Statistical Graphics in Quail: An Overview

Catherine B. Hurley
National University of Ireland Maynooth
Department of Mathematics
Maynooth, Ireland
churley@maths.may.ie

R.W. Oldford
University of Waterloo
Department of Statistics and Actuarial Science
Waterloo, Canada
rwoldford@uwaterloo.ca

1 INTRODUCTION

It has been suggested (Wainer, 1989) that the system first proposed by C.S. Peirce to organise knowledge is particularly suited to describing statistical graphics. Peirce felt that all information could be broken down into three different types – monadic information, which describes something in and of itself, dyadic information, which describes a relationship between two things, and triadic information, which describes the relation between two things mediated by a third. We can see how this applies in statistical graphics by considering the scatterplot. There, each case in a dataset is represented in the display by a glyph, which is monadic in nature. The scatterplot is a dyad; by positioning the case glyphs in the plane according to the values each case has on two variates X and Y, any empirical relationship between the variates can be seen. Triadic information is available by linking the scatterplot with another plot, say a dot-plot of a third variate, Z. Colouring the glyphs in the dot-plot within a given range of Z values, causes the corresponding glyphs in the scatterplot to be coloured in the same way. Here the relationship between X and Y is seen mediated by the third variate Z. This description of information is reflected in the design and implementation of our graphics software, which is part of the Quail system (Oldford et al).

Quail (for QUantitative Analysis in Lisp) is a programming environment for statistical and quantitative computing. It has extensive arithmetical, mathematical, statistical and display facilities. This paper gives a brief overview of the principles underlying the statistical graphics facilities. The original software model was first illustrated in a video (Hurley and Oldford, 1988), and was described in Hurley and Oldford (1991). Perhaps surprisingly, we have not altered the original software model, rather we have extended and enriched its scope over the intervening years.

The statistical graphics system in Quail has an object-oriented design, which we outline in Section 2. In the language of Peirce, individual objects have a monadic nature. We provide basic building blocks consisting of simple graphical objects such as point symbols and lines and container objects within which the simple objects are positioned to display relationships, ultimately forming plots. Container objects present dyadic information; Section 3 describes some such objects available in Quail. In Section 4, we outline a few of the ways triadic information is available; generally this involves comparison of plots, and, if the plots are displayed over time, interactive graphics.

2 ORGANISING INFORMATION

Each glyph representing a case is its own data structure, or object, called a `point-symbol`; similarly there is an `axis` object, and a `label` object. Even though a scatterplot is composed of point symbols, axes and a few labels, it is convenient to introduce an intermediate object called a `2d-pointcloud` object consisting of the point symbols. In general then, a statistical plot is a hierarchy of objects; the `scatterplot` object consists of axes, label and a point cloud which itself consists of point symbols, while a scatterplot matrix consists of many point clouds and labels.

The plot and each of its components are termed a *view*. A *simple view* is a view such as a point symbol or label which contains no other views. All other views are referred to as *compound views*. In our software organisation, information pertaining to a particular view is localised in that view. All views have the slots summarised in the following table:

Slot	Purpose	Example
Viewed object	The data structure being 'viewed'	scatterplot: dataset point symbol: case function-view: single variable function
Drawing style	Controls appearance	colour, highlighting
Viewports	Screen locations where view appears	
Menus	User interface to view	Change colour, variate Access viewed object

A compound view also has slots for its constituent views or *subviews* and the subview locations. Note that these are not the same as the viewport locations: compound views position subviews using some abstract coordinate system which is meaningful for that compound view. When a compound view is first drawn in a viewport, the subview locations are used to determine viewports for the subviews.

3 ORGANISING INFORMATION GRAPHICALLY

A graphical display consists of graphical objects drawn on the screen. The relative positions of the graphical objects drawn is crucial: in a scatterplot the point symbol positions display the relationship between two variates while in a scatterplot matrix the panels are positioned so that in each row (column), the vertical (horizontal) variate is held constant with the horizontal (vertical) variate changing from one panel to the next. Layout, then, refers to the task of positioning views relative to each other. Clearly, layout is a key step in organising information graphically.

3.1 Basic Data Displays

We begin with the familiar two dimensional point cloud. Point cloud layout is the positioning of each constituent point symbol according to values for the x and y variates. Positioning is carried out in the coordinate system of the variates themselves. When the point cloud is drawn, the point symbol locations in variate coordinates are transformed into viewport locations in screen coordinates. Thereafter we describe layouts where views are positioned within a *bounding region* in some abstract coordinate system.

Views such as the `2d-pointcloud` that require two values per case are collectively known as 2-d views. Other examples of 2-d views are the `lines-view`, which has line segments connecting consecutive pairs of coordinates, ordered by the x values, the `fitted-line`, which uses the coordinates to compute a (by default, least squares) fitted line, and a `smooth`. Similarly, views such as a `histogram-view` and a `boxplot-view`, which use one coordinate per case are collectively known as 1-d views and a `rotating-point-cloud` is a 3-d view. In summary, a p-d view is a view that given a dataset and p variates, calculates p values per case.

3.2 Plot Layout

Typically, the component subviews of a scatterplot are two axes, three labels and a 2-d point cloud. The scatterplot lays out these components by positioning the axes and labels in the left and bottom margins, and the title centred at the top border, leaving most of the space for the point cloud. The responsibility of the scatterplot for layout stops here. Layout is a recursive task and it is up to each of the scatterplot subviews to layout their own subviews when present.

¿From the perspective of laying out the scatterplot, the point cloud could just as well be replaced by a `lines-view` or even a `histogram-view` or a `density-view`. Similarly, some other kind of view – perhaps a `boxplot-view` – could be used in place of the axes in the margins. In fact, we have defined a type of view – called a `plot` – of which the scatterplot is a special case, whose task it is to position various views in the margins around an interior view. The plot format encompasses many different kinds of plot, since there is no restriction on the type of view that may be placed at each location. Typically, only the left and bottom margin views are present, and these are axes. More generally, plots allow for multiple interior views which are overlaid on the interior region, similarly for left, bottom, top and right views. Of these, the "layers" of interior view are by far the most useful and interesting: one can superimpose smooths or a least-squares fit on a point cloud, or overlay a `density-view` on a `histogram-view` for instance.

For the scatterplot to be meaningful there must be some connection or "glue" between its constituent point cloud and axes. For instance, a point symbol at $x = 0$ should line up with the x-axis tic mark at 0. More precisely, the transformation from abstract to screen coordinates for x locations should be the same for the point cloud and the x-axis, (similarly for y locations and the y-axis.) In our system, this occurs because (i) the scatterplot assigns locations to its subviews so that they line up in the appropriate way and (ii) a constraint system ensures that the horizontal extent of the interior view is the same as the horizontal extent of the bottom and top views; similarly for vertical extents.

3.3 Grid Layout

Grids are a compact way of arranging multiple displays of data, and the layout allows for easy comparison of displays across rows and columns. A `grid-layout` is a view that positions its subviews in a grid format. No restrictions are placed on the types of these subviews.

As with plots, grid layouts also support overlays where multiple subviews may be superimposed on a tile. This is useful, for instance, when showing point clouds with superimposed fits. Grid layouts, then, offer a very general way of arranging arbitrary subviews in a grid format. In many common applications, there is a consistent pattern across the subviews, where all are of the same type but show different variate or case subsets from a dataset. We have designed some particular kinds of grid layouts for these situations, as summarised in the table below:

Layout type	Data	Format	Default subview
`1d-layout`	n variates	row/column of n	`boxplot-view`
`xy-layout`	n x-variates, m y-variates	m rows, n columns	`2d-pointcloud`
`pairs-layout`	n variates	n rows, n columns	`2d-pointcloud` `label on diagonal`
`batch-layout`	data subsets eg levels of k factors	k=1 row/column $k > 1$ grid	`boxplot-view`
`table-layout`	data subsets	One tile per subset Tiles are scaleable Default: size $\propto$ subset size	`bar`

Like plots, `grid-layouts` need a constraint system or "glue" to ensure that meaningful comparisons between the subviews are possible. The constraint system used by `grid-layout` is necessarily richer than that of `plot`; it has options which require all, or none, of the subviews to have the same horizontal (or vertical) extents, furthermore constraints can be imposed separately on views across rows or down columns.

The `grid-plot` is a special kind of `plot`, where the interior view is required to be a `grid-layout`. The `grid-plot` acts as a wrapper around grid-layouts that will provide associated axes and labels, in the same way that the basic `plot` does for the 1 and 2-d views; in fact a scatterplot matrix, is actually a `grid-plot` with a `pairs-layout` as its interior view. Grid plots have a constraint system similar to grid-layout, except here the constraints apply to the margin views of the grid plot and the subviews of the grid layout.

4 MEDIATION

Triadic information, according to Peirce, describes the relation between two things mediated by a third. There are many ways in which mediation can be used in statistical graphics. Generally, mediation involves comparison of plots displayed simultaneously on the screen or page, or over time on the screen as in dynamic graphics.

In a statistical context, the most common form of mediation is by the value of some categorical variate. For example, a categorical variate Z mediates the relationship between the Y and X variates of a scatterplot when point symbols are coloured according to their Z values. We could also show the relationship between two variates mediated by a third by laying out a grid of pointclouds, one for each distinct value of Z. A mechanism for producing these kinds of plots is the so-called `batch-plot`, a kind of `grid-plot` whose interior view is a `batch-layout` with associated axes and labels in the margins.

Much of interactive statistical graphics involves some form of mediation. Brushing "linked" plots is perhaps the best known example. In the Quail system, there are two different implementations of linking. The first (see Hurley and Oldford, 1991) is the most efficient but also the most restrictive. At any time, a single view object can be a component of more than one compound view. The scatterplot matrix makes use of this feature: there is one point symbol for each case and this point symbol is a component of all the point clouds in the matrix. Since there is just one point symbol for the case, the point symbol has just a single drawing style and the change of highlight status via the brush in one location is automatically reflected in the other point symbol locations.

The second linking implementation (Hurley, 1993, 1997) is far more flexible. Basically, any two simple views can be linked to each other, with the result that a change of drawing style in one view is reflected in a change of drawing style in the second view. In the default mode of operation, two simple views are linked to each other if they display "the same" data, but, more generally, the user controls which views should be linked by choosing or specifying a function which, given a pair of views, decides whether or not they should be linked.

5 CONCLUDING REMARKS

The purpose of this paper was to give a flavour of the philosophy of statistical graphics underlying the software design in Quail. The software itself is more extensive than could be detailed here, and more importantly, the architecture is open-ended so that other graphical displays beyond those provided can be constructed; Oldford (1997) gives an example of implementing a statistical strategy using the graphical and computing facilities of Quail. The Quail system itself is based on the Common Lisp programming language, and a programming environment (either Macintosh Common Lisp from Digitool or Allegro Common Lisp from Franz, Inc). For details on availability, etc, see `http://setosa.uwaterloo.ca/~ftp/Quail/Quail.html`.

REFERENCES

Hurley, C. (1997) 'Brushing Tools for Multilevel and Multiway data.', *American Statistical Association, Proceedings of the Section on Statistical Graphics*, pp 15–24.

Hurley, C. (1993) 'The Plot-Data Interface in Statistical Graphics', *Journal of Computational and Graphical Statistics*, vol. 2, no. 4, pp.365–379.

Hurley, C., and Oldford, R.W. (1988) 'Plots as Hierarchical Views of Statistical Objects", 19 minute video (VHS format), STAT-22-88, Statistics Technical Report Series, University of Waterloo, Waterloo, Ont. Also available from the ASA sections on Statistical Computing and Graphics video library.

Hurley, C., and Oldford, R.W. (1991) "A Software Model for Statistical Graphics", In *Computing and Graphics in Statistics*, IMA Volumes in Mathematics and its Applications, vol. 36, Buja, A.,Tukey, P. (editors), New York: Springer-Verlag.

Oldford, R.W. (1997) 'Computational Thinking For Statisticians: Training by Implementing Statistical Strategy', *Computing Science and Statistics: Proceedings on the Interface*, vol. 29, pp. 88–97.

Wainer, H. (1989) Comments on 'Whither and Whence', ASA Sesquicentennial Invited Paper Sessions, pp. 382–390.