Using SAS® to Identify Cancer Treatment Patterns in Administrative Claims Data
Laurie Hamilton

ABSTRACT
This paper presents an approach using Base SAS® routines and the SAS® DATA Step to identify multiple sequential chemotherapy regimens (lines of therapy) composed of one or more drugs. The identification of specific regimens is data-driven rather than pre-specified. The claims are pre-processed using date arithmetic and the resulting dataset is then passed once through a series of DATA Steps. These steps key on RETAIN statements and use DATA Step standbys such as DO loops with exit controls and one-dimensional arrays to capture regimen begin and end dates and interruption events with associated switch dates. A SAS® character function is applied to created text versions of regimen descriptions directly from the source claims. Strategic boundaries and allowable gaps in drug-on-board are controlled by global macro variables, making the program easily adaptable to a variety of specifications. Processing concludes with a DATA Step/ODS strategy to identify and output random samples including both outcomes and input claims for efficient quality control.

INTRODUCTION
The complexity of treatment patterns for cancer using chemotherapy and biologic drugs continues to grow as new drugs are developed and the number of regimens (lines) of treatment prescribed increase, especially for metastatic cancers. Administrative claims data add much-needed information on real-world treatment patterns to the published recommended regimens, and SAS® programmers in Outcomes Research settings are being called on more and more to develop pattern-recognition routines for multi-drug, multi-line chemotherapy regimens as a prelude to cost and utilization analyses.

The challenges to the SAS programmer in this instance are twofold. First, the data sources containing information on chemo-biologic regimen drugs include both Inpatient and Outpatient medical claims and Outpatient pharmacy claims. These data sources frequently have different data structures which the programmer must reconcile to build an input dataset for regimen identification. Second, as information grows, the specifications for a regimen-identification project become more and more complex, which points to the need for data-driven solutions to increase efficiencies in the associated programming.

This paper focuses on the challenge to the programmer to develop a data-driven SAS® pattern recognition solution which is flexible enough to be adapted to varying specifications without sacrificing coding simplicity. We will consider common elements involved in defining chemo-biologic regimens from claims data: identification of boundary dates for the beginning and ending of a regimen, incorporating information on drug onboard with specifications for allowed gaps in therapy to identify drugs eligible for a regimen, and considering when to define the appearance of claims for a new agent as a regimen switch point. The discussion below assumes that the input data has been reconciled to one claim/day for a chemo-biologic drug, that the associated Days Supply has been imputed if needed, and that the original NDC or Procedure Codes have been reconciled to one standard format per drug in one database field.

STEP 1: UNDERSTANDING THE SPECIFICATIONS

OVERVIEW AND SPECIFICATIONS FOR THIS EXAMPLE
Specifications for a project of this type might be as general as a few lines in an analysis plan, or as specific as a detailed listing of required analysis variables and associated rules for their creation. Table shells, if provided, can be a supplemental source of guidance on study requirements. One overriding consideration to keep in mind is that programming will need to accommodate the fact that the drugs comprising a line and associated line begin and end dates, will vary among the study patients.

Table 1 below contains a summary of the rules for the example presented here. A line begins on the date of the first drug claim in the target period which is refilled with a specified time period. Any additional drugs with a qualifying refill which have a first fill within 28 days of the line begin date will belong to the line – 28 days is the specified Days To Establish a Line. The target period begins on Day 1 of each patient’s study period for Line 1, and is variable for all subsequent lines. A line will continue until any of the following occurs: 1) at least one drug in the line has a Days Supply which continues beyond the patient’s study end date 2) there is a 90-day gap in drug on board for all drugs in the line, or 3) a regimen switch is identified.
Table 1. Regimen-Identification Specifications for the Example Presented in this Paper

<table>
<thead>
<tr>
<th>What starts a line?</th>
<th>A line begins on the date of the first drug claim in the target period which is refilled within a specified grace period after the prescription runs out. The grace period in this example is either 0 or 28 days.</th>
</tr>
</thead>
<tbody>
<tr>
<td>How to we identify additional drugs eligible for a line?</td>
<td>All drugs with a first claim within 28 days of the first identified qualifying drugs first claim, which are refilled within the grace period.</td>
</tr>
<tr>
<td>What defines the end of a line?</td>
<td>A line ends if:</td>
</tr>
<tr>
<td></td>
<td>• There is a 90 day gap in drug on-hand for all drugs in the line.</td>
</tr>
<tr>
<td></td>
<td>• The Days Supply of any of the drugs in the line extends past the end of a patient’s study period.</td>
</tr>
<tr>
<td></td>
<td>• The line is interrupted by the beginning of a new line.</td>
</tr>
<tr>
<td>What constitutes a valid switch to a new line?</td>
<td>A line is considered to be ended by the beginning of a new line by a prescription for a new drug which is refilled within the grace period.</td>
</tr>
</tbody>
</table>

DATA-DRIVEN DRUG IDENTIFICATION

The SAS code discussed in this paper takes an input dataset containing chemo-biologic drug claims with no further specifications of the actual drugs or expected regimens. It does assume, however, that the input data contains only chemo-biologic claims of interest to the study being programmed, and within the target time period specific to each patient’s study eligibility boundaries.

The SAS code discussed in this paper takes an input dataset containing chemo-biologic drug claims with no further specifications of the actual drugs or expected regimens. It does assume, however, that the input data contains only chemo-biologic claims of interest to the study being programmed, and within the target time period specific to each patients’ study eligibility boundaries.

Preparation of the initial set of claims is beyond the scope of this paper, but in general would include: 1) subsetting from all Inpatient, Outpatient and Pharmacy claims for the study population to targeted drug claims based on a master list of target drugs provided in the study specifications, 2) conversion of multiple codes for a drug to a single code held in one database field, and 3) imputation of Days Supply if needed to fill in missing data.

STEP 2: PREPARING THE DRUG CLAIMS DATA

As a starting point for preparing the claims data, we are assuming an input dataset containing only chemo-biologic claims for drugs relevant to the cancer being studied. This drug claims database, which is at least one step removed from the raw source claims, contains the drug identification code (DRUG) and Days Supply (DAYSUPP). Any imputations for Days Supply and claims will have been rolled up to one claim/drug/day.

In order to streamline programming, we next use a DATA Step to enhance these input claims by adding the allowed refill grace period (GRACE_PERIOD) using direct assignment or a PROC FORMAT statement and by converting critical dates into DAYS within each patient’s study period. The Patient Boundaries and Intermediate Drug Claims Sections of Table 2, below, contains the flow of the enhancements for the claims for the three sample patients considered in this paper.

First we consider the boundaries of each patient’s study period. The beginning date for study participation becomes DAY 1, and all other dates are converted into a DAY within the study period using date arithmetic.

\[
\text{STUDY\_END} = \text{Study End Date} – \text{Study Begin Date} + 1
\]

We continue using similar logic to convert each drug claim day into a DRUG\_START DAY and finalize the enhanced claims by calculating a DRUG\_END DAY. Calculation of the DRUG\_END DAY takes into account both the DAYS SUPPLY for the prescription and the allowed the GRACE\_PERIOD for the specific drug

\[
\text{DRUG\_END} = \text{DRUG\_START} + \text{DAYSUPP} + \text{GRACE\_PERIOD} – 1
\]
Table 2. Example Intermediate and Enhanced Claims

<table>
<thead>
<tr>
<th>Patient Study Boundaries</th>
<th>Intermediate Claims</th>
<th>Enhanced Claims (Include all Fields From Patient Boundaries and Intermediate Claims)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Patient</strong></td>
<td><strong>Claim Date</strong></td>
<td><strong>DRUG_CODE</strong></td>
</tr>
<tr>
<td>P1</td>
<td>12/07/07</td>
<td>Chemo6</td>
</tr>
<tr>
<td></td>
<td>12/14/07</td>
<td>Chemo6</td>
</tr>
<tr>
<td>P2</td>
<td>11/28/08</td>
<td>Chemo2</td>
</tr>
<tr>
<td></td>
<td>12/22/08</td>
<td>Chemo2</td>
</tr>
<tr>
<td></td>
<td>01/09/09</td>
<td>Chemo2</td>
</tr>
<tr>
<td></td>
<td>12/01/08</td>
<td>Chemo4</td>
</tr>
<tr>
<td></td>
<td>12/22/08</td>
<td>Chemo4</td>
</tr>
<tr>
<td></td>
<td>12/01/08</td>
<td>Chemo6</td>
</tr>
<tr>
<td></td>
<td>12/22/08</td>
<td>Chemo6</td>
</tr>
<tr>
<td></td>
<td>01/09/09</td>
<td>Chemo6</td>
</tr>
<tr>
<td>P3</td>
<td>05/30/07</td>
<td>Bio2</td>
</tr>
<tr>
<td></td>
<td>04/22/07</td>
<td>Bio3</td>
</tr>
<tr>
<td></td>
<td>05/09/07</td>
<td>Bio3</td>
</tr>
<tr>
<td></td>
<td>06/19/07</td>
<td>Chemo2</td>
</tr>
<tr>
<td></td>
<td>07/10/07</td>
<td>Chemo2</td>
</tr>
<tr>
<td></td>
<td>04/22/07</td>
<td>Chemo4</td>
</tr>
<tr>
<td></td>
<td>05/09/07</td>
<td>Chemo4</td>
</tr>
<tr>
<td></td>
<td>06/26/07</td>
<td>Chemo5</td>
</tr>
<tr>
<td></td>
<td>07/17/07</td>
<td>Chemo5</td>
</tr>
</tbody>
</table>

Patient 1 (P1), for example, enters the study on 11/19/07 (DAY 1) and ends on 12/31/07 DAY 43:

\[
\text{STUDY\_END} = \text{‘31dec2007’} - \text{‘19nov2007’} + 1 = 43
\]

This patient had two claims for the drug Chemo6. The first claim on 12/07/07 is on DAY 19 of this patient’s study period. With a DAYSUPP of 28 and a GRACE_PERIOD of 0, the drug supply for this claim ends on DAY 46:

\[
\text{DRUG\_START} = \text{Claim Date} - \text{Study Start Date} + 1 = 19
\]

\[
\text{DRUG\_END} = \text{DRUG\_START} + \text{DAYSUPP} + \text{GRACE\_PERIOD} - 1 = 19 + 28 + 0 - 1 = 46
\]

**STEP 3: CONTROLLING THE PROGRAM WITH GLOBAL MACRO VARIABLES**

We will next establish a set of global macro variables which will allow control of critical variables and boundaries with one declaration, avoiding hard coding at multiple points in the program. The values for the first set of global macro variables discussed below come from the specifications. The values from the second set will be specific to the dataset being analyzed.
PARAMETERS FROM SPECIFICATIONS

The specifications contain a number of parameters which may be addressed multiple times during programming so lend themselves to being specified once at the beginning of the program as global macro variables. In all cases, the syntax to assign and subsequently address parameters specified as global macro variables is:

```bash
%let <name of global macro variable) = <value to be assigned>
&<name of global macro variable>
```

The variables to be assigned as global for this program are in Table 3 below. In addition to the number of the Line being programmed (&LN), these include the number of days required to establish a line (&ESTABLISH) and the gap allowed before a Line is considered ended (&ENDGAP).

Table 3. Global Macro Variables From Specifications

<table>
<thead>
<tr>
<th>Parameter from Specifications</th>
<th>Global Macro Variable Assignment</th>
<th>Calling Global Macro Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of line being established In this iteration</td>
<td>%let ln = 1;</td>
<td>&amp;ln</td>
</tr>
<tr>
<td>Days to establish a line</td>
<td>%let establish = 28;</td>
<td>&amp;establish</td>
</tr>
<tr>
<td>Gap days to end a line</td>
<td>%let endgap = 90;</td>
<td>&amp;endgap</td>
</tr>
<tr>
<td>Number of previous line established (Used for all except Line 1)</td>
<td>%let prevln = ;</td>
<td>&amp;prevln</td>
</tr>
</tbody>
</table>

DATA-DRIVEN PARAMETERS

Two parameters specific to unique sets of claims data, the maximum number of unique drug codes and the maximum length of a drug code, will also be referenced frequently during the program so we would also like to put them into global macro variables. The steps to create these global macro variables are in Table 4 below. We can follow the steps to assign the data-driven global macro variable for the maximum length of a drug code in Column A of Table 4. We first use a DATA Step and the LENGTH function to capture the length of each code followed by PROC MEANS to output a dataset containing the maximum value of length. The macro variable &MAXLEN is then created using the DATA Step Interface routine CALL SYMPUT, which allows us to communicate information from within a DATA Step to the program as a whole:

```bash
data _null_;
  set max_length;
  call symput ('maxlen',left(put(maxlen,20.)));
run;
```

Table 4. Data-Driven Global Macro Variables

<table>
<thead>
<tr>
<th>A. Maximum Drug Code Length</th>
<th>B. Maximum Number of Drug Codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global macro variable = &amp;MAXLEN</td>
<td>Global macro variable = &amp;MAXDRUGS</td>
</tr>
</tbody>
</table>

/* capture the length of each code */
data lengths;
  set claims;
  code_length = length(drug);
run;

/* get the max length in the data */
proc means data = lengths noprint;
  var code_length;
  output out = max_length max = maxlen;
run;

/* assign max length to macro var */
data _null_;
  set max_length;
  call symput ('maxlen',left(put(maxlen,20.)));
run;

/* create unique list of codes/patient */
proc sort data = claims
  out = tempsort nodupkey;
  by pid drug;
run;

/* count number of drug codes per patient */
proc summary data = tempsort nway missing;
  class pid;
  output out = drug_counts
    (rename = (_freq_ = num_drugs));
run;

/* get the max number of drugs in data */
proc means data = drug_counts noprint;
  var num_drugs;
  output out = max_drugs
    max = maxdrugs;;
run;

/* assign max # of drugs to macro var */
data _null_;
  set max_drugs;
  call symput ('maxdrugs', put(maxdrugs,20.));
run;
STEP 4: IDENTIFYING THE START OF A REGIMEN

For this application, the start of a line of therapy is determined by the drug with the earliest DRUG_START date in the target period which has a qualifying refill. The Line will also include any additional drugs with a prescription during the initiation window (&ESTABLISH), as long as the additional drugs also have a qualifying refill. We will begin by processing the Line 1 target period, which begins on DAY 1 of each patient's study period and looks for claims throughout the period (through STUDY_END).

ORGANIZING THE DATA

First we arrange the data by sorting by patient/drug/claim day. The results for our sample patients are in Table 5 below. We'll refer to this version of the enhanced claims throughout the remainder of the paper as we check the results of regimen assignment.

Table 5. Enhanced Claims Sorted and Ready for Processing

<table>
<thead>
<tr>
<th>Patient (PID)</th>
<th>Study Period End Day (STUDY_END)</th>
<th>Drug Code (DRUG)</th>
<th>Claim Day (DRUG_START)</th>
<th>Day of Last Drug Availability (DRUG_END)</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>43</td>
<td>Chemo6</td>
<td>19</td>
<td>46</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Chemo6</td>
<td>26</td>
<td>53</td>
</tr>
<tr>
<td>P2</td>
<td>215</td>
<td>Chemo2</td>
<td>1</td>
<td>44</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Chemo2</td>
<td>25</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Chemo2</td>
<td>43</td>
<td>86</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Chemo4</td>
<td>4</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Chemo4</td>
<td>25</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Chemo6</td>
<td>4</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Chemo6</td>
<td>25</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Chemo6</td>
<td>43</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Chemo6</td>
<td>178</td>
<td>205</td>
</tr>
<tr>
<td>P3</td>
<td>379</td>
<td>Bio2</td>
<td>43</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bio3</td>
<td>5</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bio3</td>
<td>22</td>
<td>59</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Chemo2</td>
<td>63</td>
<td>116</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Chemo2</td>
<td>84</td>
<td>141</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Chemo4</td>
<td>5</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Chemo4</td>
<td>22</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Chemo5</td>
<td>70</td>
<td>90</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Chemo5</td>
<td>91</td>
<td>118</td>
</tr>
</tbody>
</table>

IDENTIFY THE EARLIEST QUALIFYING CLAIM (FIRST REFILL) FOR EACH DRUG

Here we use BY and RETAIN statements to control the relationship among claims read sequentially and to give us a window to data values in previous claims. The BY statement referencing PID and DRUG allows us to differentiate the first and last claims for patients and drugs within patients. The RETAIN variables will contain critical values we need to “remember” between claims. Refilled_drug, firstfill_start and firstfill_end contain the DRUG, DRUG_START and DRUG_END days for the current and previous claims for the drug being processed. Refilled is a DO Loop control flag to indicate a qualifying refill has been found. The code below initializes these variables for the first claim. Note that it also uses our global macro variables &:MAXLEN and &LN:

```sas
data line&ln._first_refills;
  set claims;
  by pid drug;
  /* use global macro variable for length statement */
  length refilled_drug $&maxlen;

  retain firstfill_start
  firstfill_end
  refilled_drug
  refilled;
  /* first claim for a drug */
  if first.drug then do;
    refilled_drug = drug;
    firstfill_start = drug_start;
    firstfill_end   = drug_end;
    refilled       = 0;
  end;
```
All claims except the first for a PID/DRUG will be processed by an ELSE clause:

```plaintext
/* any other claim for the drug */
else do;
  if refilled = 0 then do;
      /* refilled in time, capture */
      if (firstfill_start) lt drug_start le (firstfill_end + 1) then do;
        refilled = 1;
        refilled_start = drug_start;
        refilled_end = drug_end;
      output;
  end;
  /* not a qualifying refill, use as the beginning of the next set */
  else firstfill_start = drug_start;
end;
run;
```

The DO Loop in this clause is controlled by the `refilled` flag, which was initialized to 0 on the first claim for each a PID/DRUG. As long as we haven’t found a qualifying refill the ELSE clause will be executed to check if the current prescription date is within the qualifying period. If the claim qualifies, the claim end date is captured, the `refilled` flag is set to 1 and the claim is output to the first refills dataset. If the claim does not qualify, it is considered as a potential first claim and its DRUG_START day is moved to the RETAINed `firstfill_start` variable for comparison to the next claim in date order.

At the completion of this data step, we have a dataset with one claim for each patient/drug combination that meets the qualifying refill requirements. For our sample:

<table>
<thead>
<tr>
<th>pid</th>
<th>refilled_drug</th>
<th>firstfill_start</th>
<th>firstfill_end</th>
<th>refilled_start</th>
<th>refilled_end</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Chemo6</td>
<td>19</td>
<td>46</td>
<td>26</td>
<td>53</td>
</tr>
<tr>
<td>P2</td>
<td>Chemo2</td>
<td>1</td>
<td>44</td>
<td>25</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td>Chemo4</td>
<td>4</td>
<td>31</td>
<td>25</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td>Chemo6</td>
<td>4</td>
<td>31</td>
<td>25</td>
<td>52</td>
</tr>
<tr>
<td>P3</td>
<td>Bio3</td>
<td>5</td>
<td>32</td>
<td>22</td>
<td>59</td>
</tr>
<tr>
<td></td>
<td>Chemo2</td>
<td>63</td>
<td>116</td>
<td>84</td>
<td>141</td>
</tr>
<tr>
<td></td>
<td>Chemo4</td>
<td>5</td>
<td>35</td>
<td>22</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td>Chemo5</td>
<td>70</td>
<td>90</td>
<td>91</td>
<td>118</td>
</tr>
</tbody>
</table>

P1’s first claim for Chemo6 is in the first_refills dataset based on the subsequent qualifying refill on DAY 26. The results are similar for the remaining PID/DRUG combinations with the exception of P3/Bio2, which had no qualifying refill.

**STEP 5: SELECT DRUGS IN THE LINE**

Next we sort the dataset containing all qualifying earliest claims by PID/firstfill_day and use a somewhat more complicated DATA Step to select the drugs in the line. This step creates two output datasets. The first is the Line start summary, which will contain one record for each PID with prescriptions in the first_refills dataset. The second is a vertical list of all the drugs in the Line by PID to be used to identify claims for processing in subsequent steps. The information for all drugs for each patient is captured in two one-dimensional arrays – one containing drug codes and the other containing the associated firstfill_day for that drug. The length of the variables in the drug codes array, and the number of elements in the array, are controlled by the global macro variables `&MAXLEN` and `&MAXDRUGS` created in Step 2:

```plaintext
data line&ln._start_summary /* Line summary, one record per PID */
line&ln._drugs;  /* Vertical list of line drugs for each PID */
set line&ln._first_refills_sort;
by pid;
length line&ln._regimen $ 50;
/* arrays for drug codes and fill days */
length drug1-drug&maxdrugs. $ &maxlen;
array adrug drug1-drug&maxdrugs.;
array adrug_start drug_start1-drug_start&maxdrugs.;
```
To control processing for this step we will RETAIN the Line start date, the drug arrays and a variable to indicate the current position in the arrays:

```plaintext
/* retained variables */
retain line&ln._start /* Line start day */
   drug1-drug&maxdrugs /* Line drug codes array */
   drug_start1-drug_start&maxdrugs /* Drug start days array */
   drugnum /* Current Array position */
```

The information in the first claim for each patient is entered into the first positions in the arrays, which are retained and not output until the last claim for each PID is processed. Since the first claim automatically belongs to the Line, the PID and DRUG are also output to the dataset containing the vertical drug list for each patient:

```plaintext
/* first claim for a patient */
if first.pid then do;
   /* initialize the arrays */
   do i = 1 to &maxdrugs;
      adrug{i} = ' '; adrug_start{i} = .;
   end;
   /* set drug counter and capture information for first drug */
   drugnum = 1;
   line&ln._start = firstfill_start;
   drug_start{1} = line&ln._start;
   drug1 = drug;
   output line&ln._drugs;
end;
```

For all subsequent drugs for a patient, we test whether the firstfill_day is within the study-specified window (&ESTABLISH). If the drug qualifies, we increment the array counter to the next position, and add the drug code and first-fill day information to the array. The drug is added to the line drugs vertical dataset as well:

```plaintext
/* subsequent claims for a patient */
else do;
   /* found a drug refilled in the required timeframe, capture */
   if (firstfill_start - line&ln._start) le (&establish) then do;
      drugnum = drugnum + 1;
      adrug{drugnum} = drug;
      adrug_start{drugnum} = firstfill_start;
   end;
   output line&ln._drugs;
end;
```

Once processing is completed for a patient, we create the text string containing a list of all drugs in the Line regimen and output the summary record. The regimen field will be composed of the codes for all qualifying drugs, in alpha-numeric order (because of the sort going into this step), separated by a hyphen.

```plaintext
/* finished processing a patient */
if last.pid then do;
   /* assign regimen name and capture in a separate dataset */
   line&ln._regimen = left(drug1);
   do i = 2 to &maxdrugs;
      if adrug{i} ne ' ' then do;
         line&ln._regimen = left(trim(line&ln._regimen)) ||'-'||left(adrug{i});
      end;
   end;
   output here.line&ln._start_summary;
end;
run;
```

All patients in this sample have qualifying drugs in Line 1. For P2, the Line begins on Day 1, driven by drug Chemo2 as the earliest qualifying refill and joined by the DAY 4 qualifying claims for drugs Chemo4 and Chemo6:

<table>
<thead>
<tr>
<th>pid</th>
<th>line1_start</th>
<th>line1_regimen</th>
<th>drug1</th>
<th>drug_start1</th>
<th>drug2</th>
<th>drug_start2</th>
<th>drug3</th>
<th>drug_start3</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>19</td>
<td>Chemo6</td>
<td>Chemo6</td>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td>1</td>
<td>Chemo2-Chemo4-Chemo6</td>
<td>Chemo2</td>
<td>1</td>
<td>Chemo4</td>
<td>4</td>
<td>Chemo6</td>
<td>4</td>
</tr>
<tr>
<td>P3</td>
<td>5</td>
<td>Bio3-Chemo4</td>
<td>Bio3</td>
<td>5</td>
<td>Chemo4</td>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
STEP 6: DETERMINE END OF LINE

For this paper, a line ends under one of three conditions: 1) a 90 day gap in drug in-hand for all drugs in the regimen (&ENDGAP), 2) no gap, but drug in-hand extends past period end date or 3) the line is interrupted by the qualifying refill of a non-line drug.

CHECK FOR LINE ENDING WITH A GAP OR END OF PATIENT’S STUDY PERIOD

In this section, we will be testing for the first two conditions and assigning line-end dates and line-end reasons, if applicable. This step also includes creation of a QA variable identifying the type of gap found. We begin by selecting all claims for regimen drugs except for the first in the period which established the drug as in the regimen. This is done in a DATA Step merging the original claims dataset with the vertical list of regimen drugs for each patient created in Step 4. For efficiency, this step creates a second set of non-line drug claims which will be used later to check for interrupts in the Line:

```sas
data line&ln._claims_for_gap_check /* Claims to check for gaps */
  line&ln._claims_for_interrupt_check; /* Claims to check for interrupts*/
merge line&ln._drugs (keep = pid drug refilled_start in = inline
  claims (keep = pid study_end drug drug_start drug_end)); by pid drug;
/* all line drug claims starting with qualifying refill claim */
if inline and drug_start ge refilled_start then
output line&ln._claims_for_gap_check;
/* all non-line drug claims */
else if not inline then
output line&ln._claims_for_interrupt_check;
run;
```

The claims to be used for the Gap Check are:

<table>
<thead>
<tr>
<th>pid</th>
<th>study_end</th>
<th>drug</th>
<th>drug_start</th>
<th>drug_end</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>43</td>
<td>Chemo6</td>
<td>26</td>
<td>53</td>
</tr>
<tr>
<td>P2</td>
<td>215</td>
<td>Chemo2</td>
<td>25</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Chemo2</td>
<td>43</td>
<td>86</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Chemo4</td>
<td>25</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Chemo6</td>
<td>25</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Chemo6</td>
<td>43</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Chemo6</td>
<td>178</td>
<td>205</td>
</tr>
<tr>
<td>P3</td>
<td>379</td>
<td>Bio3</td>
<td>22</td>
<td>59</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Chemo4</td>
<td>22</td>
<td>64</td>
</tr>
</tbody>
</table>

The gap-check DATA Step is similar to the steps we used to establish the line. The critical RETAINed variable (last_claim_end) contains the DRUG_END day for each preceding claim. This variable is initialized on the first claim for a PID/DRUG. For each subsequent PID/DRUG claim, the preceding drug end day is compared to the drug start day of the current claim to calculate the drug-in-hand gap in days. Processing is controlled only by this value and FIRST. And LAST. values of PID/DRUG:

```sas
data line&ln._gaps_checked;
set line&ln._claims_for_gap_check;
by pid drug;
length gap_type $ 100;
retain last_claim_end;
/* this is the refill, set the parameters */
if first.drug then do;
  this_claim_gap = 0;
  last_claim_end = drug_end;
end;
else this_claim_gap = drug_start - last_claim_end - 1;
```
If the calculated gap is greater than the study specific gap required to end a Line (&ENDGAP), we capture the gap start date (the first day following the last day of drug-in-hand) and assign a QA variable containing a description of the type of gap identified:

/* found one */
if this_claim_gap ge &endgap then do;
  gap_type = 'claims gap';
  gap_start = last_claim_end + 1;
output;
end;

If the calculated gap is not large enough to end the Line, processing depends on whether or not we are looking at the last claim for a PID/DRUG. In the case of the last claim for a PID/DRUG, we conclude by calculating the days between the DRUG_END Day for the current claim and the patient's STUDY_END Day (post_gap). A post_gap value greater than or equal to &ENDGAP ends the line with a gap. If we have not found a gap and are not processing the last claim for a PID/DRUG, the current claim becomes the first claim for identification of a possible subsequent gap.

/* no gap yet */
else do;
  /* last claim for a drug */
  if last.drug then do;
    /* calculate days from drug in-hand end to end of pat study period */
    post_gap = study_end - drug_end;
    /* gap between last script and study end */
    if post_gap ge &endgap then do;
      gap_type = 'last claim gap';
      gap_start = last_claim_end + 1;
    output;
  end;
  /* last script is on or > than end of patient study period, no gap */
else last_claim_end = drug_end;
end;
run;

For our example, the gap analysis identified drug-level qualifying gaps for patients P2 and P3:

<table>
<thead>
<tr>
<th>pid</th>
<th>drug</th>
<th>gap_type</th>
<th>gap_start</th>
<th>post_gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>P2</td>
<td>Chemo2</td>
<td>last claim gap</td>
<td>69</td>
<td>129</td>
</tr>
<tr>
<td></td>
<td>Chemo4</td>
<td>last claim gap</td>
<td>53</td>
<td>163</td>
</tr>
<tr>
<td></td>
<td>Chemo6</td>
<td>claims gap</td>
<td>71</td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td>Bio3</td>
<td>last claim gap</td>
<td>60</td>
<td>320</td>
</tr>
<tr>
<td></td>
<td>Chemo4</td>
<td>last claim gap</td>
<td>65</td>
<td>315</td>
</tr>
</tbody>
</table>

We conclude the gap check by sorting the gap information, selecting the gap drug with the latest start day. This information is then added to a list of all study patients' start_summary to create the gap_summary dataset:

```
proc sort data = line&ln._gaps_checked
  out = gaps_checked_sort; /*sort by PID,drug with latest gap first*/
run;

data line&ln._gap;
set gaps_checked_sort (rename = (drug = gap_drug));
by pid;
  if first.pid then output; /* select one gap claim per patient */
run;

/* create the gap summary dataset with one claim for each study patient */
data line&ln._gap_summary;
merge line&ln._start_summary (keep = pid line&ln._regimen line&ln._start)
  line&ln._gap
by pid;
run;
```
Two of the three patients considered in this example had drug-in-hand gaps which ended this Line. For P2, the line was ended starting on DAY 71 with a claims gap for drug Chemo6. The Line ended in a gap between the end of the last Chemo4 claim and the end of the study period for P3. As discussed above, for completeness this dataset contains a record for P1 although this patient had no gap (identified as gap_drug = ").

<table>
<thead>
<tr>
<th>pid</th>
<th>gap_drug</th>
<th>gap_type</th>
<th>gap_start</th>
<th>post_gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td>Chemo6</td>
<td>claims gap</td>
<td>71</td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td>Chemo4</td>
<td>last claim gap</td>
<td>65</td>
<td>315</td>
</tr>
</tbody>
</table>

CHECK FOR LINE ENDING WITH AN INTERRUPTION

The third condition ending a line is an interrupt by a non-line drug with a qualifying refill. We begin by subsetting the dataset set of non-line drugs created in Step 5 to keep only those claims with DRUG_START day beyond the line initiation boundary (Line Start Day + Days to Establish a Line):

```
data line&ln._for_interrupt_use;
  merge line&ln._start_summary (keep = pid line&ln._start)
    line&ln._claims_for_interrupt_check (in = inclaims
      keep = pid drug drug_start drug_end);
  by pid;
  if inclaims and drug_start > (line&ln._start + &establish) then output;
run;
```

The DATA Step below searches for all non-line drug claims with a qualifying refill using a coding strategy similar to that used in Step 3 to identify first refills. The control flag in this case is the RETAINED variable drug_done which is used to bypass processing of claims if a qualifying claim pair has already been identified. Note that this step only identifies potential interrupts. The actual end reason for the Line (gap, interrupt, line continues through end of study) will be determined when we combine all the information we have accumulated in STEP 8 which follows.

```
data line&ln._possible_interrupts;
  set line&ln._for_interrupt_use;
  by pid drug;
  length interrupt_drug $ &maxlen;
  retain drug_done
    last_claim_start
    last_claim_end;
  /* first claim for a non-line drug */
  if first.drug then do;
    drug_done = 0;
    last_claim_start = drug_start;
    last_claim_end = drug_end;
  end;
  else do;
    if drug_done = 0 then do;
      /* refilled in time */
      if drug_start le last_claim_end then do;
        interrupt_drug = drug;
        interrupt_start = last_claim_start;
        drug_done = 1;
        output;
      end;
      /* not filled in time, recycle */
      else do;
        last_claim_end = drug_end;
        last_claim_start = drug_start;
      end;
    end;
  end;
run;
```
/* select the potential interrupt drug claim with the earliest drug_date */
proc sort data = here.line&ln._possible_interrupts
   out = line&ln._possible_interrupts;
   by pid interrupt_start;
run;

data line&ln._interrupt;
   set line&ln._possible_interrupts;
   by pid;
   if first.pid then output;
run;

data line&ln._interrupt_summary;
   merge line&ln._start_summary (keep = pid line&ln._regimen line&ln._start)
       line&ln._interrupt;
   by pid;
run;

For our sample, only p3 has a non-line drug with a claim positioned to be a possible line interrupt:

<table>
<thead>
<tr>
<th>pid</th>
<th>line1_start</th>
<th>interrupt_drug</th>
<th>interrupt_start</th>
</tr>
</thead>
<tbody>
<tr>
<td>P3</td>
<td>5</td>
<td>Chemo2</td>
<td>63</td>
</tr>
</tbody>
</table>

**STEP 7: FINALIZE AND ASSIGN REGIME AND LINE OUTCOME**

We have now reached the point in the program where all needed information is in summary datasets which can be merged to flag final outcomes. We include a master list of all study patients not previously referenced in this program to cover the case of patients in the study but with no claims qualifying for the Line being processed. This can occur for Line 1 when no drugs have a qualifying refill to establish the Line or when searching for multiple Lines which are not present for all patients.

/* combine line start, gap, and possible interrupt results with master list of study patients */
data line&ln._outcomes;
   merge lib.study_patients /* master list of all study patients */
       line&ln._start_summary /* line initiation */
       line&ln._gap_summary /* gap information */
       line&ln._interrupt_summary /* possible interrupts */
   by pid;
length line&ln._outcome $ 50;

/* study patient with no drugs for this line */
if line&ln._regimen = '' then line&ln._outcome = 'no line';

For patients with the Line established, the next step is to determine the Line End Date and to capture the Line End Reason. An interrupt takes precedence because it signals the existence of a subsequent line. An interrupted line ends on the day prior to the DRUG_START day of the interrupting drug. If the Line was not interrupted, it ends either with a gap or continues through the end of the patient’s study period and the Line End dates are assigned:

/* patient has line */
else do;
   /* interrupt take precedence because it will start next line */
   if interrupt_drug ne '' then do;
      line&ln._end = interrupt_start - 1;
      line&ln._outcome = 'line ends, interrupt';
   end;
   else if gap_drug ne '' then do;
      line&ln._end = gap_start - 1;
      line&ln._outcome = 'line ends, gap';
   end;
   else do;
      line&ln._end = study_end;
      line&ln._outcome = 'last line';
   end;
end; end; run;
Below are the basic summary outcome variables for Line 1 for our three test patients. The Line outcomes dataset also contains information on the gap drugs and dates and interrupt drugs and dates.

<table>
<thead>
<tr>
<th>pid</th>
<th>line1_regimen</th>
<th>line1_outcome</th>
<th>line1_start</th>
<th>line1_end</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Chemo6</td>
<td>last line</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td>Chemo2-Chemo4-Chemo6</td>
<td>line ends, gap</td>
<td>1</td>
<td>70</td>
</tr>
<tr>
<td>P3</td>
<td>Bio3-Chemo4</td>
<td>line ends, interrupt</td>
<td>5</td>
<td>62</td>
</tr>
</tbody>
</table>

**STEP 8: PROCESSING ADDITIONAL LINES**

To continue using the code above to process the original enhanced claims dataset for additional Lines of therapy, we only need to change the values of two global macro variables and use the Line_Outcomes summary dataset just created to subset the claims to a new target period. To process Line 2, for example, we would first:

- Change the value of the global macro variable &LN:
  \%let LN = 2;
- Assign the number of the previous Line to the global macro &PREVLN:
  \%let PREVLN = 1;

The new claims dataset is created by merging the current Line End Day and Line Outcome text with the original enhanced claims and selecting only claims for patients who have the potential for additional Lines. The INDEX function is used to test the line_outcome variable for the presence of the text string “line ends.” Based on the coding rules for this variable, “line ends” indicates a Line ending either in a gap or an interrupt and therefore the possibility of additional lines.

```sas
data claims;
  /* capture previous line end day and outcome */
  merge line&ln._outcomes (keep = pid line&ln._end line&ln._outcome)
  /* original enhanced claims dataset */
  claims
  ;
  by pid;
  /* patients with a line ending in a gap or an interrupt */
  if index(line&ln._outcome, 'line ends') ne 0 then do;
    /* only those claims on days subsequent to end day of previous line */
    if drug_start gt line&ln._end then output;
  end;
run;
```

**STEP 9: QA WORKBOOK FOR A RANDOM SAMPLE OF STUDY PATIENTS**

We can now use the temporary datasets created during the program run to produce a QA workbook with multiple sheets containing claims and outcomes. For this example we will select a random sample of two patients and print the enhanced claims and the data sets associated with the gap and interrupt calculations in STEP 6.

**SELECTING THE RANDOM SAMPLE**

We select the random sample using the patient list from the Line Outcomes dataset and the RANUNI function. After each PID is assigned random number, the resulting dataset is sorted by that number and the desired number of sample patients selected from the sort:

```sas
/* seed RANUNI function and set sample size desired with global macro variables*/
%let seed          = 1239;
%let randsize      = 2;
/* assign random number to each patient and sort by rand number */
data random;
  set here.line&ln._outcomes;
```
rand = ranuni(&seed);
run;
proc sort data = random;
  by rand;
run;
/* select sample */
data sample;
  set random (obs=&randsize);
run;

CREATING A FORMATTED LIST OF THE SAMPLE IDs

To avoid hard coding QA patient lists, we next dynamically put all sample PIDs into one text string (pat_list) formatted specifically to be used with a selecting IN clause in a later step. We use concatenation and the TRIM function to enclose each PID in single quotation marks and separate the PIDs with a comma. For this example, if PIDs P1 and P3 are selected as the random sample, after the last record in input dataset is processed, pat_list would contain the string 'P1', 'P3':

data _null_
  set sample end = eof; /* EOF = 1 on the last record in the input dataset */
  retain id_list;
  length id_list $ 200;
  /* add first PID to id list string */
  if _n_ = 1 then id_list = "'||pid||'",";
  /* PIDs except first and last, remove trailing blanks from id_list and add PID,*/
  else if not eof then id_list = trim(id_list)|| "'" || pid || "'",";
  /* last record */
  if eof then do;
    * finalize id list */
    id_list = trim(id_list)|| "'" || pid || "'";
    /* put list into global macro variable */
    call symput("plist",id_list);
  end;
run;

Note that we are using DATA _NULL_ rather than outputting the final string into a SAS dataset. This allows us to use the strategy introduced in STEP 3 to CALL SYMPUT and pass the formatted list of PIDs to the global macro variable &PLIST once the formatted string is complete.

SAS® ODS AND THE QA WORKBOOK

The final step is to use the SAS Output Delivery System with the tagsets.excelxp output destination to create the QA Workbook. The syntax for this output destination allows us to create multiple sheets within a single Excel workbook. After closing the listing output destination, we first name the output Excel Workbook and use the sheet_interval option to instruct SAS® that a new worksheet is required with each listed PROC:

ods listing close;
ods tagsets.excelxp file = "line&ln._qa.xls" /* output Excel workbook name */
  style = sansprinter
  options (embedded_titles = 'yes' sheet_interval = 'proc');

Next, we repeat a portion of code multiple times for each dataset we wish to print as a separate spreadsheet within the Excel file line&ln._qa.xls. We use the global macro variable &PLIST to subset records for the sample IDs.

/* claims data */
aids tagsets.excelxp options (sheet_name = "claims");
CONCLUSIONS
The focus of this paper is to provide the SAS® programmers with practical guidance on using the SAS® DATA Step and BASE SAS® procedures and functions in a data-driven approach to identifying Lines of Therapy in administrative claims data. Global macro variables are created and used to increase flexibility and ease of re-use for the program.

ACKNOWLEDGMENTS
SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

CONTACT INFORMATION
Laurie Hamilton
laurie.hamilton@lshamilton.com
open, go to View and click new window as many times as additional sheets in workbook, then hit arrange all, tiled.