Paper 140-27

Visualizing Patterns with Scrollable Web Graphics

Eric Brinsfield, Meridian Software, Inc., Raleigh, NC
Caroline Bahler, Meridian Software, Inc., Raleigh, NC

ABSTRACT

By combining the power of SAS with the visual clarity of HTML tables, you can create scrollable Web graphics that illuminate your data. Although advanced analytical methods can be used to detect patterns or anomalies in data, some users prefer to or need to visually scan the raw data directly. This paper discusses a technique used to present spectral data with color graphics over the Web. The scrollable charts are both fast and easy to use and to implement.

The method uses base SAS, SAS/Intrnet, SAS/Access, HTML, JavaScript, and Microsoft's Active Server Pages (ASP) with VBScript and ADO. We will demonstrate one proven implementation of using this approach and propose additional uses and applications, such as scrollable comparative histograms.

INTRODUCTION

Anytime you create a graph, your next wish is usually to compare that graph to a similar graph for a different time period or for a different population. If you try to compare more than a few true graphics on the same page, you quickly discover that

- you can only squeeze so many graphs on a page, so comparison is still difficult
- the response time for loading graphic pages is painfully long
- you cannot process all of the information provided in detailed graphs due to the level of random noise.

To address these issues, but still satisfy those users who really want to compare unlimited numbers of samples or subsets of data across a sequence of positions or subgroupings, we devised a method of using SAS to generate HTML tables that present data in a scrollable and graphic display.

In the following discussion, we define a case study that provides an example of using this technique and we explain the details of the implementation. Finally, we also discuss how our technique can be applied to a wide variety of situations.

USER REQUIREMENTS AND OBJECTIVES

DESCRIPTION OF CASE STUDY

Our case study and demonstration addresses a need within a manufacturing environment. Our client monitors several types of defects during their manufacturing process. Three of the defect types are monitored automatically and continuously. In addition, several are collected manually at predetermined intervals. The process involves the treatment of long rolls of film that are ultimately sliced lengthwise into 82 different strips. From a manufacturing and quality perspective, those 82 strips are tracked separately from beginning to end regardless of whether they have been sliced or not.

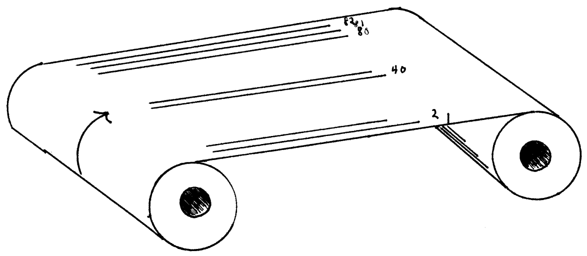

Figure 1: Roll of Film

Historically, technicians maintained large sheets of graph paper (about the size of a desktop) that allowed them to track the occurrence of defects on each roll. Each row across the page represented a roll of film and each column represented one of the 82 strips. They color-coded each cell depending on the type of defect, if one existed. They liked this method, because it provided a visual method to detect patterns over time or on specific strips or groups of strips.

The technicians did not want to lose this tool, so when we were asked to automate their quality analysis process, we had to devise a way to provide the same functionality with (at least) the same ease of use they experienced today. Our objective was to make it easier.

SPECIFIC REQUIREMENTS

In order to accept an online version of their color-coded inspection table, the technicians required the following capabilities:

- Color coding based on the type of defect
- Ability to scroll back in time to compare each roll to the roll processed before or after it.
- Ability to go back in time historically to look at older rolls (2 to 3 months)
- Ability to drill-down on details about any suspicious rolls
- Ability to see all 82 strips at the same time
- Ability to limit view to selected defect types
- Easy or automatic data entry
- Real-time access to all data in the color coded table
- Quick response time, because they found it rather easy to look at their paper now.

SOLUTION

In our first version of the quality system, we utilized SAS/AF sitting on top of Microsoft SQL Server, which we accessed with SAS/Access. We wrestled with several possible solutions for squeezing 82 columns of information onto one screen with room left for labeling information. Ultimately, we settled on using SAS/Graph to create the color-coding and to fit the information in the allotted space. We decided to utilize the DATA step graphics interface, which provided all of the capabilities required, except that the performance was somewhat sluggish.

After running in production for several months and moving into a different phase of the project, we decided to convert many of the analytical features of the system over to the Web. Based on user feedback and the fact that we could now use Web tools in the project, we decided to re-evaluate the technicians' inspection table.

Our new solution used SAS to extract the necessary data from SQL Server and process the results, which were then displayed in a browser using HTML tables, where each cell in the table was color-coded to provide the color graphic impact. The built-in functionality of the browser provided automatic sizing and quick and easy scrolling through the rows.

By default, the system displays the most recent 3 days worth of rolls. So, users can open the browser window and quickly view the status of the system. They can sit in the window and easily refresh the data with new rolls as they are processed. Using a control panel at the bottom of the screen, users can filter defects or change date ranges to access more data or older data.

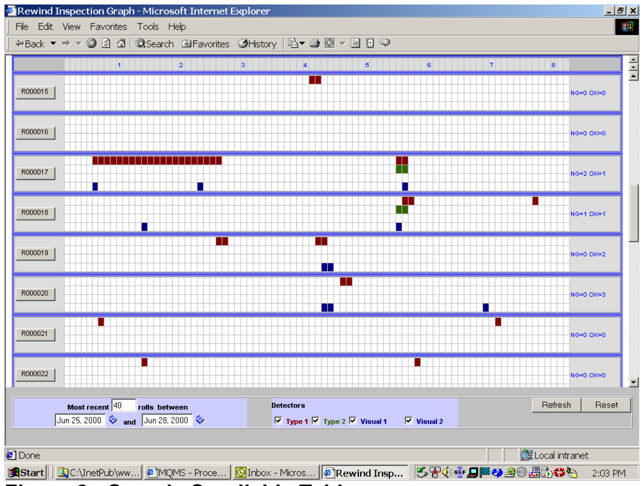
Rewind Inspection Graph - Microsoft Internet Explorer
R000015
R000016
R000017
R000018
R000019
R000020
R000021
R000022
Most recent 40 rolls between Jun 25, 2000 and Jun 28, 2000
Detectors
Type 1
Type 2
Visual 1
Visual 2
Refresh
Reset

Figure 2: Sample Scrollable Table

Note that different defects are shown in different rows within a cluster of rolls. In other words, if all defects are displayed, each roll is associated with four (4) individual rows in the table. A thick blue line demarks each roll and cluster of 4 rows.

METHODS

TECHNOLOGIES

We used the following technologies to implement the inspection Web page:

- SAS/Access to read data from SQL Server
- Base SAS to process the data and prepare the results
- SAS/Intrnet, to pass the data back to the browser
- Active Server Pages (ASP), a Microsoft Internet Information Server technology that utilizes VBScript and ADO to access SQL Server directly and generate HTML
- JavaScript and HTML

COMPONENTS

The inspection Web page is comprised of three discrete sections:

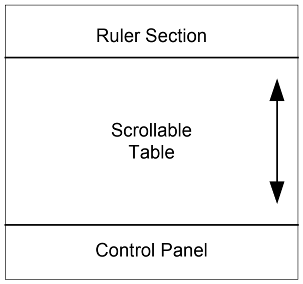

Figure 3: Web Page Components

- Section 1 is a fixed “ruler” or axis bar that marks off and labels each of the 82 strips. This section does not move during scrolling.
- Section 2 is a scrollable table that contains the color-coded defect indicators
- Section 3 is a control panel that contains buttons and filter controls that communicate with section 2.

In our example, we utilize the following specific components:

- INSPECTFRAME.ASP – an active server page
- RULER.HTM – the static HTML that creates the ruler at the top of the page
- INSPECT.SAS – a SAS program that is submitted via the application broker to read the data, build the results, and return the formatted results to the Web page.
- INSPECTMENU.ASP – an active server page that creates the control menu at the bottom of the page.

THE FRAME (THE FRAMEWORK)

INSPECTFRAME.ASP is an ASP file that utilizes VBScript to generate the HTML <FRAMESET> and <FRAME> tags. The <FRAMESET> tag was used to divide the browser window into the three (3) sections. <FRAME> tags were used to define which file to display within sections 1 and 3. The <FRAME> tag used to define section 2 calls the application broker and runs INSPECTANAL.SAS returning to the Web browser the HTML generated.

THE RULER (THE COLUMN LABELS)

RULER.HTM is a HTML file created using a HTML editor. It is a static page that will not change. Its purpose is to be the header for the inspection graph that is positioned directly below it. This page was created and placed within the top section so that when the user scrolled up and down in the actual graph the header was always present.

THE GRAPH (THE SCROLLABLE TABLE)

INSPECT.SAS creates the scrollable inspection graph. This program reads data from SQL Server, using SAS/Access, prepares the data, and uses a DATA _NULL_ to create HTML and write it to _WEBOUT.

The dynamic HTML file created is essentially a table that contains a colored cell where a defect occurred. If no defect occurred then the cell is white. A dynamic HTML file is one that is streamed directly from the application server to the web server and needs to have the same HTML sections (Head and Body) as a static file. In addition, the file needs to have a Content-type statement as the very first line in the file so that the web server can identify the type of file that is being sent by the application server.

To simplify text formatting within the table a STYLE section was used. A STYLE section creates a set of formats for text associated with specific HTML tags and has the advantage of setting the font information once instead of bracketing each piece of text with <FONT> tags.

The following code created the table. Note that to accomplish the necessary display format, tables were nested within each other.

```
/* html tags for each defect color */
red  ='<td width="1%" bgcolor="#800000"
       bordercolor="#800000">&nbsp;</td>';
green='<td width="1%" bgcolor="#336600"
       bordercolor="#336600">&nbsp;</td>';
blue ='<td width="1%" bgcolor="#000080"
       bordercolor="#000080">&nbsp;</td>';
reg  ='<td width="1%" bgcolor="#FFFFFF">&nbsp;</td>';
```

```
if first.rollno then do;
  put  @01 '<table border="1" width="100%"
            cellspacing="1" cellpadding="0"'
      /@01 '       bordercolor="#3333FF"
                   bgcolor="#C0C0C0">'
      /@01 ' <tr>'
      /@01 '  <td>'
      /@01 '   <table border="0" width="100%"
               cellspacing="1" cellpadding="0"'
      /@01 '     bgcolor="#C0C0C0">'
      /@01 '     <tr>'
      /@01 '      <td width="9%" bgcolor="#DDDDDD"
                  rowspan="' dn  '">'
      /@01 '        <font face="Arial" size="1"
                    color="#0000FF">'
      /@01 '         <button>'  rollno '</button></td>'
  ;
  do i = 1 to 82;
     if type='A1' and strips{i} ne ' '
        then put @01 red;
     else if type='A2' and strips{i} ne ' '
        then put @01 green;
     else if type in ('V1','V2') and stips{i} ne ' '
        then put @01 blue;
     else put @01 reg;
  end;
     put  @01 '      <td width="9%"bgcolor="#DDDDDD"
                      rowspan="' dn '">'
         /@01 '       <font face="Arial" size="1"
                      color="#0000FF">'
         /@01 '        NG=' NG ' OK=' ok '</td>'
         /@01 '     </tr>'
    ;
end;
else do;
   put  @01 '      <tr>';
   do i = 1 to 82;
      if type='A1' and strips{i} ne ' '
         then put @01 red;
      else if type='A2' and strips{i} ne ' '
         then put @01 green;
      else if type in ('V1','V2') and strips{i} ne ' '
         then put @01 blue;
      else put @01 reg;
   end;
   put  @01 '     </tr>';
end;
if last.rollno then
 put  @01 '   </table>'
     /@01 '  </td>'
     /@01 ' </tr>'
     /@01 '</table>'
 ;
if eof then
   /* end HTML document */
 put   @01 '</body>'
      /@01 '</html>'
 ;
```

Figure 4: Sample Code

THE CONTROL PANEL

The control panel section contains an ASP file that generates an HTML form after querying the appropriate SQL Server table for the most recent three days worth of data. This information is accessed using ADO connectivity via OLE DB within a VBScript. In addition, a set of JavaScript functions were created and utilized to control the functionality of the buttons and check boxes.

The control panel allows users to change the number of rolls displayed, change the timeframe selected, and change the defects displayed within the graph. If a user “unchecks” one of the defects (or checks one that was not checked), a JavaScript function submits INSPECT.SAS, which re-creates the inspection graph with or without the selected defect. Here is an example of the JavaScript function used to submit INSPECT.SAS:

```
/* Refresh the INSPECT graph with the menu
options selected */
function refreshed() {
  document.INSPECT.target="contents"
  document.INSPECT.action="/cgi-
bin/broker.exe"
  document.INSPECT.submit()
}
```

JavaScript syntax requires that an element or action within an HTML form be identified by document.*formname*, INSPECT in this case.

- Target is the frame section (middle) where the output from INSPECT.SAS should be displayed.
- Action calls the SAS/IntrNet application broker.
- Submit tells the browser to send the request to the web server, which performs the action.

To change the number of rolls or the timeframe, users simply specify the number of rolls or a new timeframe and click on the REFRESH button to re-create the graph. The REFRESH button calls a JavaScript function, which tells the browser to submit the user’s request to the Web server.

The RESET button was defined to reset all of the check boxes and other items within the menu to default values. This action is performed by the browser only and does not submit the page to the Web server to refresh the default data.

GENERAL APPLICABILITY

To create a dynamic Web graphic, we used SAS software on a server to process and generate Web pages. By using the best tool in the right place, we succeeded in pleasing our users. In addition, we developed a tool that not only replaced their old system, but offered more capability and expansion potential.

Our solution to this specific quality control application can be applied in many different situations, even outside the quality arena. Anytime you have an event, a lot, a sample, a unit that can be subdivided into sequential sections, such as hours, days, locations, or steps, you may want to display this data comparatively on one Web page so you can scan for patterns or trends.

The technique is easy to implement and should run quickly in most situations. Once a graph is created, you can visually compare across the sequence or scroll up and down to compare units. Colored table cells provide the visual impact necessary to draw your attention to problems without the overhead (and slow speed) of true graphics.

Although our example only addresses a binary problem (showing whether the anomaly exists in the cell or not), you could apply this technique to show discrete levels of a measurement. For example, instead of using the cluster of rows for 4 different defects, you could build comparative histograms, where vertical cells would be filled based on the magnitude of the value for that column.

CONCLUSION

Our example demonstrates a simple technique that can be used to quickly display and compare multiple charts. We utilized low graphic resolution, which trades high-level detail for faster loading and scrolling. This approach would be used typically for quick scanning with the capability to drill-down on detailed data or higher resolution graphs.

SAS software was used to extract, process, and analyze the data and to generate HTML. Although we used Active Server Pages, VBScript, and JavaScript, the same results could just as easily been developed with Java Server Pages and JAVA.

CONTACT INFORMATION

Eric Brinsfield

Meridian Software, Inc.

12204 Old Creedmoor Road

Raleigh, NC 27613

(919) 847-6750

merecb@meridiansoftware.com

www.meridiansoftware.com

Caroline Bahler

Meridian Software, Inc.

12204 Old Creedmoor Road

Raleigh, NC 27613

(919) 387-6735

merccb@meridiansoftware.com

www.meridiansoftware.com

TRADEMARKS