Paper DS14

Data Quality in Pooled Studies: Overcoming Challenges and Ensuring Accuracy

Monika Ludwinska, Intego Group, Warsaw, Poland

ABSTRACT

Pooling data from multiple studies to accomplish an extensive knowledge of a treatment and disease or to conduct a robust analysis of drug's safety and efficacy is a commonly recognized strategy in the pharmaceutical industry. This method requires integrating data from studies, potentially having similar methodologies. Collecting data and subsequently examining it introduce unique challenges to be effectively considered. These obstacles involve maintaining consistency, compliance with CDISC standards and managing heterogeneity across studies.

The integration of such diverse datasets requires meticulous planning and coordination to ensure compatibility and accuracy. The quality of the pooled data depends on the individual studies, as consolidating can mask significant differences. This paper provides a guidance on planning a pooled database from studies with similar structure with overview, typical pitfalls and key considerations to develop a highly efficient pool. It illustrates topics including: planning, metadata compliance, data standards, programming strategies, validation and documentation.

INTRODUCTION

Pooling data involves combining data from multiple sources into a single, separate project. There are several approaches of building pooled database but this paper focuses on the ADaM-based strategy. This method involves programming individual ADaM datasets from the ADaMs of individual projects to support the required analysis. This approach is commonly used in pharmaceutical industry for various purposes. It can be utilized for integrated studies for regulatory submission support, extension studies to support the safety aspects of a drug development, exploratory analysis or publications. Although the definition may appear straightforward, it is far more complex in practice. Pooling is not just about combining data – it also involves addressing differences in study designs, definitions, metadata, formats and algorithms from multiple trials.

Regardless of the purpose, the goal and scope of pooling data remain the same: to create a harmonized, CDISC-compliant, and high quality package. However, the initial attempt to pool data from two or more studies can be overwhelming and discouraging. Programmers might assume that simply combining ADaM datasets will suffice, but this often results in more inconsistencies than harmonization. The primary challenge lies in the fact that it is highly unlikely that the same team has been working on all the individual studies and that the specifications have been written in a similar manner. Consequently, the initial harmonization effort often reveals numerous differences in variable names, labels, codelists, and algorithms. These discrepancies lead to time-consuming validation and re-derivation efforts to unify and harmonize the final deliverables.

The initial question that often arises is: how should the harmonization process be performed? Should one study be used as the reference? Should the same team that worked on the individual studies be involved in this activity, if feasible? Is it necessary to allocate significant resources to the project or should timelines be extended to accommodate the complexity? These questions, which occurs at the beginning of the process, are crucial and must be addressed promptly to ensure proper planning and a confident start of programming activity.

This paper focuses exclusively on the ADaM-based strategy for pooling data. It is applied for three studies (dummy) with similar designs and is built on lessons learned and personal experiences. It aims to identify the most efficient and effective approaches to this process, addressing challenges while maintaining quality. The paper begins with recommendations for planning pooling activities, followed by a discussion of metadata and data standards compliance issues. It outlines programming strategies and validation methods, highlighting common pitfalls and offering practical solutions to address them. Achieving a high-quality pooled database requires significant effort and the ability to overcome various obstacles. This paper seeks to provide a comprehensive overview of these steps.

ORGANIZATION OF POOLING DATA AND PROGRAMMING

To efficiently begin and progress with pooling activities, it is essential to have a well-structured plan that includes ADaM creation. This involves not only validating the data from each individual study but also reviewing their metadata and algorithms. As noted earlier, numerous issues can arise as soon as programming begins. Therefore, before diving into programming, it is beneficial to thoroughly review the data from the individual studies, gather all relevant information, and prepare a comprehensive pooling plan.

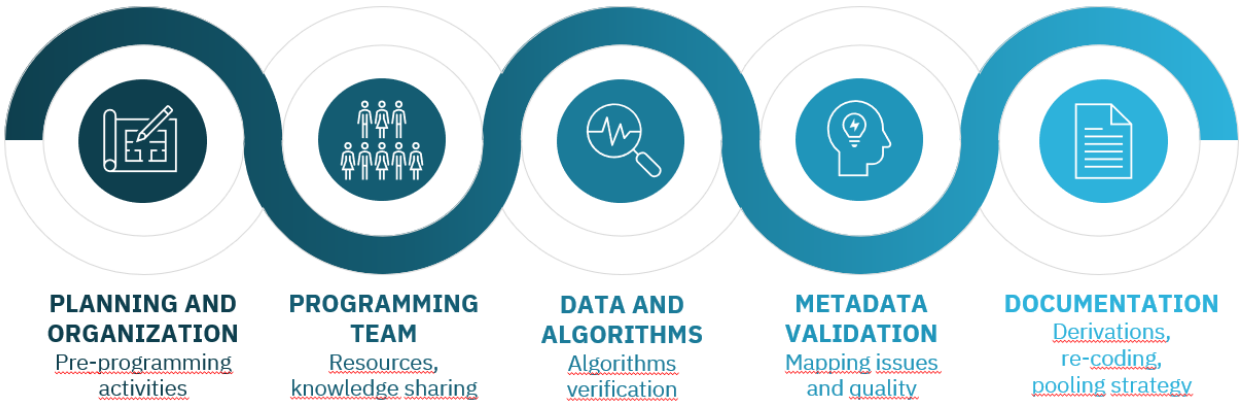

Figure 1. Diagram of organization of pooling data and programming.

PLANNING

The first step in pooling activities requires meticulous planning. The earlier planning and programming begin, the better. Along with establishing a timeline for the activity, it is important to ensure the team becomes familiar with the designs and endpoints of all individual studies, as well as any new endpoints expected for analysis. Depending on the number of pooled studies and the degree of similarity or difference in their designs, reviewing individual documentation can be a time-consuming process. Additionally, the team must carefully assess the datasets selected for pooling.

Organizing ADaMs and their metadata - for instance, in an excel spreadsheet - can help to streamline the process. Recording variable names, labels and types in one excel file enables easy filtering, identification of discrepancies, and more efficient programming of the final ADaM dataset. A critical approach to metadata review is essential, as pooled studies often reveal errors that require recalculations or other adjustments, adding to the workload. It is also helpful to centralize important documents for easy access. For instance, storing all individual define files in one location allows for quick searches of definitions or codelists, while keeping study designs in the same place aids in comparing drug periods, follow-ups, or other key aspects.

STUDY A			STUDY B			STUDY C			INTEGRATED ADaM		
VARIABLE	TYPE	LABEL	VARIABLE	TYPE	LABEL	VARIABLE	TYPE	LABEL	VARIABLE	TYPE	LABEL
BRTHDTC	char	Date/Time of Birth	BRTHDTC	char	Date/Time of Birth	BRTHDTC	char	Date/Time of Birth	**BRTHDTC**	**char**	**Date/Time of Birth**
BRTHDT	num	Analysis Date of Birth	BRTHDT	num	Date of Birth	BRTHDT	num	Date of Birth	**BRTHDT**	**num**	**Date of Birth**
		TO BE DERIVED	BRTHDTF	char	Date of Birth Imputation Flag	BRTHDTF	char	Date of Birth Imputation Flag	**BRTHDTF**	**char**	**Date of Birth Imputation Flag**
		TO BE DERIVED	ALTBL	num	ALT Level (IU/L) at Baseline	ALTBL	num	ALT Level (IU/L) at Baseline	**ALTBL**	**num**	**ALT Level (IU/L) at Baseline**
ALTBGR1	char	ALT at Baseline Group 1	ALTBLG1	char	Baseline ALT Group 1	ALTBLG1	char	ALT at Baseline Group 1	**ALTBLG1**	**char**	**ALT at Baseline Group 1**
ALTBG1N	num	ALT at Baseline Group 1 (N)	ALTBLG1N	num	Baseline ALT Group 1 (N)	ALTBLG1N	num	ALT at Baseline Group 1 (N)	**ALTBLG1N**	**num**	**ALT at Baseline Group 1 (N)**

Figure 2. Excel spreadsheet with variables from individual studies with potential issues.

Furthermore, individual studies should be validated against the Analysis Data Model Implementation Guide versions used and the appropriate dictionaries (e.g., WHO Drug, Medical Dictionary for Regulatory Activities (MedDRA)). Variations in dictionary versions across studies are common, and the general requirement is to recode the datasets using the latest version to avoid data mismatches.

As mentioned earlier, the pooling approach should be tailored based on the initial review of the data and metadata from the individual studies. Grouping ADaMs and centralizing metadata, as suggested, can help evaluate the quality of individual studies. While selecting one study as a reference may seem practical, if it contains significant errors or lacks CDISC compliance, it is often better to build the pool from scratch.

PROGRAMMING TEAM

It is a misconception that larger teams lead to better results. For ongoing or regular studies, the team size can expand based on the volume of deliverables, as new programmers can quickly familiarize themselves with key documents, such as the protocol and Statistical Analysis Plan and begin programming. However, pooled studies present additional complexities. These include an increased volume of required documents, more complex output shells and extensive metadata that frequently involves re-derivations.

In this context, a smaller, dedicated team is often more effective. With pooled studies, the number of issues and discussions grows over time, along with the team's accumulated knowledge. Sharing this knowledge with new team members later in the process can be both challenging and time-consuming, making it overwhelming for both existing and new programmers. On the other hand, the number of resources allocated should align with the deadlines to ensure timely delivery.

DATA AND ALGORITHMS VALIDATION

Data evaluation is one of the most time-consuming steps in pooling but is critical for identifying the commonalities and differences among the contributing datasets. This step provides the programming team with the information needed to normalize the data effectively. The quality of the pooled dataset is heavily dependent on the quality of the individual studies and the team's ability to address inconsistencies.

Sometimes, there may not be a reliable reference study due to significant inconsistencies or errors. In such cases, relying on an inaccurate reference could decrease the quality of the pooled dataset. Therefore, team members should not fully trust the algorithms in individual studies. Instead, it is recommended to validate each variable and its values against the algorithms provided in the define files, as well as SDTMs. Issues such as mapping errors or inconsistent laboratory test units may arise, requiring careful validation. At first glance, it might seem straightforward to create a pooled variable by simply copying values from individual studies into the pooled dataset. However, differing algorithms across studies can complicate this process. Even when algorithms appear consistent, some individual studies may include additional, undocumented steps that require further validation and adjustments by the programming team.

METADATA VALIDATION

The programming team may consider copying the metadata of the variable X (name, label, length, type) directly from the reference study and adjusting the metadata of other studies to match. This method can save time but should only be used if the reference study is of high quality and reliability. Even in such cases, it is recommended to validate every individual variable against established standards to ensure accuracy and maintain confidence in the process.

To streamline the validation of variable names and labels, a pre-created spreadsheet can be highly effective. Instead of opening each ADaM dataset individually to locate a specific variable, programmers can use the spreadsheet to quickly identify how the variable is named and labelled across studies. This aids significantly during programming.

Codelists verification can be also time consuming, as can be collected differently in individual studies. The order of categories can vary along with text chosen, grouping can be based on different thresholds.

DOCUMENTATION

The documentation - including the Analysis Data Reviewer's Guide and other important documents for the pooled study - should be prepared with the study purpose, whether it is for integration, an extension study, or another objective. It should clearly outline the chosen strategy and provide a detailed explanation of its implementation. Additionally, the documentation should include information about the Analysis Data Model Implementation Guide versions used in the individual studies, as well as the versions of the dictionaries. Any re-derivations performed during the process should also be explicitly highlighted.

KEY FACTORS AND POTENTIAL CHALLENGES

Depending on the purpose of pooling - whether for integration, an extension study, or other exploratory analyses or publications - the key factors may vary in importance, being more or less critical depending on the specific objectives of the pooling effort.

TREATMENT ISSUE

In integration for safety and/or efficacy, the key challenge is not only combining ADaMs into a single dataset but also addressing treatment mappings, which can be complex due to variations in study designs, treatment periods, background therapies, and administered treatments. The primary issue lies in selecting the appropriate approach

to treatment variables based on the purpose of the outputs and the endpoint being considered for the pool. The Analysis Data Model Implementation Guide recommends the use of variables such as TRxxPGy/TRxxPGyN (Planned Pooled Treatment y for Period xx) and TRxxAGy/TRxxAGyN (Actual Pooled Treatment y for Period xx) (1). Depending on the specific requirements of the outputs and the designs of the studies being pooled, these variables can be adapted accordingly and validate with provided shells for displays.

BASELINE DEFINITION

One of the most critical considerations when pooling data, after establishing proper treatment variables, is defining the baseline. The baseline definition can vary depending on the purpose of pooling. Pooling data from individual studies often results in discrepancies in baseline flags because baseline definitions may differ across studies.

Integration Studies

It is possible that the integration Analysis Plan specifies a separate baseline definition. In such cases, it is recommended to include a variable like *BASETYPE* to distinguish between different baseline types, ensuring individual study data is preserved without losing traceability. If multiple baseline definitions are required within the same ADaM dataset, each baseline type should have its own set of rows in the dataset structure. This approach is essential for maintaining data traceability, retaining the original baseline data, and including the new baseline definition as needed for integration purposes.

Extension Studies

In extension studies, the primary challenge lies in clearly defining the baseline and ensuring proper mapping of SDTM data. Several scenarios for baseline definitions in extension studies may arise:

1. Baseline as the first visit in the extension study
    - In this straightforward scenario, the baseline is defined as the subject's first visit in the extension study. Mapping is relatively simple, as the baseline is the first visit recorded in the extension study for each subject.
2. Baseline as the last visit in the parent study
    - The baseline is defined as the last visit from the parent study before the subject rolls over into the extension study. This approach may encounter challenges if the study designs of the individual parent studies differ in the timing of their last visits. If the timepoints for end-of-study visits vary, the SDTM team must carefully map the last visit based on the appropriate timepoints from each individual study.
3. Baseline as the baseline visit in the parent study
    - The baseline in the extension study is defined as the baseline visit from the parent study. This requires a different mapping approach in SDTM to ensure consistency with the original baseline definition from the parent study.

DICTIONARIES VERSIONS UPGRADE

As individual studies may have been completed at different times, it is possible that the versions of MedDRA and WHO Drug dictionaries have been updated. During pooling, the previously created ADaMs need to be upgraded to the latest dictionary versions, such as in datasets like ADAE and ADCM.

The ADaM Structure for Occurrence Data (OCCDS) Implementation Guide version 1.1, paragraph 3.2.10 (*Original or Prior Coding Variables*), emphasizes the importance of maintaining traceability by preserving original variables (2). To achieve this:

- Original variables must be retained with appropriate suffixes for traceability to the original data.
- Metadata should clearly document the prior dictionary names and versions.

MISSING VARIABLES

When pooling data based on ADaM datasets, programmers may encounter missing variables in the ADaMs from individual studies. This often arises because the pooled Analysis Plan may differ from the Analysis Plans of individual studies. It is common for certain variables, such as specific baseline values required for subgroup analyses, to be absent in some ADaMs. To address this issue:

- Missing variables can be populated using data from individual SDTM datasets, provided the data is available and meets the analysis requirements.

- Alternatively, the variables can be derived from other ADaMs, provided this decision is well-documented and agreed upon (e.g., deriving specific baseline laboratory data).

ONGOING STUDIES

Pooling can be performed even when some studies are still ongoing. While this approach introduces several challenges, it also offers significant benefits for improving the quality and consistency of ongoing studies.

One of the main challenges of pooling during ongoing studies is ensuring effective communication between teams. Since ongoing studies often involve multiple programmers, the information flow can become difficult to manage. Another challenge is maintaining up-to-date ADaMs during project milestones. As data from ongoing studies is updated, the pooled ADaMs must also be revised and re-run to ensure that no discrepancies arise and that the pooled dataset remains consistent and accurate.

Despite these challenges, pooling during ongoing studies provides notable benefits. A key advantage is the additional layer of review and validation performed by the pooled study team. This team can identify and address issues in metadata and data from the ongoing studies, leading to early improvements in data quality. Resolving these issues during pooling can minimize errors and streamline the overall process.

DOCUMENTING ISSUES AND SOLUTIONS

Effective documentation is essential for maintaining transparency and consistency across data analyses. All inconsistencies, re-derivations, and modifications should be thoroughly recorded, ideally in resources such as the Analysis Data Reviewer's Guide. The documentation should begin with a clear explanation of the pooling methodology used across studies, specifying the Analysis Data Model Implementation Guide versions applied in individual studies. It is important to document all changes resulting from re-coding efforts and highlight significant codelist unifications, such as standardized visit codes. Comprehensive details of any re-derivations performed should also be included.
For BDS ADaMs, it is crucial to track and document the derivation of parameters and the harmonization approaches used. Additionally, units should be consistent across datasets, with verification of any synonyms or variations in unit terminology.

DATA STANDARDS AND METADATA COMPLIANCE

There are many aspects to be addressed when working on a pooled database, especially concerning compliance, which is a key factor in the pharmaceutical industry. When handling multiple individual studies, numerous differences can be encountered, ranging from easily noticeable issues like improper naming conventions to more complex challenges such as discrepancies in algorithms. Validating these differences in each individual study can become a time-consuming task. The following section describes the harmonizations that should be applied in pooled studies to overcome these challenges and achieve compliance efficiently.

ALGORITHMS HARMONIZATION

It is inevitable that differences will arise in the algorithms used for variable creation when working with pooled data. To ensure consistency, derivations from individual studies must be thoroughly reviewed before pooling any variable. Without this verification, pooling cannot be accurately performed, as the same variable may be derived using different logic across studies. In cases where algorithms differ, it is crucial to select the appropriate algorithm and apply re-derivation in the necessary individual studies. Programmers must ensure that the variable is derived consistently, with no acceptable differences in the algorithm. Although this process is time-consuming, it is essential for maintaining data integrity. A common example of this issue is seen with analysis flags in BDS ADaMs, where derivation approaches can vary based on factors such as visit schedules, AVAL values, and other study-specific criteria.

Algorithms validation also extend to unit validation. It is possible that baseline variables, needed for datasets such as ADSL or laboratory ADaMs, were derived using different units. This requires initial validation to ensure there are no errors in labelling and to determine whether re-derivation is necessary. An effective way to identify potential unit inconsistencies is by reviewing minimum and maximum values; unusually broad ranges can indicate mixed units. Additionally, it is important to examine individual SDTMs carefully and not rely solely on pre-existing ADaMs, as errors may have been carried over without proper validation.

VARIABLES' ATTRIBUTES HARMONIZATION

As mentioned earlier, it is preferable to develop the entire metadata from scratch rather than relying blindly on individual studies. Ensuring compliance with relevant documentation, such as the Analysis Data Model Implementation Guide, is critical. In individual studies, it is common to encounter variables that store similar information but are named differently or, conversely, variables with similar names that may not adhere to the established standards. Therefore, each variable, along with its label, must be thoroughly reviewed and cross-checked against the CDISC standards to create a compliant, high-quality pooled ADaM dataset.

Another key consideration is proper classification of variable types. For instance, float variables must be stored with appropriate precision, and discrepancies in precision can arise during the pooling process. Such inconsistencies can affect data quality and analysis outcomes. An example illustrating this issue is provided in the *programming strategies* section.

CODELISTS HARMONIZATION

After validating the attributes of variables for compliance with naming and labelling conventions, the next critical step is insuring that the proper codelist is assigned to each variable. When working with pooled data, discrepancies between codelists are inevitable, even when the variable exists consistently across individual studies. These inconsistencies may manifest in various forms, such as differences in the order of coded values, variations in decode formats (e.g., uppercase versus title case), or divergent grouping thresholds based on study-specific criteria.

To achieve consistency, all the codelists should be reviewed and aligned with the latest version released by the CDISC. This ensures that standardization is maintained across all studies within the pooled dataset. Where discrepancies are identified, codelists may require restructuring to harmonize values, formats, and grouping logic. The goal is to establish a single, standardized codelist version that applies uniformly across all the datasets, thereby enhancing data integrity and facilitating accurate analysis.

PROGRAMMING STRATEGIES

In pooling projects, it's valuable for the programming team to identify key approaches and strategies that can be applied throughout the project. One effective practice is the development of programming macros tailored to the project's needs. Allocating time for macro creation during the planning phase ensures that this effort is accounted for within the overall project timeline. These macros can later streamline various tasks, such as performing checks on individual ADaM datasets before generating the pooled ADaM, ultimately improving efficiency and consistency across the project.

PROGRAMMING SETUP

As mentioned in the introduction, gathering all individual study documents and files in a centralized location is beneficial for easy access and organization. Individual study ADaMs can be copied into a designated area and structured into separate folders for each study. This setup simplifies the assignment of programming libraries specific to each folder, ensuring efficient data handling. Additionally, this approach supports the development of pooled macros, enhancing programming efficiency. The section below demonstrates how ADaMs from individual studies can be organized into separate folders, each linked to its own library.

```
* Setting libnames to individual studies;
libname studyA "/data/pooled_project/data/studyA" access=readonly;
libname studyB "/data/pooled_project/data/studyB" access=readonly;
libname studyC "/data/pooled_project/data/studyC" access=readonly;
```

PROC MEANS

Baseline values in ADSL datasets from individual studies, such as those for laboratory tests, may be calculated using different units. When pooling the data, programmers should be mindful of these discrepancies to ensure consistency.

A reliable approach to identifying potential inconsistencies is using the SAS® procedure PROC MEANS, which helps to display the minimum and maximum values of numeric variables across individual ADaM datasets. Additionally, specifying ODS OUTPUT correctly allows programmers to capture variable names and labels for better transparency. An example of PROC MEANS usage is provided below.

```
ods output summary=studyA_min_max;
proc means data=studyA.ADSL stackods min max;
run;
ods output close;

data pool_min_max;
      set
            studyA_min_max (in=A)
            studyB_min_max (in=B)
            studyC_min_max (in=C);

      if A then study = "STUDY A";
      if B then study = "STUDY B";
      if C then study = "STUDY C";
run;
```

A laboratory test can serve as an example to demonstrate the use of this procedure across multiple studies. In this case, PROC MEANS is applied to three studies to identify values for the *XXXBL* laboratory test at baseline before pooling. The results generated by PROC MEANS for the specified variable are summarized in the table below, providing an overview of minimum and maximum values across studies.

Table 1. Laboratory test values at baseline in studies A, B and C from PROC MEANS procedure.

Variable	Label	Min	Max	Study
XXXBL	Complement C3 (g/L) at Baseline	0.590000	1.940000	STUDY A
XXXBL	Complement C3 at Baseline	0.710000	2.880000	STUDY B
XXXBL	Complement C3 (mg/dL) at Baseline	68.000000	191.000000	STUDY C

There are two key findings from the *Table 1*:

1. Study A and Study B report laboratory test values in different units than Study C.
2. Study B does not include the unit in the variable label. However, based on the values, it appears similar to Study A, likely using g/L.

There are four actions to be done by the programmer:

1. Confirm the required unit for the pooled ADaM dataset by referring to the pooled study Analysis Plan.
2. Verify the unit used in Study B, which can be cross-checked with ADLB or LB data from Study B.
3. Convert values if necessary based on the pooled SAP, ensuring consistency in the required unit for the specific laboratory test.
4. Set an appropriate label when creating the pooled variable. Since Study B lacks a unit in the label, it is advisable to include it to prevent similar issues and facilitate programming.

PROC FREQ

The usage of SAS® PROC FREQ procedure can be utilized for validation of codelists. There might be significant differences while pooling grouping variables into the pool, such as: differences in the order of controlled terms, inclusion or exclusion of units and different threshold for creation of groups. All of these potential issues should be validated before the pooled variable creation. PROC FREQ procedure can be a valuable tool for validating codelists, which are part of the study's metadata. When pooling grouping variables, several differences may arise, including:

- Variations in the order of controlled terms
- Differences in the inclusion or exclusion of units
- Different thresholds used for defining similar groups

All these potential discrepancies should be addressed before creating the pooled variable to ensure consistency and accuracy. The example below demonstrates this approach using the BMI grouping variable. A macro variable (*var*) is created to facilitate the execution of PROC FREQ within a macro. This method works effectively for Study A and Study B, as both the character and numeric grouping variables share the same prefix, "BMIBLG1", allowing for seamless processing.

```
%let var = BMIBLG1;

proc freq data=studyA;
      tables &var.N * &var. / out=studyA_CT nocum nocol norow;
run;

proc freq data=studyB;
      tables &var.N * &var. / out=studyB_CT nocum nocol norow;
run;
```

The example below, for Study C, demonstrates the same PROC FREQ procedure applied to a similar type of variable and pairing. However, in this case, the macro variable approach using a shared prefix does not work. The issue arises because the character and numeric grouping variables do not share the same prefix. Specifically:

- The character variable starts with "BMIBLGR1" to meet the 8-character length requirement.
- The numeric variable is created using the "BMIBLG1" prefix.

Due to this inconsistency, the macro logic needs to be adjusted to accommodate different prefixes when processing character and numeric grouping variables.

```
%let num = BMIBLGR1;
%let char = BMIBLG1N;

proc freq data=studyC;
      tables &num * &char / out=studyC_CT nocum nocol norow;
run;

data pool_CT;
      set
            studyA_CT (in=A)
            studyB_CT (in=B)
            studyC_CT (in=C);

      if A then STUDY = "STUDY A";
      if B then STUDY = "STUDY B";
      if C then STUDY = "STUDY C";
run;
```

Table 2. BMI values in studies A, B and C from PROC FREQ procedure.

BMIBLG1N	BMIBLG1	COUNT	PERCENT	BMIBLGR1	STUDY
.		267	.		STUDY A
1	< 30	420	92.1053		STUDY A
2	>= 30	36	7.8947		STUDY A
.		45	.		STUDY B
1	< 30	97	89.8148		STUDY B
2	>= 30	11	10.1852		STUDY B
.		647	.		STUDY C
1		904	90.4000	<30 kg/m2	STUDY C
2		96	9.6000	>=30 kg/m2	STUDY C

There are three key findings from the *Table 2*:

1. Study C does not follow The Analysis Data Model Implementation Guide v1.1 and higher, which recommends using the same truncation for character and numeric variable pairs: "*Note that GRy can be abbreviated to Gy*

when necessary to comply with the variable name length limit of 8 characters. The corresponding numeric version of the variable will use the suffix GRyN (or GyN if the Gy abbreviation is used).“ (1).

2. Study C includes units in values and does not include spaces between numeric values and mathematical operators, whereas Study A and Study B do not include units and do include spaces.
3. Subjects with missing BMI are present in all three studies.

There are four actions to be done by the programmer:

1. Confirm whether units should be included in values and whether spaces should be retained between numbers and mathematical operators. Although minor, maintaining consistency is essential when creating a pooled dataset.
2. Verify with the pooled Analysis Plan whether a "Missing" category should be explicitly included for subjects with missing BMI.
3. Decide on proper variable naming based on differences observed across studies. Consistent naming conventions should be followed to align with ADaM standards.
4. Address differences in maximum variable lengths for BMIBLG1 and BMIBLGR1 across studies, ensuring that truncation does not lead to data loss during programming.

The other example of codelists differences can be found in the *Table 3*.

Table 3. Laboratory test values in studies A and B from PROC FREQ procedure.

COVPCRN	COVPCR	COUNT	PERCENT	STUDY
.	.	271	.	STUDY A
1	POSITIVE	26	5.7522	STUDY A
2	NEGATIVE	426	94.247	STUDY A
.	.	46	.	STUDY B
2	Positive	95	88.750	STUDY B
1	Negative	12	11.2149	STUDY B

There are three key findings from the *Table 3*:

1. The order of codelists differs between Study A and Study B,
2. Decoded values are inconsistent – uppercase is used in one study while title case is applied in the other.
3. Study C lacks the variable with results present in Study A and Study B.

There are two actions to be done by the programmer:

1. Create the corresponding variable in Study C based on available data.
2. Determine the appropriate codelist order for output consistency and decide whether to display decoded values in uppercase or title case for standardized reporting.

MAXIMUM LENGTH OF CHARACTER AND INTEGER/FLOAT VARIABLES

When pooling datasets from multiple studies, variable lengths - both for character and numeric variables - can vary significantly. This can lead to issues such as truncation of character variables during data stacking in a SAS® DATA step. To prevent truncation, it's recommended to set default lengths when creating variables at the start of the DATA step. A practical approach is to:

1. Extract pooled variables from individual studies.
2. Identify the maximum length across studies.
3. Assign this maximum length when defining variables in the pooled dataset.

Numeric variables present additional challenges, as they may differ in both:

- Data type: Some datasets may store the same variable as an integer in one study and as a float in another.
- Precision: The number of decimal places can vary.

To address these issues:

- Review the pooled Analysis Plan: Check if specific precision requirements are defined.
- Programmer's judgment: If not specified, the programmer should determine an approach. The easiest is to avoid the need for re-derivations in individual studies.

The macro below extracts the maximum length of integer and decimal parts for a variable to be pooled (e.g., baseline creatinine, *CREATBL*). It processes all individual study datasets and generates a summary table showing the required precision. This macro can be extended to loop through all numeric or character variables in the dataset.

```
%let studies = studyA studyB studyC;
%let adams = adsl adsl adcore;
%let variables = CREATBL CREATBL CREATBL;

%macro checkLengthNum(studies=, adams=, variables=);

      %do i = 1 %to %sysfunc(countw(&studies));
         %let study = %scan(&studies, &i);
         %let adam = %scan(&adams, &i);
         %let var = %scan(&variables, &i);

         data &study._&var;
             set &study..&adam(keep=&var where=(&var. ^=.));
             format _all_;

            varCHAR = strip(put(&var, best.));
            lenmain = length(scan(strip(varCHAR), 1, "."));

            if index(varCHAR, '.') then lendp = length(scan(strip(varCHAR), 2, "."));
            else lendp = 0;

            keep &var varCHAR len:;
            proc sort nodupkey;
            by &var;
         run;

         proc sql noprint;
             create table numLen_&study._&var. as select
             "&var." as name length = 8, max(lenmain) + max(lendp) as LENGTH,
             max(lendp) as SIGNIFICANT,
             case when calculated significant = 0 then "Integer"
             else "Float" end as NOTE
             from &study._&var.;
         quit;
      %end;

      data pool_num;
         set
         %do i = 1 %to %sysfunc(countw(&studies));
            numLen_%scan(&studies, &i)_%scan(&variables, &i)
         %end; ;
      run;

%mend checkLengthNum;
%checkLengthNum(studies=&studies, adams=&adams, variables=&variables);
```

After running the macro on three individual studies, the output shows that the same variable is stored with different precision across studies.

Table 4. Results of macro checkLengthNum for CREATBL variable from studies A, B and C.

NAME	LENGTH	SIGNIFICANT	NOTE	STUDY
CREATBL	5	2	Float	STUDY A
CREATBL	5	2	Float	STUDY B
CREATBL	12	9	Float	STUDY C

There are two possible actions to take in the pooled dataset from the *Table 4*:

1. Enhance precision: If the goal is to improve precision, variable in Study A and Study B will require re-derivation to match the higher precision.
2. Standardize to 5.2 precision: If choosing to store the variable with 5.2 precision, only Study C will require rounding, which is simpler and avoids re-derivation in other studies.

There is no universal approach for handling these discrepancies. The decision should be based on the analysis requirements, data integrity, and programming efficiency.

CONCLUSION

Pooled studies are complex projects, involving time-consuming tasks even before actual programming begins. The integration approaches necessary to harmonize data can further complicate database structures, making them both complicated and resource-intensive. Creating metadata for a pooled study requires thorough validation from the very beginning. Developing programming from scratch, while adhering to established standards, leads to a high-quality pooled dataset and reduces the time needed for subsequent steps, such as Case Report Tabulation (CRT) review.

This paper has emphasized the importance of meticulous planning, highlighting the key activities such as assembling the programming team, conducting metadata checks, thorough documentation, and strategic planning. It has also addressed the primary challenges encountered in data pooling, including data standardization and compliance issues. Moreover, it has presented programming strategies to identify and resolve common issues in pooled studies effectively.

Despite their complexity and demands, pooled studies offer significant professional development opportunities for programmers. Allocating sufficient time for proper planning, validation, and integration not only ensures the creation of a harmonized, compliant database but also enriches the programmers' expertise and experience in handling versatile data environments.

REFERENCES

1. Analysis Data Model Implementation Guide Version 1.1.: CDISC, 2016.
2. ADaM Structure for Occurrence Data (OCCDS) Implementation Guide Version 1.1.: CDISC, 2021.
3. *The Integration Dilemma.* Tinazzi Angelo. 2022.

CONTACT INFORMATION

Author Name: Monika Ludwinska
Company: Intego Group LLC
Address: Grzybowska 78
City / Postcode: Warsaw 00-844
Work Phone: +48 222 500 032 ext. 2583
Email: monika.ludwinska@intego-group.com
Web: https://intego-group.com/