Interactive Data Analysis in a Manufacturing Setting – A Case Study

David A. James

Bell Laboratories, Lucent Technologies
700 Mountain Ave
Murray Hill, NJ 07974

Summary

We present a case study in the application of interactive data analysis in the context of semiconductors and optical fiber manufacturing. Our approach to interactive graphics is based on well-known principles of data analysis and interactive graphics (Chambers, Cleveland, Kleiner & Tukey 1983, Cleveland 1993, Tufte 1983, Tufte 1990, Stuetzle 1987, Becker, Cleveland & Weil 1988*b*, Becker, Eick & Wilks 1991, Young & Lubinsky 1995, Buja, Cook & Swayne 1996), and its implementation is written in the S-language for data analysis and graphics (Becker, Chambers & Wilks 1988*a*, Chambers & Hastie 1992).
The case study also highlights the need for improved interactive software tools in S-based systems to support the type of programmable interactive graphics that we envision. These limitations have motivated us to explore some recent developments in software technology, such as S Version 4 (Chambers 1998), Java, LispStat (Tierney 1996), XploRe (Haerdle, Klinke, Turlach & Ardle 1995), CORBA (Siegel 1996), and PVM (Geist, Beguelin, Dongarra, Jiang, Mancheck & Sunderam 1994), and asses how they can be used to alleviate some of the deficiencies we see in existing tools.

Keywords: Interactive data analysis, integrated circuits, optical fiber, linked plots, object-based programming

1 Introduction

Increasingly graphics is permeating all aspects of modern data analysis, from the exploratory phase to model building and validation to final communication of results to the consumers of information. Interactive graphics, in particular, are playing an expanding role in our own industrial environment, since it is now widely recognized that interactivity is a powerful mechanism for studying complex systems (Stuetzle 1987, Becker & Cleveland 1987, Becker et al. 1988*b*, Young & Lubinsky 1995, Buja et al. 1996). However, good interactive graphics often need to be tailored to the application under study to exploit contextual information that general purpose tools may not be able to easily incorporate, for instance, displays such as traffic on a telephone network (Becker et al. 1991), the location of changed code on software listings (Eick 1994), and wafer geometry as in Section 2.1. Therefore there is still a need for programmable interactive tools to support quick development of interactive statistical applications.

Section 2 illustrates how interactive graphics is used to monitor manufacturing data as a means of early problem identification and resolution. Section 3 goes behind the scenes and discusses the graphical linking infrastructure behind our interactive displays. Finally, Section 4 discusses deficiencies in our current implementation and reviews evolving software technologies that may alleviate some problems in our current environment.

2 Case Study

The following case study describes some of the interactive data analysis tools that we have developed as part of our collaborations with Lucent Technologies (formerly AT&T) Microelectronics and Transmission Systems manufacturing facilities.

2.1 Integrated Circuit (IC) Manufacturing

The manufacturing of IC's is a high-volume, complex process that lasts several weeks and it is conducted in a clean room environment. Electronic devices are created by the hundreds, arranged in a rectangular grid on silicon disks called *wafers*; wafers are manufactured in groups or batches called *lots*.

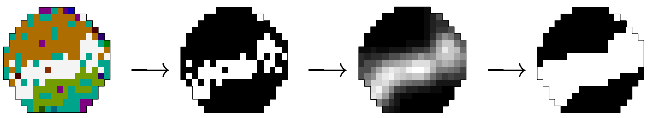

Figure 1: *Wafer data: binned, binary, smoothed, and thresholded wafers.*

While the processing can be quite different depending on the particular IC product being fabricated, a common feature is that wafers consist of many individual devices, each device a micro-chip to be sold.

At the end of production, every chip on each wafer is subjected to a sequence of functionality tests; each chip is assigned to a "bin" corresponding to the first test it fails, or considered "good" if it does not fail any test. For instance, Figure 1 depicts the outcome of testing one wafer. The various grey levels on the first wafer depict different failure modes, and white denotes working chips. The second wafer depicts a binary representation of the same physical wafer where black denotes any failure mode. The third wafer is a smoothed version of the previous binary data, and the last wafer consists of binary data computed by thresholding the smoothed wafer.

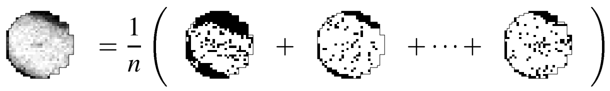

Figure 2: *A composite wafer is the sitewise average of a sample of wafers. Composite wafers are widely used to ferret out systematic patterns of defects in lots.*

Since many lots finish manufacturing every day, it is necessary to implement simple monitoring procedures that give engineers a realistic picture of the state of the production. Traditional lot yield summaries (proportion of defective chips) hide important spatial information that simple visualization techniques, such as augmenting boxplots with wafer composites (Figures 2 and 3) reveal, helping engineers focus their attention on problem lots and even on those lots with acceptable yield but systematic loss.
The composite wafers that are superimposed on the boxplots represent sitewise averages for the lower 25%, middle 50% and upper 25% of the yield distribution. Sitewise wafer averages easily reveal systematic defect patterns in a group of wafers (e.g., the two parallel stripes of defective chips at the top and bottom of the wafers

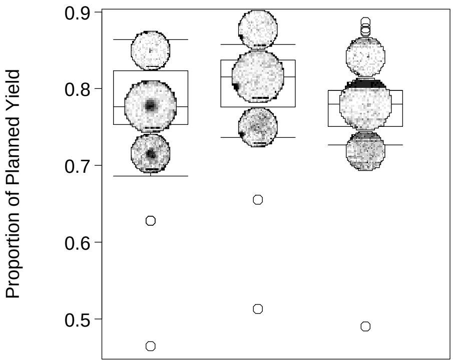

Figure 3: *Boxplots augmented with composite wafers easily incorporate spatial information. The boxplots show the distribution of yield for three lots; the composite wafers represent the lower 25%, middle 50% and upper 25% of the yield distribution. Notice that although somewhat similar with respect to yield, these three lots exhibit quite different patterns of defective chips.*

in Figure 2.)

Engineers routinely browse Web-based augmented boxplots like Figure 3 of recently finished wafers to identify "interesting" lots as an early warning of production problems. Once they select a lot for study, engineers take a closer look at the various failure modes to study how they are distributed over the surface of the wafers and compare these patterns of defective chips with known equipment and processing characteristics. Buja et al. (1996) refer to tasks like these as "finding Gestalt" and "posing queries," respectively.

Although possible with static displays, interactive tools allow engineers to immediately see whether failure modes cluster on wafers, and thus efficiently construct a map detailing spatial correlation among failed chips. This information, in turn, often suggests possible causes and corrective actions. In Figure 4 we show a snapshot of such an interactive tool.

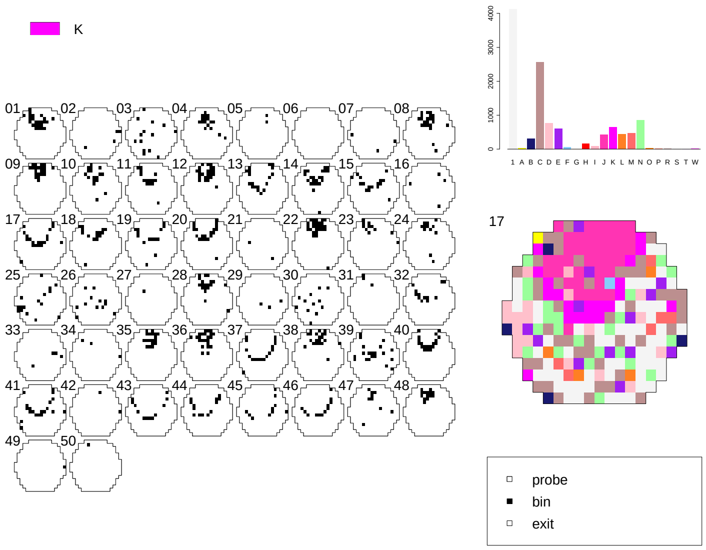

Figure 4: *Investigating the spatial distribution of various failure modes on a lot of wafers.*

On startup the large display on the left panel depicts all wafers of a given lot in full color. The smaller bar plot on the top-right displays the number of chips that fall in each of several failure categories plus the number of good chips (left-most bar labelled as "1"). Engineers select failure categories to study by clicking on the corresponding bar; this click causes a link to be instantiated between the barplot and the wafer plot on the left. The link causes the wafer plot to be updated by applying a transformation to the wafers — chips that belong to the selected category are colored in black and all others in white. The lower right panel allows the engineer to reset all failure modes by clicking on the "probe" button, which instantiates another link to the wafer plot. Finally, users find it convenient while browsing binary wafers to display all failure modes of a single wafer, and this is naturally done by clicking on the wafer of interest. This click instantiates a link to the middle right panel, which displays the selected wafer with all its failures in color. From this snapshot we clearly see that failures of mode “K” are *not* randomly distributed over the wafer surface.

What this snapshot fails to show clearly — and interactivity reveals easily — is the spatial nesting of failure modes. Failure mode “D” is the innermost cluster, which is nested in mode “E”, and these two nested in “K”, . . . “J” and “C”. This nesting of failure modes can be explained by the fact that some failure modes are more severe manifestations of similar intrinsic (unmeasured) physical defects, together with the fact that testing is sequential and is stopped on first failure.

Another strategy (Hansen & James 1997) commonly employed to narrow down root causes of manufacturing problems is to group wafers with similar defect patterns, and then look for steps on which wafers within groups were processed close in time or under the same conditions (for instance, the same chamber on a specific oven). Figure 5 is a snapshot of a wafer clustering tool. In this example we have assigned the value of one to failed chips and zero to working chips. We want to group wafers according to their spatial distribution of failed chips. Reasonable data to use for clustering can be the raw binary data, but we may want to explore smoothed and thresholded data as well (see Figure 1).

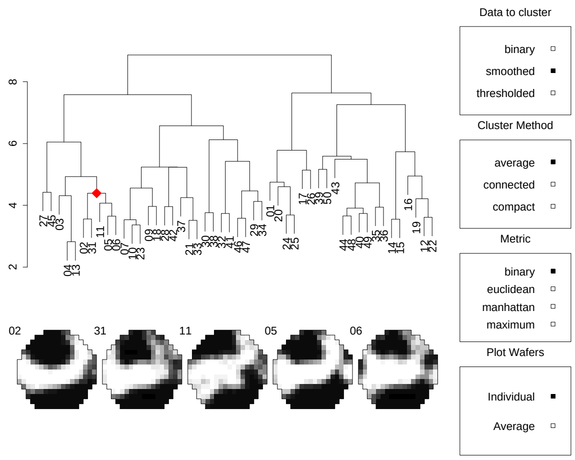

Figure 5: *Grouping wafers according to their spatial distribution of defects.*

The largest panel in Figure 5 depicts a dendrogram computed by a hierarchical agglomerative algorithm (Kaufman & Rousseeuw 1990) according to the data, metric, and clustering method indicated on the various panels on the right. Since hierarchical clustering algorithms do not produce a fixed number of groups (which in our experience with IC data is typically unknown), the user needs to explore various possible groupings before proceeding to further analyses; this exploration is done by clicking on the internal nodes of the dendrogram. The selected node is highlighted by plotting a red diamond on it. As in our previous example, this click instantiates a link between the dendrogram and the lower wafer plot according to the state defined in the lowest right panel. This state is either plotting of all wafers under the selected node or plotting of the composite wafer for the node.

The goal of this exercise is to define a "reasonable" set of wafer groups according to the spatial distribution of defective chips. These groups are then used in a search for possible process or equipment culprits. For more details see (Hansen & Nair 1997, Hansen & James 1997, Friedman, Hansen & Nair 1997, Denby & James 1995).

2.2 Optical Fiber Manufacturing

The first stage in the manufacture of optical fibers by the modified chemical vapor deposition (MCVD) process consists of depositing vaporized chemicals in the interior of a glass tube and collapsing the tube to a solid rod to produce a *preform*; the deposited doped silicon forms a glass core inside the glass tube, and geometrical and optical properties of the core and outside tube determine various quality characteristics of the final fiber.

Preforms are built on machines called "lathes" (the term refers to these machines' resemblance to carpenter's wood lathes). Measurements on a preform are made immediately after manufacturing in order to characterize its geometry and the ability of the subsequent fiber to guide light; machine number (lathe), date, an index of "ovality", core diameter, ratio of the inner core to outside diameter, differences of key features of the reflective index in the core and outside tube are some of the data that are routinely recorded.

The location of the lathes on the factory floor is important both from a product-flow and an environmental point of view; atmospheric particles and impurities in the input gases can contaminate the glass and adversely affect future manufacturing steps. Thus it is useful to display the location of the lathes during initial exploratory analyses. A map of the factory, besides being familiar to engineers, thus provides a natural canvas on which to encode the available data.

In Figure 6 we show a snapshot of a data browser depicting the factory floor on the upper left panel. A list of metrics available for visualization is displayed on the upper right panel. The lower left panel displays a time series for the selected metric and selected lathe. The last panel shows two boxplots, one for the data corresponding to the chosen lathe and the other for all available data. The rectangles on the factory floor encode the metrics' mean, standard deviation, and sample size as the location, height, and width of each rectangle. You may notice a slight tendency of lathes to drift in adjacent pairs by looking at the rectangles' vertical offsets from their target levels (these targets are denoted by the small tick marks on the aisles' perimeter).

This interactive display shows that lathe 129 had an average `ccratio1` that is below the factory's average and a slightly higher standard deviation; clicking on that rectangle we see that it was running consistently low for almost half the time period. Engineering intervention then brought the process closer to its target.

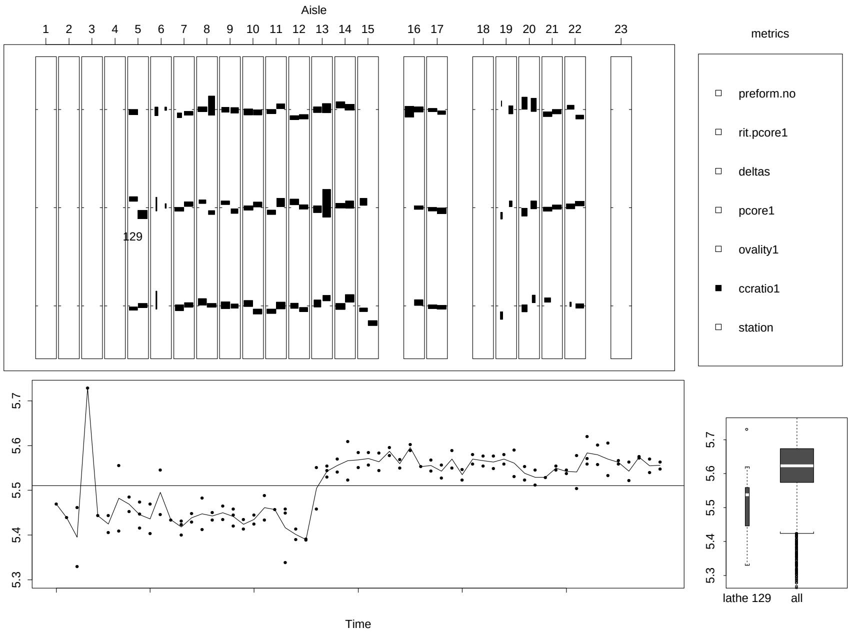

Figure 6: *A map of the factory showing the location of the various lathes. The rectangles encode the selected metrics' mean, standard deviation, and sample size. The time series panel on the bottom shows a change in the level of* `ccratio1` *for lathe 129. The boxplots in the lower-right panel compare the distribution of* `ccratio1` *between lathe 129 and all other lathes.*

This interactive display thus allows us to easily visualize data over the factory, peruse individual lathes with potentially "interesting" features by clicking rectangles of interest, and comparing its data over time and to the overall lathe population. Clicks on the time series plot allow us to restrict the data to specific time periods.

A similar data browser is displayed for different data in Figure 7. This browser is more traditional in the sense that it implements scatterplots of lathe means and standard deviations depicted as "crosses"; clicking on a "cross" instantiates links to a scatterplot of the data for the corresponding lathe in the lower left panel and time series on the remaining panels. Although simple, tools like this one make browsing a moderate number of metrics an interesting and simple exercise.

Being able to easily put together interactive displays that exploit geography, wafer spatial information, hierarchical groupings, time, or any other contextual information meaningful to the data analyst can help end-users extract information more quickly from their data. For instance, interactivity helped us see the relatively complex nesting of failure modes on wafers, investigate wafer groupings as computed by agglomerative clustering, browse fiber quality metrics over the plant. From these various examples we see that there are significant advantages in providing pro-

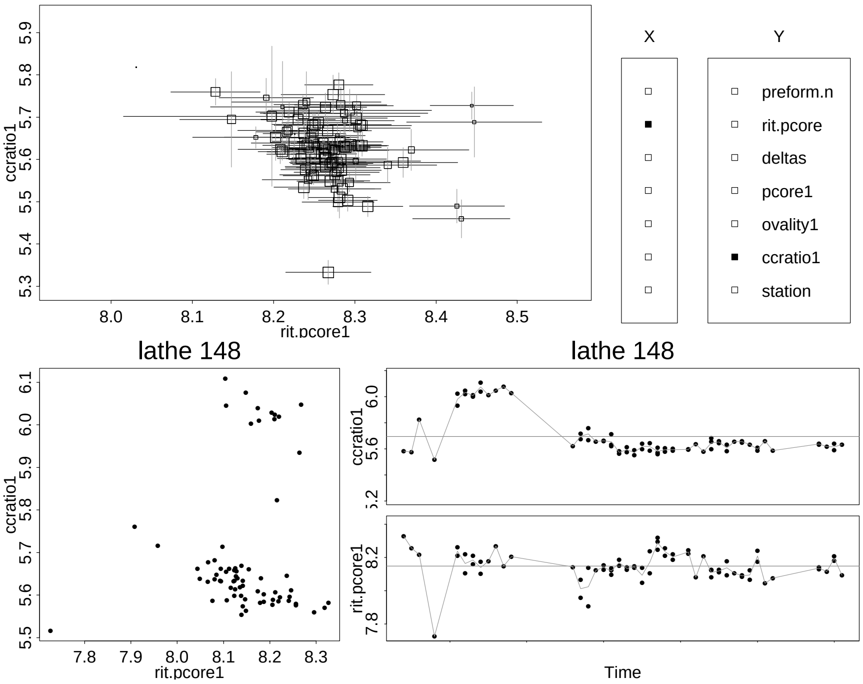

Figure 7: *Crosses depict means, standard deviations, and sample sizes for variables* `rit.core` *and* `ccratio1` *for all lathes. Clicking on a cross instantiates links to the other panels to display the data from the selected lathe as a scatter plot and as time series.*

grammable interactive tools in a standard data analysis toolbox.

3 Behind the Scenes

Some of the underlying principles behind our examples are well documented (Becker et al. 1988*b*, Stuetzle 1987, Becker & Cleveland 1987, Young & Lubinsky 1995, Buja et al. 1996). These ideas include "finding Gestalt," linked plots, encoding ancillary information into glyphs (Tufte 1983, Tufte 1990), multiple views on the data, etc. A general purpose linking prototype system was developed by Daryl Pregibon, Linda A. Clark, and David A. James, and it provides the infrastructure used in implementing the above examples.

Linking in S requires, in addition to the object-oriented environment, the three key ingredients identified in various of the above references: multiple views on the data, user interaction with these views, and links among the data views.

In order to allow fully extensible tools for interactive analysis, we need to allow the programming of the semantics of graphical interactions and the actions (links). In other words we need to be able to adjust the response to mouse events depending on the type of data display the user is interacting with. In some cases, clicking on a point in a scatterplot may highlight the same observation on another data display, as in the case of scatterplot brushing (we refer to this as case linking and Young (1994, Revised 1996) calls it *empirical* linking); in other cases we may want to extract other aspects or statistics that a point represents. For instance, in a C_p plot (Mallows 1973) clicking on a point may be interpreted to mean "produce a normal quantile plot of the residuals of the model associated with the clicked point" or "produce a Cook's distance (Cook & Weisberg 1982) plot," (this is referred as one-to-many linking, *algebraic* linking, or *logical* zooming.) More general links, as in the dendrogram case, are defined in terms of computations in the base language — S, LispStat, or other. However, to accomplish this level of programmability we need a rich object-oriented language in which to express arbitrary numerical and graphical computations.

The ViSta system (Young 1994, Revised 1996) written in LispStat implements many of these ideas (in addition to providing guided statistical analysis), although its primary application is in teaching statistics. Cook & Weisberg (1994) provide a very useful set of interactive tools called *R-code* for regression graphics, also written in LispStat. More recently, the XploRe system (Haerdle et al. 1995) also provides linked plots.

In our applications, a link object consists of the following three slots:

- a `split.screen` (Chambers & Hastie 1992) specification to carve up the graphics windows into "screens" on which each data view is displayed;
- a set of initializing S expressions (possibly null) for each display; and
- a set of expressions that represent messages that a displayed object sends to other displayed objects.

Instantiating linked objects is simply done by (1) dividing up the graphics window into screens, (2) evaluating the initializing expressions inside each screen, and (3) evaluating the links by recognizing graphical input (mouse clicks). On exit, an object with expressions corresponding to all the user's interactions is returned. This object may be used to play back the analysis.

In practice, the above approach to programmable interactive data analysis inherits a number of deficiencies from current computing environments, in particular the S-based systems R (Ihaka & Gentleman 1996), S-Plus (S-P 1995), and Bell Labs' S (Becker et al. 1988*a*). LispStat and XploRe seem to provide a better graphical environment for developing interactive tools like these.

S-based systems have not integrated their graphics subsystem closely with the host windowing systems. This creates problems, since input to the S interpreter is suspended until the user finishes graphical input. Furthermore, graphical input consists only of clicks — operations such as dragging or simply reading mouse positions are unavailable. Standard user interface widgets, such as buttons, list boxes, dialog boxes, and menus, are either lacking or very primitive. Some of these limitations can be circumvented by developing S code, like the button objects in the examples above, but their aesthetics and functionality are poor.

A related problem with S-based systems is their reliance on an older graphics paradigm. This older graphics model supports static graphics rather well, and produces very good publication-quality output, such as Trellis displays (Becker, Cleveland & Shyu 1996), but it lags behind in interactivity. Thus, plotting functions are primarily used for their side effects, and as a matter of course they do not return any object that may capture the essence of what the plot just produced contains. For instance, the expression

```
> p <- plot(x, y)
```

produces a scatter plot for the `x` and `y` vectors, but the value of the object `p` is NULL. We can add graphical elements to the plot (lines, curves, legends, etc.), but the graphical state and its data are captured nowhere. It is either impossible or very hard to find out whether a particular display contains, say, a legend, or the class of data being displayed, or whether the current display has a scatterplot, a time-series, or a dendrogram. This information is critical to allow object-oriented tools adjust their behavior to the class of displayed graphical objects.

Thus, for our application it was necessary to write wrapper functions around most common plotting functions to return meaningful objects with appropriate classes. For instance,

```
> scatterPlot <- function(x, y, ....)
{
   plot(x, y, ....)
   out <- list(x = x, y = y, call = match.call())
   class(out) <- "scatterPlot"
   return(out)
}
```

which returns enough information to have the S interpreter dispatch the proper method during graphical input. Note that languages like LispStat and XploRe produce bona-fide graphical objects, so these issues are only relevant to S.

Links among graphical displays are specified as unevaluated S expressions and stored in a $k \times k$ array E, where k refers to the number of graphical displays. Upon clicking on the $i'th$ display, the software evaluates the expression on the $E[i,i]$ diagonal, which causes the $i'th$ displayed object to be updated (e.g., to highlight a selected metric on a buttons panel); then the off-diagonal expressions on the $i'th$ row are sequentially evaluated from left to right (i.e., lower display number to higher display number). The object `.PT` in the evaluating frame (Becker et al. 1988*a*) fully describes the graphical interaction, storing both the display that was activated as well as the point (x,y) that was selected. Similarly the object `.XY` stores the graphical objects displayed on each of the k views or screens.

The main advantage of this approach is its programmability. Any type of links, not only *empirical* and *algebraic*, may be implemented by defining the links as S expressions.

4 Discussion

Interactive data analysis is very useful in a complex manufacturing setting. It provides a simple mechanism for visualizing moderate amounts of data and uncovering information that static displays sometimes cannot easily reveal. Besides the convenience of browsing moderately large amounts of data (both in terms of variables and observations) interactive graphics has allowed us to communicate the results of our analyses to our colleagues and managers quite effectively. The scaffolding we have built on which we developed the above interactive displays is somewhat unsophisticated, and the application tools could be improved with various recent advances in technology, such as S Version 4(Chambers 1998), Java, threading in S (Temple Lang 1997), various implementations of distributed computing, such as CORBA (Siegel 1996) PVM (Geist et al. 1994), or Java's Remote Method Invocation. All these developments seem quite promising for creating an environment where one can quickly and easily build objects that implement fairly sophisticated interactive displays by re-using the best of each system: statistical modeling in one system, visualization in, say, an OpenGL-based application, all communicating by means of well-defined object interfaces and coordinated on a distributed environment by a standards based object requester and broker. Some preliminary work on distributed computing and interactive statistics includes an S-Java interface by John M. Chambers and Mark Hansen, and described at

`http://cm.bell-labs.com/stat/project/java/html/`.

Also see the UCLA Electronic Statistics Textbook by de Leeuw at

`http://www.stat.ucla.edu/textbook`.

A Java client/server for XploRe is discussed at

`http://www.xplore-stat.de`.

Tierney (1996) also identifies some of these same technologies as possible extensions to LispStat. Chambers, Hansen, James & Temple Lang (1998) explore similar issues in distributed computing with data.

5 Acknowledgements

Daryl Pregibon, together with Linda A. Clark and David A. James, designed the tools for linking graphics used to implement the interactive displays shown throughout the paper. Lorraine Denby implemented many of the ideas in this paper using a VisualBasic(Vis 1994) front-end to S-Plus, see (Denby & James 1995). The projects behind this case study have been joint work with Mark Hansen (Bell Labs), Vijay N. Nair (U. of Michigan) and Daryl Pregibon (now at AT&T Labs). I also want to thank Linda A. Clark for her many comments and suggestions on this paper.

References

Becker, R. A. & Cleveland, W. S. (1987), 'Brushing scatterplots', *Technometrics* **29**, 127–142.

Becker, R. A., Chambers, J. M. & Wilks, A. R. (1988*a*), *The New S Language*, Wadsworth, Pacific Grove, California.

Becker, R. A., Cleveland, W. S. & Shyu, M.-J. (1996), 'The visual design and control of Trellis display', *Journal of Computational and Graphical Statistics*.

Becker, R. A., Cleveland, W. S. & Weil, G. (1988*b*), The use of brushing and rotation for data analysis, *in* 'Dynamic Graphics for Statistics', pp. 247–275.

Becker, R. A., Eick, S. G. & Wilks, A. R. (1991), 'Basics of network visualization', *IEEE Computer Graphics and Applications* **11**(3), 12–14.

Buja, A., Cook, D. & Swayne, D. F. (1996), 'Interactive high-dimensional data visualization', *Journal of Computational and Graphical Statistics*.

Chambers, J. M. (1998), *Programming with Data: A Guide to the S Language*, Springer, New York.

Chambers, J. M. & Hastie, T., eds (1992), *Statistical Models in S*, Wadsworth, Pacific Grove, California.

Chambers, J. M., Cleveland, W. S., Kleiner, B. & Tukey, P. A. (1983), *Graphical Methods for Data Analysis*, Wadsworth, Pacific Grove, California.

Chambers, J. M., Hansen, M. H., James, D. A. & Temple Lang, D. (1998), Distributed computing with data: A corba-based approach, *in* 'Computing Science and Statistics', Inteface Foundation of North America.

Cleveland, W. S. (1993), *Visualizing Data*, Hobart Press, Summit, NJ.

Cook, R. D. & Weisberg, S. (1982), *Residuals and Influence in Regression*, Chapman and Hall, New York.

Cook, R. D. & Weisberg, S. (1994), *An Introduction to Regression Graphics*, Wiley, New York.

Denby, L. & James, D. A. (1995), A graphical user interface for spatial data analysis in integrated circuit manufacturing, *in* M. Meyer & J. Rosenberg, eds, 'Computing Science and Statistics', Vol. 27, Interface Foundation of North America.

Eick, S. G. (1994), 'Graphically displaying text', *Journal of Computational and Graphical Statistics* **3**(2), 127–142.

Friedman, D. J., Hansen, M. H. & Nair, V. N. (1997), 'Monitoring wafer map data from integrated circuit fabircation processes for spatially clustered defects', *Technometrics*.

Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Mancheck, R. & Sunderam, V. (1994), *PVM: Parallel Virtual Machine. A User's Guide and Tutorial for Networked Parallel Computing*, MIT Press, Cambridge, MA.

Haerdle, W., Klinke, S., Turlach, B. A. & Ardle, W. H. (1995), *XploRe: An Interactive Statistical Computing Environment*, Springer Verlag, New York.

Hansen, M. H. & James, D. A. (1997), 'A computing environment for spatial data analysis in the microelectronics industry', *Bell Labs Technical Journal* **2**(1), 114–129.

Hansen, M. H. & Nair, V. N. (1997), 'Process and productivity improvement in semiconductor manufacturing', *Computing Science and Statistics* **27**, 3–7.

Ihaka, R. & Gentleman, R. (1996), 'R: A language for data analysis and graphics', *Journal of Computational and Graphical Statistics* **5**(3), 299–314.

Kaufman, L. & Rousseeuw, P. J. (1990), *Finding Groups in Data*, Willey, New York.

Mallows, C. L. (1973), 'Some comments on Cp', *Technometrics* **15**, 661–675.

S-P (1995), *S-PLUS Software Documentation*. Version 3.3.

Siegel, J. (1996), *CORBA Fundamentals and Programming*, Wiley, New York.

Stuetzle, W. (1987), 'Plot windows', *Journal of the American Statistical Association* **82**, 466–475.

Temple Lang, D. (1997), A Multi-Threaded Extension to a High Level Interactive Statistical Computing Environment, PhD thesis, University of California, Berkeley, Berkeley, CA.

Tierney, L. (1996), 'Recent development and future directions in LispStat', *Journal of Computational and Graphical Statistics* **5**(3), 250–262.

Tufte, E. R. (1983), *The Visual Display of Quantitative Information*, Graphics Press, Cheshire, Connecticut.

Tufte, E. R. (1990), *Envisioning Information*, Graphics Press, Cheshire, Connecticut.

Vis (1994), *Visual Basic Language*. Version 3.0.

Young, F. W. (1994, Revised 1996), *ViSta: The Visual Statistics System*, L.L. Thurstone Psychometric Laboratory, Chapel Hill, NC. 94-1.

Young, F. W. & Lubinsky, D. J. (1995), 'Guiding data analsys with visual statistical strategies', *Journal of Computational and Graphical Statistics* **4**, 229–260.