MWSUG 2018 - Paper HS-074

Use of SAS Macros to Automate the Production of Statistical Process Control Charts

Lynn (Xiaohong) Liu, Ann & Robert H. Lurie Children's Hospital of Chicago, Chicago, IL
Roderick Jones, Ann & Robert H. Lurie Children's Hospital of Chicago, Chicago, IL

ABSTRACT

The SAS/QC® SHEWHART procedure generates statistics and statistical process control (SPC) charts used to measure improvement and special cause variation in a process. To produce output automatically on a repeating schedule for a large number of metrics, a system of sequential SAS programs with macro processing was developed and implemented.

The foundation for the process is a parameter file, which stores information including each metric's definition, record-level data set, variable name and label, temporal unit of analysis, starting point of the time period to be analyzed and SPC chart type. The parameter file is imported and each row (or 'run') is converted into macro variables using %LET and %SYSFUNC. Moreover, CALL SYMPUT, SQL SELECT INTO and %SYMEXIST assign or detect macro variables in real time, which allows a dynamic response from the system.

The system of SAS programs acts as a series of pathways with each run directed according to its macro variables using %IF - %THEN/%ELSE logic. Each pathway has code to read record-level data sets, calculate necessary summary statistics and produce outputs.

PROC SHEWHART is applied iteratively, the number of times depending on OUTTABLE data set contents and the detection of shifts that require new phases to be defined. With a %DO loop the pathways culminate in the generation and delivery to document libraries of a SPC chart image files, metric description image file, and Microsoft Excel® summary statistics file for each run.

INTRODUCTION

Monitoring and evaluating variation in a process often requires periodic generation of a large amount of various statistical process control (SPC) charts. In this paper we introduce a system of sequential SAS programs with macro processing to automate SPC production. Although we use healthcare as the setting to discuss the automation process, In light of the similarity of components involved in SAS/QC automation, it also applies to automating the production of SPC charts for manufacturing.

There are two challenges in accomplishing this automation process. One is unique attributes of each metric. The SPC charts produced vary in terms of chart types, number of data points, baseline requirement, shift presence, date time format, temporal unit of analysis and metric definition. The second challenge is to make the system dynamically respond to what happens when reading in each metric. This challenge is reflected on the algorithms in the flowcharts of Appendix 1 and 2. In healthcare the purpose of SPC is often to quantify improvements. A recommended technique for detecting improvement is to create an initial center line using baseline points and then extend this center line into the future (Provost and Murray 2011). Thus dynamically detecting shifts in comparison to a baseline is an important part of the automating process.

The two flowcharts in Appendix 1 and 2 differ in their baseline requirements. Appendix 1 is used when the client does not have a pre-defined baseline time period requirement, whereas Appendix 2 pertains to scenarios where the baseline period is pre-determined. As it is simple to define baseline data points based on specific data points from the clients, we need to follow more steps to use as many data points as possible for no specific baseline requirement metrics. After going through the pathways of defining baseline and detecting the first shift, all the pathways converge toward a pathway of cyclic shift detection.

We use macro variables and macro functions to implement these pathways dynamically along accommodating various metrics. We add a run number after the name of each macro variable, which is essential for the success of the system, since it is common that the metrics within a project produce some macro variables with same names, but with a specific &run number after each macro variable, the data and outputs generated for a metric is unique in a project.

This paper first presents every step in the system and then introduces the tools that connect these steps and pave the pathways, thus accomplishing the automation process.

CREATE AND USE MACRO VARIABLES FROM A STATIC PARAMETER FILE

A parameter file lays a foundation for automating the process by storing specific information for each metric. It includes chart type, metric definition, record-level input data set, variable name and label, temporal unit of analysis, starting point, time period to be analyzed, baseline definition, metric population and population exclusion. Each value in the file is converted into a macro variable and then used in the system.

CONSTRUCT A PARAMETER FILE

An initial parameter has the basic information about the metrics. Tables 1.1 - 1.4 illustrate how the information is stored. For example, for run1, a metric (variable) called _Order_set_used is from a data set named ALL and to produce *p* chart. This metric is reported by month and the text present on the SPC chart is 'Order set used'. The time period to be analyzed for a SPC cycle is from March 2016 until &LastMonthEndDate (SAS resolves &LastMonthEndDate corresponding to the specific SPC cycle). The date variable used to select the data for the period is Metric_Date and the metric requires 20 data points to calculate the baseline mean.

RUN	VALUE	VAR2	VAR2LABEL		CHARTS
1	Yes	_Order_set_used	Order set used	ALL	P chart
2		_Length_of_stay	Length of stay	ALL	Median chart
3		_Time_to_order	Time to order	ITT	Xbar-S chart
4	Yes	_Antibiotic_not_given	No antibiotic given	LEG	C chart
5		_Lab_time	The time for lab	ALL	I - MR chart
6		_intervention_required	Intervention required	ALL	Rare events chart

Table 1.1. Parameter File, continued

RUN	CAL_FIS	PERIOD	START	END
1	Calendar	March 2016 - &LastMonthEndDate	21MAR16	&LastMonthEndDate
2	Calendar	November 2015 - &LastMonthEndDate	01Nov15	&LastMonthEndDate
3	Fiscal	November 2015 - &LastMonthEndDate	01Nov15	&LastMonthEndDate
4	Fiscal	November 2015 - &LastMonthEndDate	01Nov15	&LastMonthEndDate
5		November 2015 - &LastMonthEndDate	01Nov15	&LastMonthEndDate
6	Calendar	March 2016 - &LastMonthEndDate	01Mar16	&LastMonthEndDate

Table 1.2. Parameter File, continued

N	SUBUNIT	UNIT	BASELINE_REQUIRED	TIME_VAR	TIME_UNIT
1	month	Month	20	Metric_Date	_Year_Month
2	month	Month	20	Metric_Date	_Year_Month
3	QTL	Quarter	12	Metric_Date	_Year_Quarter
4	QTL	Quarter		Metric_Date	_Year_Quarter
5		Observation		Metric_Date	
6				Metric_Date	Adm_Date_Time

Table 1.3. Parameter File, continued

In SAS/QC a rare events chart (RAREEVENTS procedure) works differently from other charts. Instead of plotting counts of events, the chart plots counts of 'missed opportunities' or 'measures of elapsed time' between events (Ransdell, 2016), so the way to count total data points, define baseline and the layout of the chart are different from other traditional SPC charts, therefore, as shown in run 6 (Table 1.4) we store rare events chart information in additional columns within the parameter file.

RUN	BASLINE_REQUIRED_R	LABEL_SEEN_R	LABEL_PAT_R	FOOTNOTE_R	SORT_DISPLAY_R	SORT_UNIT_R
1						
2						
3						
4						
5						
6	28Feb18	Readmission_Date	Patients discharged	Patients were discharged	Discharge_Date	Hosp_Discharge_Time

Table 1.4. Parameter File, continued

We use proc import to read in the parameter file and then INTCK function to count the total data points for the specific SPC cycle. The differences are considered when total data points are counted, such as different temporal units or special treatments for some metrics. For example, in healthcare the period of reporting a readmission metric needs an extended observation time. Generally we extend 30 more days to report this metric than other metrics, so when counting points for the metric, &LastMonthEndDaten is used instead of &ThisMonthStartDaten which results in 1 point less than the other metrics:

```
data parameter_file;
       set parameter_file;
              startd=INPUT(start,date7.);

       if UPCASE(time_unit)="_YEAR_WEEK"
        then points=INTCK("week",startd,&thisweekstartdaten);

       else if UPCASE(time_unit)="_YEAR_QUARTER"
        then points=INTCK("qtr",startd,&thisweekstartdaten);

       else if UPCASE(var2) in ("_REMOVE_30D","_READMIT_30D")
        then points=INTCK("month",startd,&lastMonthEndDaten);

        else points=INTCK("month",startd, &thismonthstartdaten);
run;
```

As stated previously the data points of rare events charts are either the number of opportunities between rare events or the time lapse between rare events, so the data points are calculated from the input data after identifying the rare events, not from the counting of the temporal unit points between a start date and an end date, therefore, the initial counting of total data points in the parameter file is incorrect. A replacement step is carried out to compute the opportunities or time between the rare events and replace the previously-generated one. The same correction needs to be done for Individuals – Moving Range (I - MR) charts, since I - MR charts are present as individual measures in a sequential order, so total data points are not associated with any temporal units or. Instead they are usually the total observation number of a data set.

Below is the code to correct the macro variable value for total data points for a rare events chart:

```
proc sort data=input_data
          out=input_data_&run;
```

```
           by &sort_unit_r;
run;

data input_data_&run;
       format seen mmddyy8.;
       label seen ="&label_seen_r";
              seen=DATEPART(&time_unit);
              visit_num=_n_;
run;

data event&run;
       set input_data_&run (where=(&v=" &value"));
              patsbetween=visit_num-LAG(visit_num);
       label patsbetween ="&label_pat_r. between events";
run;

proc sql noprint;
       select COUNT(*)
       into :points_&run
       from event&run;
quit;

data parameter_file;
       set parameter_file;
              if run=&run
              then points=&&points_&run;
run;
```

Below is the code to correct the macro variable value for total data points for an I - MR chart:

```
data input_data_&run;
       set input_data;
run;

proc sql noprint;
       select count(*)
       into :points_&run
       from input_data_&run;
quit;

data parameter_file;
       set parameter_file;
              if run=&run
              then points=&&points_&run;
run;
```

Based on the algorithms in Appendix 1 and 2, if the number of data points is less than or equal to 12, a ‘run’ chart is produced, so baseline points are needed. If there is no specific baseline requirement, the system will initially use 12 points as the baseline; if there is a specific client requirement for baseline, the baseline data points can be either 12 or 20. The code below reflects the process above:

```
data parameter_file;
       set parameter_file;

              if points<=12 and not MISSING(points)
              then do;
                     baseline_points=.; check8_points=.;
       end;
```

```
                else if MISSING(baseline_required)
                then do;
                        baseline_points=12; check8_points=20;
        end;
```

Define the 20th point as the checking point

```
                else if baseline_required =12
                then do;
                        baseline_points=12; check8_points=.;
        end;

                else if baseline_required =20
                then do;
                        baseline_points=20; check8_points=.;
        end;

                else do;
                        check8_points=.; baseline_points=.;
        end;
run;
```

The definition of the baseline period needs more work when there is no specific request for baseline. After the system initially uses 12 data points as the baseline, it checks whether 20 points are available and whether there is a shift before reaching the 20th point. The code framed in the rectangle above is to initially define 12 points as the baseline, then use the 20th point (check8_points) to determine if 20 points are available and if a shift exists before reaching to 20th point. We will give more details on how this variable is used in pathway #4.

Like some of data analysis that uses a date or time value to segment data out of a database for a specific time period, we generally use an admission date/time or discharge date/time value to extract data for baseline mean calculation, so we need to convert the baseline points or checking points (check8_points) into a specific time value. For example, the data steps below use INTNX function to convert baseline_points in Week unit into a baseline end time and then the value is used in PROC SHEWHART to fragment the data for baseline mean calculation:

```
If UPCASE(time_unit)="_YEAR_WEEK" and points-baseline_points>=0
then baseline_enddatetime=((INTNX("week",startd,(baseline_points))) -1)*24*60*60;

proc shewhart data=&subunit.&run (where=(&&TIME_VARIABLE&run le
                                         datepart (&baseline_enddatetime) and
                                         &&TIME_VARIABLE&run ge &startd));
        xschart &v*&time_unit
run;
```

USE THE CONVERTED MACRO VARIABLES FROM PARAMETER FILES

%LET and %SYSFUNC are two functions to convert each variable in the parameter file into a macro variable. Below is the code to open a parameter file with 14 metrics, fetch every variable in each run and convert the variable into either a numeric macro variable or a character macro variable:

```
%do run= 1 %to 14;

%let dset=parameter_file;
%let dsid=%SYSFUNC(open(&&dset));
%let rc=%SYSFUNC(fetchobs(&dsid,&run));

*Convert a numeric variable into a macro variable;
%let period=%SYSFUNC(getvarC(&dsid, %SYSFUNC(varnum(&dsid,period))));

*Convert a numeric variable into a macro variable;
%let baseline_required=%SYSFUNC(getvarN(&dsid, %SYSFUNC(varnum
                        (&dsid,baseline_required))));
```

The converted macro variables are used repeatedly in several steps of the system: to prepare data for SPC charts, to serve as a parameter in PROC SHEWHART, to output data externally and to use as a condition for a pathway. We present three examples of how these macro variables function in the system.

The first example uses the converted &baseline_enddatetime macro variable to extract the baseline data and uses FREQ procedure to prepare data before PROC SHEWHART:

```
ods output crosslist=&subunit.&run;
proc freq data= input_data_&run
       (where=(&&time_variable&run le DATEPART(&baseline_enddatetime) and
               &&time_variable&run ge &startd));
        tables &time_unit*&v/crosslist;
run;
```

The second example shows how the macro variables are used in PROC SHEWHART to create a SPC chart. For instance, run 2 (Table 1.1) creates a median chart for the Length of stay metric with _Year_Month as the temporal unit. If the monthly SPC cycle ends in June 2018, the end data, &LastMonthEndDate, resolves to June 2018, and then the chart has the title as 'Median chart of Length of stay, November 2015 - June 2018 by month':

```
proc shewhart data=&subunit.&run;
       mchart &v * &time_unit /
       odstitle= "&chart of &label., &period by &unit";
run;
```

The third example selects each temporal unit of the data points in a baseline data set into a macro variable for future baseline definition. The macro variable holds a list of temporal units that are defined as baseline for the metric. For run3 (Table 1.1 - 1.3), if the 1st quarter of 2016 until the 4th quarter of 2018 are defined as the baseline for the metric, the macro variable &&subunit.list&run (QTLlist3) stores a list of values: 2016-Q1 until 2018-Q4:

```
proc sql noprint;
       select catt ("'", &time_unit, "'")
       into :&subunit.list&run separated by ', '
       from base&subunit.&run;
quit;
```

Base&subunit.&run is the predefined data set that has the baseline temporal units of each metric.

DYNAMICALLY CREATE AND USE MACRO VARIABLES

Besides creating the macro variables from a static parameter file, we also generate macro variables dynamically during the execution of the system. It is especially important for paving different pathways for a variety of metrics. For instance, if a shift is present, Test 2 is shown as positive in the OUTTABLE data set from PROC SHEWHART. The information has to be captured in the system in order to direct the metric to its pathway accordingly. Additionally, different metrics may have different formats of time variables. Digesting these different formats is essential for the system to accommodate various metrics.

CREATE MACRO VARIABLES AND DETECT THEIR EXISTENCE

We use CALL SYMPUT or the INTO clause of the SQL procedure to convert what happens in real time in the system into a macro variable, and then use macro functions to detect its existence. Whether or not these macro variables exist serves as a condition in directing the pathways. Below are three examples that show how the information is captured and used in the system.

The first example converts the name of a data set to a macro variable and detects its existence. An OUTTABLE data set is created from PROC SHEWHART to store the summary statistics for the metric. Not all metrics have the table produced and generally it depends on whether or not a shift is present. The EXIST function verifies the existence. The code below uses %SYSFUNC (EXIST) to check whether the

OUTTABLE data set named as statistics&run exists or not. If it exists, the system proceeds to a pathway for producing a shift. If it does not, a different pathway is taken:

```
proc shewhart data=&subunit.&run;
       outtable=statistics&run;
run;

%let outdata = statistics&run;

%if %SYSFUNC(EXIST(&outdata)) %then %do;
```

The second example creates a macro variable from a data set’s observation count. Sometimes even though the OUTTABLE data sets are created from PROC SHEWHART, there are not any observations in the data sets, which also indicates there is no valid shift existing for the metric.

There are several papers that have been published related to identifying empty data sets. This code is adapted from the second method that Childress and Welch presented in 2011 works fine in the system:

```
%let indata = statistics&run;
%let numobs = 0;
%let dsnid = %SYSFUNC(OPEN(&&indata));
%let dsobs = %SYSFUNC(ATTRN(&dsnid.,nobs));
%let rc = %SYSFUNC(CLOSE(&dsnid.));
%let numobs = &dsobs.;

%if &numobs gt 0 %then %do;
```

The method initially defines a macro variable named as &numobs to be 0, opens the OUTTABLE data set, reads its observation number through the ATTRN function and then assigns the numeric value to &numobs. If the resulted &numobs is greater than 0, it means the data set is not empty. However, the simpler approach below works as well as the method above:

```
proc sql noprint;
      select count(*)
      into: check
      from statistics&run;
quit;

%if (&check) %then %do;
```

In %IF - %THEN - %DO processing, a non-zero and nonmissing result causes the expression to be true; a result of zero or missing causes the expression to be false. If &check is not missing, the condition is true and the pathway continues.

The third example converts a temporal unit point into a macro variable. Wherever there is a positive Test 2 – a value of 2 in the variable _TEST_ in THE OUTTABLE data sets from PROC SHEWHART, the system converts the temporal unit corresponding to its row into a macro variable and then the macro variable is used as a condition in defining the pathways. %SYMEXIST is a function to return an indication of the existence of a macro variable:

```
proc sql  noprint ;
       select &time_unit
       into: shift&run
       from statistics&run (obs=1)
       where index( _tests_,'2');
quit;

%if %SYMEXIST(shift&run) %then %do;
```

DIGEST DIFFERENT DATE FORMATS

When a date variable is used to select data a defined time period, the date variables may come with different formats. They could be in date format or datetime format. While it is simple to use it when it is in date format, a date function called DATEPART needs to be used if it is in datetime format. The code below dynamically creates a macro variable with or without the DATEPART function:

```
data input_data;
       set input_data;
            timevar=DATEPART(&Time_var);
            if _n_ = 1
            then CALL SYMPUT ("timevarcheck", TRIM(LEFT(timevar)));
run;
```

&Time_var is a macro variable from the parameter file (Table 1.3). The method above takes advantage of the property of DATEPART function itself. If a date is in datetime format, DATEPART returns a normal date, but if it is in date format already, it returns a value of 0, so the macro variable &timevarcheck has a value of either date or 0:

```
%macro timevar;
       %global time_variable&run;
       %if &timevarcheck=0
       %then %do;
              %let time_variable&run=&time_var;
       %end;
       %else %do;
              %let time_variable&run=DATEPART(&time_var);
       %end;
%mend;
%timevar;
```

The code above is to assign a value to &&time_variable&run based on the value of &timevarcheck. When &timevarcheck is equal to 0, &&time_variable&run gets the value of &time_var, but if &timevarcheck is something else (a date), &&time_variable&run has the value of DATEPART (&time_var). &&time_variable&run is the date variable that functions in the code.

PROCEED DOWN PATHWAYS

Using %IF - %THEN - %DO logic, the macro variables and macro functions, we pave 7 pathways in accordance with the flowcharts in Appendix1 and 2. The pathways differ in the total number of data points, the availability of baseline points and the presence of a shift, the chart type and whether or not a pre-determined number of baseline points is requested.

In every pathway with a shift, the algorithms try to use either 8 or 20 data points to calculate the center line mean, which depends on whether 20 data points are available after the start of the shift. The conditions below are used in all the pathways that have a shift:

```
%if &&shift_20_end&run > &points %then %do;
%if &&shift_20_end&run =< &points %then %do;
```

The macro variable &&shift20_end&run is the position number of the 20th data point of a shift in the OUTTABLE data set from PROC SHEWHART and it is assigned a value by the code below:

```
data statistics&run;
       set statistics&run;
           rec=_n_;
run;

proc sql  noprint ;
       select &time_unit
       into: shift&run
       from statistics&run(obs=1) where INDEX( _tests_,'2');
quit;
```

```
proc sql  noprint ;
       select rec +12
       into: shift20_end&run
       from statistics&run  where &time_unit="&&shift&run";
quit;
```

The rec variable is derived from _n_ in the statistics&run data set. It is the 8th data point of the shift when &time_unit is equal to &&shift&run, so rec adding 12 points brings the points to the 20th. If &&shift20_end&run is bigger than the total data points of the statistics&run, it means the point does not exist, so the system has to use 8 points to calculate the center line mean. When &&shift20_end&run is less than or equal to the total data points of the data set, it indicates the 20th point of the shift does exist and then 20 points are used for the calculation of the center line mean.

PATHWAY #1 PRODUCE ‘RUN’ CHARTS

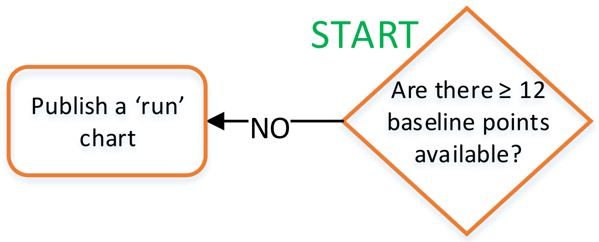


Figure 1. Pathway #1 publishes a ‘run’ chart when 12 or fewer baseline points are available

When a metric’s data set does not include enough data points to produce a valid SPC chart, a ‘run’ chart can be produced.

The pathway is directed by the statement below:

```
%if &points <=12 %then %do;
```

The macro variable &points comes from the total point counting step. When producing the ‘run’ chart, we keep upper and lower control limits in the chart, but remove any special cause test results. Accordingly, we state the limitation of the ‘run’ chart in the descriptive table. Below is part of the code to produce the ‘run’ chart descriptive table:

```
data descriptive_table&run;
       format  item $30. detail $1000.;
run;

proc sql;
       insert into descriptive_table&run
       values ("Special cause tests", ‘This chart doesn't have minimal
               number points for baseline calculation or statistical
               testing.");
quit;
```

PATHWAY #2 PRODUCE INITIAL BASELINE CHARTS

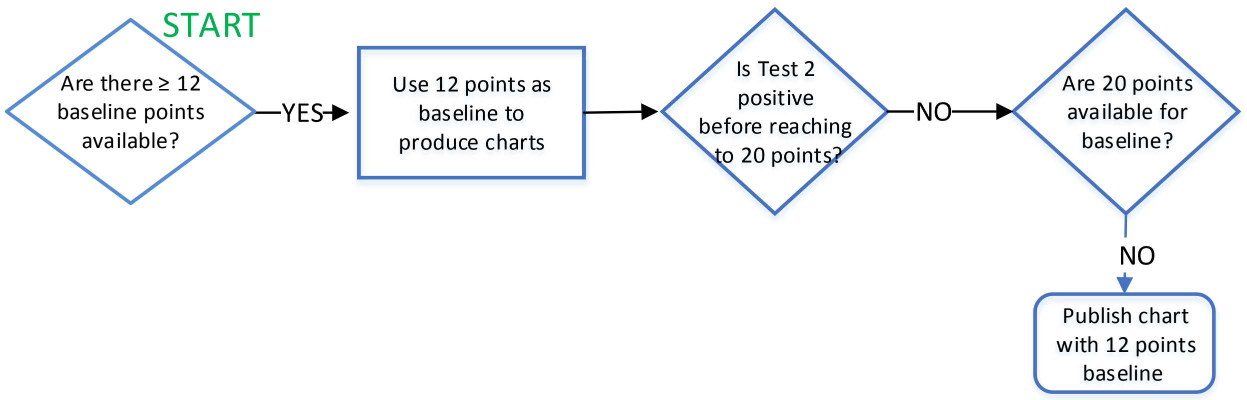

START
Are there ≥ 12 baseline points available?
YES
Use 12 points as baseline to produce charts
Is Test 2 positive before reaching to 20 points?
NO
Are 20 points available for baseline?
NO
Publish chart with 12 points baseline

Figure 2.1. Pathway #2 publishes a chart with a 12-point baseline in cases when the baseline period is not pre-determined

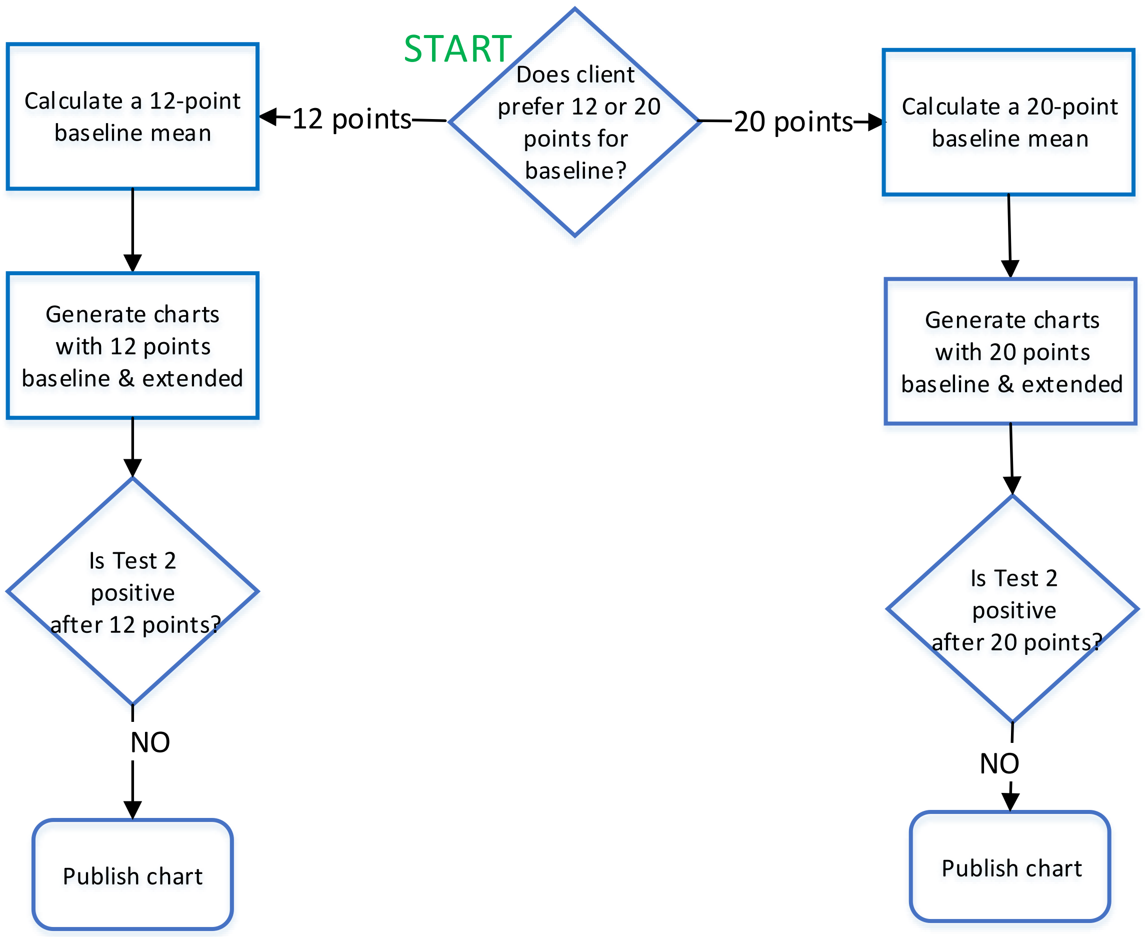

START
Does client prefer 12 or 20 points for baseline?
12 points
20 points
Calculate a 12-point baseline mean
Calculate a 20-point baseline mean
Generate charts with 12 points baseline & extended
Generate charts with 20 points baseline & extended
Is Test 2 positive after 12 points?
Is Test 2 positive after 20 points?
NO
NO
Publish chart
Publish chart

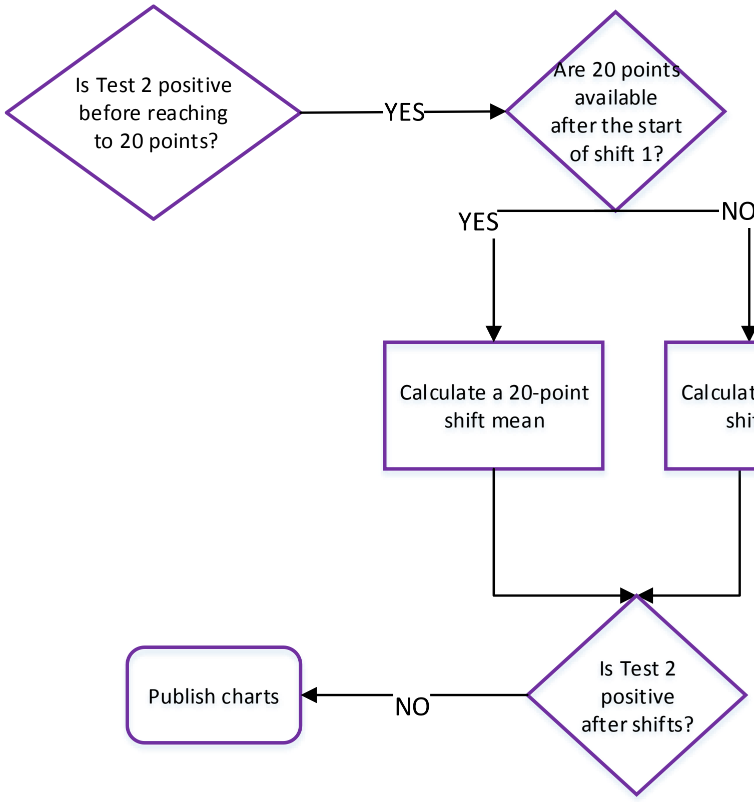
Figure 2.2. Pathway #2 publishes a chart with a 12- or 20-point baseline in cases when the client specifies the baseline period

A chart with an initial baseline is the start pointing for a shift detection. When a baseline is pre-determined as 12 or 20 points, they system uses the macro variable &baseline_points to calculate the center line. Without a pre-determined baseline, the system uses 12 points first. If 20 points are available, it proceeds down pathway #4.

The pathway is directed by the statement below:

```
 %if &points > 12 %then %do;
```

PATHWAY #3 PRODUCE 12 POINTS BASELINE AND SHIFT 1 – NO SPECIFIC BASELINE REQUESTS



Is Test 2 positive before reaching to 20 points?
YES
Are 20 points available after the start of shift 1?
YES
NO
Calculate a 20-point shift mean
Calculate an 8-point shift mean
Publish charts
NO
Is Test 2 positive after shifts?

Figure 3. Pathway #3 publishes a chart when a shift is detected prior to the 20th point

If there is a shift after 12 data points and before 20 data points, the system uses 12 points as a baseline. Since PROC RAREEVENTS does not feature tests for special cause variation, it is excluded from this pathway:

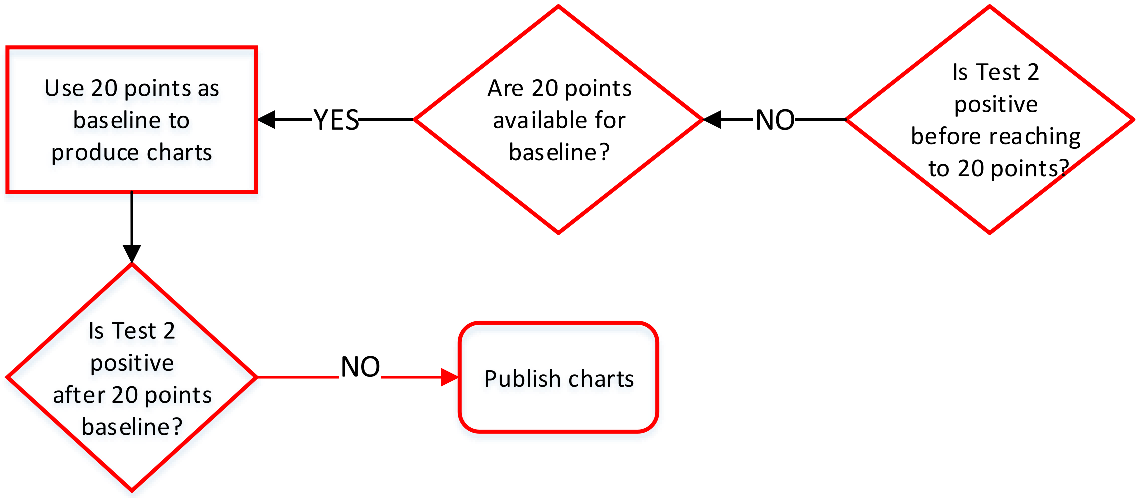
```
%if &baseline_required=.  and &points >12 and %SYSFUNC(SUBSTR(&chart,1,1)) ^=r
%then %do;

proc sql  noprint ;
       select &time_unit
       into: firstshift&run
       from statistics&run(obs=1)
       where INDEX( _tests_,'2')
              and rec gt&baseline_points
              and rec le &check8_points;
quit;

%if %SYMEXIST(firstshift&run) %then  %do;
```

As stated in the previous paragraphs &check8_points is the 20th point of an input data set and serves as the checking point for 20 points status, so the first shift needs to be detected between &basline_points and &check8_points. If there is a Test 2 positive, the pathway goes on to produce SPC charts with a baseline and the first shift. The shift mean is calculated either from 8 points or 20 points.

PATHWAY #4 PRODUCE CHARTS WITH 20 POINTS BASELINE – NO SPECIFIC BASELINE REQUESTS



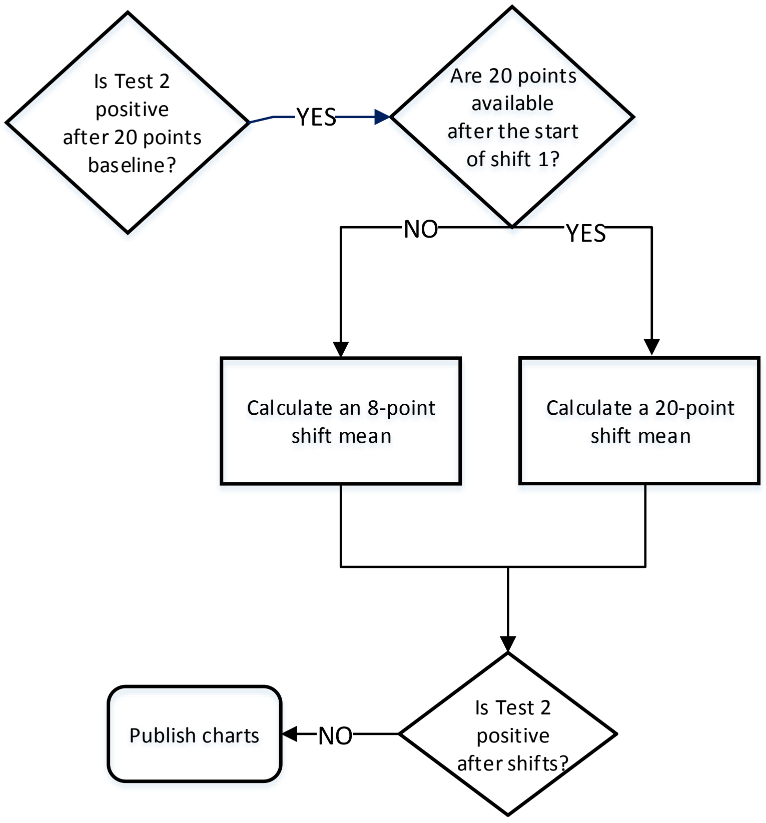
Is Test 2 positive before reaching to 20 points?
NO
Are 20 points available for baseline?
YES
Use 20 points as baseline to produce charts
Is Test 2 positive after 20 points baseline?
NO
Publish charts

Figure 4. Pathway #4 publishes a chart with a 20-point baseline

When no shifts are detected prior to the 20th point, the 20 points are used to calculate the baseline mean for purposes of future shift detection. %SYMEXIST (firstshift&run)=0 indicates the shift does not exist and &check8_enddatetime^=. means 20 points are available:

```
%if %SYMEXIST(firstshift&run)=0 and &check8_enddatetime^=. %then  %do;
```

PATHWAY #5 DETECT SHIFTS AFTER 20 POINTS – NO SPECIFIC BASELINE REQUESTS



Is Test 2 positive after 20 points baseline?
YES
Are 20 points available after the start of shift 1?
NO
YES
Calculate an 8-point shift mean
Calculate a 20-point shift mean
Is Test 2 positive after shifts?
NO
Publish charts

Figure 5. Pathway #5 publishes a chart when a shift is detected after 20 points

Pathway #5 tries to detect the first shift. The system uses the conditions below to screen the metrics initially:

```
%if &baseline_required=. and &points >20 and %SYSFUNC(SUBSTR(&chart,1,1)) ^=r
%then %do;
```

If 20 points are available for baseline, the OUTTABLE data set from PROC SHEWHART in pathway #4 exists and has observations:

```
proc shewhart data=input_data&run  limits=limits&run;
       irchart &v * &time_unit / odstitle= "&chart of &label., &period
       outtable= outtable_from_20_points&run;
run;
```

The system ensures that the OUTTABLE data set from PROC SHEWHART in pathway #4 is available and not empty at the beginning of pathway #5. The condition serves to connect pathway #4 and #5:

```
%let indat = outtable_from_20_points&run;
%if %SYSFUNC(EXIST(&indat)) %then %do;

proc sql noprint;
       select COUNT(*)
       into: check
       from outtable_from_20_points&run;
quit;

%if (&check) %then %do;
```

And then detect a shift after 20 data points:

```
proc sql  noprint ;
       select &time_unit
       into: firshift&run
       from outtable_from_20_points&run (obs=1) where index( _tests_,'2')
                                                and rec gt &check8_points;
quit;
%if %SYMEXIST(firshift&run) %then  %do;
```

PATHWAY #6 FIRST SHIFT DETECTION - SPECIFIC BASELINE REQUESTS

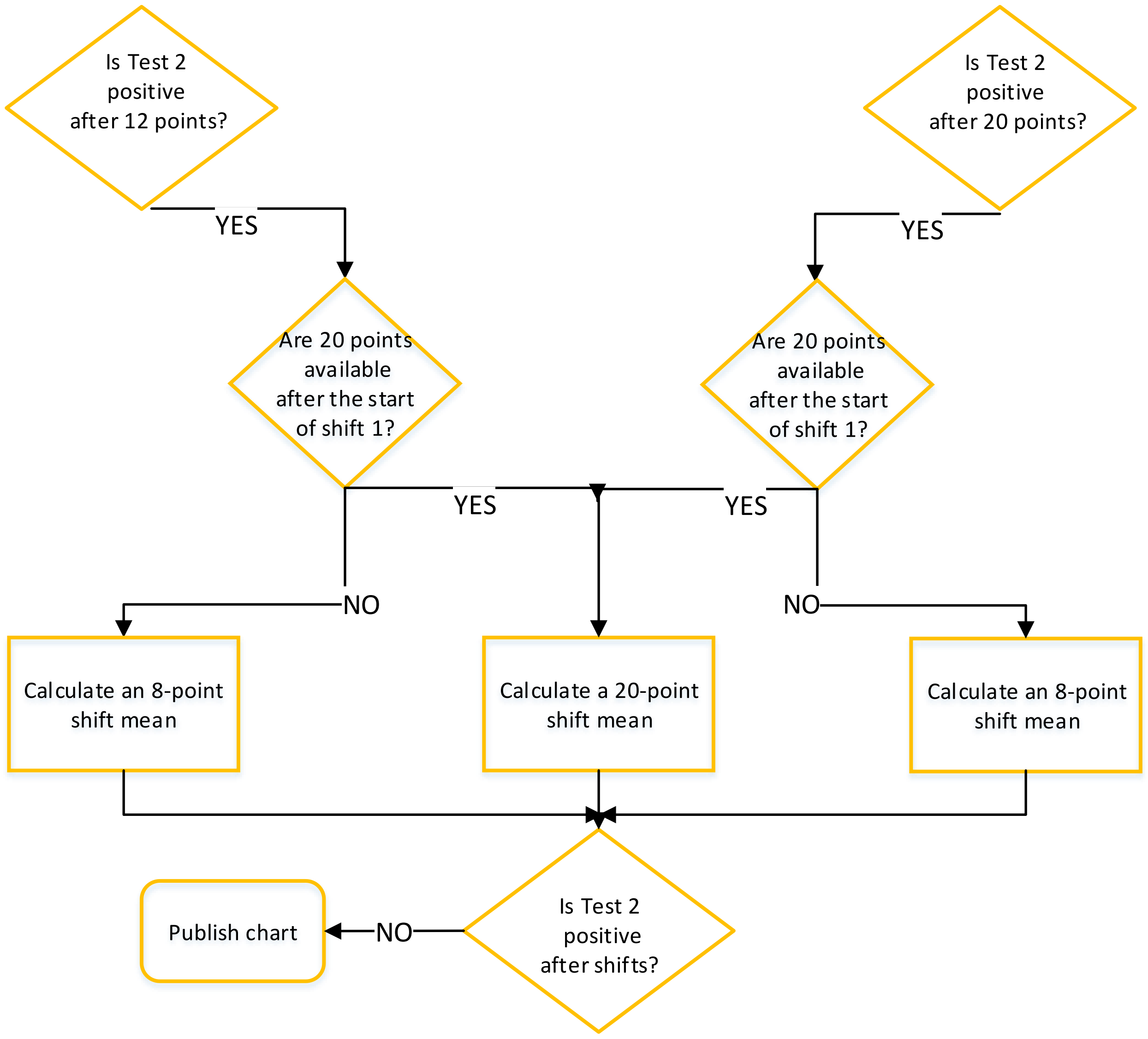

Is Test 2 positive after 12 points?
Is Test 2 positive after 20 points?
YES
YES
Are 20 points available after the start of shift 1?
Are 20 points available after the start of shift 1?
YES
YES
NO
NO
Calculate an 8-point shift mean
Calculate a 20-point shift mean
Calculate an 8-point shift mean
Publish chart
NO
Is Test 2 positive after shifts?

Figure 6. Pathway #6 publishes a chart after detecting the first shift

For scenarios in which the client specified a baseline to use, after the baseline either from 12 points or 20 points is created the system detects the first shift with it. The pathway first is filtered by the statement below:

```
%if &baseline_required^=.  and &points >12 and %SYSFUNC(SUBSTR(&chart,1,1)) ^=r %then
%do;
```

Then detects the first shift after the baseline:

```
proc sql  noprint ;
       select &time_unit
       into: firstshift&run
       from statistics&run (obs=1)
       where INDEX( _tests_,'2')
              and rec gt &baseline_points;
quit;
```

```
%if %SYMEXIST(firstshift&run) %then  %do;
```

PATHWAY #7 CYCLIC SHIFT DETECTION

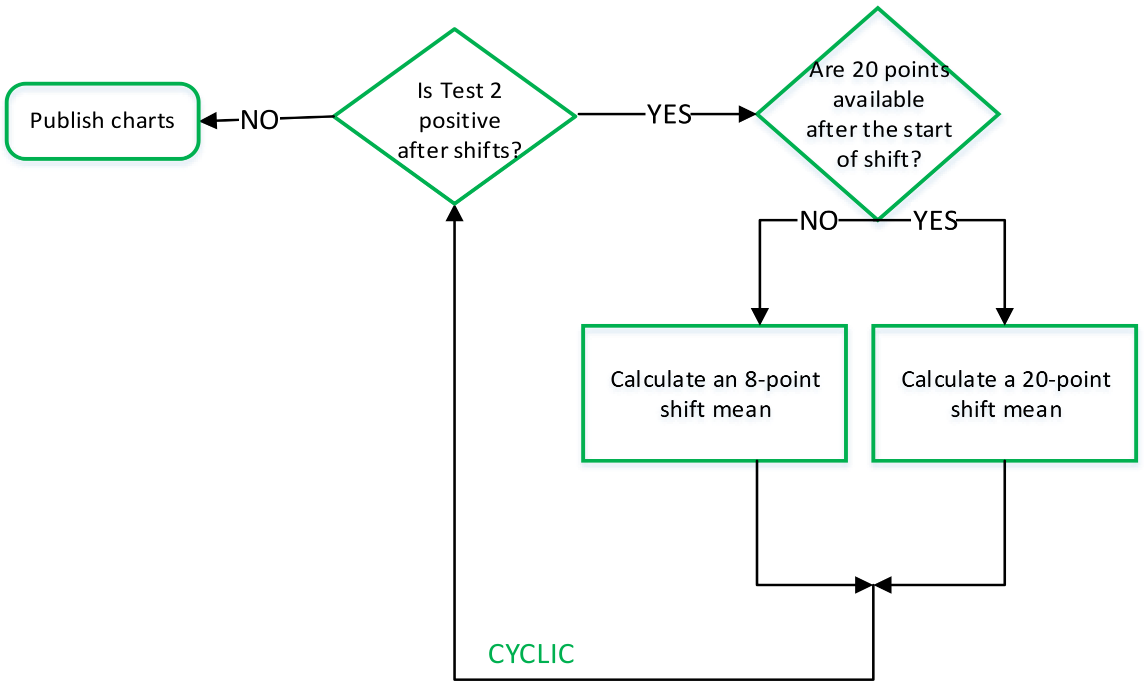

Publish charts
NO
Is Test 2 positive after shifts?
YES
Are 20 points available after the start of shift?
NO
YES
Calculate an 8-point shift mean
Calculate a 20-point shift mean
CYCLIC

Figure 7. Pathway #7 publishes a chart after iterative shift detection is complete

After the first shift is detected, detecting any additional shifts follows the same algorithm. First of all, as all other pathways with a shift, we filter the metrics with the same first condition. The difference is that the shift detection after the first one is cyclic until there are no more shifts, so a DO loop is essential to complete the process:

```
%global i;
%do i= 1 %to 10; *note: the number is here meant to be the anticipated shift
                  number;
```

For each iteration the system first checks if the previous shift exists by checking the status of the OUTTABLE data set from PROC SHEWHART:

```
%let indat = outtable_from_shift &i.&run;
%if %SYSFUNC(EXIST(&indat)) %then %do;

proc sql noprint;
       select COUNT(*) into: check
       from outtable_from_shift &i.&run;
quit;

%if (&check) %then %do;
```

The next shift after &i is &seq_shift shift and the 8th point of the new shift temporal unit is converted into a macro variable:

```
%let seq_shif=%EVAL(&i+1);
```

```
proc sql  noprint ;
       select &time_unit
       into: shift&seq_shif&run
       from outtable_from_shift &i.&run (obs=1)
       where INDEX( _tests_,'2')
              and rec gt &&shift_&i._end&run;
quit;

%if %symexist(shift&seq_shif&run) %then  %do;
```

The challenge in this cyclic shift detection is that the number of shifts for each metric is unpredictable, so accordingly the phase definition before PROC SHEWHART varies in line with the number of shifts. The manipulation of center line values is also dependent on the number of the shifts.

While it is simple to define the phase for the last shift (&i) and next shift (&seq_shif) shift, the tricky part is defining the phases of previous shifts. In order to dynamically define the shifts, a DO loop within the input data set is used until it reaches back to the first shift. The process is controlled by the condition that &assign_shif is greater than or equal to 1. When &assign_shif gets to 1, it starts to define the first shift phase.

Below is one of examples that show how the macro variables involved in shift phase definition are defined:

```
proc sql noprint;
       select catt("'", &time_unit, "'")
       into :shift&Seq_shif._extend&run separated by ', '
       from outtable_from_shift &i.&run
       where rec ge  &&shift&Seq_shif._start&run;
quit;
```

The code above selects all the temporal units starting at the first point of the next shift until the end of temporal unit sequence into a macro variable called &&shift&seq_shif._extend&run, so &&shift&Seq_shif._extend&run includes all the shift &Seq_shif temporal units and the units after. Similarly &&shift&i._extend&run has all the temporal units starting at the first point of shift &i until the end, therefore, phase &i includes all the temporal units that are in &&shift&i._extend&run, but not &&shift&seq_shif._extend&run.The same logic goes to the previous phase definitions.

For example, if a metric has 5 shifts, &I = 4 and &seq_shift = 5, so defining the fifth and fourth shift is accomplished by the code in the first rectangle. After excluding (ELSE IF) all the temporal units in either shift 4 or 5, The DO loop in the second rectangle defines the previous phases. First j = 1 and then &assign_shift = 3, so if the temporal units are on the list of &&shift3_extend&run, but not &&shift4_extend&run, the units are labeled as shift 3. Before the DO loop goes to the next cycle, j increases to 2 by the statement of %let j = %EVAL (&j+1), so now &assign_shift is equal to 2 and it is still bigger than 1, therefore, the DO loop continues. For all the temporal units that are not in shift 5, shift4 or shift 3, If they are on the list of &&shift2_extend&run, the units are labeled as shift 2. The loop continues until &assign_shift gets to 1, which defines the last shift. The similar cyclic flow follows in converting center line means into macro variables and assigning the values to the limit data sets:

```
data input_data&run;
       set input_data&run;
              format _phase_ $20.;
       %let j=1;
       %let assign_shif=%EVAL(&i-&j);

              if &time_unit in (&&shift&seq_shif._extend&run)
              then _phase_="shift&seq_shif";
              else if &time_unit in (&&shift&Seq_shif._extend&run)
                      and &time_unit not in (&&shift&seq_shif._extend&run)
              then _phase_="shift&i";
```

```
%do %while (&assign_shif>=1);
       else if &time_unit in (&&shift&assign_shif._extend&run)
       and &time_unit not in shift&i._phase_extend&run
       then _phase_="shift&assign_shif";
       %let j=%EVAL(&j+1);
       %let assign_shif=%EVAL(&i-&j);
%end;
```

```
       else _phase_='baseline';
       label &time_unit="&unit";
run;
```

Convert the center line mean from each phase into a macro variable:

```
%macro extract_shift;
       %let j=1;
       %let assign_shif=%EVAL(&i-&j);

       %do %while (&assign_shif>=1);
       %global shift&assign_shif&run;

              proc sql  noprint ;
                     select _p_
                     into: shift&assign_shif&run
                     from outt_shift&assign_shif._&run (obs=1);
              quit;

       %let j=%EVAL(&j+1);
       %let assign_shif=%EVAL(&i-&j);
       %put &assign_shif;
       %end;
%mend;
%extract_shift;
```

Replace each phase mean in the limit data set with the converted the macro variable:

```
data limits&run;
       set limits&run;
              base_p_=&&centerline&run/100;
       %let j=1;
       %let assign_shif=%EVAL(&i-&j);

       %do %while (&assign_shif>=1);
              s&assign_shif._p_=&&shift&assign_shif&run/100;
              %let j=%EVAL(&j+1);
              %let assign_shif=%EVAL(&i-&j);
       %end;

              s&i._p_=&&shift&i&run/100;
              s&seq_shif._p_=&&shift&seq_shif&run/100;


              if _index_="baseline" then _p_=_base_p_;

       %let j=1;
       %let assign_shif=%EVAL(&i-&j);
       %do %while (&assign_shif>=1);
              if _index_="shift&assign_shif" then _p_=s&assign_shif._p_;
              %let j=%EVAL(&j+1);
              %let assign_shif=%EVAL(&i-&j);
       %end;

              if _index_="shift&i" then _p_=s&i._p_;
```

```
            if _index_="shift&seq_shif" then _p_=s&seq_shif._p_;
run;
```

COMPLETE THE AUTOMATION PROCESS

A %DO loop is to execute a section of a macro repetitively based on the value of an index. With the %DO loop the pathways run through every metric in a project to produce a SPC chart image file

```
 %do run= 1 %to 14 * 14 is the total number of metrics;
```

Finally, the SAS programs that implement the automation process are integrated into data preparation steps with %INCLUDE statements.

CONCLUSION

SAS Macro processing and %DO loops are powerful tools in automation process. The system developed in this paper uses macro variables, macro functions, %IF - %THEN/%ELSE logic and DO loops to complete the automation process. Below are the key actions driving the process.

- Create and use macro variables from a static parameter file with general information for the metrics.
- Fetch the variables for a SPC chart, and convert them into macro variables.
- Capture signals dynamically during the execution of the system and convert them to macro variables.
- Pave pathways with macro variables, macro functions and %IF - %THEN/%ELSE logic to implement the flow chart algorithms.
- Complete the automation processes by using DO loops and integrating the programs into data preparation steps.

The automation process is developed in the setting of healthcare, but in light of the similarity of components involved in SAS/QC automation, it also sets an example for the production of SPC charts in manufacturing.

REFERENCES

Provost, L. P. and Murray, S. K. 2011.*The Health Care Data Guide: Learning from Data for Improvement*. San Francisco, CA: John Wiley & Sons.

Childress, S. and Welch, B. 2011. *Three Easy Ways around Nonexistent or Empty Datasets*, presented at SouthEast SAS User Group 2011 Conference in Alexandria, Virginia (Paper CC-19).

Ransdell, B. 2016. *Improving Health Care Quality with the RAREEVENTS Procedure,* presented at the SAS Global Forum 2016 Conference in Las Vegas, Nevada (paper SAS4040-2016).

RECOMMENDED READING

- *SAS /QC® 14.1 User's Guide*

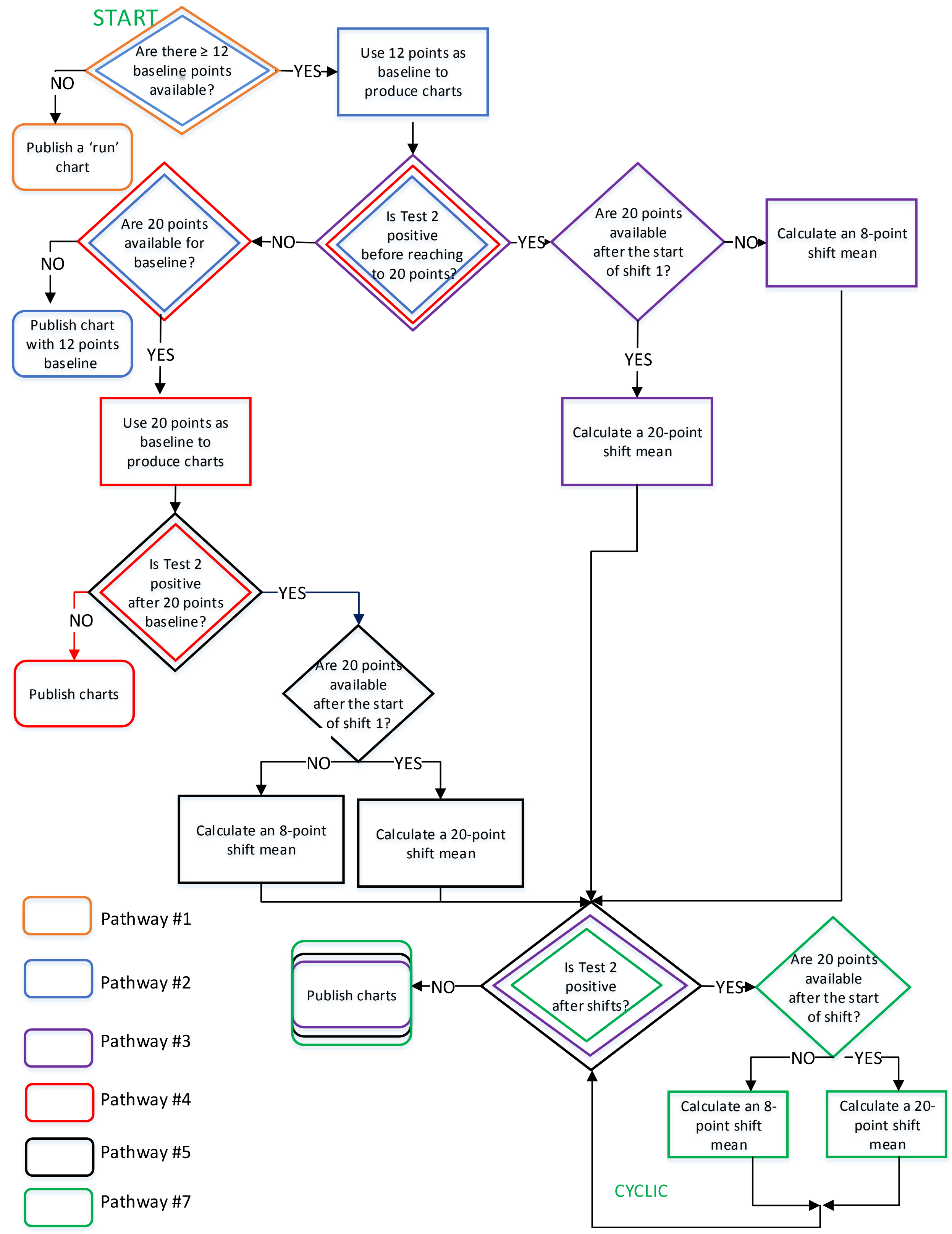
CONTACT INFORMATION

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

Lynn (Xiaohong) Liu
Ann & Robert H. Lurie Children's Hospital of Chicago
xlliu@luriechildrens.org

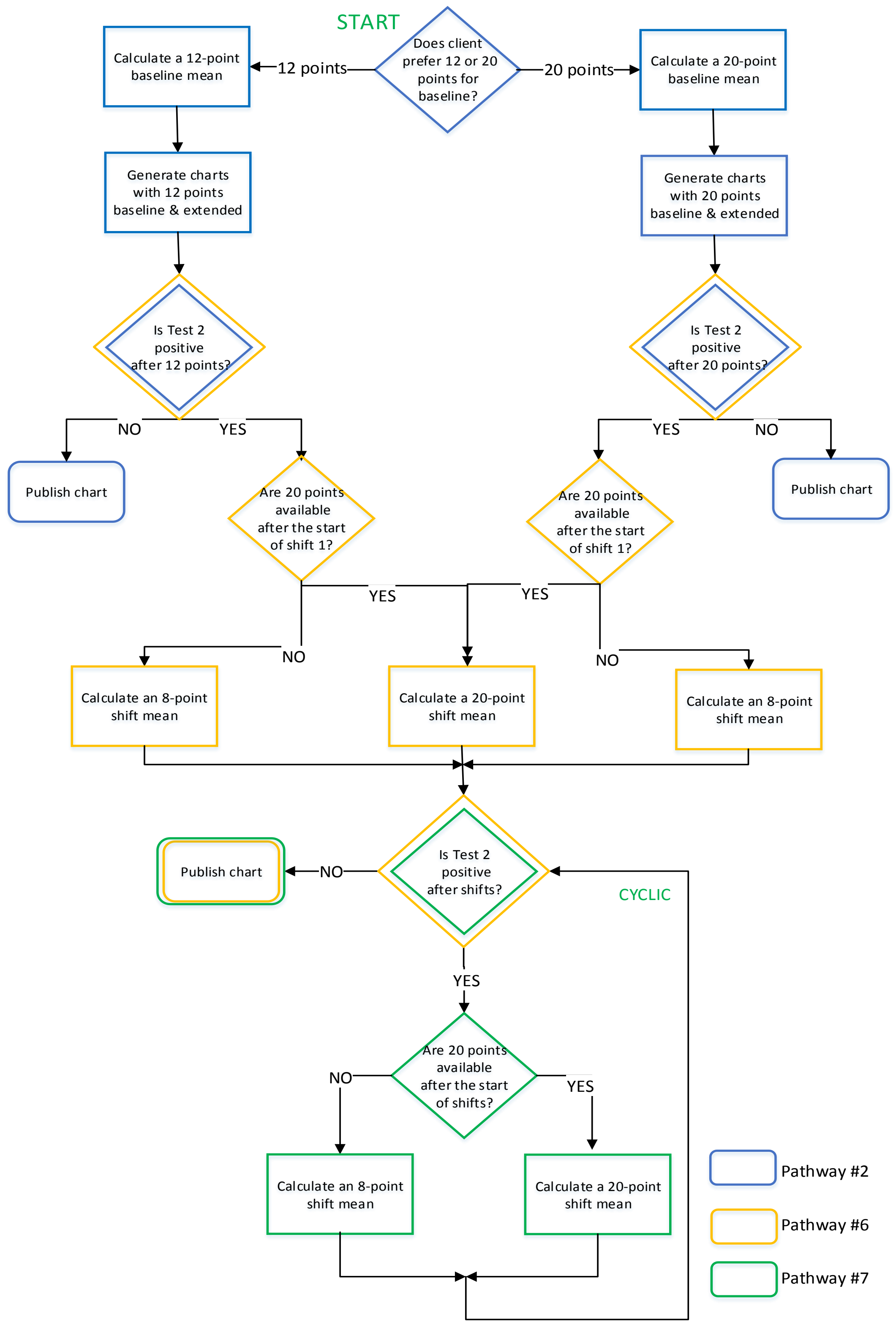
Appendix 1

When clients have no specific request on baseline points


START
Are there ≥ 12 baseline points available?
YES
NO
Use 12 points as baseline to produce charts
Publish a 'run' chart
Is Test 2 positive before reaching to 20 points?
NO
YES
Are 20 points available for baseline?
NO
YES
Publish chart with 12 points baseline
Use 20 points as baseline to produce charts
Are 20 points available after the start of shift 1?
NO
YES
Calculate an 8-point shift mean
Calculate a 20-point shift mean
Is Test 2 positive after 20 points baseline?
NO
YES
Publish charts
Are 20 points available after the start of shift 1?
NO
YES
Calculate an 8-point shift mean
Calculate a 20-point shift mean
Pathway #1
Pathway #2
Pathway #3
Pathway #4
Pathway #5
Pathway #7
Publish charts
NO
Is Test 2 positive after shifts?
YES
Are 20 points available after the start of shift?
NO
YES
Calculate an 8-point shift mean
Calculate a 20-point shift mean
CYCLIC

Appendix 2

When clients have a specific request on baseline points


START
Does client prefer 12 or 20 points for baseline?
12 points
20 points
Calculate a 12-point baseline mean
Calculate a 20-point baseline mean
Generate charts with 12 points baseline & extended
Generate charts with 20 points baseline & extended
Is Test 2 positive after 12 points?
Is Test 2 positive after 20 points?
NO
YES
YES
NO
Publish chart
Publish chart
Are 20 points available after the start of shift 1?
Are 20 points available after the start of shift 1?
YES
YES
NO
NO
Calculate an 8-point shift mean
Calculate a 20-point shift mean
Calculate an 8-point shift mean
Is Test 2 positive after shifts?
NO
Publish chart
CYCLIC
YES
Are 20 points available after the start of shifts?
NO
YES
Calculate an 8-point shift mean
Calculate a 20-point shift mean
Pathway #2
Pathway #6
Pathway #7