The Matched Molecular Pair Application
or "KNIME to UNIX and Back Again"

Man-Ling Lee
KNIME Fall Summit - Austin 11/02/2017
Where I work and what I do

- Senior Scientist (Cheminformatics) in Small Molecule Drug Discovery
- Support Drug Discovery project teams:
  - Data management & analysis
  - Application development
Previously in 2013 ...

KNIME and Command Line Tools - The Best of Two Worlds

Man-Ling Lee
KNIME User Group Meeting 03/07/2013
Previously in 2013 ...

Dynamic Command Line Node Configuration

XML file with definition of the command line programs

- Generate one node per `<command>` element
- Deduce ports from `<ports>` element
- During startup of knime/eclipse node set are initialized

A. Gobbi, M. Lee, T. Gabriel, B. Wieswedel
Previously in 2013 ...

Unix Pipe compilation – Past, Present, and Future

Creating this workflow took about one hour

- Pipe command manually copied into a csh script
- In Progress: Automate the insertion
2016 - Finally open sourced at github.com/chemalot

😊 **chemalot**  
A suite of cheminformatics command line tools

😊 **chemalot_knime**  
A KNIME extension to auto-generate command-line KNIME nodes from a configuration file that defines command line programs
The Matched Molecular Pair Application
or “KNIME to UNIX and Back Again”

Man-Ling Lee
KNIME Fall Summit - Austin  11/02/2017
Chemical structures have impact on molecular properties!

Cumming et. al, Nature Reviews Drug Discovery (2013) 12, 948-962
Supervised Matched Molecular Pair analysis

Chemical structures have impact on molecular properties!

Cumming et. al, Nature Reviews Drug Discovery (2013) 12, 948-962

Visualize the effect of different substitutions on molecules properties

![Graph showing the effect of different substitutions on molecules properties.](image-url)
## A Matter of reformatting input SD files

### User Input

<table>
<thead>
<tr>
<th>Structure</th>
<th>Compound ID</th>
<th>cKinSol_prob</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Structure 1" /></td>
<td>CID-01</td>
<td>0.581</td>
</tr>
<tr>
<td><img src="image2.png" alt="Structure 2" /></td>
<td>CID-02</td>
<td>0.334</td>
</tr>
</tbody>
</table>

![Graph](graph.png)

### ???

![Smiles_core](smiles_core.png)
A Matter of reformatting input SD files

The iMatch program creates the columns needed for the iMatch plot

<table>
<thead>
<tr>
<th>Structure</th>
<th>Compound_ID</th>
<th>cKinSol_prob</th>
<th>core_smiles</th>
<th>index_id</th>
<th>index_smiles</th>
<th>reaction</th>
<th>matched_pair</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="cid-01.png" alt="Image" /></td>
<td>CID-01</td>
<td>0.591</td>
<td><img src="image" alt="N" /></td>
<td>CID-01</td>
<td><img src="cn-to-cl.png" alt="image" /></td>
<td>reactant</td>
<td></td>
</tr>
<tr>
<td><img src="cid-02.png" alt="Image" /></td>
<td>CID-02</td>
<td>0.334</td>
<td><img src="image" alt="image" /></td>
<td>CID-01</td>
<td><img src="cn-to-cl.png" alt="image" /></td>
<td>product</td>
<td></td>
</tr>
</tbody>
</table>
Prototyping using chemalot_knime nodes

Genentech Settings

Remote Sdf Reader

Normalize

sdf TagTool

sdf Transformer

Filter rows

sdf TagTool

sdf Normalizer

sdf TagTool

rename SMI=SMI_inMol

input mol to expected mol structure

keep correctly transformed ones

add Smiles_iMatch=GNumber, Matched Pair=no

transformed mol

Matched Pair=yes

Genentech Settings

Rxn files

Create coreInfo.tab

sdf TabMerger

sdf TabMerger

sdf Groovy

sdf TagTool

sdf TabMerger

r1Left.tab by IdentifierCol

r1Right.tab by CTISMILES

remove unpaired molecules

create link_field

coreInfo.tab by link_field & link_col

sdf TagTool

sdf TagTool

Remote SDF Writer

remove columns

pick an assay

output
Prototyping using chemalot_knime nodes

<table>
<thead>
<tr>
<th>Mol Block</th>
<th>Compound_ID</th>
<th>CTISMILES</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CID-07</td>
<td>c1ccc(cc1)C#N</td>
</tr>
<tr>
<td></td>
<td>CID-08</td>
<td>c1ccc(cc1)Cl</td>
</tr>
<tr>
<td></td>
<td>CID-09</td>
<td>c1ccc(cc1)C(=O)O</td>
</tr>
</tbody>
</table>
## Prototyping using chemalot_knime nodes

![Chemical structures and nodes diagram]

<table>
<thead>
<tr>
<th>Mol</th>
<th>Mol Block</th>
<th>Compound ID</th>
<th>CTISMILES</th>
<th>transformedBy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CID-07</td>
<td>c1ccc(cc1)C=#N</td>
<td>CN_to_Cl.rxn</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CID-08</td>
<td>c1ccc(cc1)Cl</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>CID-09</td>
<td>c1ccc(cc1)C(#O)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Genentech Settings
- sdf TagTool
- sdf Transformer
- rename SMI=SMI_inMol
- input mol to expected mol structure

### Remote SDF Reader
- Normalize
- input Molecules

### sdf TabMerger
- sdf Groovy
- sdf TagTool

### sdf TagTool
- remove unpaired molecules
- create link_field

### coreInfo.tab
- by link_field & link_col

### Additional steps:
- remove columns
- pick an assay
- output
## Prototyping using chemalot_knime nodes

### Table:

<table>
<thead>
<tr>
<th>Mol Block</th>
<th>Compound_ID</th>
<th>CTISMILES</th>
<th>transformedBy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>c1ccc(cc1)C#N</td>
<td>CI5_to_Cl.rxn</td>
</tr>
<tr>
<td>CID-07</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CID-08</td>
<td>c1ccc(cc1)Cl</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CID-09</td>
<td>c1ccc(cc1)C(=O)O</td>
<td></td>
</tr>
</tbody>
</table>

### Diagram:

- **Chemical Structure**
  - Carbon (C)
  - Nitrogen (N)
  - Chlorine (Cl)

- **Nodes:**
  - sdf TagTool
  - sdf Transformer
  - sdf TabMerger
  - sdf Groovy
  - sdf TagTool
  - Remote SDF Reader
  - Normalize
  - Genentech Settings
  - Rxn files
  - coreInfo.tab
  - link_field
  - link_col

- **Connections:**
  - Input molecules
  - Transform mol structure
  - Generate transformed molecules
Prototyping using chemalot_knime nodes

- sdf TagTool
- sdf Transformer
- Filter rows
- sdf TagTool
- sdf Normalizer
- sdf TagTool

Genentech Settings

Remote SDF Reader

Normalize

sdf 2Tab

Genentech Settings

Rxn files

Create coreinfo.tab

sdf TabMerger

sdf TabMerger

sdf Groovy

sdf TagTool

sdf TabMerger

r1Left.tab

r1Right.tab

sdf TagTool

sdf TagTool

Remote SDF Writer

r1Left.tab reactant

<table>
<thead>
<tr>
<th>Compound_ID</th>
<th>Smiles_iMatch</th>
<th>iMatch_rxn</th>
<th>Index_iMatch</th>
<th>Matched Pair</th>
</tr>
</thead>
<tbody>
<tr>
<td>CID-07</td>
<td>c1ccc(cc1)C#N</td>
<td>CN_to_Cl.rxn</td>
<td>CID-07</td>
<td>no</td>
</tr>
</tbody>
</table>
Prototyping using chemalot_knime nodes

<table>
<thead>
<tr>
<th>Compound_ID</th>
<th>Smiles_iMatch</th>
<th>iMatch_rxn</th>
<th>Index_iMatch</th>
<th>Matched Pair</th>
</tr>
</thead>
<tbody>
<tr>
<td>CID-07</td>
<td>c1ccc(cc1)C#N</td>
<td>CN_to_Cl.rxn</td>
<td>CID-07</td>
<td>no</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Compound_ID</th>
<th>Smiles_iMatch</th>
<th>iMatch_rxn</th>
<th>Index_iMatch</th>
<th>Matched Pair</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1ccc(cc1)Cl</td>
<td>c1ccc(cc1)C#N</td>
<td>CN_to_Cl.rxn</td>
<td>CID-07</td>
<td>yes</td>
</tr>
</tbody>
</table>
Prototyping using chemalot_knime nodes

<table>
<thead>
<tr>
<th>Mol</th>
<th>Mol Block</th>
<th>S Compound_ID</th>
<th>S CTISMILES</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Nitrile</td>
</tr>
<tr>
<td>phenyl</td>
<td>CID-07</td>
<td>c1ccc(cc1)C#N</td>
<td></td>
</tr>
<tr>
<td>chloro</td>
<td>CID-08</td>
<td>c1ccc(cc1)Cl</td>
<td></td>
</tr>
<tr>
<td>ketone</td>
<td>CID-09</td>
<td>c1ccc(cc1)C(=O)O</td>
<td></td>
</tr>
</tbody>
</table>

**r1Left.tab reactant**

- CID-07: c1ccc(cc1)C#N
- CID-08: c1ccc(cc1)Cl
- CID-09: c1ccc(cc1)C(=O)O

**r1Right.tab product**

- c1ccc(cc1)Cl
- c1ccc(cc1)C#N

**Genentech Settings**

- Create coreInfo.tab

**r1Left.tab**

- sdf TagTool
- sdf Normalizer
- sdf TagTool
- sdf 2Tab
- sdf TabMerger
- r1Left.tab

**r1Right.tab**

- sdf TagTool
- sdf Normalizer
- sdf TagTool
- sdf 2Tab
- sdf TabMerger
- r1Right.tab

**Remote SDF Reader**

- Normalize
- input Molecules

**Genentech Settings**

- Rxn files
- Genentech Settings
- Create coreInfo.tab
## Prototyping using chemalot_knime nodes

### Table

<table>
<thead>
<tr>
<th>Mol Block</th>
<th>Compound_ID</th>
<th>CTISMILES</th>
<th>Smiles_iMatch</th>
<th>iMatch_rxn</th>
<th>Index_iMatch</th>
<th>Matched Pair</th>
</tr>
</thead>
<tbody>
<tr>
<td>reactant</td>
<td>CID-07</td>
<td>c1ccc(cc1)C#N</td>
<td>c1ccc(cc1)C#N</td>
<td>CN_to_Cl.rxn</td>
<td>CID-07</td>
<td>no</td>
</tr>
<tr>
<td>product</td>
<td>CID-08</td>
<td>c1ccc(cc1)Cl</td>
<td>c1ccc(cc1)C#N</td>
<td>CN_to_Cl.rxn</td>
<td>CID-07</td>
<td>yes</td>
</tr>
<tr>
<td>no match</td>
<td>CID-09</td>
<td>c1ccc(cc1)C(=O)O</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

### Diagram

- **Remote SDF Reader**
- **Normalize**
- **sdf TabMerger**
  - r1Left.tab by IdentifierCol
  - c1ccc(cc1)C\#N
- **sdf TabMerger**
  - r1Right.tab by CTISMILES
- **sdf Groovy**
  - remove unpaired molecules
- **sdf TagTool**
  - create link_field
- **Remote SDF Writer**
  - output
Prototyping using chemalot_knime nodes
Prototyping using chemalot_knime nodes

### Table: Chemalot Knime Nodes

<table>
<thead>
<tr>
<th>Mol Block</th>
<th>Compound_ID</th>
<th>Smiles_core</th>
<th>Smiles_iMatch</th>
<th>iMatch_rxn</th>
<th>Index_iMatch</th>
<th>Matched Pair</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CID-07</td>
<td>CC#N</td>
<td>c1ccc(cc1)C#N</td>
<td>CN_to_Cl</td>
<td>CID-07</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td>CID-08</td>
<td>CCl</td>
<td>c1ccc(cc1)C#N</td>
<td>CN_to_Cl</td>
<td>CID-07</td>
<td>yes</td>
</tr>
</tbody>
</table>

### Diagram:

- **Remote SDF Reader**
- **Normalize**
- **Genentech Settings**
- **Prototyping using chemalot_knime nodes**
- **coreInfo.tab by link_field & link_col**
- **sdf Groovy**
- **sdf TagTool**
- **sdf TabMerger**
- **sdf TagTool**
- **sdf TagTool**
- **Remote SDF Writer**
- **sdf Groovy**
- **sdf TagTool**
- **sdf TagTool**
- **Remote SDF Reader**

---

- sdf TagTool
- sdf Transformer
- Filter rows
- sdf TagTool
- sdf Normalizer
- sdf TagTool
Prototyping using chemalot_knime nodes

Command line text can be retrieved for direct Linux execution
Prototyping using chemalot_knime nodes

Command line text can be retrieved for direct Linux execution
sdfIMatch.pl
Retrieve the molecular match pairs as specified in the reaction files
USAGE: sdfIMatch.pl [options] <list of space separated rxn files>

cat #TMP_IN_FILE# \
| sdfMultiplexer.pl -in .sdf -out .sdf -nProc #nProc# \
  -cmd 'sdfTransformer.csh -in .sdf -out .sdf -makeHImplicit -trans #RXN#' \
| sdfGroovy.csh -in .sdf -out .sdf \
  -c 'if( tVal($mol,"transformedBy").length() == 0 ) return false;' \
> #STATUS_TRANS_PREFIX#_#RXN_COUNT#.sdf
Importance of being user friendly
Embedding sdfIMatch in a KNIME workflow

KNIME workflow called from Vortex

1. Process input from KNIME webservice server
2. IF Switch (Variable)
3. Found SDF and the transformations?
   - Correct input
     - Remote SDF Reader
     - Execute sdfIMatch program on the computer cluster
     - Reformatting
     - Remote SDF Writer
   - Incorrect input
     - Return with an error message for the KNIME Webservice server
4. Post processing
   - Get Vortex template
   - Prepare report
Embedding sdfIMatch in a KNIME workflow
Embedding sdfIMatch in a KNIME workflow

Process input

- IF Switch (Variable)
  - from KNIME webservice server
  - Found SDF and the transformations?

Incorrect input

Remote SDF Reader

bsub CommandLine

Reformatting

Remote SDF Writer

Execute sdfIMatch program on the computer cluster

Post processing

Get Vortex template

Prepare report

Output variables - 6:294:1 - Java Edit Variable

<table>
<thead>
<tr>
<th>Flow Variables</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>_appErrorMessage</td>
<td>top</td>
<td></td>
</tr>
<tr>
<td>_If_PortSwitch</td>
<td>com_addlIMatchPair -c 'String s=&quot;yes&quot;; String idVal = tVal($mol,&quot;GNumber&quot;); setVal($mol,&quot;Compound_ID&quot;, idVal&quot;); if(idVal.e...</td>
<td></td>
</tr>
<tr>
<td>_com_chmod</td>
<td>chmod 777 /gne/research/data/smdd/tmp/knime/webservice/manle_lIMatchTest/*</td>
<td></td>
</tr>
<tr>
<td>_com_copyVortexTemplate</td>
<td>cp */gne/research/data/smdd/ApplicationData/Vortex/prd\lIMatch\lIMatch_coreAgainstAssayDisplay_t...</td>
<td></td>
</tr>
<tr>
<td>_com_IMatch</td>
<td>sdfIMatch.pl -in .sdf -out .sdf -id 'GNumber' -debug -jobReport -nProc 5 'test.rxn'</td>
<td></td>
</tr>
<tr>
<td>_com_reorderCols</td>
<td>reorder 'Compound_ID'</td>
<td>MATCH \nreorder_Matched Pair</td>
</tr>
<tr>
<td>_com_sdf2Tab</td>
<td>-tags 'GNumber</td>
<td>Smiles.IMatch' &gt; smiles.tab</td>
</tr>
<tr>
<td>_com_tabMerger</td>
<td>-tab smiles.tab -mergeTag Index.IMatch -mergeCol 'GNumber'</td>
<td></td>
</tr>
</tbody>
</table>
Embedding sdfIMatch in a KNIME workflow

Process input

- from KNIME webservice server

IF Switch (Variable)

- Found SDF and the transformations?
- Incorrect input

Remote SDF Reader

bsub CommandLine

- Execute sdfIMatch program on the computer cluster

Reformatting

Remote SDF Writer

Post processing

Get Vortex template Prepare report

Output variables - 6:294:1 - Java Edit Variable

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>_appErrorMessage</td>
<td>top</td>
</tr>
<tr>
<td>_If_ProtSwitch</td>
<td>String s=&quot;yes&quot;; String idVal = tVal($mol,&quot;GNumber&quot;); setVal($mol,&quot;Compound_ID&quot;, idVal ); if(idVal.e...</td>
</tr>
<tr>
<td>com_addIdMatchPair</td>
<td>chmod 777 /gne/research/data/smd/orders/tmp/knime/webservice/manle__iMatchTest/*; if(idVal.e...</td>
</tr>
<tr>
<td>com_chmod</td>
<td>cp /gne/research/data/smd/orders/ApplicationData/Vortex/nrd/iMatch/iMatch_coreAgainstAssayDisplay_t...</td>
</tr>
<tr>
<td>com_convVortexTemplate</td>
<td>sdfsMatch.pl -in .sdfs -out .sdfs -id 'GNumber' -debug -jobReport -nProc 5 'test.rxn'</td>
</tr>
<tr>
<td>com_iMatch</td>
<td>reorder Compound_ID[]iMatch_rxn_noMatchedPair</td>
</tr>
<tr>
<td>com_reorderCols</td>
<td>tags 'GNumber</td>
</tr>
<tr>
<td>com_sdf2Tab</td>
<td>-tab smiles.tab -mergeTag Index_iMatch -mergeCol 'GNumber'</td>
</tr>
</tbody>
</table>
Embedding sdfIMatch in a KNIME workflow

Execute sdfIMatch program on the computer cluster

Post processing
Get Vortex template
Prepare report

The "string" parameter is controlled by a variable.
Embedding sdfIMatch in a KNIME workflow

Execute sdfIMatch program on the computer cluster

The "string" parameter is controlled by a variable.
Inside the “bsub CommandLine” Wrapped Metanode
Inside the “bsub CommandLine” Wrapped Metanode

You decide which flow variables flow into your Wrapped node!

⇒ Avoid overwriting internal Flow Variables by incoming Flow Variable values
Inside the “bsub CommandLine” Wrapped Metanode

You decide which flow variables flow out of your Wrapped node!

⇒ Avoid overwriting external Flow Variables by outgoing Flow Variable values
Embedding sdfIMatch in a KNIME workflow

KNIME workflow called from Vortex

- Remote SDF Reader
- bsub CommandLine
- Reformatting
- Remote SDF Writer

Flow:
1. Process input from KNIME webservice server
2. IF Switch (Variable)
3. Found SDF and the transformations?
4. Incorrect input
5. Return with an error message for the KNIME Webservice server
6. Execute sdfIMatch program on the computer cluster
7. Post processing
8. Get Vortex template
9. Prepare report

M. Lee      KNIME Fall Summit – Austin 2017
Embedding sdfIMatch in a KNIME workflow

Upon the completion of the output file is loaded into Vortex.
Using the prototype workflow for “debugging”

The workflow helps to identify two potential issues:

- no molecule has been transformed
- no matched molecular pair exists
Prototype

Debug

iMatch

Webservice

Embed

KNIME

Implement

Linux

sdfIMatch.pl

Command Line Program

Summary & Conclusion
Summary & Conclusion

Conclusion

✓ KNIME and Linux are still best friends
Summary & Conclusion

My take on Wrapped and “Classic” Metanodes

- Both type of Metanodes are great for creating reusable workflows

Conclusion

- KNIME and Linux are still best friends
- KNIME is great for prototyping and debugging applications involving a lot of data processing
- Command line programs embedded in KNIME workflows is an easy way to enhance the performance
My take on Wrapped and “Classic” Metanodes

✓ Both type of Metanodes are great for creating reusable workflows

✓ Advantage of Wrapped Metanodes over “Classic” Metanodes
  • Avoid overwriting Flow Variable values by restricting access to incoming Flow Variables
  • Reduce the pollution on Flow Variable stream by restricting release of Flow Variables
  • Behaves almost like a “real” node

Conclusion
✓ KNIME and Linux are still best friends
✓ KNIME is great for prototyping and debugging applications involving a lot of data processing
✓ Command line programs embedded in KNIME workflows is an easy way to enhance the performance
Acknowledgements

Command Line (chemalot_knime) Nodes Development
   Alberto Gobbi
   Thomas Gabriel (KNIME.com)
   Bernd Wieswedel (KNIME.com)

iMatch Development / Revision
   Chandra Goliva
   Kevin P. Clark
   Hans Purkey

Technical Support
   Slaton Lipscomb
   Simran Hansrai
   Elena Kochetkova
   Jim Fitzgerald

Managerial Endorsement
   Jeff Blaney
   Michael Berthold (KNIME.com)
Inside the “bsub CommandLine” Wrapped Metanode