**Paper PH08-2011**

# Using a Double DOW Loop to Compute Progression Free Survival

Matthew Nizol, United BioSource Corporation, Ann Arbor, MI

## Abstract

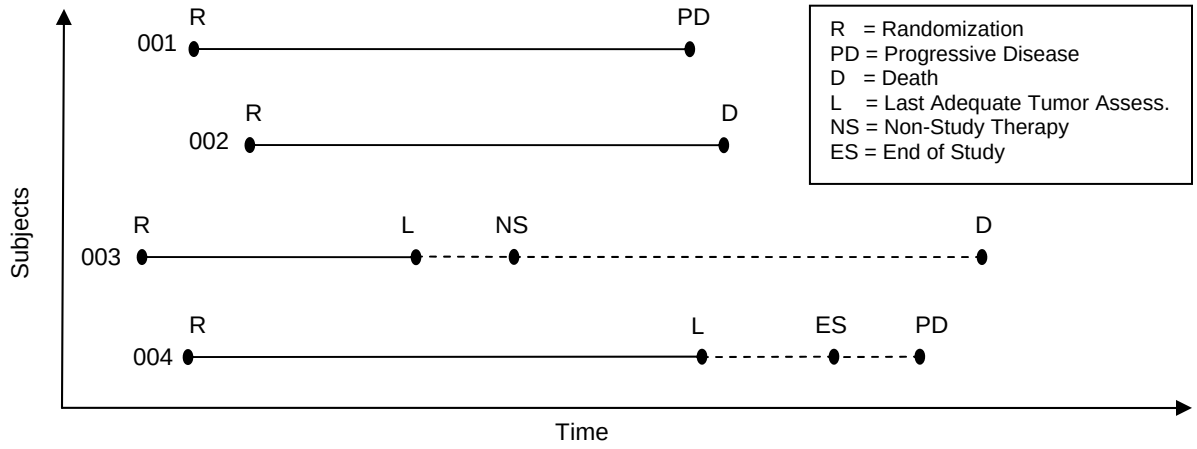
The DOW loop is a powerful construct in SAS. By combining two DOW loops in a single DATA step, a programmer may compute summary information across a BY-group and then merge that information back onto the detail records within the same DATA step. This paper will describe an application of the double DOW loop in the pharmaceutical industry. In oncology clinical trials, progression free survival (PFS), defined by the FDA as "the time from randomization until objective tumor progression or death,"[3] is sometimes an endpoint used to assess the efficacy of the study drug. The computation of PFS may be complicated by the inclusion of censoring rules. In order to support the statistical analysis of PFS, a clinical programmer must create an analysis data set which records each subject's PFS value and censoring status. This paper will describe an elegant method to compute PFS and flag the detail records used in the PFS computation in a single DATA step by using a double DOW loop.

## Introduction

Progression free survival is sometimes an efficacy endpoint in cancer trials. In general, it is the time from a subject's randomization (when they are assigned by chance to a treatment group at the start of the study[4]) until the subject experiences tumor progression or, in the absence of tumor progression, death due to any cause. The criteria to objectively identify tumor progression will vary based upon the study design (one possibility is the use of RECIST criteria), but the basic idea is that the investigator or an independent evaluator has determined that the subject's disease has worsened (perhaps the target tumor has increased sufficiently in size or new tumors have appeared). The computation of progression free survival is complicated, however, by the concept of censoring. Censoring cuts off a subject's data prior to the event of interest (tumor progression or death). The subject's data up to the censoring date are still considered in the analysis, but subjects whose PFS time is censored are treated differently in the statistical analysis than subjects whose PFS time is not censored; thus, the analysis data set must contain a flag to identify whether PFS is censored for each subject. Some reasons to censor a subject's PFS time may include:[3]

- A subject might not experience tumor progression or death as of the time of analysis.
- A subject might commence non-study anti-cancer therapy before experiencing tumor progression
- A subject might miss multiple study visits in which a tumor assessment was planned.

If a subject meets any of the above criteria (which are not exhaustive), the statistical analysis plan (SAP) might call for that subject's PFS to be censored at the date of the last adequate tumor assessment prior to the censoring event; this date is when the subject was last known to be without progression, and it replaces the date of tumor progression or death in the PFS calculation for that subject. The below chart graphically illustrates the concept of censoring with respect to PFS analysis. Solid lines represent PFS; however, if a reason for censoring occurs (as for subjects 003 and 004), PFS for those subjects is cut-off prior to the actual event at the date of the last adequate tumor assessment in which progression was not observed. Dashed lines represent time after censoring which is not considered in PFS.



Therefore, in order to compute progression free survival, we must take into account several events: death, tumor progression, randomization, and any events which might result in censoring. Rather than simply providing an analysis data set which contains the computed PFS, it would be helpful to a reviewer if the data set contained records for each event during the trial that might take part in the PFS computation, with a flag variable identifying the events actually used in the computation. This paper will describe a method, using a double DOW loop, to identify the candidate events that might contribute to PFS, to perform the PFS computation, and to set a flag on the records used in the computation. These actions will occur in a single DATA step. To begin, this paper will briefly explain the concept of a DOW loop, and then extend the explanation to a double DOW loop.

## The DOW Loop

The DOW loop has a rich history in SAS culture. Developed by Don Henderson[5] and later presented in a posting to the SAS-L newsgroup by Ian Whitlock in February 2000[2], it has been discussed, analyzed, and applied to numerous problems in many subsequent user group papers. While the machinery of the DATA step is the first lesson taught to new SAS programmers, it may help to briefly review the basics in order to explain how a DOW loop works. The DATA step is an implicit loop. At the top of the loop (the DATA statement), the observation counter _N_ is incremented and all variables in the program data vector (PDV) are reset to missing unless they have been explicitly retained by the programmer or read in via SET, MERGE, MODIFY, or UPDATE. Within the loop, SAS executes statements written by the programmer; in its most ubiquitous form, a DATA step will contain a single SET statement which reads a single observation from the input data set followed by SAS statements intended to modify that data in some way. At the end of the loop, one observation is written to the output data set (unless there is an explicit OUTPUT statement within the step) and control is returned to the top of the loop. The loop terminates, in the default case, when SAS reaches the end of the input data set.[1]

A DOW loop is a construct in which the SET statement is wrapped in an explicit DO loop which terminates after some condition is met. In a DOW loop, it is the explicit DO loop rather than the implicit DATA step loop which controls the reading of observations. It is critical to understand, however, that the implicit DATA step loop and all implicit actions that SAS performs in the background (such as setting assigned variables to missing at the top of the DATA step) still occur. They just occur after the explicit loop terminates. In its most common form, a DOW loop reads a data set using BY-group processing and terminates the DO loop after each BY-group. For example:

```
data outdata;
   do until(last.id);
      set indata;
      by id;
      /* SAS statements */
   end;
run;
```

An entire BY-group is read from the input data set before the end of the implicit loop (at the RUN statement) is ever reached. Within the DOW loop, any newly created variables retain their values without an explicit RETAIN statement, and then have their values automatically set to missing after each BY-group (when the implicit DATA step loop iterates). Excluding an explicit OUTPUT statement inside the DOW loop leaves the implicit output of the DATA step in place, which means that the above step will output one record per BY-group. This makes summarizing data easy and elegant. Including an explicit OUTPUT statement inside the DOW loop will output one record for each record in the input data set. The programmer may add SAS statements before and after the DOW loop. Statements above the loop will execute before each BY-group is read; statements below the loop will execute after each BY-group is read. To cement these concepts, notice that the following DATA steps are equivalent:

```
/* Without DOW Loop */
data outdata;
   set indata (keep=id code);
   by id;
   length list $100;
   retain list;
   if first.id then list='';
   list = catx(',', list, code);
   if last.id;
run;
```

```
/* With DOW Loop */
data outdata;
   do until(last.id);
      set indata (keep=id code);
      by id;
      length list $100;
      list = catx(',', list, code);
   end;
run;
```

In this simple example, the code which employs the DOW loop is shorter by only a single line. However, use of the DOW loop removes the need to explicitly retain and reset the variable LIST; in a longer program with more retained variables, the resulting improvement in code brevity may be substantial. But, the more important benefit of the DOW

loop is the clarity with which it can represent BY-group processing in a DATA step. Programs are much more difficult to read than they are to write, and so the readability of code must always be a priority if a program is to be maintainable. There are certain classes of problems which, at least in the author's opinion, may be coded much more clearly and cleanly by utilizing the DOW loop construct than via more traditional DATA step techniques.

## The Double DOW Loop

The concept of the DOW loop can be extended to include two iterations over each BY-group. The first iteration of the DOW loop can be used to compute summary statistics for the entire BY-group. The second iteration of the DOW loop can then merge those statistics back onto the detail records. In other words, variables containing summary statistics computed during the first DOW loop retain their value during the iteration of the second DOW loop, and therefore those values can be carried forward onto every record of the BY-group during the iteration of the second DOW loop. The same effect could be achieved in two DATA steps (the first DATA step would create a temporary data set consisting of one summary record per ID, and the second DATA step would merge the temporary data set with the original data set by ID). The combination of two DOW loops in a single DATA step is known as a double DOW loop, illustrated in skeleton form below:

```
data outdata;
   do until(last.id);
      set indata;
      by id;
      /* Statements operate on each record.  Use this section to
         compute summary information representative of the BY-group
      */
   end;

   /* Statements operate on values computed in first DOW loop, which are
      representative of the BY-group as a whole
   */

   do until(last.id);
      set indata;
      by id;
      /* Statements operate on each record. Use this section to merge
         summary information computed earlier back onto the detail records
      */
      output; ** Write one record for each input record;
   end;
run;
```

The key to understanding the double DOW loop is to realize that SAS treats each SET statement independently, as if it were reading from two separate data sets.[1] Thus, you can think of each SET statement having its own independent record pointer. When the first DOW loop terminates, the first SET statement has just read the last observation of the first BY-group and is ready to read from the first observation of the second BY-group. When the second DOW loop commences, SAS treats the second SET statement independently of the first and begins reading the data set from the first observation of the first BY-group. Because both DOW loops break on the same condition (last.id), the SET statements remain synchronized.[2] Thus, every time the implicit DATA step loop iterates, the double DOW loop performs two passes over the current BY-group. It is this two-pass feature that we will exploit in our algorithm to compute PFS and assign flags onto the detail records used in that computation; each BY-group in our algorithm will contain a single subject's records, and so each pass over the BY-group will represent the processing of all of that subject's records. The RUN statement (and thus the implicit iteration of the DATA step) will only be reached after two complete passes over each subject's records, by which time PFS will be computed and all records for the subject will be output.

## The Problem Statement

Recall from the introduction that progression free survival is the time from randomization until tumor progression or, in the absence of tumor progression, death from any cause. Recall also that some subjects might not experience tumor progression or death as of the date of the analysis, or they may experience some intervening event (such as the initiation of non-study therapy) which prevents a fair analysis of that subject's PFS. Such subjects are said to have their PFS time censored, and they will instead have their PFS computed from the date of randomization to a date preceding the censoring event such as their last adequate tumor assessment where progression was not observed.

The statistical analysis plan should detail the rules for censoring; for this paper, we will assume the following rules:

| Situation | Result | PFS Date |
|---|---|---|
| Tumor progression observed by blinded, independent radiologist | Progressed | First assessment date at which progression is documented |
| Death due to any cause in the absence of tumor progression | Progressed | Date of Death |
| Initiation of non-study cancer treatment prior to progression or death | Censored | Last adequate tumor assessment prior to start of non-study therapy |
| Treatment discontinuation prior to progression or death | Censored | Last adequate tumor assessment prior to treatment discontinuation |
| Subject still on study as of time of analysis without tumor progression | Censored | Last adequate tumor assessment prior to date of analysis |
| No baseline or adequate post-baseline images | Censored | Randomization date |

The term “adequate” in the above table means a tumor assessment with a non-missing result which is not marked “Not Evaluable (NE)”; the last adequate tumor assessment is the last time a subject was known to be without disease progression. In order to implement a PFS computation following the above censoring rules, we will need to collect the following data points (note that more complicated censoring rules may require the collection of additional data points for each subject):

- Randomization status and dates for each subject
- MRI/CT imaging assessments of subject tumors, with each assessment graded, via RECIST criteria by an independent radiologist, into the categories of CR (Complete Response), PR (Partial Response), SD (Stable Disease), PD (Progressive Disease), or NE (Not Evaluable).
- Death status and dates for each subject
- Progression status and dates (in this example we assume we will have a declaration of progression made by an independent radiologist distinct from the presence of Progressive Disease (PD) records in the imaging data).
- Non-study anti-cancer therapy start dates
- Treatment discontinuation dates for each subject

It is an exercise left to the reader to concatenate these data sources into an intermediate form consisting of one record per subject per “event”, where the event could be an imaging assessment or some relevant milestone during the study such as subject death or the initiation of non-study therapy. The intermediate data set might look like the following:

| USUBJID | ADT | PRIORITY | PARAMCD | AVALC |
|---|---|---|---|---|
| 001 | 2010-02-17 | 99 | MILESTNE | Baseline Image |
| 001 | 2010-02-22 | 99 | MILESTNE | Randomized |
| 001 | 2010-04-01 | 1 | IMAGE | Stable Disease (SD) |
| 001 | 2010-05-13 | 1 | IMAGE | Partial Response (PR) |
| 001 | 2010-06-24 | 2 | IMAGE | Progressive Disease (PD) |
| 001 | 2010-06-24 | 3 | MILESTNE | Progressed |
| 001 | 2010-07-29 | 5 | MILESTNE | Non-Study Therapy |
| 002 | 2010-04-07 | 99 | MILESTNE | Baseline Image |
| 002 | 2010-04-13 | 99 | MILESTNE | Randomized |
| 002 | 2010-06-03 | 1 | IMAGE | Stable Disease (SD) |

| | | | | |
|---|---|---|---|---|
| 002 | 2010-07-21 | 1 | IMAGE | Partial Response (PR) |
| 002 | 2010-08-01 | 4 | MILESTNE | Death |
| 003 | 2010-04-08 | 99 | MILESTNE | Baseline Image |
| 003 | 2010-05-04 | 99 | MILESTNE | Randomized |
| 003 | 2010-06-03 | 1 | IMAGE | Partial Response (PR) |
| 003 | 2010-06-09 | 0 | IMAGE | Not Evaluable (NE) |
| 003 | 2010-07-08 | 5 | MILESTNE | Non-Study Therapy |
| 003 | 2010-07-08 | 6 | MILESTNE | Treatment Discontinuation |
| 004 | 2010-05-31 | 99 | MILESTNE | Randomized |
| 005 | 2010-05-04 | 99 | MILESTNE | Baseline Image |
| 005 | 2010-05-10 | 99 | MILESTNE | Randomized |
| 005 | 2010-06-12 | 1 | IMAGE | Stable Disease (SD) |
| 005 | 2010-08-30 | 6 | MILESTNE | Treatment Discontinuation |
| 005 | 2010-09-13 | 2 | IMAGE | Progressive Disease (PD) |
| 005 | 2010-09-13 | 3 | MILESTNE | Progressed |
| 005 | 2010-09-16 | 5 | MILESTNE | Non-Study Therapy |
| 006 | 2011-01-13 | 99 | MILESTNE | Baseline Image |
| 006 | 2011-02-01 | 99 | MILESTNE | Randomized |
| 006 | 2011-03-11 | 1 | IMAGE | Stable Disease (SD) |
| 006 | 2011-04-22 | 1 | IMAGE | Partial Response (PR) |
| 006 | 2011-06-03 | 1 | IMAGE | Stable Disease (SD) |
| 006 | 2011-07-02 | 0 | IMAGE | Not Evaluable (NE) |

In the above data set, USUBJID is the subject identifier; ADT is the date of the image assessment or milestone; PARAMCD identifies whether the record represents a tumor assessment (via CT/MRI imaging) or a subject milestone; AVALC describes the imaging assessment result or milestone; and PRIORITY is a tie-breaking integer in the event that images or milestones occur on the same day. In the above example, priority is coded as follows:

| | | |
|---|---|---|
| IMAGE | 0 | Not Evaluable (NE) |
| | 1 | Stable Disease (SD) / Partial Response (PR) / Complete Response (CR) |
| | 2 | Progressive Disease (PD) |
| MILESTNE | 3 | Progressed |
| | 4 | Death |
| | 5 | Non-Study Therapy |
| | 6 | Treatment Discontinuation |
| | 99 | Other milestones not used for censoring |

If more than one record has the same ADT, we want to pick the last IMAGE and the first MILESTNE from that date; thus, the ordering within PRIORITY is such that we favor PD images over non-PD images and favor progression events over death and death over censoring events if they all occur on the same day.

Given the above input data set, we have two goals:

1.) Create a derived record for each subject containing that subject's PFS computation.

2.) Create five new variables which identify whether PFS is censored, provide the reason for censoring (or a description of whether the subject progressed or died), flag the source records used in the PFS computation, flag the PFS summary record itself, and identify the PFS summary record as a derived record.

## The Solution

While there may be many ways to solve this problem, the focus of this paper is a solution using a double DOW loop in a single DATA step. The use of a double DOW loop encourages code brevity by removing the need to explicitly retain and reset temporary variables and by removing the need to create an intermediate summary data set which is later merged onto the original data set. But perhaps more importantly, once the reader becomes comfortable with the idiom of the DOW loop, the logic becomes very natural to follow – some problems are easily conceptualized in terms of processing data via passes through BY-groups, and that is exactly the logic of a DOW loop. In the words of Paul Dorfman, "the DOW-loop lends itself as an ideal logical vehicle by greatly simplifying the alignment of stream-of-consciousness and SAS code."[2] While the step is not particularly long, its explanation will be easier if it is presented in four pieces: the first DOW loop, the code between the DOW loops, the second DOW loop, and the code following the second DOW loop.

The first DOW loop represents the first pass through the BY-group; in other words, the first pass through all records for a given subject. There is a loop counter, OBS, which represents the relative observation number within the BY-group during any given iteration. The relative observation number will turn out to be crucial when synchronizing with the second DOW loop, as the primary purpose of this first DOW loop is to identify key events for each subject and remember the relative observation numbers of those events. During this first pass we determine whether the subject has a baseline image and whether they were randomized and when. Critically, we also need to determine the date of the event (tumor progression or, in the absence of progression, death due to any cause); if there is no event or if the event is preceded by a censoring reason (initiation of non-study treatment, treatment discontinuation, etc.), then we must identify the last adequate imaging assessment prior to that censoring reason. The first DOW loop is shown below:

```
data pfs (
   keep=usubjid adt priority paramcd paramtyp avalc cnsr evntdesc anl01fl crit01fl
);
    length evntdesc $40;

   ** PASS ONE THROUGH BY-GROUP: DETERMINE RELATIVE LOCATION OF VARIOUS EVENTS;
   do obs=1 by 1 until(last.usubjid);
      set indata;
      by usubjid adt priority;

      ** DOES SUBJECT HAVE A BASELINE IMAGE? ;
      if upcase(avalc) = 'BASELINE IMAGE' then baseobs = obs;

      ** REMEMBER RANDOMIZATION;
      if upcase(avalc) = 'RANDOMIZED' then do;
         randobs = obs;
         randdt = adt;
      end;

      ** MILESTONES WITH PRIORITY < 99 ARE POTENTIAL EVENTS OR CENSORING REASONS;
      if evntobs = . and adt > .z and paramcd='MILESTNE' and priority < 99 then do;
         evntobs  = obs;
         evntdesc = avalc;
      end;

      ** IF WE HAVE NOT ALREADY FOUND THE EVENT, UPDATE LAST ADEQUATE IMAGE;
      evalimg = (paramcd='IMAGE' and upcase(avalc) ne 'NOT EVALUABLE (NE)');

      if evntobs = . and evalimg then lastimg = obs;
   end;
```

At the conclusion of the first DOW loop, we have read all observations for a given subject once, and we know the relative observation numbers of the first event or censoring reason and the last evaluable image prior to the event or censoring reason. The next section of code executes between the DOW loops, and therefore the values that it operates on are representative of the subject as a whole. It is in this section that we can finally determine whether PFS is censored and the reason for censoring, making decisions that required us to have already scanned all records for the subject once. Specifically, if a subject does not have any baseline image or evaluable post-baseline images, we censor the PFS time with a reason of "No Baseline and/or Evaluable Images." If the subject died, then it is the death date and not the last image date that we will use in our PFS calculations, so we set the last image observation number to missing. Finally, if the subject did not die, progress, or experience any other censoring reason, we censor the PFS time at their last evaluable image with a reason of "Follow-up Ongoing." This section of code is displayed below:

```
      ** NO BASELINE OR EVALUABLE POST-BASELINE IMAGES OVERRIDES ALL OTHER EVENTS;
      if nmiss(baseobs, lastimg) > 0 then do;
         evntdesc = 'No Baseline and/or Evaluable Images';
         evntobs = .;
         lastimg = randobs;
      end;

      ** IF THE EVENT IS DEATH, THE PFS DATE SHOULD NOT BE BASED UPON THE LAST IMAGE;
      if upcase(evntdesc) = 'DEATH' then lastimg = .;

      ** FLAG CENSORING STATUS;
      if upcase(evntdesc) in ('DEATH', 'PROGRESSED') then cnsr=0;
      else cnsr=1;

      ** IF NO EVENT OR CENSORING REASON OCCURED, ASSIGN DEFAULT CENSORING REASON;
      if missing(evntdesc) then evntdesc = 'Follow-up Ongoing';
```

We now know the final censoring status for the subject as well as the relative observation numbers of the event and of the last adequate image prior to the event. It is now time for the second DOW loop, during which we will carry summary information (such as the censoring reason) onto every record for the subject and we will flag (via ANL01FL) the one or two detail records (the last image record and the event record) which contributed to our determination of censoring status. Finally, we will use the second pass through the subject's data to identify the PFS date, which will either be the date of progression, the date of death, the date of the last adequate image where progression was not observed, or the randomization date depending upon the censoring status. The second DOW loop will output every record for the subject, so that the output data set will contain all original records augmented with new variables.

```
      ** SECOND PASS THROUGH BY-GROUP: FLAG ANALYSIS RECORDS AND OUTPUT EACH RECORD;
      do obs=1 by 1 until(last.usubjid);
         set indata;
         by usubjid adt priority;

         if obs = evntobs or obs = lastimg then do;
            anl01fl='Y';
            if pfsdt = . then pfsdt = adt;
         end;
         else anl01fl='';

         output;
      end;
```

After the second DOW loop, we have output all records for the subject augmented with censoring status, a description of the event, and a flag (ANL01FL) to identify the records that will contribute to PFS. The final section of the DATA step creates a derived record containing the PFS computation (PFS date minus the date of randomization plus one) and a flag, CRIT01FL, which identifies the derived record that relates to the detail records flagged by ANL01FL. CRIT01FL is helpful if an analysis data set contains more than one type of derived record (for example, PFS with censoring and PFS without censoring); in that case, we would have two sets of flags identifying detail records involved in the two derivations (ANL01FL and ANL02FL) and the corresponding CRITxxFL variables would tie the detail records to the derived records. The final section of the DATA step is shown below:

```
      ** CREATE SUMMARY PFS RECORD;
      paramtyp = 'DERIVED';
      paramcd  = 'PFS';
      anl01fl  = ' ';
      crit01fl = 'Y';
      adt      = pfsdt;
      avalc    = put(pfsdt - randdt + 1, 8.0 -L);
      output;
   run;
```

Running the above DATA step on the input data set shown earlier yields the data set on the following page.

## The Resulting Data Set

| USUBJID | ADT | PRIORITY | PARAMCD | PARAMTYP | AVALC | EVNTDESC | CNSR | ANL01FL | CRIT01FL |
|---|---|---|---|---|---|---|---|---|---|
| 001 | 2010-02-17 | 99 | MILESTNE | | Baseline Image | Progressed | 0 | | |
| 001 | 2010-02-22 | 99 | MILESTNE | | Randomized | Progressed | 0 | | |
| 001 | 2010-04-01 | 1 | IMAGE | | Stable Disease (SD) | Progressed | 0 | | |
| 001 | 2010-05-13 | 1 | IMAGE | | Partial Response (PR) | Progressed | 0 | | |
| 001 | 2010-06-24 | 2 | IMAGE | | Progressive Disease (PD) | Progressed | 0 | Y | |
| 001 | 2010-06-24 | 3 | MILESTNE | | Progressed | Progressed | 0 | Y | |
| 001 | 2010-07-29 | 5 | MILESTNE | | Non-Study Therapy | Progressed | 0 | | |
| 001 | 2010-06-24 | 5 | PFS | DERIVED | 123 | Progressed | 0 | | Y |
| 002 | 2010-04-07 | 99 | MILESTNE | | Baseline Image | Death | 0 | | |
| 002 | 2010-04-13 | 99 | MILESTNE | | Randomized | Death | 0 | | |
| 002 | 2010-06-03 | 1 | IMAGE | | Stable Disease (SD) | Death | 0 | | |
| 002 | 2010-07-21 | 1 | IMAGE | | Partial Response (PR) | Death | 0 | | |
| 002 | 2010-08-01 | 4 | MILESTNE | | Death | Death | 0 | Y | |
| 002 | 2010-08-01 | 4 | PFS | DERIVED | 111 | Death | 0 | | Y |
| 003 | 2010-04-08 | 99 | MILESTNE | | Baseline Image | Non-Study Therapy | 1 | | |
| 003 | 2010-05-04 | 99 | MILESTNE | | Randomized | Non-Study Therapy | 1 | | |
| 003 | 2010-06-03 | 1 | IMAGE | | Partial Response (PR) | Non-Study Therapy | 1 | Y | |
| 003 | 2010-06-09 | 0 | IMAGE | | Not Evaluable (NE) | Non-Study Therapy | 1 | | |
| 003 | 2010-07-08 | 5 | MILESTNE | | Non-Study Therapy | Non-Study Therapy | 1 | Y | |
| 003 | 2010-07-08 | 6 | MILESTNE | | Treatment Discontinuation | Non-Study Therapy | 1 | | |
| 003 | 2010-06-03 | 6 | PFS | DERIVED | 31 | Non-Study Therapy | 1 | | Y |
| 004 | 2010-05-31 | 99 | MILESTNE | | Randomized | No Baseline and/or Evaluable Images | 1 | Y | |
| 004 | 2010-05-31 | 99 | PFS | DERIVED | 1 | No Baseline and/or Evaluable Images | 1 | | Y |
| 005 | 2010-05-04 | 99 | MILESTNE | | Baseline Image | Treatment Discontinuation | 1 | | |
| 005 | 2010-05-10 | 99 | MILESTNE | | Randomized | Treatment Discontinuation | 1 | | |
| 005 | 2010-06-12 | 1 | IMAGE | | Stable Disease (SD) | Treatment Discontinuation | 1 | Y | |
| 005 | 2010-08-30 | 6 | MILESTNE | | Treatment Discontinuation | Treatment Discontinuation | 1 | Y | |
| 005 | 2010-09-13 | 2 | IMAGE | | Progressive Disease (PD) | Treatment Discontinuation | 1 | | |
| 005 | 2010-09-13 | 3 | MILESTNE | | Progressed | Treatment Discontinuation | 1 | | |
| 005 | 2010-09-16 | 5 | MILESTNE | | Non-Study Therapy | Treatment Discontinuation | 1 | | |
| 005 | 2010-06-12 | 5 | PFS | DERIVED | 34 | Treatment Discontinuation | 1 | | Y |
| 006 | 2011-01-13 | 99 | MILESTNE | | Baseline Image | Follow-up Ongoing | 1 | | |
| 006 | 2011-02-01 | 99 | MILESTNE | | Randomized | Follow-up Ongoing | 1 | | |
| 006 | 2011-03-11 | 1 | IMAGE | | Stable Disease (SD) | Follow-up Ongoing | 1 | | |
| 006 | 2011-04-22 | 1 | IMAGE | | Partial Response (PR) | Follow-up Ongoing | 1 | | |
| 006 | 2011-06-03 | 1 | IMAGE | | Stable Disease (SD) | Follow-up Ongoing | 1 | Y | |
| 006 | 2011-07-02 | 0 | IMAGE | | Not Evaluable (NE) | Follow-up Ongoing | 1 | | |
| 006 | 2011-06-03 | 0 | PFS | DERIVED | 123 | Follow-up Ongoing | 1 | | Y |

## Conclusion

Progression free survival is sometimes an efficacy endpoint for oncology clinical trials. While the basic definition of PFS provided by the FDA sounds simple, the addition of censoring rules can make the determination of PFS for each subject complex. This paper illustrated an application of the double DOW loop to compute PFS based upon an ordered input data set of tumor assessments and milestones. The solution preserved the original source records with those used in the PFS computation flagged for ease of review. The use of a double DOW loop lends elegance to the solution. For, it adds brevity by removing the need to explicitly retain and reset temporary variables and by removing the need to create an intermediate summary data set which is later merged onto the original data set. But perhaps more importantly, once the reader becomes comfortable with the idiom of the DOW loop, the logic becomes very natural to follow – some problems are easily conceptualized in terms of processing data via passes through BY-groups, and PFS computation would seem to fall within this class of problems.

## References


1. *SAS Online Doc 9.1.3*. http://support.sas.com/onlinedoc/913/docMainpage.jsp.

2. Dorfman, Paul M. "The DOW-Loop Unrolled." Proceedings of the 2010 Southeast SAS User's Group Conference. Paper BB-13.

3. Food and Drug Administration. "Guidance for Industry: Clinical Trial Endpoints for the Approval of Cancer Drugs and Biologics." May 2007.

4. ClinicalTrials.gov. "Glossary of Clinical Trials Terms." http://clinicaltrials.gov/ct2/info/glossary.

5. Brucken, Nancy. "2 PROC TRANSPOSEs = 1 DATA Step DOW-Loop." Proceedings of the 2011 Pharmaceutical SAS User's Group Conference. Paper CC26.


## Acknowledgements


I would like to thank Paul Slagle for his supporting my writing of this paper and for designing the PFS analysis data set structure which motivated the algorithm that I developed for this paper. I would also like to thank Mei Du, Lisa Fine, Nancy Brucken and Jeff Parno for reading this paper and providing extremely constructive feedback. Of course, any remaining errors in this paper are my own.


## Contact Information


Matt Nizol
United BioSource Corporation
2200 Commonwealth Blvd, Suite 100
Ann Arbor, MI 48105
Phone: +1 734 994 8940
Internal Extension: 1605
Email: matt.nizol@unitedbiosource.com
www.unitedbiosource.com