It doesn't have to be a one-way street: Open-source software for drug discovery from big pharma.

Gregory Landrum, Yiqun (Eddie) Cao, Andrew Palmer
NIBR IT
Novartis Institutes for BioMedical Research
Basel, Cambridge, Emeryville
ACS 2012 Spring Meeting, San Diego
Overview

- Open Source? Why?
- RDKit: what is it?
- RDKit, PostgreSQL, and Knime
- Case study: matched pairs analysis
- Case study: target prediction
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
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)

N. Barnes, "Publish your computer code: it is good enough" Nature 467:753 (2010)

“We argue that, with some exceptions, anything less than the release of source programs is intolerable for results that depend on computation. The vagaries of hardware, software and natural language will always ensure that exact reproducibility remains uncertain, but withholding code increases the chances that efforts to reproduce results will fail.”
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
  - Eli Lilly
  - Boehringer Ingelheim
  - Astra Zeneca
  - Sanofi Aventis
  - others
Practical Considerations in Big Pharma

- Noncompetitive

- 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
Acknowledgements

- **NIBR**
  - Tom Digby (Legal)
  - Richard Lewis (GDC)
  - Nik Stiefl (GDC)
  - Peter Gedeck (GDC)
  - John Davies (CPC)
  - Patrick Warren (NIBR IT)
  - Steve Litster (NIBR IT)
  - Node development
    - Manuel Schwarze (NIBR IT)
    - Dillip Kumar Mohanty (NIBR IT)
    - Sudip Ghosh (NIBR IT)
  - Marc Litherland (NIBR IT)

- **knime.com**
  - Michael Berthold
  - Bernd Wiswedel
  - Thorsten Meinl
  - Peter Ohl

- **PostgreSQL cartridge:**
  - Michael Stonebraker
  - Oleg Bartunov
  - Teodor Sigaev
  - Pavel Velikhov

- **Simon Richards (Lilly)**

- **The RDKit Open-Source Community**
Overview

- Open Source? Why?
- **RDKit: what is it?**
  - RDKit, PostgreSQL, and Knime
  - Case study: matched pairs analysis
  - Case study: target prediction
RDKit: What is it?

- Python (2.x), Java, and C++ toolkit for cheminformatics
  - Core data structures and algorithms in C++
  - 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
What is this all about?

Exact same algorithms/implementations accessible from many different endpoints
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
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 (\(\kappa^3\), 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
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
- Licensing: new BSD
- Documentation:
  - HTML/PDF “Getting Started” documentation
  - 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: Documentation?

The documentation:

Built using Python’s standard docs tool: Sphinx
RDKit: Documentation?

Sample section from introductory docs:

Reading and Writing Molecules

Reading single molecules
The majority of the basic molecular functionality is found in module rdkit.Chem:

```python
>>> from rdkit import Chem
```

Individual molecules can be constructed using a variety of approaches:

```python
>>> m = Chem.MolFromSmiles('Cc1cccc1')
>>> m = Chem.MolFromMolFile('data/input.mol')
>>> stringWithMolData = file('data/input.mol', 'r').read()
>>> m = Chem.MolFromMolBlock(stringWithMolData)
```

All of these functions return a Mol object on success:

```python
>>> m
<rdkit.Chem.rdchem.Mol object at 0x...>
```

Note: docs that include python code snippets are tested.
RDKit: Documentation?

The wiki: http://code.google.com/p/rdkit/w/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:
  - Big pharma
  - Small pharma/biotech
  - Software/Services
  - Academia
- Starting to see contributions coming from the community (wiki pages, code patches, changes to the build system, etc.) as well as active use in other systems.
- Community contributions for packaging:
  - rpms/debs for Fedora/Debian linux
  - homebrew recipe for MacOS
Overview

- Open Source? Why?
- RDKit: what is it?
- RDKit, PostgreSQL, and Knime
- Case study: matched pairs analysis
- Case study: target prediction
The database cartridge

- Integration of RDKit fingerprinting and substructure search functionality with PostgreSQL
- Bit-vector and counts-based fingerprints along with common similarity metrics (Tanimoto and Dice) integrated with PostgreSQL indexing system to allow fast searches
- Available similarity fingerprints:
  - Morgan (ECFP-like), available as bit vectors or counts
  - FeatMorgan (FCFP-like), available as bit vectors or counts
  - RDKit (Daylight-like), available as bit vectors
  - atom pairs, available as bit vectors or counts
  - topological torsions, available as bit vectors or counts
- SMILES- and SMARTS-based substructure querying integrated with indexing system
- Standard Lipinski-like descriptors (logp, tpsa, MW, etc.)
- Part of the RDKit open-source distribution since July 2010
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**:

<table>
<thead>
<tr>
<th>Query Set</th>
<th>SSS</th>
<th>Similarity (0.8)</th>
<th>Similarity (0.6)</th>
<th>Similarity (0.5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zinc Fragments</td>
<td>23.3</td>
<td>14.6</td>
<td>36.4</td>
<td>37.1</td>
</tr>
<tr>
<td>Zinc Leads</td>
<td>8.2</td>
<td>14.8</td>
<td>36.2</td>
<td>38.5</td>
</tr>
</tbody>
</table>
Knime integration

- Out of the box Knime is strong on data processing and mining, weak on chemistry.
- We have developed a set of *open-source* RDKit-based nodes for Knime
- Standard cheminformatics functionality + some nice extras
- Distributed from knime community site
- Binaries available as an update site (no RDKit build/installation required)
- Work in progress: more nodes being added (new wizard makes it easy)

---

1 Work done together with knime.com
Sponsored Knime node development

- Modifications to naïve Bayes nodes to support fingerprints
- Fingerprint naïve Bayes supporting unbalanced datasets
- Database schema browser
- Improvements to python integration
- Improvements to database connector, readers
- Ensemble tree classifier (in progress)
Overview

- Open Source? Why?
- RDKit: what is it?
- RDKit, PostgreSQL, and Knime
- Case study: matched pairs analysis
- Case study: target prediction
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.

- Easily done from Python or Knime using the RDKit, but it becomes time consuming as the number of molecules increases ($N^2$ similarity calculations required).

- The cartridge enables efficient calculation and hashing of the pairs
Atom-Pair and Topological-Torsion Fingerprints

related descriptors from the “distant past”

- **Atom-type:**
  (Element, #heavy neighbors, #pi electrons)

- **Atom Pair**¹:
  Atom-type – topological distance – Atom-type

- **Topological Torsion**²:
  Atom-type – Atom-type – Atom-type – Atom-type

- **Both fingerprints can use counts (not 1/0 values)**

Difference fingerprints

- Subtracting two count-based fingerprints from each other gives a difference fingerprint where the bits describe the different feature counts of the two molecules.

- An example:

```
<table>
<thead>
<tr>
<th>Bit</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>C_1_0-C_3_0-C_2_0-C_2_0</td>
<td>2</td>
</tr>
<tr>
<td>C_3_0-C_2_0-C_2_0-C_1_0</td>
<td>1</td>
</tr>
</tbody>
</table>
```

- Difference fingerprint:

```
<table>
<thead>
<tr>
<th>Bit</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>C_1_0-C_3_0-C_2_0-C_2_0</td>
<td>0</td>
</tr>
<tr>
<td>C_3_0-C_2_0-C_2_0-C_1_0</td>
<td>-1</td>
</tr>
<tr>
<td>C_3_0-C_2_0-C_2_0-O_1_0</td>
<td>1</td>
</tr>
</tbody>
</table>
```

- Pairs that differ by the same structural unit will have the same difference fingerprint.
- This allows us to easily recognize repeated pairs in a dataset.
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
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)
Now make that useable!

- Generate matched pairs for all tyrosine-kinase assays in ChEMBL
  - Only calculate pairs within a particular assay
  - Check if there are pairs that show up repeatedly
Scale

- 133 targets
- 10843 "assays"
- 67846 data points

Limit to assays with at least 50 data points and transformations that occur more than 5 times:
  - 5124 pairs (=2562 unique)
  - 341 transformations (=170 unique)

Runtime: a few minutes
Exploring the results
Exploring the results
Exploring the results

<table>
<thead>
<tr>
<th>target_classification</th>
<th>mol_1</th>
<th>mol_2</th>
<th>standard_units</th>
</tr>
</thead>
<tbody>
<tr>
<td>enzyme kinase protein kinase tyr tk egrf</td>
<td><img src="image1.png" alt="mol_1" /></td>
<td><img src="image2.png" alt="mol_2" /></td>
<td>3,000 nM</td>
</tr>
<tr>
<td>enzyme kinase protein kinase tyr tk egrf</td>
<td><img src="image3.png" alt="mol_1" /></td>
<td><img src="image4.png" alt="mol_2" /></td>
<td>800 nM</td>
</tr>
<tr>
<td>enzyme kinase protein kinase tyr tk egrf</td>
<td><img src="image5.png" alt="mol_1" /></td>
<td><img src="image6.png" alt="mol_2" /></td>
<td>6,000 nM</td>
</tr>
<tr>
<td>enzyme kinase protein kinase tyr tk egrf</td>
<td><img src="image7.png" alt="mol_1" /></td>
<td><img src="image8.png" alt="mol_2" /></td>
<td>3,000 nM</td>
</tr>
<tr>
<td>enzyme kinase protein kinase tyr tk egrf</td>
<td><img src="image9.png" alt="mol_1" /></td>
<td><img src="image10.png" alt="mol_2" /></td>
<td>800 nM</td>
</tr>
<tr>
<td>enzyme kinase protein kinase tyr tk egrf</td>
<td><img src="image11.png" alt="mol_1" /></td>
<td><img src="image12.png" alt="mol_2" /></td>
<td>6,000 nM</td>
</tr>
<tr>
<td>enzyme kinase protein kinase tyr tk egrf</td>
<td><img src="image13.png" alt="mol_1" /></td>
<td><img src="image14.png" alt="mol_2" /></td>
<td>6,000 nM</td>
</tr>
<tr>
<td>enzyme kinase protein kinase tyr tk egrf</td>
<td><img src="image15.png" alt="mol_1" /></td>
<td><img src="image16.png" alt="mol_2" /></td>
<td>850,000 nM</td>
</tr>
</tbody>
</table>
Overview

- Open Source? Why?
- RDKit: what is it?
- RDKit, PostgreSQL, and Knime
- Case study: matched pairs analysis
- Case study: target prediction
Predicting which target a molecule will hit

- **Goal:** build a model to predict which of a set of targets a molecule is most likely to hit
- **Method:** using RDKit atom-pair fingerprints and a new KNIME learner that builds ensembles of truncated decision trees. (sponsored development with knime.com)
- **Data set:** active molecules from 50 different ChEMBL assays

---

Predicting which target a molecule will hit

- 11561 data points, 50 classes
- Atom-pair fingerprints as descriptors
- Ensemble learner: a bag of decision trees
- 50 truncated trees, random descriptor selection
Predicting which target a molecule will hit

- 11561 data points, 50 classes
- 50 truncated trees, random descriptor selection
- out-of-bag prediction error: 5.8%
- mean error from cross validation: 4.2%
Predicting which target a molecule will hit

- mistakes tend to be in families
Drilling into the confusion matrix

<table>
<thead>
<tr>
<th>Row ID</th>
<th>SDF Molecule</th>
<th>S Name</th>
<th>S Target</th>
<th>S Target (Out-of-bag)</th>
<th>D Target...</th>
<th>D phosp...</th>
<th>D 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row48#28...</td>
<td></td>
<td>CHENBL1963... glucocorticoid receptor</td>
<td>progesterone receptor</td>
<td>0.333</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Row51#28...</td>
<td></td>
<td>CHENBL1971... glucocorticoid receptor</td>
<td>progesterone receptor</td>
<td>0.625</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Row55#28...</td>
<td></td>
<td>CHENBL2182... glucocorticoid receptor</td>
<td>progesterone receptor</td>
<td>0.533</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Row57#28...</td>
<td></td>
<td>CHENBL2209... glucocorticoid receptor</td>
<td>progesterone receptor</td>
<td>0.368</td>
<td>0.053</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Row64#28...</td>
<td></td>
<td>CHENBL2214... glucocorticoid receptor</td>
<td>progesterone receptor</td>
<td>0.417</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
Drilling into the confusion matrix

Confusion Matrix - 0:15 - Scorer

Filter to selected elements
- Scorer
  - Node 15
  - HiLite Filter
    - Node 14
    - Tree Ensemble Learner
      - Node 50

Check if there's a ChEMBL assay that matches the predicted value

ChEMBL: get assays for CHEMBL Id
  - Node 38
  - Column Resorter
  - String Manipulation
    - Node 52
    - Node 54
  - Row Filter
    - Node 55
Drilling into the confusion matrix

<table>
<thead>
<tr>
<th>Row ID</th>
<th>app Molecule</th>
<th>Name</th>
<th>Value</th>
<th>Target</th>
<th>Value</th>
<th>Prediction</th>
<th>Db lookup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row48#28...</td>
<td>heatmap</td>
<td>glucocorticoid receptor</td>
<td>progesterone receptor</td>
<td>Heatmap</td>
<td>glucocorticoid receptor</td>
<td>progesterone receptor</td>
<td>210</td>
</tr>
<tr>
<td>Row51#28...</td>
<td>heatmap</td>
<td>glucocorticoid receptor</td>
<td>progesterone receptor</td>
<td>Heatmap</td>
<td>glucocorticoid receptor</td>
<td>progesterone receptor</td>
<td>30</td>
</tr>
<tr>
<td>Row88#28...</td>
<td>heatmap</td>
<td>glucocorticoid receptor</td>
<td>progesterone receptor</td>
<td>Heatmap</td>
<td>glucocorticoid receptor</td>
<td>progesterone receptor</td>
<td>185</td>
</tr>
<tr>
<td>Row221#2...</td>
<td>heatmap</td>
<td>glucocorticoid receptor</td>
<td>progesterone receptor</td>
<td>Heatmap</td>
<td>glucocorticoid receptor</td>
<td>progesterone receptor</td>
<td>32</td>
</tr>
<tr>
<td>Row312#2...</td>
<td>heatmap</td>
<td>glucocorticoid receptor</td>
<td>progesterone receptor</td>
<td>Heatmap</td>
<td>glucocorticoid receptor</td>
<td>progesterone receptor</td>
<td>59</td>
</tr>
</tbody>
</table>
Another target prediction exercise

- **Goal:** build a model to predict which of a set of tyrosine kinases a molecule is most likely to hit

- **Method:** using RDKit fingerprints and a new KNIME learner that builds ensembles of truncated decision trees. (sponsored development with knime.com)

- **Data set:** Active molecules from Tyrosine Kinase assays in ChEMBL
  - active = <100nm IC50
  - active in a single assay
  - all targets with at least 20 actives
  - 3219 active molecules, 12 targets
Predicting which Tyr kinase a molecule will hit

- 3219 data points, 12 classes
Predicting which Tyr kinase a molecule will hit

- 3219 data points, 12 classes
- RDKit fingerprints
- 100 trees
- depth limit: 15
- random descriptor selection
Predicting which Tyr kinase a molecule will hit

- 3219 data points, 12 classes
- Out-of-bag prediction error: 5.8%
- Mean error from cross validation: 5.0%
Wrapping up: the RDKit

- **What is it?**
  - Cheminformatics toolkit useable from C++, Python, Java
  - PostgreSQL cartridge for substructure/similarity searching and descriptors
  - Open-source Knime nodes for cheminformatics

- **Web presence:**
  - Main site: [http://www.rdkit.org](http://www.rdkit.org)
  - Knime nodes: [http://tech.knime.org/community/rdkit](http://tech.knime.org/community/rdkit)
Save the date!

- The first RDKit Users’ Group Meeting is coming this year.

- It will be held at the Institute for Cancer Research in London

- Probable date: September

- Organizer: Nathan Brown

- Contact me to be put on the mailing list.
Thanks!

C++:
Core data structures and algorithms