This document is served as the manual to build a new version of FI network

**Overview of Workflow**

The workflow to build a functional interaction (FI) network as described in Figure 1. Some changes may be made version by version. So the following Figure may be updated periodically.

Macintosh HD:Users:gwu:Documents:EclipseWorkspace:caBigR3:doc:FIConstructionWorkFlow.pdf

*Figure 1. Overall workflow to build the Reactome FI network.*

There are two types of data sources are used: Data sources for predicted FIs and data sources for annotated FIs. The data sources for predicted FIs should be downloaded from protein-protein interaction databases: IntAct, HPRD, and BioGrid, pFam (for domain interactions), and from various journals (see below for details). The data sources for annotated FIs are several human curated pathway databases.

**Data Sources for Annotated FIs**

Annotated FIs are FIs extracted from human curated pathways. Multiple pathway databases have been used. Pathways in other non-Reactome databases are converted to curator tool project first, and then dumped into a modified gk\_central Reactome databases. Different file formats are used for different databases.

**Reactome:**

Reactome is used as the foundation for the FI network build. Public released Reactome database have augmented predicted objects from other non-human species. To minimize the effects of these predicted non-human objects, we use a snap-shot of gk\_central, though many non-human pathways have been dumped into gk\_central now (e.g. many objects from plant Cyc).

1. Generate a snapshot for gk\_central @reactomedev.oicr.on.ca: Log into reactomedev, and generate a gk\_central dump by using the following command:

*mysqldump –u{user\_name} –p gk\_central >gk\_central\_{mmddyyyy}.sql*

You will be asked to enter your mysql password. Replace text in {} by correct values.

If you want to do build at your local computer, zip the generate sql dump as following:

*jar –Mcvf gk\_central\_{mmddyyyy}.sql.zip gk\_central\_{mmddyyyy}*

1. Create a snapshot of gk\_central: Copy the above zipped dump file into your build computer (e.g. your desktop machine), unzip it using the following command:

*jar –xvf gk\_central\_{mmddyyyy}.sql.zip*

Log into your mysql database using an user account that can create a database. Create a database as following:

*mysql> create database reactome\_{release\_number}\_plus\_i;*

Here release\_number should be replaced by actual release number (e.g. 39). The release number should be the number of closest public release. For example, we build a FI network in January, 2012. The closest Reactome public release is 39, which was released in Decemeber, 2011. Note: Reactome has a quarterly release schedule.

After the empty database is created, run the following two commands to load the generated gk\_central snapshot mysqldump:

*mysql> use reactome\_{release\_number}\_plus\_i;*

*mysql> source gk\_central\_{mmddyyyy}.sql*

Note: In order to make sure mysql can find the dump file, please start mysql from the directory containing the unzipped mysqldump file. Or you have to use absolute path to your dump file (no quotation marks)

1. Modify Reactome database schema: the regular Reactome database schema needs to be expanded for the FI network. A new attribute called “dataSource” should be added to the top-level class, “DatabaseObject”. A new class called “Interaction” should be created. Interaction is a concrete sub-class to Event. (TO\_DO: Modify the final database schema for Perl API: need a new Protégé project)

*mysql> source {absolute\_path\_to /SchemaModification.sql*

Note: You should find a copy of SchemaModifcation.sql in resources folder in project caBigR3 in CVS.

After a successful run of the above sql, check if a new Interaction class, and new attribute dataSource for DatabaseObject have been created by connecting your curator tool to your newly created database. Check the database schema view to make sure these two things have been there.

In order to use this new expanded Reactome schema for future procedures, we need to export the new schema from the database into a local computer. In the curator tool, after connecting to the modified database, do the following:

In the database schema view, choose “Export Schema” in the File menu, and save the schema into the resource folder in your caBigR3 project folder or any working folder that contains the resource folder. This schema file will be needed by the Reactome Java API, which is used extensively in the following procedures.

Note: the file name has to be “schema” without any extension. Don’t change it!

**UniProt**

UniProt is used as the protein reference database in the FI network construction. We need the downloaded the latest version of UniProt so other data sets can be mapped to it. The download site for UniProt is: [ftp.uniprot.org](ftp://ftp.uniprot.org). You don’t need any registration. We only need to human data sets.

1. After log into [ftp.uniprot.org](ftp://ftp.uniprot.org), download two files from knowledgebase/taxonomic\_divisions/: uniprot\_sprot\_human.data.gz and uniprot\_trembl\_huma.dat.gz. You may also download reldate.txt or README files for future reference. After download these files, unzip them, and place them into a directory. Change the constant value in R3Constants in package org.reactome.r3.util to point the directory, e.g.:

**public** **static** **final** String *UNIPROT\_DIR* = *DATA\_SET\_DIR* + "UniProt/release\_2011\_12/";

1. You need to run a method UniProtAnalyzer.generateUniProtIDsMap() in package, org.reactome.r3, to create two mapping files that are used to normalize any UniProt accession number usages. For example, some pathway databases use UniProt access numbers that have been merged to other numbers. These two files are used for normalizing such usages. Two files will be generated: one for SwissProt only (SwissProtACIDMap.txt) and another for both SwissProt and Trembl (ACIDMap.txt). These two generated files should be placed in the same directory as original data files.

Note: Method generateUniProtIDs() is a JUnit3 method! The method should run very fast (around several seconds).

1. You also need to download the fasta file for Isoform sequences to be used in normalizing features for NBC training and predictions from this web site: <http://www.uniprot.org/downloads>

**IPRO\_CLASS**

A mapping file is needed to map entrez gene id to UniProt accession number. We need to download an original mapping file from the PIR web site, and do some pre-process using a Java method.

1. Download the mapping file: [ftp.pir.georgetown.edu/databases/](ftp://ftp.pir.georgetown.edu/databases/). Go to idmapping/mapping\_by\_sp, and download file h\_sapiens.tb. Remember where you place your file.
2. Modify the following constant in R3Constant so that it can point to correct file, which will be generated in Step 3:

// This file is used to map Entrez id to UniProt accession number

**public** **static** **final** String *ENTREZ\_TO\_UNIPROT\_MAP\_FILE\_NAME* = *DATA\_SET\_DIR* + "iproclass/011612/EntrezToUniProt.txt";

1. Pre-process the mapping file: In class org.reactome.r3.UniProtAnalyzer, there is a method generateEntrezGeneToUniProt(). Make sure the inputFile value points to your downloaded file in step 1.

**KEGG**

KEGG has many good pathway diagrams and many disease pathways. We import KEGG pathways using its KGML export. The KEGG KGML is downloaded from KEGG ftp folder. You need to have a license first in order to access its ftp site.

1. Download the following files from the KEGG ftp site: [ftp.bioinformatics.jp](ftp://ftp.bioinformatics.jp)
   1. kegg/xml/kgml/non-metabolic/organisms/hsa.tar.gz: we used human non-metabolic pathways only. Reactome has enough metabolic pathways.
   2. kegg/pathway/pathway.list: the whole list of pathways. From this list, we can choose what pathways we want to use. This file is optional.
   3. kegg/genes/links/genes\_uniprot.list.gz: a mapping file from kegg ids to UniProt ids. After unzipping, this file is around 140 Mb (January 12, 2012). You can use the following command to get a human only mapping file for easy view and quick processing (Mac or Linux only):

*grep hsa: genes\_uniprot.list > hsa\_genes\_uniprot.list*

The output file hsa\_genes\_uniprot.list is around 540 K.

1. Convert KGML files into a Reactome curator tool project:

In the caBigR3 project, there is a package called “org.reactome.kegg”. All KEGG related process Java classes have been grouped in this package. Class “KeggToReactomeConverter” in this package is used to convert KGML file into a reactome curator tool project (.rtpj file). To use classes in this package, please specify some constants in the org.reactome.r3.util.R3Constants class:

**public** **static** **final** String *KEGG\_DIR* = *DATA\_SET\_DIR* + "KEGG/011112/";

// Unzipped human KGML files should be in this directory

**public** **static** **final** String *HSA\_KGML\_DIR* = *KEGG\_DIR* + "KGML/hsa/";

// The converted KEGG pathways is saved in this project file

**public** **static** **final** String *CONVERTED\_KEGG\_FILE* = *KEGG\_DIR* + "011312.rtpj";

// This file is used to map KEGG gene ids to UniProt ids

**public** **static** **final** String *KEGG\_ID\_TO\_UNIPROT\_MAP\_FILE* = *KEGG\_DIR* + "hsa\_genes\_uniprot.list";

Note: change the above constants as appropriate values in your local environment.

The actual method that is used to do KEGG converting is runBatchConvert() in class KeggToReactomeConverter. Besides the above variables, you also need to make a change to dbAdaptor in this method, so that it can point to your local “reactome\_{release\_number}\_plus\_i”.

Now run method runBatchConvert(). The method is a JUnit method, run it using Run as JUnit Test after selecting it in the outline.

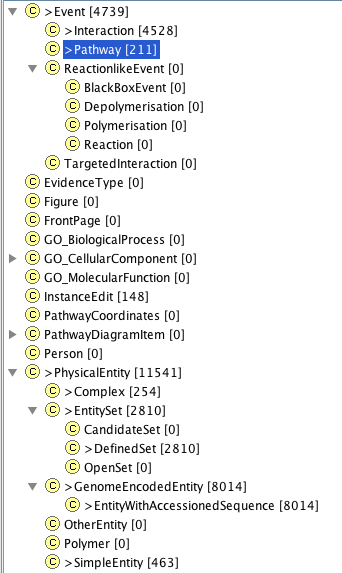
Notes: 1). Make sure to assign enough memory for running this method (e.g. –Xmx1024m).

2). On the KEGG download in January, 2012, there are 26 KEGG ids cannot be mapped to UniProts. Check to ID mapping in UniProt, find three mappings. These unmapped KEGG ids are left as there since the number is very small. The unmapped ids are listed in file UnMappedKeggIds.txt.

3). In the version of FI network constructed in 2009, one KEGG id in pathway has been mapped to one UniProt only. This has been changed in the code now. A KEGG id can be mapped to multiple UniProt via DefinedSet. So a DefinedSet converted may contain another DefinedSet.

4). This converting should run pretty fast (< 1 min for sure).

5). Here are some numbers in the 2012 version may be useful:



Note: 1). ReferenceIsoform instances may have been fetched out from the database during converting. However, our converting cannot take use of isoforms yet. So the Isoform instances should be treated as the top-level ReferenceGeneProduct instance!

2). We use KGML to import KEGG pathways. However, many complexes have not been described in KGML files. So interactions from these complexes cannot be extracted. For example, see complex HAP1-HTT-Dyneim-Dunactin in Huntington’s disease pathway: <http://www.genome.jp/kegg/pathway/hsa/hsa05016.html>. It seems there is no good file that can be used to extract these complexes. Probably a future import can use map files directly (e.g. hsadd05016.conf).

3). Because of the above problem, the pathway to protein/gene mapping file extracted for KEGG uses KGML files directly, instead of imported Reactome pathways (see below).

**NCI-PID**

There are two kinds of pathways available in this database: pathways curated by NCI-PID curators and pathways imported from BioCarta and Reactome. Indeed, we don’t need to import Reactome pathways here. BioCarta pathways have not been updated since they were imported in this database. But because the changes of the Reactome schema, these pathways should be re-imported each time if we want to use them.

1. Download pathways from NCI-PID: The data format we are going to use is BioPAX level 2. Download these two BP2 files in its download site, <http://pid.nci.nih.gov/download.shtml>: NCI-Nature Curated Data (BioPAX Level 2), and BioCarta data (BioPAX Level 2). After unzipping, the file names should be:
   1. NCI-Nature\_Curated.bp2.owl
   2. BioCarta.bp2.owl
2. Modified these four constants in org.reactome.r3.util.R3Constants so that they point to correct locations:

// Used for the Nature-PID database files

**public** **static** **final** String *NATURE\_PID\_DIR* = *DATA\_SET\_DIR* + "NCI-Pathways/011612/";

**public** **static** **final** String *NATURE\_PID\_CURATED* = *NATURE\_PID\_DIR* + "NCI-Nature\_Curated.bp2.owl";

**public** **static** **final** String *NATURE\_PID\_CURATED\_CONVERTED* = *NATURE\_PID\_DIR* + "NCI-Nature\_Curated.bp2.rtpj";

**public** **static** **final** String *NATURE\_PID\_BIOCARTA* = *NATURE\_PID\_DIR* + "BioCarta.bp2.owl";

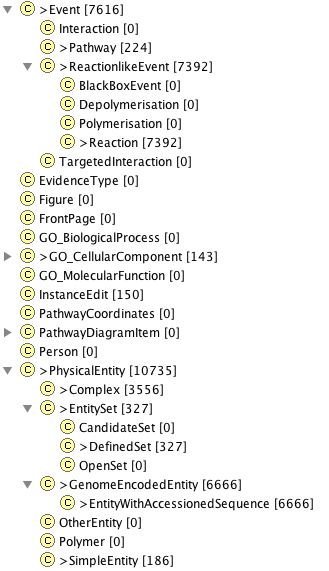
**public** **static** **final** String *NATURE\_PID\_BIOCARTA\_CONVERTED* = *NATURE\_PID\_DIR* + "BioCarta.bp2.rtpj";

1. Find the BioPAX to Reactome mapping file: resources/BioPAXToReactomeMappers.xml. Make sure the following XML element is NOT commented out so that we can use NciPIDBToRPostProcessor class in package org.reactome.b2rPostProcessor during mapping:

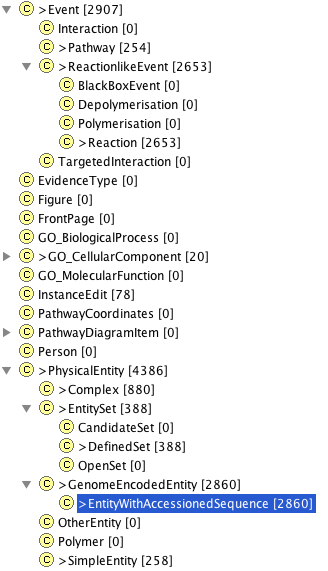
<!-- This PostProcessor is for NCI Pathways -->

<postProcessor class=*"org.reactome.b2rPostProcessor.NciPIDBToRPostProcessor"* />

1. Find package org.reactome.b2rPostProcessor, and class NciPIDConverterRunner in this package.
   1. For converting curated pathways downloaded from NCI-PID, run method runConvertOfCurated()
   2. For converting BioCarta pathways, run method runConvertOfBioCarta().
   3. Note: Make sure you have assigned enough memory to these two methods (e.g. -Xmx1024m).
2. For reference, here are some numbers for a version converted in 2012 (curated):



1. Here are some numbers for the imported NCI-PID BioCarata data set:



**Panther**

Pathways in the Panther database have not been updated since December, 2007, though the version number has been changed from 2.5 to 3.0.1. Basically the annotation of pathways in Panther has stopped for quite a while. However, because our Reactome schame has been evolved during the process, we still need to re-convert pathways from Panther into the Curator Tool project in order to merge with other pathways.

1. Download Pathways: download pathways from Panther’s ftp site, [ftp.pantherdb.org/pathway/current\_release](ftp://ftp.pantherdb.org/pathway/current_release). Please download these two files: SBML\_{version}.zip and SequenceAssociationPathway3.0.1.txt. We use the SBML files, which are the file format used by the Panther annotation. The second file is used to map pathway component ids used in pathways into UniProt accession numbers.

Note: One pathway component used in panther pathways can be mapped to multiple UniProt identifiers because these components are actually based on HMM families constructed by Panther. In order to control the quality of imported pathways, we use mappings based on the following reliable confidence only:

"IGI",

"IPI",

"IDA",

"IEP",

"TAS",

"IC",

"IMP",

"RCA"

1. Modify the following constants related to Panther in class R3Constants in package org.reactome.r3.util:

**public** **static** **final** String *PANTHER\_DIR* = *DATA\_SET\_DIR* + "Panther/Version3.0.1/"; // Download on Jan 18, 2011

**public** **static** **final** String *PANTHER\_MAPPING\_FILE* = *PANTHER\_DIR* + "SequenceAssociationPathway3.01.txt";

**public** **static** **final** String *PANTHER\_CONVERTED\_FILE* = *PANTHER\_DIR* + "Panther\_3\_0\_1.rtpj";

1. Running converting: Find method testNewBatchConvert() in class org.reactome.panther.PantherToReactomeConverterTest(). This is a JUnit 3 test method. You may have to change the setting in Eclipse and also assign enough memory to it. The final output should be specified in step 2. To run this method, please also modify the following method, so that MySQLAdaptor can point to a correct database:

**private** MySQLAdaptor getDBA() **throws** Exception {

MySQLAdaptor dba = **new** MySQLAdaptor("localhost",

R3Constants.REACTOME\_DB\_NAME,

R3Constants.DB\_USER,

R3Constants.DB\_PWD,

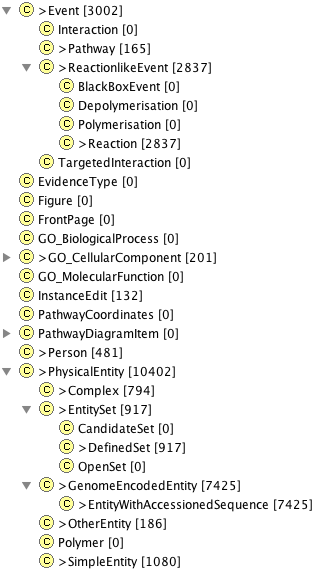
3306);

**return** dba;

}

*Note: Some part of Panther converted was developed by Andreas Hoelzlwimmer* [*andreash@ebi.ac.uk*](mailto:andreash@ebi.ac.uk) *during his internship at EBI. Needs to give him an acknowledgement in next FI related publication.*

1. For reference, here are some numbers in the converted Curator Tool project:



**CellMap**

Not needed any more!

**TRED**

TRED provides us transcription factors (TF) and their targets interactions. Many of interactions in this database are computationally predicted. We extract only manual curated interactions, which have been supported by literatures if not considering curation errors. The content in the TRED database should not be updated any more. What we need to do is just convert the data in the TRED database into a Curator tool project using the latest Reactome data model. To fetch the data from the TRED database, we use a customized hibernate based API. So have to make sure you have set up your class path correctly in order to use this hibernate API.

1. Install the TRED database into your local computer: If you are building the FI network at your local computer, you may need to install the TRED database for quick performance. If you don’t want to install it, you can access this database at reactomedev.oicr.on.ca. The database name is test\_TRED. To install this database, do a mysqldump from reactomedev, and load the mysqldump into your local mysql database.
2. Configure the hibernate configuration file: Find file TREDHibernate.cfg.xml in the resources folder. Make changes to these three properties so that they point to the correct values:

<property name=*"connection.url"*>jdbc:mysql://localhost:3306/TRED</property>

<property name=*"connection.username"*>root</property>

<property name=*"connection.password"*>macmysql01</property>

1. Make changes to these two constants in class R3Constants in package org.reactome.r3.util:

// For TRED files

**public** **static** **final** String *TRED\_DIR* = *DATA\_SET\_DIR* + "TRED/";

**public** **static** **final** String *TRED\_CONVERTED\_FILE* = *TRED\_DIR* + "TRED\_011912.rtpj";

1. Find package org.reactome.tred, and class TREDToReactomeConverter in the package. Run method doConvert(), which is a JUnit 4 method. Make sure you have assigned enough memory in the running configuration dialog.

Note: As of January, 2012, we use gene names, which are used in the TRED database, in our target Reactome database directly to map to UniProt. Some of gene names in TRED cannot be mapped. See below (partial list):

2012-01-19 14:56:22,763 [main] WARN org.reactome.convert.common.PostProcessTemplate - E2F-4 for EWAS -3 cannot be mapped to UniProt!

2012-01-19 14:56:22,900 [main] WARN org.reactome.convert.common.PostProcessTemplate - DNA Pol alpha for EWAS -42 cannot be mapped to UniProt!

2012-01-19 14:56:22,935 [main] WARN org.reactome.convert.common.PostProcessTemplate - RNAPII large subunit for EWAS -58 cannot be mapped to UniProt!

2012-01-19 14:56:22,947 [main] WARN org.reactome.convert.common.PostProcessTemplate - DNA topoisomerase I for EWAS -68 cannot be mapped to UniProt!

2012-01-19 14:56:22,991 [main] WARN org.reactome.convert.common.PostProcessTemplate - PCNA p120 for EWAS -97 cannot be mapped to UniProt!

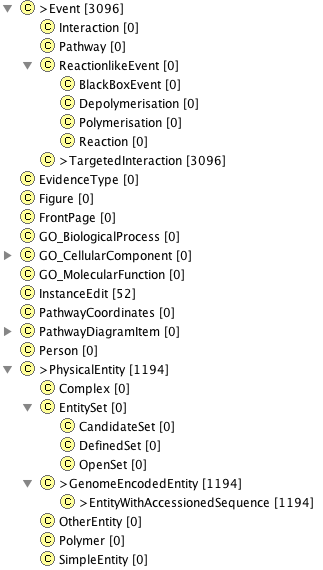
2012-01-19 14:56:23,004 [main] WARN org.reactome.convert.common.PostProcessTemplate - p107 for EWAS -110 cannot be mapped to UniProt!

2012-01-19 14:56:23,026 [main] WARN org.reactome.convert.common.PostProcessTemplate - E2F-1 for EWAS -129 cannot be mapped to UniProt!

…………

Total unmapped names: 38

1. Here are some numbers:



Note: All TF/Target interactions have been converted into TargetedInteractions.

Note: We may get a rough idea how many proteins can be merged into the original Reactome by running this Java method: checkUniProtNumbersInConvertedDBs() in class org.reactome.r3. ProteinAndInteractionCount. For release 40, we may get the following numbers (ids for UniProt accession numbers):

Total ids from Reactome: 7447 // This number may be bigger than the release one

Total ids from KEGG: 7897

Total ids after merging: 11525 (0.5163986323920476, 4078)

Total SwissProt ids after merging: 8534 (0.421473725800079)

Total ids from Nature-PID: 2573

Total ids after merging: 11984 (0.17839098328799066, 459)

Total SwissProt ids after merging: 8977 (0.44335242986961676)

Total ids from Biocarta-PID: 1526

Total ids after merging: 12785 (0.5249017038007864, 801)

Total SwissProt ids after merging: 9076 (0.4482418016594232)

Total ids from TRED: 1142

Total ids after merging: 12951 (0.14535901926444833, 166)

Total SwissProt ids after merging: 9242 (0.4564401422362703)

Total ids from Panther: 1424

Total ids after merging: 13120 (0.11867977528089887, 169)

Total SwissProt ids after merging: 9411 (0.4647866455946266)

**Dump converted pathways into the Reactome source database**

All these converted rtpj files should be saved into the Reactome source database to be used by the FI network as the sources for extracted FIs.

1. Find class ConvertedPathwayDbDumper in package org.reactome.b2rPostProcessor. Make sure you have configured log4j.properties as what you want in the resources folder. Usually you can just output logging into the Eclipse console.
2. Run method dump() in the class. Before running it, make sure you have assigned enough memory to this method (say: -Xmx2048m)
3. After the above method run, all converted pathways should be in the database. You can check your imported events by searching using dataSource in the curator tool, database browser.

Note: Some of UniProt identifiers that cannot be mapped to the original ReferenceGeneProducts from different databases may be duplicated in the database. Just leave them as is. But have to remember to merge them when counting how many UniProt accessions in the database. Avoid checking the numbers based on ReferenceGeneProduct instances only.

1. Dump all extracted FIs from Reactome and converted Pathways:
   1. Make sure assign corrected DB\_IDs for different data sources in method getPathwayDbAnalyzers() in class org.reactome.r3.ReactomeAnalyzer.
   2. Run method dumpPathwayFIs() in class org.reactome.r3.FIFileAnalyzer. You should get similar numbers or more as follows (March, 2012):

$ wc -l FIs\_\*.txt

6761 FIs\_BioCarta - Imported by PID.txt

48819 FIs\_Kegg.txt

15694 FIs\_Pathway Interaction Database.txt

140040 FIs\_Reactome.txt

2858 FIs\_TRED.txt

15400 FIs\_pantherdb.txt

229627 total

**Data sources for predicted FIs**

**Ensembl-Compara**

Ensembl-compara is used to map PPIs from non-human species to human. We need to have a local compara database to do this mapping by downloading some files from the Ensembl-Compara database:

1. Download files: In this web page, <http://www.ensembl.org/info/data/ftp/index.html>, choose Comparative/MySQL to its ftp site. From the ftp site, choose ensemble\_compara\_xx (xx for release number, 66 as of February, 2012. However, for some reason, many yeast UniProt ids are not in this current release. So release 62 is used!). Download member.txt.gz, family.txt.gz, and family\_member.txt.gz. Also download the database schema file: ensembl\_compara\_62.sql.gz. Unzip all files.
2. Create ensemble\_compara database: log into a local mysql database, create a ensembl\_compara database by using, “create database ensembl\_compara\_xx” (xx should be the release number for compara). After the database is created, load the schema by using ensembl\_compara\_xx.sql with command: source ensembl\_compara\_62.sql, assuming you start mysql from the directory containing this sql file. Otherwise, provide the absolute path to it.
3. Load data into the database: log out from the mysql database, and run the following command:

mysqlimport –u{mysql\_db\_user} -p --local ensembl\_compara\_62 family.txt family\_member.txt member.txt

Note: make sure you use correct mysql user name and password

1. Make changes to the following constants in R3Constances class to point correct files:

// For ensembl related files

// public static final String ENSEMBL\_DIR = DATA\_SET\_DIR + "Ensembl/release\_53/";

**public** **static** **final** String *ENSEMBL\_DIR* = *DATA\_SET\_DIR* + "Ensembl/release\_62/";

**public** **static** **final** String *ENSEMBL\_COMPARA\_DATABASE* = "ensembl\_compara\_62";

**public** **static** **final** String *ENSEBML\_PROTEIN\_FAMILIES* = *ENSEMBL\_DIR* + "ProteinFamilies.txt";

1. Create ENSEBML-Compara protein families: run method dumpProteinFamilies in class org.reactome.r3.EnsemblAnalyzer. Make sure you have enter correct information in method getConnection() in the method.
2. Note: we may also connect to the ensembl public mysql databases. However, it is much faster to load the above tables into a local mysql database.

**Protein-Protein Interactions (PPIs)**

Protein protein interactions are downloaded from iRefIndex, a comprehensive protein-protein interaction databases by grouping many popular protein interaction databases together, including IntAct, BioGrid, HPRD, BIND, and others, and creating a non-redundant interaction database using customized hash keys. One problem using this database is that we need to make sure this data source is updated regularly. As of now (February, 2012, release 9.0), it is updated pretty regularly.

1. Download interaction files: the PPIs in iRefIndex are provided in PSIMI-TAB format. The following files (these files names are for release 9.0. Names may have different date in future releases) should be downloaded from its ftp site via this web page: <http://irefindex.uio.no/wiki/README_MITAB2.6_for_iRefIndex>, and unzip them.

4932.mitab.10182011.txt.zip (baker’s yeast): this file is not used!

6239.mitab.10182011.txt.zip (worm)

7227.mitab.10182011.txt.zip (fly)

9606.mitab.10182011.txt.zip (human)

559292.mitab.10182011.txt.zip (yeast S288c)

10090.mitab.10182011.txt.zip (mouse)

1. Based on the above data files, make necessary changes in the follow configuration constants in class R3Constants:

// Constants for iRefIndex data files

**public** **static** **final** String *IREFINDEX\_DIR* = *DATA\_SET\_DIR* + "iRefIndex/9.0/";

**public** **static** **final** String *IREFINDEX\_HUMAN\_FILE* = *IREFINDEX\_DIR* + "9606.mitab.10182011.txt";

**public** **static** **final** String *IREFINDEX\_HUMAN\_PPI\_FILE* = *IREFINDEX\_DIR* + "HumanPPIsInUniProt022712.txt";

**public** **static** **final** String *IREFINDEX\_YEAST\_FILE* = *IREFINDEX\_DIR* + "559292.mitab.10182011.txt";

**public** **static** **final** String *IREFINDEX\_YEAST\_PPI\_FILE* = *IREFINDEX\_DIR* + "YeastPPIsInUniProt022812.txt";

**public** **static** **final** String *IREFINDEX\_FLY\_FILE* = *IREFINDEX\_DIR* + "7227.mitab.10182011.txt";

**public** **static** **final** String *IREFINDEX\_FLY\_PPI\_FILE* = *IREFINDEX\_DIR* + "FLyPPIsInUniProt022812.txt";

**public** **static** **final** String *IREFINDEX\_WORM\_FILE* = *IREFINDEX\_DIR* + "6239.mitab.10182011.txt";

**public** **static** **final** String *IREFINDEX\_WORM\_PPI\_FILE* = *IREFINDEX\_DIR* + "WormPPIsInUniProt022812.txt";

**public** **static** **final** String *IREFINDEX\_MOUSE\_FILE* = *IREFINDEX\_DIR* + "10090.mitab.10182011.txt";

**public** **static** **final** String *IREFINDEX\_MOUSE\_PPI\_FILE* = *IREFINDEX\_DIR* + "MousePPIsInUniProt031412.txt";

**public** **static** **final** String *IREFINDEX\_MOUSE\_TO\_HUMAN\_PPI\_FILE* = *IREFINDEX\_DIR* + "HumanPPIsFromMouseInUniProt031412.txt";

1. Run the following methods to extract PPIs from the above downloaded files in class IRefIndexMITTabAnalyzer in package org.reactome.r3:
   1. loadHumanPPIs(): extract human PPIs
   2. loadFlyPPIs(): extract Fly PPIs
   3. loadWormPPIs(): extract worm PPIs
   4. loadYeastPPIs(): extract yeast PPIs
   5. loadMousePPIs(): extract mouse PPIs
2. You may check how many PPIs you have got by using this simple Unix command:

$ wc -l \*PPIs\*

38,940 FLyPPIsInUniProt022812.txt

95,959 HumanPPIsInUniProt022712.txt

11,626 WormPPIsInUniProt022812.txt

12,787 MousePPIsInUniProt031412.txt

214,848 YeastPPIsInUniProt022812.txt

1. Map non-human PPIs to human PPIs using ENSEMBL compara protein families so that these PPIs can be used as features in Naïve Bayes Classifier
   1. Map yeast PPIs to human PPIs by running method generateHumanPPIsFromYeastInUniProt() in class org.reactome.psi.data.PsiMiOrthologyAnalyzer.
   2. Run generateHumanPPIsFromWormInUniProt() for worm
   3. Run generateHumanPPIsFromFlyInUniProt() for fly
   4. Run generateHumanPPIsFromMouseInUniProt() for mouse
   5. The final numbers as of March, 2012:

376,781 HumanPPIsFromFly030112.txt

122,547 HumanPPIsFromWorm030112.txt

1,304,755 HumanPPIsFromYeast030112.txt

770,385 HumanPPIsFromMouse031412.txt

1. You can check the odds ratio for the human PPIs and mapped PPIs by running this method: testPPIsCoverage() in class org.reactome.psi.data.PsiMiOrthologyAnalyzer. Here are some results from running this method in March, 2012:

File: datasets/iRefIndex/9.0/HumanPPIsInUniProt022712.txt

Total checked pairs: 95959

Total: 140359

Mapped to ppi: 8637 (0.061535)

Mapped to random: 156 (0.001111)

……

Average odds ratio: 63.731439292269116 +- 3.6748121713411286 (from 10 tests)

File: results/v4/HumanPPIsFromYeast030112.txt

Total checked pairs: 1304755

Total: 140359

Mapped to ppi: 2208 (0.015731)

Mapped to random: 188 (0.001339)

……

Average odds ratio: 11.907809031980854 +- 0.6991789633961197 (from 10 tests)

File: results/v4/HumanPPIsFromWorm030112.txt

Total checked pairs: 122547

Total: 140359

Mapped to ppi: 592 (0.004218)

Mapped to random: 24 (0.000171)

……

Average odds ratio: 19.746145664598135 +- 2.75733192050712 (from 10 tests)

File: results/v4/HumanPPIsFromFly030112.txt

Total checked pairs: 376781

Total: 140359

Mapped to ppi: 1943 (0.013843)

Mapped to random: 91 (0.000648)

……

Average odds ratio: 25.21456370938219 +- 2.4931316261019165 (from 10 tests)

File: results/v4/HumanPPIsFromMouse031412.txt

Total checked pairs: 770385

Total: 140359

Mapped to ppi: 4628 (0.032973)

Mapped to random: 131 (0.000933)

Mapped to random: 150 (0.001069)

Mapped to random: 143 (0.001019)

Mapped to random: 143 (0.001019)

Mapped to random: 130 (0.000926)

Mapped to random: 128 (0.000912)

Mapped to random: 135 (0.000962)

Mapped to random: 129 (0.000919)

Mapped to random: 144 (0.001026)

Mapped to random: 121 (0.000862)

Average odds ratio: 35.45707647231835 +- 2.395786420492694 (from 10 tests)

Note: To run the above method, we need FIs extracted from the Reactome database. To extract FIs from the Reactome database, run method extractInteractions() in class org.reactome.r3.ReactomeAnalyzer. Be sure to configure R3Constants.Reactome\_FI\_FILE correctly.

**Gene-Expression Correlations**

Two gene expression data sets have been used as NBC features: Lee’s Gene Expression and Prieto’s Gene Expression. The original data files were downloaded from their paper’s web sites, and should not be changed any more. However, we need to update protein-pairs to latest version of UniProt.

Note: The original downloaded data files contain gene pairs only. In order to be used as features in NBC, we have to map gene names to UniProt accession numbers. We use downloaded UniProt data file to do this mapping. We use the swissprot part of UniProt data for doing the mapping for the Prieto data file. For Lee’s data file, we use the original downloaded mapping file and normalize with the latest UniProt data.

1. Get the archived source files from CVS: synchronize data\_archive/data\_archive.zip. Unzip this zipped file. You should find two files: GeneExpWith3FromPavlidis.txt under the LeeGeneExp folder, and union60.txt in the PrietoGeneExp folder.
2. Make sure these two constants in R3Constants have been set correctly (usually you don’t need to make a change):

// public static final String LEE\_GENE\_EXP\_FILE = "results/v3/PavlidisCoExp\_Norm.txt";

**public** **static** **final** String *LEE\_GENE\_EXP\_FILE* = *RESULT\_DIR* + "LeeGeneExp.txt";

// public static final String PRIETO\_GENE\_EXP\_FILE = "results/v3/CarlosCoExp\_Norm.txt";

**public** **static** **final** String *PRIETO\_GENE\_EXP\_FILE* = *RESULT\_DIR* + "PrietoGeneExp.txt";

1. Normalize Lee Gene Expression: Run method normalizeLeeGeneExp() method in class org.reactome.r3. MicroarrayDataAnalyzer. Make sure you have set the correct inFileName in the method. The output name should not be changed.
2. Normalize Prieto Gene Expression: run method generatePrietoCarlosGeneExpFile() in class org.reactome.r3. MicroarrayDataAnalyzer. Make sure the srcFileName in the method points to the union60.txt file.

Note: We use the SwissProt part of UniProt data only in order to map gene names to UniProt accessions. This is used to control the size of the final UniProt pairs. There are just two many mappings if we use STREMBL too: many of HLA proteins. Since our FI network focuses on the SwissProt proteins, this should be fine.

1. Checked the performance of these two gene expression data as NBC features using Odds ratio by running methods in class MicroarrayDataAnalyzer
   1. checkCoExpFromPavlidis() for the Lee data

Total Co-expression: 205903

Total checked pairs: 205903

Total: 140359

Mapped to ppi: 5853 (0.041700)

Mapped to random: 312 (0.002223)

Mapped to random: 338 (0.002408)

Mapped to random: 338 (0.002408)

Mapped to random: 339 (0.002415)

Mapped to random: 312 (0.002223)

Mapped to random: 309 (0.002201)

Mapped to random: 338 (0.002408)

Mapped to random: 341 (0.002429)

Mapped to random: 355 (0.002529)

Mapped to random: 333 (0.002372)

Average odds ratio: 18.416709742008 +- 0.8653644305136787 (from 10 tests)

* 1. checkCoExpFromPrietoCarlos() for the Prieto data

Co-expressed genes: 19204

Total checked pairs: 19204

Total: 140359

Mapped to ppi: 1477 (0.010523)

Mapped to random: 32 (0.000228)

Mapped to random: 38 (0.000271)

Mapped to random: 26 (0.000185)

Mapped to random: 25 (0.000178)

Mapped to random: 29 (0.000207)

Mapped to random: 33 (0.000235)

Mapped to random: 28 (0.000199)

Mapped to random: 30 (0.000214)

Mapped to random: 21 (0.000150)

Mapped to random: 35 (0.000249)

Average odds ratio: 51.64469900367323 +- 9.323084566995446 (from 10 tests)

**Gene Ontology Annotation**

The sharing of GO annotation (Biological Process only) is used as another feature in our NBC training and prediction. In our usage, we check a simple sharing only without considering any parent-child relationships among GO terms.

1. Download GO annotation file for homo sapiens from the gene ontology’s web site: <http://www.geneontology.org/GO.downloads.annotations.shtml>.
2. Modify the following value in R3Constants class to point to the correct directory that holds the above downloaded file:

**public** **static** **final** String *GO\_DIR* = *DATA\_SET\_DIR* + "GO/030712/";

1. This is optional: you may check the performance of the GO term sharing features using method, testGOFeatures() in class org.reactome.r3.GODataAnalyzerV2. Similar results are as follows:

GO BP:

Total: 140359

Mapped to ppi: 83184 (0.592652)

Mapped to random: 11991 (0.085431)

……

Average odds ratio: 15.607724990002925 +- 0.1401811632893593 (from 10 tests)

GO MF:

Total: 140359

Mapped to ppi: 69984 (0.498607)

Mapped to random: 28428 (0.202538)

……

Average odds ratio: 3.916952909578359 +- 0.04072282915792495 (from 10 tests)

GO CP:

Total: 140359

Mapped to ppi: 96002 (0.683975)

Mapped to random: 47236 (0.336537)

……

Average odds ratio: 4.2744730649963945 +- 0.02449319661431225 (from 10 tests)

Note: the odds ratio for GO BP is much higher than others. This is the reason why we have not chosen other two GO aspects.

**Domain-Domain Interactions**

We use domain-domain interactions from the pFam database.

1. Generate UniProt accession to pFam domain ids mapping file: run method genreateUniToPfamMap() in class UniProtAnalyzer.
2. Download domain interaction file from pFam: go to pFam release ftp site, <ftp://ftp.sanger.ac.uk/pub/databases/Pfam>, go to the release folder, the latest release, and database\_files. Dowload two files: pfamA.txt.gz and pfamA\_interactions.txt.gz, unzip them.
3. Specify the following constants as being used in step 2:

// Directory for pFam

**public** **static** **final** String *PFAM\_DIR\_NAME* = *DATA\_SET\_DIR* + "Pfam/26.0/";

1. Generate simple domain-domain interaction file: In class org.reactome.r3.PfamAnalyzer, run method convertIntToPfamIDs() method.
2. Check the odds ratio to make sure the converted files are correct to be used: run method testPfamFeature() in class PfamAnalyzer. The results should be similar to the following:

Total: 140359

Mapped to ppi: 26325 (0.187555)

Mapped to random: 2417 (0.017220)

Mapped to random: 2502 (0.017826)

Mapped to random: 2450 (0.017455)

Mapped to random: 2379 (0.016949)

Mapped to random: 2469 (0.017591)

Mapped to random: 2525 (0.017990)

Mapped to random: 2536 (0.018068)

Mapped to random: 2592 (0.018467)

Mapped to random: 2546 (0.018139)

Mapped to random: 2438 (0.017370)

Average odds ratio: 12.814447338094961 +- 0.3469475098844736 (from 10 tests)

Note: the odds ratio should be around 10.

**Gene Ways Protein-Protein Interactions**

Not used any more because many PPIs are actually extracted from literatures manually!

**Naïve Bayes Classifier (NBC) Training and FIs Prediction**

Classes related to NBC training and FI prediction using a trained NBC are in package org.reactome.weka.

1. Validate the features used in class FeatureHandlerForV3
   1. In method getFeatureList(), make sure all features to be used by NBC have been listed there.
   2. In method loadFatureToPairs(), make sure all PPIs features have point to correct file names. Usually this should be correct if you have set up correct constants in org.reactome.r3.util.R3Constants class.
2. Training NBC:
   1. Run method calculateNBCBasedOnReactome() in class NBCAnalyzer.
   2. The output from the above running may look like this (the values for cutoff sections have been edited):

// Total FIs extracted from Reactome

Total pairs: 140040

// Total FIs used in the training data set

After filtering: 92314

Prior probability: 0.005421774137184735

// Negative in the training data set: 100:1 is used.

Total negative: 14004000

// Total FIs in the testing data set

test fis before filtering: 80521

// Total FIs actually used in the test data set

Total test fis: 19316

TP rates and FP rates under different cutoff values.

|  |  |  |
| --- | --- | --- |
| Cutoff | False\_Positive\_Rate | True\_Positive\_Rate |
| 0 | 1 | 1 |
| 0.1 | 0.007763129 | 0.233899358 |
| 0.2 | 0.007610401 | 0.226392628 |
| 0.3 | 0.007008323 | 0.200662663 |
| 0.4 | 0.001423898 | 0.115396562 |
| 0.5 | 0.001276366 | 0.10669911 |
| 0.6 | 9.06E-04 | 0.087647546 |
| 0.7 | 4.51E-04 | 0.051615241 |
| 0.8 | 4.43E-04 | 0.051149306 |
| 0.9 | 3.62E-04 | 0.047991303 |
| 1 | 0 | 0 |

// Check posterior probability if only one feature is used.

One feature contribution:

pavlidisGeneExp: 0.006974915512111727

mousePPI: 0.014130702196887439

celPPI: 0.007328678807142349

humanInteraction: 0.023837402836575705

pfamDomainInt: 0.007155776198220065

carlosGeneExp: 0.019159908466993246

dmePPI: 0.010126542317059548

scePPI: 0.004865204578762476

goBPSharing: 0.02421823647125449

// Conditional probabilities for each feature calculated from NBC.

Learning probabilties:

carlosGeneExp\_false|false: 0.9997215081405313

carlosGeneExp\_false|true: 0.9840002599822345

carlosGeneExp\_true|false: 2.784918594687232E-4

carlosGeneExp\_true|true: 0.015999740017765454

celPPI\_false|false: 0.999707012282205

celPPI\_false|true: 0.9935762722880603

celPPI\_true|false: 2.929877177949157E-4

celPPI\_true|true: 0.006423727711939684

dmePPI\_false|false: 0.999296772350757

dmePPI\_false|true: 0.9789414390016682

dmePPI\_true|false: 7.032276492430734E-4

dmePPI\_true|true: 0.021058560998331782

goBPSharing\_false|false: 0.8901596686660954

goBPSharing\_false|true: 0.09851160170721662

goBPSharing\_true|false: 0.1098403313339046

goBPSharing\_true|true: 0.9014883982927834

humanInteraction\_false|false: 0.9986046843758926

humanInteraction\_false|true: 0.9074788222804775

humanInteraction\_true|false: 0.001395315624107398

humanInteraction\_true|true: 0.0925211777195225

mousePPI\_false|false: 0.9987644244501571

mousePPI\_false|true: 0.9496826050219902

mousePPI\_true|false: 0.001235575549842902

mousePPI\_true|true: 0.050317394978009834

pavlidisGeneExp\_false|false: 0.9967849185946872

pavlidisGeneExp\_false|true: 0.9365968325497758

pavlidisGeneExp\_true|false: 0.0032150814053127676

pavlidisGeneExp\_true|true: 0.06340316745022423

pfamDomainInt\_false|false: 0.9819181662382177

pfamDomainInt\_false|true: 0.7160344043157051

pfamDomainInt\_true|false: 0.018081833761782348

pfamDomainInt\_true|true: 0.2839655956842949

scePPI\_false|false: 0.9983254070265638

scePPI\_false|true: 0.9760816344216479

scePPI\_true|false: 0.001674592973436161

scePPI\_true|true: 0.023918365578352144

* 1. You can also try to run class NBCGUITest.main() to study the results using different combinations of feature selections. Here is a screenshot from this application:



1. Plot ROC curve (receiver operating characteristic curve) and calculate AUC (area under curve):
   1. Run method calcualteROCPoints() in class org.reactome.weka.NBCAnalyzer. Make sure you have assigned correct file name in this method.
   2. Find a R script called “ROCCurveDrawing.R” in the folder RSource, change the value fileName to the above generated ROC point file. Run this R script in R. You should get the ROC curve similar to the following:



* 1. Calculate AUC using R:

> calculate.AUC(rocData)

[1] "AUC: 0.858892192875038"

1. Check protein coverage by merging predicted FIs and extracted FIs under different cutoff values
   1. Generate protein pairs having domain-domain interactions by running method checkSharedBPPairAndDomainPair()
   2. Run method checkCutoffValueForPredictedFIs() in class NBCAnalyzer. The results should be similar as following (Note: prediction FIs have excluded pathway FIs):

Total pairs: 3693723

Cutoff: 0.1

FIs from pathways: 193901 (9609)

SwissProt IDs: 7378

SwissProt Coverage: 0.3643816673251679

FIs from prediction: 1307694 (27683)

SwissProt IDs: 12690

SwissProt Coverage: 0.6267285657842749

FIs merged: 1501595 (29794)

SwissProt IDs: 13440

SwissProt Coverage: 0.6637692611615962

Cutoff: 0.2

FIs from pathways: 193901 (9609)

SwissProt IDs: 7378

SwissProt Coverage: 0.3643816673251679

FIs from prediction: 1260342 (26405)

SwissProt IDs: 12610

SwissProt Coverage: 0.6227775582773607

FIs merged: 1454243 (28588)

SwissProt IDs: 13376

SwissProt Coverage: 0.6606084551560648

Cutoff: 0.30000000000000004

FIs from pathways: 193901 (9609)

SwissProt IDs: 7378

SwissProt Coverage: 0.3643816673251679

FIs from prediction: 1021057 (22947)

SwissProt IDs: 12480

SwissProt Coverage: 0.6163571710786251

FIs merged: 1214958 (25397)

SwissProt IDs: 13269

SwissProt Coverage: 0.655323982615567

Cutoff: 0.4

FIs from pathways: 193901 (9609)

SwissProt IDs: 7378

SwissProt Coverage: 0.3643816673251679

FIs from prediction: 72977 (10893)

SwissProt IDs: 8456

SwissProt Coverage: 0.4176214934808376

FIs merged: 266878 (15276)

SwissProt IDs: 10897

SwissProt Coverage: 0.538176610035559

Cutoff: 0.5

FIs from pathways: 193901 (9609)

SwissProt IDs: 7378

SwissProt Coverage: 0.3643816673251679

FIs from prediction: 67892 (10366)

SwissProt IDs: 8024

SwissProt Coverage: 0.3962860529435006

FIs merged: 261793 (14884)

SwissProt IDs: 10591

SwissProt Coverage: 0.523064006321612

Cutoff: 0.6000000000000001

FIs from pathways: 193901 (9609)

SwissProt IDs: 7378

SwissProt Coverage: 0.3643816673251679

FIs from prediction: 51293 (9169)

SwissProt IDs: 7141

SwissProt Coverage: 0.3526768075859344

FIs merged: 245194 (14155)

SwissProt IDs: 10124

SwissProt Coverage: 0.5

Cutoff: 0.7000000000000001

FIs from pathways: 193901 (9609)

SwissProt IDs: 7378

SwissProt Coverage: 0.3643816673251679

FIs from prediction: 37767 (6663)

SwissProt IDs: 4761

SwissProt Coverage: 0.23513433425523508

FIs merged: 231668 (12933)

SwissProt IDs: 9013

SwissProt Coverage: 0.4451303832477282

Cutoff: 0.8

FIs from pathways: 193901 (9609)

SwissProt IDs: 7378

SwissProt Coverage: 0.3643816673251679

FIs from prediction: 34390 (6442)

SwissProt IDs: 4751

SwissProt Coverage: 0.2346404583168708

FIs merged: 228291 (12740)

SwissProt IDs: 9010

SwissProt Coverage: 0.4449822204662189

Cutoff: 0.9

FIs from pathways: 193901 (9609)

SwissProt IDs: 7378

SwissProt Coverage: 0.3643816673251679

FIs from prediction: 25230 (5735)

SwissProt IDs: 4532

SwissProt Coverage: 0.223824575266693

FIs merged: 219131 (12157)

SwissProt IDs: 8885

SwissProt Coverage: 0.43880877123666534

Note: You may change the cutoff values to other to get more detailed values.

1. Based on results from above, choose a reasonable cutoff value for high specificity. Usually cutoff value 0.50 should be chosen though you may get a pretty low true positive rate (recall rate).
2. Predict FIs using the training NBC:
   1. Make sure a correct file names has been specified in the R3Constant class:

**public** **static** **final** String *PREDICTED\_FI\_FILE* = *RESULT\_DIR* + "PredictedFIs\_031512.txt";

* 1. Run method generatePredictedFIs() in class NBCAnalyzer using the selected cutoff value at step 5. You may see some output like follows:

FIs from pathways: 193901 (9609)

SwissProt IDs: 7378

SwissProt Coverage: 0.3643816673251679

FIs from prediction: 67892 (10366)

SwissProt IDs: 8024

SwissProt Coverage: 0.3962860529435006

FIs merged: 261793 (14884)

SwissProt IDs: 10591

SwissProt Coverage: 0.523064006321612

1. Generate a FI database:
   1. Create an empty database in mysql: create database FI\_{YYYY} (YYYY should be the year of building, eg. 2012).
   2. Modify the hibernate configuration file for this database: resources/ funcIntHibernate.cfg.xml for the following values:

<property name=*"connection.url"*>jdbc:mysql://localhost:3306/FI\_2012</property>

<property name=*"connection.username"*>root</property>

<property name=*"connection.password"*>macmysql01</property>

……

<!-- Drop and re-create the database schema on startup -->

<property name=*"hbm2ddl.auto"*>create</property>

……

* 1. In package org.reactome.hibernate, find class HibernateFIReader, and run method, setSetting(). After running this method, several tables should be generated by hibernate API automatically. Check mysql, and make sure you get the following tables:

mysql> show tables;

+----------------------------+

| Tables\_in\_fi\_2012 |

+----------------------------+

| Evidence |

| Interaction |

| InteractionReactomeSources |

| Protein |

| ProteinDbReferences |

| ReactomeSource |

| dbreference |

+----------------------------+

7 rows in set (0.00 sec)

Note: For some reason, hibernate uses lower case for table dbreference. You may change the table name by the following two SQLs:

mysql> rename table dbreference to DbReference1; rename table DbReference1 to DbReference;

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

* 1. Turn auto-create database schema in the hibernate configuration file off by reset the following configurations as follows:

<!-- Echo all executed SQL to stdout -->

<property name=*"show\_sql"*>false</property>

<property name=*"use\_sql\_comments"*>false</property>

<!-- Drop and re-create the database schema on startup -->

<!-- property name=*"hbm2ddl.auto"*>create</property -->

* 1. Dump extracted pathway FIs into the FI database by running method HibernateFIWriter.dump():

Total empty interactions: 0

Total empty interactions: 0

Cannot extract interactions: [Interaction:2161776] CDH1, Arc-1, CD324, CDHE, ECAD, LCAM, UVO-CDH1, Arc-1, CD324, CDHE, ECAD, LCAM, UVO

Cannot extract interactions: [Interaction:2161775] MAG, GMA, S-MAG, SIGLEC-4A, SIGLEC4A-MAG, GMA, S-MAG, SIGLEC-4A, SIGLEC4A

Cannot extract interactions: [Interaction:2159845] NVL-RNR2, MTRNR2, MT-RNR2

……

Total empty interactions: 199

total interactions: 193901

Total time to extract: 125578

……

Note: You may see many lines saying “Cannot extract interactions…”. Since our FIs are protein centric around UniProt ids, if an EWAS has no referenceEntity to a UniProt id, no interaction will be extracted though it may have gene names there.

* 1. Dump predicted FIs into the database by running method, HibernateFIWriter.dumpPredicted():

Total predicted FIs: 67972

……

Total protein from db: 9609

Total FIs will be added to the FI database: 67892

Save proteins: 5275

Save evidences: 67892

* 1. Generate a FI files using protein or gene names by running method, HibernateFIReader. generateFIFileInGeneInHibernate(). Before running this method, make sure the following constants have been set correctly in R3Constants:

**public** **static** **final** String *GENE\_FI\_FILE\_NAME* = *RESULT\_DIR* + "FIsInGene\_031612.txt";

**public** **static** **final** String *GENE\_FI\_PATHWAY\_FILE\_NAME* = *RESULT\_DIR* + "FIsInGene\_Pathway\_031612.txt";

**public** **static** **final** String *GENE\_FI\_PREDICTED\_FILE\_NAME* = *RESULT\_DIR* + "FIsInGene\_Predicted\_031612.txt";

The output from the method running should be similar to the following:

Total interactions from prediction: 67892

Total interactions from pathways: 193901

Time for getting interactions: 5594

Total predicted FIs: 38450

Total pathway FIs:135432

Total FIs: 172235

Total predicted proteins: 8102

Total pathway proteins: 7402

Total proteins: 10696

*======The END======*