Paper 055-2010

Reverse-engineer a Reference Curve: Capturing Tabular Data from Graphical Output

Brian Fairfield-Carter, ICON Clinical Research, Redwood City, CA

ABSTRACT

The pharmaceutical industry is a competitive arena: new drugs inevitably have to find a market among competitors targeting the same indication. Comparisons against these competitor drugs can be problematic, however, since proprietary and legacy data may not be available in analysis-ready format. In some cases data must be captured or 'reverse-engineered' from graphical output. How would you meet a sponsor's request to overlay a reference curve on an efficacy plot where the data that produced the reference curve was not available? The most obvious but least appealing approach would of course be to estimate 'by eye', or by using pencil and ruler, the X and Y coordinates for each data point on the reference curve as it appears in graphical output. This paper introduces a more accurate and less labor-intensive alternative: to capture screen coordinates using an application like Windows Paint, and scale the screen coordinates to X,Y coordinates which can then be plotted by a SAS/Graph procedure. To illustrate the technique, a 'case study' is presented where a legacy Kaplan-Meier survival curve is added as a reference curve to a survival plot.

INTRODUCTION

Developing and marketing a new compound can not only involve demonstrating efficacy against placebo, but also against other compounds. In an industry typified by active research programs and a steady stream of new data, planned analyses are often supplemented by unplanned and ad hoc comparisons against newly-published information. Such comparisons are often hindered by the absence or non-availability of raw or electronic data for the competitor drug.

SETTING THE STAGE

Let's say you're working on an Oncology study, and you've just completed a progression-free survival plot comparing two treatment groups:

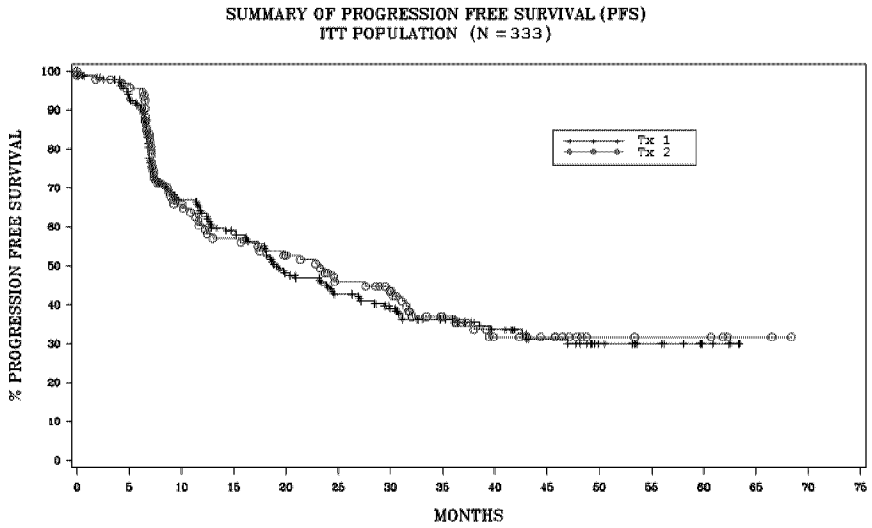

Figure 1. Sample Plot Requiring Reference Curve

The project statistician then comes to you with a paper from an industry conference presenting a similar analysis for a competitor drug:

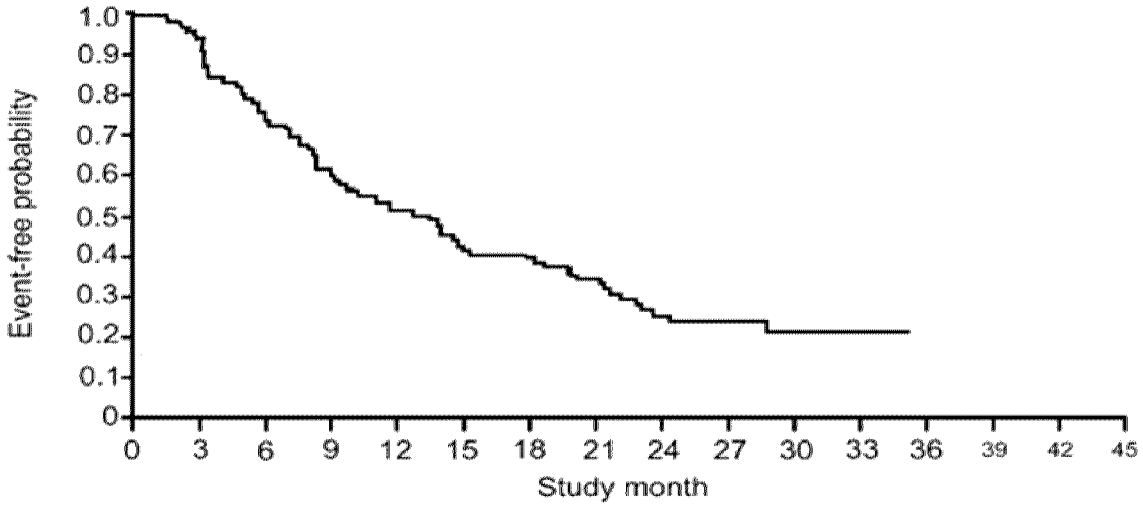

Figure 2. Sample Reference Curve

A request is then made to add the 'Event-free probability' curve for the competitor drug as a reference line in the summary of progression free survival. The only problem is that the data used in generating the event-free probability curve is not available; the dataset to reproduce the curve using SAS/Graph must somehow be created from scratch based on the graphical output. The question then is how best to retrieve the data points from the graph.

A POSSIBLE SOLUTION

One approach might be to estimate the X,Y coordinates by eye:

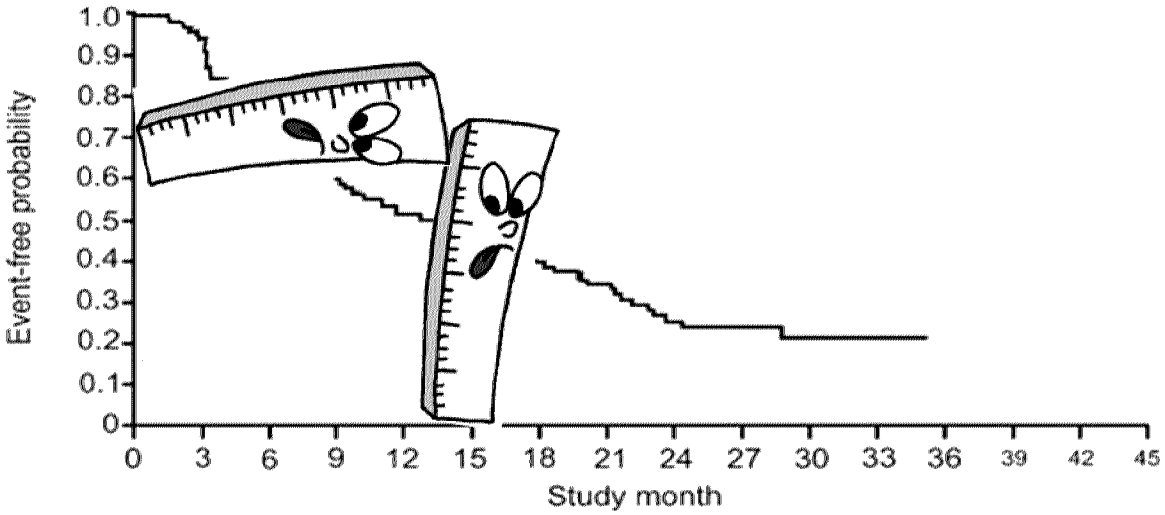

Figure 3. Manual Estimation of Plot Coordinates

Aside from the obvious unpleasantness of this approach, it would also be terribly inaccurate; achieving drop-lines parallel to the axes would be next to impossible, and position relative to the axis tick-marks provided on the graph would have to be guessed at.

AN ALTERNATIVE: REVERSE-ENGINEER DATA FROM SCREEN COORDINATES

A preferable solution would be one that enabled you to capture precise coordinates for each data point (or at least, for each noticeable inflection point on the graph), and do so without requiring a tremendous expenditure of time and

energy. By capturing the graph as a screen-shot and pasting it into Windows Paint, provided the 'status bar' in Windows Paint is activated (select the View/Status Bar drop-down menu), the screen coordinates of the mouse pointer can be displayed:

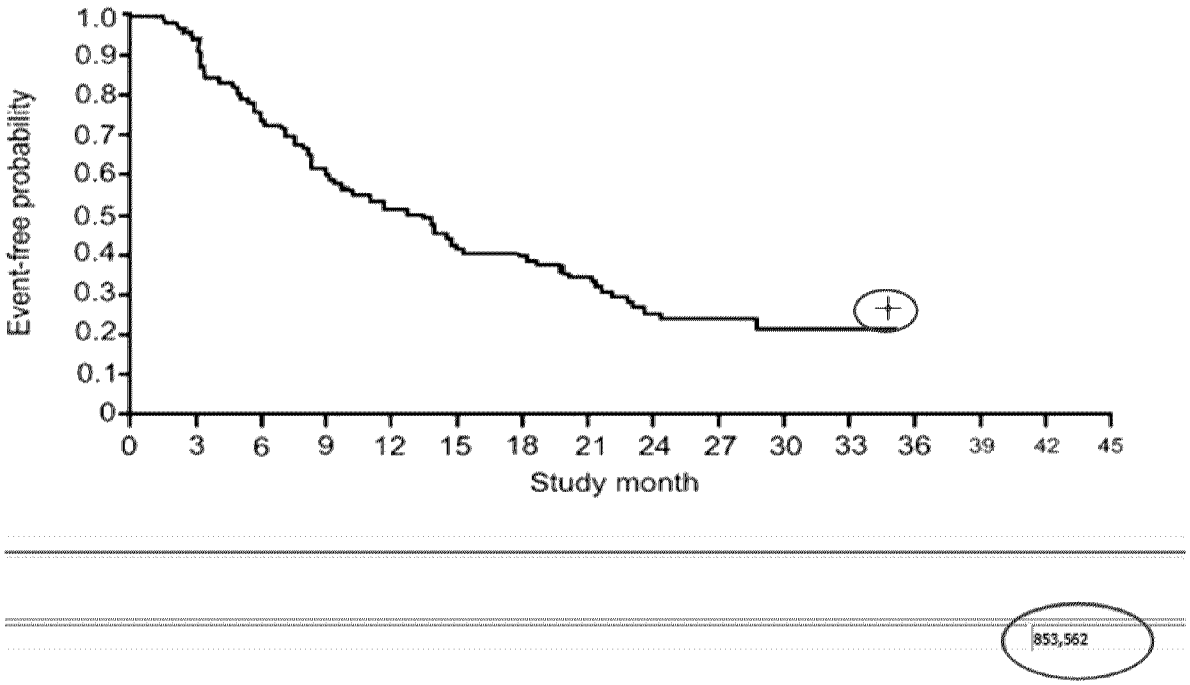

Figure 4. Displaying Screen Coordinates in Windows Paint

Placing the mouse pointer over each inflection point on the curve allows screen coordinates for each data point to be displayed and recorded. Note however that the screen coordinates are not relative to the graph's axes, and are not in the same units as the graph axes; the screen coordinates must be transformed to X,Y graph coordinates. Additionally, note that the axis ranges on the reference curve are different from those of the plot to which the reference curve is to be added; again, data from the reference graph must be transformed to match the axes of the target graph.

TRANSFORMING SCREEN COORDINATES TO PLOT COORDINATES

In order to transform screen coordinates from the reference plot to plot coordinates on the target graph, you must first capture the screen coordinates of minimum and maximum values for the horizontal and vertical axes. These minimum and maximum screen values are then equated to minimum and maximum plot axis values (in this example, these range from 0 to 1.0 on the vertical axis, and from 0 to 45 on the horizontal axis) as a simple ratio problem, in order to scale screen coordinates for each data point to X,Y coordinates on the plot.

In the above example, screen coordinates 83, 303 relate to the axis origin 0, 0 on the plot (in other words, the position on the plot denoted by Study Month=0, Event-free probability=0), and screen coordinates 815, 5 relate to the plot coordinates 45, 100. (This may seem counter-intuitive, but note that contrary to the plot axis origin, screen coordinates 0, 0 actually refer to the top left corner of the screen).

```
Screen x            Plot x
---------------------------------
83                  Study Month 0
815                 Study Month 45

Screen y            Plot y
---------------------------------
303                 Event-free probability 0
5                   Event-free probability 1.0

(However, note that Event-free probability of 1.0 on the reference plot
relates to a % Progression Free Survival of 100% on the target figure)
```

To convert the screen coordinates of each captured data point to plot coordinates, simply solve the following ratio problem: to convert screen X values to plot X values, subtract 83 (the screen value associated with the minimum plot x-axis value) from the screen X value for the data point, divide by the range of the axis in screen coordinates (815-83), and multiply by the plot axis range (45).

To convert screen Y values to plot Y values, subtract the screen Y value for the data point from the screen Y value associated with the minimum plot Y value (303), then divide by the range of the axis in screen coordinates (303-5) and multiply by the plot axis range (remember, though the reference plot Y axis ranges from 0 to 1, the target plot ranges from 0 to 100, so we need to multiply by 100).

The calculation can be carried out in a simple data step:

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

data ref_;
  set ref_;
  plot_x=((screen_x-83)/(815-83))*45;
  plot_y=((303-screen_y)/(303-5))*100;
run;
```

ADDING THE REFERENCE CURVE TO THE SURVIVAL PLOT

Now that we have the 'raw data' for the reference line, transformed to plot coordinates appropriate for the target graph, the reference line can be fairly simply added by assigning an arbitrary 'treatment group' to the data:

```
data ref_;
  set ref_;
  trt=3;
run;

data final;
  set final ref_;
run;

proc format;
  value trtcode_ 1="Tx 1"
                 2="Tx 2"
                 3="Reference";
run;
```

Plotting the two treatment groups along with the reference curve in PROC GPLOT would look something like this:

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

, producing the final output:

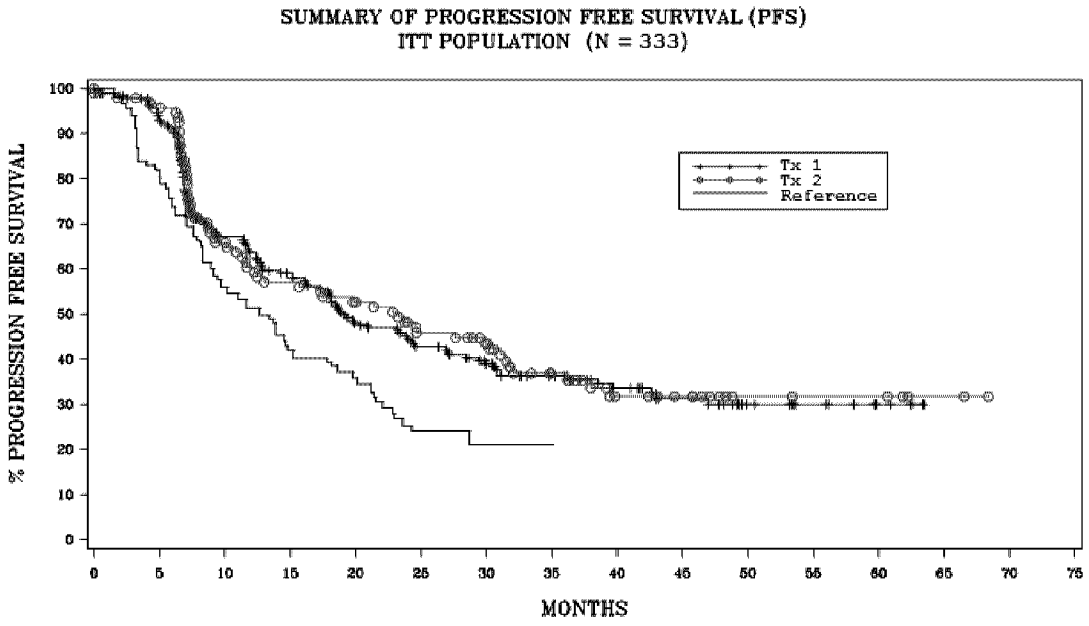

Figure 5. Sample Plot with Added Reference Curve

THE NEXT STEP

While this approach does provide a substantial improvement over the 'by eye' approach, since it provides far greater accuracy for far less work, it is by no means labor-free; the process of displaying and transcribing screen coordinates from Windows Paint can become fairly agonizing with large numbers of data points or if multiple reference curves are being captured. Is there a way of automating the process of logging screen coordinates, perhaps through a simple mouse-click?

As it happens, capturing mouse events (including capturing mouse pointer position) is a very common aspect of many applications-development environments. While it is beyond the scope of this paper to discuss the development of custom applications, what follows is a brief description of a very simple work-in-progress Windows application (named 'GraphPirate", available in the 'Shellout' collection on SourceForge (http://sourceforge.net/projects/shellout/)) for capturing and logging screen coordinates.

GraphPirate consists of 2 'panes', the larger of which displays the graph (as a bitmap image), and the smaller of which logs screen coordinates at each mouse-click. Graphs (stored as bitmap files) are opened from a standard Windows dialog called from a typical 'File/Open' drop-down menu. When the graph is opened, the file path is displayed in the title bar, along with the screen coordinates of the current mouse pointer position:

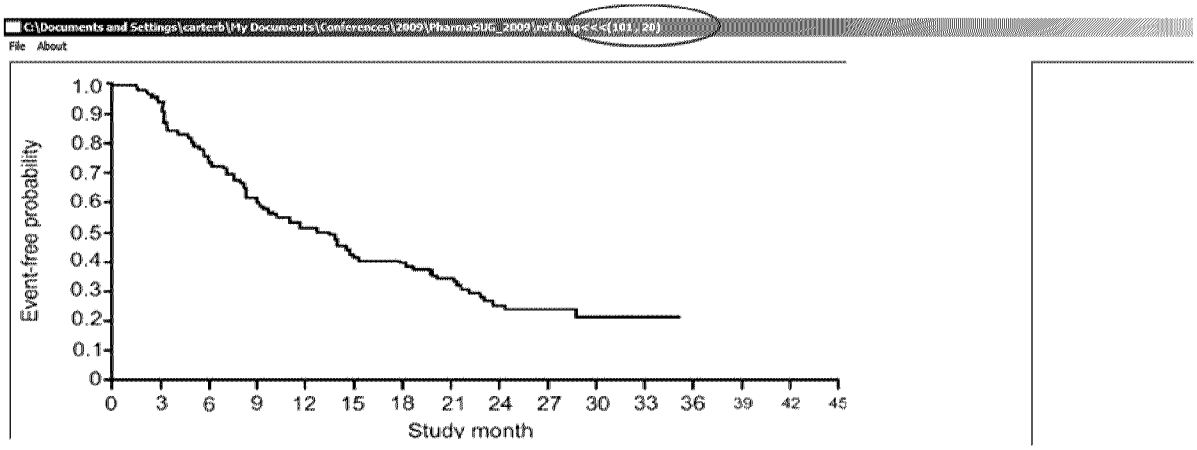

Figure 6. A Simple Application to Capture Screen Coordinates

In order to capture screen coordinates, the mouse pointer is simply placed over each target data point, and a click of the left mouse button logs the screen coordinates of the mouse pointer to the text pane, and places a small square over the selected data point:

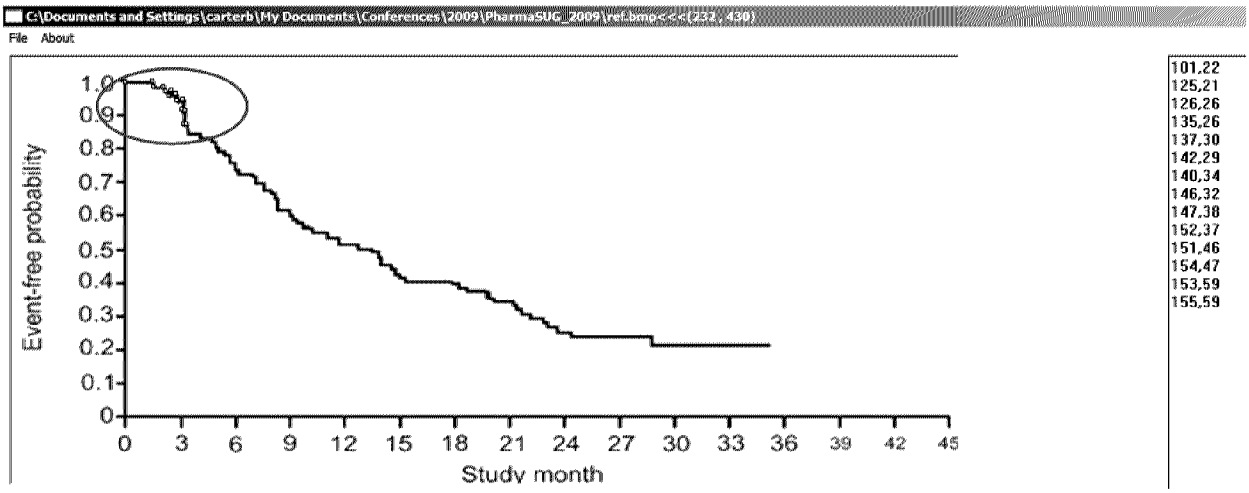

Figure 7. Screen Coordinates Logged for Target Data Points

While GraphPirate includes additional 'under construction' functionality, even the bare bones described here are sufficient to remove essentially all the agony of 'reverse-engineering' tabular data from a reference curve. Screen coordinates in the text pane can simply be cut and pasted into the SAS program editor and read in using a 'cards' statement as shown previously, and then transformed to plot coordinates with a few simple SAS statements.

CONCLUSION

While the problem of generating tabular data from graphical output may not arise very often, when it does crop up it can be both time-consuming and frustrating, and as such it is worth-while having a few tools and techniques at your disposal. Estimating X,Y coordinates by eye is both time-consuming and inaccurate; exploiting an application like Windows Paint to capture screen coordinates provides a substantial improvement in accuracy, but less of an improvement in speed. However, a number of programming interfaces exist in which simple home-made applications can be assembled to automate the process of accurately capturing and logging screen coordinates, removing most of the pain involved in reverse-engineering tabular data from graphical output.

ACKNOWLEDGMENTS

I would like to thank ICON Clinical Research for consistently encouraging and supporting conference participation, and all my friends at ICON for their great ideas, enthusiasm and support, in particular Syamala Schoemperlen and Jackie Lane.

This paper is dedicated to Cole ('Cole-Angel'), my tiny hero.

CONTACT INFORMATION

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

Brian Fairfield-Carter
ICON Clinical Research
Email: fairfieldcarterbrian@gmail.com