# Towards Ubiquitous Brushing for Information Visualization

Jonathan C. Roberts,
Computing Laboratory,
University of Kent, UK
J.C.Roberts@kent.ac.uk

Michael A. E. Wright
School of Computer Science,
University of Nottingham, UK
maw@cs.nott.ac.uk

## Abstract

*Brushing is a collection of techniques to dynamically query and directly select elements on the visual display. Such interaction allows the user to explore the visualization, to interactively select a subset of points and see how these changes are updated in other related views. Traditionally, the artefacts that are 'brushed' are the plotted elements in the visualization (e.g. the points on a scatterplot, or the bars of a bar chart). In this paper we discuss the concept of Ubiquitous Brushing (UB), which brings together various different types of selection (whether in data space, screen space or views). Not only can users brush over elements in the display but also they can brush over various meta-information such as menus, legends or axis to affect the highlighted elements. The paper discusses the basic idea and demonstrates how subsequent UB operations can be compound together to provide useful dynamic filter operations.*

*Keywords*— **Brushing, Exploratory Visualization, Multiple Linked Views**

## 1 Introduction

Brushing is used in almost every interactive visualization environment. Typically, the technique is simple; the user moves the brush to different positions on the plotted graphic to select (and highlight) certain elements that are bounded by the brush. By doing so the user is able to graphically highlight and hence investigate various aspects of the data. Brushing is an extremely important investigative tool because the queries are instantaneous, linked with other representations, which gives insight into how they appear in other dimensions, and above all directly applied to the visualization.

The origins of brushing can be traced back to the PRIM-9 system by Tukey et al [7] but it was Becker and Cleveland who defined the term "brushing" in their seminal paper on "Brushing Scatterplots" [2]. They state "The salient features of brushing are visual manipulation of the graph by the analyst, instantaneous change, the movement of the brush to different positions on the scatterplot matrix, the ability to change the shape of the brush ... The brushing methodology provides a medium within which a data analyst can invent data analytic methods, which we call brush techniques".

Over the years various researchers have indeed invented and extended this basic idea, developing new brush techniques (see section 2). However, all these techniques only allow the user to brush on the plotted data points. In this paper we introduce "ubiquitous brushing". The concept is that it utilizes the whole display: allowing the user to brush over any element of the interface, whether plotted elements, labels or menus. Obviously, these additional components provide the context for the actual visualization; hence, when used in a compound brush the user can instantaneously view how these items relate to other elements. This is significant as some operations could be easier or quicker to perform through a UB operation, than through traditional techniques.

Ubiquitous brushing hence requires a meaning or operation to be attached to any part of the visualization system, such as data-table, menus or a legend, thus giving the user a rich and unique set of operations from which to view and amend the brushed elements. This allows the user to directly explore from one context to another. This breaks the common assumption that a brush can only be active on the plot itself. In fact, as we shall see, it is possible and useful to brush over this additional meta-information to accordingly change and update the highlighted elements. One way to implement UB is through the technique named 'click and brush' [24]; where a compound brush can be built up from a series of clicks – to fix the current information – and subsequent brushes to filter the operation.

## 2 Overview of brushing & Related work

Brushing has been called by different names; for instance, Buja et al [5] described it as "painting multiple views". However, the principles remain the same; that each of the realizations still assumes that the user will control,

in a visual manner, a brush of a certain shape and area, and that the results are simultaneously updated on each data plot (i.e. that the plots are dynamically linked).

Dynamic linked brushing is important: the user can brush in one view in one dimension and see the results of that operation in other dimensions in other views. These subsequent additional views may be different projections of the data all plotted in the same form, such as those provided by a matrix of scatterplots, or can be completely different forms. For example, systems like the early PRIM-9 [7], and tools by Becker and Cleveland [2] principally used scatterplots, while others like HyperSlice [19] provide a matrix of similar visualizations. Whereas, tools such as XmdvTool [20], Snap [11], Waltz [13], GeoVISTA [17] and CommonGIS [1] link complimentary visualizations. It is often by interacting with these different visualizations that the user gains a better understanding of the underlying data.

In fact, dynamically linked brushing is part of the wider more general field of multiple linked views. For example, Siirtola [14] looked at brushing multivariate data using a parallel coordinate graph and the reorderable matrix. For an in depth discussion of multiple linked tools and issues see the paper on "Exploratory visualization with multiple linked views" [12].

Developers over the years have created various brushes. The brush must provide a method by which to designate 'what is selected' and 'what is not selected'. Generally a bounding box or freehand lasso is used to encapsulate the required elements [23]. However the shape of the brushes can be point, line or area based. E.g. Stuetzle [15] allows the user to highlight individual points; Becker, Cleveland and Weil[3] describe a line-brush that selects points constrained to an axis; while many systems use a bounding box to select items of interest [5, 2, 16, 20]. Brushes are usually dynamically painted over the plot. For instance, Ward [21] describes an example which he names "direct, user-specified', where the centre of a bounding-box brush can be moved to the location of a mouse click, and the limits of the brush changed by selecting on the boundary of the brush.

Becker and Cleveland specify four brush operations highlight, shadow highlight, delete and label. Highlight, changes the visual appearance of the element in both the current view and any linked view (this is known as masking in the PRIM-9 system [7]); with shadow highlight, both the visual appearance of the selected elements is changed, and the other unselected elements are removed from other linked views (this is similar to the isolation operation that was implemented in the PRIM-9 system, which extracts any sub-region and displays it in isolation); Delete is the opposite to highlight in that elements are taken out of the display; whereas, Label displays additional information about the elements (usually a text label appears to the side of the selected elements).

Finally, some systems allow multiple brushes, where each new selection instance is displayed in a new color [21]. These multiple brushes can then be joined or subtracted from previous selected elements. This method is a 'compound brush' [6] and is most relevant to Ubiquitous Brushing. Wills [23] comprehensively describes different possible combinations in his selection calculus. It is a pity that compound brushing, especially using the selection calculus principles, has not been more widely adopted, perhaps because it is harder to implement memory based systems or it is hard for a user to ascertain which items were joined and how they were joined. Chen [6] provides a solution to the latter problem in their flexible compound brushing system based on higraphs. In their system various components of the system can be linked together via various logical operations and expressions (e.g. AND, OR, XOR, LESS, EQUAL etc). The higraphs provide a clear visual representation of the selection calculation.

## 3 Towards Ubiquitous Brushing

Let us imagine a user exploring a multidimensional dataset of cars. The developer may wish to generate scatterplot displaying the weight (x-axis), acceleration (y-axis) and number of cylinders (symbols), see Figure 1.

Our first observation is that there are many components to the display (the plot area, legend, axis and labels, menu etc.) and it is these components that provide the context to the plot data. For instance, x,y axis labels and tick marks allow the user to read off specific values, the legend provides meaning to the symbols, while menus give control over the current information.

It is possible to imagine a system that allowed the user to brush over the legend; brushing over the legend symbol could cause the associated elements to be highlighted, or displayed in isolation, or removed from the plot. Remember, the elements are instantly updated, so a user can move from one legend symbol to another and instantly observe the result. Additionally, one could envisage brushing over the axis. This could act as a beam-brush to select all values above the position of the pointer.

The second observation is that the exact meaning of a component is dependent on where it is or what is around it. For example, the button labelled "X" of a sub-window in a multiple window system, typically on the top right of the window, will close that specific sub-window down. Furthermore, values that are plotted in the same vicinity share similar traits and the elements themselves provide a context for other elements. In this situation the user may understand the data, not necessarily because of a legend, axis or other information, but because of adjacent and congruent information.

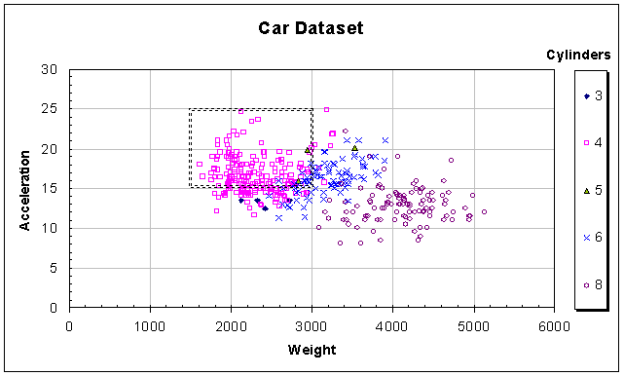


**Figure 1. A demonstration of a scatterplot visualization. (Realized using Excel).**

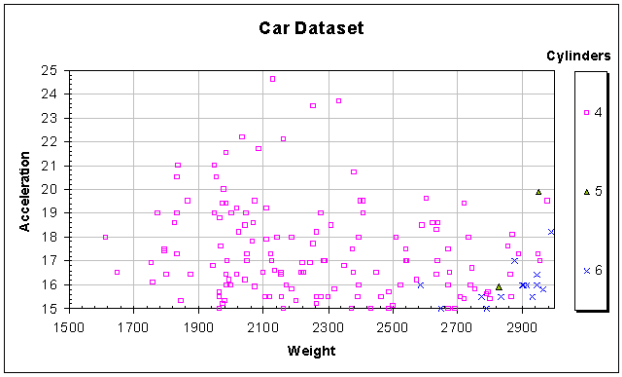


**Figure 2. A subset of the data selected from Figure 1, is shown in *isolation*. The user could brush over the legend to select every item of a particular type.**

Although brushing over legend symbols would highlight elements in one display and would clarify the visualization when many categories are represented, it would be most useful in multiple-view visualization where the information in linked-views would be simultaneously highlighted. This is because the meaning of each could be different. Consider when the user selects a sub-area which mostly contains 4 cylinder cars. These are displayed in isolation in a new view, as shown in Figure 2. Assuming that brushing operations in figures 1 and 2 are linked together; by brushing over the 4-cylinder symbol in the legend of Figure 1 all the 4-clynder elements would be selected, while brushing over the elements in Figure 2 only a subset of corresponding elements in Figure 1 would be highlighted in corresponding views.

Another idea could be to exploit the position of the plotted points, such that all elements, which are in the same vicinity to a selected point, would be highlighted. So for example, in a multi-window system the user could have created multiple windows with elements in *isolation*, as soon as the user moves into the window (i.e. brushes on the window) all the elements that are displayed in that particular window are instantaneously selected and highlighted in all corresponding windows.

Our third observation is that the information in the data-table or exact-details of some of these components change as the user explores. For example, as the user manipulates and explores the information through brushing (or any other manipulation technique), not only does the state of the individual elements change (0 for unselected, 1 for selected), but the specific legend for the new visualization would also change. Now, not only the data-table, that is used to create this view, contains a subset of the original data, but also the legend for the new view contains only three symbols. Also, both x,y axis are smaller and show more labels, and it is possible to see more individual items because they are shown at a larger scale. This zoomed view then can be subsequently brushed and explored.

Perhaps the user is interested in specific values of some points, and then the user can view details-on-demand where specific details appear in a popup at the request of the user. This popup would contain both labels and values. Hence, it is conceivable to imagine that not only the labels but also the new popup information could be brushed. By brushing over the label, any information that contains that variable could be highlighted; by brushing over the value, any data point that also has that value could be highlighted in any linked view. Furthermore, it is feasible to use this idea to operate compound brushes; i.e. brush operations that are made up of multiple sequential operations.

Let us analyze further these concepts. First, each of these examples assumes that a meaning has been applied to the component. Thus, one challenge to the developer is to apply an appropriate and meaningful operation to these components. In reality, it may not be appropriate to allocate a 'brushed' meaning to every part of the display, e.g. it is unclear what brushing over certain menu items such as 'File' would mean. Second, the area or hotzone of the elements to be brushed needs to be defined. This may be straightforward, such as in the case of symbols on the legend, or may be more complex such as brushing on the axis, because there may be multiple appropriate meanings. Third, it is possible to utilize ubiquitous brushing as multiple brushes. This is perhaps more interesting because, as we presented in our third observation above, the context often changes after one brushing operation, which could be used to explore multiple scenarios sequentially: brush some data, which would generate a new plot, which could be used to do another brush, and so forth.

**In definition**, Ubiquitous Brushing (UB) is therefore an interactive technique that allows the user to brush over any part of the user interface, including (1) any of the plotted

values, (2) any of the surrounding context-giving components (such as legends, keys, symbols, labels etc), (3) labels or data points that popup or (4) windows and containers. Appropriate meaning needs to be applied to these elements to generate consequential operations, while the extents of the brushed areas also need to be defined. Furthermore, because the components change through an exploration it useful to allow compound UB operations. Thus, the system needs to have memory to control the subsequent brushing operations.

## 4 Demonstration

We have implemented a demonstration system that visualizes mail messages. It is not meant to be a complete mail visualization tool, but is designed to tryout some of these interactive techniques. MailView, as shown in Figure 3 currently demonstrates ubiquitous brushing of labels and other contextual elements. The system consists of three main visualization windows: the Graph View is a scatterplot that displays the emails as date vs arrival-time, the Information Panel shows the main message (i.e. presents details-on-demand), the Data Table either lists the complete data or shows only those elements that are selected. Also, the Tool Box window provides buttons and menus to change the operation of the system.

In this implementation the user can ubiquitously brush over any element in the Data Table as well as most labels and information in the details-on-demand windows (e.g. within the Information Panel). Additional scatterplots (Graph windows) can be created as well to show selected elements in *isolation*, but these are not shown here. Each of the windows are linked together such that any brushing operation in one window is automatically highlighted in subsequent windows. This system was developed from previous incarnations, and we encourage the reader to consult these for more details of the workings of the system [8, 24].

Let's start by selecting some elements. Figure 3 shows MailView with some elements selected, in this case we have initially brushed over one point which gets highlighted in blue, and circular brush of a larger size is used to select a further six elements which are highlighted in red. Each subsequent selected elements are highlighted in different colors, and the details are shown in the Data Table view, see Figure 3 top. So far, this demonstrates traditional brushing operations. However, ubiquitous brushing permits us to brush over any component of the system.

The user can brush over any item of the data table. Figure 4 shows the situation after the user has brushed over the From field in the fifth row of the Data Table. In this case the brush falls over "To CS-staff" and the ubiquitous brush dictates that any element that is also sent "To CS-staff" is additionally highlighted. These additional elements are also added to the Data Table, as shown in black at point B on the figure. Any other relevant information is also updated, such as the data in the Information Panel. Because the brushing is dynamic, the user can subsequently move and brush over any other items. This is shown in Figure 5, where the user brushes over the email address "S.J.Thompson" in the From field. Subsequently, any other email "From S.J. Thompson" is highlighted in all the other views.

The user could also brush over the menus. Thus if they brush over the "TO" menu item in the Data Table then all emails that were sent "To" any of the addresses currently listed in the Data Table list will get highlighted in any view.

The user can also brush over the details-on-demand information in the Information Panel. Figure 6 shows that the user has taken the state of Figure 4, where the user had brushed over the To field. This operation had resulted in the new email From D.J.Sowrey being displayed in the Information Panel. In this operation the user subsequently brushes over that email address in the Information Panel to discover other emails from that person. Alternatively, the user could brush over the item labelled 15:00 to display any other email sent at 3pm.

Such dynamic brushing operations allow the user to explore from one situation to another. This technique is implemented using the Click and Brush principle [24]; where the user can brush over some elements, click on them to fix them which consequently stores them into the Selection Data Table. Users can then make subsequent brushing operations ubiquitously on any part of the display.

## 5 Discussion and Conclusions

UB is a direct manipulation brushing technique that extends current brushing ideas by allowing the user to brush over any element in the display, as well as the plotted data points. By using this technique the user may be able to select elements more efficiently, and a developer could build resourceful systems. This is most useful when used as a compound brush to allow the user to directly explore through multiple scenarios. In fact, the most relevant related-work is the structure-based brushes by Fua, Ward and Rundensteiner [9]. Their techniques do allow multiple drill-down/roll-up operations but their work focussed on brushing hierarchies and did not allow the user to brush over a wider area. Even though UB is a creative technique there are obviously some challenges to overcome:

First, because UB is a direct manipulation approach the interface is transparent to the user, and it may be difficult for a user to understand the exact result of the particular ubiquitous brush operation. This is a general problem with direct brushing especially compound brushing techniques. One solution is to use a history list, where every operation is stored and displayed in a list and the user can role back

to a previous instance. Alternatively, different techniques could be use to clarify the operation; such as annotating details as transparent layers to generate a meta-visualization of the operation [22] or by explicitly drawing a graph representation of the brushed results, such as used by Chen [6] in their hygraph representation.

Second, the refresh rate needs to be kept high. This is an important consideration, especially with large datasets. It is ideal to aim for an update rate of 10 frames per second [18]. Thus, appropriate data structures need to be used to highlight the correct information. We utilize associated arrays to locate the right information.

Third, the information needs to be instantaneously updated in multiple views; various associated lists need to be kept to allow different information to be retrieved in a timely manner. We use the linking model by Boukhelifa et al [4] to achieve this coordination.

Fourth, highlighting is another important issue that needs to be considered; we use color to represent different states of the information and multiple views to display the information in isolation. However, there are alternatives, such as focus and context, masking and distortion.

Finally, in this paper we have discussed Ubiquitous brushing that fits in well with other brushing operations. It is imagined that this technique can be further developed and integrated with other techniques, such as the hygraphs of Chen [6], selection calculus [23] and high-dimensional brushes [10].

## Acknowledgments

We acknowledge Simone Frau for an earlier version of the MailView tool and Andrew Runnalls for beneficial discussions of this idea.

## References

[1] G. Andrienko and N. Andrienko. Making a GIS intelligent: CommonGIS project view. In *AGILE'99*, pages 19–24. Crete University Press, April 1999.

[2] R. A. Becker and W. S. Cleveland. Brushing scatterplots. *Technometrics*, 29(2):127–142, 1987.

[3] R. A. Becker, W. S. Cleveland, and G. Weil. *The use of Brushing and Rotation for Data Analysis*, pages 247–275. Wadsworth Brooks/Cole, 1988.

[4] N. Boukhelifa, J. C. Roberts, and P. Rodgers. A coordination model for exploratory multi-view visualization. In *Proc. CMV'03*, pages 76–85. IEEE, July 2003.

[5] A. Buja, J. A. McDonald, J. Michalak, and W. Stuetzle. Interactive data visualization using focusing and linking. In *Proc. Visualization '91*, pages 156–163. IEEE Computer Society Press, 1991.

[6] H. Chen. Compound brushing explained. *Information Visualization*, 3(2):96–108, 2004.

[7] M. A. Fisherkeller, J. H. Friedman, and J. W. Tukey. *PRIM-9: An Interactive Multidimensional Data Display and Analysis System*, pages 91–109. Wadsworth Brooks/Cole, 1988.

[8] S. Frau, J. C. Roberts, and N. Boukhelifa. Dynamic coordinated email visualization. In V. Skala, editor, *WSCG05*, pages 187–193, Plzen, Czech Republic, January 2005.

[9] Y.-H. Fua, M. O. Ward, and E. A. Rundensteiner. Navigating hierarchies with structure-based brushes. In *Proc. Information Visualization*, pages 58–64, San Francisco, California, USA, October 1999. IEEE Computer Society Press.

[10] A. R. Martin and M. O. Ward. High dimensional brushing for interactive exploration of multivariate data. In *Proc. Visualization '95*, pages 271–278. IEEE, 1995.

[11] C. North and B. Shneiderman. Snap-together visualization: A user interface for coodinating visualizations via relational schemata. In *Advanced Visual Interfaces*, pages 128–135, 2000.

[12] J. C. Roberts. *Exploratory visualization with multiple linked views*. In Exploring Geovisualization A.MacEachren and M-J.Kraak and J.Dykes, Eds, Chapter 8, Amsterdam: Elseviers, pages 159-180, 2005.

[13] J. C. Roberts. Waltz: an exploratory visualization tool for volume data using multiform abstract displays. In *Visual Data Exploration and Analysis V, Proc. SPIE Vol. 3298*, pages 112–122, January 1998.

[14] H. Siirtola. Combining parallel coordinates with the reorderable matrix. In J.C.Roberts, editor, *Proc CMV'03*, pages 63–74, July 2003.

[15] W. Stuetzle. *Plot Windows*, volume 82, pages 466–475. Wadsworth Brooks/Cole, 1987. Also in Dynamic Graphics for Statistics, W.S.Cleveland, M.E.McGill ed. Wadsworth Brooks/Cole, 1988, 225-245.

[16] D. F. Swayne, A. Buja, and D. T. Lang. Exploratory visual analysis of graphs in ggobi. *Proc. 3rd Int. Workshop on Distributed Statistical Computing*, March 2003. Vienna.

[17] M. Takatuska and M. Gahegan. GeoVISTA Studio: A codeless visual programming environment for geoscientific data analysis and visualization. *Computers and Geosciences*, 28:1131–1144, 2002.

[18] J. J. Thomas and K. A. Cook. *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. IEEE Press., 2005.

[19] J. J. van Wijk and R. van Liere. Hyperslice: visualization of scalar functions of many variables. In *Proc. Visualization '93*, pages 119–125, 1993.

[20] M. O. Ward. Xmdvtool: integrating multiple methods for visualizing multivariate data. In *Proc. Visualization '94*, pages 326–333. IEEE Computer Society Press, 1994.

[21] M. O. Ward. Creating and manipulating n-dimensional brushes. In *Proc Joint Statistical Meeting*, pages 6–14, 1997.

[22] C. Weaver. Visualizing coordination in situ. In *Proc InfoVis'05*, Minneapolis, MN, October 2005. IEEE.

[23] G. J. Wills. 524,288 ways to say 'this is interesting'. In *Proc. IEEE Symposium on Information Visualization*, pages 54–61. IEEE Computer Society, 1996.

[24] M. A. Wright and J. C. Roberts. Click and brush: A novel way of finding correlations and relationships in visualizations. In L. Lever and M. McDerby, editors, *Proc. Theory and Practice of Computer Graphics*, pages 179–186. EG, June 2005.

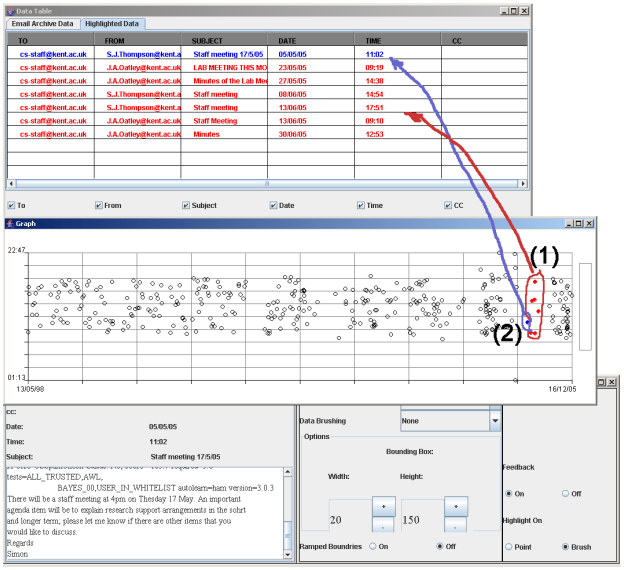


**Figure 3. The MailView application consists of three main visualization windows and one control window. The Data Table view (top) either shows the detail of every element or solely the selected elements (in this example it is showing the selected elements). The Graph view (center) depicts emails as date vs arrival-time, the Information Panel (lower left) shows the email details. The user has selected one element, annotated as (1), and a further six annotated as (2), these are shown in *isolation* in the Data Table view.**

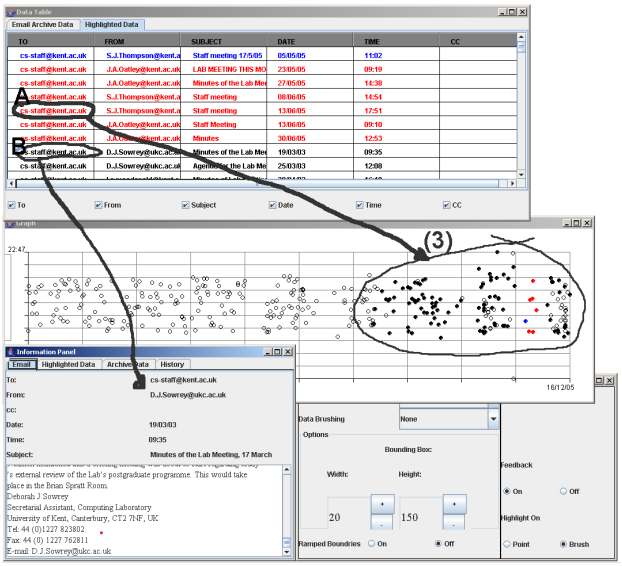


**Figure 4. Ubiquitous brushing means that any item in the Data Table view can be brushed. So, continuing on from Figure 3, this screenshot shows the situation after the user has brushed over the TO field in the fifth row of the Data Table (labelled as A). By doing so, any item that is also sent TO this person is displayed in the Graph View (3). Plus these new elements are appended to the Data Table. Moreover, the Information Panel currently shows the first element of the new information (as annotated by B).**

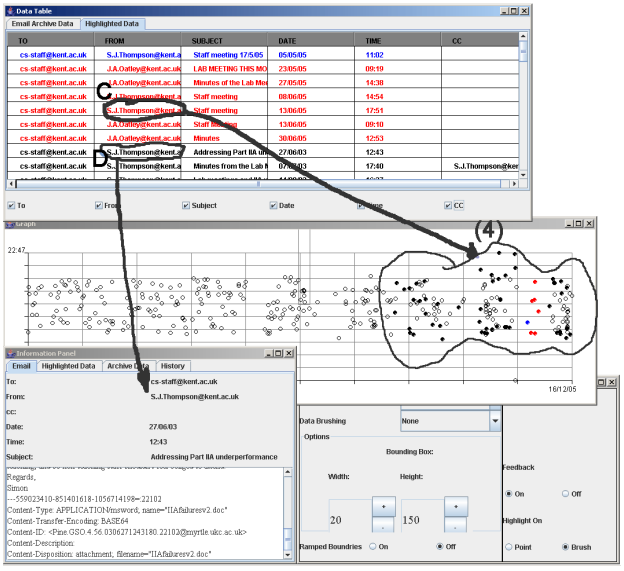


**Figure 5. The user can move the brush, from the position in Figure 4 to explore other elements. I.e. the user is brushing over the From field, and it selects a different set of elements as shown by label (4).**

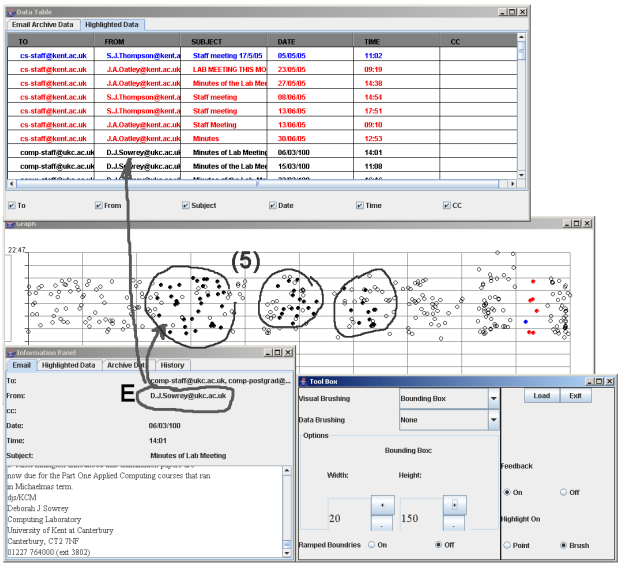


**Figure 6. Additionally, the user can brush over other labels and elements, to explore other scenarios. This shows the user brushing over another From field (this time on the Information Panel) continuing from the state in Figure 4.**