PharmaSUG 2021 - Paper DS-081

Adding Rows to a BDS Dataset: When to Use DTYPE in Addition to Metadata

Sandra Minjoe, PRA Health Sciences

ABSTRACT

In simple ADaM Basic Data Structure (BDS) datasets, each SDTM row is the basis of one ADaM row. Things get more complicated when you need to add rows, such as to impute missing data or to create an entirely new parameter. In these more complicated cases, metadata is the primary way to explain how these derived rows differ from other rows. Sometimes BDS variable DTYPE (Derivation Type) can also be used, in addition to metadata, to describe a derivation within the data itself.

This paper describes how different levels of metadata can be used for explaining derived rows, gives some examples of when to use and not use variable DTYPE, suggests what to do when variable DTYPE isn't appropriate, and provides automatable ways to determine if variables DTYPE and PARAMTYP are being used correctly.

AUTHOR BACKGROUND

I've been on the CDISC ADaM team since 2001. I have led and co-led sub-teams, including those that wrote ADaMIG v1.1 (*Reference 3*) and OCCDS v1.0 (*Reference 5*). I joined the ADaM Leadership team in 2014, and was the ADaM Team Lead for the years 2018 and 2019. As an active participant on the ADaM team, I review all ADaM documents for adherence to the various ADaM rules. While I can't officially speak for the ADaM team, I am very familiar with the content and intent of all the ADaM documents.

DISCLAIMER REGARDING METADATA EXAMPLES

The examples within this paper do not show all components of metadata. Components relevant to the purpose of the paper are included, but unrelated content, even if required, may not be shown. The ADaM Implementation Guide (ADaMIG) (*References 2,3,4*) and Define-XML (*Reference 7*) should be referenced for more information about all necessary metadata.

INTRODUCTION TO DTYPE

Variable DTYPE has been part of the ADaMIG since version 1.0, but it seems to be often misunderstood and misused. We are to use DTYPE only when describing how a row is derived differently than the rest of the rows within a parameter, and not when describing how an entire parameter is derived.

DTYPE DOCUMENTATION FROM ADAMIG V1.2

Here is some of what the ADaMIG v1.2 (*Reference 4*) has to say about variable DTYPE.

ADaMIG v1.2 Section 3.3.5 states

> *The variable DTYPE … is to be used to identify records within a given parameter that contain these special-case analysis values. The value of DTYPE indicates the method used for populating the analysis value; a null value of DTYPE indicates the analysis value was not a special case.*

The CDISC Notes for variable DTYPE in ADaMIG v1.2 Table 3.3.5.1 gives three common situations where DTYPE is to be populated:

- *A new row is added within a parameter with the analysis value populated based on other rows within the parameter.*
- *A new row is added within a parameter with the analysis value populated based on a constant value or data from other subjects.*

- *An analysis value (AVAL or AVALC) on an existing record is being replaced with a value based on a pre-specified algorithm.*

ADaMIG v1.2 Section 3.3.5 also states:

> *In short, when the analysis value on a record within a parameter has been imputed or modified, DTYPE will indicate the method used to populate the analysis value. DTYPE would be used if there are special cases within the new parameter that should be identified. If a parameter is wholly derived, such as a Time-to-Event parameter, then it is a misapplication to populate DTYPE for all records in that parameter because, by definition, all records are derived using the same method.*

DTYPE CONTROLLED TERMINOLOGY

Variable DTYPE uses controlled terminology that is extensible. That means that you should review all of the standard terms, and use one of those if it is applicable. However, if none of the controlled terms are applicable, you can create a term for your situation.

Table 1 shows the list of DTYPE terms from the November, 2020 CDISC Controlled Terminology release (*Reference 6*):

Table 1: DTYPE Controlled Terminology

CDISC Submission Value	CDISC Synonym(s)
AVERAGE	Average
BC	Best Case
BLOCF	Baseline Observation Carried Forward
BOC	Best Observed Case
BOCF	Best Observation Carried Forward
EXTRAP	Extrapolation
HALFLLOQ	One Half of Lower Limit of Quantification
INTERP	Interpolation
LLOD	Lower Limit of Detection
LLOQ	Lower Limit of Quantification
LOCF	Last Observation Carried Forward
LOV	Last Observed Value
LVPD	Last Value Prior to Dosing
MAXIMUM	Maximum
MINIMUM	Minimum
ML	Maximum Likelihood
MOTH	Mean of Other Group
MOV	Mean Observed Value in a Group
PHANTOM	Phantom Record
POCF	Penultimate Observation Carried Forward
SOCF	Screening Observation Carried Forward
ULOD	Upper Limit of Detection
ULOQ	Upper Limit of Quantification
WC	Worst Case
WOC	Worst Observed Case
WOCF	Worst Observed Value Carried Forward
WOV	Worst Observed Value in a Group

Notice that these terms are designed to provide just a clue of how a row was derived. For example, the term AVERAGE doesn't let you know what was averaged. That's why metadata is so important!

EXAMPLES OF WHERE TO USE DTYPE

First, let's look at a couple examples of how to properly use DTYPE.

EXAMPLE 1: ADDING NEW ROWS WITHIN A PARAMETER

Consider this case where the Statistical Analysis Plan (SAP) specifies these derivations for all vital signs:

- Baseline is the average of all pre-treatment values
- Report change from baseline at analysis visits Week 2, Week 4, Week 8, and Week 12. Analyze in three separate tables using different values at these post-baseline analysis visits:
    - One table where any missing values are imputed with last prior post-baseline observation
    - One table where any missing values are imputed with the worst (highest) prior post-baseline observation
    - One table where missing post-baseline values are not imputed

This means we're going to need to create an average baseline record for each subject and parameter, plus add rows whenever any of the 4 analyzed visits is missing. Reviewing the controlled terminology for DTYPE in Table 1, we are going to need to use terms "AVERAGE", "LOCF", and "WOCF".

Example 1 Metadata

We begin by explaining, at a high-level, how rows are added to what is copied from SDTM. Table 2 shows an example of what we might include in dataset-level metadata:

Table 2: Example 1 Dataset-level Metadata

Dataset Name	Comment
ADVS	In addition to all rows copied from SDTM dataset VS, rows are added for AVISIT of "Baseline" (average of all pre-treatment values), for any missing post-baseline analysis visit ("Week 2", "Week 4", "Week 8", or "Week 12") with last observation carried forward, and for any missing post-baseline analysis visit ("Week 2", "Week 4", "Week 8", or "Week 12") with worst (highest) observation carried forward.

For additional detail, we need to look at variable and value-level metadata. Table 3 shows some selected variable-level metadata, and Table 4 shows the value-level metadata for variable AVAL.

Table 3: Example 1 Variable-level Metadata

Variable Name	Derivation
PARAMCD	Set to VS.TESTCD
PARAM	Character format of ADVS.PARAMCD as per controlled terminology
VSSEQ	Null if more than one row is averaged for the baseline. Otherwise set to the value of VS.VSSEQ.
ABLFL	Set to "Y" for the baseline row (AVISIT = "Baseline"); otherwise null.
AVAL	*See value-level metadata*
BASE	Set to the baseline value of the vital sign measurement (ADVS.AVAL from the row with ADVS.ABLFL = "Y")
CHG	Set to the difference of the current vital sign measurement (ADVS.AVAL) and the baseline value (ADVS.BASE) for all post-baseline records (ADVS.ADT is greater than ADVS.ADT where ADVS.ABLFL = 'Y'). Set to null otherwise.
DTYPE	Set to "AVERAGE" on the row with AVISIT = "Baseline"; set to "LOCF" on any rows imputed as last observation carried forward, set to "WOCF" on any rows imputed as worst observation carried forward; otherwise null. LOCF and WOCF are only used when there is not a result in the visit window for AVISIT values of "Week 2", "Week 4", "Week 8", or "Week 12".

Table 4: Example 1 Value-level Metadata

Variable Name	Where	Derivation
AVAL	DTYPE = “AVERAGE”	Set to the average of AVAL across all pre-dose records (AVISITN < 0)
AVAL	DTYPE = “LOCF”	If there is no record for an AVISIT in “Week 2”, “Week 4”, “Week 8”, or “Week 12”, set to the most recent prior post-baseline result (ADVS.AVAL)
AVAL	DTYPE =”WOCF”	If there is no record for an AVISIT in “Week 2”, “Week 4”, “Week 8”, or “Week 12”, set to the worst (highest) prior post-baseline result (ADVS.AVAL)
AVAL	DTYPE is null	Set to VS.VSSTRESN

Notice in Table 4 that all values of DTYPE, both populated and null, are included. When creating metadata for AVAL, we tend to focus on the special cases, where DTYPE is populated, but don’t forget to describe what happens when DTYPE is null!

Now you could instead put all the value-level metadata (here shown in Table 4) directly into the derivation for AVAL. However, this would make the derivation quite long, so that when displayed in define.xml it would wrap and make the cell very tall.

Example 1 Data

Table 5 shows some collected SDTM data for one subject’s systolic blood pressure. Table 6 shows the same content but in an ADaM dataset with imputations for baseline and missing Week 8.

Table 5: Example 1 SDTM Data

row	VSTEST	VSSEQ	VSSTRESC	VSSTRESU
1	Systolic Blood Pressure	3821	120	mmHg
2	Systolic Blood Pressure	3822	116	mmHg
3	Systolic Blood Pressure	3823	115	mmHg
4	Systolic Blood Pressure	3824	118	mmHg
5	Systolic Blood Pressure	3825	126	mmHg
6	Systolic Blood Pressure	3826	122	mmHg
7	Systolic Blood Pressure	3827	134	mmHg

Table 6: Example 1 ADaM Data

row	PARAM	VSSEQ	AVISIT	AVISITN	ABLFL	AVAL	BASE	CHG	DTYPE
1	Systolic BP (mmHg)	3821	Screening	-4		120	117	.	
2	Systolic BP (mmHg)	3822	Run-In	-2		116	117	.	
3	Systolic BP (mmHg)	3823	Week 0	-1		115	117	.	
4	Systolic BP (mmHg)		Baseline	0	Y	117	117	0	AVERAGE
5	Systolic BP (mmHg)	3824	Week 2	2		118	117	1	
6	Systolic BP (mmHg)	3825	Week 3	2		126	117	9	
7	Systolic BP (mmHg)	3826	Week 4	4		122	117	5	
8	Systolic BP (mmHg)	3826	Week 8	8		122	117	5	LOCF
9	Systolic BP (mmHg)	3825	Week 8	8		126	117	9	WOCF
10	Systolic BP (mmHg)	3827	Week 12	12		134	117	17	

Notice the derived rows 4, 8, and 9 are the special cases: they have DTYPE values, which give a clue to how these rows came about. The rest of the rows have a missing value for DTYPE, meaning those are not special cases.

Also notice that the DTYPE values, especially the value “AVERAGE”, are not sufficient alone to understand how a row was derived. We also need the metadata, as shown in Table 2, Table 3, and Table 4, in order to fully understand what is going on with this data. However, when viewing a dataset, seeing these values in DTYPE gives a nice clue about how each row was derived.

Side note: including the VSSEQ value on the LOCF and WOCF rows provides traceability back to SDTM.

EXAMPLE 2: REPLACING A VALUE ON AN EXISTING RECORD

Consider another example, where LBSTRESC has a value of “<2” and no value in LBSTRESN. How is this handled in ADaM? The answer depends on what the SAP says:

- If the SAP does not describe how to impute the missing value, then AVAL would be null. In other words, no imputation is done. In this case, there is no need to use DTYPE.
- However, if the SAP says that when a value is less than detectable by the laboratory, represented in LBSTRESC with a less-than (“<”) symbol, then to impute the missing value to that level of detection. For example, a value of “<2” would be imputed to a numeric 2 in AVAL. Here we need to explain where this value came from, since this derivation is different than most of rows (where AVAL is copied from LBSTRESN). Looking at Table 1, the value of LLOD is the appropriate choice for DYPE.

Example 2 Metadata

Here we aren’t adding new rows, so we probably don’t need extra documentation at the dataset level.

At the variable level, in a simple dataset, where the only imputation is LLOD and it is done the same way for all parameters, we can describe this in variable-level metadata, as shown in Table 7.

Table 7: Example 2 Variable-level Metadata

Variable Name	Derivation
AVAL	Set to LB.LBSTRESN when it is non-missing. When LB.LBSTRESN is missing but LB.LBSTRESC begins with “<”, then set to the value after the “<”.
DTYPE	Set to “LLOD” when LBSTRESN is missing and LBSTRESC begins with “<”.

However, in a more complicated example, where each parameter has different imputation rules, it would be better to create value-level metadata, as show in Table 8.

Table 8: Example 2 Value-level Metadata

Variable Name	Where	Derivation
AVAL	PARAMCD = “XXXXXX” and DTYPE = “LLOD”	Set to the value after the “<” in LB.LBSTRESC
AVAL	PARAMCD = “XXXXXX” and DTYPE NE “LLOD”	Set to the value of LBSTRESN
AVAL	PARAMCD = …	

Example 2 Data

Table 9 shows some data for one subject’s lab parameter (PARAMCD = “XXXXXX”). Notice that row 5 has a LBSTRESC value of “<2” and a missing value in LBSTRESN, so AVAL is set to 2 and DTYPE is “LLOD”. The rest of the rows have a missing value for DTYPE, meaning those are not special cases.

Table 9: Example 2 Data

row	PARAMCD	VSSEQ	LBSTRESC	LBSTRESN	AVAL	AVALC	DTYPE
1	XXXXXX	1	2	2	2		
2	XXXXXX	2	3.1	3.1	3.1		
3	XXXXXX	3	2.5	2.5	2.5		
4	XXXXXX	4	2.1	2.1	2.1		
5	XXXXXX	5	<2		2		LLOD
6	XXXXXX	6	2.4	2.4	2.4		
7	XXXXXX	7	2.6	2.6	2.6		

Side note: Do not set AVALC to LBSTRESC for parameter XXXXXX. This parameter is obviously being analyzed as numeric, or there would not have been a need to impute AVAL. If you did set AVALC to LBSTRESC, you would not have a 1:1 map between AVAL and AVALC, since the AVALC value of "2" on row 1 and "<2" on row 5 would correspond to the same AVAL value of 2.

INCLUDING METADATA FOR DERIVED ROWS

The purpose of DTYPE is to provide some information within the dataset to help the reviewer understand that the value of AVAL has been derived differently than others. The content of variable DTYPE alone is often not sufficient to explain how DTYPE is derived. In Example 1, when DTYPE = "AVERAGE", we need to explain what was averaged. Was it all visits, the last 2 prior to dosing, or something else? This is where metadata comes in.

As described in the ADaM model document (*Reference 1*), metadata is always needed as a part of traceability: "It may not always be practical or feasible to provide datapoint traceability … However, metadata traceability must always clearly explain how an analysis variable was populated regardless of whether datapoint traceability is also provided." So we use DTYPE in addition to, not instead of, dataset-, variable-, value-, and even results-level metadata.

Be sure to include in the metadata a detailed explanation of how the rows have been derived.

WHEN DTYPE SHOULD NOT BE USED

While DTYPE can be very helpful in the situations described above, DTYPE is not to be used to describe how a whole parameter is derived. Let's look at some examples of derived parameters and describe what should be used instead of DTYPE.

EXAMPLE 3: DO NOT USE DTYPE FOR DERIVING BMI FROM HEIGHT AND WEIGHT

For our first example of when not to use DTYPE, consider deriving Body Mass Index (BMI) from collected height and weight. The SAP for a pediatric study states to derive a value of BMI when the subject has a non-missing value for both height and a weight at each analysis visit, using the standard formula: Weight in kg/(Height in m squared). Table 10 shows an example of some height and weight data which will be used to derive BMI.

Table 10: Example 3 Initial ADaM Dataset

row	VSSEQ	AVISIT	PARAM	AVAL
1	1	Month 1	Height (m)	1.3
2	10	Month 12	Height (m)	1.4
3	2	Month 1	Weight (kg)	28.1
4	5	Month 6	Weight (kg)	30.2
5	12	Month 12	Weight (kg)	30.9

Note that we have Month 1 and Month 12 data for both height and weight, but only Weight data for Month 6. This means we will not be able to determine BMI for Month 6, since the SAP does not include any imputation rules.

For Month 1, the BMI value will be 28.1 / (1.3 x 1.3) = 16.6

For Month 12, the BMI value will be 30.9 / (1.4 x 1.4) = 15.8

Since BMI is derived from multiple input parameters (height and weight), it will need to be designed as a new parameter. This could be described in variable-level metadata, as shown in Table 11, but it can be difficult to follow the content when collapsed into a single cell of variable-level metadata. Because the derivation of parameter BMI is quite different than the derivation of height and weight, value-level metadata as shown in Table 12 would be a better choice for AVAL: it splits out what was 1 cell of variable-level metadata into 2 rows (based on PARAM) and 2 columns (one for the "where" and the other for the derivation itself).

Table 11: Example 3 Variable-level Metadata (allowed, but not optimal)

Variable Name	Derivation
AVAL	Set to the value of VS.VSSTRESN for PARAMCD values of "HEIGHT" and "WEIGHT". For derived PARAMCD "BMI", set to the value of WEIGHT/(HEIGHT * HEIGHT), where WEIGHT is the value of AVAL for PARAMCD = "WEIGHT", HEIGHT is the value of AVAL for PARAMCD = "HEIGHT", and where the height and weight records are matched by the same value of AVISIT for the same subject.

Table 12: Example 3 Value-level Metadata (best solution)

Variable Name	Where	Derivation
AVAL	PARAMCD is "HEGHT" or "WEIGHT"	Set to the value of VS.VSSTRESN
AVAL	PARAMCD = "BMI"	Set to the value of WEIGHT/(HEIGHT * HEIGHT), where WEIGHT is the value of AVAL for PARAMCD = "WEIGHT", HEIGHT is the value of AVAL for PARAMCD = "HEIGHT", and where the height and weight records are matched by the same value of AVISIT for the same subject.

Table 13 shows the same ADaM dataset as in Table 10, but with the new BMI records added.

Table 13: Example 3 Final ADaM Dataset

row	VSSEQ	AVISIT	PARAM	AVAL
1	1	Month 1	Height (m)	1.3
2	10	Month 12	Height (m)	1.4
3	2	Month 1	Weight (kg)	28.1
4	5	Month 6	Weight (kg)	30.2
5	12	Month 12	Weight (kg)	30.9
6		Month 1	BMI (kg/m2)	16.6
7		Month 12	BMI (kg/m2)	15.8

A few things to note about the data in Table 13:

- Derived BMI rows 6 and 7 have AVISIT populated using the same terminology as the rows for height and weight. Here AVISIT was a merge key, so we can be assured the values match.

- Derived BMI rows 6 and 7 do not have a value for VSSEQ. This is because, in each case, there is more than one source row providing content into that row's AVAL.
- If you had some sort of imputation rule for the missing height, not only would you have an additional row for Month 6 in Table 13, you would also need to update the metadata shown in Table 12. This makes it even more important to have the derivation in value-level metadata, rather than combined with height and weight in variable-level metadata (Table 11).

You might be tempted to add a column for DTYPE to the data in Table 13, to explain how rows 6 and 7 were derived, but this is not how DYPE is used! Recall that the purpose of DTYPE is to explain how a row in a parameter is derived differently from other rows for the same parameter. However, AVAL is derived in exactly the same way for all rows of Parameter "BMI (kg/m2)", so any value you would put in DTYPE would be the same on all rows for the parameter.

Parameter value-level metadata sufficiently describes how AVAL is derived for both the collected data (height and weight) and the derive data (BMI). There are additional ways to denote within the data that the BMI parameter is derived. This topic is discussed later in the paper.

EXAMPLE 4: DO NOT USE DTYPE FOR TOTALLING DOSES IN EXPOSURE

In other example of when not to use DTYPE, consider an exposure dataset, where we need to derive the total exposure to study drug. Let's first look at some data from dataset EX in Table 14.

Table 14: Example 4 SDTM EX Data

row	EXSEQ	VISIT	EXDOSE
1	1	Week 1	10
2	2	Week 2	10
3	3	Week 3	10
4	4	Week 4	9
5	5	Week 5	10

In this case, we want to create a record that holds the total amount given (10+10+10+9+10 = 49). How do we do that in a BDS dataset?

First, we need to create this as a new row, since it will be derived from multiple rows.

Second, we need to decide whether to use the same parameter as the Week 1 – Week 5 rows, or a different parameter. The Week 1 – Week 5 rows would likely be analyzed together, such as to create a graphical display, but the total dose taken would never be analyzed with those individual weeks. This means we should create a separate parameter to hold the total, as shown in Table 15, rather than use the same parameter as the weekly doses.

Table 15: Example 4 ADaM Data

row	EXSEQ	VISIT	EXDOSE	PARAM	AVAL
1	1	Week 1	10	Weekly Dose	10
2	2	Week 2	10	Weekly Dose	10
3	3	Week 3	10	Weekly Dose	10
4	4	Week 4	9	Weekly Dose	9
5	5	Week 5	10	Weekly Dose	10
6				Total Dose	49

Table 16 shows how value-level metadata is used to describe how AVAL is derived for both parameters.

Table 16: Example 4 Value-level Metadata

Variable Name	Where	Derivation
AVAL	PARAM = “Weekly Dose”	Set to the value of EX.EXDOSE
AVAL	PARAM = “Total Dose”	Set to the sum of the values of AVAL from all rows with PARAM = “Weekly Dose” for the same subject

Looking at Table 15, it might be tempting to add a DTYPE for the PARAM value of Total Dose, maybe something like “SUM”, but this is not appropriate. Recall that DTYPE describes how a derivation differs on rows within a parameter, but if we added a DTYPE of “SUM” it would be the same on all rows within PARAM “Total Dose”.

Next we’ll see ways to denote within the data that a parameter is derived.

WHAT TO DO INSTEAD OF USING DTYPE FOR PARAMETER-LEVEL DERIVATIONS

If all rows in the parameter have the same value of DTYPE, you need to remove this content from DTYPE and find another way to show that the parameter was derived.

ADaMIG v1.0 (*Reference 2*) and ADaMIG v1.1 (*Reference 3)* include variable PARAMTYP, with a single non-extensible controlled term of “DERIVED”. PARAMTYP is used to show when an entire parameter is derived. PARAMTYP is especially nice to include when only one or a few of many parameters in a dataset are derived, such as a derived total score in a questionnaire, or derived BMI in a vital signs dataset, since it highlights those parameter during manual review of the dataset.

ADaMIG v1.1 (*Reference 3*) CDISC Notes for PARAMTYP states “This variable will be retired from the ADaMIG in the next version because it was confused with the concept of DTYPE and therefore was being misused. The variable metadata should be adequate to indicate when a parameter is wholly derived.” PARAMTYP is not included in ADaMIG v1.2 (*Reference 4*).

However, just because PARAMTYP is not included in ADaMIG v1.2, doesn’t mean it can’t still be used! Variables can be added to a dataset as long as they don’t break the rules described in ADaMIG (*References 2, 3, and 4*) sections 3.1 and 4.2, including a variable named PARAMTYP. In fact, if you’ve used PARAMTYP on older studies, it would add consistency to continue using it in newer studies.

If, on the other hand, you’re working with ADaMIG v1.1 or later and have not used PARAMTYP in prior studies, you have other options to signal within the dataset itself that a parameter is derived. For example, you can create a flag variable such DPARAMFL (“Derived Parameter Flag”) that is set to “Y” for all records of a derived parameter, and missing for other parameters.

INCLUDING METADATA FOR DERIVED PARAMETERS

Even if you include a variable in the dataset, be it DTYPE or something else, always be sure that you have sufficient metadata to describe the details of how the parameter is derived. Since the derivation is based on parameter, parameter value-level metadata is usually the most appropriate place to describe it.

Examples of parameter value-level metadata for derived parameters were shown in Table 12 and Table 16.

CHECKING DATA

CHECKING DYPE

As mentioned throughout this paper, DTYPE is only used to describe how a row is derived differently than other rows within the same parameter.

A quick way to determine if DTYPE is being used correctly is to summarize the unique values of DYTPE by parameter, such as with SAS® PROC FREQ. What you should see is a relatively small number of rows within each parameter that have DYPE populated, and a large number of rows with a null value. If all rows in the parameter have the same value of DTYPE, then DTYPE is not being used correctly.

Another way to check whether DTYPE is being used correctly is to summarize the unique values of DTYPE without regard to parameter. Most of the time, you should be able to use one of the terms in the CDISC controlled terminology, as shown in Table 1. Although DTYPE is extensible, some values entered in DTYPE that look particularly suspicious are “DERIVED”, “TOTAL” and “SUM”. In each case, these are probably trying to describe how the whole parameter was derived, not individual different rows.

If you determine that DTYPE is not appropriate, remove this content from DTYPE (or remove the whole variable if none of the content is appropriate). While you must include metadata to explain how the parameter was derived, you can also include variable PARAMTYP or something else in your dataset.

CHECKING PARAMTYP

PARAMTYP is not required and has been removed starting from ADaMIG v1.2 (*Reference 4*). However, it still can be used, and for consistency it is especially good to use if prior studies for the same submission have used PARAMTYP. If you are using variable PARAMTYP in your dataset, an easy check is to summarize the unique values of PARAMTYP by parameter, such as with SAS® PROC FREQ. Within each parameter, variable PARAMTYP must be either populated as “DERIVED” on every row or null on every row. If a parameter has some rows null and some not, PARAMTYP is not being used correctly.

CHECKING METADATA

Metadata is required to describe derivations, whether or not variables like DTYPE or PARAMTYP are included within the data.

If DYPE is appropriate, then you need to describe in the metadata how the records with this value of DTYPE are derived differently than other records. Examples 1 and 2 in this paper showed cases where DTYPE was used appropriately, including the use of value-level metadata.

If instead you have a derived parameter, then you need to describe in the metadata how the parameter is derived differently than other records. Examples 3 and 4 in this paper showed how parameter-level metadata was used to describe one parameter derived differently than others. When a whole parameter is derived, a variable such as PARAMTYP can be used in addition to (but not instead of) parameter-level metadata.

Anytime new rows are added to a dataset, check that all metadata (dataset-, variable- and value-level, as appropriate) is complete and will be understandable by reviewers.

CONCLUSION

Reminders as you prepare your dataset:

- Ensure your metadata sufficiently describes the details of how rows were derived.
- DTYPE is a useful tool in ADaM, showing how a row within a parameter is derived differently from other rows within that same parameter.
- Be sure not use to DTYPE to explain how a whole parameter is derived.
- Simple automated checks on DTYPE and PARAMTYP can quickly show you if these variables are not being used correctly

REFERENCES

1. ADaM v2.1: CDISC. 2009. Accessed March 29, 2021. https://www.cdisc.org/standards/foundational/adam/adam-v2-1.
2. ADaMIG v1.0: CDISC. 2009. Accessed March 29, 2021. https://www.cdisc.org/standards/foundational/adam/adamig-v1-0.
3. ADaMIG v1.1: CDISC. 2016. Accessed March 29, 2021. https://www.cdisc.org/standards/foundational/adam/adamig-v1-1.

4. ADaMIG v1.2: CDISC. 2019. Accessed March 29, 2021. https://www.cdisc.org/standards/foundational/adam/adamig-v12.
5. OCCDS v1.0: CDISC. 2016. Accessed March 29, 2021. https://www.cdisc.org/standards/foundational/adam/adam-structure-occurrence-data-occds-v1-0.
6. CDISC Controlled Terminology for ADaM, November 2020 release: NIH. 2020. Accessed March 29, 2021. https://evs.nci.nih.gov/ftp1/CDISC/ADaM/.
7. Define-XML (multiple versions): CDISC. Accessed March 29, 2021. https://www.cdisc.org/standards/data-exchange/define-xml.

CONTACT INFORMATION

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

Sandra Minjoe
MinjoeSandra@prahs.com