**Documentation**

# System Prerequisites

1. Windows. This process was developed and executed on Windows 10 Pro.
2. OHDSI Usagi mapping tool. For this process, we used Usagi v1.2.7.
3. PostgreSQL database. Database storage and operations were enabled using PostgreSQL 9.3.
4. Pgadmin III PostgreSQL SQL client. Use this client to run the SQL scripts.
5. Python. Python scripts were executed on Python 3.6.
6. RStudio. R scripts were executed on R version 4.0.0.
7. SAS. SAS scripts were executed on SAS 9.
8. MATLAB. MATLAB scripts were executed on MATLAB R2020b.

# OHDSI CDM Vocabulary Schema

Load the OHDSI CDMV5 Vocabulary tables into a separate PostgreSQL schema called *cdmv5*. The vocabulary data files, and the PostgreSQL table load scripts are available for download from the Athena website: <https://athena.ohdsi.org/search-terms/start> (Download tab). On the download page keep the pre-selected vocabularies and also select the MedDRA vocabulary (review the EULA link for MedDRA).

# Set working directory

Select **CRESCENDDI-main/** folder as the working directory in Python, RStudio, SAS and MATLAB.

# DDI Reference Set

## Web Data Extraction

* Run **bnf\_ddi\_extraction.py**, **micromedex\_ddi\_extraction.py** and **thesaurus\_ddi\_extraction.R** scripts to get DDI data from the original sources.
* Create an empty schema called *drug\_interaction\_compendia* on Pgadmin III.

### BNF

* Create a table named *bnf\_data* with 5 columns: drugname\_1\_original (varchar), drugname\_2\_original (varchar), description (text), severity (text), evidence (text).
* Import BNF scraped data from BNF Interactions website (as of November 2018) to *bnf\_data* table using the file **bnf\_ddi\_data.csv** (PgAdmin III wizard).

### Thesaurus

* Create a table named *thesaurus\_data* with 4 columns: drug\_1\_original (varchar), drug\_2\_original (varchar), description (text), severity (integer).
* Import Thesaurus data (as of September 2019) to *thesaurus\_data* table using the file **thesaurus\_ddi\_data.csv** (PgAdmin III wizard).

### Micromedex

* Create a table named *micromedex\_data* with 5 columns: drugname\_1\_original (varchar), drugname\_2\_original (varchar), description (text), severity (text), evidence (text).
* Import Micromedex scraped data from IBM Micromedex website (as of September 2018) to *micromedex\_data* table using the file **micromedex\_ddi\_data.csv** (PgAdmin III wizard).

## Drug Name Mapping

### BNF

1. Create a mapping table (drug name to RxNorm and RxNormExtension concepts) named *bnf\_drug\_usagi\_mapping* by running the **create\_bnf\_drug\_usagi\_mapping.sql** script.
2. Import drug mappings created with USAGI to *bnf\_drug\_usagi\_mapping* table using **bnf\_drugnames.csv** (PgAdmin III wizard).
3. Run the **map\_bnf\_drugnames.sql** script to map all drug concepts from BNF to an RxNorm/RxNormExtension concept (if available on the mapping table) to a new table named *bnf\_with\_mapped\_drugnames*. This table also contains a column “flag” that indicates the numerical order of the two drug concept IDs (flag = 1 **if drugname\_1\_concept\_id < drugname\_2\_concept\_id**, flag = 2 **if drugname\_1\_concept\_id > drugname\_2\_concept\_id**, flag missing in cases where **drugname\_1\_concept\_id = drugname\_2\_concept\_id**) and another column “ordered\_drug\_list” containing the drug pair concept IDs in ascending order, separated by ‘|’.
4. Create a table named *bnf\_drug\_pairs* that only contains drug pairs with their drug names and mapped concepts from BNF by running the **generate\_drug\_pairs\_bnf.sql** script.

### Thesaurus

1. Create a mapping table (drug name to RxNorm and RxNormExtension concepts) named *thesaurus\_drug\_usagi\_mapping* by running the **create\_thesaurus\_drug\_usagi\_mapping.sql** script.
2. Import drug mappings created with USAGI to *thesaurus\_drug\_usagi\_mapping* table using **thesaurus\_drugnames.csv** (PgAdmin III wizard).
3. Run the **map\_thesaurus\_drugnames.sql** script to map all drug concepts from Thesaurus to an RxNorm/RxNormExtension concept (if available on the mapping table) to a new table named *thesaurus\_with\_mapped\_drugnames*. This table also contains a column “flag” that indicates the numerical order of the two drug concept IDs (flag = 1 **if drugname\_1\_concept\_id < drugname\_2\_concept\_id**, flag = 2 **if drugname\_1\_concept\_id > drugname\_2\_concept\_id**, flag missing in cases where **drugname\_1\_concept\_id = drugname\_2\_concept\_id**) and another column “ordered\_drug\_list” containing the drug pair concept IDs in ascending order, separated by ‘|’.
4. Create a table named *thesaurus\_drug\_pairs* that only contains drug pairs with their drug names and mapped concepts from Thesaurus by running the **generate\_drug\_pairs\_thesaurus.sql** script.

### Micromedex

1. Create a mapping table (drug name to RxNorm and RxNormExtension concepts) named *micromedex\_drug\_usagi\_mapping* by running the **create\_micromedex\_drug\_usagi\_mapping.sql** script.
2. Import drug mappings created with USAGI to *micromedex\_drug\_usagi\_mapping* table using **micromedex\_drugnames.csv** (PgAdmin III wizard).
3. Run the **map\_micromedex\_drugnames.sql** script to map all drug concepts from Micromedex to an RxNorm/RxNormExtension concept (if available on the mapping table) to a new table named *micromedex\_with\_mapped\_drugnames*. This table also contains a column “flag” that indicates the numerical order of the two drug concept IDs (flag = 1 **if drugname\_1\_concept\_id < drugname\_2\_concept\_id**, flag = 2 **if drugname\_1\_concept\_id > drugname\_2\_concept\_id**, flag missing in cases where **drugname\_1\_concept\_id = drugname\_2\_concept\_id**) and another column “ordered\_drug\_list” containing the drug pair concept IDs in ascending order, separated by ‘|’.
4. Create a table named *micromedex\_drug\_pairs* that only contains drug pairs with their drug names and mapped concepts from Micromedex by running the **generate\_drug\_pairs\_micromedex.sql** script.

## Intersection of DDI online resources

* Run the **generate\_common.sql**script to create a table called *common* that contains only the common drug pairs among the three compendia with their corresponding BNF and Micromedex text descriptions (in their blinded form). The last part of the script **creates two tables** in the output pane that summarise the unique BNF and Micromedex text descriptions with their frequencies. Each table is exported to a .csv file named **common\_bnf\_descriptions\_blinded.csv** and **common\_micromedex\_descriptions\_blinded.csv**, respectively. Those files will be imported to USAGI for adverse event mapping.

## Adverse Event Mappings

### BNF

* 1. Create a mapping table (text description to MedDRA LLT/PT concepts) named *bnf\_event\_usagi\_mapping* by running the **create\_bnf\_event\_usagi\_mapping.sql** script.
  2. Import event mappings created with USAGI to *bnf\_event\_usagi\_mapping* table using **bnf\_eventnames.csv** (PgAdmin III wizard).

### Micromedex

* + 1. Create a mapping table (text description to MedDRA LLT/PT concepts) named *micromedex\_event\_usagi\_mapping* by running the **create\_micromedex\_event\_usagi\_mapping.sql** script.
    2. Import event mappings created with USAGI to *micromedex\_event\_usagi\_mapping* table using **micromedex\_eventnames.csv** (PgAdmin III wizard).

## Positive Controls

* Run the **ddi\_positive\_controls.py** script.

## Negative Controls

* Run the **ddi\_negative\_controls.py** script.

# Single Drug Information

## Web Data Extraction

### BNF

* Run the **bnf\_adr\_extraction.py** scriptto get single-drug data for adverse drug reactions (ADRs) from BNF as a CSV file called **bnf\_single\_data.csv**.

### SIDER

* Download and unpack (if needed) the following files from the SIDER website (<http://sideeffects.embl.de/download/>).   
  Put the extracted files in the **CRESCENDDI/data\_extraction/data/SIDER/** subfolder.

## Drug Name Mapping

### BNF

* Run the **map\_bnf\_single\_drugnames.py** script.

### SIDER

* Run the **map\_sider\_drugnames.py** script.

## Adverse Event and Indication Mappings

### BNF

* Run the **map\_bnf\_single\_eventnames.py** script.

### SIDER

* Run the **map\_sider\_eventnames.py** script.

## Negative Controls

* Run the **single\_drug\_negative\_controls.py** script.

## Combine files to a single output

* Run the **single\_drug\_generate\_output.py** script to create the final Data Record.

# FAERS Screening

## Input Data

* Spreadsheets with FAERS counts.

## Algorithms for DDI Surveillance

### Omega

* Run the **omega\_calculation.py** script to get Omega scores for the DDI controls.

### delta\_add

* Run the **delta\_add\_calculation.sas** script to get delta\_add scores for the DDI controls.

### Interaction Signal Score (IntSS)

* Run the **intss\_calculation.R** script to get IntSS scores for the DDI controls.

## Algorithms for Drug-Event Pair Surveillance

* Run the **single\_drug\_sda\_calculation.R** script to get Proportional Reporting Ratio (PRR), Empirical Bayes Geometric Mean (EBGM) and Bayesian Confidence Propagation Neural Network (BCPNN) scores for the single drug controls.

## Receiver Operating Characteristic (ROC) Analysis

* Load the data files with scores as data tables to MATLAB.
* Run the **roc\_ddi.m** scriptto get Area Under the Curve (AUC) scores with 95% confidence intervals for each of the DDI surveillance algorithms.
* Run the **roc\_single\_drug.m** script to get Area Under the Curve (AUC) scores with 95% confidence intervals for each of the drug-event pair surveillance algorithms.