WUSS 2022 – Session # 101

Cautionary Notes when Working with Interim Data

William Coar, Cytel

ABSTRACT

Constructing a hypothesis, conducting an experiment, collecting and analyzing data, and sharing results are fundamental components of the scientific method. In many experiments (such as clinical trials), collecting data may take years. Furthermore, some analyses may be performed along the way with interim data which is incomplete and often erroneous. While there are statistical concerns with analyses and interpretation of interim data, we do not attempt to discuss them here. Rather, we focus on working with interim data as it relates to the conduct of the experiment. Interim data review can be critical to ensuring the experiment goes as expected, mitigating unforeseen circumstances if needed. This helps maintain the integrity of the original experiment.

When working with interim/incomplete/sometimes erroneous data, programmers and statisticians should be cautious as definitions of expected data values, expected data structures, and derivations needed in analyses may differ from what may be found at the end of the experiment. Furthermore, programming with interim data should include additional defensive programming efforts as it is nearly impossible to predict what may be seen in future data.

In the pharmaceutical industry, interim versions of data may be used to support activities such as manual data review, data cleaning, annual safety updates required by regulatory agencies, and support of Data Monitoring Committees(DMC). Manual data review and data cleaning result in high quality data. Safety updates and support of DMCs are critical to ensure patient safety. Such analyses cannot wait until the end of the experiment, yet the expected data values, expected data structures, and derivations at interim timepoints during study conduct may differ than those used in the final analysis. The focus of this presentation is to emphasize the cautionary use of early and interim data, and lessons learned from many years of supporting DMCs.

INTRODUCTION

Consider an experiment to determine of a new drug in combination with current standard of care prolongs life in patients with cancer. They primary question, or hypothesis, may be that patients with this cancer live longer with the addition of add-on therapy. Ideally, we can conduct an experiment that means following several hundred patients who receive 1 of two treatments: standard of care (SoC), or standard of care + add-on treatment. Patients would be followed from the start of treatment until they die, but this often is not feasible. As an alternative, it may be sufficient to demonstrate that the add-on drug either slowed the progression of cancer, or even eliminated it. They primary question is to determine if progression of cancer (ie, *disease progression*) slows with the addition of add-on therapy. This can be done by comparing the hazard rate of SoC+add-on treatment (T) to that of SoC (C) using a hazard ratio.

$$H_o: \frac{\lambda_T}{\lambda_C} \geq 1 \; vs \; H_a: \frac{\lambda_T}{\lambda_C} < 1$$

Note that if the add-on therapy is effective, the hazard of experiencing disease progression with SoC+Add-on should be less than that of SoC. A hazard ratio HR<1 suggests effective treatment.

The endpoint, or quantitative measure captured to answer the primary question, is a measure of time-to-event. Once sufficient patients are enrolled, they are followed until enough patients experience disease progression, which includes death. Patients may come in at fixed timepoints to receive treatment and medical care. A lot of da

ta is captured a long the way in an **assumed (or expected)** format.

Current technology allows data to be captured directly into an electronic data capture system (EDC). While these systems can be designed to ensure **expected data values** (what we call clean data), many data captured need to be reviewed manually or algorithmically checked to ensure it makes logical sense and is consistent with expected values. If not, a query is generated so the data point is questioned. Data integrity is critical to the final interpretation of the results.

Since the study will be ongoing for years, there may be updates to the EDC which potentially change the structure of the output dataset. Additionally, certain tables in the EDC may not be fully populated, or populated at all early in the study. This presents challenges to programming when either the **expected structure** of the resulting SAS dataset changes, or if information on the **expected structure** of future data needs to be predicted. Furthermore, any transformations after the EDC may impact **expected data structure**.

While the experiment is designed to determine if the new add-on therapy is efficacious in slowing cancer progression, it is extremely important to ensure it can be done in a relatively safe manner. Toxicity is assessed objectively by capturing adverse reactions that patients experience while receiving the new treatment. They can also be assessed through blood chemistry, vital signs, and electrocardiographs. Collectively, these data are referred to as *safety data*.

As patients come in and out of the experiment, their data need to be reviewed and queried for potential safety concerns. This happens throughout the entire duration of the experiment, not at the end. Interim data reviews are essential to ensure safety of current and future patients.

Some toxicity is expected. Too much toxicity may warrant halting the experiment. How much is too toxic? That depends on how much benefit a patient may gain from receiving the new add-on therapy. When assessing risk/benefit, **programming rules used at interim reviews may differ** from those at the end of the study.

Interim reviews of data are often required to allow independent oversite of risk/benefit. This independent review ensures that trial conduct is not changed (intentionally or unintentionally) as a result of reviewing observed data, thereby maintaining trial integrity.

While the trial may take months or years to conduct, there are many programming activities along the way, on interim data.

EXPECTED DATA VALUES

Data are captured directly into an EDC by staff where patients receive medical treatment using patient records, or *source documents*. Data are entered by humans, and humans make mistakes. Programmers are often tasked with running data through pre-defined algorithms to ensure data make sense. These are sometimes called *edit checks*.

During the development of the EDC, an electronic case report form is used to define exactly what data are captured, what the possible values may be (for categorical data) and ranges of values for continuous data. Other data captured are free text fields, meaning that anything can be entered. This is referred to as metadata.

Collection of unsolicited adverse reactions is through free text fields. For this to be analyzed, the values need to be mapped to defined terminology generally agreed upon by the medical community using medical dictionaries. This mapping may be done automatically, or manually. In either case, medical experts review the mapped result to verify it make sense given the text captured in the EDC.

Additional information may be controlled data entry, such as the severity of the adverse reaction(mild/moderate/severe), seriousness (yes/no), and outcome (resolved/ongoing). It is controlled in the sense that only pre-specified values may be entered. Onset and resolution dates are also captured, though sometimes dates may be incomplete.

Metadata can be used to understand the expected data values that can be entered (and exported) from the EDC. Furthermore, metadata provide the mapping between character and numeric code representation of categorical data. However, this metadata is not always available to external vendors or the statistical data analysis center (SDAC) supporting an independent monitoring committee.

Early in the study, there simply isn’t much data. If the SDAC is trying to summarize the reasons for patients discontinuing study treatment, there is limited observed data available to help the programmer understand expected values (numeric and character representation) of categorical data. Below is example metadata for reasons for treatment discontinuation:

Numeric Value	Character Value
1	Adverse Event
2	Clinical Progression
3	Radiologic Progression
4	Patient Withdrew Consent
5	Study Terminates
6	Physician Decision
7	Other

In the absence of metadata, character values may be obtained from the case report form. However, the numeric equivalent may not be available.

Other data captured may be unpredictable. For example, visits have a numeric and character label that describe the sequence or point in time when the patient visited the doctor. For example, the visit associated with the start of treatment may have a visit label ‘Cycle 1 Day 1’ with a corresponding numerical representation of 1. As the study progresses, more data are available at later visits. It is often quite challenging to predict what future visit labels and numbers will be. This is especially true when patients have unscheduled visits, or enter a follow-up phase where intervals of time between visits changes, and data collected at each visit changes. Some example metadata for patient visit schedule is below:

Numeric Value	Character Value
-1	Screening
1	Cycle 1 Day 1
2	Cycle 1 Day 15
3	Cycle 2 Day 1
4	Cycle 2 Day 15
…	…
20	End of Treatment
21	Safety Follow-up

As data accumulate, the mapping between numeric and character representations are eventually reviewed. But programming done early usually requires updating along the way to accommodate new values observed.

EXPECTED DATA STRUCTURE

Industry standards for data structure are now mandated for submission to regulatory agencies for drug approval. We refer to these as CDISC standards. Data from an EDC usually require transformation to CDISC standards. Having an expected data structure including variable names, types, etc can greatly assist with programming and data review. However, many of the rows and columns are derived, and when it is derived on interim data, their meaning may change, or yield unexpected values.

Data collected Multiple Places

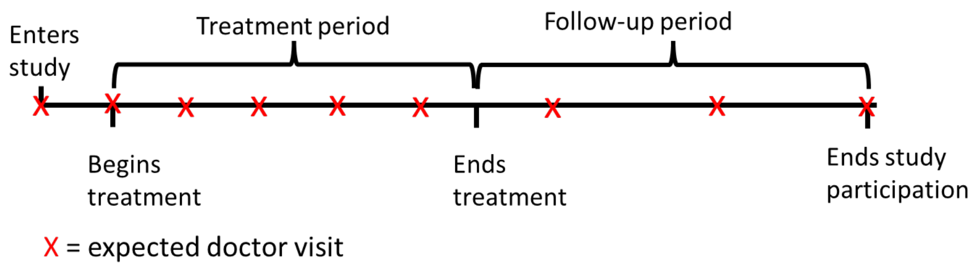
Let's consider an example where data for the same information is collected in different places, and resides in different datasets. An example relevant to assessing safety is a binary flag to identify if a patient dies. This can be found in any of the following:

- If a patient dies, they are expected to have a record in a disposition dataset (DS) with a text field indicating a death event has occurred.
- There is a patient demographics (DM) dataset that is one record per patient which has a flag to indicate if the patient died.
- We know that adverse events (AE) can have an outcome of death.
- If there is a follow-up period, there may be a CRF page specifically to capture death information.

At the conclusion of a study, when all data are collected, queried, and cleaned, we expect consistency across these different datasets. We can expect to find a single record in DS to identify if a patient dies. However, this is often not the case with interim data.

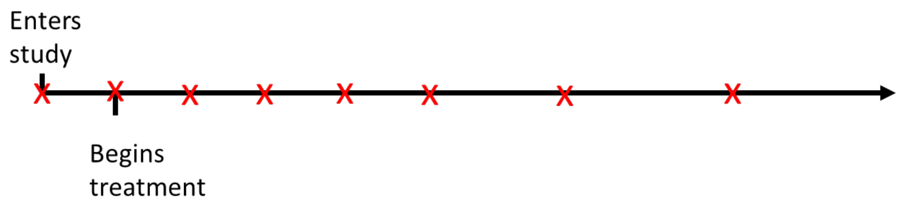
Change in Variable Meaning

An example of change in variable meaning is with reference dates. At the conclusion of a study, it is possible have reference start/stop dates associated with a patient's study participation, and reference start/stop dates associated with study treatment.

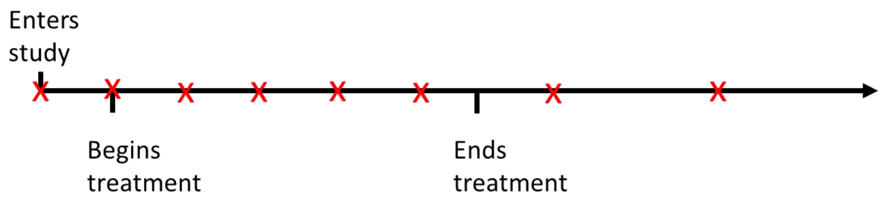


Consider reference start/stop dates associated with treatment administration, which are found in the demographics data DM where there is a single record for each patient. These dates are also found in other drug administration type data, but usually in a different format where multiple records exist.

In the interim setting, a patient may still be ongoing treatment, so there is no treatment end date.



Alternatively, a patient may have ended treatment but is continuing to be evaluated in a follow-up period.



While a patient is ongoing in the study, certain reference start/stop dates may not actually have been observed. However, these are often approximated (intentionally or unintentionally) by some other data which can actually change their meaning. In the two scenarios above, a treatment end date may be approximated by the last known date a drug was taken. A study end date may be approximated by the date of the last known visit.

Exposure data usually has its own case report form that captures relevant information such as what dose (if any) was received, were there any dose interruptions, modifications, etc. This data can be scanned to determine the earliest date in which a dose was received (reference start date) and the latest date in which a dose was received (reference stop date). With interim data, we should recognize that a programming algorithm will identify the last date in which a dose was received, but it is not actually a stop date*. It is simply the last date in which we know data were entered*. The patient may actually be continuing treatment. Programmers should be mindful of this in subsequent algorithms that may use the reference end date.

Experiments such as those that allow for patients to cross-over to a specific treatment can be for more complicated.

Given uncertainty when reference stop dates have not technically occurred, derivations for analysis may need to be modified in the interim setting.

Data Structures Change

Even a seasoned programmer can have a hard time predicting when data structures will change. The impact is very much dependent on the change itself. Changes may occur in the original datasets and/or the subsequent versions in CDISC format. The biggest defense here is communication. Some changes will have little to no impact, others may require re-programming from scratch. In an interim setting we feel it is best practice to do some sort of comparisons of data structure between each interim look.

DERIVATIONS

Binary Flag Example

Consider the earlier case where the desire is to identify if a patient dies on study. At the conclusion of the study, it is straightforward:

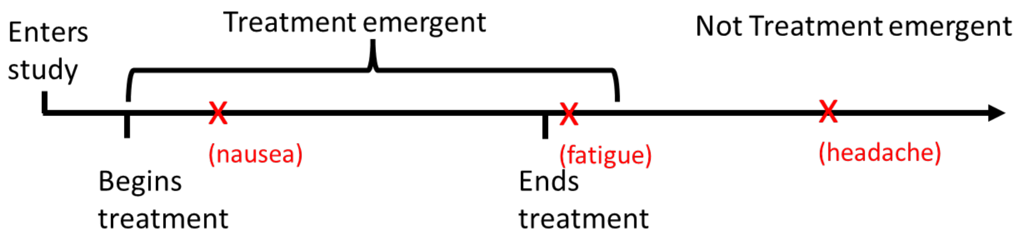
```
Proc sort data=DS out=ptdeath(keep=usubjid);
     by usubjid;
     where DSDECOD='DEATH';
run;
```

In the interim setting, the programmer may need to scan multiple datasets to identify patients that died.

```
Proc sort data=DS out=dsdeath(keep=usubjid);
     by usubjid;
     where DSDECOD='DEATH';
run;
Proc sort data=AE nodupkeys out=aedeath(keep=usubjid);
     by usubjid;
     where aetoxgrn=5;
run;
data ptdeath;
     merge DM(in=a where=(DTHFL='Y')) DSDEATH(in=b) AEDEATH(in=c);
     by usubjid;
     if a or b or c;
run;
```

Interval Related Example

Consider the use of reference end date for treatment exposure. At the end of the patient's participation in the study, this truly be the end date of treatment. This is important for determining a *treatment emergent* flag for adverse reactions. When analyzing patient safety, only adverse events that are treatment emergent are typically included. Rules involve comparing the onset of the reaction with a date range associated with treatment exposure. However, it is not simply the start and stop of treatment. A lag such as 14 or 30 days after last dose is used, though others may be clinically indicated.



In the above example, nausea and fatigue would both be considered as treatment emergent for analysis. Headache, however, started long after treatment ended. It is unlikely that headache resulted from study treatment. It may or may not be even captured in the database.

Given interim data, the programming logic may need to be modified. If a patient is ongoing treatment (ie, had not discontinued), then the following rule is more appropriate:

```
if (trtsdt ≤ astdt ) and dctrt='N' then trtemfl = Y;

else if (trtsdt ≤ astdt  ≤ (trtedt + 14)) then trtemfl = Y;
```

At the end of the study, programming can be simplified to:

```
if (trtsdt ≤ astdt  ≤ trtedt + 14) then trtemfl = Y;
```

Without the modification with interim data, an algorithm is likely to exclude many adverse events from analysis simply because of a lag in data entry associated with exposure data. This could have enormous consequences as it may imply safety is acceptable for the new treatment, yet safety was never fully evaluated because of faulty programming logic.

Interval Related Example

For DMC to assess potential benefit they request that time to disease progression be summarized. This may be the same analysis as the primary analysis at the conclusion of experiment to make formal decisions. However, in the interim setting, the intent is to allow assessment of potential benefit, not to make formal inference.

To analyze this time-to-event data, we know two things:

- Indicator (0/1): whether or not a patient has disease progression or is *censored* (ie, followed but no progression has occurred). Recall the disease progression is defined to be cases where cancer worsened or the patient dies, whichever comes first.
- Measurement of time: How many months have they been followed until disease progressed? If disease has not progressed, how many months has the patient been followed?

For a final analysis, there may be censoring rules associated with missed tumor assessments and/or subsequent anti-cancer therapy. If there is a long lag that suggests a missed tumor scan, an event may become censored. If a subject is known to receive subsequent anti-cancer therapy without documented disease progression, an event may be censored. For interim data and safety reviews, neither of these may be appropriate.

With interim data, lags in the data may simply be a result if delays in data entry. These cases may trigger other questions, such as questions of whey there is a lag in data entry which is related to study conduct and/or data integrity. Most importantly, this censoring tends to occur if the patient dies. This can mask potential safety concerns.

Similarly, if summarizing overall survival (ie, time to all-cause mortality), censoring rules may be different in the interim setting. For a final analysis, a rule may be to scan all datasets and identify the latest date found for each patient that did not die. This can still be done with interim data, but the censoring distribution because we know that follow-up time is likely more than what is observed in the data. An alternative may be simply to use the data cutoff, which assumes that if a patient has died, we would likely know about it. Both censoring rules can be reasonable, and provide a DMC with reasonable information. One however, is much more simplified than the other.

Denominators

At the conclusion of a study, data at collected at each planned study visit are often summarize by planned study visit. For example, the number of subjects with abnormal vital signs is summarized by visit, which allows a reader to understand when abnormalities occur, and if they resolve. The count and percent of patients with such abnormalities are summarized by visit. At the conclusion of a study, all patients will have had the opportunity to complete all visits. It may be reasonable to simply use the total numbers of patients for the denominator to estimate the rate of abnormalities. However, in the interim setting, not all patients have had the opportunity to reach all visits. It is misleading to calculate a percentage based on a large denominator (ie, total number of patients) at later visits when few patients may have actually reached that point in the study. In the interim setting, rules for defining a denominator may be necessary.

DEFENSIVE PROGRAMMING

Predicting what future data may be observed is hard. Even with information such as metadata, it can be challenging to develop code early in the experiment that accommodates future observations. Some might say this is impossible. Regardless, there are ways to mitigate with defensive programming.

Early in a study, it is common for programmers to write code that works for a given set of data. Afterall, that is what we expect. But this code often fails as data accumulate which can sometimes go unnoticed. This is primary associated with categorical type data. One of the simplest defensive programming technique the author has found is to put messages to the log when either (1) something unexpected is found with IF statements or (2) something you haven't seen before is observed. These can easily be identified when LOG files are reviewed.

Proactive programming to print data can similarly be used to identify potential issues.

While these two approaches may seem simplistic, they work. They are also a good starting point. With experience, programmers better understand where defensive programming may be more applicable to individual situations.

CONCLUSION

Interim versions of data may be used to support activities such as manual data review, data cleaning, annual safety updates required by regulatory agencies, and support of a DMC. Programming is critical to ensure the experiment goes as expected, mitigating unforeseen circumstances if needed.

While the example provided is associated with testing a new treatment in a clinical trial, it is not limited to this setting. Even with an understanding of expected data values and data structure, updates to early programming are inevitable. Changes to algorithms and/or interpretation may also be necessary. Higher quality can be achieved when our teams approach projects with this in mind.

ACKNOWLEDGMENTS

The author would like to thank David Kerr for his review and insights from the perspective of a statistician supporting DMCs for over 20 years.

CONTACT INFORMATION

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

William Coar
Axio Research, A Cytel Company
William.coar@cytel.com