

“Dataflow vs. Scripting Languages”

Andrew Dalke

“The Evils of KNIME”

Moral

Propose your own talk ...

... or one will be assigned to you

Me and KNIME

KNIME Desktop for Mac OS X

Note: this is still highly experimental. KNIME 2.1 is based on Eclipse 3.4.2, whereas for a functioning KNIME Mac OS X integration we needed to use Eclipse 3.5. Hence the update site is not working due to known problems and incompatibilities with p2. Therefore this version includes all KNIME plugins. If you encounter any bugs, please let us know. Note, however, that we will likely not be able to seriously support this version before we switch over to Eclipse 3.5.

Good for me, but ...

Just how many pharmas use Macs?

IBM's Data Explorer - OpenDX

The image displays the Visual Program Editor for Streamline.net, showing a data flow diagram and a 3D visualization of streamlines.

Visual Program Editor (Top Window):

- Menu: File, Edit, Execute, Windows, Connection, Options, Help
- Categories: (ALL), Annotation, DXLink, Debugging, Flow Control, Import and Export, Interactor, Interface Control, Macros, Options, RAMS, Realization, Rendering
- Tools: (ALL) Tools: AmbientLight, Append, Arrange, ArrangeMember, Attribute, AutoAxes, AutoCamera, AutoColor, AutoGlyph, AutoGrayScale, AutoGrid, AutoScale, BSpline, Band, BandColors, BarChart, BlackScholes, Camera, CappedIsosurfaceMar, Caption, Categorize, CategoryStatistics
- Diagram Components: Selector, Sequencer, Switch, Switch, streamline_head, slab_position, label
- Text: "This page allows the user to choose whether the sequencer should control the heads of the streamlines or the starting points of the streamlines (the slab position)"

Sequence Control (Middle Window):

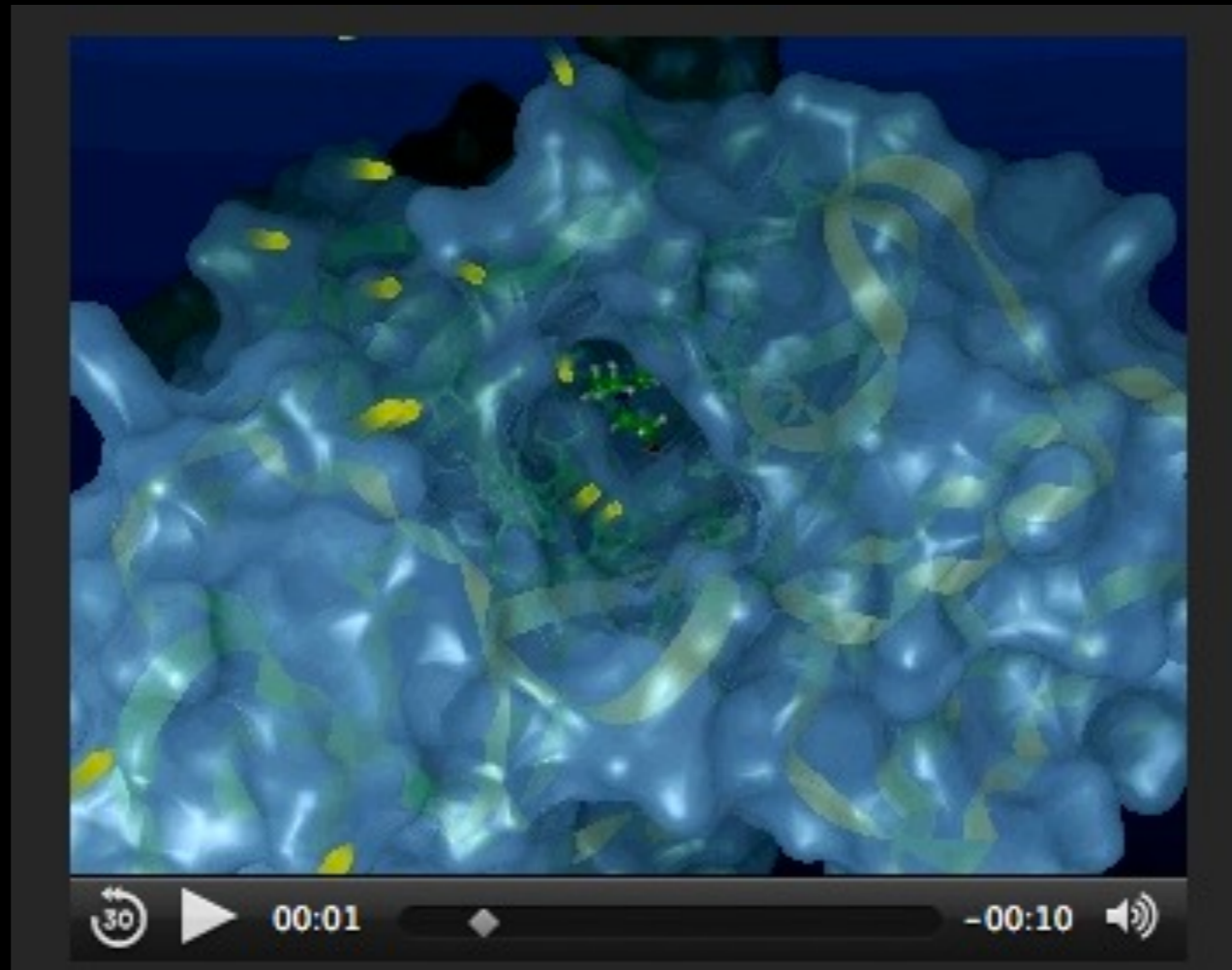
- Buttons: Undo, Redo, Play, Stop, Pause, Previous, Next
- Value: 12

View Control (Bottom Window):

- Buttons: Undo Ctrl+Z, Redo Ctrl+Y
- Mode: None
- Set View: Top
- Projection: Orthographic
- View Angle: 30.000
- Buttons: Close, Reset Ctrl+F

3D Visualization (Bottom Right Window):

- Menu: File, Execute, Windows, Connection, Options, Help
- Image: A 3D visualization of streamlines within a wireframe box. The streamlines are colored based on time, showing a complex, swirling pattern.
- Text: "colors are based on time sequencer changes head of streamlines"



`` Visualizing Enzyme Electrostatics with IBM Visualization Data Explorer". R. Gillilan and D. Ripoll, (1994). In `` *Data Visualization in Molecular Science.*" J. Bowie Ed. Manning Publications Co. and Addison-Wesley Publishing Company Inc. Chapter 4, pp. 61-81.

Michel Sanner's ViPEr (ca 2002)

The screenshot displays the ViPEr software interface, which is a visual programming environment for molecular visualization. The main window is titled "ViPEr" and contains a menu bar (File, Edit, Networks) and a toolbar with buttons for "Load Network", "Save Network", and "Run Network". Below the toolbar are tabs for "Std" and "User", and a library of components categorized into "mapper", "input", "output", and "filter".

The central workspace shows a network of interconnected nodes and connections. The workflow starts with "Read Molecule" (loading "1cm.pdb") and "Assign Radii" (using "united radii"). The network includes nodes for "Select Atoms", "Extract Atom Property" (radius), "Dial" (set to 0.56), "Array Ufunc2" (multiply), "Extract Atom Property" (coords[0]), "Color", "MSMS", "IndexedPolygons", "CPK", "Viewer", "Filter Image" (set to "CONTOUR"), "grab Image", "Scale", and "Show Image".

On the right side, there is a "Color Map" panel showing a color gradient from blue to red. Below the gradient are input fields for "Min: 0.0" and "Max: 255.0", and buttons for "Hue", "Saturation", "Brightness", "Opacity", "Reset", "Read", "Write", and "Dismiss".

At the bottom, there are two windows: "Show Image" displaying a 2D contour plot of the molecule, and "Viewer" displaying a 3D CPK model of the molecule with a color gradient from blue to red.

THE ON-LINE GRAPHICAL SPECIFICATION OF COMPUTER PROCEDURES

by

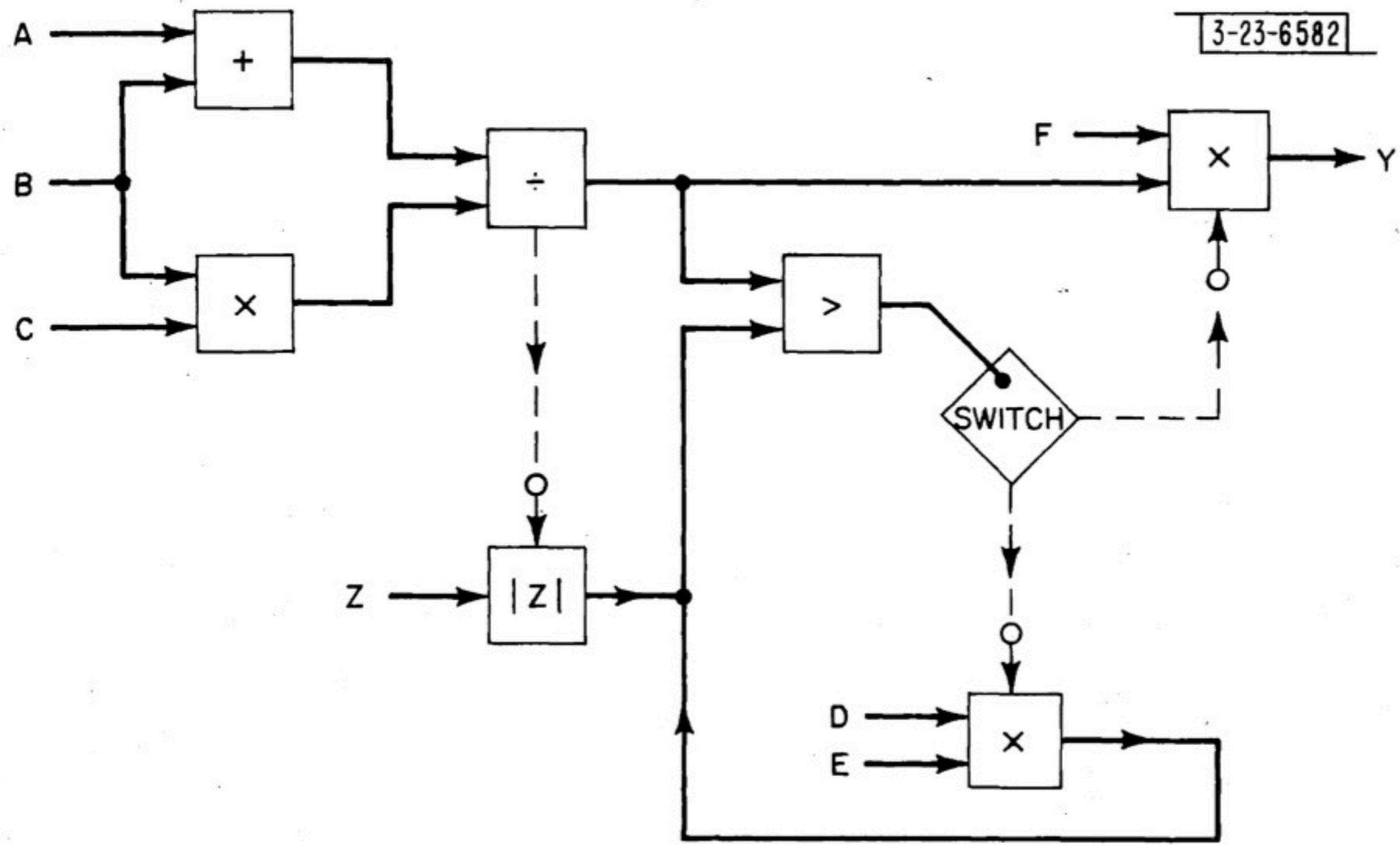
WILLIAM ROBERT SUTHERLAND

An interactive computer graphics system may be used for describing procedures in a two-dimensional programming language as well as for the more usual task of manipulating graphical data. The work described here is concerned with the graphical specification of procedures which then may be applied to problem data obtained from any source. The notation used to form a two-dimensional description of a procedure and the conventions used to control the procedure's execution are described.

An experimental graphical programming system has been created for the TX-2 Computer. With this system one may draw an arbitrary symbol and give it a meaning. The system has a "macro" capability enabling a new symbol to be defined as a combination of operators. A procedure may be executed after initial values are assigned. A number of debugging features are included in the system.

The conventions described make a graphical procedure description a natural way of expressing parallel operations. The standard notions of flow must be substantially modified. Explicit flow control is generally omitted but may be included at the users option. Some aspects of compiling from a graphical program are also discussed.

Thesis Supervisor: Claude E. Shannon



Flow Path Pieces
Figure 4.6

http://en.wikipedia.org/wiki/Visual_programming_language

Lists about 70 systems and leaves out a lot

What has been will be again,
what has been done will be done again;
there is nothing new under the sun.

Ecclesiastes 1:9-14 NIV

Chemistry Toolkit Rosetta

<http://ctr.wikia.com/>

Solve the same cheminformatics problem using different toolkits.

Read an SD file and list the heavy atom counts

Read a SMILES file and list the number of rings

Convert a SMILES string to canonical SMILES

Working with SD tag data

Detect and report SMILES and SDF parsing errors

Report how many SD file records are within
a certain molecular weight range

Convert SMILES file to SD file

Report the similarity between two structures

Find the 10 nearest neighbors in a data set

Depict a compound as an image

Highlight a substructure in the depiction

Align the depiction using a fixed substructure

Unique SMARTS matches against a SMILES string

Calculate TPSA

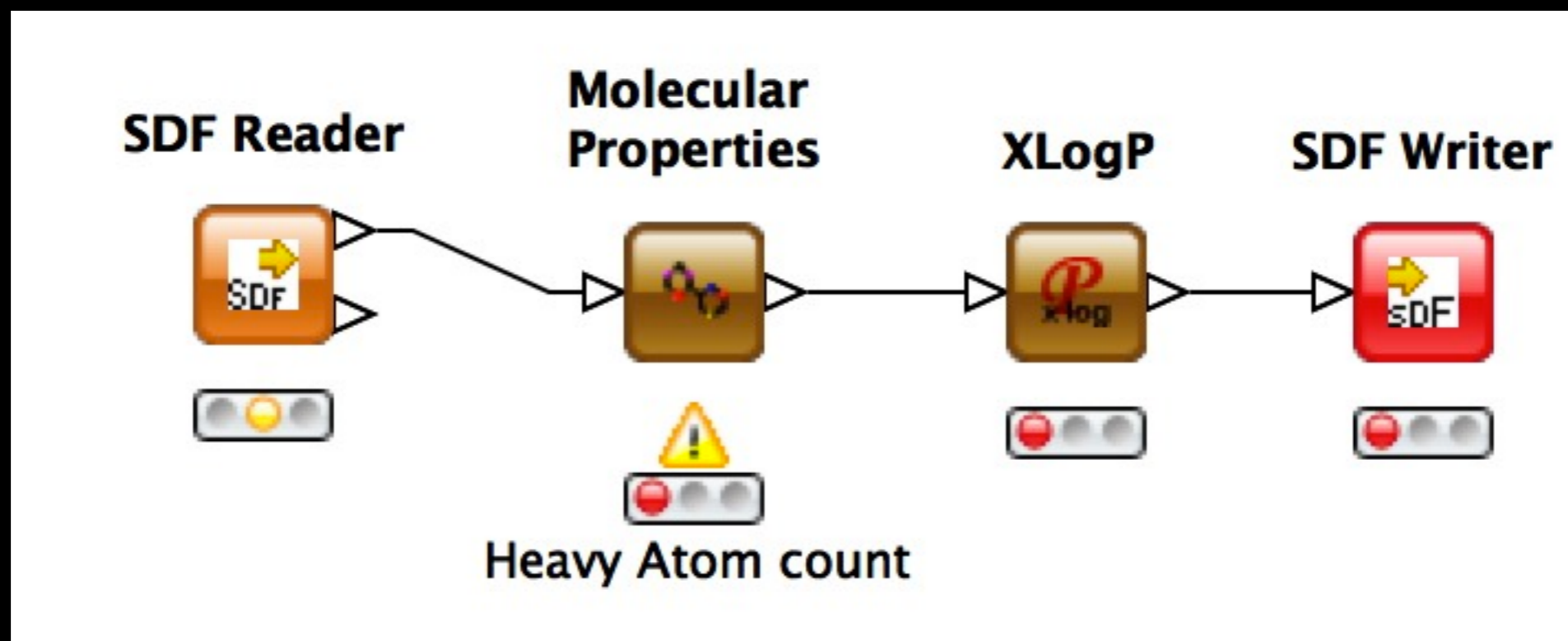
Find the graph diameter

Break rotatable bonds and report the fragments

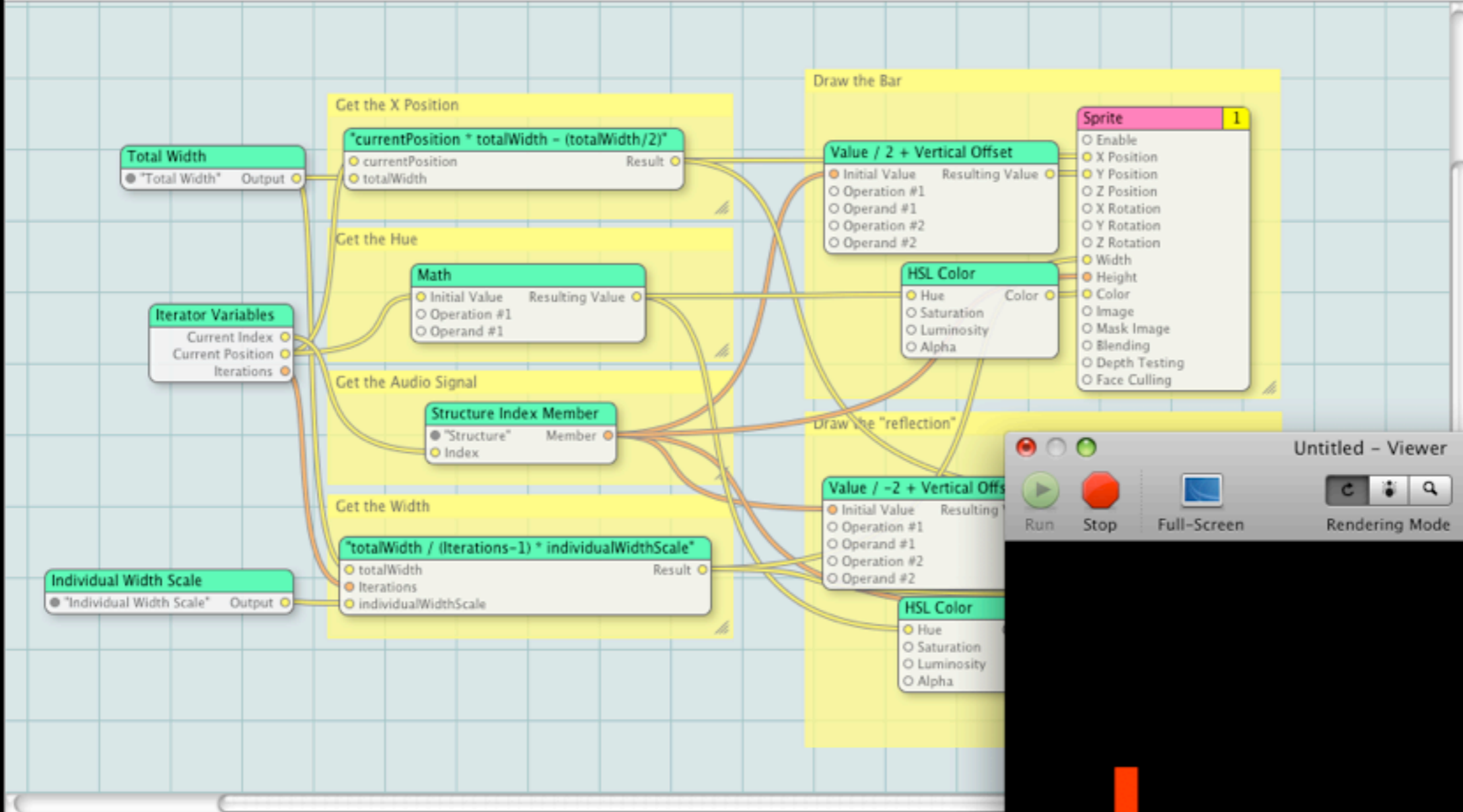
Read an SD file and list
the heavy atom counts

Sometimes decorations can help editorialize about the substance of the graphic. But it's wrong to distort the data measures—the ink locating values of numbers—in order to make an editorial comment or fit a decorative scheme.

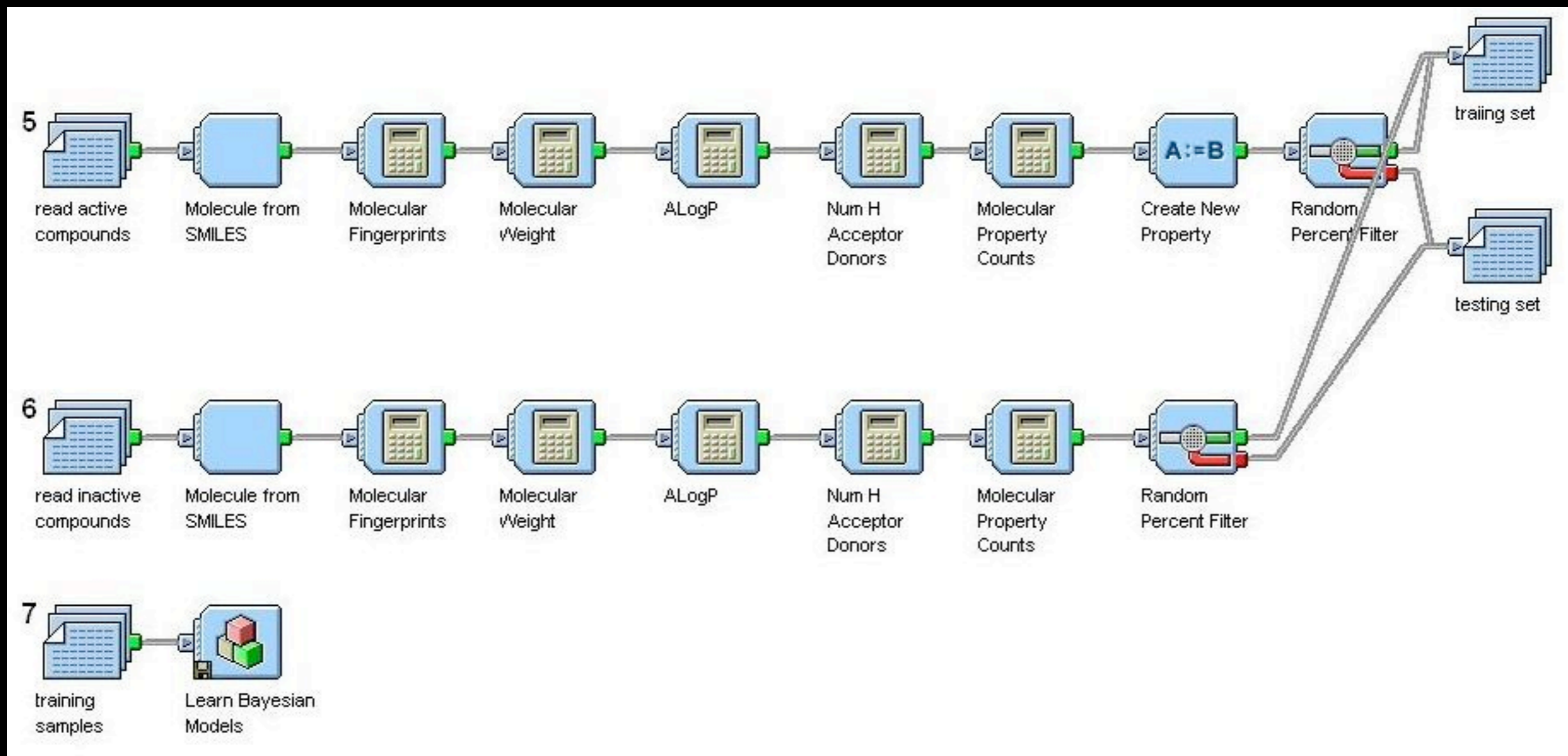
Edward Tufte, “*Visual Display*”



Root Macro Patch > Audio Spectrum > Iterator



$$\text{model} = (0.056 * \text{MW}) + (0.03 * \text{ACDLogD74}) + (0.06 * \text{ACDLogP}) - (0.07 * \text{NumCharged}) + (0.017 * \text{NumHDonors})$$



<http://cheminfo.wikispaces.com/High+Throughput+Predictive+Models+for+PubChem+Bioassays>

Little boxes ~~on the hillside,~~ in the pipeline,
Little boxes made of ticky tacky,
Little boxes ~~on the hillside,~~ in the pipeline,
Little boxes all the same.

There's a green one and a pink one
And a blue one and a yellow one,
And they're all made out of ticky tacky
And they all look just the same.

- Malvina Reynolds

Dependencies

`_orig_smiles` → “C1(=C(C=C...”
`smiles, _oemol` → `register_input`
`ACDLogD74` → `calc_acd`
`model` → `compute_model`

```
def register_input(properties):
```

```
    input_smiles = properties["_orig_smiles"]
```

```
    ... register ...
```

```
    properties["smiles"] = registered_smiles
```

```
    properties["_oemol"] = mol
```

```
def compute_model(properties):
```

```
    MW = properties["MW"]; ...
```

```
    properties["model"] = ((0.056*MW)+ (0.03*ACDLogD74) +  
                           (0.06*ACDLogP) - (0.07*NumCharged)  
                           (0.017*NumHDonors))
```

Dependencies resolved as needed

```
props = PropertyCalculator(smiles="c1ccccc1O")  
print prop["model"]
```

Easy to extend

```
props = PropertyCalculator(extra_rules =  
    {"rat_model": make_rat_prediction})  
print prop["rat_model"]
```

Data processing is molecule-oriented, not batch-oriented

Why I don't like visual programming languages

- hides important information
- GUI display uses a lot more space than text
- dependency management
- does not scale beyond a few dozen nodes
 - (“that node where I open the ‘test.sdf’ file”)
- how do I add new capabilities?
- wide gap between node user and node author

But some people use and like them.

Why?

“Don’t need to program”

Remember, these are “visual programming languages”

“It’s easier”

Counter-proof by Microsoft

VisualBasic

IDE integration

```
>>> help(OEParseSmiles)
```

```
Help on function OEParseSmiles in module openeye.oechem:
```

```
OEParseSmiles(*args)
```

```
OEParseSmiles(OEMolBase mol, char str, bool canon=False, bool strict=False) -> bool
```

```
OEParseSmiles(OEMolBase mol, char str, bool canon=False) -> bool
```

```
OEParseSmiles(OEMolBase mol, char str) -> bool
```

```
OEParseSmiles(OEMolBase mol, string str, bool canon=False, bool strict=False) -> bool
```

```
OEParseSmiles(OEMolBase mol, string str, bool canon=False) -> bool
```

```
OEParseSmiles(OEMolBase mol, string str) -> bool
```

“It’s easy to find things”

Noel O'Boyle's 'cinfony'

```
>>> help(cinfony.obabel.readstring)
```

Help on function readstring in module cinfony.pybel:

```
readstring(format, string)
```

Read in a molecule from a string.

Required parameters:

format - see the `informats` variable for a list of available

input formats

string

Example:

```
>>> input = "C1=CC=CS1"
```

```
>>> mymol = readstring("smi", input)
```

```
>>> len(mymol.atoms)
```

```
5
```

APIs developed for ease of use

```
import pybel
```

```
for mol in pybel.readfile("sdf", "benzodiazepine.sdf.gz"):  
    print mol.OBMol.NumHvyAtoms()
```

```
from openeye.ochem import *
```

```
ifs = oemolistream()
```

```
ifs.open("benzodiazepine.sdf.gz")
```

```
for mol in ifs.GetOEGraphMols():
```

```
    print OECCount(mol, OEIsHeavy())
```

“Pipelines are easy to build and run”

True! Sometimes.

```
results = []  
for mol in readfile("example.sdf"):  
    if matches(mol, "...BOND == 1"):  
        results.append(mol)  
view_in_vida(results)
```

```
Pipeline(readfile("example.sdf"),  
         filter(... BOND == 1),  
         view_in_vida)
```

```
view_in_vida(mol for mol in readfile("example.sdf") if matches(mol, "...BOND == 1"))
```

The screenshot displays a graphical user interface for a pipeline execution tool. It features three stacked panels: 'ToOEB', 'SDFilter', and 'VIDA'. The 'SDFilter' panel is active, showing a filter expression 'PUBCHEM_CACTVS_ROTATABLE_BOND == 1' and an unchecked checkbox for 'Skip matching fields'. Below the panels is a progress bar and a log window. The log window contains the following entries:

Log	Duration
✓ ToOEB completed	0.207 seconds
✓ SDFilter completed	0.180 seconds
✓ VIDA completed	1.256 seconds

“With a GUI I get to see the results”

Also true.

At the Python prompt, how do I
view a molecule? Plot a graph?
Inspect a database?

(VIDA, PyMol, VMD, ... do make this easier.)

Utility functions to call helper apps

```
>>> view(mol)
```

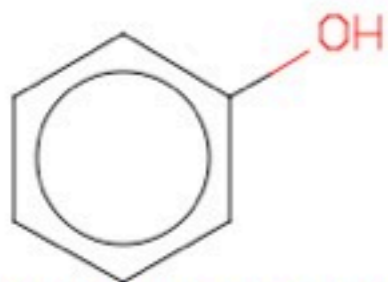
```
>>> view(oematch)
```

```
>>> view(fingerprint)
```

In [2]: `smilin("c1ccccc1O")`

Help:

Out[1]:



Download [smiles](#) [SDF](#) [mol2](#)

In [2]: `mol = Out[1]`

In [3]: `len(mol.atoms)`

Out[3]: 7

In[4]: `ls -l *.sdf`

```
-rw-r--r-- 1 dalke staff 4152 Sep 30 2006 292626083396408575.sdf
-rw-r--r-- 1 dalke staff 3019 Apr 2 2007 a_drug.sdf
-rw-r--r-- 1 dalke staff 92057 Apr 14 2003 compounds.sdf
```

In [5]: `db = load_database("compounds.sdf")`

In [6]: `db`

Out[6]:

Search:

Id	Structure	MW
ABC00001	<chem>c1ccccc1O</chem>	94.05
ABC00002	<chem>O</chem>	18.01

View structure as

Download [tab-delimited](#) [Excel](#) [Spotfire](#)

In [7]: `db.search("100 < MW < 1000 and tanimoto($query) > 0.9",
... sortBy = "tanimoto($query)", query = mol)`

Out[7]:

Id	Structure	MW	tanimoto
XYZ1		334.412	0.9814

Workbook

Workbench

The screenshot displays the Galaxy web interface. The top navigation bar includes 'Galaxy', 'Analyze Data', 'Workflow', 'Data Libraries', 'Help', and 'User'. The left sidebar lists various tools under categories like 'Get Data', 'Text Manipulation', 'Filter and Sort', 'Join, Subtract and Group', 'Extract Features', 'Fetch Sequences', 'Fetch Alignments', 'Get Genomic Scores', 'Operate on Genomic Intervals', 'Statistics', 'Graph/Display Data', 'Regional Variation', 'Multiple regression', 'Evolution', 'Metagenomic analyses', 'EMBOSS', 'NGS TOOLBOX BETA', 'NGS: QC and manipulation', 'NGS: Mapping', 'NGS: SAM Tools', and 'NGS: Peak Calling'. The main workspace shows the 'Branch Lengths' tool configuration. The 'Fasta file' is selected, and the 'Substitution Model' is set to 'F81'. The 'Base Frequencies' are set to 'Nucleotide frequencies collected from the data file'. An 'Execute' button is visible. Below the configuration, a description states: 'This tool takes a single or multiple FASTA alignment file and estimates branch lengths using HYPHY, a maximum likelihood analyses package. For the tree definition, you only need to specify the species build names. For example, you could use the tree ((hg17,panTro1),(mm5,rn3),canFam1), if your FASTA file looks like this:'. A sample FASTA file is shown with headers for hg17, panTro1, mm5, rn3, and canFam1, each with a chromosome and coordinates, followed by a sequence of 'GTGGGAGGT'. The right sidebar shows the 'History' panel with an 'Options' dropdown. It contains an 'Unnamed history' section with a single entry: '1: Get Microbial Data (CDS for Aster yellows witches'-broom phytoplasma AYWB:NC_007718)'. Below this, it says '1 regions, format: bed, database: 13478' and 'Info: Get Microbial Data (CDS for Aster yellows witches'-broom phytoplasma AYWB:NC_007718)'. A 'view in GeneTrack' link is provided. A table with 4 columns (1. Chrom, 2. Start, 3. End, 4) is shown with 4 rows of data: NC_007718 0 1149 rep;AYWB_pII0, NC_007718 1440 1911 AYWB_pII02,GI, NC_007718 1914 2397 AYWB_pII03,GI, and NC_007718 3012 3327 ssb;AYWB_pII0.

Tools

- Get Data
- Send Data
- ENCODE Tools
- Lift-Over
- Text Manipulation
- Convert Formats
- FASTA manipulation
- Filter and Sort
- Join, Subtract and Group
- Extract Features
- Fetch Sequences
- Fetch Alignments
- Get Genomic Scores
- Operate on Genomic Intervals
- Statistics
- Graph/Display Data
- Regional Variation
- Multiple regression
- Evolution
 - Branch Lengths Estimation
 - Neighbor Joining Tree Builder
 - Mutate Codons with SNPs
- Metagenomic analyses
- EMBOSS
- NGS TOOLBOX BETA
- NGS: QC and manipulation
- NGS: Mapping
- NGS: SAM Tools
- NGS: Peak Calling

Branch Lengths

Fasta file:

Tree Definition:

For example: ((hg17,panTro1),(mm5,rn3),canFam1)

Substitution Model:

Base Frequencies:

This tool takes a single or multiple FASTA alignment file and estimates branch lengths using HYPHY, a maximum likelihood analyses package.

For the tree definition, you only need to specify the species build names. For example, you could use the tree ((hg17,panTro1),(mm5,rn3),canFam1), if your FASTA file looks like this:

```
>hg17.chr7(+):26907301-26907310|hg17_0
GTGGGAGGT
>panTro1.chr6(+):28037319-28037328|panTro1_0
GTGGGAGGT
>mm5.chr6(+):52104022-52104031|mm5_0
GTGGGAGGT
>rn3.chr4(+):80734395-80734404|rn3_0
GTGGGAGGT
>canFam1.chr14(+):42826409-42826418|canFam1_0
GTGGGAGGT

>hg17.chr7(+):26907310-26907326|hg17_1
AGTCAGAGTGTCTGAG
>panTro1.chr6(+):28037328-28037344|panTro1_1
AGTCAGAGTGTCTGAG
>mm5.chr6(+):52104031-52104047|mm5_1
AGTCAGAGTGTCTGAG
>rn3.chr4(+):80734404-80734420|rn3_1
AGTCAGAGTATCTGAG
>canFam1.chr14(+):42826418-42826434|canFam1_1
AGTCAGAGTGTCTGAG

>hg17.chr7(+):26907326-26907338|hg17_2
```

History

Unnamed history

1: Get Microbial Data (CDS for Aster yellows witches'-broom phytoplasma AYWB:NC_007718)

1 regions, format: bed, database: 13478

Info: Get Microbial Data (CDS for Aster yellows witches'-broom phytoplasma AYWB:NC_007718)

1. Chrom	2. Start	3. End	4
NC_007718	0	1149	rep;AYWB_pII0
NC_007718	1440	1911	AYWB_pII02,GI
NC_007718	1914	2397	AYWB_pII03,GI
NC_007718	3012	3327	ssb;AYWB_pII0

Galaxy - <http://main.g2.bx.psu.edu/>

Bioclipse

The screenshot displays the Bioclipse software interface. The central window shows the 2D chemical structure of reserpine, a complex alkaloid with a pentacyclic core and a triphosphate group. The structure is rendered with atoms in various colors: carbon (grey), oxygen (red), nitrogen (blue), and phosphorus (green).

On the left, the 'Bioclipse Navigator' panel shows a tree view of 'Sample Data' with subfolders for '2D structures' and '3D Structures'. The '2D structures' folder contains files like '0037.cml', '0037.mdl', 'ATP.mol', 'polycarpol.mol', 'reserpine.mol', and 'thiamin.mol'. The '3D Structures' folder contains files like '2-methylpropanal.cml', 'pentan-1-ol.mol', 'pentanal.cml', 'propan-2-ol.cml', and 'tetracosane.cml'. There are also folders for 'Javascripts', 'PDB', and 'SDF'.

At the bottom left, the 'Properties' panel displays a table of molecular properties:

Property	Value
General	
Has 2D Coords	yes
Has 3D Coords	no
InChI	InChI=1/C10H19N5O13P3/c11-8-5-9(13-2-12-8)15(3-14-5)10-7(17)6(16)4(
InChIKey	VYU5ENGRDRMH-UHFFFAOYAU
Molecular Format	MDL Molfile (2D)
Molecular Formula	C10H19N5O13P3
Molecular Mass	510.2055
SMILES	O=P(O)(O)OP(=O)(O)OP(=O)(O)OCC3OC(N2CNC1C(N)NCNC12)C(O)C3(O)

On the right, the 'Outline' panel lists the atoms and bonds in the structure. The 'Atoms' section lists 10 oxygen atoms (O) with their hybridization states (O.sp3 or O.sp2) and 4 carbon atoms (C) with their hybridization states (C.sp3). The 'Bonds' section lists 14 bonds, including 1 N-C (single), 10 C-N (single, aromatic), and 3 C-C (single, aromatic).

Those are technological solutions.

What about improving the
programming skills of chemists?

<http://dalkescientific.com/training/>

Web Applications for Cheminformatics

Python, Django, OEChem, Javascript

Leipzig, 18-20 May, 2010

Designed for computational chemists who
need to develop web-based tools for others.
All examples based on cheminformatics.

- Django web application framework
- OEChem for cheminformatics
- handling csv, SMILES and SD files
- structure and substructure depictions
- SQL databases and Excel
- descriptor and fingerprint calculations
- Javascript for AJAX and widgets
- plotting, calling binaries, and more.

See <http://dalkescientific.com/training/> or
email training@dalkescientific.com

Andrew Dalke
dalke@dalkescientific.com
+46 (0)73 980 70 09
Göteborg, Sweden



Code Reviews