Investigating Adverse Drug Reactions

Denis Kazakov
CSCI 4502
102298967

Tyler Fox
CSCI 5502
810913549

Nicole Woytarowicz
CSCI 4502
101423791

ABSTRACT

We will investigate how data mining techniques can be used on reported side effects patients experience during medical drug therapy to determine which drugs are causing which side effects and to determine which drugs react with each other.

KEYWORDS

data mining, data classification, pharmacovigilance, drug safety, adverse drug effect, adverse drug reaction

1. INTRODUCTION

Adverse drug effects (ADEs)[1], afflictions that occur during the use of medical drug therapy (whether actually caused by the drug(s) or not), cause severe health problems every year for millions of patients in the United States. According to a study by the U.S. Food and Drug administration, adverse drug effects are one of the leading causes of morbidity in healthcare and approximately 6% of all hospitalized patients will suffer from some sort of adverse drug effect. The number of instances of these effects occurring has continued to rise in recent years, even with overall improvements in medical care. This is primarily due to the increased amount of medications prescribed to patients and the increased amount of medications taken concurrently, the latter of which causes the probability of an adverse effect occurring to escalate exponentially once a patient is on four or more medications at the same time.

Adverse drug effects are the main focus of the field of pharmacovigilance, which concerns itself with detecting and preventing unwanted side effects caused by medical drugs in a timely manner. However, the field often suffers from the inability to correlate adverse drug effects with any specific drug(s) patients take, as they are reported either by the patients themselves or by patients' medical practitioners and can result from a variety of causes, such as drug-drug interactions, unknown allergies, or reasons completely unrelated to the specific medical treatment. While attempts are made by healthcare providers to stop adverse effects once they arise (by altering dosages, switching drugs used, or ending therapy completely), the principle problem continues to lie in the inability to easily detect exactly what causes adverse effects, especially when multiple medications are taken at once.

In 2014, the U.S. Food and Drug administration launched a public API database, openFDA, of adverse drug effects reported by patients and their healthcare providers. The database contains reports from 2004

[1] Also known as adverse drug reactions (ADRs)

through present day, and separates them based on attributes such as gender, age, reason for use, and if the patient was also taking nonsteroidal anti-inflammatory drugs (NSAIDs), as they are commonly taken by enough of the population to necessitate their own category. The database also groups drugs, either as possible suspects for one or more adverse effect(s) or as concomitants, meaning they are taken at the same time as another drug but are not suspected to be responsible for the adverse effect (usually common painkillers such as Aspirin or Tylenol fall into the concomitants category).

The openFDA database unfortunately suffers from the main problem of pharmacovigilance, in that if a patient is taking multiple drugs at the same time, it is difficult to ascribe a specific symptom to any one drug. Even when the patient is only taking one drug, it can also be difficult to determine whether a certain reaction is actually caused by that drug or by an unrelated reason (such as other conditions the patient suffers from).

In this project, we used the openFDA database to determine which drugs are most likely to cause which side effects and which drugs seem to interact negatively with each other.

2. RELATED WORK

In a similar study by Columbia University's Department of Biomedical Informatics, a team was able to identify 1,167 drug-drug interactions with their adverse affects using 162,744 reports from the same openFDA database and strong association rule mining[1]. The team found that a medical domain expert could clinically validate 67% out of these 1,167 interactions as potential drug-drug interactions previously recognized by the medical community, while 4% were interactions between drugs that were already known and well established. Several potentially medically sound interactions that had been unknown to the medical community prior to this study were also discovered.

The team used an optimized version of the Apriori algorithm (an algorithm that relies on the monotonicity rule and generates frequent itemsets from transactions based on a minimum support count) that hashed drugs to effects to generate their strong association rules, which they did so in order to run the algorithm quickly enough for such a large dataset. Only individual drugs with a high enough support count were hashed in the algorithm, because if an individual drug did not pass the support threshold, then any combinations of that drug with another drug would not pass the support threshold and could not be considered a frequent drug-drug interactions. They also only generated strong association rules with drugs in the antecedent and side effects in the consequent (e.g., drug 1 + drug 2 $\Rightarrow$ side effect), which helped to optimize the algorithm further. The team managed to optimize their implementation of Apriori sufficiently enough to run in four hours on their entire dataset, which included the association rule generation step.

3. DESIGN & IMPLEMENTATION

3.1 Choosing technologies

While we initially thought we might need to use MATLAB and R for modeling/analyzing our data, we ultimately found that Python was able to accomplish everything we sought to do, including gathering our data and implementing our algorithm. Writing all our code in Python made the overall debugging process easier as well, since everyone in the group is experienced in Python and could easily contribute when errors needed to be corrected. It also simplified our code base; with everything written in Python, there was no need to have adapter classes between different languages, decreasing the amount of work needed on the program architecture and allowing us to focus more on mining the data.

3.2 Data Gathering & Preprocessing

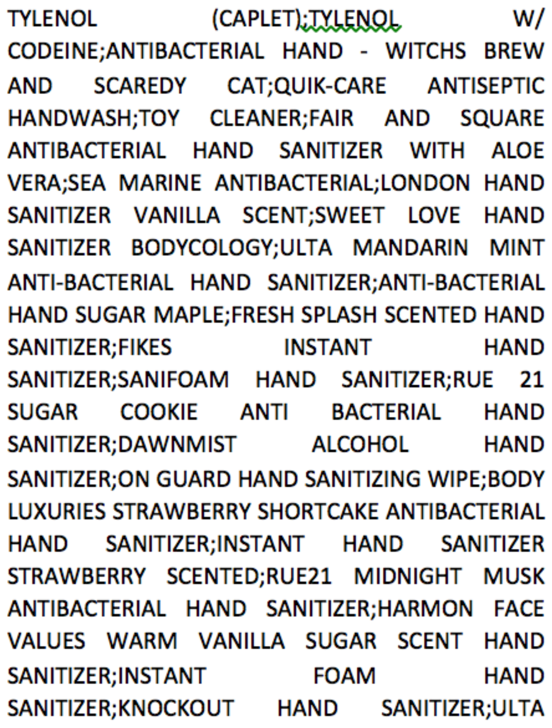
Our very first task in the project was to gather all the data, which we found posed a few problems, mostly due to the fact that the openFDA database is still in beta. We did this by querying openFDA's API to access their dataset, which was done by including the number of results we wanted from each call to the API and an API key provided by openFDA for accessing more results per day (if using the API key, a max of 100 reports could be returned by each call, with a total of 200,000 reports per day). To get more than 100 results, we had a skip variable in our code that would skip through the database with each call to the API. This variable was dependent on the date the report was received, spanning from 2004 to 2008, which allowed us to attempt to gather a large number of reports. Throughout the process of data collecting, we were gathering around the 200,000 a day max with approximately two-thirds of the reports gathered containing valid side effects and drugs. In total, we collected 665,470 reports

Each call to the API returned a JSON object, which we parsed to extract the data we wanted. This included the ID of the report, the date the report was created, the age of the patient (given in hours, weeks, months, years, or decades, so we converted them all to years), the sex of the patient, the brand names of drugs, and the adverse reactions.

The majority of problems with the database being in beta arose here. We learnt that in the majority of reports, the drugs contained in each report were listed under the category "medicinal product," while in some others reports there were multiple fields under "medicinal product" that the drugs were found in, so we had to modify our code account for this. We also found that that were redundancies in certain reports in how the drugs were listed (e.g., CAPSIL and (CAPSIL) occurring in the same report). We had to strip unnecessary characters like quotation marks, parentheses, apostrophes, etc. out of the drug names, and we also had to account for the redundancies themselves, so we would not have an inaccurate number of drugs when trying to find possible associations between drugs and side effects.

We created a separate Python file to make this data more accessible, which essentially would call functions from the data gathering file and create a given number of CSV files with a given number of reports in each file. We created multiple CSV files in case our program stopped in error while collecting reports (either due to hitting the daily query limit or running into some sort of inconsistency in the reports that we had not yet encountered, such as an apostrophe in the drug name), we would not lose all the data we had collected in just one instance of running the program. Essentially, we created checkpoints for the data as we gathered it, so we could just re-instantiate the program from our last data checkpoint in case the program shut down.

There were a few instances we ran into while creating our CSV files where the given list of drugs for a report was so long, that it would break Python's maximum length of a string type. This happened at around the 32,000-character mark, and seemed to result only from errors in the database itself. An example of this that occurred more than once is when the drugs listed for a patient seemed to consist of normal drugs followed by every single antibacterial soap/sanitizing product registered by the FDA, consisting of ten whole pages and approximately 3,500 words when pasted into a Microsoft Word document. A brief segment of this data is included in the following figure:



TYLENOL (CAPLET);TYLENOL W/ CODEINE;ANTIBACTERIAL HAND - WITCHS BREW AND SCAREDY CAT;QUIK-CARE ANTISEPTIC HANDWASH;TOY CLEANER;FAIR AND SQUARE ANTIBACTERIAL HAND SANITIZER WITH ALOE VERA;SEA MARINE ANTIBACTERIAL;LONDON HAND SANITIZER VANILLA SCENT;SWEET LOVE HAND SANITIZER BODYCOLOGY;ULTA MANDARIN MINT ANTI-BACTERIAL HAND SANITIZER;ANTI-BACTERIAL HAND SUGAR MAPLE;FRESH SPLASH SCENTED HAND SANITIZER;FIKES INSTANT HAND SANITIZER;SANIFOAM HAND SANITIZER;RUE 21 SUGAR COOKIE ANTI BACTERIAL HAND SANITIZER;DAWNMIST ALCOHOL HAND SANITIZER;ON GUARD HAND SANITIZING WIPE;BODY LUXURIES STRAWBERRY SHORTCAKE ANTIBACTERIAL HAND SANITIZER;INSTANT HAND SANITIZER STRAWBERRY SCENTED;RUE21 MIDNIGHT MUSK ANTIBACTERIAL HAND SANITIZER;HARMON FACE VALUES WARM VANILLA SUGAR SCENT HAND SANITIZER;INSTANT FOAM HAND SANITIZER;KNOCKOUT HAND SANITIZER;ULTA

Figure 3.1: An example of the outlier drug format consisting of antibacterial products.

Another problem we ran into with our data was the fact that the majority of reports found contained drugs listed by their brand names, rather than by the active ingredient in the drug. This is problematic when a single active ingredient has multiple brand names, as this could make it seem like there are fewer connections in our data than there actually are. In order to avoid this, we created a text file from an online list of over 1,000 generic drugs with their corresponding brand names and used regular expressions in Python to create a hash map where brand names were hashed to keys with their generics as values. In doing this, we were able to quickly map the brand names that arose in our data to their corresponding active ingredient.

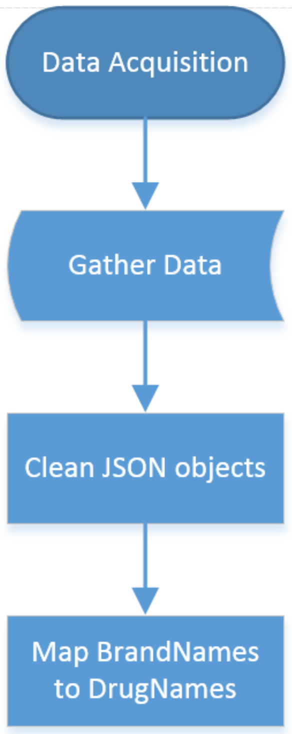


Figure 3.2: The visual representation of our data collection process.

3.2 Choosing an Approach

During our brainstorming phase, our main concern was the size of our dataset. Since we had approximately 650,000 reports, thrice as many as the team in the Columbia study, we wanted to optimize our work as much as possible; therefore, we initially felt reluctant to use the Apriori algorithm in our implementation and tried to use Bayesian networks instead following a suggestion from a faculty professor.

We learned the theory behind Bayesian Networks and saw a potential application to our problem. There are dozens of packages available online for making statistical dependencies between elements of Bayesian network (e.g., PyStan and eBay's Bayesian network framework in Python). Unfortunately, all of these packages were either too complicated to use or had little flexibility in their approach. Thus, we abandoned the idea of using Bayesian belief networks and came back to the Apriori algorithm.

Our main problem with Apriori was the time complexity if we were to use it on our entire dataset. To be precise, with an implementation of the regular Apriori algorithm, it took approximately 40 minutes for Apriori to run on only 5,000 reports. Because we had so many reports collected, we wanted to run our algorithm on every single report to find as many interesting patterns as we could. We solved this problem by pre-computing what drugs are likely to result in a specific side effect and then only running Apriori for reports containing those drugs and side effects. This reduced the amount of combinations that Apriori needed to check, which allowed our algorithm to run in around four minutes on all 665,470 reports.

3.3 Improving the Apriori algorithm

A visual representation of the pre-computation step is shown in Figure 3.3. It is a Python dictionary where the key is an adverse drug effect and values associated with that key are the names and counts of the drugs that resulted in that given adverse effect.

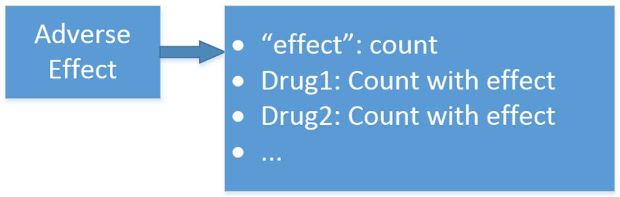


Figure 3.3: Structure of the effect-drugs map

With this pre-computation step we implemented an anti-monotonic rule:

If (DrugA $![L, S, C] \rightarrow$ ADE)

$\Rightarrow$ (!∃ superset {DrugA, …} $[L, S, C] \rightarrow$ ADE)

That is, if a drug does not result in an ADE with given [Lift, Support, Confidence] values, then no superset of that drug can result in an ADE with the given [Lift, Support, Confidence] values.

This rule is notably different from the normal anti-monotonic rule that Apriori relies on:

If (Element $[L, S, C] <$ [minL, minS, minC]) $\Rightarrow$ (!∃ superset {Element, …} $[L, S, C] >$ [minL, minS, minC])

That is, if an element does not occur with given [Lift, Support, Confidence] values, then no superset including that element can occur with given [Lift, Support, Confidence] values.

After completing the pre-computation step, we proceeded to use the Apriori algorithm for finding possible frequent drug-drug interactions that resulted for a given ADE. This way, on average, Apriori only had to search for possibly frequent combinations of three to five drugs that are known to result in a given ADE, as opposed to possibly frequent combinations of all the drugs and side effects.

3.4 Algorithmic complexity

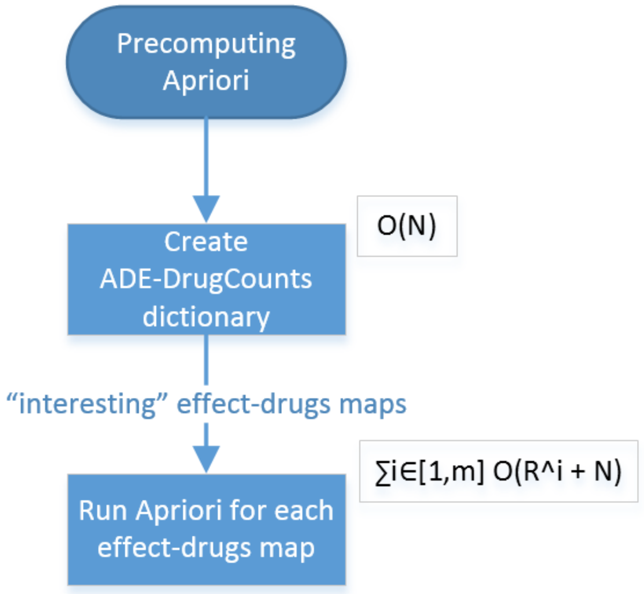

Precomputing Apriori
Create ADE-DrugCounts dictionary
O(N)
"interesting" effect-drugs maps
Run Apriori for each effect-drugs map
∑i∈[1,m] O(R^i + N)

Figure 3.4: For the Apriori algorithm, R – number of elements, N – number of transactions, i – length of a set, m – number of iterations.

By looking at the time complexity of the Apriori algorithm, we can see that R (the number of elements we are combining) contributes the most to the overall complexity of the algorithm; therefore, by cutting it down from a million to just a few, we decreased the time that our algorithm takes by immense amount. We also used dictionaries in Python because they are essentially hash tables, which have an access time of O(1).

4. EVALUATION

After cleaning up our data, we found a total of 12,880 unique adverse effects and 148,610 drugs. The drug number is slightly inaccurate in terms of uniqueness because there were many cases where drugs would have multiple forms of administration (e.g., liquid and oral) or they would come in different forms (e.g., Advil gel caps and Advil tablets). Our algorithm for mapping drugs to their generic was also not 100% accurate, as the list of generics we used did not contain every single drug in existence, so there were many instances where we had to leave the brand names. We also found that 277,931 of the patients in the reports were men, 385,369 were women, and 2,168 had no gender listed. There were approximately 22.5 drugs per report (however, this number is influenced by the 56,617 outlier reports that contained over 100 drugs; without these, the average would only be 6.4) and 4.1 reactions per report. We also found the distributions of the patients' ages, with the most common age group being 50-75 (in 304,079 reports) and the second being 25-50 (192,987 reports).

After running our algorithm with a support count of 100, a confidence value of .3, and a lift value of 10, we discovered approximately 237 frequent adverse effect-drug associations that fit our minimum support thresholds. There were almost no outliers in our final results, except for the "no adverse event" frequent adverse event reports (with a support count of 748), which seemed to have been entered into the openFDA database as actual adverse effects, so they were not caught in the data cleaning stage.

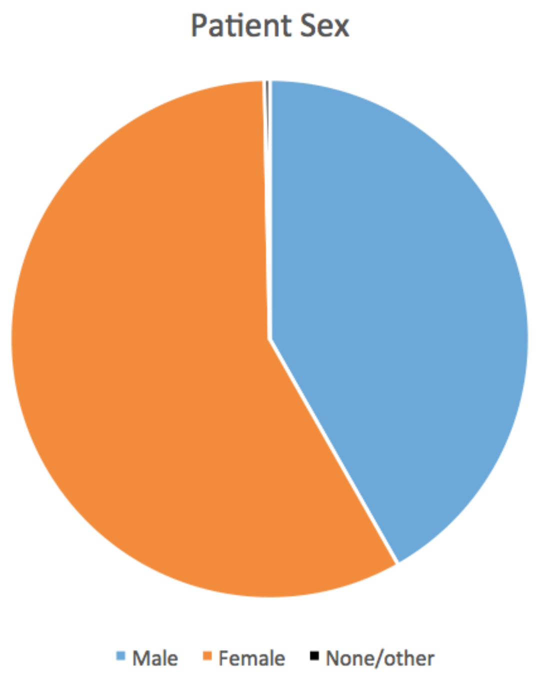


Figure 4.1: Distribution of patient genders

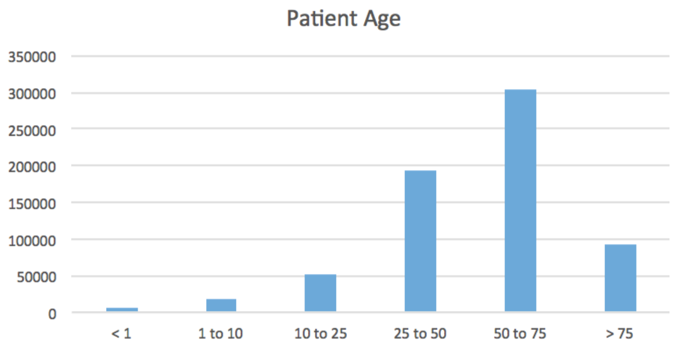


Figure 4.2: Distribution of patient ages

Using our algorithm, we found a few different classes of frequent adverse effects associated with drugs that, after conducting outside research, we found to be interesting. Due to the magnitude of frequent side effects, we decided to not include every single side effect in this report, but rather give examples of some different classes of adverse effect-drug associations we believe to be potentially useful and interesting.

Frequent ADE classes

Class	Description
a	Medically verified adverse effects (for single drug)
b	Medically verified adverse effects (for drug-drug interactions)
c	Potentially novel adverse effects
d	Adverse effect not likely to be caused by the given drug

Examples of each class

	ADE – drug	(L, S, C)
a	Myocardial infarction – Rofecoxib	(13.35, 9878, 0.53)
b	Breast cancer – conjugated estrogens & meroxyprogesterone	(93.67, 1431, 0.45)
c	Loss of taste – Lunesta	(43.91, 1895, 0.36)
d	Death – Thalidomide	(21.40, 4521, 0.34)

A brief description of each example follows:

a: Rofecoxib was an anti-inflammatory drug similar to Tylenol or Advil that was approved by the FDA in 1999. It was discontinued in September 2004 due to the frequency with which it caused heart problems, including myocardial infarctions (heart attacks) and strokes. This is especially interesting because our data begins in 2004, meaning in just a few months almost 10,000 people reported having a heart attack while on this drug.

b: Postmenopausal women are given conjugated estrogens & meroxyprogesterone in hormone replacement therapy. Interestingly enough, cited research says that estrogen by itself seems to have no correlation to an increased risk of breast cancer, but when combined with meroxyprogesterone, breast cancer risk is increased.

c: Lunesta is a common, prescription-only sleeping pill. We were unable to find any medical studies or reports that verified loss of taste as a possible side effect of Lunesta, but we did find some online message boards with people complaining about it resulting from taking the drug, which, combined with the high support count in our data, makes it a potentially novel discovery.

d: Thalidomide is given as a treatment for multiple myeloma (blood plasma cancer), which only has a 45% five-year survival rate when caught early. Although we conducted research to attempt to find any links between thalidomide resulting in death, the closest results we could find were that thalidomide creates a risk of fetal death if taken during pregnancy. This means that, most likely, thalidomide itself is not causing death, but that patients are dying of their cancer and their doctors reported these to the FDA.

5. CONCLUSIONS & FUTURE WORK

Throughout the course of the project, we learnt that the most time consuming parts of finding interesting patterns in our data was cleaning the data itself and then analyzing our final results. Frequently we would come across reports that had poor formatting or simply needed to be thrown out because they were missing vital information, and due to the nature of our project, there was no way to supplement this missing information. Analyzing the final results also became a time consuming endeavor, as we had to research the possible frequent ADE – drug associations after returning our final results. This was also difficult because none of the group members are medical professionals and, therefore, did not have much medical expertise to guide us in any conducted research. We also felt that, because of this, we could not definitively assign certain adverse events to the classes we created for describing our results, so we did not attempt to classify anything that had conflicting research.

Although our algorithm was not perfect (as evidenced by it associating Thalidomide with death), it was able to efficiently and accurately find many frequent adverse effects associated to drugs. The incorrect associations also stemmed from the data itself, and not our algorithm, since there is no quality control involved in voluntary report submitting of adverse effects experienced while taking medications (e.g., people under or over reporting experienced adverse effects). Some of our discovered associations had already been verified by the medical community years before we set out with this project, but these results were still helpful in verifying the correctness of our algorithm.

While minor adverse effects such as nausea or fatigue do not pose severe threats to patients' lives, they can still diminish the quality of life for patients experiencing them. Knowing what drugs cause certain adverse effects is important, not only for the present in keeping patients informed of possible side effects, but also in future drug manufacturing for preventing these same adverse effects in new drugs. We believe that approaches similar to our project will become more and more common in the coming years to discover possible adverse effects, especially in the manufacturing of new drugs; if drug companies know what drugs similar to their formulation caused certain side effects in the past, they can attempt to avoid that occurring in their product.

Further work that could be conducted using our algorithm would involve attempting to make the results as specific as possible by factoring in the age and gender of each patient in the report as well; however, we decided to sacrifice specificity to keep our algorithm as efficient as possible.

6. REFERENCES

[1] Harpaz, Rave; Chase, Herbert; and Friedman, Carol. *Mining multi-item drug adverse effect associations in spontaneous reporting systems*, 2010. University of Columbia Department of Biomedical Informatics.

[2] Medicines Safety: Pharmacovigilance, 2015. World Health Organization. http://www.who.int/medicines/areas/quality_safety/safety_efficacy/pharmvigi/en/

[3] OpenFDA: Drugs – Adverse Effects, 2014. U.S. Food and Drug Administration. https://open.fda.gov/drug/event/

[4] Preventable Adverse Drug Reactions, 2014. U.S. Department of Health and Human Services. http://www.fda.gov/drugs/developmentapprovalprocess/developmentresources/druginteractionslabeling/ucm110632.htm

[5] Simovici, Dan. *Data Mining of Medical Data: Opportunities and Challenges*. University of Massachusetts, Boston.

[6] Sneed, Kevin, and Robert Campbell. "Acute Congestive Heart Failure Induced by Rofecoxib." *Journal of the American Board of Family Medicine* (2004).

[7] Ross, Ronald, Annlia Paganini-Hill, Peggy Wan, and Malcolm Pike. "Effect of Hormone Replacement Therapy on Breast Cancer Risk: Estrogen Versus Estrogen Plus Progestin." *Journal of the National Cancer Institute* (2000).

[8] "Side Effects of Lunesta (Eszopiclone) Drug Center." http://www.rxlist.com/lunesta-side-effects-drug-center.htm

[9] Palumbo, Antonio, Thierry Facon, Pieter Sonneveld, et al. "Thalidomide for Treatment of Multiple Myeloma: 10 Years Later." *Blood Journal* (2008).

[10] "Lists of Generic and Brand Name Drugs." http://www.emedexpert.com/lists/brand-generic.shtml

7. MEMBER CONTRIBUTIONS

On our honor, as University of Colorado at Boulder students,
we have neither given nor received unauthorized assistance on this work.

Team Member	**Contributions**
Tyler Fox	- Gathered and cleaned all the data (involved outlier detection/removal and converting data into a uniform format) - Supervised the process of all code implementation to ensure efficiency and accuracy - Advised in choosing final methodology to find interesting patterns in data - Conducted statistical analysis on the data set to include in final report - Researched methods for finding interesting patterns in the data
Denis Kazakov	- Implemented the improved Apriori algorithm - Assisted in initial data collection - Made presentation slides - Assisted in writing and created diagrams for final report - Researched methods for finding interesting patterns in the data
Nicole Woytarowicz	- Implemented the improved Apriori algorithm - Wrote initial proposal and checkpoint/final reports - Mapped brand name drugs to generic names - Devised project idea and found data source - Researched frequent ADE – drug results returned by algorithm

All group members read and approved the final report before submitting.