Python for Chemistry in 21 ~~days~~ *minutes*

Dr. Noel O'Boyle

Dr. John Mitchell and Prof. Peter Murray-Rust

UCC Talk, Sept 2005

Available at http://www-mitchell.ch.cam.ac.uk/noel/

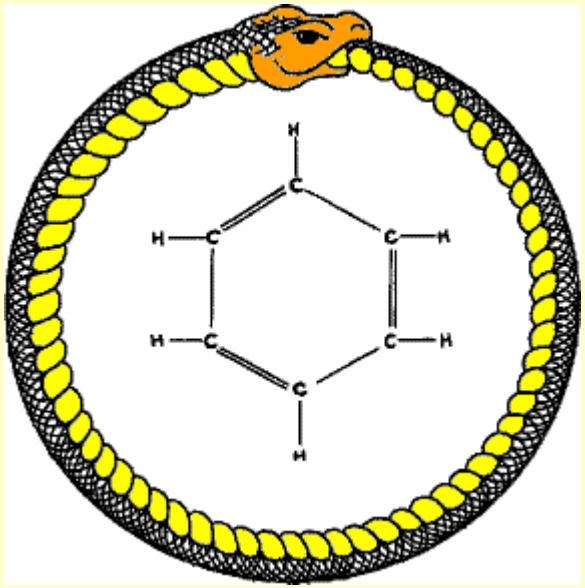
Introduction

- This talk will cover
  - what Python is
  - why you should be interested in it
  - how you can use Python in chemistry

Introduction

- This talk will cover
  - what Python is
  - why you should be interested in it
  - how you can use Python in chemistry
- This talk will **not** cover
  - how to program in Python
- See references at end of talk

Word of warning!!



“My mental eye could now distinguish larger structures, of manifold conformation; long rows, sometimes more closely fitted together; all twining and twisting in snakelike motion. But look! What was that? One of the snakes had seized hold of its own tail, and the form whirled mockingly before my eyes. As if by the flash of lightning I awoke...Let us learn to dream, gentlemen”

Friedrich August Kekulé (1829-1896)

What is Python?

- For a computer scientist...
  - a high-level programming language
  - interpreted (byte-compiled)
  - dynamic typing
  - object-oriented

What is Python?

- For a computer scientist...
  - a high-level programming language
  - interpreted (byte-compiled)
  - dynamic typing
  - object-oriented
- For everyone else...
  - a scripting language (like Perl or Ruby) released by Guido von Rossum in 1991
  - easy to learn
  - easy to read (!)

What is Python?

- For a computer scientist...
  - a high-level programming language
  - interpreted (byte-compiled)
  - dynamic typing
  - object-oriented

- For everyone else...
  - a scripting language (like Perl or Ruby) released by Guido von Rossum in 1991
  - easy to learn
  - easy to read (!)
  - named after Cambridge comedians

The Great Debate

Sir Lancelot:
We were in the nick of time. You were in great Perl.
Sir Galahad:
I don't think I was.
Sir Lancelot:
You were, Sir Galahad. You were in terrible Perl.
Sir Galahad:
Look, let me go back in there and face the Perl.
Sir Lancelot:
No, it's too perilous.

(adapted from *Monty Python and the Holy Grail*)

Why you should be interested (1)

- Python has been adopted by the cheminformatics community
- For example, AstraZeneca has moved some of its codebase from 'the other scripting language' to Python

Job Description – Research Software Developer/Informatician
[section deleted]
Required Skills:

At least one object oriented programming language, e.g., Python, C++, Java.
Web-based application development (design/construction/maintenance)
UNIX, UNIX scripting & Linux OS

Position in AstraZeneca R&D, 02-09-05

Why you should be interested (2)

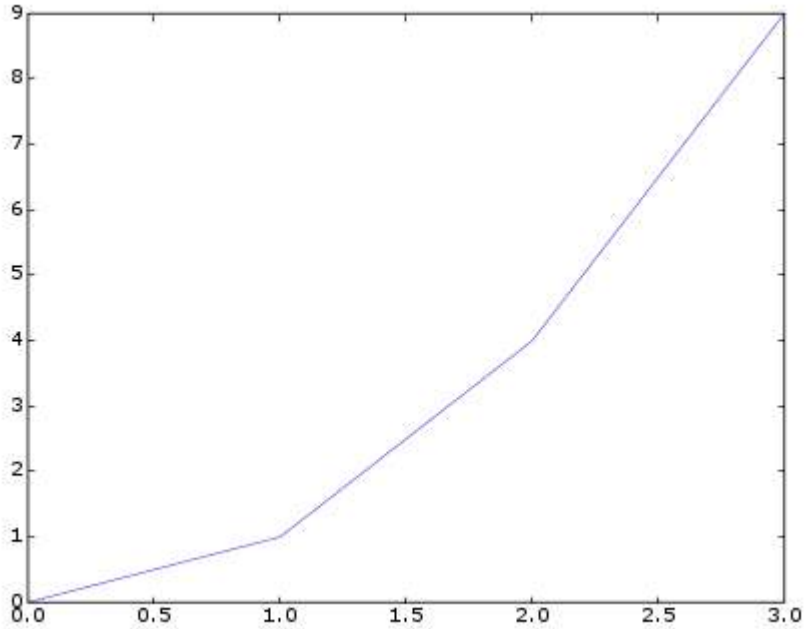
- Scientific computing: scipy/pylab (like Matlab)
- Molecular dynamics: MMTK
- Statistics: scipy, rpy (R), pychem
- 3D-visualisation: VTK (mayavi)
- 2D-visualisation: scipy, pylab, rpy
- coming soon, a wrapper around OpenBabel
- cheminformatics: OEChem, frowns, PyDaylight, pychem
- bioinformatics: BioPython
- structural biology: PyMOL
- computational chemistry: GaussSum
- you can still use Java libraries...like the CDK

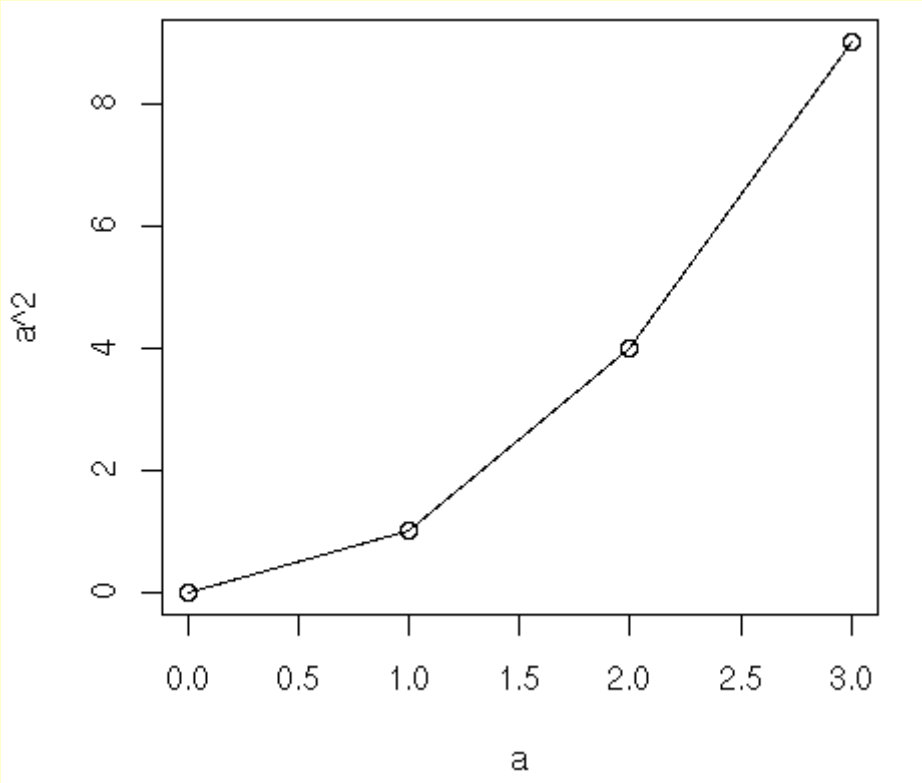
scipy/
pylab

```
>>> from scipy import *
>>> from pylab import *
>>> a = arange(0,4)
>>> a
[0,1,2,3]
>>> mean(a)
1.5
>>> a**2
[0,1,4,9]
>>> plot(a,a**2)
>>> show()
```

```
> a <- seq(0,3)
> a
[1] 0 1 2 3
> mean(a)
[1] 1.5
> a**2
[1] 0 1 4 9
> plot(a,a**2)
> lines(a,a**2)
```

R





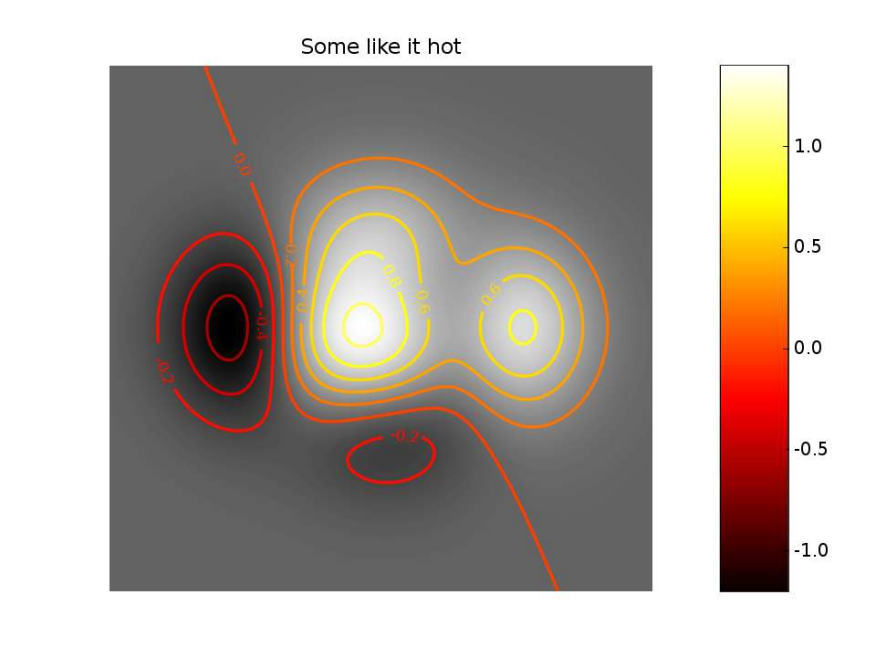
Scipy

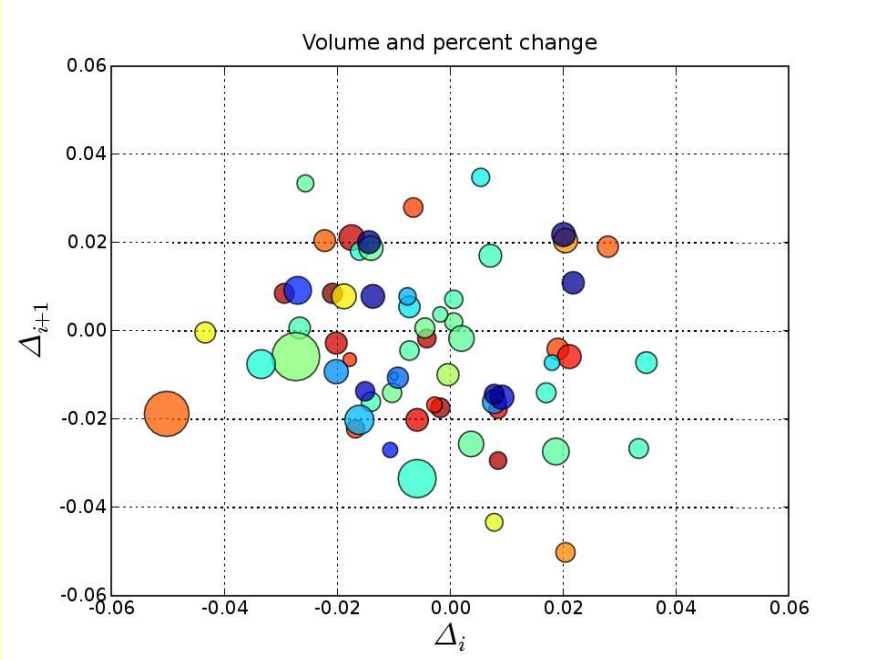
- cluster: information theory functions (currently, vq and kmeans)
- weave: compilation of numeric expressions to C++ for fast execution
- cow: parallel programming via a Cluster Of Workstations
- fftpack: fast Fourier transform module based on fftpack and fftw when available
- ga: genetic algorithms
- io: reading and writing numeric arrays, MATLAB .mat, and Matrix Market .mtx files
- integrate: numeric integration for bounded and unbounded ranges. ODE solvers.
- interpolate: interpolation of values from a sample data set.
- optimize: constrained and unconstrained optimization methods and root-finding algorithms
- signal: signal processing (1-D and 2-D filtering, filter design, LTI systems, etc.)
- special: special function types (bessel, gamma, airy, etc.)
- stats: statistical functions (stdev, var, mean, etc.)
- linalg: linear algebra and BLAS routines based on the ATLAS implementation of LAPACK
- sparse: Some sparse matrix support. LU factorization and solving sparse linear systems

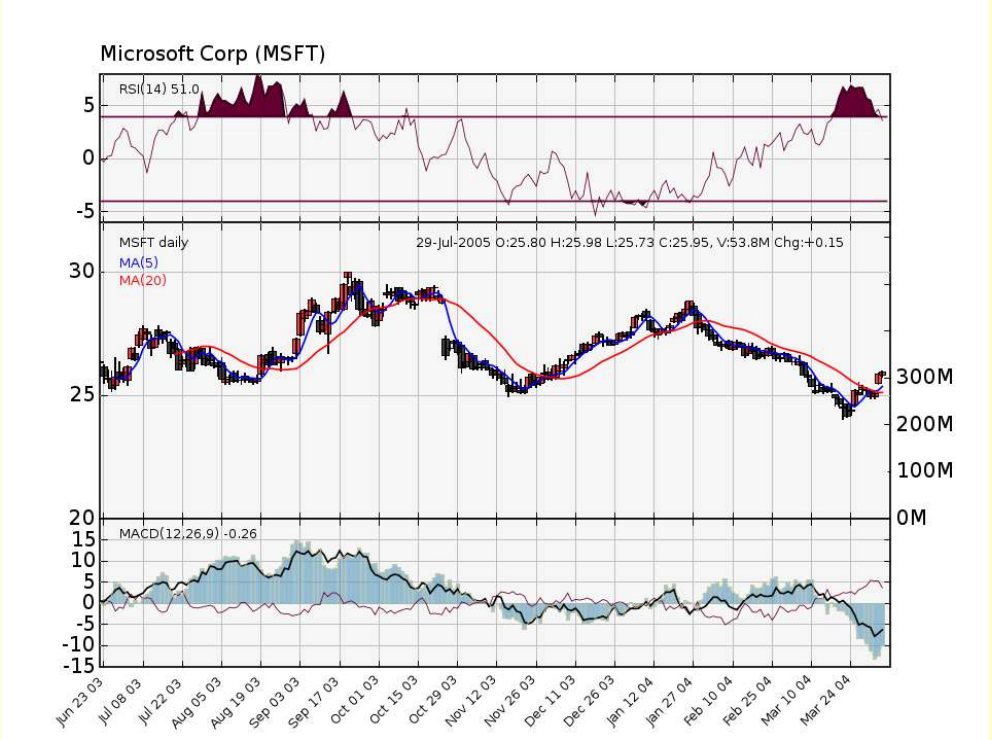
Scipy statistical functions

- descriptive statistics: variance, standard deviation, standard error, mean, mode, median
- correlation: Pearson r, Spearman r, Kendall tau
- statistical tests: chi-squared, t-tests, binomial, Wilcoxon, Kruskal, Kolmogorov-Smirnov, Anderson, etc.
- linear regression
- analysis of variance (ANOVA)
- (and more)

pylab


Some like it hot
1.0
0.5
0.0
-0.5
-1.0


Volume and percent change
0.06
0.04
0.02
0.00
-0.02
-0.04
-0.06
Δ_{i+1}
Δ_i


Microsoft Corp (MSFT)
RSI(14) 51.0
MSFT daily
29-Jul-2005 O:25.80 H:25.98 L:25.73 C:25.95, V:53.8M Chg:+0.15
MA(5)
MA(20)
MACD(12,26,9) -0.26
300M
200M
100M
0M
Jun 23 03
Jul 08 03
Jul 22 03
Aug 05 03
Aug 19 03
Sep 03 03
Sep 17 03
Oct 01 03
Oct 15 03
Oct 29 03
Nov 12 03
Nov 26 03
Dec 11 03
Dec 26 03
Jan 12 04
Jan 27 04
Feb 10 04
Feb 25 04
Mar 10 04
Mar 24 04

pychem: Using scipy for chemoinformatics

- Many multivariate analysis techniques are based on matrix algebra
- scipy has wrappers around well-known C and Fortran numerical libraries (ATLAS, LAPACK)
- pychem can do:
  - principal component analysis, partial least squares regression, Fisher's discriminant analysis
  - clustering: k-means, hierarchical (used Open Source clustering library)
  - feature selection: genetic algorithms with PLS
  - additional methods can be added

Python and R

- Advantages of R:
  - a large number of statistical libraries are available
- Disadvantages of R:
  - difficult to write algorithms
  - slow (most R libraries are written in C)
  - chokes on large datasets (use scan instead of read.table)

Reading in data

Method	300K	600K	1.6M
Python	6.8	13.9	41
R (read.table)	42	105	
R (scan)	9	20	56

Principal component analysis

Method	300K	600K	1.6M
Python	2.2	3.6	42
R (read.table)	5	10	
R (scan)	3	5	29

http://www.redbrick.dcu.ie/~noel/RversusPython.html

Python and R

- rpy module allows Python programs to interface with R
- have the best of both worlds
  - access to the statistical functions of R
  - access to the numerous modules available for Python
  - can program in Python, instead of in R!!

Python

```
>>> from rpy import r
>>> x = [5.05, 6.75, 3.21, 2.66]
>>> y = [1.65, 26.5, -5.93, 7.96]
>>> print r.lsfit(x,y)['coefficients']
{'X': 5.3935773611970212,
 'Intercept': -16.281127993087839}
```

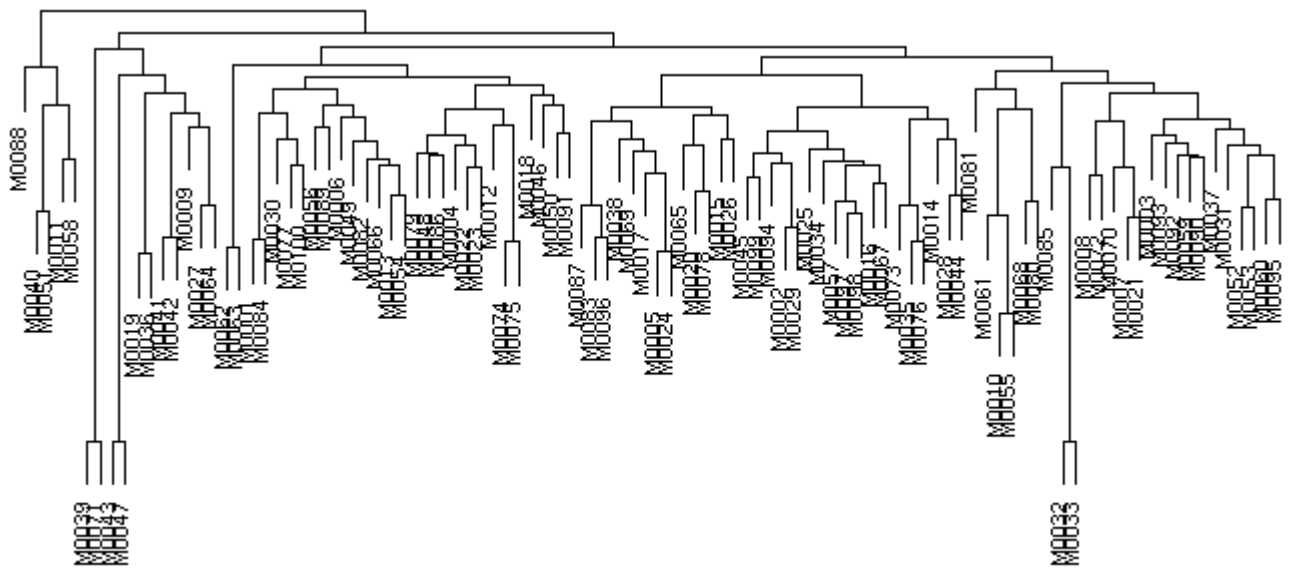
R

```
> x <- c(5.05, 6.75, 3.21, 2.66)
> y <- c(1.65, 26.5, -5.93, 7.96)
> lsfit(x, y)$coefficients
 Intercept          X
-16.281128   5.393577
```

Python and R

Problem: Analyse a hierarchical clustering

Solution: Use R to cluster, and Python to analyse the merge object of the cluster



```
> hc$merge
      [,1] [,2]
 [1,]  -32  -33
 [2,]  -39  -71
 [3,]  -43  -47
 [4,]  -10  -55
 [5,]  -19  -36
 [6,]   -5  -24
 [7,]  -62  -63
 [8,]  -74  -75
 [9,]  -35  -76
[10,]   -1  -84
[11,]  -41  -42
[12,]  -83  -96
[13,]   -2  -29
[14,]   -7  -21
[15,]  -61    4
```

Problem 1

Graphically show the distribution of molecular weights of molecules in an SD file. The molecular weight is stored in a field of the SD file.

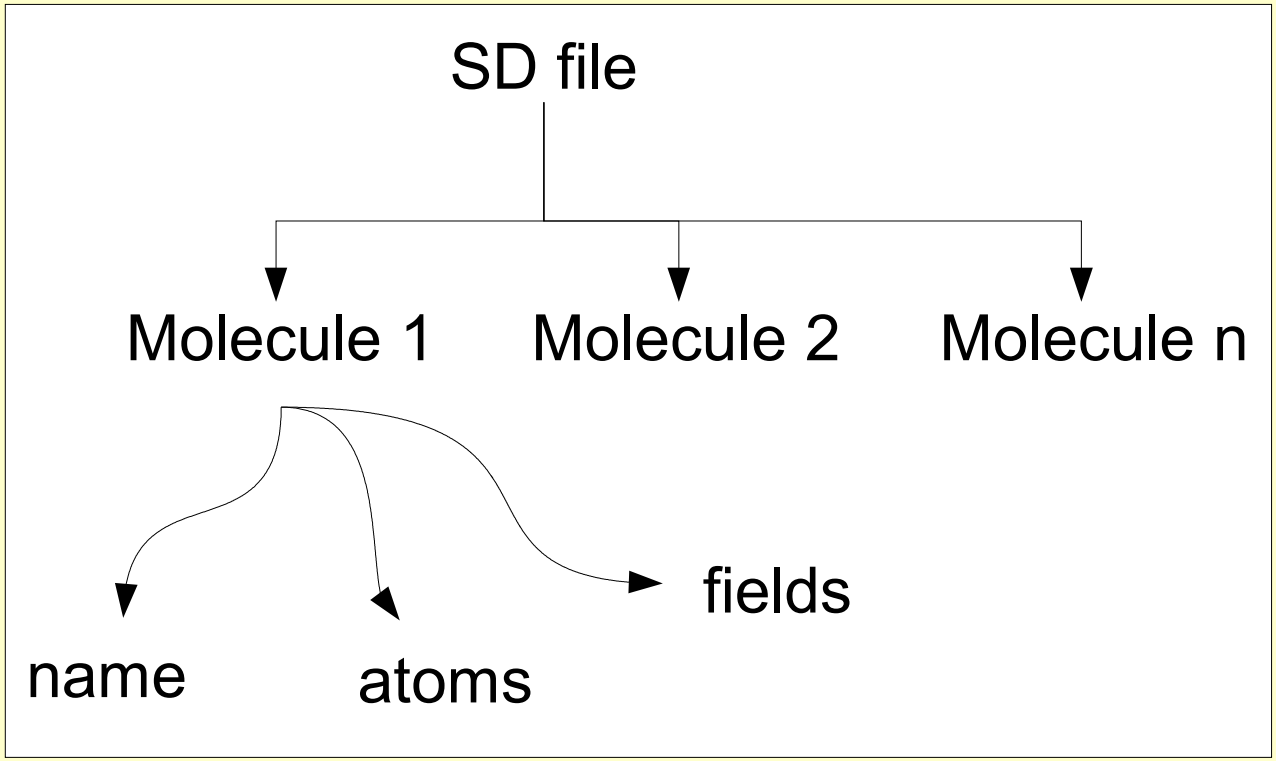
```
1,2-Diaminoethane
  MOE2004           3D

  6 3 0  0  0  0  0  0  0  0999 V2000
   -0.6900   -0.6620    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
    0.5850   -1.9590    0.8240 H   0  0  0  0  0  0  0  0  0  0  0  0
    0.5350    1.5040    0.0000 N   0  0  0  0  0  0  0  0  0  0  0  0
    0.6060    2.0500    0.8460 H   0  0  0  0  0  0  0  0  0  0  0  0
    1.3040    0.8520   -0.0460 H   0  0  0  0  0  0  0  0  0  0  0  0
  1  2  1  0  0  0  0
  1  3  1  0  0  0  0
  1  4  1  0  0  0  0
M  END
>  <chem.name>
1,2-Diaminoethane

>  <molecular.weight>
60.0995

$$$$
```

Object-oriented approach

object

object

attributes



The code for creating these objects is stored in sdparser.py, which can be imported and used by any scripts that need to parse SD files.

Solution

```
from sdparser import SDFile
from rpy import *

inputfile = "mddr_complete.sd"

allmolweights = []

for molecule in SDFile(inputfile):
    molweight = molecule.fields['molecular.weight']
    allmolweights.append(float(molweight))

r.png(file="molwt_r.png")
r.hist(allmolweights,xlab="Mol. weights",main="MDDR", col="red")
r.dev_off()
```

Solution

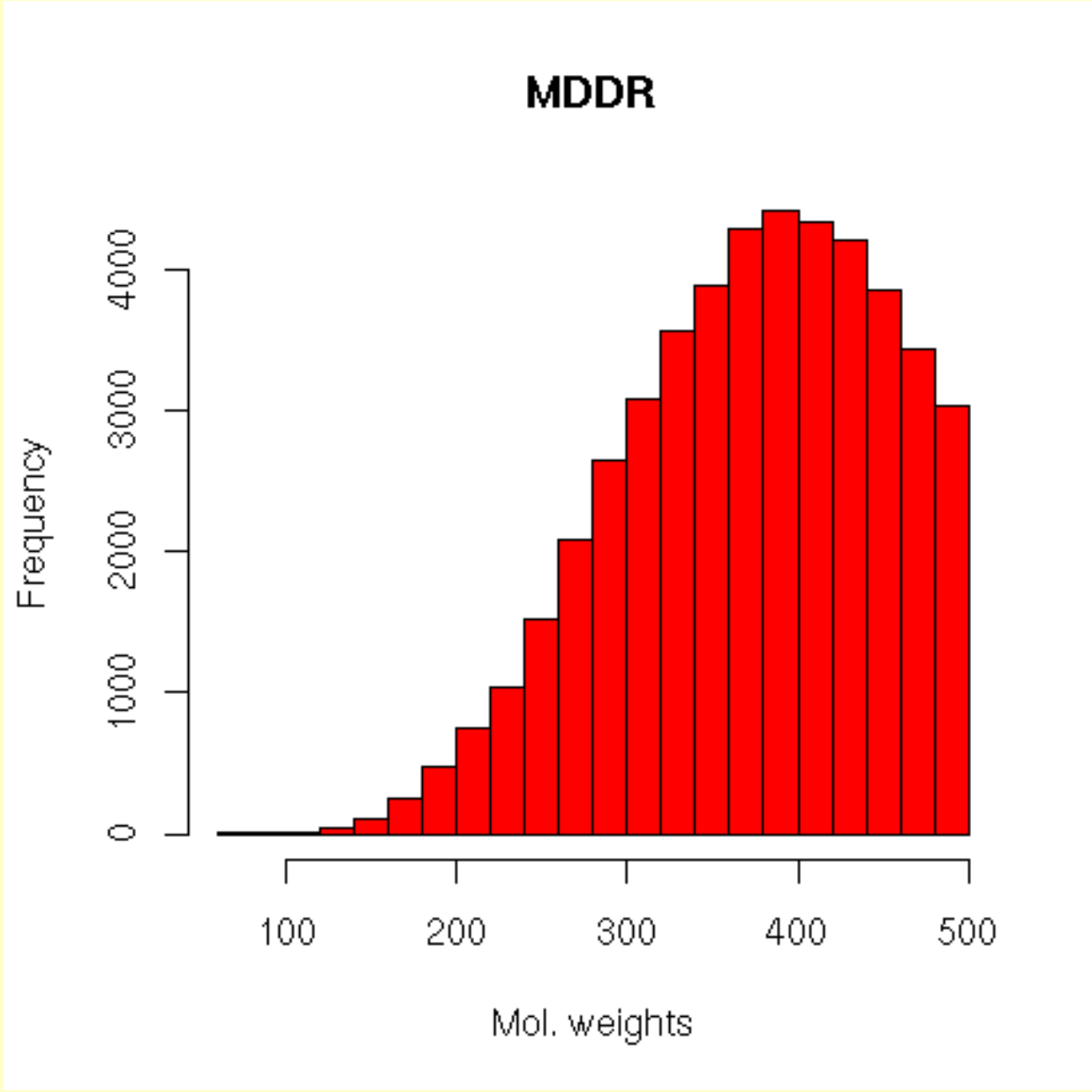
```
from sdparser import SDFile
from rpy import *

inputfile = "mddr_complete.sd"

allmolweights = []

for molecule in SDFile(inputfile):
    molweight = molecule.fields['molecular.weight']
    allmolweights.append(float(molweight))

r.png(file="molwt_r.png")
r.hist(allmolweights,xlab="Mol. weights",main="MDDR", col="red")
r.dev_off()
```

Solution

```
from sdparser import SDFile
from rpy import *

inputfile = "mddr_complete.sd"

allmolweights = []

for molecule in SDFile(inputfile):
    molweight = molecule.fields['molecular.weight']
    allmolweights.append(float(molweight))

r.png(file="molwt_r.png")
r.hist(allmolweights,xlab="Mol. weights",main="MDDR", col="red")
r.dev_off()
```

Solution

```
from sdparser import SDFile
from rpy import *

inputfile = "mddr_complete.sd"

allmolweights = []

for molecule in SDFile(inputfile):
    molweight = molecule.fields['molecular.weight']
    allmolweights.append(float(molweight))

r.png(file="molwt_r.png")
r.hist(allmolweights,xlab="Mol. weights",main="MDDR", col="red")
r.dev_off()
```


MDDR
Frequency
0
1000
2000
3000
4000
100
200
300
400
500
Mol. weights

Problem 2

Every molecule in an SD file is missing the name. To be compatible with proprietary program X, we need to set the name equal to the value of the field “chem.name”.

```
(MISSING NAME!)
  MOE2004           3D

  6 3 0  0  0  0  0  0  0  0999 V2000
   -0.6900   -0.6620    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
    0.5850   -1.9590    0.8240 H   0  0  0  0  0  0  0  0  0  0  0  0
    0.5350    1.5040    0.0000 N   0  0  0  0  0  0  0  0  0  0  0  0
    0.6060    2.0500    0.8460 H   0  0  0  0  0  0  0  0  0  0  0  0
    1.3040    0.8520   -0.0460 H   0  0  0  0  0  0  0  0  0  0  0  0
  1  2  1  0  0  0  0
  1  3  1  0  0  0  0
  1  4  1  0  0  0  0
M  END
>  <chem.name>
1,2-Diaminoethane

>  <molecular.weight>
60.0995

$$$$
```

Solution

```
from sdparser import SDFile

inputfile = "mddr_complete.sd"
outputfile = "mddr_withnames.sd"

for molecule in SDFile(inputfile):
    molecule.name = molecule.fields['chemical.name']
    outputfile.write(molecule)

inputfile.close()
outputfile.close()

print "We are the knights who say....SD!!!"
```

Solution

```
from sdparser import SDFile

inputfile = "mddr_complete.sd"
outputfile = "mddr_withnames.sd"

for molecule in SDFile(inputfile):
    molecule.name = molecule.fields['chemical.name']
    outputfile.write(molecule)

inputfile.close()
outputfile.close()

print "We are the knights who say....SD!!!"
```

Python and Java

- It's easy to use Java libraries from Python
  - using either Jython or JPype
  - see http://www.redbrick.dcu.ie/~noel/CDKJython.html

Example: using the CDK to calculate the number of rings in a molecule (given a string variable containing CML)

```
from jpype import *
startJVM("jdk1.5.0_03/jre/lib/i386/server/libjvm.so")
cdk = JPackage("org").openscience.cdk
SSSRFinder = cdk.ringsearch.SSSRFinder
CMLReader = cdk.io.CMLReader

def getNumRings(molecule):
    # Convert to a CDK molecule
    reader = CMLReader(java.io.StringReader(molXmlValue))
    chemFile = reader.read(cdk.ChemFile())
    cdkMol = chemFile.getChemSequence(0).getChemModel(0).getSetOfMolecules().getMolecule(0)
    # Calculate the number of rings
    sssrFinder = SSSRFinder(cdkMol)
    sssr = sssrFinder.findSSSR().size()
    return sssr
```

3D visualisation

- VTK (Visualisation Toolkit) from Kitware
  - open source, freely available
  - scalar, tensor, vector and volumetric methods
  - advanced modeling techniques such as implicit modelling, polygon reduction, mesh smoothing, cutting, contouring, and Delaunay triangulation
- MayaVi
  - easy to use GUI interface to VTK, written in Python
  - can create input files and visualise them using Python scripts

Demo

Python Resources

- http://www.python.org
- Guido's Tutorial
  - http://www.python.org/doc/current/tut/tut.html
- O'Reilly's “Learning Python” or Visual Quickstart Guide to Python
  - Make sure it's Python 2.3 or 2.4 though
- For Windows, consider the Enthought edition
  - http://www.enthought.com/


...AND SO, IN CONCLUSION, OUR PROPOSED METHOD INCREASES PERFORMANCE BY OVER 0.0001%.
I WANT TO THANK YOU FOR LISTENING TO MY TALK. YOU'VE BEEN A GREAT AUDIENCE...
ANY QUESTIONS?
YES, CAN I CHANGE YOUR SHEETS, NOW?
JORGE CHAM © 2004
www.phdcomics.com

THANKS!