

An Introduction to Visit Window Challenges and Solutions

Mai Ngo, SynteractHCR

ABSTRACT

In clinical trial studies, statistical programmers often face the challenge of subjects' visits not occurring on the exact scheduled visit dates. As a result, visit windowing is often needed for analysis purposes. This paper aims to provide a general introduction to visit window programming. I present some different scenarios where visit windowing is needed and sample SAS[®] codes for each of these cases. I also discuss strategies for applying the visit window rules to multiple programs with different visit frequencies and window lengths and cover validation strategies in these cases.

INTRODUCTION

A large amount of clinical trial data to evaluate the safety and efficacy of new drugs is by-visit assessments that were designed to follow a schedule of assessment specified in the trial protocol. The typical schedule of assessment lists target visit days and associated tests to be conducted on these visits; some also specify associated visit windows for these visits. However, it is often difficult to schedule visits and keep patients on schedule. Mistakes may also be made at clinics when calculating the proper day to set the next appointment or when entering visit labels into the database. Because of this, statistical programmers of clinical data often work with data where the visit labels in the raw data do not match with the intended visit numbering in the schedule of assessment.

While presenting summaries of data by visit, one can choose to summarize either by the visit labels in the raw data or by analysis visits. Analysis visits are selected from the application of an algorithm to identify visits that are closest to the intended visit in the schedule of assessment. This practice is called 'visit windowing'. It has the advantage of allowing for more comparable comparison of test results temporally and correcting for mistakes that may have occurred when visit label information is entered at the sites. Another scenario where visit windows can be used is to identify 'missed visits', or visits that are out-of-window, and determine how severe these issues are. This information could be useful in reviewing protocol deviations.

The visit window rule ultimately depends on the specific study and the analysis goal. In this paper, I will present an introduction of the most common rules that I have encountered in my work and some example SAS codes for programming visit window corresponding to these rules. While these are simple codes, my goal is to highlight some common threads in the visit window algorithms often used in analyzing clinical trial data.

VISIT WINDOW RULE

OVERVIEW OF A TYPICAL CLINICAL TRIAL VISIT WINDOW ALGORITHM

Below is a snapshot example of the Schedule of Assessment in a clinical protocol. A visit's target study day is based on the Schedule of Assessment in the clinical protocol. In other words, the day in this schedule is the Study Day (days since the treatment first started), not a calendar day. Visit window range used in programming could follow what is specified in the protocol or follow different ranges as defined in the Statistical Analysis Plan (SAP). The visit-window ranges cannot overlap but they may either cover the entire study period or there may be gaps between them. Follow-up visits are typically not included in visit windows, or assigned to 'Follow-up' visit regardless of the nominal visits. Baseline visits or the visits when the treatment first starts (Day 0 or Day 1) can either be windowed or not.

A visit window algorithm consists of the following 2 components:

1. Specification of a visit window range (earliest and latest study day) for each visit in the period that the visits need to be 'windowed';
2. Specification of a rule that enables the selection of a unique visit for all visits that fall within a visit window range. This selected visit is the 'analysis visit' that will be used in by-visit summaries.

IMPLEMENTATION OF THE VISIT WINDOW RULE – SAS PROGRAMMING EXAMPLES IN THE CLINICAL TRIAL CONTEXT

Writing the programming specifications

The visit window rule specified in the Statistical Analysis Plan (SAP) is translated into the programming specifications for Analysis Data Model, hereafter called ADaM specs. I have seen visit window rules specified both as a separate tab in the ADaM specs Excel spreadsheet, or as a part of the specs for each corresponding ADaM dataset.

Table 1 shows an example of the ADaM specs for visit window programming presented in a separate tab in the ADaM specs spreadsheet. ADY, study day, is the day from the date where the treatment first started (TRTSDT) for a subject and is calculated as $ADY = \text{visit date (ADT)} - TRTSDT + (ADT \geq TRTSDT)$ if the treatment start date is counted as Day 1, or $ADY = ADT - TRTSDT$ if the treatment start date is counted as Day 0. AWTARGET is the target study day for a visit, often specified in the Schedule of Assessment in the protocol. AWLO and AWHI are lower limit and upper limit (study day) of the visit window, respectively. AWTDIFF is the difference between the visit's actual study day and the target study day for that visit, AWTDIFF could be actual or absolute value of this difference. AVISIT is the label of the selected analysis visit after applying the visit window range and tie-breaker rule and AVISITN is its corresponding numeric variable.

Variable	Label	Define Comments
USUBJID	Unique Subject ID	
TRTSDT	Treatment Start Date	Date of first dose of study drug
VISITNUM	Visit (Numeric)	Visit (from Raw Data)
VISIT	Visit	Visit (from Raw Data). One-to-one map to VISITNUM
AVISITN	Analysis Visit (Numeric)	The numeric code for AVISIT. One-to-one map to AVISIT.
AVISIT	Analysis Visit	Derived using visit windowing per SAP Appendix A
ADT	Analysis Date	Visit date
ADY	Analysis Relative Day (Study Day)	If $ADT \geq TRTSDT$ then $(ADT - TRTSDT) + 1$; else $(ADT - TRTSDT)$.
AWTARGET	Analysis Window Target	The targeted day for the analysis visit (per SAP Appendix A and B).
AWTDIFF	Analysis Window Diff from Target	Absolute value of $ADY - AWTARGET$.
AWLO	Analysis Window Beginning Time Point	The beginning time point (day) for the analysis visit. Visit windows are defined in SAP Appendix A.
AWHI	Analysis Window Ending Time Point	The ending time point (day) for the analysis visit. Visit windows are defined in SAP Appendix A.
AWU	Analysis Window Unit	For all records that have a non-missing AWTDIFF, set to 'Days'.

Table 1. Programming specifications for Analysis Data Model (ADaM specs).

Programming visit window in SAS

A. The Basics

In this section, I give an example of SAS codes to derive the analysis visits (AVISITN) following a visit window rule. Here is the example dataset I will use to illustrate the visit window programming steps:

```
data allrec;
  infile datalines dlm=' ';
  length visit $12;
  input usubjid visit $ visitnum ady vistyp aval;
  datalines;
```

```

1, Month3, 3, 100, 1, 78
1, Unscheduled, 99, 100, 4, 76
1, Month6, 6, 227, 1, 60
1, Unscheduled, 99, 182, 4, 90
1, Unscheduled, 99, 178, 4, 120
1, Unscheduled, 99, 250, 4, 75
1, Unscheduled, 102, 250, 4, 80
;
run;

```

The first step is to identify all visits that fall in the visit window. This step can be broken down into 2 sub-steps:

(1a) Set up visit window ranges:

The visit window ranges for the planned 3-month interval visits are:

Visit	Target Study Day (AWTARGET)	Analysis Window Study Day	
		Low (AWLO)	High (AWHI)
Month 3	90	2	135
Month 6	180	136	225
Month 9	270	226	315

Table 2. Visit window ranges.

In the code below, I set up a shell visit window data set based on the ranges specified above:

```

data awindow;
  input avisitn awtarget awlo awhi;
  datalines;
3 90 2 135
6 180 136 225
9 270 226 315
;
run;

```

(1b) Choose all visits where study day (ADY) is between AWLO and AWHI; hence after, these visits are called 'windowed' visits:

```

proc sql;
  create table windowed as
  select a.*, b.*
  from allrec a left join awindow b
  on awlo <= ady <= awhi;
quit;

```

The second step is to select the analysis visit (one per visit window range) following the tie-breaker rule in the SAP. Below are some of the common rules that I have encountered in my work.

(2a) Tie-Breaker Rule – Example 1 - Selecting the closest visit to the planned target visit study day

To do this, we first need to calculate the distance (number of days), AWTDIFF, between a visit and a target study day:

```

data windowed;
  set windowed;
  if ady ne . and awtarget ne . then AWTDIFF=abs(ady-awtarget);

```

```
run;
```

Depending on idiosyncrasies in the data, such as whether there are multiple visits on the same date, etc., the tie-breaker rule for selecting the closest visit can be simple or more complex. For some data, (A1) and (A2) would be sufficient for selecting unique analysis visits for each target visit. For other data, additional tie-breaker rules need to be specified and programmed for, such as (B1) and (B2). In the example code below, USUBJID is a patient's unique ID number.

A1. Select the visit with closest to the target study day, i.e. the one with the smallest AWTDIFF

A2. If 2 assessments are the same distance from the target day for a particular analysis window, the later assessment is chosen:

```
proc sort data = windowed;
  by usubjid avisitn descending awtdiff ady;
run;
```

```
data allrec2;
  set windowed;
  by usubjid avisitn descending awtdiff ady;
  if last.avisitn then anl01fl='Y';
run;
```

B. If there are two or more assessments taken on the same date or same date and time, the rules for choosing the assessment are:

(B1) First take the assessment from the scheduled visit, then take the assessment from the re-test visit (applicable to laboratory tests), then take the assessment from the discontinuation visit, then take the assessment from the unscheduled visit.

(B2) Take the assessment associated with the smallest (raw) visit number.

In the example code below, variable VISTYP has been programmed to take the value of 1 to 4 corresponding to assessment from the scheduled visit, from the re-test visit, from the discontinuation visit, and from an unscheduled visit, respectively:

```
proc sort data = windowed;
  by usubjid avisitn descending awtdiff ady descending vistyp
  descending visitnum;
run;
```

```
data final;
  set windowed;
  by usubjid avisitn descending awtdiff ady descending vistyp
  descending visitnum;
  if last.avisitn then anl01fl='Y';
run;
```

(2b) Tie Breaker Rule – Example 2 - Selecting the visit with the worst test result (most abnormal value)

In the example code below, variable AVALAB has been programmed such that abnormal value is in increasing order of AVALAB:

```
proc sort data=windowed;
  by usubjid avisitn avalab;
run;
```

```
data final;
  set windowed;
  by usubjid avisitn avalab;
```

```

    if last.avisitn then anl01fl='Y';
run;

```

The third step is to set a value for AVISIT and AVISITN for records that are not 'windowed'. For example, a visit window rule can specify to display raw visit labels for pre-treatment and follow-up visits; otherwise, display 'Not Windowed' for visits where the visit dates are missing or partial.

Finally, for records that are considered for visit windowing, we tidy up the visit window variables (AVISIT, AVISITN, ANL01FL (optional), AWTARGET, AWTDIFF, AWLO, AWHI). There are two approaches to do this. In the first approach, we display values of these variables for all visits that fall in a visit window. An analysis flag, e.g. ANL01FL, is created that takes the value of 'Y' for only the selected analysis visit. For other visits, the value of this flag is set to missing. We can then create by-visit summary in tables by selecting only records where ANL01FL equals 'Y'. In the second approach, all the visit-window variables above are set to missing for a visit record that is not the selected analysis visit even if this visit falls in the visit window range. As a result, by-visit summaries can be created by selecting records with non-missing values of AVISIT. To follow this approach, the data set FINAL can be merged back to the raw data set. Then for visits that are not selected by the tie-breaker rule, all the visit window variables can be set to missing.

B. Additional Levels of Complexity

B1. What about the last on-study visit?

There are two approaches for programming visit window for the last on-study visit or end-of-study visit. The first approach is simply to use the end-of-study visit as labeled in the raw data. The second approach is to either specify an upper limit study day for the window of the last visit or leave that range to be infinite (i.e. AWHI is blank). The third approach requires applying a check to ensure that the analysis visit is selected among only visits that did not occur after the last day the subject is on the study. In essence, this means that AWHI for the last visit is the last day the subject is on the study. Below is one example of this rule:

- (a) Get the last value of SVSTDTC (start date of an activity in a visit) for each subject, compare it with the last dose date and take the later of these 2 dates.
- (b) Ensure that the dates for all visits are prior to/on the date from Part (a).

B2. Visit window rules vary across different protocol versions

While this requires more lengthy set up of the visit window range depending on the protocol version the subject is enrolled under, this could be done relatively easily in terms of programming once the visit window rule is specified

B3. Distinguishing 'Out-of-Window' visits from 'Missed' visits

Here is an example of the rule for programming protocol deviation categories for MRI testing that I encountered in a study: if an assessment was performed at an assigned visit but during an out of a defined window period, label it as an 'out of window assessment'. Otherwise, if there is no visit that falls in the defined visit window for Visit X, Visit X is flagged as 'missed visit'. Each assessment should be counted once. While this requires more complicated coding, the principle for programming visit windows is still the same.

C. Writing visit window codes more efficiently – user-defined macros for visit window programming

A user-defined SAS macro would be useful when the visit windowing needs to be applied for many datasets; especially when the visit window ranges, i.e. the difference between AWLO and AWHI for each planned visit, and the selection rule for analysis visits are similar across different analysis datasets. Below is an example of a user-defined macro that I wrote. In the case where the visit window ranges are very similar across datasets, the macro can also be tweaked to include the steps of generating the visit

window shell (the AWINDOW dataset) instead of having this dataset generated outside the macro in each individual analysis dataset program. However, if the visit window ranges vary widely across different datasets, setting up the visit window shell independently in each program allows for the needed flexibility. For validation purpose, the validation programmer can independently write a validation visit window macro to be called in the validation programs:

```
%macro wind(indat=,windat=,param=,crit=,out=,anlfl=);
proc sql;
  create table temp1 as
  select a.*, b.*
  from &indat a left join &windat b
  on awlo <= ady <= awhi;
quit;
data temp1;
  set temp1;
  awtdiff=abs(ady-awtarget);
run;
proc sort data=temp1; by usubjid avisitn &crit ; run;
data &out;
  set temp1;
  by &param usubjid avisitn &crit ;
  %if &anlfl ^= %then %do;
    if last.avisitn then &anlfl='Y';
  %end;
  %else %do;
    if last.avisitn;
  %end;
run;
%mend;
```

For example, this macro call replicates the steps illustrated in (B2):

```
%wind(indat=allrec, windat=awindow, param=, crit= descending awtdiff ady
descending vistyp descending visitnum, out=final, anlfl=Y);
```

CONCLUSION

Programming visit windows is a common task for analyses of clinical trial data. While the specific visit window rule varies across studies, I describe in this paper common building blocks for programming visit windows. At the same time, note that many levels of complexity could be added to these building blocks. Also, programmers should examine the data to ensure that (1) the visit window algorithm makes sense, (2) the selection of the analysis visit is unique for the dataset, and (3) the program is dynamic so that it still works as more data come in.

CONTACT INFORMATION

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

Mai Ngo
 SynteractHCR
 430 Davis Dr, Morrisville, NC
 Work Phone:
 E-mail: maianh.ngo@gmail.com
 LinkedIn: <https://www.linkedin.com/in/maiango/>