Paper 3560-2015

A SAS Macro to Calculate the PDC Adjustment of Inpatient Stays

Anping Chang, IHRC INC; Cathleen Gillespie, Epidemiology & Surveillance Branch of Division for Heart Disease and Stroke Prevention, CDC

ABSTRACT
The Center for Medicare & Medicaid Services (CMS) uses the Proportion of Days Covered (PDC) to measure medication adherence among their beneficiaries. There is also some PDC-related research based on Medicare Part D Event (PDE) Data. However, “Under Medicare rules, beneficiaries who receive care at an Inpatient (IP) may receive Medicare covered medications directly from the IP, rather than by filling prescriptions through their Part D contractors; thus, their medication fills during an IP stay would not be included in the PDE claims used to calculate the Patient Safety adherence measures.” (Medicare 2014 Part C&D star rating technical notes). Therefore, the previous PDC calculation method underestimated the true values. Beginning with the 2013 Star rating, PDC calculations are adjusted for IP stays, by censoring IP stays during the measurement period. Further, If the patient also has measured drug coverage during the IP stay, the number of days a drug was supplied during that stay will be shifted to the post IP time period. This shift also causes a chain of shifting for subsequent medication fills. This paper presents a SAS Macro utilizing the SAS Hash Object to match IP stays, censoring the IP stays, shifting ending dates, and calculating the adjusted PDC.

INTRODUCTION
The CMS uses the proportion of days covered (PDC) to measure the Medicare patients’ medication adherence. There is also some PDC-related research based on Medicare Part D Event (PDE) Data (Lihua Zhang et al 2011, Simoni-Wastila L. et al 2012). The PDC calculation is based on the date of medication prescription fills and the days of supply for each prescription fill in a defined interval, for example a calendar year. There are existing methods to calculate the PDC rate (R. Scott Leslie 2007, Li-Hao Chu et al 2011. Stacy Wang et al 2013), but none of these methods adjusted for inpatient (IP) stays. This paper presents a SAS macro program to calculate the PDC for single medication use adjusted for IP stay days. It utilizes the SAS lag function, hash object, and do loop operation. It contains three parts: data preparation; dates adjustment; and PDC calculation.

1: PREPARING THE PDE DATA
The PDC calculation for the Medicare population is based on the Part D Event data (PDE). The PDE data set contains variables like service date, number of days' supply, and medication name. When calculating the covered days, we need to know the service start date and end date of a prescription fill and the next fill start date. However, the PDE data does not contain the end service date and next fill start date. The end date can be created by service start date of the fill plus the number of days' supply. When the service date plus the number of days' supply is greater than the next fill service date, the end date will be adjusted to the next fill date minus one. The leftover medication will be carried over to the next fill. This process repeats for each fill, but the next fill service date is not available in the same observation since one observation only represents one prescription fill. Fortunately, by utilizing the SAS PROC SORT and LAG functions, we can easily create the next fill date in the current prescription fill observation. Tables 1 and table 2 show examples of a patient’s original PDE data and the processed PDE data, respectively. The following code creates the Next_fill_dt and rx_end_dt variables:

Proc sort data=&rxDsn out= rx_sorted nodupkey;
    by &pat_id &rx_class DESCENDING &rx_fill_dt ;
RUN;

Data RX_dates;
    set rx_sorted;
by &pat_id &rx_class;
next_srvc_dt = lag(&Rx_fill_dt);
if first.&rx_class then next_srvc_dt = .;
format next_srvc_dt date9.;
RUN;

Proc sort data=RX_dates ;
by &pat_id &rx_class &rx_fill_dt ;
RUN;

/* Create service ending date of each prescription fill */
Data RX_dates2;
set RX_dates;
format rx_end_dt date9.;
retain Rx_leftover ;
by &pat_id &rx_class &rx_fill_dt ;

if first.&rx_class then do;
if &rx_fill_dt + days_suply_num < next_srvc_dt then do;
rx_end_dt = &rx_fill_dt + days_suply_num;
rx_leftover = 0;
end; else do;
rx_end_dt = next_srvc_dt -1;
rx_leftover = (&rx_fill_dt + days_suply_num) - next_srvc_dt + 1;
end;
end; else if last.&rx_class then do;
rxd End_dt = &rx_fill_dt + rx_leftover + days_suply_num;
r_x_leftover = 0;
end; else do;
if &rx_fill_dt + rx_leftover + days_suply_num <next_srvc_dt then do;
rx_end_dt = &rx_fill_dt + rx_leftover + days_suply_num;
rx_leftover = 0;
end; else do;
rx_end_dt = next_srvc_dt -1;
x_leftover =(&rx_fill_dt + rx_leftover + days_suply_num)-next_srvc_dt+ 1;
end;
end;
if first.&rx_class and last.&rx_class then rx_end_dt = &rx_fill_dt + + days_suply_num;
if next_srvc_dt = . then next_srvc_dt = rx_end_dt + 1;
RUN;

<table>
<thead>
<tr>
<th>Pat_id</th>
<th>Srvc_dt</th>
<th>Medication_name</th>
<th>Num_days_supply</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>02/13/2012</td>
<td>Lovastatin</td>
<td>30</td>
</tr>
<tr>
<td>001</td>
<td>03/16/2012</td>
<td>Lovastatin</td>
<td>30</td>
</tr>
<tr>
<td>001</td>
<td>04/18/2012</td>
<td>Lovastatin</td>
<td>90</td>
</tr>
<tr>
<td>001</td>
<td>07/17/012</td>
<td>Lovastatin</td>
<td>90</td>
</tr>
<tr>
<td>001</td>
<td>11/12/2012</td>
<td>Lovastatin</td>
<td>90</td>
</tr>
</tbody>
</table>

Table 1. Original PDE data

<table>
<thead>
<tr>
<th>Pat_id</th>
<th>Srvc_dt</th>
<th>Medication_name</th>
<th>Num_days_supply</th>
<th>Next_fill_dt</th>
<th>Rx_end_dt</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>02/13/2012</td>
<td>Lovastatin</td>
<td>30</td>
<td>03/16/2012</td>
<td>03/14/2012</td>
</tr>
<tr>
<td>001</td>
<td>03/16/2012</td>
<td>Lovastatin</td>
<td>30</td>
<td>04/18/2012</td>
<td>04/15/2012</td>
</tr>
<tr>
<td>001</td>
<td>04/18/2012</td>
<td>Lovastatin</td>
<td>90</td>
<td>07/17/012</td>
<td>07/16/2012</td>
</tr>
<tr>
<td>001</td>
<td>07/17/012</td>
<td>Lovastatin</td>
<td>90</td>
<td>11/12/2012</td>
<td>10/16/2012</td>
</tr>
<tr>
<td>001</td>
<td>11/12/2012</td>
<td>Lovastatin</td>
<td>90</td>
<td>.</td>
<td>02/10/2013</td>
</tr>
</tbody>
</table>

Table 2. After process
2: ADJUST THE INPATIENT STAY DAYS

The principle of the adjustment is that when a patient has an IP stay during the prescription fill period, the stay days will be censored and the days of medication supply during the IP stay will be carried over to the post-IP stay period or the next fill. If the patient has multiple periods of IP stays, then the process will be repeated. Tables 3 and 4 are the simplified examples. This is a 16 day period with four IP days and four uncovered days. The medications in day 2, 3, 6, 7 will be moved to day 11, 12, 13 and 16, to account for the patient's IP stay of four days. Before the adjustment, the PDC is defined as (12/16)*100% = 75%; after the adjustment, the PDC is defined as (12/12)*100% = 100%.

<table>
<thead>
<tr>
<th>days</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drug covered</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Inpatient</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
</tbody>
</table>

Table 3. Before adjustment

<table>
<thead>
<tr>
<th>Days</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drug covered</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Inpatient</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

Table 4. After adjustment

Since PDE data and Inpatient data both have multiple records for each patient, it is difficult to use merge or SQL operations to process the adjustment. Fortunately, SAS provides Hash Object, and by combining the methods of Hash Object and do-loop operation, the process is simplified and more efficient. The following code processes the adjustment:

```sas
Data adj_rx_date;
length &pat_id. &inpt_begin_dt. &inpt_end_dt. &adj_rx_end_dt. 8.;
format &inpt_begin_dt. &inpt_end_dt. &adj_rx_end_dt. date9. ;
if _N_ = 1 then do; /*Define the Hash table and put the inpatient stay dataset */
   dcl hash inpt(dataset: "&inptDSN", multidata: 'Y', hashexp: 20, ordered: 'a');
   inpt.defineKey("&pat_id");
   inpt.definedata("&inpt_begin_dt", "&inpt_end_dt");
   inpt.definedone();
   Call missing(&pat_id, &inpt_begin_dt, &inpt_end_dt);
End;
set RX_dates2;
by &pat_id &rx_class &Rx_fill_dt;
retain numRx_leftover_4inpt censored_days;
if first.&pat_id then do;
   numRx_leftover_4inpt = 0;
   censored_days = 0;
end;
rc = inpt.find();
If rc NE 0 then do;
   output;
   goto exist;
End; else do;
if (&rx_fill_dt<&inpt_begin_dt<rx_end_dt) and (&rx_fill_dt<&inpt_end_dt < rx_end_dt) then do;
   numRx_leftover_4inpt = &inpt_end_dt - &inpt_begin_dt;
   censored_days = &inpt_end_dt - &inpt_begin_dt;
   if (rx_end_dt + numRx_leftover_4inpt) < next_srvc_dt then do;
      adj_rx_end_dt = rx_end_dt + numRx_leftover_4inpt;
      numRx_leftover_4inpt = 0;
   end;
end; else do;
```

3
numRx_leftover_4inpt = (rx_end_dt + numRx_leftover_4inpt) - next_srvc_dt;
adj_rx_end_dt = next_srvc_dt - 1;
end;
end;

if &rx_fill_dt <&inpt_begin_dt < rx_end_dt and &inpt_end_dt >= rx_end_dt then do;
censored_days = rx_end_dt - &inpt_begin_dt;
numRx_leftover_4inpt = rx_end_dt - &inpt_begin_dt;
if &inpt_end_dt < next_srvc_dt then do;
  adj_rx_end_dt = &inpt_end_dt + numRx_leftover_4inpt;
  numRx_leftover_4inpt = 0;
end;else do;
  numRx_leftover_4inpt = numRx_leftover_4inpt - (next_srvc_dt - &inpt_end_dt);
adj_rx_end_dt = next_srvc_dt - 1;
end;
end; else do;
  numRx_leftover_4inpt = numRx_leftover_4inpt;
  adj_rx_end_dt = next_srvc_dt - 1;
end;
end;

drop r rc rnext;
exist: ;
RUN;

Tables 5 and 6 show a real example of PDE data and Inpatient data, and table 7 shows the result after the adjustment.

<table>
<thead>
<tr>
<th>Pat_Id</th>
<th>Srvc_dt</th>
<th>Num_days_supply</th>
<th>Medication_name</th>
<th>Next_srvc_dt</th>
<th>Rx_end_dt</th>
</tr>
</thead>
<tbody>
<tr>
<td>002</td>
<td>01/18/2012</td>
<td>30</td>
<td>STORVASTATIN</td>
<td>04/24/2012</td>
<td>02/17/2012</td>
</tr>
</tbody>
</table>
Table 5. Medication Dates before Adjusted

<table>
<thead>
<tr>
<th>Pat_Id</th>
<th>Inpt_begin_dt</th>
<th>Inpt_end_dt</th>
<th>Medication_name</th>
<th>Next_rx_end_dt</th>
<th>Adj_rx_end_dt</th>
<th>Rx_end_dt</th>
<th>Censored_days</th>
</tr>
</thead>
<tbody>
<tr>
<td>002</td>
<td>04/24/2012</td>
<td>06/23/2012</td>
<td>STORVASTATIN</td>
<td>06/22/2012</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>002</td>
<td>05/23/2012</td>
<td>08/29/2012</td>
<td>STORVASTATIN</td>
<td>08/23/2012</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>002</td>
<td>08/29/2012</td>
<td>10/30/2012</td>
<td>STORVASTATIN</td>
<td>10/28/2012</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>002</td>
<td>10/30/2012</td>
<td>12/30/2012</td>
<td>STORVASTATIN</td>
<td>12/29/2012</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>003</td>
<td>01/06/2012</td>
<td>04/17/2012</td>
<td>ROSUVASTATIN</td>
<td>04/05/2012</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>003</td>
<td>04/17/2012</td>
<td>07/15/2012</td>
<td>ROSUVASTATIN</td>
<td>07/14/2012</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>003</td>
<td>07/15/2012</td>
<td>10/15/2012</td>
<td>ROSUVASTATIN</td>
<td>10/14/2012</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>003</td>
<td>10/15/2012</td>
<td>01/15/2013</td>
<td>ROSUVASTATIN</td>
<td>01/14/2013</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6. Inpatient Stay dates

<table>
<thead>
<tr>
<th>Pat_Id</th>
<th>Srvc_dt</th>
<th>Num_days_supply</th>
<th>Medication_name</th>
<th>Next_svc_dt</th>
<th>Rx_end_dt</th>
<th>Adj_rx_end_dt</th>
<th>Censored_days</th>
</tr>
</thead>
<tbody>
<tr>
<td>002</td>
<td>01/18/2012</td>
<td>30</td>
<td>STORVASTATIN</td>
<td>04/24/2012</td>
<td>02/17/2012</td>
<td>02/17/2012</td>
<td>0</td>
</tr>
<tr>
<td>002</td>
<td>04/24/2012</td>
<td>60</td>
<td>STORVASTATIN</td>
<td>06/23/2012</td>
<td>06/22/2012</td>
<td>06/22/2012</td>
<td>0</td>
</tr>
<tr>
<td>002</td>
<td>06/23/2012</td>
<td>60</td>
<td>STORVASTATIN</td>
<td>08/29/2012</td>
<td>08/23/2012</td>
<td>08/23/2012</td>
<td>8</td>
</tr>
<tr>
<td>002</td>
<td>08/29/2012</td>
<td>60</td>
<td>STORVASTATIN</td>
<td>10/30/2012</td>
<td>10/28/2012</td>
<td>10/28/2012</td>
<td>8</td>
</tr>
<tr>
<td>002</td>
<td>10/30/2012</td>
<td>60</td>
<td>STORVASTATIN</td>
<td>12/30/2012</td>
<td>12/29/2012</td>
<td>12/29/2012</td>
<td>16</td>
</tr>
<tr>
<td>003</td>
<td>01/06/2012</td>
<td>90</td>
<td>ROSUVASTATIN</td>
<td>04/17/2012</td>
<td>04/05/2012</td>
<td>04/05/2012</td>
<td>0</td>
</tr>
<tr>
<td>003</td>
<td>04/17/2012</td>
<td>90</td>
<td>ROSUVASTATIN</td>
<td>07/15/2012</td>
<td>07/14/2012</td>
<td>07/14/2012</td>
<td>0</td>
</tr>
<tr>
<td>003</td>
<td>07/15/2012</td>
<td>90</td>
<td>ROSUVASTATIN</td>
<td>10/15/2012</td>
<td>10/14/2012</td>
<td>10/14/2012</td>
<td>9</td>
</tr>
<tr>
<td>003</td>
<td>10/15/2012</td>
<td>92</td>
<td>ROSUVASTATIN</td>
<td>01/15/2013</td>
<td>01/14/2013</td>
<td>01/14/2013</td>
<td>9</td>
</tr>
</tbody>
</table>

Table 7. After Adjustment

3: CALCULATE PDC

After the adjustment, the PDC calculation becomes easy. The following code calculates the numerator and denominator for the PDC:

```sql
/* create total inpatient stay days */
Data Inpt_days;
  set adj_rx_date ;
  by &pat_id &rx_fill_dt &rx_class ;
  if last.&pat_id ;
RUN;

/* create total medication covered days */
Data Rx_covered_days;
  set adj_rx_date ;
  if adj_rx_end_dt . then adj_rx_end_dt = rx_end_dt ;
  rx_covered_days = min(adj_rx_end_dt, "&Study_end_dt"d) - &rx_fill_dt ;
RUN;

Proc sql;
  create table tot_Rx_covered_days
    as select distinct &pat_id , &rx_class
```

5
, min(&rx_fill_dt) as period_start format = date9.
, max(adj_rx_end_dt) as period_end format = date9.
, sum(rx_covered_days) as tot_rx_covered_days
from rx_covered_days
  group by &pat_id, &rx_class;

cREATE TABLE pre_pdc
  AS SELECT a.* , b.censored_days
  FROM tot_Rx_covered_days AS a LEFT JOIN inpt_days AS b
  ON a.&pat_id = b.&pat_id;
QUIT;

/* Calculate the pdc */
DATA &outdsn;
  SET pre_pdc;
  denominator = "&Study_end_dt" - period_start - censored_days;
  IF denominator = 0 THEN PDC = 0;
  ELSE PDC = tot_rx_covered_days/denominator;
RUN;

The whole Macro program is in the appendix.

CONCLUSION

This SAS Macro program provides an efficient tool to calculate the PDC for the Medicare population. By adjusting for IP stays, the PDC estimate is closer to the true PDC value. So far, this macro only works for single drug use. The macro for multiple drug use will be available in the near future.

REFERENCES


RECOMMENDED READING

- *SAS® Hash Programming*
- *Base SAS® Procedures Guide*

CONTACT INFORMATION

Anping Chang
APPENDIX

%MACRO inpt_PDC_Calculate(rxDSN=, pat_id =, Rx_fill_dt =, rx_class =, inptDSN =, inpt_begin_dt =, inpt_end_dt =, outdsn =, Study_end_dt =);
/* Preparing the Rx data, create next fill date variable by using Lag function and proc sort*/
Proc sort data=&rxDSN out=rx_sorted nodupkey;
   by &pat_id &rx_class DESCENDING &Rx_fill_dt ;
RUN;
Data RX_dates;
   set rx_sorted;
   by &pat_id &rx_class;
   next_srvc_dt = lag(&Rx_fill_dt);
   if first.&rx_class then next_srvc_dt = .;
   format next_srvc_dt date9. ;
RUN;
Proc sort data=RX_dates ;
   by &pat_id &rx_class &rx_fill_dt ;
RUN;
/* Create service ending date of each prescription fill */
Data RX_dates2;
   set RX_dates;
   format rx_end_dt date9. ;
   retain Rx_leftover ;
   by &pat_id &rx_class &rx_fill_dt ;
   If first.&rx_class then do;
      if &rx_fill_dt + days_suply_num < next_srvc_dt then do;
         rx_end_dt = &rx_fill_dt + days_suply_num;
         rx_leftover = 0;
      end; else do;
         rx_end_dt = next_srvc_dt -1; 
         rx_leftover = (&rx_fill_dt + days_suply_num) - next_srvc_dt + 1;
      end;
   end; else if last.&rx_class then do;
      rx_end_dt = &rx_fill_dt + rx_leftover + days_suply_num;
      rx_leftover = 0;
   end; else do;
      if &rx_fill_dt + rx_leftover + days_suply_num <next_srvc_dt then do;
         rx_end_dt = &rx_fill_dt + rx_leftover + days_suply_num;
         rx_leftover = 0;
      end; else do;
         rx_end_dt = next_srvc_dt -1;
         rx_leftover = (&rx_fill_dt + rx_leftover + days_suply_num) - next_srvc_dt + 1 ;
      end;
   end;
   if first.&rx_class and last.&rx_class then rx_end_dt = &rx_fill_dt + + days_suply_num;
   if next_srvc_dt = . then next_srvc_dt = rx_end_dt + 1;
RUN;
Proc sort data=&InptDSN out=inpt_stay;
  by &pat_id &inpt_begin_dt;
RUN;
/* Adjust the inpatient stay days */
Data adj_rx_date;
  length &pat_id. &inpt_begin_dt. &inpt_end_dt. 8.;
  format &inpt_begin_dt. &inpt_end_dt. adj_rx_end_dt. date9. ;
  if _N_ = 1 then do; /* Define the Hash table and put the inpatient stay dataset */
    do file hash inpt(dataset: "&InptDSN", multidata: 'Y', hashexp: 20, ordered: 'A');
      inpt.defineKey("&pat_id");
      inpt.definedata("&inpt_begin_dt", ", &inpt_end_dt");
      inpt.definedone();
      Call missing(&pat_id, &inpt_begin_dt, &inpt_end_dt);
    End;
  set RX_dates2; by &pat_id &rx_class &Rx_fill_dt; retain numRx_leftover_4inpt censored_days;
  if first.&pat_id then do;
    numRx_leftover_4inpt = 0;
    censored_days = 0;
  end;
  rc = inpt.find();
  If rc NE 0 then do;
    output;
    goto exist;
  end; else do;
    if (&rx_fill_dt <&inpt_begin_dt < rx_end_dt) and (&rx_fill_dt <&inpt_end_dt < rx_end_dt) then do;
      numRx_leftover_4inpt = &inpt_end_dt - &inpt_begin_dt;
      censored_days = &inpt_end_dt - &inpt_begin_dt;
      if (rx_end_dt + numRx_leftover_4inpt) < next_srvc_dt then do;
        adj_rx_end_dt = rx_end_dt + numRx_leftover_4inpt;
        numRx_leftover_4inpt = 0;
      end; else do;
        numRx_leftover_4inpt = (rx_end_dt + numRx_leftover_4inpt) - next_srvc_dt;
        adj_rx_end_dt = next_srvc_dt - 1;
      end;
    end;
  end;
  if &rx_fill_dt <&inpt_begin_dt <&inpt_end_dt >= rx_end_dt then do;
    censored_days = rx_end_dt - &inpt_begin_dt - &inpt_end_dt;
    if &inpt_end_dt < next_srvc_dt then do;
      if numRx_leftover_4inpt < next_srvc_dt - &inpt_end_dt then do;
        adj_rx_end_dt = &inpt_end_dt + numRx_leftover_4inpt;
        numRx_leftover_4inpt = 0;
      end; else do;
        numRx_leftover_4inpt = numRx_leftover_4inpt - (next_srvc_dt - &inpt_end_dt);
        adj_rx_end_dt = next_srvc_dt - 1;
      end;
    end;
  end; else do;
    numRx_leftover_4inpt = numRx_leftover_4inpt;
    adj_rx_end_dt = next_srvc_dt - 1;
  end;
end;
next = inpt.find_next();
if next NE 0 then output;
do while(next = 0);
  if (&rx_fill_dt <&inpt_begin_dt < rx_end_dt) and (&rx_fill_dt <&inpt_end_dt < rx_end_dt) then do;
    numRx_leftover_4inpt = &inpt_end_dt - &inpt_begin_dt;
    censored_days = &inpt_end_dt - &inpt_begin_dt - censored_days;
    if (rx_end_dt + numRx_leftover_4inpt) < next_srvc_dt then do;
      adj rx_end_dt = rx_end_dt + numRx_leftover_4inpt;
      numRx_leftover_4inpt = 0;
    end; else do;
      numRx_leftover_4inpt = (rx_end_dt + numRx_leftover_4inpt) - next_srvc_dt;
      adj_rx_end_dt = next_srvc_dt - 1;
  end;
  next = inpt.find_next();
if next NE 0 then output;
do while(next = 0);

end;
if &rx_fill_dt <&inpt_begin_dt < rx_end_dt and &inpt_end_dt >= rx_end_dt then do;
censored_days = rx_end_dt - &inpt_begin_dt + censored_days ;
numRx_leftover_4inpt = rx_end_dt - &inpt_begin_dt;  
if &inpt_end_dt < next_srvc_dt then do;
  if numRx_leftover_4inpt < next_srvc_dt - &inpt_end_dt then do;
    adj_rx_end_dt = &inpt_end_dt + numRx_leftover_4inpt;
  numRx_leftover_4inpt = 0 ;
end;else do;
  numRx_leftover_4inpt = numRx_leftover_4inpt - (next_srvc_dt - &inpt_end_dt);
  adj_rx_end_dt = next_srvc_dt -1;
end;
end;
end; else do;
numRx_leftover_4inpt = numRx_leftover_4inpt;
adj_rx_end_dt = next_srvc_dt -1;
end;
end;
end;
inpt.has_next(result: r) ;
if r = 0 then output ;
rnext = inpt.find_next();
end;
drop r rc rnext;
exist: ;
RUN;

/* create total inpatient stay days */
Data Inpt_days;
  set adj_rx_date ;
  by &pat_id &rx_fill_dt &rx_class;
  if last.&pat_id ;
RUN;

/* create total medication covered days */
Data Rx_covered_days;
  set adj_rx_date;
  if adj_rx_end_dt = . then adj_rx_end_dt = rx_end_dt;
  rx_covered_days = min(adj_rx_end_dt, "&Study_end_dt"d) - &rx_fill_dt;
RUN;

Proc sql;
create table tot_Rx_covered_days
  as select distinct &pat_id, &rx_class
      , min(&rx_fill_dt) as period_start format = date9. 
      , max(adj_rx_end_dt) as period_end format = date9. 
      , sum(rx_covered_days)as tot_rx_covered_days 
from rx_covered_days
  group by &pat_id, &rx_class ;
create table pre_pdc
  as select a.*
      , b.censored_days
from tot_Rx_covered_days as a left join inpt_days as b
  on a.&pat_id = b.&pat_id;
Quit;

/* Calculate the pdc */
Data &outdsn;
  set pre_pdc;
  denominator = "&Study_end_dt"d - period_start - censored_days;
  if denominator = 0 then PDC = 0;
  else PDC = tot_rx_covered_days/denominator;
RUN;

%MEND;
 /* Macro call */
%inpt_PDC_Calculate(rxDSN = PED_STATIN 
  , pat_id = bene_id 
  , Rx_fill_dt = srvc_dt 
  , rx_class = medication_name
, inptDSN = inpatient
, inpt_begin_dt = inpt_begin_dt
, inpt_end_dt = inpt_end_dt
, outdsn = pdc_result
, Study_end_dt = 31Dec2012
);