

## MACE Tools

The Michigan Alliance for Cheminformatic Exploration

Kerby Shedden  
Department of Statistics  
University of Michigan  
kshedden@umich.edu

July 15, 2006

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Installation and system requirements</b>	<b>4</b>
2.1	Preparing Microsoft Windows for MACE Tools . . . . .	4
2.2	Preparing Linux/Unix for MACE Tools . . . . .	4
2.3	Preparing MacOS for MACE Tools . . . . .	5
2.4	Installing MACE Tools . . . . .	5
2.5	Executing MACE Tools scripts . . . . .	5
<b>3</b>	<b>Using MACE Tools to create and maintain a local copy of PubChem</b>	<b>6</b>
<b>4</b>	<b>Using the MACE Tools data access API</b>	<b>8</b>
4.1	Using the Python implementation of the data access API . . . . .	9
4.2	Other implementations of the data access API . . . . .	9
<b>5</b>	<b>Using the MACE Tools application development class library</b>	<b>10</b>
<b>6</b>	<b>Using the MACE Tools data visualization application</b>	<b>11</b>
6.1	Viewing data . . . . .	12
6.1.1	Summary statistics . . . . .	14
6.2	Highlighting and selecting sets of points . . . . .	15
6.2.1	Highlighting and selecting from the data view . . . . .	15
6.2.2	Highlighting and selecting from the controller . . . . .	16
6.2.3	Setting the match mode . . . . .	17
6.3	Growth inhibition profiles . . . . .	18
6.4	Structure profiles . . . . .	19

# Chapter 1

## Introduction

The MACE Tools are a set of software components and application programs centered on the PubChem collection of chemical structures and assay results ([pubchem.ncbi.nlm.nih.gov](http://pubchem.ncbi.nlm.nih.gov)). The MACE Tools have been developed to support the following types of work:

- The MACE Tools support the creation and maintenance of a local copy of PubChem and related data.
- The MACE Tools provide a uniform means of accessing the local copy of PubChem in commonly used high-level analysis packages such as Python, R and Matlab. The tools also provide data access routines written in C and Java to support the rapid development of analysis software centered on PubChem. Although the data access routines are provided for different programming languages, they follow a common API (application programming interface).
- The MACE Tools provide a set of reusable Python classes supporting the rapid development of PubChem-centered data analysis application software in Python.
- The MACE Tools provide several end-user application programs that provide powerful analysis capabilities and also serve to demonstrate the lower-level components of the MACE Tools. Presently, these application programs include an interactive version of the NCI/DTP COMPARE analysis, and an interactive visualization and dynamic graphing package.

When fully developed, the MACE Tools will be an open-source, highly documented, Python-based collection of reusable software components and application programs. It will facilitate the analysis of PubChem data in analysis packages such as R and Matlab, by providing a consistent means of data management and data retrieval for this large, heterogeneous collection of data. The MACE Tools will also support the rapid development and prototyping of specialized software for interactive data analysis of PubChem and similar collections of data.

Concretely, Mace Tools consists of:

- A set of Python scripts for constructing a local data collection of PubChem and related data. Under the present design, this data collection consists of a number of flat binary and text files. The binary files are accompanied by index files that allow random access to data for specific compounds, assays, etc. These files should not be accessed directly, since their layout may change in the future to accommodate growth in PubChem. Instead, the data access API described below should always be used to retrieve data from the local data collection.
- A data access API specifying function calls that efficiently retrieve slices of data from the local data collection. We provide a reference implementation of the API in Python, and also implementations in several other languages including C and R.

- A set of Python classes for common PubChem-related data management and analysis tasks. These classes are intended to support the rapid development of interactive data analysis application software for PubChem and other collections of high-throughput small molecule screening data.
- A set of application programs written in Python that demonstrate the capabilities of lower-level MACE Tools components.

## Chapter 2

# Installation and system requirements

The MACE Tools are entirely implemented in Python, but make heavy use of several libraries that are not part of the standard Python distribution. The MACE Tools should be usable on Windows, Linux and MacOS platforms that support Python and the required libraries.

*Note: As of July 2006 only a pre-release version of MACE Tools is available. The first official release is expected in September 2006.*

*Note: The pre-release version of MACE Tools was primarily developed on a Microsoft Windows system, with limited testing on Linux. Future releases will be tested on both Windows and Linux.*

### 2.1 Preparing Microsoft Windows for MACE Tools

Download and install the following packages:

- **Python** ([www.python.org](http://www.python.org)). The pre-release version of MACE Tools was developed using Python version 2.4.2.
- **Numeric Python** ([www.numpy.org](http://www.numpy.org)). The pre-release version of MACE Tools was developed using Numeric Python 0.9.8.
- **GTK+ for Windows** Currently a relatively easy way to install GTK+ for Windows is to use the installer provided by the Glade for Windows project ([gladewin32.sourceforge.net](http://gladewin32.sourceforge.net)). Download and install the GTK+/Win32 runtime environment. When installing, select C:\GTK as the install directory (otherwise you will need to modify the `gtkdir` variable in the `Configure.py` file of MACE Tools to point to the GTK+ install directory). The pre-release version of MACE Tools was developed using GTK+ for Windows version 2.8.
- **PyGTK** An installer for PyGTK for Windows is available from [www.pcpm.ucl.ac.be/~gustin/win32\\_ports](http://www.pcpm.ucl.ac.be/~gustin/win32_ports). Download and install both the `pygtk` and `pycairo` installers for Python version 2.4.

If you are only interested in using the MACE Tools to do data management and batch mode data analysis, you do not need GTK+ or PyGTK. These libraries are only used for interactive and graphical applications.

### 2.2 Preparing Linux/Unix for MACE Tools

Most Linux and Unix systems already have Python and GTK. Source tarballs and pre-compiled x86 Linux binaries for Numeric Python and PyGTK are available from the URL's given above.

## 2.3 Preparing MacOS for MACE Tools

No testing of the MACE Tools on MacOS has been carried out. However since Python and the key libraries are available for MacOS, it is likely to work with minimal modification.

## 2.4 Installing MACE Tools

To install MACE Tools download the zip archive from the MACE website ([www.stat.lsa.umich.edu/~kshedden/MACE](http://www.stat.lsa.umich.edu/~kshedden/MACE)) and unzip it. Then open the `Configure.py` file located in the `Common-Scripts` directory of the MACE Tools installation directory using a text editor. The `PATH` variable should point to a directory on the local drive capable of holding around 2GB of data. This directory will be referred to as the “workspace” below. On a Windows system, the `gtkdir` variable should point to the install directory for GTK+ (see above). On Linux/Unix systems the `gtkdir` variable is not used.

## 2.5 Executing MACE Tools scripts

**Note:** Before any MACE Tools scripts can be executed, see below for instructions on building a local data collection.

MACE Tools scripts are located in several subdirectories of the main MACE Tools install directory. Scripts can generally be executed on a Windows system by double clicking the `.py` or corresponding `.pyc` file. Alternatively a script can be run from the command prompt by typing `python` followed by the full script name (be sure that the directory containing `python.exe` is in the search path).

On a Unix or Linux system, MACE Tools scripts can generally be executed by typing “`python`” followed by the script name at the shell prompt. Again, be sure that the Python executable file is located in the search path.

## Chapter 3

# Using MACE Tools to create and maintain a local copy of PubChem

The higher-level components of MACE Tools draw data from a local copy of PubChem and related data. The scripts in the subdirectory `Database-Scripts` of the MACE Tools installation directory are used to build and maintain this data collection. The script `Generate_All.py` is the master script to build or rebuild the entire data collection. It transfers the XML bioassay result and compound structure files from the PubChem ftp site at NCBI, extracting from each file the information that is needed by MACE Tools.

**Note:** The `Generate_All.py` script will take roughly 8 hours to run with a fast internet connection. Also note that for the pre-release version of MACE Tools, you must have at least 1GB physical RAM to successfully run `Generate_All.py`. This restriction will be reduced in the near future.

The workspace (i.e. the directory given by the `PATH` variable in `Configure.py`) will contain a number of raw and processed files once the data collection build process is complete. As noted above, the layout of binary files is subject to change. In some cases, the text files located in the subdirectory `Processed-Text` may be of interest (generally all data should be accessed via the API described in chapter 4). Some text files of possible interest are:

- `assay-descriptions`: This is a text file summarizing the results of the individual assay description files provided by the screening centers (which are stored in the `Resources` subdirectory of the MACE Tools workspace). Each assay is described by several lines of text, with `//` separating the text blocks for different assays. The first line contains the PubChem assay ID, followed by the short text description of the assay, followed by the long text description of the assay, followed by one line for each result field. The result fields are described by a field index (1, 2, ...), then the name of the result field, then the text description of the result field.
- `Bioassay.gz`: Each line consists of a tab-delimited list of values describing the assay results for one substance in one assay. The first value is the PubChem assay identifier (AID), followed by the PubChem substance identifier (SID), followed by the result for each result field that is of `ival` or `fval` type. If the result data type is not `ival` or `fval`, then `-999` is given.
- `Frag_Table_Details`: This file describes a set of 2D that are matched against PubChem compounds. Currently, the fragments are either single atoms (e.g. `ATOM:O` for oxygen), augmented atom codes (e.g. `AAC:C:C1,C1,C2` for a carbon atom singly bonded to two other carbon atoms and doubly bonded to a third carbon atom), or rings (e.g. `RING:C2C1C2C1C2C1` for benzene). The total number of occurrences of each fragment in PubChem compounds is given in column 2, and the number of distinct compounds containing at least one occurrence of the fragment is given in column 3.

- `Topo1.gz`: This file describes the fragments (as listed in `Frag_Table_Details`, described above) that are contained in each compound. Each line consists of a sequence of tab-delimited integer numbers. The first number is the PubChem compound identifier (CID). The following numbers are paired, with the first value in each pair being the index of a fragment (i.e. the row in the `Frag_Table_Details` file describing the fragment, with the first line indexed as 0), and the second value of the pair being the number of occurrences of the fragment in the compound.

**Note:** *By default, the MACE Tools only calculate and save structure data for the compounds with assay data (i.e. compounds that are linked to at least one substance that has a bioassay result). In the future we will provide the option of saving structure results for all PubChem compounds (this will require a lot of local storage).*

**Note:** *Currently the MACE Tools do not support incremental updates to the local data collection. Therefore the entire data collection must be rebuilt to acquire new data from PubChem. Capabilities for incremental updating will be included in the near future.*

## Chapter 4

# Using the MACE Tools data access API

The data access API specifies the following functions:

**(SID,Z) = get\_bioassay(aid)** Returns a matrix  $Z$  of bioassay data for assay  $aid$ , with each row corresponding to a substance and each column corresponding to one of the assay result fields (which are ordered as specified in the XML bioassay description file). The return value  $SID$  is a map from substance identifiers to the row of  $Z$  where the data for the substance is stored.

**(SID, CL, Z) = get\_nci\_gi\_bioassays(min\_sid, max\_missing, impute\_missing)** Construct an array containing all molar log GI50 result fields for the NCI cell line GI50 assays. The columns of  $Z$  correspond to the cell line assays, with  $CL$  providing a map from cell line names to the corresponding column indices of  $Z$ . The return value  $SID$  is a map from substance identifiers to the row of  $Z$  where the data for the substance is stored. The argument  $min\_sid$  specifies the minimum number of assayed substances required for an assay to be included. The argument  $max\_missing$  specifies the maximum number of missing values allowed for a substance to be included as a column of  $Z$ . If the logical argument  $impute\_missing$  is true, then missing values are replaced with the substance mean across the observed assay values; if it is false then missing values are represented by -999.

**(PSID, GN, Z) = get\_expression\_data\_averaged()** Returns a list of Affymetrix probeset identifiers (PSID), a list of the corresponding gene names (GN), and a data matrix  $Z$  of gene expression values (averaged over the replicate experiments). Each row of  $Z$  contains the gene expression data for one probe set, for each cell line.

**(frag\_id, count) = fragments\_in\_compound(cid)** Returns a list of fragment identifiers contained in compound  $cid$ , along with the number of occurrences of each fragment in the compound.

**cid\_set = CIDs\_with\_fragment(frag\_id)** Returns a set of compound identifiers containing the fragment specified by  $frag\_id$ .

**gen = generate\_CIDs\_with\_fragment(cid)** Returns an iterator that allows the fragments to be accessed in sequence, with successive calls yielding the set of CIDs containing a given fragment.

**sid\_set = SIDs\_with\_fragment(frag\_id)** Returns a set of substance identifiers containing the fragment specified by  $frag\_id$ .

**sid\_list = SIDs\_with\_assay(aid)** Returns a list of substance identifiers for which data exist for assay  $aid$ .

The data access API also specifies the states of the following variables:

**SID** An array of substance identifiers, in increasing numerical order.

**CID** An array of compound identifiers, in 1-1 correspondence with SID. If a SID is not mapped onto a CID, zero is stored as the CID. This mapping is based on the “standardized form” mapping between CIDs and SIDs as provided by PubChem.

**CID\_from\_SID** A map from a substance identifier to the corresponding compound identifier.

**SIDs\_from\_CID** A map from a compound identifier to the corresponding list of substance identifiers.

**frag\_desc** A map from numerical fragment keys (0, 1, ...) to the corresponding text string describing the structure.

## 4.1 Using the Python implementation of the data access API

The Python implementation of the data access API takes the form of a class called PCData. To create an instance of PCData and, retrieve the data for bioassay aid 1, use the following.

```
PC = PCData()
(SID,Z) = PC.get_bioassay(1)
```

**Note:** Since a PCData instance uses a fair amount of RAM and takes a few seconds to construct, generally only a single instance of PCData should be created in an application and shared throughout the application.

The following code demonstrates how to use an iterator to efficiently cycle through a list of 2D structure fragments. For each fragment, the compound id's containing the fragment are compared to a compound set of interest denoted ID. The log odds ratio (with cell pseudocounts of one) assessing the association between the event that a compound contains the fragment and the event that the compound is in ID (a Python set) is calculated. If the log odds ratio exceeds 2 in magnitude, the fragment description is printed to the screen. The value of Tot is the total number of compounds in the compound collection.

```
for Q in PC.generate_CIDs_with_fragment():

    ## Cell counts.
    N11 = len(Q & ID)
    N12 = len(Q) - N11
    N21 = len(ID) - N11
    N22 = Tot - N11 - N12 - N21

    LOR = numpy.log(N11+1) + numpy.log(N22+1) - numpy.log(N21+1) - \
        numpy.log(N12+1)

    if abs(LOR) > 2:
        D.append("%-30s%6.2f\n" % (PC.frag_desc[k], LOR))
```

## 4.2 Other implementations of the data access API

Implementations of the data access API in C, R, and possibly other languages will be developed and documented here.

## Chapter 5

# Using the MACE Tools application development class library

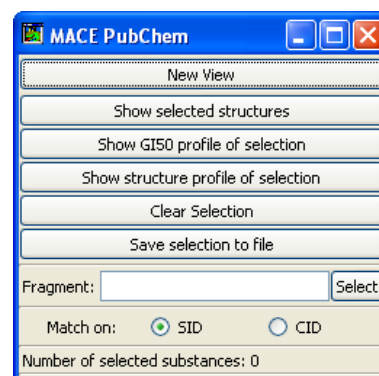
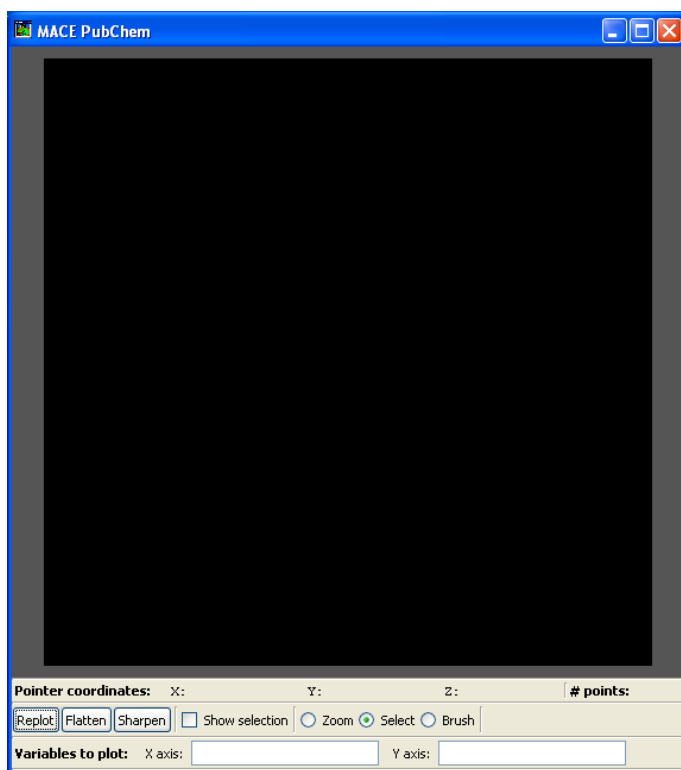
Much of the higher level functionality in the Python version of MACE Tools is organized into reusable classes. Classes exist for searching for data fields based on unstructured text queries, making bar charts, dynamic scatterplots, and structure fingerprinting. We have not yet fully developed and documented these classes as separate entities from the application software (described below) for which they were originally developed.

## Chapter 6

# Using the MACE Tools data visualization application

The MACE Tools data visualization program, temporarily called MACE-Vis, is intended to demonstrate the lower-level components of MACE Tools, and also provide valuable end-user analytic capabilities. It is designed to facilitate interactive and visual analysis of PubChem data and related data, aided by quantitative statistical methods where appropriate. MACE-Vis has some of the functionality of commercial scientific data visualization packages like Spotfire. It is much less developed than Spotfire and probably will remain less polished than such commercial efforts. However it has the advantage of being open source and adapted specifically to PubChem centered analysis.

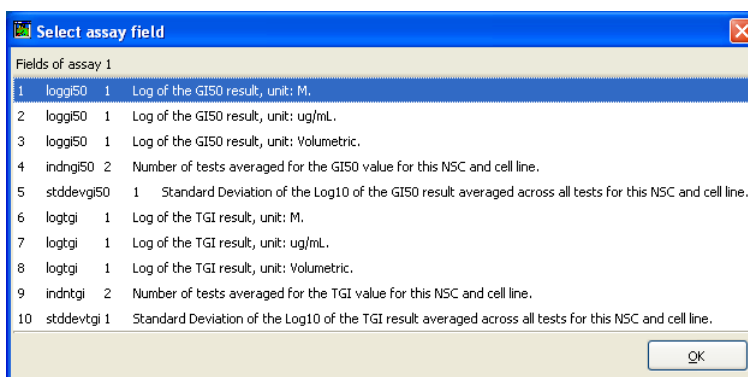
To run MACE-Vis, execute the `MACE-Vis.py` script in the `Viewer-Tools` directory of the distribution. You should see a data view window and a control window on the screen, as seen on the left and right below.



When running MACE-Vis there will always be one control window visible and one or more data view windows visible.

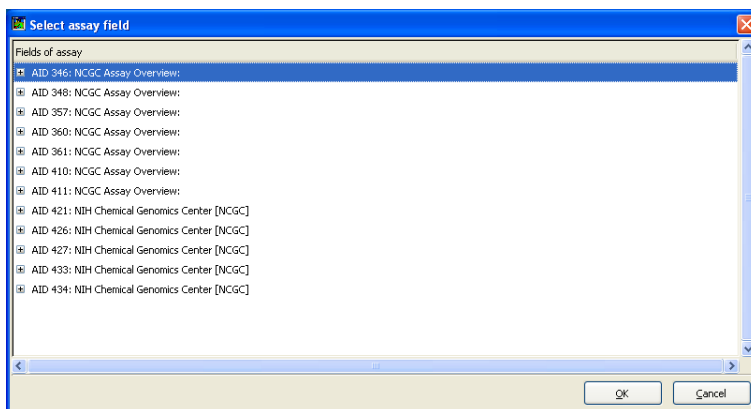
## 6.1 Viewing data

To generate a view of PubChem data, enter the name of a bioassay into the **X** axis and **Y** axis text boxes on the view window. You can specify an assay using a text string of the form "bioassay XX:YY," where XX is the assay identifier and YY is the result field number. Alternatively, you can provide only the assay identifier, and you will be prompted to select from the possible result fields. For example, if you type bioassay 1, you will be prompted as shown below.



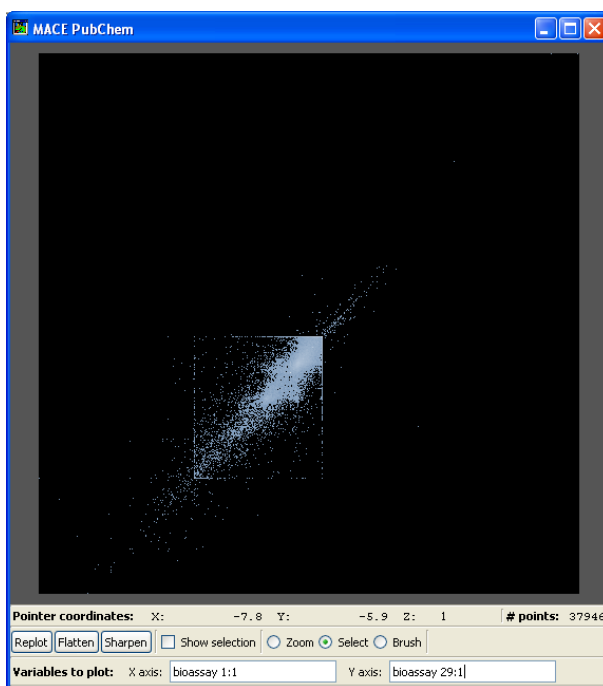
You should select one of the result fields that you wish to view, then press OK.

Another alternative for acquiring bioassay data is to enter a text string into the box, which will be used to search against descriptions of each bioassay. For example, if you enter "NCGC," you will be able to select from all the assays containing that string, as follows.



In addition to bioassay data, you may also plot chemical property data. Note that property data is not generated automatically during the data collection build, and must be entered separately using the chemical property import tool. The properties that can be plotted are limited to what has been imported.

Once two values have been selected for plotting, the data view window will appear as below, which shows the GI50 data for the H23 cell line plotted on the horizontal axis, with the GI50 data for the MALME-3M cell line plotted on the vertical axis.



The data in the above example are displayed as a density, so that if multiple compounds appear at the same point, the color is relatively more white. Bluer colors correspond to lower compound densities, with black denoting zero

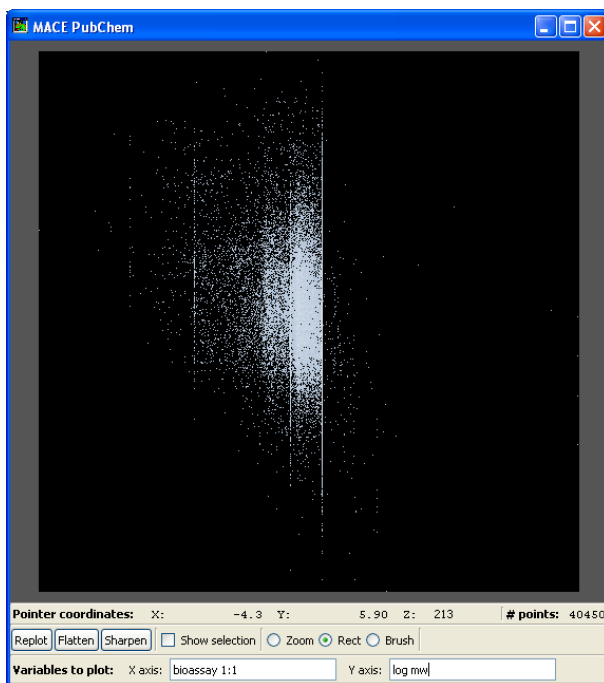
density. The pointer coordinates shown below the plot provide the X and Y coordinates, and the Z coordinate represents the number of compounds underlying a 3-pixel square window under the pointer. To maximize the color contrast and depth perception of the plotted density, adjust the flatness of the image using the “Flatten” and “Sharpen” buttons. Note that for many PubChem assays, certain points constitute floor or ceiling values, where large numbers of points are overplotted (e.g. -4 for the log GI50 results).

You can zoom in on the plot by clicking the right mouse button on the plot, and dragging the cursor to select a rectangular region. The points in the selected region will appear red before the zooming occurs. Press the “Replot” button to return to the original view.

You can view the 2D structure of any point showed on the plot by clicking on it with the left mouse button. The 2D structures are obtained from the NCBI web site, so you will need to have an internet connection to use this feature.

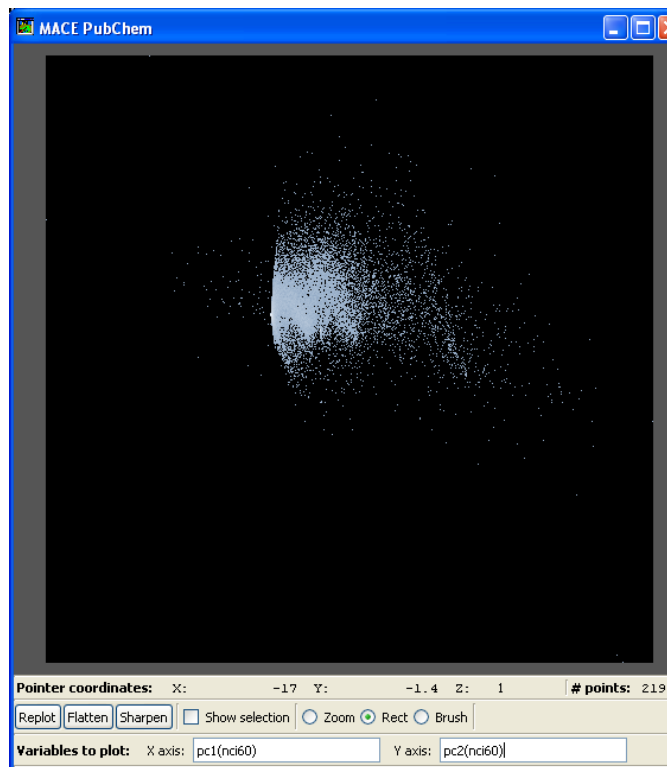
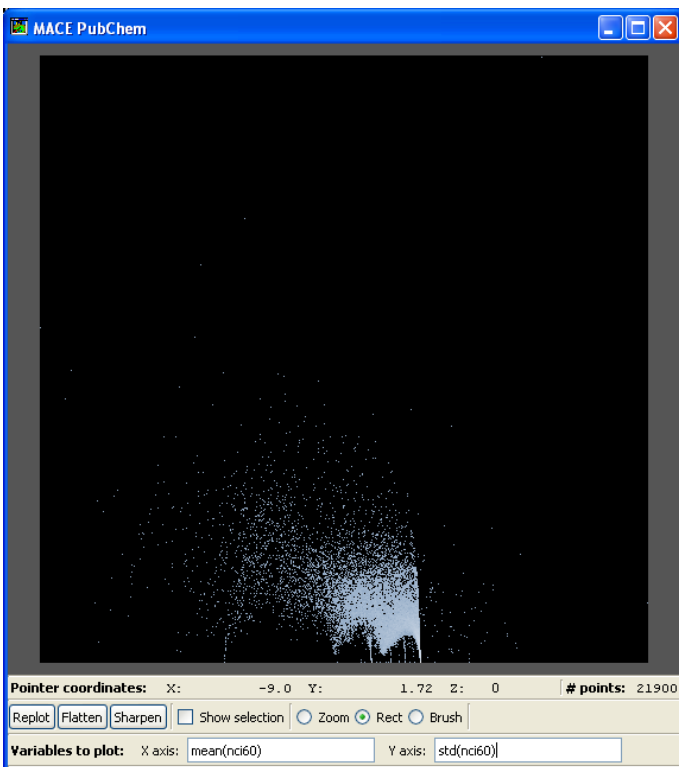
An arbitrary number of data views may be active at any time. To launch a new data view, use the “New View” button on the control window. Note that for a compound to appear in a data view, it must have data for both quantities (assay result or property) being plotted. However compounds appearing in one data view need not have data for fields plotted in another data view.

Property data may be viewed along with assay data. Property data names are not standardized by PubChem, but are specified using the property import tool. In the example below, log GI50 values for the H23 cell line are plotted against the logarithm of molecular weight. Note that the transforms log and sqrt (square root) may be used with any positive assay or property value.



### 6.1.1 Summary statistics

It is possible to build views based on summary statistics from multiple assays, including the mean, standard deviation, and principal components. It is particularly useful to do this with the NCI growth inhibition cell line assays. The images below show data views constructed from the mean and standard deviation of these assays (left) and from the first two principal components of these assays (right).



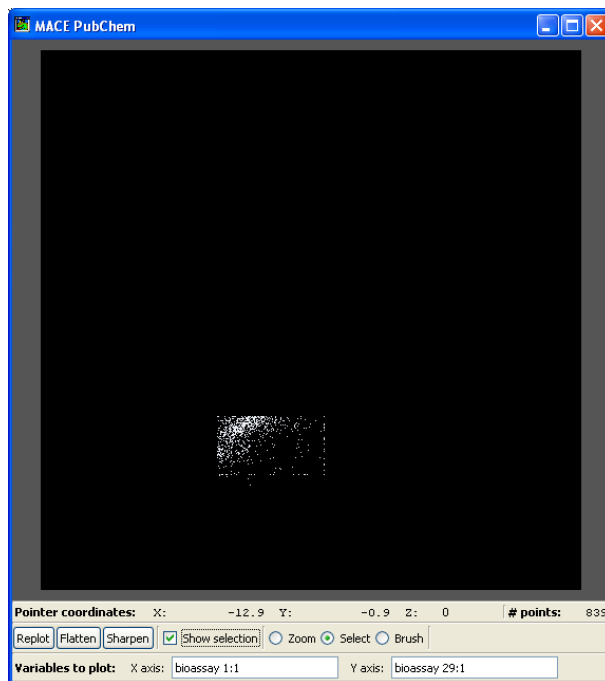
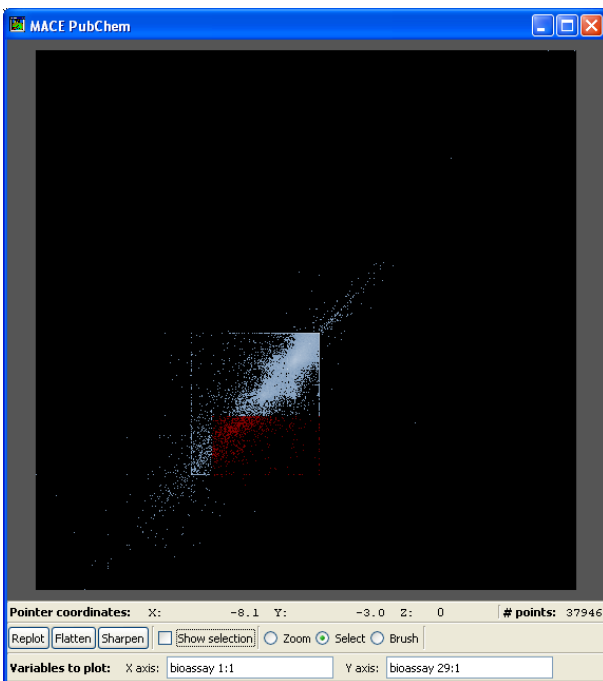
## 6.2 Highlighting and selecting sets of points

### 6.2.1 Highlighting and selecting from the data view

The key idea behind MACE-Vis is that a set of points may be “selected” based on a combination of bioassay results, topological properties, and chemical properties. This selection is generated by intersecting the “highlighted” points on all data views, and also intersecting with any topological constraints set at the controller window. We will begin by discussing how points are highlighted. This may be done using either a rectangular highlighting region, or using a brush. To set the highlighting mode, use the buttons near the bottom of the data view window. Brushing is used to highlight complex regions of points, while when the “Rect” button is pressed, a rectangular region is highlighted. Both types of highlighting require the right mouse button to be pressed. The highlighted points appear red in the data view where they were highlighted.

The “selection” is the intersection of all highlighted points in all data views, which is also intersected with any topological constraints that are specified at the control window (as discussed below). It is important to remember that there is only one set of “selected points” active at any point in time, but the highlighted points are specific to each data view. To see the selected points on a data view, press the “Show selection” button. When this button is pressed, only the selected points are shown. Otherwise all points are shown.

The images below show a set of points that have been highlighted (shown on the left in red), and the same points appearing as the selection (shown on the right, where the “Show selection” button has been toggled). In this case, no other selection has been made in another data view or at the controller, so the highlighted points are identical to the selected points.

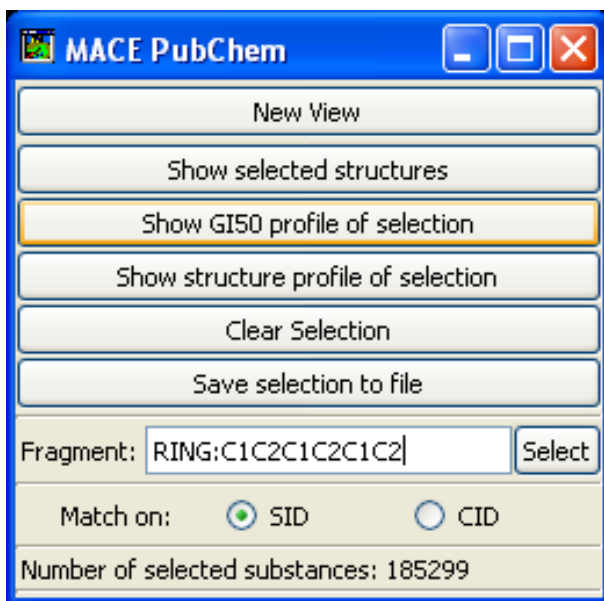


A sequence of highlighting operations in one window are each individually intersected with the global set of selected points. Therefore it is easy to reduce the set of selected points to nothing by highlighting different regions of the plot in sequence. Remember to press the “Clear Selection” button on the control window to begin a new selection. Note that the number of selected points is shown at the bottom of the control window. Compound and substance identifiers for the selected points can be saved to a text file using the “Save selection to file” button on the controller.

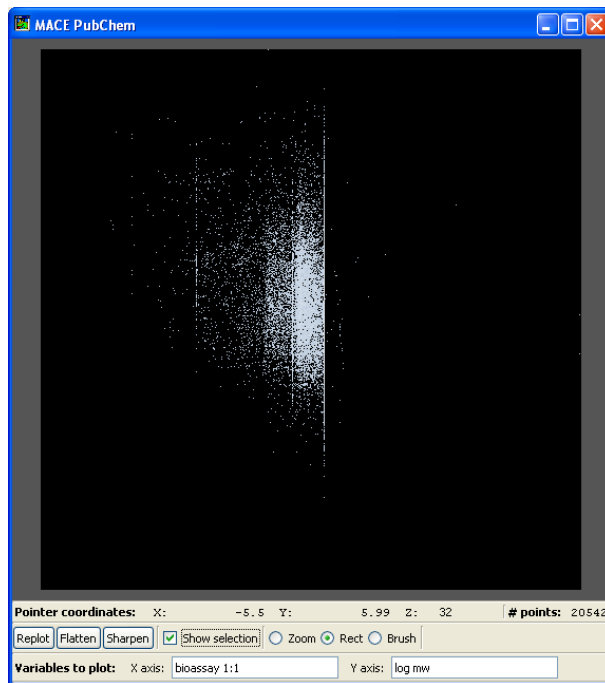
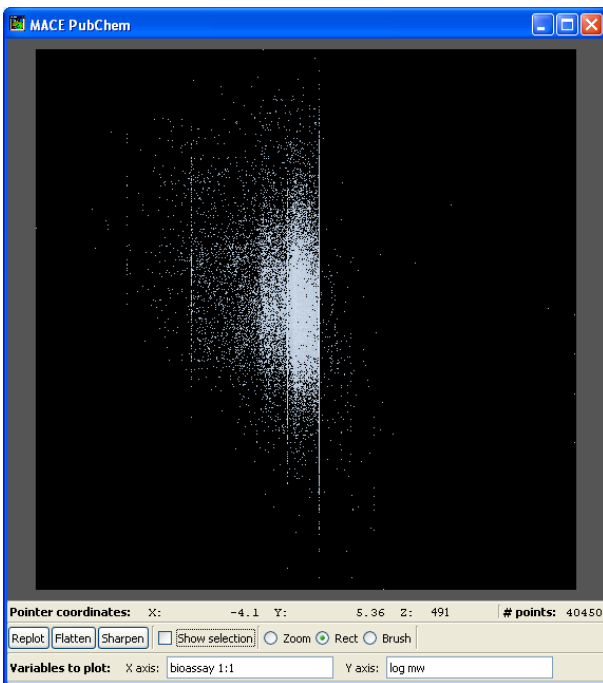
To reiterate, the highlighted region is specific to each window, while the set of selected points arises through linkages between windows. The intersection logic of the selection process can be used, for example, to select drug-like compounds that possess values in a target range for each of several properties (e.g. Lipinski’s rule). It then becomes possible to see how these compounds compare to less drug-like compounds in assay space.

### 6.2.2 Highlighting and selecting from the controller

The selection may be modified from the controller to select points with specific 2D structure fragments. The particular fragments available for selection are listed in the `Frag_Table_Details` file of the `Resources` directory of the MACE Tools distribution. For example, in the controller below a benzene ring has been selected, and it is shown that 185299 compounds in the overall database have a benzene ring (this is around half of the total).



Continuing with this example, the images on the left and right below show log molecular weight plotted against H23 GI50 data for all compounds (left) and the compounds containing benzene rings (right). It is seen that of 40450 compounds having assay and molecular weight data, 20542 have benzene rings.



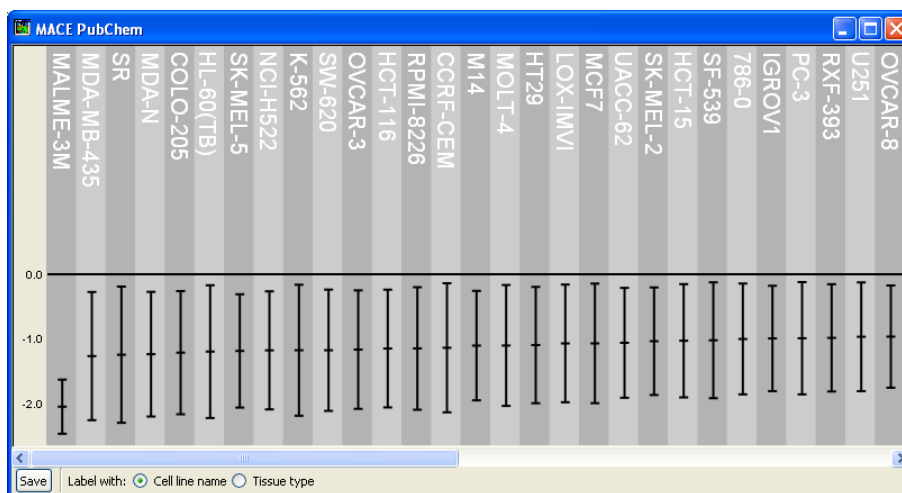
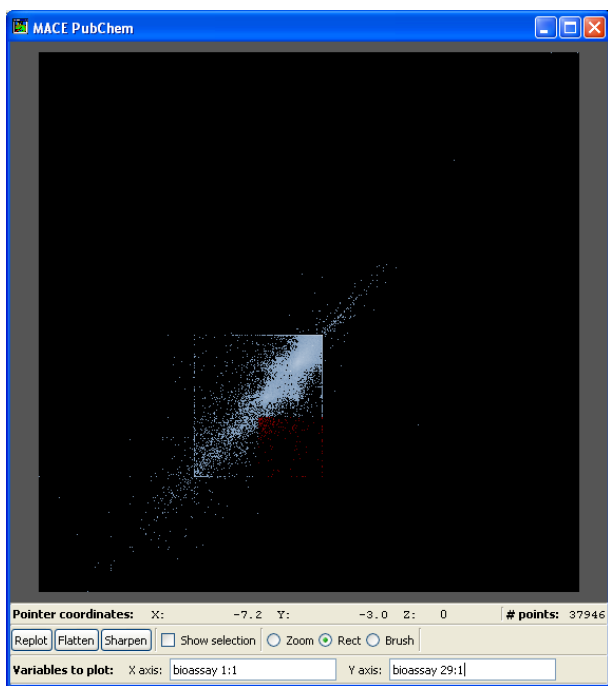
### 6.2.3 Setting the match mode

By default, the selection is formed by intersecting the highlighted regions based on common substance identifiers (SID's). To match on compound identifiers (CID's), select the "Match on CID" button on the controller. This

enables comparison of assay results from different screening centers that may assay the same compounds under different SID's. However note that for viewing data where this is not an issue (e.g. different fields of the same assay, or an assay against a property) it is better to link on SID's since otherwise replicated data (multiple assayed substances that link back to a common CID) are lost.

### 6.3 Growth inhibition profiles

A growth inhibition profile of the selected compounds is shown by pressing the "Show GI50 profile of selection" button on the controller. A set of selected points, and the corresponding GI50 profile are shown below. Note that in the GI50 profile window, the results are too big to display in the window, so the scrollbar is used to view different parts of the chart.



The GI50 profile chart shows the mean  $\pm$  one standard deviation of the GI50 results (molar concentration units) for the selected compounds. The means for each cell line assay are translated relative to the overall mean for that assay. Thus a result of zero indicates that the selected compounds have the same average GI50 value as the set of all compounds, while positive values indicate lesser than average activity for the selected points, and lower values indicate greater than average activity for the selected points. The cell lines are sorted so that the cell lines that are most differentially sensitive to the selected compounds appear on the left while the compounds that are most differentially resistant to the selected compounds appear on the right. The option is provided to annotate the cell lines by their name or by their tissue type.

## 6.4 Structure profiles

A structure profile of the selected compounds is shown by pressing the “Show structure profile of selection” button on the controller, then pressing the “Calculate” button in that window. The results shown below are from the same selection used to illustrate the GI50 profile above.

Structure Descriptor	Log Odds Ratio	Number of Selected Compounds	Number of Non-Selected Compounds
RING:C1N1C1C1C1N2	5.82	1	10
RING:N1C2N1N1N2	5.22	1	19
AAC:C:H1,H1,Sn6,C1	5.17	1	20
AAC:Cl:Pb1	5.12	0	10
RING:C2N1C2N1C1O1	5.12	0	10
AAC:C:H1,H1,H1,B1	5.12	0	10
AAC:C:H1,H1,S1,Br1	5.12	0	10
RING:C1S1C2N1N1C2	5.12	0	10
RING:C1C1N1N1N1	5.12	0	10
RING:C1S1N1C2N1	5.12	0	10
AAC:Ru:N6,N6,N6,N6,N6,N6	5.12	0	10
RING:C1N1N2C1N1N1	5.12	0	10
AAC:Ag:N6,O6	5.12	0	10
RING:C1S1C1N1S1	5.12	0	10
AAC:S:N1,N2,O1,O2	5.12	0	10
AAC:O:Cl,Ir6	5.12	0	10
AAC:Cl:Cr1	5.12	0	10
AAC:S:Co6,C1	5.12	0	10
AAC:O:Cl,U6	5.12	0	10
RING:N1C1N1P1N1P1	5.12	0	10
AAC:S:P1,P1	5.12	0	10
AAC:C:H1,P1,C11,C11	5.12	0	10
RING:O1C1C1C1O1S1	5.12	0	10
RING:C1S1C1C1O1C2	5.12	0	10
AAC:Cl:Zr1	5.12	0	10

A list of structure fragments is provided that are differentially present or absent in the selected compounds compared to all other compounds. The log odds ratio statistic is used to assess differential presence of a fragment. The results are displayed in decreasing order of their log odds ratios. The first column of the text output is the structure descriptor, the second column is the log odds ratio (positive values indicate greater than expected frequency of occurrence in the selected compounds while negative values indicate less than the expected frequency of occurrence). Column 3 is the number of selected compounds containing the fragment and column 4 is the number of non-selected compounds containing the fragment.