



The RDKit: open source cheminformatics now for Knime too!

Gregory Landrum

NIBR IT

Novartis Institutes for BioMedical Research, Basel

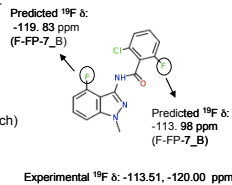
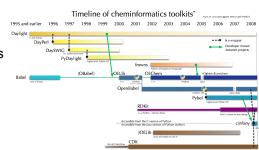
2011 Knime Users Group Meeting

Zurich, 3 March 2011

The RDKit



- Input/Output: SMILES/SMARTS, SDF, TDT, SLN¹, Corina mol2¹
- "Cheminformatics":
 - Substructure searching
 - Canonical SMILES
 - Chirality support (i.e. R/S or E/Z labeling)
 - Chemical transformations (e.g. remove matching substructures)
 - Chemical reactions
 - Molecular serialization (e.g. mol <-> text)
- 2D depiction, including constrained depiction
- 2D->3D conversion/conformational analysis via distance geometry
- UFF implementation for cleaning up structures
- Fingerprinting:
 - Daylight-like, atom pairs, topological torsions, Morgan algorithm, "MACCS keys", etc.
 - Similarity/diversity picking (including fuzzy similarity)
- 2D pharmacophores
- Gasteiger-Marsili charges
- Hierarchical subgraph/fragment analysis
- RECAP and BRICS implementations
- Feature maps
- Shape-based similarity
- Molecule-molecule alignment
- Shape-based alignment (subshape alignment)
- Integration with PyMOL for 3D visualization
- Database integration
- Molecular descriptor library:
 - Topological (κ3, Balaban J, etc.)
 - Electrotopological state (Estate)
 - clqP, MR (Wildman and Crippen approach)
 - "MOE like" VSA descriptors
 - Feature-map vectors
- Machine Learning:
 - Clustering (hierarchical)
 - Information theory (Shannon entropy, information gain, etc.)
 - Decision trees, *naïve Bayes*¹, *KNN*¹
 - Bagging, random forests
 - Infrastructure (data splitting, shuffling, enrichment plots, serializable models, etc.)



```

from rdkit import Chem
from rdkit.Chem import AllChem
import sys, cPickle

refFilename = sys.argv[1]
probeFilename = sys.argv[2]

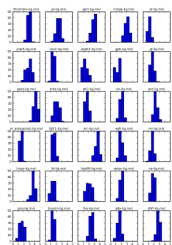
suppl = Chem.SDMolSupplier(refFilename)
refs=[]
for mol in suppl:
    if mol is None: continue
    refs[mol.GetProp('_Name')]=mol

probes = cPickle.load(file(probeFilename, 'rb'))

outF = file(probeFilename.split('.')[1].split('.')[0]+'.'+'.rms.pkl', 'wb+')
for nm, ref in refs.iteritems():
    probe = probes.get(nm, None)
    if probe is None: continue
    query = Chem.RemoveHs(probe)
    matches = ref.GetSubstructMatches(query)
    maps = []
    for match in matches:
        t=[]
        for j, idx in enumerate(match):
            t.append((j, idx))
        maps.append(t)
    molRMS=[]
    for conf in probe.GetConformers():
        bestRMS = AllChem.GetBestRMS(ref, probe, -1, conf.GetId(), maps)
        molRMS.append((bestRMS, conf.GetId()))
    molRMS.sort()
    cPickle.dump((nm, molRMS), outF)

import sys, cPickle
import pylab

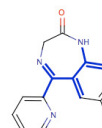
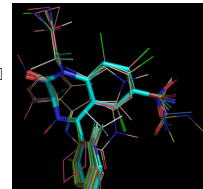
inF = file(sys.argv[1], 'rb')
data=[]
while 1:
    try:
        row = cPickle.load(inF)
    except:
        break
    if row[0].find('self')<=0:
        data.append(row)
baseN=sys.argv[1].split('.')[1].split('.')[0]
nRows=6
perRow=len(data)/nRows + 1
pylab.figure(1, figsize=(10, 10), dpi=100)
pylab.gcf().text(0.5, 0.95, baseN)
horizontalalignment='center', fontsize=16)
pylab.subplots_adjust(hspace=0.3)
for i, (nm, molRMS) in enumerate(data):
    pylab.subplot(nRows, perRow, i+1)
    pylab.hist([x for k, y in molRMS],
               bins=[0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5])
    pylab.title(nm, fontsize=8)
    pylab.axis([0, 5, 0, 50])
    pylab.xticks(fontsize=8, visible=(i/perRow==nRows-1))
    pylab.yticks(fontsize=8, visible=(i/perRow==0))
    pylab.savefig(baseN+'.png')
    pylab.show()
    
```



```

>>> from rdkit import Chem
>>> from rdkit.Chem import AllChem
>>> supplier = Chem.SDMolSupplier('set1.sdf')
>>> patt = Chem.MolFromSmarts('cc(N)c(C=N)c')
>>> ms = [x for x in suppl if x and x.HasSubstructMatch(patt)]
>>> refMol = ms.pop(0)
>>> refMatch = refMol.GetSubstructMatch(patt)
>>> mMatch=ms[0].GetSubstructMatch(patt)
>>> map=zip(mMatch, refMatch)
>>> AllChem.AlignMol(ms[0], refMol, atomMap=map)
0.1027353353855964
>>> for m in ms:
...     mMatch=m.GetSubstructMatch(patt)
...     map=zip(mMatch, refMatch)
...     AllChem.AlignMol(m, refMol, atomMap=map)
...
>>> from Chem import PyMol
>>> v = PyMol.MolViewer()
>>> v.ShowMol(refMol, name='reference')
>>> for m in ms: v.ShowMol(m, name=m.GetProp('_Name'), showOnly=False)
...

>>> from rdkit import Chem
>>> from rdkit.Chem import AllChem
>>> supplier = Chem.SDMolSupplier('set1.sdf')
>>> patt = Chem.MolFromSmarts('cc(N)c(C=N)c')
>>> ms = [x for x in suppl if x and x.HasSubstructMatch(patt)]
>>> refMol = ms.pop(0)
>>> refMatch = refMol.GetSubstructMatch(patt)
>>> mMatch=ms[0].GetSubstructMatch(patt)
>>> map=zip(mMatch, refMatch)
>>> AllChem.AlignMol(ms[0], refMol, atomMap=map)
0.1027353353855964
>>> for m in ms:
...     mMatch=m.GetSubstructMatch(patt)
...     map=zip(mMatch, refMatch)
...     AllChem.AlignMol(m, refMol, atomMap=map)
...
>>> from Chem import PyMol
>>> v = PyMol.MolViewer()
>>> v.ShowMol(refMol, name='reference')
>>> for m in ms: v.ShowMol(m, name=m.GetProp('_Name'), showOnly=False)
...
    
```



```

from rdkit import Chem
from rdkit.Chem import AllChem

# read the molecules once
supplier = Chem.SDMolSupplier('catalog.sdf')
mols = [x for x in supplier if x]

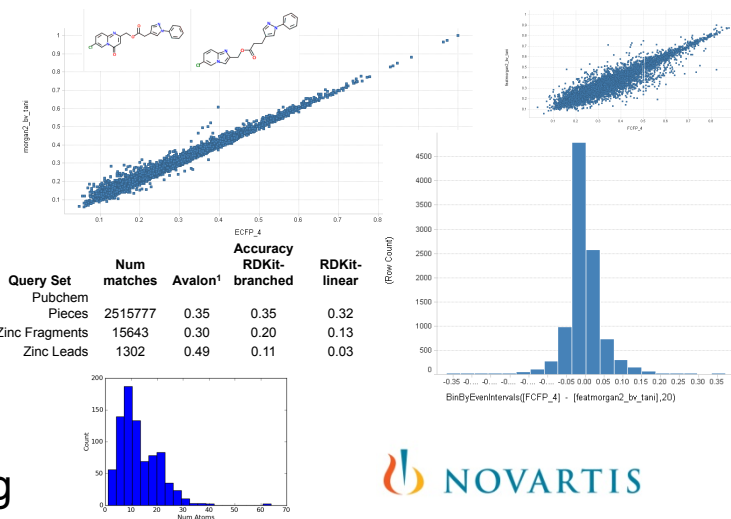
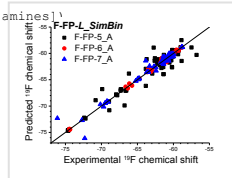
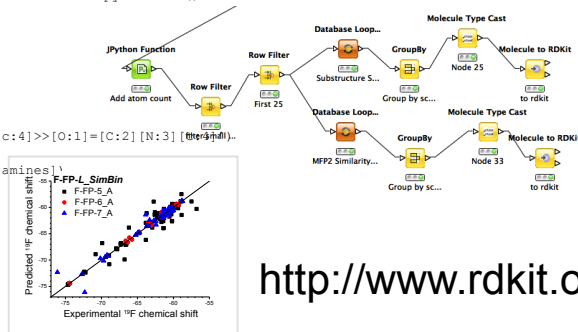
# find primary aromatic amines:
pattern = Chem.MolFromSmarts('[NH2]c')
amines = [x for x in mols if x.HasSubstructMatch(pattern)]

# find carboxylic acids:
pattern = Chem.MolFromSmarts('[O;H;-]C(=O)')
acids = [x for x in mols if x.HasSubstructMatch(pattern)]

# construct a reaction object:
rxn = AllChem.ReactionFromSmarts('[O:1]=[C:2]-[O].[N:3;H2][c:4]>>[O:1]=[C:2][N:3][c:4]')

# returns a generator that yields all the products:
products = AllChem.EnumerateLibraryFromReaction(rxn, [acids, amines])

# write the unique products to an sdf file:
w = Chem.SDWriter('products.sdf')
uniq=set()
for product in products:
    smi = Chem.MolToSmiles(product, True)
    if smi in uniq: continue
    AllChem.Compute2DCoords(product)
    w.write(product)
    uniq.add(smi)
    
```



<http://www.rdkit.org>



Overview

- RDKit: what is it?
- RDKit + PostgreSQL
- RDKit + Knime

Acknowledgements

- Novartis:
 - Tom Digby (Legal)
 - John Davies (CPC/LFP)
 - Richard Lewis (GDC/CADD)
 - Andy Palmer (NIBR IT)
 - Nik Stiefl (GDC/CADD)
 - Peter Gedeck (GDC/CADD)
- Rational Discovery:
 - Santosh Putta (currently at Nodality)
 - Julie Penzotti
- RDKit open-source community
- postgresql cartridge:
 - Michael Stonebraker
 - Oleg Bartunov
 - Teodor Sigaev
 - Pavel Velikhov
- knime.com
 - Michael Berthold
 - Thorsten Meinl
 - Bernd Wiswedel

Overview

- **RDKit: what is it?**
- RDKit + PostgreSQL
- RDKit + Knime

Cheminformatics?

- Definition 1:

“Chem[o]informatics is the mixing of ... information resources to transform data into information and information into knowledge for the intended purpose of making better decisions faster in the area of drug lead identification and optimization.” F. K. Brown <http://dx.doi.org/10.1016%2FS0065-7743%2808%2961100-8>

- Definition 2:

“Cheminformatics enables the combination of molecular information with other types of data to answer questions more quickly, accurately, or completely. In the best case, it allows new questions to be asked and answered ”

- Definition 3:

“Cheminformatics helps to provide answers to the questions chemists ask about (and using) molecules.”



RDKit: What is it?

- Python (2.x) and C++ toolkit¹ for cheminformatics
 - Core data structures and algorithms in C++
 - Heavy use of Boost libraries
 - Python wrapper generated using Boost.Python
- Functionality:
 - 2D and 3D molecular operations
 - Descriptor generation for machine learning
 - Molecular database cartridge
 - Supports Mac/Windows/Linux
- History:
 - 2000-2006: Developed and used at Rational Discovery for building predictive models for ADME, Tox, biological activity
 - June 2006: Open-source (BSD license) release of software, Rational Discovery shuts down
 - to present: Open-source development continues, use within Novartis, contributions from Novartis back to open-source version

¹ Evolving support for Java

RDKit: Where is it?

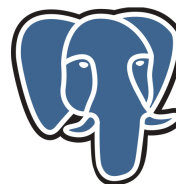
- Web page: <http://www.rdkit.org>
- Sourceforge: svn repository, bug tracker, mailing lists, downloads
 - <http://sourceforge.net/projects/rdkit>
- Google code: wiki, downloads
 - <http://code.google.com/p/rdkit/>
- Releases: quarterly (more or less)
- Licensing: new BSD
- Documentation:
 - “Getting Started in Python” document
 - in-code docs extracted by either doxygen (C++) or epydoc (python)
- Getting help:
 - Check the wiki and “Getting Started” document
 - The rdkit-discuss mailing list

RDKit: Who is using it?

- Hard to say with any certainty
- ~300 downloads of each new version
- Active contributors to the mailing list from:
 - Small pharma/biotech
 - Software/Services
 - Academia

Overview

- RDKit: what is it?
- **RDKit + PostgreSQL**
- RDKit + Knime



PostgreSQL

+



The database cartridge

- Integration of RDKit fingerprinting and substructure search functionality with PostgreSQL
- Similarity metrics (Tanimoto and Dice) integrated with PostgreSQL indexing system to allow fast searches (~1 million compounds/sec on a single CPU)
- Available similarity fingerprints:
 - Morgan (ECFP-like)
 - FeatMorgan (FCFP-like)
 - RDKit (Daylight-like)
 - atom pairs
 - topological torsions
- Bit vector and count-based fingerprints are supported (searches using count-based fingerprints are slower).
- SMILES- and SMARTS-based substructure querying
- Part of the RDKit open-source distribution since July 2010

The database cartridge

- Using the cartridge:

Similarity search with Morgan fingerprint:

```
vendors=# select \  
  id,tanimoto_sml(morganbv_fp('N=C1OC2=C(C=CC=C2)C=C1',2),mfp2) \  
  from fps where morganbv_fp('N=C1OC2=C(C=CC=C2)C=C1',2)%mfp2 ;  
  id      |      tanimoto_sml  
-----+-----  
  9171448 | 0.538461538461538  
  765434  | 0.538461538461538  
(2 rows)
```

Substructure Search:

```
vendors=# select count(*) from mols where m@>'N=C1OC2=C(C=CC=C2)  
C=C1';  
  count  
-----  
    2854  
(1 row)
```

Cartridge performance

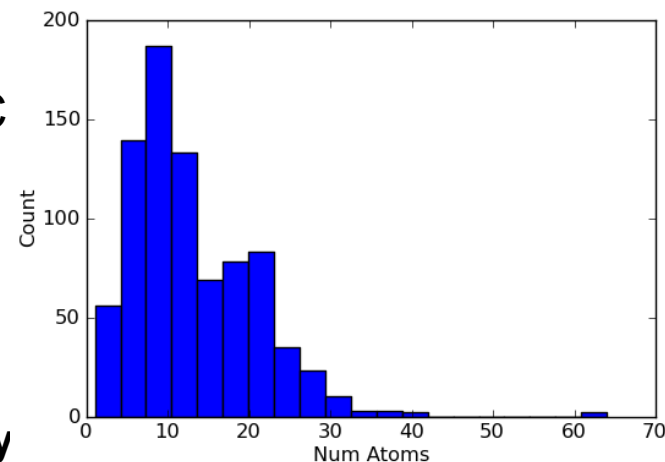
- Database: 100K diverse drug-like molecules from ZINC
 - Molecules load/index time: 109 sec / 343 sec
 - Fingerprints (Morgan2) calculation/index time: 23.1 sec / 9.3 sec
- "Fragments" queries: 500 diverse fragment-like molecules from ZINC
- "Leads" queries: 500 diverse lead-like molecules from ZINC
- Hardware: MacBook Pro (2.5GHz Core2 Duo)
- Do queries via a cross join (i.e. 500 queries x 100K database molecules = 50M possible comparisons/searches)
- Results:

| Query Set | SSS | Run time (sec) | | |
|----------------|------|------------------|------------------|------------------|
| | | Similarity (0.8) | Similarity (0.6) | Similarity (0.5) |
| Zinc Fragments | 23.3 | 14.6 | 36.4 | 37.1 |
| Zinc Leads | 8.2 | 14.8 | 36.2 | 38.5 |

About the substructure fingerprints

- Benchmarking: determine screening accuracy (= number of SSS hits found / number of fingerprint matches) for three different types of queries run against 100K diverse drug-like molecules from ZINC:
 - 823 pieces constructed by doing a BRICS fragmentation of a set of molecules from the pubchem screening set. Size range from 1->64 atoms
 - 500 diverse lead-like molecules from ZINC
 - 500 diverse fragment-like molecules from ZINC

- Results:



Accuracy

| Query Set | Num matches | Avalon ¹ | RDKit-branched | RDKit-linear |
|----------------|-------------|---------------------|----------------|--------------|
| Pubchem Pieces | 2515777 | 0.35 | 0.35 | 0.32 |
| Zinc Fragments | 15643 | 0.30 | 0.20 | 0.13 |
| Zinc Leads | 1302 | 0.49 | 0.11 | 0.03 |

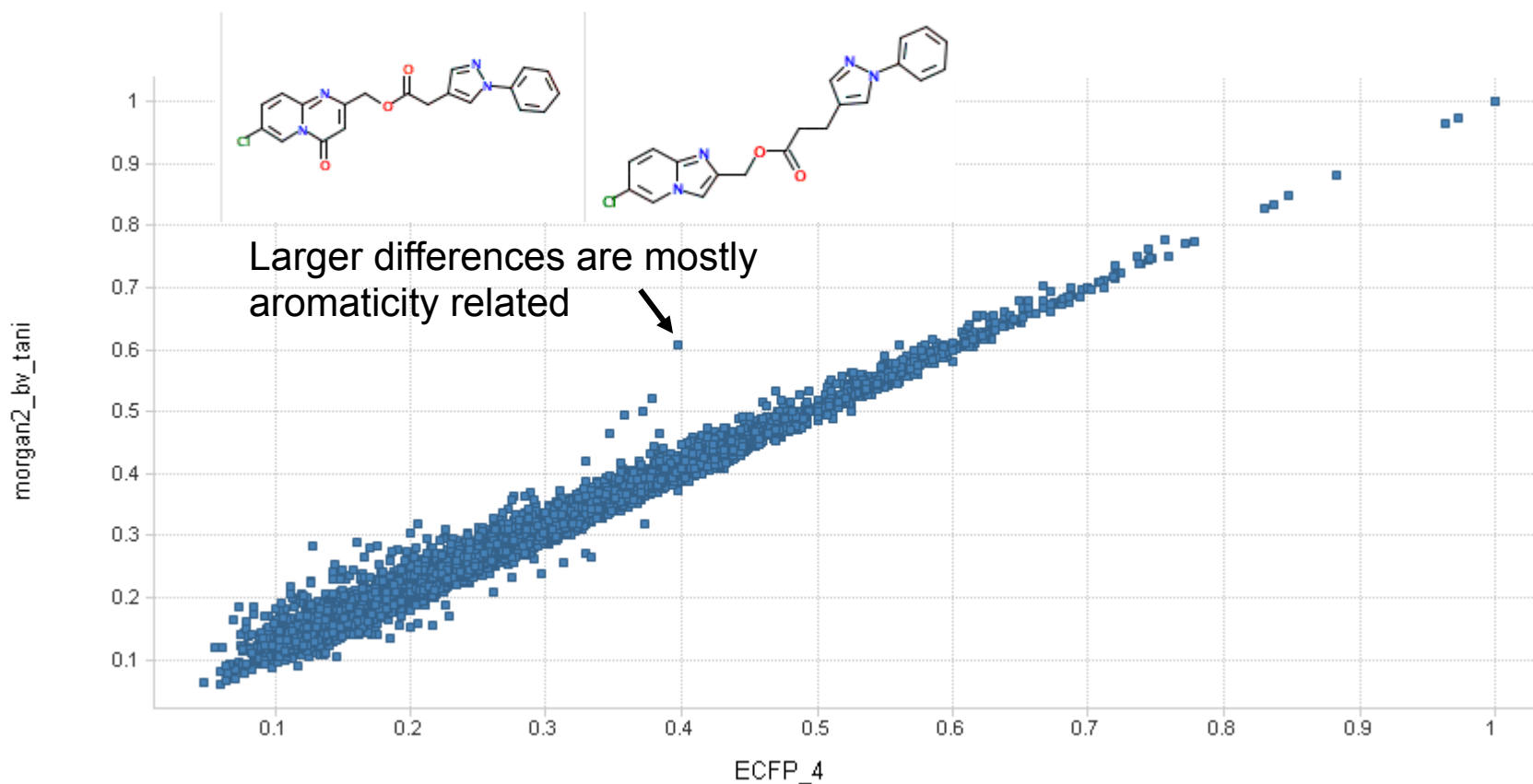
1. Gedeck, P., Rohde, B. & Bartels, C. JCIM **46**:1924-36 (2006).

Comparing similarity measures

- Pick 10K random pairs of vendor compounds that have at least some topological similarity to each other (Avalon similarity ≥ 0.5)
- Compare similarities calculated with Pipeline Pilot and the RDKit

Comparing similarity measures

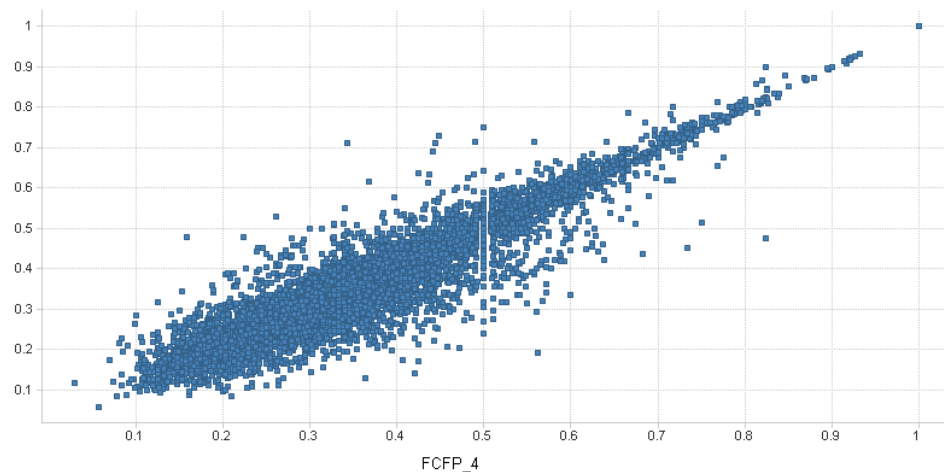
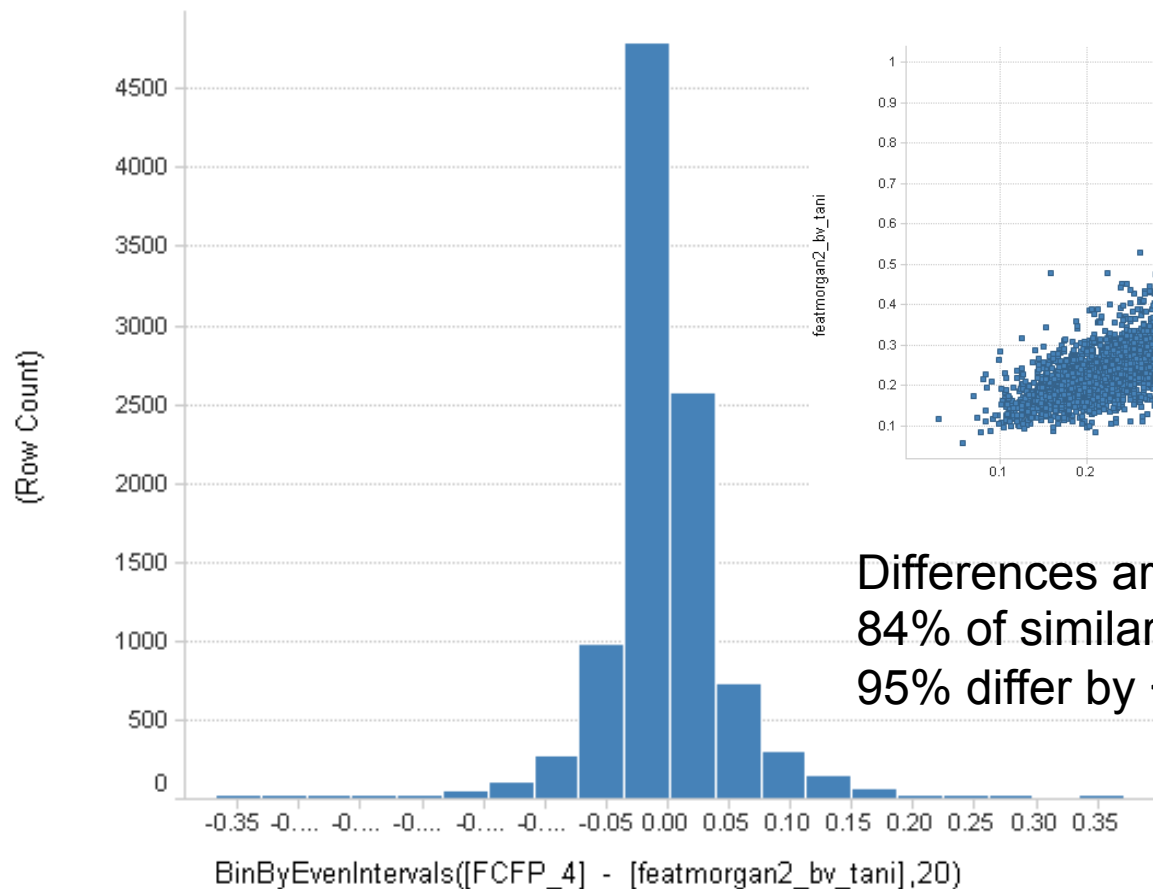
- RDKit Morgan2 vs PP ECFP4



- RDKit Morgan3 vs PP ECFP6 is similar

Comparing similarity measures

- RDKit FeatMorgan2 vs PP FCFP4



Differences are due to feature definitions
84% of similarities differ by <0.05
95% differ by <0.1

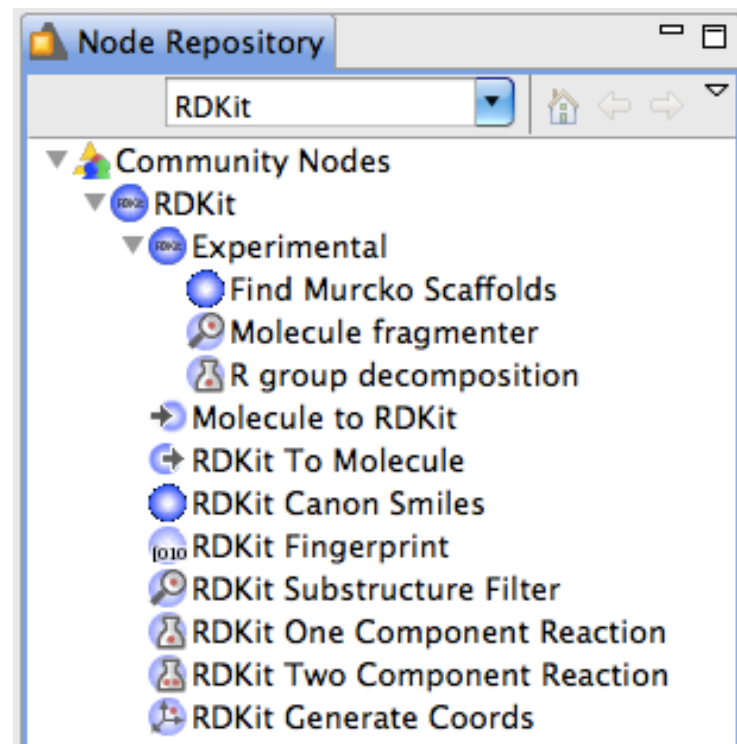
Overview

- RDKit: what is it?
- RDKit + PostgreSQL
- **RDKit + Knime**

Knime integration¹

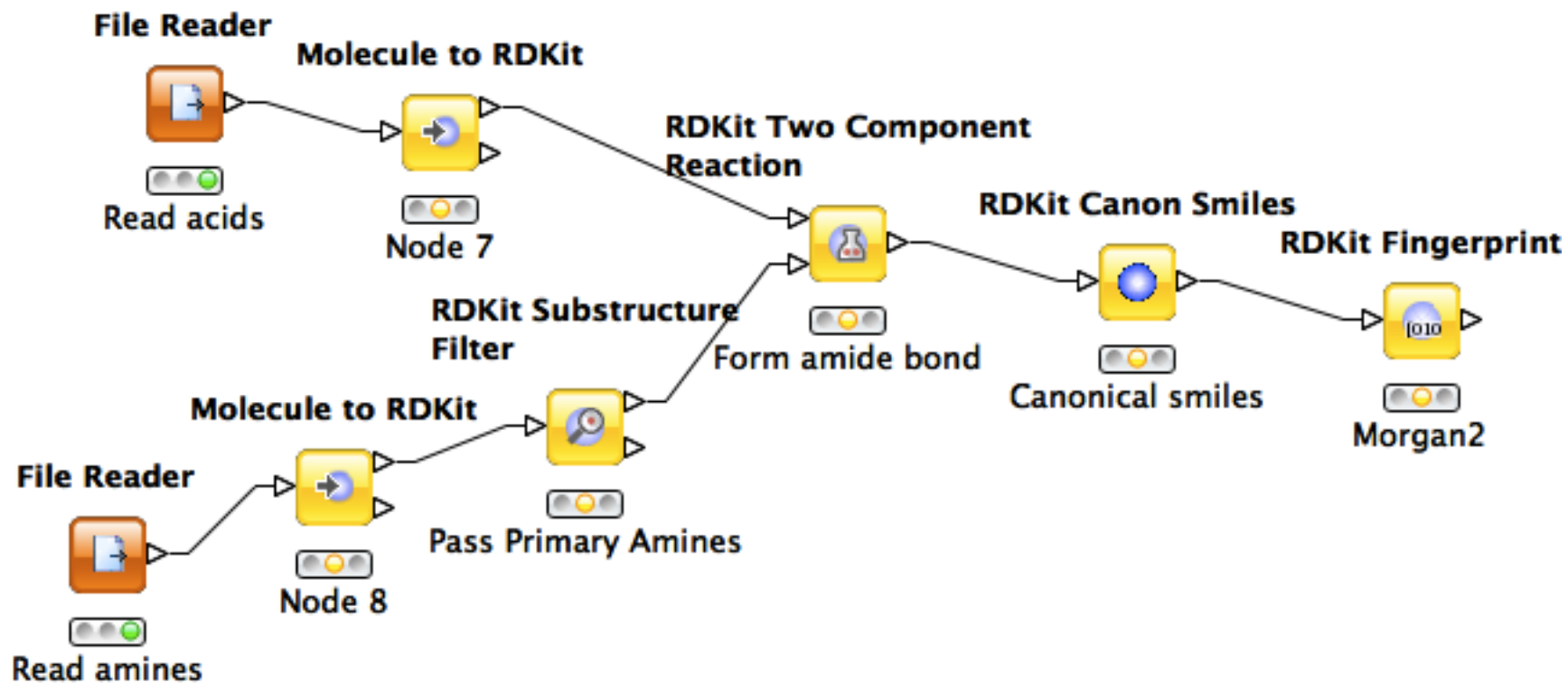
- Out of the box Knime is strong on data processing and mining, weak on chemistry.
- Goal: develop a set of *open-source* RDKit-based nodes for Knime that provide basic cheminformatics functionality

- Distributed from knime community site
- Binaries available as an update site (no RDKit build/installation required)
- Work in progress: more nodes being added frequently



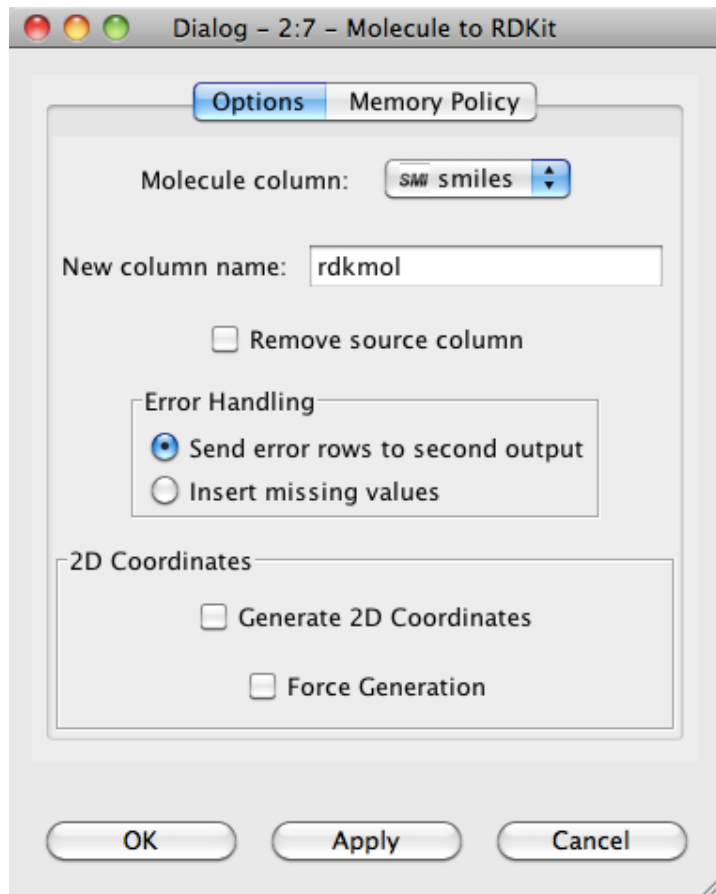
¹ Work done together with knime.com

Knime integration



Handling molecules

Molecule to RDKit



Output data - 2:7 - Molecule to RDKit

File

Table "default" - Rows: 6 Spec - Columns: 2 Properties

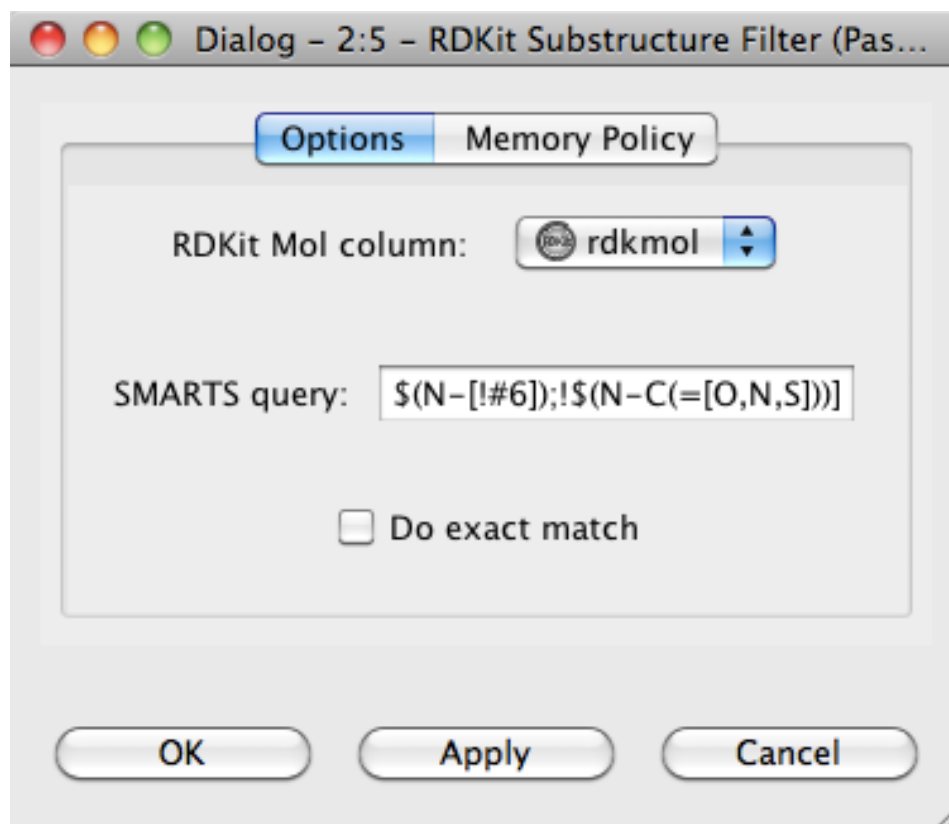
| Row ID | SMI smiles | rdkmol |
|--------|----------------|--------|
| 3 | C1CC1C(=O)O | |
| 4 | c1ccncc1C(=... | |
| 5 | c1ccccc1 | |

Substructure search

RDKit Substructure
Filter

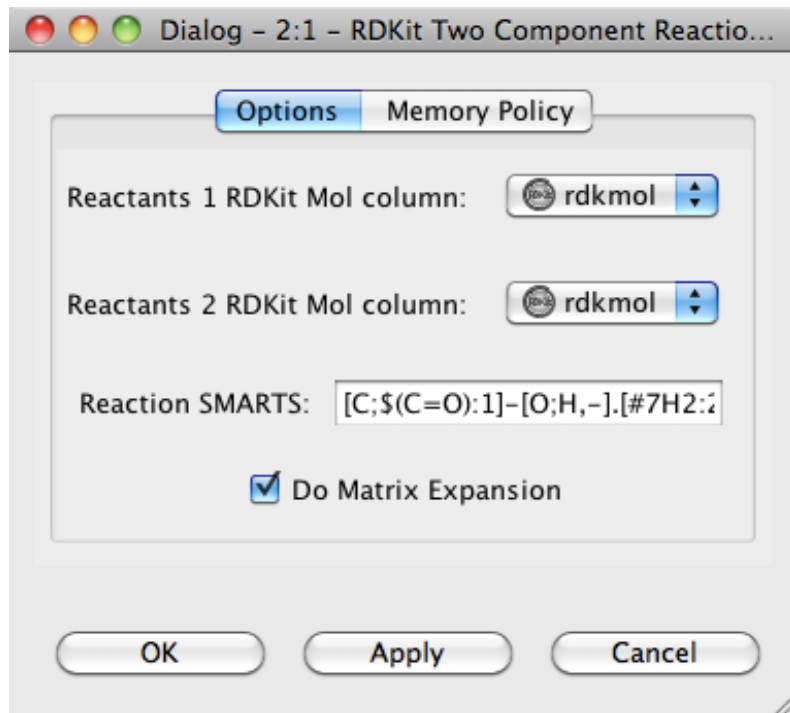
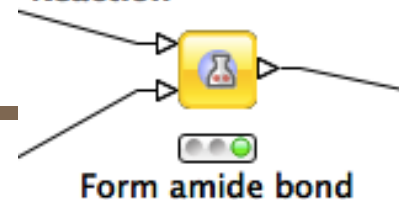


Pass Primary Amines



Reaction handling

RDKit Two Component Reaction



Product molecules - 2:1 - RDKit Two Component Reaction (Form amide bond)

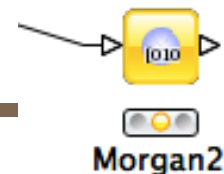
File

Table "output" - Rows: 12 Spec - Columns: 6 Properties

| Row ID | Product | Product... | Reacta... | Reactant 1 | Reacta... |
|---------|---------|------------|-----------|------------|-----------|
| 2_0_0_0 | | 0 | 2 | | 0 |
| 2_1_0_0 | | 0 | 2 | | 1 |
| 3_0_0_0 | | 0 | 3 | | 0 |
| 3_1_0_0 | | 0 | 3 | | 1 |

Fingerprinting node

RDKit Fingerprint



Dialog - 2:6 - RDKit Fingerprint (Morgan2)

Options Memory Policy

RDKit Mol column: Product

FP type: morgan

New column name: morgan2

Remove source column

Num Bits: 1024

Radius: 2

Min Path Length: 1

Max Path Length: 7

LayerFlags: 65535

OK Apply Cancel

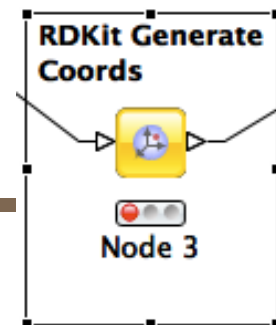
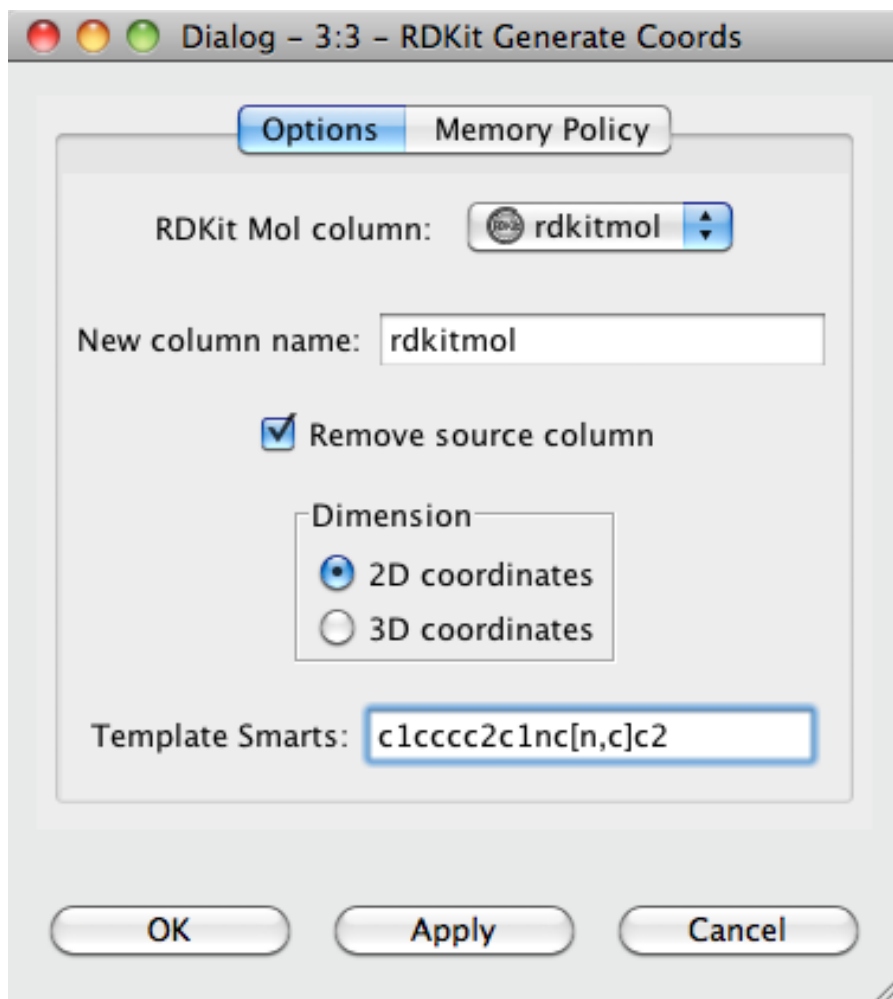
morgan
featmorgan
atompair
torsion
rdkit
layered

FP type

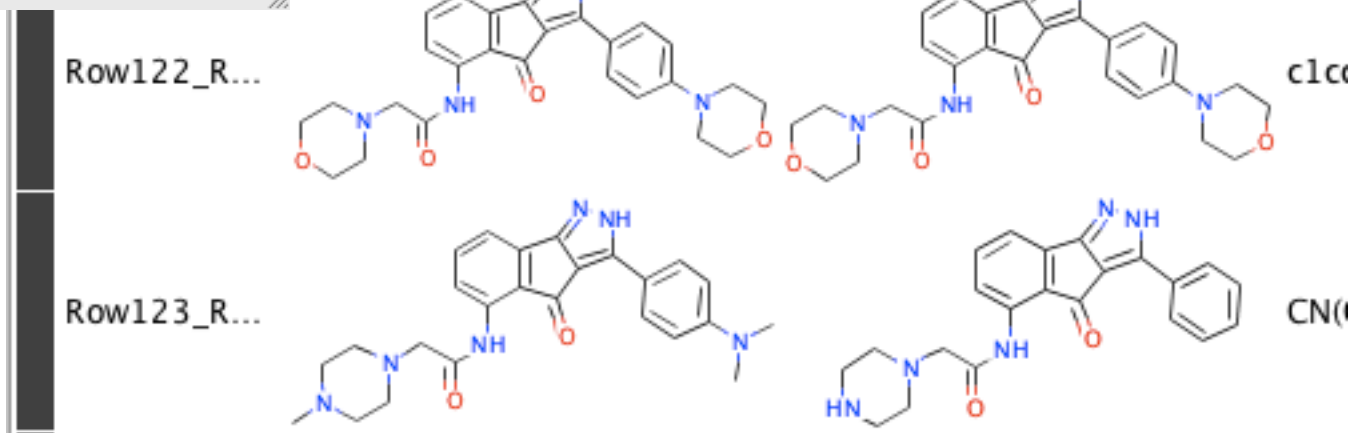
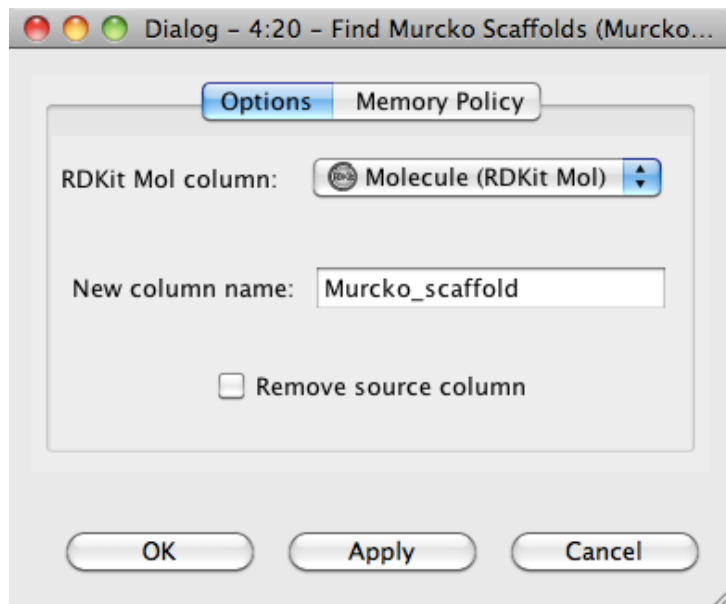
The type of fingerprint to generation. Choices are:

- Morgan: Circular fingerprint based on the Morgan algorithm and connectivity invariants (ECFP-like)
- FeatMorgan: Circular fingerprint based on the Morgan algorithm and feature invariants (FCFP-like)
- AtomPair: Atom-pair fingerprint
- Torsion: Topological-torsion fingerprint
- RDKit: Daylight-like topological fingerprint
- Layered: An experimental substructure-matching fingerprint

Coordinate generation

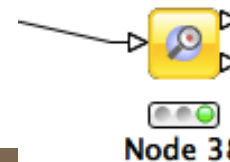


Murcko Scaffolds



Fragmentation

Molecule fragmenter



Dialog - 4:38 - Molecule fragmenter

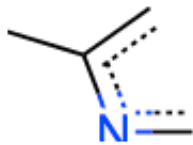
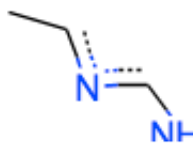
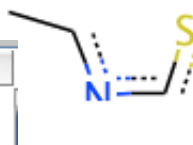

Options Memory Policy

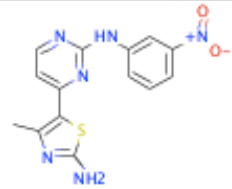
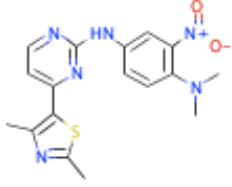
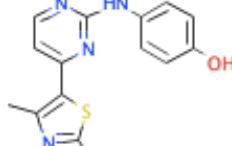
RDKit Mol column:

Min Path Length:

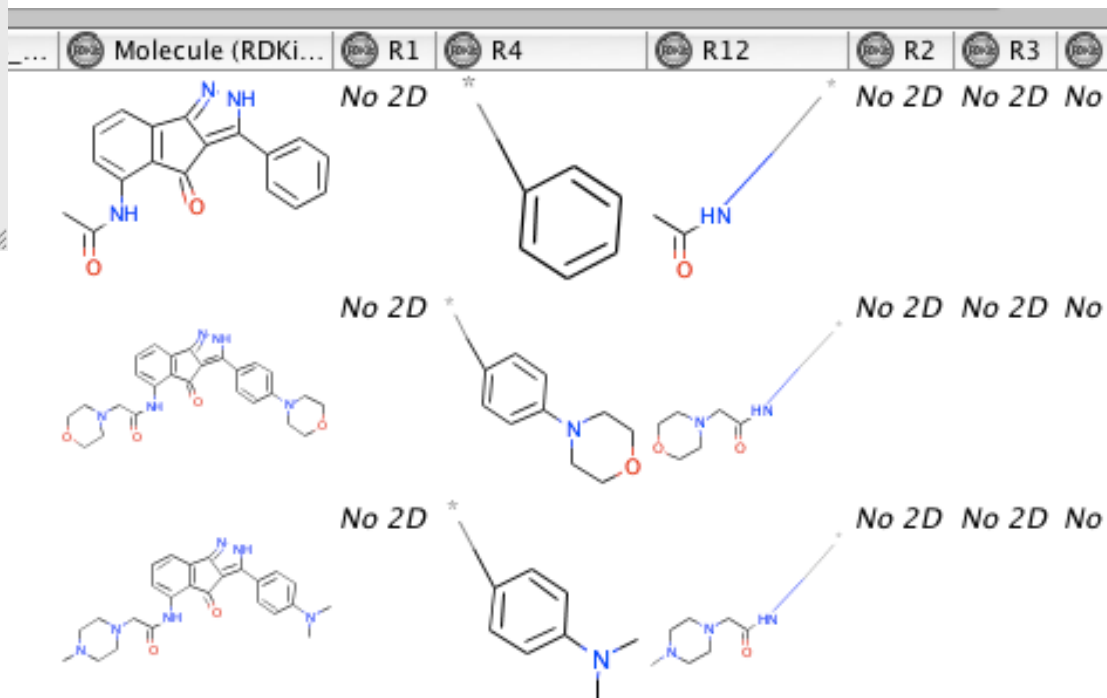
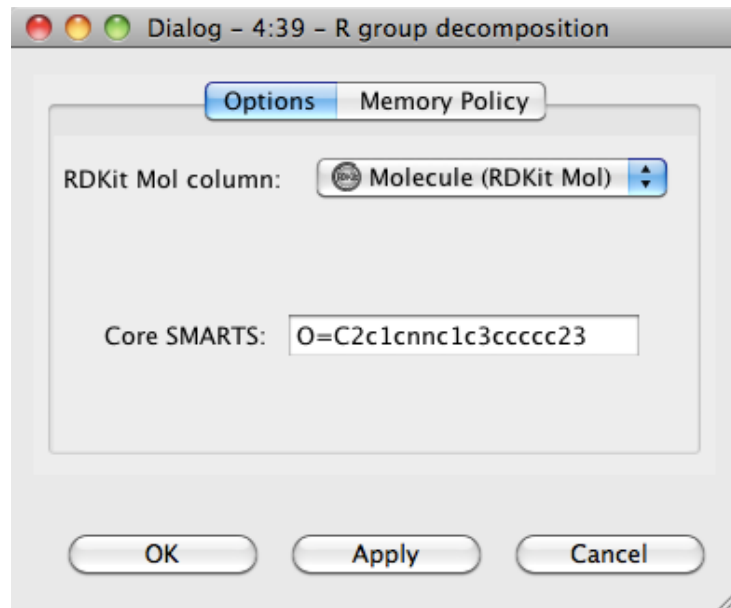
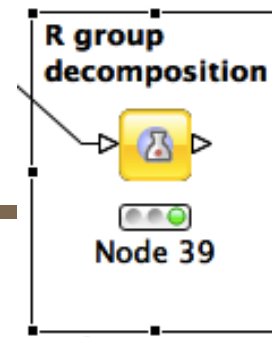
Max Path Length:

OK Apply Cancel

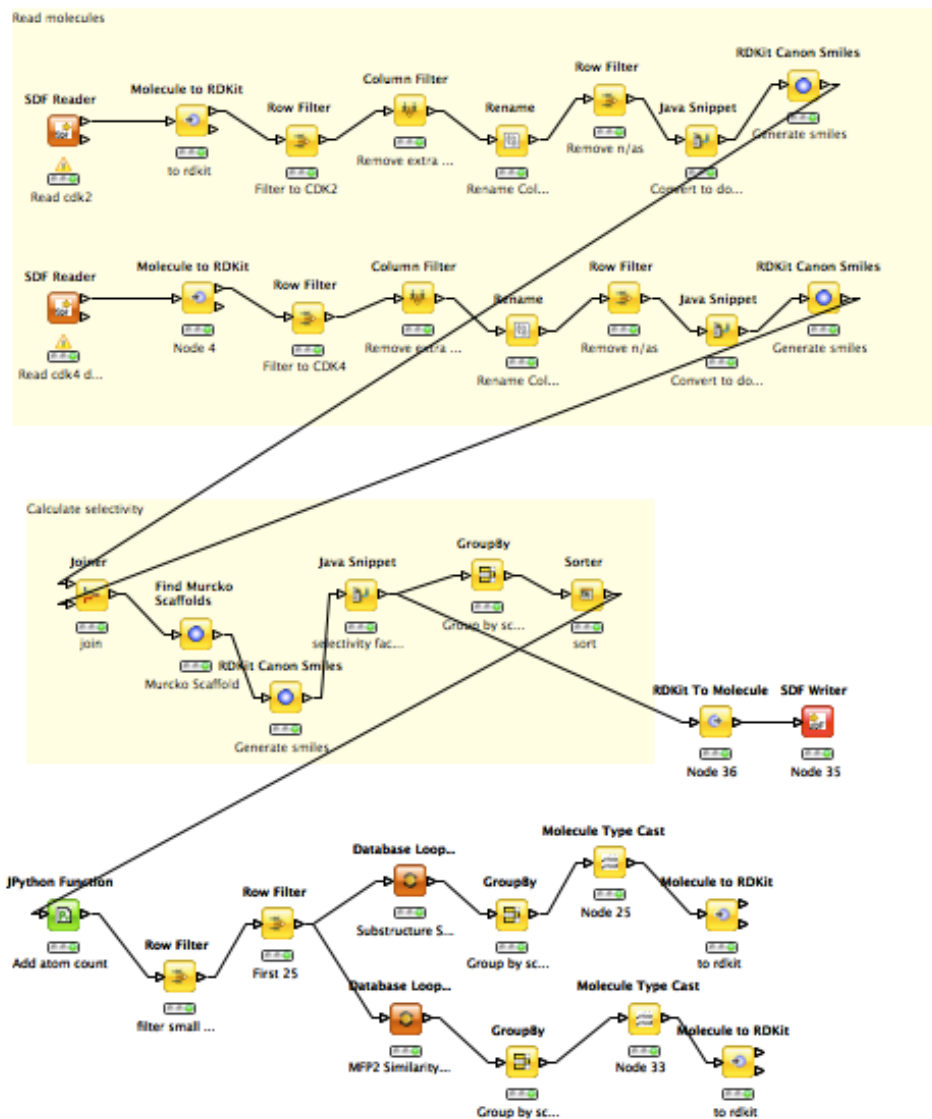
| Row ID | Frags... | Fragment | SM Fragm... | Frags... | Count |
|---------|----------|--|-------------|----------|-------|
| frag_01 | 1 |  | cnc(C)c | 4 | 20 |
| frag_11 | 2 |  | CcncN | 4 | 5 |
| | |  | Ccncs | 4 | 12 |
| | |  | cnc-c | 4 | 20 |

| Row ID | name | Enzym... | Molecule (RDKit Mol) | Fragment indices |
|--------|-----------------|----------|---|------------------|
| Row1 | idine deriv.... | n/a |  | [1,2,3,...] |
| Row2 | idine deriv.... | n/a |  | [1,560,3,...] |
| Row3 | idine deriv.... | n/a |  | [1,2,3,...] |

R Group decomposition

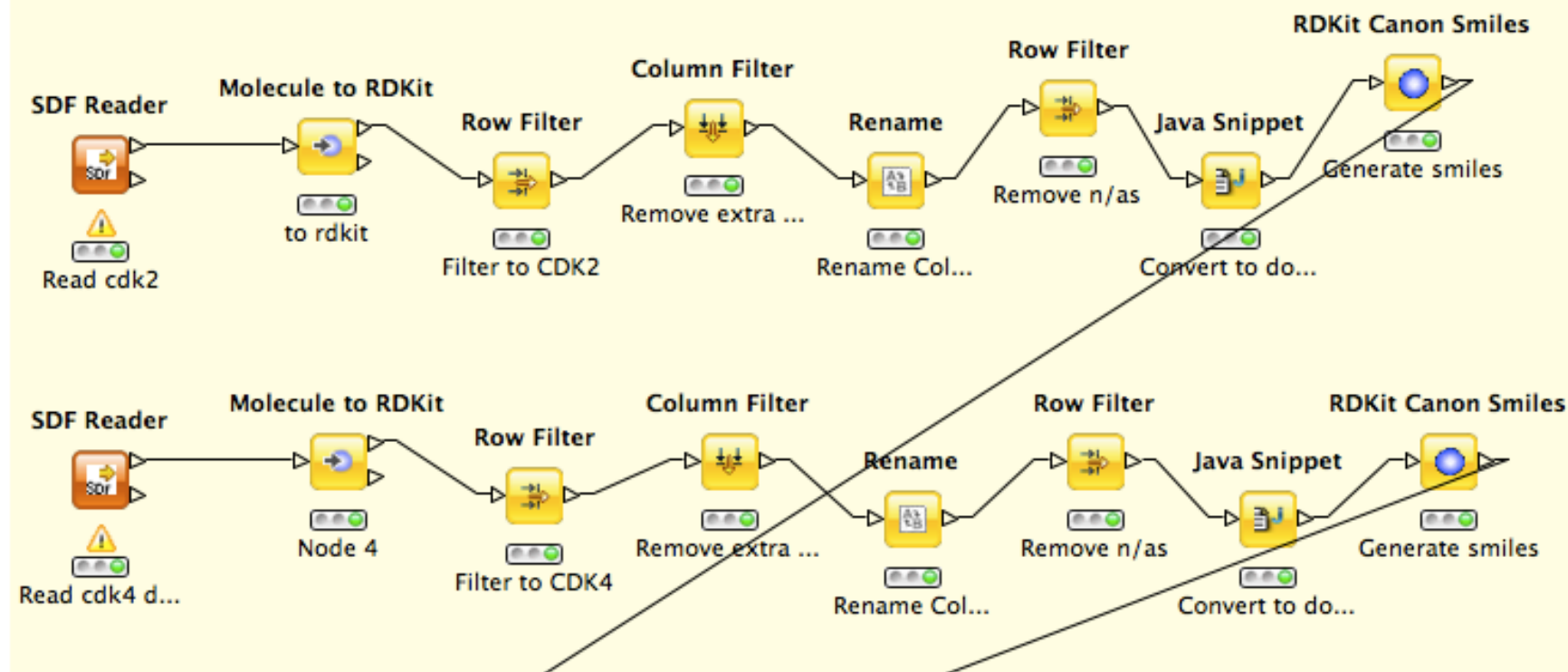


More complex example: CDK2/CDK4 selectivity

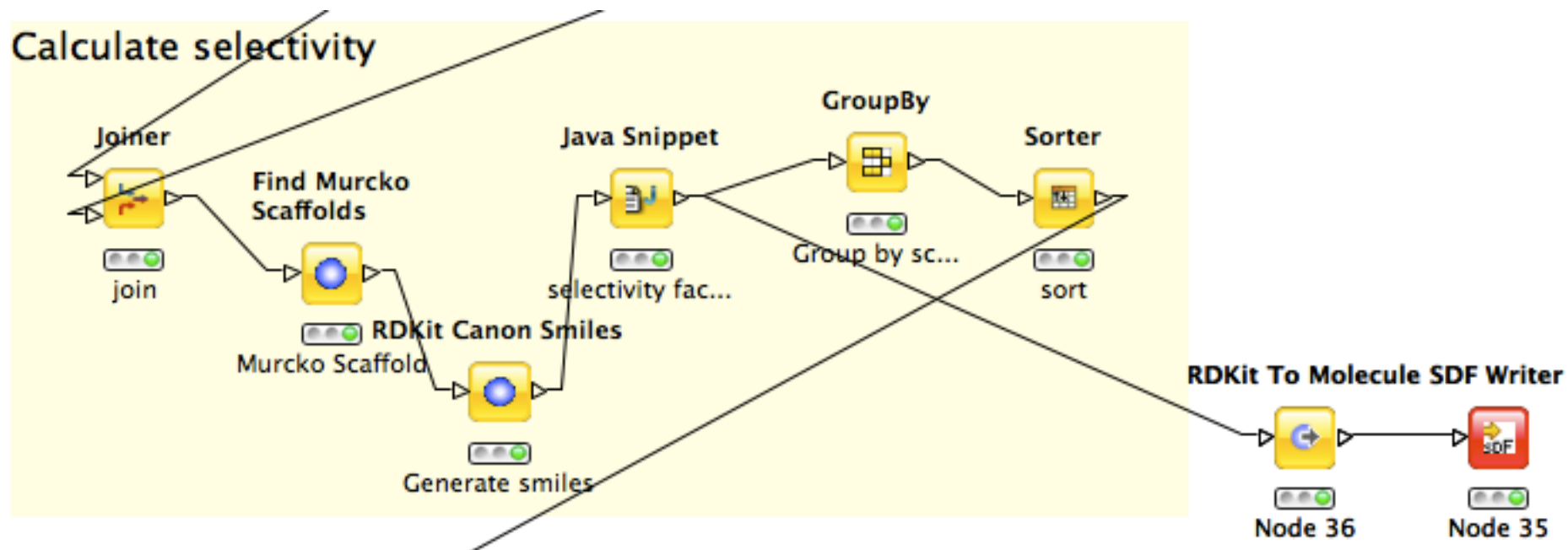


More complex example... cont.

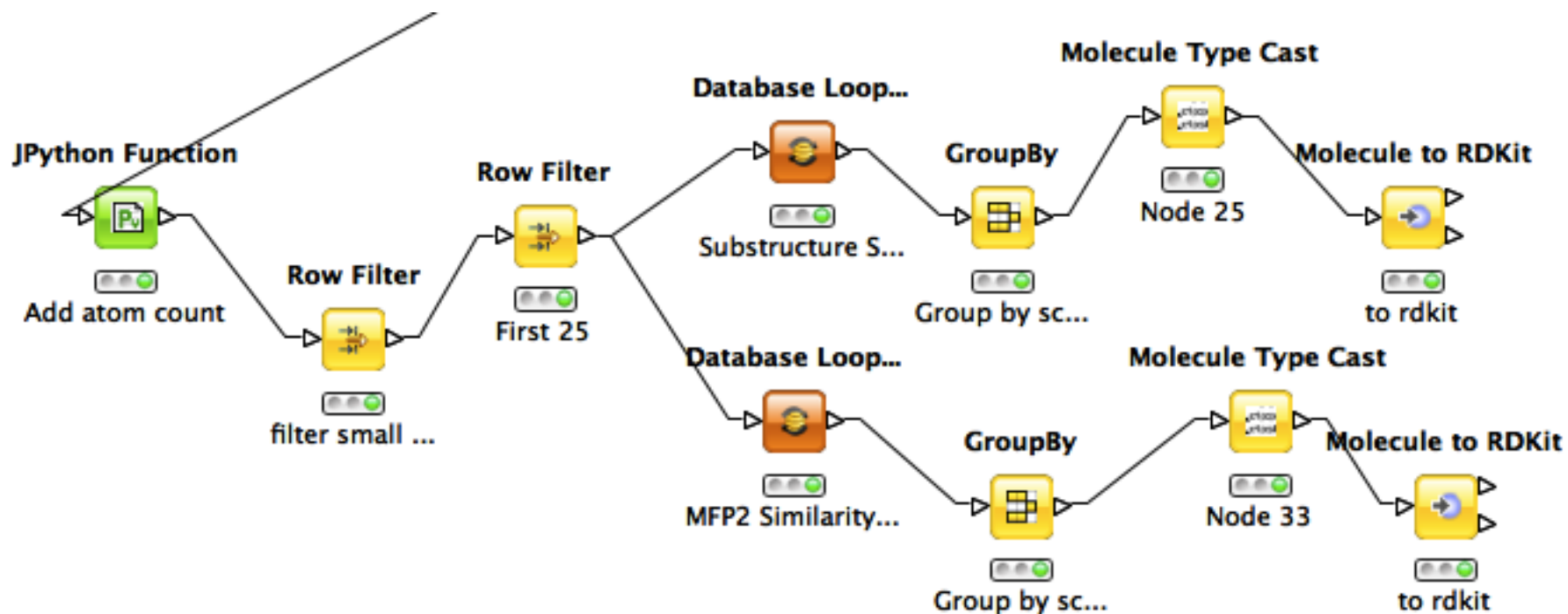
Read molecules



More complex example... cont.

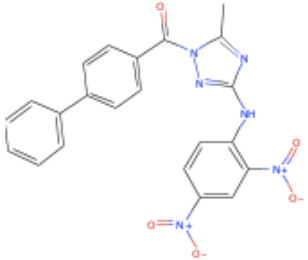
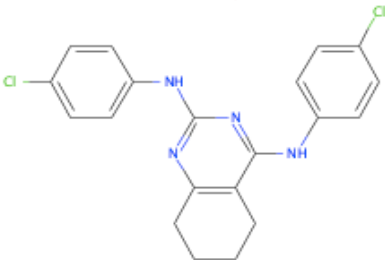
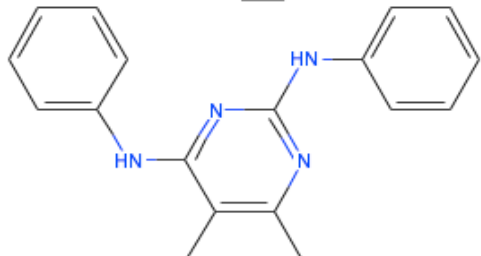


More complex example... cont.

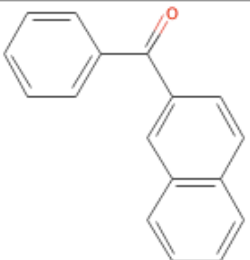
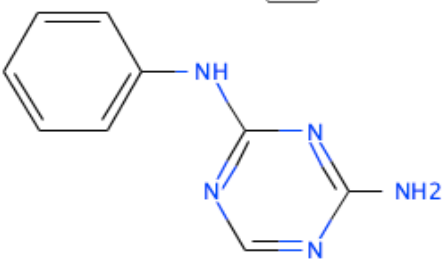
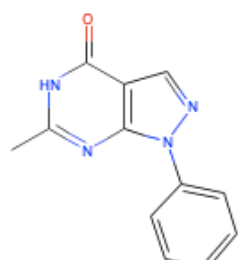


More complex example... cont.

SSS Results

| Row ID | id | Molecule (RDKit Mol) |
|--------|--------|--|
| Row0 | 162583 |  |
| Row1 | 178192 |  |
| Row2 | 247618 |  |

Similarity Results

| Row ID | id | First(si... | Molecule (RDKit Mol) |
|--------|--------|-------------|--|
| Row0 | 167309 | 0.5 |  |
| Row1 | 178689 | 0.5 |  |
| Row2 | 196649 | 0.674 |  |

Wrapping up

- What is it?
 - Cheminformatics toolkit useable from C++, Python, Java
 - Postgresql cartridge for substructure/similarity searching
 - Open-source Knime nodes for cheminformatics
- Web presence:
 - Main site: <http://www.rdkit.org>
 - Knime nodes: <http://tech.knime.org/community/rdkit>