Dynamic Dynamic Queries (DDQ)

P.J. Teuben

University of Maryland, College Park

Abstract.
We describe an implemention of Dynamic Queries for arbitrary tabular data.

1. Introduction

Dynamic Queries (DQ, see also Shneiderman, 1993) is a recently developed querying technique which proved to be a good alternative to the often difficult to learn but very general and flexible SQL-type interface to a relational database. The query in DQ is formulated using a number of graphical widgets that represent a column from a table (columns can be numeric as well as textual). Using two sliders a range search can be set for a column, and the resulting view of this multiply OR-ed selection is dynamically viewed in a canvas. The canvas can display simple points, but additional information can optionally be added to these points (color, size, etc.). Once sufficient selections have been made, some implementations of DQ then allow the displayed data to be queried in more detail by e.g. clicking on the point. A number of specific examples of DQ have been discussed in recent years (e.g. HomeFinder xxx 19xx, MovieFinder yyy 19yy).

This paper discusses a generic implementation where the input dataset can be an arbitrary table (currently all collumns have to be numeric). A selection of the viewing coordinates X and Y is made, as well as the columns that can be queried. In addition to the above described features, we also implement a generic (user definable) interactive analysis tool that is associated with the dynamic query which will be useful for many situations.

2. Other

2.1. PDQ Trees

Browsing hierarchical data is exemplified with multi-level dynamic queries and pruning ("PDQ Trees", see Kumar et al, 1995).

2.2. Starfield

What we do in this paper is really the starfield approach (see e.g. Jog & Shneiderman, 199x) including interactive zooming of the canvas.

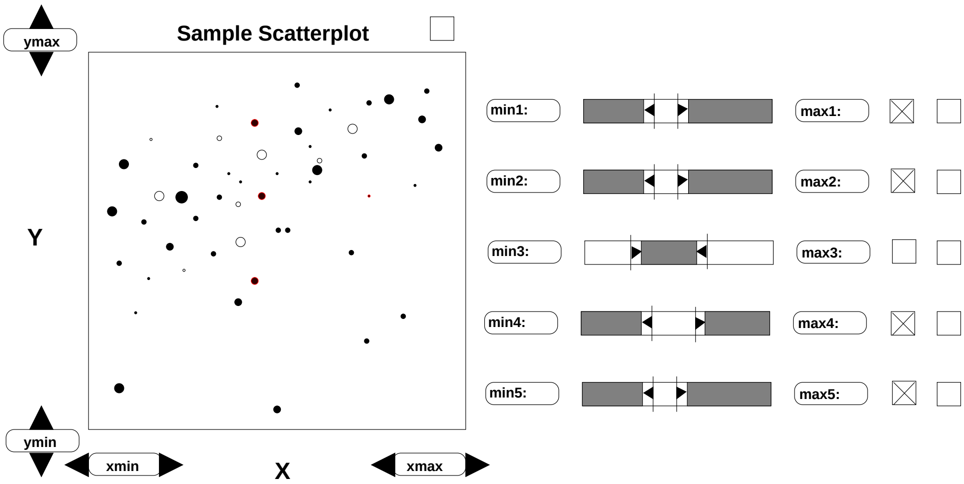

Figure 1. Overall layout of DDQ; sliders on the right (see also Fig. 2), the dynamic view of the selected X and Y coordinates in a scatter plot on the left. The X-Y viewport can be controlled by the xmin/xmax and ymin/ymax buttons as well as selecting a center and zooming in or out.

3. Implementation

The current "ascii-widget" implementation can be found in the `tabview` program in the NEMO (Teuben 1995) package. It currently uses the PGPLOT (Pearson 19xx) graphics package as a display engine.

3.1. Data Structures

The input table is considered to be a M-dimensional (columns) dataset with N values (rows). N would be the number of observations, which M parameters for each observation (ref. multi-variate?). Sticking to the two-dimensional aspect of our display screen (the canvas), we choose 2 variables (or mathematical expressions derived from any of the M columns).

3.2. Formats

The following data-structures can be (well: are to be) read:

- unstructured "commented" ascii tables
- FITS ascii TABLE (e.g. ADS-1 cdrom)
- FITS BINTABLE (e.g. ADS-1 cdrom)
- IHL
- RawSpice

3.3. Startup: ascii widgets

In the "ascii widget" implementation a generic graphics library was used to plot the data, but the slider widgets were implemented in an interactive command-line based mode.

At startup a number of sliders (columns) are selected for DQ interaction, and the program first overviews simple statistics and all their possible correlations are computed. A sample output with 6 sliders would be:

```
nemo> tabview in=p100.list col=1,2,3,4,5,6 xvar=%1 yvar="sqrt(%2**2+%3**2)"
Reading table p100.list with 6 columns and 100 rows

### Overview statistics:
Slider    : Npts      min      max      mean    sigma skewness kurtosis
1         :  100  -7.1859   5.7947  1.43e-07   1.2811  -1.1083   12.732
2         :  100  -2.3444   7.3121 -5.83e-08   1.1374   3.5281   19.521
3         :  100  -3.4209   5.2131 -1.97e-07   1.1072   1.0584   5.8222
4         :  100 -0.78246   1.1218 -4.76e-08   0.3856   0.3441  -0.1828
5         :  100 -0.95056   1.0854  7.13e-08   0.3906  -0.0894  -0.1902
6         :  100  -1.1176   1.0182  4.87e-08   0.3762  -0.0878   0.3303

### Pearson correlation matrix:
   :     1     2     3     4     5     6
 1 :   1.00
 2 :   0.22  1.00
 3 :   0.54  0.14  1.00
 4 :  -0.07 -0.03 -0.10  1.00
 5 :   0.07  0.02 -0.06  0.06  1.00
 6 :  -0.12 -0.07 -0.08  0.02 -0.06  1.00
Xvar %1: displayed min= -7.51038 max= 6.11921
Yvar %2: displayed min= -2.58581 max= 7.55352
```

The prompt in "asii-widget" mode is that of a slider:

```
Slider 1 [-7.18587 5.7947 0 5.7947]:
```

displaying, from left to right, the minimum, low slider, high slider and maximum of the data. The following commands currently implemented:

```
    l <num>     modify 'lo' slider
    h <num>     modify 'hi' slider
    l s <step>  set step in 'lo' slider
    h s <step>  set step in 'hi' slider
    b <step>    step both 'lo' and 'hi' slider
    <digit>     change slider to interact with
    r           reset lo/hi to min/max for this slider
    s           show min/lo/hi/max for all sliders
    u           update screen new
    o <file>    output table, w/ optional override out= filename
    x           swap lo and hi (invert logic)
    q           quit
    !cmd        execute a shell command 'cmd'
    |cmd        pipe visible data as ascii table to 'cmd'
    ?           this help plus status
```

Notice that moving the low slider to the right of the high slider is equivalent to inverting the logic (the `x`-command.

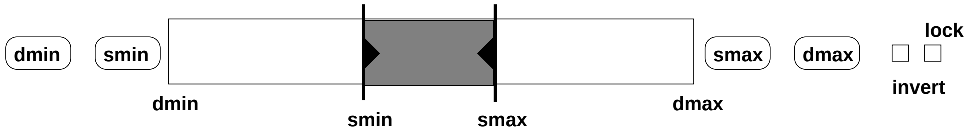

Figure 2. An expanded view of the slider widget that would be needed. The four buttons 'dmin', 'dmax', 'smin', 'smax' on either side of the slider can also be edited. The lock and invert check buttons on the right determine the way the slider moves and if the inside or outside is greyed.

3.4. Post-Processing

At each step of the interaction the visible data (or it's complement) can be post-processed. At each step during the DQ interaction, an action (the | command, modeled after the unix pipe, pipes the X-Y data to any command of the users choice. One can think for example of a post-analysis program computing the slope and intercept and displaying these grapically in another display (naturally the state of the DQ is also saved in some accompanying database for later analysis). The selection of this post-analysis program is completely up to the user. Some example analysis programs, like a general least squares fitting program, have been very useful. It also turned out to be useful to allow interactively flagging of the visual data directly in the canvas using either nearest point or polygonic areas, making this a very powerful interactive analysis tool.

4. Missing features

Apart from a GUI interface, there are still some fundamental features missing from the current implementation for which some more general datastructure may be needed. The current slider widget allows simple range selections, but is lacking some other frequently used selection mechanisms with their SQL equivalence:

- alphaslider - where the entries are textual instead of real numbers
- radio (?) button - select one of many
- radio (?) button - select any of many
- multiple ranges for the same column

Acknowledgments. I want to thank the Human-Computer Interaction Laboratory at the University of Maryland, and B.S. & C.P. specifically, for their inspiring work that lead to this investigation.

References

Shneiderman, B. (Jan. 1993) Dynamic Queries: for visual information seeking, IEEE Software, vol. 11, #6 (Nov. 1994) 70-77

Kumar et al. 1995 (PDQ Trees)

Jog & Shneiderman, 199x
Brotzman, L. (FTB: FITS Table Browser)
Teuben, P.J. - ADASS-V conference
Pearson, T.J. - ULR?