SUBGROUPING AND SUBSETTING BEFORE INVOKING THE SHEWHART PROCEDURE

Mark A. Sobolai
Wheeling-Pittsburgh Steel Corporation

I. BACKGROUND AND INTRODUCTION

SAS* software was licensed at Wheeling-Pittsburgh Steel Corp. (WPSC) primarily for quality control work. The major focus of this was to be Statistical Process Control (SPC) implementation. The reasons for choosing Base SAS* and SAS/QC software were several, and were largely dictated by our particular situation in 1986.

- A wide network of remote terminals and printers already existed which were tied to our corporate mainframe computer. At the time, there were very few personal computers in these locations.
- A significant database was stored on tape and/or disk at the computer center. This included such things as order specifications, rejection and claims data, and production records.
- Our plants are vertically integrated. Each of our finishing plants gets 100% of its semi-finished steel from other WPSC plants. Thus there was a need to share data.
- The amount of data we were collecting was enormous, and we were only scratching the surface. Several process control systems were in planning stages.
- The Quality Control staffs had no experience in computer operations, and didn't want the additional responsibility of maintaining a separate data collection system. By installing SAS software on the mainframe that responsibility remained at the computer center.

We were given access to SAS/QC when it became available for beta test in 1986. We were quickly sold on it. The commands are fairly simple. A Proc Shewhart statement and a Chart statement produces a control chart with centerlines, limits, and automatically scaled. The package can do any type control chart a creative quality engineer might want. The options available with the Shewhart procedure allow a great deal of versatility. Also, the line printer generated output is adequate and quite suitable for plant situations.

Another advantage of SAS/QC software is that it is combined with the data management power of Base SAS software. Manipulating raw data before and after using the SAS/QC procedures gives the quality engineer the ability to customize data input and output to his exact needs. While we at WPSC appreciated this power, it quickly became apparent that we needed to enhance our programming skills.

Two programming tasks usually required when building an SPC application with SAS/QC are subgrouping the observations and subsetting the database. There are several subroutines I have found useful in this regard. There is nothing unique in them. The important point is that they have worked and helped better utilize an already good tool, SAS/QC software.

As the title of this paper indicates, the examples and explanations discussed herein refer particularly to use of the Shewhart procedure. This is where we have concentrated our efforts. Moreover, our operations generally require use of variables data. However, I believe that the programming concepts can be extended to use with other Proc's within the SAS/QC repertoire and also to attributes data.

II. SUBGROUPING

One of the preliminary decisions the quality engineer must make when designing a control chart application is the subgroup size. Determining a rational subgroup is supposed to be based on theoretical sensitivity and process knowledge. In real world situations several other factors are also involved. These include such things as how many samples are economically possible, the workload of the inspector or operator, and past practices. Certainly, the quality engineer doesn't want a software package that imposes further restrictions.

It's necessary to understand how Proc Shewhart reads data. For variables charts there are two ways that a SAS dataset can be structured for recognition by Proc Shewhart.

- For raw data to be read the dataset must be arranged as below. Two variables are necessary: one that identifies the subgroup to which the measurement belongs and the result of the measurement or test. The Proc Shewhart procedure allows a great deal of versatility. Also, the line printer generated output is adequate and quite suitable for plant situations.

Dataset example:

<table>
<thead>
<tr>
<th>Group</th>
<th>Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.1</td>
</tr>
<tr>
<td>1</td>
<td>1.2</td>
</tr>
</tbody>
</table>

- For subgrouped data the dataset must be handled differently. Two variables are necessary: one that identifies the subgroup to which the measurement belongs and the result of the measurement or test. The Proc Shewhart statement should identify the SAS dataset with the 'data=' option.
SAS statements:
```
proc shewhart data=datl;
  xchart measure*sample;
```

Summary data which is the result of doing calculations for subgroup average (x̄), range (r), standard deviation (s), and size (n) can also be handled. The SAS dataset must be of form as below. The variable names for subgroup statistics must be the same name plus the proper letter suffix (x, n, s, r). The suffix identifies the statistic to Proc Shewhart. In addition, the dataset must be referenced with the "histoty=" option.

Dataset example: datl
```
sample 1 2 2 2
measure 1.3 1.6 1.4 1.8
```

SAS statements:
```
proc shewhart history=datl;
  xchart sample*measure;
```

An obvious way to input your SPC data is in one of these two layouts. However, the first method requires that the subgroup be specified for each measurement. The second method requires preliminary calculation. Many times this is not possible nor desirable. Many alternatives are available with use of base SAS file manipulations.

EXAMPLE 1. Assigning Numbered Subgroups

If the measurement data for a particular characteristic is input in desired order for control charting, it is possible to assign a subgroup number in the data step. The statement "count+1;" creates a variable that is incremented by 1 (one) each time a record is processed. Subgroups of any predetermined size n can then be assigned by adding n-1 to variable "count", dividing that sum by n and using the "int" function to return the integer part of the quotient.

```
Input:
```
```
    21.86
    25.36
    24.67
    24.35
    24.24
```
```
Program:
data datl;
  input measure;
  count+1;
  subgroup=int{(count+Z}/3);
  proc shewhart data=datl;
  xchart measure*subgroup;
```

The above example Data step assigned the same numeric subgroup variable to successive groups of 3(three) measurements.

EXAMPLE 2. Subgroup Measurements Input on a Single Record

Use of SAS statistical functions is possible when the measurements pertaining to a subgroup are input in a string on a single record(line). The advantage of the SAS functions is the way they ignore missing data and return statistics calculated on only nonmissing values. Using variable names with proper suffix allows the resulting SAS dataset to be read by a Proc Shewhart statement with the "history=" option.

```
Input:
```
```
  2057 2 5 8 2 7
  2085 6 4 2 4 8 5
  2087 6 7 4 2 1 5
```
```
Program:
data datl:
  infile input missover;
  input lot $ measure;
  sample+1;
  measurex=mean(of measure);
  measures=std(of measure);
  measurern=of measure);
  drop measure;
  proc shewhart history=datl;
  xchart measure*sample=
```

A nice feature of this program is that it handles varying sample sizes. The "missover" option in the infile statement causes non-existent measurements to be set to missing values, which the SAS functions then ignore. A numeric ordinal subgroup variable, "sample", is created and used to identify the order in which the lots were measured (or manufactured, received, etc.).

EXAMPLE 3. Editing and Numbering Subgroups

Computers don't solve the problem of bad or missing data. Mechanized data systems force the user to decide ahead of time how such situations are to be addressed. The example program below is set up to skip over missing data, and to separate and highlight out-of-range data by outputting it to another dataset. The logic to verify the input is repeated for two variables. Because consecutive subgroups are assigned only when data is in range, the two variables, yield and tensile, are grouped slightly different. Thus two chart statements are used after the Proc Shewhart statement.

```
Input:
```
```
  081387 081387
  081587 082487
  082587
```
```
Program:
data datl errorl;
  infile input missover;
  input lot $ measure;
  sample+1;
  check=of measure;
  error=of measure;
  sample=
  proc shewhart history=datl;
  xchart measure*sample=
```

The result is a SAS dataset of form that can be read by Proc Shewhart with "data=" option.
**EXAMPLE 4. Using Character Variables for Subgroups**

With Proc Shewhart, subgroup variables can be either numeric or character. When using numeric subgroups, the data must be sorted in ascending subgroup order. When control charts are generated with numeric subgroups, the horizontal axis is scaled over the range of observed subgroup values. Many times subgroups are not defined by a numeric value. Unit identifications are often used, such as lot no., batch no., or coil no. These often contain alpha characters or blanks. In addition, there are times when scaling of the horizontal axis over the range of actual subgroup nos. doesn’t make sense, as when there are gaps in the data. Other times, the quality engineer may want to reorder the subgroups, perhaps to study possible trends caused by a previous or subsequent process.

In such cases, the subgroup variable should be defined as a character variable. This can be done in the Input statement, or by converting numeric data with the put function.

**Proc Shewhart charts character subgroups in the sequence they appear in the dataset. Thus, it is necessary that all data for each subgroup be back to back in the dataset. Character subgroups are scaled uniformly so that the appearance of a chart depends on the number of subgroups.**

**EXAMPLE 5. Converting Date & Shifts to SAS Datetimes**

Date and shift designations are often used to define subgroups. The previous example showed one way to handle this. It also pointed out some shortcomings of that method. Another way to construct control charts in time order using date and shift input data is to assign pseudotimes to each shift. These can then be output with SAS datetime formats to make an easily interpretable chart.

When a variable is identified as a SAS date, it is converted to a numeric value representing the number of days since January 1,1960. For example, 09/01/87 is 10,048. Similarly, if a date and time is identified as a SAS datetime variable, it is converted to the number of seconds since 12 midnight January 1,1960.

In the example below, a date is input and identified as a SAS date thru an Informat statement. Each day has three shifts: A, B, and C. This variable is also input. A new variable named “time” is created as a numeric which will be given a SAS datetime format for printing purposes. For ‘A’ shifts (midnight to 8 am), the variable ‘time’ is determined by multiplying that days SAS date representation by 86400. As a result ‘time’ is equal to the number of seconds from January 1,1960 to the beginning of that shift. For ‘B’ shift which start at 8 am, the logic is similar. In this case the pseudotime variable “time” is equal to the SAS date multiplied by 86400 plus 28800 which is the number of seconds in 8 hours. Likewise the ‘C’ shift begins 57600 seconds after midnight.

The Proc Shewhart output is labeled with SAS datetimes across the subgroup axis (horizontal axis). Since the variable “time” is actually a numeric, the scaling is done over the range of times. As a result, if shifts are missing (not worked) the scaling is unaffected. There is simply...
no point plotted in that space.

<table>
<thead>
<tr>
<th>Input1</th>
<th>090187 A</th>
<th>0.14</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>090187 A</td>
<td>0.37</td>
</tr>
<tr>
<td></td>
<td>090187 A</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>090187 A</td>
<td>-0.17</td>
</tr>
<tr>
<td></td>
<td>090187 A</td>
<td>0.14</td>
</tr>
<tr>
<td></td>
<td>090187 B</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>090187 B</td>
<td>-0.15</td>
</tr>
</tbody>
</table>

Program:

```sas
data dat1;
  infile input1;
  informat date mmddyy6.:
  format date mmddyy6.:
  format time datetime7.:
  input date shift $ measure;
  if shift='a' then time=(date*86400);
  else if shift='b' then time=(date*86400)+28800;
  else if shift='c' then time=(date*86400)+57600;
  proc sort data=dat1;
  by time;
  proc shewhart data=dat1;
  xchart measure*time='o'/ npanel=-8 repeat;
```

The `npanel=` option after the chart statement can be used to control the number of days on a page. In the example, `-8` causes exactly 8 days to be scaled on each page, which is actually 7 days data. In cases where more than one page or panel will be output, the `repeat` option is necessary when using this type of subgrouping.

EXAMPLE 6. Combining Date, Shift and Sequence Logic

The program below is another example of using pseudotimes. In this case, an individual's moving range chart is needed. During a normal shift a measurement is made every two hours that equipment operates. When equipment delays occur the number of measurements for that shift is fewer. The data is input in time order.

After the initial Data step to read in the raw data, a second Data step is used with a `By` statement. This creates the variables "first.date" and "first.turn" which are used to reset a sequencing variable, "seq" to "0" whenever a shift change occurs. If/then logic is used to create a numeric variable that corresponds to the character variable "turn".

```sas
data dat1;
  informat date mmddyy6.:
  input date turn $ gauge;
  data dat2;
  set dat1;
```

by date turn;
  if turn='a' then turnl=0;
  else if turn='b' then turnl=1;
  else if turn='c' then turnl=2;
  if first.date or first.turn then seq=0;
  else seq+1;
  time=(date*86400)+(turnl*28800)+ (seq*7200);
  format time datetime7.;
  proc shewhart data=dat2;
  xchart gauge*time='*' noconnect repeat;
```

It is important that the programmer knows the maximum number of measurements per shift in this example. This is what determines the multiplier for variable "seq". In the example, up to four(4) measurements are taken. Dividing 28800 by 4 sets the pseudotime interval at 7200 seconds between samples.

III. SUBSETTING

There are two general reasons for subsetting the database when using Proc Shewhart for SPC applications. First, it is often necessary to subset data for certain parts, or operators or raw materials. The other reason for subsetting is to limit the amount of data to be displayed at a given time.

There are no statements or options that can be contained in the Proc Shewhart step to accomplish the subsetting tasks. However, the myriad of SAS functions, formats, and logic statements available in the Data step allow the quality engineer the means to structure the data to fit his needs.

The examples that follow deal with subsetting to control the amount of data being displayed at a particular time. A rule of thumb that is often followed is that an ongoing control chart not display more than one page's worth of points. About 30 characters of output linesize are required for labeling of vertical axis and control limits. This leaves space for 100 points when printing on normal 14" wide paper. However, Proc Shewhart scales numeric subgroups by round units—2, 5 or 10. The net result is that a chart should be planned for a maximum of 90 to 95 points. If more points are forced onto a page than it can logically print, some "ticks" may have two points plotted over them.

EXAMPLE 7. Subsetting to Keep the Last X Days Data

Most sampling plans are consistent in the amount of data collected over a given time period. Thus a natural solution to controlling the subsetting of data for output purposes is to key on the date a measurement was made.
In the sample program below the variable "today" is assigned the current SAS date (no. of days since January 1, 1960). The input variable "date" is also a SAS date. The expression "(today-date)" is then the number of days duration since a particular observation was made. By deleting all observations for which this duration is greater than 7, a subset of recent data is created.

EXAMPLE 8. Subsetting Previous Calendar Week Plus Current

Rather than display only the last so many days, it is sometimes desirable to keep the entire previous week's data. In this case, the "intck" function can be used to count calendar weeks. In the example below the variable "thisweek" is the number of calendar weeks between January 1, 1987 and the current date. Similarly, the variable "dataweek" is the number of calendar weeks between January 1, 1987 and the date of the observation. It follows that the expression "(thisweek-dataweek)" is the number of calendar weeks between the current date and when data collected. By deleting all observations for which this expression is greater than 1, only the previous and current week's data is processed.

```
input date turn $ coilid $ reflect;
infile input1;
informat date mmddyy6.;
if turn='A' then time=(date*86400);
else if turn='B' then time=(date*86400)+28800;
else if turn='C' then time=(date*86400)+57600;
format time datetime7.;
proc shewhart data=dat1;
xrchart reflect*time='0'/npanel=-9;
```

Note that the control limits are calculated based only on the data for past 7 days, since pre-established limits were not referenced. The "npanel=" option is set so that up to 14 days worth of data is put on one page. Also, in this example a known mean and standard deviations are specified. Proc Shewhart calculates control limits using these values and adjusts for subgroup size.

EXAMPLE 9. Subsetting by Last X points

Another common situation is to plot a specific number of points each time the chart is generated. This approach is workable when the subgroup variable is character or an ordinal numeric. It is not as clean a solution when using date or date-time subgroups.

The sample program below was accomplished by the subsetting without doing several sorts. This is a real plus when working with large databases. It's worthy to acknowledge that we received some help from the staff at SAS Institute, Inc. on this. They serve as a nice backup resource.

Unlike the previous examples, two Data steps are used to accomplish the subsetting. In the first Data step, raw data is read and a variable "subgroup" is assigned. The second Data step creates a subset of the first dataset by using a Set statement. Two "setoption" variables are specified that are key to the subsetting logic. The variable "x" is defined by the "nobs=" option and is equal to total number of observations in "dat1". The variable "n" is the record number (or line number). The Set statement is processed iteratively by the Do statement. For example, if there were 100 observations in "dat1" then x = 100. The Do statement will process observations (100-49) to 100. Thus the dataset "dat2" contains the last 50 observations. In the
Proc Shewhart statement "dat2" is then referenced.

0 Input: 032588 A 67.3  
032788 A 66.4  
032788 C 71.5  
032788 C 68.5  
032688 A 69.3  
032888 A 70.2

0 Program:
data dat1;
  infile input1;
  informat date mmddyy6.;
  input date turn %measure;
  count+1;
  subgroup=int((count+2)/3);
data dat2;
do n=(x-49) to x;
  set dat1 point=n nobs=x;
  output;
  if _error_ then abort;
end;
stop;
proc shewhart data=dat2;
xrchart measure*subgroup/ noconnect;

This example assumes that the input data is in proper sequence. If not, a sort is required before assigning subgroup numbers. The statement "if _error_ then abort;" is to prevent an error causing an endless loop.

IV. CONCLUSION

While the examples presented in this discussion have been simplified for the sake of presentation, the concepts are easily applied to larger databases and more detailed analyses. Their purpose was to demonstrate the flexibility of the Base SAS and SAS/QC software combination.

An interesting question that needs to be addressed is how much software know-how does a quality engineer need? Does he have to be a programmer and/or analyst also? This writer's opinion is that he needs all the software know-how he can absorb. Practical situations dictate what an engineer should "know" versus what he can reference. In cases where SPC implementation is still evolving, the quality engineer needs to be able to adapt. Products change, processes change, standards change and more significantly, our understanding of our processes changes. SPC computer programs must be changed accordingly. Reliance on outside services for programming leaves the quality engineer quite vulnerable. How do you cost justify these changes? More importantly, do you have the time to write cost justifications? Why not do it within your department and move on to the next project!

This is the approach we've taken at WPSC. SAS/QC software is a user's tool. The QA and Operations staff "own" it and use it. We've had good success. We have user designed and written programs working in every plant and almost every department.

The key word is "working". In most cases, the program was written by the person who uses it. He wrote it to help him do his job.

Whether a program is sophisticated or crude is not important. Does it work? Does it make someone's task easier? Does it help improve the process?

Much of our growth in using SPC has come thru computerizing the applications. Our growth in computerizing has come from sharing ideas in the use of SAS/QC software. The subgrouping and subsetting techniques discussed in the several examples have been found to work, so one person passed them on to another. If one of them can be helpful to you, please use it.

SAS, Base SAS and SAS/QC are registered trademarks of SAS Institute, Inc., Cary, NC, USA.