PharmaSUG 2026 - Paper DS-134

Human Beings Still Needed: Manual ADaM Checks that AI Can't Do

Sandra Minjoe, ICON plc;
Mario Widel, Independent

ABSTRACT

ADaM Conformance Rules v5.0 contains almost 2000 spreadsheet rows of automatable checks. Running that many checks might seem like it would be sufficient to ensure that data is ADaM-compliant. However, the ADaM Conformance Rules Guide states that "conformance rules are machine-readable (i.e., programmable within computer software) and capable of being implemented by ADaM users", and we are told that manual review should also be done. But what constitutes a manual review?

This paper and presentation provide many checks that are not fully automatable (they need a human component), though many of them are partially automatable. These checks are designed to help you find potential ADaM conformance issues that the fully automatable checks cannot. Adding these types of checks to your review will aid you in determining whether your ADaM data is compliant with both ADaM rules and fundamental principles.

INTRODUCTION

Automated review tools for ADaM (Analysis Data Model) have improved dramatically over the years. No longer are we dealing with hordes of false-positive issues popping up because the tool writers understood the Study Data Tabulation Model (SDTM) far better than ADaM.

Even though automated reviews have gotten better, a manual review is still supposed to be part of the process. In fact, the ADaM Conformance Rules Guide v4.0 lists examples of checks that "cannot be tested by a software program", including these in Figure 1:

- Many ADaM variables are conditionally required (i.e., if a condition is true), but some conditions are not testable by a software algorithm.
- One of the key components of ADaM is the inclusion of thorough and well-defined metadata. The thoroughness and clarity of metadata cannot be tested by a machine-readable algorithm but are necessary to enable the traceability that ADaM requires.

Figure 1: Examples not testable by a software program (*from ADaM Conformance Rules v4.0 Guide*)

So what can you do to test for issues such as these, if the tool can't do it for you? Well, that's where manual review comes in.

Other conference papers have described why manual review is important (see References at the bottom of this paper), but here we give many concrete examples on how to perform that review. We also describe automation can be done to aid in manual review, noting where it takes a human to actually look at the content to determine compliance. Many of our recommendations include example data and metadata issues, example content to review, or example SAS code to help you find possible issues.

We've broken the list of topics into two sets: one for the ADaM Fundamental Principles, and the other for some ADaM rules that can't be properly checked using just an automated tool. Many of these manual checks can be done using specs, so content can be reviewed and modified even before datasets are programmed.

FUNDAMENTAL PRINCIPLES OF ADAM

The Analysis Data Model (ADaM) v2.1 describes the fundamental principles of ADaM, copied here into Figure 2:

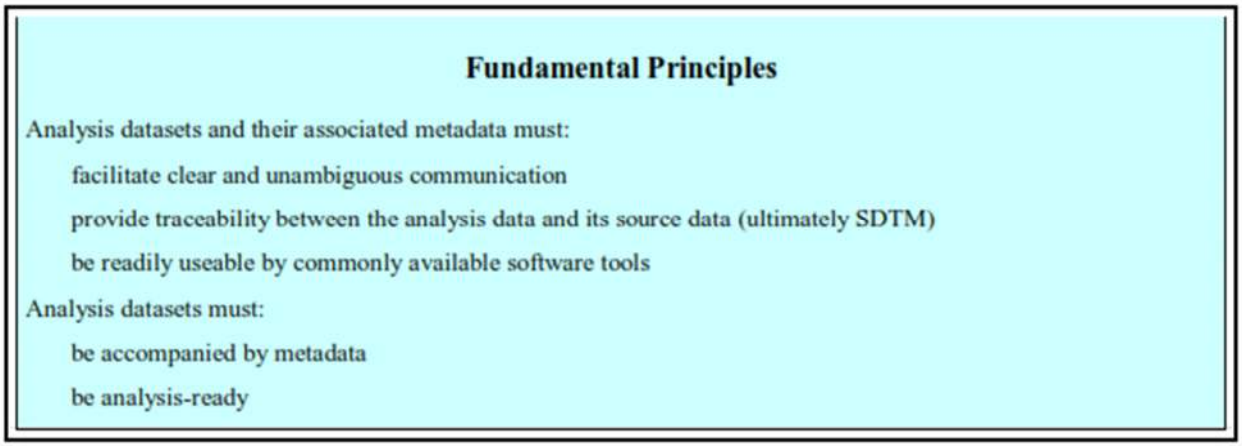

Figure 2: Fundamental Principles of ADaM (*from ADaM v2.1*)

The name "Fundamental Principles" is quite the description: these are the things that determine whether a dataset is or is not ADaM. All fundamental principles must be met for a dataset to be called ADaM. The fundamental principles are over and above any rules about dataset structure or variable names, and they are difficult to test using fully-automated tools. For example, how could a tool determine whether something is clear and unambiguous?

Let's discuss these fundamental principles one by one, and see how you can determine whether they have been met.

Fundamental Principle: Clear and Unambiguous Communication

The point of this fundamental principle is to ensure that everyone will understand how the ADaM datasets were developed and how they are used for analysis. Often this comes down to having good metadata, but good dataset names and labels, variables names and labels, and even parameter names and codes also play into this principle.

Some things you might do:

- Generate a list of unique PARAM values. Review each one to determine whether all qualifiers needed to explain AVAL part of that PARAM. For example, the SDTM LB dataset in Table 1 shows content in LBSTRESU, LBSPEC, and LBMETHOD that are not currently included in ADLB PARAM in Table 2, some of which may need to be added, depending on the analysis needs.

LBSEQ	LBTESTCD	LBTEST	LBCAT	LBSTRESC	LBSTRESU	LBSPEC	LBMETHOD
1	GLUC	Glucose	CHEMISTRY	5.1	mmo/L	SERUM	
2	GLUC	Glucose	CHEMISTRY	101	mg/dL	BLOOD	TEST STRIP
3	GLUC	Glucose	URINALYSIS	NORMAL		URINE	TEST STRIP
4	GLUC	Glucose	URINALYSIS	+2		URINE	DIPSTICK

Table 1: Example Content from SDTM LB

LBSEQ	PARAMCD	PARAM	AVAL	AVALC
1	GLUC	Glucose	5	
2	GLUC	Glucose	101	
3	GLUC	Glucose		NORMAL
4	GLUC	Glucose		+2

Table 2: Example Content from ADaM ADLB

- Generate a list of dataset names and labels, of variable names and labels, and of parameter names and codes. Would someone unfamiliar with the study but knowledgeable about ADaM find these values somewhat informative? For example, while there is no official rule that a dataset named ADVS must contain vital signs data, if it instead contained Concomitant Medications data that would not be clear and unambiguous!
- Generate a list of long derivation methods (for example, derivations longer than 80 characters) and review them. Table 3 is an example of some variables with long derivations:

Variable	Label	Origin	Derivation/Comment
PARAM	Parameter	Assigned	If LBTESTCD=GLUC and LBSPEC=SERUM and LBSTRESU=mg/dL then set to "Serum Glucose (mg/dL)"; If LBTESTCD=GLUC and LBSPEC=BLOOD and LBMETHOD=TEST STRIP and LBSTRESU=mg/dL then set to "Blood Glucose Test Strip (mg/dL)"; If LBTESTCD=GLUC and LBCAT=URINALYSIS and LBMETHOD=DIPSTICK then set to "Urine Blood Glucose Dipstick"
PARAMCD	Parameter Code	Assigned	If PARAM="Serum Glucose (mg/dL)" then set to GLUC; If PARAM="Blood Glucose Test Strip (mg/dL)" then set to GLUB; If PARAM="Urine Blood Glucose Dipstick" then set to UGLUC;

Table 3: Example Variable Metadata that might be better split into Value Level Metadata

Some things to consider:

  - Are any variable-level derivations of the format “IF-THEN-ELSE”, where it might be more appropriate to split this content into value-level metadata? (This is the case in Table 3.)
  - Should this content instead be summarized in the define.xml metadata, and point to a longer explanation in the ADRG to give the rest of the detail?

Fundamental Principle: Provide Traceability

Providing traceability means that you show where this content came from. There are two kinds of traceability: metadata and datapoint.

- Dataset, variable, and value-level metadata should contain information about the source dataset, variable, and values, which satisfies metadata traceability.
- Datapoint traceability is simple to include in many datasets, such as by keeping the SDTM --SEQ variable when copying in data to ADaM. While datapoint traceability variables are not required, they provide useful information, and, as stated in the Analysis Data Model Implementation Guide (ADaMIG) v1.3 Section 3.3.9, they “should be included whenever practical and feasible”.

Some things you might do to check for datapoint traceability:

- Check each dataset (other than ADSL) for source variables: --SEQ or the suite of SRC* variables [SRCDOM + SRCSEC in the Occurrence Data structure (OCCDS), or SRCDOM + SRCSEC + SRCVAR in the Basic Data Structure (BDS)].
  - Consider the exceptions, where you would not expect to find any source variables, such as when the entire dataset is made up of summary rows. For example, an exposure summary dataset that doesn’t contain any of the actual EX rows, but instead has parameters like “Total Dose Received (mg)” and “Duration of Exposure (min)” would never have a single row from the input dataset to point back to, so there is no need for source variables.
  - If any source variables are in the dataset, are they populated on all rows where they should be? For example, if a new row was added to contain the average of a set of collected triplicate across a timepoint, then that row with the average would not have a single sequence number to point back to. Conversely, if a new row is created by using exactly one row from the source, such as is done when using the methods of Last Observation Carried Forward (LOCF) or Worst Observation Carried Forward (WOCF),

then this value does come from a single source and a sequence number should be populated, as described in ADaMIG v1.3 Section 4.10.5.

For example, in Table 4, row 6 is a derived parameter, BMI, that was calculated in ADaM from the WEIGHT and HEIGHT values on rows 2 and 3, respectively. As a row derived from multiple other rows, BMI does not (and should not) have a value for VSSEQ.

Row	VSSEQ	PARAMCD	PARAM	AVAL	VISIT
1	10	PULSE	Pulse Rate (bpm)	72	SCREENING
2	11	WEIGHT	Weight (kg)	78.0	WEEK 1
3	12	HEIGHT	Height (cm)	181.0	WEEK 1
4	13	DBP	Systolic Blood Pressure (mmHg)	78	WEEK 1
5	14	SBP	Diastolic Blood Pressure (mmHg)	120	WEEK 1
6		BMI	BMI (kg/m2)	24.0	WEEK 1

Table 4: Example of a Derived Parameter without a sequence number populated

- In cases where a row is derived from multiple source rows, is there some other variable(s) that might help with traceability? For example, when averaging across rows, do these all have the same value across some SDTM variable? Be sure to include variables that will help anyone reviewing the dataset to see where to find the source data. Notice that in Table 4, above, the SDTM VISIT variable is populated on the derived row 6, since the value of VISIT is the same on all SDTM rows used in that derivation and makes sense for the derived row. Populating SDTM variables on derived ADaM rows is further explained in ADaMIG v1.3 Section 4.10.5.

Fundamental Principle: Be Readily Useable by Common Tools

As of this writing, the only format that meets this "Readily Useable by Common Tools" criterion is SAS v5 transport files, because it is open source. To ensure that nothing got lost when creating a SAS v5 transport file from an original SAS dataset, convert the SAS v5 transport file back to a SAS dataset and then compare the converted dataset with the original to confirm nothing changed along the way.

If another data format, such as JSON, becomes acceptable for CDISC deliveries, then a similar process would be followed: convert the JSON content back to a SAS dataset and compare to the original SAS dataset to confirm nothing changed.

Fundamental Principle: Accompanied by Metadata

Every dataset needs dataset metadata. Every variable with each dataset needs variable metadata. Value-level metadata is used when the creation of the variable content differs depending on something in the data, such as by parameter.

Some checks you can do to help you review your metadata include:

- Generate a list of variables that are derived but don't have a method. Ensure that method is missing only for those that are described in value-level metadata.
- Generate a list of variables that are assigned but don't have a comment. Should a comment be added to explain how the variable was assigned?
- As described above in the section "Fundamental Principle: Clear and Unambiguous Communication", generate a list of long derivations (for example, derivations longer than 80 characters) and review them.
    - Are any variable-level derivations of the format "IF-THEN-ELSE", where it might be more appropriate to split this long content into value-level metadata? Table 3, above, shows a couple cases where creating value-level metadata might be helpful.
    - Should this content instead be summarized in the variable- or value-level metadata, and point to a longer explanation in the ADRG to explain the detail?

- Look at the dataset metadata of each dataset. Is the label sufficient to describe the data? Should a comment be included to cover special cases, like sub-setting the source data, interleaving, merging multiple source datasets, or transposing?

  Table 5 shows an example where it might be helpful to include content in the "Documentation" column. On the ADAECE row, wouldn't it be helpful to understand how the AE and CE data were combined and for what purpose? On the two questionnaire datasets, wouldn't it be helpful on each row to describe the subsetting performed, probably using SDTM variable QSCAT, such as "Subset of the SDTM QS domain, including only records with QSCAT = "…"?

Dataset	Description	Clas – SubClass	Structure	Purpose	Keys	Documentation	Location
ADAECE	Adverse + Clinical Events Analysis Dataset	OCCURRENCE DATA STRUCTURE	One record per …	Analysis	STUDYID, USUBJID, …		adaece.xpt
ADQSBR23	EORTC BR23 Questionnaire Analysis Dataset	BASIC DATA STRUCTURE	One record per …	Analysis	STUDYID, USUBJID, …		adqsbr23.xpt
ADQSPRMS	PROMIS Questionnaire Analysis Dataset	BASIC DATA STRUCTURE	One record per…	Analysis	STUDYID, USUBJID, …		adqsprms.xpt

Table 5: Example Dataset Metadata that would benefit from some Documentation

Fundamental Principle: Analysis-Ready

An ADaM dataset is analysis-ready when it only needs simple steps like sorting and subsetting prior to producing the analysis results. More complex programming, like merging or deriving new variables, needs to be part of the dataset itself and not part of the analysis table or figure program.

This fundamental principle is tough to fully automate, but here are some checks you can do:

- Do a code review of table and figure programs. Prior to including the data in a statistical procedure, is any merging or transposing being done, or are any variables being derived? If so, this work should be moved into dataset programming. In some cases an additional ADaM dataset that is truly analysis-ready may need to be added.
- Check that AVISIT terminology matches what is used for analysis. In most cases it should not contain the text string "UNSCHEDULED", unless "UNSCHEDULED" is actually used on the analysis output. In Table 6, the last row shows that there are three records with AVISIT = 'UNSCHEDULED'. We would need to refer to the SAP and table mocks to determine if these should be changed.

Analysis Visit		
AVISITN	**AVISIT**	**Frequency**
1	**Baseline**	138
4	**Week 4**	137
…		
28	**Week 28**	115
30	**Follow-Up**	90
99	**UNSCHEDULED**	3

Table 6: Example of AVISIT frequency

ADAM RULES

While many of the rules provided in the ADaMIG, the ADaM Structure for Occurrence (OCCDS) Implementation Guide v1.1, the ADaMIG for Medical Devices v1.0, the ADaMIG for Non-compartmental

Analysis Input Data v1.0, and the Basic Data Structure for ADaM popPK Implementation Guide v1.0 have been successfully translated into automatable checks within the ADaM Conformance Rules (and then into the automated tool itself), there are still rules in these documents which were too difficult to automate. Below are some ideas on additional checks that require a manual component.

Generate a List of all character variables in each dataset without a Codelist

Most ADaM character variables should have terminology. Exceptions include dictionary coding and free text fields. Review the list of character variables to ensure that there are no variables missing terminology that should have it.

For example, variable AVALC should have Terminology used to perform categorical analysis. If you find parameters where AVALC is instead a text version of AVAL, as shown in the example in Table 7, and this is why it doesn't have terminology, then AVALC was not used properly (see ADaMIG v1.3 Section 3.3.4.2) and either the content in AVALC needs to be removed or the variable needs to be renamed.

Variable	Label	Key	Type	Length	Controlled Terms or Format	Origin	Derivation/Comment
AVAL	Analysis Value		float			Assigned	Some derivation…
AVALC	Analysis Value(C)		Text			Assigned	Set to character version of AVAL

Table 7: Example Variable Metadata showing AVALC not used properly

Generate a Frequency of all DTYPE content by parameter

Since the point of variable DTYPE is to describe how a row is derived differently from other rows within the parameter, most records within each parameter should have a null value for DTYPE. Review this list, looking for parameters where all or most rows have the same non-missing value of DTYPE, because this means that DTYPE was not used to explain exceptions to the rule. DTYPE values of 'DERIVED' and 'SUMMARY' are most likely trying to describe how an entire parameter was derived, which is an incorrect use of this variable. See ADaMIG Section 3.3.5 for more details on when to use (and not use) DTYPE.

Table 8 shows an example of this type of frequency list. If there are hundreds of rows for each parameter, it is reasonable that only 17 of the weight parameter were derived by average, and 48 of the temperature parameter were derived by LOCF. However, here <u>all</u> of the BMI records have a DTYPE = 'DERIVED', which is <u>not</u> appropriate.

PARAM by DTYPE				
	DTYPE (Derivation Type)			
PARAM (Parameter)	AVERAGE	LOCF	MAXIMUM	DERIVED
Height (cm)	0	48	48	0
Weight (kg)	17	48	48	0
Diastolic Blood Pressure (mmHg)	0	48	48	0
Systolic Blood Pressure (mmHg)	0	48	48	0
Temperature (C)	0	48	48	0
BMI (kg/m2)	0	0	0	336

Table 8: Example of DTYPE frequency within PARAM

Review all the Extended Terms of CDISC Controlled Terminology

This list might come out of the automated tool, but it needs manual attention. Compare each of these extended terms to the original CDISC Controlled Terminology list. Do any of these extended terms have a similar enough meaning that the CDISC term should instead be used?

Review non-standard ADaM variables

Here, when we say “non-standard ADaM variables”, we mean those that are not in the ADaM standard, not in the SDTM standard, and not a study SUPPQUAL QNAM. There are many different ways to look at these variables, such as:

- Generate a list of all non standard ADaM variables. Check each variable on this list to ensure that ADaM naming conventions from ADaMIG v1.3 Section 3.1 are used, and that all the rules for generating rows vs. columns, from ADaMIG v1.3 Section 4.2, are followed.
- Generate a list of variables that contain only the values of ‘N’, ‘Y’, or null and don’t have a variable name suffix of ‘FL’. Do any of these variables need to be renamed to end in ‘FL’? Note that SDTM variables and SUPPQUAL QNAM values that were copied unchanged and use this terminology are fine to leave without the ‘FL’ suffix, and sometimes our categorical analysis value (AVALC) might use just the values of ‘N’, ‘Y’, or null. Program 1 contains some SAS code to do this type of check:

```
data yn_vars;
set adam.adxx;
array vars{*} $ _CHARACTER_;
do i=1 to dim(vars);
   if vars{i} in ("Y","N") and
   prxmatch('/.+FL$/', vname(vars{i}))=0 then output;
end;
run;
```

Program 1: Finds variables with a name not ending in ‘FL’ that contain just 'Y' or 'N'

- Generate a list of variables that use a date, time, or datetime format but don’t have the required ADaM suffix of DT, TM, or DTM, respectively. ADaMIG v1.3 Section 3.1.5 includes the fragments DT, TM, and DTM as required suffixes, so these must be used for date, time, and datetime variables, respectively. Do any of these variables need to be renamed? Program 2 contains some SAS code to do this type of check:

```
proc sql;
create table bad_dates as
select * from dictionary.columns
where libname="ADAM" and memname= "ADXX"
and format like '%DATE%' and name not like '%DT';
quit;
```

Program 2: Finds variables with a name not ending in ‘DT’ but formatted as a date

- Generate a list of variables with the suffix of DTC that are not complete dates, then look for corresponding variables with suffix of DT and DTF to ensure imputation is being done when it should be and described appropriately within the metadata. Similar checks can be done for incomplete times and datetimes. Program 3 contains some SAS code to do this type of check:

```
data part_dates;
set adam.adxx;
array vars{*} $ _CHARACTER_;
do i=1 to dim(vars); /* find SDTM partial dates */
  if prxmatch('/.*DTC$/', vname(vars{i})) > 0
     and length(vars{i}) < 10 then output;
end;
do i=1 to dim(vars); /* find imputation flags  */
  if prxmatch('/.*DTF$/', vname(vars{i}))> 0
     and vars{i}=' ' then output;
end;
run;
```

Program 3: Finds date variables with imputation done

- Generate a list of variables with the suffix DTF, then check that you have an imputation rule in the SAP for the corresponding variable with suffix DT or DTM. Similar checks can be done for variables with the suffix TMF and corresponding variable with suffix TM or DTM.
- Generate a list of non-standard variables in OCCDS datasets and compare them with standard BDS variables. Do any of these variables not make sense in OCCDS? For example, PARAM, AVAL, ABLFL, and CRITy should not be part of OCCDS datasets.

Review Conditionally-Required variables for the dataset Class (and Sub-Class, if used)

Generate a list of all the ADaM conditionally required variables within the dataset class (and subclass, if used) that are not in your dataset. Review the CDISC notes for each of these variables to determine whether any of these meet the condition and should be included.

CONCLUSION

Manual review is not trivial, but it can be very helpful in finding issues that automated review cannot. Just like automated checks, manual checks should be done early in the development process, so that changes can be made before too much work gets done (and it becomes harder to fix the issue). Any manual checks done using specifications/metadata can be done even before programming even begins.

Most of our suggestions include an automated component to get you started, so the review isn't completely manual. However, a manual review itself may require some ADaM and/or study knowledge, often resulting in a more senior person reviewing the automated results to determine what is truly an issue that should be fixed.

Performing manual review early and often will aid you in determining ADaM compliance and prevent many last-minute discoveries that could push out deliverable timelines.

REFERENCES

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

- ADaM Conformance Rules v5.0
- ADaM Conformance Rules Guide v4.0
- Analysis Data Model (ADaM) v2.1
- Analysis Data Model Implementation Guide (ADaMIG) Version 1.3
- ADaM Structure for Occurrence (OCCDS) Implementation Guide v1.1
- ADaMIG for Medical Devices v1.0
- ADaMIG for Non-compartmental Analysis Input Data v1.0
- Basic Data Structure for ADaM popPK Implementation Guide v1.0

CDISC Terminology is available at https://www.cdisc.org/standards/terminology/controlled-terminology.

The following conference papers also talk about manual review of ADaM content:

- CDISC rules + FDA requests + PMDA requirements = Guaranteed Compliance? Goodfellow, D. https://www.lexjansen.com/phuse-us/2019/si/SI12.pdf.
- Quality Control and Validation – More than Just PROC COMPARE. Starostin, E. and Rimler, M. https://www.lexjansen.com/phuse/2019/pp/PP08.pdf.
- Complying with the ADaM Compliance Rules. Wittle, A. M. and Kirby, S. https://www.lexjansen.com/pharmasug/2018/DS/PharmaSUG-2018-DS23.pdf.

- How to Prepare High-quality Metadata for Submission. Debbeti, V. https://www.lexjansen.com/phuse-us/2018/si/SI12.pdf.
- Step up your ADaM Compliance Game. Ayyappath, R. and Oakley, G. https://www.lexjansen.com/phuse-us/2018/si/SI13.pdf.
- Watch Out For Blind Spots While Keeping Up With the Speed of Evolving Standards and Regulations. Veeragoni, S. Hegarty, P. https://www.lexjansen.com/pharmasug/2018/SS/PharmaSUG-2018-SS22.pdf.
- ADaM Compliance Starts with ADaM Specifications. Mankus, T. and Letourneau, K. https://www.lexjansen.com/pharmasug/2017/DS/PharmaSUG-2017-DS16.pdf.
- Conformance, Compliance, and Validation: An ADaM Team Lead's Perspective. Troxell, J. https://www.lexjansen.com/pharmasug/2016/DS/PharmaSUG-2016-DS12.pdf.
- A Road Map to Successful CDISC ADaM Submission to FDA: Guidelines, Best Practices & Case Studies. Jain, V. and Minjoe, S. https://www.lexjansen.com/pharmasug/2014/DS/PharmaSUG-2014-DS15.pdf.
- "Analysis-ready" – Considerations, Implementations, and Real World Applications. Lin, E. and Ding, B. https://www.lexjansen.com/pharmasug/2012/DS/PharmaSUG-2012-DS13.pdf.

ACKNOWLEDGMENTS

We want to thank Constanza Widel for her helping with some of the example content shown in this paper.

We also want to acknowledge all the people who think they don't need to do manual checks, for they inspired us to write this paper.

CONTACT INFORMATION

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

Sandra Minjoe
ICON plc
sandra.minjoe@iconplc.com

Mario Widel
Independent
mhwidel@gmail.com