Developing Executable Phenotype Algorithms Using the KNIME Analytics Platform

William Thompson, PhD (Northwestern University)
Huan Mo, MD, MS (Vanderbilt University)
Jennifer Pacheco (Northwestern University)
Robert Carroll, PhD (Vanderbilt University)
Disclosure

• William Thompson is co-founder of Textractor Technologies LLC
Funding

This work has been supported in part by funding from

- PhEMA (R01 GM105688)
- eMERGE (U01 HG006379, U01 HG006378 and U01 HG006388)
- PheWAS (R01 LM010685)
- iPgx (R01 GM103859)
Acknowledgements

• PhEMA members:
  • Jyoti Pathak (Weill Cornell)
  • Joshua Denny (Vanderbilt)
  • Abel Kho (Northwestern)
  • Enid Montague (DePaul)
  • Guoqian Jiang (Mayo Clinic)
  • Luke Rasmussen (Northwestern)
  • Richard Kiefer (Mayo Clinic)
  • Peter Speltz (Vanderbilt)

• PheWAS Members:
  • Pedro Teixeira
  • Wei-Qi Wei
  • Laura Wiley

• eMERGE Members:
  • Ken Borthwick
  • Gerard Tromp
  • Peggy Peissig
Learning Objectives

• Create KNIME workflows, perform data I/O, and data transformation
• Adapt downloaded KNIME workflows for existing phenotype algorithms and adopt them with local data, and create sharable workflows for collaboration
• Have a picture of the variety of KNIME extensions and possible applications
KNIME Overview – What can KNIME do?

• Double click:
  • Examples (guest@http://...)/Please login to access the server
  • Or: Welcome to KNIME/Browse example workflows
Example: TextMining/009005 Gene Term Cooccurrence Heatmap (credit: KNIME group)

https://tech.knime.org/term-cooccurrence-heatmap-example
What have this workflow done?

• Input 1: 5 gene names (RNASE3, ARCN1, IL5, ADAM33, CYSLTR1)
• Downloaded 10,000 abstracts from PubMed that have mentioned each gene
• Input 2: 12 terms (leukocytes, inflammation, edema, etc.) as dictionary.
• **Tag** the terms in the abstracts (documents), and **extract** them out
• **Aggregate** the counts, **pivot** the table
• Make the output table nice
• **Plot** heat map in R
Outputs

<table>
<thead>
<tr>
<th>Row ID</th>
<th>ADAM33</th>
<th>CYSLTR1</th>
<th>IL5</th>
<th>RNASE3</th>
</tr>
</thead>
<tbody>
<tr>
<td>allergy</td>
<td>46</td>
<td>18</td>
<td>67</td>
<td>37</td>
</tr>
<tr>
<td>asthma</td>
<td>134</td>
<td>37</td>
<td>72</td>
<td>45</td>
</tr>
<tr>
<td>atopy</td>
<td>18</td>
<td>4</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>inflammation</td>
<td>27</td>
<td>9</td>
<td>61</td>
<td>36</td>
</tr>
<tr>
<td>sputum</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>13</td>
</tr>
<tr>
<td>bronchitis</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>edema</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>leukocytes</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>
KNIME Toolbox(es)

• **Data I/O:** csv, Excel, database (mysql, postgres, jdbc), etc.
• **Connectivity:** RESTful, PubMed, etc.
• **Data manipulation:** whatever you can do with tables in R
• **Workflow control:** loops, if
• **Special data:** XML, json, NLP documents, PMML
• **Scripting and integration:** Java, R, Python, Groovy (?), etc.
• **Data mining:** classification, Weka, R, PMML, statistics
• If you know what you are doing: social media, NGS, chemistry CDK, Street map, etc.
• **Bottom line:** KNIME is growing!
Basic concepts of KNIME

Huan Mo

Demo workflow: “Hello_Word”
Hello world!

• Create a new workflow: right click KNIME Explorer (Upper Left)/LOCAL (Local Workspace).

• Create a first node: **Table creator** (search it from Node Repository) drag it to the canvas (Node 1)

• Double-click the new node

• Input the following table

• Double Click the **column headers** to set up **column names** and **types**

<table>
<thead>
<tr>
<th></th>
<th>Words</th>
<th>Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row0</td>
<td>Hello</td>
<td>0</td>
</tr>
<tr>
<td>Row1</td>
<td>World</td>
<td>1</td>
</tr>
<tr>
<td>Row2</td>
<td>Hi</td>
<td>2</td>
</tr>
</tbody>
</table>
Run the node!

• Close the configure window
• Right click the node, select “Execute”
• Right click the node, select the last item on the menu
• This table is a data object that flows through the KNIME workflow!
Manipulate the table!

- Search a new node “Row Filter” and drag to canvas (Node 2)
- **Connect** the old node (table creator) to the new node (by dragging from the out-arrow to the in-arrow)
- Configure the “Row Filter”
  - Column to test: Numbers
  - Check “use range checking”, and set “lower bound” to 2
  - Run
- Check the output
Table Manipulation – Common Nodes

• GroupBy: aggregative, projection (as relational algebra)
• Ungroup: Array to rows
• Filtering/splitting: row filter, reference row filter, rule-based
• Pivoting:
• Joiner: natural, cross, left, right joins
Next concept: flow variable

• Search and add node “Table Row to Variable” (Node 3)
• Connect Node 2 (Row filter) out-port to Node 3 in-port
• Execute Node 3, and check result
• Compare the tabs of Node 2 and Node 3 results
Understand Flow variable

• Search and add node “Java Snippet” (Node 4)
• Connect Node 1 out-port (arrow) to Node 4 in-port
• Connect Node 3 out-port (red dot) to Node 4’s Left Upper corner
  • If it fails, right click Node 4, and select “Show Flow Variable Ports”
Java Snippet

• Double click and configure Node 4
  • Put cursor to lines after “Enter your code here:”
  • And click Output/Add
  • Check Append, fill in a name “new words”, type is String
  • Put the cursor to after “out_newwords = “
  • Double click “Column List/Words”
  • Finish your java program, don’t forget to try “Flow Variable List/Words”
  • `out_newwords = c_Words + " " + c_Numbers + " " + v_Words;`

• Run, and check result
Meta Nodes

• Use **Shift** to select both Node 2 and Node 3
• Right click **“Collapse into Meta Node”**
• Double click the new meta node

• Close the meta node tab, and **right click** the meta node
• Add one more **in port** and one more **out port**
• Double click in the meta node again, see what has changed
Data I/O

• Common formats:
  • Csv: CSV Writer, File reader
  • xls(x): XLS Reader, XLS Writer
  • database I/O (MySQL, Postgres, JDBC)

• Native format:
  • I: Table Reader
  • O: Table Writer
  • Benefit: preserve data types (especially data time, documents)
Export your workflow!

• Save the workflow and close all tabs
• Right click the workflow in KNIME Explorer
• Click “Export KNIME Workflow…”
• (Share your phenotype workflow on PheKB.org)

• Remained question: if your workflow requires to import a file, must the recipient of the shared workflow need to reconfigure the file path?
KNIME Path

• Go back to the “Hello_World” workflow, and configure the “XLS Writer” or any other writer

• For file path, use the following instead of an absolute path:

  knime://knime.workflow/hellow_world_out.xls

• Execute the writer

• In your computer, go to your “knime-workspace”, and find your workflow, go under it. Did you see the output file there?
eMERGE type 2 diabetes algorithm

Jennifer Pacheco

Demo workflow: Diabetes-Case-Assignment and Diabetes-Control-Assignment
T2DM: case definition
NOTE: This workflow requires an input comma-separated value (CSV) file with the correctly configured format, i.e., with the right column names and data types. For an example, look at the included sample file. To run this workflow, highlight all nodes, right-click, and select 'reset' (will be grayed out if nodes are already reset). Then execute all nodes, e.g. by using (Shift+F7)
T2DM: KNIME Workflow Sharing
Question: How can I adapt the workflow to my local EHR repository? (Case study with i2b2)

• I2b2 observation_fact table
  • Encounter_num
  • Patient_num
  • Concept_cd
  • Start_date
  • Nval_num

• Input table for T2DM eMERGE algorithm (case)
  • T2dm_dx_cnt
  • T2dm_dx_cnt
  • T2dm_physcn_dx_cnt
  • Max_rndm_gluc_lab_val
  • Max_fast_gluc_lab_val
  • Max_hba1c_lab_val
  • T1dm_rx_dt
  • T2dm_rx_dt
Local adaptation can be challenging!
Another eMERGE Phenotype: AAA

This algorithm also easily shared among sites:

A: either paste in data into Table Creator nodes, or replace w/ any other input node, to retrieve 3 basic types of data in the fmt. in the Table Reader nodes

B: the actual algorithm, which simply runs (no updating needed) & produces a list of patients w/ AAA & their covariates

Natural Language Processing (NLP)

William Thompson
KNIME Text Processing

• KNIME plug-ins for text processing enable:
  • Reading collections of documents in different formats
  • Pre-processing text
  • Tagging text
  • Text analytics and mining

• Additional functionality can be integrated with KNIME using Java snippet or web service nodes
  • Enables integration with cTAKES, Metamap, etc.
  • Output of NLP can be seamlessly used in workflows integrating both structured and unstructured data
KNIME Workflow Example

• Workflow queries PubMed with list of gene names
• Dictionary Tagger used to match relevant terms
• Documents converted to Bag of Words
• R View node is used to create a heat map based on word frequencies.

https://tech.knime.org/term-cooccurrence-heatmap-example
Text IO

- KNIME **text IO nodes** contain parsers for multiple formats, including DML, SDML, PubMed (XML format), PDF, Word, and flat files.
- **Parser nodes** generate a data table consisting of a single column of *DocumentCells*.
- Each *DocumentCell* contains one document, which can be further processed by additional KNIME text processing nodes.
Text Enrichment

• Text enrichment nodes add linguistic and semantic information to text
• Taggers can be used to assign part of speech (POS) categories to words
• Dictionary lookup can be used to do named entity recognition (NER)
Text Transformation

• Transformation nodes turn the Document data into numerical or string data that can be used by other KNIME nodes (e.g., data mining or visualization)
• Documents can be converted to **bag of words**
• Generate **term vectors**
• **Tags assigned** during enrichment process can be converted to strings
Text Preprocessing

Text preprocessing nodes can be used to modify and filter out terms in order to reduce noise including:

• Case converter
• Stemming
• Stop-word filter
• Punctuation erasure
Text Frequencies

• Node for calculating term frequencies
• Inverse document frequency
• N-grams
• Term co-occurrences within specified chunk (sentences, paragraphs, sections)
Text Mining

• Keyword and topic extraction
• At this stage, unstructured data have been converted to **structured data**
• Data extracted from documents can now be fed into other KNIME nodes for **data mining, visualization**, etc.
Custom NLP Integration

- KNIME Java Snippet nodes allow for integration of arbitrary Java code.
- Steps for integrating custom NLP:
  - Create Java NLP software for processing text, returning results (e.g., as JSON result strings)
  - Generate a JAR file containing Java NLP software
  - Add a Java Snippet node to KNIME workflow.
  - Add JAR file as dependencies in the Java snippet node.
  - Call the NLP library from Java snippet node
  - Parse the results (e.g., JSON strings) in subsequent workflow nodes
Custom NLP Integration: Java Snippets
Custom NLP Integration: Java Snippets

```
10// system imports
12 // Your custom imports:
14 import edu.northwestern.fsm.ColonPathReportProcessor;
16 // system variables
25 // Your custom variables:
26
27// expression start
29 // Enter your code here:
31 out_nlp_re = ColonPathReportProcessor.processText(c_path_text);
33
35// expression end
```

Column Filter

Joiner

Input

<table>
<thead>
<tr>
<th>Column / Flow variable</th>
<th>Java Type</th>
<th>Java Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>path_text</td>
<td>String</td>
<td>c_path_text</td>
</tr>
</tbody>
</table>
```java
public synchronized static String[] processText(String text) {
    List<String> retVals = new ArrayList<String>();
    try {
        jcas.reset();
        jcas.setDocumentText(text);
        engine.process(jcas);
        setJCas(jcas);
        select(type: UMLSRelation).each { UMLSRelation relation ->
            RelationArgument arg1 = relation.getArg1();
            RelationArgument arg2 = relation.getArg2();
            IdentifiedAnnotation mention1 = (IdentifiedAnnotation) arg1.mention;
            IdentifiedAnnotation mention2 = (IdentifiedAnnotation) arg2.mention;

            if (arg1.role == ROLE_FINDING && arg2.role == ROLE_LOCATION) {
                UmlsConcept concept1 = mention1.ontologyConcepts.find { it.codingScheme == CODING_SCHEME_SNOMED };
                UmlsConcept concept2 = mention2.ontologyConcepts.find { it.codingScheme == CODING_SCHEME_SNOMED };
                JsonObject jsonObject = new JsonObject();
                jsonObject.addProperty(arg1.getRole(), concept1.getCode());
                jsonObject.addProperty(arg2.getRole(), concept2.getCode());
                jsonObject.addProperty("polarity", mention1.getPolarity());
                retVals.add(jsonObject.toString());
            } else if (arg1.role == ROLE_FINDING && arg2.role == ROLE_DYSPLASIA) {
            }
        } catch (AnalysisEngineProcessException e) {
            e.printStackTrace();
        }
    }
    return retVals.toArray(new String[retVals.size()]);
}
```
Custom NLP Integration: Java Snippets
R, PheWAS, and Data Mining

Robert Carroll
R integration with KNIME

• KNIME enables users to pass data into and out of R.
• Very flexible; KNIME has many packages, but so does R!
• Ask yourself a few questions:
  • Why am I using KNIME?
  • Why am I using R?
• We are going to use phenome wide association studies (PheWAS) as our case study.
Phenome Wide Association Study

• https://phewas.mc.vanderbilt.edu/
• PheWAS originated as a way to “flip” the genome wide association study paradigm.
• GWAS investigate hundreds of thousands (or more) single nucleotide polymorphisms (SNPs) for a disease or phenotype of interest.
• PheWAS uses the rich clinical data available in electronic health records (EHRs) to broadly survey diseases or phenotypes.
• GWAS are fairly common, and there are well documented software available.
• PheWAS are less common due to the intense data requirements, but there is an R package.
Our scenario:

• We are working with a group that has an EHR connected to a biobank with genotype data available.
• We would like perform a PheWAS using these data on a SNP in the gene *FTO*.
• This gene is known to be associated with type 2 diabetes and obesity, but it is unclear what other associations there may be, or if body mass index may drive some of these associations.
The real study and our version

• This study was published by Cronin *et al.* Front Genet 2014; 5: 250. [http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4134007/](http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4134007/)

• We are using a simulated data set that will replicate some of these findings!
What do we need?

**Inputs**
- ICD9 billing code data
- Demographic information
- Genetic data

**Outputs**
- Adjusted regression association measures for 1600 phenotypes
- Visualization of results
PheWAS: Phenome-Wide Association Study

- EHR
- ICD9 Codes
- PheWAS Code Translation
- Phenotypes
- Genotypes
- Demographics
- Association Testing
- Results
- Plotting

Biobank
R PheWAS Package

- EHR
  - ICD9 Codes
    - PheWAS Code Translation
  - Demographics
  - R PheWAS Package
- Biobank
  - Genotypes
  - Results
  - Plotting
What now?
Example 1: R PheWAS in KNIME
Example 1 Debrief

Problems:
1. We can’t tell where data is coming from.
2. We can’t make changes without resetting everything.
3. We might as well just run this in R.
Example 2: Back to Basics

Let’s start again using some design decisions:

1. Use KNIME data import where possible
2. Break up R calls as possible to run through KNIME
3. Follow the roadmap we have for PheWAS
Data Import
Loading RData files
Filtering plink raw files
R Snippet: Translate PheWAS codes
Merge our data together
Load the data into R
R source (workspace)

```r
# Set the working directory to the workspace
setwd(knime.flow.in[["knime.workspace"]])
# Load the PheWAS package:
library(PheWAS)
```
Add Table to R

```r
phenotype_names = names(knime.in)[-1]
rm(knime.in)
```
Perform the analysis
# Run our PheWAS
results = phewas(phenotypes=phenotype_names,
genotypes=c("rs8050136_A"),
covariates=c("age", "is.male"),
data=data,
cores=2)
R to table

• Allows us to print our results to a KNIME table for other uses.
• In our case, we just save the table.
• As with the other nodes, there is a simple template: Set knime.out to the data.frame you wish to export
R View (Workspace)

- This node allows us to generate a plot from our R workspace.
- The R View (Table) node allows us to do it from a KNIME table as well.
- `phewasManhattan(results, title='rs8050136_A', annotate.angle=0)`
- Note that the “PNG Settings” tab can change how your plot looks!
XML, RESTful and Value Set Authority Center (VSAC), RxNORM API

Huan Mo

Demo workflow: “XML_RESTful_VSAC”
RESTful API instructions

• Value Set Authority Center (VSAC) API
  • Step 1: **POST** UMLS account name/password to obtain Ticket Granting Ticket (TGT)
  • Step 2: **POST** TGT and Service to obtain ticket
  • Step 3: **GET** value set XML with OID and ticket

• RxNORM API

• If your RESTful connection fails despite...
  • Check **Header Settings**
  • Accept: text/xml (sometimes text/plain)
XML in KNIME

- To XML
- **XPath** (To parse VSAC/RxNORM API responses)
- XSLT
- XML Column/Row Combiner
- Use Java Snippet to do more