by chance
enhancing interaction with large data sets through statistical sampling

Alan Dix
Computing Department
Lancaster University
Lancaster, LA1 4YR, UK
+44 7887 743 446
alan@hcibook.com

Geoff Ellis
School of Computing and Mathematics
University of Huddersfield
Queensgate, Huddersfield, HD1 3DH, UK
+44 1484 472912
g.p.ellis@hud.ac.uk

http://www.hcibook/com/alan/papers/avi2002/

Abstract
The use of random algorithms in many areas of computer science has enabled the solution of otherwise intractable problems. In this paper we propose that random sampling can make the visualisation of large datasets both more computationally efficient and more perceptually effective. We review the explicit uses of randomness and the related deterministic techniques in the visualisation literature. We then discuss how sampling can augment existing systems. Furthermore, we demonstrate a novel 2D zooming interface – the Astral Telescope Visualiser, a visualisation suggested and enabled by sampling. We conclude by considering some general usability and technical issues raised by sampling-based visualisation.

Categories and Subject Descriptors
H.5.2 User Interfaces – graphical user interfaces, G.3 Probability and Statistics – probabilistic algorithms, H.2.8 Database Applications – data mining

General Terms
Human factors

Keywords
Random sampling, Visualisation, Very large data sets, Astral Telescope Visualiser, Sampling from databases.

1. Introduction

In which we consider some of the problems of visualising large data sets and also some of the uses of randomness in other areas of computing.

Einstein said "God doesn't play dice". Whether or not Einstein's theological insight equalled his cosmological genius, it is clear that randomness is not only unavoidable, but often desirable in many situations. Randomness makes the insoluble soluble, the intractable tractable and the impossible possible. For the omniscient and omnipotent, randomness is an optional extra, for the finite and fallible, it is an essential part of our theoretical and practical armoury. Many areas of computing explicitly involve random elements: sometimes simply to improve efficiency, sometimes to solve problems that would otherwise be impossible.

In this paper we will look at ways of deliberately using random elements in visualisation. Random sampling in particular can improve the efficiency of visualisation algorithms, especially when dealing with very large data sets that might otherwise be impossibly expensive or time consuming to process.

Visualisation is essentially about interaction with people to give them an insight and understanding of information. We will find that random sampling is often acceptable because Gestalt visual processing often depends on approximate rather than exact properties of the data. In addition, random sampling may improve interfaces as faster processing allows better interaction. Furthermore, the data reduction effect of sampling makes visualisation and interaction possible in cases where the user would otherwise be swamped in data.

In the remainder of this section, we will consider some of the problems of visualising large data sets and also some of the uses of randomness in other areas of computing. These will be used to elaborate our proposals for using random sampling in visualisation.

1.1 problems of large data sets
Dealing with large data sets causes two main problems:

- visual limits – the shear number of items makes it hard to comprehend the data set due to perceptual or cognitive limitations of the user or hardware limitations of the display device.
- computational limits – the data volume is too great in terms of the necessary processing power, data storage or network traffic, especially when requiring interactive control of the visualisation.

The word 'large' here is itself somewhat ambiguous and does depend on the visualisation technique and type of data – some visualisation systems regard 10,000 or more as 'large', but in data mining applications many millions of records are common. The Google search engine has over a billion Web pages indexed. The critical size is when the data set becomes too big to view in totality for either visual or computational reasons.

1.1.1 visual limits

The visual limits of visualisations are apparent in many systems. For example, figure 1 is from Netmaps's Web site [28], illustrating the use of Netmap™ Link Analysis to visualise the relationships in a large database of movies. On the circumference of the circle, 900,000 actors and 250,000 movies are drawn as points. Orange lines are drawn between each movie and the actors who appeared in it. Of course, no lines are visible – the entire centre of the circle is solid orange!

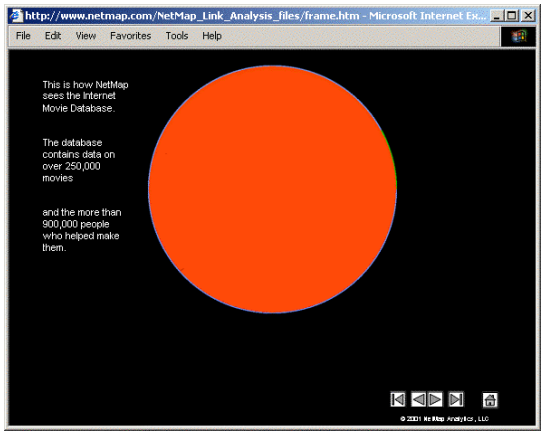


Figure 1. NetMap movie database

In general, when the number of points or lines being plotted becomes large compared to the number of pixels on the display the screen becomes a solid mass of colour and hence no visualisation is possible. This is rather like overexposing a photograph.

Point and line visualisations rely on the human Gestalt visual system to extract trends, rules and interesting issues. We perceive wholes or patterns, rather than pieces or parts, but achieving the right density is critical – too few points and we see spurious connections or cease to see patterns just points, too many and the data becomes an amorphous blob.

If we try to add more visual attributes to each point (such as colour, shape, size) the problems become worse. Closely spaced points of different colours will be merged by the eye to become a mixed colour as in a pointillist painting, while icons or glyphs will overlap and obscure each other.

Interactive 3D visualisations can make overlapping objects acceptable because we expect closer items to obscure more distant ones. However, density control is essential for seeing 'into' a 3D space and not just the outer surface [13]. Both point or iconic representations will reach critical limits when the number of items displayed makes it impossible to see more than the closest items – the foggy day syndrome.

Some kinds of data cannot even be visualised as coded attributes but require full data or summarised forms of it. For example, in an image database one at least needs to see a thumbnail but for a database of unclassified images, seeing more than a hundred thumbnails on-screen becomes difficult. Also, methods such as rapid serial visual presentation implies no more than a hundred or so items per minute [6].

Similarly, deciding whether a document is interesting or relevant, typically requires at minimum the title/keywords and often an abstract or summary. In these cases attempts to view even a few hundred let alone thousands or millions of documents is unrealistic. As a result, search engines typically return 10 items per search and the Scatter/Gather Browser [32] shows 30 document titles on screen, 3 titles for each of the 10 clusters (and also keywords for each cluster).

1.1.2 computational limits

Some visualisation algorithms require substantial processing power. As an example, consider two data sets: set A with 100 items and set B with 100,000 items (still not large by data mining or Web standards). Plotting the points will take time proportional to the number, o(N), so set B will take 1000 times longer to draw. Simple processing like filtering or calculating display attributes will also take time proportional to N, so again 1000 times longer for set B. However, if we want to do even moderately interesting things such as sort the data, this will take o(NlogN) time. Set B will therefore take 10,000 times longer to sort than set A. More complex manipulations are likely to take times that rises quadratically, $O(N^2)$, or even exponentially. So set B would take at least 1,000,000 times longer to process than set A, thus implying that algorithms that have rapid interactive feedback for set A may take days to refresh for set B!

The data set size may also be too great to store in full on a local machine. Consequently, data reduction and processing will be performed remotely. For example, HiBrowse displays summary statistics based on SQL queries, with the real calculations being performed on the database server [15]. Off-loading computation remotely, introduces network delays thus slowing down interactive feedback.

Sometimes it is possible to precompute meta-data and use this for visualisation, only retrieving detailed data on demand. For example, in QPit, documents are mapped onto 3D coordinates based on similarity to referenced documents [3]. Only the 3D coordinates of the documents and a few additional attributes are required for the interactive visualisation although the full text is required to calculate the similarity matrix.

Even reduced meta-information may be too voluminous for very large datasets. Chalmers [8] points out that meta-data, such as index information for Web documents would be too great for normal storage systems, thus implying that meta-meta-data is required. This meta-meta-data would be aggregate data as a few bytes per Web page would fill most disks! Note that using a Web search engine effectively off-loads this storage problem in the same way that an SQL server does.

1.2 randomness in computing

Traditional algorithms are deterministic, attempting to find the unique or the best solution. In contrast, 'modern' algorithmics (modern here really goes back at least 30 years!), including neural networks, genetic algorithms and simulated annealing, makes heavy use of randomness. These algorithms are non-deterministic and find 'a solution' rather than 'the solution', and 'good' rather than 'best'. Because of this more relaxed and inexact approach to solutions, these algorithms can tackle problems that are otherwise intractable, including NP-hard ones. Quality is traded for computation. In some cases, this is a simple cost-benefit trade-off, in others this is because the computation for the exact solution would be impossible.

Examples of these algorithms include:

- optimisation/search – simulated annealing, genetic algorithms, neural nets all involve some degree of randomness which enables solutions to NP problems. This works because there are many reasonably good solutions

albeit in a complex space. Finding pretty good solutions is therefore acceptable.

- cryptography – keys need to be random and often very large prime numbers. Usually, numbers are randomly generated and tested to see if they are prime. However, primality tests would take far too long, so approximate tests are used which rely on random probes [38]. Digital watermarking also uses spreadspectrum techniques (see below) [14].
- signal processing – added white noise can, sometimes, increase the sensitivity of analogue to digital circuitry.
- wireless communication – spreadspectrum techniques use random shifts between frequencies, for example,. Code Division Multiple Access (CDMA) modulates a signal by multiplying it against a pseudo-random signal [7].
- telephone routing – some years ago the US phone network collapsed due to systematic failure of deterministic routers. When a large central exchange broke down, calls were diverted to the next lowest cost route. Consequently, a large numbers of calls hit that exchange, which also collapsed; this continued until the whole system broke down. Calls are now randomly routed through 'cheap enough' paths.
- parallel computation – placement of computation on processors may be randomised to avoid computations which would hit worst case behaviour of parallel algorithms [35]. Routing is also randomised between multiple processors and memory elements to avoid contention problems, like the telephone network.

In some of these cases randomness is used simply to reduce the computational effort – if one could do the calculation in full it would be better but the random version is just 'good enough'. However in many cases, the randomness is essential otherwise the system would be worse or incorrect.

1.3 randomness for visualisation

We have seen that it is often difficult to deal with large data sets for perceptual and computational reasons and also seen that randomness is useful in other areas of computing. Moreover, the data set is often a sample from the real world as the underlying data is intrinsically continuous (or otherwise vast). The example data used by Influence Explorer is an illustration of the latter - the model used has continuous as well as discrete input parameters [42, 43]. Even if all the data is available electronically, it may have to be sub-sampled or summarised for privacy reasons, as is the case with census data [18, 30].

Our central proposition is that deliberately introducing (further) random sampling can improve visualisation algorithms:

- when calculations mean that information is lost anyway (e.g. average, representative histogram)
- when there are too many data points to show
- when details are only required for some data items

In the rest of the paper, we will expand on this theme. Section 2, will examine existing uses of randomness in visualisation and also alternative visualisation techniques that address some of the same problems that we wish to solve using sampling. Section 3 will consider several systems where random sampling can be used to improve current visualisation. This will include augmenting standard techniques (e.g. density plots, histograms, pie charts) and also techniques used previously by the authors. We will also demonstrate how sampling can lead to a new visualisation technique, the 'Astral Telescope Visualiser'.

Two classes of problems arise from the explicit use of random sampling: firstly, user interface issues, such as the number of samples and potential sources of confusion and secondly, computational issues like sampling methods with conventional or bespoke databases. Sections 4 and 5 deal with these issues respectively.

2. Existing randomness and alternatives

In which we review existing visualisation techniques that use random effects and also techniques that achieve similar aims.

It has been difficult to find examples that use random sampling or random effects in general within the visualisation literature. If one wants to visualise some aspect of real world data, it is normal to capture only a sample. But why is it so rare to find subsampling used for large electronic data sets? Perhaps because the data is 'there' it seems that it ought to be used and therefore effectively discarding information by sampling feels, in some way, wrong. It is easier to find more deterministic methods that address some of the concerns we wish to tackle with sampling.

2.1 using randomness now

The clearest case of sampling is the model data used in Influence Explorer [44]. The visualisation space is the relationship between input and output parameters in a mathematical engineering model. This is effectively an infinite data set and so only a finite subset of input parameters can be put through the model. Tweedie [44] describes the selection of parameter values: the designer first selects a range of values where the final design is expected to lie, and then "Within this region a large number of points (e.g. over 500) are generated randomly, each point representing a design." The issues arising from this random selection are not really discussed in either the papers on Influence Explorer and associated Viki toolkit or Bob Spence's recent visualisation book [41].

There are various algorithms where random initial values or presentation orders are used. These include numerous ball and spring visualisations for the Web and other information domains [19, 5] and also neural network techniques such as Kohonen Maps [22, 25, 26].

2.2 related deterministic techniques

2.2.1 summary statistics and aggregates

Perhaps the oldest way to deal with large amounts of data is the use of summary statistics, either numerically as tables (e.g. mean, standard deviation) or graphically in the form of histograms and similar graphs. The large number of individual data values is reduced to a few numbers or bars on a chart. Similarly density and contour plots reduce data to the number (or some other measure) within a certain area.

2.2.2 single pixel per record

Some techniques use a single pixel to represent a data point, hence reducing the screen real estate needed to display a large dataset. Point plot visualisations such as starfields work on this principle and can display up to tens of thousands of data points. VisDB [20] takes this to an extreme in not just representing each record as a pixel, but also in packing the pixels together using various space filling curves. GridFit [21] a related technique, handles overlap of plotted data points for

2D spatial data by moving points to a close unoccupied pixel position. However, in both these systems the maximum number of data items that can be displayed is limited by the number of pixels on screen (of the order of a million) and in practice several smaller displays are shown, limiting the maximum number to only hundreds of thousands. TableLens [36] uses a similar technique but it represents most records as a single pixel in the vertical direction. Again this technique has limits as the users of TableLens have to scroll to view more than a few hundred records.

2.2.3 overview and zooming

Because users often need to see the detail of individual data items, many techniques allow both overview and details to be seen. In focus+context techniques such as Fish-eye views [16] and the Hyberbolic Browser [24], this is determined by location. In contrast, techniques following Shneiderman's visualisation mantra "overview, zoom and filter, detail on demand" [39] are based on time. So when zoomed out or unfiltered, one just sees points, when zoomed in or heavily filtered, details of individuals become visible. Constant density displays [46, 47] makes this even more dynamic, where the amount of detail of an individual item is dependent on the local density.

2.2.4 cutting corners

The Hyperbolic Browser also uses two more techniques to deal with large data sets. First, it does not try to represent points very close to the edge of the viewing circle as they get too small to see. This is a form of filtering, but driven by visual perceptual limits. Second, the links between nodes in the hyperbolic model should really be circle arcs and are drawn like this when the scene is still, but, during user interaction the links become straight lines which are faster to draw. As the number of links is proportional to the number of points, this is a significant time saving.

2.2.5 filtering

Filtering is also at the heart of dynamic queries such as in Homefinder [45] and Starfield Displays [1], however these filters are user-defined and related to the task. As a result, the user has a trade-off if the number of points is too large to see the structure. This trade-off is between zooming in (and loosing sense of the whole) or filtering on particular attributes that are not task related. Some systems attempt to deal with this by filtering, based on some form of relevance/importance. For example in Salton [37], word co-occurrence links between documents are filtered based on the strength of the link so that only the stronger links are shown. This reveals a structure that would be hidden by drawing every single link.

2.2.6 clustering and representative values

Clustering techniques reduce the density of large datasets by grouping the data into a small number of groups of similar items. These clusters can then be treated as individual items to be visualised depending on their average or typical group attributes [23] and perhaps filtered based on the cluster size. Alternatively, representative members may be selected, as is the case with the Scatter/Gather Browser [10, 32].

Note that all these techniques work with the full set of data (or at least meta-data) behind the scenes even when only a subset or summary is shown to the user.

3. Using randomness

In which we suggest ways of using randomness to enhance or enable different forms of visualisation and interaction.

3.1 aggregates

Virtually any visualisation based on aggregate or summary statistics can use sampled data to give approximations. For example, a histogram can be created using a sample of the full data set. To produce a histogram that shows the heights of males in Italy, one would choose a sufficiently large sample of males and measure them, but not every single male. Likewise, histograms, pie charts, density and contour plots can all use sampled data. In one of the key papers on database sampling [9], the use of database sampling to generate good enough histograms is suggested, however there is no record of this actually being followed through.

Similarly, interfaces with numerical data can use sampling. For example, the HiBrowse interface constantly displays for each value of each attribute (including hierarchical attributes such as taxonomies) the number of records that have that value, and hence how many would be chosen if that attribute were selected to filter [15]. Where these numbers are large, for instance in the initial stages of a search, approximate values based on samples would be sufficient as their purpose is to guide the users search – a form of information scent [33].

3.2 point and line data

As discussed previously, a key problem with point or line data for very large datasets is when the points or lines saturate the display. In many cases, simply sampling the data can make these readable. For example, a supermarket has till data and uses a ‘wagonwheel’ visualisation. The goods are drawn as points on the circumference of a circle grouped by type (beers, frozen pizza, chips, tomato sauce etc.) and for each pair of items on a till receipt, a line is drawn between them. The aim is that the types frequently bought together will show up as having lots of lines between them. If these lines were drawn for every pair of items, the circle would become a solid mass of lines, as shown in figure 1. But if till receipts were selected at random, the drawing can stop when there are just enough lines to show features and relationships.

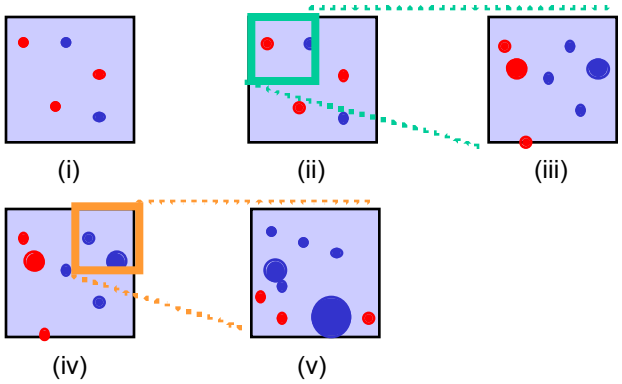
As mentioned earlier, the TableLens will start to breakdown when faced with millions of records. However, if the data were sampled, the majority of correlations would be visible. On the other hand, a single outlier amongst the million records would probably not be visible. Sampling helps us see bulk properties, not single special cases.

Similarly, 2D or 3D point data can be sampled to keep its density low enough to be able to see structures by simply using a standard visualisation on a sample of a large dataset. However, it is also possible to sample and resample dynamically during interaction.

3.2.1 the Astral Telescope Visualiser

An example of this dynamic resampling is a new visualisation technique based on the metaphor of a star-gazers telescope. As you look at the night sky with your naked eye, you see a small cluster of 5 stars. Take a telescope and look at the same area, you not only see the same constellation of 5 stars brighter and bigger, but also more stars appear that were previously too dim to see. If you increase the magnification of the telescope, yet more stars appear as the field of view shrinks.

The Astral Visualiser works in a very similar fashion. Two attributes or derived values are chosen for the x,y coordinates and a small set of a few hundred sampled records are originally chosen and plotted. The user can select an area of interest. As Astral zooms in, it samples more records in that area (that is a sample constrained by the x and y coordinates). The sample, is chosen so that the density of sampling increases with the square of the zoom value, this means that the actual visible density remains constant (see figure 2).



(i) initial user view, (ii) user selects top left and zooms, (iii) more points become visible, (iv) selects top right and zooms, (v) yet more points appear

Figure 2. User views of Astral Visualiser (sizes exaggerated)

One way of thinking about this, is that we determine the x and y coordinates from the attributes, but randomly allocate a z coordinate and then determine how far away we see by the current zoom factor. Figure 3 shows this model. For illustration purposes there are a very small number of points and only three discrete z depths. Also, all the points are shown with their associated z value, in the real system the z values are calculated only when they are sampled from the database.

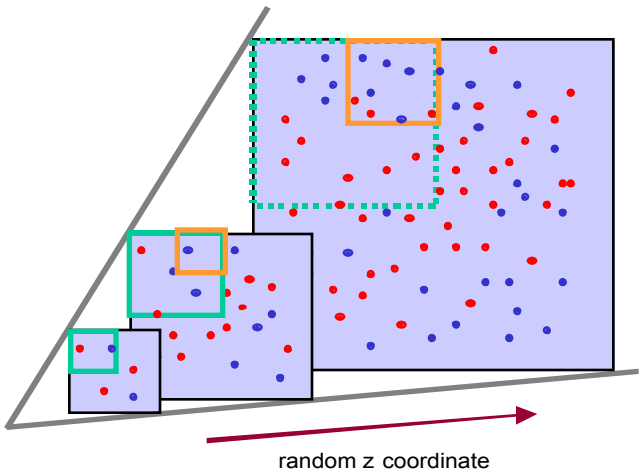


Figure 3. Model behind Astral Visualiser

We can step through the interaction using this model in figure 3 (which also shows the zoom selections from figure 2). Initially, only the first z plane from figure 3 is visible in figure 2.i. The user then selects the top left hand corner. As the system zooms in, extra points are sampled from this region; so the points from the second layer become visible. The points in the first layer become brighter and correspond to the dotted area in the third layer. However, these points are not visible at this zoom factor. Finally as the user selects and zooms into the top right of figure 1.iv, the points in this area are re-sampled to reveal the points in the solid marked region of the third layer.

To summarise, the Astral Visualiser can work with 2D views of arbitrarily large data sets (not just millions) allowing both the overall structure of the data to be seen and also the ability to zoom into individual data items.

3.3 individual data items

As discussed in Section 2, it is often important to see at least some details of each data item. "Data is probably the only thing that people have in common with computers" Benyon [4]. Sometimes this may be an iconic, summary or thumbnail version of the data item. Here too, sampling can be used to reduce the data set to a size that allows visualisation of individual items. For instance, in the Astral Visualiser the 'closer' items (which are a random selection) could be shown in summary form. This is similar to Pad++ and other 'zoomable' interfaces where information items change their form and reveal more detail as the user 'zooms' closer [2]. Starfield-style visualisations [1] have the same behaviour. Small numbers of iconic or summary representations can coexist with more point/line style displays so long as the density of the icons does not obscure too many points.

In other cases, such as the Scatter/Gather Browser, more details are required because it is essential to see real document titles to judge whether a cluster is appropriate. Usually these are chosen as the best representative documents but random selection could also be used to give representatives. This may be problematic if the cluster is very diffuse, for the random sample could be (albeit unlikely) right at the edge of the cluster. However, the use of a small number of representative documents (standard Scatter/Gather uses 3 anyway) makes this unlikely to be a problem. The clustering itself can also be performed on samples of the document set, which can significantly improve such algorithms (this technique has been used for machine learning [40]).

3.3.1 Query-by-Browsing

Query-by-Browsing (QbB) was first proposed as in idea 10 years ago as a platform to demonstrate interaction issues when using AI and machine learning in user interfaces [11]. The first running implementation was produced a few years later [12] and it has recently become available in a Web version [34]. QbB is an intelligent database interface that uses examples of records to generate a query as opposed to generating a list of records in response to a query!

In QbB the user first sees a list of records in the database and goes through the records either selecting them for inclusion with a tick ✓ or exclusion with a cross ✘ (figure 4).

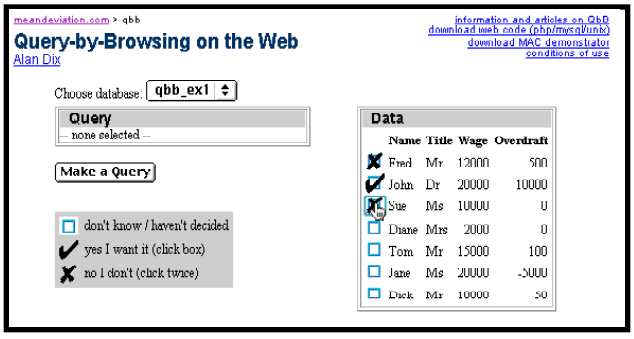


Figure 4. User chooses records

When sufficient records have been classified the system generates an SQL query that matches the marked records and highlights all other records in the query result (figure 5). If the user is satisfied with the query, it can be used in subsequent database actions, perhaps copied for use in a CGI script. If the query is not right, the user can mark more records until the query is as required.

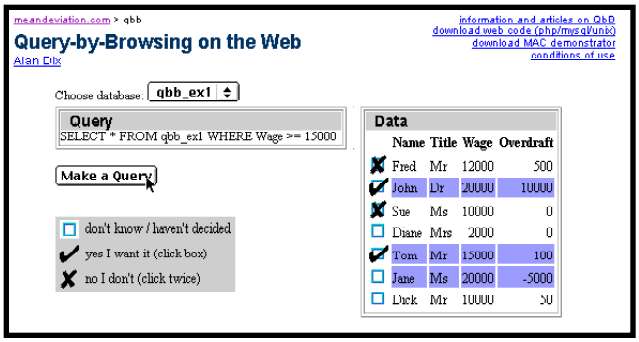


Figure 5. QbB generates query

QbB is based on a number of principles:

- **data centric** – the focus is always on real data with the abstraction of the SQL query being secondary.
- **interpreting not creating** queries – a user may be able to see whether a query is correct, but not create the query themselves.
- **multiple representations** – the existence of both the query and the list of selected records is easier to interpret (especially for boolean connectives and negations) than the query on its own.
- **precision and audit** – for many database queries and updates, a precise and auditable result is required. Similarity based IR techniques cannot be used alone, the SQL query is important.

The first principle is significant here. The focus on individual records is essential but how can this scale to very large datasets?

Given the context of this paper, one solution is clear: the user can be presented with a random sample of records and subsequently, classification and query formation can be based on that sample. Once the query has been confirmed it can be applied to the whole dataset.

This random solution is better than most deterministic alternatives. For instance, it is possible to present the user with the first 100 records based on some sort order, say by department. However this would mean that all the records classified by the user may be in the first department (probably accounts), thus leading to an unrepresentative and potentially incorrect query formulation.

The random sample is the best sample!

4. Randomness and interaction

In which we discus some of the issues sampling raises for interaction and how to choose correct sampling levels, taking into account human perception and understanding.

In general, we want to minimise the sample size in order to minimise computation, but only insofar as this does not have a significant effect on the accuracy of the visualisation. The need to see Gestalt patterns also suggests making samples small enough to reveal structure, but if the sample is too small, the patterns may not be seen either. We will now consider these related issues and others arising from random sampling.

4.1 perceptual limits

One way to determine if a sample is big enough, is when the visualisation obtained from the sample is indistinguishable from that obtained from the full data set. Our example will consider histograms, but the issues are similar for most visualisations.

If the error in a histogram bar height due to the sampling is less than 1 pixel, the sample is clearly big enough! If the number of items within column i is n_i, then the standard deviation of this is approximately $\sqrt{n_i}$. This means that the proportional error in the height of the column is proportional to $1/\sqrt{n_i}$, so a column based on 100 data items has a 10% error and one with 10,000 items has a 1% error. Note that the *proportionate* error reduces with the number of items, but the *absolute* error increases. If the column height is 200 pixels we need approximately 40,000 data items to get the standard deviation within 1 pixel. This is rather large!

Of course, the important point is not that the visualisation is identical at a pixel level, but that it is effectively the same view for the user. Therefore, to see the general shape of a histogram, errors far greater than a single pixel are acceptable. Given the square root in the error formula, accepting a 5 pixel error allows us to have 25 times less data items sampled! Also, we may be more interested in the proportionate error rather than absolute error; an error of 5 pixels in a 10 pixel high histogram is more visually significant than a 5 pixel error in a 200 pixel column.

Because the error depends on the number of items, it has been suggested that the width of columns can be adapted to give the same number in each column [31]. This reflects the common practice in human drawn histograms of pooling data in the smaller regions – making the shorter bars wider. This minimises the worst proportionate error in column height. Strangely enough, to minimise the *absolute* error in column heights (say to within 1 pixel), the higher columns need to be made wider!

Pie charts and density plots have similar behaviour to histograms, but contour plots are more complex. Contours are plotted at points where the quantity changes past some value. The position of contours is very sensitive in areas where the quantity changes slowly. This is also true of physical contours in situations where there is a plateau near one of the contour levels. Often it is the differences that are most important to the user not absolute values or even approximate ones. This is also true of point plots, histogram columns, and so on.

4.2 statistical awareness

We have seen that statistical error is an issue when dealing with sampled data, either because the full data set is a sample from the real world or because we are sub-sampling from the dataset. If the error is small enough, this is not a major problem and the user can treat the visualised data as if it were real. However, as shown by the calculations above, the numbers of points that need to be sampled to reduce errors to pixel levels, may be very large indeed. Political opinion polls, involving a few thousand respondents at most, would lead to errors that exceed a pixel in all but the smallest graphic.

With real statistical data, it is therefore crucial that users understand that the values they see are approximate. Numerical data can be presented with +/– error figures and graphs can

show error bars, but often this can obscure the structure inherent in the visualisation. Sometimes we can use more subtle indications. For example, the Google results page clearly indicates an approximate number of results and reinforces this with trailing zeroes in the total.

Result 1 - 10 of about 77,900.

For graphical data, explicit error bars may add to visual clutter, but one can get the effect of 'trailing zeroes' by visual techniques such as blurred edges. However, the best solution may be to allow the randomness to become more apparent by reducing the amount of 'smoothing'. For example, rather than making the histogram columns wide enough to minimise error you can make them narrow enough that the column heights become ragged and hence suggest randomness. The user's eye can then do the smoothing. The Influence Explorer (figure 6) demonstrates this feature; the number of data points is far too small to give 'accurate' histogram heights, but the overall jaggedness of the histograms clearly indicates the randomness.

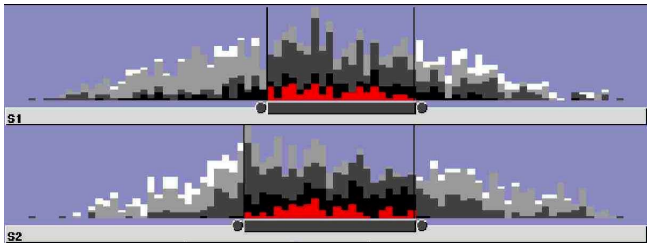


Figure 6. Influence Explorer [41]

Similarly, point plots intrinsically give a better idea of the underlying randomness than density or contour plots. Showing points slightly blurred or with a ‘halo’ may well combine the two effects.

4.3 handling interaction

The random effects of sampling may be confusing when interacting with a visualisation. For example, when zooming into an area, some resampling may be necessary (as with the Astral Visualiser). However, it would be confusing to the user if the existing points disappeared when a different random sample was used. Maintaining continuity in sampling, both within a session and possibly between sessions, is important.

The resampling may also take time, which implies either slowing down interaction or perhaps adding the new data as it is sampled. With the Astral Visualiser the stars could appear gradually over time as we zoom in, or the bars of a histogram could be shown 'settling down'. Such effects that expose the underlying sampling, may maintain user awareness of the statistical nature of the visualisation. Indeed it has been shown that smooth transitions between data sets is beneficial to the user (discussion of cone trees in Spence [41]).

Zooming out also needs careful consideration on how to remove the extra ‘resampled’ data from the previous zooming in operation. The previous data can be reinstated or a new sample can be generated. The former solution will have the overhead of remembering the data at each level whilst the latter solution will not maintain continuity.

Another issue arises from examining the behaviour of the Influence Explorer. If the model is resampled to give more detail, the histograms end up with large peaks where resampling has occurred. Two possibilities exist, either hide the resampled points when dealing with overviews, or weight the points appropriately in resampled areas.

4.4 spurious patterns

Patterns are clearly visible in the night sky (when it’s not raining of course!) and certain ones such as the Great Bear, the Plough, the Big Dipper are named. However, these pictures in the stars are simply random arrangements as the distance from Earth to their component stars vary considerably. The patterns are in our minds. Similarly, it is easy to see spurious correlations, clusters and more in sampled data. For example, look at the points in Figure 7. Can you see groups and lines? In fact the points are completely random with no underlying structure whatsoever.

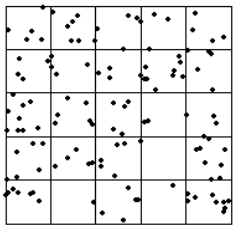


Figure 7. Random dots [17]

In this case, explicit sampling may help. Normally the raw data is itself some sample from the real world and increased awareness of this can make users more alert to false conclusions from the data. For example, in figure 7 there is an apparent line of points in the second square on the top row. If we then zoomed in on this, the line would disappear as we saw more points making it clearly an artefact of the sampling. In fact, a standard statistical technique is to compare some sort of calculation on a sub-sample of data with a similar calculation on another sub-sample. This gives an indication of the robustness of the statistic. Likewise, in machine learning and neural networks it is commonplace to train on a sub-sample and then test the learnt rules on another sub-sample.

This cycle of hypothesis testing could be valuable in visualisation.

visualise on sample 1 $\rightarrow$ propose relationship $\rightarrow$ test on sample 2

Even when we could show the whole data set, this sub-sampling technique can be used to build more robust hypotheses.

4.5 distribution of sampling

When we make a sample we make choices. If the sample is simply a uniform selection (such as 1 in 10 of the data set), then perhaps we do not need to worry, but most sampling regimes introduce some form of sampling distribution and bias. For example, a telephone poll will disproportionately exclude those who do not have a phone. If we sample from a file by randomly choosing a byte offset and then selecting the record that contains the byte, although this at first seems fair, it is in fact biased towards longer records.

A very clear example of this is the sampling used in the Influence Explorer. A parameter space is chosen for the engineering model and points are chosen randomly within it; but what distribution is used? Imagine it was a GIS model and one of the parameters was related to density of housing in rural areas. We could either include this as (i) number of dwellings per square kilometre, or as (ii) average distance between dwellings. A uniform sample against (i) would have a 1/x distribution in (ii) and vice versa. Similarly with sound, a decibel scale is logarithmic compared with an energy scale, frequency vs. wavelength. Clearly the right distribution should not be a matter of how one measures a parameter.

An arbitrary choice of distribution causes various problems in subsequent visualisations. We will discuss these in the context of Influence Explorer because it makes sampling explicit. However, these issues also arise in any visualisation where the data being visualised is not necessarily representative of the real world.

First, the shape of the histograms in the Influence Explorer is affected by the underlying sampling distribution. Given the sampling distribution is not necessarily meaningful, the histograms are effectively meaningless! Of course the *change* in the colouring of the histograms as various parameters are altered is very important, but the initial shapes are not. How many users are aware of that?

Even more problematic, the Influence Explorer can be used to explore yields. This involves selecting bounds for input parameters (in their example these are material, filament diameter etc.) and then seeing what proportion of the sample satisfies the output targets. Obviously, the sampling distribution will be uniform within the input parameter bounds, whereas in reality it is likely to have some more complex shape depending on the manufacturing process. If the filament is extruded, the diameter is likely to be peaked towards the middle of the range. So, a uniform distribution may be misleading during yield optimisation.

As noted above, the reason why these issues are apparent in the Influence Explorer are not due to using random sampling per se, but because sampling is obvious within the algorithm. If the underlying data had been from a telephone poll and we used all of it, we would be facing the same problems of distorted data. Statisticians deal with this either by using stratified samples (right number in each group based on real distribution) or by using weighting during later processing to fix sampling bias.

Again, considering the Influence Explorer, there is no clear 'real' distribution to use during initial exploration. Perhaps, because the users are engineers, we may be able to rely on their professional experience with previous experimental data. With a less experienced user group it may be worth making the histograms all rescale to full range initially, similar to the colour balance in film processing. For yield maximisation, the expected distributions from the production process (and perhaps introducing different processes as additional input variables) can be used to allow weighting of samples so that the displayed histograms match those that would be obtained in real production.

5. Sampling databases

In which we examine ways of extracting random samples from existing databases, look at some research literature on sampling from large data sets and see how this may be used to help design bespoke data storage.

The data we would like to sample is typically stored in a database, either a standard SQL database or some other database – perhaps OO database, flat file or one in a bespoke format. It is important to obtain samples efficiently and correctly. Efficiency is reasonably easy to characterise: we would ideally like samples to be obtained within a time proportional to the size of the sample (say n) or at very worst the size of the database (say N). Correctness is perhaps not quite so clearly defined. We will first discuss some of the issues about appropriate samples and then look at practical techniques and research on sampling from databases.

5.1 sampling issues

5.1.1 statistical properties

The sample we take should have the right statistical properties: the right size, uniformly chosen (not biased towards any particular values) and each sample independent of the rest. However, we may want to relax some of these properties to improve efficiency. For example, if we are prepared to have approximately 100 in our sample rather than exactly 100, we may be able to generate a sample quicker. Similar issues arise in real world sampling. A recent telephone poll used a ‘snowball sampling’ technique, where subjects were asked if they knew others who might be prepared to take part in the poll. Clearly the responses between those who know one another are not independent, but it is a rapid way to build up a given size of sample. We will see similar bucket sampling issues arise when looking at databases.

5.1.2 multiple samples

We also need to consider multiple samples. In some cases, we need the samples to always overlap. In the Astral Visualiser for example, if we have 1000 points and then zoom into an area with only 100 points, we need to get 1000 points in the new region. However, to ensure display continuity, these points should include the original 100. In other situations we may deliberately want independent samples.

5.1.3 stratified samples

To improve the accuracy of visualisation we may wish to have a stratified sample: that is one where we force appropriate numbers of the sample to be in particular subgroups. Assuming we take a random sample of 100 from a data set, where most of the data items belong to a particular group A but 1% are in a second group B. There is a 40% chance that none of the sample will be from group B and a 20% chance that group B is over-represented with 2 or more items. One solution is to randomly choose 1 representative item from group B and 99 from group A. Another solution is to over-bias the sampling, choosing 10 from group B and 90 from group A, but weight the samples in subsequent calculations in order to improve the accuracy.

5.1.4 what is random anyway!

Ideally, we would use a 'really random' method of choice, perhaps counting decay of a radioactive isotope. However, it is more likely that we will be using a pseudo-random number generator, typically based on prime modulus arithmetic. This does raise the question of what is really random and what is random enough. One answer is that something is random if its statistical properties are uncorrelated with the things we are interested in! For instance, the middle 4 digits of a person's telephone number is likely to be a pretty good 'random' number for many purposes.

5.2 tricks for sampling standard SQL databases

Most SQL databases do not give any support for random sampling, so we have to resort to 'tricks' to get a sample.

It is tempting to simply use the SQL `'RAND()'` function, but this can be problematic as some SQL servers treat this as a 'constant' for optimisation purposes or conversely they may recalculate it more than expected. A better choice is to make use of a 'random field', either an existing field, as suggested above or a precomputed random field (the `'RAND()'` function can be used for this).

Since random sampling may involve filters or sorting against this field it may be worth indexing it and/or creating a separate table with just the record key and 'random' field. This table will then have much smaller records than the full data record and thus be faster to process. However, it will need to be joined to the original table for random sampling of query results.

The chosen 'random' field can then either be (a) sorted and the top n chosen or (b) filtered to choose n out of N records (e.g. "`MOD(the_field,1000) < 10`" would give 1 in 100 of the records). Of these (a) is more accurate giving exactly n records whereas (b) only gives approximately n records. But (a) will usually take O(NlogN) time as the database has to be sorted (or O(Nlogn) for clever top n sort), whereas (b) is a simple filter and thus takes O(N) time.

More details of SQL sampling techniques, including SQL examples, can be found on the web pages for this paper.

5.3 implementing sampling in the database - SQL and bespoke data

To our knowledge, although not included in any commercial databases, there has been considerable research on adding random sampling as primitives to relational databases [29, 30, 9, 27]. As well as being available for external use, such sampling can also be used internally for query optimisation.

The advantage of sampling as a primitive is that it can use knowledge of the internal structure of the database to give faster sampling and get closer to the desired O(n) behaviour. For example, if the size of the sample is such that most database blocks do not contain a chosen data item (that is if n x B << N, where B is the block size), then it is possible to obtain faster sampling by first randomly choosing blocks then choosing records randomly from the chosen blocks. This involves accessing n blocks whereas a filter such as case (b) would involve accessing all blocks. Some algorithms deliberately choose all the items from a block if these are known to be unrelated, for instance if the block placement is based on a hash index. This implies that only n/B blocks are accessed.

Although most of this research targets relational databases many of the techniques are about lower level structures such as B-trees. Therefore the basic results can be used to add sampling to other kinds of database or bespoke data storage.

6. Conclusions

In which we sum up that randomness is a jolly good thing and the next AVI should be held in Monte Carlo :-)

Virtually all electronic data is some sort of sample and so it is surprising that there are not more examples of sub-sampling of that data for visualisation. Possibly there is more in practice than reported. However, this may also reflect a certain reluctance to 'lose information' that is in the electronic domain and also a bias towards deterministic algorithms within computer science education.

However, in many areas of computing, random algorithms have proved to be exceedingly powerful. We believe we have demonstrated that the same benefits can be obtained within visualisation.

We have discussed several specific examples of ways in which random sampling can be used with existing algorithms. In some cases this simply reduces the time needed to calculate a visualisation, both saving resource and making interaction possible where it would otherwise be too slow. In the case of Query-by-Browsing and other rich record-focused interactions, sampling makes the system scalable. Moreover, in the Astral Visualiser we have seen how even something as simple as a 2D plot can become a powerful visualisation technique with the aid of sampling and interaction.

Sampling introduces errors and approximations over and above those already present in all real world data. But the statistical techniques to estimate and control these errors are well understood. However we need, not just recruit the established mathematics but also relate it to a rich understanding of display and perceptual limits.

Although there is an active area studying random sampling from databases, we do not know of this being included to date in any commercial products. But we have shown that there are practical techniques to achieve sampling without built-in support.

For smaller datasets and simpler algorithms, none of this is necessary. However, for many applications the difference between a good, possibly impractical, idea and an effective interactive visualisation may be in the roll of the dice.

7. References

For additional information and links to online versions of many of the refernces below , see the paper web page at:
http://www.hcibook/com/alan/papers/avi2002/

[1] Ahlberg, C., Shneiderman, B. Visual Information Seeking: Tight Coupling of Dynamic Query Filters with Starfield Displays. Proceedings ACM Conference on Human Factors in Software, CHI '94, Boston, April 1994, ACM Press, 313-317

[2] Bederson, B.B and Hollan, J.D. Pad++: A Zooming Graphical Interface for Exploring Alternate Interface Physics. Proceedings ACM UIST '94, Marina del Rey, CA, November 1994, 17-26

[3] Benford, S. and Mariani, J. Virtual environments for data sharing and visualisation – populated information terrains. Proc. IDS '94 2nd Int'l Workshop on User Interfaces to Databases, Lancaster, UK, April 1994. Springer Verlag: Workshops in Computer Science, 168–182

[4] Benyon, D. Task analysis and system design: the discipline of data. Interacting with Computers. 4(1):246–249, 1992

[5] Brodbeck, D., Chalmers, M., Lunzer, A. and Cotture, P. Domesticating Bead: Adapting an Information Visualization System to a Financial Institution. Proc. IEEE Information Visualization 97, Oct. 1997, 73-80

[6] de Bruijn, O. and Spence, R. Rapid serial presentation: a space–time trade-off in information presentation. Proceedings AVI '2000, ACM Press, 2000, 51-60

[7] CDMA Develpment Group. What is CDMA (Code Division Multiple Access)? (accessed 17th Nov 2001, dated © 2000). http://www.cdg.org/tech/about_cdma.asp

[8] Chalmers, M. Informatics, Architecture and Language. Social Navigation in Information Space. A. Munro, K. Hook & D. Benyon (eds.), Springer, 1999.

[9] Chaudhuri, S., Motwani, R. and Narasayya, V. Random Sampling for Histogram Construction: How much is enough? Proceedings SIGMOD '98, Seattle, ACM Press,1998

[10] Cutting, D.R., Karger, D.R., Pedersen, J.O. and Tukey, J.W. A cluster-based approach to browsing large document collections. Proceedings SIGIR '92, Copenhagen, 1992, ACM Press, 318-329

[11] Dix, A. Human issues in the use of pattern recognition techniques. Neural Networks and Pattern Recognition in Human Computer Interaction, 1992. Eds. R. Beale and J. Finlay, Ellis Horwood, 429-451.

[12] Dix, A. and Patrick, A. Query By Browsing. Proc. IDS '94: The 2nd Int'l Workshop on User Interfaces to Databases, Lancaster, UK, April 1994. Springer Verlag: Workshops in Computer Science. 236-248.

[13] Dix, A. Time, space and interaction. Proceedings FADIVA 3, Ed. I. Catarci. Gubbio, University of Rome, Italy, 1996, 99-103.

[14] Dugelay, J-L. Digital watermarking (tutorial). SAICSIT 2001, South African Institute of Computer Scientists and Information Technologists Annual Conference, University of South Africa, Pretoria, Sept. 2001, 25-28

[15] Ellis, G.P., Finlay, J.E. and Pollitt, A.S. HIBROWSE for Hotels: bridging the gap between user and system views of a database. Proc. IDS '94 2nd Int'l Workshop on User Interfaces to Databases, Lancaster, UK, April 1994. Springer Verlag: Workshops in Computer Science, 45-58

[16] Furnas, G. W. Generalized Fisheye Views. Proceedings ACM CHI '86, Boston, April 1986. ACM Press. 16-23

[17] Dix, A. Statistics tutorial: Gheisra – a story. 1998. http://www.meandeviation.com/tutorials/stats/Gheisra/

[18] Guthrie, D. Statistical models and analysis on auditing. Panel on nonstandard mixture of distributions, Statistical Science 4, 2-33

[19] Hendley, R.J., Drew, N.S., Wood, A.M. and Beale, R. Narcissus: visualizing information. Proceedings IEEE Information Vizualization '95, IEEE 1995. 90-96, 146

[20] Keim, D.A. and Kreigal, H-P. VisDB: database exploration using multidimensional visualization. IEEE Computer Graphics and Applications, Sept. 1994, 40–49

[21] Keim, D. A. and Herrmann, A. The Gridfit Algorithm: An Efficient and Effective Approach to Visualizing Large Amounts of Spatial Data. Proceedings Visualization '98, Research Triangle Park, NC, 1998, 181-188, 531

[22] Kohonen, T. The self-organizing map. Proceedings of the IEEE, 78(9):1464–1480, 1990

[23] Kreuseler, M. and Schumann, H. Information visualization using a new Focus+Context Technique in combination with dynamic clustering of information space. Proceedings NPIV '99 (New Paradigms in Information Visualization and Manipulation), Missouri, November 1999, 1-5

[24] Lamping, J. and Rao, R. Visualizing Large Trees Using the Hyperbolic Browser. Proceedings CHI '96, Vancouver, April 1996, ACM Press, 388-389

[25] Lin, X. Visualization for the document space. Proceedings IEEE Visualisation '92. IEEE, 1992, 274–281

[26] Lin, X. Map displays for information retrieval. Journal of the American Society for Information Science, 48(1):40–54, 1997

[27] Manku, G. S., Rajagopalan, S. and Lindsay, B.G. Random sampling techniques for space efficient online computation of order statistics of large datasets. Proceedings SIGMOD '99 Int'l Conf. on Management of Data, Philadephia, May 1999, ACM Press, 251-262

[28] NetMap Link Analysis: making the invisible, visible. 2001. http://www.netmap.com/ > Presentations > Link Analysis

[29] Olken, F. and Rotem, D. Random Sampling from Relational Databases. Proceedings VLDB '86, August 1986, Kyoto, Japan, Morgan Kaufmann, 160--169

[30] F. Olken. Random Sampling from Databases. Ph.D. dissertation, UC Berkeley, April 1993, LBL Technical Report 32883

[31] Piatetsky-Shapiro, G. and Connell C. Accurate estimation of the number of tuples satisfying a condition. Proceedings SIGMOD '84, Boston, June 1984, ACM Press, 256–276

[32] Pirolli, P., Schank, P., Hearst, M. and Diehl, C. Scatter/Gather browsing communicates the topic structure of a very large text collection. Proceedings CHI '96, Vancouver, May 1996, ACM Press, 213–220

[33] Pirolli, P. Computational Models of Information Scent-Following in a very Large Browsable Text Collection. Proceedings CHI '97, March 1997, ACM Press, 3-10

[34] Dix, A. Query-by-Browsing on the Web. 2001. http://www.meandeviation.com/qbb/qbb.php

[35] Raman, R. Random Sampling Techniques in Parallel Computation. IPPS/SPDP Workshops 1998, 351–360

[36] Rao, R. and Card, S. The Table Lens: Merging graphical and symbolic representations in an interactive focus + context visualization for tabular information. Proceedings CHI '94, Boston, ACM Press, 1994, 111–117

[37] Salton, G., Allan, J., Buckley, C. and Singhal, A. Automatic analysis, theme generation and summarization of machine-readable texts. Science, 264:141–1426, 1994

[38] Schneier, B. Applied Cryptography second edition. Wiley, 1996

[39] Shneiderman, B. Designing the User Interface, Third Edition, Addison-Wesley, 1998

[40] Skalak, D. B. Prototype and feature selection by sampling and random mutation hill climbing algorithms. Proceedings of the Eleventh International Machine Learning Conference, Morgan Kaufmann, New Brunswick, NJ, 1994, 293--301

[41] Spence, R. Information Visualisation. Addison-Wesley, 2001

[42] Tweedie, L., Spence, R., Williams, D. and Bhogal, R. The Attribute Explorer. Video proceedings CHI '94, ACM Press, 1994

[43] L. Tweedie, R. Spence, H. Dawkes and H. Su. The Influence Explorer. Companion Proceedings CHI '95. ACM Press, 1995, 129-130

[44] Tweedie, L., Spence, R., Dawkes, H. and Su, H. Externalizing abstract mathematical models. Proceedings CHI '96, ACM Press, 1996, 406–412

[45] Williamson, C. and Shneiderman, B. The Dynamic HomeFinder: Evaluating dynamic queries in a real-estate information exploration system. Proceedings SIGIR '92, ACM Press, 339-346

[46] Woodruff, A., Landay, J. and Stonebraker, M. Constant Information Density in Zoomable Interfaces. Proceedings AVI '98, L'Aquila, Italy, ACM Press, 57-65

[47] Woodruff, A., Landay, J. and Stonebraker, M. Constant Density Visualizations of Non-Uniform Distributions of Data. Proceedings UIST'98, San Francisco, 1998, 19-28