Paper DH01
Slicing Codes: A Modular approach to ADaM programming

Mahendran Venkatachalam, Novo Nordisk, Bangalore, India.

ABSTRACT

A significant amount of time and resources are required to complete the submission process for a phase 3 project with multiple trials. A lot of this effort is spent aligning the programming across several trials. In this presentation, we demonstrate a new ADaM programming experiment at Novo Nordisk that helps study teams to efficiently produce trial and submission deliverables, including DSUR, PSUR, ISS, ISE, and country-specific submissions based on a single set of ADaM programs. Our approach to developing a single set of ADaM programs uses stacked SDTM datasets for multiple Phase 3 trials within the same project area. We describe how we took advantage of R programming to slice the ADaM codes for each individual trial based on the program for the pooled ADaM datasets, resulting in reduced code duplication and streamlined programming across all Phase 3 trials. By avoiding the creation of multiple programs for the same type of ADaM datasets, we were able to optimize our programming resources and improve efficiency in the submission process.

INTRODUCTION

The conventional approach to managing trial activities involves a Trial programmer collaborating with a group of support programmers to work on ADaM programs, as well as tables, listings, and figures. If two or more phase 3 trials are running concurrently within the same project area, the approach remains consistent, with each trial having its dedicated trial programmer and a group of other programmers for support. Finally, we will combine all the phase 3 trials for submission and encountered a few issues, as listed below.

- Indeed, the issue of **alignment** is very real, and colleagues are facing significant challenges in delivering the summary documents. During the pooling phase, we are essentially addressing the dis-alignment resulting from multiple concurrent teams, which in turn adds to the workload.
- The amount of **duplicated code** in each trial is substantial. What percentage of unique code have we written for the same type of ADaM program in Trial-A compared to Trial-B?
- A significant amount of **time** and **resources** are required to complete the submission process for a phase 3 project with multiple trials.

To address the issues mentioned above, we experimented with an unconventional approach in one of our project areas. We first created pooled datasets by performing pooled ADaM programming using data from the individual phase 3 trials, and then we redistributed the programs to the individual trials.

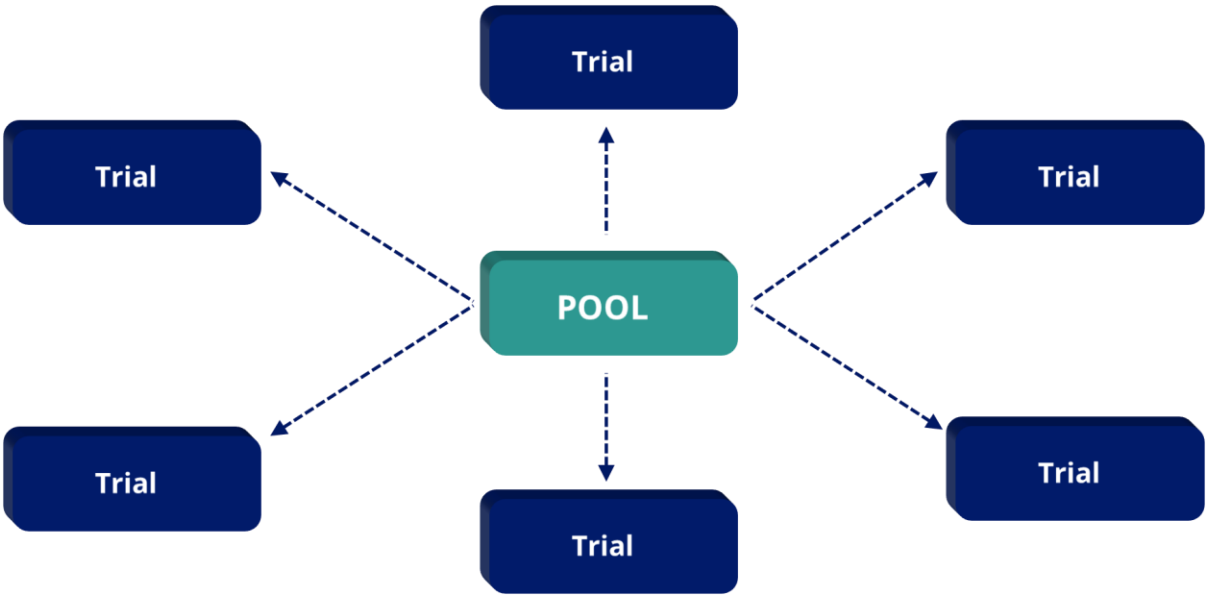


Figure 1: Re-distribution sketch from Pool to individual trial.

The idea behind the pool is to minimize the amount of code we need to write for all phase 3 trials and to prevent the creation of multiple programs for the same type of ADaM datasets. This implies that the programs in the pool are derived from the combined SDTM datasets rather than from the ADaM datasets of individual trials. We developed a system, using the R programming language, to slice/cut the codes and distribute it. All ADaM programs were divided based on their tags. The tags were utilized to redistribute the code from the pool folder to the respective individual trial folders, effectively constructing the ADaM programs for each specific trial by including only the relevant components.

The benefits of pooling at the outset of all trial activities are:

- Highly repetitive tasks.
- Submission required pooled data.
- Number of programmers are limited.
- Alignment & Save lot of time.

The pool programming activities are approached as follows:

- Pooled data from the start.
- Code is initially stored in one repository and then 'sliced' into individual repositories.
- Everyone shares responsibility for all pooled trials.
- You must coordinate – one repository.

POOL PROGRAMMING

Stacking RAW & SDTM

The stacking of RAW and SDTM datasets was performed using **R programs** within the pool instance, sourced from the individual trial folders. The need for RAW datasets is to facilitate the creation of metadata datasets, which in turn aid in the development of ADaM datasets.

ADaM programming

The vision behind the pool is that we reduce the amount of code we write across all the phase 3 trials and be align across all the trials.

The below illustration show that we have stacked all the SDTM of individual trial and done single ADaM programing activity across all the trials and redistributed codes to the individual trial folder. The ADaM programs in the pool are based on the stacked SDTM datasets.

Trial specific derivations can exist in many forms and do not always require that you restrict the derivation to an individual trial. Sometimes you can create a piece a code where all the data passes through, but only a specific trial will be affected by the code because of the special structure of data from that trial. The code written in the pool instance should be functional across all trials, for some trials, and for individual trials.

We once again employed an **R program** to segment or slice the programs into individual trials from the pool folder. The tags were used to re-distribute the code from the pool programs to the individual trial folders.

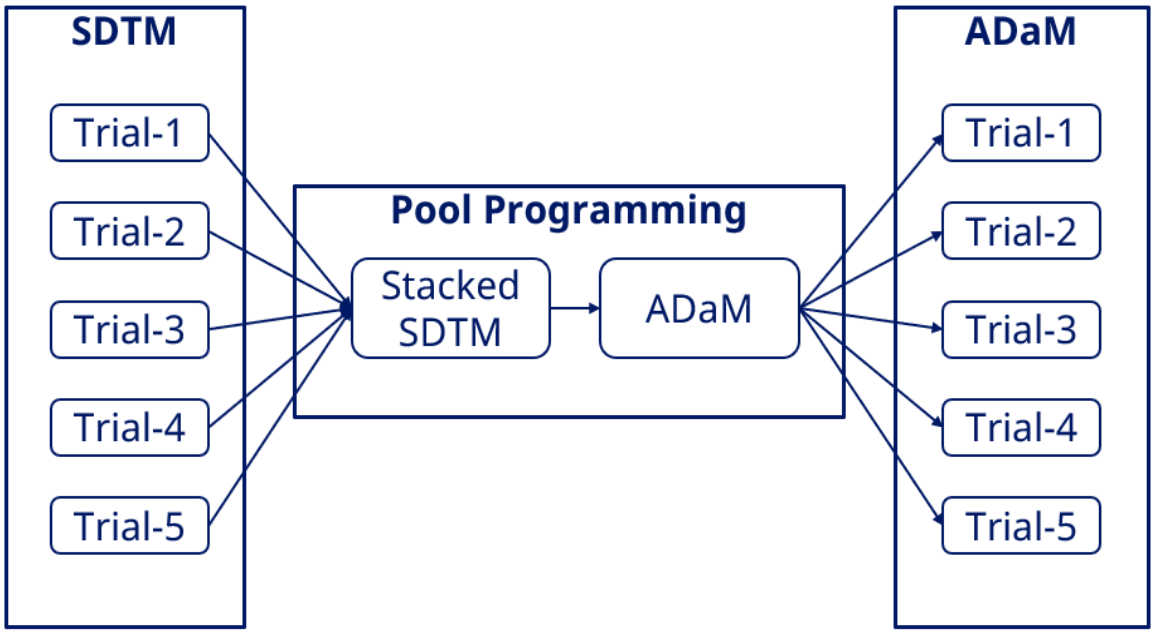


Figure 2: Stacking SDTM and re-distribution of ADaM programs to individual trial.

Tagging

Trial specific derivations must be enclosed in tags of the following nature:

- start: *>>2222; / /*>>3333*/
- end: *<<2222; / /*<<3333*/ **For SAS**

- start: #>>2222; #>>3333
- end: #<<2222; #<<3333 **For R**

To indicate any trial-specific derivation, we use two open parentheses (>>), followed by the trial number, at the beginning of the code. The derivation is concluded by closing the parentheses with (>>), followed by the trial number.

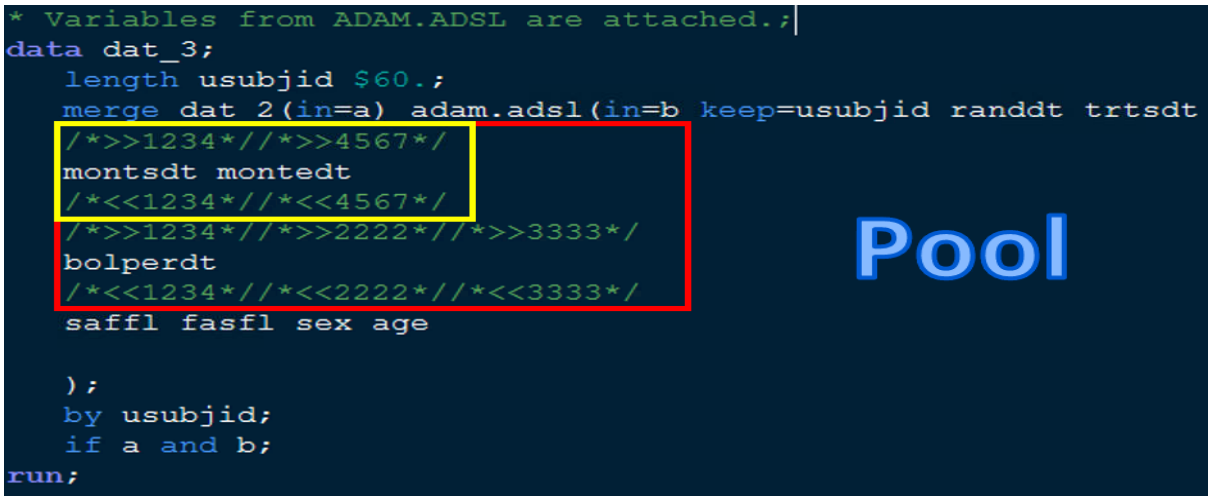


Figure 3: Tagging example in pool instance

The R programs read each line of the code as an element in an R vector and then encapsulate all these code vectors into a list. Next, the R programs identify all the entries in the vectors that contain the start and end tags we intend to keep, and we exclude entries that contain tags other than the ones we are looking for. The contrast between Figure 3 and Figure 4 illustrates how the programs appear in the pool instance and in the individual trial folder.

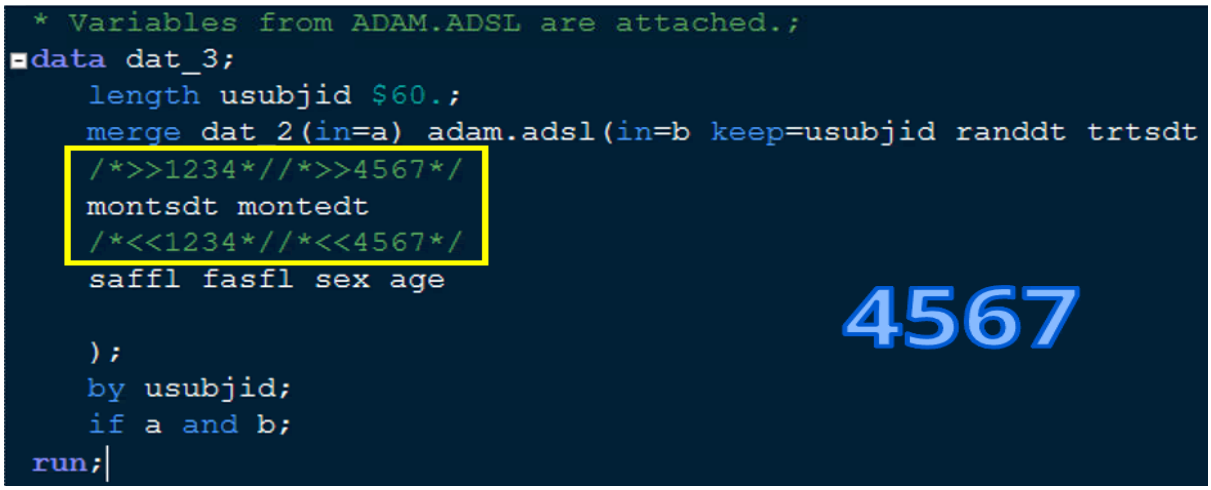


Figure 4: Program in 4567 trial after the slice from the pool instance.

Program Slice:

The illustration below demonstrates how the code chunks appear in the pool in comparison to the individual trials. All trial data passes through the piece of code written in the pool, and trial-specific derivations are marked with tags.

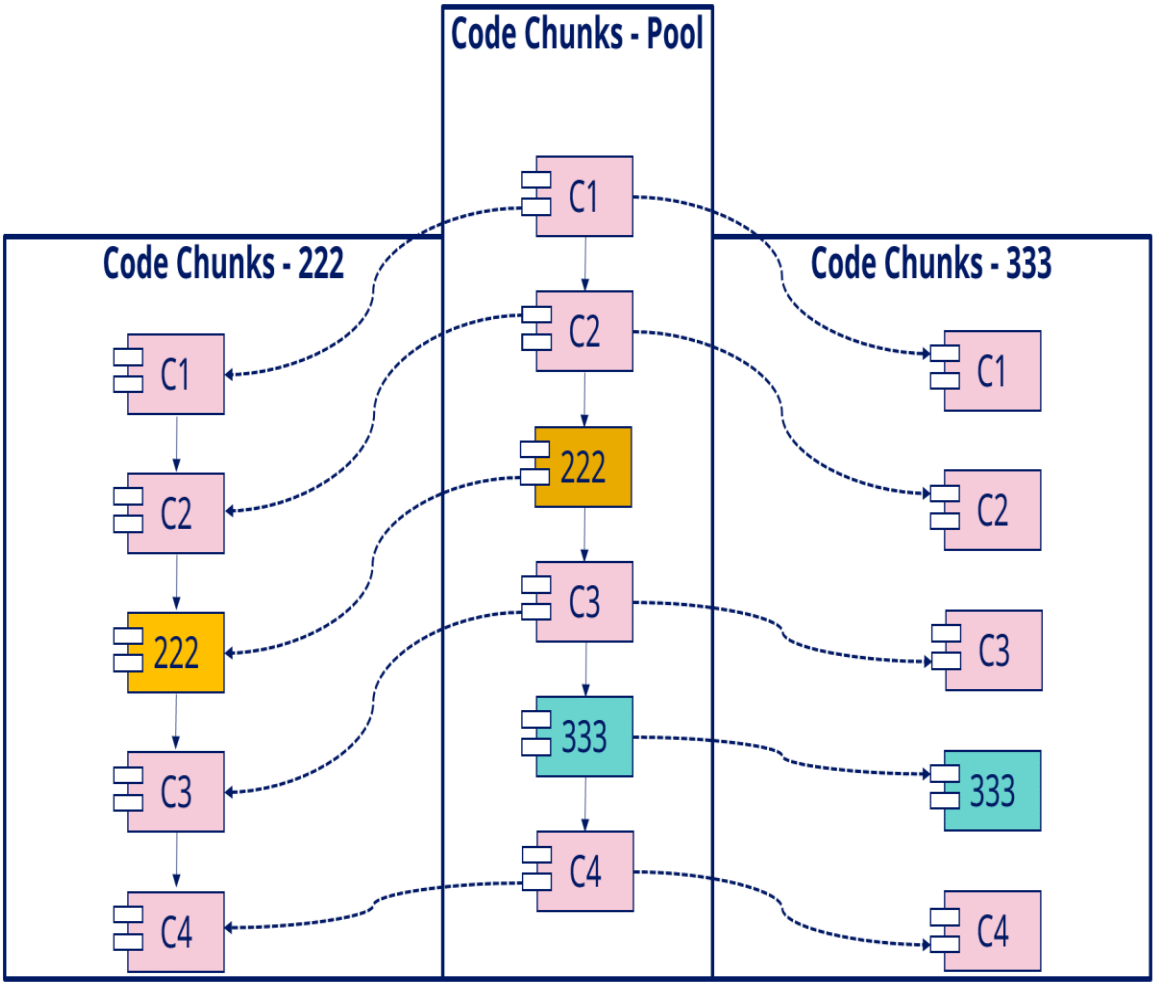


Figure 5: Code chunks in pool and individual trials as flowchart.

Figure 6 illustrates code in the pool instance with tags for trial-specific elements, while Figure 7 illustrates the code in the individual trial after slicing. This indicates that tags could be applied within the code chunks, as shown in Figure 3, and across the entire code chunks, as demonstrated in Figure 6. Its clearly seen that the code chunks missing the trial specific derivation done for the trial 222 which is not needed in trial 333. When slicing the ADaM programs, the R program slices(cut) out the tags done for the trial 222 and copied the remaining to the trial(333) folder.

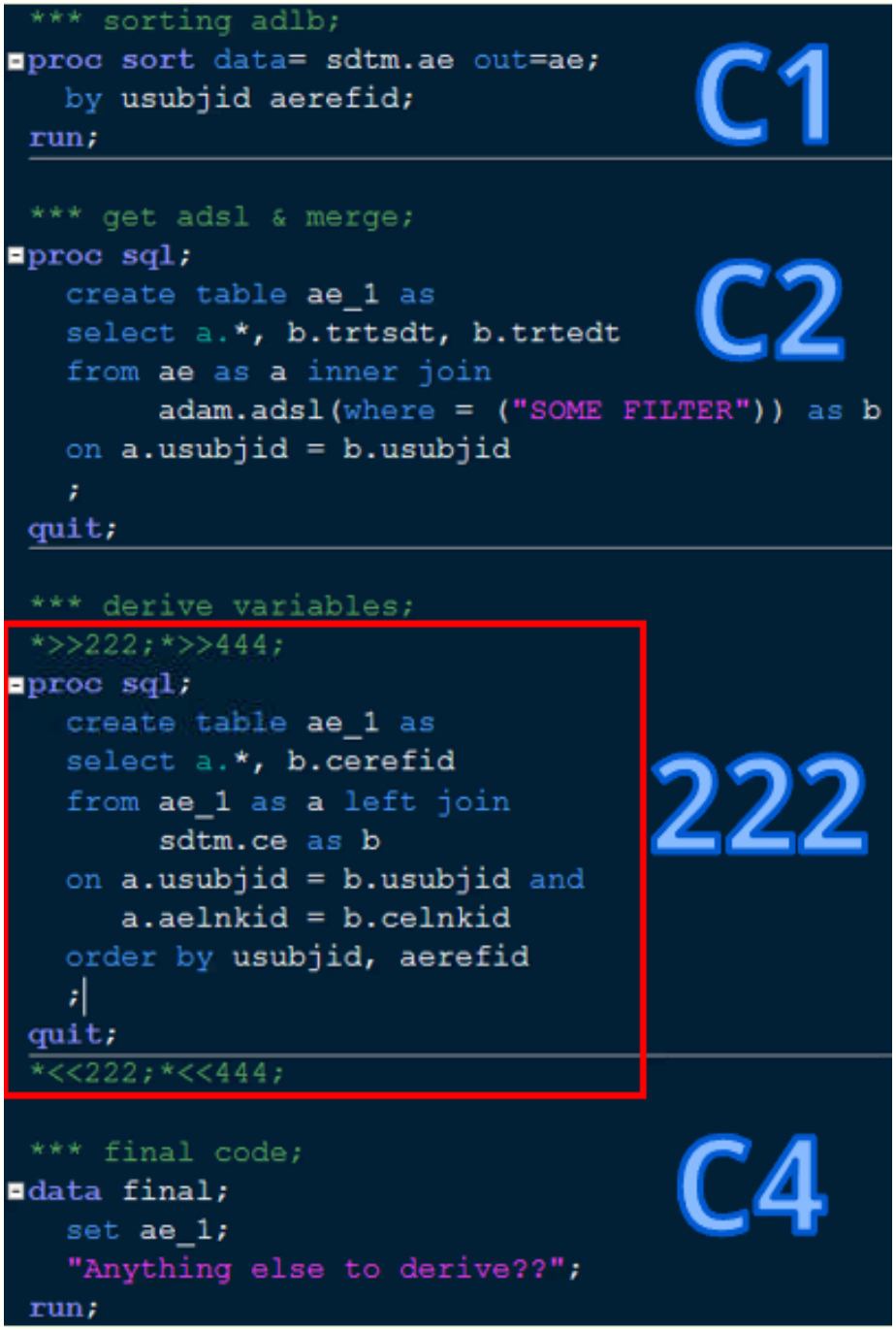


Figure 6: Code chunks real example in pool

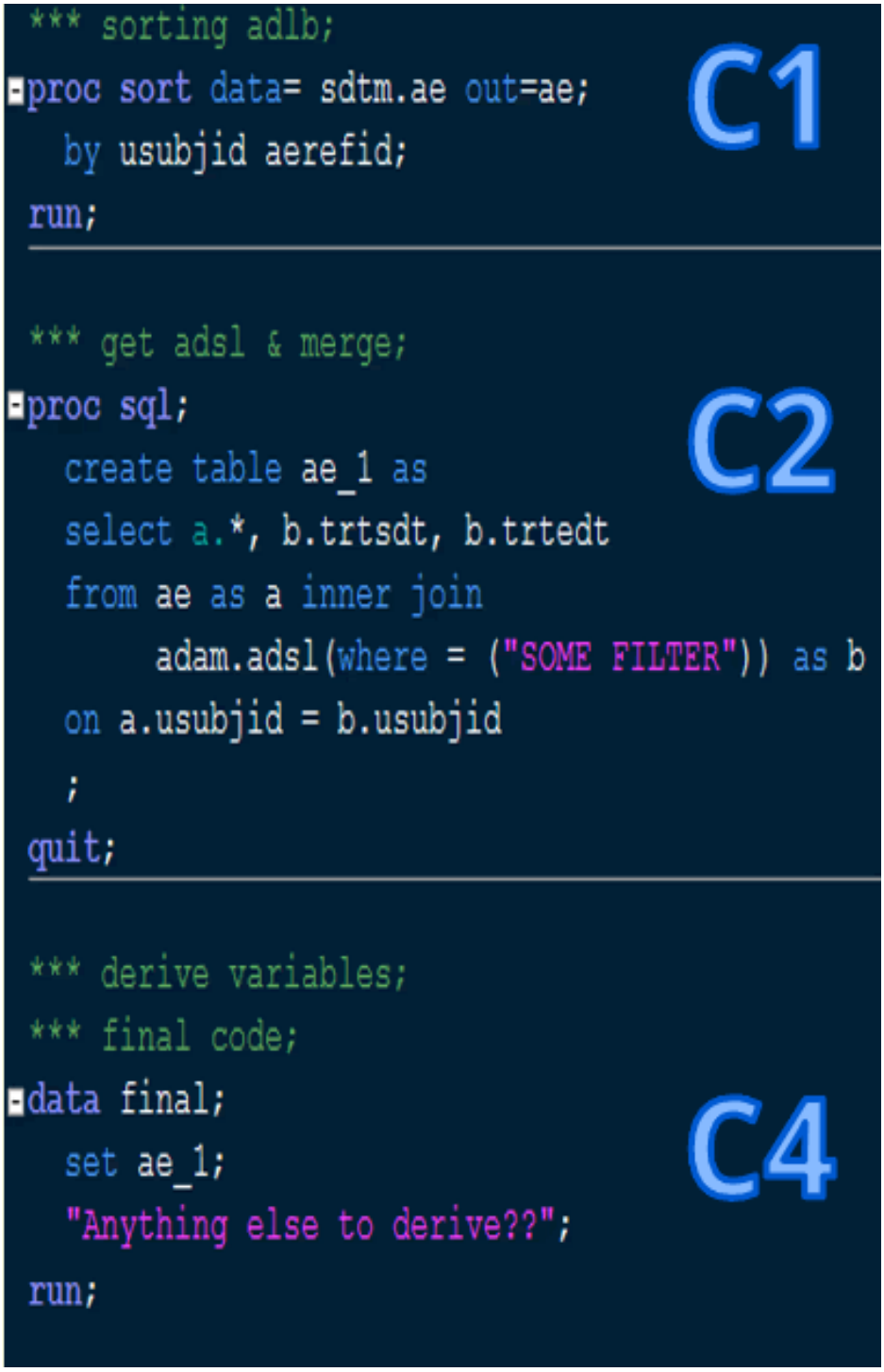


Figure 7: Code chunks real example in trial number 333

Metadata Slice:

We utilized a single specification for pooled ADaM programming, which aided in ensuring alignment across all the trials. The 'deliverable' column is meant to be filled with either trial IDs or the name of the deliverable, depending on the pool. If it's left blank, it indicates that the variable applies to all deliverables. When specific derivations for a particular deliverable are needed, you should duplicate the row containing the variable, and the 'deliverable' column must be filled accordingly. Figures 8 and 9 illustrate the specification in the pooled folder and the individual trial folder, respectively, after the slice.

include_in_trial	include_in_submission	deliverable	corefl	table	column	label
Y	Y	1234		ADSL	DCTFL	Discontinuation from Treatment Flag
Y	Y	-1234		ADSL	DCTFL	Discontinuation from Treatment Flag
Y	Y			ADSL	DCTREAS	Reason for Discontinuation of Treatment
Y	Y			ADSL	DCTREASP	Reason Specify for Discont of Treatment

Figure 8: ADaM programming specification in Pool

include_in_trial	include_in_submission	deliverable	corefl	table	column	label
Y	Y	1234		ADSL	DCTFL	Discontinuation from Treatment Flag
Y	Y			ADSL	DCTREAS	Reason for Discontinuation of Treatment
Y	Y			ADSL	DCTREASP	Reason Specify for Discont of Treatment

Figure 9: ADaM programming specification in trial 1234

Slice Call:

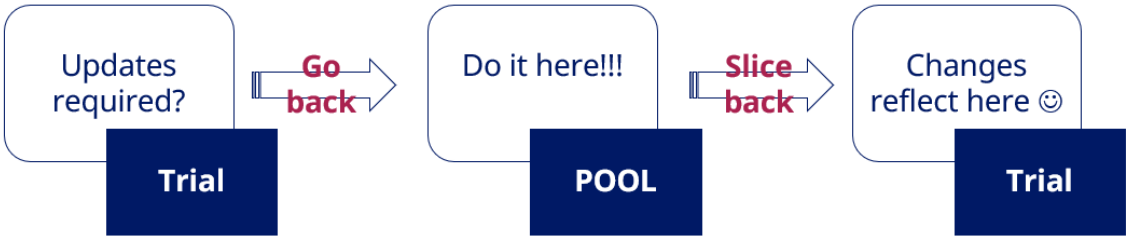
In addition to ADaM programs and ADaM specifications, we have also created macros, output programs, and parallel programs in the pooled instance, and subsequently sliced them into the individual folders. We also maintained a single MedDRA queries dataset across all the trial data for pooled programming and sliced it accordingly. Figure 10 illustrates the macro call used to copy/slice/cut the programs from the pool instance to trial 1234.

The R program copy and as well as slice the files such as .xlsx, .csv, .sas7bdat, .rds, .sas & .R

```r
# to Trial 1234 ------------------------------------------------------

trial <- "1234"

# Slice Place, CST and external data if any
metaslice(
  "place",
  "cst",
  "external_data",
  in_instance = "current",
  in_trial = "pool",
  in_project = "someNNproject",
  out_trial = trial
)

# Slice programs
slice(
  "macros",
  "metadataprog",
  "adamprog" = c("program1.R", "program2.R", "program3.R", "program4.R",
                 "program1.sas", "program2.sas", "program3.sas", "program4.sas" ),
  "statprog",
  "parprog/adamprog",
  "parprog/statprog",
  in_trial = "pool",
  in_project = "someNNproject",
  out_trial = trial
)
```

Figure 10: ADaM programming specification in trial 1234

If any changes need to be made to any programs after slicing in the trial folder, you should go to the pool instance, update the program in pool instance, and then slice it back to the respective trial. This will ensure that the changes are reflected in the trial folder.



CONCLUSION

Pool programming streamlined the development and maintenance of code and specifications, resulting in reduced redundancy and more efficient allocation of valuable time and resources in our project. The pool experiment resulted in all trial activities being prepared, including summary documents such as ISS and ISE.

With this experiment, in many of our projects, we are adopting the pool programming approach, which means that **trials are out**, and the **pools takes precedence**.

DISCLAIMER

The views and opinions expressed in this presentation belong solely to the presenter and do not necessarily reflect those of the presenter's employer, organization, committee, or any other group or individual. The purpose of this presentation is to highlight the experience gained in one of our projects. No data or confidential information belonging to Novo Nordisk has been utilized in this presentation**.**

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:
Mahendran Venkatachalam
Novo Nordisk Service Centre India Private Ltd,
Mind Comp Tech Park, 4th Floor,
EPIP Area, Whitefield
Bangalore – 560087
India
Work phone: +91-6366924741
Email: mveh@novonordisk.com
web: www.novonordisk.co.in