

An Interactive SDF Viewer

RDKit UGM 3

23-Oct-2014

Axel Pahl

- Medicinal Chemist (12+ yrs. experience)
 - still working in the lab
- Linux / OpenSource Enthusiast
- Programming Skills
 - Python, Postgresql, RDKit, HTML
 - Pipeline Pilot, KNIME
- Married, two children (10 & 12 yrs. old)



CV Summary

- 1998 PhD in Organic Chemistry
- 1998 - 2012 Research at Solvay Pharmaceuticals / Abbott Products (Hannover)
 - 2 yrs. Chemical Pilot Plant (GMP)
 - 9+ yrs. Medicinal Chemistry
 - Metalloprotease Inhibitors (ECE, NEP)
 - HCV Polymerase Inhibitors (NS5B palm II site)
- 2013 - today AVIRU Project (TU Munich)
 - Virulence Inhibitors (a.o. ClpP) for MRSA

SDF Viewer

- SD File Viewer
- Display properties that are stored in the SD file
- Interactive visualization of the properties
 - color data points by category properties (e.g. compound family) (slide 8)
 - compare a subset (e.g. from a substructure search) to the whole SD file graphically (slide 9)
 - display molecules from data points
 - clicking a data point once displays the record in the SDF Viewer
- Create subsets / new SD files
 - fact or substructure (SMILES) searches
 - manual selection of molecules, then <create db from selection>
 - in the SDF Viewer directly or
 - by clicking data points twice in the scatter plots
 - subsets can always be saved as new SD file
 - program sessions can be stored
 - special behaviour for SD files saved from Aldrich or Chemspider (slide 10)
- Written in Python (2.7), RDKit (2014_03_1), PyQt (4.11.2), Matplotlib (1.4.0)
 - platform-independent (but written on and for Linux)
 - runs on Linux and Windows (Mac not tested)

Example Data Set

- Excerpt from the Sutherland Data Set
J. Chem. Inf. Comput. Sci. 2003 (43) 1906 - 1915
(as found on <http://www.cheminformatics.org/datasets/#qsar>)

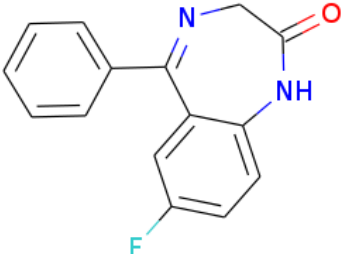
Histograms - For a Quick Overview

ChemDB

database: ugm2014_bzr

current db: ugm2014_bzr querytype: fact query: stored in: start

db	origin	type	query	hits	remark
1	ugm2014_bzr			134	original data...



1	2
1	rotb 1
2	set 1
3	tpsa 41
4	formula C15H11FN2O
5	hba 3
6	hbd 1
7	molwt 254.26
8	logp 2.62
9	family A.1
10	name Ro05-3061
11	molid 1
12	plC50 7.40

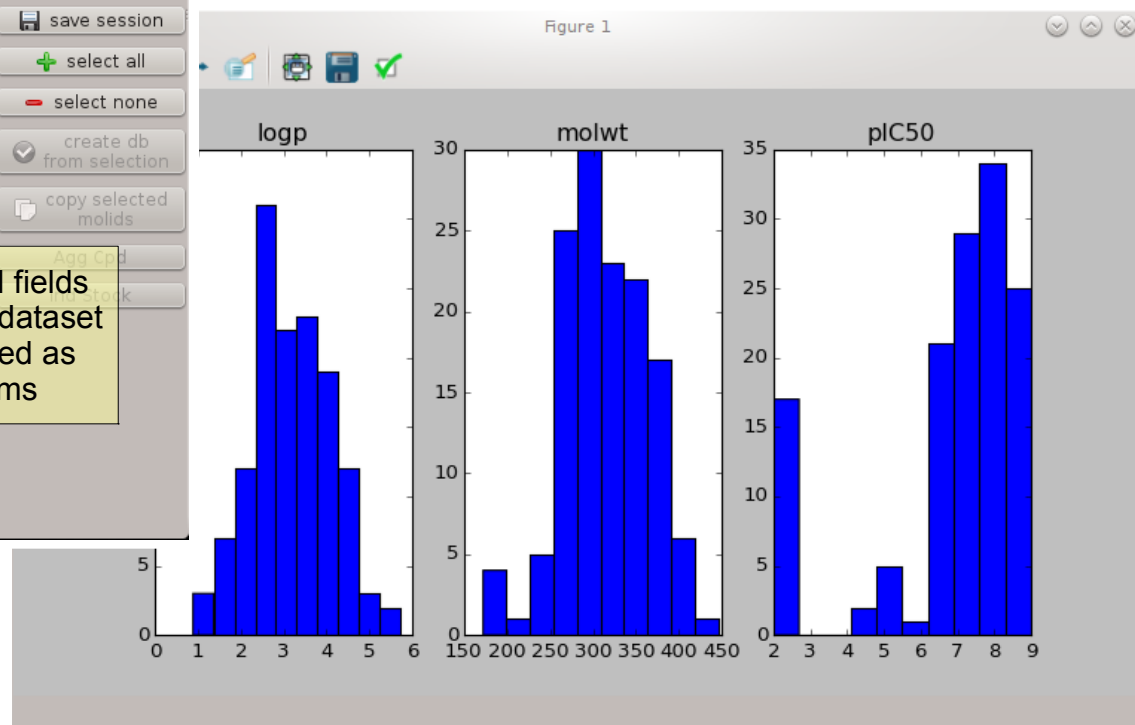
1 of 134

selected highlight Lipinski

histogram color by scattermatrix <none> compare to scattermatrix2

The selected fields of the whole dataset are displayed as histograms

Histograms

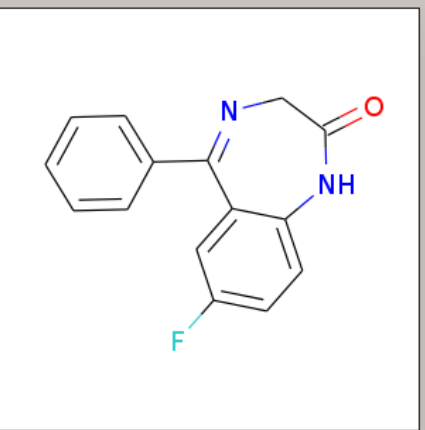


Scatter Matrix

database: ugm2014_bzr

current db: ugm2014_bzr querytype: fact query: stored in: start

db	origin	type	query	hits	remark
1	ugm2014_bzr			134	original data...



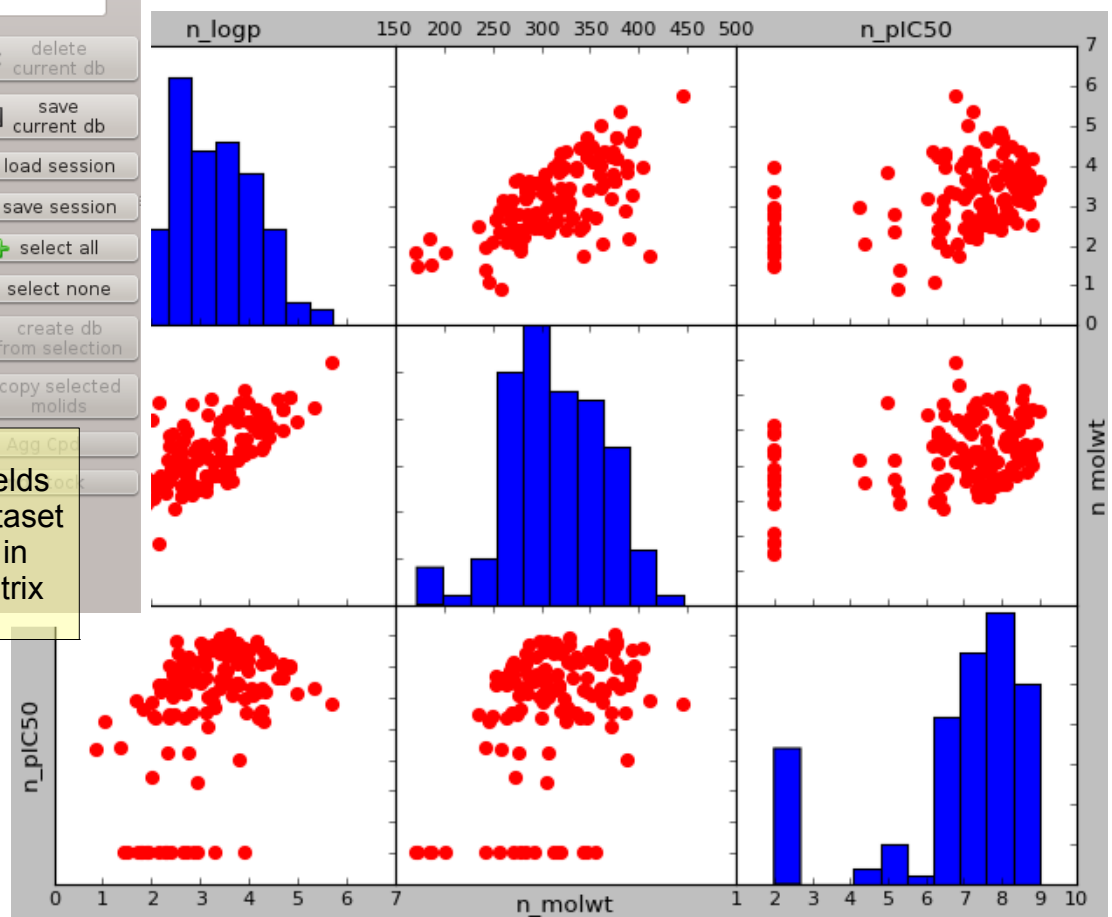
	1	2
1	rotb	1
2	set	1
3	tpsa	41
4	formula	C15H11FN2O
5	hba	3
6	hbd	1
7	molwt	254.26
8	logp	2.62
9	family	A.1
10	name	Ro05-3061
11	molid	1
12	pic50	7.40

1 of 134

selected highlight Lipinski

histogram scattermatrix color by <none> compare to scattermatrix2

Scatter Matrix: all selected fields are plotted against each other, with histograms on the diagonale



The selected fields of the whole dataset are displayed in the scatter matrix

Scatter Matrix - Color by Property

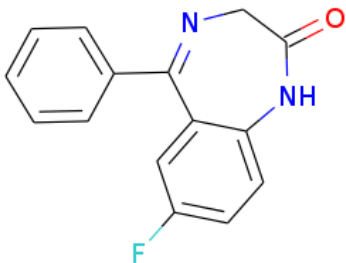
- color data points by category properties (e.g. compound family)

ChemDB

database: ugm2014_bzr [load] [load prev] [load next]

current db: ugm2014_bzr querytype: fact query: stored in: [start]

db	origin	type	query	hits	remark
1	ugm2014_bzr			134	original data...

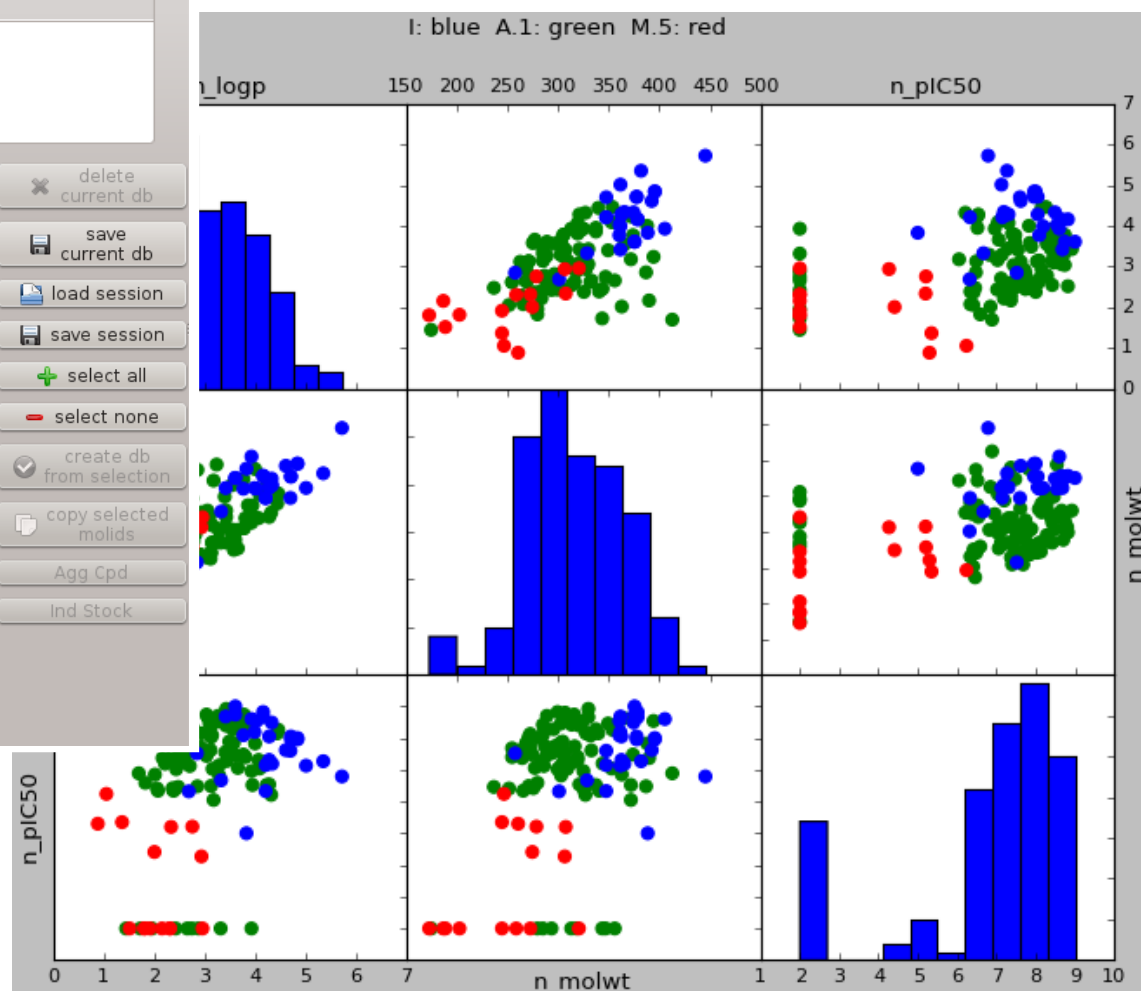
O=C1CN=C(c2ccccc2)c3cc(F)ccc3N1

	1	2
1	rotb	1
2	set	1
3	tpsa	41
4	formula	C15H11FN2O
5	hba	3
6	hbd	1
7	molwt	254.26
8	logp	2.62
9	family	A.1
10	name	Ro05-3061
11	molid	1
12	pic50	7.40

[delete current db] [save current db] [load session] [save session] [select all] [select none] [create db from selection] [copy selected molids] [Agg Cpd] [Ind Stock]

1 of 134 [selected] [highlight Lipinski]

histogram color by scattermatrix s_family compare to scattermatrix2



Compare a Subset to the Original SD File

- compare a subset (e.g. from a substructure search) to the whole SD file graphically

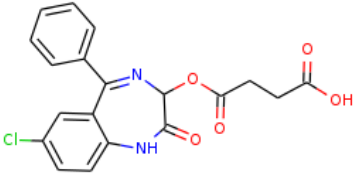
ChemDB

database: ugm2014_bzr

current db: esters querytype: substruct query: COC=O stored in: esters

start

	db	origin	type	query	hits	remark
1	ugm2014_bzr				134	original data...
2	esters	ugm2014_bzr	substruct	COC=O	12	

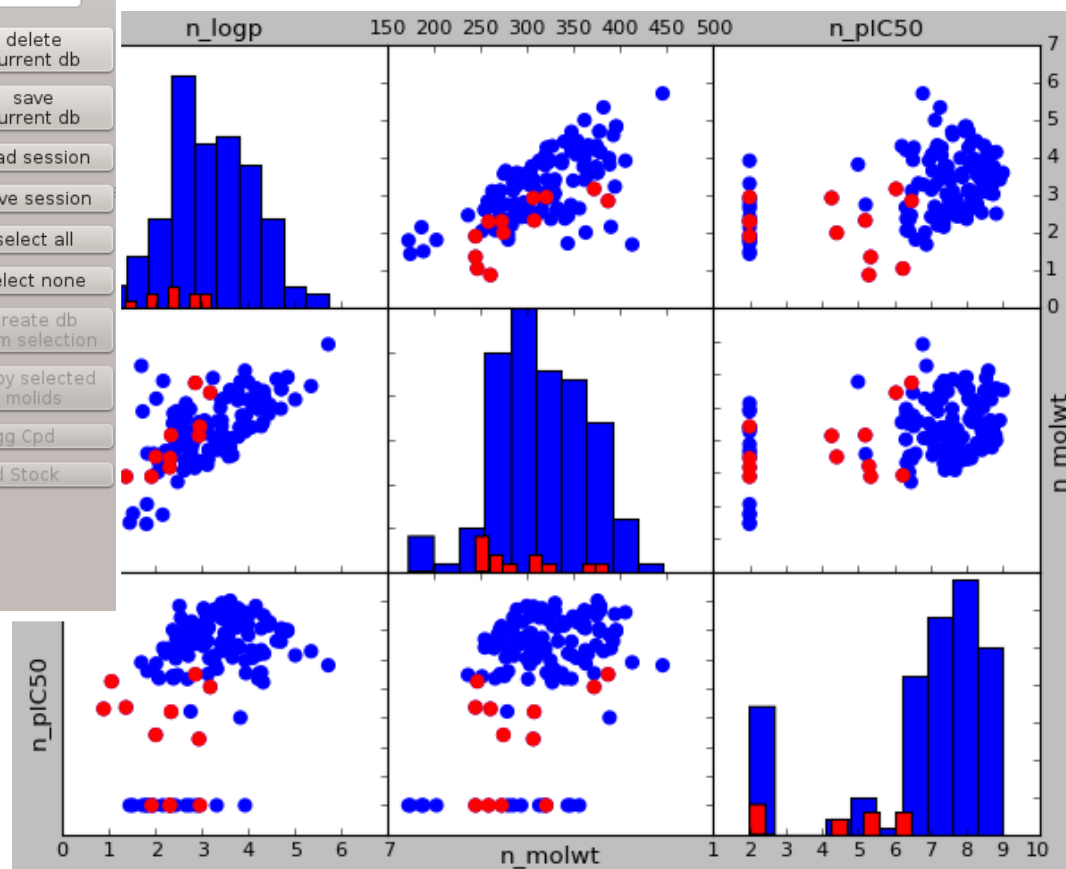
O=C(O)CCOC1=NC2=CC(=CC=C2C1=O)N3C(=O)C=CC=C3Cl

	1	2
1	rotb	5
2	set	2
3	tpsa	105
4	formula	C19H15ClN2O3
5	hba	7
6	hbd	2
7	molwt	386.79
8	logp	2.86
9	family	A.1
10	name	Oxazepam-h...
11	molid	88
12	pIC50	6.48

delete current db
save current db
load session
save session
select all
select none
create db from selection
copy selected molids
Agg Cpd
Ind Stock

1 of 12 selected highlight Lipinski

histogram scattermatrix color by <none> compare to ugm2014_bzr
scattermatrix2



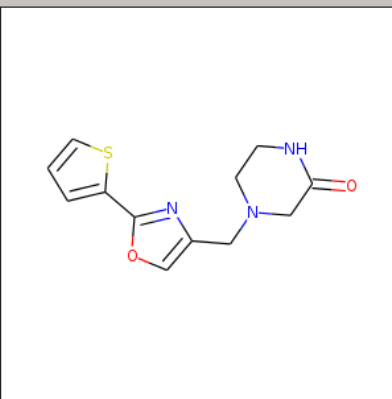
Link to Online Records

- special behaviour for SD files saved from Aldrich or Chemspider
- clicking the button opens the Chemspider (or Aldrich) record in a browser

database
cs_vs_res_140508

current db: cs_vs_res_140508 querytype: fact query: k_molid <= 100 stored in: first100

db	origin	type	query	hits	remark
cs_vs_res_1...				658	original data...



	1	2
3	vs_order	0
4	tpsa	58
5	vs_sel	1
6	vs_score_delt	0.20
7	vs_num_atm	18
8	date	20140508
9	vs_score_max	7.70
10	formula	C12H13N3O2S
11	hba	5
12	hbd	1
13	molwt	263.32
14	vs_score_class	low
15	logp	1.33
16	vs_score2	7.70
17	molid	5
18	vs_score1	7.50

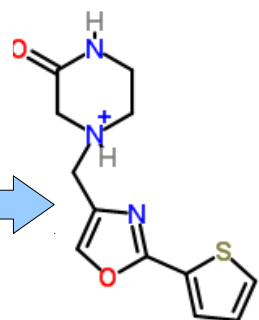
6 of 658 selected highlight Lipinski

Agg Cpd

ChemSpider

ChemSpider
Search and share chemistry

About | More Searches | Web APIs | Help



2D 3D Save Zoom

Charge

3-Oxo-1-([2-(2-thienyl)-1,3-oxazol-4-yl]methyl)piperazin-1-ium

ChemSpider ID: **7254021**

Molecular Formula: C₁₂H₁₄N₃O₂S

Average mass: 264.322906 Da

Monoisotopic mass: 264.080109 Da

▼ Systematic name

3-Oxo-1-([2-(2-thienyl)-1,3-oxazol-4-yl]methyl)piperazin-1-ium

► SMILES and InChIs

► Cite this record

sdf_tools Provides a Set of Convenience Functions

sdf_tools.py:

```
def load_sdf(fn="testset.sdf", large_sdfdb=False):
```

- returns the SD file as a list of RDKit Mol objects
- large SD files can be returned as file objects
- most other functions in the module can work with both so that arbitrarily large SD files can be handled
a typical scenario is to perform a search on a large SD file as a file object and handle the smaller search result as list in memory. A substructure search in 120 000 compounds (315MB file) takes about 70s on my laptop.

```
def write_sdf(sdf_list, fn, conf_id=-1):
```

```
def list_fields(sdf_list_or_file):
```

```
def show_db(db_list, force=False):
```

```
def merge_prop_from_file(sdf_list, fn, prop):
```

```
def remove_props(mol_or_sdf_list, props):
```

```
def calc_props(mol_or_sdf_list, counterprop="k_molid", dateprop="k_date", include_date=True, force2d=False):
```

```
def sort_sdf(sdf_list, field, reverse=True):
```

```
def factsearch(sdf_list_or_file, query, invert=False, max_hits=2000, count_only=False, sorted=True,  
               reverse=True):
```

```
def substruct_search(sdf_list_or_file, smarts, invert=False, max_hits=5000, count_only=False, add_h=False):
```

```
def similarity_search(sdf_list_or_file, smarts, similarity=0.8, max_hits=2000, count_only=False):
```

... and more

- Can be used e.g. in an IPython Session
- The viewer uses a subset of these functions
- Functions that return the same number of records (sorting, renaming fields, ...) modify the list in place
- Functions that modify the number of records (e.g. substructure search) return a new list
- Plotting of properties is possible just like in the viewer (but without interactivity)

Example of Plotting With sdf_tools in the IPython Notebook

```
In [1]: from __future__ import absolute_import, division, print_function
```

```
from rdkit.Chem import AllChem as Chem
from rdkit.Chem import Draw
from rdkit.Chem.Draw import IPythonConsole
Draw.DrawingOptions.atomLabelFontSize = 18
```

```
In [2]: import sdf_tools as sdft
```

```
> interactive session, trying to import display_png from IPython... success!
> found environment var /home/apl/aviru/db_reports
```

```
In [3]: %pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

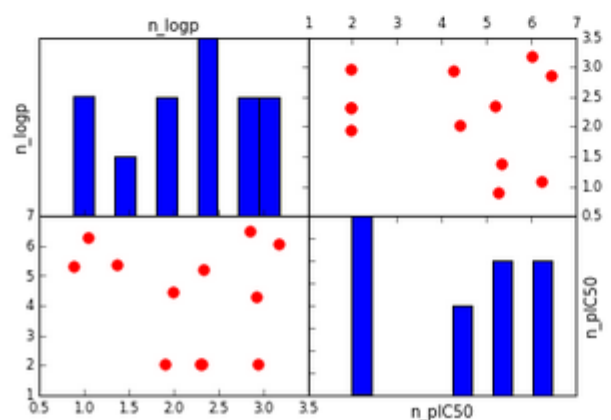
```
In [8]: sdf_list = sdft.load_sdf("sdf/testset.sdf")
```

```
> sdf sdf/testset loaded with 134 records.
```

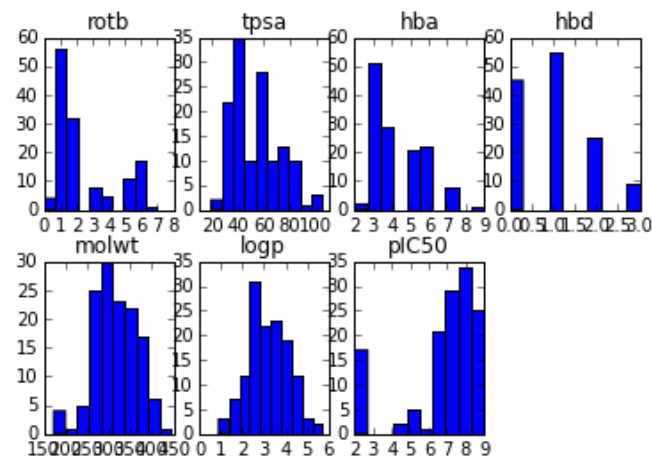
```
In [9]: esters = sdft.substruct_search(sdf_list, "COC=O")
```

```
> searching...
processed: 134 found: 12
done.
```

```
In [10]: sdft.show_scattermatrix(esters, ["n_pIC50", "n_logp"])
```



```
In [8]: sdft.show_hist(db)
```



Interesting Bits and Caveats

- The images of the structures are generated on-the-fly
 - stored as a temporary png file
 - and displayed in a QtGui.QLabel as a QtGui.QPixmap

```
with tempfile.NamedTemporaryFile() as f:  
    Draw.MolToFile(self.curr_db[self.curr_db_mol_index],  
                   f.name, imageType="png")  
    self.label_molimage.setPixmap(QtGui.QPixmap(f.name))
```

Interesting Bits and Caveats

- The Matplotlib data point picker and selector (*onpick*, next slide)
 - for the data picker to work in the viewer, it has to be taken care of that usually not every compound is tested in every assay
 - for this a dict of molids is kept per individual plot in the scatter matrix (`self.axes_molindex_dict`)

The Matplotlib Data Point Picker and Selector

```
class App(QtGui.QMainWindow, Ui_MainWindow):
    ...
    def onpick(self, event):
        x = event.mouseevent.xdata
        y = event.mouseevent.ydata
        ind = event.ind
        ax = event.mouseevent.inaxes
        if ind:
            ind = ind[0]
            if self.curr_db_index != 0:
                # use root database to display structures
                self.init_curr_db(self.db_name_order[0])

            molid = self.axes_molindex_dict[ax][ind]
            molindex = self.db_mol_index[molid]

            if self.curr_db_mol_index != self.db_mol_index[molid]:
                self.curr_db_mol_index = molindex
                print "x= %s, y= %s" % (x, y)
                print "index %2d: molid %s" %(molindex, molid)
                self.display_mol()

            else:
                # toggle the selection checkbox only on the second click on the data point:
                # self.curr_db_mol_index == self.db_mol_index[molid]
                # (this automagically triggers on_check_rec_selected_stateChanged)
                self.check_rec_selected.setChecked(not self.check_rec_selected.isChecked())

    @QtCore.pyqtSlot()
    def on_btn_scatter_clicked(self):
        colorby = str(self.combo_colorby.currentText())
        print "colorby:", colorby
        if colorby == "<none>":
            colorby = None
        selected_fields = self.get_selected_fields()
        self.statusbar.showMessage("generating scatter matrix...", 2000)
        self.fig, self.axes_molindex_dict = qdb.show_scattermatrix(self.curr_db,
                                                                    fields=selected_fields, colorby=colorby, mode="gui")
        self.statusbar.showMessage("scatter matrix generated.", 2000)
        self.fig.canvas.mpl_connect('pick_event', self.onpick)
        self.fig.show()
```

Thanks



Questions ?

- Code can be found on GitHub
(https://github.com/apahl/sdf_viewer)