PharmaSUG 2025 - Paper DS-105

ADaM Pet Peeves: Things Programmers Do That Make Us Crazy

Nancy Brucken, IQVIA;
Sandra Minjoe, ICON PLC

ABSTRACT

Both authors have been actively involved in the Clinical Data Interchange Standards Consortium (CDISC) Analysis Data Model (ADaM) team for a long time – one for more than 20 years, and the other for over 10 years. During that time, we've led and contributed to a variety of ADaM documents, and we are both authorized CDISC ADaM trainers. Because of our extensive ADaM expertise, we each end up reviewing a lot of ADaM submissions before they are sent to regulatory agencies. This paper highlights some of the common issues that we've seen ADaM developers make. For each issue, we provide a better and/or more conformant approach.

INTRODUCTION

This paper includes six topics that we regularly see done either incorrectly or just not optimally:

- Variable Ordering
- Intermediate Datasets
- Findings About
- The Number of ADaM Datasets
- Use of DTYPE
- Parameter Categories and Qualifiers

For each of these issues, we provide a better and/or more conformant approach.

VARIABLE ORDERING

Have you ever opened an ADaM dataset looking for one of the analysis values contributing to a table, and discovered that you had to scroll through over 100 variables before finding AVAL and its associated parameter? Unlike SDTM, there is no required order for ADaM variables. The ADaM Model says that "the ordering of variables follows a logical model (not simply alphabetic)". It also goes on to say, "It is recommended that the sponsor define a convention for ordering of variables within a dataset and then apply this ordering consistently for all analysis datasets." So that leaves a lot of room for interpretation. Most companies put STUDYID and USUBJID as the first two variables in all of their ADaM datasets, but there is a considerable amount of divergence after that. So what is the best approach, and how should "a logical model" be defined?

Our recommendation is to put the important variables in a dataset, such as dataset keys and those variables used for analysis, at the beginning of the record; place variables only displayed on a listing, such as concatenations of demographic variables, at the end of the record; and define a consistent order for the rest, such as variables copied in from ADSL and SDTM, across all datasets in that class.

For example, in a BDS dataset sorted by subject, parameter and visit, start the record with STUDYID and USUBJID, followed by PARAM/PARAMCD/PARAMN, AVISIT/AVISITN, and then AVAL/AVALC. Chances are that if someone opens the dataset, they'll be searching for analysis values associated with a particular subject, parameter and visit, and that way, those variables will be immediately visible. For an OCCDS dataset, the topic variable (--TERM, --TRT) is probably going to be important, so put that at the beginning of the record immediately following STUDYID and USUBJID, along with any associated dictionary terms summarized on the tables.

Traceability variables should also be somewhere near the beginning of the record, for easy access, and located in a consistent spot across all datasets for a study. It's also helpful to have SDTM counterparts to analysis variables, such as --STRESC or VISIT, placed near the corresponding analysis variables so any differences can be easily identified.

INTERMEDIATE DATASETS

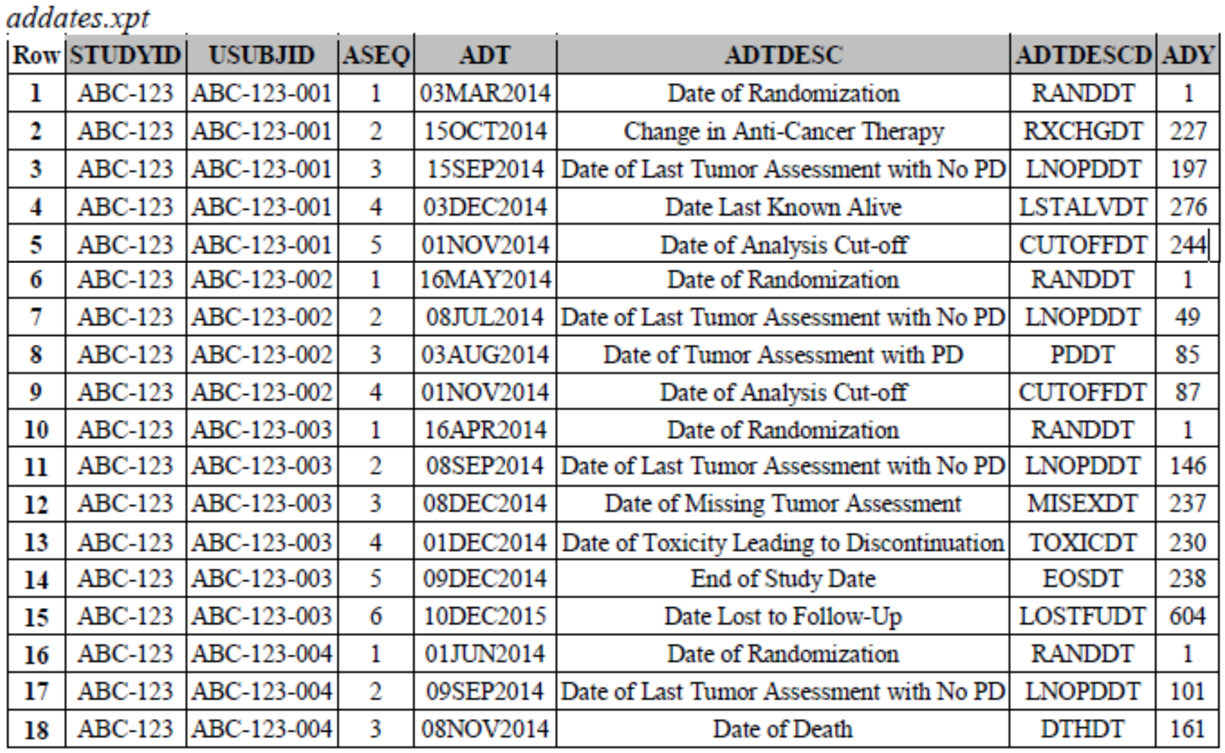
As a student, did your teachers insist on seeing not only the answer to a math problem, but the steps you followed to arrive at that solution? The same principles hold for validation/QC programmers, statisticians, reviewers, and everyone else using analysis datasets. Traceability is defined in the ADaMIG as the "property that enables the understanding of the data's lineage and/or the relationship between an element and its predecessor(s)", and is built by establishing a path from the result in a Table, Listing, or Figure (TFL) back to the records and variables from the analysis dataset used to produce that result, from there back to the SDTM dataset(s) and variable(s) used to produce that analysis dataset, and from there back to the source or collected data. Complex analysis datasets featuring variables with complicated derivations make it difficult for someone to identify and trace the links between SDTM and ADaM – the equivalent of a teacher trying to determine if you actually understood the problem, or simply made a lucky guess, when only the final answer was provided.

For example, suppose you are working on a study where overall survival (i.e., time to death) is a key endpoint. If a subject dies during the study, overall survival is calculated by taking the difference between the death date (DTHDT) and the date at which the subject became at risk for dying during the study, typically the randomization date (RANDDT) or date of informed consent (RFICDT). However, what if the subject leaves the study while they are still alive? They are censored on the date at which they are no longer at risk for dying during the study; that could be the study participation end date (RFPENDTC), discontinuation date, last assessment date, or any other date specified in the Statistical Analysis Plan (SAP). You could build your ADTTE dataset directly from SDTM, populating ADT with the correct event/censoring date and setting the censoring flag (CNSR) accordingly, and adding the SRCDOM/SRCVAR/SRCSEQ traceability variables to show where ADT came from, and that might be sufficient. On the other hand, someone who is trying to trace your path to ADT is still going to have to open up multiple SDTM datasets in order to find its potential sources. A better solution is to create an intermediate dataset containing all of the potential event and censoring date records for the study, and then using that dataset as the source of the final ADTTE dataset.

While overall survival is a relatively simple application of a survival analysis endpoint, oncology studies may have far more complicated endpoints such as progression-free survival (PFS), which require hunting through multiple datasets for potential event and censoring dates, including start dates for prohibited medications and last readable tumor scan dates. In those cases, having an intermediate dataset in which to collect all of those potential event and censoring dates, and then setting the SRCDOM/SRCVAR/SRCSEQ variables in both the intermediate and final datasets to point to their immediate predecessors provides a clear roadmap through your analysis process.

Here's an example of an intermediate dataset used for determining PFS, taken from the Prostate Cancer Therapeutic Area User Guide (TAUG):

Figure 1: Example ADDATES from Prostate Cancer TAUG



addates.xpt

Row	STUDYID	USUBJID	ASEQ	ADT	ADTDESC	ADTDESCD	ADY
1	ABC-123	ABC-123-001	1	03MAR2014	Date of Randomization	RANDDT	1
2	ABC-123	ABC-123-001	2	15OCT2014	Change in Anti-Cancer Therapy	RXCHGDT	227
3	ABC-123	ABC-123-001	3	15SEP2014	Date of Last Tumor Assessment with No PD	LNOPDDT	197
4	ABC-123	ABC-123-001	4	03DEC2014	Date Last Known Alive	LSTALVDT	276
5	ABC-123	ABC-123-001	5	01NOV2014	Date of Analysis Cut-off	CUTOFFDT	244
6	ABC-123	ABC-123-002	1	16MAY2014	Date of Randomization	RANDDT	1
7	ABC-123	ABC-123-002	2	08JUL2014	Date of Last Tumor Assessment with No PD	LNOPDDT	49
8	ABC-123	ABC-123-002	3	03AUG2014	Date of Tumor Assessment with PD	PDDT	85
9	ABC-123	ABC-123-002	4	01NOV2014	Date of Analysis Cut-off	CUTOFFDT	87
10	ABC-123	ABC-123-003	1	16APR2014	Date of Randomization	RANDDT	1
11	ABC-123	ABC-123-003	2	08SEP2014	Date of Last Tumor Assessment with No PD	LNOPDDT	146
12	ABC-123	ABC-123-003	3	08DEC2014	Date of Missing Tumor Assessment	MISEXDT	237
13	ABC-123	ABC-123-003	4	01DEC2014	Date of Toxicity Leading to Discontinuation	TOXICDT	230
14	ABC-123	ABC-123-003	5	09DEC2014	End of Study Date	EOSDT	238
15	ABC-123	ABC-123-003	6	10DEC2015	Date Lost to Follow-Up	LOSTFUDT	604
16	ABC-123	ABC-123-004	1	01JUN2014	Date of Randomization	RANDDT	1
17	ABC-123	ABC-123-004	2	09SEP2014	Date of Last Tumor Assessment with No PD	LNOPDDT	101
18	ABC-123	ABC-123-004	3	08NOV2014	Date of Death	DTHDT	161

As you can see, the dataset has been classified as following an ADaM Other structure, and contains one record for each of the important dates feeding into the PFS determination. It can then be used to build the final ADTTE dataset, which might look something like this:

Table 1: Example ADTTE Dataset using data from Prostate TAUG

STUDYID	USUBJID	PARAMCD	STARTDT	ADT	AVAL	CNSR
ABC-123	ABC-123-001	PFS	03MAR2014	15SEP2014	197	1
ABC-123	ABC-123-002	PFS	16MAY2014	03AUG2014	85	0

SRCDOM/SRCVAR/SRCSEQ are not shown here due to lack of space, but in the final ADTTE dataset, would point back to ASEQ of the record selected for analysis from the intermediate dataset. In the intermediate dataset, they would point back to the source record (SDTM or other ADaM dataset) associated with the date.

FINDINGS ABOUT

Have you ever opened up a list of datasets for a study, and noticed the presence of a dataset named ADFA, with a label of “Findings About Analysis Dataset”? Were you able to tell at a glance what that dataset might contain for that particular study, without having to dig into the Value-Level Metadata (VLM) or PARAM/PARAMCD codelists? Since almost anything can (and often does) get mapped to FA, the contents of an “ADFA” dataset can vary considerably between studies.

According to the ADaM Model, the only reserved ADaM dataset name is ADSL. Otherwise, all ADaM dataset names must be limited to a maximum of 8 characters and start with “AD”, but the rest of the name is completely up to the sponsor. Our recommendation is to provide a meaningful name for the remaining characters. For example, an ADaM dataset built from an FA dataset containing attributes of migraine events could be named ADMIGRNE with a label of “Migraine Attributes Analysis Dataset”, which provides much more information about its contents.

In addition to the problem of a likely non-informative name and label, an ADaM dataset formed by taking records from FA and adding ADSL variables also may not meet analysis needs. Data mapped to FA often consists of related values strewn across multiple records due to the structure of FA itself, and that information frequently needs to be reorganized for summarization. For example, we reviewed a study with an FA dataset consisting of 57 unique FATESTCD values. The initial ADFA dataset, formed by merging FA with ADSL, contained 57 parameters and could not produce any of the required TFLs. It turned out that FA actually consisted of 3 different types of records – a migraine history questionnaire, eCRF event-level information on the current migraine episode, and ePRO data recorded every 30 minutes for several hours after study drug administration. Each of these data types was summarized separately. In addition, there was a table summarizing derived ePRO data by migraine episode. We ended up restructuring FA into 4 separate BDS analysis datasets, each of which were analysis ready for a set of summary tables.

THE NUMBER OF ADAM DATASETS

There is nothing in the ADaM Model or ADaMIG that says there must be a 1-1 relationship between SDTM and ADaM datasets. It's entirely acceptable to split an SDTM dataset into multiple ADAM datasets for analysis purposes, even if they are not big enough to hit the maximum size allowed for regulatory submissions.

For example, QS may contain records from multiple questionnaires, all of which are scored differently and summarized separately on the tables. The ADaM ADQRS sub-team has recommended splitting QS into separate analysis datasets for each questionnaire. That makes the VLM much simpler, because the variable derivations, including total score calculations, only have to refer to parameters from a single questionnaire, and makes a potentially large QS dataset much smaller and faster to process.

Lab data represents another case where it may be helpful to create multiple analysis datasets, especially if LB contains a mixture of lab tests used for safety and efficacy analyses. The resulting efficacy lab parameters may be summarized on a different subset of subjects, and with different definitions of baseline values and analysis visits, than the safety lab parameters.

Another common situation for splitting an SDTM dataset arises with ECG data. EG often contains a combination of qualitative ECG tests, such as heart rate, QTcF and QTcB measurements, and quantitative ECG tests, such as normal heart rhythms identified. The former are often summarized on a table displaying values and changes from baseline by visit, and the latter are often simply counted. Creating a BDS dataset for the qualitative ECG results and a separate OCCDS dataset for the quantitative results leads to analysis datasets that more closely meet the analysis needs.

USE OF DYPE

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 <u>only</u> when describing how a row is derived differently than the rest of the rows within a parameter (special cases), and <u>not</u> when describing how an entire parameter is derived.

The CDISC Notes for variable DTYPE 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.

The ADaMIG also states '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."

We'll walk through both cases and describe what should be done instead.

EXTENDING DTYPE WHEN A TERM ALREADY EXISTS

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. Note that, while many of the DTYPE controlled terms are 8 characters or less, DTYPE itself can actually contain up to 200 characters.

Here is the list of DTYPE terms from the September, 2024 CDISC Controlled Terminology release:

Table 2: DTYPE Controlled Terminology from September, 2024

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
COPY	
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
NOCB	Next Observation Carried Backward; Next Value Carried Backward; NVCB
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 some clue as to 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!

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 (or AVALC) has been derived differently than others. The content of variable DTYPE alone is often not sufficient to explain how DTYPE is derived. However, 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, 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.

Invalid extensions to DTYPE Controlled Terminology

Although DTYPE is extensible, that doesn't mean we can use whatever term we want. In the above example, "AVERAGE" was not sufficient on its own to describe the details of what was done on the row. While this is easily explained in metadata, some people want to extend DTYPE to help explain what was averaged. For example, they might incorrectly use AVGALL to describe that all collected rows were averaged to create an analysis visit of "Average", and then use of AVGLAST2 to describe that only the last two non-missing collected visits were averaged to create an analysis visit of "EndPoint":

Table 3: Incorrect DTYPE Values

row	PARAM	VSSEQ	AVISIT	AVISITN	AVAL	DTYPE
1	Pulse (bpm)	101	Baseline	0	80	
2	Pulse (bpm)	102	Week 1	1	70	
3	Pulse (bpm)	103	Week 2	2	60	
4	Pulse (bpm)		Average	6	70	AVGALL
5	Pulse (bpm)		EndPoint	7	65	AVGLAST2

In this case, DTYPE controlled term AVERAGE should be used on both rows, since both were derived by averaging. As described above, metadata will fill in any gaps that DTYPE isn't able to explain.

However, if you feel the need to explain in the data how the average was derived, you could add a variable for that purpose in addition to DTYPE. For example, here is a non-standard variable named DTYPEDTL (Detail for variable DTYPE), created to hold that explanation.

Table 4: Correct DTYPE Values plus non-standard, non-required Variable DTYPEDTL

row	PARAM	VSSEQ	AVISIT	AVISITN	AVAL	DTYPE	DTYPEDTL
1	Pulse (bpm)	101	Baseline	0	80		
2	Pulse (bpm)	102	Week 1	1	70		
3	Pulse (bpm)	103	Week 2	2	60		
4	Pulse (bpm)		Average	6	70	AVERAGE	All rows averaged
5	Pulse (bpm)		EndPoint	7	65	AVERAGE	Last 2 rows averaged

Always check the CDISC controlled terminology for DTYPE. If a term exists that explains how the row was derived, use it. If that term doesn't provide as much detail as you like, like we saw here with "AVERAGE", explain in the metadata and consider adding another variable to the dataset.

DTYPE AND DERIVED PARAMETERS

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: Deriving BMI from Height and Weight

The SAP for a pediatric study states to derive a value of Body Mass Index (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)^2. Table 10 shows an example of some height and weight data which will be used to derive BMI.

Table 5: Example Height and Weight use to derive BMI

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. Table 6 shows the same ADaM dataset as in Table 5, but with the new BMI records added in rows 6-7.

Table 6: Example Height, Weight, and derived BMI

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 6:

- 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.

You might be tempted here to add a column for DTYPE to the data in Table 6, 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 <u>explain how a row in a parameter is derived differently from other rows for the same parameter</u>. 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 would sufficiently describe 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: Deriving Total Dose of Exposure

In other example of when <u>not</u> 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 SDTM dataset EX:

Table 7: Example 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, rather than use the same parameter as the weekly doses.

Here we see this done in an example ADaM dataset, which we've called ADEX. It brings in all the EX data from Table 7, above, plus adds total dose as a new parameter in row 6.

Table 8: Example ADEX with Total Dose row

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

Looking at Table 8, it might be tempting to add a DTYPE for the PARAM value of Total Dose, maybe extending the terminology with something like "SUM", but this is <u>not</u> appropriate. Recall that the purpose of DTYPE is to <u>explain how a row in a parameter is derived differently from other rows for the same parameter</u>. However, AVAL is derived in exactly the same way for all rows of Parameter "Total Dose", so the DTYPE of "SUM" would be the same on all rows for the parameter.

We've described how metadata can be used to describe how new parameters are derived, but let's now look at what we can do within the data itself.

What to do instead of using DTYPE for Parameter-level derivations

If all rows in the parameter would have the same value of DTYPE, then DTYPE is not being used correctly. DTYPE is only used to explain how a row in a parameter is derived differently from other rows for the same parameter. So what else can we do to describe how a whole parameter was derived?

ADaMIG v1.0 and ADaMIG v1.1 include variable PARAMTYP, with a single non-extensible controlled term of "DERIVED". PARAMTYP was 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 we've seen in our examples above, since it highlights those parameter during manual review of the dataset.

ADaMIG v1.1 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.

However, just because PARAMTYP is not included in ADaMIG v1.2 or later, 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 sections 3.1 and 4.2, including using 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 a newer ADaMIG 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.

Even if you include a variable in the dataset, be it DTYPE, PARAMTYP, or something else, always be sure that you have sufficient metadata to describe the details of how the row was derived.

CHECKING FOR MIS-USE OF DTYPE

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 a variable in your dataset for that purpose.

PARAMETER CATEGORIES AND QUALIFIERS

The ADaM model and all versions of the ADaMIG (v.1.0, 1.1., 1.2, 1.3) state "PARAM must include all descriptive and qualifying information relevant to the analysis purpose of the parameter". This allows analysis and review to make use of a single variable, PARAM (Parameter Name), to describe the result. That means when bringing over a test from SDTM to use in analysis, you often need to tack on content from some SDTM qualifiers to construct a parameter. Here we see a few ADaM parameters and the variables from SDTM needed to create them.

Table 9: Example ADaM Parameters Constructed from Multiple SDTM Variables

PARAM content	SDTM variables needed
Weight (kg)	VSTEST, VSSTRESU
Supine Systolic Blood Pressure (mmHg)	VSPOS, VSTEST, VSSTRESU
Urine Glucose (mg/dL)	LBCAT, LBTEST, LBSTRESU

It's good to consider whether all rows with the same PARAM value would make sense to be analyzed together. If Supine vs. Sitting blood pressure would be analyzed separately, they would need to be created as separate parameters. Since urine glucose is not analyzed with blood glucose, those must also be separate parameters.

An automated check exists in the ADaM Conformance Rules v4.0 to look for more than one baseline for a parameter. This will catch cases where you don't have PARAM fully qualified, since there you would have a baseline for each combination of PARAM and any external qualifiers. It would not be appropriate here to leave the data as is and instead choose to explain the issue in the Analysis Data Reviewer's Guide (ADRG). Instead, run automated checks early so that you can find and fix this type of issue well before developing analysis results and delivering data.

Let’s now consider disease progression results that have been determined both by the investigator and by an independent reviewer. In that case, you might have SDTM data such as shown here for one subject and visit:

Table 10: Example SDTM Response Data

RSSTRESC	QEVAL from SUPPRS
Stable Disease	Investigator
Partial Response	Independent Review Facility

When creating the ADaM dataset, it might be tempting to create a single parameter, such as “Disease Progression”, and use PARCAT1 to contain the Reviewer information, as shown here:

Table 11: Example of incorrect ADaM Response Data

~~AVALC~~	~~PARAM~~	~~PARCAT1~~
~~Stable Disease~~	~~Disease Progression~~	~~Investigator~~
~~Partial Response~~	~~Disease Progression~~	~~Independent Review Facility~~

However, assuming Disease Progression as determined by the Investigator and Disease Progression as determined by the independent reviewer are to be summarized separately, this is not ADaM-compliant. Again, automated checks would fire because there would be two baselines for this parameter: one for the Investigator Disease Progression and another for the Independent Reviewer Disease Progression. Another check would fire because the same value of PARAM is found under two different PARCAT1 values.

You can fix this issue by ensuring the parameter contains the information about the reviewer, as shown here:

Table 12: Example of correct ADaM Response Data

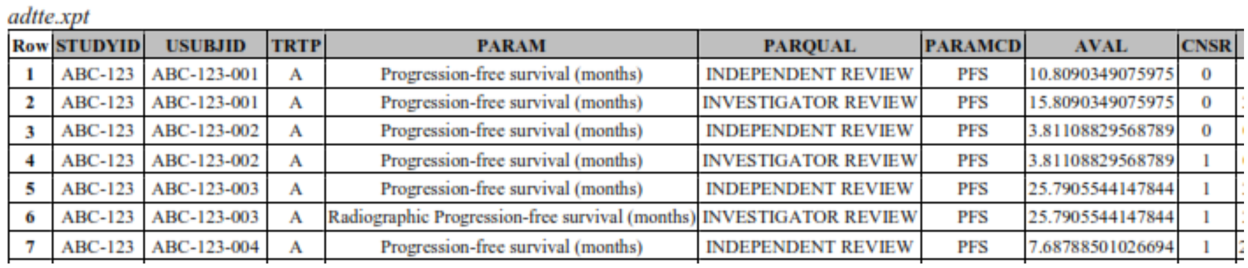
AVALC	PARAM	PARCAT1
Stable Disease	Disease Progression – Investigator	Investigator
Partial Response	Disease Progression – Independent Review Facility	Independent Review Facility

Note that as long as you include the reviewer information in the parameter, you can also put that information into a PARCATy variable, which might make it easier to subset your data, such as choosing which rows would be included in a specific table.

However, because analysis tables would likely only need the text “Disease Progression” on any rows, with the reviewer information as part of the table title, there would be some work needed to pull content out of PARAM in order to produce those tables.

To avoid that extra work and table development, you could create separate datasets, one for the Investigator review results, and another for the Independent Review Facility results. This would allow both datasets to have the same parameter name of “Disease Progression”, and, instead of a PARCATy variable, the information about who performed the review would be part of the dataset metadata. Admittedly, this solution comes with the additional overhead of maintaining two ADaM datasets instead of one.

The ADaM team has discussed whether, in cases like this, where there are multiple reviewers and their reviews are to be analyzed separately, it might be helpful to have some qualifier information allowed outside of PARAM. This would make analysis simpler when the qualifier is described in a table title or section header, rather than within the rest of the parameter. The team went as far as to create a qualifier variable named PARQUAL that was intended to be part of ADaMIG v1.2. At that time, some of the TAUGs were being developed, and PARQUAL was included in a couple of them. The FDA also uses PARQUAL in at least one of their documents. Figure 2 shows how PARQUAL is used in the Prostate Cancer TAUG to describe the assessor (“INVESTIGATOR REVIEW” or “INDEPENDENT REVIEW”):



adtte.xpt

Row	STUDYID	USUBJID	TRTP	PARAM	PARQUAL	PARAMCD	AVAL	CNSR
1	ABC-123	ABC-123-001	A	Progression-free survival (months)	INDEPENDENT REVIEW	PFS	10.8090349075975	0
2	ABC-123	ABC-123-001	A	Progression-free survival (months)	INVESTIGATOR REVIEW	PFS	15.8090349075975	0
3	ABC-123	ABC-123-002	A	Progression-free survival (months)	INDEPENDENT REVIEW	PFS	3.81108829568789	0
4	ABC-123	ABC-123-002	A	Progression-free survival (months)	INVESTIGATOR REVIEW	PFS	3.81108829568789	1
5	ABC-123	ABC-123-003	A	Progression-free survival (months)	INDEPENDENT REVIEW	PFS	25.7905544147844	1
6	ABC-123	ABC-123-003	A	Radiographic Progression-free survival (months)	INVESTIGATOR REVIEW	PFS	25.7905544147844	1
7	ABC-123	ABC-123-004	A	Progression-free survival (months)	INDEPENDENT REVIEW	PFS	7.68788501026694	1

Figure 2: Part of the Prostate Cancer TAUG ADTTE Example

There was great concern among many ADaM team members about the potential for misuse or abuse of PARQUAL, and the team was unable to come up with a solution before the release of ADaMIG v1.2 that would allow PARQUAL to be used in some situations but not in others. Due to this issue, between the draft ADaMIG v1.2 and the final version, PARQUAL was completely removed and has not yet found a home in any official ADaM document.

PARQUAL has continued to be brought up in team discussions, and different proposals have been made to "put a fence" around PARQUAL so that it would be used for only the qualifiers the ADaM team feels are appropriate. The use of non-extensible controlled terminology (CT) seemed like a good solution, and could work for something like assessor that is used in the Prostate TAUG. However, the other use case for PARQUAL where the ADaM team sees a need is with drug name, such as used in PK or exposure analysis, and it would be impossible for CT to keep with the all the possible drug names under development.

More recently, the ADaM team has been looking at a solution that adds another variable, possibly called PARQTYPE (Parameter Qualifier Type) that is subject to non-extensible controlled terminology and must be used whenever PARQUAL is used. Here we see how the pair of variables would be used:

Table 13: Proposed use of PARQUAL and PARQTYPE

PARQUAL	PARQTYPE (tentative name)
Parameter Qualifier	Parameter Qualifier Type
Contains the qualifying text needed to fully describe PARAM	Is one of the allowed types of qualifications
Not subject to CDISC CT	Uses CDISC Non-Extensible CT
Must be used with variable PARQTYPE	Must be one of the CDISC CTs

This solution would allow PARQUAL to be used only in certain specific conditions. Conformance rules would need to be developed for both PARQUAL and PARQTYPE to make it simple to discern whether PARQUAL is used correctly:

- A check can fire if PARQUAL is used without PARQTYPE
- A check can fire if PARQTYPE is not one of the terms in the CDISC CT

Additionally, there is already a CT process in place for proposing new terms to add to non-extensible CT to PARQTYPE, so the ADaM team can evaluate each proposal in a timely manner.

While it may not be perfect, this solution seems reasonable and we're fairly confident it (or something similar) will happen. If it is finalized, then any existing documents that use PARQUAL, such as the Prostate TAUG, would need updating to include PARQTYPE. Until a solution for PARQUAL is finalized, you might consider doing one of the following:

1. Do not use PARQUAL at all, but instead continue to either put all qualifying information into PARAM or create separate datasets for each "qualifier" (e.g., one dataset for investigator review and a separate dataset for independent review).
    - Advantage: breaks no rules
    - Disadvantage: might make programming some outputs a little more difficult

2. Use PARQUAL only in situations that the ADaM team is planning to allow (Assessor, Drug Name) and be clear in the ADRG how PARQUAL was used
    - Advantage: Matches existing TAUG use cases which are likely to be part of a future standard
    - Risks: Compliance tool messages will need to be explained, plus it is possible that PARQUAL won't be part of any upcoming standard

Our recommendation is to do choice 1 if that doesn't feel too constraining, and even choice 2 doesn't seem very risky. We strongly advise against using PARQUAL for anything other than Assessor or Drug Name, since that is exactly the scenario that the ADaM team has had concerns about. For example, it would not be appropriate to put SDTM units, position, or lab panel into PARQUAL.

CONCLUSION

This paper has described several situations that we've encountered regularly while reviewing analysis data submission packages. In each case, we've provided suggestions for alternative, more conformant approaches to handling such data that we hope will prove useful in your ADaM implementation journey.

REFERENCES

CDISC ADaM documents are available at https://www.cdisc.org/standards/foundational/adam. This paper referenced the following documents, all available at this site:

- Analysis Data Model (ADaM) v2.1
- Analysis Data Model Implementation Guide (ADaMIG) v1.3
- Analysis Data Model Implementation Guide (ADaMIG) v1.2
- Analysis Data Model Implementation Guide (ADaMIG) v1.1
- Analysis Data Model Implementation Guide (ADaMIG) v1.0
- ADaM Conformance Rules v4.0

CDISC Therapeutic Area User Guides (TAUGs) are available at https://www.cdisc.org/standards/therapeutic-areas/published-user-guides. This paper referenced the following TAUG:

- Therapeutic Area User Guide for Prostate Cancer v1.0 (Provisional)

CDISC Terminology is available at https://www.cdisc.org/standards/terminology/controlled-terminology. This paper referenced the following version of ADaM Terminology:

- ADaM Terminology 2024-09-27

See the following conference papers for more information on some of the topics covered in this paper:

- Leveraging Intermediate Data Sets to Achieve ADaM Traceability. Yun, J. PharmaSUG 2019 – Paper DS-185.
- Adding Rows to a BDS Dataset: When to Use DTYPE in Addition to Metadata. Minjoe, S. PharmaSUG 2021 – Paper DS-081.
- ADaM Grouping: Groups, Categories, and Criteria. Which Way Should I Go? Shostak, J. PharmaSUG 2017 – Paper DS17.
- Proposal for New ADaM Paired Variables: PARQUAL/PARTYPE. Dennis, E., Kawohl, M., Slagle, P. PharmaSUG 2023 – Paper SS-127.

ACKNOWLEDGMENTS

We would like to thank the active members of the CDISC ADaM team who have provided insight into many of the issues presented here.

CONTACT INFORMATION

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

Nancy Brucken: Nancy.Brucken@iqvia.com

Sandra Minjoe: Sandra.Minjoe@iconplc.com

Any brand and product names are trademarks of their respective companies.