



PhUSE 2010 - Paper TS09

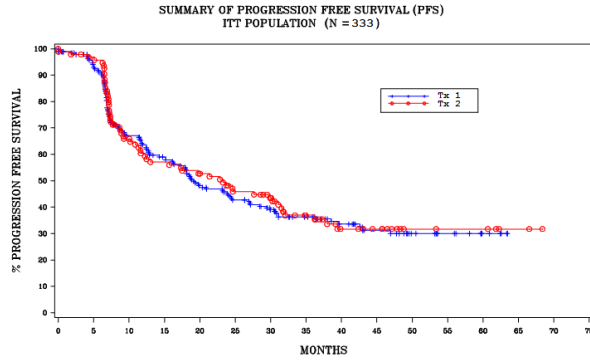
Capturing Tabular Data from Graphical Output: Web and Windows- Based Tools

**Brian Fairfield-Carter
and
Stephen Hunt,
ICON Clinical Research,
Redwood City, CA**

Outline

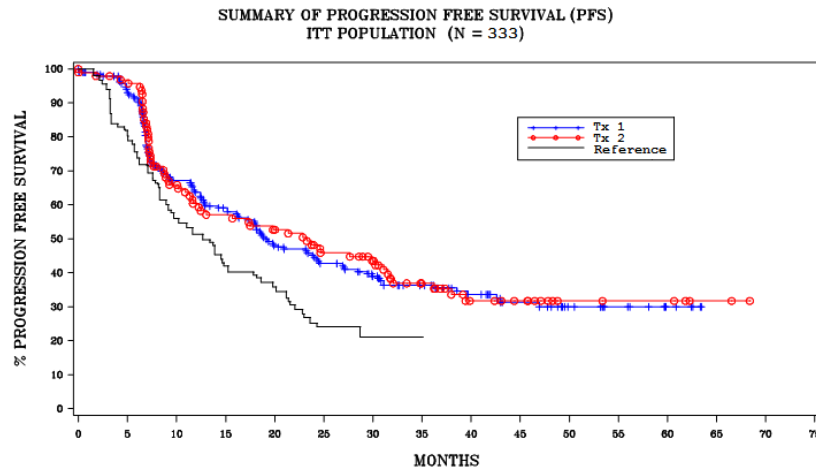
- ‘Reverse engineering’ data: when and why
- Capturing screen coordinates in Windows Paint, transforming to plot coordinates
- Two ‘Home-built’ applications:
 - HTA (web-based)
 - Windows API
- Working with XML
 - Importing XML to SAS
 - Displaying XML
 - EXTensible Stylesheet Language
 - Vector Markup Language

Why reverse-engineer graphical data?

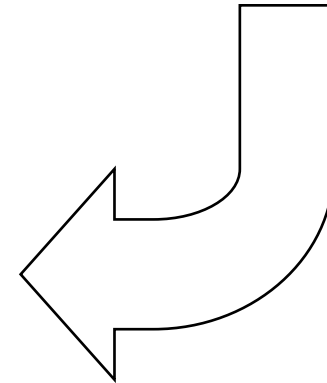


NOTE: PROGRESSION FREE SURVIVAL IS CALCULATED FROM RANDOMIZATION.
NOTE: THE CURVE IS ESTIMATED BY KAPLAN-MEIER METHOD.

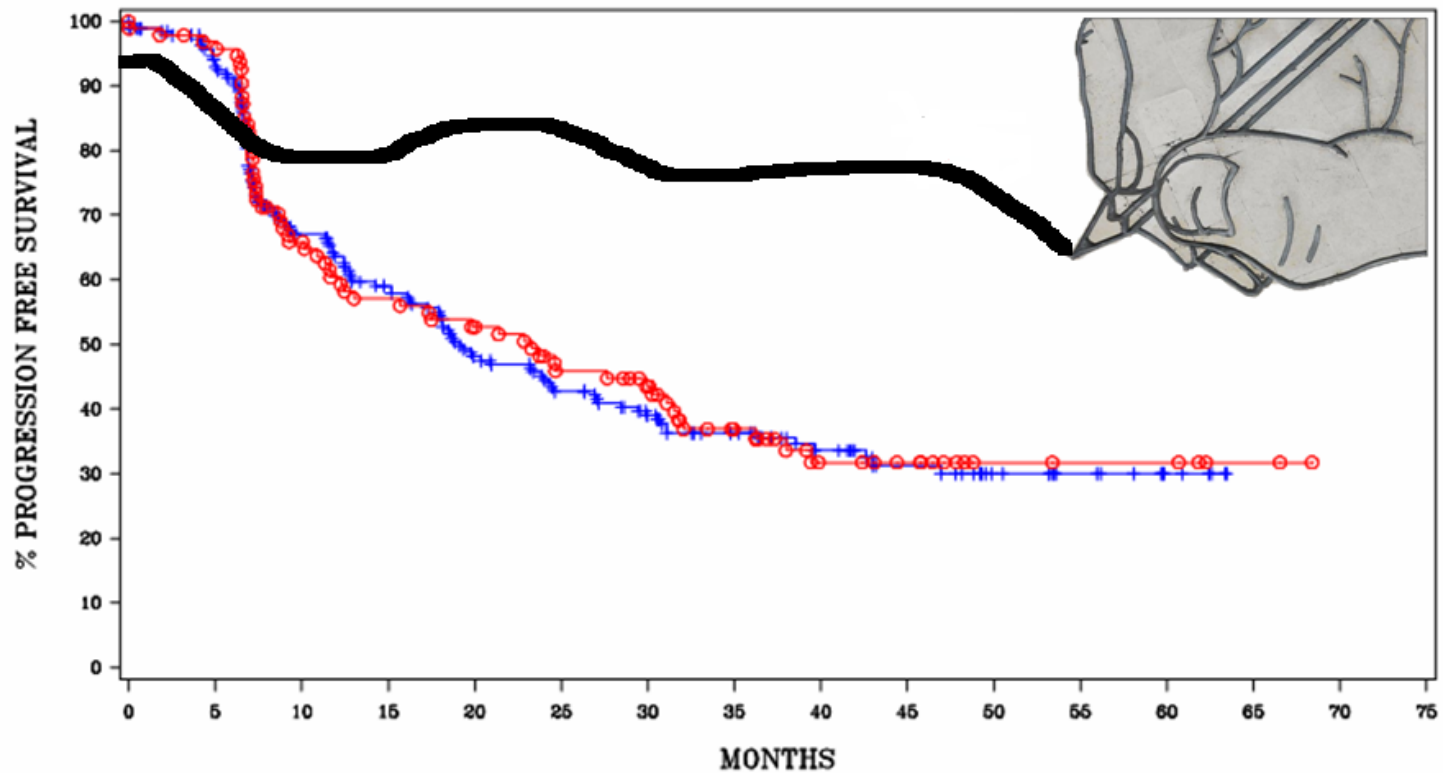
+



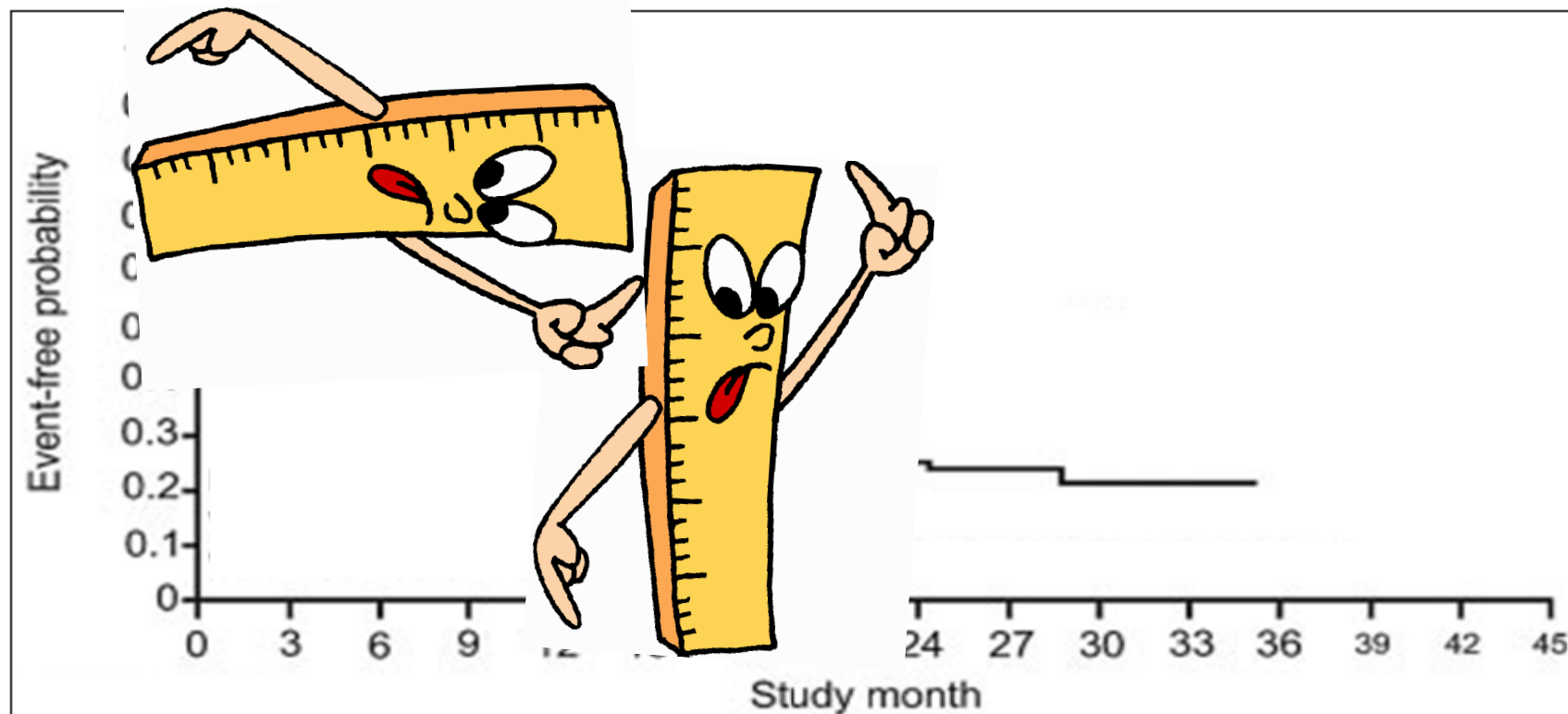
NOTE: PROGRESSION FREE SURVIVAL IS CALCULATED FROM RANDOMIZATION.
NOTE: THE CURVE IS ESTIMATED BY KAPLAN-MEIER METHOD.



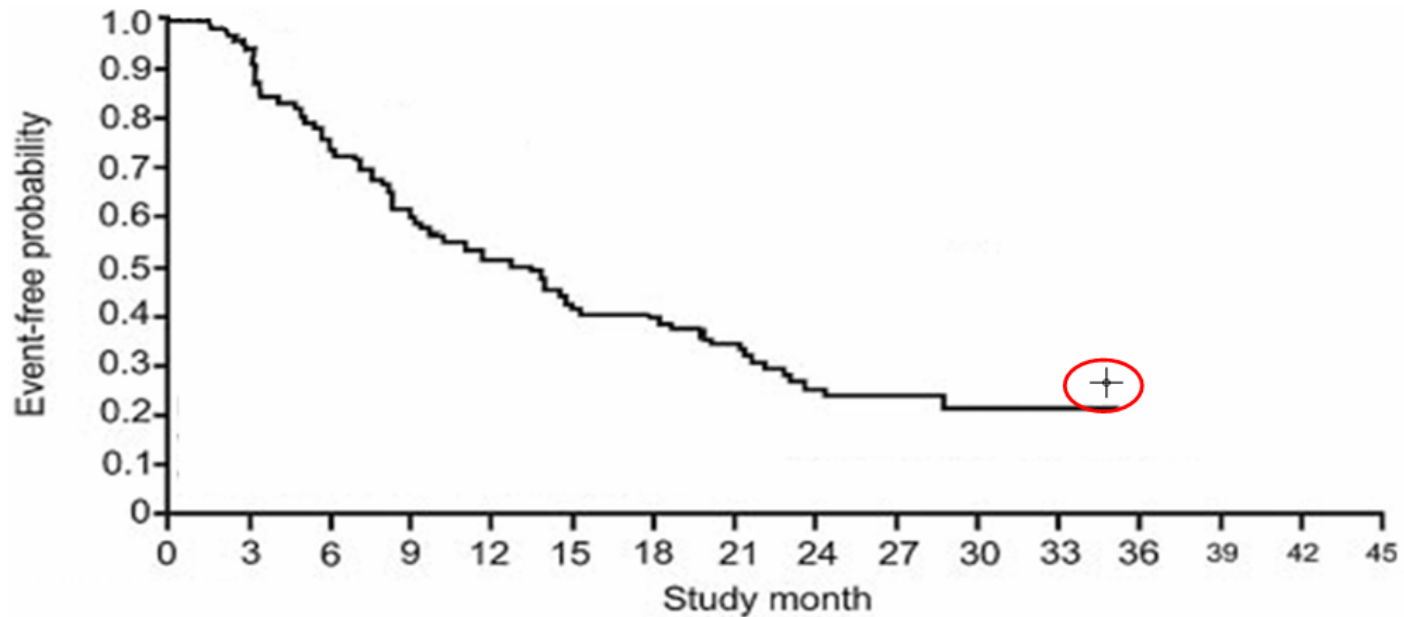
**SUMMARY OF PROGRESSION FREE SURVIVAL (PFS)
ITT POPULATION (N = 333)**



NOTE: PROGRESSION FREE SURVIVAL IS CALCULATED FROM RANDOMIZATION.
NOTE: THE CURVE IS ESTIMATED BY KAPLAN-MEIER METHOD.



Capturing screen coordinates in Windows Paint



853,562

Record screen coordinates and transform to plot coordinates

```
data ref_;
  input screen_x screen_y;
  cards;
  83    5
  108   5
  108  11
  ...
run;
```

Distance from the screen x-axis origin

```
data ref_;
  set ref_;
  plot_x=((screen_x-screen_x_min)/(screen_x_max-screen_x_min))
    *plot_x_range;
  plot_y=((screen_y_max-screen_y)/(screen_y_max-screen_y_min))
    *plot_y_range;
run;
```

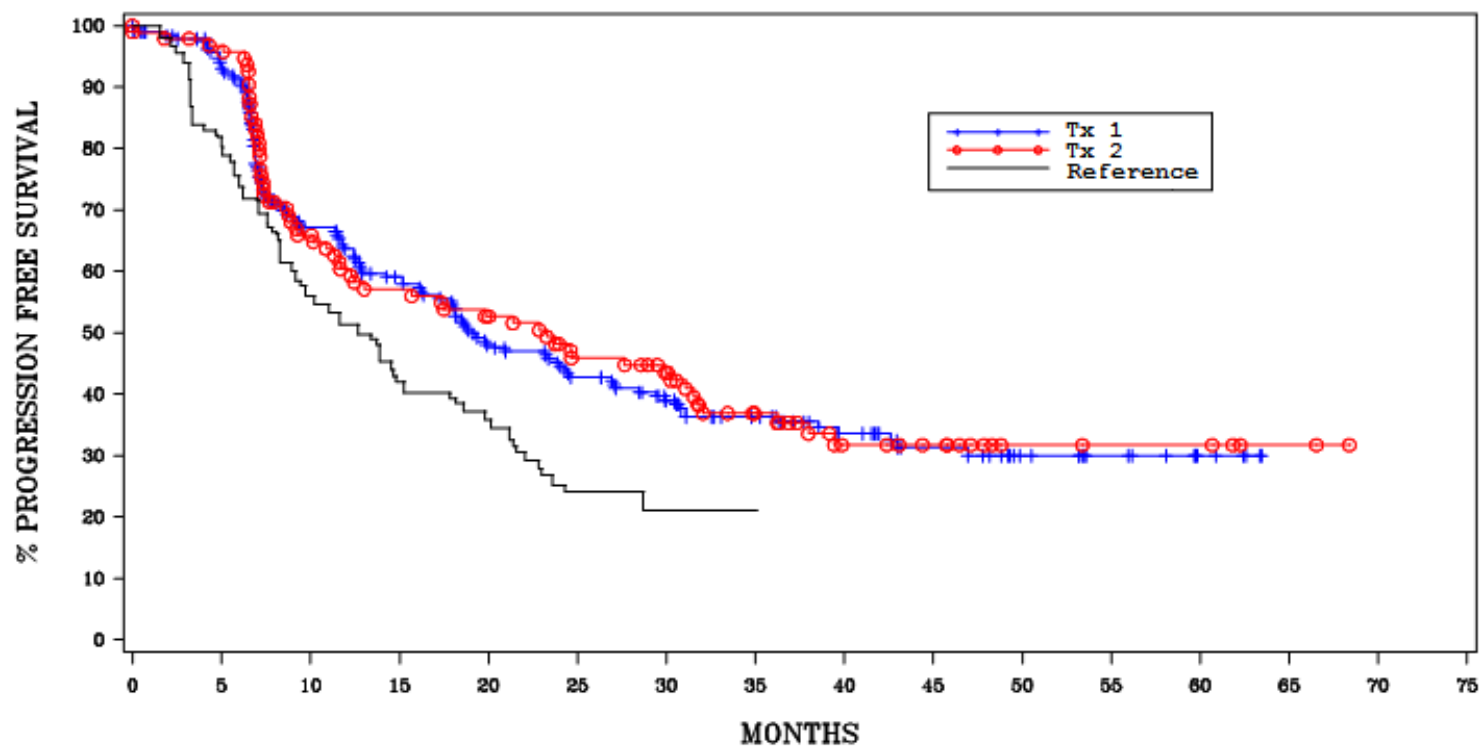
Length of the screen x-axis

Plot x-axis range

Add the reference line...

```
data ref_;  
    set ref_;  
    trt=3;  
run;  
  
data final;  
    set final ref_;  
run;  
  
proc gplot data=final;  
    plot plot_y*plot_x = trt;  
        / vaxis=axis1 haxis=axis2 legend=legend1;  
run;
```

**SUMMARY OF PROGRESSION FREE SURVIVAL (PFS)
ITT POPULATION (N = 333)**



NOTE: PROGRESSION FREE SURVIVAL IS CALCULATED FROM RANDOMIZATION.
NOTE: THE CURVE IS ESTIMATED BY KAPLAN-MEIER METHOD.

In summary...

- Windows Paint offers a partial (but still labor-intensive) solution...

A Useful application would...

- Display a graphical image
- Track mouse pointer position
- Determine screen coordinates at key 'events' (i.e. mouse-clicks)
- Write screen coordinates to file

Web Applications

[Gmail](#) [more](#) ▼



[Advanced Search](#)
[Language Tools](#)

Google Search

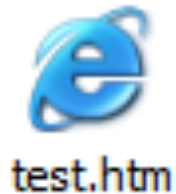
I'm Feeling Lucky

Search: ☒ the web ☐ pages from Canada

dHTML vs HTA

- dHTML: HTML with embedded script components; security rules assume communication with remote servers
- HTA: similar to dHTML, but assumes no communication with remote servers (so avoids a lot of security issues)

dHTML



```
<html>
  <script language=jscript for=mybutton event=onclick>
    alert("Hi");
  </script>

  <input type=button id=mybutton value="Hi"/>
</html>
```

 To help protect your security, Internet Explorer has restricted this webpage from running scripts or ActiveX controls that could access your computer.

Hi

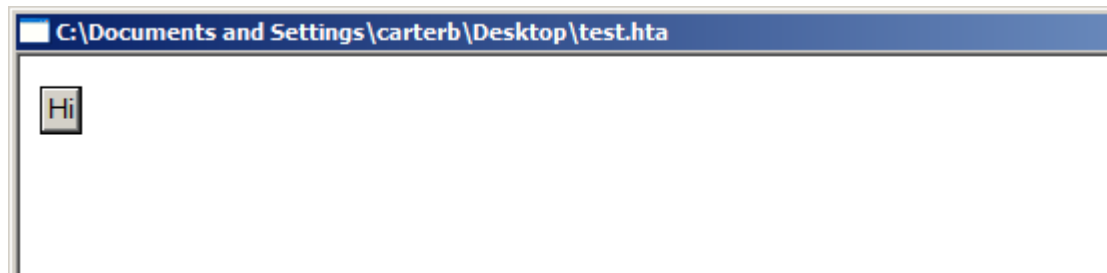
HTA



test.hta

```
<html>  
  <script language=jscript for=mybutton event=onclick>  
    alert("Hi");  
  </script>
```

```
  <input type=button id=mybutton value="Hi"/>  
</html>
```

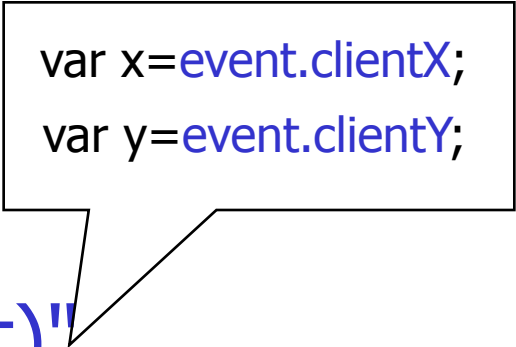


HTA: Display an Image

```
<body id="bodyid"  
  onClick="capture(event)"  
  onMousemove="getcoord(event)"  
  onUnload="endcapture(event)"  
  background="graph.bmp">  
</body>
```

HTA: Track Mouse Pointer Position

```
<body id="bodyid"  
  onClick="capture(event)"  
  onMousemove="getcoord(event)"  
  onUnload="endcapture(event)"  
  background="graph.bmp">  
</body>
```



```
var x=event.clientX;  
var y=event.clientY;
```

HTA: Capture Screen Coordinates

```
<body id="bodyid"
  onClick="capture(event)"
  onMousemove="getcoord(event)"
  onUnload="endcapture(event)"
  background="graph.bmp">
</body>
```

```
var x=event.clientX;
var y=event.clientY;
mytext.Writeline("<X_COORD>" +
x + "</X_COORD>");
```

HTA: Write Coordinates to File

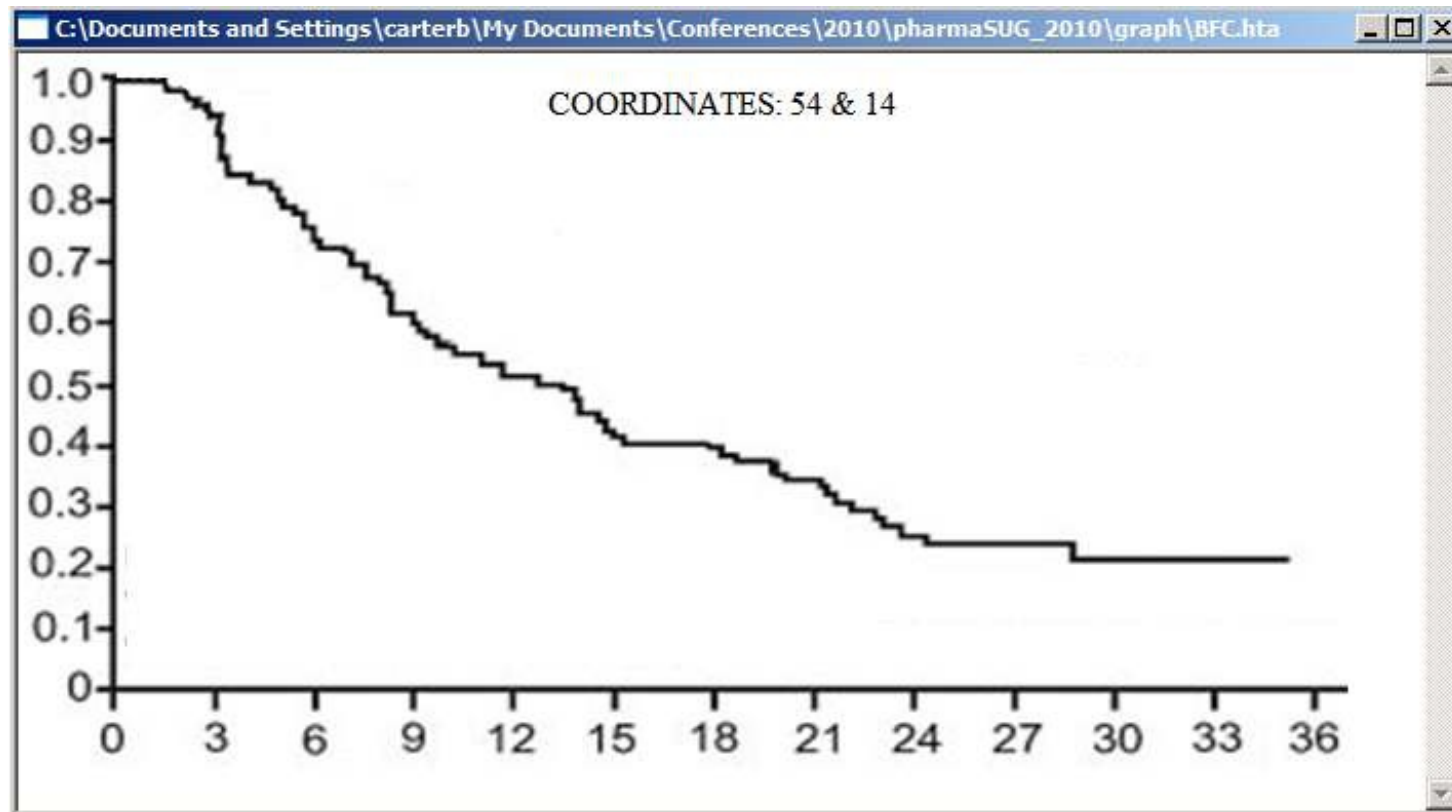
- During the implicit 'onLoad' event:

```
var mytext=fso.CreateTextFile("MyCoordinates.xml",true);  
mytext.WriteLine("<?xml version='1.0'?>")  
---(etc.)---
```
- At each mouse click:

```
mytext.WriteLine("<X_COORD>" + x + "</X_COORD>");
```
- At the 'onUnload' event ('endcapture' function):

```
mytext.WriteLine("</catalog>");  
mytext.Close();
```

HTA: Running the Application



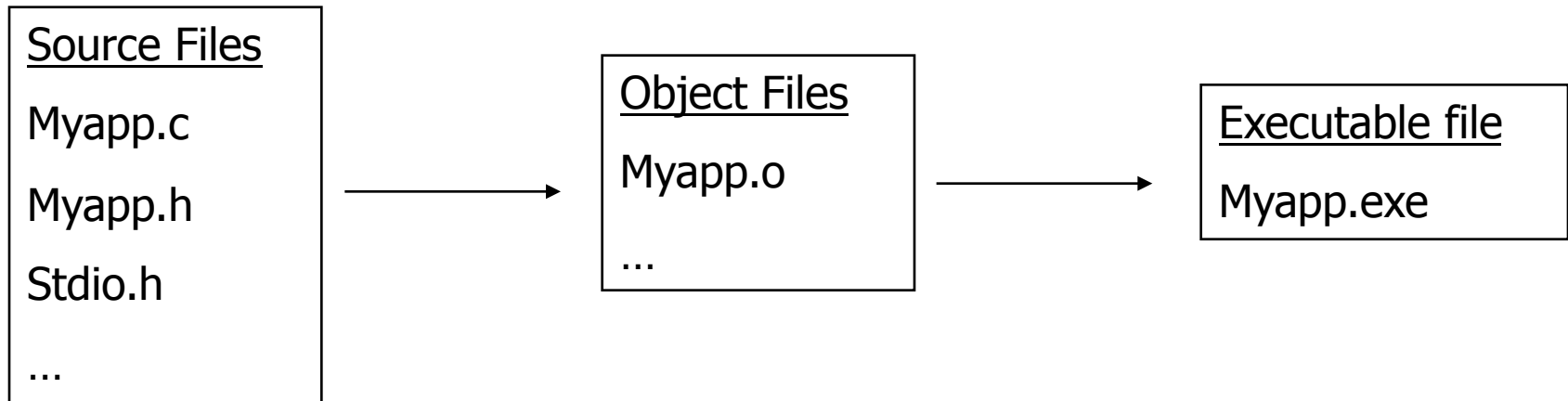
HTA: Output File

```
<?xml version='1.0'?>
<?xml-stylesheet type='text/xsl' href='table.xsl'?>
<catalog>
  <COORDINATES>
    <X_COORD>72</X_COORD>
    <Y_COORD>1005</Y_COORD>
  </COORDINATES>
  ...
  <COORDINATES>
    <X_COORD>176</X_COORD>
    <Y_COORD>911</Y_COORD>
  </COORDINATES>
</catalog>
```

A Windows Application

- Uses the Windows API
- Greater programming overhead, but more control & flexibility
- Open-source, written in C, built on MinGW (Minimalist GNU for Windows)
- For info on the MinGW framework, and on building from source, refer to:
 - *SAS, GNU & Open Source: MinGW Development Tools and Sample Applications*. Brian Fairfield-Carter & Stephen Hunt. Proceedings of the 2006 Pharmaceutical Industry SAS Users Group Conference.

Compilation steps – carried out by GCC (the GNU Compiler Collection),
'orchestrated' by the GNU Make facility...



Win API: Display an Image

```
hBmp = LoadImage(<instance>,<file>,IMAGE_BITMAP, 0, 0,  
    <options>);  
RedrawWindow(<bitmap window handle>,  
    0,0,RDW_INVALIDATE);
```

Win API: Track Mouse Pointer Position

```
switch (message)
```

```
{
```

```
case WM_MOUSEMOVE:
```

```
    xPos = LOWORD(IParam);
```

```
    yPos = HIWORD(IParam);
```

Traps movements of the mouse pointer, captures coordinates of pointer

```
case WM_LBUTTONDOWN:
```

```
    sprintf(cX,"%i",CursorPoint.x);
```

```
    sprintf(cY,"%i",CursorPoint.y);
```

```
    strcpy(cCoordBuffer,strcat(cCoordBuffer,cX));
```

```
    strcpy(cCoordBuffer,strcat(cCoordBuffer,","));
```

```
    strcpy(cCoordBuffer,strcat(cCoordBuffer,cY));
```

```
    strcpy(cCoordBuffer,strcat(cCoordBuffer,"\r\n"));
```

Win API: Capture Screen Coordinates

```
switch (message)
```

```
{
```

```
case WM_MOUSEMOVE:
```

```
    xPos = LOWORD(IParam);
```

```
    yPos = HIWORD(IParam);
```

Traps mouse-click events,
writes coordinates of pointer
to text buffer

```
case WM_LBUTTONDOWN:
```

```
    sprintf(cX,"%i",CursorPoint.x);
```

```
    sprintf(cY,"%i",CursorPoint.y);
```

```
    strcpy(cCoordBuffer,strcat(cCoordBuffer,cX));
```

```
    strcpy(cCoordBuffer,strcat(cCoordBuffer,","));
```

```
    strcpy(cCoordBuffer,strcat(cCoordBuffer,cY));
```

```
    strcpy(cCoordBuffer,strcat(cCoordBuffer,"\r\n"));
```

Win API: Write Coordinates to File

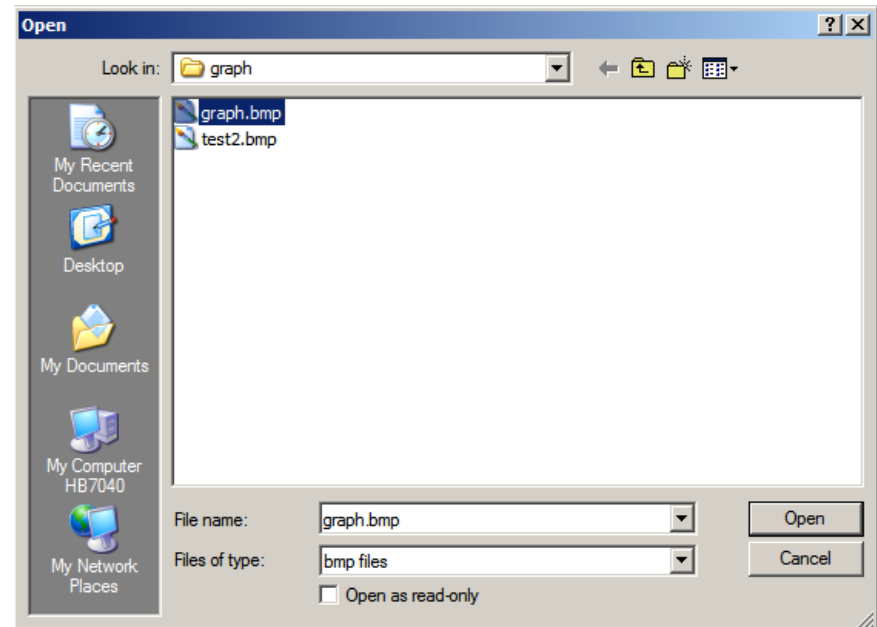
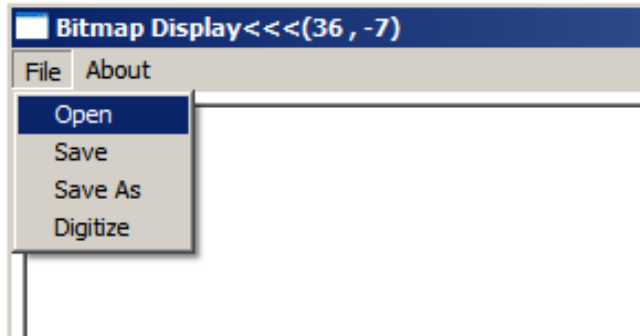
```
bSaveFileName = GetSaveFileName(&sfn);
f=fopen(sfn.lpstrFile,"w");
fprintf(f,"%s","<?xml version='1.0'?>\n");
...(etc.)...
```

Launches 'Save/Save As' dialog

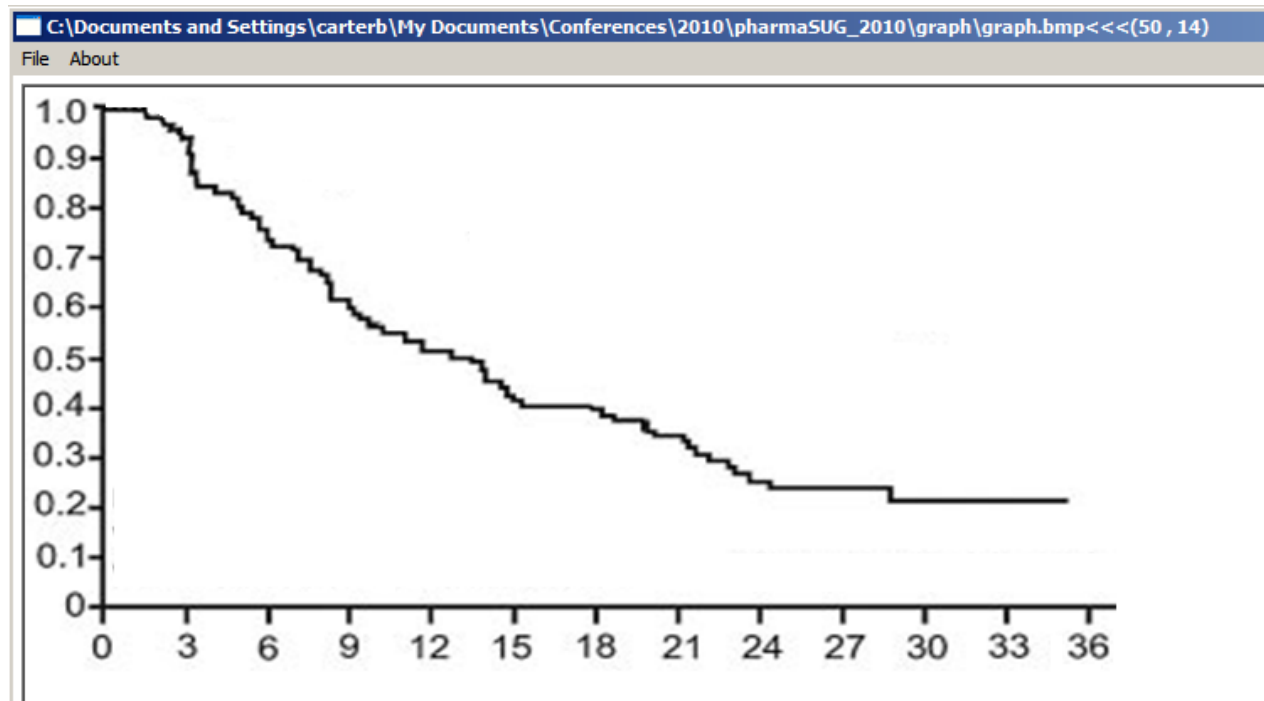
Opens text file for writing

Writes to text file

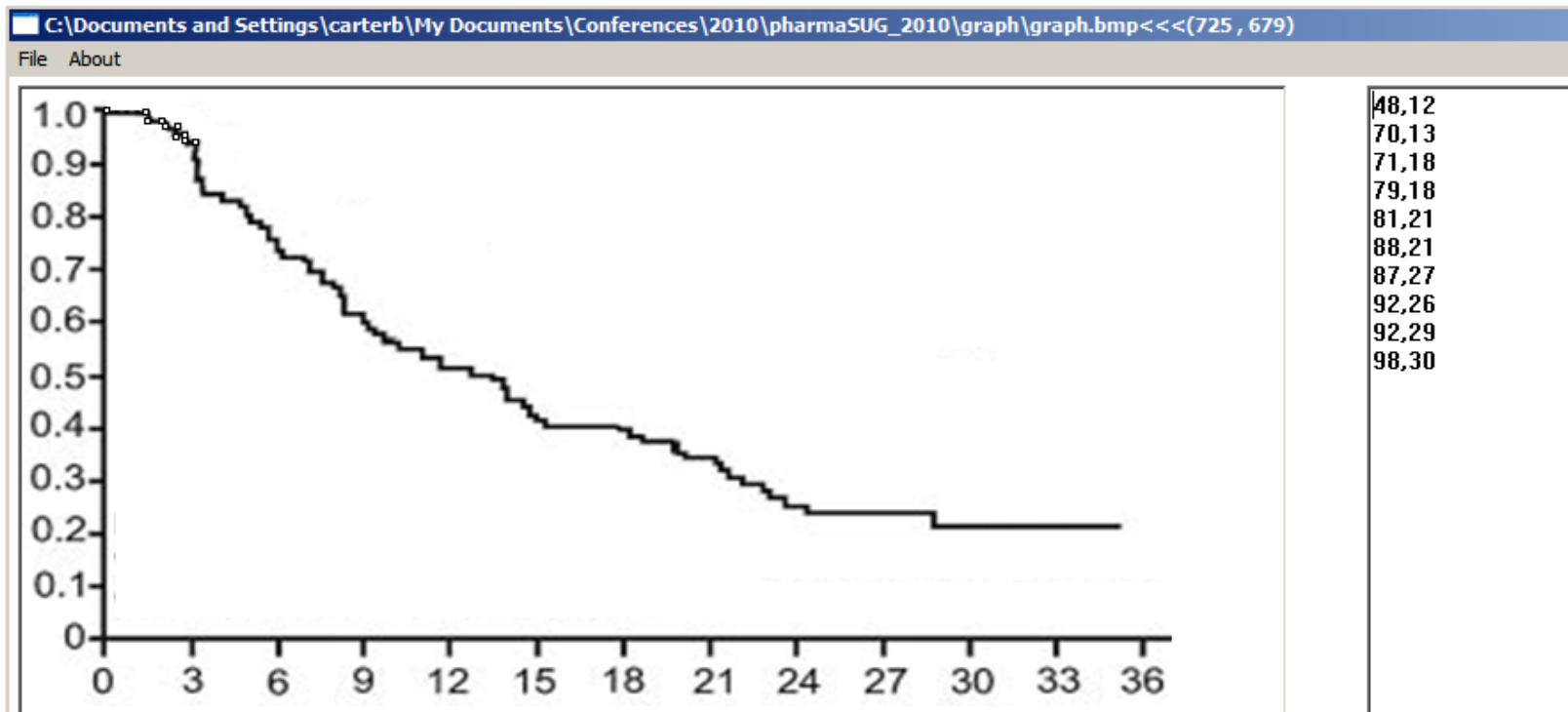
Win API: Running the Application



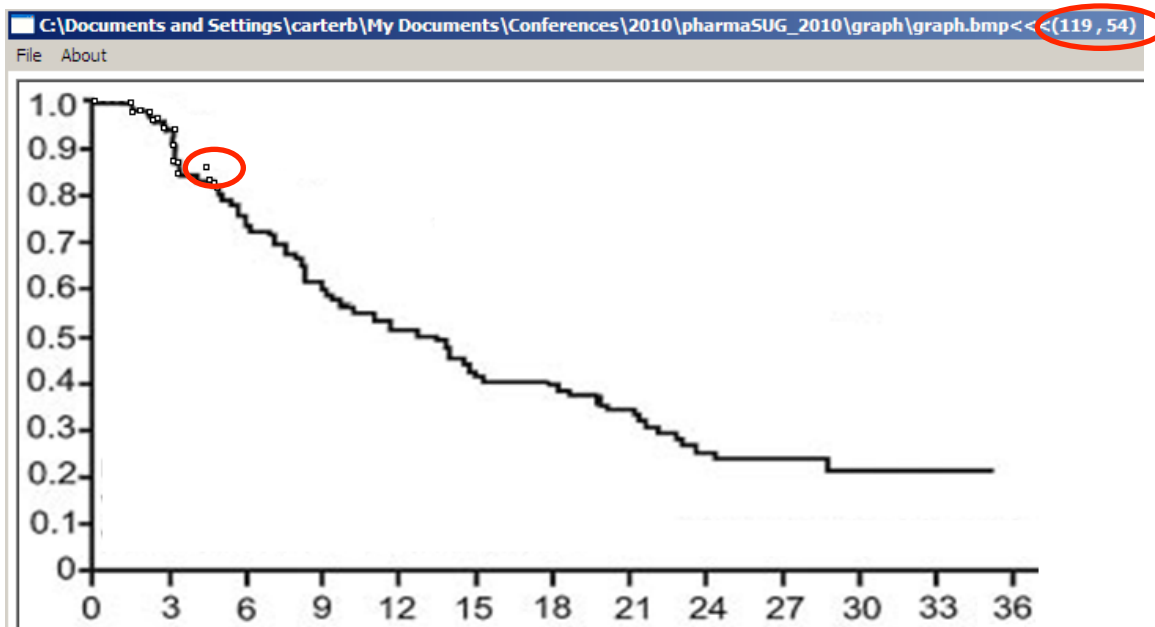
Win API: Running the Application



Win API: Running the Application

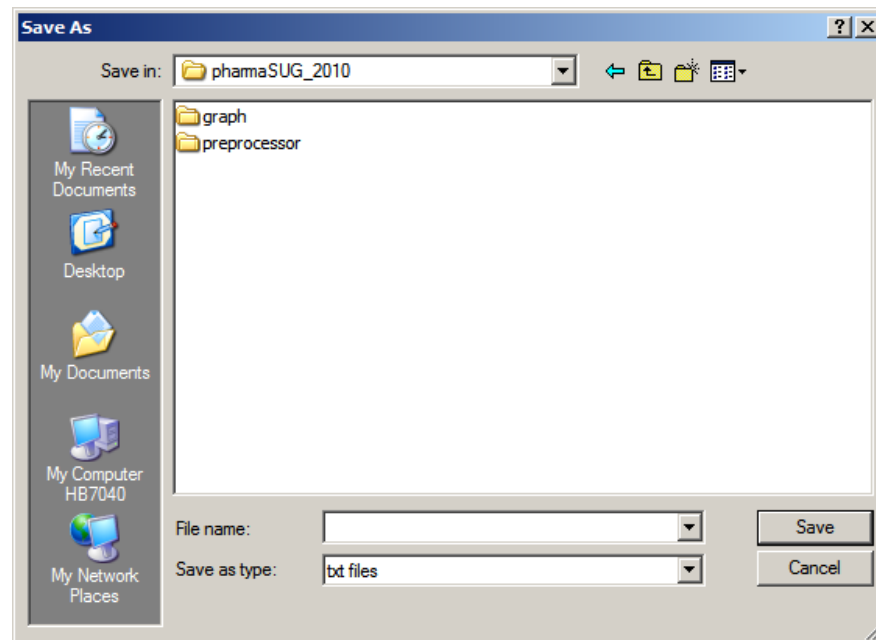
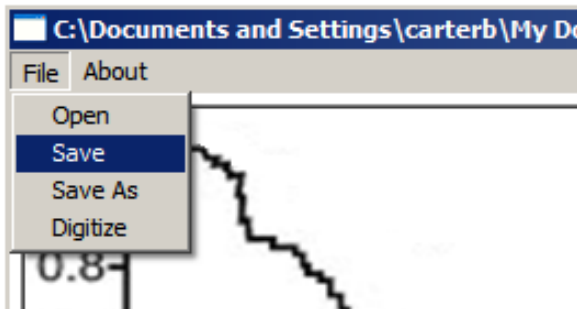


Win API: Running the Application



48,12
71,13
72,19
77,18
83,19
85,24
88,23
92,29
99,30
98,40
98,50
101,51
101,58
119,54
121,62
124,64
126,67

Win API: Running the Application



Win API: Output file

```
<?xml version='1.0'?>
<?xml-stylesheet type='text/xsl' href='chart.xsl'?>
<catalog>
  <COORDINATES>
    <X_COORD>48</X_COORD>
    <Y_COORD>12</Y_COORD>
  </COORDINATES>
  <COORDINATES>
    <X_COORD>70</X_COORD>
    <Y_COORD>13</Y_COORD>
  </COORDINATES>
  ...(etc.)...
```

Importing XML to SAS

```
filename myxml 'graph.xml';  
filename sxlemap 'graph.map';  
libname myxml xml xmlmap=sxlemap;  
data graph;  
    set myxml.coordinates;  
run;
```

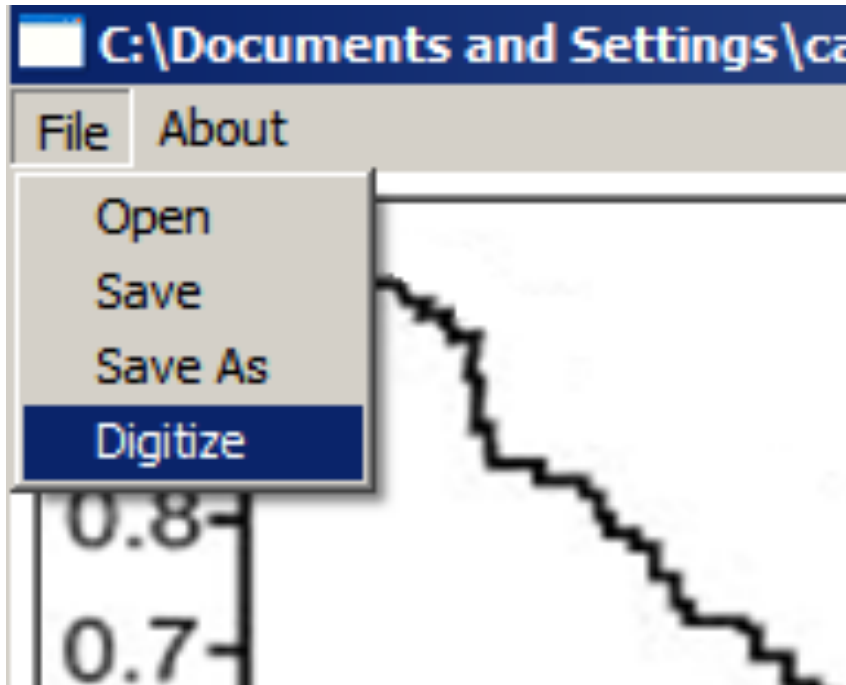
Specifies XML 'libname engine'

Provides info on how to parse the XML file

Additional functionality



Experimental 'digitize' function



```
for (int x_=0;x_<rcClient.right;x_++) {
    for (int y_=0; y_<rcClient.bottom;y_++) {

        CurrentPixel=GetPixel(BmpDC,x_,y_);
        RedValue=GetRValue(CurrentPixel);
        GreenValue=GetGValue(CurrentPixel);
        BlueValue=GetBValue(CurrentPixel);
```

Displaying XML using XSL

- XML tags have no meaning to a web browser
- XML must be transformed to HTML in order to be rendered in a browser
- `<?xml-stylesheet type='text/xsl' href='coordinates.xsl'?>`

A static HTML table

```
<html>
<body>
  <table valign="top" align="left" width="20%" border="1" style="font-
    family:Arial Narrow;font-size:12px">
    <tr>
      <td valign="top" align="left">X_COORD<span cols="18"></span></td>
      <td valign="top" align="left">Y_COORD<span cols="18"></span></td>
    </tr>
    <tr>
      <td valign="top" align="left">72<span cols="18"></span></td>
      <td valign="top" align="left">1005<span cols="18"></span></td>
    </tr>
  </table>
</body>
</html>
```

A static HTML table

X_COORD	Y_COORD
72	1005

XML → HTML transformation using XSL

```

<html>
  <body>
    <table>
      <xsl:for-each select="catalog/*">
        <tr align="left">
          <td valign="top" align="left"><xsl:value-of select="X_COORD"/>
            <span cols="18"></span></td>
          <td valign="top" align="left"><xsl:value-of select="Y_COORD"/>
            <span cols="18"></span></td>
        </tr>
      </xsl:for-each>
    </table>
  </body>
</html>

```

XML → HTML transformation using XSL

X-Coordinate	Y-Coordinate
72	1005
103	979
136	947
176	911

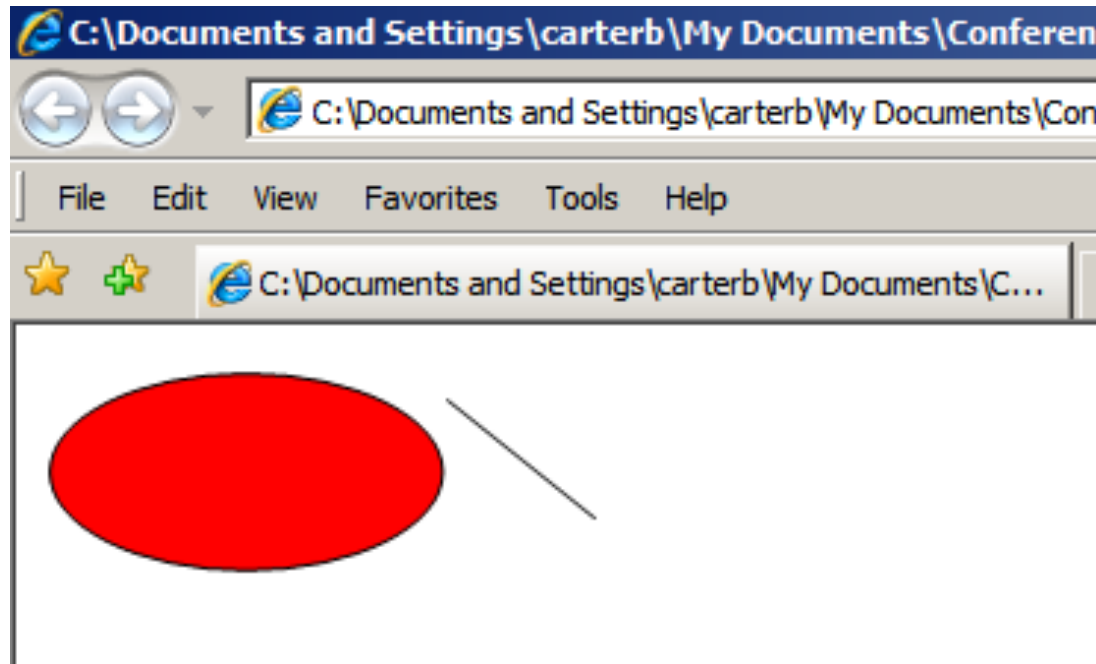
Displaying XML using VML

- VML is embedded in HTML
- (Takes the XML→HTML transformation a step further by adding VML drawing instructions)
- VML consists of graphic elements (rectangles, circles, lines, etc.) and attributes (color, etc.)

A Static VML Example

```
<html>
  <body>
    <!-- Include the VML behavior -->
    <style>v\: * { behavior:url(#default#VML); display:inline-block }</style>
    <!-- Declare the VML namespace -->
    <xml:namespace ns="urn:schemas-microsoft-com:vml" prefix="v" />
    <v:oval style="width:100pt;height:50pt" fillcolor="red">
    </v:oval>
    <v:line from="0,10" to="50,50">
    </v:line>
  </body>
</html>
```

A Static VML Example



XML → HTML/VML

transformation using XSL

```
<?xml version='1.0'?>
<?xml-stylesheet type='text/xsl' href='chart.xsl'?>
<catalog>
  <COORDINATES>
    <X_COORD>48</X_COORD>
    <Y_COORD>12</Y_COORD>
  </COORDINATES>
  <COORDINATES>
    <X_COORD>70</X_COORD>
    <Y_COORD>13</Y_COORD>
  </COORDINATES>
  ...(etc.)...
```

XML → HTML/VML

transformation using XSL

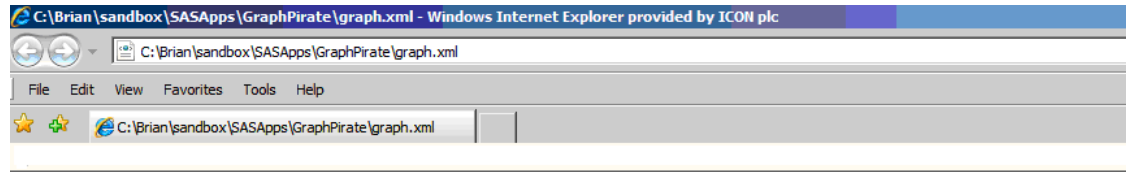
```

<xsl:text disable-output-escaping="yes">
  <![CDATA[<v:polyline style="position:absolute" points="]]>
</xsl:text>
  <xsl:for-each select="catalog/COORDINATES">
    <xsl:value-of select="X_COORD"/>
    <xsl:text> , </xsl:text>
    <xsl:value-of select="Y_COORD"/>
    <xsl:text> </xsl:text>
  </xsl:for-each>
  <xsl:text disable-output-escaping="yes">
    <![CDATA["> </v:line>]]>
</xsl:text>

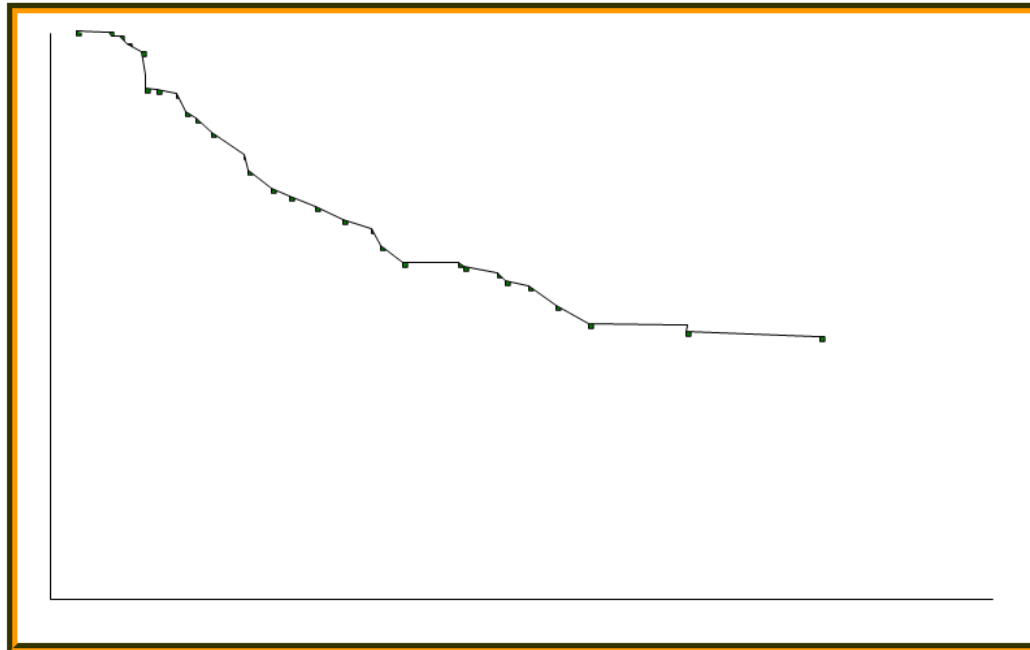
```

XML → HTML/VML

transformation using XSL



Points Graphed Using VML



References

- *SAS, GNU & Open Source: MinGW Development Tools and Sample Applications*. Brian Fairfield-Carter & Stephen Hunt. Proceedings of the 2006 Pharmaceutical Industry SAS Users Group Conference.
- <http://sourceforge.net/projects/shellout/>
- fairfieldcarterbrian@gmail.com