Paper DH04

Pre ADSL – Revolutionize your ADaM workflow

Korak Datta, AstraZeneca, Cambridge,UK

Taniya Chatterjee, AstraZeneca, Warsaw, Poland

ABSTRACT

PRE-ADSL, an ADaM OTHER class dataset introduced in ADaMIG v1.2, is a convenient location for Analysis Results or Flags that may be used in multiple derivations across ADaMs. We plan to provide real-life examples of PRE-ADSL from our real-life experiences.

While the ADaM ADSL dataset is usually the first ADaM to be created, it often contains several variables that are dependent on complex derivations driven by values or flags from other ADaM datasets. This creates a Circular Dependency problem, tricky to manage in the programming run-cycles as well as challenging to document in Specification documents. Also in certain situations, derivations in ADSL may need row-level data-point traceability from SDTM to ADaM which does not align with the ADSL structure.

We hope to convince you that using PRE-ADSL can help to de-duplicate effort, derivations, and streamline ADaM programming workflow by centralising programming logic.

INTRODUCTION

A common workflow for creation of ADaM datasets for a clinical trial typically begins with the creation of a ADaM.ADSL dataset. This dataset is crucial as it provides essential variables to subsequent ADaMs, such as treatment arm, treatment start and end dates, baseline measurements, stratification information, as well as variables key to deriving various trial endpoints. Other ADaMs then derive parts of their data from the variables inherited from ADSL. For example, the Treatment Arm is copied from ADSL, and the Actual and Planned Treatment Codes are copied and renamed to match the per-visit structure of BDS ADaMs. Treatment Start & End Date variables from ADSL can be used in derivation of On-treatment Events or Findings in ADaMs like ADTTE & ADLB and many other examples.

While being the predominant ADaM workflow, complexities such as partially blinded data, operational needs, processes, and complex derivations can trigger changes to the default method.

In this paper, we intend to illustrate two case studies, where ADSL (& other ADaMs) were broken up and created in parts, then assembled 'LEGO® style' at the end of the ADaM workflow. This approach resulted in a reduction of code complexity, enhanced traceability of intermediate data points & derivations, and provided a solution that we could then come and present at a conference!

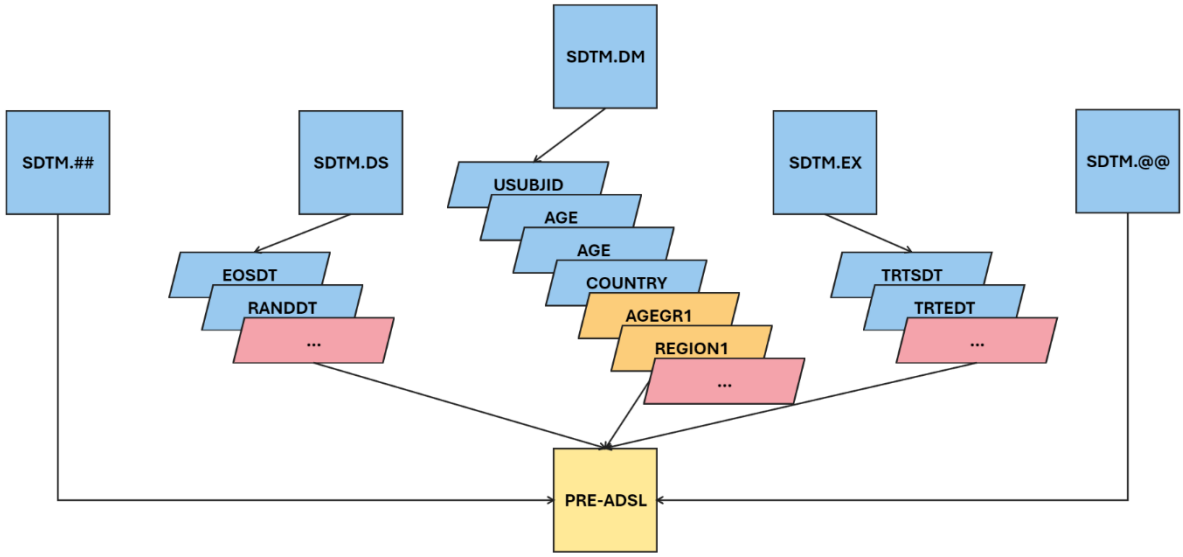
PRE-ADSL & OTHER PRE-ADAMS

As the name suggests, these are datasets which are one step before fully compliant ADaM datasets. These comply with basic tenets of CDISC for their relevant class (ADSL, BDS, ADTTE and others) however are not considered complete from an analysis & submission perspective.

These usually are mapped from their respective SDTM sources, where the source variables undergo basic transformation like renaming, formatting, mapping. They however, on principle, lack any variables which involve complex derivations – something that involves transposing, or calculations based on multiple variables or SDTM rows, etc.

For example, USUBJID would exist in a 'PRE' dataset, and so would TRT01P which is mapped from SDTM datasets. A variable like TRTSDT can also exist in a PRE-ADSL when it is simply derived from filtering records from SDTM.EX. As we will demonstrate via our examples though, this reasoning behind which variables to keep in a PRE dataset and which not to keep can depend heavily on the study situation and complexity and is a decision that can have an impact on the process flow of ADaM programming. At PHUSE EU Connect 2020, "*Flow Dependencies in ADaM Generation*" by Jeppe Rømer Juul *et. al.* explored the usage of PRE-ADSL to manage Circular Dependencies in derivation workflows.

Figure 1: PRE-ADSL Illustration



EXAMPLE 1: PRE-ADAMS AND CROSS DATASET

There are a few pieces of information in ADaM which are needed twice, once in the main ADaM dataset and in ADSL in some form. This can either be solved by deriving them in specific ADaM & then incorporating in ADSL or derive it twice – once in ADSL & again the specific ADaM. Both the solutions come with their own challenges. In the first solution, we would need to create ADSL, followed by ADaMs, and again followed by the final ADSL (to incorporate the new variables). In the second solution, we need to document the derivation twice in the specifications, as well as program it twice. Any updates during the study lifecycle will also need to be done in multiple places. The first solution presents us with an issue of recursion, which can cause issues with Computing Environments and Audit Trails. Our solution: PRE-ADaM(s) & CROSS dataset!

Sandra Minjoe explores the wider topic of the usage of intermediate datasets to ease programming workflows in the paper "*It Depends On Your Analysis Need*" at PharmaSUG 2015. A systematic approach can help ensure that common variables are managed efficiently and that derived variables are created in a consistent manner. This approach can help manage consistency and avoid issues with maintaining multiple derived variables. By creating PRE-ADSL and CROSS, we can generate the necessary variables once and then reuse them as needed, reducing the need to derive those multiple times.

In our study, we need the information coming from a questionnaire and medical history to form grouping variables in ADSL (e.g. Baseline DSQ Score Category, Prior Stricture Dilation, etc.). In case of the Prior Stricture Dilation variable in ADSL, we start by creating a PRE-ADMH dataset which is a basic ADaM for Medical History data. Then, we create a PRE-ADSCRI which is an internal dataset standard containing pre-specified Respiratory event details along with derived parameters for each event of interest, in this case 'Stricture Dilation'. We then create an intermediate dataset (CROSS) which contains the baseline value taken from PRE-ADSCRI. Finally, we merge the PRE-ADSL with CROSS to obtain the final ADSL. Similarly, we merge CROSS with other PRE-ADaMs to create the final ADaMs ensuring the ADSL common variables are populated in all final ADaMs.

Let us investigate a code snippet in Table 1 with a dummy example where we derive the grouping variable for ADSL from the PRE-ADSCRI parameter "Time Since Last Stricture Dilation".

Table 1: Example Code for Creating Grouping Variable for "Time Since Last Stricture Dilation"

```
* To create Grouping variable in ADSL based on Time since Stricture
Dilation;

data tssd1;
      set adam.pre_adscri;
      where PARAMCD eq "TSSD"; /* Time since Stricture Dilation */
      keep USUBJID PARAMCD AVAL;
run;
```

```
proc transpose data=tssd1 out=tssd2 (drop=_NAME_);
      by USUBJID;
      id PARAMCD;
      var AVAL;
run;

data tssd3(keep=USUBJID TIME GROUP1 GROUP1N);
      set tssd2;
      if TSESYS ne . then do;
            if TSESYS le @@@@ then do;
                  GROUP1 = "CO1";
                  GROUP1N = 1;
            end;
            else if TSESYS gt @@@@ then do;
                  GROUP1 = "CO2";
                  GROUP1N = 2;
            end;
      end;
run;
* @@@@ not disclosed due to data sensitivity reasons;
```

Now, let us look at another data point which we want to use from a PRE-ADaM in the final ADSL.

Figure 2: The Dysphagia Symptom Questionnaire (Version 4.0)

Question	Response options	Score
1. Since you woke up this morning, did you eat solid food?[b]	No	–
	Yes	–
2. Since you woke up this morning, has food gone down slowly or been stuck in your throat?	No	0
	Yes	2
3. For the most difficult time you had swallowing food today (during the past 24 hours), did you have to do anything to make the food go down or to get relief?	No, it got better or cleared up on its own	0
	Yes, I had to drink liquid to get relief	1
	Yes, I had to cough and/or gag to get relief	2
	Yes, I had to vomit to get relief	3
	Yes, I had to seek medical attention to get relief	4
4. The following question concerns the amount of pain you have experienced when swallowing food. What was the worst pain you had while swallowing food over the past 24 hours?[c]	None, I had no pain	0
	Mild	1
	Moderate	2
	Severe	3
	Very Severe	4

DSQ Dysphagia Symptom Questionnaire
[a]The scoring algorithm was constructed from responses to questions 2 and 3, to ensure that the final DSQ score was driven by the frequency and severity of dysphagia
[b]Responses to question 1 were unscored
[c]Responses to question 4 were not included as part of the psychometric analysis; question 4 is a standalone item on the DSQ

The Dysphagia Symptom Questionnaire (DSQ) is a patient-reported outcome measure that assesses the frequency and severity of dysphagia in patients with Eosinophilic Esophagitis (EoE) by capturing dysphagia symptoms in EoE patients. Using a form like the one displayed in Figure 2, Daily DSQ questions and their response values (when applicable) are captured in the database.

If no solid food intake is confirmed (*Q1=No*), *Q2* and *Q3* will be skipped. So, for every subject at every visit, the total score can be derived based on a varying combination of the 4 questions.

To calculate DSQ score we take the data of the patients over a 14-day baseline period. Higher scores indicate more frequent and/or severe dysphagia.

DSQ 14-day Score = Sum of Response (AVAL)x14 / Number of diary days reported with non-missing data.

We derive the DSQ score in PRE-ADQS for all relevant visits (14-day time windows). The baseline DSQ score is required in the final ADSL as a covariate for the efficacy analysis. Therefore, the baseline DSQ Score is taken from PRE-ADQS, and stored in the CROSS dataset first, and in subsequent programming steps it is merged in the final ADSL.

Table 2: Example Code to Integrate Baseline DSQ Score in CROSS Dataset

```
*To create DSQ score Grouping variables based on DSQ score from ADQS;
data adqs1(keep=USUBJID AVAL);
      set adam.pre_adqs;
      where ABLFL eq "Y" and PARAMCD eq "DSQSUMM";
run;

data adqs2;
      set adqs1;
      if AVAL  ne . then do;
         if AVAL ge @@@@ then do;
            GROUP2 = "CO1"; GROUP2N = 1;
         end;
         else if AVAL le @@@@ then do;
            GROUP2 = "CO1"; GROUP2N = 2;
         end;
      end;
run;
*Merging all the datasets together to create CROSS dataset;
data pre_cross;
      merge adam.pre_adsl(keep=USUBJID) tssd3 adqs2 … ;
/*<Merge all other required datasets>*/
      by USUBJID;
run;

data adam.cross;
      set pre_cross;
      attrib
  STUDYID   label="Study Identifier"
  USUBJID   label="Unique Subject Identifier"
  GROUP1    label="Group 1 Category"
  GROUP1N   label="Group 1 Category (N)"
  GROUP2    label="Group 2 Category"
  GROUP2N   label="Group 2 Category (N)"
…
;
run;
* @@@@ not disclosed due to data sensitivity reasons;
```

Once the CROSS dataset has been setup as illustrated in Table 2, the information from CROSS can then be fed into the final ADSL and other ADaM datasets. In Table 3 we demonstrate how information is built up stepwise, first in PRE-ADSL, then in CROSS and finally in ADSL.

Table 3: Dataset Contents

PRE_ADSL	CROSS	ADSL
Data Set Name: ADAM.PRE_ADSL Observations 417 Variables 85	Data Set Name: ADAM.CROSS Observations 417 Variables 55	Data Set Name: ADAM.ADSL Observations 417 Variables 140

EXAMPLE 2: PRE-ADSL AND BLINDING

Let us start by visualising ourselves at a Data and Safety Monitoring Committee meeting (also referred to as independent Data Monitoring Committee meeting). This is a group of external experts reviewing the data from a blinded (or sponsor blinded) study, to assess if the study is worth continuing or is safe to continue. One major component of this meeting is the review of safety signals, majorly from Adverse Event data tables.

During the Open/Blinded Session, the data is discussed at an aggregate level, followed by the Closed/Unblinded session where 'by treatment group' data is reviewed. One of the ways to ensure that the data discussed in both sessions is of similar nature, the Total column from the Open Session is cross verified with the Total column in the Closed Session. Also, from an operational and validation perspective, the independent/unblinded programming (IP) team is expected to do minimal writing of code, and code needs to be written and provided by blinded programmer (BP).

In the study in question, the two treatment groups are comprised of Treatment A (fixed dose BID) and Treatment B (Chemotherapy once in 3 weeks, dose based on Body Surface Area (BSA)). Given the two different dosing regimens, the Treatment Emergent Adverse Event (TEAE) definition is varied between the two treatment arms. For "Treatment A", it was the last dose of study drug + 30 days, while for "Treatment B", it was the last date of infusion + 21 days. So, it was critical to the team that the TEAE derivation between blinded & unblinded areas were aligned and balanced. Additionally, the two study arms have different schedules for vital signs, and thus this dataset treated as sensitive and unblinding in nature, and raw VS data was only available in the unblinded area. However, the availability of BSA is essential for calculation of planned and actual dose for the subjects in arm "Treatment B".

The solution to this conundrum was devised by creation of an intermediate PRE-ADSL dataset and incorporating it in the ADaM workflow. The steps used were as follows:

1. PRE-ADSL code was developed by blinded programmer (BP) using dummy data.
2. Validated Code was passed over to independent programmer (IP).
3. IP create PRE-ADSL dataset with real exposure data & derived baseline BSA values.
4. PRE-ADSL dataset with real dates and values passed over to blinded programming area.
5. PRE-ADSL is used in both blinded and unblinded area to build the final ADSL dataset.

Table 4: Actions and Responsibilities

Action	Working Area	Responsible
Create PRE-ADSL code	Blinded	BP
Copy PRE-ADSL to Unblinded area	Unblinded	IP
Create Unblinded PRE-ADSL	Unblinded	IP
Copy PRE-ADSL to Blinded area	Unblinded	IP
Create ADSL	Blinded	BP
Create ADSL	Unblinded	IP

The above example only uses two data points, the treatment dates & BSA. This PRE-ADSL dataset contained more than ten distinct types of variables which were deemed to be necessary for the analysis and consistency across blinded and unblinded area was important for the IDMC.

Thus, creation of a PRE-ADSL dataset, in this example, allows for the creation of a final ADSL which is complete in all respects and allows for the analysis of data to occur seamlessly, irrespective of the blinded status of the data.

CONSIDERATIONS

SPECIFICATION

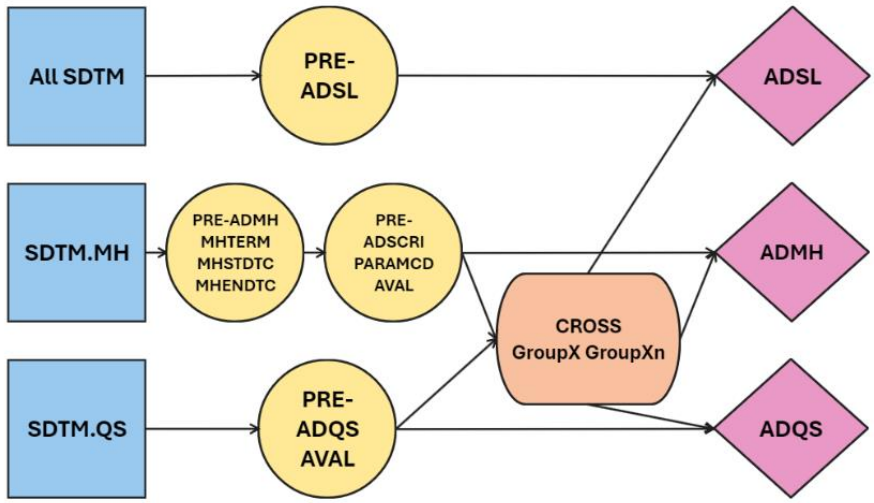
In the ADaM specification, only the final submission ADaM datasets are included. We differentiate each variable's source (SDTM, PRE-ADSL, CROSS) based on colour coding as illustrated in Table 5. The variables which have a PRE-ADaM source (and hence a SDTM source) including those from PRE-ADSL are marked in green, while those coming from CROSS dataset are coloured in yellow.

Table 5: Example of Dataset Specification Highlighting Sources for Variables

Dataset	Variable	Source	Input dataset*
ADSL	USUBJID	Predecessor	SDTM.DM
ADSL	GROUPXX1	Derived	ADAM.CROSS
ADSL	GROUPXX2	Derived	ADAM.CROSS
ADMH	USUBJID	Predecessor	ADaM.ADSL
ADMH	MHTERM	Derived	SDTM.MH
ADMH	MHSTDTC	Derived	SDTM.MH

ADMH	MHENDTC	Derived	SDTM.MH
ADSCRI	USUBJID	Predecessor	ADaM.ADSL
ADSCRI	PARAMCD	Derived	ADaM.PRE_ADMH
ADSCRI	AVAL	Derived	ADaM.PRE_ADMH
ADQS	USUBJID	Predecessor	ADaM.ADSL
ADQS	PARAMCD	Derived	SDTM.QS
ADQS	AVAL	Derived	SDTM.QS

Figure 3: Data Flow Across SDTM, PRE-ADaM, CROSS & Final ADaM



SUBMISSION

The CROSS & PRE- datasets in the examples above are not included in the regulatory submission. The eCRT documentation do not mention these datasets, and a unified dataset specification as illustrated above is used. Caution is advised when finalising the eCRT packages to ensure no references have been made to these intermediate datasets.

ALTERNATIVE SOLUTIONS & CHALLENGES

MANAGING RECURSION

Recursion is phenomenon where a code calls itself on its own output.

In a situation where ADSL is created first, then used in an ADaM, and then updated again with additional information from the other ADaM, the batch run can malfunction, and the audit trail can reflect recursive use. In the example in Figure 4 we see an ADSL dataset which uses ADVS, ADCM & ADQS as inputs and hence would need to contain defensive coding to account for presence/absence of these ADaMs. Also, during subsequent runs of the datasets, additional checks would need to be in place to ensure that the ADSL is using the latest ADaMs rather than the ones generated from a previous run.

Figure 4: Illustration of ADSL Re-Use in ADaM Workflow

MACROS

As per SAS® documentation, the macro facility is a tool for extending and customizing SAS® and for reducing the amount of text that you must enter to do common tasks.

Writing a macro is a commonly used solution to a situation where the same code may need to be executed multiple times. In our use cases, while macros are a helpful solution in some scenarios, in other scenarios they may not be. Macros work very well to centralise the code, and any updates need to be made only in a single place. They can,

however, function as bottle neck as well if the code maintenance of the different domains falls on different individuals.

OTHER DATASET WORKFLOWS

Specifically in the case of ADSL, various workflows have been suggested to optimise the storage and usage of baseline and post baseline subject level information. Some involve creation of ADBASE at the very end of the ADaM workflow, others advise ADCORE at the very beginning. Hongyu Liu and Hang Pang describe some of these workflows in their 2015 paper titled "*Challenges in Developing ADSL with Baseline Data*". These solutions can be used as appropriate to fit the purpose of the analysis.

CONCLUSION

Among many other cases, in two specific instances CDISC has proposed the creation of additional intermediate ADaM datasets to capture complete traceable information. The Therapeutic Area User Guides suggest usage of ADDATES and ADEVENT to ease ADTTE and ADRESP derivation and traceability, respectively. This points towards a level of acceptance of intermediate or precursor datasets, under the "ADaM Other" classification. John Troxell discusses this as well, in his paper from 2015 PharmaSUG titled "*What is the "ADAM OTHER" Class of Datasets, and When Should it be Used?*". On similar lines, we encourage users to explore the possibility of using PRE-ADSL and Other PRE-ADaMs in their programming workflow and discuss possibility of using them in the submission package as needed.

ACKNOWLEDGEMENT

Sarada Gonuguntla, Vishnu Bapu Naidu, Praveen Reddy, Sadanand Deshmukh, Valentine Jehl, Marine Weil, and Prabhakar Munkampalli, Katarzyna Pindor, Linda Simonsson, Christina Johansson and Marta Tomczyk for introducing us to PRE-ADSL and helping develop the workflow.

Sarah Berenbrinck, Timothy Lachlan-Cope, Jinesh Patel, Asif Karbhari, Katarzyna Pindor, Beata Pawlikowska, Iain Timmins, Piotr Karasiewicz and Daniel Graham for reviewing the abstract, paper & slides and for constant encouragement and support.

GLOSSARY

ADaM: CDISC Analysis Data Model

ADaMIG: CDISC Analysis Data Model Implementation Guide

ADBASE: Baseline Variable Analysis Dataset

ADCORE: Core Variable Analysis Dataset

ADDATES: Event Dates Analysis Dataset

ADEVENT: Event Analysis Dataset

ADMH: Medical History Analysis Dataset

ADQS: Questionnaire Analysis Dataset

ADSCRI: Respiratory Subject Characteristics Analysis data

ADSL: Subject-Level Analysis Dataset

ADTTAE: Adverse Event Time-to-Event Analysis Dataset

ADTTE: Time-to-Event Analysis Dataset

BDS: CDISC ADaM Basic Data Structure

BID: Twice-a-Day dosing regimen

BP: Blinded Programmer

BSA: Body Surface Area

CDISC: Clinical Data Interchange Standards Consortium

DSQ: Dysphagia Symptom Questionnaire

eCRT: Electronic Case Report Tabulations

EoE: Eosinophilic Esophagitis

IDMC: Independent Data Monitoring Committee

IP: Independent Programmer

SDTM: Study Data Tabulation Model

TAUG: Therapeutic Area User Guide

TEAE: Treatment Emergent Adverse Event

REFERENCES

1. Getting Started with the Macro Facility. SAS® 9.4 and SAS® Viya® 3.5 Programming Documentation.
2. Hongyu Liu, Hang Pang (2015). Challenges in Developing ADSL with Baseline Data. PharmaSUG 2015 - Paper PO22
3. Jeppe Rømer Juul, Niels-Kristian Kjøller, Ari Siggaard Knoph (2020). Flow Dependencies in ADaM Generation. PHUSE EU Connect 2020 – Paper DH11.
4. John Troxell (2015). What is the “ADAM OTHER” Class of Datasets, and When Should it be Used? PharmaSUG 2015 - Paper DS16.
5. Nancy Brucken, Deb Goodfellow, Brian Harris, Terek Peterson, Alyssa Wittle, (2019). Incremental Changes: ADaMIG v1.2 Update. PharmaSUG 2019 - Paper DS082.
6. Pamela Shaw (2022). Data and Safety Monitoring Committees: Why, What, Who, and How? NIH VideoCast, YouTube.
7. R. M. Pokrzywinski, B. Goodwin, E.S. Dellon, et al (2023). Qualitative assessment of the suitability of the Dysphagia Symptom Questionnaire to monitor dysphagia in children aged 7–10 years with eosinophilic esophagitis. Journal of Patient-Reported Outcomes.
8. Sandra Minjoe (2015). It Depends On Your Analysis Need. PharmaSUG 2015 - Paper DS11.
9. Stacie Hudgens, Christopher Evans, Elaine Phillips, Malcolm Hill (2017). Psychometric validation of the Dysphagia Symptom Questionnaire in patients with eosinophilic esophagitis treated with budesonide oral suspension. Journal of Patient-Reported Outcomes. 1. 10.1186/s41687-017-0006-5.
10. Therapeutic Area Data Standards User Guide for Breast Cancer Version 1.0 (Provisional) (2016-05-16).
11. Therapeutic Area User Guide for Prostate Cancer Version 1.0 (Provisional) (2017-07-10).

CONTACT INFORMATION, AFFILIATIONS, FUNDING, DISCLAIMER

Your comments and questions are valued and encouraged.

Author: Korak Datta, Biometrics, Late-stage Development, Late Oncology, Oncology R&D, AstraZeneca, Cambridge, UK, Email: korak.datta@astrazeneca.com.

Author: Taniya Chatterjee, Biometrics, Late-stage Development, Respiratory and Immunology (R&I), BioPharmaceuticals R&D, AstraZeneca, Warsaw, Poland, Email: taniya.chatterjee@astrazeneca.com.

The authors are AstraZeneca employees and may hold stocks or stock options in AstraZeneca entities. This paper was authored using funding from AstraZeneca. Brand and product names are trademarks of their respective companies.