RDKit, PostgreSQL, and Knime: Open-source cheminformatics in big pharma

Gregory Landrum, Richard Lewis, Andrew Palmer, Nikolaus Stiefl.

NIBR IT and Global Discovery Chemistry

Novartis Institutes for BioMedical Research, Basel and Cambridge

MIOSS 2011

Hinxton, 4 May 2011

Overview

- RDKit: what is it?
- RDKit + PostgreSQL
- RDKit + Knime
- Case study: matched pairs analysis
- Contributing to open source from big pharma

NOVARTIS

Acknowledgements

- Novartis:
  - Tom Digby (Legal)
  - John Davies (CPC/LFP)
  - Eddie Cao (NIBR IT)
  - 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
- Entagen (SWIG wrappers):
  - Chris Bouton
  - Erik Bakke
  - James Hardwick
- knime.com
  - Michael Berthold
  - Thorsten Meinl
  - Bernd Wiswedel

NOVARTIS

Overview

- **RDKit: what is it?**
- RDKit + PostgreSQL
- RDKit + Knime
- Case study: matched pairs analysis
- Contributing to open source from big pharma

NOVARTIS

RDKit: What is it?

- Python (2.x), Java, and C++ toolkit for cheminformatics
  - Core data structures and algorithms in C++
  - Heavy use of Boost libraries
  - Python wrapper generated using Boost.Python
  - Java wrapper generated with SWIG
- 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

NOVARTIS

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

NOVARTIS

What can you do with it?

A laundry list

- Input/Output: SMILES/SMARTS, SDF, TDT, SLN[1], Corina mol2[1]
- "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[1]
- Gasteiger-Marsili charges
- Hierarchical subgraph/fragment analysis
- RECAP and BRICS implementations

[1] *functional, but not great implementations*

NOVARTIS

What can you do with it?
A laundry list, cntd

- Feature maps
- Shape-based similarity
- Molecule-molecule alignment
- Shape-based alignment (subshape alignment) [1]
- Integration with PyMOL for 3D visualization
- Database integration
- Molecular descriptor library:
  - Topological (κ3, Balaban J, etc.)
  - Electrotopological state (Estate)
  - clogP, 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.)

[1] *functional, but not great implementations*

NOVARTIS

Things you should know

- Molecules should be “correct”: i.e. there should be a valid Lewis-dot structure. If not, they will be rejected:

```
>>> Chem.MolFromSmiles('CC(F)(Cl)(Br)I')
[08:58:09] Explicit valence for atom # 1 C, 5, is greater
than permitted
```

- The software generally doesn’t try and read the user’s mind

Nitro groups and N-oxides are repaired:

```
>>> Chem.CanonSmiles('CN(=O)=O')
'C[N+](=O)[O-]'
>>> Chem.CanonSmiles('c1ccccn1=O')
'[O-][n+]1ccccc1'
```

but some odd constructs (this one from CHEMBL) are not:

```
>>> Chem.MolFromSmiles('CN=N#N')
[16:30:08] Explicit valence for atom # 2 N, 5, is greater
than permitted
  ... snip ...
>>> Chem.CanonSmiles('CN=[N+]=[N-]')
'CN=[N+]=[N-]'
```

NOVARTIS

Overview

- RDKit: what is it?
- **RDKit + PostgreSQL**
- RDKit + Knime
- Case study: matched pairs analysis
- Contributing to open source from big pharma

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

NOVARTIS

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)
```

NOVARTIS

Cartridge performance

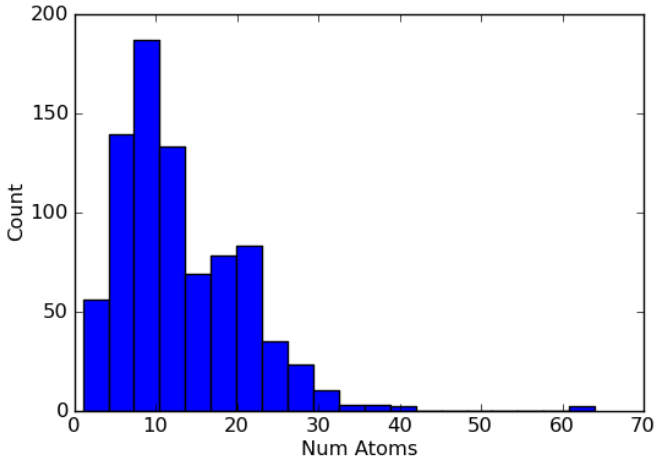
- Database: 100K diverse drug-like molecules from ZINC
  - Molecules load/index time: 109 sec / 343 sec
  - Fingerprints (Morgan2) calculation/index time: 23.1 sec / 9.3 sec
- "Fragments" queries: 500 diverse fragment-like molecules from ZINC
- "Leads" queries: 500 diverse lead-like molecules from ZINC
- 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:

		Run time (sec)		
Query Set	**SSS**	**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

NOVARTIS

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).

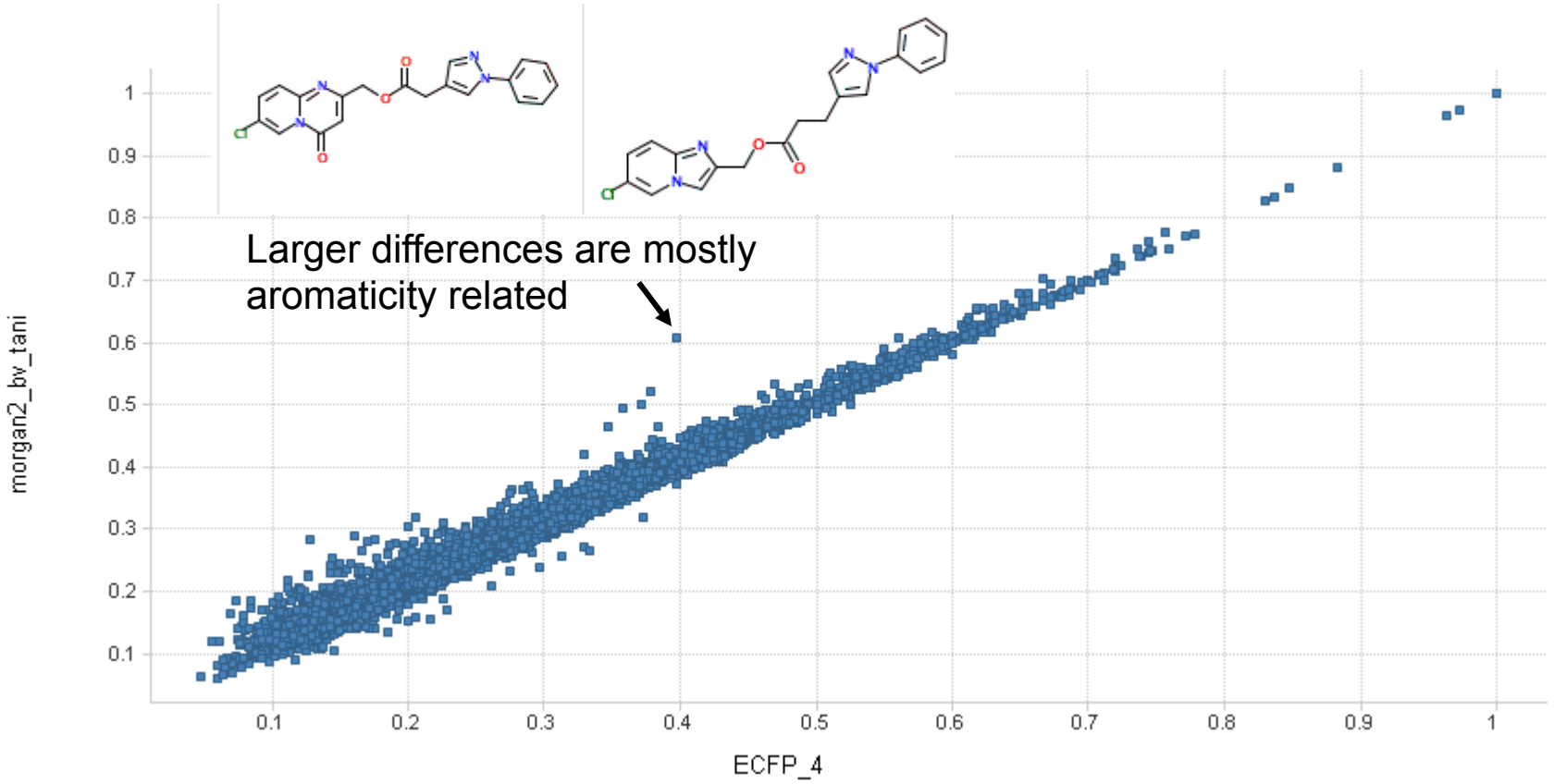
NOVARTIS

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

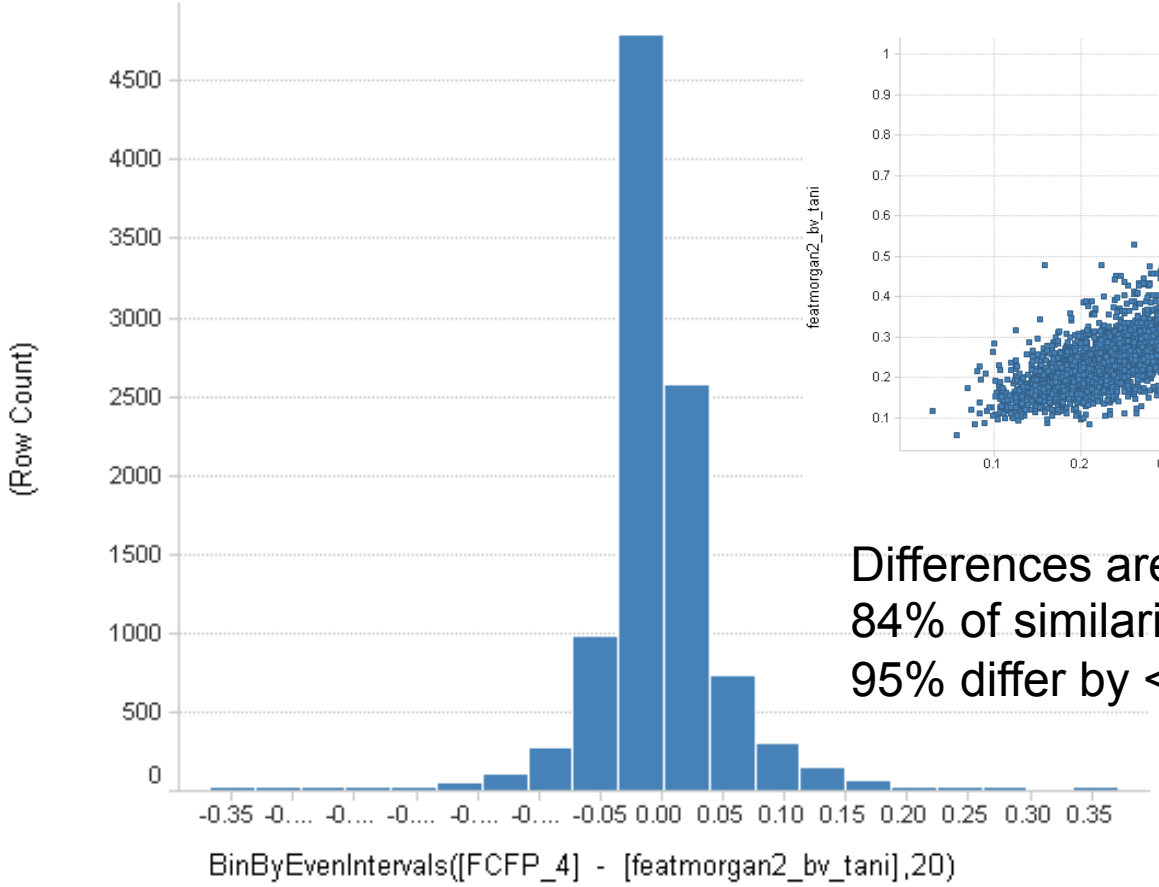


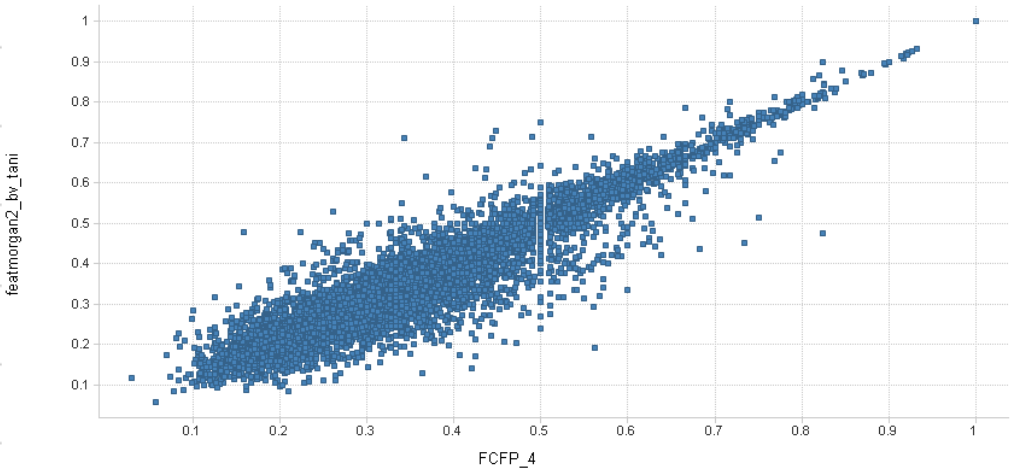
- RDKit Morgan3 vs PP ECFP6 is similar

NOVARTIS

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

NOVARTIS

Overview

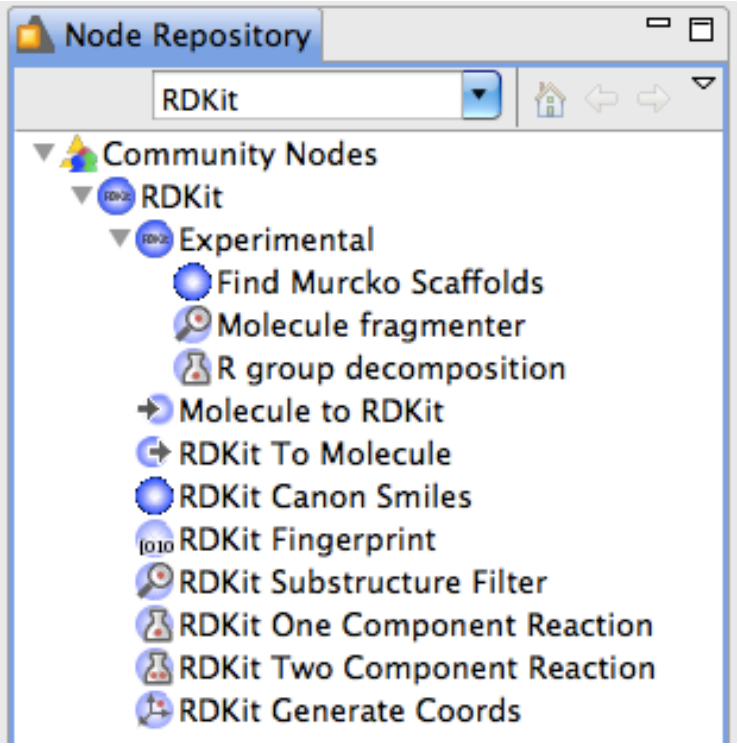
- RDKit: what is it?
- RDKit + PostgreSQL
- **RDKit + Knime**
- Case study: matched pairs analysis
- Contributing to open source from big pharma

NOVARTIS

Knime integration[1]

- 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

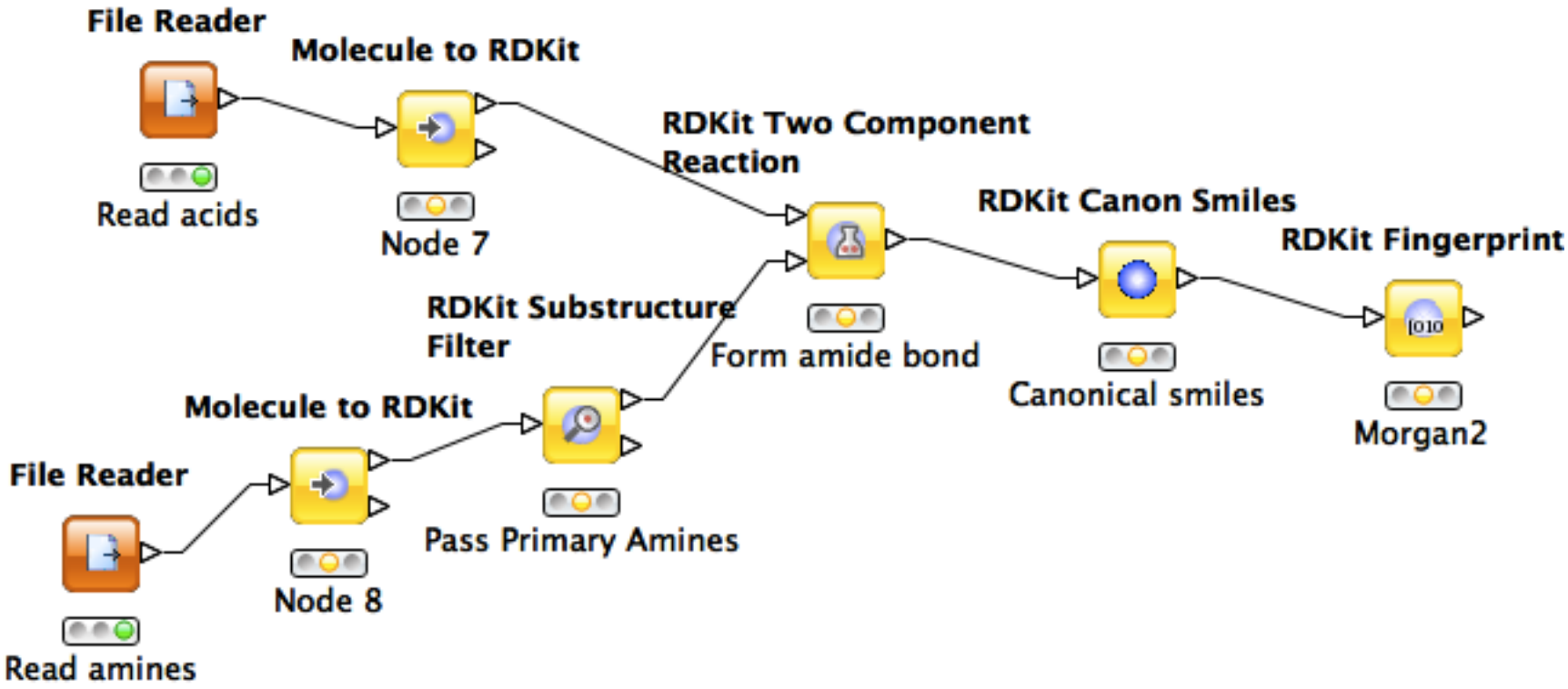
- Distributed from knime community site
- Binaries available as an update site (no RDKit build/installation required)
- Work in progress: more nodes being added frequently



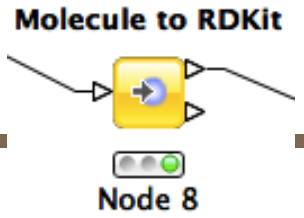
[1] Work done together with knime.com

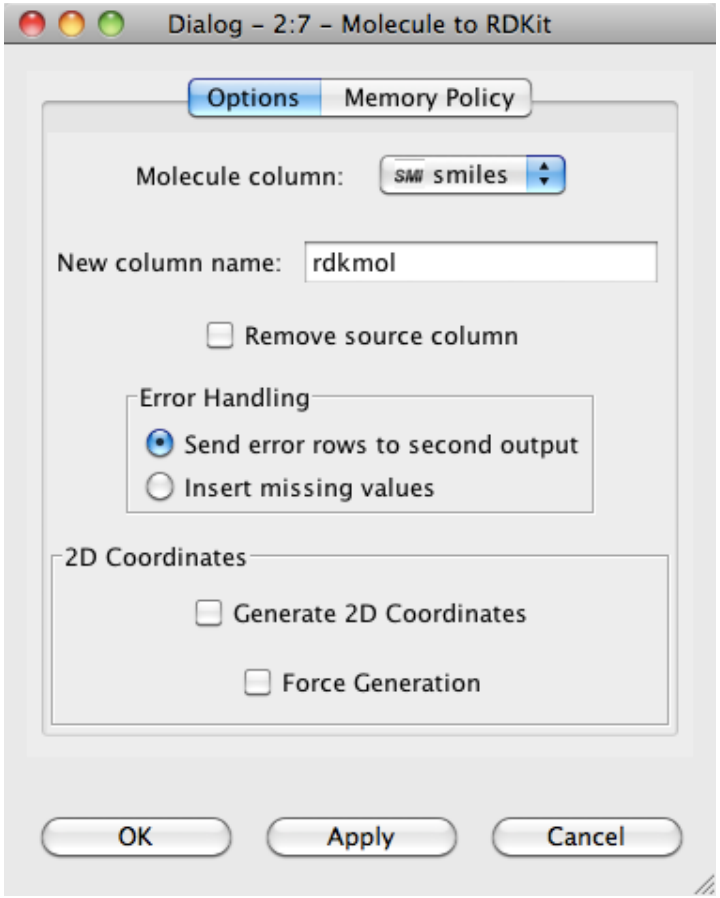
NOVARTIS

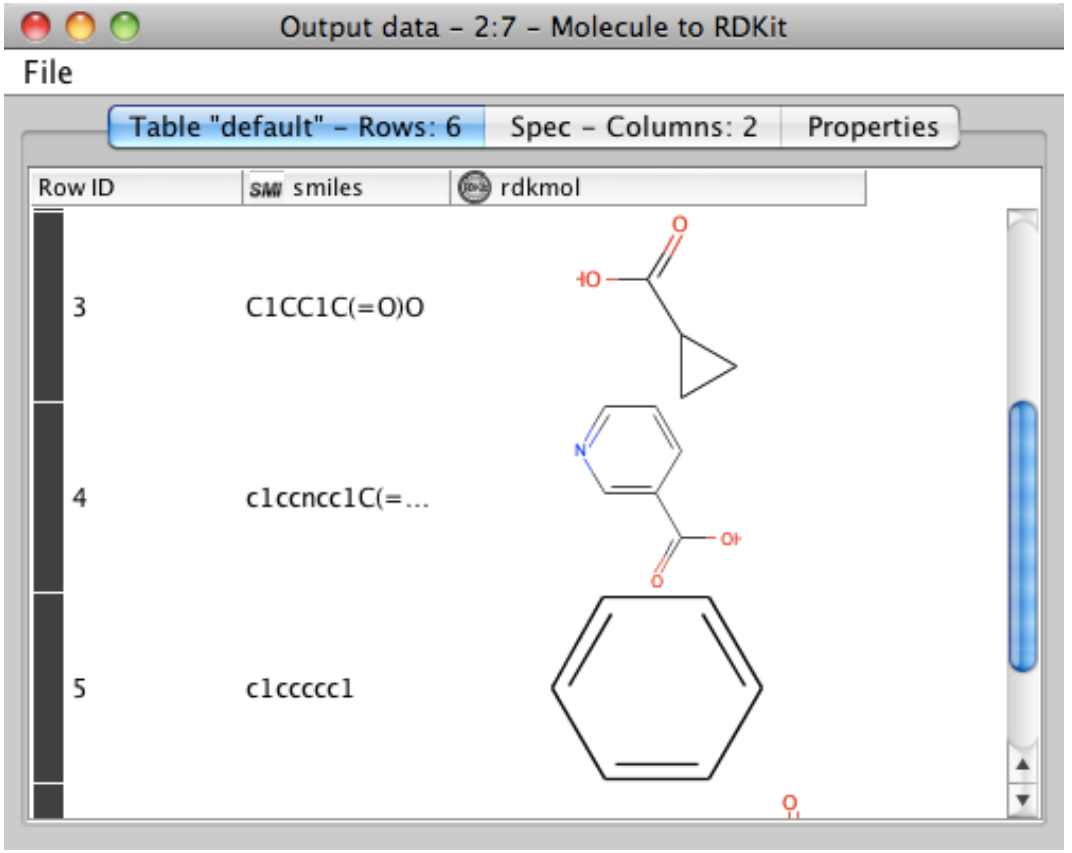
Knime integration



Handling molecules

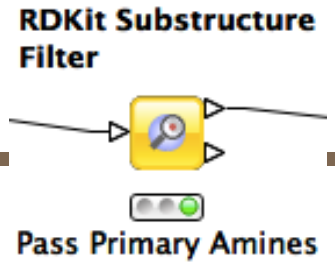


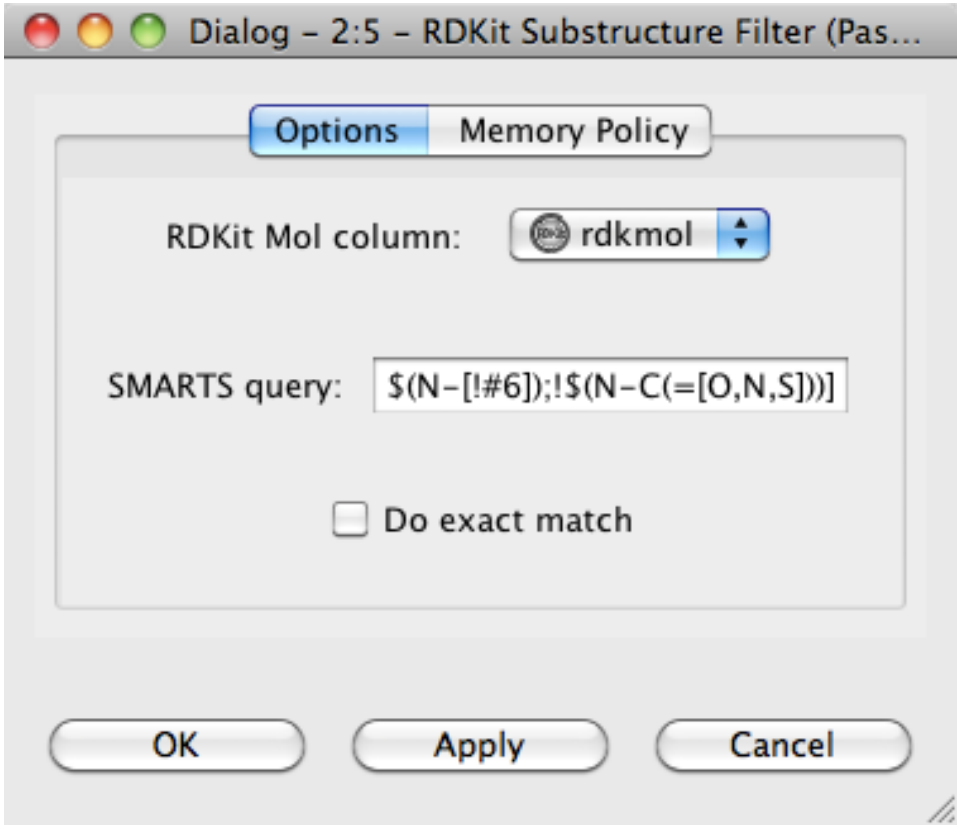




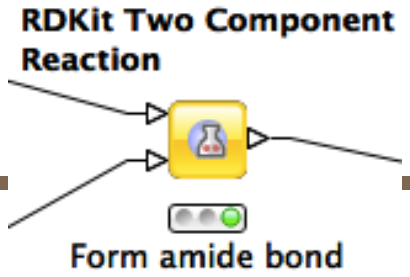
NOVARTIS

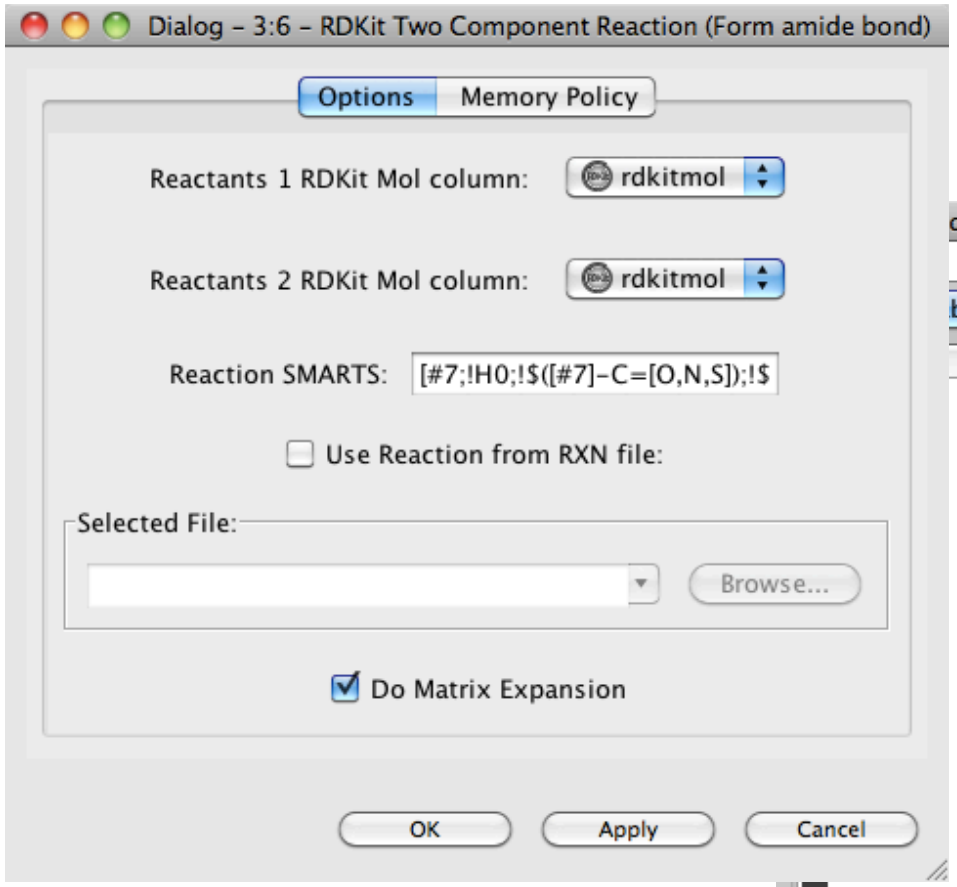
Substructure search

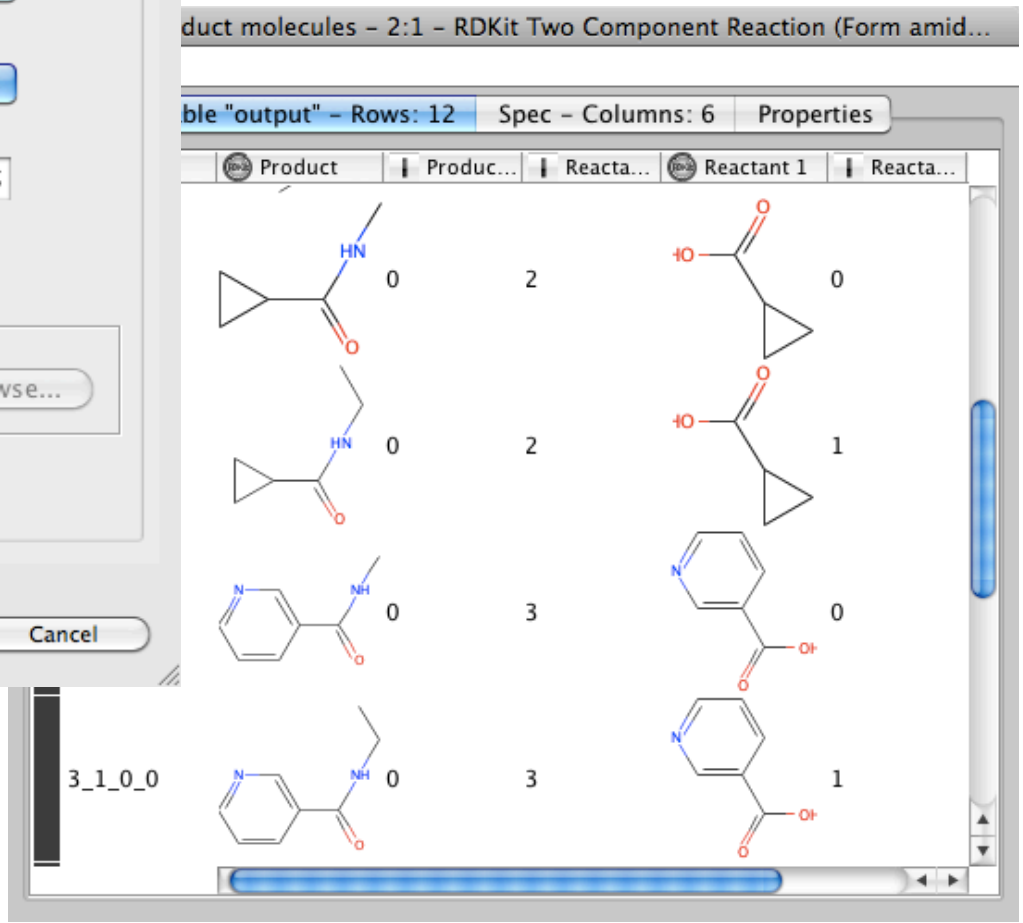




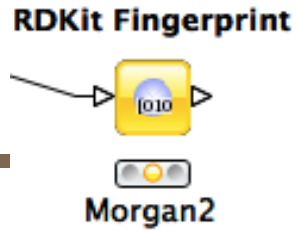
Reaction handling

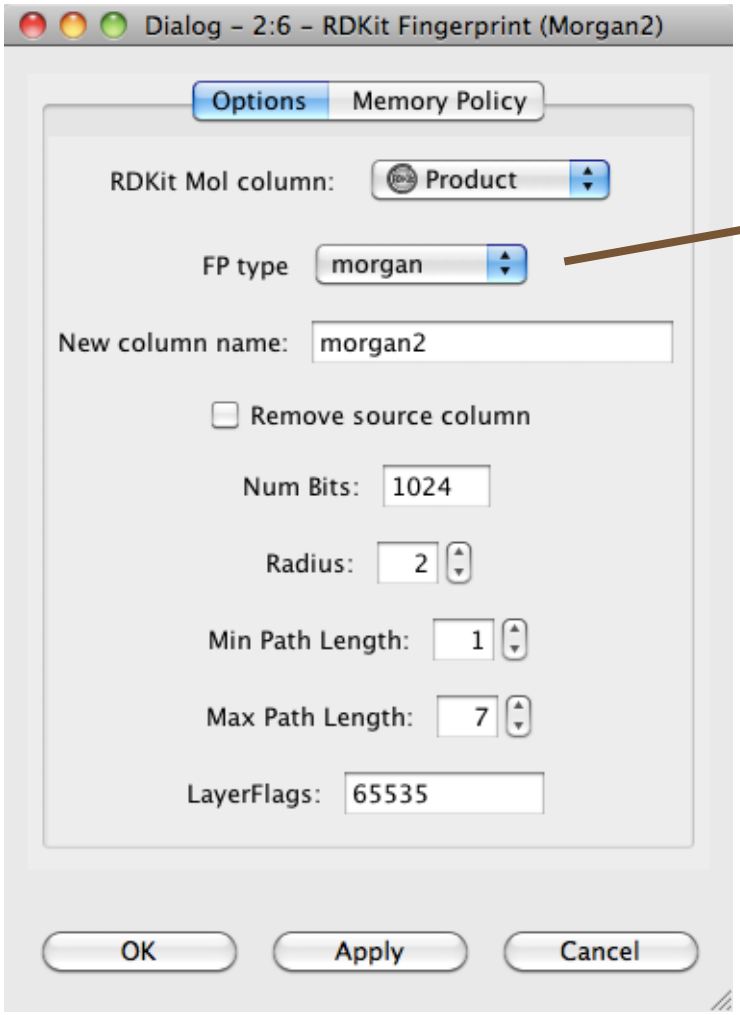






Fingerprinting node



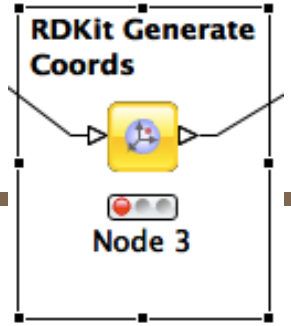


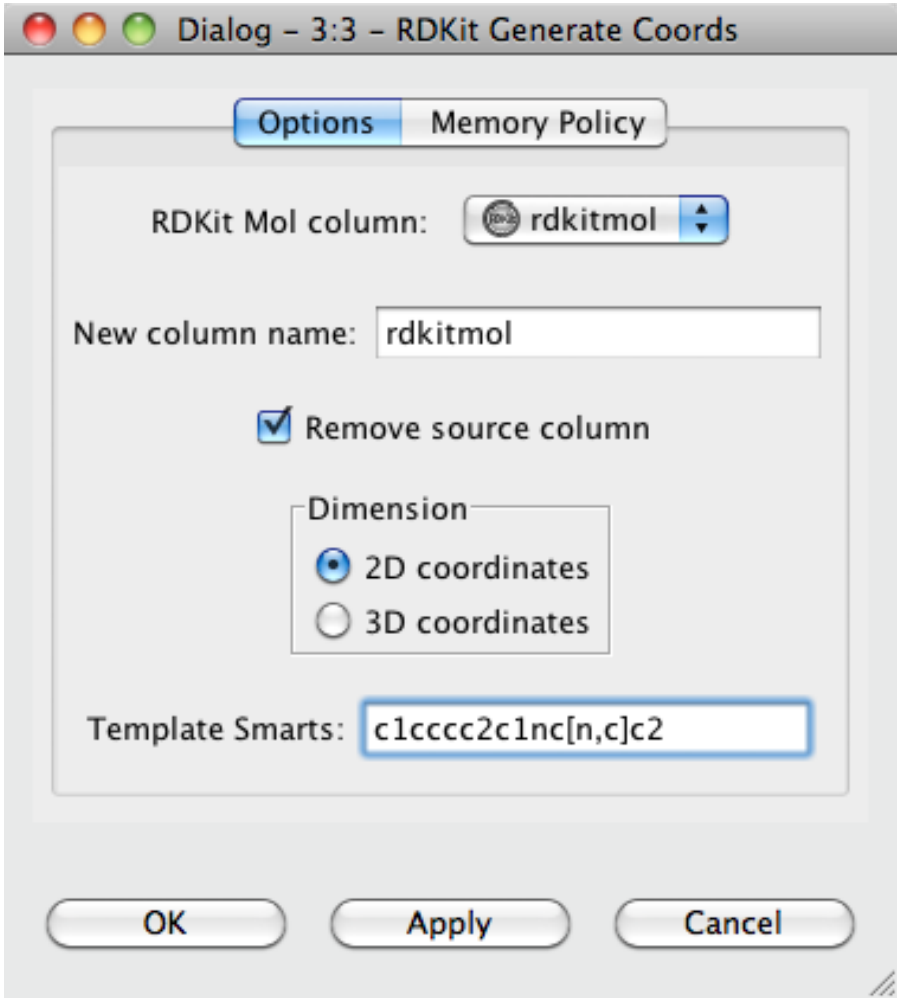
FP type

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 fingeprint
- RDKit: Daylight-like topological fingerprint
- Layered: An experimental substructure-matching fingerprint

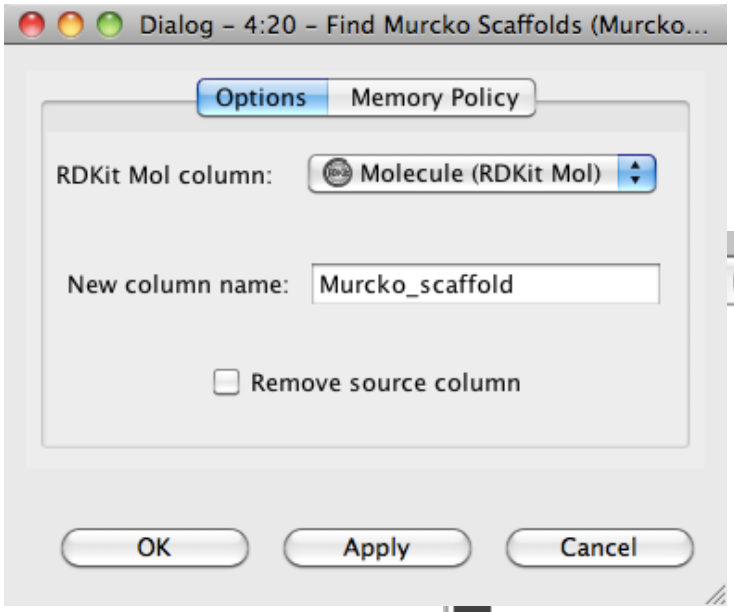
Coordinate generation

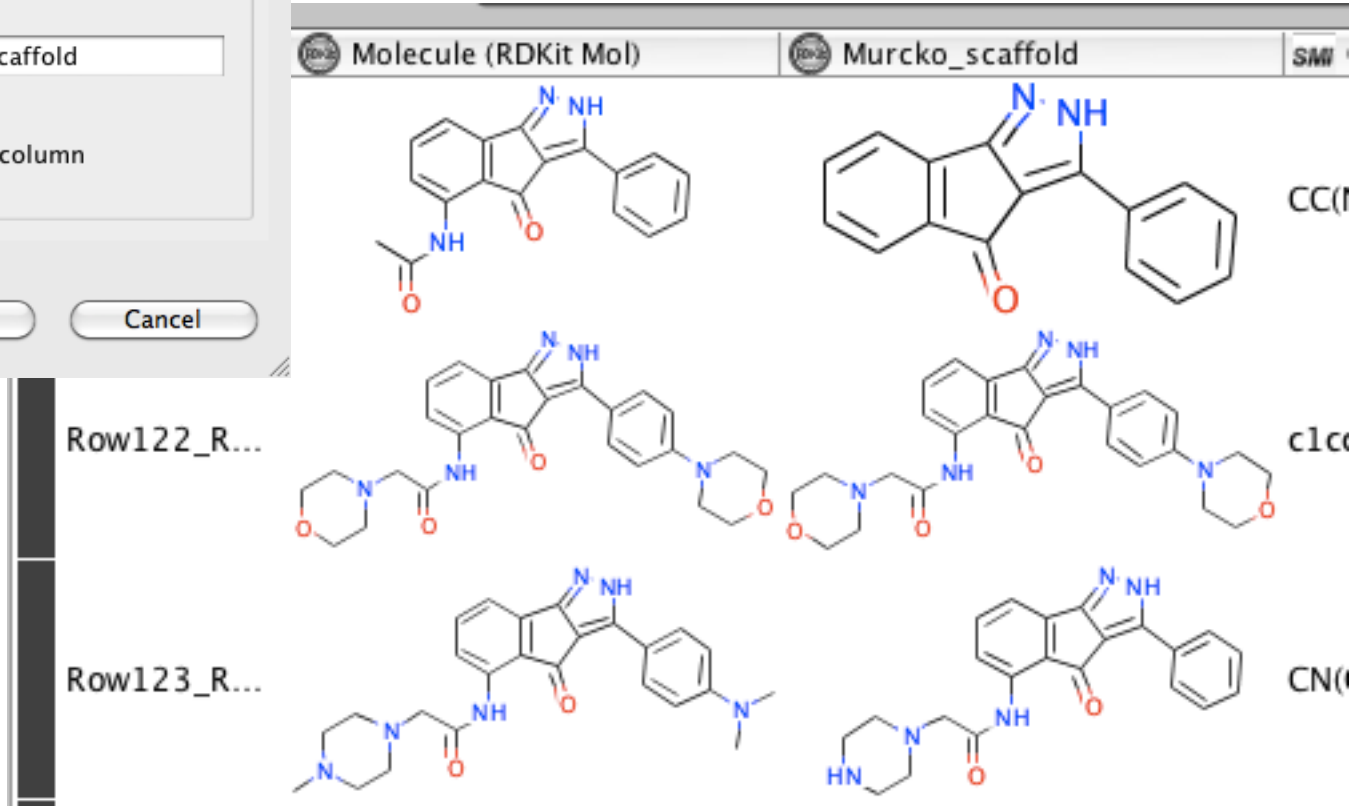




NOVARTIS

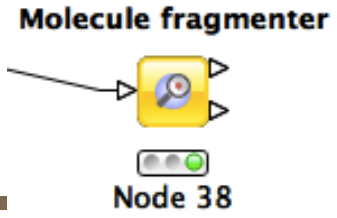
Murcko Scaffolds

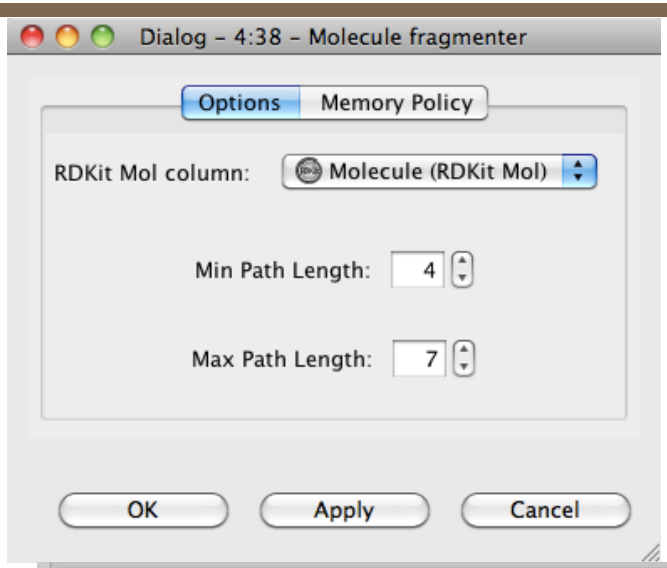




NOVARTIS

Fragmentation



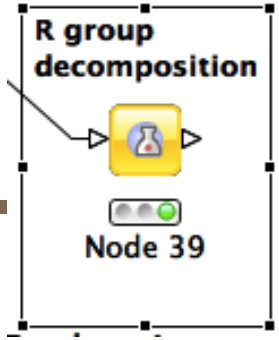


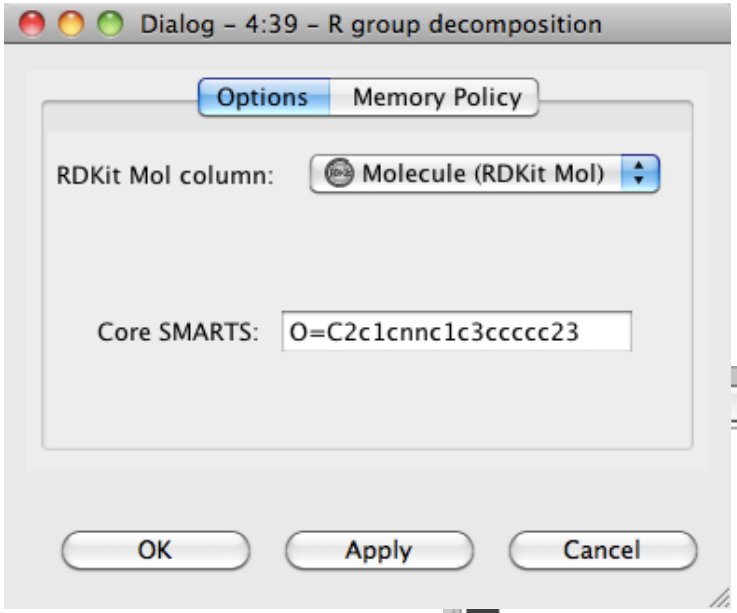
Row ID	Fragm...	Fragment	Fragm...	Fragm...	Count
frag_01	1		cnc(C)c	4	20
frag_11	2		CcncN	4	5
			Ccncs	4	12
			cncc-c	4	20

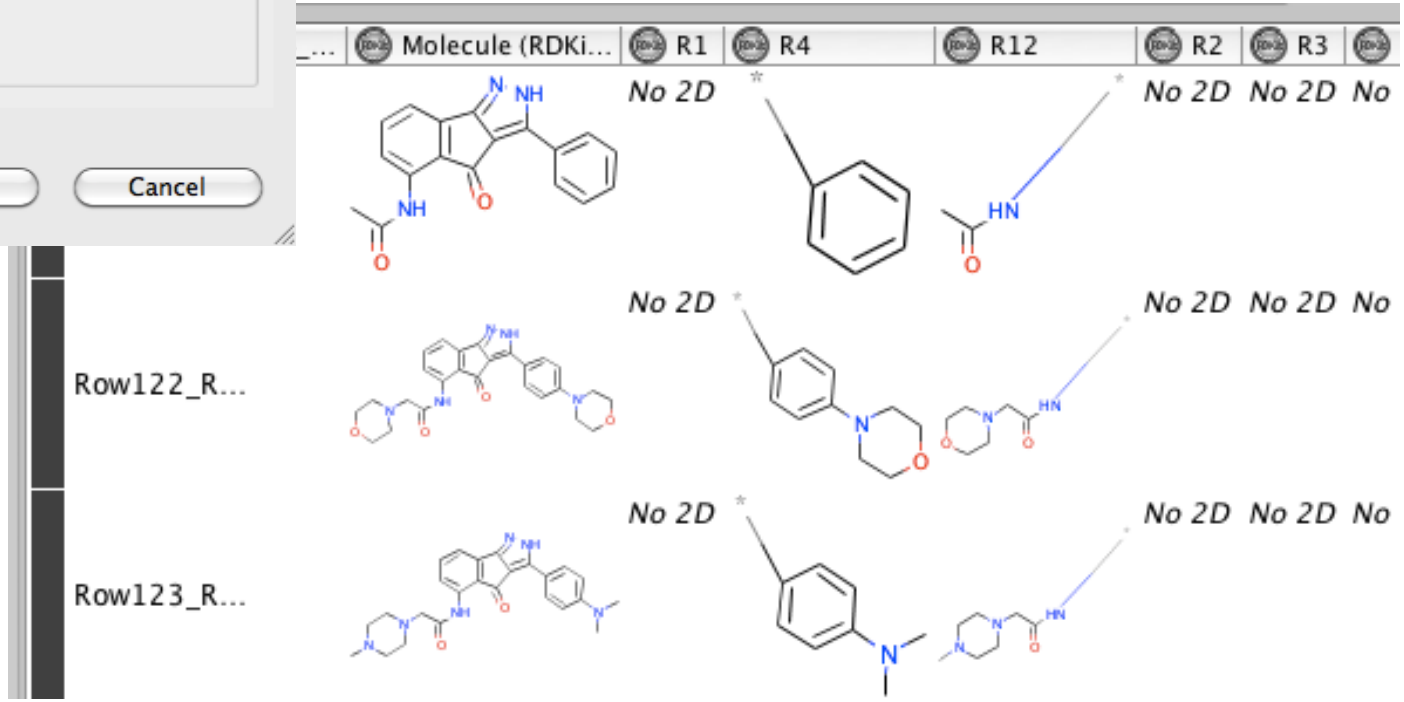
Row ID	ıme	Enzym...	Molecule (RDKit Mol)	Fragment indices
Row1	idine deriv....	n/a		[1,2,3,...]
Row2	idine deriv....	n/a		[1,560,3,...]
Row3	idine deriv....	n/a		[1,2,3,...]

NOVARTIS

R Group decomposition

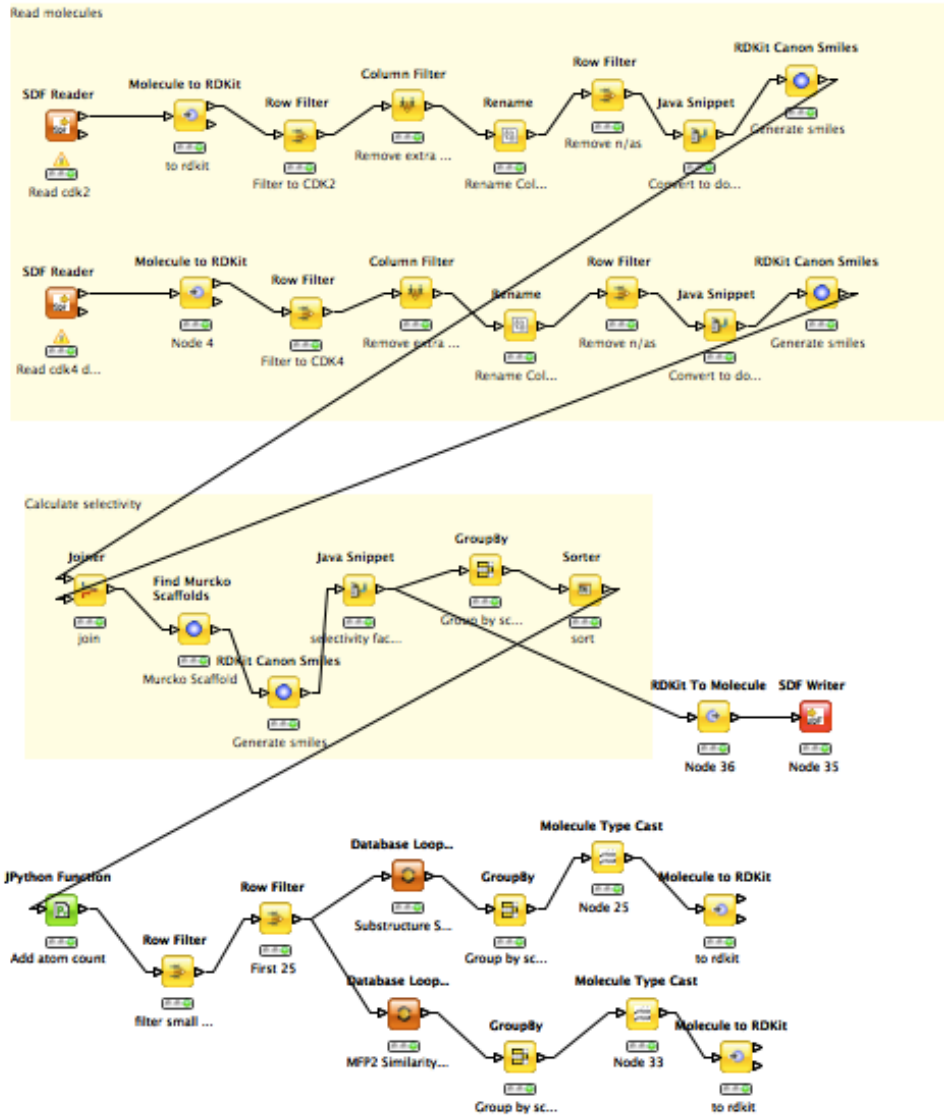






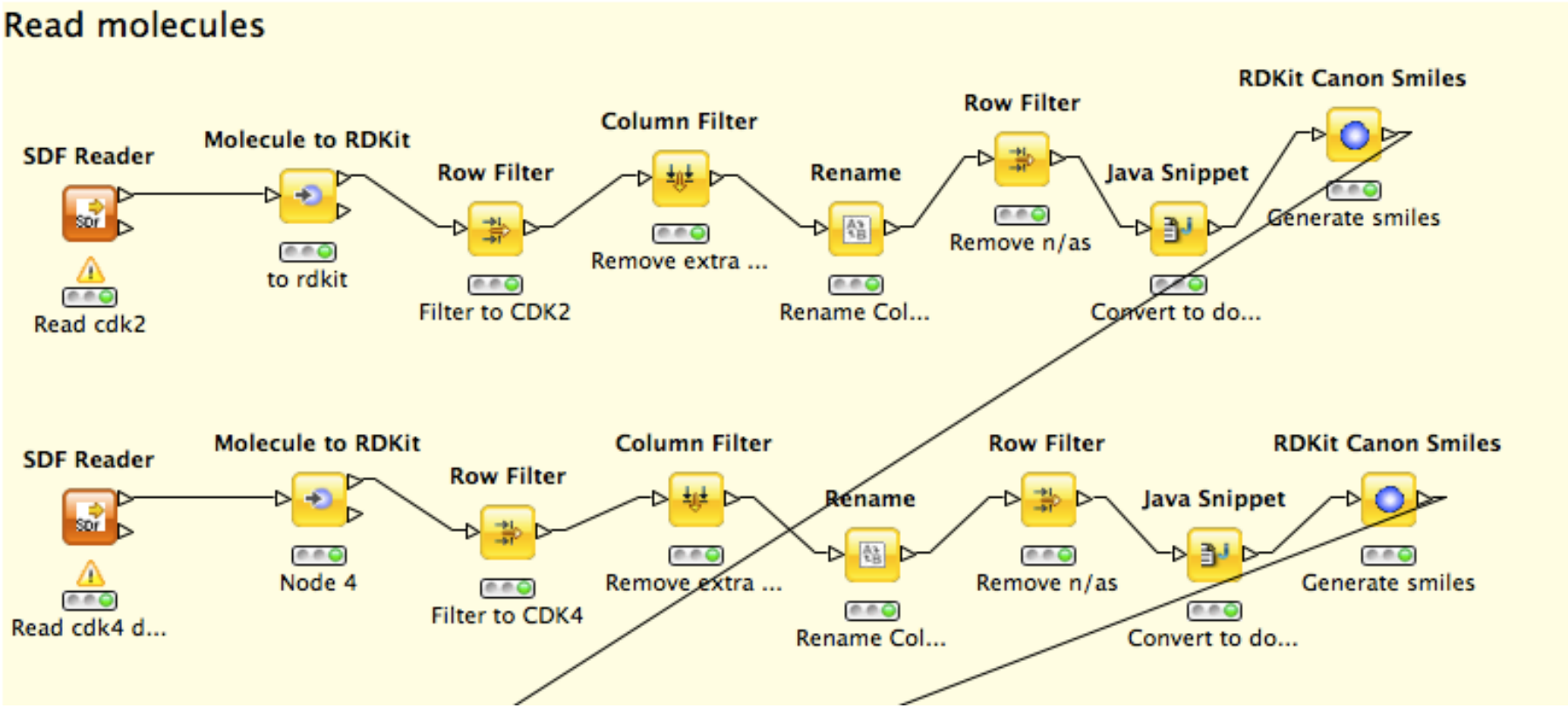
NOVARTIS

More complex example: CDK2/CDK4 selectivity



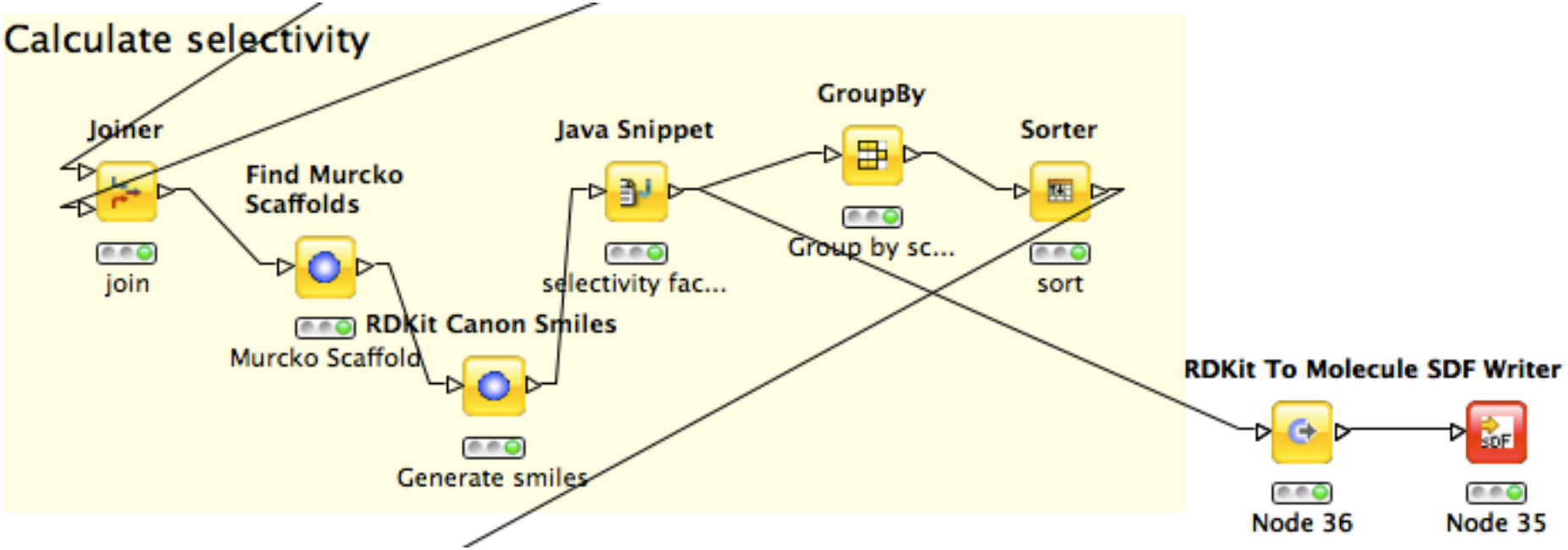
NOVARTIS

More complex example… cont.



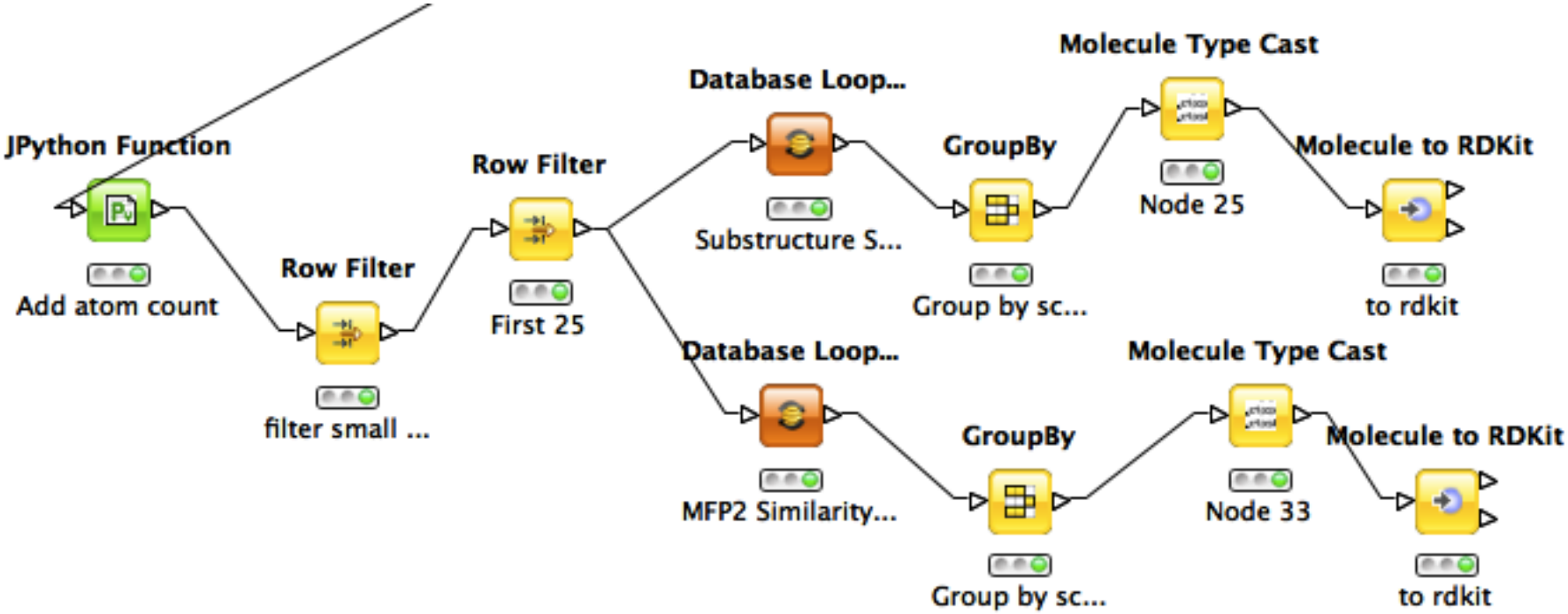
NOVARTIS

More complex example… cont.



NOVARTIS

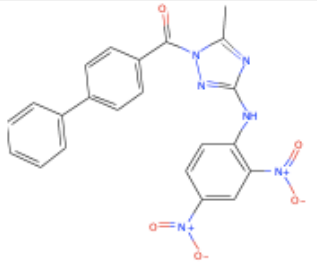
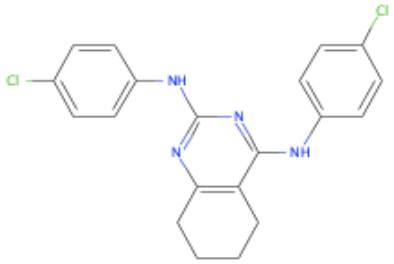
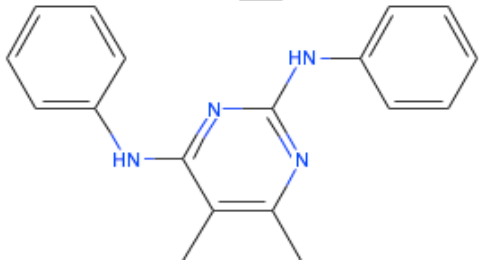
More complex example… cont.



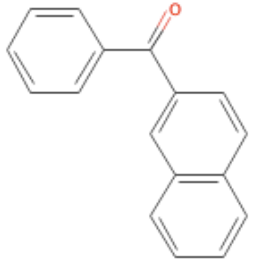
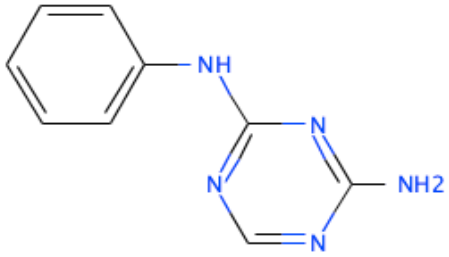
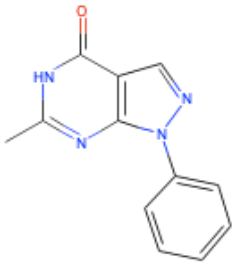
NOVARTIS

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	

NOVARTIS

Overview

- RDKit: what is it?
- RDKit + PostgreSQL
- RDKit + Knime
- **Case study: matched pairs analysis**
- Contributing to open source from big pharma

NOVARTIS

Bringing the pieces together

- Combine the RDKit + the PostgreSQL cartridge + Knime to do matched-pairs analysis
- Idea: find pairs of molecules that are structurally similar but that have quite different activities to identify interesting/useful transformations.
- Key concept is disparity: ΔActivity / (1-similarity)
- Easily done from Python using the RDKit, but it becomes time consuming as the number of molecules increases (N^2 similarity calculations required).

A standard molecular db schema

```
Table "herg_data.mols"
   Column     |  Type    | Modifiers
--------------+---------+-----------
 id           | integer | not null
 compound_id | text     |
 m            | mol      |
Indexes:
    "mols_pkey" PRIMARY KEY, btree (id)
    "molidx" gist (m)
Triggers:
    process_mol AFTER INSERT OR DELETE OR UPDATE ON
herg_data.mols FOR EACH ROW EXECUTE PROCEDURE
herg_data.process_update_mol()
```

```
      Table "herg_data.molvals"
     Column      |  Type    | Modifiers
-----------------+---------+-----------
 id              | integer | not null
 pIC50           | float    |
 ACTIVITY_CLASS | text     |
 CompoundName    | text     |
 MDLPublicKeys   | text     |
Indexes:
    "molvals_pkey" PRIMARY KEY, btree (id)
```

```
 Table "herg_data.countfps"
  Column    |  Type    | Modifiers
-----------+---------+-----------
 id         | integer | not null
 pairfp     | sfp      |
 torsionfp | sfp      |
 mfp2       | sfp      |
Indexes:
    "countfps_pkey" PRIMARY KEY, btree (id)
    "apfpcountidx" gist (pairfp sfp_low_ops)
    "mfpscountidx" gist (mfp2 sfp_low_ops)
    "torsionfpcountidx" gist (torsionfp sfp_low_ops)
```

```
    Table "herg_data.fps"
 Column |  Type    | Modifiers
--------+---------+-----------
 id     | integer | not null
 mfp2   | bfp      |
 ffp2   | bfp      |
Indexes:
    "fps_pkey" PRIMARY KEY, btree (id)
    "ffp2idx" gist (ffp2)
    "mfp2idx" gist (mfp2)
```

```
Table "herg_data.pairbvfps"
  Column    |  Type    | Modifiers
-----------+---------+-----------
 id         | integer | not null
 pairfp     | bfp      |
 torsionfp | bfp      |
Indexes:
    "pairbvfps_pkey" PRIMARY KEY, btree (id)
    "apfpbvidx" gist (pairfp)
    "torsionfpbvidx" gist (torsionfp)
```

Matched pairs in SQL(simple)

```
select *,(pact1-pact2)/dist disparity,pact2-pact1 dact from
 ( select ms1.id id1,ms1.m smiles1,ms2.id id2,ms2.m smiles2,dist,
         -1*log(vs1.ic50_nm*1e-9) pact1,
         -1*log(vs2.ic50_nm*1e-9) pact2
    from ( select fp1.id id1,fp2.id id2,
                  1.0-dice_sml(fp1.torsionfp,fp2.torsionfp) dist
           from cdk2.countfps as fp1
                cross join cdk2.countfps as fp2
           where fp1.torsionfp#fp2.torsionfp and fp1.id!=fp2.id
         ) cliff_pairs
         join cdk2.mols ms1 on (id1=ms1.id)
         join cdk2.mols ms2 on (id2=ms2.id)
         join cdk2.molvals vs1 on (id1=vs1.id)
         join cdk2.molvals vs2 on (id2=vs2.id)
    where dist>0
 ) tmp
 where pact1>=pact2 and (pact1-pact2)>.1
 order by disparity desc
```

NOVARTIS

Matched pairs in SQL(simple)

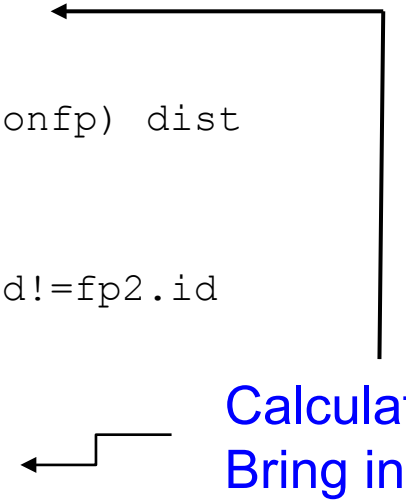
```
select *,(pact1-pact2)/dist disparity,pact2-pact1 dact from
 ( select ms1.id id1,ms1.m smiles1,ms2.id id2,ms2.m smiles2,dist,
         -1*log(vs1.ic50_nm*1e-9) pact1,
         -1*log(vs2.ic50_nm*1e-9) pact2
    from ( select fp1.id id1,fp2.id id2,
                  1.0-dice_sml(fp1.torsionfp,fp2.torsionfp) dist
           from cdk2.countfps as fp1
                cross join cdk2.countfps as fp2
           where fp1.torsionfp#fp2.torsionfp and fp1.id!=fp2.id
         ) cliff_pairs
         join cdk2.mols ms1 on (id1=ms1.id)
         join cdk2.mols ms2 on (id2=ms2.id)
         join cdk2.molvals vs1 on (id1=vs1.id)
         join cdk2.molvals vs2 on (id2=vs2.id)
    where dist>0
 ) tmp
 where pact1>=pact2 and (pact1-pact2)>.1
 order by disparity desc
```

Generate pairs

NOVARTIS

Matched pairs in SQL(simple)

```
select *,(pact1-pact2)/dist disparity,pact2-pact1 dact from
 ( select ms1.id id1,ms1.m smiles1,ms2.id id2,ms2.m smiles2,dist,
         -1*log(vs1.ic50_nm*1e-9) pact1,
         -1*log(vs2.ic50_nm*1e-9) pact2
    from ( select fp1.id id1,fp2.id id2,
                  1.0-dice_sml(fp1.torsionfp,fp2.torsionfp) dist
           from cdk2.countfps as fp1
                cross join cdk2.countfps as fp2
           where fp1.torsionfp#fp2.torsionfp and fp1.id!=fp2.id
         ) cliff_pairs
         join cdk2.mols ms1 on (id1=ms1.id)
         join cdk2.mols ms2 on (id2=ms2.id)
         join cdk2.molvals vs1 on (id1=vs1.id)
         join cdk2.molvals vs2 on (id2=vs2.id)
    where dist>0
 ) tmp
 where pact1>=pact2 and (pact1-pact2)>.1
 order by disparity desc
```



Calculate pIC5
Bring in smiles

NOVARTIS

Matched pairs in SQL(simple)

```
select *,(pact1-pact2)/dist disparity,pact2-pact1 dact from
 ( select ms1.id id1,ms1.m smiles1,ms2.id id2,ms2.m smiles2,dist,
        -1*log(vs1.ic50_nm*1e-9) pact1,
        -1*log(vs2.ic50_nm*1e-9) pact2
   from ( select fp1.id id1,fp2.id id2,
                 1.0-dice_sml(fp1.torsionfp,fp2.torsionfp) dist
          from cdk2.countfps as fp1
               cross join cdk2.countfps as fp2
          where fp1.torsionfp#fp2.torsionfp and fp1.id!=fp2.id
        ) cliff_pairs
        join cdk2.mols ms1 on (id1=ms1.id)
        join cdk2.mols ms2 on (id2=ms2.id)
        join cdk2.molvals vs1 on (id1=vs1.id)
        join cdk2.molvals vs2 on (id2=vs2.id)
   where dist>0
 ) tmp
 where pact1>=pact2 and (pact1-pact2)>.1
 order by disparity desc
```

← Calculate disparity

NOVARTIS

Matched pairs in SQL (complete)

```
select *,(pact1-pact2)/dist disparity,pact2-pact1 dact from
 ( select ms1.id id1,ms1.m smiles1,ms2.id id2,ms2.m smiles2,dist,
         -1*log(vs1.ic50_nm*1e-9) pact1,
         -1*log(vs2.ic50_nm*1e-9) pact2,
         t4v_hash
    from ( select fp1.id id1,fp2.id id2,
           1.0-dice_sml(fp1.torsionfp,fp2.torsionfp) dist,
           md5(subtract(fp1.torsionfp,fp2.torsionfp)::text) t4v_hash
           from cdk2.countfps as fp1
           cross join cdk2.countfps as fp2
           where fp1.torsionfp#fp2.torsionfp and fp1.id!=fp2.id
         ) cliff_pairs
         join cdk2.mols ms1 on (id1=ms1.id)
         join cdk2.mols ms2 on (id2=ms2.id)
         join cdk2.molvals vs1 on (id1=vs1.id)
         join cdk2.molvals vs2 on (id2=vs2.id)
    where dist>0
 ) tmp
 where pact1>=pact2 and (pact1-pact2)>.1
 order by disparity desc
```

Label transformations

NOVARTIS

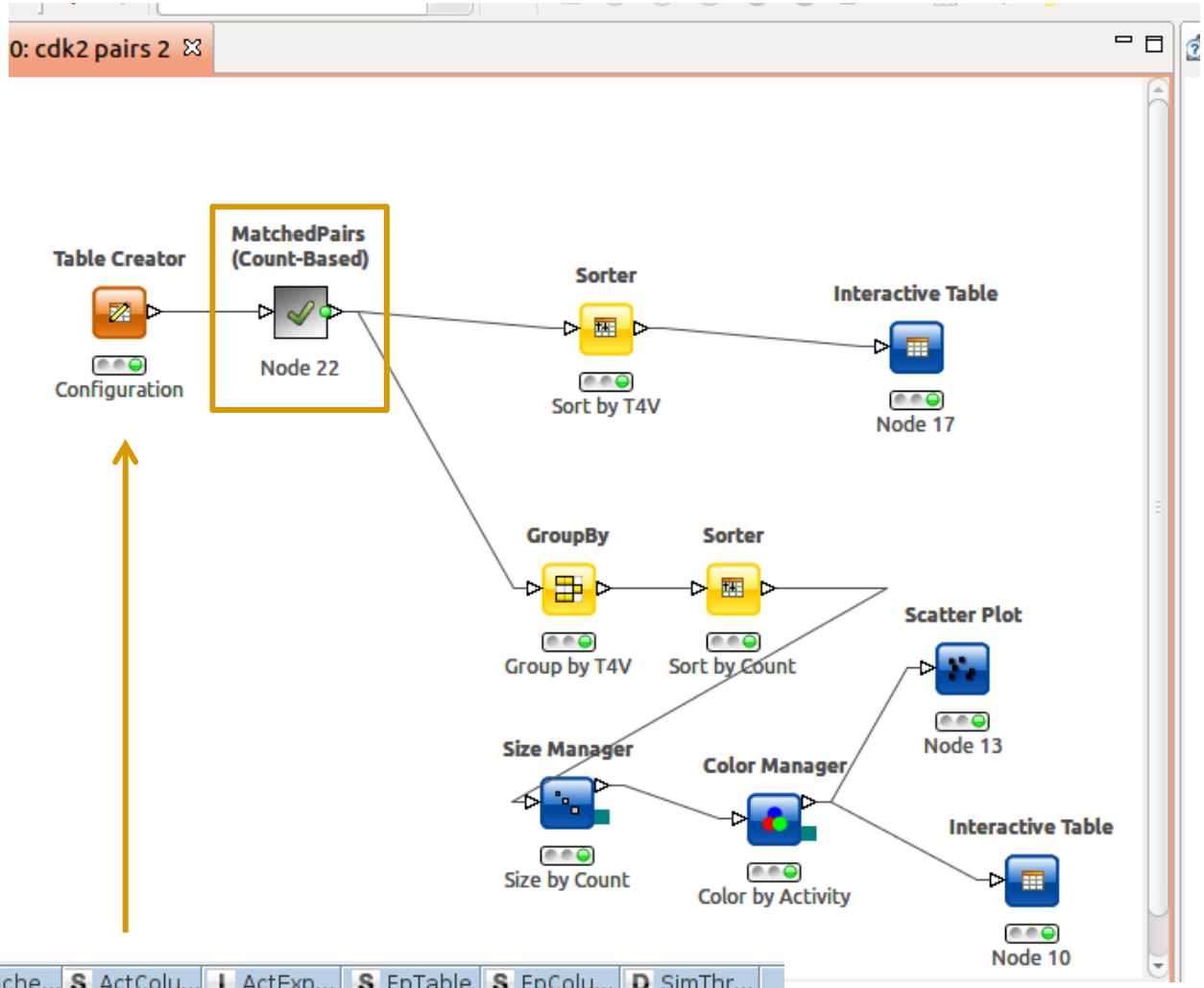
Performance

- Dataset: 1181 molecules with measured CDK2 IC50s

(source: binding db)

- Fingerprints: topological torsions (count-based)
- Counting results:
  - Similarity cutoff 0.90: 1400 pairs, 0.39 sec
  - Similarity cutoff 0.85: 3719 pairs, 0.53 sec
  - Similarity cutoff 0.75: 11541 pairs, 0.85 sec
- Retrieving results:
  - Similarity cutoff 0.90: 1400 pairs, 2.0 sec
  - Similarity cutoff 0.85: 3719 pairs, 4.9 sec
  - Similarity cutoff 0.75: 11541 pairs, 14.1 sec
- Hardware: Dell Studio XPS (i7 870, 64bit)

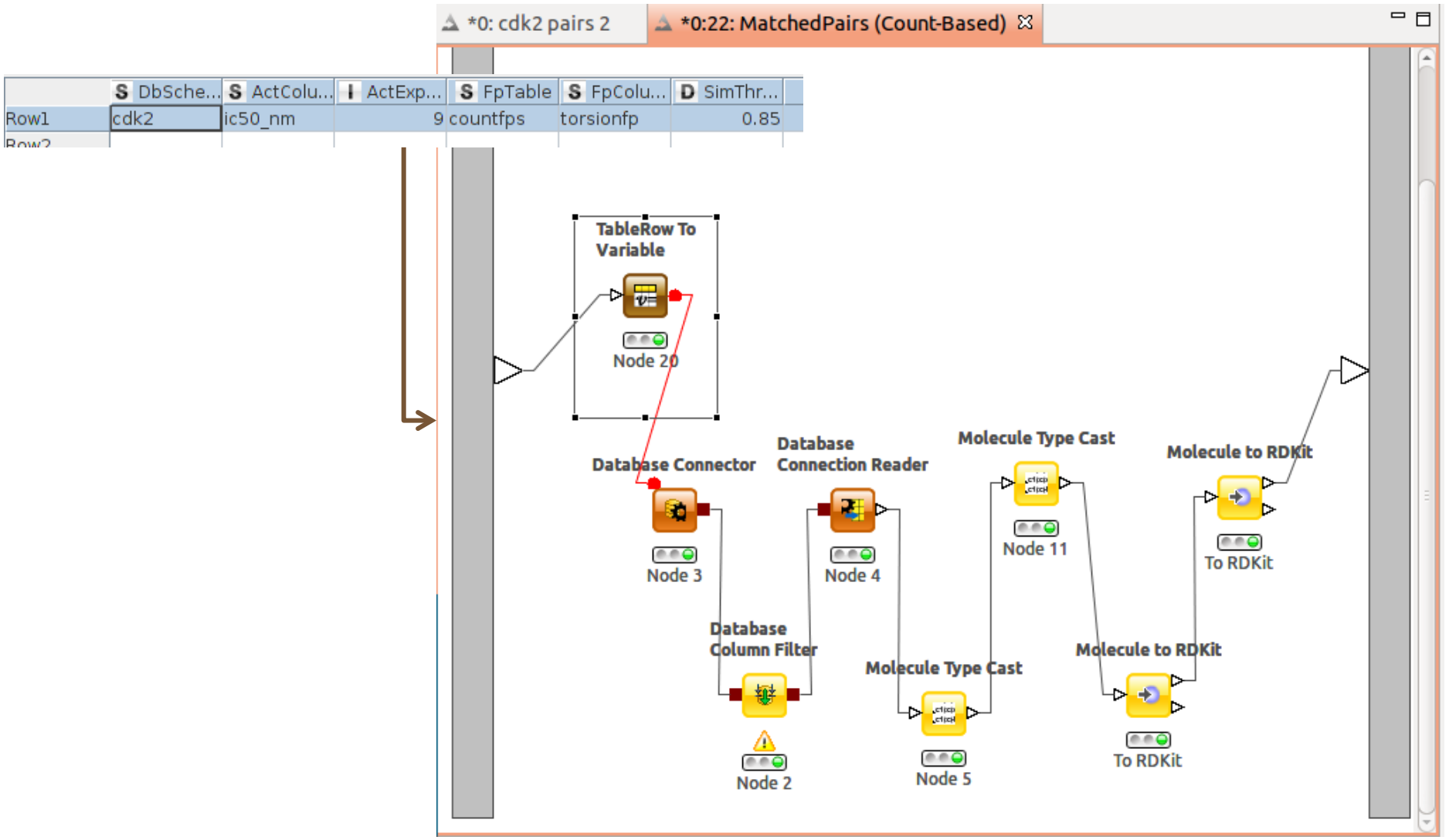
Knime implementation



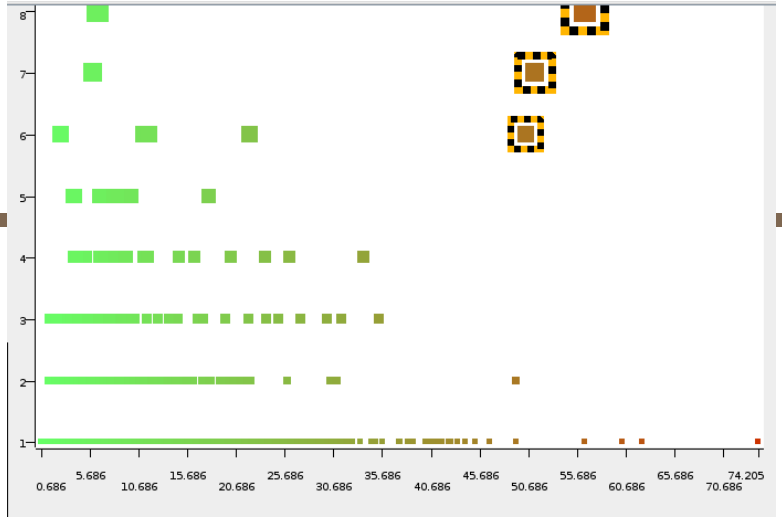
	DbSche...	ActColu...	ActExp...	FpTable	FpColu...	SimThr...
Row1	cdk2	ic50_nm	9	countfps	torsionfp	0.85
Row2						

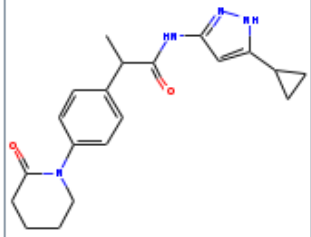
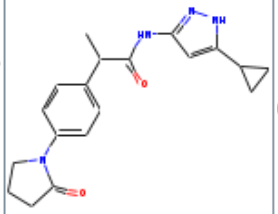
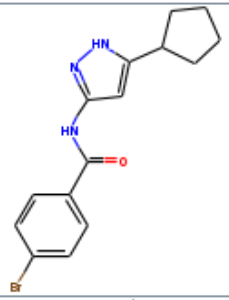
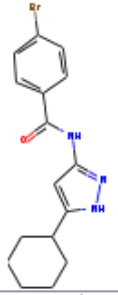
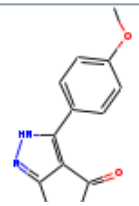
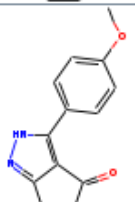
NOVARTIS

Knime implementation

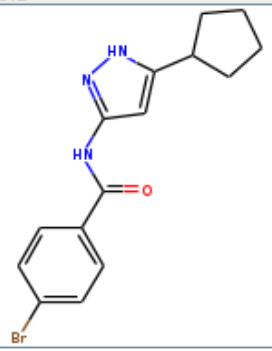
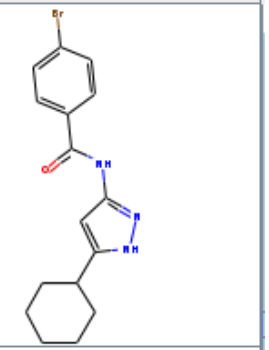
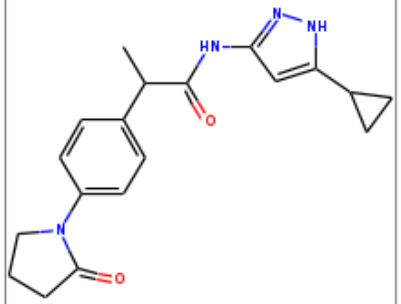
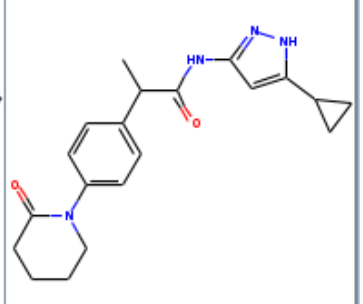
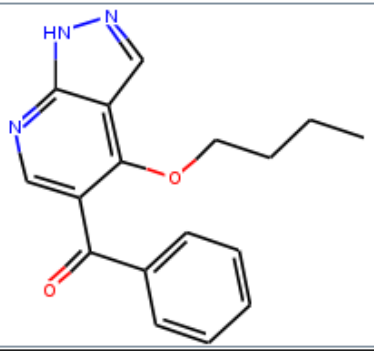
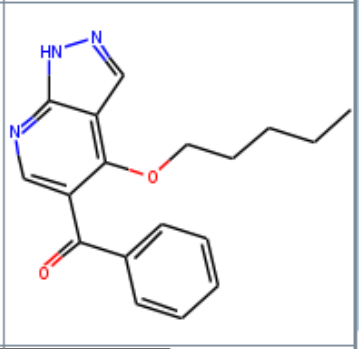


Knime implementation

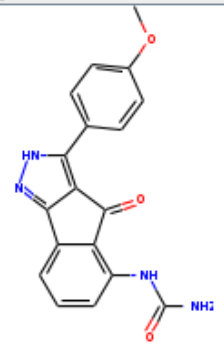
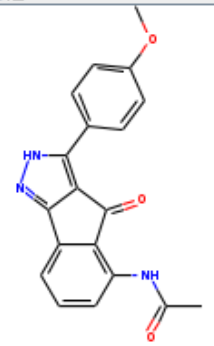
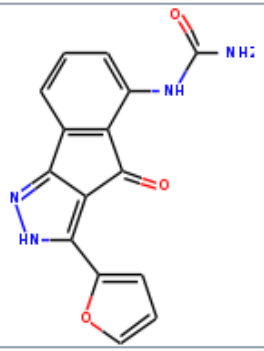
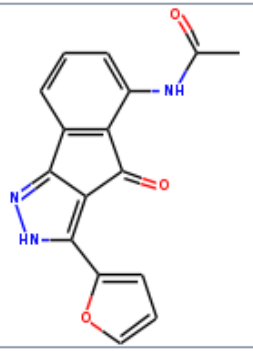
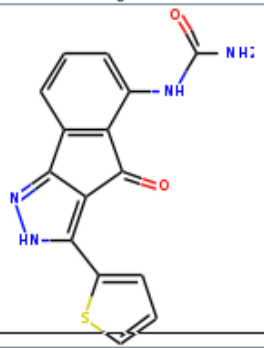
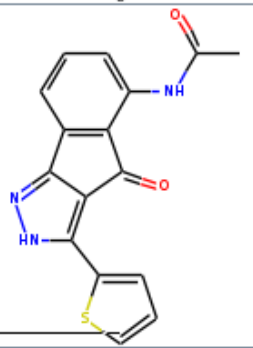


Row ID	t4v hash	Mean(...	First(mol1)	First(mol2)	Count(...	Max(pact1)	Mean(dact)	
Row2649	d3049eaf3c90e2ea202380a780a5b...	50.461			6	7.699	-0.463	0
Row937	4890255b136b311a2078798761e0...	51.404			7	7.824	-0.491	0
Row1523	78e01712a4f11b2e7adf2b33b21d0...	56.469			8	8.155	-1.053	0

Knime implementation

S t4v hash	D disparity	D dact	mol1	mol2
4890255b136b311a2078798761e00...	100.179	-1.301		
4890255b136b311a2078798761e00...	85.261	-0.812		
4890255b136b311a2078798761e00...	30.155	-0.339		

Knime implementation

S t4v hash	D disparity	D dact	mol1	mol2
78e01712a4f11b2e7adf2b33b21d067b	93.59	-1.586		
78e01712a4f11b2e7adf2b33b21d067b	69.981	-1.296		
78e01712a4f11b2e7adf2b33b21d067b	65.55	-1.214		

NOVARTIS

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

Overview

- RDKit: what is it?
- RDKit + PostgreSQL
- RDKit + Knime
- Case study: matched pairs analysis
- **Contributing to open source from big pharma**

NOVARTIS

Contributing to open source: why bother?

- Scientific argument for releasing source:

  ACS ethical guidelines: "A primary research report should contain sufficient detail and reference to public sources of information to permit the author's peers to repeat the work." (http://pubs.acs.org/userimages/ContentEditor/1218054468605/ethics.pdf)

Z. Merali, "ERROR: why scientific programming does not compute." *Nature* **467**:775-7 (2010)
N. Barnes, "Publish your computer code: it is good enough" *Nature* **467**:753 (2010)

Contributing to open source: why bother?

- Philosophy
- Improved code quality : users = testers/peer reviewers
- Gather new ideas/contributions from others
- Altruism : give something back to "the community"
- Selfishness : guarantee your own access to your work

NOVARTIS

Contributing to open source: why bother?

- We aren't the only big company doing this:
  - IBM
  - Apple
  - Google
  - Nokia
  - Microsoft(!)
  - many, many others
- We aren't even the only pharma company:
  - Sunesis
  - Lilly
  - Boehringer Ingelheim
  - Astra Zeneca
  - Sanofi Aventis
  - others

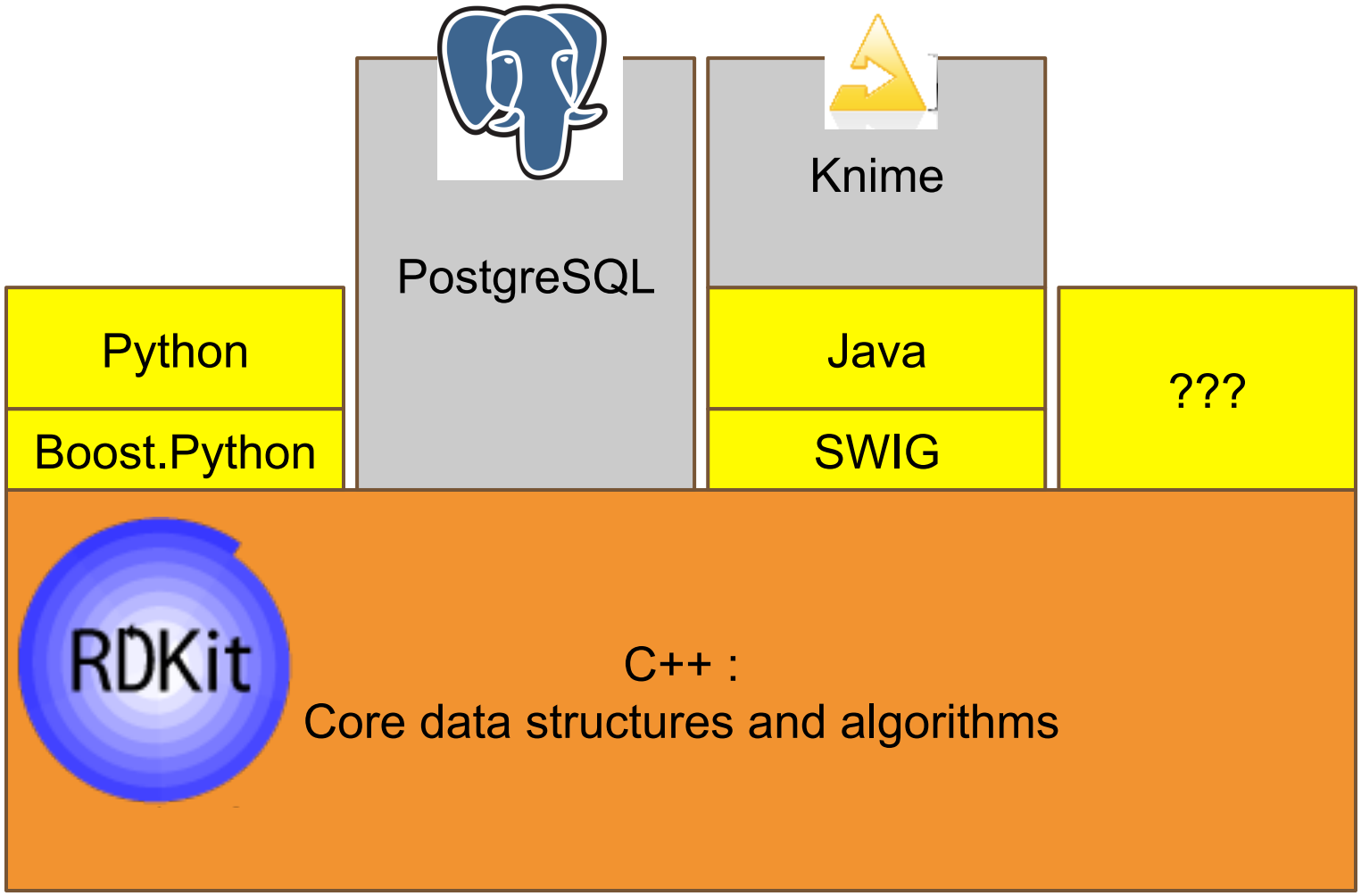
NOVARTIS

Practical Considerations

- Precompetitive
- Treat code publication process the same as publication of a scientific paper
- Pick a license carefully; use a standard one
- Management understanding and support
- Support from legal/patent department

NOVARTIS

Thanks!



NOVARTIS