracted from Demographic, Findings, Events, and Interventions domains.

ADAM DATASETS

For ADaM datasets, the step of renaming variables will not be necessary. Every ADaM dataset that uses BSD, TTE, or another defined structure will use the same set of variables.

ADaM Domain	Grouping vars	Question vars	Answer vars	Timing vars (date/day)	Timing vars (period/visit)
BSD	PARCATy, PARCATyN	PARAMCD, PARAM	AVAL, AVALC	ADT, ASTDT, AENDT, ADY, ASTDY, AENDY	AVISITN, AVISIT, ATPT, APERIOD
TTE	PARCATy, PARCATyN	PARAMCD, PARAM	AVAL, EVNTDESC	ADT, STARTDT	AVISITN, AVISIT

Table 7.

ADaM datasets typically contain much more information, such as demographic grouping, population flags, baseline, and outcome flags, etc. However, if we limit the datasets to only grouping, question, answer, and timing variables, then stacking any number of such datasets from one or multiple studies should not require additional pre-processing.

PROCESSING STACKED DATASETS

In short, by standardizing datasets of each model (findings, interventions, events) to a uniform structure, as explained earlier, we make it possible to stack the datasets prior to processing them for the extraction of result-level metadata. Instead of processing each dataset separately, it is possible to process the combined model datasets "in bulk", as illustrated below.

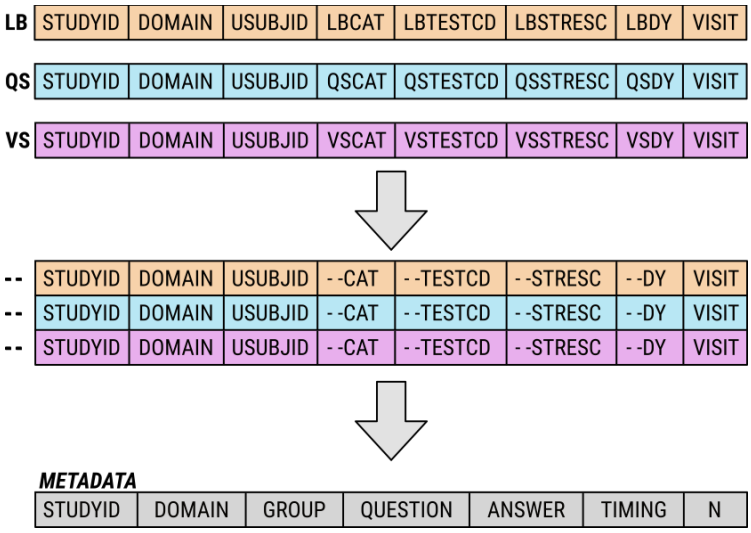

Figure 4. Stacking FINDINGS datasets before processing

When dealing with datasets from multiple studies, it is straightforward to extract the result-level metadata from each study and combine them.

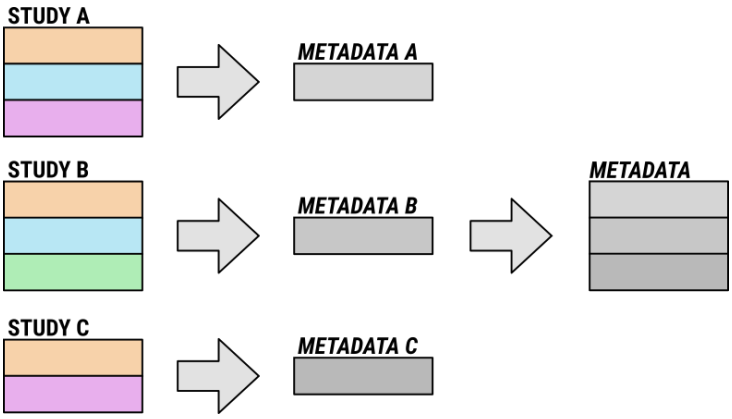

Figure 5. Stacking metadata after processing studies

Furthermore, because the combined datasets from each study are uniform in structure, it is possible – where it makes sense – to combine them and extract result-level metadata from this combined multi-study super-dataset.

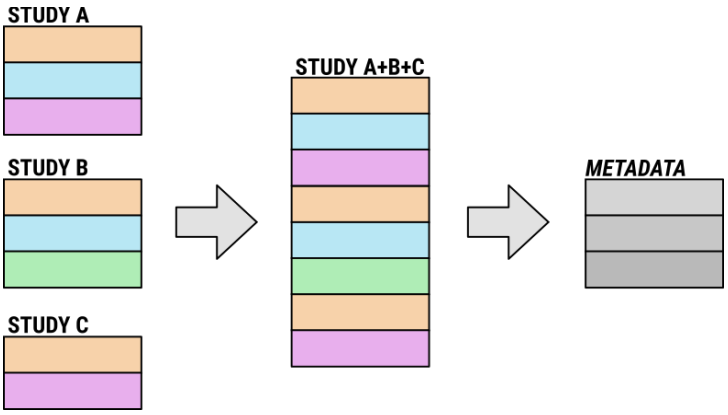

Figure 6. Stacking studies' data before processing

The choice between the two options will depend on the project's goals. It is also possible to combine the two options: process all historical studies in one go, but only add new, incoming studies one by one.

GENERALIZED PROGRAMS TO PROCESS DATASETS

The purpose of making it possible to stack datasets to use them as a metadata source is to build a small set of programs/macros that can take any SDTM or ADaM dataset and produce consistent, structured content metadata. This approach also provides flexibility in deciding whether it is more efficient to process datasets individually or to stack all study datasets into demographics, findings, interventions + events, BSD, TTE super-datasets and process them in that form. Depending on the programming and database environment, it may be more desirable to process many small datasets or a handful of large ones.

INITIAL ASSUMPTIONS AND CONSIDERATIONS:

- Some variables may not be present in some datasets. Pooling means they will likely be present in the combined datasets, but it is a good practice to start with an empty dataset with the required structure and append all other datasets to it.
- Decide which variables will be used as categories or groups to achieve the right level of detail.
- Consider the date/study day variables. Some datasets may use __STDY + __ENDY instead of __DY, or vice versa. Cater for such situations by using the COALESCE function to take the first non-missing value from all available.
- __ORRES contains results as captured in the study. __STRESC/__STRESN contain results converted to standard units. In the context of pooling results from multiple studies, only the results in standard units are relevant, so there is no need to process the __ORRES variables.

- __STRESC contains the result in a character format. __STRESN contains the result as numeric. The former will be present regardless of the type of result, while the latter is likely to be missing for character or categorical values. Therefore, __STRESC is useful when checking if the result is non-empty, while __STRESN can be used for descriptive statistics calculations.

BASIC RESULT LEVEL METADATA

At its most basic, the metadata can be limited to listing parameters found in datasets, with no regard for their frequency of occurrence or values. Example programs to extract this information from Events and Findings domains are straightforward and may look similar to those shown below.

```
create table RLM_EVENTS as
 select distinct
  STUDYID, DOMAIN
  ,__CAT, __SCAT
  , __DECOD
 from EVENTS;

create table RLM_FINDINGS as
 select distinct
  STUDYID, DOMAIN
  , __CAT, __SCAT
  , __TESTCD, __TEST
 from FINDINGS;
```

ADDING MORE DETAILS

Since each dataset will be processed to extract Result Level Metadata, it's worth upgrading the code to gain a better understanding of the contents of each dataset. The code below has been amended to find details such as:

- Count of records with a particular test (1)
- Number of unique subjects with non-missing values (2)
- Proportion of all subjects it represents (3)
- Simple descriptive statistics can be produced for all numeric results (4)
- Units added for better readability.

```
create table RLM_EVENTS as
 select distinct
  STUDYID, DOMAIN
  , __CAT, __SCAT
  , __DECOD
  , count(distinct USUBJID) as N_SUBJS                          --(2)
  , (calculated N_SUBJS)/_TOT_SUBJS as P                        --(3)
  , count(*) as N                                               --(1)
 from EVENTS
  , (select count(distinct USUBJID) as _TOT_SUBJS from data.DM) --(3)
 group by STUDYID, DOMAIN, __CAT, __SCAT, __DECOD;

create table RLM_FINDINGS as
 select distinct
  STUDYID, DOMAIN
  , __CAT, __SCAT
  , __TESTCD, __TEST
  , __STRESU                                                    --(5)
  , count(distinct USUBJID) as N_SUBJS                          --(2)
  , (calculated N_SUBJS)/_TOT_SUBJS as P                        --(3)
  , count(*) as N                                               --(1)
  , mean(__STRESN) as MEAN, median(__STRESN) as MEDIAN          --(4)
  , min(__STRESN) as MIN, max(__STRESN) as MAX                  --(4)
 from FINDINGS
  , (select count(distinct USUBJID) as _TOT_SUBJS from data.DM) --(3)
 where __STRESC <> ''                                           --(2)
 group by STUDYID, DOMAIN, __CAT, __SCAT, __TESTCD, __TEST, __STRESU;
```

The selection of grouping and category variables will depend on the expectations and requirements of the system we are building and the queries we intend to respond to. Categorical variables like __CAT and __SCAT are obvious candidates. Other variables can be added depending on the requirements. For example, VISITNUM/VISIT can be used as grouping variables to provide more granular information about tests on a visit-by-visit level. Another possibility is to use __BLFL to consider all baseline and post-baseline records as different categories. There is no technical limit to how many grouping/category variables can be introduced; however, it is worth considering only those present in all or the majority of datasets of a certain type.

ONE TABLE TO RULE THEM ALL

Events, Findings, Interventions, as well as Demographics and any other group of metadata sets, will each have a slightly different structure driven by the design of those models and variables found in them. However, those structures are similar enough to apply one more step of unification, creating a common structure to which all models will fit.

By concatenating grouping/categorical variables, as well as tests where applicable, and applying consistent names to variables, we can build a mega set that will contain all RLM from all types of datasets.

The code shown previously can be amended to restructure the outputs to the common form:

```
create table RLM_EVENTS as
 select distinct
  STUDYID, DOMAIN
  , catx(' :: ', __CAT, __SCAT, __BODSYS) as GRP
  , __DECOD as Q
  ,count(distinct USUBJID) as N_SUBJS
  ,(calculated N_SUBJS)/_TOT_SUBJS as P
  ,count(*) as N
 from EVENTS
  ,(select count(distinct USUBJID) as _TOT_SUBJS from data.DM)
 group by STUDYID, DOMAIN, __CAT, __SCAT, __BODSYS, __DECOD;

create table RLM_FINDINGS as
 select distinct
  STUDYID, DOMAIN
  , catx(' :: ', __CAT, __SCAT) as GRP
  , catx(' :: ', __TESTCD, __TEST) as Q
  , __STRESU as UNIT
  ,count(distinct USUBJID) as N_SUBJS
  ,(calculated N_SUBJS)/_TOT_SUBJS as P
  ,count(*) as N
  ,mean(__STRESN) as MEAN, median(__STRESN) as MEDIAN
  ,min(__STRESN) as MIN, max(__STRESN) as MAX
 from FINDINGS
  ,(select count(distinct USUBJID) as _TOT_SUBJS from data.DM)
 where __STRESC <> ''
 group by STUDYID, DOMAIN, __CAT, __SCAT, __TESTCD, __TEST, __STRESU;
```

We can also include other special case datasets, such as the demographic dataset. However, this will require some pre-processing, specifically transposing the dataset into a vertical layout. The need for special treatment of this unique domain is justified by the value of information it contains.

```
data DM;
  set data.DM;
  AGE1=ceil(AGE/10)*10;
  AGEGRP=compress(catx('-',put(AGE1-9,best.),put(AGE1,best.))); ** 10-y groups **;
  TOTALCOUNT='TOTALCOUNT';  ** used for counting total number of participants **;
run;

proc transpose data=DM out=DMT;
  by STUDYID DOMAIN USUBJID;
  var TOTALCOUNT SEX AGEGRP RACE COUNTRY;
run;
```

```
create table RLM_DEMO as
 select distinct
  STUDYID, DOMAIN
  , _NAME_ as GRP
  , COL1 as Q
  ,count(distinct USUBJID) as N_SUBJS
  ,(calculated N_SUBJS)/_TOT_SUBJS as P
  ,count(*) as N
 from DMT
  ,(select count(distinct USUBJID) as _TOT_SUBJS from data.DM)
 group by STUDYID, DOMAIN, _NAME_, COL1;
```

The need for special treatment of this unique domain is justified by the value of information it contains.

The resulting combined metadata set will feature the following columns:

- **STUDYID**
- **DOMAIN**
- **GRP:** Concatenated category values
- **Q:** Concatenated test/question values
- **N_SUBJS:** Number of unique subjects with a value for the GRP + Q combination
- **P:** Proportion of participants with a value
- **N:** Number of records with a value
- **UNIT:** Unit assigned to Q
- **MEAN, MEDIAN, MIN, MAX:** Descriptive statistics for the GRP + Q combination (only present for numeric results)

Additionally, timing variables can be included. While they will be treated as grouping variables, they can be separated for improved readability and query formulation:

- **BASEPOST**: Variable grouping all baseline and post-baseline records
- **VISIT**: Visit name

An example output dataset can be found in Appendix 4.

CREATE THE METADATA, USE THE METADATA

We have demonstrated that the data for which metadata is to be created can be unified in structure prior to processing. We have also observed that there is a small and finite number of metadata tables that will contain all relevant information. It then follows that those metadata tables can also be stacked seamlessly, regardless of the type or size of the studies they describe and how many or how few datasets each of them consists of. We should only need one table for each type of information (demographics, findings, events, interventions, visits, etc.) across all the studies.

We have also opted for unifying all tables for all types of information into a single structure. This means that this single table or a handful of tables can be used as the backbone of a search tool. In turn, this makes it possible to answer questions about our data, which would have otherwise required complex queries (never mind the fact that the clinical data would not be available for querying without requesting them in the first place).

By feeding this metadata to a simple search engine and opening it up for general use within our organization, we can address the discoverability challenge we identified and set out to solve at the beginning of this paper.

Below are but a few examples of questions that can be easily answered using a simple search tool or by querying the metadata table:

> How many (and which) studies had participants in France and/or Spain?
>
> **>STUDYID: GRP=COUNTRY & (Q=FRA | Q=ESP) & N>0**
>
> Which studies had participants between 41 and 50 years old?
>
> **>STUDYID: GRP=AGEGRP & Q=41-50 & N>0**
>
> Have there been studies where the prevalence of diabetes was >50% among participants?
>
> **>STUDYID: DOMAIN=AE & Q=DIABETES & P>50**

How many studies have been ongoing in the year 2021?

```
>STUDYID: GRP=STARTYEAR & Q=2021 & N>0
```

Which datasets contain MRI results?

```
>DOMAIN: GRP like 'MRI%' | Q like 'MRI%'
```

In which countries was study ALPHA conducted?

```
>GRP, Q, N: STUDYID=ALPHA & GRP=COUNTRY
```

Let's go back to our original example. Adam, the researcher, intended to run an analysis of HbA1c levels using pooled data from historical studies. He requested access to studies ALPHA, GAMMA, and SIGMA since he knew those studies collected HbA1c levels. To make sure he's not missing any valuable data, Adam can search the metadata to check if there are other studies that also match his criteria.

```
>STUDYID: Q=HbA1c
```

Straight away, the list is longer than what Adam expected (**`STUDYID: ALPHA, GAMMA, KAPPA, LAMBDA, SIGMA`**). But he's curious how many participants each study had.

```
>STUDYID, N: Q=HbA1c
```

He can see that including the LAMBDA and KAPPA studies will make his population twice as large (**`STUDYID: ALPHA (245), GAMMA (187), KAPPA (309), LAMBDA (348), SIGMA (192)`**). Next, he can even find out what the baseline HbA1c levels were in each study.

```
>STUDYID, Q, MEAN, MIN, MAX, UNIT: BASEPOST=baseline & Q=HbA1c
```

Or what inclusion/exclusion criteria were used.

```
>STUDYID, Q, N: STUDY in (ALPHA, GAMMA, KAPPA, LAMBDA, SIGMA) & DOMAIN=IE
```

And learn other details about high-level demographics for these studies.

```
>STUDYID, GRP, Q, N: STUDY in (ALPHA, GAMMA, KAPPA, LAMBDA, SIGMA) & DOMAIN=DM
```

Now, Adam can request not only the three studies he was familiar with but two additional ones he has just DISCOVERED, which more than double the population for his analysis and will improve the quality of his analysis.

PRIVACY CONCERNS (HINT: THERE SHOULD BE NONE LEFT)

Once the result-level metadata has been extracted from the clinical data, the two can be separated. Due to the sensitivities around clinical data discussed earlier, it may not be advisable or even possible to make it accessible without request, approval, and access procedures. In other words, it cannot be queried at will to learn its contents.

However, those same sensitivities do not apply to well-designed result-level metadata. Consequently, clinical data can be access-controlled, while metadata can be open and used to identify data sets relevant to potential analysis or a project. But how can we be sure that privacy concerns, commercial confidentiality issues, or unblinding risks don't apply to metadata?

A common way of assessing if privacy is an issue is through three tests: linkability, singling out, and inference. The result-level metadata in the form suggested in this paper has an aggregate nature. There are no longer subject-level records that describe individual study participants. Instead, simple statistics or frequencies describe the ranges of lists of values across datasets. For this reason, as well as the fact that subject IDs are not stored, listed, or aggregated in the metadata, it is not possible to link any of this information to any one participant.

The fact that there are no subject-level records also means that singling out is not possible. If an event has only been observed once in a study, effectively having an N of 1, it is only the event/characteristic that is singled out, and not the individual. There are no means of merging this information with others. For example, if in the same study, there was also only 1 instance of a particular medication, and only 1 participant of a certain age, there is no mechanism to assess if they belong to the same individual or to three different ones.

There is no possibility of inferring values of any parameter for a particular participant, either in isolation or even if other characteristics are known. This is because each parameter is described separately, and metadata only stores final results and no substrates needed to recreate them.

The same constraints as described above also alleviate the unblinding risks. However, those concerns can be further addressed by not including treatment assignments as one of the grouping or question variables.

The main motivation behind creating the result-level metadata is to enable the discovery of data sets that can be requested and used for secondary research. Studies that have not yet reached the stage at which they can be fully or partially shared should not be on the list of studies available for sharing. Creating metadata only once this stage has been reached is an additional step preventing and addressing any unblinding risks and commercial confidentiality issues.

CONCLUSIONS

Clinical data are not open for free access and use due to privacy issues and other sensitivities. While necessary, this greatly affects the transparency and discoverability of historical data. A good way of overcoming this and making data more transparent and available for secondary use is by creating and opening up metadata that describe the contents of clinical datasets.

SDTM and ADaM datasets have reliable, strict naming conventions and consistent structure within and across studies. We have shown how these qualities make it possible to create generalized programs that stack and process datasets in an automated manner, regardless of their contents. The result is a set of descriptive result-level metadata tables, not constrained by privacy concerns. Such metadata can be made open and searchable while access to data remains restricted. It makes it possible to discover tests, events, visits, and characteristics across datasets and studies before requesting access to relevant data assets. This type of meaningful and well-designed metadata, especially when combined with a dedicated search engine, is a way to unlock the value of clinical data and increase its transparency and discoverability for researchers and data scientists within and across organizations.

CONTACT INFORMATION

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

Lukasz Kniola
Biogen, UK
email: lukasz.kniola@biogen.com

APPENDIX 1

Example Python code to replace prefixes with underscores in datasets.

```
import pandas as pd

def rename_columns(df):
    # Create a copy of the DataFrame to avoid modifying the original
    new_df = df.copy()

    # Get the name of the DataFrame
    prefix = [name for name in globals() if globals()[name] is df][0].upper()

    # Find and rename columns that start with the DataFrame name
    for column in df.columns:
        if column.startswith(prefix):
            new_column = column.replace(prefix, '__')
            new_df.rename(columns={column: new_column}, inplace=True)

    return new_df

# Read in data
vs = pd.read_sas("data/vs.sas7bdat")

# Call the function to rename columns
vs_r=rename_columns(vs)

# Print the new DataFrame
print("\nNew DataFrame with renamed columns:")
print(vs_r.columns)
```

APPENDIX 2

Example SAS code to replace prefixes with underscores in datasets.

```
%macro rename_columns(ds);
  data _null_;
    set sashelp.vcolumn(where=(upcase(catx('.',libname,memname)) eq upcase("&ds")))
        end=last;

    ** Start data step to read &ds / write &ds.r. once at the beginning **;
    if _N_ eq 1 then
      call execute("data &ds.r; set &ds(");

    ** Rename all variables starring with domain name **;
    if upcase(substr(name,1,2)) eq upcase(memname) then
      call execute('rename=('||strip(name)||'=__'||strip(substr(name,3))||')');

    ** Finish data setp. once at the end **;
    if last then
      call execute("); run;");
  run;
%mend rename_columns;

%rename_columns(work.ae);
```

APPENDIX 3

Example SQL query to add baseline/post-baseline grouping detail.

```
create table FINDINGS_BL as
 select
  *,
  case
    when __BLFL = 'Y' then 'BL0'
    when __DY > 0 THthenEN 'PBL'
    else '_delete_'
  end as BASEPOST
 from FINDINGS
 having BASEPOST ^= '_delete_';

create table RLM_FINDINGS as
 select distinct
  STUDYID, DOMAIN
  , catx(' :: ', __CAT, __SCAT) as GRP
  , catx(' :: ', __TESTCD, __TEST) as Q
  , __STRESU as UNIT
  , BASEPOST as BASEPOST
  , count(distinct USUBJID) as N_SUBJS
  , (calculated N_SUBJS)/_TOT_SUBJS as P
  , count(*) as N
  , mean(__STRESN) as MEAN, median(__STRESN) as MEDIAN
  , min(__STRESN) as MIN, max(__STRESN) as MAX
 from FINDINGS_BL
  ,(select count(distinct USUBJID) as _TOT_SUBJS from data.DM)
 where __STRESC <> ''
 group by STUDYID, DOMAIN, __CAT, __SCAT, __TESTCD, __TEST
  , __STRESU, BASEPOST;
```

APPENDIX 4

Example combined RLM table.

STUDYID	DOMAIN	GRP	Q	N_SUBJS	P	N	UNIT	MEAN	MEDIAN	MIN	MAX
Phuse23	AE	ADVERSE EVENT	BACK PAIN	34	0.22666	39					
Phuse23	AE	ADVERSE EVENT	CHILLS	24	0.16000	25					
Phuse23	AE	ADVERSE EVENT	DIARRHOEA	19	0.12666	21					
Phuse23	AE	ADVERSE EVENT	DIZZINESS	9	0.06000	9					
Phuse23	AE	ADVERSE EVENT	NAUSEA	24	0.16000	27					
Phuse23	VS	BLOOD PRESSURE	DIABP :: Diastolic Blood Pressure	150	1.00000	2071	mmHg	99.38	99	61	140
Phuse23	VS	BLOOD PRESSURE	PULSE :: Pulse Rate	150	1.00000	2071	beats/min	111.21	111	48	173
Phuse23	VS	BLOOD PRESSURE	SYSBP :: Systolic Blood Pressure	150	1.00000	2071	mmHg	120.52	120	71	170
Phuse23	VS	BODY MEASUREMENTS	HEIGHT :: Height	150	1.00000	150	cm	174.91	175	152	201
Phuse23	VS	BODY MEASUREMENTS	WEIGHT :: Weight	150	1.00000	2071	kg	87.99	87	47	131
Phuse23	DM	AGEGRP	21-30	112	0.74666	112					
Phuse23	DM	AGEGRP	31-40	38	0.25333	38					
Phuse23	DM	COUNTRY	DEU	28	0.18666	28					
Phuse23	DM	COUNTRY	FRA	42	0.28000	42					
Phuse23	DM	COUNTRY	IND	35	0.23333	35					
Phuse23	DM	COUNTRY	USA	45	0.30000	45					
Phuse23	DM	RACE	ASIAN	43	0.28666	43					
Phuse23	DM	RACE	BLACK OR AFRICAN AMERICAN	14	0.09333	14					
Phuse23	DM	RACE	WHITE	93	0.62000	93					
Phuse23	DM	SEX	F	81	0.54000	81					
Phuse23	DM	SEX	M	69	0.46000	69					
Phuse23	DM	TOTALCOUNT	TOTALCOUNT	150	1.00000	150					