TAP TO GO BACK TO KIOSK MENU

SAS® GLOBAL FORUM 2020

MARCH 29 - APRIL 1
WASHINGTON, DC

USERS PROGRAM

ACHI

CREATING EPISODES OF CARE & FINDING COVERAGE GAPS WITH ADMINISTRATIVE MEDICAL RECORDS

TAP TO GO BACK TO KIOSK MENU

Stephen D. Lein, MPH

Arkansas Center for Health Improvement (ACHI)

Please use the headings above to navigate through the different sections of the poster

Abstract

Analysis and research on administrative healthcare data often involve the manipulation of multiple records with contiguous or overlapping date fields. Claims data contain transaction records with multiple records for each individual across multiple dates of service, while enrollment data often contain multiple records per person reflecting that individual's enrollment change history. In many cases, it is necessary to roll up the date fields in these types of files in order to establish continuous enrollment, episodes of care, and the gaps in coverage for those periods.

In this presentation, a flexible SAS® program is introduced that rolls up multi-record date fields to establish continuous segments, segment lengths, and segment gap lengths, and to calculate the number of segments and gaps for an individual ID variable. Using SAS functions such as LAG, RETAIN, and INTCK, as well as procedures such as PROC SQL and PROC TRANSPOSE, the target data are transformed from long to wide. This enables the user to more easily process hospitalization data into episodes of care or evaluate claims and enrollment records against standardized performance measures such as the Healthcare Effectiveness Data and Information Set (HEDIS).

Stephen D. Lein

Stephen D. Lein, MPH
Arkansas Center for Health Improvement (ACHI)

Introduction

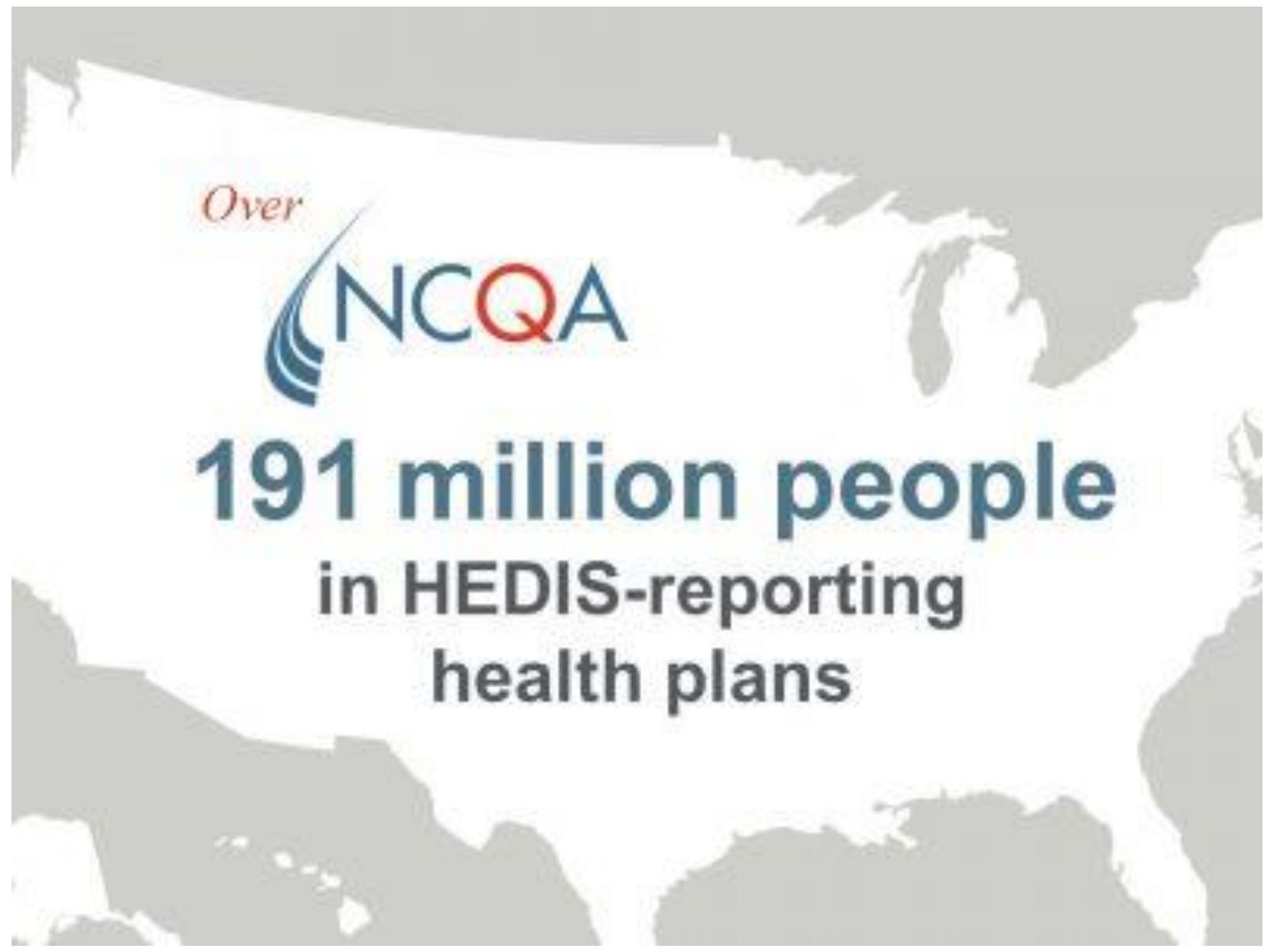
Please use the headings above to navigate through the different sections of the poster

Administrative medical claims data, such as Medicare, Medicaid, and commercial coverage plans like Blue Cross and Blue Shield are often used to study healthcare utilization and outcomes. However, these types of claims were designed to facilitate billing, not research. When processing this type of medical claims data, it is often necessary to roll up an individual's experience along dates of service or dates of enrollment for the purposes of checking for gaps between services or gaps between insurance coverage dates. One of the most common uses for this type of roll up is calculating HEDIS measures.

HEDIS measures are a comprehensive set of standardized performance measures designed to evaluate and compare health plan performance. Each measure will typically have allowable gap criteria, such as:

- "No more than one gap in enrollment of up to 45 days during each year of continuous enrollment."
- "The member may not have more than a 1-month gap in coverage during each year of continuous enrollment."
- "No gaps in enrollment during the continuous enrollment period."

The SAS program discussed here — although designed for use with this type of administrative medical enrollment and claims data — could easily be adapted to roll up any time series data containing paired begin and end dates.



ACHI CREATING EPISODES OF CARE & FINDING COVERAGE GAPS WITH ADMINISTRATIVE MEDICAL RECORDS

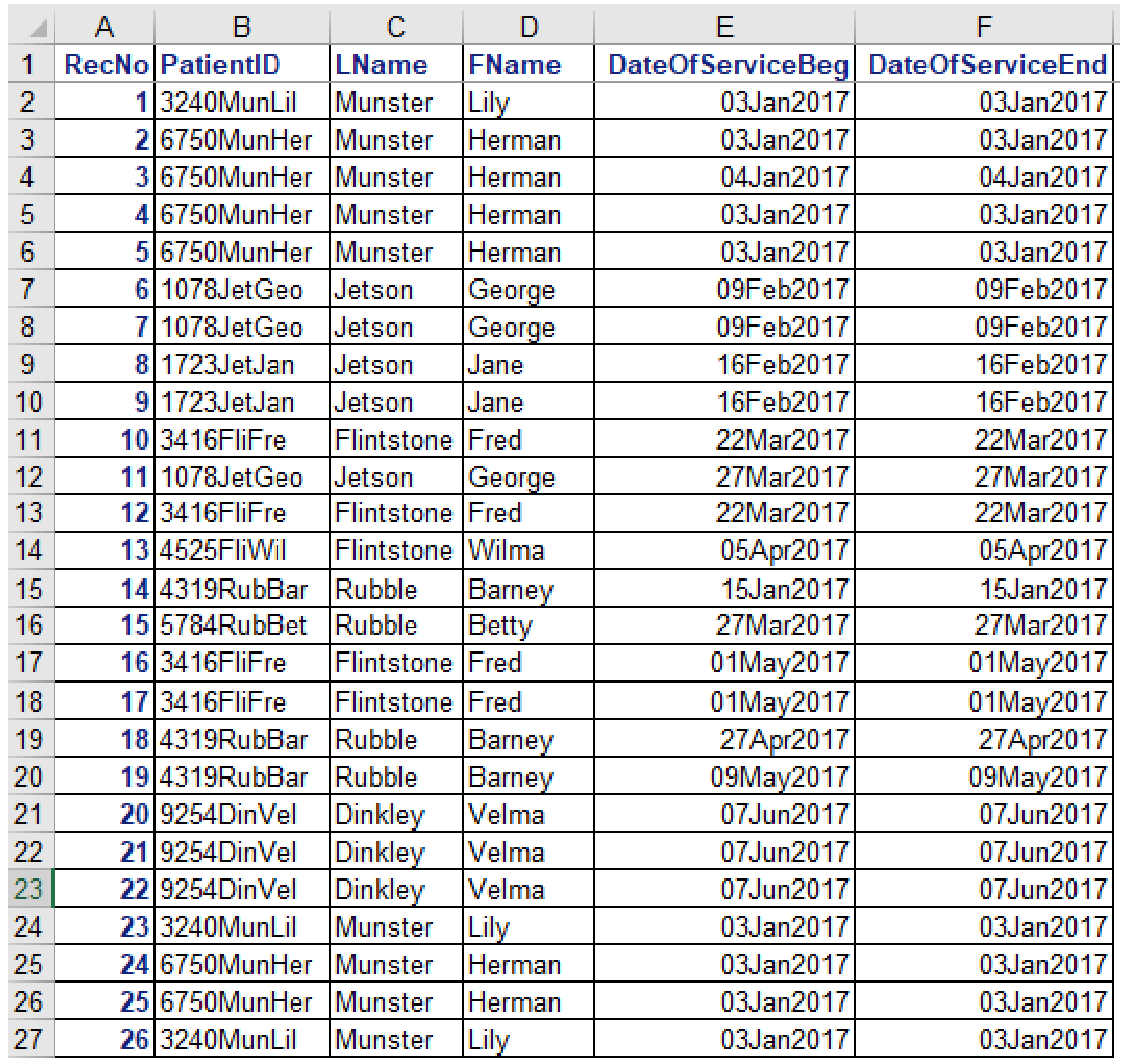
Stephen D. Lein, MPH

Arkansas Center for Health Improvement (ACHI)

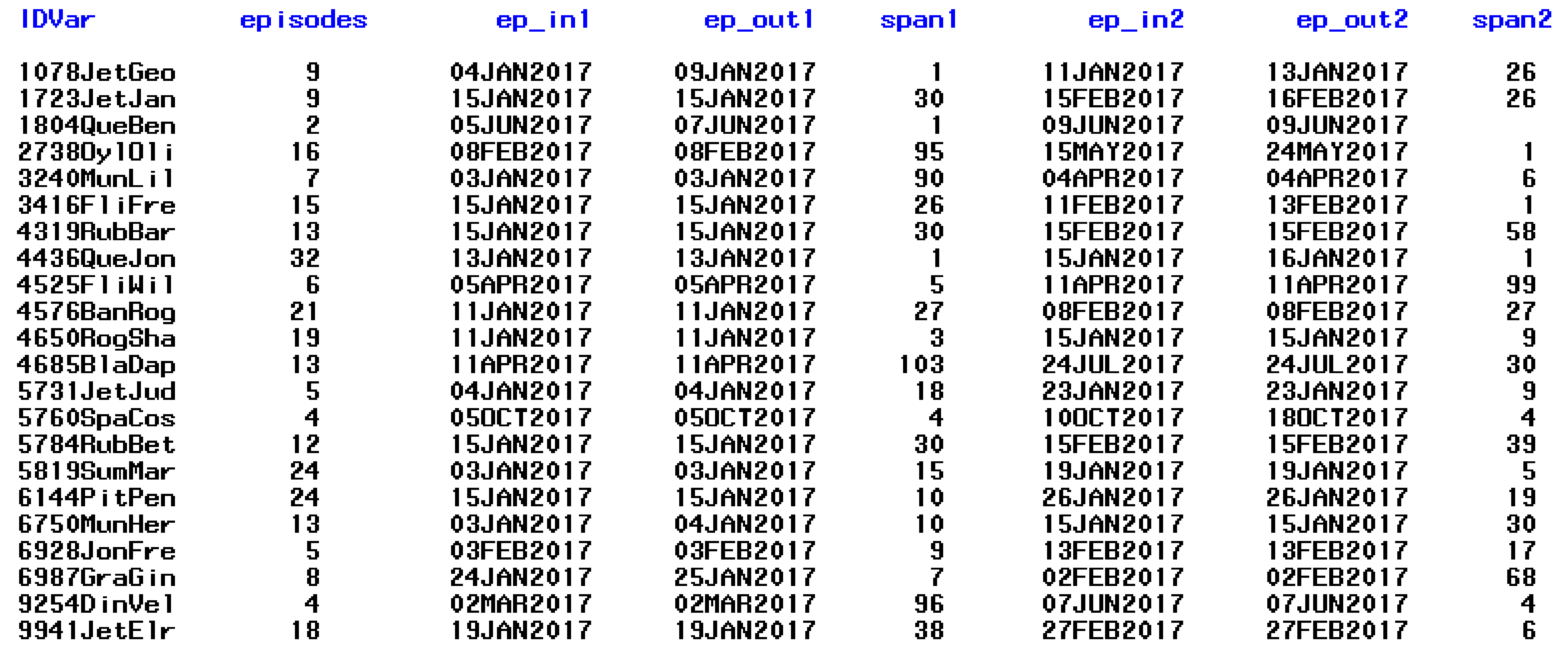
Please use the headings above to navigate through the different sections of the poster

Purpose

In this presentation, paired time series data will be transformed from long to wide, resulting in a single record per unique ID variable. The resulting table will contain the number of episodes, the begin and end dates for each episode and the spans (i.e., gaps) between episodes.



	A	B	C	D	E	F
1	RecNo	PatientID	LName	FName	DateOfServiceBeg	DateOfServiceEnd
2	1	3240MunLil	Munster	Lily	03Jan2017	03Jan2017
3	2	6750MunHer	Munster	Herman	03Jan2017	03Jan2017
4	3	6750MunHer	Munster	Herman	04Jan2017	04Jan2017
5	4	6750MunHer	Munster	Herman	03Jan2017	03Jan2017
6	5	6750MunHer	Munster	Herman	03Jan2017	03Jan2017
7	6	1078JetGeo	Jetson	George	09Feb2017	09Feb2017
8	7	1078JetGeo	Jetson	George	09Feb2017	09Feb2017
9	8	1723JetJan	Jetson	Jane	16Feb2017	16Feb2017
10	9	1723JetJan	Jetson	Jane	16Feb2017	16Feb2017
11	10	3416FliFre	Flintstone	Fred	22Mar2017	22Mar2017
12	11	1078JetGeo	Jetson	George	27Mar2017	27Mar2017
13	12	3416FliFre	Flintstone	Fred	22Mar2017	22Mar2017
14	13	4525FliWil	Flintstone	Wilma	05Apr2017	05Apr2017
15	14	4319RubBar	Rubble	Barney	15Jan2017	15Jan2017
16	15	5784RubBet	Rubble	Betty	27Mar2017	27Mar2017
17	16	3416FliFre	Flintstone	Fred	01May2017	01May2017
18	17	3416FliFre	Flintstone	Fred	01May2017	01May2017
19	18	4319RubBar	Rubble	Barney	27Apr2017	27Apr2017
20	19	4319RubBar	Rubble	Barney	09May2017	09May2017
21	20	9254DinVel	Dinkley	Velma	07Jun2017	07Jun2017
22	21	9254DinVel	Dinkley	Velma	07Jun2017	07Jun2017
23	22	9254DinVel	Dinkley	Velma	07Jun2017	07Jun2017
24	23	3240MunLil	Munster	Lily	03Jan2017	03Jan2017
25	24	6750MunHer	Munster	Herman	03Jan2017	03Jan2017
26	25	6750MunHer	Munster	Herman	03Jan2017	03Jan2017
27	26	3240MunLil	Munster	Lily	03Jan2017	03Jan2017



```
IDVar         episodes      ep_in1      ep_out1    span1      ep_in2      ep_out2    span2

1078JetGeo        9      04JAN2017   09JAN2017       1     11JAN2017   13JAN2017      26
1723JetJan        9      15JAN2017   15JAN2017      30     15FEB2017   16FEB2017      26
1804QueBen        2      05JUN2017   07JUN2017       1     09JUN2017   09JUN2017
2738OylOli       16      08FEB2017   08FEB2017      95     15MAY2017   24MAY2017       1
3240MunLil        7      03JAN2017   03JAN2017      90     04APR2017   04APR2017       6
3416FliFre       15      15JAN2017   15JAN2017      26     11FEB2017   13FEB2017       1
4319RubBar       13      15JAN2017   15JAN2017      30     15FEB2017   15FEB2017      58
4436QueJon       32      13JAN2017   13JAN2017       1     15JAN2017   16JAN2017       1
4525FliWil        6      05APR2017   05APR2017       5     11APR2017   11APR2017      99
4576BanRog       21      11JAN2017   11JAN2017      27     08FEB2017   08FEB2017      27
4650RogSha       19      11JAN2017   11JAN2017       3     15JAN2017   15JAN2017       9
4685BlaDap       13      11APR2017   11APR2017     103     24JUL2017   24JUL2017      30
5731JetJud        5      04JAN2017   04JAN2017      18     23JAN2017   23JAN2017       9
5760SpaCos        4      05OCT2017   05OCT2017       4     10OCT2017   18OCT2017       4
5784RubBet       12      15JAN2017   15JAN2017      30     15FEB2017   15FEB2017      39
5819SumMar       24      03JAN2017   03JAN2017      15     19JAN2017   19JAN2017       5
6144PitPen       24      15JAN2017   15JAN2017      10     26JAN2017   26JAN2017      19
6750MunHer       13      03JAN2017   04JAN2017      10     15JAN2017   15JAN2017      30
6928JonFre        5      03FEB2017   03FEB2017       9     13FEB2017   13FEB2017      17
6987GraGin        8      24JAN2017   25JAN2017       7     02FEB2017   02FEB2017      68
9254DinVel        4      02MAR2017   02MAR2017      96     07JUN2017   07JUN2017       4
9941JetElr       18      19JAN2017   19JAN2017      38     27FEB2017   27FEB2017       6
```

TERMS

- **Episode**: A series of contiguous or overlapping dates merged to form a single "episode." For example, 01Jan2017 – 10Jan2017, 11Jan2017 – 12Jan2017, 01Feb2017 – 05Feb2017 would become two episodes consisting of 01Jan2017 – 12Jan2017 and 01Feb2017 – 05Feb2017.
- **Span (Gap)**: The number of days between each episode. Continuing the above example, there would be 2 episodes and one span of 19 days. This would be the equivalent of a 19-day gap in medical insurance enrollment coverage or a 19-day gap between episodes of care.

CREATING EPISODES OF CARE & FINDING COVERAGE GAPS WITH ADMINISTRATIVE MEDICAL RECORDS

Stephen D. Lein, MPH
Arkansas Center for Health Improvement (ACHI)

Please use the headings above to navigate through the different sections of the poster

Code Walkthrough

PREPROCESSING

In order to apply this code, the input file must have at least one unique ID variable, a begin date variable and an end date variable. In the preprocessing step, missing/blank date field values are handled by:

- Delete entire record if both dates are missing/blank
- If only one of the two dates are missing/blank, set the missing date equal to the non-missing date (creates a <u>same day period</u>)
- If the end date contains a far future date such as 31Dec9999, it can either be left as is and it will be processed as if it were <u>continuous through the next valid end date</u> (appropriate for enrollment records), it can be handled as if the date is missing/blank creating a <u>same day period</u>, or it can be deleted, depending (appropriate for claims records) on user needs.

PatientID	LName	FName	DateOfServiceBeg	DateOfServiceEnd
1078JetGeo	Jetson	George	12Jan2017	13Jan2017
1078JetGeo	Jetson	George	09Feb2017	09Feb2017
1078JetGeo	Jetson	George	09Feb2017	09Feb2017
1078JetGeo	Jetson	George	09Feb2017	09Feb2017
1078JetGeo	Jetson	George	09Feb2017	31Dec9999
1078JetGeo	Jetson	George	27Mar2017	27Mar2017

Same day period

EpisodeBeg	EpisodeEnd
09Feb2017	09Feb2017

Continuous through

EpisodeBeg	EpisodeEnd
09Feb2017	27Mar2017

Click for related code

DATE ROLL UP

In this section, the records are sorted chronologically by the ID variable. Then each begin and end date are evaluated to establish the true periods associated with each unique ID variable.

PatientID	LName	FName	DateOfServiceBeg	DateOfServiceEnd
1078JetGeo	Jetson	George	12Jan2017	13Jan2017
1078JetGeo	Jetson	George	09Feb2017	09Feb2017
1078JetGeo	Jetson	George	09Feb2017	09Feb2017
1078JetGeo	Jetson	George	09Feb2017	09Feb2017
1078JetGeo	Jetson	George	09Feb2017	31Dec9999
1078JetGeo	Jetson	George	27Mar2017	27Mar2017

Same day period – appropriate for claims

12Jan2017	13Jan2017	09Feb2017	09Feb2017	27Mar2017	27Mar2017

Continuous through – appropriate for enrollment

12Jan2017	13Jan2017	09Feb2017	27Mar2017

Click for related code

ACHI CREATING EPISODES OF CARE & FINDING COVERAGE GAPS WITH ADMINISTRATIVE MEDICAL RECORDS

Stephen D. Lein, MPH

Arkansas Center for Health Improvement (ACHI)

Please use the headings above to navigate through the different sections of the poster

Code Walkthrough

REMOVE DUPLICATE RECORDS & TRANSPOSE

In this step, duplicate records are removed. The begin and end dates are transposed separately using PROC TRANSPOSE and then merged back together. This populates the episode begin (ep_in) and episode end (ep_out) variables.

Click for related code

PROCESS THE TRANSPOSED FILE

In this section, the transposed file is used to populate the episodes variable (episodes) by counting the number of episodes for each unique ID. This step also sets the following macro variables:

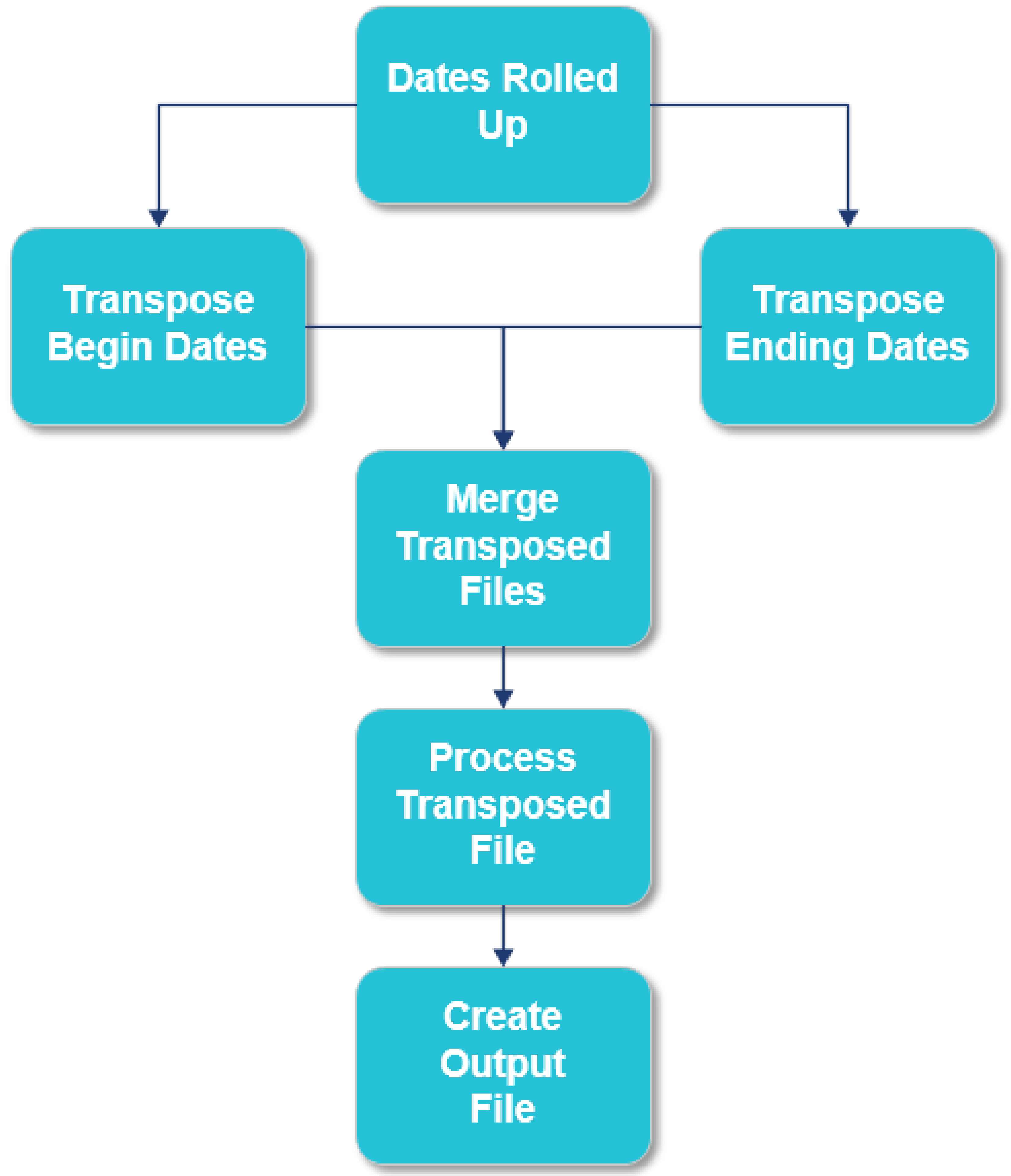
- tHigh = the number of episodes for each unique ID variable
- tSpans = the number of spans, or gaps, between each episode

Click for related code

POPULATE THE SPAN VARIABLES & CREATE OUTPUT FILE

In this final step, the span variables are populated by counting the number of days between each episode for each unique ID variable. The output data set now contains a count of the number of episodes (episodes), each episode begin (ep_in1 – ep_in*n*) and end (ep_out1 – ep_out*n*) dates and the number of days between each episode (span1 – span??) for each unique ID variable.

Click for related code



Stephen D. Lein, MPH
Arkansas Center for Health Improvement (ACHI)

Please use the headings above to navigate through the different sections of the poster

Conclusion

After rolling up an individual's experience along dates of service or dates of enrollment, creating episodes of care or continuous enrollment segments respectively, a user can then use the final output file to aggregate information such as medical or pharmacy claims costs that occur within an episode of service or identify diagnosis or procedure codes that occur within an episode of service.

A final word of caution: Because this program creates a wide record that is dependant on the number of episodes an individual ID variable has accumulated over time, the number of variables in the final output can become very large, depending on your range of dates and how often episodes occur. For example, a year's worth of data for an individual with frequent short duration medical claims generates 200 episodes. The resulting file would contain roughly 600 variables per unique ID variable.

REFERENCES

Lisa Fine, "Ready, Set, Retain, and Then Maybe Reset" *Paper 091-2011*, United Biosource Corporation, Ann Arbor, MI: SAS Global Forum 2011. Retrieved from http://support.sas.com/resources/papers/proceedings11/091-2011.pdf

Cody, Ron. (2004). *SAS® Functions by Example*. SAS® Institute Inc., Cary, NC.

Wicklin, Rick "INTCK and INTNX: Two essential functions for computing intervals between dates in SAS." Retrieved from https://blogs.sas.com/content/iml/2017/05/15/intck-intnx-intervals-sas.html

Base SAS® 9.4 Procedures Guide, TRANSPOSE Procedure.

```
************************************************************************************************
  Preprocessing
************************************************************************************************;
data rollup1; set testdata(rename=(PatientID=IDVar DateOfServiceBeg=adm_date DateOfServiceEnd=dis_date));
  ** Records with missing/blank begining and ending dates are removed **;
  if adm_date eq . and dis_date eq . then delete;
  ** Create single day date pair when one of the two dates is missing/blank **;
  ** If just the beginning date is missing/blank set the beginning date equal to the ending date **;
  if adm_date eq . and dis_date ne . then adm_date = dis_date;
  ** If just the ending date is missing/blank set the ending date equal to the beginning date **;
  if adm_date ne . and dis_date eq . then dis_date = adm_date;
  ** If the ending date contains a far future date, replace it with the beginning date.
     You may choose to comment out this edit and allow the future date to be processed in the steps
     below as if it were continuous through the next valid ending date. ;
  *if dis_date eq '31DEC9999'd then dis_date = adm_date;
run;


************************************************************************************************
  Begin Date Roll Up

  This section checks each begin date against each ending date and vice versa to get
  the true "periods" associated with each unique ID variable.
************************************************************************************************;

*****  Sort to get the dates in chronological order  *****;
proc sort data=rollup1; by IDVar adm_date dis_date;
run;
```

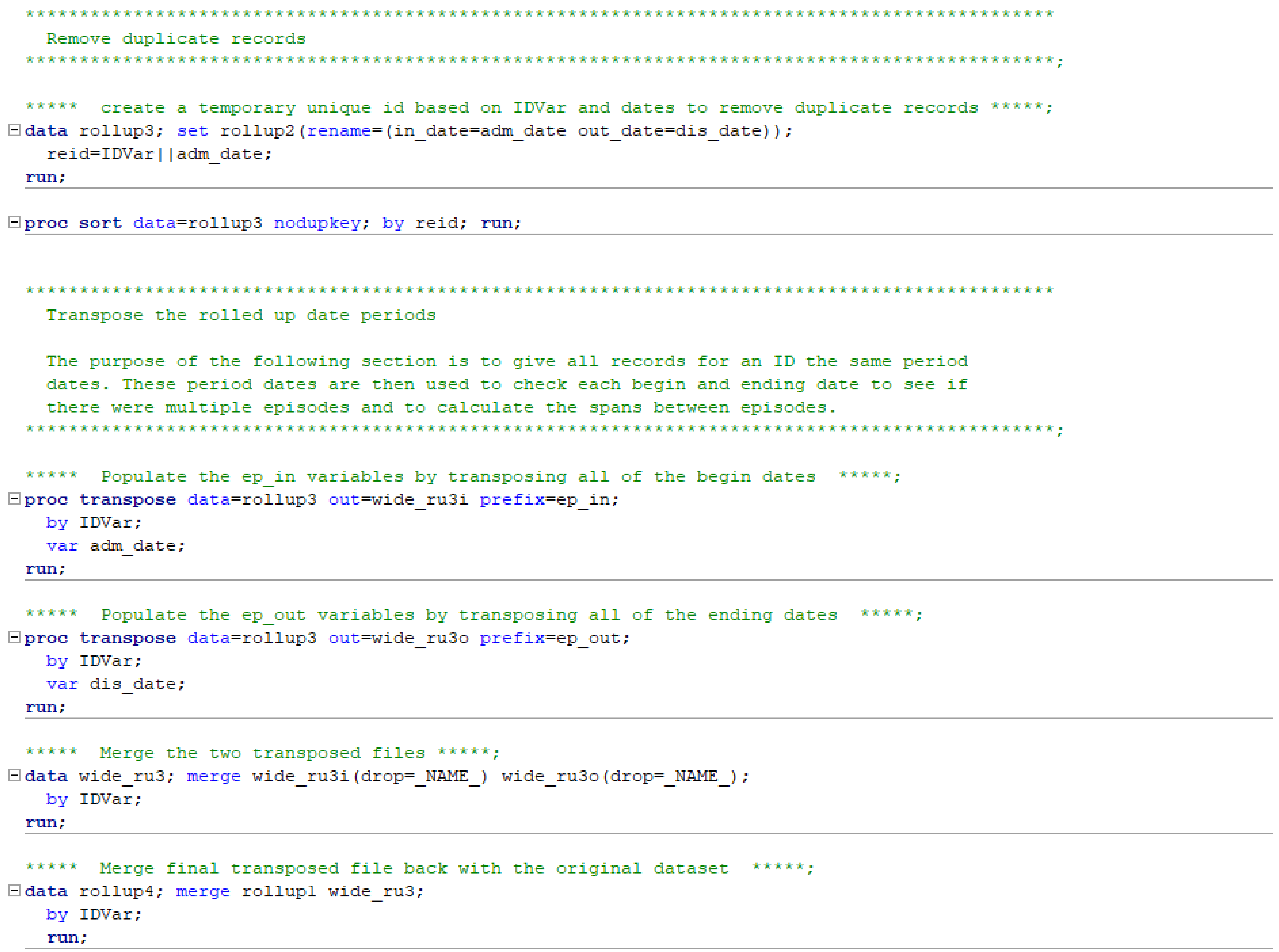
Click to return to previous slide

```
 *****  Begin rolling up the dates **;
data rollup2(keep= IDVar in_date out_date); set rollup1;
  by IDVar;
  *****  on the first id, set values and retain them  *****;
  if first.IDVar then do;
    in_date=adm_date;
    out_date=dis_date;
    retain in_date out_date;
  end;

  *****  on all records after the first...  *****;
  else do;

    *****  dates like '31DEC9999' present  *****;
    if out_date gt today() then do;
      out_date = adm_date - 1;
      retain out_date;
    end;

    *****  check to see if the current adm_date is the   *****;
    *****  same or earlier than the retained out_date    *****;
    if adm_date le out_date then do;
      *****  check to see if the retained out_date needs  *****;
      *****  to be reset to an older date                 *****;
      if out_date lt dis_date then do;
        out_date=dis_date;
        retain out_date;
      end;
    end;

    *****  check to see if the current adm_date is greater *****;
    *****  than the retained out_date                      *****;
    if adm_date gt out_date then do;
      *****  if the gap is only one day then no readmit so  *****;
      *****  reset out_date to dis_date                     *****;
      if intck('DAYS',out_date,adm_date) eq 1 then do;
        out_date=dis_date;
      end;
      *****  if the gap is more than one day output the  *****;
      *****  current values for in_date and out_date     *****;
      *****  then reset in_date and out_date to the new  *****;
      *****  period begin dates                          *****;
      else if intck('DAYS',out_date,adm_date) gt 1 then do;
        output;
        in_date=adm_date;
        out_date=dis_date;
        retain in_date out_date;
      end;
    end;

  end;
  *****  output everything when the last record for the  *****;
  *****  current id is reached                           *****;
  if last.IDVar then output;
  format adm_date dis_date in_date out_date date10.;
run;
```

Code split into two columns for readability

Note: The INTCK function counts the full time period between dates such that INTCK(‘DAYS’,’09Jan2017’d,’11Jan2017’d) returns 2. The program is configured so that a returned value of 1 does not initiate a break in the current episode because the dates would be consecutive.

Click to return to previous slide

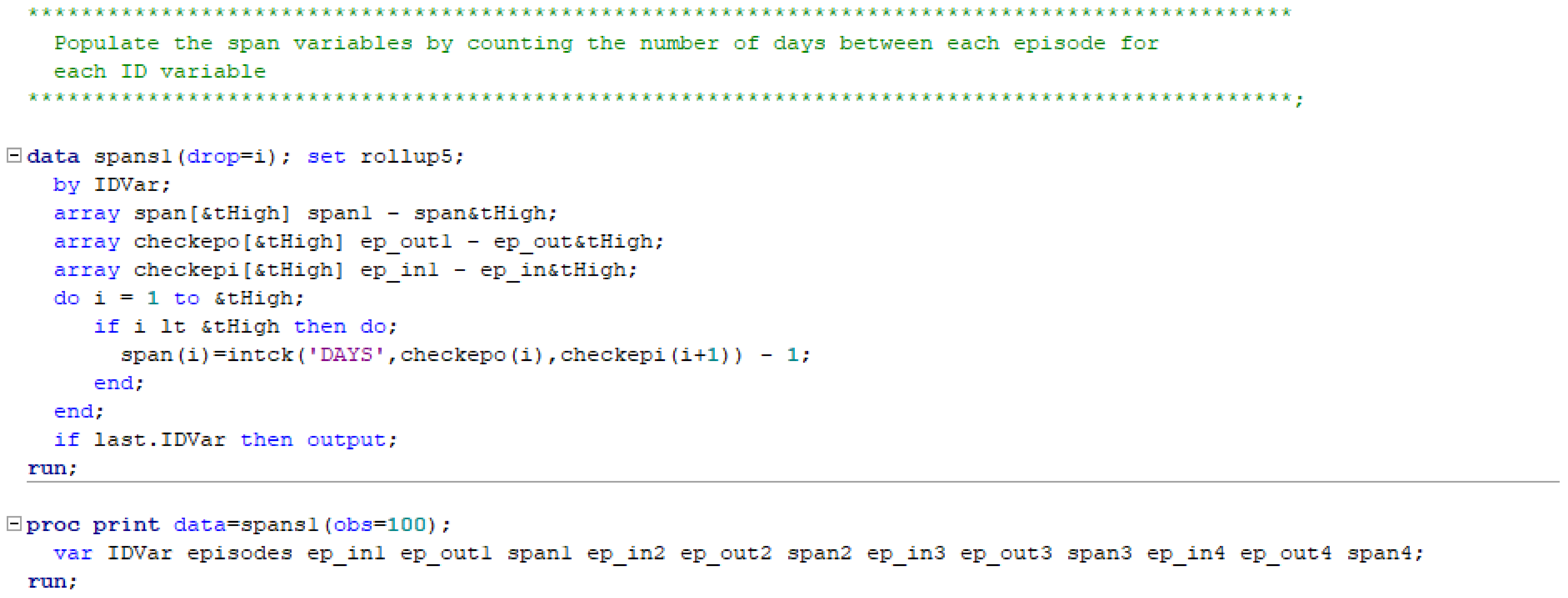


```
*********************************************************************************************
  Remove duplicate records
*********************************************************************************************;

*****  create a temporary unique id based on IDVar and dates to remove duplicate records *****;
data rollup3; set rollup2(rename=(in_date=adm_date out_date=dis_date));
  reid=IDVar||adm_date;
run;

proc sort data=rollup3 nodupkey; by reid; run;


*********************************************************************************************
  Transpose the rolled up date periods

  The purpose of the following section is to give all records for an ID the same period
  dates. These period dates are then used to check each begin and ending date to see if
  there were multiple episodes and to calculate the spans between episodes.
*********************************************************************************************;

*****  Populate the ep_in variables by transposing all of the begin dates  *****;
proc transpose data=rollup3 out=wide_ru3i prefix=ep_in;
  by IDVar;
  var adm_date;
run;

*****  Populate the ep_out variables by transposing all of the ending dates  *****;
proc transpose data=rollup3 out=wide_ru3o prefix=ep_out;
  by IDVar;
  var dis_date;
run;

*****  Merge the two transposed files *****;
data wide_ru3; merge wide_ru3i(drop=_NAME_) wide_ru3o(drop=_NAME_);
  by IDVar;
run;

*****  Merge final transposed file back with the original dataset  *****;
data rollup4; merge rollup1 wide_ru3;
  by IDVar;
  run;
```

Click to return to previous slide

```
*********************************************************************************************
  Process the transposed file
*********************************************************************************************;

*****  Use PROC CONTENTS to get a list of all of the different episode dates  *****;
proc contents data=rollup4 noprint out=rollup4cont; run;

*****  The resulting file is run through this macro to count the number of episodes  *****;
*****  and assign values to several macro variables.                                 *****;
%let tVarName = ep_in;
proc sql noprint;
  select count(*)
    into :tNobs from rollup4cont(where=(name=:"&tVarName"));
  quit;
*%put number of admits is &tNobs;

%let tHigh= %eval(&tNobs);
*%put high= &tHigh;
%let tSpans= %eval(&tNobs - 1);
*%put high= &tSpans;
run;

*****  Populate the episodes variable by counting the number of episodes for each ID variable  *****;
data rollup5(drop=i); set rollup4;
  episodes = 0;
  array checki[&tHigh] ep_in1 - ep_in&tHigh;
  array checko[&tHigh] ep_out1 - ep_out&tHigh;
  do i = 1 to &tHigh;
    if adm_date ge checki(i) and dis_date le checko(i) then episodes=i;
  end;
run;

proc sort data=rollup5;
  by IDVar adm_date dis_date;
run;
```

Click to return to previous slide



```
*****************************************************************************
  Populate the span variables by counting the number of days between each episode for
  each ID variable
*****************************************************************************;

data spans1(drop=i); set rollup5;
  by IDVar;
  array span[&tHigh] span1 - span&tHigh;
  array checkepo[&tHigh] ep_out1 - ep_out&tHigh;
  array checkepi[&tHigh] ep_in1 - ep_in&tHigh;
  do i = 1 to &tHigh;
     if i lt &tHigh then do;
       span(i)=intck('DAYS',checkepo(i),checkepi(i+1)) - 1;
     end;
  end;
  if last.IDVar then output;
run;

proc print data=spans1(obs=100);
  var IDVar episodes ep_in1 ep_out1 span1 ep_in2 ep_out2 span2 ep_in3 ep_out3 span3 ep_in4 ep_out4 span4;
run;
```

Click to return to previous slide

SAS® GLOBAL FORUM 2020

USERS PROGRAM

MARCH 29 - APRIL 1 | WASHINGTON, DC | #SASGF