Code

Analysis of Occurrence Frequency and Pattern of Reported Adverse Events in select Drug Classes in the United States

[redacted]

Abstract

We present an analysis the occurrence frequency and patterns of FDA reported Adverse Drug Events (ADEs) using Reporting Odds Ratio (ROR) and Ward's method. We find ROR to be an effective method in detection of elevated reporting of ADEs in a drug. We also find that there is a strong correlation between the types of ADEs reported and the drug class despite most reactions could is not strongly specific to drug classes.

Introduction

Terminology

- Adverse Drug Event (ADE) - An adverse drug event is any undesirable experience that occurs at any dose and at any time during the use of a drug, whether or not it is considered drug related. An adverse drug event may be a serious adverse event or a non-serious adverse event. (FDA)
- Reporting Odds Ratio (ROR) - The reporting odds ratio (ROR) is a measure of the strength of association between an exposure and an outcome. It is the ratio of the odds of reporting the outcome in the presence of the exposure to the odds of reporting the outcome in the absence of the exposure. (FDA)
- Established Pharmacolog Class (EPC) - The established pharmacologic class (EPC) is a classification system that groups drugs based on their pharmacologic properties that is determined to be scientifically valid and clinically meaningful. (FDA)
    - First and Second Generation Antipsychotics (FGA, SGA) are medications used to treat schizophrenia and bipolar disorder. Well known adverse events include Neuroleptic Malignant Syndrome (NMS), sedation and weight gain, with NMS more common in FGA patients.
    - Benzodiazepines (BZD) are medications used to treat anxiety, insomnia, and seizures. Well known adverse events associated with BZDs include sedation, dizziness and drug dependence.
    - Nonsteroidal Anti-inflammatory Drugs (NSAID) are medications used to treat pain, fever, and inflammation. Well known adverse events associated with NSAIDs include gastrointestinal bleeding, renal failure and liver toxicity.
    - Beta Blockers (BB) are medications used to treat hypertension, angina, and arrhythmias. Well known adverse events associated with BBs include bradycardia, hypotension and bronchospasm.
- Pharmacovigilance is the science and activities relating to the detection, assessment, understanding and prevention of adverse effects or any other drug-related problem.

Objective

One of the most important roles of pharmacovigilance in drug safety is to detect previously unforseen ADEs based on real-world prescription data and one of the methods to do so is statistical analysis the voluntary reports of ADEs to regulatory agencies such as the FDA.

The objective of this project is analyze ADEs reported to the FDA, perform necessary cleanup and normalization using third party data such as Medicare Part D prescription claims data, and:

- Discover drug-event pairs with elevated reporting frequency using ROR analysis.

- Identify correlation patterns of which ADEs tend to be reported in the same group of drugs using hierarchical clustering analysis.

Methods

Data Acquisition

Three datasets were used in this study and joined together by the drug generic name.

Drugs at FDA

Dataset `drugsfda` was obtained from FDA published data on drugs that have been approved for use in the United States. The dataset is accessible at https://open.fda.gov/apis/drug/drugsfda/ (https://open.fda.gov/apis/drug/drugsfda/) as a JSON file. It was downloaded and transformed with `jq` for further analysis.

- The unnecessary " [EPC]" suffix in drug classification were stripped off.
- Each drug's shortest generic name and EPC classification are selected as columns.

Code

```
## successfully generated datasets/drug_epc.csv 2.488 s elapsed
```

Medicare Part D Prescription Spending

Dataset `medicare_part_d` was obtained from the Centers for Medicare and Medicaid Services (CMS) published data on Medicare Part D prescription drug plans. The dataset is accessible at https://data.cms.gov/summary-statistics-on-use-and-payments/medicare-medicaid-spending-by-drug/medicare-part-d-spending-by-drug (https://data.cms.gov/summary-statistics-on-use-and-payments/medicare-medicaid-spending-by-drug/medicare-part-d-spending-by-drug) as a single CSV file.

```
wget -O datasets/medicare/part_d/Medicare_Part_D_Spending_by_Drug_2020.csv \
 "https://data.cms.gov/sites/default/files/2022-01/bdc2d0f7-fcdb-4429-9304-f97dfdc3c7ab/
DSD_PTD_R21_P04_V21_D20_BGM.csv"
```

FDA Reported Drug Adverse Event

Dataset `drug_event` was obtained from the FDA published data on reported adverse drug events. This dataset is very large so we used the following query & shell script to download the data for drug categories and date range we are interested in.

▸ Click to show/hide URL component

▸ Click to show/hide shell script

We received a total of 423795 records. The records were further processed into CSV with `jq` :

- Reports that contain more than 10 drugs are excluded.
- The report receipt date, list of reactions, list of shortest generic name of the drugs, and the report

source qualification are selected.

This transformation is very CPU intensive so we performed the query on the server with high parrallelism and then cached the results with knitr caching options.

Code

```
## successfully generated datasets/drug_event.csv 96.562 s elapsed
```

Data Cleanup

Code

Drug Categorization

Firstly we need to add categorization to the drugs in `drug_event` and `cms_partd` datasets. We read in data from Drugs at FDA dataset, select EPCs we are interested in and add a column `epc_cate` as a factor of their respective abbreviations. The rest of the drug list is coded as "OTHER" for the `epc_cate` column.

Code

```
## Rows: 1202 Columns: 2
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (2): generic_name, epc
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Code

Medicare Part D Spending Data

We read in the CSV data, removed two summary columns, removed manufacturer-specific rows leaving only the summarized data for each drug, added a column containing the first word of the generic name and used this column to inner join with the drug classification data.

We then pivoted the columns ending with a year and normalized the column names to snake_case to create a tidy dataset. For drugs that has the same first word but different full generic names, the shortest one was selected for convenience.

In this join we lost about half of the rows because the `drugs@FDA` dataset did not have EPC classification for all the drugs approved. Since our analysis is crucially dependent on classification being available for each drug, we decide to exclude drugs that does not have an EPC classification from the analysis.

The end product is a tibble with each row representing the Medicare claims, spending and dosages dispensed in that year for a particular drug along with its EPC classification.

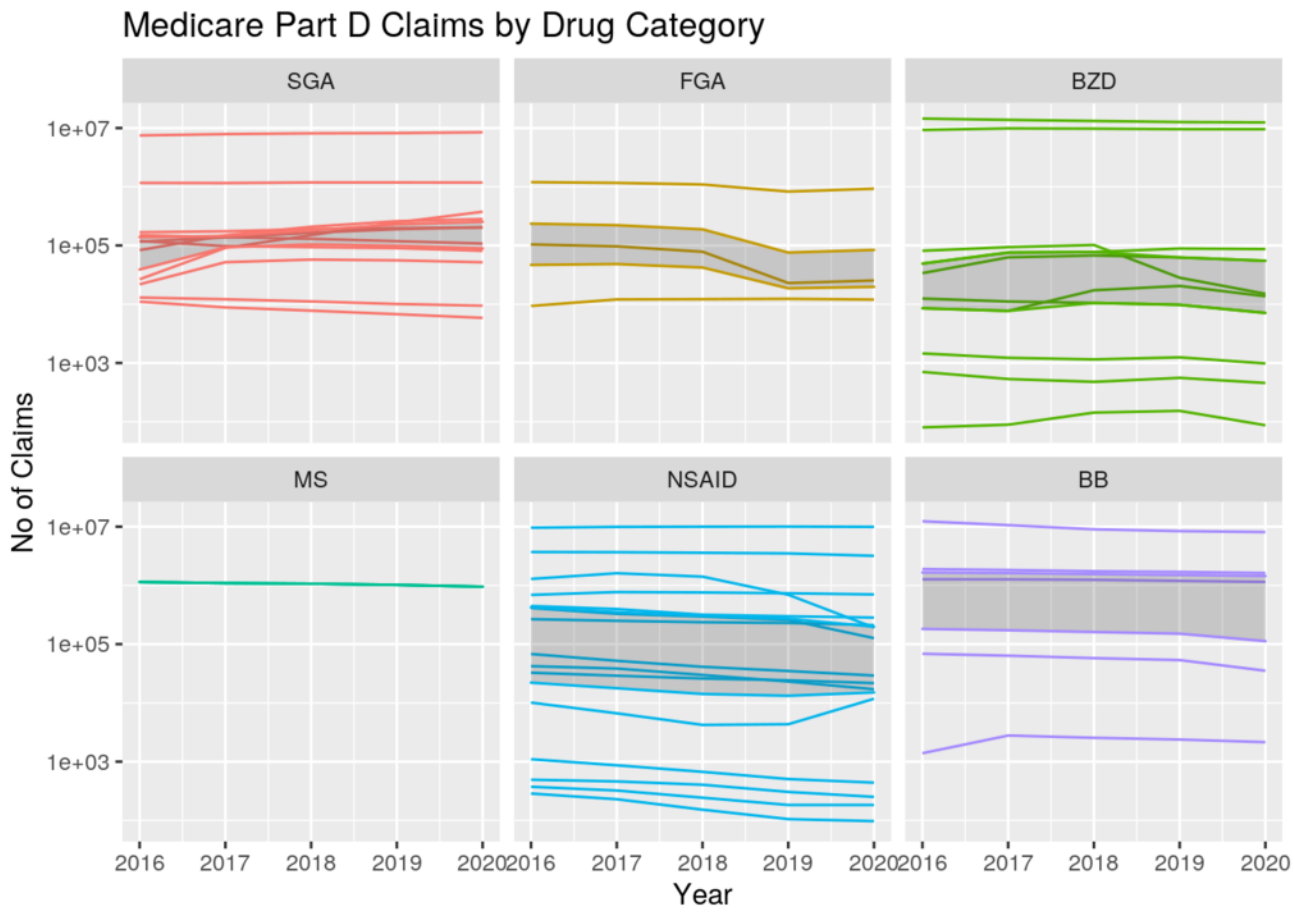
Code

```
## Rows: 13570 Columns: 46
## ── Column specification ────────────────────────────────────────────────────
## Delimiter: ","
## chr  (3): Brnd_Name, Gnrc_Name, Mftr_Name
## dbl (43): Tot_Mftr, Tot_Spndng_2016, Tot_Dsg_Unts_2016, Tot_Clms_2016, Tot_B...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
## Joining . and drug_epc by c(drug_name = "generic_name") with inner_join()
## .: 12900 -> 4945 rows
## drug_epc: 698 -> 4945 rows
## .$drug_name: na.count=0, chr [1:4945] "ARIPIPRAZOLE" "ARIPIPRAZOLE" "ARIPIPRAZOLE" "A
RIPIPRAZOLE" ...
## .$drug_full_name: na.count=0, chr [1:4945] "Aripiprazole" "Aripiprazole" "Aripiprazol
e" "Aripiprazole" ...
## .$year: na.count=0, int [1:4945] 2016 2017 2018 2019 2020 2016 2017 2018 2019 2020
...
## .$dosage_units: na.count=0, num [1:4945] 117225 138043 165452 189937 204265 ...
## .$claims: na.count=0, num [1:4945] 116208 137385 164759 189140 203218 ...
## .$spending: na.count=0, num [1:4945] 2.02e+08 2.62e+08 3.46e+08 4.22e+08 4.60e+08 ...
## drug_epc$epc: na.count=0, chr [1:4945] "Atypical Antipsychotic" "Atypical Antipsychot
ic" ...
## drug_epc$epc_cate: na.count=0, Factor w/ 7 levels "SGA","FGA","BZD",..: 1 1 1 1 1 1 1
1 1 1 ...
```

In order to verify the sensibility of the transformed data we plotted the number of claims of each drug over time, faceted by drug category. We expected each category to have comparable claims and change of spending over time. The plot seem to confirm that. We also observed that while there is big variance in the number of claims across drugs, there is no obvious difference in dosage units claimed in each drug category.

Code



FDA Reported Adverse Events

Most of our data cleaning efforts were concentrated on the FDA adverse events dataset. In short, we unnested the | concatenated columns, removed events that are obviously not drug reactions, removed events that has been reported less than 10 times over the whole dataset, recoded common synonyms into a single term, and tallied the number of reports per reaction per drug.

We then joined the dataset with the previously joined `cms_partd` dataset using the year and drug name as the ID. The end result was a dataset with the year-drug-reaction combination, count of reports, and the associated Medicare dosage units dispensed that year and the drug classification.

We further summarized the dataset, eliminating the year column by summing up the dosage units and number of reports.

Note that in this join we lost about half of the drug-event pairs, which are mainly drugs previously dropped in the previous join due to lack of EPC classification and drugs that are not covered by Medicare Part D. They are excluded from the analysis because our pipeline was not designed to handle incomplete rows.

Code

Code

```
## Joining . and cms_partd by c("year", drug = "drug_name") with inner_join()
## .: 1346001 -> 667173 rows
## cms_partd: 2275 -> 667173 rows
## .$year: na.count=0, int [1:667173] 2016 2016 2016 2016 2016 2016 2016 2016 2016 2016
...
## .$drug: na.count=0, chr [1:667173] "ABATACEPT" "ABATACEPT" "ABATACEPT" "ABATACEPT" "A
BATACEPT" ...
## .$reaction: na.count=0, chr [1:667173] "Abasia" "Abdominal discomfort" "Abdominal pai
n" ...
## .$n: na.count=0, int [1:667173] 1 1 1 1 1 5 1 1 5 1 ...
## cms_partd$drug_full_name: na.count=0, chr [1:667173] "Abatacept" "Abatacept" "Abatace
pt" "Abatacept" "Abatacept" ...
## cms_partd$dosage_units: na.count=0, num [1:667173] 206759 206759 206759 206759 206759
...
## cms_partd$claims: na.count=0, num [1:667173] 47105 47105 47105 47105 47105 ...
## cms_partd$spending: na.count=0, num [1:667173] 1.86e+08 1.86e+08 1.86e+08 1.86e+08 1.
86e+08 ...
## cms_partd$epc: na.count=0, chr [1:667173] "Selective T Cell Costimulation Modulator"
...
## cms_partd$epc_cate: na.count=0, Factor w/ 7 levels "SGA","FGA","BZD",..: 7 7 7 7 7 7
7 7 7 7 ...
```

Code

```
## `summarise()` has grouped output by 'drug'. You can override using the
## `.groups` argument.
```

Analysis

ROR

In order to detect elevated report of certain adverse events in a given drug, we used the Reporting Odds Ratio (ROR) as a measure of the increase in likelihood of a certain reported event in a drug.

The core idea of ROR is for each drug/reaction pair, we calculate the ratio of the frequency of it being reported for this drug to the frequency of it being reported in other drugs.

	Drug j	**Other Drugs**	**Total**
Event i	n_{ij}	n_{i0}	$n_{i\_}$
Other Events	n_{0j}	n_{00}	$n_{0\_}$
Total	$n_{\_j}$	$n_{\_0}$	$n_{\_\_}$

$$ROR_{ij} = \frac{OR_i}{OR_0} = \frac{\frac{n_{ij}}{n_{i0}}}{\frac{n_{0j}}{n_{00}}} = \frac{n_{ij}}{n_{i0}} \cdot \frac{n_{00}}{n_{0j}}$$

Then the variance of $\log(ROR_i)$ is defined as:

$$\mathrm{Var}\left[\log(ROR_{ij})\right] = \frac{1}{n_{ij}} + \frac{1}{n_{i0}} + \frac{1}{n_{0j}} + \frac{1}{n_{00}}$$

We provided a custom implementation based on the description by Ahmed et al. (2010) (https://onlinelibrary.wiley.com/doi/10.1111/j.1541-0420.2009.01262.x).

We implemented this instead of using existing implementations such as PhViD::ROR (Ismaïl Ahmed et al.) namely because:

- The p values are too small for machine precision so we had to write a version that works with log p values.
- The PhViD implementation cannot change the alpha value, and precisely because of the less-then epsilon p values we cannot reinterpret the CI to a diffent alpha value.

Here we called the custom ROR implementation, filtered for drug/event pairs that has at least 20 reports and 99.9% CI of log(ROR) is still greater than 0.

Note that there is difference between elevated reports and elevated occurence and causation, namely:

1. Reporters tend to report more events that are indicated in the use of the drug, such as:
    - Reporting pain when using an analgesic
    - Reporting psychological symptoms when using a psychotropic drug
2. Reporters are more likely to report events that are related to the illness, such as:
    - Reporting tumor progression and diarrhea in Ondansetron
3. Most patients were taking more than one drug at the same time, increase in ROR of certain events may be because of another drug that are likely to be taken together, such as:
    - Stomatitis, a common side effect in chemotherapy, was not directly indicated in the side effects of Afatinib despite its very high ROR

Thus for each high ROR signal it is imperative that further investigation is done to determine whether it is a genuine drug adverse effect.

Code

```
## Warning in ror(., alpha = 0.001): 5412/308642 of p values reached machine eps!
```

We made some remark on a few discoveries.

Code

▸ AFATINIB (OTHER, Kinase Inhibitor), 5 signals.

Afatinib is a tyrosine kinase inhibitor used to treat non-small cell lung cancer. One of its most common side effects is diarrhea (>90%). As expected we see that the top signal is diarrhea. Note that Stomatitis, is likely the result of concurrent chemotherapy as it was not directly indicated in the side effects of Afatinib.

▸ ALPRAZOLAM (BZD, Benzodiazepine), 114 signals.

Alprazolam (Xanax) is a benzodiazepine used to treat anxiety and panic disorder. It is one of the most commonly abused drugs in the US. As expected we Substance abuse and Withdrawl syndrome in the top signals.

▸ OXYCODONE (OTHER, Opioid Agonist), 96 signals.

Oxycodone is an opioid analgesic used to treat moderate to severe pain. Cases of fatal opioid overdose usually kills by respiratory depression. Out results corroborate with this.

▸ CLOZAPINE (SGA, Atypical Antipsychotic), 78 signals.

Clozapine is an atypical antipsychotic used to treat schizophrenia. However unlike other antipsychotics on this list there is an extremely high ROR for Granulocytopenia (marked decrease in a type of white blood cells). The FDA has issued a black box warning and requires that patients taking clozapine have their white blood cell count monitored in the REMS (https://www.newclozapinerems.com) program precisely because of this risk.

▸ HALOPERIDOL (FGA, Typical Antipsychotic), 59 signals.

Haloperidol is a typical (first generation) antipsychotic drug used to treat schizophrenia and other psychotic disorders. Compared to second generation antipsychotics, it is assodicated with a higher risk profile of neurological side effects such as neuroleptic malignant syndrome (NMS). Our research corrroborates with this.

▸ RISPERIDONE (SGA, Atypical Antipsychotic), 80 signals.

Risperidone is a second generation antipsychotic drug used to treat schizophrenia and bipolar disorder. A common side effect among antipsychotics is weight gain and abonormal blood glucose levels, which could be seen in our results.

▸ IBUPROFEN (NSAID, Nonsteroidal Anti-inflammatory Drug), 198 signals.

Ibuprofen (Advil) is a non-steroidal anti-inflammatory drug. One of the well known dangerous side effects of NSAIDs is gastrointestinal bleeding.

▸ ONDANSETRON (OTHER, Serotonin-3 Receptor Antagonist), 52 signals.

Ondansetron (Zofran) is a serotonin 5-HT3 receptor antagonist commonly used in combination with other drugs to treat chemotherapy-induced nausea and vomiting (as seen in our signals). Due to its serotonergic effects, serotonin syndrome could occur in case of overdose or drug-drug interactions.

▸ LIDOCAINE (OTHER, Amide Local Anesthetic), 40 signals.

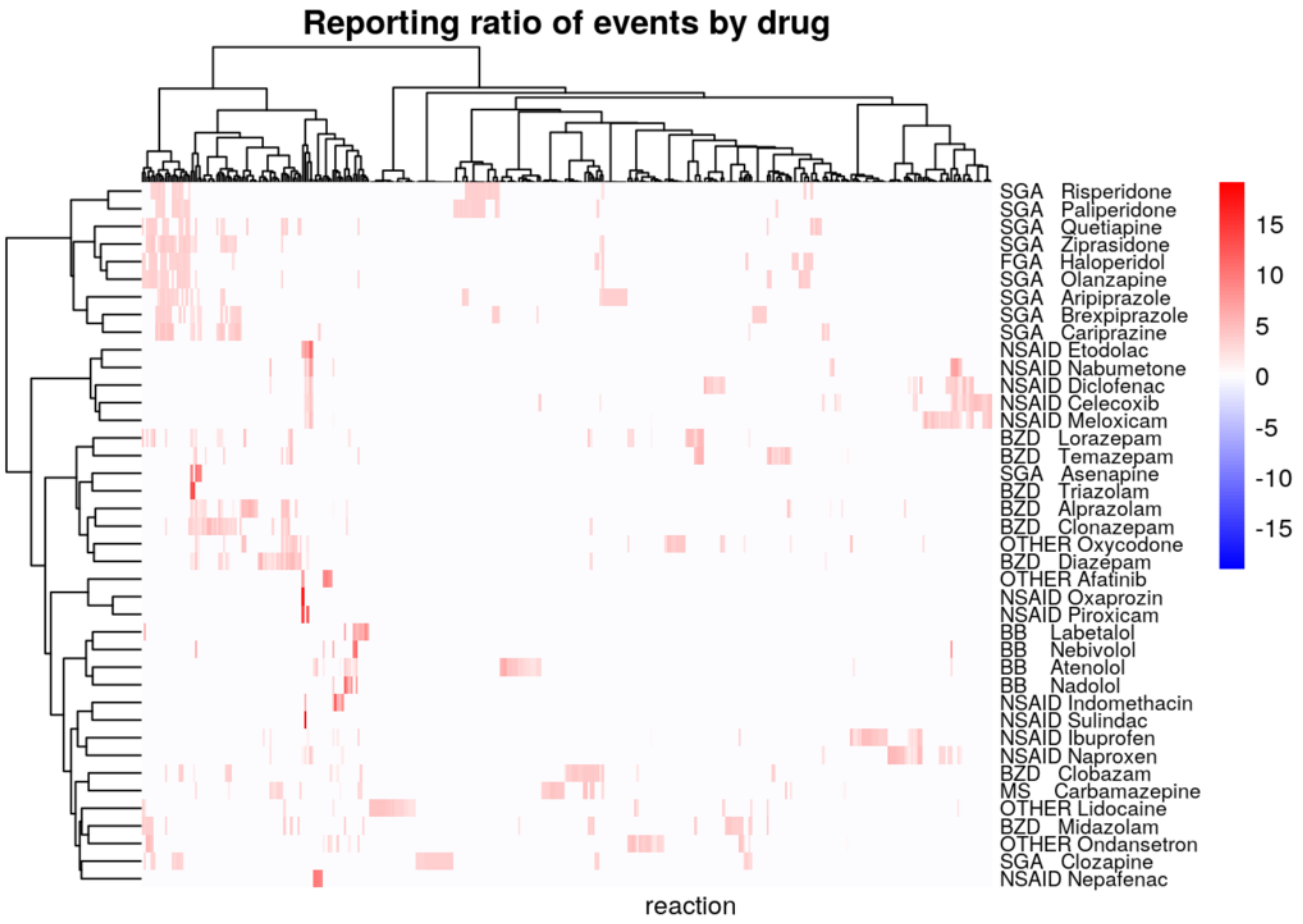
Lidocaine is a local anesthetic used to treat pain. This is an obvious demonstration of how correlation does not imply causation. most of the signals are related to pain, which is most likely the drug indication and not a side effect. People are also more likely to report pain for pain killers compared to other drugs which could also contribute to the inflated ROR.

Correlation of reactions within drug classes

Hierarchical clustering is a common unsupervised learning technique that attempt to find drugs similar in their reaction profiles and build a tree-like structure to represent how the drugs could be split into groups with similar reaction profiles.

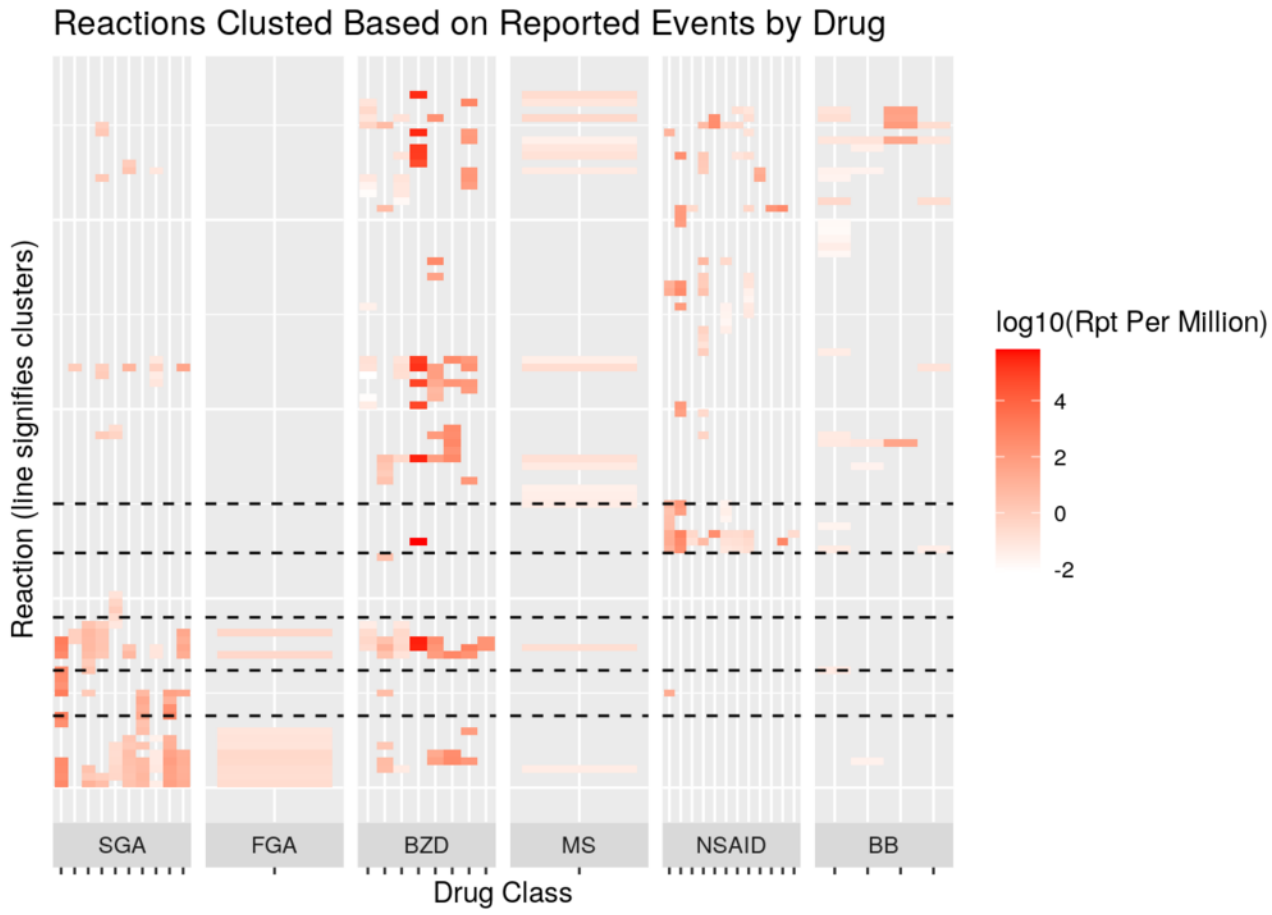
Here we compuated a distance matrix based on the ratio of observed and expected counts, and then clustered the drugs using the Ward method and plotted the matrix using the `pheatmap` package. As expected, we observed that drugs within the same drug class tend to be clustered together, and we can see certain reactions being mostly observed within a specific drug class.

Code



We also attempted to cluster reactions together, and plotted a heatmap of number of reports by drug and clustered reactions, faceted by drug class. However we did not observe obviously different patterns of occurence between reactions. While we could see some patterns such as NSAIDs reactions are mostly in the second-to-top cluster, while FGA and SGA reactions are mostly in the lower clusters. This may be because the distance matrix is not constructed optimally, such as due to incorrect estimation of distance for events that are not reported at all in a certain drug.

Code



Discussion

We have demonstrated that the FDA Adverse Event Reporting System (FAERS) is a valuable data source for adverse event pattern mining and drug safety monitoring.

Additionally, we have demonstrated that the classification of the drug has significant impact on the observed modes of adverse events. Future work could be done to investigate whether this observation could be used to assist in distinguishing increased reporting frequency due to an side effect other than confounding factors.

References

Ahmed, I., Dalmasso, C., Haramburu, F., Thiessard, F., Broët, P., & Tubert-Bitter, P. (2010). False Discovery Rate Estimation for Frequentist Pharmacovigilance Signal Detection Methods. *Biometrics*, *66*(1), 301–309. https://doi.org/10.1111/j.1541-0420.2009.01262.x (https://doi.org/10.1111/j.1541-0420.2009.01262.x)

Dalmasso, C., Broët, P., & Moreau, T. (2005). A simple procedure for estimating the false discovery rate. *Bioinformatics (Oxford, England)*, *21*(5), 660–668. https://doi.org/10.1093/bioinformatics/bti063 (https://doi.org/10.1093/bioinformatics/bti063)

Kass-Hout, T. A., Xu, Z., Mohebbi, M., Nelsen, H., Baker, A., Levine, J., Johanson, E., & Bright, R. A.

(2016). OpenFDA: An innovative platform providing access to a wealth of FDA's publicly available data. *Journal of the American Medical Informatics Association*, *23*(3), 596–600. https://doi.org/10.1093/jamia/ocv153 (https://doi.org/10.1093/jamia/ocv153)

Kolde, R. (2019). *pheatmap: Pretty Heatmaps* (1.0.12). https://CRAN.R-project.org/package=pheatmap (https://CRAN.R-project.org/package=pheatmap)

Medicare Part D Spending by Drug. (n.d.). Centers for Medicare & Medicaid Services Data. Retrieved October 22, 2022, from https://data.cms.gov/jsonapi/node/dataset/54426646-2108-48e7-a339-730dcfabbe9a (https://data.cms.gov/jsonapi/node/dataset/54426646-2108-48e7-a339-730dcfabbe9a)

Pounds, S., & Cheng, C. (2006). Robust estimation of the false discovery rate. *Bioinformatics*, *22*(16), 1979–1987. https://doi.org/10.1093/bioinformatics/btl328 (https://doi.org/10.1093/bioinformatics/btl328)

Wickham, H., Chang, W., Henry, L., Pedersen, T. L., Takahashi, K., Wilke, C., Woo, K., Yutani, H., Dunnington, D., & RStudio. (2022). *ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics* (3.3.6). https://CRAN.R-project.org/package=ggplot2 (https://CRAN.R-project.org/package=ggplot2)

Wickham, H., & RStudio. (2022). *tidyverse: Easily Install and Load the "Tidyverse"* (1.3.2). https://CRAN.R-project.org/package=tidyverse (https://CRAN.R-project.org/package=tidyverse)

Appendix

All ROR Signals

Hide

```
print.ror.table(drug_event_ror, use.remark = FALSE)
```

- AFATINIB (OTHER, Kinase Inhibitor), 5 signals.
- ALPRAZOLAM (BZD, Benzodiazepine), 114 signals.
- OXYCODONE (OTHER, Opioid Agonist), 96 signals.
- CLOZAPINE (SGA, Atypical Antipsychotic), 78 signals.
- HALOPERIDOL (FGA, Typical Antipsychotic), 59 signals.
- RISPERIDONE (SGA, Atypical Antipsychotic), 80 signals.
- IBUPROFEN (NSAID, Nonsteroidal Anti-inflammatory Drug), 198 signals.
- ONDANSETRON (OTHER, Serotonin-3 Receptor Antagonist), 52 signals.
- LIDOCAINE (OTHER, Amide Local Anesthetic), 40 signals.
- ARIPIPRAZOLE (SGA, Atypical Antipsychotic), 164 signals.
- ASENAPINE (SGA, Atypical Antipsychotic), 4 signals.
- ATENOLOL (BB, beta-Adrenergic Blocker), 83 signals.

- BREXPIPRAZOLE (SGA, Atypical Antipsychotic), 74 signals.
- CARBAMAZEPINE (MS, Mood Stabilizer), 47 signals.
- CARIPRAZINE (SGA, Atypical Antipsychotic), 27 signals.
- CELECOXIB (NSAID, Nonsteroidal Anti-inflammatory Drug), 98 signals.
- CLOBAZAM (BZD, Benzodiazepine), 35 signals.
- CLONAZEPAM (BZD, Benzodiazepine), 180 signals.
- DIAZEPAM (BZD, Benzodiazepine), 87 signals.
- DICLOFENAC (NSAID, Nonsteroidal Anti-inflammatory Drug), 45 signals.
- ETODOLAC (NSAID, Nonsteroidal Anti-inflammatory Drug), 5 signals.
- INDOMETHACIN (NSAID, Nonsteroidal Anti-inflammatory Drug), 6 signals.
- LABETALOL (BB, beta-Adrenergic Blocker), 9 signals.
- LORAZEPAM (BZD, Benzodiazepine), 117 signals.
- MELOXICAM (NSAID, Nonsteroidal Anti-inflammatory Drug), 79 signals.
- MIDAZOLAM (BZD, Benzodiazepine), 27 signals.
- NABUMETONE (NSAID, Nonsteroidal Anti-inflammatory Drug), 13 signals.
- NADOLOL (BB, beta-Adrenergic Blocker), 6 signals.
- NAPROXEN (NSAID, Nonsteroidal Anti-inflammatory Drug), 83 signals.
- NEBIVOLOL (BB, beta-Adrenergic Blocker), 6 signals.
- NEPAFENAC (NSAID, Nonsteroidal Anti-inflammatory Drug), 4 signals.
- OLANZAPINE (SGA, Atypical Antipsychotic), 102 signals.
- OXAPROZIN (NSAID, Nonsteroidal Anti-inflammatory Drug), 1 signals.
- PALIPERIDONE (SGA, Atypical Antipsychotic), 44 signals.
- PIROXICAM (NSAID, Nonsteroidal Anti-inflammatory Drug), 2 signals.
- QUETIAPINE (SGA, Atypical Antipsychotic), 129 signals.
- SULINDAC (NSAID, Nonsteroidal Anti-inflammatory Drug), 1 signals.
- TEMAZEPAM (BZD, Benzodiazepine), 26 signals.
- TRIAZOLAM (BZD, Benzodiazepine), 2 signals.
- ZIPRASIDONE (SGA, Atypical Antipsychotic), 44 signals.

Relevant Rcpp Code

Match a vector of strings to a list of regex

Hide

```cpp
#if defined(_OPENMP)
#include <omp.h>
#endif

#include <regex>
#include <vector>
#include <Rcpp.h>

// Vector broadcast regex search
// RESULT[i] = any(regexp.match(patterns[:], input[i]))
//
// [[Rcpp::plugins(cpp14)]]
// [[Rcpp::plugins(openmp)]]
// [[Rcpp::export]]
Rcpp::LogicalVector str_broadcast_search(const Rcpp::CharacterVector inputs,
                                         const Rcpp::CharacterVector patterns)
{
    Rcpp::LogicalVector output(inputs.size());
    std::vector<std::regex> compiled_patterns(patterns.size());
    for (size_t i = 0; i < patterns.size(); i++)
    {
        compiled_patterns[i] = std::regex(patterns[i], std::regex_constants::egrep);
    }

#if defined(_OPENMP)
#pragma omp parallel for schedule(guided)
#endif
    for (size_t i = 0; i < inputs.size(); i++)
    {
        const auto input = Rcpp::as<std::string>(inputs[i]);
        for (const auto &pattern : compiled_patterns)
        {
            if (output[i] = std::regex_search(input, pattern))
                break;
        }
    }

    return output;
}
```

ROR Statistics

Hide

```cpp
#include "check_eps.hpp"

#include <vector>
#include <algorithm>
#include <Rcpp.h>

// Compute ROR and local false discovery rate
//
// Input: A Data frame with at least 3 columns:
//        - First Column: Drug label
//        - Second Column: Event label
//        - Third Column: Number of events
//        Each Drug/Event combination must be unique.
//
// Output: The input data frame bound with additional columns:
//         - n.expect: The expected number of events
//         - margin.drug: The margin of the drug
//         - margin.event: The margin of the event
//         - ror: The relative odds ratio
//         - stat.z: The z-score
//         - stat.p: The p-value
//         - stat.log.p: The log p-value
//         - stat.lb: The lower bound of the confidence interval
//         - stat.lfdr: The local false discovery rate per Ahmed et al. 2010
//         - stat.log.lfdr: The log local false discovery rate
//         - stat.log.bh: The log Benjamini-Hochberg p-value (calculated with p.adjust)
//
// [[Rcpp::plugins(cpp14)]]
// [[Rcpp::export]]
Rcpp::DataFrame ror(const Rcpp::DataFrame &df,
                    const double alpha = 0.0025)
{
    //
    //                  |   Drug j=1    | Other drugs j=0
    // Event i=1        |
    // Other Event i=0  |
    std::unordered_map<std::string, uint64_t> drug_margin;
    std::unordered_map<std::string, uint64_t> event_margin;
    Rcpp::CharacterVector df_drug = df[Rcpp::_, 0];
    Rcpp::CharacterVector df_event = df[Rcpp::_, 1];
    Rcpp::NumericVector df_n = df[Rcpp::_, 2];

    for (const auto &drug : df_drug)
        drug_margin[Rcpp::as<std::string>(drug)] = 0;

    for (const auto &event : df_event)
        event_margin[Rcpp::as<std::string>(event)] = 0;

    uint64_t n__ = 0;

    for (int i = 0; i < df.nrow(); i++)
```

```
{
    n__ += df_n[i];
    drug_margin[Rcpp::as<std::string>(df_drug[i])] += df_n[i];
    event_margin[Rcpp::as<std::string>(df_event[i])] += df_n[i];
}

Rcpp::NumericVector df_ror(df.nrow());
Rcpp::NumericVector df_log_ror(df.nrow());
Rcpp::NumericVector df_ror_se(df.nrow());
Rcpp::NumericVector df_z(df.nrow());
Rcpp::NumericVector df_p(df.nrow());
Rcpp::NumericVector df_log_p(df.nrow());
Rcpp::NumericVector df_lb(df.nrow());
Rcpp::NumericVector df_n_expect(df.nrow());
Rcpp::NumericVector df_margin_drug(df.nrow());
Rcpp::NumericVector df_margin_event(df.nrow());

uint64_t f_05_times_n = 0;

for (int i = 0; i < df.nrow(); i++)
{
    const auto n11 = df_n[i];
    const auto n_1 = drug_margin[Rcpp::as<std::string>(df_drug[i])];
    const auto n1_ = event_margin[Rcpp::as<std::string>(df_event[i])];
    const auto n10 = n_1 - n11;
    const auto n01 = n1_ - n11;
    const auto n00 = n__ - n_1 - n1_ + n11;

    const auto ror = (double)(n11 * n00) / (double)(n10 * n01);
    const auto logRor = log(n11) + log(n00) - log(n10) - log(n01);
    const auto varLogRor = 1.0 / n11 + 1.0 / n00 + 1.0 / n10 + 1.0 / n01;
    const auto seLogRor = std::sqrt(varLogRor);

    const auto z = logRor / seLogRor;
    const auto p = R::pnorm(-std::abs(z), 0, 1, 1, 0);
    const auto log_p = R::pnorm(-std::abs(z), 0, 1, 1, 1);
    const auto lb = R::qnorm(alpha, logRor, seLogRor, 1, 0);

    df_ror[i] = ror;
    df_log_ror[i] = logRor;
    df_ror_se[i] = seLogRor;
    df_z[i] = z;
    df_p[i] = p;
    df_log_p[i] = log_p;
    df_lb[i] = lb;
    df_n_expect[i] = n1_ * n_1 / n__;
    df_margin_drug[i] = n_1;
    df_margin_event[i] = n1_;

    if (p < 0.5)
        f_05_times_n++;
```

```
}
check_eps(df_p, "p", &df_z);

// f(a): portion of p < alpha
// v(a): function of f(a)
std::vector<double> f_over_df(df.nrow());
for (size_t i = 0; i < df.nrow(); i++)
    f_over_df[i] = (i + 1) / (double)df.nrow();
std::stable_sort(f_over_df.begin(), f_over_df.end(),
                 [&df_p](double i1, double i2)
                 { return df_p[i1] < df_p[i2]; });
check_eps(f_over_df, "f^(a)");

double a_hat = 0;
for (int i = 0; i < df.nrow(); i++)
    a_hat += 2 * std::min(df_p[i], 1 - df_p[i]);

double pi_hat = std::min(1., 2 * a_hat / df.nrow());

auto v_hat = [=](int i)
{
    if (df_p[i] <= .5)
        return pi_hat * df_p[i];
    return pi_hat * 0.5 + f_over_df[i] -
           (double)f_05_times_n / df.nrow();
};
auto v_hat_log = [=](int i)
{
    if (df_p[i] <= .5)
        return log(pi_hat) + df_log_p[i];
    // p is big enough it is prob more accurate
    // to just use original value
    return log(pi_hat * 0.5 + f_over_df[i] -
               (double)f_05_times_n / df.nrow());
};

Rcpp::NumericVector df_lfdr(df.nrow());
Rcpp::NumericVector df_lfdr_log(df.nrow());
for (int i = 0; i < df.nrow(); i++)
{
    df_lfdr[i] = v_hat(i) / f_over_df[i];
    df_lfdr_log[i] = v_hat_log(i) - log(f_over_df[i]);
}
check_eps(df_lfdr, "lfdr", &df_p);

Rcpp::Environment stats = Rcpp::Environment::namespace_env("stats");
Rcpp::Function p_adjust = stats["p.adjust"];
Rcpp::NumericVector df_fdr_bh = p_adjust(df_p, "BH");

Rcpp::Environment base = Rcpp::Environment::namespace_env("base");
Rcpp::Function cbind = base["cbind"];
```

```
    return cbind(df, Rcpp::DataFrame::create(
                          Rcpp::Named("n.expect") = df_n_expect,
                          Rcpp::Named("margin.drug") = df_margin_drug,
                          Rcpp::Named("margin.event") = df_margin_event,
                          Rcpp::Named("ror") = df_ror,
                          Rcpp::Named("stat.z") = df_z,
                          Rcpp::Named("stat.p") = df_p,
                          Rcpp::Named("stat.log.p") = df_log_p,
                          Rcpp::Named("stat.lb") = df_lb,
                          Rcpp::Named("stat.lfdr") = df_lfdr,
                          Rcpp::Named("stat.log.lfdr") = df_lfdr_log,
                          Rcpp::Named("stat.fdr.bh") = df_fdr_bh));
}
```

Splitting a hierarchical clustering tree

Hide

```cpp
#include <vector>
#include <algorithm>
#include <Rcpp.h>
#include <functional>
#include <queue>

// [[Rcpp::export]]
Rcpp::NumericVector hclust_split(const Rcpp::NumericMatrix &merge, const int n_split)
{

    const auto nrow = merge.nrow();
    Rcpp::NumericVector ret(nrow + 1);

    std::queue<int> q;
    q.push(nrow);

    while (q.size() != n_split)
    {
        const auto i = q.front();
        q.pop();
        if (i < 0)
        {
            q.push(i);
        }
        else
        {
            q.push(merge(i - 1, 0));
            q.push(merge(i - 1, 1));
        }
    }
    for (int i = 0; i < n_split; i++)
    {
        const auto root = q.front();
        q.pop();
        std::function<void(int)> dfs = [&](int v)
        {
            if (v < 0)
            {
                ret[-v - 1] = i + 1;
            }
            else
            {
                dfs(merge(v - 1, 0));
                dfs(merge(v - 1, 1));
            }
        };
        dfs(root);
    }
    return ret;
}
```

```
## # A tibble: 23 × 3
##    key              modifier    count
##    <chr>            <chr>       <dbl>
##  1 dplyr::arrange   ""              2
##  2 dplyr::filter    ""              6
##  3 dplyr::group_by  ""              4
##  4 dplyr::group_by  "cate"          4
##  5 dplyr::group_by  "num"           1
##  6 dplyr::join      ""              3
##  7 dplyr::join      "inner"         3
##  8 dplyr::mutate    ""             12
##  9 dplyr::mutate    "convert"       2
## 10 dplyr::mutate    "depend"        5
## 11 dplyr::mutate    "rename"        1
## 12 dplyr::mutate    "transmute"     1
## 13 dplyr::pivot     ""              1
## 14 dplyr::pivot     "longer"        1
## 15 dplyr::pivot     "pattern"       1
## 16 dplyr::select    ""              3
## 17 dplyr::select    "rowwise"       1
## 18 dplyr::summary   ""              2
## 19 dplyr::summary   "cate"          2
## 20 dplyr::summary   "num"           4
## 21 dplyr::summary   "tally"         1
## 22 ggplot           ""              2
## 23 ggplot           "geom_line"     2
```