Tox21Enricher-Shiny
Setup Guide

(Last Updated 6/29/2022)

http://hurlab.med.und.edu/Tox21Enricher

Junguk Hur, Ph.D.
Department of Biomedical Sciences
University of North Dakota, School of Medicine and Health Sciences
Grand Forks, North Dakota 58202, USA
Email: InformaticsTools@gmail.com
http://hurlab.med.und.edu

Table of Contents

I. Introduction to Tox21Enricher-Shiny

Humans are exposed to tens of thousands of chemicals that are used in daily life, some at levels that may pose a health risk. There is limited toxicological information for many of these chemicals, which makes risk assessment difficult or impossible. The United States Toxicology Testing in the 21st Century (Tox21) program was established to develop more efficient and human-relevant toxicity assessment methods. The Tox21 program is currently screening a set of over 10,000 chemicals, the Tox21 10K library, using quantitative high-throughput screening (qHTS) of assays that measure effects on toxicity pathways. To date, more than 70 assays have yielded >12 million concentration-response curves by Tox21 researchers. To efficiently apply these data for identifying potential hazardous compounds and for informing further toxicological testing, the United States National Toxicology Program (NTP) has developed several web applications (Tox21 Toolbox: **http://ntp.niehs.nih.gov/tbox/**), including tools for data visualization (Tox21 Curve Browser) and exploration (Tox21 Activity Profiler).

One critical usage of this dataset is to perform chemical-relational analysis based on the patterns of activity across the Tox21 assays and then to use nearest neighbor-based prediction to infer the toxicological properties of untested chemicals via their association with tested chemicals. One approach to inferring the specific properties is to perform chemical annotation enrichment of chemical neighborhoods.

Here, we present Tox21Enricher-Shiny, a web-based chemical annotation enrichment tool for Tox21 assay data built using the R Shiny framework. Tox21Enricher-Shiny identifies significantly over-represented chemical biological annotations among sets of chemicals (neighborhoods), which facilitates the identification of the toxicological properties and mechanisms in the chemical sets.

II. Setup

II.1. Getting a Copy of Tox21Enricher-Shiny

The Tox21Enricher-Shiny code for both the server-side and client-side applications may be found at https://github.com/hurlab/tox21enricher. Assuming you have cloned the repository into a folder called "/home/user/tox21enricher/" on your machine, server-side code can be found in `/home/user/tox21enricher/` and client-side code can be found in `/home/user/tox21enricher/tox21enricher/`.

II.2. Server Application Setup

There are three main components to the Tox21Enricher-Shiny server-side utilities: the Plumber API, the request queue, and the database. This installation guide assumes you are using a *nix environment, preferably Ubuntu 20.04 LTE.

II.2.a. Server Configuration

Assuming you have cloned the code to /home/user/tox21enricher/, the server-side code's configuration file can be found at /home/user/tox21enricher/config.yml. This configuration file has two namespaces, `tox21enricher` and `tox21enricher-queue`, for the **tox21enricher database** and the **tox21queue database**, respectively.

The importance of each of the variables in this configuration file is explained below:

- **driver** – The database driver. This should always be set to `"Postgres"`.
- **server** – The location of the Tox21Enricher-Shiny database in the form `<host_ip>:<port>/<database_name>`. For example, if you are hosting the database on the same machine at the default Postgres port of 5432 with a database name of "tox21enricher", this should be set to `"127.0.0.1:5432/tox21enricher"`. For the tox21queue database, this should be `"127.0.0.1:5432/tox21queue"`.
- **host** – The name or IP address of the machine where the database is hosted. If you are hosting the database on the same machine, this should be set to `"127.0.0.1"` or `"localhost"`.
- **uid** – The username of the Postgres user that owns the tox21enricher database for the tox21enricher namespace / the username of the Postgres user that owns the tox21queue database for the tox21enricher-queue namespace.
- **pwd** – The password of the Postgres user that owns the tox21enricher database for the tox21enricher namespace / the password of the Postgres user that owns the tox21queue database for the tox21enricher-queue namespace.
- **port** – The port that the database is listening on. If you set up PostgreSQL to use the default port, this should be set to `"5432"`.
- **database** – The name of the Tox21Enricher-Shiny database on the Postgres server. This will be `"tox21enricher"` for the tox21enricher namespace and `"tox21queue"` for the tox21enricher-queue namespace.

Unique to **tox21enricher** namespace:

- **appversion** – The current version of the application. This is used to know which version of the Tox21Enricher-Shiny manual to serve to users.
- **appdir** – The location of the server-side project files. This should be wherever you cloned the repository to (i.e., `"/home/user/tox21enricher/"`).
- **indir** – The location of the directory where input files from requests should be stored.
- **outdir** – The location of the directory where result (output) files from requests should be stored.
- **archivedir** – The location of the directory where the zipped archives from old requests should be saved after they can no longer be accessed from the application.
- **cores** – The number of cores available to use when using multicore processing. This is initially set to 4.
- **apiHost** – The hostname or IP address where the API will be hosted.
- **apiPort** – The port on the host where the API will be hosted. If left as the empty string `""`, Tox21Enricher-Shiny will assume the default is used (port 80 for HTTP or port 443 for HTTPS) and omit the port from any displays of the API's address.
- **apiSecure** – `true` or `false` value. Should be `true` if the host where the API is located is using HTTPS and `false` if it is using HTTP.

Unique to **tox21enricher-queue** namespace:

- **cleanupTime** – The number of hours a transaction will be kept alive in the queue before being cancelled and marked for deletion. Also, the number of hours until the expiration of a cookie created on the client-side that stores a request's UUID.
- **deleteTime** – The number of days before an old transaction is archived.
- **inputMax** – The number of concurrent sets that may be accepted as part of a single request. This value *n* must be 1 <= *n* <= 16. If More than 16 sets are specified, the application will set this value to 16. If less than 1 set is specified, the application will set this value to 1.
- **apikey** – Your API key needed to access the ipstack API to fetch user geolocation data (https://ipstack.com)

II.2.b. Setting Up the Plumber API

The Plumber API serves as the primary point of contact between the Tox21Enricher-Shiny client application and the other components of the Tox21Enricher-Shiny server-side utilities. Requests to perform enrichment and to access the application's database are processed through the Plumber API via HTTP requests. The Plumber API should ideally run as a daemon on the same machine as the queue and the database.

To set up the Plumber API:

1. If they do not already exist, create input and output directories in the project's root directory. Assuming you have cloned the code to /home/user/tox21enricher, these directories should be created at /home/user/tox21enricher/<input> and /home/user/tox21enricher/<output>. If these directories already exist when you cloned the project, delete the README.md files in them.
2. Download and install R from the appropriate CRAN mirror: https://cran.r-project.org/mirrors.html.
3. Install a few necessary packages using `apt-get update && apt-get install`:
    - curl, libcurl4-openssl-dev, and libxml2-dev (all needed for cURL)
    - libbz2-dev
    - libpcre2-dev
    - libpq-dev
    - libssl-dev
    - libz-dev
    - r-base-dev
4. Use `install.packages('package_name',dependencies=TRUE)` to install the necessary R libraries for the API:
    - config
    - data.table
    - DBI
    - future
    - ggplot2
    - httr
    - parallel
    - plumber
    - plyr
    - pool
    - promises
    - rjson
    - RPostgres
    - stringr
    - tidyverse
    - utils
    - uuid
5. Ensure you have properly updated the server's configuration file.
6. Ensure the Tox21Enricher-Shiny PostgreSQL server is running and listening on an open port.

7. Open the startPlumberAPI.sh script in a text editor and set `port=9000` to whatever port you want the API to listen on.
8. Ensure the startPlumberAPI.sh script has executable permissions by running `chmod +x /home/user/tox21enricher.startPlumberAPI.sh`.
9. Run the API with the command `/home/user/tox21enricher.startPlumberAPI.sh`.

II.2.c. API Demos

Examples for API usage relating to the headless submission of requests, can be found in the `<project_root>/demos/` directory.

II.2.d. Setting Up the Database

Tox21Enricher-Shiny uses a PostgreSQL database to store chemical and annotation data. Specifically, the application uses the PostgreSQL cartridge for RDKit (https://www.rdkit.org/docs/index.html).

To install RDKit and the PostgreSQL database:

1. Download and install RDKit, following the directions at https://www.rdkit.org/docs/Install.html.
2. This will require you to download and install the Conda package manager and Python.
3. Install and initialize the RDKit PostgreSQL cartridge, following the directions at https://www.rdkit.org/docs/Install.html#installing-and-using-postgresql-and-the-rdkit-postgresql-cartridge-from-a-conda-environment.
4. Make sure the PostgreSQL server is running by using `<conda_folder>/envs/my-rdkit-env/bin/pg_ctl -D /<data_folder>/`.
5. Obtain a backup or copy of the Tox21Enricher-Shiny database. Instructions for setting up the database from scratch can be found in **Section IV**.
6. Create a schema called “tox21enricher” and either:
    a. Log into it by using `<conda_folder>/envs/my-rdkit-env/bin/psql tox21enricher`. Run the command `\i /<path_to_database_backup>/` from within Postgres to restore the database from the backup.
    b. Use the pg_restore utility to restore the database from backup by running from the command line using `<conda_folder>/envs/my-rdkit-env/bin/pg_restore /<path_to_database_backup>/`.
7. Obtain a backup or copy of the Tox21Enricher-Shiny queue database. Instructions for setting up the database from scratch can be found in **Section IV**.
8. Create a schema called “tox21queue” and either:
    a. Log into it by using `<conda_folder>/envs/my-rdkit-env/bin/psql tox21queue`. Run the command `\i /<path_to_database_backup>/` from within Postgres to restore the database from the backup.
    b. Use the pg_restore utility to restore the database from backup by running from the command line using `<conda_folder>/envs/my-rdkit-env/bin/pg_restore /<path_to_database_backup>/`.

II.2.e. Other Server-side Utilities

Tox21Enricher-Shiny also features additional scripts that handle some quality-of-life features to help make maintaining the application easier.

- **queueCleanup.R**
  - This script will run indefinitely once started. It will continually check if the age of requests in the database exceeds the number of days defined as the **deleteTime** variable in the config.yml file. Requests that exceed this value are considered "old" and will have a "delete" flag set to 1 in the **transaction** table in the **tox21queue** database. The result files on the host filesystem for these old requests will be deleted, and a zipped archive of the result files will be moved to the directory defined as the **archivedir** variable in the config.yml file.
  - Failure to properly define the correct location of the output directory may result in the unintended deletion of data.
  - Passing a value of "" (empty string) to the **archivedir** variable will disable the archive feature of the cleanup script. All deleted result files will be permanently deleted.
  - The script will also continually check if any request has remained active in the queue for longer than the time defined as the **cleanupTime** variable in the config.yml file. It will automatically cancel those requests that have remained in the queue for too long.
  - It is recommended that the server host run this script continually in the background to ensure that the filesystem does not become overpopulated with old files and to help prevent congestion of the queue.
  - This script may also be run with the included executable, `./startCleanup.sh`.
- **fetchUserData.R**
  - This script will fetch from the database all logged IP addresses that are not already in the **user_data** table in the **tox21queue** database. It will then query the ipstack API (https://ipstack.com/) for each unique, non-loopback IP address to retrieve geolocation data for the address.
  - To use this script, you must supply a working, valid API key for ipstack in the config.yml file as the variable **apikey**.
  - It is recommended that the server host run this script one or a few times a month.
  - This script may also be run with the included executable, `./startFetchUserData.sh`.

II.3. Client Application Setup

The Tox21Enricher-Shiny client application provides a graphical user interface for performing enrichment analysis and interacting with the data stored in the Tox21Enricher-Shiny database. The client application may be run on the same machine as the server-side utilities and exposed to the internet so that other remote clients may use the application via a web browser. Otherwise, the client application may also be run directly on the client machine as a standalone executable and connect remotely to the server utilities running on a remote host. To access the Tox21Enricher-Shiny client application via an internet browser, your browser must have JavaScript enabled for the application to load properly.

II.3.a. Client Configuration

Assuming you have cloned the code to /home/user/tox21enricher/, the server-side code's configuration file can be found at /home/user/tox21enricher/tox21enricher/config.yml. This configuration file is notably simpler than the configuration file used for the Tox21Enricher-Shiny server. The client configuration file has only one namespace, `tox21enricher-client`.

The importance of the variables in this configuration file is explained below:

- **host** – The name or IP address of the machine where the Tox21Enricher-Shiny Plumber API is hosted. If you are hosting the API on the same machine, this should be set to `"127.0.0.1"` or `"localhost"`.
- **port** – The port that the Tox21Enricher-Shiny Plumber API is listening on. By default, this is set to `"9000"`. If port is unknown or not specified (i.e., if the API is located at an address like this: `hostname/subdir1/subdir2`), set to `""` (empty string).
- **secure** – Should be `true` or `false` (without any quotation marks). Set to `true` if the API address uses HTTPS. Set to `false` if it uses HTTP instead.
- **mode** – The mode of operation that the client application will use. "server" should be used if you are intending to host the client application on a web server that can be accessed by multiple users via a web browser. "client" should be used if you are intending to use the client application for only yourself. Supplying anything other than "server" or "client" will cause the application to default to "client" mode upon loading.

II.3.b. Setting Up the Client Application

The Tox21Enricher-Shiny client application is a Shiny (https://www.rstudio.com/products/shiny/) application and thus can be hosted on a machine and accessed from a remote client via an internet browser.
To set up the client application as a web application:

1. If they do not already exist, create the `<project_root>/tox21enricher/www/docs` and `<project_root>/tox21enricher/www/local/`. Assuming you have cloned the code to /home/user/tox21enricher, these directories should be created at /home/user/tox21enricher/tox21enricher/www/. If these directories already exist when you cloned the project, delete the README.md files in them.
2. Download and install R from the appropriate CRAN mirror: https://cran.r-

project.org/mirrors.html.

3. Install a few necessary packages using `apt-get update && apt-get install`:
    - libbz2-dev
    - libgdal-dev
    - libpcre2-dev
    - libudunits2-dev
    - libz-dev
    - r-base-dev
4. Use `install.packages('package_name',dependencies=TRUE)` to install the necessary R libraries for the client application:
    - bslib
    - catmaply
    - config
    - dplyr
    - DT
    - ggVennDiagram
    - httr
    - igraph
    - igraphdata
    - plotly
    - rclipboard
    - rjson
    - shiny
    - shinyBS
    - shinycssloaders
    - shinydashboard
    - shinyjs
    - shinythemes
    - stringr
    - uuid
    - visNetwork
5. Ensure you have properly updated the client application's configuration file.
6. Launch the application according to the **Running in a Separate Process** section in the instructions here: https://shiny.rstudio.com/articles/running.html.
7. Open the port that the client application runs on. By default, this is a random port, but you can configure the application to run on the same port every time by specifying a specific port for the `port` argument of the `shiny::runApp()` function as described in the documentation here: https://shiny.rstudio.com/reference/shiny/1.0.1/runApp.html.

III. Deployment

The server components of Tox21Enricher-Shiny may be deployed onto an RStudio Connect server (if in a commercial/organizational environment) or onto a Shiny Server (if in a personal environment). The client application can be deployed in this manner too if you are running the client application as a public-facing web application.

It is not necessary to make the PostgreSQL server's port publicly available (in fact, it is not recommended). The Plumber API's port must be open if the client application is running as a remote client and not as a web application. If the API's port is not public-facing, then direct API querying and remote client application access will not work properly.

If running the client application as a web application, its port must be publicly available.

IV. Database Specifications and Maintenance

This section will detail the specifications of the Tox21Enricher-Shiny database. It will also explain how to periodically update the data stored in the database.

IV.1. Building the Database from Scratch

The Tox21Enricher-Shiny database is made up of two schemas: **tox21enricher** and **tox21queue**. **tox21enricher** stores read-only data relating to the chemicals in the Tox21 dataset, while **tox21queue** contains a few writable tables that store information about submitted requests.

To build the Tox21Enricher-Shiny database from scratch, follow these specifications:

1. First, create the role that will own the **tox21enricher** database.
2. Next, create the tox21enricher database and set the owner to the user created in step 1.
3. Create the following tables and functions:
    a. annotation_class

Name	Type	Constraints
annoclassid	INTEGER	NOT NULL, AUTO INCREMENT, PRIMARY KEY
annoclassname	VARCHAR(50)	NOT NULL
firsttermid	INTEGER	NOT NULL
lasttermid	INTEGER	NOT NULL
numberoftermids	INTEGER	NOT NULL
baseurl	VARCHAR(255)	NOT NULL
annotype	VARCHAR(100)	
annogroovyclassname	VARCHAR(100)	
annodesc	VARCHAR(7000)	
networkcolor	VARCHAR(11)	

    b. annotation_detail

Name	Type	Constraints
annotermid	INTEGER	NOT NULL, AUTO INCREMENT, PRIMARY KEY
annoclassid	INTEGER	NOT NULL, FOREIGN KEY REFERENCES annotation_class(annoclassid)
annoterm	VARCHAR(7000)	NOT NULL

    c. annotation_matrix

Name	Type	Constraints
casrn	VARCHAR(15)	NOT NULL
name	VARCHAR(500)	NOT NULL
annotation	VARCHAR(40000)	

    d. annotation_matrix_terms

Name	Type	Constraints
id	INTEGER	

term	VARCHAR(7000)	

e. annoterm_pairwise

Name	Type	Constraints
pairwise	INTEGER	NOT NULL, AUTO INCREMENT, PRIMARY KEY
term1uid	INTEGER	NOT NULL
term2uid	INTEGER	NOT NULL
term1size	INTEGER	NOT NULL
term2size	INTEGER	NOT NULL
common	INTEGER	NOT NULL
union	INTEGER	NOT NULL
jaccardindex	DOUBLE	NOT NULL
pvalue	DOUBLE	NOT NULL
qvalue	DOUBLE	NOT NULL

f. chemical_detail

Name	Type	Constraints
casrnuid	INTEGER	NOT NULL, AUTO INCREMENT, PRIMARY KEY
casrn	VARCHAR(15)	NOT NULL
testsubstance_chemname	VARCHAR(500)	NOT NULL
molecular_formular	VARCHAR(50)	
iupac_name	VARCHAR(1000)	
inchis	VARCHAR(1000)	
inchikey	VARCHAR(30)	
smiles	VARCHAR(600)	
smiles_qsar_ready	VARCHAR(600)	
dtxsid	VARCHAR(20)	
stoichiometric_ratio	VARCHAR(10)	
cid	VARCHAR(20)	
mol_formula	VARCHAR(200)	
mol_weight	DOUBLE	
dtxrid	VARCHAR(20)	

g. fps_2

Name	Type	Constraints
casrn	VARCHAR(15)	
torsionbv	BFP	
mfp2	BFP	
ffp2	BFP	

h. mols_2

Name	Type	Constraints
casrn	VARCHAR(15)	
m	MOL	
cyanide	INTEGER	
isocyanate	INTEGER	
aldehyde	INTEGER	

epoxide	INTEGER	

i. term2casrn_mapping

Name	Type	Constraints
term2casrnmappinguid	INTEGER	NOT NULL, AUTO INCREMENT, PRIMARY KEY
annotermid	INTEGER	NOT NULL
annoclassid	INTEGER	NOT NULL
casrnuid_id	INTEGER	NOT NULL

j. get_mfp2_neighbors

Description

```
CREATE OR REPLACE FUNCTION public.get_mfp2_neighbors(smiles text)
 RETURNS TABLE(casrn character varying, m character varying, similarity
double precision, cyanide integer, isocyanate integer, aldehyde integer,
epoxide integer)
 LANGUAGE sql
 STABLE
AS $function$
select casrn,mol_to_smiles(m)::character
varying,tanimoto_sml(morganbv_fp(mol_from_smiles($1::cstring)),mfp2) as
similarity,cyanide,isocyanate,aldehyde,epoxide
from fps_2 join mols_2 using (casrn)
where morganbv_fp(mol_from_smiles($1::cstring))%mfp2
order by morganbv_fp(mol_from_smiles($1::cstring))<%>mfp2;
$function$
```

4. Revoke all privileges on every table in this database for the user created in step 1.
5. Grant only the SELECT privilege on each table in the database for the user created in step 1.
6. Next, create the role that will own the **tox21queue** database.
7. Next, create the **tox21queue** database and set the owner to the user created in the previous step.

8. Create the following tables and functions:
    a. queue

Name	Type	Constraints
mode	VARCHAR(15)	
uuid	VARCHAR(50)	NOT NULL, PRIMARY KEY
annoselectstr	VARCHAR(1500)	
cutoff	INTEGER	
finished	INTEGER	DEFAULT 0
index	INTEGER	NOT NULL, AUTO INCREMENT
error	VARCHAR(500)	
cancel	INTEGER	NOT NULL, DEFAULT 0
lock	INTEGER	NOT NULL, DEFAULT 0

    b. status

Name	Type	Constraints
step	INTEGER	NOT NULL, DEFAULT 0
uuid	VARCHAR(50)	NOT NULL, FOREIGN KEY REFERENCES queue(uuid), ON DELETE CASCADE
setname	VARCHAR(500)	

    c. transaction

Name	Type	Constraints
original_mode	VARCHAR(15)	
mode	VARCHAR(15)	NOT NULL
uuid	VARCHAR(40)	NOT NULL, PRIMARY KEY
annotation_selection_string	VARCHAR(1000)	
cutoff	INTEGER	NOT NULL
input	VARCHAR	
original_names	VARCHAR	
reenrich	VARCHAR	
colors	VARCHAR	
timestamp_posted	VARCHAR(50)	NOT NULL, DEFAULT CURRENT_DATE
timestamp_started	VARCHAR(50)	DEFAULT 'not started'
timestamp_finished	VARCHAR(50)	DEFAULT 'incomplete'
cancel	INTEGER	NOT NULL, DEFAULT 0
reenrich_flag	INTEGER	NOT NULL, DEFAULT 0
casrn_box	VARCHAR	
delete	INTEGER	NOT NULL, DEFAULT 0
ip	VARCHAR	

d. user_data

Name	Type	Constraints
ip	VARCHAR	NOT NULL, PRIMARY KEY
type	VARCHAR	
continent	VARCHAR	
country	VARCHAR	
region	VARCHAR	
city	VARCHAR	
zip	VARCHAR	
is_proxy	INTEGER	
proxy_type	VARCHAR	
is_crawler	INTEGER	
crawler_name	VARCHAR	
crawler_type	VARCHAR	
is_tor	INTEGER	
threat_level	VARCHAR	
threat_types	VARCHAR	

9. Revoke all privileges on every table in this database for the queue-owning user.
10. Grant only the SELECT, INSERT, and UPDATE privileges on each table in the database for the queue-owning user.

IV.2. Updating the Database

It will sometimes be necessary to update the database used by Tox21Enricher-Shiny. Included in the application's GitHub repository under the `/<project_root>/tox21enricher-database/update/` directory are several scripts that must be run to properly update the database.

The following instructions must be run to ensure the database update proceeds smoothly.

1. Navigate to the `/<project_root>/tox21enricher-database/update/` folder.
2. Make sure that three subdirectories, **Annotation**, **ReOrganized**, and **tmp**, all exist in this directory. If they do not, create them. **Note the capitalization of the names**.
3. Make sure the files **Tox21_CASRN_Names.anno** and **Tox21_CASRN_Names_Full.anno** exist in the **Annotation** folder. They should already be in this folder if you cloned the project from the GitHub repository.
4. Place your annotation files in the **Annotation** folder. There are a few specifications these files need to follow:
    a. You must place ALL annotation files in the **Annotation** folder, not just the new/updated files.
    b. Each line must be structured like this:
    `<CASRN> semicolon and; space; separated; list; of; annotations`
    Note that the CASRN and annotations should be tab-separated, and each annotation should be separated by a semicolon AND a space. The last annotation on each line should be followed immediately by the newline character ("\n") and NO semicolon.
    c. Each file should be named like this:
    `DrugMatrix_EXAMPLE_NAME.txt`
    Note that each file must be prefixed with "DrugMatrix_", class names should be in all capital letters, and multiple-word names must be split by single underscores ("_").
5. Run the first four scripts using Perl. Note that some or all of these scripts may take hours to days to complete, depending on processing speed, memory, and the number of cores available.
    a. Run the **first** script to reorganize the annotation data using the command `perl 01_ReOrganize_Annotation_Data_v2.1-20161003.pl`.
    b. Run the **second** script to perform the Fisher exact test and create most of the database files using the command `perl 02_Split-perform_Fisher-Exact-Test.pl`.
    c. Run the **third** script to create the final pairwise calculation file using the command `perl 03_Merge_Pvalues-and-calculate-qvalues.pl`.
    d. Run the **fourth** script to create the initial annotation matrix files using the command `perl 04_Create_complete_annotation_matrix_v1.0_11182020.pl`.
6. Run the next two scripts using Python.
    a. Run the **fifth** script to create the file **annotation-matrix-names.sql** which should store the names and IDs of each annotation in the matrix using the

command `python 05_Create_annotation-matrix-names.py`.
    b. Run the **sixth** script to create the file **newmatrix.sql** which should store the new annotation matrix using the command `python 06_Create_annotation-matrix.py`.
7. The **seventh** script is optional and should only be run if you are updating the **mols_2** table. In most cases, you can ignore this script.
8. Before you can import the generated text files into the PostgreSQL database, a little bit of formatting needs to be done:
    a. The headers (first line) of each file must be deleted. The easiest way to do this is by running the **remove_headers.sh** bash script. Ensure this script has execute permissions before running.
    b. The table_annotation_class_v2.1.txt file needs to be padded with placeholder data that will be overwritten with a later script. To do this, run the **pad_annotation_class.py** script with Python.
9. You should now be able to import the files into the database. First, ensure that all database tables exist. They should be:
    a. annotation_class
    b. annotation_detail
    c. annotation_matrix
    d. annotation_matrix_terms
    e. annoterm_pairwise
    f. chemical_detail
    g. fps_2
    h. mols_2
    i. term2casrn_mapping

    The *italicized* tables should not have to be updated.
10. You can use the PostgreSQL command `COPY <table_name> FROM '/path_to_file/file.txt';` to fill each table with the data from the respective text files. You should only copy from the files:
    a. table_annotation_class_v2.1.txt
    b. table_annotation_detail_v2.1.txt
    c. table_annoterm_pairwise_v2.1_FINAL.txt
    d. table_chemical_detail_v2.1.txt
    e. term2casrn_mapping_v2.1.txt
11. After filling in the above tables, run the eighth script, `08_Add_additional_info_annotation_class.sql`, with PostgreSQL to finish filling in the annotation_class table.
12. To fill the annotation_matrix and annotation_matrix_terms tables, run the **annotation-matrix-names.sql** and **newmatrix.sql** scripts.

V. Common Issues

Both the client and server code for Tox21Enricher-Shiny are still in continued development, so there are likely issues that may arise when trying to either perform certain tasks or set up the application. Known bugs in the code and planned features are documented on the project's GitHub repository at: https://github.com/hurlab/tox21enricher/issues. Other known issues not necessarily related to the code itself are documented in this section of the manual.

V.1. Setup Issues

These are known issues that you may experience when attempting to set up Tox21Enricher-Shiny.

- **Problem installing RDKit PostgreSQL cartridge** – Even if you successfully install RDKit using Conda, Conda may fail to install the RDKit PostgreSQL cartridge due to package version conflicts. A workaround for this is to create a new Conda environment using `conda create -n <environment_name>`, delete or move the contents of the new environment's `/bin/` directory, install the RDKit PostgreSQL cartridge to that environment, then finally move the newly installed files from the new environment's `/bin/` directory into the RDKit environment's /bin/ directory.

END OF THE SETUP GUIDE