maxdView User Guide

Version 1.0.5 − March 2005

http://www.bioinf.man.ac.uk/microarray/maxd/

This document was generated automatically from the built−in help pages
using htmldoc (see http://htmldoc.org). These help files can also be viewed at the 'maxd' website, or via the help browser built into the application.

Table of Contents

Table of Contents

Table of Contents

1 maxdView Overview

1.1 What is maxdView?

maxdView is a system for integrating techniques for analysis and visualisation of expression data.

The design is focused on providing a comprehensive model of expression data and a flexible, extensible architecture to allow functionality to be added as desired, and as new techniques are developed.

maxdView provides three ways of integrating algorithms and visualisations.

- Write Java 'plug−in' modules which are installed in the **maxdView** menu hierarchy and have full access to the internal data representation and underlying event model (see more information on plugins).
- Enter 'fragments' of code directly into a dialog box and compile and execute it dynamically with a single button press. Code run in this way has complete access to the internal data representation (see the Code Runner plugin).
- Interface to external programs or scripts, with two−way exchange of both symbolic and numeric data. The data transfer can be via standard input and output or via temporary files (see the Run External plugin).

maxdView is by no means a complete analysis package, for example it has no facilities for image processing. It is not intended to be an all−encompassing solution. Instead, it's purpose is to provide an infrastructure into which existing tools can be fitted, and to assist rapid prototyping of new algorithms and visualisation techniques.

1.2 What sort of data does it handle?

The core of the **maxdView** data model is the **ExprData** class which represents expression data, comprising inner classes modeling the following features:

- **Measurements** represent collections of expression levels (or other data associated with image analysis process) recorded under a known condition.
- **Annotation** of each expression value records the name of the array, spot, probe, and gene(s) associated with each numerical value, and provides methods for retrieving textual descriptions of biological entities from different sources. Annotation can be retrieved from different sources then browsed and searched.
- **Clusters** are a fairly general purpose way of handling connections between groups of entities within a data set. **maxdView** models clusters as a hierarchical collection of sequences of items.

1.3 What can it display?

The main window of **maxdView** displays the conventional table view of expression data, with the rows showing the different spots on an array, and the columns showing the different measurement conditions for which values are known.

Rows can be labelled with the name of the spot, the probe in the spot, or the names of biological entities known to be targeted by the probe. The table view is augmented with a dendrogram showing relationships between rows. These relationships can be created and modified interactively in a variety of ways.

Additional visualisations of the data are provided by plugin modules, including a co−relation plots (Stack Plot, Scatter Plot), multi−dimensional projections (Profile Viewer, Web Plot and HyperCube Plot) and distribution histograms (Distogram).

1.4 What can it do?

maxdView provides a number of useful data transformation operations, such as sorting, and filtering the data values in a variety of ways.

New transformations can be added via any of the three integration routes outlined above. For example, if a new filtering technique is installed as a plugin class, it will become available to all of the other parts of **maxdView**.

The current version of **maxdView** provides plugins for reading and writing native XML and plain text files and for loading data to and from **maxdSQL** databases. Additional methods for importing and exporting data can be added via the plugin system.

See Also:

- Concepts
- Glossary
- Data Model
- Commands
- Plugins
- File formats

2 maxdView Concepts

2.1 Spots, Names and Name Attributes

Spots are **maxdView**'s basic unit of data. **Spot**s represent all the values associated with a single cell on an array.

Each **Spot** has a name and also records the name of the **Probe** contained in that **Spot**. **Probe** refers to the biological material placed in the cells of an array. Each **Probe** may be linked to zero or more **Gene** names.

The **Spot** names must be unique within a data set as it is these names which are used to identify a specify spot on an array. The **Gene** and **Probe** names can be replicated to model duplicate elements on the array.

All three names of name can have optional **Name Attributes**. Attributes are text values that can be associated with Spot, Probe and Gene names. Gene names, for example, might have "Accession number", "Description" and "Function" attributes used to record information about each gene. Names and their attrbiutes are available for use in searching and other text matching operations.

The Name Tag Editor is used manipulate all types of name and their attributes. It is created from the popup menu that appears over spot and name columns in the main display.

The Name Munger plugin can be used to manipulate both names and name attributes tags in a variety of ways.

maxdView does not presently specify, or support, any particular naming convention. Names are stored and manipulated as Java String objects, and conventional string matching rules determine whether one name is the same as another.

See a picture of the Data Model for more information about this topic.

2.2 Measurements

A **Measurement** is collection of numbers representing values recorded from a set of **Spots**. It corresponds closely to a single column a numbers in a spreadsheet or traditional flat file.

Measurements have a **Data Type** which determines how they will be coloured for display. The **Data Type** can also be used to group **Measurement**s together, for example applying a command to all **Measurement** of a particular type.

Any number of **Measurement Attribute**s, i.e. *name−value* pairs, can be associated with each **Measurement**. This data can be accessed via the "Attributes" tab found on the Measurement Manager plugin.

Additional data values for each **Spot** can also be linked to a **Measurement**. This sort of data, called **SpotAttribute**s can record supplimentary information about the numerical levels, such as confidence levels, signal intensities, and derived statistical values. The Spot Attributes plugin can be used to see these values.

See a picture of the Data Model for more information about this topic.

2.3 Filters

A Filter is a rule applied to each of the **Spot**s in turn. **Spot**s which do not obey the rule are trapped by the filter and are temporarily ignored by the rest of maxdView. This feature is useful for reducing the complexity of analyses or visualisations by discarding uninteresting **Spot**s. It can also be used to locate **Spot**s which match particular criteria.

Using Filters does not permanently remove Spots, they are only hidden whilst the Filter is enabled. Disabling, or altering the filter will cause hidden Spots to reappear.

maxdView allows multiple **Filter**s to be active simultaneously. They are applied in turn, in the order in which they were created. If a **Spot** is trapped by *any* of the **Filter**s, it will be ignored.

A collection of general−purpose **Filter**s is supplied with **maxdView**. New ones can be created internally using the Code Runner, or defined externally as **Plugin** classes.

See also:

- Filter by Name or Value
- RegExp Filter
- Filter By Clusters
- Filter By Selection
- Math Filter
- Multi Filter
- Profile Filter

2.4 Clusters

A Cluster is a collection of Spots or Measurements that have been grouped together. Things might be grouped because they have similar expression profiles, share some common annotation, or for some other reason. The Spots or Measurements in a Cluster are termed its *elements*.

In addition to elements, **Cluster**s can also contain one or more child **Cluster**s. This allows hierarchical structures to be contructed, to capture, for example, the output of an external clustering program.

Each **Cluster** has a number of attributes associated with it; including a name and a coloured glyph. These attributes, and the hierarchy of **Cluster**s themselves, can be manipulated in the Cluster Manager.

Clusters can be defined using any of the names associated with a **Spot**. A **Cluster** with elements defined by **Spot** name can be re−applied to any other **Measurement** from the same type of array. **Cluster**s with elements defined using **Probe** or **Gene** names can be reapplied to **Measurement**s from arrays containing any of these entities.

See also:

- Tutorial: Working with Clusters

2.5 Annotation

Annotation refers to any text which is associated with a Probe or Gene name. Annotation for a particular entity can be gathered from multiple sources (i.e databases like EMBL and SwissProt) and stored locally to reduce data loading times.

More information:

- Annotation Loader
- Annotation Viewer

2.6 Events

maxdView is an *event−driven* system, which means that as the data is updated, each of the system's components are notified by being sent an Event.

Different types of event are dispatched depending on how the data has changed. This enables the various components of the system decide whether they need to take any action in response to the change.

To see behind the scenes, use the Event Watcher which reports events as they occur.

2.7 Plugins

Plugins are Java classes which can be loaded into **maxdView** whilst it is running. The code for the plugin classes is not loaded until the plugin is needed, which saves space and reduces the applications start−up time.

Plugins provide an easy way to extend the **maxdView** core functionality to add extra capabilites. Plugins can have a user interface and manipulate and display data in any way they want. Plugin classes can also receive **Event**s, enabling then to react to changes made by other components.

See also:

- Using Plugins
- Programmer's Guide

2.8 Colourisers

Colourisers are objects which convert a numerical value into a colour for display.

Colourisers are created and edited using the Colouriser panel accessed from the "Display" menu. The assignment of Colourisers to Measurement is done ia the Measurement Manager plugin or the Measurement name popup menu in the main display window.

2.9 Data Merging

When data is loaded (via any of the importing plugins; "Read Native", "Load Plain Text", and "Database Loader") it can either replace the existing data or be merged with it.

Merging is done by matching either the Spot, Probe or Gene names associated with each spot. Names which are not recognised can be either ignored or allocated to new Spots. More details can be found in the Merging data help page.

2.10 Custom Menu

The Custom Menu is accessed via the context sensitive popup menu used in the main display. Press the *right* mouse button whilst pointing at a Name, Spot, Measurement or Cluster in the display. The popup menu which appears has a "Custom" submenu containing the option "Edit" which opens the custom menu editor.

The editor allows to add and remove *command* enties from the custom menu. Commands are associated with **plugins** and allow you to access a plugin functions without launching it's user interface.

2.11 Drag and Drop

Drag−and−drop is an easy method for moving information (such as Spot, Measurement and Cluster names) between parts of **maxdView**.

Press and hold the left mouse button to start the drag. A small folder icon appears beside the mouse cursor. This icon can then be 'dragged' to another window and 'dropped' by releasing the mouse button.

Drag−and−drop is presently supported by:

- Spots, Measurements and Clusters in the main display
- The "Find" dialog box
- Clusters in the *Cluster Manager*'s tree
- The *Notepad* plugin

- The *Sort Clusters* plugin
- The *Scatter Plot* plugin
- The *RegExp Filter* plugin
- The *Stack Plot* plugin
- The *Profile Filter* plugin
- The *HyperCube Plot* plugin
- The *Profile Viewer* plugin
- The *Just−o−Clust* plugin
- The *Compare Clusters* plugin

2.12 Selection

Data can be selected by clicking on elements in the main display window, or programatically via the selection API. Spots, Measurements and Clusters can be selected independantly.

Selected items are highlighted in the main display. Selected Spot and Measurement names are drawn in inverted colours and selected Clusters are drawn with a surrounding box.

The Filter By Selection plugin can be used to show or hide the selected Spots.

Spots are selected by clicking on them in the main display window. An extended selection can be made by clicking on the start Spot and then holding the 'Shift' key and clicking on the end Spot. Holding the 'Ctrl' key when clicking on a Spot toggles the selection of that Spot.

Measurements are selected in the same way, i.e. by clicking on their names. The 'Shift' and 'Ctrl' key can be used as before.

Clusters are selected as follows:

click	select just this Cluster
Shift + click	add this Cluster to the selection
Ctrl + click	toggle selection of this Cluster

Note: The 'Ctrl' key is no longer used to activate the popup menu (for systems with a single−button mouse) instead use 'Ctrl' and 'Alt' mouse−click to activate the popup menu.

The **Selection Menu** is accessed via the context sensitive popup menu used in the main display. It contains entries which manipulate the selection. See the Selection Menu help page for full details.

See Also:

- Glossary
- Overview
- Commands
- Plugins
- File formats

3 Plugins

3.1 About plugins

Plugins provide an easy way to extend **maxdView** to add extra capabilites. Plugins can have a user interface and manipulate and display data in any way they want using the **maxdView** core as a support environment.

The Java code for the plugins is not loaded until the plugin is *launched* by the user. This speeds up the loading of the main application, and reduces the memory footprint as functions which are not used never take up any space.

Plugins are kept in a directory called `plugins/`, which is futher subdivided into directories for the main types of plugin. When *scanning* for plugins, the application will search any class files and directories in the file system starting at the "plugins/" directory.

You can find out more about the plugins that are currently known to the system on the "About" help page.

Plugins can provide commands which allow you to access them from within your own code. You can write simple scripts which use these commands with the Code Runner plugin. A complete list of all plugin commands is available.

Plugins that are downloaded from the **maxd**website, or created 'locally' by a user will be installed in the *per−user* plugin hierarchy, which means they will only be available to the user who installed them. The *per−user* plugin hierarchy is seperate from the *shared* plugin hierarchy, which contains the set of 'standard' plugins that are shipped with **maxdView**. If a plugin in the *per−user* hierarchy has the same name as one in the *shared* hierarchy, then it will be used preferentially.

3.2 The Plugin Manager

The Plugin Manager is activated using the entry on the **System** menu.

The Plugin Manager can automatically download new plugins (and updated versions of existing plugins) from the **maxdView** web site. It can also be used to un−install plugins that are not required and to change the order in which plugins are listed in the main menu.

In addition, the Plugin Manager provides the "Rescan Plugins" feature which is used to 'manually' register new plugins with the system.

3.3 The supplied plugins

The set of plugins distributed with this version of the **maxdView** is:

- App In A Box
- Cluster Manager
- Code Runner
- Compare Clusters
- Database Loader
- Distogram
- Event Watcher
- Filter By Clusters
- Filter By Name Or Value
- Filter By Selection
- HyperCube Plot
- Just−o−Clust

- Load Plain Text
- Math Filter
- Measurement Manager
- Multi Filter
- Name Munger
- Normalise
- Notepad
- Old Load Plain Text
- Profile Filter
- Profile Viewer
- Read Native
- RegExp Filter
- Raw Write
- Reorder Measurements
- Run External
- Save As Text
- Save To Database
- Scatter Plot
- Simple Maths
- Sort Clusters
- Sort by Name or Value
- Space Plot
- Spot Attributes
- Stack Plot
- Super Grouper
- SVD
- Text Explorer
- Web Plot
- Weka Cluster
- Write Native
- XCluster
- XML Exporter

(plugins coloured like this utilise public domain software that is distributed with maxdView)

(plugins coloured like this require extra software that is not part of the maxdView distribution)

(plugins coloured like this are provided for compatibility or testing reasons)

(plugins coloured like this are new and/or experimental and therefore might not work as advertised and may have incomplete or misleading documentation)

The Commands help page contains an entry for each plugin currently available.

See also:

- Programmer's Guide
- Writing a Plugin tutorial

4 The Popup Menu

The popup menu appears in the main display when the right mouse button is pressed (alternatively via 'Ctrl+Alt+left mouse' or 'Ctrl+SpaceBar').

The popup menu can be removed by pressing the 'Escape' key.

Each menu has the common entries "Selection" and "Custom". In addition to these, the menus contain entries relevant to the entity under the mouse cursor when the menu was opened.

There are four possible menus depending on where in the display the menu is invoked:

- Name or Name Attribute
- Measurement Name
- Cluster
- Spot

The *Name or Name Attribute* menu

Display Annotation	Open a new Annotation Viewer and load the annotation for this Spot
Edit Names &Attrs	Start the Name Tag Editor for this Spot
Show in this column...	Select which Name or Name Attribute to use in this column (see the layout control panel)
Add another column	Add another name column after this one
Remove this column	Removes this name column from the display
Sort this column	Reorders the Spot rows so that the values in the column are in alphabetical order

The *Measurement Name* menu

Hide this Measurement	Hide this Measurement (see the Measurements control panel)
Show all Measurements	Make all hidden Measurements visible
Show Properties	Open the Measurement control panel
Pick Colouriser	Display a list of available Colourisers
Show Attributes	Open the Measurement attributes panel

The *Cluster* menu

Show Properties	Open the Cluster Manager
Hide children	Hide the children of this Cluster
Show children	Show all of the children of this Cluster
Hide parent	Hide the parent of this Cluster
Show parent	Show the parent of this Cluster
Show parent and siblings	Show the parent of this Cluster and all of the other children of the parent
Hide all others	Hide all other Clusters except this one
Show all others	Show all other Clusters

The *Spot* menu

Display Annotation	Open a new Annotation Viewer and load the annotation for this Spot
Edit Names & Attrs	Start the Name Tag Editor for this Spot
Sort ascending	Reorder the Spots based on the values in this column
Sort descending	Reorder the Spots based on the values in this column

The Selection Menu

The **Selection** sub−menu contains options for manipulating the data selection. See the Selection Menu help page for full details.

The Custom Menu

All popup menus contain the **Custom** sub−menu where you can collect frequently used commands for easy access. See the Custom menu help page for more information.

See also:

- The Custom menu
- The Selection menu

4.1 The Custom Menu

The *Custom Menu* allows quick access to frequently used commands. You can define submenus within the *Custom Menu* to group related commands together.

The custom menu can be found on the popup menu available in the main display window.

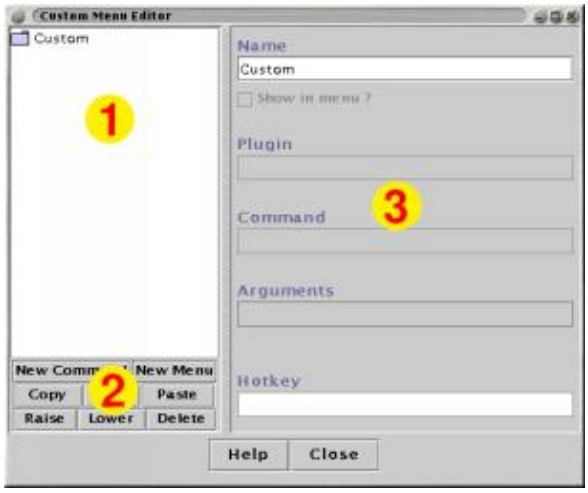
Custom menu commands can also be activated using a keyboard shortcut or *hotkey*. The key sequence used to execute the command is assigned using the editor.

Note that the custom menu and the hotkey functions are only available in the main display window and not in windows created by plugins.

4.1.1 The Custom Menu Editor

To open the *Custom Menu Editor* choose "Custom−>Edit" from the popup menu in the main display. The popup menu is triggered using the *right* mouse button (or `Ctrl`,`Alt` and the left mouse button).

The *Custom Menu Editor* is divided into three sections:

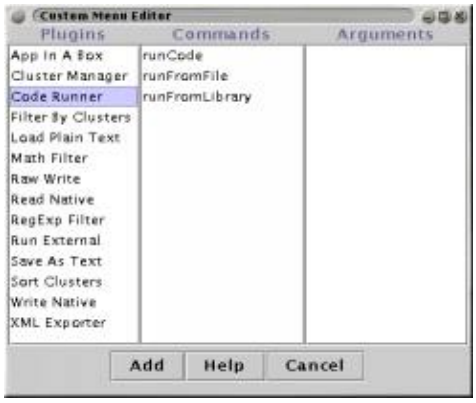


1. the menu hierarchy; double click on a menu to expand or collapse it
2. control buttons for manipulating commands and menus in the hierarchy
3. type−in fields for defining the hotkey and arguments of the selected command

---

4.1.2 Adding a command

Use the "New Command" button to add a new entry to the currently selected Menu. The *Add Command* panel is displayed:



The *Add Command* panel is composed of three lists, "Plugins", "Commands" and "Arguments". When one of the "Plugins" is selected, the commands offered by that plugin are displayed. When one of the "Commands" is selected, the arguments (parameters) understood by that command are displayed.

Full details about each plugins commands and their arguments are given on the help page of the relevant plugin.

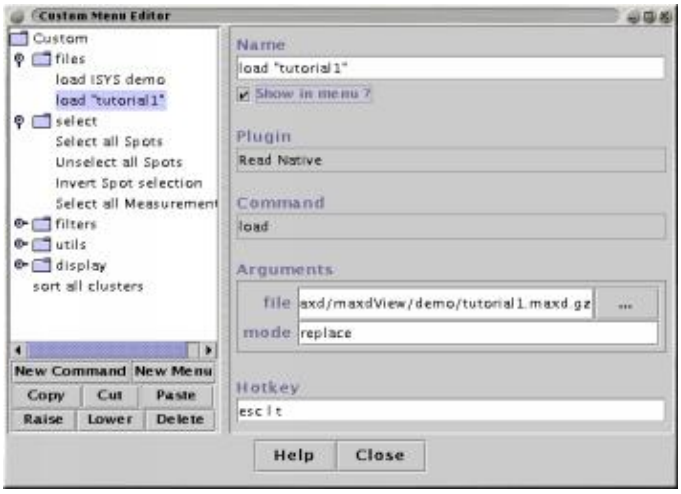
Built−in commands are can also be accessed via this panel.

Select a plugin and a command then use the "Add" button to add this new command.

4.1.3

Setting the name, arguments and hotkey

Select one of the Menus or Commands in the tree to display it's details in the fields on the right hand side of the panel.



Commands and Menus can be given a descriptive "**Name**" which defines the text that will used in the popup *Custom Menu*. By default, new commands are named "Plugin.command".

Most Commands understand one or more "**Arguments**". The *Custom Menu Editor* displays a list of the arguments of the selected command. The values for these arguments can be edited via the type−in fields.

Full details about the commands offered by each plugin are given on the help page of the repective plugin.

Commands can optionally have a "**Hotkey**". There are three methods for specifying the hotkey(s) that will run the command without you having to open the custom menu.

- **ALT−C mode**

  The prefix `Alt−C` followed by a `Alt` plus a unique character can be used as a hotkey. The required single character should be typed into the "Hotkey" field. For example, setting a command's "Hotkey" value to:

```
X
```

will cause the command to be executed when "Alt−C" then "Alt−X" are pressed.

- **Escape mode**

The 'escape' key followed by any sequence of letters or digits can be used to trigger a command. The format for specifying the sequences is as follows:

```
esc F A 1
```

This hotkey specification causes the command to be run when the user presses 'escape' then types 'F' followed by 'A' followed by '1'. The sequence must begin with the string '`esc`' and contain one or more letters or digits. Whitespace in the hotkey specification is ignored.

When using the 'escape' hotkey mode, the first hotkey specification to match the sequence of keys pressed will be executed. If you specify one commands hotkey as "`esc B`" then specifications such as "`esc B 2`" will not be accessible. The "`esc B`" command will always be chosen before the "`esc B 2`" specification can be seen.

- **Function key mode**

The function keys can be used to trigger commands. The format to specify a function key hotkey is the string "`fn`" followed by a number in the range 1 to 16. For example, setting a command's "Hotkey" value to:

```
fn 12
```

will cause the command to be executed when "F12" is pressed.

4.1.4 Editing the menu hierarchy

- New Command
  Add a new Command to the currently selected Menu (see above)

- New Menu
  Add a new Menu as a child of the currently selected Menu
  You can change the name of the Menu in the same way as for Commands

- Cut
  Remove the currently selected Menu or Command to the clipboard

- Copy
  Copy the currently selected Menu or Command to the clipboard

- Paste
  Insert the clipboard contents into the currently selected Menu

- Raise
  Move the currently selected item up the tree

- Lower
  Move the currently selected item down the tree

- Delete
  Remove the currently selected item from the tree

4.1.5 Built−in Commands

```
[ExprData]

 Select all Spots
 Unselect all Spots
 Invert Spot selection
 Select all Measurements
 Unselect all Measurements
 Invert Measurement selection

[DataPlot]

 Set zoom scale
 Zoom in
 Zoom out

 Open Measurements Dialog
 Open Find Dialog
 Open Layout Dialog
 Open Colouriser Dialog
 Open Annotation Loader Options Dialog
 Open Custom Menu Editor Dialog

 New View
 Open Print Dialog
 Apply Filter
 Do Not Apply Filter

[maxdView]

 Rescan plugins
 Exit
```

See Also:

- Commands
- Plugins

4.2 The Selection Menu

The **Selection Menu** contains options for manipulating the data selection. It is accessed via the context sensitive popup menu used in the main display. Press the right mouse button (or 'Ctrl + 'Alt' + left button) whilst pointing at a Spot, Measurement or Cluster in the display. The popup menu which appears has a "Selection" submenu with different options depending on what was pointed at when the menu was opened.

When over a Spot or a name column...

Select all	Puts all Spots in the selection
Unselect all	Clear the Spot selection
Invert selection	Select all Spots which are currently deselected and vice−versa
Add filtered Spots	Add the Spots which pass the current filter(s) to the selection
Remove filtered Spots	Remove the Spots which pass the current filter(s) to the selection
Send to...	Send the selected Spots to a registered data sink

When over a Measurement name...

Select all	Selects all Measurements
Unselect all	Clears the Measurement selection
Invert selection	Select all Measurements which are currently deselected and vice−versa
Send to...	Send the selected Measurements to a registered data sink

When over a Cluster...

Unselect all	Clear the Cluster selection
Convert to spot selection...	Selects all of the Spots which are in the selected Clusters
Send to...	Send the selected Clusters to a registered data sink

---

See also:

- The Custom menu
- The Popup menu
- Concepts: Selection

5 File Menu

5.1 Save As Text

5.1.1 Overview

Export some or all of the data in a tabular ASCII file

5.1.2 User Interface

The interface is split into four tabs:

Formatting Options	*basic layout options*
Column contents	*choose which Measurements & Spot Attributes to include*
Row Contents	*choose which Names and Name Attrbiutes to use as labels and choose whuch Measurement Attribute data to include*
Preview	*see a preview of how the file will appear and optionally adjust the order of the columns*

The controls on the first three of these tabs are used to define which data will be included in the file. The fourth tab displays a preview of the file contents. Once a satisfactory layout has been selected, press the "**Save**" button and use the file chooser that is then displayed to specify the file location and name.

The **Compress with GZIP** option employs a common compression technique to reduce the file size. The *Load Plain Text* plugin can load compressed files directly, so this is a convenient way to save disk space.

5.1.3 1. Formatting Options

Use the "**Delimiter**" drop−down menu to select between tab, space or comma delimited files.

The "**Significant Digits**" slider controls the number of decimal places that will be used for numerical data.

The "**Missing value**" field allows a custom text string to be used to represent any missing numerical values.

5.1.4 2. Column contents

The main control on this tab is a list of all Measurements and optionally the Spot Attributes of these Measurements. Select one or more items from this list for inclusion in the output file.

The "**Include column labels**" option determines whether the names of Measurements and Spot Attributes will be included at the head of each column.

The "**Remove whitespace from labels**" option replaces any whitespace (i.e. space and TAB characters) in column labels with an underscore character. This is useful if the file is to be parsed by other software which expects no whitespaces in it's input.

5.1.5 3. Row Contents

The "**Apply filter**" option restricts the output to only the Spots not removed by any currently active Filters.

The "**Include row labels for Measurement Attributes**" controls whether Measurement Attributes are labelled in the output.

The "**Remove white from labels**" option replaces any whitespace (i.e. space and TAB characters) in Measurement Attribute labels with an underscore character. This is useful if the file is to be parsed by other software which expects no

whitespaces in it's input.

Select any number of entries from the list of "**Spot Names and Name Attributes**" for inclusion in the file.

If any Measurement Attributes exist then they will be displayed in the list of the right hand side of the tab. Select any number of these attributes for inclusion in the file. Selected attributes will occur at the top of the file (as can be seen by examining the preview tab)

5.1.6 4. Preview

The preview tab shows exactly how the output file will look (although only the first twenty rows are included in the preview)

Columns in the table can be re−ordered by dragging the column heading left or right. The columns in the output file will be arranged in the same order.

Note: *this help page is incomplete.*

5.2 Save To Database

5.2.1 Overview

Store a Measurement in a maxdSQL database

See the maxdSQL and maxdLoad documentation for full details about building and populating a maxdSQL database.

When this plugin is started, the Database Connection panel will be displayed. Use this panel to login to a maxdSQL database.

5.2.2 User Interface

(In the following steps the names of maxdSQL objects are written `like this`.)

- Connect to a maxdSQL database in the same way as in the DatabaseLoader plugin.
- Pick a Measurement from the list then press "Next"
- Identify which database `Image` that the Measurement should be linked to. The "Pick" button in the "Image" panel will display a list of all `Image`s in the database. Select the one that corresponds to the Measurement.

  The selected `Image` identifies a single `ArrayType`. Details of this ArrayType will be shown in the "Array Type" panel once an `Image` has been selected. The `ArrayType` determines the collection of Spots in the database that can be matched to Spots in the Measurments.

  All of the Spot Names in the Measurement must be matched with the Spot Names on the selected `ArrayType`. (Not all of the Spots on the `ArrayType` have to be supplied, so subsets of the `ArrayType` can be saved.)

  If the Measurement was previously loaded from a maxdSQL database it will have attributes that identify which `Image` it is linked with. These attributes will be detected by the *Save To Database* plugin.
- Press the "Pick" button on the "Image Analysis Protocol" panel and choose one protocol from the list that will be displayed.

  The selected `ImageAnalysisProtocol` determines which Spot Attributes can be saved along with the Measurement.

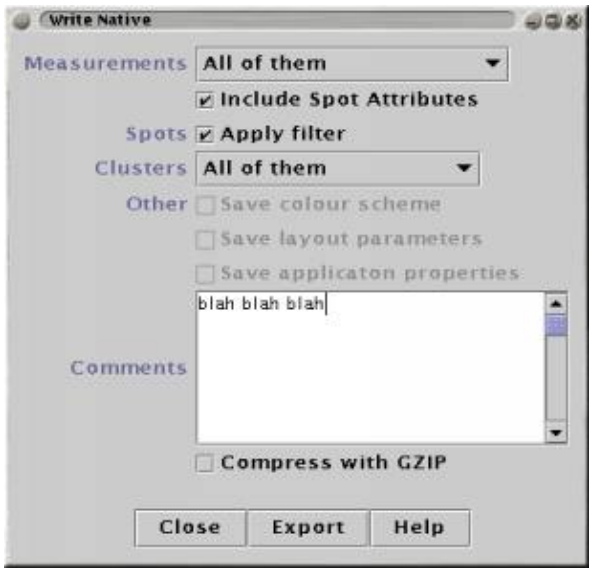
Note: *this help page is incomplete.*

5.3 Write Native

5.3.1 Overview

Export some or all of the data in **maxdView**'s native file format (which uses XML syntax).

5.3.2 User Interface



Measurements chooses between all measurements, or just those that are curently visible (Measurement visibility is controlled using the "Show?" checkboxes in the Measurements control panel).

Include Spot Attributes controls whether any Spot Attributes are stored in the file.

Use the **Apply filter** checkbox to determine whether the currently active Filters (if any) are applied to the spots as they is saved.

Clusters chooses between saving no Clusters, all of them or just those that are curently visible (Cluster visibility is controlled using the "Show?" checkboxes in the Cluster Manager).

Free−text **Comments** can also be stored with the along with the data.

The final option is to **Compress** the file. The *Read Native* plugin can load compressed files directly, so this is a convenient way to save disk space.

5.4 Database Loader

5.4.1 Overview

The *Database Loader* plugin can import data from a **maxdSQL** database using a JDBC connection.

When this plugin is started, the Database Connection panel will be displayed. Use this panel to login to a maxdSQL database.

5.4.2 User Interface

Once you have logged in, a new panel appears. This panel is initially empty

1. Choose browse table

Use the "**Browse by**" dropdown menu to select which database table is used to generate the list of Measurements. You can choose between Array, ArrayType, Experiment, Extract, Hybridisation ,and Image.

The "**and show**" dropdown menu will be updated to contain all of the entries from the table selected in "**Browse by**".

Use the "**and show**" menu to pick one instance. The Measurement list in the top−left panel will now display all Measurements which are linked to this instance.

2. Select Measurement(s)

In the top−left panel, a list of Measurement names will be displayed. The contents of the list depend on the **Browse by** and **and show** selections. Changing either of these selections results in a new list being shown.

Select a single Measurement by clicking on it. Select a continuous range by clicking on the first Measurement in the sequence and 'Shift−Click'ing on the last one. Use 'Ctrl−Click's to toggle the selection of a single Measurement.

3. View details

The top−right panel displays the details of the database relationships of the currently selected Measurements. Details include the names of the samples, extracts, protocols used to generate this data. This data will be loaded as Measurement attributes.

4. Select Spot Attributes

By default, any Spot Attributes that Measurements have will not be loaded.

In the bottom−left panel are checkboxes for the Spot Attributes of the selected Measurements. Each Spot Attribute can be independantly selected for loading.

When more than one Measurement is selected, checkboxes for all spot attributes found in any selected Measurements are shown. If a spot attribute is not present in all selected Measurements, then the number of Measurements that do have it is shown beside the name.

5. Select Gene, Probe and Spot details

Additional data may exist in the database for each of the Spot, Probe and Gene names. This information can be loaded into **maxdView** as Name Attributes.

Controls in the bottom−right panel determine whether these details are loaded for each of the name types.

6. Load the data

Once you have selected all of the Measurements (and additional data) you are interested in, use **Merge** to add them to the existing data, or **Replace** to remove the existing data prior to loading. See the Merging data help page more details on how merging is performed.

5.5 Load Plain Text

5.5.1 Overview

This plugin imports data from a plain text file.

The file can be in any of the encodings supported by Java, including ASCII and several types of 8 and 16 bit Unicode encodings.

Any type of column−oriented data can be loaded, for example:

```
                   ROW   COL   Time_0    Valid   Time_10    Valid
hum_xrna_23         1     1    2.193427    Y     1.151119     Y
hum_xdna_23_II      1     2    8.64423     N     −1.922       Y
hum_xrna_94         1     7    2.6543      Y     +1.4226      Y
hum_xrna_94         5     2    −1.73525    Y     −1.24396     Y
hum_xrna_94 II      5    12    1.0916      Y     −8.66144     N
```

The columns can be delimited by space, commas or TAB characters.

Missing numerical values can be signified by an empty column on the row, or by a user−specified text string.

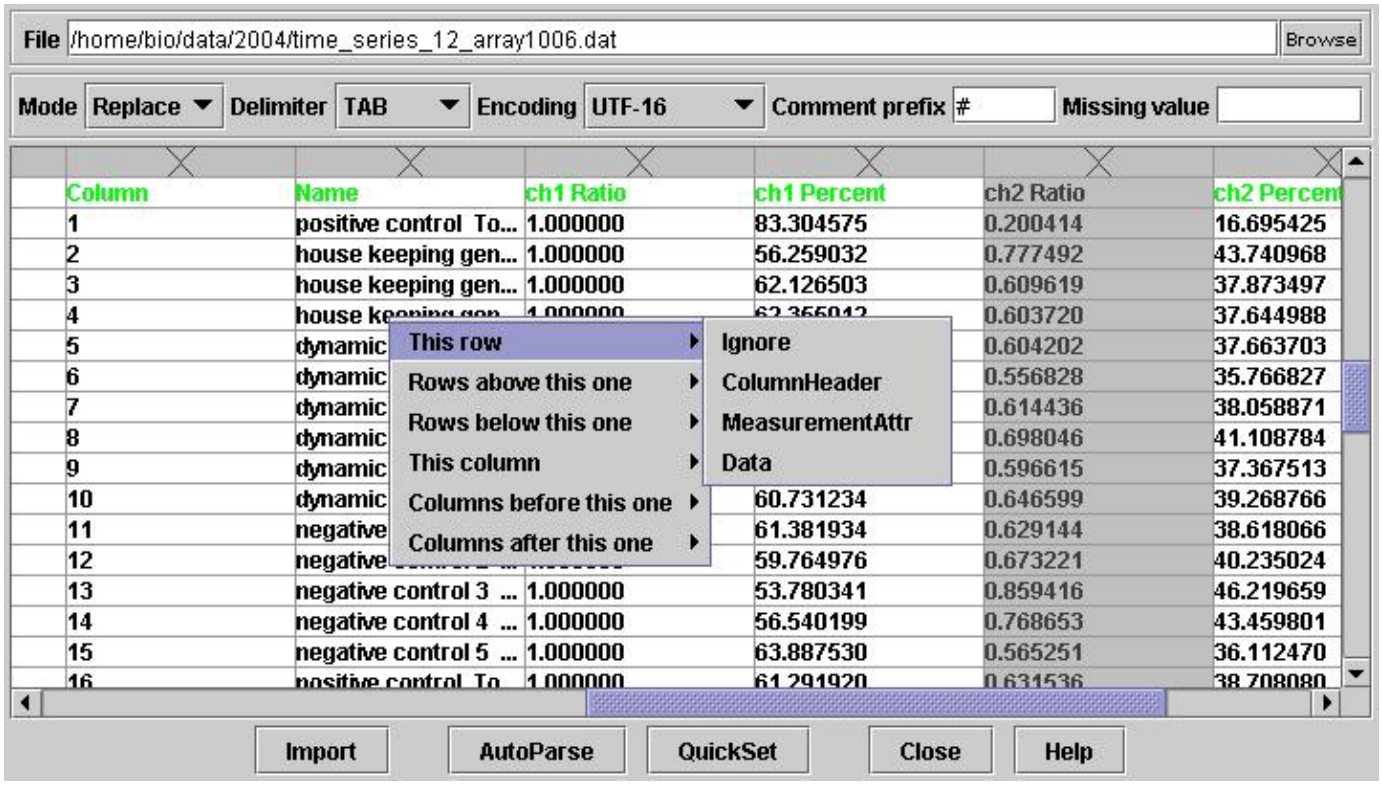
Files compressed using GZIP can be loaded directly without decompressing them first (the filename must have a ".gz" or ".zip" extension.

Data is extracted from the file by tagging each of the rows and columns to indicate what type of data they contain (or whether they should be ignored). There are three methods by which the rows and columns can be tagged:

- Drop−down controls shown on the left of each row and the top of each column
- A popup−menu which can be opened on any cell in the table
- The **QuickSet** dialog box provides a fast way to tag sequences of rows or columns in one step

Several examples are provided to illustrate the various ways in which data can be extracted files.

5.5.2 User Interface



Two lines of controls at the top of the panel are used to select the source file and determine the basic parsing options:

- Enter the **File** name directly into the type−in field, or use the **Browse** button to open a file selection dialog box. Once a file has been chosen, it will be loaded at the contents will be displayed in the table.

- The **Mode** drop−down selects between replacing the current data with the contents of the file, or merging the new data with the existing data.
  See the Merging data help page more details on how merging is performed.

- The **Delimiter**, which is the character that is used to separate columns, can be selected as either "TAB", "Space" or "Comma".
  Changing this option will cause the file to be re−parsed and re−displayed.

- A string can be specified as the **Comment prefix**. Lines in the file which begin with this prefix will be ignored.

- A string can be specified as the **Missing value**. When it occurs in the file, this string will be ignored.
  For example, if the data file contains the string "BLANK" to signal an unknown value, enter "BLANK" in this control and all occurrences of the string will be ignored.
  Missing data values will be represented as "NaNs" (Not−A−Number) when the data is loaded.

Once the file has been loaded, the main panel of the plugin displays the contents of the file in a table.
The table also contains some controls which are used to specify how to extract data from each of the rows and columns:

- The first column of the table, coloured red, shows the row numbers.
  The first row of the table, also coloured red, shows the column numbers.

- The third row (coloured white on blue) of the table contains one drop−down selection control for each of the columns.
  These controls are used to tell the plugin how to interpret the columns (see below).

- The third column (also coloured white on blue) of the table contains one drop−down selection control for each of the rows.
  These controls are used to tell the plugin how to interpret the rows (see below).

- The fifth row (coloured grey on pink) contains text edit fields with which the *names* of the columns can be specified.

The values in these fields are used to name Measurements and SpotAttributes.

For each row, the interpreting modes are:

Mode	*Frequency*	*Meaning*
Ignore	zero or more	do not load extract any data from this row
Data	zero or more	extract data values from this row
ColumnHeader	zero or one	extract column names from this row
MeasurementAttr	zero or one	extract Measurement Attributes from this row

In this table, *Frequency* refers to how many times each of the interpreting modes can be used within a file, for example, there can be at most one column marked as "ColumnHeader" but any number of columns marked as "MeasurementAttr".

For each column, the interpreting modes are:

Mode	*Frequency*	*Meaning*
Ignore	zero or one	do not load (or merge) this column
SpotName	zero or one	Spot Names are optional, but must be unique if provided
SpotNameAttr	zero or more	an attribute of the corresponding Spot name
ProbeName	zero or one	Probe Names
ProbeNameAttr	zero or more	an attribute of the corresponding Probe name
GeneName	zero or more	Gene Names
GeneNameAttr	zero or more	an attribute of the first Gene name on this row
Data	one or more	Data columns may only contain numbers, blanks or "NaN"s
DataAttrNext	zero or more	an attribute of the next Data column
DataAttrPrev	zero or more	an attribute of the previous Data column

In this table, *Frequency* refers to how many times each of the interpreting modes can be used within a file, for example, there can be at most one column marked as "SpotName" but any number of columns marked as "SpotNameAttr".

If no column is marked as "SpotName" then artifical names will be generated automatically.

The "SpotNameAttr", "ProbeNameAttr" and "GeneNameAttr" modes allow you to add Name Attributes to each of the types of name.

The "DataAttrNext" and "DataAttrPrev" modes allow you to store Spot Attributes along with the Measurement data. The data type of the Spot Attributes will be determined automatically based on the contents of the column.

The **Data Colouring Scheme** is used to highlight the different varieties of data in the table.:

- Rows tagged as **`ColumnHeader`** are shown in green.
- Rows tagged as **`MeasurementAttr`** are shown in purple.
- Rows and columns tagged as**`Ignore`** are shown with a grey background.
- Rows which will be ignored because they start with the **Comment prefix** are shown with a grey background.
- Cells that are blank or contain the specified 'missing value' are shown as white cells with a cross in the middle.

At the bottom of the panel are the **Commmand Buttons**

- The **AutoParse** button makes the plugin scan the data and guess the likely nature of each column. It also copies any names found in the **Col. names in** row.
- The **QuickSet** button opens a dialog box which allows the interpreting mode for several rows or columns to be set in one go. This is handy, if for example, the first 1000 rows of the file should be ignored.
- Once the parsing is satisfactory, the **Import** button starts the actual loading or merging process.

5.5.3 QuickSet

The QuickSet dialog box contains controls which allow the interpreting mode to be set for a sequence of rows or columns. It also provides a "**Save**" button which stores the current row and column settings in a file, and a "**Load**" button which retrieves the settings from a file. The file format for the row and column settings is very simple, and looks like this:

```
Row 1 Ignore
Rows 2 to 7 Data
Row 8 Ignore
Columns 1 to 4 Ignore
Column 5 SpotName
Columns 6 to 9 Data
```

5.5.4 Examples

The first example shows how Name Attributes and Spot Attributes can be extracted:

		1	2	3	4	
		ProbeName	SpotNameAttr	SpotNameAttr	Data	DataAttrI
1	ColumnHeader		ROW	COL	Time_0	Valid?
2	Ignore					
3	Data	hum_xrna_23	1	1	2.19342	Y
4	Data	hum_xdna_23_II	1	2	8.64423	Y
5	Data	hum_xrna_94	1	7	2.6543	N
6	Data	hum_xrna_94	5	2	-1.7352	Y
7	Data	hum_xrna_94 II	5	12	1.0916	Y

- Row 1 contains names that can be used as column headings, so it is tagged as "`ColumnHeaders`".
- Row 2 is blank, so it is tagged as "`Ignore`".
- Rows 3 to 7 contain data values and are therefore tagged as "`Data`".
- Column 1 contains values that can be used as "`ProbeName`".
- Columns 2 and 3 contain the position of the Spot on the array, and so these columns are marked as "`SpotNameAttr`". Although no columns are tagged as "`SpotName`", the "ROW" and "COL" data will be assigned to the artifically generated Spot names.
- Column 4 contains expression data values, so it is tagged as "`Data`".
- Column 5 contains data which should be linked to values in Column 4 as it records whether data in the fourth column is `valid' or not. To achieve this it has been marked as "`DataAttrPrev`".

The second example shows how Measurement Attributes can be extracted:

			1	2	3	4	
			MeasAttrName	GeneName	Data	Data	Data
1		MeasurementAttr	Sample:		RRD/1	RRD/2	RRF/1
2		MeasurementAttr	Time:		0	30	60
3		MeasurementAttr	Temp:		25c	28c	27c
4		Ignore					
5		Data		YAL123C	0.63	1.255	3.543
6		Data		YBR004W	0.76	4.32	3.449
7		Data		YBR014W	1.50	2.93	1.24
8		Data		YBL024C	6.32	5.23	3.991

- Rows 1 to 3 of this file contains data that can be used as Measurement Attributes, in this case values for "Sample", "Time" and "Temperature". These rows are tagged as "`MeasurementAttr`".

- Row 4 is blank, so it is tagged as "`Ignore`".
- Rows 5 to 8 contains the expression data values and are tagged as "`Data`".
- Column 1 contains the names for the Measurement Attributes, so it is tagged as "`MeasAttrName`".

- Column 2 contains values that will be used as "`GeneName`".

- Columns 3 to 5 contains the expression data values, so they are tagged as "`Data`".

5.6 Read Native

5.6.1 Overview

Read Native imports data from a file using **maxdView**'s native file format.

5.6.2 User Interface

The plugin is based around a normal file browser. By default, the browser only displays files ending with `.maxd` or .maxd.gz

Choose a native file and then press "Open" to start the loading process. You can double−click on a name to start the load directly. During the load, a popup dialog shows the progress.

Files in the native file format are generated by the Write Native plugin.

The two buttons under **Load how?** select between replacing the current data with the contents of the file, or merging it. See the Merging data help page more details on how merging is performed.

The buttons under **Load what?** control which parts of the file are processed. Unselected buttons cause *Read Native* to ignore portions of the file, such as the Clusters or the Colours.

See also:

- File Formats
- Write Native plugin.

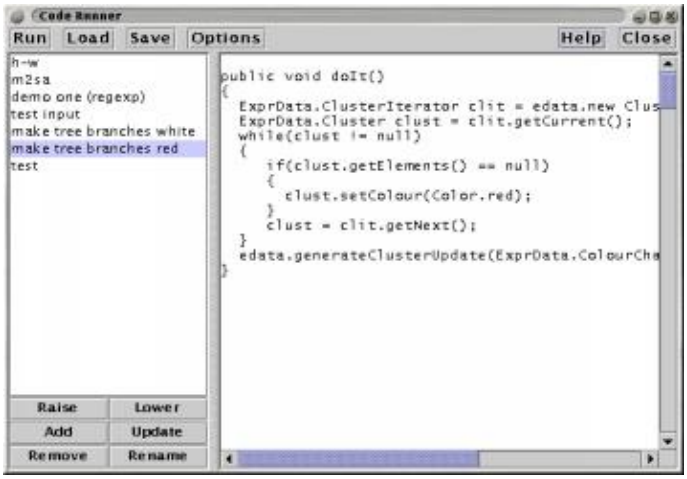
6 Transform Menu

6.1 Code Runner

6.1.1 Overview

This plugin lets you run arbitrary Java code fragments to manipulate data or run comands from plugins.

6.1.2 User Interface



The buttons on the top of the main window are:

Run the current code fragment,
Load a code fragment from a file,
Save the current code fragment to a file,
open the compiler **Options** panel,
display this **Help** information,
Close the window.

6.1.3 Code Library

You can store freqently used code fragments in the *library*.

Press the "Add" button in the bottom left corner and specify a name you want to use for the current code fragment. The library stores the code, and it's compiled counterpart. If the code currently in the edit window is not already compiled you will be asked whether you want it compiled.

A dialog box appears asking for a name for the new library entry. This can be anything you want.

Code stored in the library does not have to be recompiled and can be run directly by double−clicking on it's name in the list.

You can cause code from the library to be executed using the "`runFromLibrary`" plugin command. This command takes a single argument which specifies the name of the library entry to run.

Other buttons under the library list are:

Raise the selected entry up the list
Lower the selected entry down the list
Update (see below)
Remove any selected entries
Rename the selected entry

If you edit the code of a library entry, this change is not put into the library unless you press "Update". If you don't press "Update", the library entry will be unmodified, even if you compile and run the altered code.

The location of the library is set using the "Options" panel. you can have many different libraries in different locations. Each library directory will contain a file called "`CodeRunnerLibrary.dat`" which the plugin uses to locate the code files. Code files stored in this directory are given automatically generated names and should not be renamed.

6.1.4 Method Tree

The Method Tree displays the public methods of all the **maxdView** classes.

The method tree is displayed by selecting using the "Methods" checkbox (near the top−left corner of the panel).

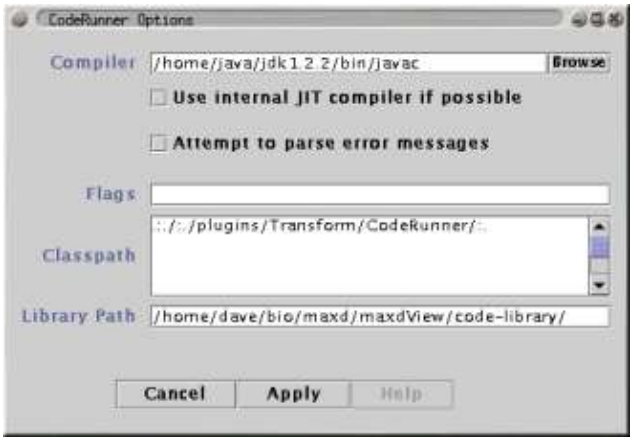
The tree is arranged by the top−level class names. Within each class, the public methods are listed in alphabetical order.

The "Find" feature can be used to locate methods using either their names, return types or arguments lists.

Double−click on a method name in the tree to insert the text into the code fragment.

Double−click on a class or interface name in the tree to expand or collapse that branch.

6.1.5 Compiler Options



Use this panel to control how your code is compiled, including specifying a customised classpath.

"Compiler" should contain the full path to a Java compiler. This progam is usually called `javac` (or `javac.exe` on MS−Windows). If you cannot find the compiler in the same place as the other Java programs on your system then it is likely that you are using a run−time only version of Java. The 'Java Runtime Environment' (JRE) does not come with a compiler. If you have installed a JRE then you need to upgrade to a 'Java Development Kit' (JDK) in order to use this plugin.

You can specify optional compiler "Flags", such as `-O` or `-g`.

The "Classpath" is by default set to the `classpath` used to start **maxdView**. This should suffice for most cases unless you want to reference some external class from within your code.

6.1.6 Example code

(this is not intended to be a complete description of all of things you can do through the APIs, for more details see the Programmers's Guide.)

Clamp all values to +/− 3.0

```
for(int s=0; s < edata.getNumMeasurements(); s++)
  for(int g=0; g < edata.getNumSpots(); g++)
  {
    if((edata.eValue(s, g)) > 3.0)
      edata.setEValue(s, g, 3.0);
    if((edata.eValue(s, g)) < −3.0)
      edata.setEValue(s, g, −3.0);
  }
```

Create a Measurement which is the mean of 2 existing Measurements:

```
double[] tmp = new double[edata.getNumSpots()];

double[] d1 = edata.getMeasurementData("time_1");
double[] d2 = edata.getMeasurementData("time_2");

for(int s=0; s < edata.getNumSpots(); s++)
{
   tmp[s] = (d1[s] + d2[s]) / 2.0;
}
edata.addOrderedMeasurement("Mean_1,2", tmp);
```

Make the values in all Measurement relative to those in the first Measurement:

```
double[] master = edata.getMeasurementData(0);

for(int m=1; m < edata.getNumMeasurements(); m++)
{
  double[] data = edata.getMeasurementData(m);
  for(int s=0; s < edata.getNumSpots(); s++)
  {
    data[s] /= master[s];
  }
  edata.setMeasurementData(m, data);
}
```

Find any duplicated gene names in the data:

```
String dupls = "Duplicates:";
for(int g=0; g < edata.getNumSpots(); g++)
{
  for(int g2=g+1; g2 < edata.getNumSpots(); g2++)
  {
    if(edata.getGeneName(g2).equals(edata.getGeneName(g)))
      dupls += edata.getGeneName(g2));
  }
}
 mview.infoMessage(dupls);
```

Display only clusters with at least 7 elements in them:

```
ExprData.ClusterIterator clit = edata.new ClusterIterator();
ExprData.Cluster clust = clit.getCurrent();
while(clust != null)
{
  clust.setShow( (clust.getSize() > 6) );
  clust = clit.getNext();
}
```

See also:

- Programmers Guide
- Method Reference

6.2 Data Munger

6.2.1 Overview

The *Data Munger* plugin creates new Measurements or Spot Attributes based on an existing data. The values from the existing data are copied to to the Measurement or Spot Attribute for all Spots except those which have been filtered. The values for the filtered Spots are set to a user−specified value.

The purpose of this plugin is to assist in 'tidying up' data, for example, setting the expression value to '0' in all cases where it is 'less than 100'.

Data Munger must be used in conjuction with filters to determine which subset of the Spots should have their values altered.

6.2.2 User Interface

The user interface is divided into three sections:

- **Select the source 'Measurement' or 'Spot Attribute'**

  Use the drop−down list to pick either a Measurement or a SpotAttribute. The SpotAttributes are listed after the Measurement to which they belong.
- **Choose which Spots to affect**

  Select either to alter the values for Spots which are filtered, or for those which are not filtered.
- **Specify how the values are to be changed**

  The values that will be changed can either be set to 'NaN' (the special *Not−a−Number* value which indicates a missing number) or to some numerical value.

6.2.3 Examples:

Create a new Measurement which is the same as Measurement 'A' except that all values which are greater than 1000 will be clamped to 1000

- Set up a filter that selects Spots in which the value in Measurement 'A' are greater than 1000. (The *MathFilter* is ideal for this task).
- Start the *Data Munger* plugin and select 'A' as the source Measurement.
- Select 'Change the value in all unfiltered Spots'. This ensures that the hidden Spots (i.e. those in which the values are 1000 or less) which not be affected.
- Select the 'Change values to' button and enter the value '1000' into the type−in box.
- Press "**Execute**" and provide a name for the new Measurement.

HTML>

6.3 Name Munger

6.3.1 Overview

The *Name Munger* plugin lets you add, replace and translate the different names and attributes that can be associated with each Spot. For most operations, data is loaded from local text files.

It is most useful for adding supplimentary information to Measurements (i.e. annotation of Probes and Genes) or for transforming from one naming or coding scheme to another.

Name Munger can also be used in conjuction with filters to alter subsets of existing names or attributes.

6.3.2 User Interface

The user interface reflects the five operations that *Name Munger* can perform:

- **Copy** one set of names or name attributes to another.
- **Translate** names or attributes using an index file or an 'old name : new name' pair. Can also translate words to upper− or lower−case.
- **Load** names or attributes from an index file.
- **Save** names or attributes as an index file.
- **Clear/Set** mode can set names or name attributes to nulls, blanks, fixed values or values created by a format string.

For all modes, the "Apply filter" checkbox determines whether the current filtering rules (if any) will be applied to restrict the spots processed by the operation.

Copy

This mode is used to move data between names or attributes. If for example you have loaded data from a file as "Probe name"s but later decide the data is really "Gene name"s then the use **Copy** mode to move the names.

Select the source name or attribute in the first dropdown and the destination name or attribute in the second.

You can choose whether null data (i.e. values that are not present) in the source will be copied or ignored. If they are ignored, then the previous data in the destination will be unaffected, otherwise it will be lost.

The "Check" button will tell you have many values will be copied and how many old non−null values will be lost. The "Apply" button

Translate

This mode translates data from one value to another. It is useful for 'decoding' one set of identifiers to another. The translations are specified using a text file in which each line contains the 'old' and 'new' values separated by a TAB character.

Translate mode can also be used to convert Names or Name Attributes to all upper− or lower−case characters.

Example Decoding Probe names:

Assume you have Probe names that use some lab or array specific identifiers and you wish to convert the names to something that can be looked up in a database.

You need to create a text file specifying how one set of identifiers relate to another set, such as this:

```
Qq456.9     EC 102.45.1
Qq382.8     EC 2.55.12
Qq299.7     EC 34.01.18
Qq573.5     EC 102.45.3
```

(in this file, the probe names are on the left of the TAB character and the database identifiers are on the right)

Select "Probe name" in the dropdown list and specify the name of the file in the first text field. You can press "Check" to find out how many names on the file match names in the current data without performing any changes. Press "Apply" to actually do the translation.

If you only want to translate a single value, you can specify a one translate 'from' and 'to' pair of values instead of using a file.

Load

This mode adds data from a file to a name or attribute. The data values are matched based on values in a different name or attribute.

Example To add descriptive information to Gene names:

Assuming you have a text file like this:

```
YGR008c   ATPASE STABILIZING FACTOR 15 KDA PROTEIN.
YHR087w   HYPOTHETICAL 12.0 KDA PROTEIN IN NAM8-GAR1.
YNR001c   CITRATE SYNTHASE, MITOCHONDRIAL PRECURSOR.
....
```

(each line contains a gene name followed by a TAB character followed by some descriptive text for the gene)

Create a new attribute for Gene names (using the Name Tag Editor). Call it something like "DESCRIPTION".

Set the "Load" dropdown to "DESCRIPTION" (or whatever you called the new name attribute) and the "indexed by" dropdown to "Gene name". Use the "file" text box to specify the name of the text file. The "from column" should be set to 2 and the "in column" set to 1 for this example.

Any gene with a name that matches one of the names in the file will have the descriptive text loaded into it's "DESCRIPTION" attribute.

Before doing the load you can press "Check" to find how how many gene names will be matched, and thus how many "DESCRIPTION" attributes will be set. If you are happy with the information press "Apply" to actually perform the load.

Save

Use this mode to write names or attributes to a plain text file. Each line will contain two elements separated by a TAB character. You can choose which names or attributes are used for both of the elements.

Example To save comment information that you have added to Probe names:

If you have created a "COMMENT" attribute for Probe names and added information to one or more Probes (using the Name Tag Editor or the data API) then you can export this information as follows.

Set the "Save" dropdown to "COMMENT" and the "indexed by" dropdown to "Probe name". Choose a file name and press "Apply" to write the file.

Clear/Set

This mode is used to reset the values or a name or attribute to empty or to a constant value.

Use the dropdown menu to select the source name or attribute to modify.

Choose between:

- ***set all values to empty***
  set all values of the specified name or name attribute to the null string.
- ***set all values to blank***
  set all values of the specified name or name attribute to a zero length string.
- ***set all values to***
  use a format string modify all values in the specified name or name attribute.

The **Apply filter** option and one or more filters can be used limit the number of values that will be changed.

Indexing

The term "file indexed by Spot names" refers to a file whose lines each begin with something to be interpreted as a Spot name, followed by a TAB character. For example:

```
spot_023  \t  probe_XYZ_32a
spot_025  \t  probe_XYZ_39a
spot_026  \t  probe_XYZ_41bb
...
```

(\t signifies the ASCII TAB character)

All lines in these "Indexed" files must have two names separated by a TAB character. All other whitespace between the names is ignored. Lines without a TAB character are ignored. Names that are not recognised are silently ignored.

The same rules apply for files indexed by any type of name or attributes.

Format strings

Format strings are used to convert a name or name attributes.

They can be used to encoding commercially sensitive names, for example, or for automatically generating unique IDs for as−yet unknown entities.

Format strings can include the variables:

`$N` ... replaced by the the current name or name attribute
`$I` ... replaced by a unique index, counting from 1.
`$i` ... replaced by a unique index, counting from 0.
`$xI` ...where 'x' is a single digit 0....9,
gives a unique index, padded to x characters, counting from 1.

All other characters in the format string will appear in the converted value. The "`$`" character can be included by specifying it as "`$$`" in the format string.

Example:
The format string "`Probe_$I`" generates a set of values of the form "`Probe_1`", "`Probe_2`", "`Probe_3`". The original names or name attributes are ignored.

Example:

The format string "BLANK" will convert any values into "BLANK".

Example:
The format string "$5i was ($N)" converts as follows:

```
ID_17 --> 00000 (was ID_17)
ID_18 --> 00001 (was ID_18)
ID_23 --> 00002 (was ID_23)
ID_119 --> 00003 (was ID_119)
```

and so on...

6.4 Centering Normaliser

See the Plugin Commands help page for details on the commands offered by this plugin.

6.4.1 Overview

Centering of logged microarray data is one of the simplest normalisation methods and one of the first developed. It is typically applied to a ratio of intensities such as one might obtain from a two colour spotted cDNA microarray experiment. The underlying assumption is that, between the two mRNA populations being compared using the ratios, no overall differential expression has occurred, i.e. on average just as much up regulation has occurred as down regulation. This is equivalent to an assumption that the total amount of mRNA is the same between the two mRNA populations. If these assumptions do not apply to your data, e.g. there is a significant level of overall of differential expression between the two mRNA populations, then another normalisation method may be more appropriate.

If the assumption of no overall differential expression is valid then the average log ratio of mRNA abundances should be zero. The average log ratio of measured (raw, un−normalised) intensities may not be zero. This difference is due to bias, which the normalisation method attempts to eliminate. This is done by subtracting the average log ratio of intensities from all the log ratios, i.e. centering the distribution of log ratios over a mean value of zero.

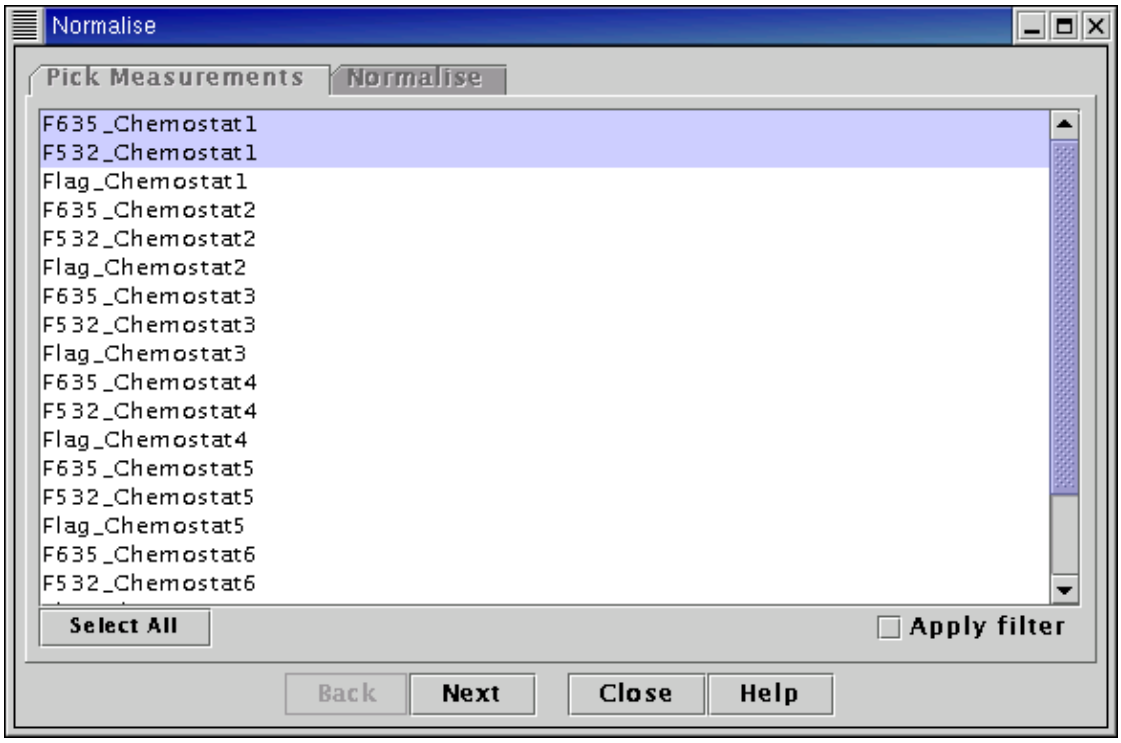
Two different centering methods are provided within MaxD − mean and median centering. Mean centering sets the mean of the logged data to zero. Calculation of the average or mean of a quantity can be significantly affected by large outliers and may be an unreliable estimate of the center of a distribution. The median of data set is a quantity that is less affected by outliers and so is a more *robust* estimate of the center of a distribution. The median centering method sets the median of the logged data to zero. Due to the more robust nature of the median, median centering is the default (recommended) centering method.

Notes

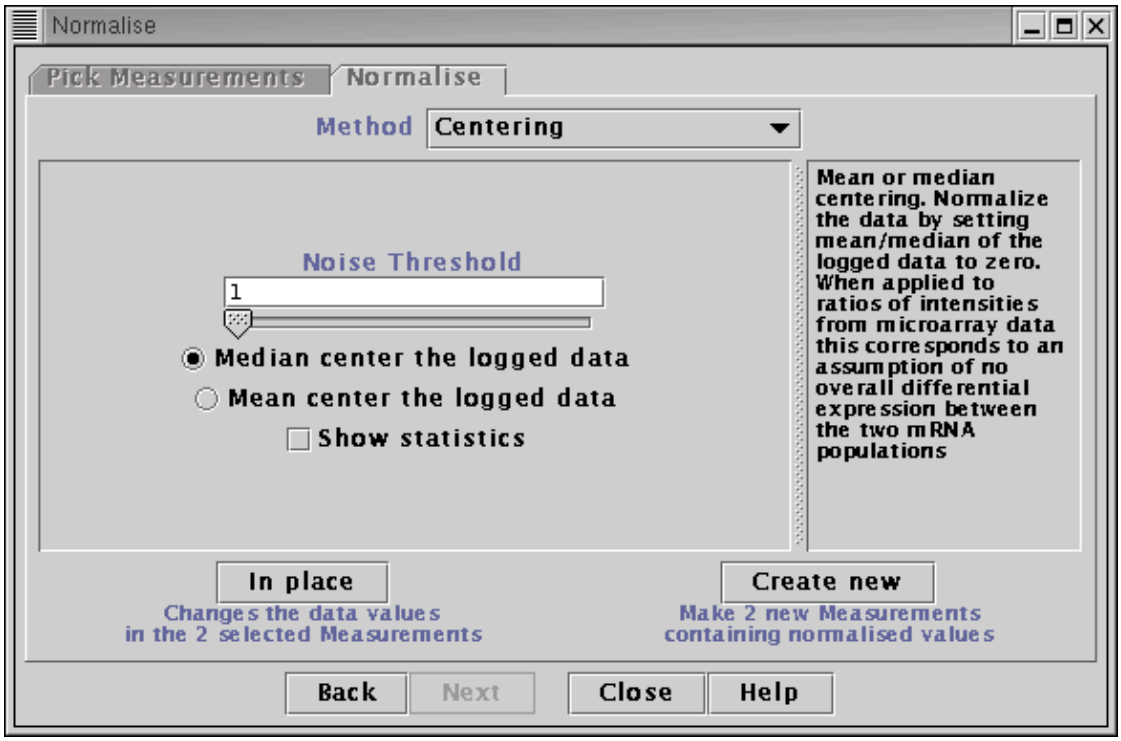
The centering methods center the logged data that is given to them. The logarithms are taken within the normalisation method. If you wish to apply mean/median centering to ratios of intensities, you must calculate the ratios first (e.g. using the "Simple Maths" menu option on the "Transform" menu) before using the normalisation algorithm.

The centering normalisation methods return normalised logged data. All logarithms are natural logarithms, i.e. logarithms to base *e*.

6.4.2 User Interface



Select Measurements to normalise from the list on the left−hand side of the panel.



Choose the centering method (mean or median). Choose threshold value − any raw data values below the threshold will be set to the threshold before taking logarithms. Choose whether to create new Measurements in the main table, or overwrite the Measurements selected with the normalised data.

6.5 Geometric Mean Normaliser

See the Plugin Commands help page for details on the commands offered by this plugin.

6.5.1 Overview

The Geometric mean normalization method corresponds to a location and scale transformation of the distribution of the logged data. The data passed to the algorithm is logged (logarithms to base *e* taken). The mean and standard deviation of the un−normalized logged data are then calculated. We then subtract the mean from the un−normalized logged data and divide by the standard deviation. The normalized logged data will have a mean of zero and a standard deviation of approximately 1. This can be useful when comparing biological data sets whose ranges of expression are expected to be of comparable size. Note that if the Geometric mean normalization method is applied to ratio data (i.e. ratios of intensities) the normalized values **DO NOT** represent normalized log−ratios. Exponentiating the normalized value will not give a fold−change. The normalized values represent values that can be used (with other data sets) to detect differential expression.

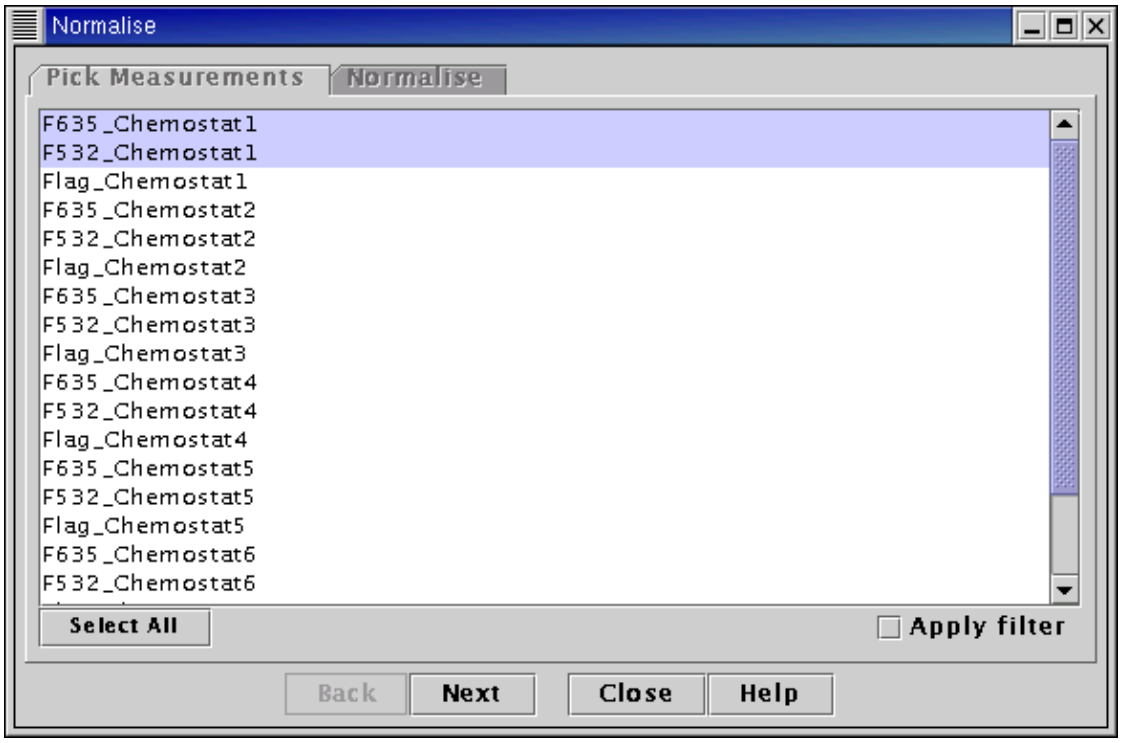
Notes

A trimmed standard deviation is actually calculated, i.e. i) first the standard deviation is calculated is using all the data points ii)the standard deviation is re−calculated from all data points that differ from the mean by +/− 3 times the un−trimmed standard deviation. This reduces the effect large outliers have upon the estimate of the standard deviation.

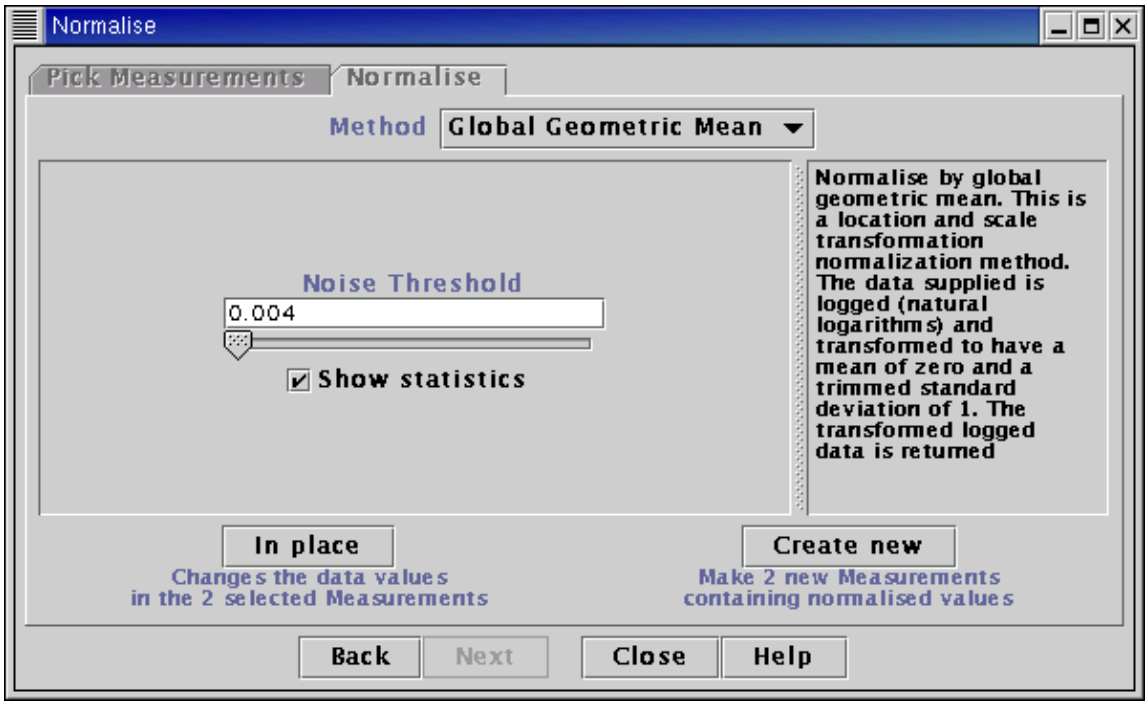
All logarithms used are natural logarithms, i.e. logarithms to base *e*. The normalized values returned are normalized logged data, transformed to a mean of zero and standard deviation of approximately 1.

If you wish to apply the Geometric mean method to ratios of intensities, you must calculate the ratios first (e.g. using the "Simple Maths" menu option on the "Transform" menu) before using the normalization algorithm.

6.5.2 User Interface



Select Measurements to normalize from the list on the left−hand side of the panel.



Set the noise threshold. Un−normalized data values below the noise threshold will be set to the threshold.

If the "show statistics" option is checked, basic summary statistics about the raw, un−normalized data will be displayed.

Choose whether to create new Measurements in the main table, or overwrite the Measurements selected with the normalized data.

6.6 Intensity Dependent Normaliser

See the Plugin Commands help page for details on the commands offered by this plugin.

6.6.1 Overview

Systematic bias in microarray data may be due to many sources. In particular for two colour microarray data (using typically Cy3 and Cy5 fluors) the log ratio values may show a systematic trend that is dependent on the total log intensity. Such systematic trends can detected using "M−A" plots. If for a two colour microarray experiment R and G represent the red and green channel intensities respectively, then M and A are defined as,

M = log(R) − log(G)

A = 0.5 * [log(R) + log(G)]

M is simply the log ratio, i.e. log(R/G), whilst A is the average (over channels) of logged intensities. We would not necessarily expect the log ratio, M, to depend upon the precise value of A. Thus any general trend in a plot of M against A would indicate a systematic intensity dependent bias.

The intensity dependent bias can be removed by first estimating the general trend in the M−A scatter plot and then defining normalised log−ratios as the difference between the raw, un−normalised log ratio and the general trend. The mean normalised log−ratio will be essentially set to zero. If this assumption is not valid you may wish to consider using a different normalisation algorithm.

One of the most popular methods for calculating the general trend in the M−A scatter plot is lowess (Locally Weighted Regression), sometimes also referred to as loess. This calculates the general non−linear trend using a series of short straight line fits to local sections of the data. For more details see Yang *et. al.* (2002) *Nucl. Acid. Res.* **30**:e15

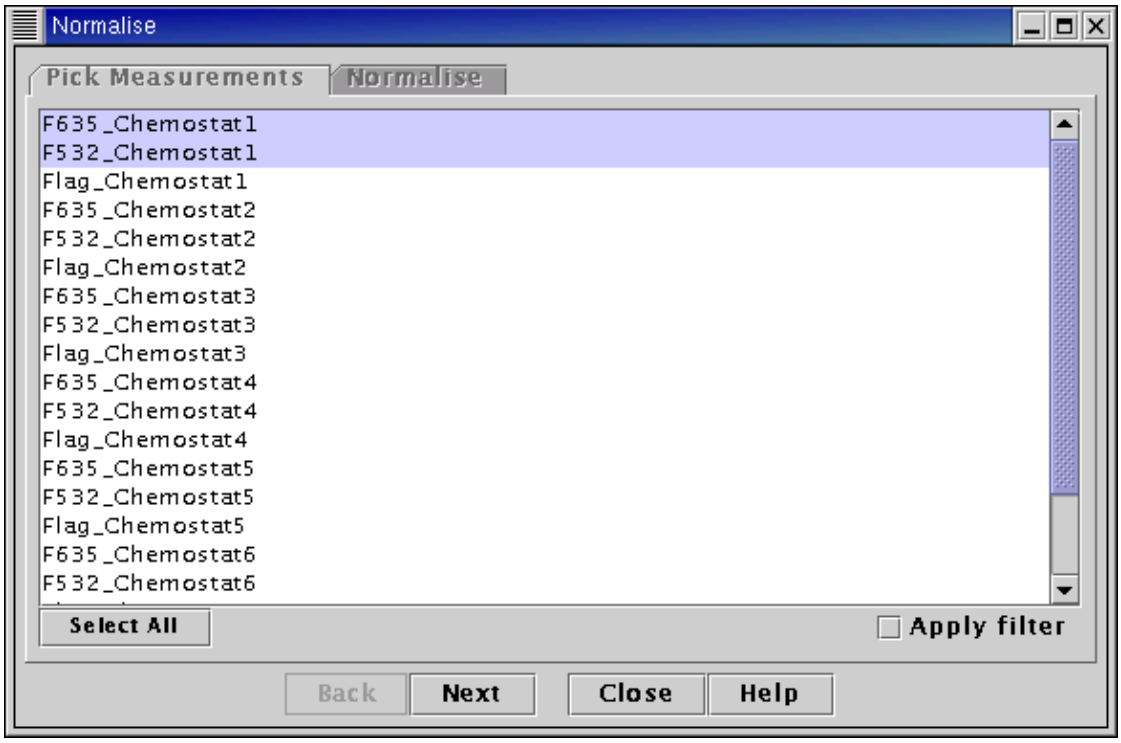
Notes

When working with a large number of spots the lowess calculation can be time consuming. Therefore rather than performing a locally weighted regression at every value of A in the raw data (i.e. for every spot), we perform a locally weighted regression at every centile of A in the data. For spots with values of A between centiles, the general trend in the scatter plot is defined to be on the straight line segment resulting from the locally weighted regression performed at the nearest centile.

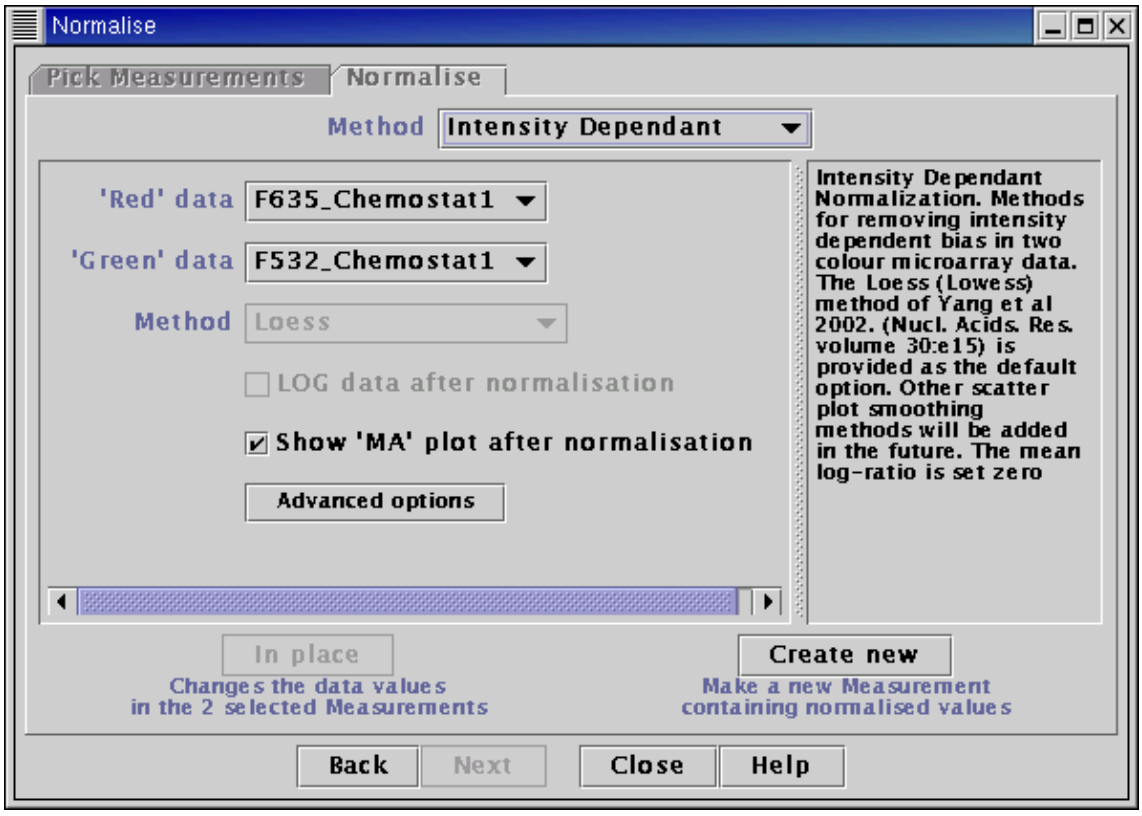
The lowess algorithm currently implemented uses a tri−cube kernel with a span corresponding to 30% of the data. Three iterations of the robust version of the lowess algorithm are performed, with a bi−square kernel on the dependent variable, i.e. M. In future releases it is hoped to make these parameter settings accessible to the user through the "Advanced Options" control panel.

The intensity dependent normalisation methods return normalised log−ratios. All logarithms are natural logarithms, i.e. logarithms to base *e*.

6.6.2 User Interface



Select Measurements to normalise from the list on the left−hand side of the panel.

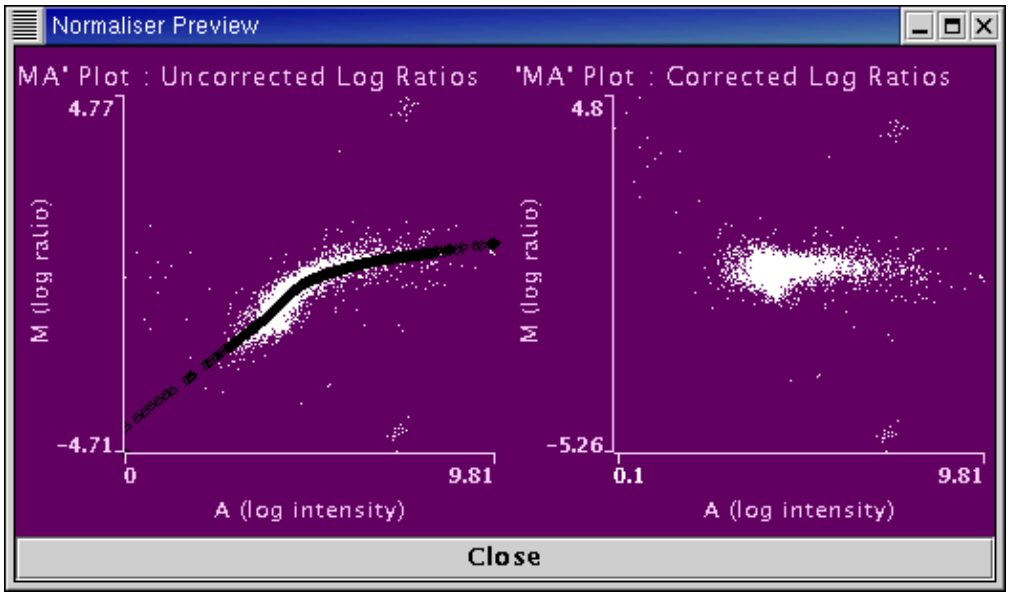


Choose the "red channel" from the list of pre−selected Measurements. Similarly choose the "green channel" data.

Choose the smoothing method. The current default method is "loess".

Choose whether to create new Measurements in the main table, or overwrite the Measurements selected with the normalised data.

If the "Show MA plot after normalisation" option is checked the M−A plots of the raw and normalised data will appear.



The left hand plot shows the un−normalised data, with the black points indicating the lowess smoothed values at each centile. The right hand plot shows the normalised log−ratios.

6.7 Least Squares Normaliser

See the Plugin Commands help page for details on the commands offered by this plugin.

6.7.1 Overview

The Least Squares normalisation algorithm is a location and scale transformation method applied to the logged data, similar in operation to the Geometric Mean normalisation method. It is applied to multiple Measurements. The data passed to the algorithm is logged (logarithms to base *e* taken). Each Measurement is considered to correspond to a linear transformation of a "reference data set". The transformation parameters (slope and intercept) for each Measurement are calculated and the inverse transformation applied to obtain the normalised data. Rather than using a single hybridisation to act as the "reference data set", this is determined collectively from all the Measurements that have been selected for normalisation. This acts as a more robust method than subjectively choosing a single hybridisation to which to compare all others.

The "reference data set" is taken to have a mean of zero and standard deviation of 1. Thus the normalised logged data will have a mean of zero and a standard deviation of approximately 1. This can be useful when comparing biological data sets whose ranges of expression are expected to be of comparable size. Note that if the Least Squares normalisation method is applied to ratio data (i.e. ratios of intensities) the normalised values **DO NOT** represent normalised log−ratios. Exponentiating the normalised value will not give a fold−change. The normalised values represent values that can be used (with other data sets) to detect differential expression.

Notes

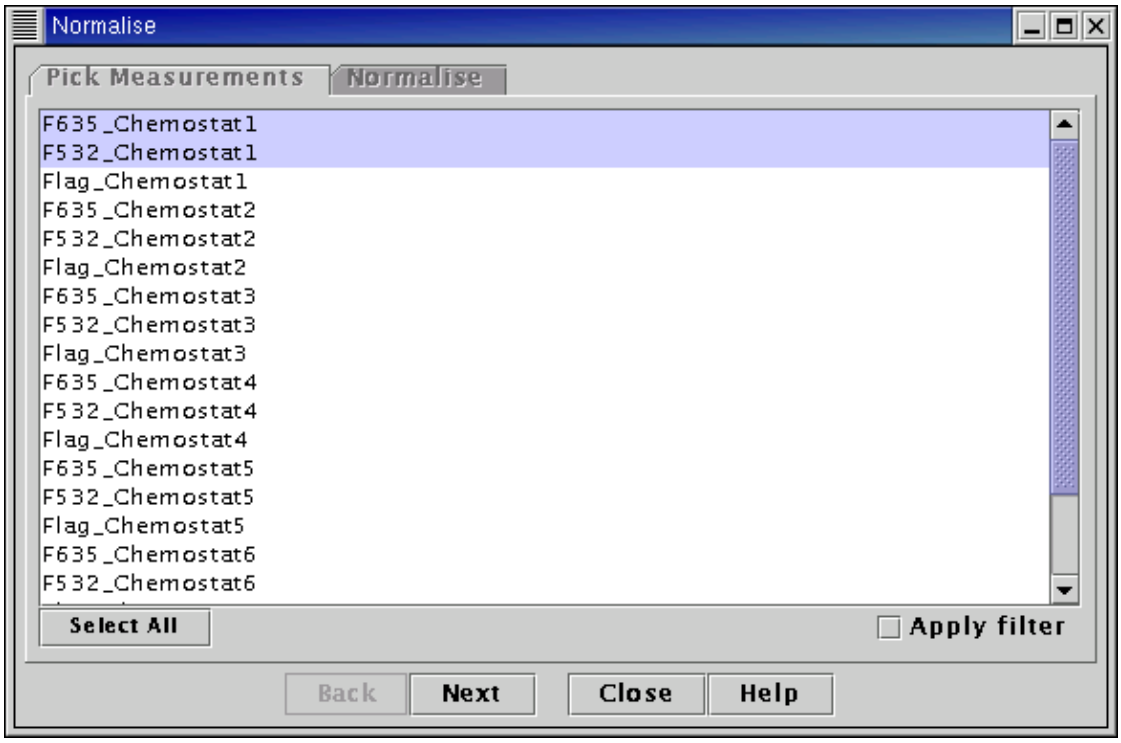
All logarithms used are natural logarithms, i.e. logarithms to base *e*. The normalied values returned are normalised logged data, transformed to a mean of zero and standard deviation of approximately 1.

If you wish to apply the Least Squares method to ratios of intensities, you must calculate the ratios first (e.g. using the "Simple Maths" menu option on the "Transform" menu) before using the normalisation algorithm.

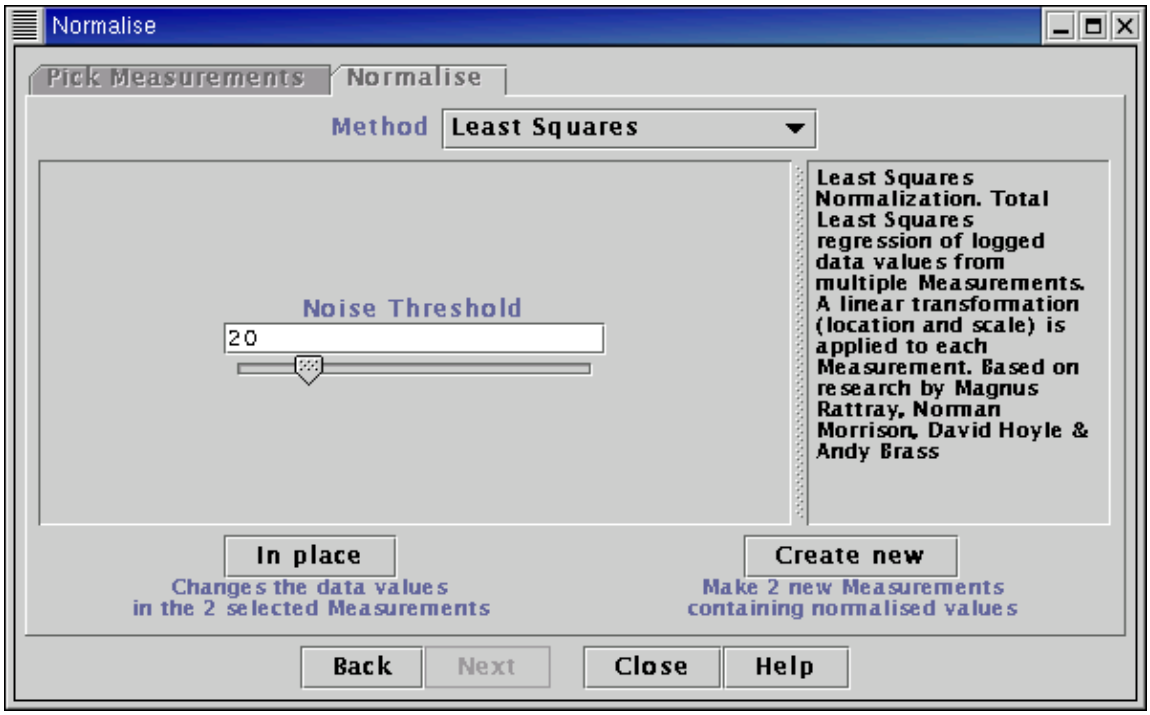
For more details see the technical report DNA microarray normalisation, PCA and a related latent variable model

---

6.7.2 User Interface



Select Measurements to normalise from the list on the left−hand side of the panel.



Set the noise threshold. Un−normalised data values below the noise threshold will be set to the threshold.

Choose whether to create new Measurements in the main table, or overwrite the Measurements selected with the normalised data.

6.8 Normalise

6.8.1 Overview

This plugin performs a normalisation of one or more Measurements.

For documentation on the individual methods that are supported, select the desired method and press the "Help" button again.

6.8.2 User Interface

The interface is divided into two panels which are accessed in sequence:

6.8.3 Pick Measurements

Use the first panel to select one or more Measurements. You can exclude one or more Spots from the normalisation process by trapping them with a filter.

Press the "**Next**" button to move to the next panel.

6.8.4 Normalise

Normalisation can be performed either "**In place**" in which case the values in the existing Measurements are changed, or as "**Create new**" in which case a new Measurement is made for each of the Measurements selected for normalisation.

6.9 Sample Normaliser

This code is provided as an example of how to write a new normliser class that can be used by the *Normaliser* plugin.

6.10 Reorder Measurements

6.10.1 Overview

This plugin lets you change the order in which Measurements are stored. This will change the order in which they are displayed in the main plot window (and in other viewers which consider the ordering).

The order can be changed manually by dragging the Measurements left or right in the display, or it can be automatically computed using the names of the Measurments or the values of one of the Measurement Attributes.

6.10.2 User Interface

Each of the Measurements in the data is represented by a icon in the panel, these icons can be dragged into new positions using the mouse.

The "**Sort by Name**" button arranges in the Measurements in alphabetical order of their names.

The "**Sort by Attribute**" button displays a list of all of the Measurement Attributes in the data and once one is chosen, arranges in the Measurements in alphabetical order of the values of that Attribute.

Once you are happy with the new order, use the 'Apply' button to commit the changes. Use the 'Cancel' button to discard any changes you have made.

6.11 Run External

6.11.1 Overview

An interface to other programs or scripts; `Run External' sends some or all of the data to another program, and then attempts to parse the results produced by that program.

`Run External' can export one or more Measurements, either to the standard input of a program, or to a temporary file which the external program can be told about via a command−line flag.

The output of the program that has been invoked can be grabbed from the program's standard output, or loaded from a temporary file which the program has produced.

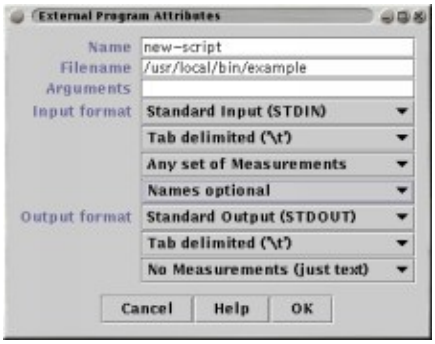
6.11.2 User Interface

Run the selected program

Edit the definition of the selected program

Delete the definition of the selected program

New create a new program definition



Name is the name that will be used in the program list (it does not have to correspond to the actual name of the executable file)

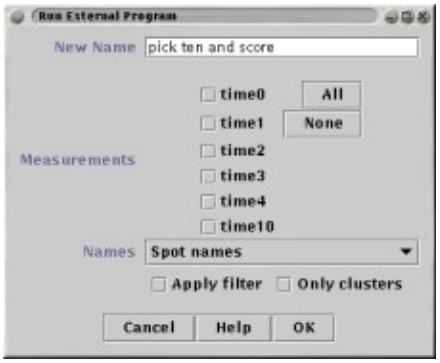
Filename specifies the full path and name of the program that you wish to be invoked

Arguments should contain any command−line parameters you want to be passed to the program as it is run. Special variables representing temporary file names can be used in this field, see below.

Input format comprises four down−down menu selections with which you describe what sort of data the external program is able to accept, and how the communication will occur.

- Choose between STDIN or temporary file mode for input data.
- Choose the delimiter that will be used to separate columns in the data.
- Choose what form of data can be understood by the script, options are 'No measurement', 'One measurement', 'Any set of measurements' and 'Measurements of one type'.
- Choose what (if any) source will be used to name the data rows.

Output format contains three down−down menu selections for describing the output format of the external program, and how the output communication will occur.



6.11.3 Example Definitions

Consider a hypothetical script called "`mean`" which accepts data in tab separated column form, like this: form:

```
21431.31    43.21    54352.42   325432.1
213.2143    3232.5   43242.5    3231.23
5121.23     4123.2   54325.12   34.451
```

and calculates the mean value on each line, producing an output file like this:

```
67531.9
41894.256
42783.435
```

The main thing to notice about this behaviour is that the same number of lines are output as were input, and that no symbolic names are present, only numbers. Any program which behaves like this can be described using the options presented in this example. A more complex example, in which the number of output lines is not the same as then number of input lines is described below.

To interface to this script, the **output format** selections should be "Tab delimited", "One or more Measurements", "No name" and the **input format** selections should be "Tab delimited", "One Measurement".

In the case where the script can operate using standard input and output stream, then no additional description information is required.

If the script does not understand standard input or output, then a temporary file can be used for either input or output (or both). Imagine that the "`mean`" script is normally operated from the command−line like this:

```
host%  ./mean −in my_data.txt −out result.txt
```

`Run External' can save the input data in a temporary file, called for example 'temp1', and then execute the command:

```
host%  ./mean −in temp1 −out temp2
```

and, once the "mean" script has completed, read the output data from the file 'temp2'.

To do this, select 'Temporary file' in both the **output format** and **input format** selections and set the **Arguments** text−field to

```
−in %IN −out %OUT
```

The symbols '%IN' and '%OUT' will be replaced by the temporary filenames that `Run External' generates when you invoke this program. You can see these filenames (and change them) in the dialog box that appears before a program runs (see Part Three above).

(more documentation to follow...)

6.12 SVD

See the Plugin Commands help page for details on the commands offered by this plugin.

6.12.1 Overview

This plugin calculates a Singular Value Decomposition (SVD) of a matrix. SVD is a method of representing the matrix in terms of a set of orthogonal vectors − the eigenvectors, with different weights − the eigenvalues. The larger the eigenvalue the greater the contribution the corresponding eigenvector makes in representing the matrix. A few of the largest eigenvalue, eigenvector pairs can then approximate accurately the original matrix, making SVD a useful method of approximating a data matrix using just a few dimensions.

Formally the SVD of a matrix **A** is defined as $\mathbf{A} = \mathbf{UDV}^T$, where $\mathbf{V}^T$ is the transpose of **V**. The matrix **D** is diagonal, whose elements contain the eigenvalues. The columns of the matrices **U** and **V** correspond to the eigenvectors of **A**.

Given a selected set of Measurements the user has the option of calculating the SVD of the sample covariance matrix formed from the selected Measurements, or the SVD of the Wishart matrix formed from the selected Measurements.

If the selected Measurements form a data matrix **X** then the Wishart matrix is equal to $\mathbf{XX}^T$ (where $\mathbf{X}^T$ is the transpose of **X**).

The sample covariance matrix is equal to $\mathbf{(X - m)(X - m)}^T$ (where **m** is the mean of **X**).

Principal Component Analysis (PCA) is essentially an SVD of the data covariance matrix.

Once the SVD of the chosen matrix has been calculated the eigenvalues are displayed. Also displayed is a plot of the eigenvalues against the expected eigenvalues from a random covariance matrix with the same total variance. This gives a (non−rigorous) means of identifying by eye how many eigenvalues and eigenvectors are significant.

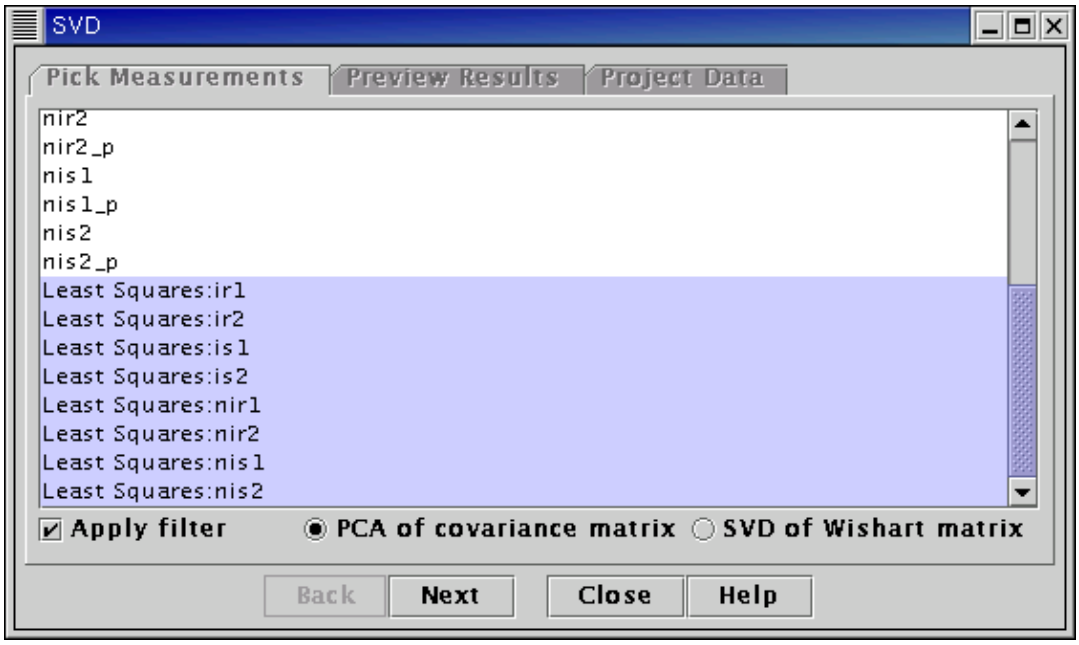
Finally the plugin allows the user to project the original data onto the eigenvectors − either the columns of **U** or the columns of **V**. If "PCA of covariance matrix" was the option originally chosen, it is **X −m** that is projected onto the eigenvectors.

Notes

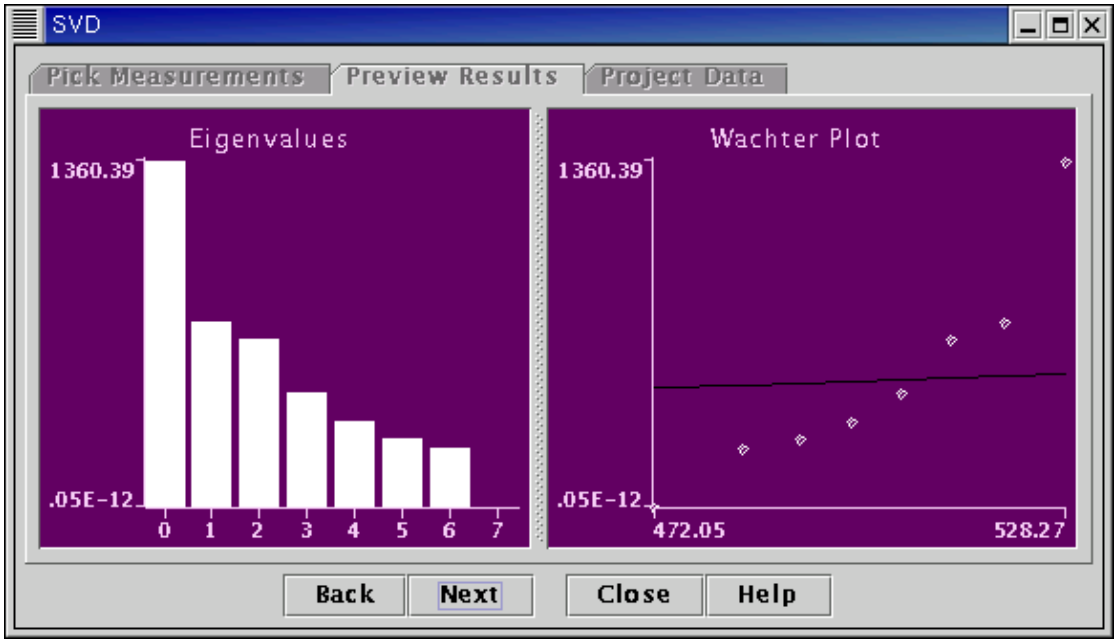
For additional details see,

- *Principal Component Analysis*, I.T. Jolliffe (Springer−Verlag, New York 1986)
- O. Alter, P.O. Brown, D. Botstein, (2000), "Singular value decomposition for genome−wide expression data processing and modelling", *Proc. Natl. Acad. Sci. USA* **97**:10101−10106.

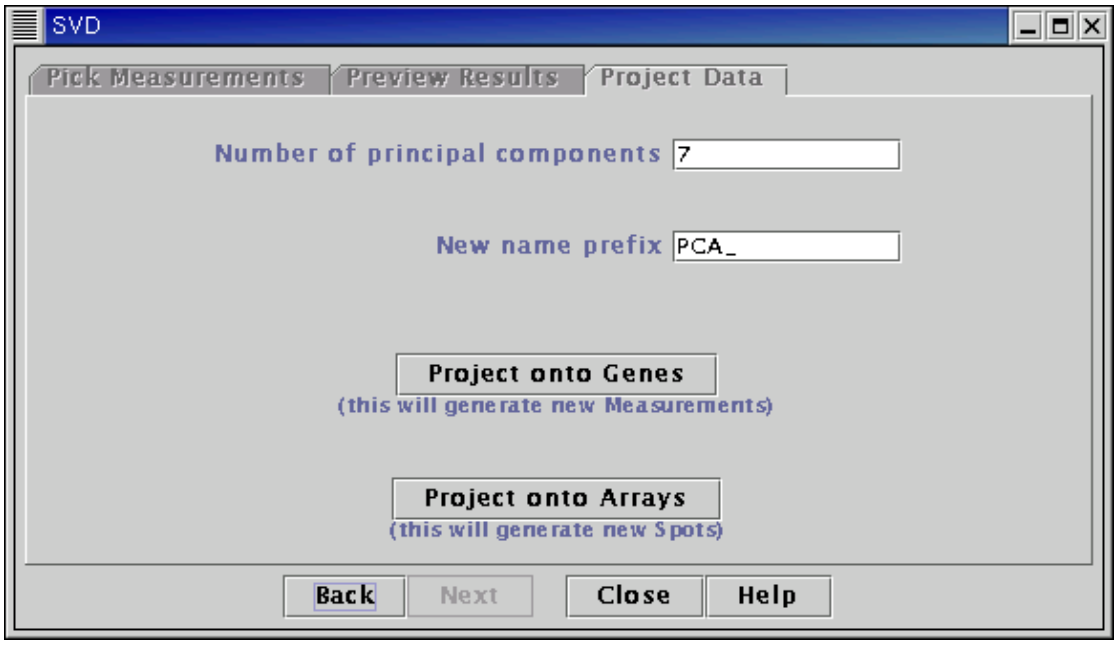
6.12.2 User Interface



Select Measurements to form the data matrix from the list on the left−hand side of the panel. Select the matrix (covariance or Wishart) on which to perform the SVD.



Eigenvalues are displayed in the left hand graph. The right hand graph shows a "Wachter plot" of the eigenvalues. This is a plot of the eigenvalues against the expected eigenvalue, at the same quantile, from a random matrix. The random matrix has independent and identically distributed (*i.i.d.*) elements drawn from a distribution whose variance is equal to the average of the observed eigenvalues. This gives a (non−rigorous) means of assessing whether an observed eigenvalue is the consequence of a genuine dominant direction in the data, or the result of just a chance fluctuation due to the finite number of Measurements in what should otherwise be data that has no dominant directions. The solid black line shows the y = x line. Eigenvalues above this line are usually taken to be of potential interest. For more details see, for example, Johnstone, I.M. (2001), "On the distribution of the largest eigenvalue in Principal Component Analysis", *Annals of Statistics* **29**:295−327.



Choose whether to project onto the eigenvectors corresponding to the columns of **U**, or the eigenvectors corresponding to the columns of **V**.

6.13 Simple Maths

The *Simple Maths* plugin generates derived Measurements.

New data is created using *expressions*, formed in a style similar to rules in a spreadsheet.

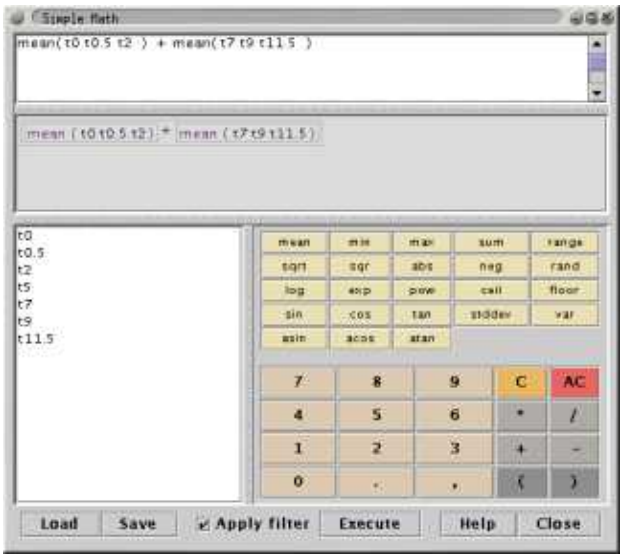
6.13.1 Overview

This plugin is used to derive new Measurements from existing ones. Mathematical expressions such as "`mean( T1, T2, T4 )`" are used to specify how the new data is calculated.

The expression is evaluated on a per−Spot basis. The current filter(s) can be use to select a subset of Spots to process.

A selection of standard arithmetic and statistical operators are provided, for example `mean`, and `log`.

6.13.2 User interface



You can type the expression directly into the top panel, or use the list of Measurements and the calculator style keypad to enter terms.

If the expression can be parsed successfuly, the parse tree will be shown in the middle panel. The order in which the terms will be evaluated is shown by the way the boxes are nested.

Press the "**Execute**" button to give the new Measurement a name and then calculate the values.

The "**Load**" and "**Save**" buttons can be used to write an expression to a file or read an expression from a file respectively. The expression is stored as plain text within the file.

6.13.3 Example expressions

The following examples assume there are Measurements called "`Time0`", "`Time10`" and so on.

- `Time0 + Time10`

  generates a new Measurement in which the value for each Spot is the sum of the values for that Spot in Measurements ***`Time0`*** *and* ***`Time10`***

- `( Time0 + Time10 + Time20 ) / 3`

  generates a new Measurement in which the value for each Spot is the average of the values for that Spot in Measurements ***`Time0`***, ***`Time10`*** *and* ***`Time20`***

- `sqrt( sqr(Time0 − Time10) + sqr(Time20 − Time60 ) )`
- `mean( Time0, Time10, Time20, Time60 )`
- `log( (Time0 + Time10) / 2 ) * log ( (Time20 + Time60 / 2) )`
- `min( max( Time0, Time10 ), max( Time20, Time60 ) )`
- `sum( Time0, Time20, Time60, min ( Time120,  Time180 ) )`
- `sttdev( log(Time0), log(Time20), log(Time60) )`

6.13.4 Available Functions

Name	*Arguments*	*Comment*
abs	1	*absolute value (discard minus sign)*
alog2	1	
alog10	1	
asin	1	
acos	1	
atan	1	
cbrt	1	*cube root*
ceil	1	*round value upwards*
cos	1	
exp	1	*exp(x) raises* ***e*** *to the* ***x****'th power*
floor	1	*round value downwards*
inv	1	*1 / x*
ln	1	*natural log*
log2	1	
log10	1	
max	> 1	*the largest of the arguments, ignores null values*
mean	> 1	*the mean value of the arguments, ignores null values*
min	> 1	*the smallest of the arguments, gnores null values*
neg	1	*negate value (i.e. invert sign)*
pow	2	*pow(x,y) raises* ***x*** *to the* ***y****'th power*
rand	0	*random value in the range 0 ... 1*
range	> 1	*i.e. max()−min(), ignores null values*
round	> 1	*convert to nearest whole number*
sin	1	
sqrt	1	*square root*

sqr	1	*squared*
stddev	1	*standard deviation, ignores null values*
sum	> 1	*ignores null values*
tan	1	
var	> 1	*variance, ignores null values*

6.13.5 Expression Grammar

```
Expression        := Constant | Variable | Function | ComplexExpression

Constant          := Double | Integer

Variable          := MeasurementName | MeasurementSpotAttributeName

Function          := FunctionName "(" [ Expression [ "," ] ]*  ")"

ComplexExpression := Expression  Operator  Expression

Operator          := '+'  |  '−'  |  '*'  |  '/'
```

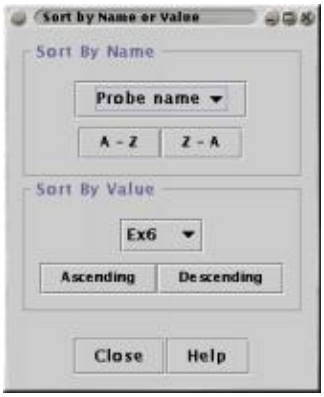
6.14 Sort by Name or Value

6.14.1 Overview

This panel lets you change the order of the rows in the data so that a name column or the numerical values of a Measurment are in ascending or descending order.

The context senstive popup menu in the main display can contain shortcuts for sorting columns.

6.14.2 User Interface



Sort by Name...

The drop down menu allows you to select which Name or Name Attribute or sort on. The "**A − Z**" and "**Z − Z**" buttons sort the Spots into forward or reverse alphabetical order respectively.

Note that blank or missing values are always at the end of the result (i.e. after 'A' or 'Z').

Sort by Value...

The drop down menu allow you to select one Measurement for sorting. The "**Ascending**" and "**Descending**" buttons sort the Spots into numerical order.

6.15 Sort Clusters

6.15.1 Overview

This plugin reorders the rows and/or columns of the data to brings the elements of any visible clusters together.

Clusters which are not currenty visible (i.e. they have been switched off in the Cluster Manager) are not considered in the sorting process.

If each Measurement or Spot in the data is in at most one cluster, then the data will be reordered to exactly match the current cluster hierarchy. If some Measurements or Spots are in more than one cluster, then a secondary algorithm will attempt to produce a ordering which groups things in same clusters together.

6.15.2 User Interface

Select **Sort by Spot clusters** to have the rows reordered, and **Sort by Measurement clusters** to have the columns reordered.

The **Show Progress** option makes the plugin display the partially sorted data from time to time as it the algorithm proceeds.

6.15.3 Plugin Commands

- `sort`

  ```
  boolean : sortSpots
  boolean : sortMeasurements
  ```

- `sortSpots`

- `sortMeasurements`

6.16 Super Grouper

The *Super Grouper* organises Spots into groups based on their similarity of their expression profiles. Hierarchical clustering can then be applied within the groups.

6.16.1 Overview

The basic idea of the *Super Grouper* is to partition spots into groups whereby all Spots in one group have similar expression profiles.

Each group has a "target profile". The expression profile of each Spot is checked against all of the target profiles, and the Spot is put into the group with the most similar target profile.

The target profiles can be adjusted by hand, or the "Auto adjust" can be used feature to iteraratively tweak the profiles.

6.16.2 User interface

The main controls for the *Super Grouper* are in the panel on the left−hand side of the window.

The top portion of this panel contains a list of Measurements. Select two or more Measurements for inclusion in the profile.

The bottom portion of this panel is divided into four sections:

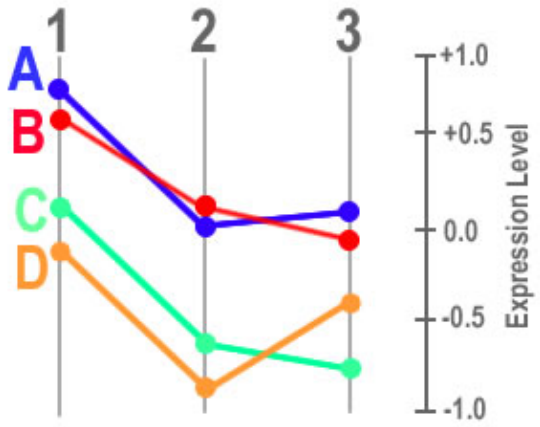
6.16.3 Similarity Metric

The similarity metric is the calculation that is used to quantify the similarity between two profiles. Three different metrics are supported in this plugin:

Distance:	*the sum of the squared differences between values*
Slope:	*the sum of the differences between the first derivatives of the values*
Direction:	*a score generated by comparing direction changes*

The three metrics are illustrated in the following example in which there are 4 spots (A,B,C and D) with values in 3 Measurements (1,2 and 3):

	Meas 1	**Meas 2**	**Meas 3**



Using the **Distance** metric, spots A and B are most similar. This is because the sum of of the absolute differences between corresponding pairs of values is smallest for these two profiles.

Formally:

```
distance_metric(A, B) = abs(A1 − B1) + abs(A2 − B2) + abs(A3 − B3)
distance_metric(A, C) = abs(A1 − C1) + abs(A2 − C2) + abs(A3 − C3)
.....
```

where `abs()` is the *absolute* function: `abs(−X) = X`

With the **Slope** metric, spots A and C are most similar. This metric calculates the sum of the differences of the first derivatives of the values, i.e. the slopes of the two line segments (the one joining Measurements 1 and 2 and the other one joining Measurements 2 and 3). The profile with the smallest sum is regarded as the most similar.

Formally:

```
slope_metric(A, B) = abs( (A2−A1) − (B2−B1) ) + abs( (A3−A2) − (B3−B3) )
slope_metric(A, C) = abs( (A2−A1) − (C2−C1) ) + abs( (A3−A2) − (C3−C3) )
.....
```

This metric therefore chooses lines which have similar (but not neccessarily identical) behaviours in terms of the changes between Measurements

The final metric, **Direction** compares only the direction of the slopes of the lines joining the Measurements. A 'score' is generated by comparing each pair of line segments, if they go in different directions then 1 is added to the score. The most similar profile is the ones with the lowest score. In the above example, spots A and D are the most similar because they both go "down, up" (spots C and B both go "down,down").

6.16.4 Groups

Groups are collections of Spots which have similar profiles. Each group has a 'candidate profile'. Spots are allocated to groups by comparing their profiles with each of the candidate profiles (using the similarity metric) and then choosing the candidate which is most similar.

The buttons in the Groups palette are used to control how many groups are required and to provide initial candidate profiles for these groups.

A specific number of groups can be requested using the type−in field. Enter a number and press "**Set**" to initialise the groups. Any number of groups can be used, although very large numbers will be time and memory consuming.

The "**Use Spot selection**" button initialises the groups using the currently selected Spots. Each Spot in the selection will become the candidate profile for a group. This is useful if you have previously identified some candiate profiles.

The "**Permutations**" button initialises the groups so that all possible permutations of the basic profile shapes are represented. If for example, there are 3 Measurments selected then there are 8 (3 to the power of 3) possible permutations

of the profile, namely: "up,up,up", "up,up,flat", "up,up,down", "up,flat,up", "up,flat,flat" and so on.

The "**Randomise**" button randomly allocates new candicate profiles to each of the current groups.

The "**Remove Empty**" button deletes any groups which currently have no Spot profiles assigned to them.

6.16.5 Options

The "**Apply filter**" option controls whether any currently enabled Filters are used to restrict the number of Spots that will be grouped.

The "**Edit profiles**" option enables interactive editing of the candidate profiles. When this option is active the mouse can be used to drag the individual points on the candidate profiles up and down the axis. After each adjustment, the Spots will be automatically regrouped.

6.16.6 Auto−adjust

Auto−adjust mode is an interative process in which the candiate profiles are repeatedly altered and the Spots re−allocated to groups. A variety of different rules can be employed to control how the profiles are adjusted.

The "**Enable**" switch turns auto−adjust mode on and off. When it is on, a count of how many iterations have been done is displayed next to the switch. Note that in order to improve performance, the display is only updated every two seconds, not every interation.

Two drop−down menus control the main aspects of auto−adjust mode:

1. Which group(s) to adjust:

Smallest:	*adjust the group with the fewest Spots*
Biggest:	*adjust the group with the most Spots*
Best:	*adjust the group with the best fit (the least difference between profiles)*
Worst:	*adjust the group with the worst fit (the biggest difference between profiles)*
Random:	*adjust a randomly chosen group*
All:	*adjust all groups*

2. How the profile(s) in the chosen groups(s) are altered:

Mean:	*adjust the candidate profile towards the mean profiles for all spots in the group*
Mean Spot:	*adjust the candidate profile towards the that of the most average spot in the group*
Random:	*randomly adjust the candidate profile*

The "**Rate**" slider controls the amount by which the candiate profiles are adjusted. Bigger values lead to greater alteration of the profiles.

The "**Noise**" slider allows an amount of random noise to be introduced into the candiate profile of the group(s) chosen for adjustment. Adding noise can be useful as it prevents the allocation of spots to groups from becoming 'stuck' in one particular configuration.

When "**Auto remove empty**" option is selected then groups which become empty during the optimisation are discarded.

6.16.7 Identifying Spots

When the mouse pointer is rested over a Spot profile for a moment, a tool−tip window appears containing an identification label. This label can be any of the Name or Name Attributes in the current data. The choice of which Name or Name Attribute to show is made via the drop−down menu in the control bar along the top of the window.

Click on a profile to add a name label to the plot. Click on the profile again to remove the label. The "**Clear all**" button removes all name labels that have been added to the plot. The "**Label selection**" button adds labels to all Spot profiles which are currently in the Spot selection.

The size of the label font can be adjusted using the "**a+**" and "**a−**" buttons.

Drag−and−drop can be used to locate particular Spots in the plot. Drag one or more Spots from the main display and drop them into the *HyperCube Plot* window. This causes the name(s) to appear next to the profiles on the plot. Dropping a Cluster onto the panel causes the names of all Spots in the cluster to be displayed.

The "**AutoSpace**" option attempts to improve the readability of the graphs by reducing the quantity of displayed labels. When this mode is enabled, labels will only be drawn if there is enough space for them. Note that when there is not a lot of space, or a lot of labels, this mode can lead to some spot labels not being displayed.

Spots which have been selected for labelling can also be displayed in colour to make them more visible. The "**Colour**" checkbox controls this feature.

6.16.8 Make clusters

The groups can be converted into clusters using one of three methods:

One cluster per group
Generates 'flat' clusters, requires little time or memory

Hierarchical, most accurate
Proper clustering (see below), can take a long time

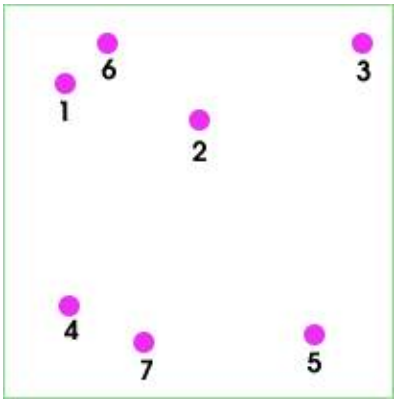
Hierarchical,quickest
Quick−and−dirty clustering, run quickly but uses a lot more memory than the proper mode

Hierarchical,experimental
Nearly proper clustering (see below), but using a heuristic to compute approximate tnearest neighbours. Runs quickly and doesn't require too much memory, but might not generate reproducable results.

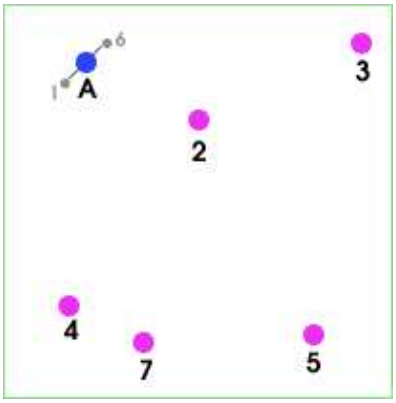
The 'proper' hierarchical clustering algorithm operates by repeatedly finding the two profiles which are closest to one another and then joining them together. The two joined profiles are removed from the group and replaced by their average. This process is repeated until there are no profiles left to join.

As an example, the following images show how the algorithm would proceed in a simple 2D case:

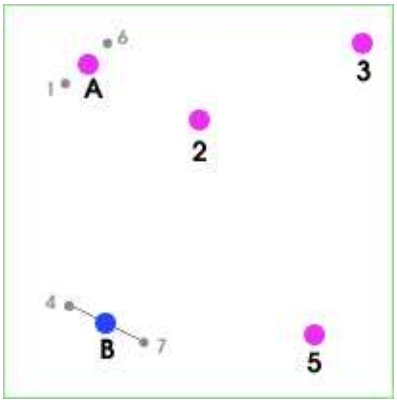
Initial state



Join 1 & 6 to make A



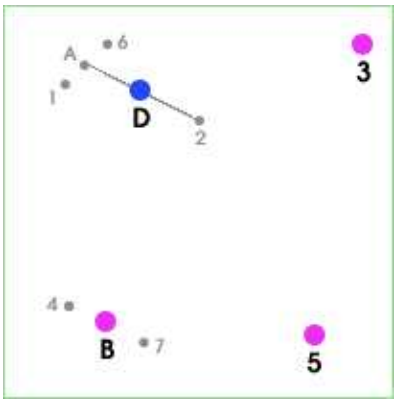
Join 4 & 7 to make B

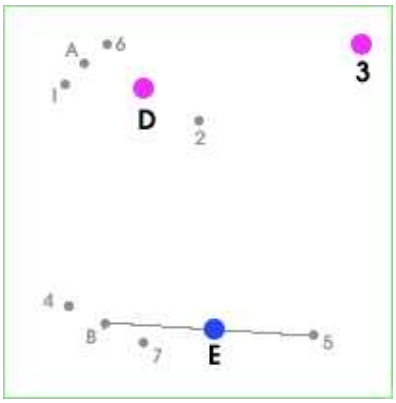


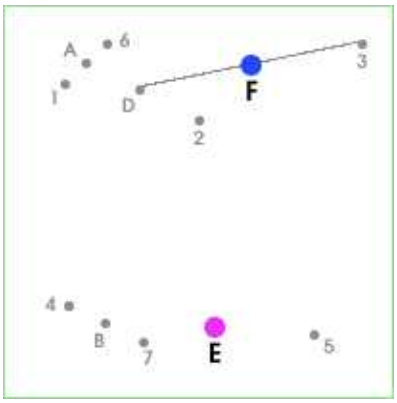
Join A & 2 to make D

Join B & 5 to make E

Join D & 3 to make F

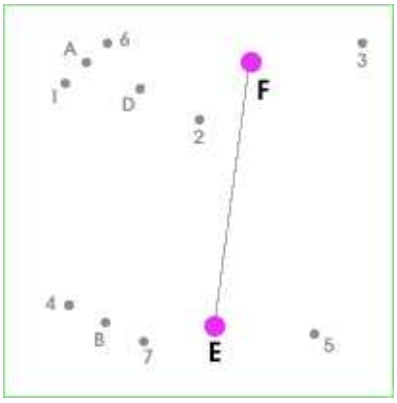


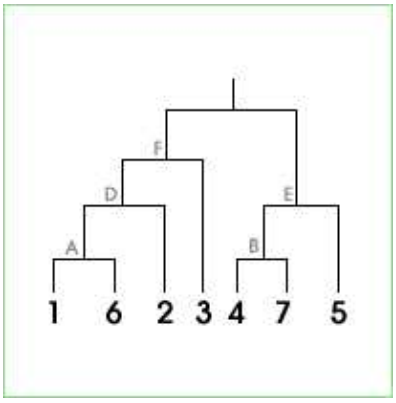




Join E & F, finished

Resulting cluster tree





6.16.9 Examples

There are a lot of parameters in the *Super Grouper*, the best way to find out how they interact is by experimentation.

.....

6.17 Student T Test

6.17.1 Overview

This plugin carries out a standard Student T Test. The T Test is often used to assess the significance of a result, especially when sample sizes are small. The T Test is used to address the following question: Given two sets of measurements, are the means of the sets significantly different? Does the difference in the means of the data reflect a real difference in the means of the underlying distribution, or is the result likely to occur just by chance?
Maxd offers two versions of the TTest. The first is the standard t test, for use when both of the sets considered are assumed to be from normal distributions, and with the added assumption that the two distributions have the same variance.
The second version is for use when the distributions are assumed normal, but the variances are not assumed to be the same.

6.17.2 Interface

To perform the t−test:

- Pick **TTest** from the **transform** menu.
- Choose the measurements for set A and for set B from the two columns. Hold down the **CTRL** key to help select the measurements you require.
- press next.
- On the options tab, select the options you require as follows
- Select the type of T Test you require. The standard t test is for use if you assume that the variances are the same, while the **vars differ** version is for use when you suspect that the variances may differ for some reason. For microarray experiments, the small number of replicates usually mean that it is expedient to assume that the variances are equal.
- select the action that you wish the program to take when some but not all of the measurements for a particular spot are NaN (not a number). This is a case which could occur if there has been a problem on one or two of the replicates for that spot. It could also occur if the user has purposefully set to NaN those measurements which have been flagged as not working. The first option here is "set to NaN", in which case the t test is not carried out for that spot, and the t and p values in the result column will be set to NaN. The second option is "calculate using remaining values." In this case, the program will ignore the NaN values, and will carry out a t test only on the remaining values for that spot.
- Check boxes in order to select which columns you wish to create.
- Change the names of the columns if you wish.
- Press "press t test".

On completing the T Test the program will display an info message confirming which measurements were used in the calculation.
Note: If the "calculate using remaining values" option is chosen, the user is recommended not to rely only on the t value column when interpreting the results. For spots where the t value has been calculated using different numbers of results, the number of degrees of freedom will differ, so the t values will not be directly comparable. The p values are calculated taking into consideration the number of degrees of freedom, so the p values will be directly comparable. The user is thus reccomended to include the p value column and/or the degrees of freedom column in this case
Note: When analysing microarray data, one is usually interested in the 2 tailed p−value. This is the probablility that a difference in means greater than that reported might occur by chance. This includes both cases, meanA>meanB and meanB>meanA. The one tailed p value is used only in the case where our a priori belief is that meanA>meanB.

6.17.3 Formulae and Explanation

The assumptions which are made in order to carry out a t test are:

- That the distributions of values follow a normal (gaussian) distribution
- That the true variances of the two distributions are the same

The formula for the T Test is as follows:

```
t = differenceOfMeans / sterrOfDifferences
```

where

```
differenceOfMeans = mean(A) − mean(B)
```

and

```
sterrOfDiffences = sqrt(alpha * beta)

alpha = (1/n(A)) + 1/n(B))

beta = (var(A)*(n(A)−1) + var(B)*(n(B)−1)) / (n(A) + n(B) −2)
```

recall that:

```
mean(x) = Sum(x)/n(x)

var(x) = Sum((x−mean(x))^2) / (n(x)−1)
```

Note that if both of the samples have the same number of elements, say n, then the formula for the value of t is reduced to:

```
(mean(A) − mean(B)) / sqrt ((var(A) + var(B))/n )
```

In order to interpret the t value, need to look on a statistical table to determine the p−value cutoff which corresponds to the number of degrees of freedom and the required statistical significance.
Note that the number of degrees of freedom is:

```
n(A) + n(B) − 2
```

The above explanation and the t test in maxd are for the case where the variances of the two distributions are assumed to be the same.
One can still apply a significance test if the variances are assumed to be different. In that case, the value calculated is given by:

```
d = mean(A) − mean(B) / sqrt(var(A)/n(A) + var(B)/n(B))
```

the effective number of degrees of freedom in this case is given by

```
degFreedom = (u^2/(n(A)−1) + (1−u)^2/(n(B)−1))^(−1)
```

where

```
u = var(A)/n(A) / (var(A)/n(A) + var(B)/n(B))
```

the significance cutoff can be obtained from relavant tables by considering the nearest integer values to the effective degrees of freedom.
Maxd will calculate the P value for the given t statistic and number of degrees of freedom. A low P value indicates a result which was unlikey to occur just by chance, and can be interpreted as a high probability that means of the underlying distributions differ to the extent reported.

6.18 License and Copyrights

These are the copyright and licence statements for the Colt Distribution, Version 1.0.2. Do not use this software if you do not accept the terms of the licenses described below. In alphabetic order:

`packages cern.colt* , cern.jet*, cern.clhep`

Written by Wolfgang Hoschek. Check the Colt home page for more info.
Copyright © 1999 CERN − European Organization for Nuclear Research.
Permission to use, copy, modify, distribute and sell this software and its documentation for any purpose is hereby granted without fee, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation. CERN makes no representations about the suitability of this software for any purpose. It is provided "as is" without expressed or implied warranty.

`package corejava`

Written by Cay S. Horstmann & Gary Cornell.
Copyright © 1997 Sun Microsystems Inc. All Rights Reserved.
Cay S. Horstmann &Gary Cornell, Core Java Published By Sun Microsystems Press/Prentice−Hall Copyright (C) 1997 Sun Microsystems Inc. All Rights Reserved. Permission to use, copy, modify, and distribute this software and its documentation for NON−COMMERCIAL purposes and without fee is hereby granted provided that this copyright notice appears in all copies. THE AUTHORS AND PUBLISHER MAKE NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF THE SOFTWARE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON−INFRINGEMENT. THE AUTHORS AND PUBLISHER SHALL NOT BE LIABLE FOR ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR DISTRIBUTING THIS SOFTWARE OR ITS DERIVATIVES.

`package com.imsl.math`

Written by Visual Numerics, Inc. Check the Visual Numerics home page for more info.
Copyright (c) 1997 − 1998 by Visual Numerics, Inc. All rights reserved.
Permission to use, copy, modify, and distribute this software is freely granted by Visual Numerics, Inc., provided that the copyright notice above and the following warranty disclaimer are preserved in human readable form. Because this software is licenses free of charge, it is provided "AS IS", with NO WARRANTY. TO THE EXTENT PERMITTED BY LAW, VNI DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ITS PERFORMANCE, MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. VNI WILL NOT BE LIABLE FOR ANY DAMAGES WHATSOEVER ARISING OUT OF THE USE OF OR INABILITY TO USE THIS SOFTWARE, INCLUDING BUT NOT LIMITED TO DIRECT, INDIRECT, SPECIAL, CONSEQUENTIAL, PUNITIVE, AND EXEMPLARY DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

`package edu.cornell.lassp.houle.RngPack`

Written by Paul Houle. Check the RngPack home page for more info.
Copyright © 1997, 1998 honeylocust media systems.
This package is released free under the GNU public license.

`package edu.oswego.cs.dl.util.concurrent`

Written by Doug Lea. Check the util.concurrent home page for more info.
Originally written by Doug Lea and released into the public domain. He says about this package: All classes are released to the public domain and may be used for any purpose whatsoever without permission or acknowledgment.

`packages hep.aida.*`

Written by Pavel Binko, Dino Ferrero Merlino, Wolfgang Hoschek, Tony Johnson, Andreas Pfeiffer, and others. Check the FreeHEP home page for more info.
Permission to use and/or redistribute this work is granted under the terms of the LGPL License, with the exception that any usage related to military applications is expressly forbidden. The software and documentation made available under the terms of this license are provided with no warranty.

`packages jal*`

Written by Matthew Austern and Alexander Stepanov. Check the JAL home page for more info.


```
package ViolinStrings
```

Written by Michael Schmeling. Check the ViolinStrings home page for more info.

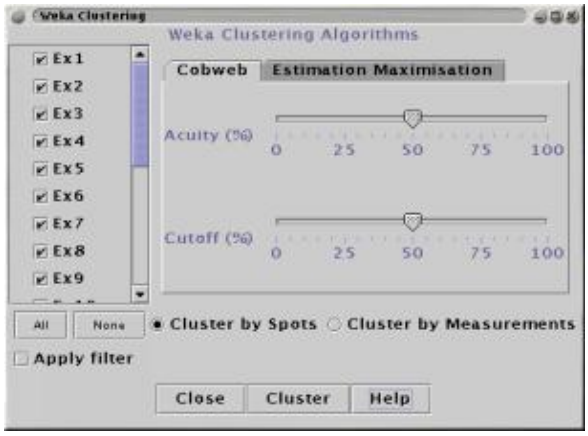
6.19 Weka Clustering

6.19.1 Overview

The Weka machine learning package provides four clustering methods, "Cobweb", "Estimation Maximisation", "Simple K−Means" and "Farthest First". For more details on the algorithms used, see the Weka documentation.

The plugin requires the file "`weka.jar`" which can be downloaded from the Weka web site. The file is distributed in a package called `weka−3−4.jar` (although newer versions made be available). **Unpack this file** as per the supplied instructions. Start the *Weka Clustering* plugin and when it states that it cannot find the Weka classes, pick the "Find" option and use the file chooser to specify the location of the file "`weka.jar`".

Note: These clustering algorithms can be very time consuming for large datasets. Using the Filter(s) to hide uninteresting Spots before clustering can be very beneficial.



6.19.2 User Interface

Clustering can be done on either Spots (i.e. finding genes with similar profiles) or Measurements (i.e. finding experiments or organisms with similar profiles).

Each Measurement is represented with a checkbox. Use these to select which Measurements will be passed to the clustering algorithm.

Apply filter restricts the clustering to the Spots which pass through the current Filter(s).

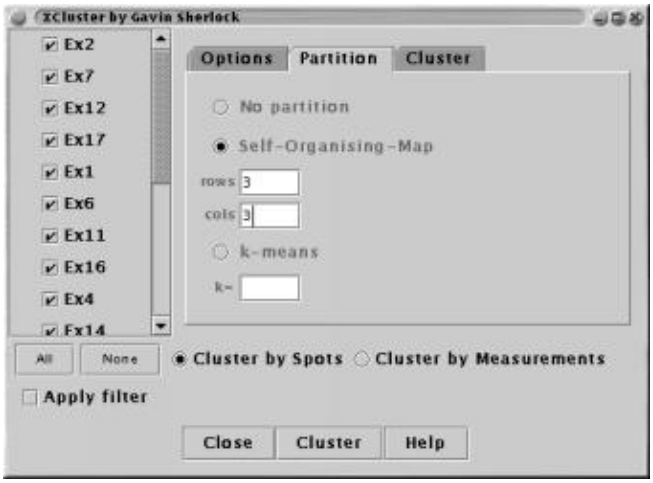
Two tabs contain the controls for the different clustering algorithms. Consult the Weka documentation for full details.

The clustering process takes place in a separate thread. You can continue to use other features of **maxdView** whilst the clustering is under way, but take care not to load new data points until the clustering has finshed.

6.20 XCluster

6.20.1 Overview

This plugin provides an interface to the XCluster software by Gavin Sherlock, for details of the parameters see the XCluster website.



6.20.2 User Interface

Clustering can be done on either Spots (i.e. finding genes with similar profiles) or Measurements (i.e. finding experiments or organisms with similar profiles).

All Measurements is displayed in a list. Use `shift−click' or `ctrl−click' to select one or more Measurements to be passed to the clustering algorithm.

Apply filter restricts the clustering to the Spots which pass through the current Filter(s).

The clustering process takes place in a separate thread. You can continue to use other features of **maxdView** whilst the clustering is under way, but take care not to load new data points until the clustering has finshed.

The controls and parameters are organised in three groups:

Options

This panel is used to specify where the XCluster package has been installed and where temporary files should be stored.

The "**Location of Cluster program**" field should contain the full path to the `cluster' binary, e.g. `/usr/local/apps/xcluster/cluster`.

The "**Directory for temporary files**" field can optionally give the path of a directory in which files can be stored. Tempoprary files are used to communicate data from **maxdView** to `XCluster'. If this field is left blank then the directory where **maxdView** is installed will be used.

The "**Delete after use**" option refers to the temporary files and should normally be switched on (unless you wish to manually inspect the output of the `XCluster' for some reason).

Partition

This panel is used to select which (if any) of two possible partitioning schemes to use whilst clustering.

"**No partition**" results in normal hierarchical clustering.

"**K−means**" attempts to group the data into a fixed number of categories, each of which is organised by a separate hierarchical clustering.

"**Self−Organising−Map**" attempts to group the data into categories as with k−means. Here the categories are further organised into a grid arranged by their similarity with one another.

Cluster

This panel contains options used by XCluster when determining how similar one Spot or Measurement is to another one. The options are explained fully in the documentation that accompanies 'XCluster'.

"**Pearson corelation**"

"**Euclidean distance**"

"**Centered distance**"

"**Log transform data**"

7 Filter Menu

7.1 Filters

A Filter is a rule applied to each of the **Spot**s in turn. **Spot**s which do not obey the rule are trapped by the filter and can be ignored by the rest of **maxdView**. This feature is useful for reducing the complexity of analyses or visualisations by discarding uninteresting **Spot**s. It can also be used to locate **Spot**s which match particular criteria.

Using **Filter**s does not permanently remove **Spot**s, they are only hidden whilst the **Filter** is enabled. Disabling, or altering the filter will cause hidden Spots to reappear.

maxdView allows multiple **Filters** to be active simultaneously. They are applied in turn, in the order in which they were created. If a **Spot** is trapped by *any* of the **Filter**s, it will be ignored.

See also:

- Filter by Name or Value
- RegExp Filter
- Filter By Clusters
- Filter By Selection
- Math Filter
- Profile Filter
- Multi Filter

7.2 Filter By Clusters

7.2.1 Overview

Filter by Clusters chooses spots based on how many clusters they are contained in.

A Cluster is only considered if it is currently visible. Visible means that the clusters "Show" box is ticked in the Cluster Manager, not that the cluster can be currently seen in the display.

At the bottom of the window the percentage of Spots which are trapped by the filter is shown in red and the percentage of Spots which are not trapped is shown in green. Note that the percentages only reflect the filtering that is done by this filter, and do not take into account any filtering being done by other filter plugins that are also active.

As with all filter plugins, closing the plugin's window deactives the filter (but iconifying it does not).

7.2.2 User Interface

There are five modes that the filter can be in:

No filter
: *displays all spots irrespective of whether they are in any clusters*

Hide spots not in any cluster
: *displays only the spots which are in a least one cluster*

Hide spots in one or more clusters
: *displays only the spots which are in a no clusters*

Hide spots in more than one cluster
: *displays only the spots which are in either one or no clusters*

Hide spots in less than two clusters
: *displays only the spots which are in two or more clusters*

7.2.3 Plugin Commands

- `start`

  `string : filter-rule`

- `stop`

7.3 Filter by Name or Value

7.3.1 Overview

This panel lets you selectively hide some of the rows of data by filtering them out. The filter is constructed from a series of rules, such as:

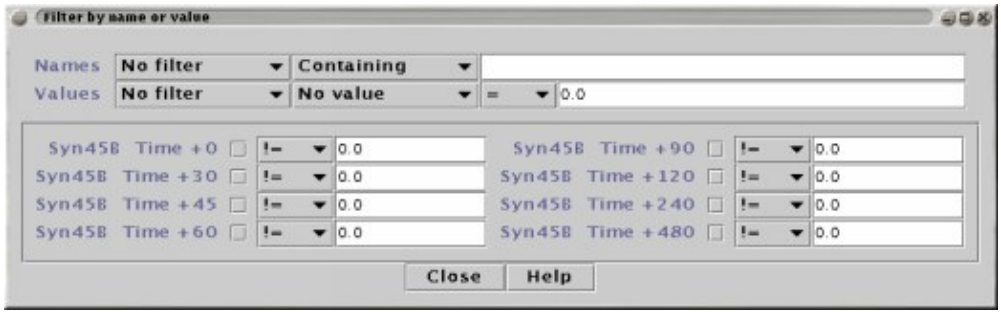
```
Probe Name contains "RNA"
PROB_2 > 0.01
TIME_2 < 3.5
```

There are two global rules, and one optional rule for each Measurement. All of the rules are combined using a *logical AND* operation, this means only genes which obey all of the rules will be displayed.

At the bottom of the window the percentage of Spots which are trapped by the filter is shown in red and the percentage of Spots which are not trapped is shown in green. Note that the percentages only reflect the filtering that is done by this filter, and do not take into account any filtering being done by other filter plugins that are also active.

As with all filter plugins, closing the plugin's window deactives the filter (but iconifying it does not).

7.3.2 User Interface



Name filtering....

The first rule applies the to the names, and can be specify that the name must include some sequence of characters, or must not include some sequence. Select which of the names to use (either Probe, Gene(s) or Spot) and select which matching criterion to use.

As you type characters into the edit field, the filter is continuously updated and spots are removed or displayed based whether whether their name matches the rule.

Value filtering....(global)

The second rule applies to the all of numerical values associated with each spot. These values are grouped based on the type assigned to them in the Measurements panel. You can select between requiring "No value" "Any value" (i.e. 1 or more) or "All values" to conform to the rule.

The *operand* selection allows to you to choose how to compare the numercial values with some constant value that you supply. The options are:

= equal to

!= not equal to
>= greater than or equal to
<= less than or equal to
> greater than
< less than
<> in this range (+/−)
!<> not in this range (+/−)

The edit field to the right of the *operand* selection is where you enter the constant value to compare the numercial values to.

Example:

To exclude any spot with a missing Expression value in any of it's Measurements, set the rule to

```
"Expression" "No values" "=" "NaN"
```

(NaN is the special value used to represent a missing number)

To exclude any spot with a very high value, set the rule to

```
"Expression" "Any value" ">" "100000"
```

Value filtering....(per Measurement rules)

The bottom part of the Filter panel contains controls for filter rules to apply to each of the individual Measurements. Each rule can be enabled or disabled separately using the checkbox next to it's name.

The *operand* selection and edit field for the constant value are the same as for the global value filter. The rule is applied only to this the Measurement, and only if the rule has been enabled.

7.3.3 Missing values (i.e. NaNs)

Note: This behaviour is different to that of versions prior to 1.0.4

All comparisions in which a NaN value is tested against any non−NaN value will be false, for example "`NaN > 3`", "`NaN >= −2000`", "`400 < NaN`" and "`NaN > NaN`" are all false.

NaN values are equal to other NaN values, for example "`NaN >= NaN`", "`NaN <= NaN`" and "`NaN = NaN`" are all true.

Comparison of NaN values with Infinite values are also always false, therefore "`Infinity < NaN`" and "`NaN >= Infinity`" are both false.

See also:

- Apply filter

7.4 Filter By Selection

7.4.1 Overview

Filter by Selection chooses spots based on whether they are in the current *selection*.

At the bottom of the window the percentage of Spots which are trapped by the filter is shown in red and the percentage of Spots which are not trapped is shown in green. Note that the percentages only reflect the filtering that is done by this filter, and do not take into account any filtering being done by other filter plugins that are also active.

As with all filter plugins, closing the plugin's window deactives the filter (but iconifying it does not).

7.4.2 User Interface

The filter can be in one of three modes:

No filter
: *displays all spots irrespective of whether they are selected*

Hide unselected
: *only displays spots that are selected*

Hide selected Spots
: *only displays spots that are not selected*

7.4.3 Plugin Commands

- `start`

  ```
  string : filter-rule
  ```

  'filter−rule' should be one of "`no filter`", "`unselected`" or "`selected`".

- `stop`

7.5 Math Filter

7.5.1 Overview

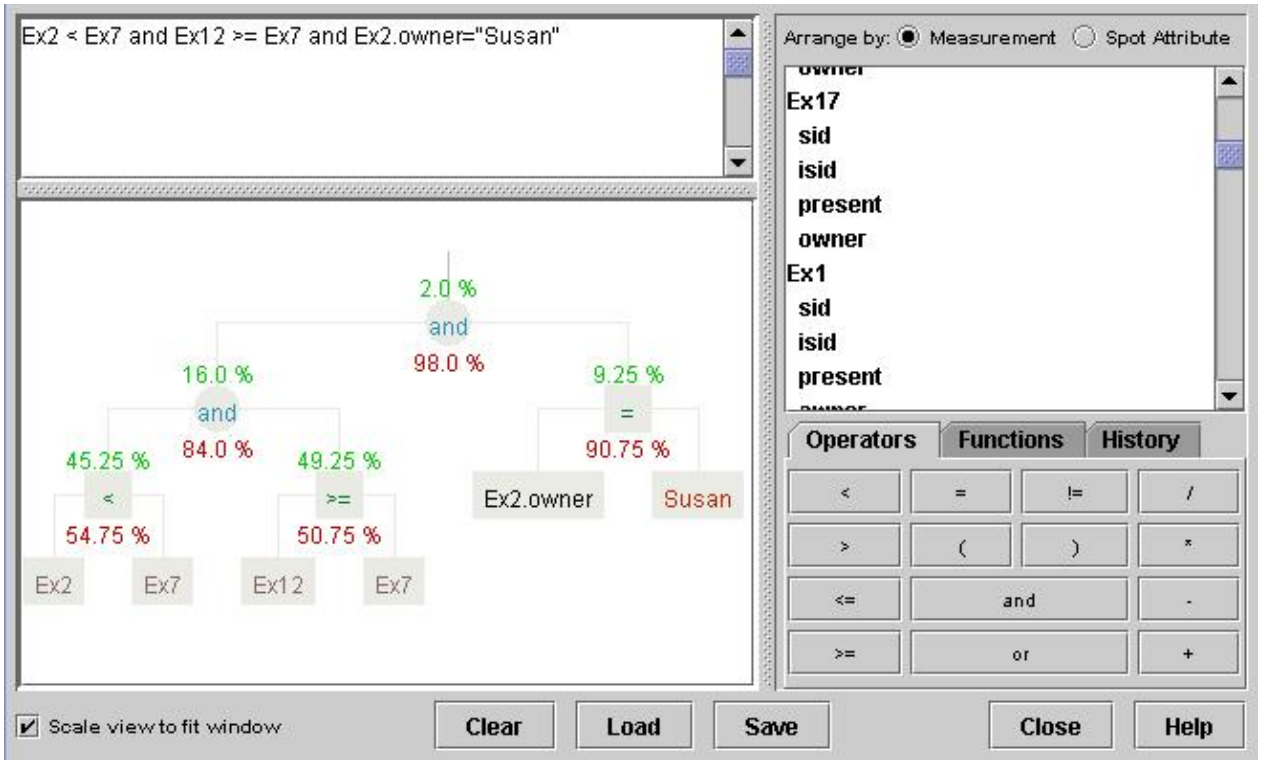
The Math Filter uses mathematical inequalites to filter the spots.

Example filters:
(assuming there are Measurements called M100, M101 and M102).

Filter Rule...	...Displays
`M100 > M101`	spots in which the value in M100 is *greater than* the value in M101.
`M100 > (5 * M101)`	spots in which the value in M100 is *greater than five times* the value in M101.
`M101 > 0 and M102 < 0`	spots in which the value in M101 is positive *and* the value in M102 is negative.
`M101.Source = "Fred"`	spots in which the value of the SpotAttribute "Source" in Measurement M101 is "Fred".

As with all filter plugins, Closing the plugin's window deactives the filter (but iconifying it does not).

7.5.2 User Interface



The interface is divided into four areas:

Top−left A text entry area where the rule is displayed and can be edited.

Bottom−left

	A panel showing a graphical representation of the current filter rule. The graph can be scaled to fit into the available space. The plugin's window can be resized to make more room for the graph.
Top−right	A list of available Measurement names, and the names of any Spot Attributes linked to these Measurements. The list can be arranged either my Measurement (in which case Spot Attributes are listed after each Measurement) or by Spot Attribute (in which case the Spot Attributes are grouped together by name). Clicking on any name in this list will insert the corresponding text into the rule.
Bottom−right	Palettes for entering mathematical operators and functions and a history list in which you can store commonly used filtering rules.

The percentage of spots which are trapped by each term in the rule is continuously displayed. The figure in green above each node shows the percentage of spots that pass through the node. The figure in red (below each node) is the percentage of spots that are trapped by this term of the rule. Note that the percentages only reflect the filtering that is done by this filter, and do not take into account any filtering being done by other filter plugins that are also active.

The **Load** and **Save** buttons allow the rule to be read from and written to a file respectively. The rule is stored in plain text format in the file.

7.5.3 The History List

The current rule can be added to history list using the "**Add current**" button. This rule will then be permanently stored and can be easily retrieved next time the plugin is used. Items can be removed from the history list by selecting them and pressing the "**Remove selected**" button.

7.5.4 Syntax

Math Filter supports the following operators:

and	**or**	(logical operators, either true or false)
<	>	(relational operators, either true or false)
<=	>=	
=	**!=**	
+	−	(arithmetic operators, type dependant)
*	/	

The operator **!=** means 'not−equal−to'.

The symbols **or may be used instead of and and the symbols | or || may be used instead of or.**

The above table is order by decreasing precedence. Expressions are evaluated left−to−right. Parentheses can be used to force precedence, for example:

```
a > 1 &b < 1 | c = 1
```

is, by default, interpreted as:

```
(a > 1 &b < 1) | c = 1
```

but, by the addition of some parentheses, it can be changed to:

```
a > 1 &(b < 1 | c = 1)
```

The graphical representation of the filter rule makes it clear how the rule is being interpreted.

Spot Attributes are referred to as `MeasurementName.SpotAttrName`. The data type of the spot attribute is determined, and the other operand in the term must match this type.

As an example, assume Measurement M102 has two spot attributes, `Cy3 Signal` (an Integer), `Call` (a Char). These attributes are referenced like this:

```
( M102.Call = 'Y' and "M102.Cy3 Signal" < 100 )
```

String constants are enclosed in "double quotes" and Char constants are enclosed in 'single quotes'. Double and Integer numerical values are specified as expected, and interpreted depending on context.

Measurement or Spot Attribute names which contain white space or characters which mean something to the Math Filter (such as < and >) must be enclosed in double quotes, "like this".

7.5.5 Missing values (i.e. NaNs)

Note: This behaviour is different to that of versions prior to 1.0.4

All comparisions in which a NaN value is tested against any non−NaN value will be false, for example "`NaN > 3`", "`NaN >= −2000`", "`400 < NaN`" and "`NaN > NaN`" are all false.

NaN values are equal to other NaN values, for example "`NaN >= NaN`", "`NaN <= NaN`" and "`NaN = NaN`" are all true.

Comparison of NaN values with Infinite values are also always false, therefore "`Infinity < NaN`" and "`NaN >= Infinity`" are both false.

7.5.6 Grammar Rules

A formal description of the grammar, in pseudo−BNF form is:

```
Filter := Operand Operator Operand

Operator := BoolOp | RelOp | MathOp

BoolOp := 'and' | 'or'

RelOp := '<' | '>' | '>=' | '<=' | '=' | '!='

MathOp := '+' | '−' | '*' | '/'

Operand := Constant | Variable| Operand Operator Operand

Constant := DoubleConstant | IntConstant | 'CharConstant' | "TextConstant"

Variable := MeasurementName | MeasurementSpotAttributeName
```

7.5.7 Plugin Commands

- `start`

  `string : filter`

- `stop`

7.6 Multi−Filter

7.6.1 Overview

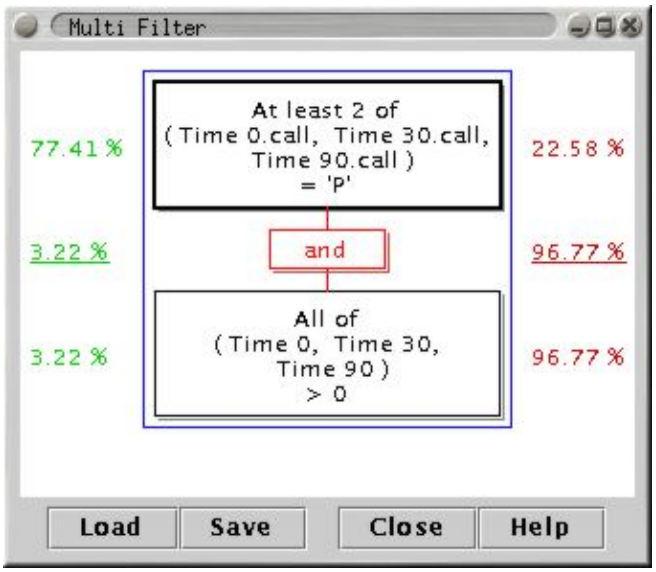
The *Multi−Filter* is a convenient way of filtering based on values in more than one Measurement at a time.

Rules such as "at least 3 of M1, M2, M3 or M4 are greater than 0" can be constructed using this filter.

Individual rules can be combined using the boolean operators 'and' and 'or'.

As with all filter plugins, closing the plugin's window deactives the filter (but iconifying it does not).

7.6.2 User Interface



Rules are combined using the boolean operators "AND" and "OR".

The currently selected rule is shown with a dark border. Click on any rule to select it.

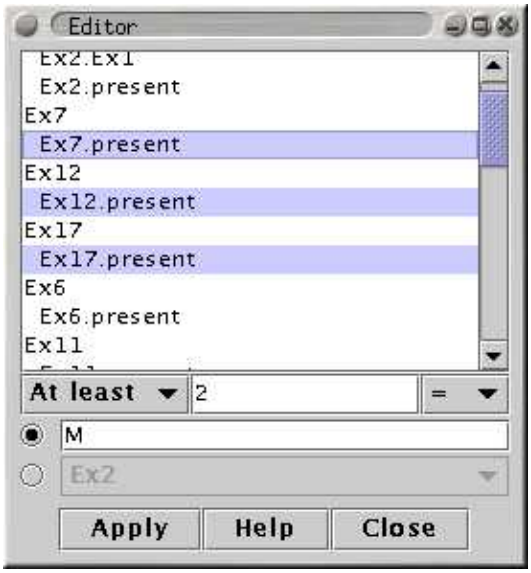
The percentage of Spots which are trapped by each rule is shown in red to the right of the rule. The percentage of Spots which pass through the rule is shown to the left of the rule. In the top level (i.e. outermost) rule, the percentage values are underlined. Note that the percentages only reflect the filtering that is done by this filter, and do not take into account any filtering being done by other filter plugins that are also active.

Use the right mouse button to open a popup−menu which allows you to add and remove rules from the filter.

Expand forwards adds a new rule after the selected rule, and **Expand backwards** add a new rule before the selected one.

The boolean operators used to connect rules are shown in red boxes. Double click on the box to toggle between "AND" and "OR".

Double−click on a rule box (or use the "Edit" menu item) to open the rule editor:



The general form of a rule is

`One_Or_More_Things  Operator  Value`

where

`One_Or_More_Things` is a collection of Measurements and/or SpotAttributes,
`Operator` is one of >, <, = or `!=`
`Value` is a constant (such as 1.5 or 'Q'), a Measurement or a SpotAttribute.

When `One_Or_More_Things` has more than one element, then the following options are available:

```
All
None
At least
At most
```

The "`At least`" and "`At most`" options require an additional numerical value, specified in the type−in field to the right of the options drop−down menu.

Press the "**Apply**" button to update the rule.

Example rules:

```
T1 < T0

( T1 < T0 ) AND ( T2 < T1 )

All of ( T1, T2, T3, T4 ) > T0

At most 2 of ( T2, T3, T7 ) < −0.5
```

7.6.3 Type conversions

When required, type conversions are applied as follows:

	to **Integer**	to **Double**	to **Char**	to **Text**
from **Integer**	−	promote	convert or fail	convert
from **Double**	coerce or fail	−	fail	convert
from **Char**	promote or fail	promote or fail	−	convert
from **Text**	convert or fail	convert or fail	convert or fail	−

promote
: change type with no loss of precision.

coerce
: change type with possible loss of precision.

convert
: convert type with possible loss of meaning.

fail
: causes a type mismatch error which happens whan a value in one type cannot be coverted sensibly to a value in some other type, for example converting the character 'q' to an integer.

7.6.4 Missing values (i.e. NaNs)

Note: This behaviour is different to that of versions prior to 1.0.4

All comparisions in which a NaN value is tested against any non−NaN value will be false, for example "`NaN > 3`", "`NaN >= −2000`", "`400 < NaN`" and "`NaN > NaN`" are all false.

NaN values are equal to other NaN values, for example "`NaN >= NaN`", "`NaN <= NaN`" and "`NaN = NaN`" are all true.

Comparison of NaN values with Infinite values are also always false, therefore "`Infinity < NaN`" and "`NaN >= Infinity`" are both false.

7.7 Profile Filter

7.7.1 Overview

The *Profile Filter* selects spots that have a similar expression profile to some `target' Spot.

An expression profile is the sequence of changes between expression levels of a spot across a series of Measurements. In a time series for example, the profile might be that the expression level rose between 't=0' and 't=1' and then fell from 't=1' to 't=2' and rose from 't=2' to 't=3'.

The *Profile Filter* finds the spots with the most similar profile to the selected `target' Spot. Two comparisons metrics are provided: 'Distance' and 'Direction and Distance'.

As with all filter plugins, closing the plugin's window deactives the filter (but iconifying it does not).

7.7.2 User Interface

On the left of the panel is a list of all Measurements in the current data. You can select two or more of the Measurements for inclusion in the profile.

Note: Selecting more than one item in the list is done using the "SHIFT" and "CTRL" keys in conjunction with mouse clicks. "SHIFT−click" selects everything between the click point and the previous click, and "CTRL−click" toggles the clicked item.

On the right of the panel is a list of all Spots, select one of these to act as the `target' Spot. You can use drag−and−drop to transfer a Spot from another window.

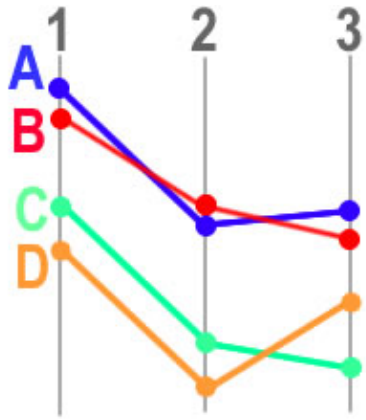
Below the two lists is a text field for selecting how many of the nearest profiles to allow through the filter and a dropdown list for adjusting the similarity metric.

The similarity metric is the calculation that is used to quantify the similarity between two profiles. Three different metrics are supported in this plugin:

Distance:	*the sum of the squared differences between values*
Slope:	*the sum of the differences between the first derivatives of the values*
Direction:	*a score generated by comparing direction changes*

The three metrics are illustrated in the following example in which there are 4 spots (A,B,C and D) with values in 3 Measurements (1,2 and 3):

	Meas 1	**Meas 2**	**Meas 3**



Using the **Distance** metric, spots A and B are most similar. This is because the sum of of the absolute differences between corresponding pairs of values is smallest for these two profiles.

Formally:

```
distance_metric(A, B) = abs(A1 − B1) + abs(A2 − B2) + abs(A3 − B3)
distance_metric(A, C) = abs(A1 − C1) + abs(A2 − C2) + abs(A3 − C3)
.....
```

where `abs()` is the *absolute* function: `abs(−X) = X`

With the **Slope** metric, spots A and C are most similar. This metric calculates the sum of the differences of the first derivatives of the values, i.e. the slopes of the two line segments (the one joining Measurements 1 and 2 and the other one joining Measurements 2 and 3). The profile with the smallest sum is regarded as the most similar.

Formally:

```
slope_metric(A, B) = abs( (A2−A1) − (B2−B1) ) + abs( (A3−A2) − (B3−B3) )
slope_metric(A, C) = abs( (A2−A1) − (C2−C1) ) + abs( (A3−A2) − (C3−C3) )
.....
```

This metric therefore chooses lines which have similar (but not neccessarily identical) behaviours in terms of the changes between Measurements

The final metric, **Direction** compares only the direction of the slopes of the lines joining the Measurements. A 'score' is generated by comparing each pair of line segments, if they go in different directions then 1 is added to the score. The most similar profile is the ones with the lowest score. In the above example, spots A and D are the most similar because they both go "down, up" (spots C and B both go "down,down").

7.7.3 Plugin Commands

- **`start`**
  Arguments **`select`**

    Comment either 'all' or 'list'
    Type string
    Default all

    `measurements`

    Comment used when 'select'='list'
    Type measurement_list
    Default

`target_spot_tag`

Comment which Name or Name Attrbiute
Type string
Default

`target_spot_value`

Comment the value
Type string
Default

`n_spots`

Comment how many nearest matches
Type integer
Default 10

`metric`

Comment one of 'distance', 'slope' or 'direction'
Type string
Default distance

- **`stop`**

7.8 RegExp Filter

7.8.1 Overview

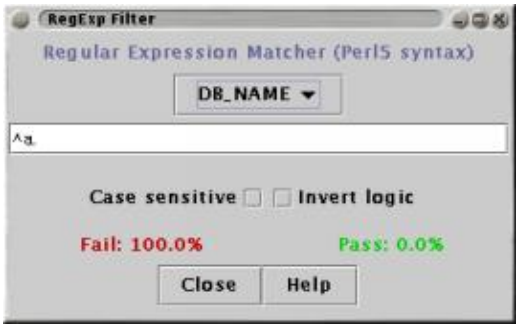
The *RegExp Filter* uses pattern matching (Perl 5 regular expressions) against the text associated with each spot.

The filter can be applied to any combination of the Spot, Probe or Gene names, and to the Annotation associated with either Gene or Probe names.

At the bottom of the window the percentage of Spots which are trapped by the filter is shown in red and the percentage of Spots which are not trapped is shown in green. Note that the percentages only reflect the filtering that is done by this filter, and do not take into account any filtering being done by other filter plugins that are also active.

As with all filter plugins, closing the plugin's window deactives the filter (but iconifying it does not).

7.8.2 User Interface



Enter the regular expression into the type−in field and select which text to match it against using the checkboxes.

By default, matching is not case sensitive, but this can be changed using the "**Case sensitive**" checkbox under the type−in field.

The "**Invert logic**" option swaps the sense of the pattern matching − things that previously matched are not matched and vice versa.

7.8.3 The ORO−Matcher Package

The *RegExp Filter* uses a package called *ORO−Matcher* available from http://www.oroinc.com/. *ORO−Matcher* is an implementation of Perl 5 regular expression matching that is freely available (but comes without source code). A version of this package is included with maxdView, and should be automatically located. If not, use the file browser to navigate to the "`external/OROMatcher−1.1.0/`" directory.

7.8.4 Regular Expression Syntax

(this is not intended to be a complete description of Perl 5 regexp syntax, for more details see any good Perl documentation.)

fish

matches any occurence of the string "fish" (including occurences within other strings such as "oafish").

.ish
matches any sequence composed of any character followed by "ish", such as "fish", "dish", "5ish".

f..t
matches any sequence composed of "f" followed by any two characters followed by "t", such as "foot", "feat", "f01t".

fish|chips
matches any occurence of the string "fish" or the string "chips".

[A−Z]*[0−9]
matches any string composed of 1 or more letters followed by a single digit.

^*hum*
matches any string which *starts with* the 3 letters "hum"

^*12.*rna$*
matches any string which starts with "12" and ends with "rna", for example "1234rna", "1289 human rna" and "12rna".

7.8.5 Plugin Commands

- `set`

```
 string : filter
boolean : case_sensitive
boolean : gene_names
boolean : probe_name
boolean : spot_name
boolean : annotation
```

- `start`

```
 string : filter
boolean : case_sensitive
boolean : gene_names
boolean : probe_name
boolean : spot_name
boolean : annotation
```

- `stop`

8 Display Menu

8.1 Display: Colours

8.1.1 Overview

Define and edit *Colourisers* using this panel. Colourisers are objects which convert a numerical value into a colour for display. Selection of which Colouriser to use for each Measurement is done using the Measurement Manager plugin (or the Measurement name popup menu in the main display window).

Colourisers operate on a *number range*, i.e. numbers from the smallest to the biggest number that this Colouriser is generating colours for. The number range depends on which Measurements are using it. Minimum and maximum values are updated whenever a Colouriser is assigned to a Measurement. There is no method for controlling this range manually.

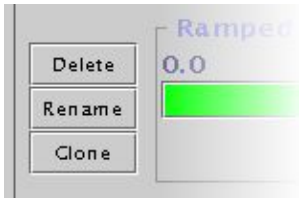
8.1.2 User Interface

The panel displays each of Colourisers that currently exist. The "Add new" button at the bottom of the panel allows the creation a new Colouriser.

There are presently four types of Colouriser:

- **Blender**, which generates colours by interpolating between a set of user defined colours representing different values.
- **Ramped**, which is similar, but allows control of the blending function.
- **Discrete**, which assigns a of a fixed set of colours to each number based on rules, such as ">15".
- **Equalising**, which examines the distribution of the data and allocates a colours to equally partition the values.

To the left of each Colouriser are three buttons:



"**Delete**" removes this colouriser, "**Rename**" opens a dialog box enabling the name of this colouriser to be changed, and "**Clone**" creates a duplicate of this Colouriser.

Blender Colourisers

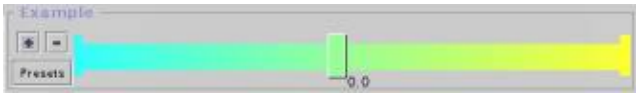
The Blender colouriser consists of two or more colours positioned at fixed points along the number range. The colour for a number which lies between any two points is determined by linear interpolation.

The simplest form of Blender has two colours, one representing the minimum of the the number range and the other representing the maximum:

Colours for numbers are determined by a linear interpolation between the two colours. The colour of the midpoint of the number range will be halfway between the minimum and maximum colours.

The colour at a fixed point can be changed by double clicking on it, which opens a colour chooser dialog box.

A new fixed colour can be added by double clicking on a point somewhere along the colour blend. Alternatively, the "+" button adds a new fixed colour to the blender.



Clicking on a colour point makes it 'selected'. The selected colour point is drawn with a shaded outline. The numerical value associated with the selected colour point is displayed beneath it.

Fixed points, except for the minimum and maximum colours, can deleted by selecting them and then using the "−" button.

The minimum and maximum colours cannot be moved. Other colour points can be moved by dragging them left and right along the colour blend. Colours can be dragged over their neighbours.

Any number of fixed point colours can be added to the colouriser to produce a wide variety of colour schemes.

Ramped Colourisers

Each Ramped Colouriser has two ramps, one for negative values and for positive values.

Both of ramps generates a seqence of colours fading from a start colour to and end colour. The end colour of the negative ramp and the start colour of the positive ramp are shared. Click on the solid colour boxes are either end of a ramp to change the start or end colours.

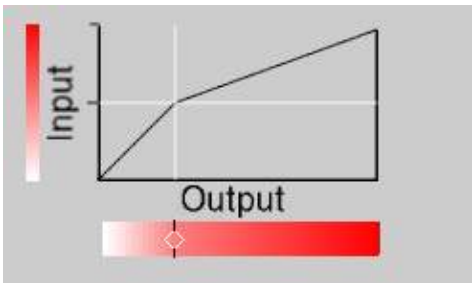
The following examples illustrate how ramps are made. In these examples the colours fade from white to red. The rate at which the fading takes place is determined by the positioning of *nodes*.

Above each ramp are two buttons,  adds a node to the ramp, and  removes the current node from the ramp. The current node is the one represented as a hollow diamond shape, other nodes on the ramp are drawn as filled in diamonds. Click on any node to make it the current node.

The simplest form of ramp is one with no nodes, shown below. The output ramp is a linear fade from white to red.

In the next example, a node has been added, and moved to the left−hand side of the ramp. This causes the fade to happen more quickly at the lower end of the number range.

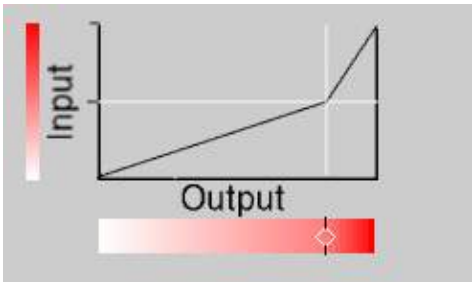
To understand how the fade rate is computed, consider the following graph which plots position in the input ramp against that of the output ramp. The input ramp is always a linear ramp from one colour to another, but, as will seen, the output can be constructed piecewise from a set of linear sub−ramps,



The graph how the output ramp is now composed of two linear sub−ramps. This gives rises to the change in fade rate that occurs at the node position. The input ramp is divided into two equally sized portions, and these portions are then scaled to fit the sub−ramps which form the output.

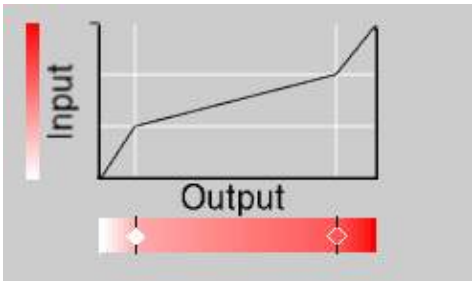
If the node is moved to the right−hand side, the fade happens more quickly at the upper end of the number range, as seen in the next image.

Another graph of input ramp against output ramp shows how the two sub−ramps have been reconfigured.



If a second node is added, the ramp becomes three linear sub−ramps.

The three ramps are arranged as follows:



In general, when `N` nodes are present, the input ramp is divided into `N+1` equally sized chunks and the output ramp is formed by squashing or stretching these chunks according to the node positions.

Discrete Colourisers

The Discrete Colouriser is a collection of colours linked to simple rules, such as "`<−10`". To generate a colour from an input number, the rules are tested in turn from left to right. The colour is chosen from the first rule which is true for this number.

For example, a Discrete Colouriser with three rules would look like this:

The rules, "`>10.0`", "`>0.0`" and "`<0.0`", partition the input numbers into one of three groups. Each rule has a colour, displayed in a panel under the rule. Clicking on this panel opens a colour chooser dialog.

Rules are tested from left−to−right. Once one rule is true, no futher rules are tried. It is therefore important that the rules are ordered correctly. If, for example, the order of the first two rules in the above example is swapped:

then values which are `>10.0` will now be coloured orange. This is because they fulfill the first rule "`>0.0`". No values will be coloured red because the "`>0.0`" rule will always be seen before the "`>10.0`" rule.

Each rule has two buttons underneath it. To insert a new rule, use the "+" button and to remove a rule, use the "−" button.

Equalising Colourisers

The Equalising Colouriser builds a histogram of the values that it is being used for. This infomation is used to partition the range into a fixed number of bands so that each band contains approximately the same number of values. A colour is then allocated to each band.

The histogram data is displayed in the colourisers panel. Under it are bars which show how the bands have been positioned and coloured.

The number of colours is selected by the "Level" control. The lowest level uses only two bands (and therefore two colours). Each higher level uses twice as many bands as the previous level.

The colour for each band is determined by interpolating between the specified start and end colours, selected using the two buttons under the "Level" control.

8.2 Display: Find

8.2.1 Overview

Search for spots or clusters by name.

Enter all or part of a name into a search box and press "Go" (or press return). If the name is found, the main display scrolls to position the relevant spot at the top of the window.

Each subsequent press of "Go" (or return) searches for the next occurrence of the name. A count of the number of matches is displayed after each search. If the name is not found, the message "search finished, not found" is shown in the Find window.

If the spot or cluster being searched for is not visible anywhere (i.e. it has been hidden by a filter) then the message "found, hidden by filter(s)" is displayed.

Drag−and−drop can be used to copy names into any of the search boxes.

8.2.2 User Interface

By default, "hum" will match "human", "Human", "sub−human" etc. Use the following controls to restrict the matching rules:

Exact Match requires the search pattern to be a perfect match, normally substring matching is used.

Case sensitive matching means that the upper− and lower−case version of each letter are considered to be different.

8.3 Display: Layout

8.3.1 Overview

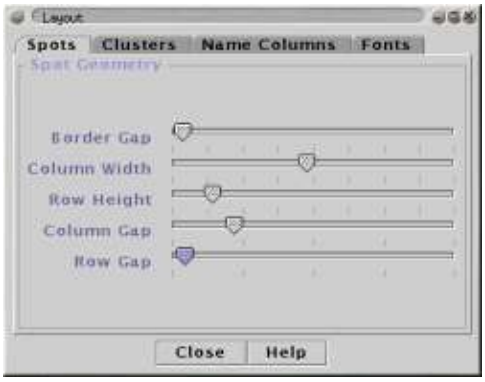
The panel contains controls for the size and layout of spots, clusters and labels in the main viewer.

It is accessed using the "Layout" option in the "Display" menu on the main window.

8.3.2 User Interface

The layout controls are divided between four panels:

Geometry

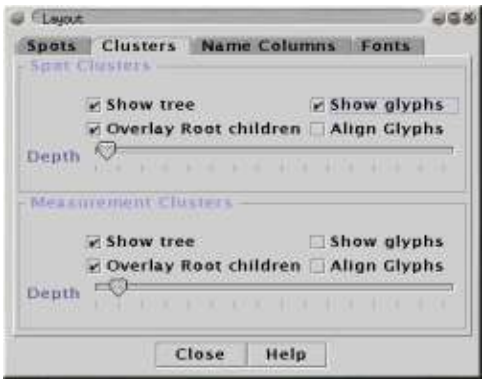


The "**Border**" slider adjusts the amount of blank space left around the border of the display.

The "**Column width**" and "**Row height**" sliders control the size of Measurements (columns) and Spots (rows) in the main display.

The "**Column gap**" and "**Row gap**" sliders adjust the amount of space between columns (Measurements) and rows (Spots).

Clusters



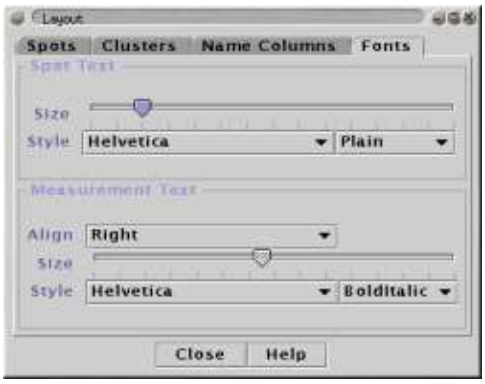
This panel controls how Clusters are displayed in the main window.

Duplicate sets of controls are provided; one for Spot clusters and one for Measurement clusters.

- **Show tree** toggles display of cluster branches
- **Show glyphs** toggles display of cluster glyphs
- **Align glyphs** switches between placing the glyphs at the clusters true depth, or aligned at the deepest depth
- **Overlay Root children** causes the Clusters which are immediate children of the Root cluster to be drawn *on top* of one another. When this option is not selected, the children of Root are drawn one *after* another.

The "**Depth**" slider adjusts the overal depth scale of the cluster tree.

Text Font

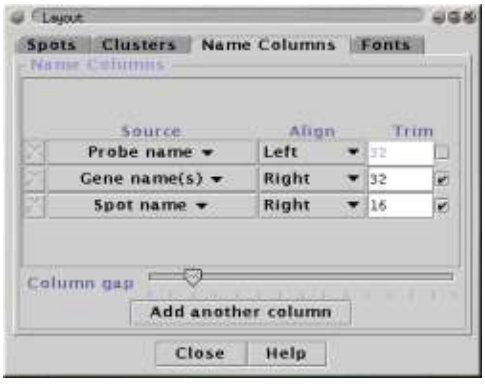


The fonts specified in this panel is used to display labels for Spots and Measurements in the main viewer.

Some viewer plugins use the "Spot label" font in their displays.

The first three font families in the list, "Helvetica", "Courier" and "Times", are 'standard' and can be expected to exist on all platforms. All other fonts in the list are system dependant. If you write a maxd native file on one system and the read it an a different system, system fonts may change.

Name Columns



This panel contains controls for the labels used for the Spot rows. Each column has a "Source", "Align" and "Trim" control.

Columns are removed from the list (and hence the display) using their "X" button. New columns are added using the "Add another column" button.

The "Source" selects which of the **Name**s or **Name Attributes** is displayed in the column. The "Trim" option can be used to automatically shorten labels longer than a specified length. Trimming is only enabled when the tick−box to the right of the trim length is ticked.

The "Column Gap" slider controls the amount of blank space left between name columns in the main display.

8.4 Display: New View

Creates a new instance of the main viewer window displaying the same data as the current viewer. Both views are displaying the same data, if you change it in some way, both views will reflect this change.

The main reason for having two (or more) views open is that each view can be positioned at a different place, and also that each view can independantly choose whether to apply the filter.

8.5 Display: Print

8.5.1 Overview

Prints some or all of the spots as seen in the main display.

The current layout settings (i.e. Row Height, Column Gap, etc) are used when printing. The only difference between the screen version and the printed version of data will be due to differences between size of the window and page.

NOTE: printing is not currently fully implemented. Some features do not work properly yet.

8.5.2 User Interface

Print format selects the destination for the print job:

- The **Printer** option will cause a platform dependant printer selection dialog box to appear to enable you to select a printer and the set page size etc.
- The **Image file** options save the data as either an ASCII or binary `.ppm` image. The binary version of the format produces considerably more compact files than the corresponding ASCII version. The `ppm` format is used because it is very easy to generate. Although the format is not supported by many applications, it can be converted to something more portable using any decent 'paint' program.

Print what selects which Spots will be included:

- **All spots** selects all of the data elements
- **All spots, apply filter** selects all of the data elements that pass the current filter(s).
- **Currently visible spots** *doesn't work properly!*

Print how controls some printer specific options:

- **Single Page** fits all of the data onto a single page
- **Multiple Pages** uses as many pages as neccesary to draw the data
- **White background** overrides the selected background colour
- **Black test** overrides the selected text colour

8.6 Display: Apply Filter

This switch controls whether the currently active filters are applied to the data displayed in this view.

When the switch is off, the filters are not applied, and all data items are displayed.

When the switch is on, the filters are applied in the order in which they were created. Data items will only be displayed if they pass through *all* of the currently active filters.

See also:

- Filters
- Filter by Name or Value
- RegExp Filter
- Filter By Clusters
- Filter By Selection
- Math Filter
- Profile Filter
- Multi Filter

9 Viewer Menu

9.1 App In A Box

9.1.1 Overview

App In A Box lets you run a text mode application in a window.

App In A Box is presently jury−rigged to communicate with the `matlab` package from www.mathworks.com. Use "Send" and "Rec'v" to load and retrieve data from Measurements to and from a named `matlab` matrix.

Remote connection is possible using the `RemoteAppServer` class located in the *App In A Box*'s directory. Run this class on the remote host and connect to it using the "Remote" option on the plugin's control panel.

(more documentation to follow)

9.1.2 User Interface

Use the "App" button to change to a different application.

9.1.3 Plugin Commands

- `start`

```
   file : app
boolean : remote
 string : host
integer : port
```

9.2 Benford Analyser

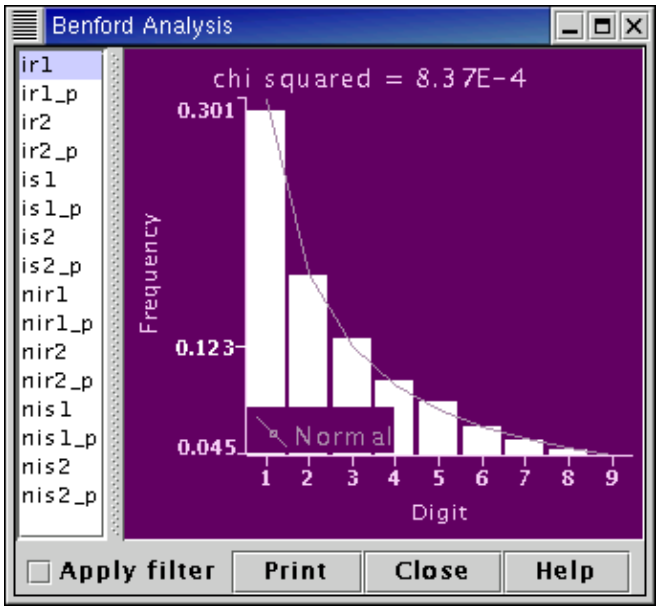
9.2.1 Overview

The distribution of 1st significant digits in many real data sets is not uniform, but follows Benford's law (F. Benford, ,*Proc. Am. Phil. Soc.*, **78** (1938), pp551−572). In base 10 Benford's law says the probability, P(D), of D being the 1st significant digit is,

$$P(D) = \log_{10}(1 + D^{-1})$$

This curious statistical law, actually first noted by Newcomb (S. Newcomb, *Amer. J. Math.*, **4** (1881), pp39−40), says that numbers whose 1st significant digit is 1 occur approximately 30% of the time, whilst those whose 1st significant digit is 9 occur approximately 5% of the time. It has been noted recently (D.C. Hoyle et al, *Bioinformatics*, **18** (2002), to appear) that spot intensities from microarray experiments follow Benford's law very closely. This application plots the frequency, f_D, of 1st significant digits for the selected data set and compares it to Benford's law. The chi−squared statistic indicates the amount of agreement. The smaller the chi−squared value is, the closer the agreement between the observed distribution of 1st significant digits and Benford's law.

9.2.2 User Interface



The frequencies of 1st significant digits are calculated for the Measurement selected by the user from the list of Measurements on the left hand side of the User−Interface. The frequencies are displayed in the graph on the right hand side of the User−Interface. The Benford's law distribution is also shown on the graph as a solid line. The chi−squared value, measuring the degree of agreement between the distribution of 1st significant digits in the selected Measurement and Benford's law, is shown above the graph.

9.2.3 Plugin Commands

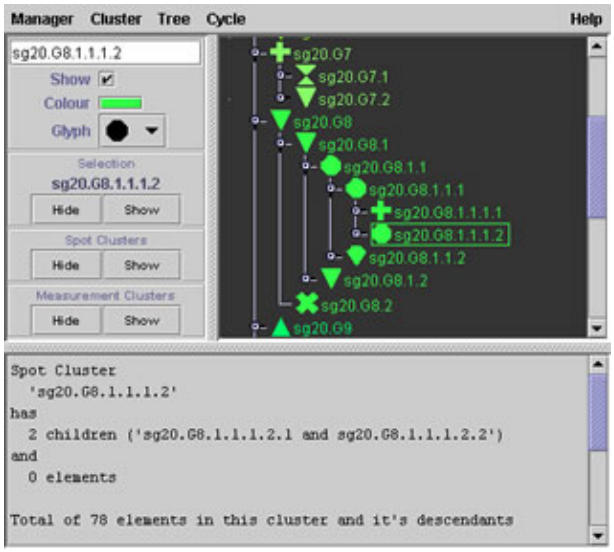
9.3 Cluster Manager

9.3.1 Overview

The Cluster Manager shows the current cluster hierarchy and provides controls for loading, saving, and manipulating clusters.

Drag−and−drop can be used to find clusters in the hierarchy (by dropping a Cluster onto the tree) or to cause clusters to be displayed in viewers (by dragging a Cluster from the tree).

9.3.2 User Interface



The Cluster Manager is divided into three panels which can be resized using the thick inner dividing bars.

The Cluster Tree appears in the top right panel

Cluster Controls for the selected cluster are in the top left panel.

Cluster Infomation for the selected cluster is displayed in the bottom panel.

9.3.3 The Cluster Tree

The top right panel displays the clusters in a tree in which individual branches are expanded or collapsed by clicking on the small switches next to them.

Single clicking on a cluster makes it the *selected* cluster. The selected cluster is name indicated by a box around its name in the tree display.

Double clicking on a cluster with no children toggles the visibilty of the cluster. If the cluster is currently visible, it is hidden and *vice versa*.

Double clicking on a cluster that has children causes the branch to be expanded if it is not already so, otherwise it is collapsed.

You can use Drag−and−drop to take a cluster from this panel to another plugin. If a cluster is dropped on the tree, the tree branches will be exanded as required to reveal the cluster.

9.3.4 Cluster Controls

In the top left panel are controls for manipulating the selected cluster.

- A type−in text field allows to to change the name of the cluster. (You must press RETURN to commit the name change.)
- The "**Show**" checkbox controls whether the cluster is visible or not. If the cluster has children, their visibilty is also controlled by this switch.
- The "**Colour**" and "**Glyph**" controls are used to alter the display attributes of the cluster and its children.

  When the selected cluster has no children, pressing the "**Colour**" button displays a conventional colour picking dialog box. When the cluster has got children, the Colour Selector dialog is used instead.

  The "**Glyph**" dropdown menu selects which of the glyph shapes will be used for the selected cluster. If the cluster has children, then the same glyph will be assigned to all of the children.

  When the selected cluster has children, an extra option is available in the "**Glyph**" menu. Cycle mode (shown as rotating arrows) causes the parent to allocate different glyphs to each of its children.
- The "**Selection**" area shows the name of the currently selected node and contains buttons which "**Hide**" or "**Show**" this cluster.
- The "**All Clusters**" area contains buttons which "**Hide**" or "**Show**" all clusters except the selected cluster.

9.3.5 Cluster Information

The bottom panel displays the names of the elements in the currently selected cluster. This panel also shows how the cluster was defined, whether by Spot names, in which case the cluster is specific to this array type, or by Probe or Gene names, in which case the cluster can be applied to data from other sources.

9.3.6 Menu Commands

Manager menu:

Load reads clusters from a file. The new clusters are added either as children of the *Root* cluster or as children of the selected cluster depending on the "**Insert Where?**" setting.

Files may be in one of three formats:

- **Native** files are in a human readable XML format (see File Formats).
- **Stanford** files are the `.gtr` files used by software from Stanford University.
- **List of Names** files contain a single cluster, specified with one element name per line. You can specify any name or name attribute to match with. If one or more of the elements in the file match existing data then a cluster will be created.

Save writes clusters to a text file.

The "**Which Clusters?**" lets you select between saving all clusters, just those that are visible or just the currently selected cluster.

When "**Include Children?**" is enabled the children of clusters are output, otherwise they are not included.

The clusters can be saved as either the "**Native**" XML format (see File Formats) or as "**List of Names**".

Close shuts the Cluster Manager window (the clusters are not removed)

Cluster menu:

Find searches for a cluster by name

Create constructs a new cluster as one of:

- a empty cluster (which can then be used to group other clusters)
- the spots visible under the current Filter.
- the spots in the current Selection.
- the measurements in the current Selection.
- the union of the elements in currently visible clusters
- the intersection of the elements in currently visible clusters

Delete removes the currently selected cluster (and all of its children)

Re−parent displays a dialog box allowing you to pick a new parent for the currently selected cluster. The cluster will be detatched from its current parent and added as a child of the chosen new parent.

Collapse moves all elements from the children of the selected cluster into the selected cluster itself. The (now empty) children of the selected cluster are then removed.

Tree menu:

Expand fully opens all of the branches in the tree.

Collapse close all of the branches in the tree.

Background allows you to change the colour used by the tree panel to make the cluster colours more easily visible

Cycle menu:

Enable the automatic cycling of cluster visibility using this checkbox.

Descend determines whether the to descend the entire tree or just travel through the children of the selected cluster.

Speed can be set to slow, medium or fast.

The 'cycle' feature is useful when you have lots of clusters on a plot and cannot see the scope of individual features. When enabled, cycle mode toggles each cluster on and off in turn, travelling either sequentially through the children or down the tree from the current position to the bottom. Once all clusters have been displayed, the process begins again at the top of the tree, or first child, and continues until you disable the cycle feature.

9.3.7 The Colour Selector

The *Colour Selector* allows you to apply colour ramps to the children of a cluster. This can make it easier to work out which clusters belong together when viewing cluster overlays.

There are two modes of operation; fixed and blend. In fixed mode the cluster and all of its children are set to the same colour. In blend mode, two colours are chosen and a range of colours is blended from them. This colour range is then distributed to the children of the cluster.

In fixed mode a single colour button is displayed, whereas in blend mode a `from' colour and a `to' colour are displayed. These are the colours that will be used to make the blend colours. Click on a colour button to display a colour picker.

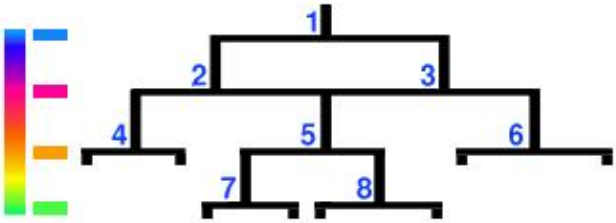
The full range of options are only available when "**Apply to children**" is selected. When "**Apply to children**" is not selected, only the "**Fixed**" mode is available.

Blend mode is available in two flavours:

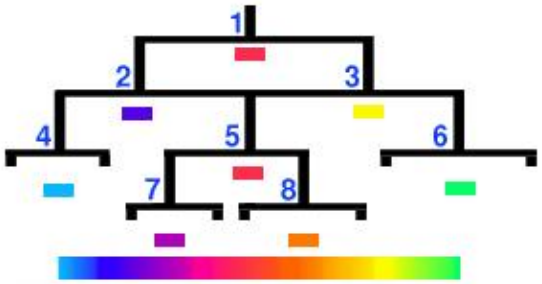
- "**Blend HSB**" generates a range of colours by interpolating the "hue" spectrum. This is the rainbow−like spectrum of colours often seen in computer graphics applications. The HSB colour space can be explored using the "HSB" panel on the standard Java colour picker that appears when you select either blend colour.
- "**Blend RGB**" generates a range of colours by interpolating red, green and blue values directly.

The blended colour range is assigned to the children of a cluster in one of two ways:

- "**Colour by depth**" assigns colours to the children based on how deep they are in the tree. All clusters at the same depth will be coloured the same.



  In this example the cluster tree has a depth of 4. Colours are blended at 4 equidistant points in the colour range and assigned to the children in depth order. Cluster `1' is at the top and will be coloured blue. Clusters `2' and `3' are at depth 1 so they are coloured pink. Clusters `4', `5' and `6' are coloured orange because they are at a depth of 3. Finally, clusters `7' and `8' are the deepest and will be coloured green.
- "**Colour by breadth**" recursively splits the range into pieces for each child such that all the of leaf nodes get a unique colour.



  This example uses the same cluster tree and colour range. The colours are assigned to clusters based on where they lie when the cluter tree is laid out along the colour range. Cluster `1' is in the center and will get a colour from the middle of the range. Cluster '4' is at the far left and is coloured blue and cluster '6' at the far right is coloured green. Other clusters are coloured based on their position in the colour range as indicated. Note that some branch nodes can be coloured that same (e.g. clusters '1' and '5') , but that no leaf nodes can share the same colour.

See also:

- Tutorial: Working with Clusters
- File Formats

9.4 Compare Clusters

Overview

The *Compare Clusters* plugin compares the contents of two clusters and reports how where they match and where they differ. It is useful for examining the differences between different clustering algorithms and/or parameters.

The Algorithm

The algorithm `compare cluster A with cluster B` proceeds as follows:

```
foreach cluster A' in A
  find smallest cluster B' in B where A' is a subset of B'
  find B'', the smallest subset of B' where A' is a subset of B''
  calculate coverage metric for A' over B''
```

The `coverage metric` is a measure of how many of the elements in one set of clusters are found in another set of clusters. If the two sets both contain exactly the same spots and no others then the coverage value is 100%.

```
coverage_metric (A over B) = ( intersect_AB_count / B_count )

where
 intersect_AB_count = number of elements in the intersection of A and B
 B_count = number of element in B
```

The operation of this plugin is best explained with an example.

User Interface

The interface is divided into three panels which are accessed in sequence:

Pick clusters

In the left−hand tree select one cluster and in the right−hand tree select the cluster to compare it to. You can use drag−and−drop to transfer a cluster from another viewer to either tree.

Use the "**Next**" button to advance to the next panel.

Set options

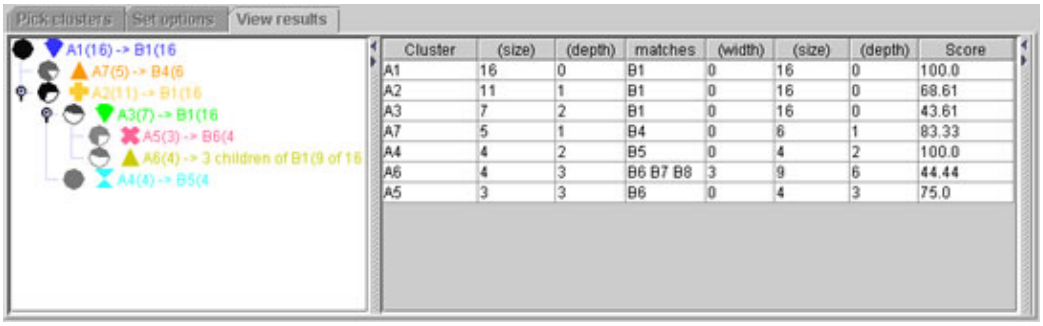
In the following description, assume that cluster X is being compared to cluster Y:

- When a cluster in X cannot be matched with a single cluster in Y, the "**Decompose imperfect matches**" forces a search to be made for the smallest collection of clusters in Y that are needed to cover the cluster in X. The list of clusters from Y that were found will be displayed in the results panel.
- The "**Calculate total coverage**" option generates an additional table of data containing the 'reverse' coverage metric values. In the case where cluster X is being compared to cluster Y, the reverse coverage shows how the elements of Y are contained in the elements of X.

Use the "**Next**" button to advance to the results panel and the "**Back**" button to return to the "Pick Clusters" panel.

View results

Once the algorithms has finished examining the clusters the results table is displayed:



Cluster	(size)	(depth)	matches	(width)	(size)	(depth)	Score
A1	16	0	B1	0	16	0	100.0
A2	11	1	B1	0	16	0	68.61
A3	7	2	B1	0	16	0	43.61
A7	5	1	B4	0	6	1	83.33
A4	4	2	B5	0	4	2	100.0
A6	4	3	B6 B7 B8	3	9	6	44.44
A5	3	3	B6	0	4	3	75.0

The cluster hierarchy of the cluster being compared is shown in the left−hand panel. Each cluster name is prefixed with a 'pie−chart' which indicates the match score that has been calculated for the cluster; a score of 100 is represented by a fully complete circle, and a score of 0 is shown as an empty circle.

The full results are shown in a table in the centre panel. A detailed explanation of the contents of this table can be found in the example below.

The rows of the table can be sorted by clicking on any of the column headings. The sort order will switch between ascending and descending on alternate clicks.

If the "Calculate total coverage" option was selected then a second table is also displayed which shows the values of the reverse coverage metric. This is also explained below.

Highlight Matches:

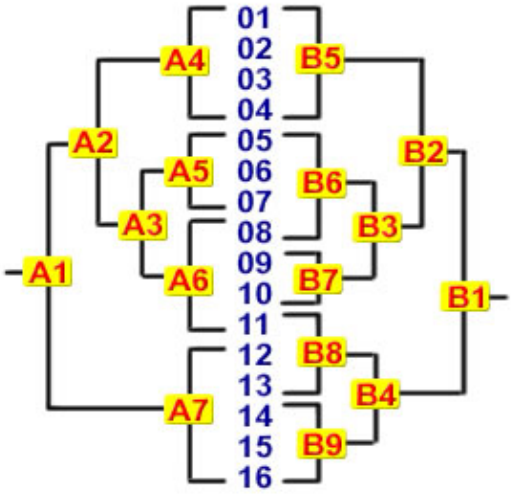
The "**Highlight Matches**" feature can be used to hide clusters which do not match well. A series of threshold values must be provided:

- Minimum match score
- Minimum cluster size
- Maximum cluster width

These values correspond to the second, fifth and eight columns of the table respectively (see below). All clusters which fall outside the provided threshold values will be 'hidden' revealing only the clusters which satisfy the specified score, size and width constraints.

Example

Consider a data set with 16 spots (named 01...16). Two different clustering algorithms (A and B) have been applied to the data resulting in two similar, but not identical cluster hierarchies as shown below:



The diagram shows the results of clustering algorithm A on the left−hand side and algorithm B on the right−hand side. For example, cluster A5 contains spots 05, 06 and 07 and cluster A3 contains clusters A5 and A6 (and thus spots 05 to 11).

The process of comparing these hierarchies would begin by selecting clusters A1 (in the left−hand tree) and B1 (in the right−hand tree) on the first panel of the plugin.

When results panel is displayed, each of the clusters in A1 will have a corresponding entry in the table:

Cluster	*(size)*	*(depth)*	*matches*	*(width)*	*(size)*	*(depth)*	*Score*
A1	16	0	B1	0	16	0	100.0
A2	11	1	B1	0	16	0	68.61
A3	7	2	B1	0	16	0	43.61
A7	5	1	B4	0	6	1	83.33
A6	4	3	B6 B7 B8	3	9	6	44.44
A4	4	2	B5	0	4	2	100.0
A5	3	3	B6	0	4	3	75.0

There are seven columns of information for each cluster; from left−to−right:

(size)	the number of elements in the cluster
(depth)	the number of parents the cluster has
matches	the cluster(s) covered by this cluster
(width)	the maximum distance between covered cluster(s)
(size)	the total size of the covered cluster(s)
(depth)	the sum of the depths of the covered cluster(s)
Score	the score as given by the coverage metric

Explanation of the results:

- A1 matches B1 with a score of 100. This is because both A1 and B1 (the two 'root' clusters) contain exactly the same spots.
- A2 matches B1; this is because there is no smaller subset of B1's children that contains the spots within A2.
- A3 matches B1 for the same reason.
- A7 matches B4 with a score of 83.3. This is because B4 contains one extra spot which is not found in A7 (note that B4 is actually the parent of B8 and B9).
- A6 matches B6,B7 and B8, but with a low score because the are a lot of spots in the union of B6,B7 and B8 which do not feature in A6. The '(width)' value is 3 because that is the maximum distance between B6,B7 and B8.
- A4 is an exact match for B5, so the 'Score' is 100.
- A5 matches B6 with a score of 75 (because only three of the four spots in B6 are contained in A5).

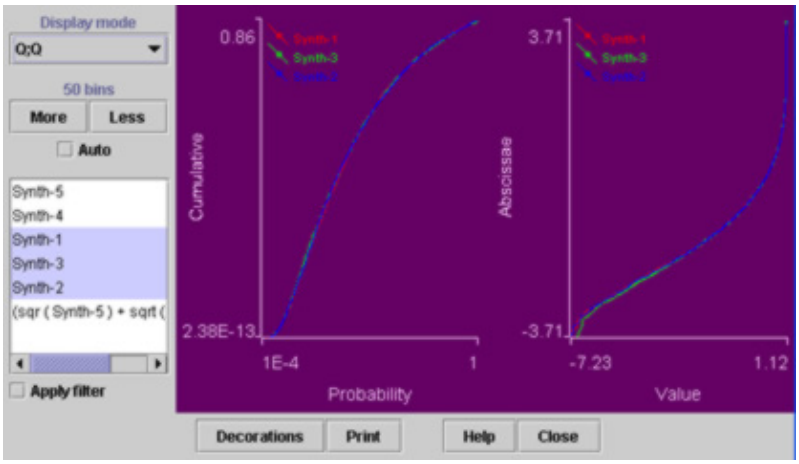
9.5 Distogram

9.5.1 Overview

The *Distogram* plugins can display a variety of distribution histograms for one or more Measurements.

Values are discretised into a number of bins, and a graph showing the number of values falling into each bin is displayed. The plot auto−scales to accomodate the values currently displayed.

9.5.2 User Interface



The control panel contains (from top to bottom):

- A label showing how many bins are being used in the discretisation process. The **More** and **Less** buttons below this label allow you to increase and decrease the number of bins.
- The **Cumulative** checkbox toggles between displaying the distribution(s) in the normal fashion, or as a cumulative graph (in which each bin is the sum of the spots in that bin, and the of the the spots in bins to the left of it).
- One checkbox for each Measurement in the data. The checkbox also shows which colour has been allocated to that Measurement. Colours are allocated sequentially using a predefined colour scheme that cannot be changed. In you really want to change colours, you can use Reorder Measurements to change which colour gets allocated to which measurement.
- Under the list of Measurements are two buttons, **All** switches on display of all Measurements, and **None** switches them all off.
- **Apply filter** selects between showing the distribution all of the data or that of only the data items which pass through the current Filter(s).

9.6 Event Watcher

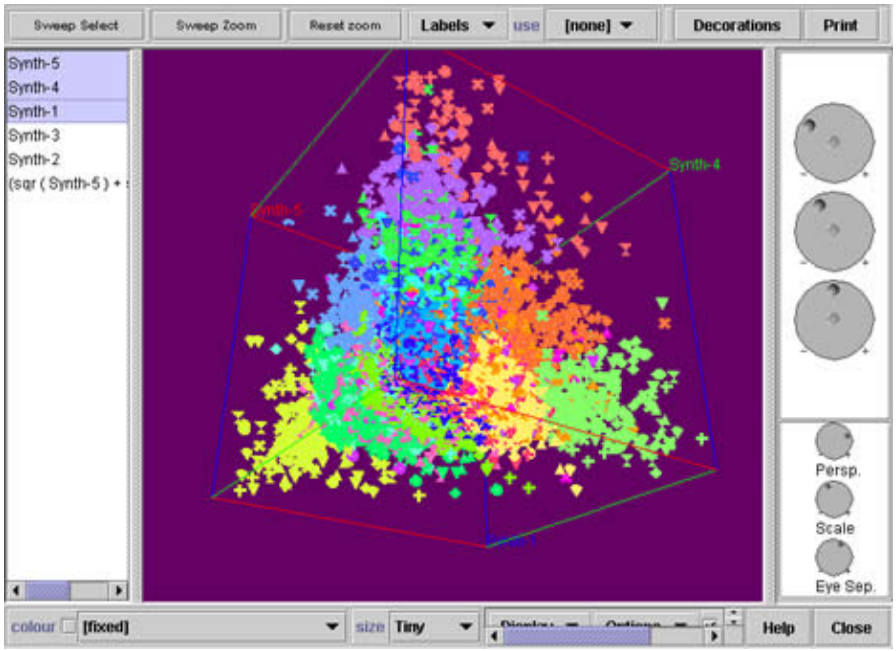
9.6.1 Overview

Event Watcher displays events as they occur.

Use it for debugging new *plugin*s, and to find out which events you should be looking out for if you want to be notified of a particular change.

9.7 HyperCube Plot

9.7.1 Overview



This plugin is an extended scatter plot for inspecting the correlation between two or more Measurements.

The *hyper*−cube is a generalisation of a cube to the *N*th dimension. The 2−cube is a flat square. The 3−cube is what we are used to as a cube. It can be thought of as an extrusion of the flat square into a new dimension. The 4−cube is an extrusion of a 3−cube in exactly the same way.

The plugin attempts to visualise hyper−cubes by projecting the N−dimensional data into 2−dimensional space and drawing dots for each data point. The data can be rotated about each of its primary axes before projection.

When used for two Measurements, this plugin provides most of the functionality of the Scatter Plot plugin.

When used with three Measurements, it provides similar functionality to the Space Plot plugin.

When used with four or more Measurements the resulting visualisations can be very disconcerting!

9.7.2 User Interface

9.7.3 Selecting axes for the plot

Select two or more Measurements from the list on the left−hand side of the window. You can use "Ctrl+click" to add and remove individual Measurements from the selection.

Each selected Measurement will be allocated to one axis of a hyper−cube. If you choose 3 Measurements then the data space will be 3 dimensional.

9.7.4 Rotating and adjusting the view

Each dimension will have a rotation knob in the top half of the panel on the right−hand side of the window. Press and hold the left mouse button on the edge of the knob and drag it to rotate the data.

The data can also be rotated by holding the left mouse button and moving the pointer in the image panel. Normally the mouse axes will control the rotation in the first and second dimensions; hold the "Ctrl" key to map the mouse axes to the third and fourth dimensions.

The rotation knobs have an `auto' feature. Click in the center of the knob to have it rotate slowly in one direction. Click it in the center again to stop the rotation. The next click causes it to start rotating in the opposite direction.

The bottom half of the right−hand panel contains three viewer controls:

- **Persp.**
  adjusts the degree of perspective in the projections
- **Scale**
  is a global size control to fit the data to the window
- **Eye Sep.**
  sets the amount of eye separation in stereoscopic views

Additional controls in the bottom panel are:

- **Show clusters**
  show cluster glyphs (for any visible clusters) in the display
- **...and edges**
  adds the edges between clusters to the display
- **Show axes**
  display the colour coded axes of the space
- **Show limits**
  display the full extent of the space
- **Apply filter**
  use the current filter(s) to hide data
- **Use wireframe**
  use a quicker drawing method to make rotating smoother
- **Uniform scale**
  force each axis to use the same scaling factor
- **Depth sort**
  draw the data points in the correct depth order
- **Stereo**
  display the data from 2 viewpoints for low−tech stereo viewing
- **Time**
  show performance statistics

9.7.5 Identifying Spots

When the mouse pointer is rested over a dot for a moment, a tool−tip window appears containing an identification label. This label can be any of the Name or Name Attributes in the current data. The choice of which Name or Name Attribute to show is made via the drop−down menu in the control bar along the top of the window.

Click on a spot to add a name label to the plot. Click on the spot again to remove the label. The "**Clear all**" button removes all name labels that have been added to the plot.

Drag−and−drop can be used to locate particular Spots in the plot. Drag one or more Spots from the main display and drop them into the *HyperCube Plot* window. This causes the name(s) to appear next to the dots on the plot. Dropping a

Cluster onto the panel causes the names of all Spots in the cluster to be displayed.

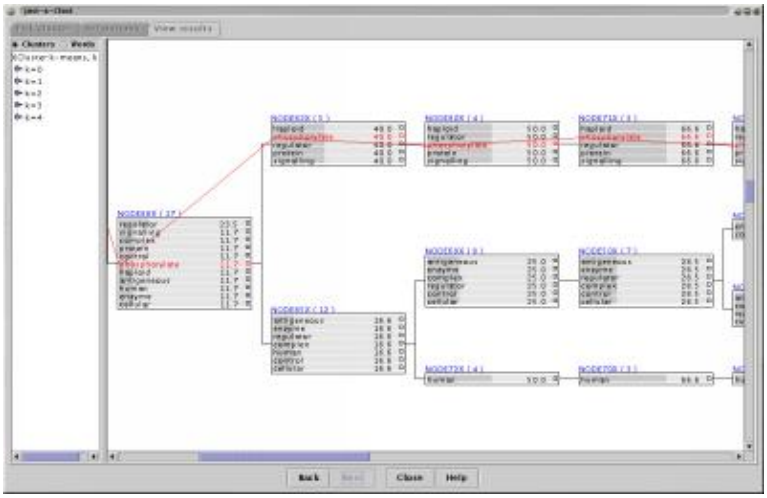
9.8 Just−o−Clust

9.8.1 Overview

The *Just−O−Clust* plugin attempts to `justify' clusters of Spots by finding repeated words in the text linked to them.

Words can be extracted from the Gene or Probe names and name attributes, or from the annotation linked to the names.

The plugin then generates a display showing how words are repeated in the contents of each cluster.



9.8.2 User Interface

The interface is divided into three panels which are accessed in sequence:

- Pick cluster
- Set options
- View results

9.8.3 Pick cluster

Select a cluster for justification. You can pick any cluster from the tree or use drag−and−drop to transfer a cluster from another viewer.

Use the "**Next**" button to advance to the next panel.

9.8.4 Set options

- **Word source**: this menu allows to you to select one of the names or name attributes to use as a source of words. The menu also has an "Annotation" option which specifies that text from the annotation cache should be used instead.
- **Delimiters**: specify characters used to split text strings into individua l words.

  The text from source specified in "**Word source**" is converted into words using the "**Delimiters**". These characters are used to identify the start of a new word in a text string. The "**Delimiters**" string usually includes characters such as " " (space) and "," comma. You can customise the "**Delimiters**" to suit the text you have.

For example: using the delimiters " ,:" (i.e. a space, a comma and a colon) the string "one:red, two:green, three:blue" is converted into six words ('one', 'red', 'two', etc.) If the delimiters string does not contain a ":" then the example string will be converted into three words ('one:red' etc.).

You can use RETURN and TAB as delimiters by refering to them as '\n' and '\t' respectively.

- **Min word length**: Words shorter than this length will be ignored
- **Detect words as substrings**: When enabled, words which occur as substrings in other words will be counted identically, otherwise they will be treated as different words.

  For example, with this mode enabled, "nonhuman" and "subhuman" will both match as "human". With this mode disabled these three words will be treated differently.
- **Case sensitive**: When enabled, the case of the text is considered during matching, otherwise upper− and lower−case characters are identical.
- **Stop words**: A list of words which should be ignored. This button opens an editor for the list of stop words. Words in the "**Word source**" which have been identified as stop words will not be counted or displayed in the results.
- **Translations**: A translation table recording pairs of words that are synonyms. This opens an editor for the list of translations. Each translation is two words (called 'from' and 'to'). Words in the "**Word source**" which match a 'from' word are translated to the corresponding 'to' word.

  The translation table is useful when the text sources use several different terms or abbreviations for the same thing.

Use the "**Next**" button to advance to the next panel and the "**Back**" button to return to the "Pick Cluster" panel.

9.8.5 View results

The results of the word matching are displayed in a scrollable panel. The selected cluster and it's children are laid out left to right.

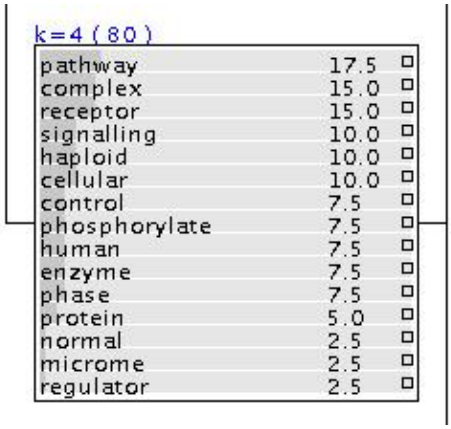
The panel on the left hand side can display either a tree of clusters or a list of all of the repeated words.

When the cluster tree is displayed clicking on a cluster in the tree scolls the right hand panel to display that cluster.

When the list of words is displayed clicking on a word selects that word causing a line to be drawn following the occurrences of the word in the cluster hierarchy.

One box like that seen below is displayed for the cluster that you picked and one for each of its descendants. Each box contains one line for each words that is repeated in text linked to spots in that cluster. Next to the word is displayed the percentage of spots in the cluster which contain that word. If for example all spots in a cluster have the word "protein" in their text then this word will appear at the top of the list with "100%" beside it.

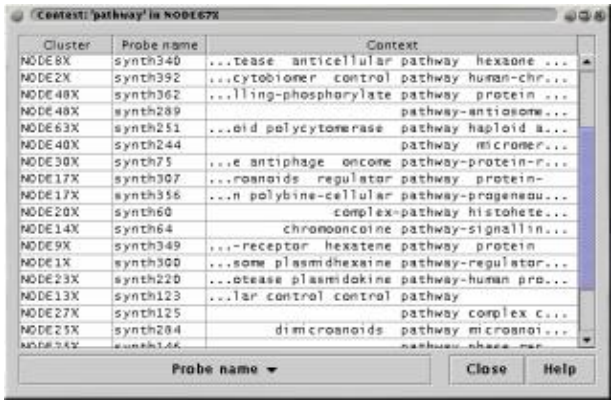
Note that the percentage results for parent clusters are inclusive of the results of their children (and grandchildren and so on).



The above example shows the repeated words in a cluster called "k=4". The total number of spots in the cluster and all of its descendants is shown in brackets beside the name.

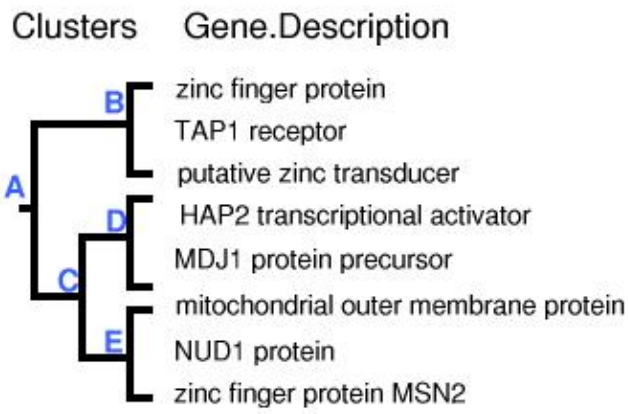
This cluster contains 80 spots. From the word source for those spots 15 words have been found to occur in more than one Spot. The repeated words are shown, ranked by the percentage of spots containing the word.

In this example the word "pathway" has been found in 17.5% of the spots and "complex" in 15% of the Spots. On each line, beside the percentage, is a small button which opens window containing more detailed results.



This popup window shows which of the spots contained the word, and shows the chunk of text in which word occurred.

9.8.6 Example



The above picture shows a simple cluster hierarchy containing eight spots. Cluster names are shown in blue. Text information about each spot is stored the "Gene.Description" name attribute.

If the *Just−o−Clust* is let loose on this example then the output would be:

- "A": zinc 36.7%, protein 50%
  (because 'zinc' occurs in 3 of the 8 spots and 'protein' in 4 of 8)
- "B": zinc 66.6%
  ('zinc' occurs in 2 of the 3 spots)
- "C": protein 60%
  ('protein' in 3 of the 5 spots)
- "D": no repeats
- "E": protein 66.6%
  ('protein' in 2 of the 3 spots)

Overview

Use the *Measurement Manager* plugin to view and manipulate Measurements. Using the controls in this panel, Measurements can be renamed, deleted and hidden, Colourisers can be selected and Measurement Attributes can be examined, altered and searched.

User Interface

On the left of the panel is a list containing all of the Measurements in the current data. One or more Measurements can be selected at a time; use the "**Select All**" button to conveniently select all of the Measurements in the list.

To the right of this list are three tabbed panels, each showing a different aspect of the currently selected Measurements:

- The Properties tab shows basic information, such as the name and datatype.
- The Attributes tab can display one or more Measurement Attributes
- The Statistics tabs shows some additional data, such as minimum, maximum and mean data values

The bar which divides the list of Measurement names from the tabbed panels can be dragged to the left or right to provide more space for the names or tabbed panels as required.

Along the bottom of the panel are some controls which are always available regardless of which tabbed panel is currently displayed:

- The "**Delete**" button removes the currently selected Measurement(s) completely. A confirmation message is displayed before the data is discarded.
- The "**Cluster**" button creates a new Cluster containing the currently selected Measurement(s). A dialog box will be displayed in which a name for the new cluster can be provided.
- When the "**Sync. Selection**" toggle is active, the selection in the Measurement list will be synchronised with the main Measurement selection. Changes to the main selection will be reflected in the list, and *vice versa*.

Measurement Properties

These controls are (from top to bottom):

- The **Name** of the Measurement can be changed by typing into the text−edit field (press return to trigger the update).
- The **Show** and **Hide** buttons determine whether this Measurement is visible, i.e. whether is will appear, or be available for selection, in viewer plugins. Use this to temporarily disable a Measurement without completely

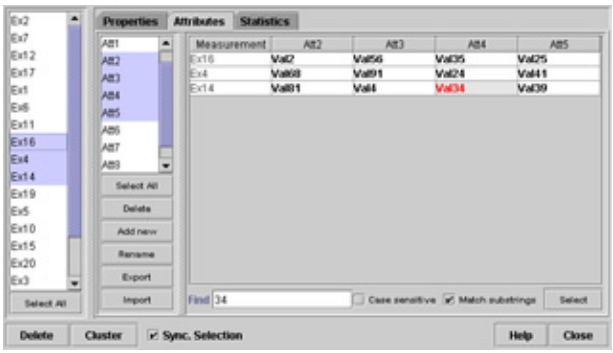
deleting it.

- A list of the Spot Attributes which exist in the currently selected Measurements(s) is displayed. This list is not editable − the Spot Attributes plugin is provided to view and maniupulate Spot Attributes.
- The **Colouriser** used by this Measurement is selected via this drop−down list. Colourisers are created, edited and deleted via their own control panel.
- The **Data Type** of the Measurement is selected via a drop−down menu. Some other parts of **maxdView** allow you to apply operations to all Measurements of a particular data type.

To change the attributes of more than one Measurement at a time, e.g. to hide several Measurements, make a multiple selection in the list (using 'Shift'−click or 'Ctrl'−click).

The "Name" cannot be changed during multiple selections, but all other attributes can be changed. This makes it easy, for example, to assign a different "Colouriser" to all Measurements at once.

Measurement Attributes



Measurement Attributes are *name:value* data items which can be associated with **Measurement**s. Each Measurement can store any number of Measurement Attributes. These attributes can be viewed, edited, searched, imported and exported using this panel. Any combination of Measurements and Measurement Attributes can be selected for display.

A list of the names of all Measurement Attributes (sorted into alphabetical order) is displayed on the left hand side of the panel. As with the Measurement list, one or more names can be selected at a time. Below this list is a set of control buttons:

To the right of this list is a table in which the names and values of the currently selected Measurement Attributes are displayed. When there is sufficient space, the time of creation and time of last modification are also displayed.

- "**Select All**" selects all Measurement Attributes in the list
- "**Delete**" permanently deletes (after confirmation) the currently selected Measurement Attributes from the current selected Measurements
- "**Add new**" opens a dialog box requesting a name and then adds a Measurement Attribute with that name to each of the currently selected Measurements. The value for the attribute can be edited directly in the display table (see below)
- "**Rename**" changes the name of the currently selected Measurement Attribute in all of the currently selected Measurements. A dialog box requesting a new name is displayed and confirmation of the name change is required. This function is only available when a single Measurement Attribute is selected.
- "**Export**" opens a dialog box which allows the currently selected Measurement Attributes to be written to a plain text file (see below)
- "**Import**" opens a dialog box which enables one or more Measurement Attributes to be extracted from a plain text file (see below). A wide range of possible file formats are supported.

Editing

The attribute values can be edited by clicking on the cell in which they are being displayed. Only the 'value' and 'source' fields can be edited, the 'created' and 'last modified' fields are controlled automatically.

Searching

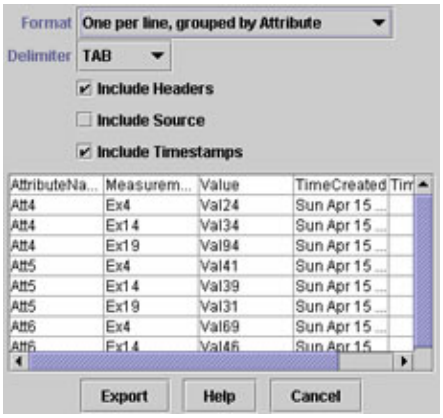
Under the table which displays the attribute fields is a type−in field which is used for searching. Type a string into this field and any attributes which contain this string will be highlighted in the table. All attribute fields are checked in the search.

The "**Case sensitive**" and "**Match substrings**" options control how the searching is performed. "**Match substrings**" toggles between allowing only complete matches or allowing partial matches (i.e. the string 'arg' partially matches 'target').

The "**Select**" button selects all Measurements and Measurement Attributes which match the current search, i.e. those which are currently highlighted.

Exporting

The currently selected Measurement Attributes (i.e. the contents of the table) can be exported to a plain text file.



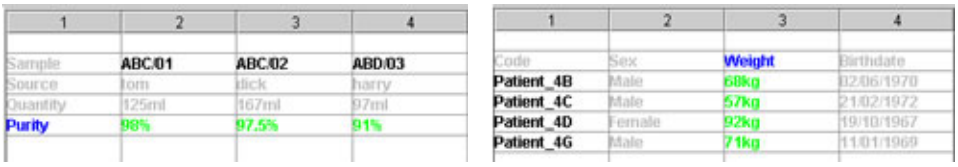
The export dialog box shows a preview how the file will be structured. The following controls are provided:

- **Format** − several methods for data layout are supported.
- **Delimiter** − columns can be separated by "TAB", "Space" or "Comma".
- **Include Headers?** − whether column headers are provided.
- **Include Source?** − whether the 'source' field is exported.
- **Include Timestamps?** − whether the 'created' and 'last modified' fields are included.

Note that the 'source', 'created' and 'last modified' fields are only available when one of the "One per line..." formats is selected.

Pressing the "**Export**" button displays a file chooser dialog in which the file name and location for the exported data can be selected. The data will be written once the file has been chosen. The "**Cancel**" button aborts the data exporting process.

Importing



The process of importing Measurement Attributes begins by choosing a file. The contents of the chosen file are then displayed in a table view. The delimiter that separates columns can be either "TAB", "Space" or "Comma".

The data to be imported is defined by tagging cells in the tabular view of the data. Regions of the table are selected by pressing and dragging the mouse or by 'shift' clicking in the usual fashion. Once a region is selected one of the three 'tag' buttons should be pressed to indicate what type of data the cells contain. The three possible types are "Measurement Names", "Attribute Names" and "Attribute Values".

The tagging process is best illustrated by example: consider the following tabular data

name	Subject 5	Subject 6	Subject 7
sex	male	male	female
weight	67.8	74.3	99.1

The "Measurement Names" are in the top row and the "Attribute Names" are in the left−most column. By selecting the revelant cells in the table and using the 'tag' buttons, the table can be coloured as follows:

name	**Subject 5**	**Subject 6**	**Subject 7**
sex	male	male	female
weight	67.8	74.3	99.1

Finally, the "Measurement Values" cells are identified in the same way, i.e. they are selected in the table and the "Tag as Measurement Values" buton is presed:

name	Subject 5	Subject 6	Subject 7
sex	male	male	female
weight	67.8	74.3	99.1

Once a valid configuration of cells have been tagged, the "**Export**" button will become enabled.

If the contents of cells that are tagged "Measurement Names" cannot be matched with the names of the current Measurement then an error message is displayed and the data is not imported. In the above example, there would have to be existing Measurements with the names "Subject 5", "Subject 6" and "Subject 7".

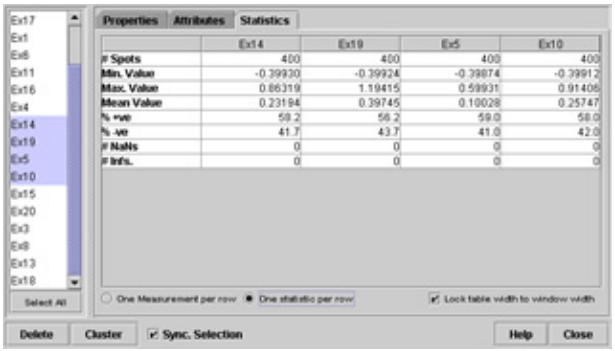
If the attribute names that are being imported match names of attributes that already exist then a warning message will be displayed with the option to replace the existing values or automatically rename the attributes as they are loaded.

This method of tagging the data to be imported means that many different layouts can be handled. The most general case is that a single attribute *name:value* pair can be imported from anywhere in the data, in the following example...

name	Subject 5	Subject 6	**Subject 7**
sex	**male**	male	female
weight	67.8	**74.3**	99.1

...Measurement "Subject 7" would be given an attribute called 'male with the value '74.3'.

Measurement Statistics



The 'statistics' tabbed panel displays a number of useful statistics for the curently selected Measurement(s):

`#Spots` the number of Spots in the Measurement
`Min.Value` the minimum value in the Measurement
`Max.Value` the maximum value
`Mean Value` the mean (i.e. max−min * 0.5) value

% +ve the percentage of Spots with values > zero
% −ve the percentage of Spots with values < zero
#NaNs the total number of Spots which are 'Not−A−Number', i.e have an undefined value
#Infs the total number of Spots with the value of 'Infinity'

The following options are available to control how the statistical data is displayed:

- "**One Measurement per row**" arranges the view so that each statistic is displayed in one column (this is generally better when there are a large number of Measurements selected)
- "**One statistic per row**" arranges the view so that all statistics for each Measurement are displayed in a single column
- "**Lock table width to window width**" fixes the table width to that of the tabbed panel so that the available space is divided equally between the columns. If there are lots of columns in the table, deactivating this option alows the table to become wider than the available space, thus freeing up more space for columns.

9.9 Notepad

9.9.1 Overview

Use *drag and drop* to copy things (such as gene names, cluster names) into the *Notepad* viewer. You can use the text entry area at the top to store information about the contents of the *Notepad*.

Use the "Save" button to write the contents of the *Notepad* to a text file.

"Clear" removes everything from the *Notepad* and resets the text area.

9.10 Profile Viewer

9.10.1 Overview

This plugin displays one or more groups of 'expression profiles'. An expression profiles is a line which represents how the expression value for a Spot changes across Measurements. A single group of hand picked Spots can be displayed, or groups can be generated from a cluster hierarchy.

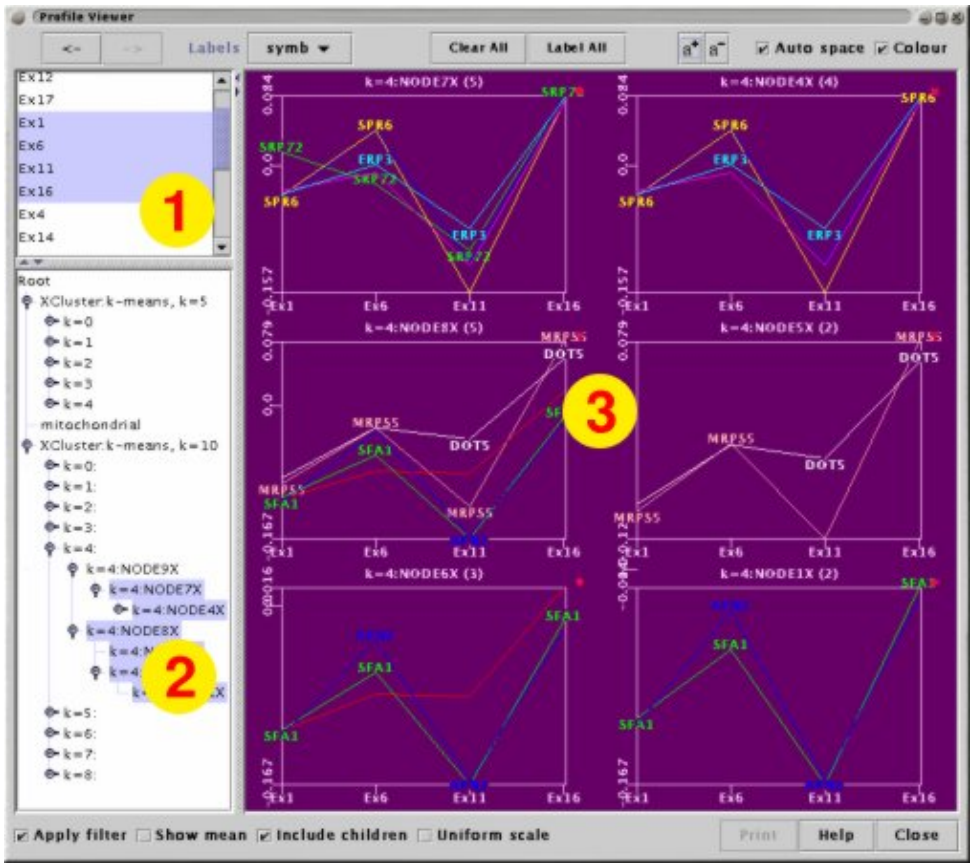
The expression profiles are essentially the same visualisation as that given by the Web Plot. The difference with this plugin is that the grouping of Spots into Clusters can also be displayed.

In "Cluster" mode you can select one or more Clusters from a tree or use *drag−and−drop* to move Clusters from other parts of **maxdView**.

You can 'drill down' into a Cluster by clicking on the glyph drawn on the top−right on each graph.

In "Spot" mode you can select one or more Spots from an alphabetically spoted list of Names or Name Attributes. Again, *drag−and−drop* can be used to select Spots in the list. Dropping a Cluster onto the list will select all Spots in that Cluster (and it's descendants).

9.10.2 User Interface



The plugin is split into three areas:

1. The Measurement selection area with which you select two or more Measurements to include in the profile.
2. The Cluster selection area with which you select one or more Clusters to draw profiles for.
3. The visualisation panel in which the profiles appear. There will be one graph for each selected Cluster. The name of the Cluster and the number of Spot profiles is shown above each graph.

When the **Include children** option is selected, each graph will include the Spots from the Cluster and all of it's children, otherwise just the Spots in the Cluster itself are displayed.

The **Show mean** option toggles between showing profiles for each Spot in the Cluster or calculating a mean profile for all Spots in the Cluster. When the mean profiles are displayed, error bars are included show the minimum and maximum values along the profile

The **Apply filter** option selects between showing all of the Spots in selected Clusters, or only showing the Spots which pass through the current filter(s).

The **Uniform scale** option determines whether all graphs are drawn to the same scale or whether each graph is allowed to optimise it's scale to the range of the data it is displaying.

The Cluster selection panel has a shortcut feature. Select a Cluster, then press a number key '1',..,'9' to expand the tree to the specified depth. Press '0' to collapse the branches of the selected Cluster.

You can use *drag−and−drop* to move Clusters to the Cluster selection panel and Measurement Names to the Measurement selection panel.

9.10.3 Identifying Spot Profiles:

When the mouse pointer is rested over a profile for a moment, a tool−tip window appears containing an identification label. This label can be any of the Name or Name Attributes in the current data. The choice of which Name or Name Attribute to show is made via the drop−down menu in the control bar along the top of the window.

Click on a profile to add a name label to the plot. Click on the profile again to remove the label. The "**Clear all**" button removes all name labels that have been added to the plot. The "**Label all**" button adds labels to all Spot profiles which are currently displayed.

The size of the label font can be adjusted using the "**a+**" and "**a−**" buttons.

Drag−and−drop can be used to locate particular Spots in the plot. Drag one or more Spots from the main display and drop them into the *HyperCube Plot* window. This causes the name(s) to appear next to the dots on the plot. Dropping a Cluster onto the panel causes the names of all Spots in the cluster to be displayed.

The "**AutoSpace**" option attempts to improve the readability of the graphs by reducing the quantity of displayed labels. When this mode is enabled, labels will only be drawn if there is enough space for them. Note that when there is not a lot of space, or a lot of labels, this mode can lead to some Spots becoming unlabelled.

Spots which have been selected for labelling can also be displayed in colour to make them more visible. The "**Colour**" checkbox controls this feature.

9.10.4 Cluster Drill Down:

You can 'drill down' into a Cluster by clicking on the glyph drawn in the top−right on each graph. (This glyph is the coloured shape that is associated with the Cluster via the Cluster Manager). Profiles for the Cluster and up to eight of it's children will be displayed. The Cluster that was drilled onto will be in the top−left corner.

9.10.5 Example:

- Load the tutorial file "tutorial1.maxd.gz" from the "demo" directory (use the *Read Native* plugin
- Select two or more Measurements in the top−left panel
- Select one or more Clusters in the bottom−left panel.
- A graph will appear for for each selected Cluster. This graph shows the expression profiles for the Spots in that Cluster.
- Add and reomve Measurements from the selection using "Ctrl" + left mouse clicks.
- Add and remove Clusters from the selection using "Ctrl" + left mouse clicks.

9.11 QC Chart

9.11.1 Overview

The QC Chart Plugin is based on a methods explained by Y. Fang et al [1]. These methods produce a **Control Chart** of the data which can be used to assess the quality of cDNA microarray slides.

The QC Chart Plugin is used to examine a set of normalised slide measurements, to assess whether each one falls close to the mean of its group (replicate set). The idea is to identify slides which are non−typical within each group.

The QC Chart plugin is used on data which has already been normalised and put into log ratio form

The user first specifies which normalised log ratio measurements are to be used in the calculation. These may be either from one group (all slides measure the same pair of conditions) or from several groups. The user can use the default list of replicate groups, or specify more meaningful group names reflcting the conditions in his experiment. The user assigns each slide to a group. The user also specifies which slides are straight repeats (forward) and which are dye flips (reverse). Finally, the user specifies the significance cutoffs to be used, or accepts the default values.

When the user has supplied this information, the QC Chart calculates the mean in a spotwise manner for each group. Then, for each slide, the spotwise difference between the slide values and the mean values are calculated. Finally, for each slide, the variance of this difference from the mean is calculated. This variance is an indicator of how far the slide is from the average.

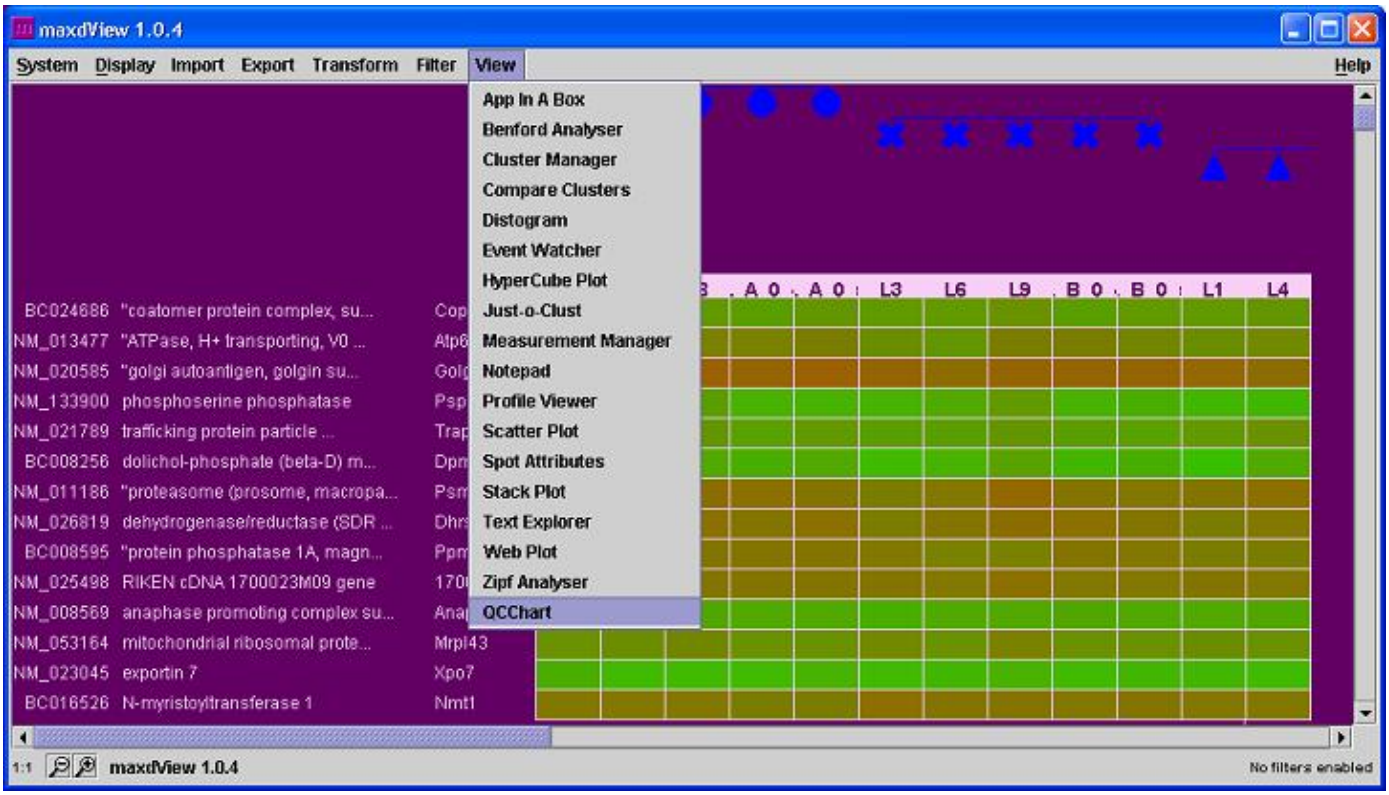
The values are reported in a table, and also on a Control Chart graph. The graph shows the log (base 2) values of the variance. We would expect these to follow a normal distribution.

The **Upper Control Line**, **Lower Control Line** and **Centre Line** are all displayed on the graph and in the table. One point per slide is plotted on the graph, giving its quality control value. If the number of slides is small, then the user may be interested in the confidence bars around the control lines, as well as the control lines themselves.

The user may infer that slides which fall above the upper lines are varying a lot from the average, and may consider excluding such slides from the study. The user may note slides which fall very low down − this corresponds to low variance, and may be an indicator that a superior technique was used to obtain that data. This could have implications for improving data collection techniques in the experiment.

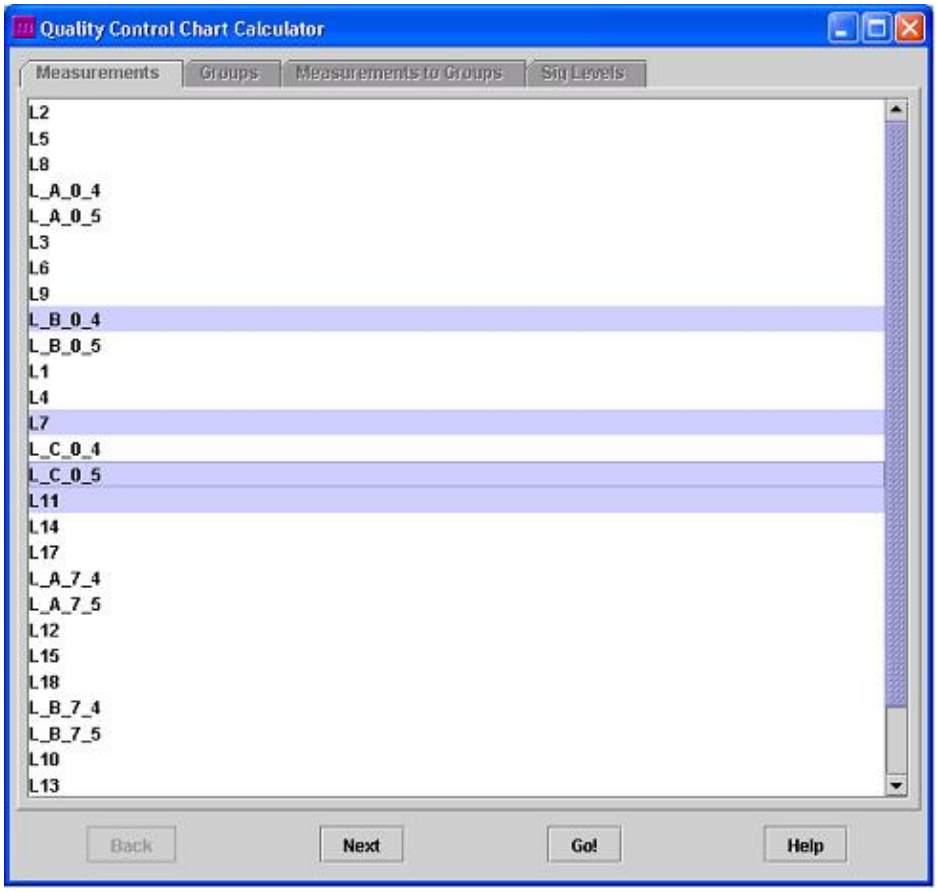
9.11.2 User Interface

To Start the Quality Control Chart Plugin, select **QCChart** from the **View** menu.



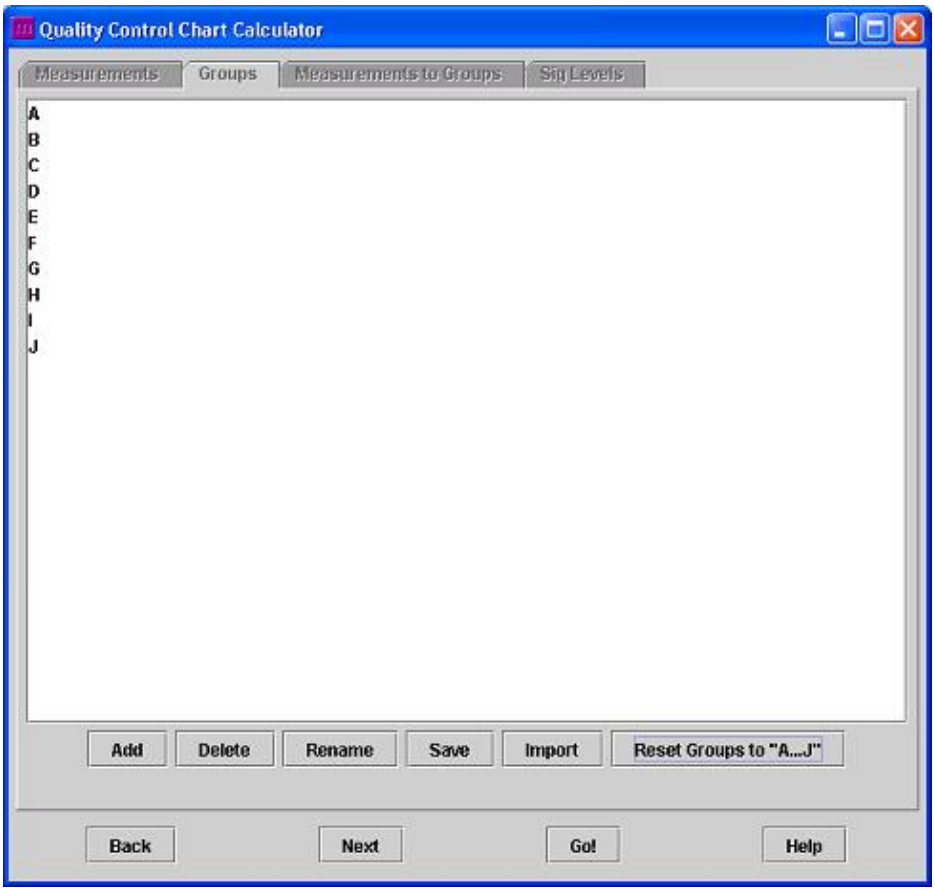
The Quality Control Chart user interface will appear, with the **Measurements** tab selected. From the list of measurements, select the measurements which you are interested in. These should be log ratio values, (or normalised log ratio values) from two−colour cDNA microarrays.

You can use CTRL−mouseclick to multiple−select the measurements you require.

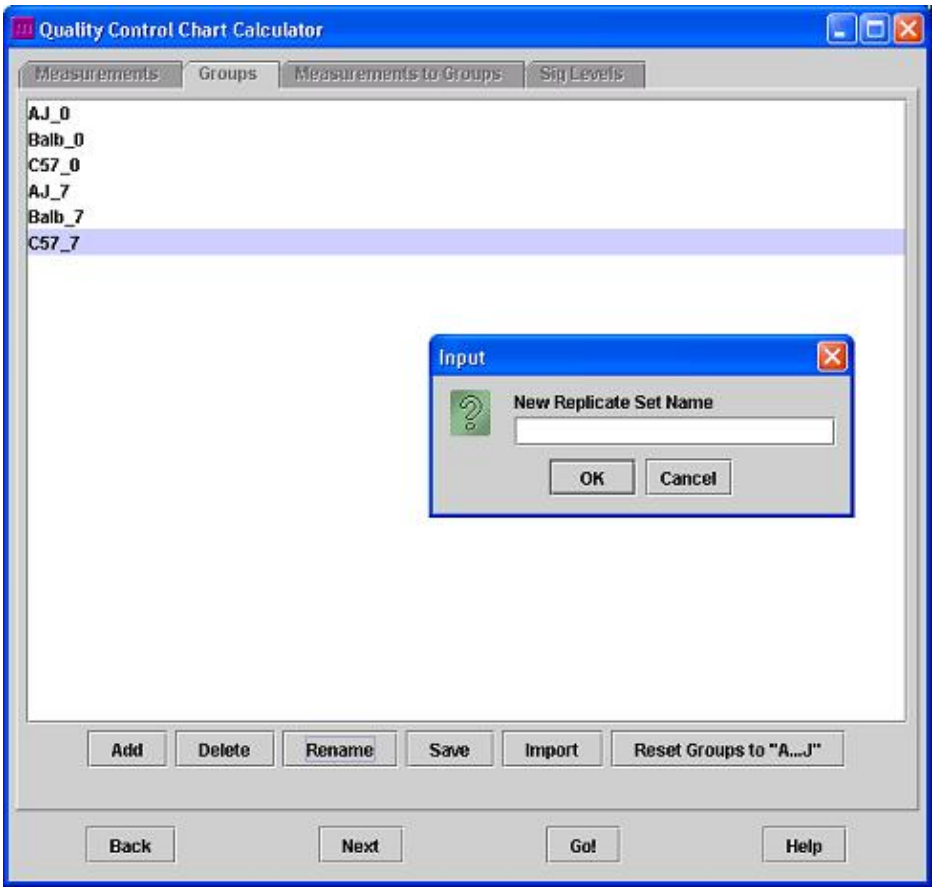


Then click on the **Next** button. The group options tab will then appear. This window contains the names of the replicate

groups to which you will assign each of your chosen measurements. The default group name list consists of the letters 'A' though 'J':



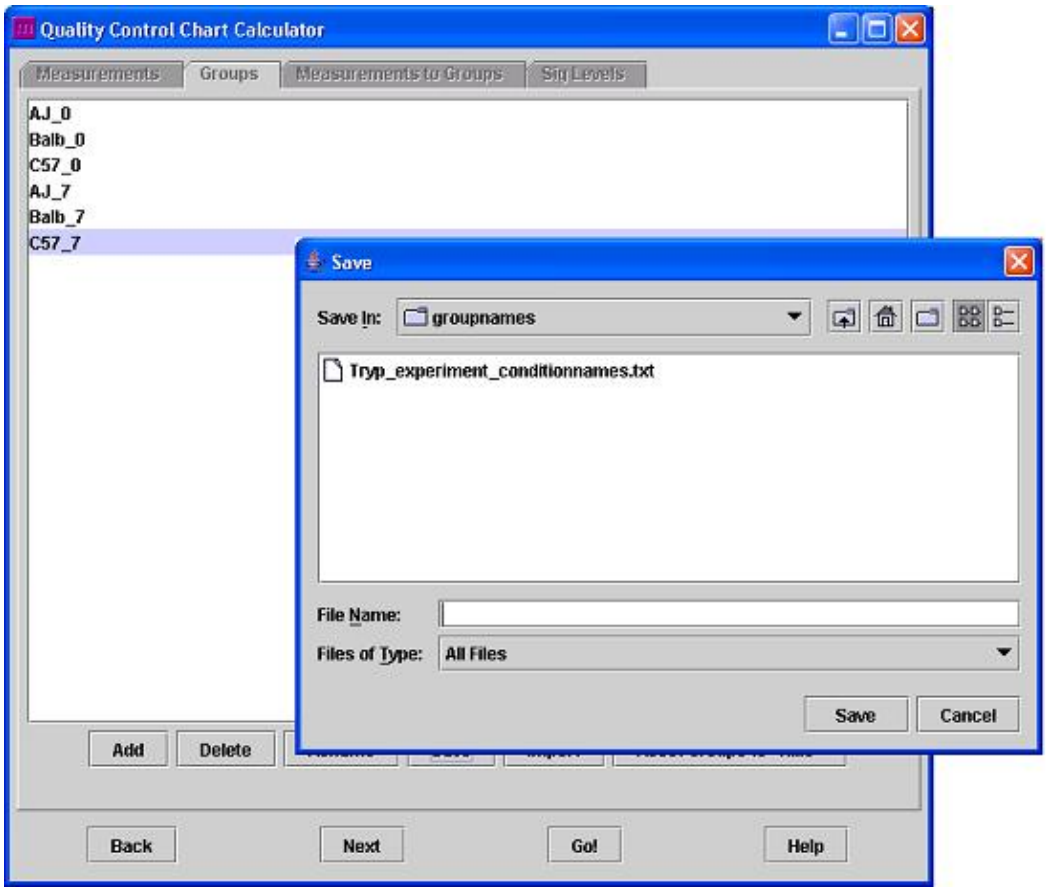
You can use the 'Add', 'Delete' and 'Rename' buttons to change the group names to something more meaningful to your experiment:



You can also import a list of group names from a file. This will replace the existing group names with those specified in the file. The format of a group names file is one group name per line, e.g:

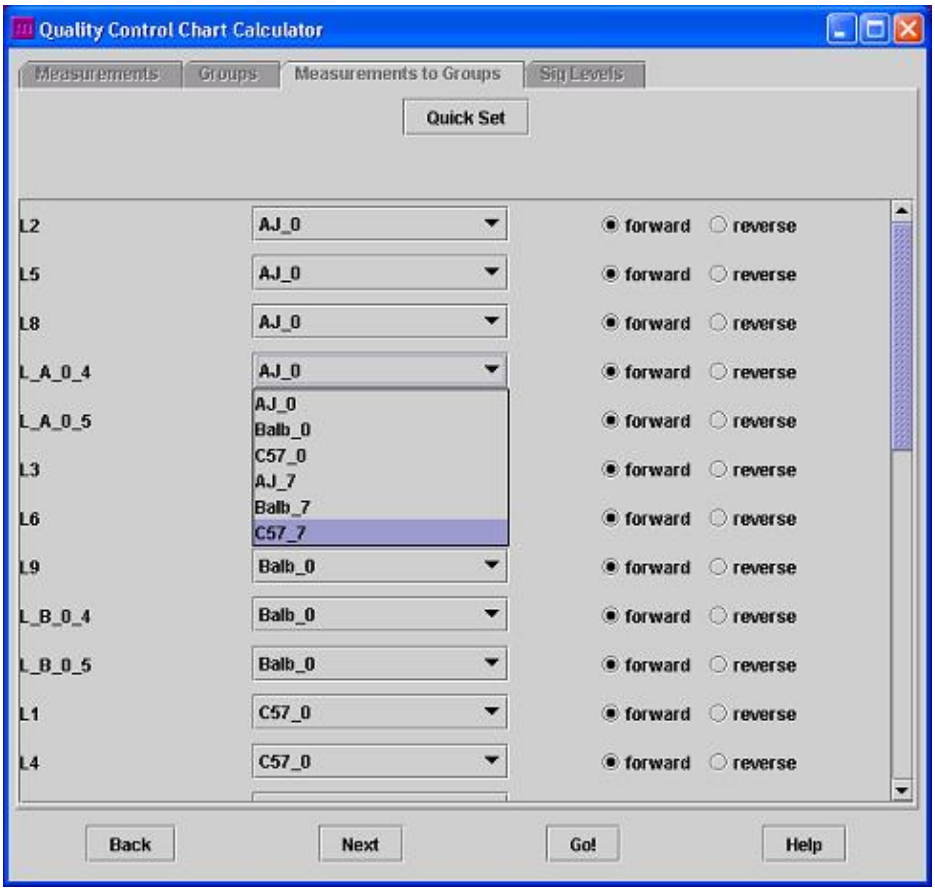
```
timepoint_0hrs
timepoint_8hrs
timepoint_24hrs
```

You can save the group names to a file of this format using the 'Save' button.



Now click on the 'Next' button to reach the tab where you assign measurements to groups. Note that you can use the 'Back' button if you want to return to previous tabs to change your selection of measurements or to edit the names of the groups.

You are now looking at the 'Measurements to Groups' tab. This contains a list of all the measurements you have selected. For each measurement, there is a drop down box where you can select the group to which that measurement belongs.



There is also a radio button where you can choose 'forward' or 'reverse' for each measurement. The purpose of this is for use when dye flips have been carried out, and where the calculated log ratios have not yet been reversed. For example, consider the following case:

```
slide 1      condition A = Red      condition B = Green     meas1 = normalise(log(R/G))
slide 2      condition A = Green    condition B = Red       meas2 = normalise(log(R/G))
```

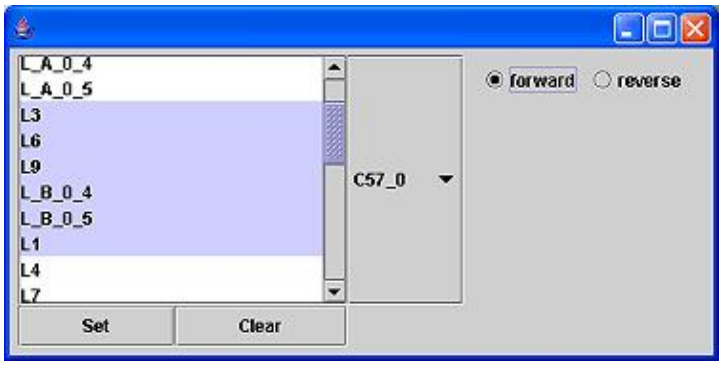
Here, slide 2 is a dye flip repeat of slide 1, and meas2 should be comparable to −meas1. So it is appropriate in this case to choose 'forward' for measurement 1 and 'reverse' for measurement 2. If you already carried out this calculation in the preliminary steps, with e.g:

```
slide 1      condition A = Red      condition B = Green     meas1 = normalise(log(R/G))
slide 2      condition A = Green    condition B = Red       meas2 = −normalise(log(R/G))
```

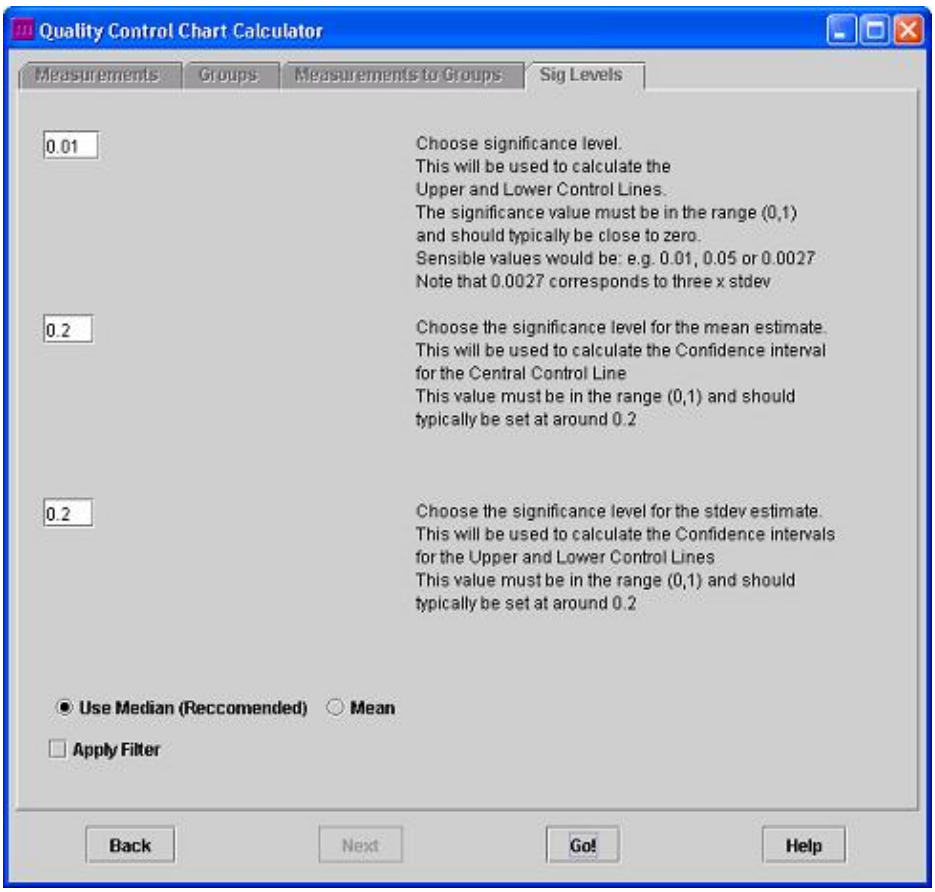
then it is appropriate to choose 'forward' for both meas1 and for meas2.

In this way, you are specifying whether the calculator should use the data values as they stand (forward hyb) or whether the calculator should take the negative of the values (reverse, dye flip case, where this adjustment has not already been made)

For an experiment with many measurements to consider, it may be inconvenient to choose each group or each orientation individually. For this reason there is a **Quickset** option. Clicking on the 'Quickset' button brings up a Quickset pane where the user can multiple−select several measurements at once. Choose the appropriate group and/or orientation for those measurements, and click on 'Set' to confirm the selection. Repeat this until all measurements are assigned as required.



Click on 'Next' to get to the significance level window, where you can set various options for how the calculation will be carried out.



Specify a numerical value for each significance level as required. Suitable values are e.g. 0.01, 0.2, 0.2. See the Maths section for explanation of these values. A small value for the main significance level will result in a strict application of the control chart, with the UCL and LCL being placed close to the center line, and more slides failing the test. A small value for the two confidence significance levels will result in a stirct application of the confidence bars, the width of the confidence bars becoming very wide.

Decide whether you want the **Median** or the **Mean** to be used in the spot−wise average step of the calculation. See the Maths section of this help page for where this is used. Usually the **Median** is chosen, because it is considered to be more robust for small numbers of replicates.

Use the **Apply Filter** checkbox to choose whether or not to apply the filter. If the checkbox is ticked, then any spots which have been filtered out of the main window of maxdview will be excluded from the calculation.

If your data included NaN or Inf values, these will be excluded from the calculation. The number of such exclusions will be reported in a dialog box.

When all selections have been set as required, hit the **Go!** button

The results of the calculation are displayed in a table and a graph. Resize the graph so that your results are displayed nicely. The table reports the variance of the difference from the (group) average for each slide, and the log (base 2) of this value.

We consider slides whose value falls below the Upper Control Line (UCL) to be normal, or exceptionally good if they also fall below the Lower Control Line (LCL). See the Maths section for details of how these values are calculated. Those values which fall above the UCL have abnormal variance from the mean, and so are considered to be bad. The table reports 'Normal' 'Excellent' and 'Bad' accordingly.

Furthermore, a confidence interval is calculated for each control line. Slides which fall within the confidence interval of the UCL or LCL are labelled 'Borderline Excellent' or 'Borderline bad' accordingly.

QC Results Table

The results of the calculation are:

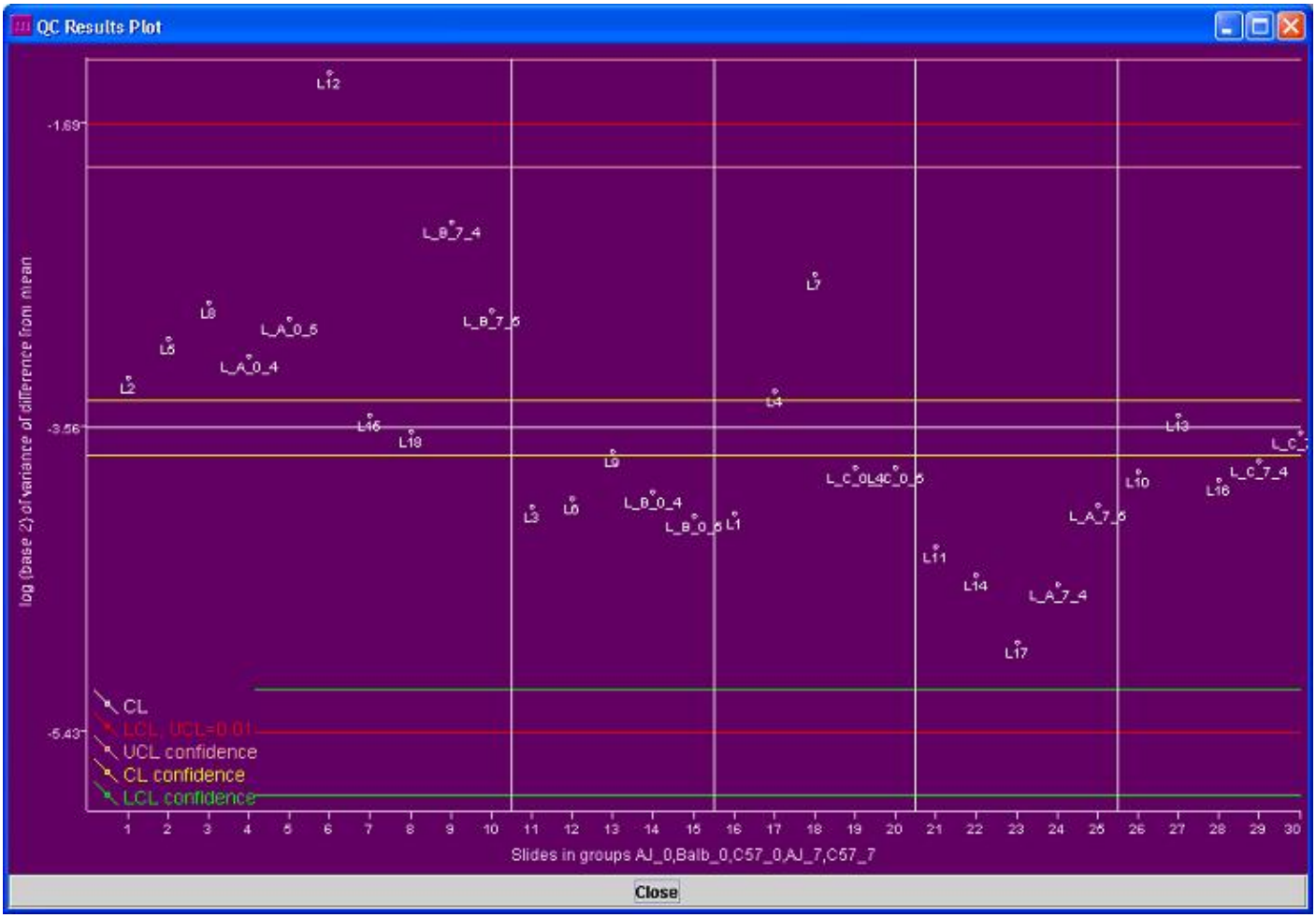
mean of logged vars: (mu)	-3.562452991990057	
stdev of logged vars: (sigma)\tt	0.7252940842952083	
Chosen significance value (alpha)	0.01	
Phi^(-1) (1-alpha/2)	2.5758293035489004	
Centerline	-3.562452991990057	
Upper Control Line	-1.6942192359717934	
Lower Control Line	-5.430686748008321	

Confidence Bars:

Chosen significance value (alpha_mu)	0.2	
Chosen significance value (alpha_sigma)	0.2	
Confidence Bar for UCL:	-1.9532477649900835	to -1.2996243613216216
Confidence Bar for CL:	-3.7321560207355904	to -3.392749963244524
Confidence Bar for LCL:	-5.825281622658492	to -5.17165821899003

measurement	group	variance of diff	log 2 var	flag
L2	AJ_0	0.1050246830...	-3.2511996623...	Normal
L5	AJ_0	0.1237379632...	-3.0146399028...	Normal
L8	AJ_0	0.1447113586...	-2.7887499291...	Normal
L_A_0_4	AJ_0	0.1151728942...	-3.1181268737...	Normal
L_A_0_5	AJ_0	0.1352877588...	-2.8858967877...	Normal
L12	AJ_0	0.3850642887...	-1.3768287626...	Borderline Bad
L15	AJ_0	0.0891176451...	-3.4881450784...	Normal
L18	AJ_0	0.0834450165...	-3.5830302973...	Normal
L_B_7_4	AJ_0	0.2032036933...	-2.2990014704...	Normal
L_B_7_5	AJ_0	0.1394809831...	-2.8418596568...	Normal
L3	Balb_0	0.0604188204...	-4.0488581703...	Normal
L6	Balb_0	0.0626006732...	-3.9976780167...	Normal
L9	Balb_0	0.0765648306...	-3.7071743350...	Normal
L_B_0_4	Balb_0	0.0643317011...	-3.9583263517...	Normal
L_B_0_5	Balb_0	0.0580391845...	-4.1068289397...	Normal
L1	C57_0	0.0588435991...	-4.0869706977...	Normal
L4	C57_0	0.0989728042...	-3.3368240335...	Normal
L7	C57_0	0.1634543714...	-2.6130401340...	Normal
L_C_0_4	C57_0	0.0715153463...	-3.8056033293...	Normal
L_C_0_5	C57_0	0.0715429261...	-3.8050470622...	Normal
L11	AJ_7	0.0511232879...	-4.2898755660...	Normal
L14	AJ_7	0.0450828855...	-4.4712763318...	Normal
L17	AJ_7	0.0339152136...	-4.8819236088...	Normal
L_A_7_4	AJ_7	0.0435316508...	-4.5217914585...	Normal
L_A_7_5	AJ_7	0.0610160837...	-4.0346666051...	Normal

The graph reports the log 2 of the variance of the diff from the group mean. The graph is divided by vertical lines which seperate one group from another. The distribution of these values would be expected to be a normal distribution. The estimated centre (mean) of this distribution is indicated in white. The LCL, UCL and confidence intervals are also marked. Spots which fall above the upper control lines may give cause for concern. Spots which fall below the lower control lines indicate a very strong resemblance to the mean value.



9.11.3 Maths

Here is an explanation of what values are calculated and reported.

We write *V* for the microarray process quality characteristic. The characteristic *V_k* of the *k* th slide *A_k* is calculated as follows:

Look at the replicate group to which slide *k* belongs. We are interested in knowing how different the slide *k* is from the true values for that condition (spot by spot), and the variance of that difference. We do not have access to the real values of the expression, so we estimate the true expression by averaging the expression of the slides in the group. That is, we calculate the average expression for each spot. Usually we use the **median** for this calculation because this is considered to be more robust than the mean when small numbers of replicates are used. One could alternatively use the mean − an option to do this is included in the plugin. Write *M* for this virtual array, which is the average of all arrays in the group.

M = spotwise average of the arrays in the group.

We then calculate the variance of the difference for each slide, and report this as the quality statistic for that slide:

V_k = mean sqared error of *k*th slide, as measured wrt spotwise groupwise median (or mean).

V_k = 1/(number Spots *−1*) * Sum over all spots *s ((M[s] − A[s])^2))*

These *V_k* are the values which will be reported in the table at the end of the calculation. For 2−colour microarray experiments, the *V_k* are well approximated by a log normal distribution [1]. We also report the *log_2* values of each of the *V_k* and use these to calculate the control lines as follows.

Write *mu* for the mean of the logged *V_k* s and *sigma* for the estimated stdev.

mu = mean over all k (log_2 (V_k))

sigma = stdev over all k (log_2 (V_k))

The Centre Line, Upper control line and lower control line are given by:

CL = mu

*UCL = mu + Phi^(−1) (1−alpha/2) * sigma*

*UCL = mu i Phi^(−1) (1−alpha/2) * sigma*

Where *Phi* is the normal distribution function and *alpha* is the chosen significance level.

If the overall number of slides is small (less than 10) then the user may wish to take into account not only the control lines, but to be aware of the confidence interval around each control line.

The confidence intervals are calculated as follows: Let *alpha* be the overall significance level, as before, *alpha_mu* be the significance level of the estimate of the mean, and *alpha_sigma* be the significance level chosen for the estimate of the standard deviation. Then the confidence interval of the CL is given by:

(mu − Phi^(−1) (1−alpha_mu /2) sigma / sqrt(n), mu + Phi^(−1) (1−alpha_mu /2) sigma / sqrt(n))

while the confidence intervals for the UCL and the LCL are given by

(mu + Phi^(−1) (1−alpha /2) K_1(n, alpha_sigma) sigma, mu + Phi^(−1) (1−alpha /2) K_2(n, alpha_sigma) sigma)

(mu − Phi^(−1) (1−alpha /2) K_2(n, alpha_sigma) sigma, mu − Phi^(−1) (1−alpha /2) K_1(n, alpha_sigma) sigma)

respectively. Here, K_1 and K_2 are given by:

K_1(n, x) = sqrt((n−1)/ ChiSquared(n−1, 1−x/2))

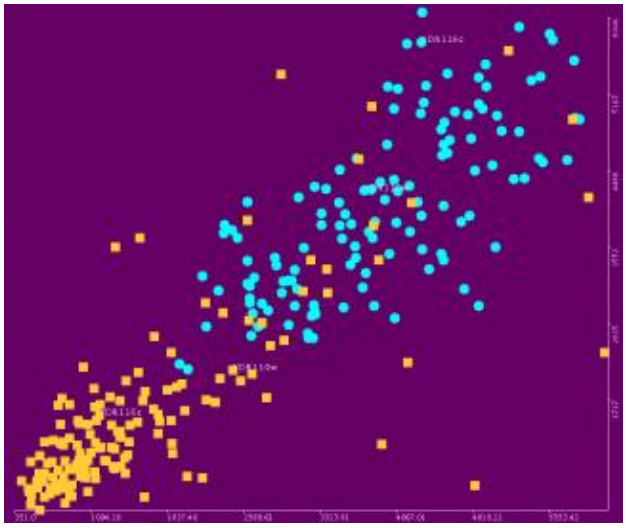
K_2(n, x) = sqrt((n−1)/ ChiSquared(n−1, x/2))

ChiSquared(n,x) being the inverse of the cumulative distribution function with *n* degrees of freedom.

"A New method for statistical control of microarray process quality", Yongxiang Fang et al, preprint available from abrass@cs.man.ac.uk

9.12 Scatter Plot

9.12.1 Overview



Show one or more corelation plots. Each plot shows how all values compare between two Measurements or two Spots. Name labels and cluster data can be overlayed on the plots.

9.12.2 User Interface

9.12.3 Defining the Plot

The panel along the top of the window controls which data is plotted and how it is displayed. The controls are:

7. controls to add and remove plots and change the plot order
   (see below).

6. choose between a pair of Spots or a pair of Measurements

5. select the two Spots or Measurements used for positioning

   Each of the dots in the display are displayed using shapes drawn at positions determined by these two selections.

4. select the two Measurements used for sizing

   The size of shapes drawn for the Spots is determined by these two selections; the height and width of 0.the shape can either be a fixed for all Spots or be scaled by the values of a Measurement (used for example, for displaying error values)

3. a pair of scale controls

   The numbers in these boxes is used to scale the width and height of the shape to a range suitable for display.

2. select the fixed colour and shape.

The shape can be oval, rectangle or diamond and can be filled or hollow. The colour can be fixed for all Spots (using the colour chooing button) or can be picked from any Measurement (using the drop−down list).

1. optionally apply the filter and/or overlay cluster glyphs.

9.12.4 Displaying labels

Click on any dot in the plot window to toggle display of the name label for that dot. The format of the names is controlled with the main display's Display Layout panel.

Buttons in the panel at the bottom of the window to control labels are: **Clear all labels** and **Label all** (actually, all of the Spots or Measurements that are currently displayed in the plot).

9.12.5 Zooming in using the mouse

Press and hold the left mouse button and drag out a *rubber box* to define a zoom region. Release the button to zoom the display into the specified region.

The **Back** button (in the bottom−left corner of the window) moves the display back after a zoom operation. Repeated presses of this button step backwards through the sequence of zooms.

The **Reset view** button adjusts the scale so that all Spots are visible in all plots.

Auto reset view causes the view to rescale itself when changing from one Measurement to another. When this switch is off, the view position and scaling is unchanged when switching between Measurements.

9.12.6 Locating Spots or Measurements

Drag−and−drop can be used to locate particular Spots or Measurements in the plot. Drag a Spot or Measuremenr from the main display and drop it into the *Scatter Plot* window and the name will appear next to it on the plot. Dropping a Cluster onto the panel causes the names of all Spots or Measurement in the cluster to be displayed.

9.12.7 Displaying more than one plot

Scatter Plot can display multiple plots at the same time. The 'add plot' button (represented as a "+") appears to the left of the plot control panel (at the top of the window).

The initial settings for a newly added plot are exactly the same as the existing plot. A new set of controls appears in the panel at the top of the window (the slider may have to be adjusted for these controls to become visible).

When more than one plot is enabled, extra controls appear to the left of each of the plot control panels. These buttons are used to raise and lower a plot in the list (plots are drawn from top−to−bottom) and to delete a plot (the "X" icon).

9.12.8 Controlling the axes

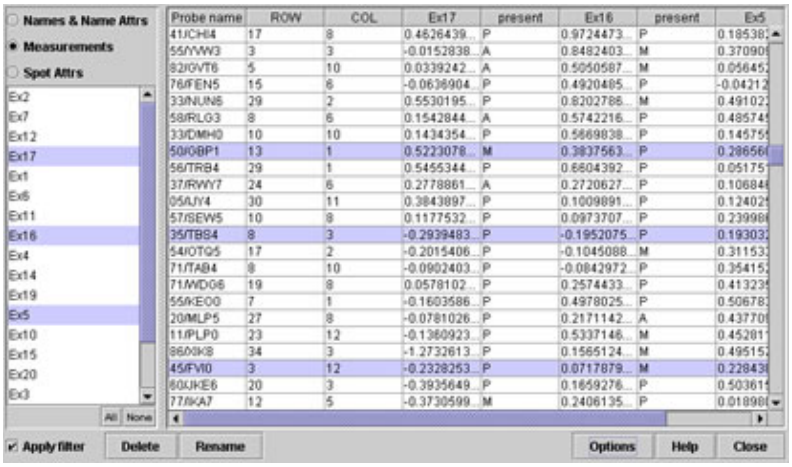
The axis scaling, labelling and title can be controlled via the panel that is displayed when the "Axes" button is pressed. More information about the axis controls can be found here.

9.13 Spot Attributes

9.13.1 Overview

The *Spot Attributes* viewer displays one or more Spot Attributes in conjuction with any combination of Names and Name Attributes

9.13.2 User Interface



One of three possible lists can be displayed on the left−hand side of the panel:

Names & Name Attrs	*a list of all Names and Name Attributes*
Measurements	*a list of all Measurements*
Spot Attrs	*a list of all Spot Attributes (which occur in one or more Measurement)*

The three buttons above the list control which of the lists is currently displayed. The "**All**" and "**None**" buttons select all entries in the current list or none of the entries in the current list respectively.

The right−hand side of the panel contains a table which displays the currently selected data. Nothing will appear in the table until at least one Measurement has been selected.

Any number of Names and Name Attributes can be selected for inclusion in the table.

Any number of Spot Attributes can be selected. A column is added to the table for each of the chosen Spot Attributes are added to the table which occurs in each of the selected Measurements.

The columns in the able can be re−ordered by dragging their headings left or right. The width of individual columns can be changed by dragging the edges of the column heading left or right.

The "**Apply filter**" option can be activated to remove Spots which are trapped by the current filter(s) from the table.

The "**Delete**" button permanently removes the currently selected SpotAttributes from the currently selected Measurements. Any combination of SpotAttributes and Measurements can be used, for example deleting one SpotAttribute from all Measurements, or deleting all SpotAttributes from a single Measurement.

The "**Rename**" button allows the currently selected SpotAttribute to be renamed in the currently selected Measurements. Only one SpotAttribute can be renamed at a time, but the name can be changed in any combination of the Measurements. Renaming a SpotAttribute does not alter the data or the datatype of the SpotAttribute.

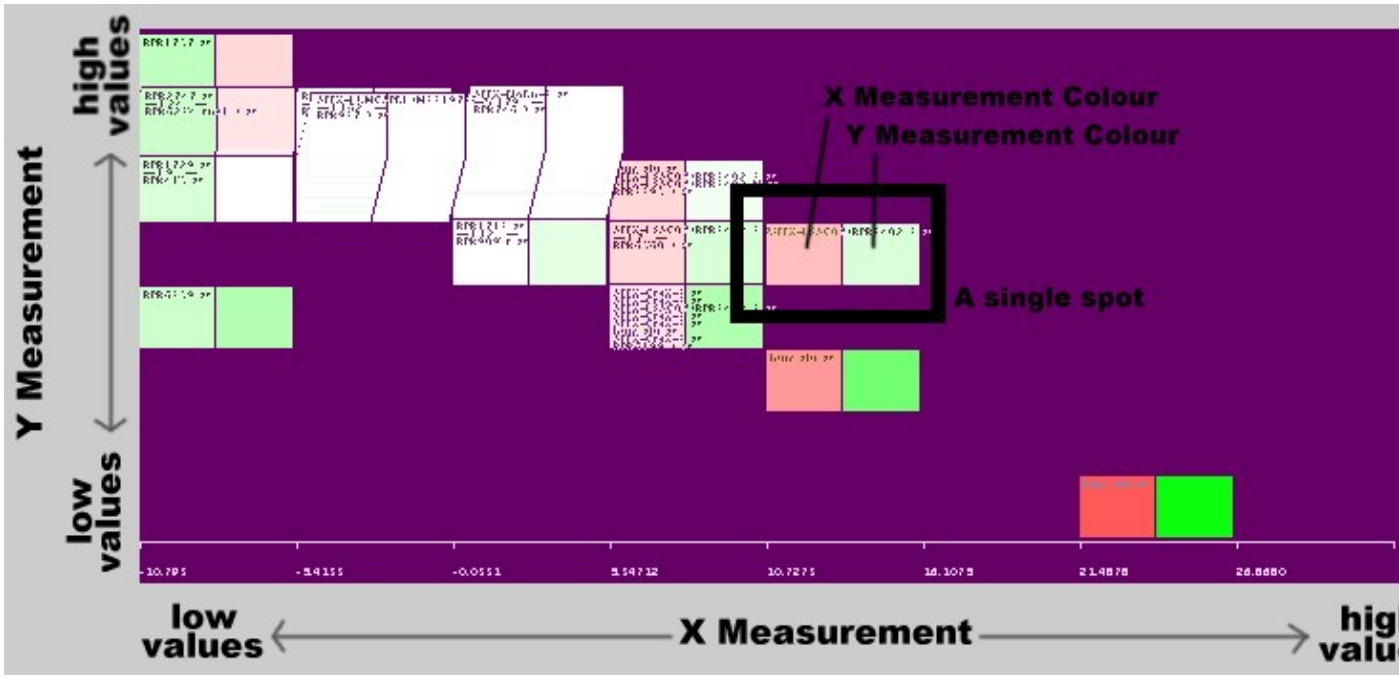
9.13.3 Options

The "**Options**" button opens a dialog box containing a number of switches:

- The "**Synchronise the Spot selection**" option keeps the row selection in the table synchronised with the current Spot selection.
- The "**Synchronise the Measurement selection**" option keeps the selection in the Measurement list synchronised with the current Measurement selection.
- The "**Always display the Measurement values**" option controls whether the expression values of selected Measurements are displayed in the table. When unselected, only Spot Attribute values are displayed.
- The "**Expand SpotAttribute Names**" option determines whether Spot Attribute names are prefixed with the name of the Measurement they belong to.
- When the "**Group SpotAttribute columns together**" option is selected, Spot Attributes columns are grouped together, when this option is not selected then Spot Attributes are grouped with the Measurement they belong to.
- The "**Attempt to remember column layout**" option controls how the order of the columns in the table is maintained when columns are added or removed. When selected, the order is preserved as columms are added or removed, when unselected, the order is defined by the current Measurement order.
- When the "**Attempt to remember column width**" option is selected, the width of columns is preserved as columns are added to or removed from the table. When unselected, column widths are reset whenever the table is altered.

9.14 Stack Plot

9.14.1 Overview



Show the co−relation between two measurements on a 2½D plot

The Stack Plot is so−called because it stacks up multiple spots which fall into the same bin. Bins are created by dividing the range of data in a Measurement into a number of discrete chunks.

9.14.2 User Interface

The configuration of X and Y axes is done by the two sets of controls in the top left corner of the window. The controls (from left to right) for each axis are the same; three drop−down menus:

- Measurement
- Number of bins − choose from a preset selection of sizes
- Boundary mode − either equally filled or equally spaced

A scrollbar on the left−hand side on the window controls the global scale of the stacks. When set to zero, i.e. fully down, the stack plot becomes completely 2D.

A set of checkboxes in the top−right corner control additional features:

Use filter selects between showing all of the data or only showing the data items which pass through the current Filter.

Auto reset view causes the view to rescale itself when changing from one Measurement to another. When this switch is off, the view position and scaling is unchanged when swtching between Measurements.

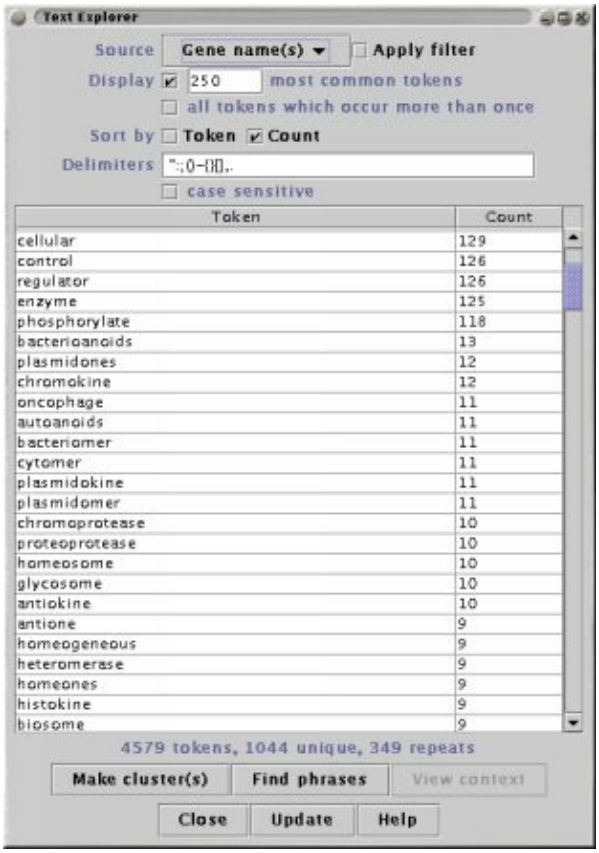
Auto levelise mode updates the bin boundaries to attempt to more evenly distribute the data. This option is only available when both axes are in *equally filled* mode

9.15 Text Explorer

9.15.1 Overview

The *Text Explorer* finds repeated words and word patterns in names and name attributes and in annotation. Spots associated with interesting words can then be tagged in a cluster.

9.15.2 User Interface



Select a "Source" for the text and which "Delimiters" (see below) to use to convert the text into tokens (i.e. words). The resulting frequency counts for the tokens are then displayed in the table.

You can order the table by "Token" (i.e. alphabetically) or by "Count" (i.e. the frequency of the token).

The "Make cluster(s)" button creates a cluster for each of the selected tokens. You can select a token in the table by clicking on it, multiple selections are made using `shift−click' and `cntrl−click'.

The "Find phrases" button opens a Phrase Viewer panel for each of the selected tokens.

9.15.3 Phrase Viewer

The *Phrase Display* portion of the *Text Explorer* shows how one fixed token occurs in conjunction with all other tokens.

The fixed token is determined when the *Phrase Display* is created. The tokenised text is then scanned for tokens which occur near the fixed token. The resulting frequency count for pairs of tokens is then displayed in a table.

Example:

If there are three text strings being explored:

```
this is the first piece of text

and this is the second piece of text

and this is the third piece of text
```

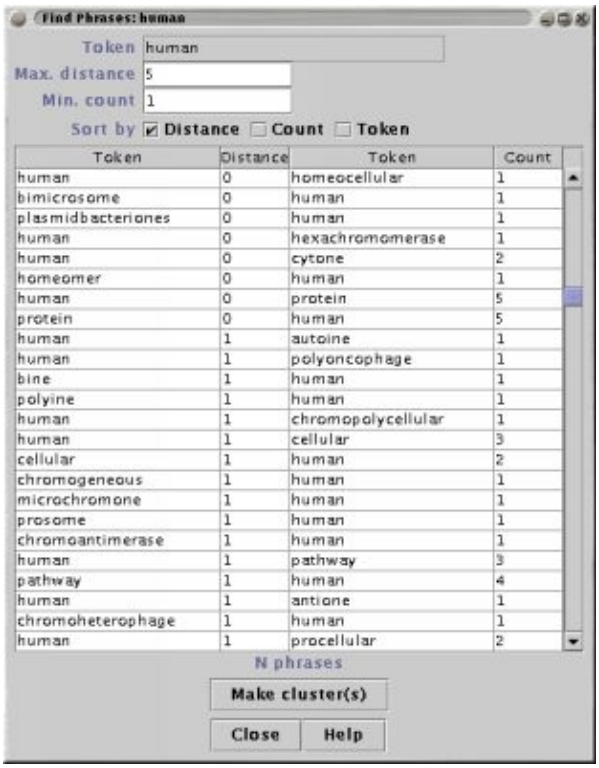
and the token "piece" has been selected, then the 'phrase' table would contain these entries:

```
second    0    piece     1
first     0    piece     1
third     0    piece     1
piece     0    of        3
the       1    piece     3
piece     1    text      3
is        2    piece     3
this      3    piece     3
and       4    piece     2
```

The first and third columns show the 'phrase' (i.e. pair of tokens) that has been found. The first two phrases in this example are "second piece" and "first piece".

The second column is the "Distance", i.e. the number of other tokens between the tokens in the phrase. For this example the distance of the first phrase is 0 meaning that "second" and "piece" occur next to one another. The distance of the last phrase, made from the tokens "and" and "piece", is 4. This means that there are 4 other tokens between "and" and "piece" in occurences of this phrase.

The fourth column in the table is the "Count" of times that the phrase has been found in the text. In the example data, the final phrase "and X X X X piece" (i.e. the tokens "and" and "piece" seperated by 4 other tokens) occurs 2 times.



The "Max. distance" setting determines how far apart two tokens can be in a phrase. Larger values increase the data processing time considerably.

The "Min. count" setting determines the minimum number of times that a phrase must occur to be displayed in the result.

You can order the table by "Distance" (between the two tokens), "Token" (i.e. alphabetically) or by "Count" (i.e. the frequency of the phrase).

The "Make cluster(s)" button creates a cluster for each of the selected phrases. You can select a phrase the table by clicking on it, multiple selections are made using `shift−click' and `cntrl−click'.

9.15.4 Delimiters

Chunks of text are converted into 'tokens' by splitting the text every time one or more of the delimiter characters are seen.

The characters to use for finding tokens are specified in the "Delimiters" string. You can customise the "**Delimiters**" to suit the format of the text you have.

The delimeters string should contain the sort of non−letter characters that are used between words, for example ' ' (space) ',' (comma) and '"' (quote).

The special characters '`\t`' and '`\n`' are used to refer to TAB and RETURN respectively.

Examples:

using the delimiters:
  " ,:" (i.e. a space, a comma and a colon)
the string:
  "one:red, two:green,three:blue"
is converted into six words:
  'one', 'red', 'two', 'green', 'three', 'blue'

using the delimiters:
  " ," (i.e. just a space and a comma)
the same string:
  "one:red, two:green,three:blue"
is converted into three words:
  'one:red', 'two:green', 'three:blue'

using the delimiters:
  "'−()," (a single quote, a dash, and open and close parentheses)
the string:
  "sample−213 'test' marker at 5'end (ignore)"
is converted into eight words:
  'sample', '213','test', 'marker', 'at', '5','end', 'ignore'

9.16 Web Plot

9.16.1 Overview

Web Plot presents a view in the data with all visible Measurements projected onto parallel or radial axes. The purpose of this plot is to show the overall distribution of many Measurements simultaneously, and as a way of visualising the results of multi−dimensional clustering.

9.16.2 User Interface

Parallel axes toggles between radial and parallel axis layout.

Apply filter selects between showing all of the data or only showing the data items which pass through the current Filter.

Uniform scaling forces each axis to use the same scale, otherwise scales are adjusted for each axis to improve visibilty of the data

Mouse tracking mode draws all of the lines in grey, except for those which pass under the current mouse position. In addition, the name or name attribute associated with lines near the mouse are displayed. Mouse tracking only occurs when the mouse is over one of the spokes, not the space inbetween spokes.

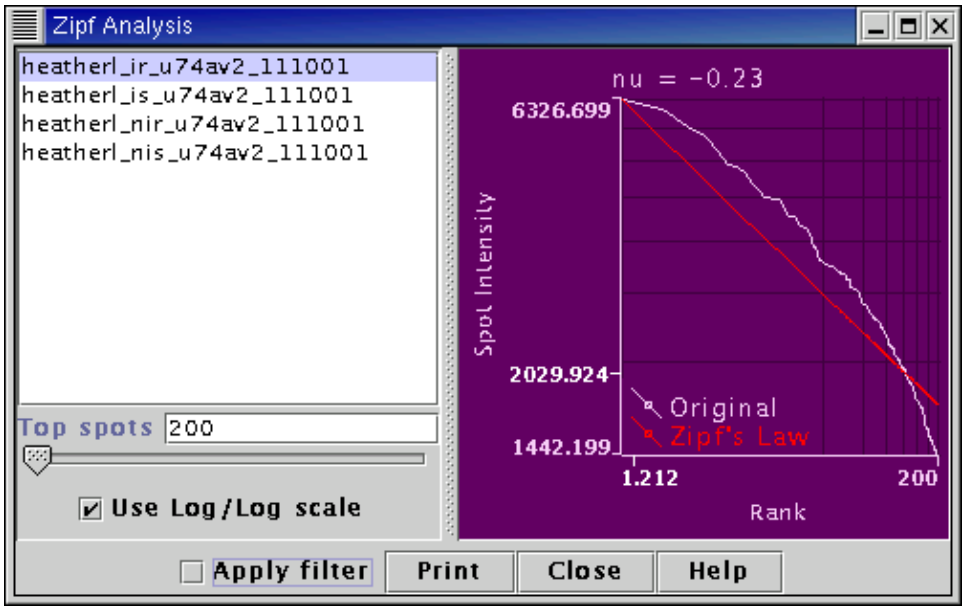
9.17 Zipf Analyser

9.17.1 Overview

Zipf's law (G.K. Zipf (1936) *The Psycho−biology of Language: An Introduction to Dynamic Philology*; G.K. Zipf (1949) *Human Behaviour and the Principle of Least Effort*) is concerned with the frequency of occurrence of objects, or with the size of objects, and how those frequencies/sizes decrease with rank r (from the highest r = 1 to the lowest). For example the frequency of occurrence of words within large pieces of text, or the sizes of cities both follow Zipf's law.

It has been noted recently (D.C. Hoyle et al, *Bioinformatics* **18** (2002), to appear) that the highest spot intensities from a microarray experiment (when ranked in order) follow Zipf's law very closely. This application plots the ordered spot intensities against rank r. When log(spot intensity) is plotted against log(rank) the highest spot intensities should follow a straight line. A Zipf's law exponent, represented by the Greek letter nu, is calculated.

9.17.2 User Interface



Select a Measurement from the list on the left−hand side of the panel. The plot of intensity against rank is displayed on the right−hand side.

The Zipf's law exponent (the *nu* value) is displayed at the top of the graph.

The slider at the bottom of the panel selects how many of the top ranking Spots will be used for the display.

A Log/Log scale can be selected as required.

9.17.3 Plugin Commands

(no plugin commands available)

10 Annotation Loader

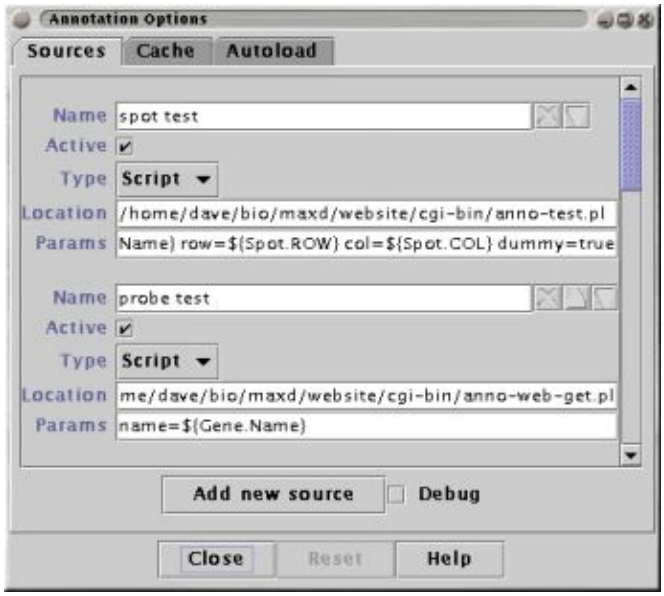
10.1 Overview

Annotation refers to any text which is associated with a Probe or Gene name. Annotation for a particular entity can be gathered from multiple sources (i.e databases like EMBL and SwissProt) and stored locally to reduce data loading times.

Annotation is displayed in the Annotation Viewer which is activated when you select "Display Annotation" in the main window's popup menu. The viewer also allows you to access the annotation loading options.

maxdView caches the annotation data as it is first loaded, this makes subsequent accesses faster. If you need to reload the data (because you have changed the data source for example) you have to empty the cache manually using the button in the loading options panel (see below).

10.2 Sources



The sources tab in the loader options panel

You can change the order in which the sources are searched using the up and down arrows to the right of each source. The delete button (immediately to the right of the source name) permanently removes the source. To temporarily disable a source, uncheck the "Active" checkbox below the source's name.

Three type of data source are currently supported; Script, URL and File. Any number and mixture of these source types can be used to retrieve annotation for a given spot.

Each source is has the following controls:

- **Name**: how the source will be titled displayed in the Annotation Viewer
- **Active**: whether the source should be used
- **Type**: sources can be 'Script', 'URL' or 'File' (see below)
- **Location** where to find the script, URL or directory for this source
- **Parameters**: the arguments to be passed to the source

The 'name' line is unimportant in the data gathering purpose − it is only used to refer the source. The use of the **Location** and **Parameters** controls depends on the type of the source and is explained in the sections below.

Script sources are programs which can be invoked to generate annotation

The full path and name of the script should be specified in the sources **Location** control, for example "`/home/people/dave/bin/srs−wrap.pl`".

Parameters are passed to the script using **variables** which refer to the names or name attributes of the spot. All variable names begin with a '$' (dollar) character and are enclosed in curly braces '{}'. For example, the probe name associated with a spot is called "`${Probe.Name}`". Any attribute that a name has can be used as a variable, for example "`${Gene.ACC_NO}`" and "`${Spot.ROW}`".

Any text in the argument list that is not recognised as a variable name is passed directly to the script. At least one variable must appear in the **Parameters** control for the source to be considered valid. Names that appear to be variable names but which cannot be matched with real attribute names will cause an error message to be displayed.

Example: Calling the `wgetz` utility (from the SRS package)

In the source's **Location** control specify the full path and name of the script:
`/home/SRS/bin/wgetz`.

In the source's **Parameters** control specify the command−line arguments to pass to the script:
`−e "[embl−all:${Gene.Name} acc:desc]"`

When the a request is made to the source, the `${Gene.Name}` part of the argument list will be replaced with the actual gene name and the script will be invoked. If, for example, the gene is called 'Y1234' then the following command will be executed:

```
/home/SRS/bin/wgetz −e "[embl−all:Y1234 acc:desc]"
```

Any text that the command produces (on either it's standard or error output streams) will be captured.

URL sources represent a web servers which generate annotation in response to CGI requests.

Note: The HTTP POST method is used to send the parameters. This will not work with URL handling scripts that use the GET method.

CGI is the mechanism used to handle the forms commonly found on web sites. A request is sent to the URL named in the source's **Location** field along with the data in the **Parameter** control.

Variables are used in exactly the same way as with script sources, see above for details.

The **Parameter**s must be in the correct *URL encoded* form. The encoding is the same as that used for specifying CGI parameters within a URL, for example:

```
http://www.demo.org/cgi−bin/demo.pl?arg1=val1NOBR>
```

An easy way to discover which parameters to use and how to encode them is to use a web browser to access the annotation source 'manually'. The URL location displayed in the browser will show you how the **Parameters** should be defined.

You do not need to specify the '?' character that separates the URL's location from it's parameters. For example, to access the above URL the source's **Location** would be:

```
http://www.demo.org/cgi−bin/demo.pl
```

and the parameters would be

```
search=all
```

Aside: Using a proxy server:

If your internet connection requires the use of a **Proxy Server** you must inform the Java environment when you start **maxdView**. The following command−line parameters are used:

```
java −Dhttp.proxyHost=proxyhost
     [−Dhttp.proxyPort=portNumber] −classpath ....
```

Aside: basic rules for encoding URL parameters:

- Each parameter should be separated by an ampersand ('character
- Spaces should be replaced by plus ('+') characters
- Any non−alphanumeric character should be represented as a 3−character string "%xy", where xy is the two−digit hexadecimal representation of the lower 8−bits of the character.

File sources are directories on a local filestore that contain annotation text.

Annotation is looked for in the directory specified in the source's **Location**. The file name will be generated using the source's **Parameters** value.

If the path separator ('/' on Unix, '\' in MS−Windows) or a non−alphahumeric character occurs in a value specified for the file name, it is replaced with an underscore ('_') before the file is searched for.

Example: Assume you have created files with names based on the position of a spot on it's array. Information about the spot at row X and column Y is stored in a file called "`spot_X_Y.dat`". A source to load this information would have it's **Parameter** set to:

```
spot_${Spot.ROW}_${Spot.COL}.dat
```

(this assume that you have created Spot name attributes called "ROW" and "COL" and populated them with values).

Note: having tens of thousands of files in the same directory is likely to make your file system very unhappy.

10.3 Caching

As data is loaded by a source it will be cached in memory. Next time the source is requested to load the same annotation, the data will be retrieved from the cache rather than it's real location.

Data is cached on a per−source basis, that is, each source stores it's data separately. If you change either the source's **Location** or **Parameters**, data cached for the old location or parameters will be ignored and the new data will be cached.

The entire cache can be written to and read from a file. This allows you to retrieve annotation one time only and save it between runs of **maxdVIew**.

The cache is controlled by a tab in the loader options panel, open this panel using the "Options" button in the *Annotation Viewer*.

The controls are:

- **Empty** the cache of all data
- **Save** the cache to a file
- **Load** the cache from a file

A panel also displays some statistics about the amount of data currently in the cache. The "Hit Rate" shows how effective the cache is being.

10.4 Autoloading

The autoload feature attempts to load the annotation for all spots using threads running in the background. You can use this to retrieve annotation for a large number of spots as a one−off process and then save the cache (see above) for later use.

Autoload generates a load requests for all of the spots and the background threads consume these load requests one at a time. The status label informs you how many load requests are currently pending and how many threads are currently active.

See Also:

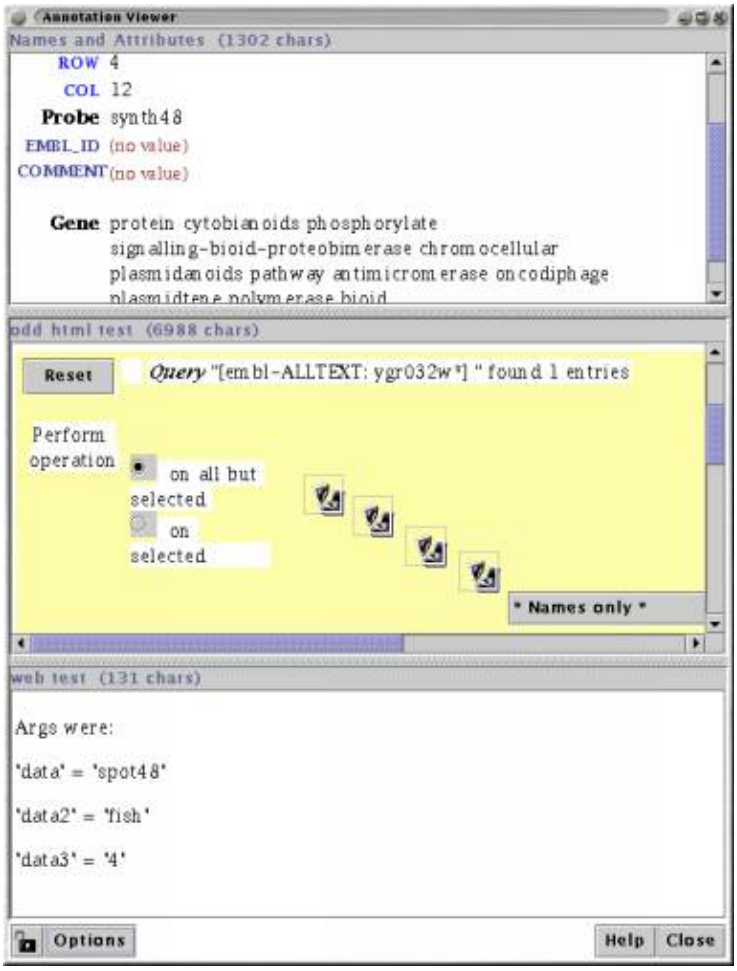
- Annotation Viewer
- Overview
- File formats

11 Annotation Viewer

The *Annotation Viewer* displays the textual annotation associated with a particular Probe or Gene name. The viewer is created whenever you click on a spot or name in the main window, or when a plugin class requests that some annotation be displayed.

The viewer contains one panel for each of the active sources. The panels can be resized by dragging the horizontal bar that divides them.

11.0.1 User Interface



The label at the top of each panel shows the source of that part of the annotation. Buttons at the bottom of the window (from left to right) lock or unlock the current view, open the loading options panel, display this help page, and close the window.

Locking...

By default, requests for annotation cause the text to be displayed in the current *Annotation Viewer*. If you want to compare two or more sets of annotation, "Lock" the current viewer by clicking on the padlock icon in the bottom left corner. The next request for annotation will cause a new window to be opened. You can have as many windows as you want open. If you unlock one of the locked windows, it will be used next time an annotation reqest is made.

Caching...

Annotation data is cached (saved in memory) when it is first loaded to make subsequent accesses faster. The operation of the cache is controlled in the Annotation Loader's options panel.

If you need to reload the data (if you have changed the data source for example) use the "Reload" option on the popup menu (you can empty the whole cache using the button in the loading options panel).

Data Sources...

Annotation from different sources is presented in the order that it is loaded. The data sources are selected in the Annotation Loader's options panel.

See Also:

- Annotation Loader
- Name Tag Editor
- Spots (and Names)
- RegExp Filter
- Text Explorer
- Just−o−Clust

12 Name and Attributes

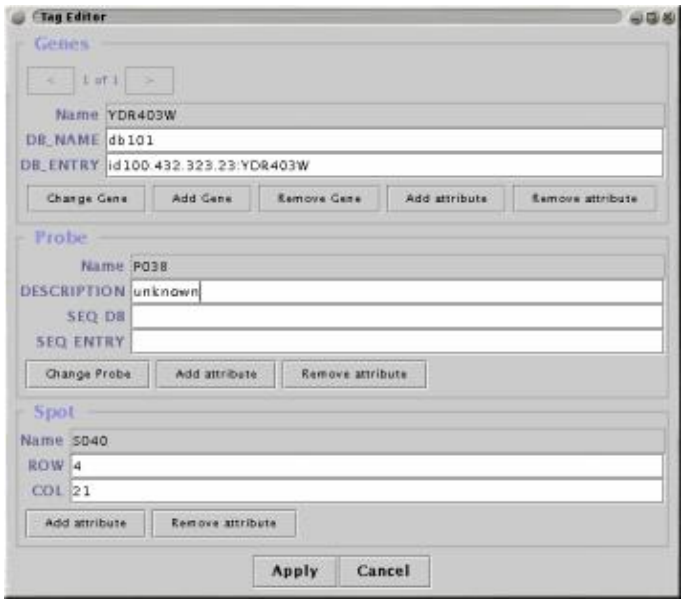
Every **Spot** in the data has a **Spot name** an optional **Probe name** and zero or more **Gene names**.

Each of these three types of name can have zero or more **Name attributes** which are extra fields in which you store data associated with the names.

An example might be "`ROW`" and "`COLUMN`" attributes for **Spot name**. Every spot can have a value for each of these attributes and these values can be used in searching and filtering operations. The Name Munger plugin can be used to copy, translate, load and save both names and their attributes. The editor can be used to alter Name Attributes by hand.

Name attributes are useful for storing alternate names or identifiers for **Gene** and **Probe** names. You might store "`EMBL_ID`" and "`ACCESSION No.`" for example. Attributes can also be used to store descriptive information such comments, functional categories, homologue families and so on.

12.1 The Name and Attribute Editor



Genes

When the Probe is linked to more than more Gene, use the left and right arrows to select between Genes.

Change Gene	alter the current gene, enter a new name, or pick from existing names
Add Gene	enter a new name, or pick from existing names
Remove Gene	removes the current gene
Add attribute	adds a new Name Attribute (to all Gene Names)
Remove attribute	removes a Name Attribute (from all Gene Names)

Probe

Change Probe	enter a new name, or pick from existing names
Add attribute	adds a new Name Attribute (to all Probe Names)
Remove attribute	removes a Name Attribute (from all Probe Names)

Spot

Add attribute	adds a new Name Attribute (to all Spot Names)
Remove attribute	removes a Name Attribute (from all Spot Names)

13 Database Connection

This panel is displayed when a database connection is being made to a **maxdSQL** database.

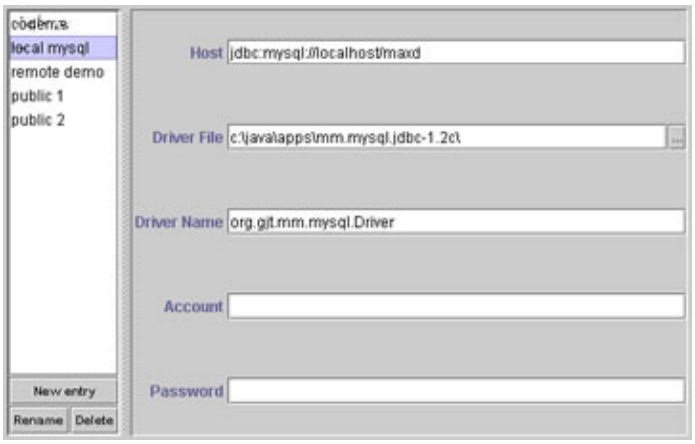
maxdView uses a language called JDBC when talking to a relational database hosting the **maxdSQL** database.

Each database provider supplies a *JDBC driver* for their database which provides the interface between the application (i.e. **maxdView**) and the underlying database engine. A specific driver is needed for each different database that **maxdView** wishes to connect to. JDBC drivers are often supplied with the database, but sometimes they have to be downloaded separately from the manufacturer's website.

13.1 Connection

The five fields on the right hand side on the panel must be completed to establish the connection.

Once the details have been correctly defined, they can be saved for future connections.



Host....

The format for the host name depends on what RDBMS system is being used, consult the database's JDBC documentation for full details.

Some common formats are:

Server	*Using the standard port*	*Using a non−standard port*
PostgreSQL	`jdbc:postgresql://HOST/DATABASE`	`jdbc:postgresql://HOST:PORT/DATABASE`
Oracle	`jdbc:oracle:thin:@HOST:DATABASE`	`jdbc:oracle:thin:@HOST:PORT:DATABASE`
MySQL	`jdbc:mysql://HOST/DATABASE`	`jdbc:mysql://HOST:PORT/DATABASE`

where `HOST` is the fully qualified domain name of the machine running the database (e.g. myhost.man.ac.uk), `PORT` is the TCP/IP port the database server is listening to, and `DATABASE` is the actual name of the database holding the **maxdSQL** tables.

If the database is running on the same machine as **maxdView**, the `HOST` part of the name does not have to be supplied.

Normally, the PORT part of the name can be omitted unless the database is listening on a non−standard port for some reason.

Driver File....

The "Driver File" field is specifies the location of the JDBC driver.

This driver is supplied by the database vendor, and does not come as part of the **maxd** software. Consult the database administrator, or the appropriate vendor's documentation (or more usefully, their web−sites) for information about specific JDBC drivers.

The driver usually comes either as a single JAR file, or in the form of a *package*, which looks like a set of nested directories. In either case, the location of the driver should be specified in the "Driver File" field.

- **If the driver is in a JAR file**: specify the full path and file name of this JAR file, for example; `C:\Oracle\drivers\JDBC_Driver_12.jar`
- **If the driver is a package**: specify the top−level directory of the package, for example; `/opt/mysql/jdbc/`

Driver Name....

The "Driver Name" field is where you specify the name of the **class** that will provide the JDBC functionality.

Some common driver names:

Server	*Driver Name*
PostgreSQL	`org.postgresql.Driver`
Oracle	`oracle.jdbc.driver.OracleDriver`
MySQL	`org.gjt.mm.mysql.Driver`

Account & Password....

Some databases will require a valid account name and password, others will allow anonymous login. This will depend on how the database administrator has configured the system. Note that the database account name and password are not necessarily the same as those used to login to the operating system.

If there is some problem with the connection attempt, i.e. the host name is not found, or account/password pair are illegal, an error message will be displayed. As this message is generated by the JDBC driver, The format of this message depends on the system, but should indicate what caused the connection to fail.

13.2 Connection problems?

In the event of connection problems, try the following steps to identify the cause:

- *Can't find the driver?*

  If a dialog box appears containing the message "No suitable driver" every time you press "Connect", then double−check each of these things:

  1. The driver's JAR file or directory path must be correctly listed in the `−classpath` part of the command used to run the application.
  2. The protocol name of the database (the word(s) after `jdbc:`, and before the hostname) must match exactly the format expected by the driver.
  3. The "Driver" field must exactly match the full package name of the driver class.

- *Can't find the database?*

Make sure the HOSTNAME part of the "Host" field is correct. Check whether the **maxdSQL** database can be accessed using any other tools.

- *Unable to log in?*

  Double−check the account/password details. Try to log into the database server using a different tool (such as the SQL console or a database administration tool).

13.3 Saving details

Connection details can be saved in a list. This list is displayed on the left−hand side of the panel.

To create a new entry in the list, press the "**New entry**" button and provide a descriptive name for the entry. The current details (i.e. the "Host", "Driver File" and other fields) will be stored in the new entry.

When an entry is selected in the list, the details (can be edited, and the updated details will permanently replace the existing details.

The "**Rename**" button allows the descriptive name of the currently selected entry to be changed.

The "**Delete**" buttons removes the currently selected entry from the list.

Note that passwords are never saved in the connection details list, and must be provided whenever required by the database.

14 Merging Data

All plugins which import data offer the choice of "Replace" the current data, or "Merge" with the current data. When "Merge" is used, the *Merge control panel* is displayed once the data is loaded:

This panel shows how the names in the new data match with the names in the current data. Each of the three types of name (Spot, Probe and Gene) are matched separately.

The *Merge control panel* shows how many of names match for each of the types. One of the three type must be selected.

If some of the names in the new data do not match existing names, the choice between "generate new spots" or "ignore unmatched spots" is offered. One of these options must be chosen before the merge can proceed.

(Note: The Name Munger is useful for altering names so that merging can be performed.)

How merging works...

If the current data has five spots, named like this:

```
spot1
spot2
spot3
spot4
spot5
```

and the new data contains three spots, named like this:

```
spot1
spot3
spot5
```

then the merge is simple because all spots are recognised. No new spots will be created, and Spots "`spot2`" and "`spot4`" in the new Measurements will be given the value "NaN" (Not−A−Number).

If the new data instead contained the four spots:

```
spot1
spot3
spot5
spot7
```

then the merge more complex because "`spot7`" is not recognised. If the "ignore" option has been selected, "`spot7`" will be discarded, otherwise a new Spot called "`spot7`" will be created. In the existing Measurments, "`spot7`" will be given the value "NaN". As before, the value for "`spot2`" and "`spot4`" will be set to "NaN" in the new Measurements.

This rule is applied to Spot, Probe and Gene name matching. When matching using Probe or Gene names, an additional rule is used to handle repeated occurences of names.

If the current data has five spots, with Probe names like this:

```
probeA
probeB
probeC
probeA
probeB
```

(note that "probeA" occurs twice)
and the new data contains four spots, with the Probe names:

```
probeA
probeA
probeA
probeD
```

then the first occurence of "probeA" in the new data will be matched with the first occurence of "probeA" in the existing data. Similarly, the second occurence of "probeA" in the new data is matched with the second occurence of "probeA" in the existing data. The third occurence of "probeA" does not match however, as there are no more "probeA"s in the existing data. This "probeA" will either be added as a new spot, or ignored depending on which merge option was selected (see above). Likewise, "probeD" does not match any existing name, it too will either become a new spot, or be ignored.

See Also:

- Name Munger
- Read Native
- Load Plain Text
- Database Loader

15 maxdView ISYS Integration

This document describes how **maxdView** can run as a client of the NCGR's ISYS integration system (see http://www.ncgr.org/isys/ for full details of ISYS).

Data tranfer is based on collections of one or more Spots. **maxdView** can send any combination of Names and Name Attributes to other ISYS components.

If some data has been sent from **maxdView** to other ISYS viewer services then **maxdView** is said to be synchronised with the components providing those services. When viewers are synchronised, ISYS keeps the data selection the same in each. Altering the selection in one component will have a corresponding effect in all other synchronised components.

The ISYS "hide and show" facility is integrated with **maxdView** via the filter mechanism. If one or more filters are enabled in **maxdView**, hide and show events will be sent to other ISYS components sharing the same data. Similarly, when Spots are hidden by an ISYS component, a corresponding filter is installed in **maxdView**. See below for more details.

15.1 Packaging data to send to ISYS

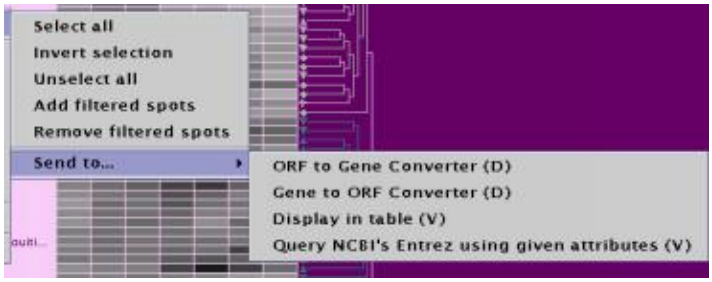
ISYS is a centralised broker that enables communication between two or more individual components (such as **maxdView**).

ISYS provides a feature called Dynamic Discovery. This allows clients (in this case **maxdView**) to find out about *services* offered by other ISYS components.

Two types of service are supported; *viewer services* which provide a visualisation capability and *data services* which perform data manipulation. **maxdView** can send Spots to either type of service (and optionally capture the output from data services, see below).

Communication to ISYS is via the current Spot selection. The ISYS Dynamic Discovery mechanism is invoked whenever the the selection is changed.

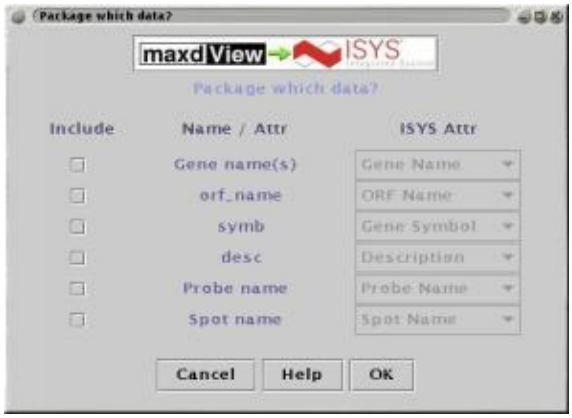
Each available service will be represented in the "Send to" submenu on the Selection popup menu.



15.1.1 Defining the "Data Packer" mapping

Before **maxdView** can talk to ISYS, a common terminology must be agreed on. The **maxdView** Spot data must be converted into the format supported by ISYS. You must specify how ISYS should interpret each of **maxdView**'s Names

and Name Attributes. This is done using the "Data Packer" dialog box:



The dialog box contains entries for all of the Names and Name Attributes in the current **maxdView** data. The "Include" check box on the left of each line controls whether the Name or Name Attribute will be included in the data sent to ISYS.

Each of the Names or Name Attributes can be linked to one ISYS Attribute. More than one Name or Name Attribute can be linked to the same ISYS Attribute.

The mapping specified in this dialog remains in force after the dialog box is closed. The dialog will not be redisplayed unless Name Attributes are added or removed from the **maxdView** data. If this has happened, then the dialog box will be redisplayed before any more data is sent to ISYS.

For best results, always include the **maxdView** `Spot Name`s. As they are guaranteed to be unique, they are ideal for subsequent matching.

15.1.2 ISYS Object model

When sending data, **maxdView** generates one ISYS Object for each Spot. The Name and Name Attributes specified as included in the mapping are stored as ISYS Attributes within the ISYS Object.

15.2 Receiving and matching data from ISYS

When **maxdView** receives data from ISYS, a *mapping* is needed to specify how the ISYS Attributes are converted into **maxdView**'s Names and Name Attributes. This mapping is normally the inverse of the mapping used to convert from **maxdView** to ISYS (see above).

Each time a new ISYS Attribute is encountered, the "Data Matcher" dialog box will be displayed:

Each of the possible ISYS Attributes can be matched with one of the Names and Name Attributes in the current **maxdView** data.

Use the dropdown menus to select which Name or Name Attribute to compare each ISYS Attribute with. The "Ignore" option can be used to cause **maxdView** to disregard certain ISYS Attributes that are not of interest.

When **maxdView** receives one or more ISYS Objects, it uses the mapping to determine how the ISYS Attributes in the received data are related to the Names and Name Attributes in the data it already has.

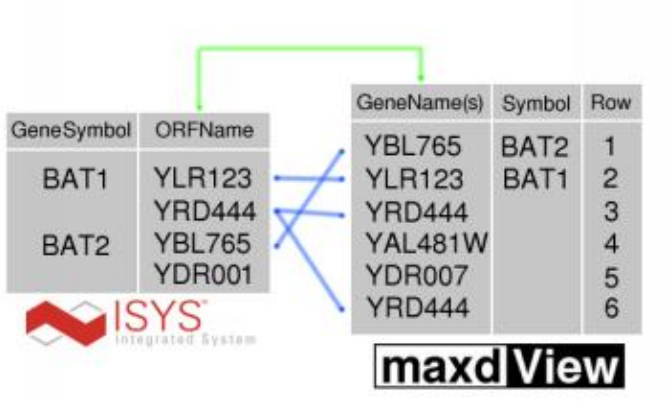
Given a suitable mapping, the incoming data can be matched with Spots in the existing data. The values of the ISYS Attributes can be compared with values in the **maxdView** Names and Attributes to determine which ISYS Objects correspond with which Spots.

Use the checkboxes on the right of each line to select which ISYS Attribute to use for matching. You may select more than one attribute as candidates for matching. When more than one match option exists, **maxdView** will choose the first one in the list that is present in the data that it receives.

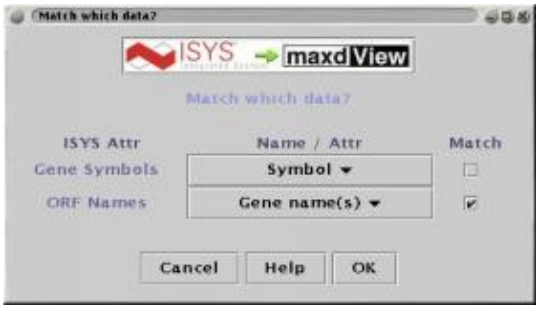
For best results, use `Spot Name`s for matching as they are guaranteed to be always be present and unique.

15.2.1 Example One

The following example shows how some ISYS format data (in the table on the left) can be matched with the same data in **maxdView**.



Using the "Data Matcher" dialog, the ISYS Attribute "ORFName" has been mapped to **maxdView**'s "Gene Name(s)" and selected to use for matching.

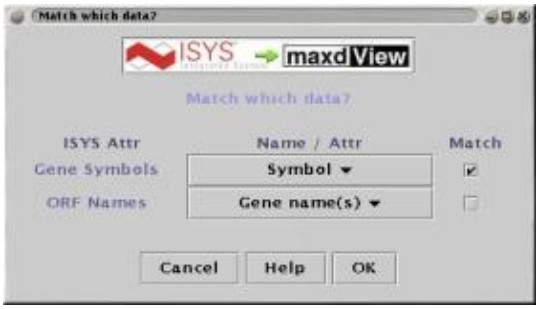


Three of the four ISYS objects can be matched with Spots by comparing "ORFName" with "Gene Name(s)". These matches are shown by the blue lines.

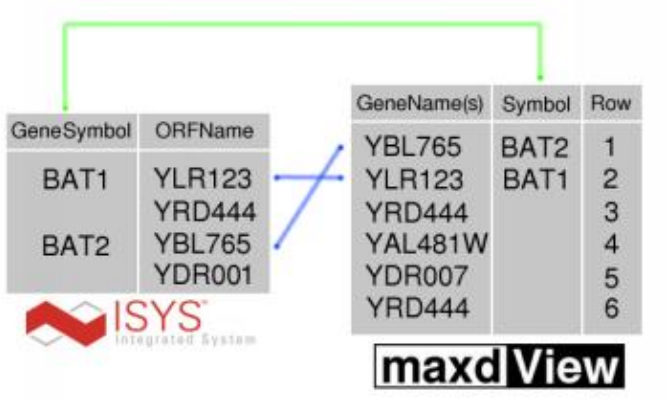
Note that one object (`YRD444`) matches with two Spots in the **maxdView** data. Matches of the form "one−to−many" are handled automatically.

15.2.2 Example Two

In this second example, the ISYS Attribute "GeneSymbol" has been mapped to a **maxdView** Name Attribute called "Symbol". This mapping has then been selected for use in matching.



The follow picture shows the reduced level of matching that is achieved:



Two of the four ISYS objects can be matched with Spots by comparing "GeneSymbol" with "Symbol". The YRD444 object (which previously matched two **maxdView** Spots) is ignored because it has no "GeneSymbol".

15.3 Show and Hide

15.3.1 Sending show/hide events

When one or more **maxdView** filters are enabled, ISYS "show" and "hide" events are sent to any synchronised components. These events will cause the components to hide ISYS Objects corresponding to filtered Spots.

The mapping specified in the "Data Packer" dialog (see above) is for converting **maxdView** Spots into ISYS Objects.

15.3.2 Receiving show/hide events

When **maxdView** notices that a synchronised component has "hidden" one or more objects, it uses the matching process to convert the objects to Spots, and then installs a filter which hides those Spots.

The 'ISYS' filter is identical to other **maxdView** filters except tht it has no user interface. The filter is controlled automatically in response to ISYS "hide" and "show" events. When there are no hidden objects the filter is removed. The "Filter" menu in the main display will indicate the percentage of data that is currently hidden.

Note: Sending and receiving of show and hide events can be disabled using the ISYS Options panel.

15.4 Data Capture

maxdView can 'capture' data that is received from ISYS data services.

Normally the result of calling an ISYS data service is sent to another ISYS component. When *capture mode* is enabled the values of the ISYS Attributes can be stored in **maxdView** Names and Name Attributes.

This feature enables useful information provided by ISYS components (such as the "ORF to Gene Converter") to be added to existing **maxdView** data.

Capture mode is enabled using the "Allow capture" checkbox in the ISYS Options Panel. In capture mode, a variant of the "Data Matcher" dialog is displayed when data is received from an ISYS data service:

This "Capture Data" dialog has the same mapping controls as the "Data Matcher" dialog (see above).

The "Capture" button at the bottom of the dialog starts the capture process. Once the ISYS Objects have been matched with **maxdView** Spots, each ISYS Attribute that is not mapped to "Ignore" is examined for 'new' values. 'New' refers to values not present in the **maxdView** Name or Name Attribute that the ISYS Attribute has been mapped to. Any new or different values in the ISYS data are stored in the **maxdView** data.

The "Capture Data" dialog also contains a dropdown menu with a list of possible ISYS services. Beside the menu is a "Send" button. You can use these controls to send the data directly back to ISYS without capturing any of it.

Note: Capture mode is disabled by default because it is possible to set up mappings that cause data to become "scambled". Not all ISYS components use the same "one spot per object" model as **maxdView**. Capturing data from components such as "ORF to Gene Converter" works correctly, but you are advised to use this feature on other components with care.

15.5 The ISYS Options Panel

The options panel is accessed using the "ISYS Options" entry on the popup menu in the main display.

*(This entry is only available when **maxdView** is running as an ISYS client.)*

- **Packing**

  The "Edit mapping" button opens the "Data Packer" mapping dialog box.

  By default, ISYS objects are cached were possible. If you are running short of memory, caching can be disabled using the "Enable caching" checkbox.

- **Matching**

  The "Edit mapping" button opens the "Data Matcher" mapping dialog box.

  The Capture mode can be enabled using the "Allow capture" checkbox.

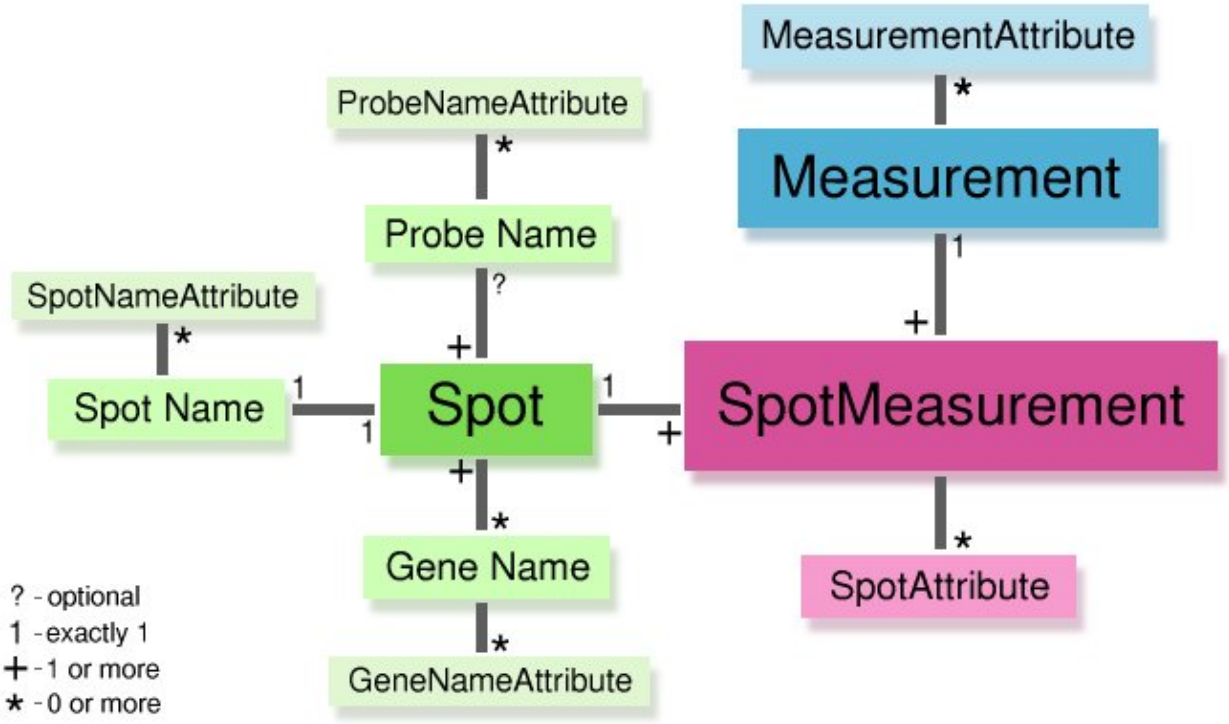
- **Show/Hide**

  The sending and receiving of "show" and "hide" events can be independantly controlled by these two checkboxes.

- **Selection**

  The "Scroll display" option makes the main display scroll to show each new Spot as they are selected.

16 The maxdView Data Model



The links between elements show how they are connected, the decorations on the links show many of each element can be linked.

For example, a "SpotName" can be linked to 0 or more "SpotNameAttribute"s but a "SpotName" must be linked to exactly one "Spot". Similarly, every "Spot" much have a "SpotName" but can have zero or more "GeneName"s.

(note that "SpotAttribute" should really be called "SpotMeasurementAttribute" but it isn't...)

See Also:

- Concepts
- Overview
- Method Reference
- Programmer's Guide

17 File Formats

17.1 Native

The native file format is written in XML syntax.

Documents are structured as follows:

```
<MeasurementGroup>

 <ArrayType NAME="atn1">

  <Probe NAME="pn1" >
   <Gene NAME="gn1" />
  </Probe>

  <Spot NAME="sn1" PROBE="pn1" />

 </ArrayType>

 <Environment>
 </Environment>

 <ApplicationProperties>
 </ApplicationProperties>

 <Measurement NAME="mn1" DATATYPE="dt">

  <Attribute NAME="att" VALUE="val" />

  <SpotAttribute NAME="sa1" UNIT="" TYPE="INTEGER">
   <DataBlock NAME="d2">
    <DataChunk START="n">
     n1 n2 n3 n4 n5 .... n256
    </DataChunk>
   </DataBlock>
  </SpotAttribute>

  <DataBlock NAME="d1">
   <DataChunk START="n">
     n1 n2 n3 n4 n5 .... n256
   </DataChunk>
  </DataBlock>

 </Measurement>

</MeasurementGroup>
```

The colour of the tag show how many times it may appear inside it's parent tag:
Exactly one. One or none, One or more, Zero or more,

(more details to follow)

17.2 Plain text data

The *Load Plain Text* plugin expects files in this format:

```
                  Time_0      Time_10     Time_20
hum_xrna_22       2.193427    1.151119    -1.042308
hum_xdna_23_II    8.64423     -1.922      -10.56623
```

```
       hum_xrna_24      6.50871    −1.73525    −8.24396
       hum_xrna_24 II   6.0916     −2.66144    −8.75304
```

The columns can be delimited by space, commas, tabs or any combination of the three. Measurement names may appear on any line of the file. Gene, probe and spot names may appear in any column, but will only be recorded if they consistently appear in the same column.

The *Save As Text* plugin can write files in this format for importing into other software packages.

17.3 Cluster data

The Cluster Manager can load cluster data from files are written in XML syntax, as follows:

```
<XML>
<CLUSTER>
<NAME>Pathways</NAME>

  <CLUSTER>
  <NAME>PathII(a)</NAME>
    <ELEMENTS>
       <GENE> TAP1 protein </GENE>
       <GENE> GDP receptor </GENE>
       <GENE> TAP6 protein </GENE>
       <GENE> putative HGFT promoter </GENE>
    </ELEMENTS>
  </CLUSTER>

  <CLUSTER>
  <NAME>PathII(b)</NAME>
    <ELEMENTS>
       <PROBE> probe_6 </PROBE>
       <PROBE> probe_23 </PROBE>
       <PROBE> probe_19 </PROBE>
       <PROBE> probe_8 </PROBE>
       <PROBE> probe_1 </PROBE>
    </ELEMENTS>

    <CLUSTER>
    <NAME>Blank Spots</NAME>
      <ELEMENTS>
         <SPOT> AA04 </SPOT>
         <SPOT> AB04 </SPOT>
         <SPOT> AB12 </SPOT>
      </ELEMENTS>
    </CLUSTER>

  </CLUSTER>

</CLUSTER>
</XML>
```

The file must begin with `<XML` and end with `</XML>`. Each `<CLUSTER>` tag starts a new cluster which must be terminated with a `</CLUSTER>` tag.

Each cluster can optionaly contain a list of elements enclosed in `<ELEMENTS>` ... `</ELEMENTS>` tags. The elements can be identified as either Gene, Probe or Spot names but not a mixture of name types within a single cluster.

Each cluster can also contain any number of other clusters.

Indenting the file as above is encouraged, but not required.

17.3.1 Optional Tags

The following optional tags can also be included within a `<CLUSTER>...</CLUSTER>` block:

- `<COLOUR> 11120867 </COLOUR>`

The COLOUR is specified as a 24−bit integer (written in base 10, not hexadecimal) made by composing three 8 bit values:

bits 16−23 store the red value
bits 8−15 store the green value
bits 0−7 store the blue value

- <GLYPH> 5 </GLYPH>

The GLYPH is specified as an integer in the range 0...8 corresponding to the list: Shield, Arrow Up, Plus, Diamond, Cross, Arrow Down, Octagon, Hour Glass

18 Tutorial: Getting started with maxdView

This is a quick tour of some of the main features of **maxdView**. The internal help system is the main source of documentation for **maxdView**. All dialogs and control panels have a "Help" button which opens the relevant page in the help browser.

Loading data....

We will begin by loading a demonstration data file stored in native format. Start the "Read Native" plugin by selecting it from the "File" menu.

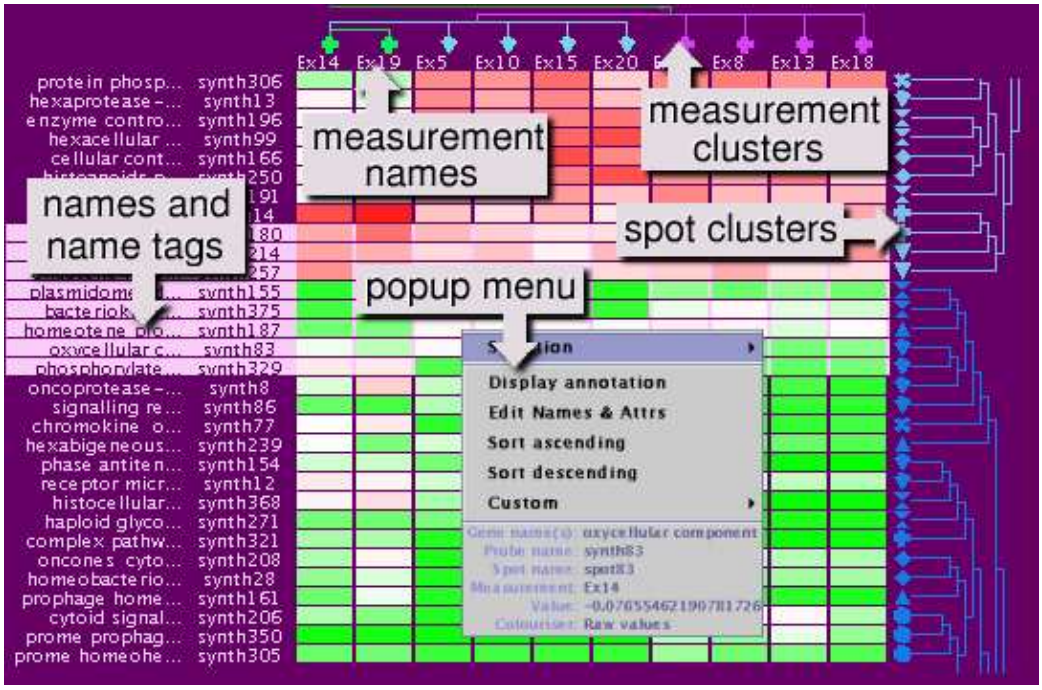
If "Read Native" does not appear in your "File" menu, the plugin could not be found or be started for some reason. You need to download an extra software package to use "Load Native". See the installation document for more details.

The "Read Native" panel is an extended file browser. Options on the right hand side let you control how files are loaded. For this demonstration we will ignore these controls. Unless you have previously changed them, the default settings are fine. If you have changed them, set them back so "Load how?" is "Replace", and all of the boxes under "Load what?" are selected.

Find the "`demo`" directory which will be located wherever you installed **maxdView**. In this directory will be a file called "`tutorial1.maxd.gz`". This file contains some synthetic data encoded using **maxdView** native XML format and compressed using GZIP. Select this file and press "Open" (or double click on the file).

Some data should appear in the main display, which will look something like this: (but without the big labels!)

The main display....



This data has 19 Measurements of 400 Spots.

Measurements (which correspond to different experiments, samples, conditions or timepoints) are laid out left to right. The Measurement Names are shown along the top of the display.

The Spots are arranged down the display. Each row starts with one or more columns of Name and Name Attributes. These columns will be explained later.

The values associated with each Spot in each Measurement are represented by the coloured boxes which take up most of the display. Two colour schemes are in use, one for the first four Measurements and the other for the rest of the Measurements.

The data also includes some Clusters, represented by coloured symbols (called glyphs) linked by a hierarchical structure (the tree).

Spot Clusters appear on the right hand side of the display. Spots in the same Cluster can be identified as they have the same coloured glyph. The tree shows how the clusters are nested.

Along the top of the display, above the Measurement names, are the Measurement Clusters. These are shown with a tree and coloured glyphs in the same way as for Spot Clusters.

We will play with the Clusters later in this tutorial.

As you move the mouse pointer around the display, a status line at the bottom of the window gives information about the thing under the pointer. This information will also appear in a *tooltip* window next to the mouse pointer.

Unless you have a very big display, not all of the Spots and Measurements will fit onscreen, so the window will have scrollbars. Note that as you scroll around, only the Spots and Measurements move, the Names and Clusters stay onscreen.

Pressing the right mouse button displays the Popup menu.
(On systems with no right button, use *Ctrl + Alt + Left mouse* instead.)

The popup menu is context sensitive, in other words a different menu is displayed for each of the different things in the display. For example, when you right−click on a Measurement name you get a menu specific to Measurements.

Popup menu...

The popup menu usually contains shortcuts to functions or options available elsewhere. The menus also contain options specific to certain elements of the display:

- The Name and Name Attributes popup menu has options for adding and removing the name columns, and for changing what is displayed in the column. The concept of Names will be explained below.

  The "Sort this column" option re−orders the Spot rows so that the values in the column are in alphabetical order. Try sorting the different name columns.
- The Cluster popup menu contains options for showing and hiding parts of the cluster tree and a shortcut which opens the Cluster Manager.
- The Measurement Name popup menu includes shortcuts to open the Measurements control panel and to show and hide the Measurement.
- The Spot popup menu contains a summary of data associated with that Spot.

  The "Sort ascending" and "Sort descending" options order the Spot rows based on the column that was clicked on. Try sorting the data on different columns.

  As the rows are rearranged, the cluster trees become all mixed up. This is because the Spots in each cluster are no longer adjacent. The Sort Clusters plugin can re−order the Spots using the cluster trees.

All popup menus contain the Custom sub−menu where you can collect frequently used commands for easy access. See the Custom menu help page for more information.

The Selection sub−menu also appears on all popup menus. This sub−menu contains options for manipulating the data selection.

Display options...

1. Layout

Now we will have a brief look at how the layout and colouring of data is controlled. Use the "Display −> Layout" menu entry to open the layout control panel.

The controls in this panel are divided into four groups. The first tab contains sliders which alter the size of the spot boxes used in the main display. By reducing the "Column Width" and "Row Height" for example, you can fit more data into the window.

Try adjusting the various layout parameters and seeing how they affect the display. Try selecting different text fonts using the "Text" tab.

The "Name Columns" tab of this control panel can also be used to add and remove Name and Name Attribute columns. The "Clusters" tab controls how Clusters are displayed. See the help page for information.

2. Colours

Use the "Display −> Colours" menu entry to open the colouriser control panel. Colours for the data used in this tutorial are generated by two things called `colourisers'. One, called "Raw values", appears as a bar of smoothly varying colours from green, through white, to red. The other, called "Error values", is represented as three sets of type−in fields, each with a colour button underneath them.

You can click on the white 'box' in the "Raw values" bar and drag it left or right to adjust the rate at which the colurs are blended. The data colouring in the main display changes whenever you release the mouse button. By double clicking on the white box, you can open a colour chooser. You can also double−click on the red and green boxes at either end of the colour bar to change their colours.

You can do a lot more with colourisers, see the help page for more details.

Filtering....

The next thing we will try is using filters to select a subset of the spots for display.

Launch the "Math Filter" plugin using the entry on the "Filter" menu. The window that appears is divided into four sections. In the top−left panel, you can write a mathematical formula, such "Ex4 > 0.55", which will select only those Spots in which the value in Measurement "Ex4" is > 0.55;

You can experiment with different formulae, such as "Ex1 > Ex2 and Ex1 < Ex3". The "Math Filter" understands "and" and "or" and lets you build simple arithmetic expressions such as "(Ex1 + Ex2) * 3 <.0"

Dismiss the "Math Filter" using it's "Close" button. Closing the window disables the filter and all Spots are shown once again.

Open the "Profile Filter" using the entry on the "Filter" menu. This filter selects Spots based on their similarity to a chosen target Spot.

In the left−hand panel use the "All" button to select all Measurements for consideration. Now pick a target spot from the list in the right hand panel. The drop−down box above this name allows you to search using the different names and Attributes of the Spots.

When a target Spot picked, the filter removes all but that spot and the 10 most similar Spots. The main display will update to show just these Spots.

To remove the "Profile Filter", close the plugin's window.

Finding....

Use the "Display −> Find" menu entry to open the finder. This panel has type−in fields for finding either Spots or Clusters. Type "25/E" into the top field and select "Probe name" in the drop−down list. Press return (or the "Go" button). The display scrolls so that the Spot containing the Probe "25/EBO0" is at the top.

Press return (or "Go") again and the message "search finished, 1 found" is displayed in the finder. This tells you that there is only one instance of "25/E" in the Probe names.

If you try finding "25/" instead, you will discover that each press of return moves to another Probe with a name begining with "25/", for example "25/OBR5", "25/JDI1" and so on.

You can also search for Gene and Spot names and their name Attributes, which are introduced below.

Clusters...

Clusters are controlled using a plugin called the Cluster Manager which appears in the "Viewer" menu. You can also open this panel using the "Show Properties" option in the Cluster popup menu.

With this control panel you can change the colour and glyph assigned to Clusters, make them temporarily invisible, load, save and delete them.

More information about Clusters can be found in the Working with Clusters tutorial.

1. Hiding and Showing

Start the Cluster Manager plugin using the entry in the "Viewer" menu. The window which appears is divided into three panels. The top−right panel displays a tree in which the branches can be expanded and collapsed using the control to the left of each Cluster name.

Clicking on a name in the tree panel selects the cluster. The name colour and glyph of the selected cluster can be changed using the controls in the top−left panel.

Select the Cluster called "k=1". This is a child of "XCluster:k−means,k=5". You can select this Cluster by hand, or use the "Cluster −> Find" menu option to find and select it.

Scroll the main display so that you can see the light blue Clusters in the "k=1" part of the tree. Press the first "Hide" button in the top−left panel. This button hides the selected Cluster and all of its children. Note how the light blue part of the tree vanishes in the main display. Turn the Clusters back on using the "Show" button in the "Selection" part of the top−left panel.

Now hide all of the Spot clusters by pressing the second "Hide" button in the top−left panel. Then turn on the "k=1" Clusters using the "Show" button again. Now only the light blue Clusters are enabled in the main display.

2. Drag and Drop

An easy way to select Clusters for editing is to use *drag−and−drop* to bring a Cluster from the main display to the plugin:

- Position the mouse over a glyph or tree branch in the main display. You will know when the mouse is correctly positioned as the status line will show the name of the Cluster.
- Press and *hold* the left mouse button.
- Move the mouse to the top−right panel of the Cluster Manager window. The mouse cursor will be different to indicate the fact that drag and drop is enabled.
- Release the left mouse button once the pointer in the top−right panel. The Cluster Manager will select the Cluster and display it in the tree.

Names and Name Attributes...

maxdView terminology : Each Spot corresponds to one well on a microarray slide. The labelled material placed in the Spot is called the Probe. This Probe is used to detect one or more Genes during the hybridisation process.

In short: each line in the display corresponds to one Spot, which has a unique "Spot Name". Each "Spot Name" can have an associated "Probe Name". The "Probe Names" do not have to be unique, when microarray data contains replicate spots the replicates should have the same "Probe Name". Each "Probe Name" can be linked with one or more "Gene Name". The "Gene Name" can be in any format, **maxdView** does not support or require a particular naming scheme.

Each type of name can have "Attributes". These are extra (key,value) pairs that can be linked to each name. For example "Spot Name" might have "Row" and "Column" attributes. Each Spot name can then be associated with its coordinates on the array.

The Name Tag Editor allows you to view and edit the names and attributes for a Spot. Point at a spot in the main display and use the right mouse button to activate the popup menu. Select the "Edit Names & Attrs" option to open the editor.

The name columns to the left of each line show one or more of these Names and Name Attributes. You can control the number of columns, what each one contains, how it is aligned and the maximum width for each column. These controls are found in the layout control panel, acessed from the "Display" menu.

The popup menu for name columns has the "Show in this column" sub−menu. Try using this to select different Names or Name Attributes for display.

You can control these columns in more detail using the layout control panel.

And now...

- The Working with Clusters tutorial
- How about Writing a Plugin ?

19 Tutorial: Working with Clusters

This document describes the different parts of **maxdView** that allow you to create and visualise clusters.

Using Clusters to group things together

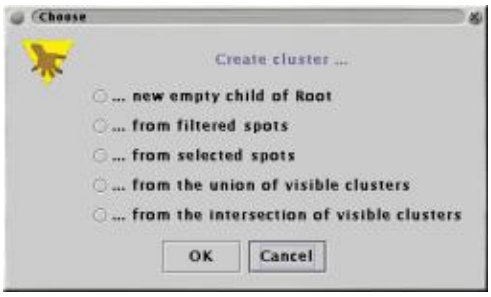
Clusters are useful for storing groups of Spots, Probes Genes and Measurements.

If you have used a filter to select an interesting set of Spots, creating a Cluster will allow you to quickly identify those Spots at a later time.

For a practical example, we will use the the demonstration data file "`tutorial1.maxd`". See the Getting started tutorial for details on where to find this file.

Once you have loaded "`tutorial1.maxd`", use a filter a isloate some of the Spots. For example use *Filter by Name or Value* to select spots with "enzyme" in their Gene names.

Start the *Cluster Manager* plugin. Selecting the "Create" option on the "Cluster" menu displays the following dialog box:



Select the second option, "...from filtered Spots", then provide a name for the new Cluster. The newly created Cluster will have all of the Spots which pass through the filter as its elements.

You will be able to see this new Cluster in the *Cluster Manager*s tree viewer. The new Cluster is allocated a random colour and glyph. The controls in the *Cluster Manager* allow you to change these properties.

After you disable the filter (by closing its window) the new Cluster `remembers' which Spots were selected by the filter. If you use a plugin which can display Clusters, for example the *Scatter Plot*, you will be able to see the Cluster glyphs identifying the previously filtered Spots.

Repeat the Cluster creating process using a different filter then use a *viewer* plugin to compare the two Clusters.

The *Cluster Manager* provides several other ways of creating Clusters, such as loading names from a file or by combining existing clusters.

Generating files for the Cluster Manager to load

One method for loading Cluster data into **maxdView** is to generate files that the *Cluster Manager* can read.

The simplest format 'List of Names' allows you to specify a single cluster in a file. Each line of the file should contain exactly one name, for example:

```
YBR042c
YBR108
YBL012a
```

```
YBL012b
YLC0981
```

You can specify any name or name attribute to match with. If one or more of the elements in the file match existing data then a cluster will be created. Names that are not recognised are ignored.

The 'Native' file format allows you to create a hierarchy of clusters from a file. The format is human−readable and uses XML syntax. See the File Formats for more details.

20 Tutorial: Commands and Hotkeys

Commands and hotkeys provide a quick way to access frequently used features of **maxdView**.

Many plugins can be accessed via *Commands* as well as via their normal user interface. It is possible, for example, to start a filter plugin and set a particular filter mode without touching the mouse.

A complete list of all plugin commands is generated whenever the plugins are scanned.

In this tutorial, we shall first define a hotkey for an existing command, and then use the Code Runner plugin to create a new command.

- Part 1: Calling an existing command
- Part 2: Defining a new command

20.1 Part 1: Calling an existing command

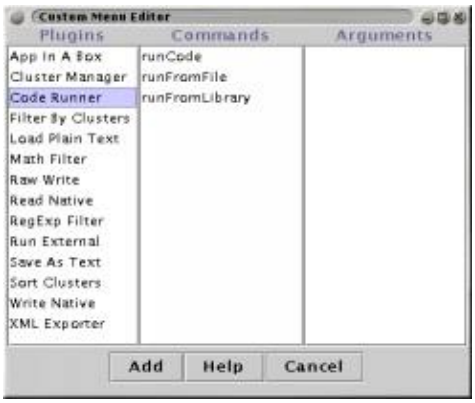
For this example, we shall use a command from the *Filter By Selection* plugin. This plugin is used for hiding (or displaying) spots which are currently filtered (or not).

This plugin provides two commands, "`start`" and "`stop`". The "`start`" command is used to fire up the plugin and takes an argument which sets the initial state (see the plugin's help page for details).

Open the custom menu editor....

We will use the Custom menu editor to install the new command. Open the editor by selecting the "Edit" option from the "Custom" menu. The "Custom" menu appears as a submenu on all Popup menus.

Press the "New Command" button to display the list of plugins and their commands.



Add the command....

Select the "`Filter By Selection`" plugin from the list in the left−hand panel. This causes the list of commands understood by this plugin to be displayed in the middle panel. Select the "`start`" command in this panel. The right−hand panel will now display the arguments accepted by the command.

Press the "Add" button to add this command to the custom menu. The window will return to the list of custom commands with the new "`Filter By Selection.start`" command at the end of the list.

Set the arguments....

The "`start`" command takes a single argument called "`filter`". This argument can be used to put the plugin into one of the three diferent filtering modes that it supports. The help page states that the value of this argument should be one of "`no filter`", "`unselected`" or "`selected`". For this example, we shall use "unselected". Type this into the

"`filter`" type−in field in the "Arguments" section.

Name the command....

Commands are given a default name, in this case "`Filter By Selection.start`".

You can change this name as required by typing a new name into the "`Name`" type−in field. The name is only used when listing the command in the menu, changing the name does not affect the command itself.

Define a hotkey....

Of more interest than the name is the "Hotkey" that will be used to activate the command. There are three forms of hotkey specification, one based on a "Ctrl−C" prefix, one based on an "Escape" prefix and one for the 'function' keys F1...F16

The first hotkey format allows you to specify a single key for the command. For this example, we might choose "F" (for **F**ilter). Type "F" into the "Hotkey" field and close the Custom menu editor. In the main display, press "Ctrl−C" ("C" for **C**ustom) then "F". This will call the command and start the plugin.

When the plugin starts, it will be in "unselected" mode. Try changing the mode by setting a different value in the "Arguments" section of the custom menu editor.

The second hotkey format is based on the "Escape" prefix. In this format, sequences of keys can be used to launch the command. This can make it easier to remember which hotkey will launch which command.

Open the custom menu editor and select the "`Filter By Selection.start`" command that you have just added. Change the hotkey specification to be "`esc fil`". Close the custom menu editor and then, in the main display window, press the "Escape" key then then keys "`f`","`i`" and "`l`" in sequence. This will start the plugin as before.

You can have as many characters as you wish in a hotkey specified using the "`esc`" format.

20.2 Part 2: Defining a new command

Using the library feature of the Code Runner plugin, it is possible to access your own code from the custom menu or via a hotkey.

In this section of the tutorial, we shall create a 'hello world' command. Whilst not very useful in itself, this example shows how any code you want can be accessed via the plugin command mechanism.

The Code Runner will be used to define the command to be executed, and then the Custom menu editor will be used to access this command.

Write the code

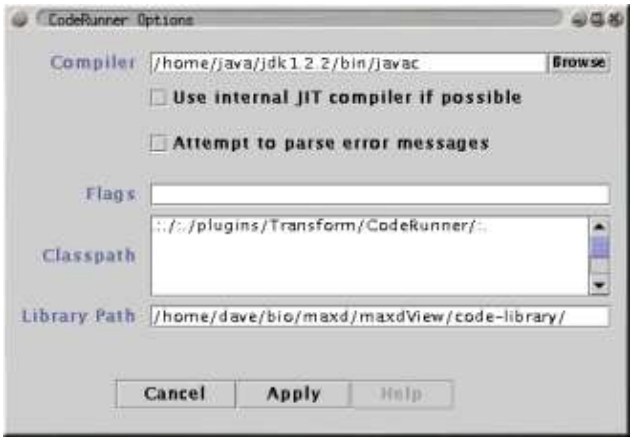
Start the Code Runner plugin. Paste the following code into the editing panel:

```
public void doIt()
{
  mview.infoMessage("Hello world");
}
```

note: an empty "`doIT()`" method body will already be in the editor, you only need to add the single line in red. Make sure to include the semi−colon at the end of the line.

Compiler options....

If you have not previously configured the options of the Code Runner plugin, press the "Options" button to display the following panel:



The "Compiler" field must contain the path to a Java compiler (called '`javac`' or '`javac.exe`') − see the Code Runner help page for more information.

Press the "Apply" button to close the options dialog box.

Add it the library

Press the "Add" button (in the bottom left−hand corner). You will be asked to confirm whether you want to compile the code and add it to the library. If the compilation succeeds, you will be asked to give the new entry a name, call it "hello".

If the compile fails you will be informed of the reason. If the compiler cannot be found, check the "options" as detailed above. If compiler error messages are displayed, make sure that the code in the editor exactly matches that shown above.

Once the new entry is in the library you can test it by double−clicking on the entry in the library list. If it works, the message will be displayed. Having verified the new library entry, you can close the Code Runner plugin.

Create a new command

Any entry in the Code Runner library can be invoked by sending the plugin a "`runFromLibrary`" command.

The "`runFromLibrary`" command is accessed in the same way as was done in the first part of this tutorial. Firstly, open the custom menu editor by selecting the "Edit" option from the "Custom" menu. The "Custom" menu appears as a submenu on all Popup menus.

Press the "New Command" button to display the list of plugins and their commands. Choose the "`Code Runner`" plugin and then the "`runFromLibrary`" command. Press "Add" to add the new command to the custom menu hierarchy.

The command takes a single argument which tells the plugin which library entry to execute. Enter the name of the new library entry (i.e. "hello") into the "`name`" field in the "Arguments" section of the editor. The command is now ready for use.

You can define a hotkey for the command as outlined above.

For more information about what you can do with plugin commands, see the Working with Plugin Commands Tutorial.

21 Tutorial: The Cluster APIs

maxdViews Cluster APIs allows you to interact with Cluster data from your own plugins, or via code−fragments in the Code Runner plugin.

Accessing existing Clusters

```
ExprData.Cluster ExprData.getRootCluster()
```

This method returns the top−level cluster. From there you can use the clusters own methods to get its children one by one.

```
ExprData.Cluster maxdView.getCluster(String message)
```

This method displays a dialog box with which the user can pick a cluster. If the user makes a selection, this cluster is returned.

Cluster Iterators

The `ClusterIterator` provides an easy way to traverse the cluster hierachy. It is used as follows:

```
ExprData.ClusterIterator iterator = edata.new ClusterIterator();
ExprData.Cluster cluster = iterator.getCurrent();
while(cluster != null)
{
   // do something with cluster
   // ...
   cluster = iterator.getNext();
}
```

Other `ClusterIterator` methods can be used to adjust the search pattern:

```
public void    positionAt(Cluster c)
  // move the iterator to the specified cluster

public Cluster getFirstLeaf()
  // find the first cluster with no children
public Cluster getNextLeaf()
  // find the first cluster with no children
```

Elements and Element names

The *elements* of a cluster is the ordered list of either Spots or Measurements that this cluster contains. A clusters *element names* is the list of Spot, Probe, Gene or Measurement names that was used to generate the *elements*.

The reason for the distinction is that when clusters are specified by for example Gene names, there may be more elements than element names. If a Spot in the Measurement is a duplicate, then there will be multiple occurrences of the Gene name in that Spot.

For example, consider a Measurement containing the folllowing Spots:

```
SpotName   ProbeName
A01        pr001
A02        pr004
A03        pr007
B01        pr026
B02        pr004
```

Note that Probe 'pr004' is duplicated. If a cluster is created using the *element names* 'pr001','pr004' then the *elements* will Spots 'A01', 'A02' and 'B02'.

The separation between elements and element names must be maintained in order that misleading cluster information is not created. The `Cluster` class provides methods for manipulating both elements and element names and attempts to keep the two synchronised.

```
int   getNumElements()
 // returns the number of elements
int[] getElements()
 // returns an array of Spot or Measurements indices

int    getElementNameMode()
 // returns one of the constants below
Vector getElementNames()
 // returns the original names used to construct this cluster
void setElementNames(int elem_name_mode, Vector elem_names)
 // use these constants for elem_name_mode
 //   SpotName,
 //   SpotIndex,
 //   ProbeName,
 //   GeneName,
 //   MeasurementName,
 //   MeasurementIndex,
 //
 // and provide a collection of names in elem_names
```

The `getElements()` method returns the actual Spot or Measurement indices. For example, the following code prints the cluster elements as Spot Names:

```
if(cluster.getElementNameMode() == ExprData.SpotIndex)
{
   int[] ids = cluster.getElements();
   for(int s=0; s <ids.length; s++)
     System.out.println( edata.getSpotName( ids[s] ) );
}
```

If the example data used above were loaded, the following code would create a cluster containing the two Probes:

```
Vector name = new Vector();
names.addElement("pr001");
names.addElement("pr004");
ExprData.Cluster cluster = edata.new Cluster("Example",
                                             ExprData.ProbeName,
                                             names);
```

This new cluster would have 3 elements.

Although the newly created cluster exists, it has not been added to the data. To do this a parent for it must be chosen. The 'root' cluster is always present and is usually a good choice for new clusters. You can install the cluster as a child of root using the `ExprData.addChildToCluster()` method:

```
edata.addChildToCluster( edata.getRootCluster(), cluster );
```

Children

Each cluster can have zero or more child clusters.

```
public int     getNumChildren()
 // return the number of children
public Vector  getChildren()
 // returns a java.util.Vector of Clusters

public void addCluster(Cluster child)
 // adds a new child
```

The `Vector` returned by `getChildren()` contains the child `Cluster` objects. You can access these objects using the usual `Vector` methods, for example:

```
...
int nc = cluster.getNumChildren();
Vector ch = cluster.getChildren();
for(int c = 0; c <nc; c++)
{
   ExprData.Cluster child = (ExprData.Cluster) ch.elementAt(c);
```

```
    // do something with this cluster....
}
```

Visiblity, Colour and Glyph

The visibility of the cluster (i.e. whether it is drawn or not, or counted in filters) is controlled using the following methods:

```
public boolean getShow()
public void setShow()
```

The coloured glyph that is used when drawing the cluster can be changed using the following methods:

```
public Color getColour()
public void  setColour(Color c)

public int  getGlyph()
public void setGlyph(int glyph_code)
```

The glyph code is a number in the range 0..8 representing the available shapes: Shield, Arrow Up, Plus, Diamond, Cross, Arrow Down, Octagon, Hour Glass

Other useful methods

```
void  deleteCluster(Cluster)
void  removeAllClusters()
```

22 Tutorial: Writing a Plugin

This document illustrates the process of writing a plugin class. A reasonable level of understanding of the Java language is assumed.

This example plugin is called "Sort by Probe Name" and, unsurprisingly, will re−order the Spots so that the Probe names are in alphabetical order. To keep the example simple, the plugin has no user interface.

The constructor

The constructor for a plugin must take a single argument of type `maxdView` which gives it a link back to the main application. This argument should be saved in an instance variable.

The constructor should only initialise the plugin, and should not do any other work. An instance of the plugin class will be created during the "rescan plugins" operation so the constructor should not display any user interface components. The plugin is not considered to be running until it's `startPlugin()` method is called. The example constructor looks like this:

```
public SortByProbeName(maxdView mview_)
{
    mview = mview_;
    edata = mview.getExprData();
}
```

In addition to storing the `maxdView` reference, this constructor also gets and stores a reference to the current `ExprData` object. You do not have to do this, but it saves repeated calls to `mview.getExprData()` later on.

The `Plugin` interface

A plugin class must implement the `Plugin` interface. This interface has 5 methods:

```
public void startPlugin()
public void stopPlugin()
public PluginInfo getPluginInfo()
public PluginCommand[] getPluginCommands()
public void runCommand(String name, String[] args, CommandSignal done)
```

These methods are implemented as follows:

`startPlugin()`

The `startPlugin()` method is called by the plugin launcher once an instance of the plugin class has been successfully created. This is where the plugin should start doing things, such as building and displaying a user interface. In this case, `startPlugin()` calls another method which actually does the work. The `doSort()` method is described below.

```
public void startPlugin()
{
   doSort();
}
```

stopPlugin()

The `stopPlugin()` method may be called by the plugin launcher, or by other plugins if they want this plugin to stop. The method should release any resources held by the plugin, and shut down any user interface components.

```
public void stopPlugin()
{
}
```

In our example, the method doesn't need to do anything.

getPluginInfo()

The `getPluginInfo()` method is used by the application to get details about ths plugin. These details are required to place the plugin in the menu hierarchy and to provide links in the help documentation.

```
public PluginInfo getPluginInfo()
{
   return new PluginInfo("Sort by Probe name",
                         "transform",
                         "Orders the Spots alphabetically by Probe names",
                         "",
                         1, 0, 0);
}
```

The `PluginInfo` class provides a single constructor, taking 7 arguments. The arguments are (in order):

- `String: name`
  The name of plugin as it will appear in the menus. This does not have to be the same as the class name.

- `String: type`
  The type of the plugin determines which menu the plugin will be listed in. Legal options are "`importer`", "`exporter`", "`transform`", "`filter`" and "`viewer`".

- `String: short_description`
  A text string that is used on the "Commands" help page. Use this to provide a brief descrption of what the plugin is for.

- `String: long_description`
  A text string that is used on the "About" help page. Use this to present more information about the plugin, such as any licencing or copyright message.

- `int: major_version_number`
- `int: minor_version_number`
- `int: build_number`

  The version number of a plugin is represented as `X.Y.Z`, where `X` is the major version number, `Y` is the minor version number and `Z` is the build number.

  At this time, version numbers are ignored, but later versions of **maxdView** might provide some form of version control and automatic updating for plugins.

`getPluginCommands()`

Plugins can optionally register one or more commands with **maxdView**. These commands can be invoked by the user using the "Custom menu" or from code, e.g. in other plugins or via the Code Runner.

```
public PluginCommand[] getPluginCommands()
{
  return null;
}
```

This example returns `null` which means 'no commands to be registered'. Later examples in this tutorial will show how commands are defined.

`runCommand()`

The final `Plugin` method is used to invoke any commands which this plugin has registered. Even if the plugin has no commands, as in this example, an empty method body must still be provided.

```
public void runCommand(String name, String[] args, CommandSignal done)
{
}
```

Providing Help documentation

Plugins should have a help page. This must be in the same directory as the plugin class, and must have the same name as the class, but with a `.html` extension. For our `SortByProbeName.class`, the help document would be called `SortByProbeName.html`. A link to this document will appear in the "Commands" help page.

The `doSort()` method

The following code does the actual sorting. It uses several `ExprData` and `maxdView` methods to access the list of names. Java's `Arrays` class provides the actual sorting routine.

```
public void doSort()
{
  // get a mapping of Probe name to a Vector of spot indices

  Hashtable pnht = edata.getProbeNameHashtable();

  // build an array of probe names...

  String[] pnames = new String[pnht.size()];

  int p = 0;
  for (Enumeration e = pnht.keys(); e.hasMoreElements() ;)
  {
      String pname = (String) e.nextElement();
      pnames[p++] = pname;
  }

  // sort this array...

  Arrays.sort(pnames);
```

```
    // and now build the Spot order array using this sorted list of names

    int[] new_order = new int[edata.getNumSpots()];

    int so = 0;

    for (p=0; p <pnames.length; p++)
    {
        Vector sids = (Vector) pnht.get( pnames[p] );

        for(int s=0; s  <sids.size(); s++)
        {
          int sid = ((Integer) sids.elementAt(s)).intValue();

          new_order[so++] = sid;
        }
    }

    // and install the new Spot order

    edata.setSpotOrder(new_order);
  }
```

Compiling the plugin

The complete code for this example can be found in `SortByProbeName.java`. You need to put this `.java` file somewhere in the `plugins/` subdirectory and compile it.

The precise location of the file makes no difference, as long as it is somewhere inside the `plugins` directory hierarchy. By convention, existing plugins are given a directory each and are grouped into five sections based in their "type". You might choose to create a new directory `plugins/Transform/SortByProbeName` and put the `.java` file there.

For sucessful compilation, the `maxdView` and `ExprData` class files must be found in the classpath you give to `javac`.

If you have put the `.java` file in `plugins/Transform/SortByProbeName` and this is the current working directory, then a suitable comand would be:

```
javac −classpath ../../.. SortByProbeName.java
```

(on MS−Windows systems use `..\..\..` instead)

Once the class is compiled, use the "File −> Rescan Plugins" command to detect the new plugin. A new entry should appear in the "Transform" menu, selecting this will run your code.

Adding a plugin command

Adding support for commands to a plugin commands has two benefits. Firstly, the functionality is available to other plugins and via the Code Runner. Additionally, the features of the plugin can be accessed via a custom menu entry or hotkey.

Most existing plugins provide "`start`" and "`stop`" commands. These commands do not make sense for this plugin as it displays no user interface. Instead, a "`sort`" command would seem more appropriate.

The commands that a plugin can understand are specified in the return value of the "`getPluginCommands()`" method. This method is queried by **maxdView** when plugins are *scanned*. The array of `PluginCommand` objects returned by this method defines the commands that will be associated with this plugin.

Register the command

The current implementation of this method is to return `null`. Modify the method to be as follows:

```
public PluginCommand[] getPluginCommands()
{
  PluginCommand[] com = new PluginCommand[1];

  String[] args = new String[]
  {
    // name       // type     //default   // flag   // comment
    "ascending",  "boolean",  "true",     "",       "in which order to sort"
  };

  com[0] = new PluginCommand("sort", args);

  return com;
}
```

The method now returns an array containing a single `PluginCommand` object. This object defines the name of the commands and describes the argument that can be provided when the command is invoked.

(Note that in this example this is only one `PluginCommand` object, but you can define as many commands as you want as long as they each have a unique name.)

The constructor for the `PluginCommand` class takes two parameters, a `String` specifying the name of the command, and an array of `String`s defining the arguments of the command. Each of the arguments of the command is specified by five `String` parameters:

- **name** *the name of the argument*
- **type** *integer, double, boolean, string, char or file*
- **default** *the default value*
- **flag** *the character 'm' implies that the argument is mandatory*
- **comment** *a short description of this argument*

As an example of a more complicated command which takes more than one argument, consider a plugin which draws a graph. It might provide a command such as:

```
String[] args = new String[]
{
  // name         // type     //default   // flag   // comment
  "width",        "integer",  "100",      "m",      "width in pixels",
  "height",       "integer",  "100",      "m",      "height in pixels",

  "error_level",  "double",   "0.00001",  "",       "minimum error threshold",

  "apply_filter", "boolean",  "false",    "",       "remove filtered Spots?"
};

com[0] = new PluginCommand("plot", args);
```

Handle the command

The plugins `runCommand()` method will be called when a command is invoked, either via the custom menu (or a hotkey) or via the **maxdView** `runCommand()` method.

The current implementation of this method does nothing. Modify the method to be as follows:

```
public void runCommand(String name, String[] args, CommandSignal done)
{
  if(name.equals("sort"))
  {
    boolean ascending = mview.getPluginBooleanArg("ascending", args, true);

    doSort();
  }
  if(done != null)
    done.signal();
```

```
    }
```

The three parameters to this method provide the name of the command to be executed, an array containing zero or more (*name*,*value*) pairs and a `CommandSignal` object.

For more information about how the values of the arguments are passed see the "Working with Plugin Commands" tutorial.

23 Tutorial: Working with Plugin Commands

The `loadPlugin()` and `runCommand()` methods provided by the `maxdView` class allow you to invoke plugin commands from within your own code. This both facilitates writing macros and enables intra−plugin communication.

This document describes how 'macro' functions can be built by glueing together existing commands.

Several larger examples are provided in the "`code−fragments`" subdirectory:

```
load−edit−save−example.java
script−demo.java
name−munger−demo.java
super−grouper−demo.java
```

23.1 Using `runCommand()`....

Any plugin command can be accessed by calling the `runCommand()` method of a `maxdView` object:

```
void runCommand( String plugin_name, String command_name,  String[] args)
```

The first two parameters give the name of the plugin (as it appears in the menus) and the name of the command. The third parameter is an array of strings in which the arguments for the command are specified. Arguments are listed as *(name, value)* pairs of strings within the array, for example:

```
String[] args = new String[]
{
  "file",            "example.dat",
  "start_line",      "1",
  "end_line",        "1024",
  "comment_prefix",  "#"
};
```

All argument values are specified as strings irrespective of their real type. For instance, the integer value 7 is represented as "7" and the boolean value true is represented as "true".

Example: To call the *Sort by Name or Value* plugin and sort the Spots based on Spot name:

```
String[] args = new String[]
{
  "name",      "Spot name",
  "order",     "descending"
};

mview.runCommand( "Sort by Name or Value", "sort", args );
```

23.2 Running more than one command....

Many plugin commands such as "`sort`" are 'one−shot' in the sense that when they are called the plugin is started, does its job and then goes away. Other commands are 'persistant' in that the plugin remains running after the command has finished. An example is the "`start`" command supported by most filter plugins. When a filter plugin is started, it remains alive until it is sent a "`stop`" command or the user closes it's window.

To be able to send more than one command to the same instance of a plugin, a handle to the plugin object is required. This is obtained using the `loadPlugin()` method which returns a `Plugin` object. This object can then be used in calls to `runCommand()`.

Example: To start a filter, use it to define a new Spot selection and then remove it:

```
// first clear the current selection
edata.clearSpotSelection();
```

```
// load the filter plugin and save the reference to it
Plugin my_filter = mview.loadPlugin( "Math Filter" );

// build an arguments array for the filtering rule
String[] args = new String[] { "filter", "Time1 > Time0 and Time2 > Time1" };

// start the filter plugin with the specified filtering rule
mview.runCommand( my_filter, "start", args );

// add filtered Spots to the current selection
edata.addFilteredSpots();

// and then stop the filter plugin
mview.runCommand( my_filter, "stop" );
```

24 Tutorial: The RMI interface

This interface enables limited access to data via the Java RMI (Remote Method Invocation) mechanism. This allows other Java applications to connect to a running **maxdView** application and interact with it. The applications can be on the same machine or connected via a network.

A set of demonstration applications is provided which show how remote methods can be used to provide a bidirectional link between **maxdView** and your Java application or applet.

The Java Tutorial provides a good introduction to RMI and contains some example programs illustrating what can be done.

To enable RMI operation in the JVM, you need do a number of things differently:

- A policy file which permits network connections and file i/o is needed. Policy files are used when a Java `SecurityManager` is enabled. When RMI is used a `SecurityManager` is required so the policy file must be provided (see below).
- The `rmiregistry` (supplied with the Java SDK) must be running.
- **maxdView** must be started with the `-allow_rmi` command line option.
- Additional RMI and security specific options must be passed to the JVM running **maxdView** (see below).

The following interfaces are defined:

- `RemoteExprDataInterface`
  to access Spot and Measurement data
- `ExprData.ClusterHandle`
  to access Clusters
- `ExprData.RemoteExprDataObserver`
  to receive update events
- `ExprData.RemoteDataSink`
  to register objects that can receive data
- `ExprData.RemoteSelectionListener`
  to monitor the Spot selection

RMI Examples

A set of examples is provided in the `rmiDemo` directory. To compile the `rmiDemo/java` class you will need a classpth that includes the **maxdView** directory and the RMI demo directory, for example:

```
cd rmiDemo

javac -classpath .:.. rmiDemo.java
```

Once compiled, the `rmiDemo` application is run as follows:

```
java -Djava.rmi.server.codebase=file:/LOCATION_OF_maxdVIEW/
```

```
-classpath LOCATION_OF_maxdVIEW:LOCATION_OF_RMIDemo
RMIDemo [-host HOSTNAME] [-demo 1|2|3|4|5]
```

Note: sections marked in in red must be replaced with the correct paths for your file system. (see below)

The `-host` option specifies the name of the machine running **maxdView**. If this option is not present `localhost` is used.

The optional `-demo` argument specifies which demo to run. If this option is not present each of the demos are run sequentially. The demos are:

1. creates a `Cluster` of Probe names
2. creates a `RemoteExprDataObserver`
3. creates a `RemoteDataSink`
4. creates a `RemoteSelectionListener`
5. creates a hierarchical `Cluster` of Spot names

The Java Policy File

The following policy file allows applications to use sockets, read and write system properties, access certain run-time features and to read and write parts of the file system.

This file is normally placed it in `~/.java.policy` although you can use a JVM option (see below) to specify a different location.

The most likely reason for the demo applications throw to `RemoteException`s is an incorrect or missing policy file. The error message will indicate what the application was trying to do when the problem arose.

Note: You will need to modify the relevant file paths in the final `grant` section (marked in red) to suit your file system.

```
grant {
  permission java.net.SocketPermission "*:1024-65535", "connect,accept";
  permission java.net.SocketPermission "*:80", "connect";
};

grant {
  permission java.util.PropertyPermission "user.*", "read";
  permission java.util.PropertyPermission "java.*", "read";
};

grant {
  permission java.lang.RuntimePermission "getClassLoader";
  permission java.lang.RuntimePermission "createClassLoader";
};

grant {
  permission java.io.FilePermission "/", "read";

  permission java.io.FilePermission "/tmp", "read";
  permission java.io.FilePermission "/tmp", "write";

  permission java.io.FilePermission "/home/", "read";

  permission java.io.FilePermission "/home/dave/", "read";
  permission java.io.FilePermission "/home/dave/", "write";

  permission java.io.FilePermission "/home/dave/-", "read";
  permission java.io.FilePermission "/home/dave/-", "write";

  permission java.io.FilePermission "/home/dave/bio/maxd/maxdView/rmiDemo/-", "read";
  permission java.io.FilePermission "/home/dave/bio/maxd/maxdView/-", "read, write";
};
```

Additional JVM Options

The JVM running **maxdView** must be informed about the policy file (if it is in a non−standard location) and about the `codebase` which is the location it will find remote classes in.

These options are set using the JVM's `−D` command line argument:

```
java −Djava.rmi.server.codebase=file:/home/dave/bio/maxd/maxdView/
     −Djava.security.policy=/home/dave/.java.policy
     −classpath ...
```

Note: the trailing slash on the `codebase` is required.

What to do when it doesn't work...

A lot of different options must be configured correctly to get RMI properly working. If you can't get the demo applications to work , make sure you have done all of the following:

- Started the `rmiregistry` (which runs in the background)
- Installed a suitable policy file
- Started **`maxdView`** with the `−allow_rmi` command line option in a JVM with the correct `codebase` and `policy.file` options.
- Compiled the **`RMIDemo`** application successfully
- Started **`RMIDemo`** in a JVM with the correct `codebase` and `policy.file` options.

25 Tutorial: Wrapping maxdView with another application

It is possible to write a `wrapper' class which instantiates a maxdView object and controls it directly. This enables you to integrate maxdView's functionality with another application.

This tutorial shows the various interfaces that maxdView provides can be used for integration.

An example wrapper class is provided in `WrapperDemo.java`.

Contents:

Creating a maxdView object

The following code creates a `maxdView` object and uses the `getBusy()` method to start the application:

```
public static void main(String[] args)
{
    try
    {
       maxdView mv = new maxdView(args);
       mv.getBusy();
    }
    catch(java.rmi.RemoteException re)
    {
    }
}
```

(the `java.rmi.RemoteException` is only thrown when the maxdView object is running in RMI mode.)

Recieve events using the `DataObserver` interface

An object of a class which implements the `ExprData.ExprDataObserver` can be registered to recieve events.

```
public interface ExprDataObserver
{
  public void dataUpdate        (DataUpdateEvent due);
  public void clusterUpdate     (ClusterUpdateEvent cue);
  public void measurementUpdate (MeasurementUpdateEvent mue);
  public void environmentUpdate (EnvironmentUpdateEvent eue);
}
```

The object is registered by a call to:

```
void ExprData.addObserver(ExprDataObserver edo)
```

and can be un−registered with:

```
void ExprData.removeObserver(ExprDataObserver edo)
```

The more information about maxdView events, see the Programmer's Guide.

The `ExternalSelectionListener` interface

The current selection (which can be Spots, Clusters or SpotMeasurements) can be monitored by installing an object of a class which implements the following interface:

```
public interface ExternalSelectionListener
{
  public void spotSelectionChanged(int[] spots_ids);
  public void clusterSelectionChanged(ExprData.Cluster[] clusters);
  public void spotMeasurementSelectionChanged(int[] spot_ids, int[] meas_ids);
}
```

When any of the selections change the correspnding method in the `ExternalSelectionListener` object will be called.

The object is registered by a call to:

```
int ExprData.addExternalSelectionListener(ExternalSelectionListener esl)
```

which returns an integer 'handle' which can be used to subsequently un−register the object:

```
void ExprData.removeExternalSelectionListener(int esl_handle)
```

The example wrapper class provided in `WrapperDemo.java` includes an demonstration of this interface.

The `ExternalDataSink` interface

An `ExternalDataSink` is an object to which the data selection can be sent. Registering an `ExternalDataSink` object causes a new option to appear in the "Send to..." entry in the selection popup menu.

```
public interface ExternalDataSink
{
  // these methods specify what sort of data that class can accept
  public boolean likesSpots();
  public boolean likesSpotMeasurements();
  public boolean likesClusters();

  // these methods are used to pass data to the class
  public void consumeSpots(int[] spots_ids);
  public void consumeSpotMeasurements(int n_spots, int n_meas, double[][] data);
  public void consumeClusters(Cluster[] clusters);

  // this method returns the name that will be used for this sink in menu entries
  public String getName();
}
```

The example wrapper class provided in `WrapperDemo.java` includes an demonstration of this interface.

Add customised menu entries to the popup menu

It is possible to add entries to the popup menu (accessed in the main display window).

The `maxdView` class provides a public method:

```
void addPopupMenuEntry( String name, ActionListener al )
```

which will add entry with the specified "`name`". The `java.awt.event.ActionListener` will be invoked if this entry is selected by the user.

26 Glossary

Cluster
: A subset of all known data, that presumably share some attrbiute, or are otherwise similar in some way. Clusters can be used to model biological relationships such as families and pathways, and also numerical relationships, such as the output of classification or clustering algorithms. Clusters can be related to one another in a hierarchy (actually a N−ary tree)

Code fragment
: A piece of Java code which can interact with *maxdView*.

Data
: Any collection of one or more Measurements is referred to as Data.

Data type
: *Measurements* are given a data type, one of 'Absolute Expression', 'Ratio Expression', 'Probability', 'Error' or 'Unknown'. The data type determines which colour scheme will be used for the data, and in some cases, how the data will be interpreted by the application.

Filter
: A filter selectively disables some (or all) of the Spots in the *Data*

Gene
: In maxd terminology, a gene represents some biological sequence, which might be named, and might have textual annotations associated with it.

launched
: Plugin application are launched by selecting them from the menus.

maxd
: **M**anchester **A**rray E**X**press **D**atabase

maxdLoad
: A platform neutral interface for loading expression data (and its related experimental data) into a **maxd** database.

maxdView
: An architecuture for integrating visualisation and analysis techniques.

maxdSQL
: An implementation in SQL of the ArrayExpress database schema.

Measurement
: A one group of expression levels, error or probability measurements for a set of Spots

Plugin
: A Java class which can interact with *maxdView*.

Probe
: A named biological entity used to detect the presence of one or more genes.

Spot
: A single cell on a membrane, slide or array. Each Spot contains one Probe.

XML
: e**X**tensible **M**arkup **L**anguage, a platform independant, self describing, file format used for interchange of data.