Interactive Dynamic Graphics for Exploratory Survival Analysis

E. Neely Atkinson*

January 31, 2021

Abstract

The availability of powerful and inexpensive hardware and software for graphical computing makes the use of dynamic interactive graphics feasible for the examination of survival data. The use of these approaches may help guide the clinical researcher in the formulation of specific hypotheses and in the selection of appropriate models.

1 Introduction.

This document describes a set of techniques for performing exploratory survival analysis using the LISP-STAT software package on a Macintosh or X11 computer system. The goal of this research is to provide clinicians direct access to survival data in an interactive intuitive fashion. While statisticians know how to look at survival data, they may not know what to look for; while clinicians know what to look for, they may not know how to look. In even the closest collaboration, there is a danger that important aspects of the data will be missed because of the difficulty in communication between individuals of vastly different and highly specialized backgrounds. One way to overcome this difficulty is to provide clinicians with methods which permit them to examine survival data directly and intuitively, rather than through mathematical models. The insights gained through such explorations can then be used to formulate models and construct testable hypotheses.

The techniques described are preliminary attempts at providing such tools. While I hope that these techniques will be useful in themselves, I also hope that they will stimulate thinking about new ways to look at survival data. Please report any bugs in the code or any difficulties in its use, but also please report any ideas you may have about capabilities which would aid you in your examinations of data.

These is a very preliminary version of the code. It may occasionally crash or fail to perform properly. It may occasionally produce cryptic error messages - these can be ignored for the most part. Also, the code is not terribly elegant; I don't want to spend a lot of time implementing features until I know what is important to the clinician and what is not.

These methods are described more fully in "Interactive dynamic graphics for exploratory survival analysis" in *The American Statistician* **49**:1, pp. 77–84.

*The Department of Biomathematics, University of Texas M. D. Anderson Cancer Center, 1515 Holcombe Blvd., Houston, Tx., 77030. Electronic mail: neely@biomath.mda.uth.tmc.edu. This work supported in part by grant CA16672-17 from the National Cancer Institute.

2 The LISP environment.

This code is implemented as a set of definitions in the LISP-STAT system, written by Luke Tierney, based on the XLISP dialect of LISP written by Dave Betz. It is possible to use the tools provided with very little knowledge of LISP; of course, with a knowledge of LISP-STAT, it is possible to do much more.

The basic form of LISP commands is (*command parameter* 1 ... *parameter p*), where *command* is a LISP command and the *parameter i* are the various parameters required by the command. Thus, to define a variable `x` with the value 3, type `(defvar x 3)`; to reset the value of `x` to 5, type `(setf x 5)`. The enclosing parentheses are required. (LISP is rumored to be an acronym for **L**ots of **I**rritating **S**purious **P**arentheses.) When a command has been typed, it is executed by pressing the "enter" key.

3 Getting started.

After starting the LISP-STAT application, load the file "expsurv.lsp" On a Macintosh, this is done by selecting "Load" from the"File" menu and then selecting the file "expsurv.lsp" in the file dialog box. On an X11 system, this is done by typing `(load "expsurv.lsp")`. This will load the required definitions.

4 Data.

Data may be read in with the "get-data" command. To read in a data file containing the data for the variables "survival," "status," "age," "grade," "stage," and "CA125," type `(get-data survival status age grade stage ca125)`. This will present you with a file dialog box, open the selected file, read the data, and store it in the names given. The data file should be a plain ASCII (text) file, with one case per line and values separated by one or more blanks. The program does not currently allow for missing values. The same variables could be stored to a text file by typing `(save-data survival status age grade stage ca125)`.

To let you get started, the command `(gen-data` n`)` will generate n sample data cases. The variables created are **arrive** (the patient's time of arrival into the study). **length** (the length of time from arrival until death or last follow-up), **status** (vital status - 1 for dead and 0 for alive), and **x**, **y**, and **z** (three covariates whose effect on survival we wish to examine.) The program currently assumes that status is always coded as 1 for uncensored (dead) and 0 for censored (alive or lost to follow-up). For the curious, the data are generated as follows. **x** and **y** are independent and uniformly distributed on (0,1); $\mathbf{z} = \mathbf{xy}$. Survival lengths are generated as exponentially distributed with $\lambda = 10x + y$. After the the survival lengths are calculated, they are rescaled by dividing by their standard deviation. Arrival times are uniformly distributed in (0,2). Time of death is computed as arrival time plus survival length. All cases with time of death greater than 2 are censored at 2 and status and survival length set accordingly. Thus, a case which arrived at 1.2 and survived for 2.3 would have a time of death of 3.5 and would therefore be marked as censored, with a survival length of 0.8 (the time from arrival at 1.2 until the close of the study at 2.0).

5 The tools.

5.1 Choosing a cut point.

A common request to statisticians from clinicians is to select some cutoff value for a continuous covariate which distinguishes normal from high-risk patients. The effects of such a division can be explored with the command `choose-cut-plot`. Assuming that you have generated some data with the `gen-data` command, you could examine the result of dichotomizing **x** by typing `(defvar ccp1 (choose-cut-plot length status x))`. (The variable name **ccp1** is arbitrary; you can use any name you want.) This will produce three windows on your screen. The first is a slider which runs from the lowest value of **x** to the highest. The second is a plot of the density of **x** with a vertical line marking the current value of the slider; if you are not familiar with density plots, you can think of it as a smoothed histogram. The third window contains two survival curves - one for cases above the slider value and one for those below. By moving the slider back and forth, you can observe the effects of changing the cutoff value. Not only does this help in selecting an appropriate point, more importantly it helps you evaluate whether the idea of dichotomizing the data is a good one in the first place. If the two curves change smoothly throughout the range of **x**, then arbitrarily selecting one value to discriminate "normal" patients may be discarding valuable information. Further, it can show you how sensitive your choice is to small variations; having selected a cutoff of 2.0, you can see what difference it makes if the cutoff is set at 1.75 or 2.25 instead.

5.2 Scatterplot matrices.

The next set of tools are based on the idea of a scatterplot matrix. A scatterplot matrix of the variables **x1**, **x2**, and **x3** is a matrix of all possible pairwise scatterplots of the variables. Further, the individual plots are all "linked" in the following fashion. If a point or points on one plot is selected (as described below), the corresponding points on the other plots will be be highlighted. Thus, by looking at the plot of **x1** versus **x2** we can see if there is a relation between those variables. By selecting only those points which have a high value for **x3**, we can see if the relation between **x1** and **x2** changes depending on the value of **x3**. Thus scatterplot matrices can be used to look for higher dimensional interactions.

Points may be selected in two ways. The first way is "selecting mode." In this mode, points are selected using the mouse in the standard Macintosh fashion. Points may be selected individually by clicking on them. Groups of points may be selected by holding the mouse button down and dragging a rectangle around them. Points can be added to the current selection be holding down the shift key while selecting additional points. Clicking on an empty space unselects all points. The second fashion for selecting points is "brushing mode." In this mode, a resizeable rectangle appears at the cursor. All points within the rectangle are selected; all points outside the rectangle are unselected. As the rectangle is moved about, points move from unselected to unselected and back again. The selection mode is chosen using the "Mouse Mode" item under the "Scatmat" menu. The brushing rectangle is resized using the "Resize Brush" item in the same menu.

There are three tools based on the scatterplot matrix available.

5.2.1 `scat-km`.

The `scat-km` command produces a scatterplot matrix linked to a Kaplan-Meier survival curve of the selected points. The curve is dynamically redrawn as the selected points change. To examine the data produced by the `gen-data` command, type `(defvar sckm1 (scat-km length status x y z))`. (The name **sckm1** may be replaced by any name of your choice.)

5.2.2 `scat-box`.

The command `scat-box` produces a scatterplot matrix linked to a censored boxplot. Censored boxplots are described in the article "Graphical Methods for Censored Data" by R. Gentleman and J. Crowley, which appeared in the September 1991 issue of the *Journal of the American Statistical Association.* Censored boxplots are similar to regular boxplots, except the boxplot is constructed from the life table, rather than the raw data. Horizontal bars are placed at the median and the upper and lower quartiles, if these exist. At the point corresponding to the last uncensored time, the value of the survival curve is printed. If a certain quartile has not yet been reached, the sides of the bar are extended to the last uncensored time. The censored boxplot is evoked by `(defvar scbx1 (scat-box length status x y z))`. (The name **scbx1** may be replaced by any name of your choice.)

5.2.3 `scat-event`.

The command `scat-event` produces a scatterplot matrix linked to an event chart. Eventcharts are described in "EVENTCHARTS: Visualizing Survival and Other Timed-Event Data" by Anne I. Goldman, which appeared in the February 1992 issue of *The American Statistician.* The vertical axis shows the time patients arrive on the study; the horizontal axis shows the length of time the patients were followed. A patient's line ends in a cross if the patient is uncensored and a circle if the patient is censored. The eventchart captures all of the original data used to construct the life table and presents it graphically. The survival curve and the eventchart are not competitors - they are complementary displays. It is not easy, for example, to read median survival from the eventchart, although it is easily seen on the survival curve. On the other hand, the survival curve considers everyone as having entered the study simultaneously, while the eventchart preserves the complete information on both time of entry and length of follow-up. Preserving the complete information allows us to examine ways in which the patient population may have changed over the course of the study. The event chart may be evoked by `(defvar scev1 (scat-event arrive length status x y z))`. (The name **scev1** may be replaced by any name of your choice.)

5.3 Examining models.

Two types of regression model are in common use for survival data: proportional hazards models and accelerated failure time models. Although these models are defined in mathematical terms which may have little intuitive appeal to a clinician, they can each be given a graphical interpretation. Suppose that our regression model contains a single covariate, which is binomial. Thus, we are comparing the survival experience of two groups. Then, according to the proportional hazards model, there exists a k such that $S_1(t) = S_2(t)^k$ for all t, where $S_i(t)$ is the survival function for group i, while according to the accelerated failure time model, there exists a k such that

$S_1(t) = S_2(kt)$. If the proportional hazards model is correct, the two curves can be moved up and down in a suitably constrained fashion until they are aligned; if the accelerated failure time model is correct, they can be moved left and right until they are aligned.

Some sample data for exploring these techniques can be generated by issuing the command `(gen-expo-example)`. This command creates two data sets. The first is stored in `time1` and `stat1` and is a sample of 100 points drawn from an exponential distribution with parameter $\lambda = 1$. The second is stored in `time2` and `stat2` and is a sample of 50 data points drawn from an exponential distribution with $\lambda = 10$. In both data sets, approximately 10% of the data are censored.

Plots for attempting to align the survival curves are produced by `(defvar ph1 (prop-haz-plot time1 stat1 time2 stat2))` and `(defvar ac1 (accel-fail-plot time1 stat1 time2 stat2))`. Each command produces a plot with a pair of survival curves and a pair of sliders for adjusting the curves. For the proportional hazards model, the curves ploted are $S_1(t)^{k_1}$ and $S_2(t)^{k_2}$, with the sliders adjusting k_1 and k_2, while for the accelerated failure model, the curves are $S_1(k_1 t)$ and $S_2(k_2 t)$.