Exploring Adverse Drug Effect Data with Apache Spark, Hadoop, and Docker

Nicholas A. Davis, PhD
Assistant Professor of Research
Department of Medical Informatics
School of Community Medicine
University of Oklahoma, Tulsa
nicholas-davis@ouhsc.edu
@argoneus

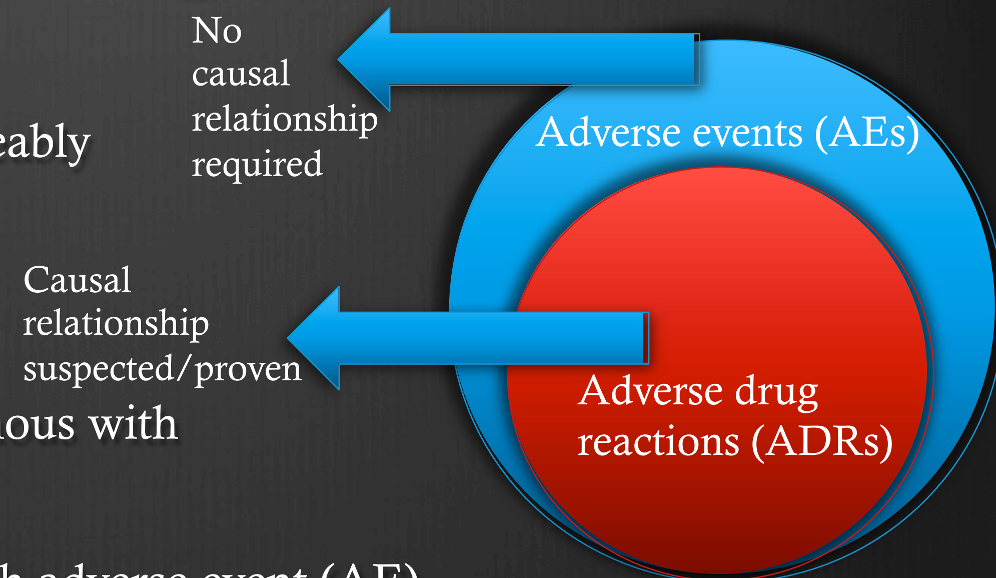
Oklahoma Supercomputing Symposium
September 23, 2015

Outline

- Adverse Drug Effects
  - Definitions
  - Motivation
- FAERS
- FAERS data characteristics
- Architecture
- Overview of Hadoop
- Overview of Apache Spark
- Overview of Docker
- Data Exploration
- Conclusions

http://www.cfhi-fcass.ca/SearchResultsNews/2010/10/01/f1ca3fcf-f9aa-4dcb-a64c-a95f2bbadd8d.aspx

Adverse Drug Effects



Inspired by: http://www.slideshare.net/DrVijayBhushanam/vj-ad-rs

- Common terms often used interchangeably
  - Adverse drug effect
  - Adverse drug event/Adverse event
  - Adverse drug reaction
  - "Side effect"
- Adverse drug effect (ADE) is synonymous with adverse drug reaction (ADR)
- Adverse drug event is synonymous with adverse event (AE)
- Adverse event (AE) is defined as any untoward medical occurrence in a patient administered a medicinal product which does not necessarily have a causal relationship with this treatment
- Adverse drug reaction (ADR) is defined as all noxious and unintended responses to a medicinal product related to any dose
- ADRs are thus a subset of AEs
- Distinction between side effect and ADR

Motivation to Study ADRs

- Significant burden on population, as 68% of US population is taking at least one prescription drug (Mayo Clinic, 2013)
- 40% of adults aged 65+ take 5 to 9 concurrent medications
- Studies have shown that ADRs result in increases in morbidity, mortality, as well as increases healthcare costs
- Most recent data shows over 2 million people in the US are impacted by ADRs by death, hospitalization or serious injury
  - 100K fatalities
  - 1.5 million hospital visits
- ADRs can occur in US patients during hospitalizations

Motivation to Study ADRs (cont.)

- Meta-analysis in 2002 found that 4.9% of hospitalizations were associated with ADRs
  - 28.9% of these were considered preventable
- More recent study in 2012 found 2% of inpatients and 1.6% of outpatients had preventable ADRs
  - This study also suggests that approximately half of ADRs are preventable among both outpatients and inpatients
- 28% of ER visits due to drugs, 70% of these were preventable

FDA Adverse Event Reporting System (FAERS)

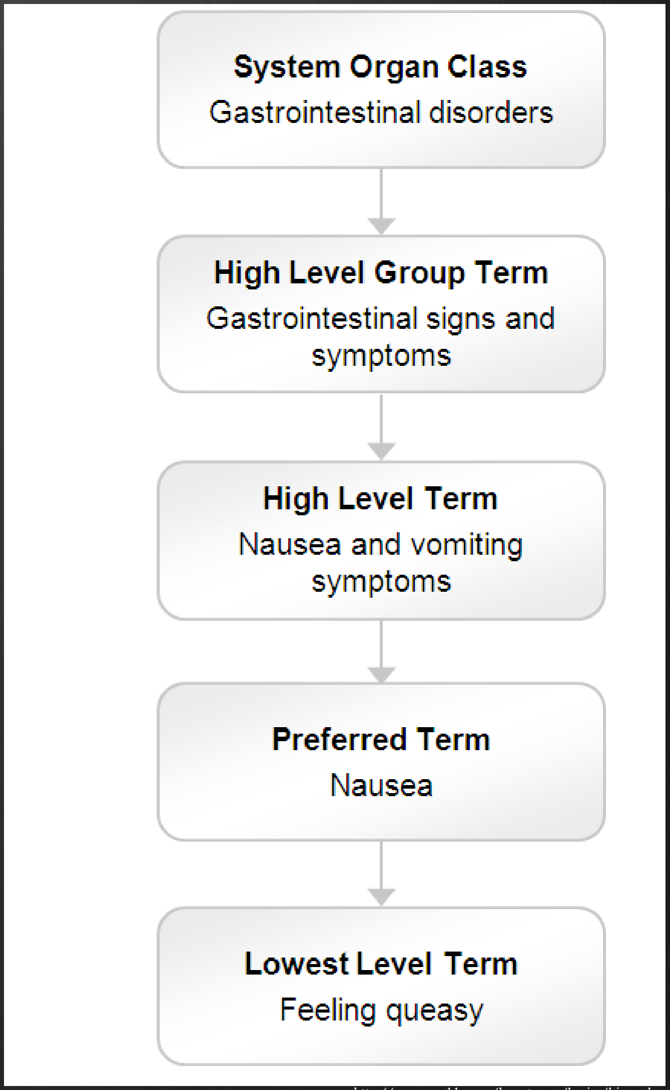
- Database that contains information on adverse event (AEs) and medication reports submitted to the US Food and Drug Administration (FDA)
- Voluntary for healthcare professionals (physicians, pharmacists, nurses) to report AEs
- Often healthcare professionals will report AEs to product manufacturer
- Mandatory for product manufacturers to submit report to FDA for entry into FAERS
- Limitations of data
  - No certainty that reported event actually due to product
  - FDA does not require causal relationship
  - FDA does not receive reports for every AE that occurs with a product
- Available to the public as an open data set!

FAERS (cont.)

- FDA uses FAERS to discover new safety concerns related to a marketed product
- Also uses FAERS to monitor manufacturer’s compliance with reporting regulations
- Data structure conforms to international safety reporting guidance issued by ICH
- AEs coded to terms in Medication Dictionary for Regulatory Activities (MedDRA) terminology

MedDRA

- Hierarchy of 5 levels
- Specific or most granular terms (LLTs) through broad groupings by etiology, manifestation site, or purpose (SOCs)
- Preferred Term (PT) is a distinct descriptor or single medical concept for a symptom, sign, disease diagnosis, therapeutic indication, investigation, surgical or medical procedure
- FAERS data uses PTs



http://www.meddra.org/how-to-use/basics/hierarchy

FAERS Data Characteristics

- OpenFDA initiative provides FAERS data via RESTful API
  - Limitations on requests per minute/per day
  - API Key available to increase limits
  - Useful for a limited project using infrequent requests
  - Difficult to leverage all available data for a broad analysis
- Data also provided via a series of downloads (used in this project)
- Data packaged in zip files by quarter
  - Packages available in both XML format and ASCII text character-delimited format files
  - Data ranges from Q1 2004 through Q1 2015

FAERS Data Characteristics (cont.)

- Each quarter of FAERS data is a separate download link (not ideal)
  - Used Firefox Web Developer to select innerHTML surrounding download URL
  - Copied HTML into vim and trimmed down to raw URL
  - Used bash script with curl to automated download process
- Once script completed there were 45 zip files, total size of 736MB compressed
- Each zip file contains a few documentation files and the raw ASCII character-delimited data in .txt files
- Used bash script to extract files to a single directory for simplicity
- Lots of duplicate documentation (Readme.doc/pdf) files, so used unzip option to prevent overwriting files with same filename
- Converted .doc/.pdf documentation to plaintext using Apache Tika, removed duplicate documentation using SHA1 sums to compare files
- Total uncompressed size of all ASCII text files is 3.8GB

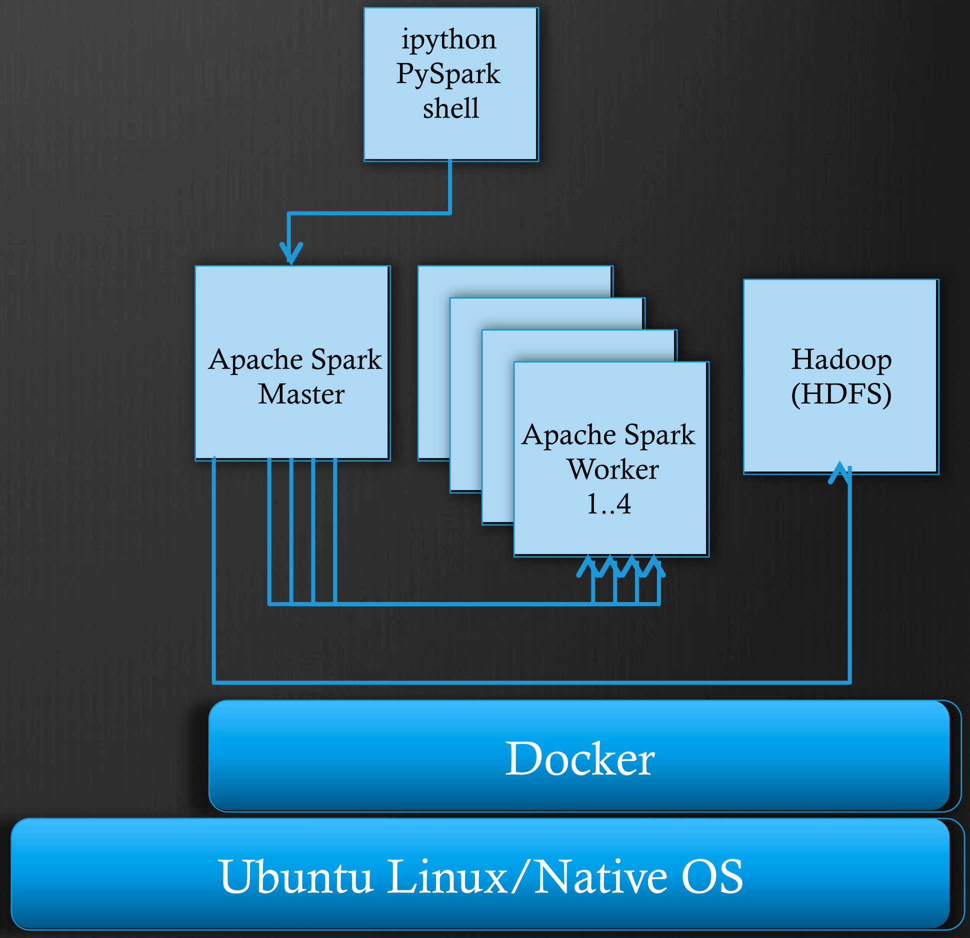
FAERS Data Characteristics (cont.)

- Total of 385 ASCII text files representing 9 discrete categories

Category	Description
DEMO	Patient demographics for each event
DRUG	Drug info report for each event
INDI	MedDRA terms coded for the indications for use (diagnoses) for the reported drugs
OUTC	Patient outcomes for each event
REAC	MedDRA terms coded for the event
RPSR	Report sources for the event
SIZE	File sizes and record counts for all data (discontinued after Q3'12)
STAT	Gives null counts and frequency (discontinued after Q3'12)
THER	Drug therapy start dates and end dates for the reported drugs

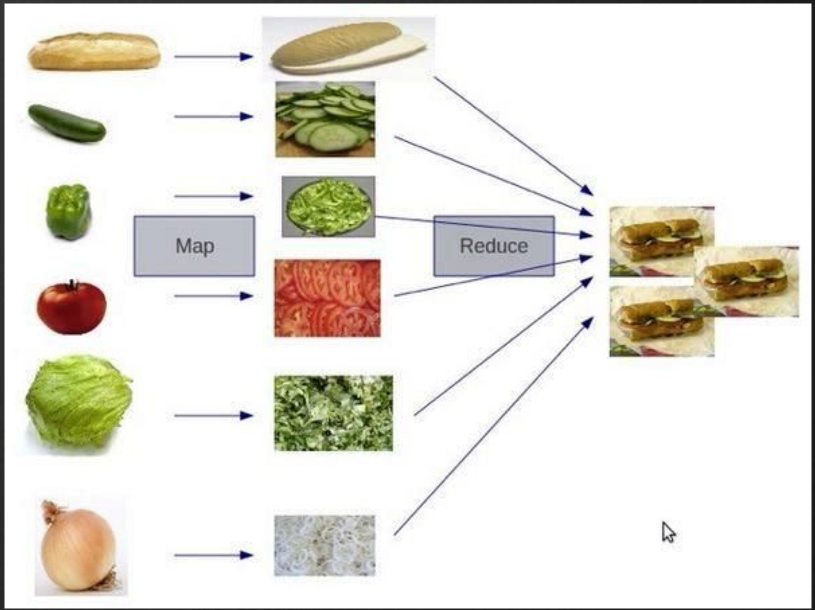
Architecture

- Want to leverage open source distributed computational tools for processing extracted FAERS data
- No easy access to multiple commodity machines for building cluster
- Instead utilize server class hardware used for research and analysis
  - Dell Precision T7610 workstation
  - 128GB RAM, 12-core Xeon E5-2697 2.7GHz
  - Ubuntu Linux 14.04
- Employ Hadoop, Apache Spark, and PySpark for convenient data loading, cleaning, and analysis
- Use Docker to run multiple isolated applications, virtual networking for communications



Hadoop Overview

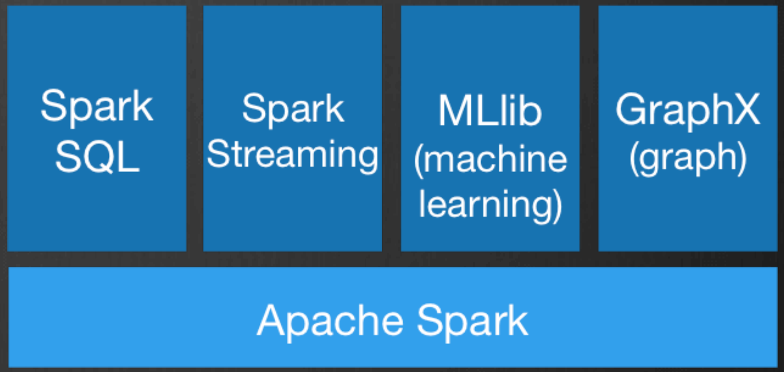
- Open source Java-based distributed storage and processing platform, derived from Google's publication of a similar proprietary platform for large data sets
- Leverages commodity hardware
- "Big Data" tool used by Yahoo, FaceBook, Twitter (large organizations with Petabytes of data) and others with smaller data sets
- Provides ecosystem of tools that accompany primary computational engine (MapReduce)
  - Hadoop distributed filesystem (HDFS)
  - Hive, Pig, HBase
- This project utilizes HDFS for distributed storage
- ASCII data files loaded into HDFS



https://twitter.com/daviottenheimer/status/532661754820829185

Apache Spark Overview

- Open source, Java-based alternative to Hadoop MapReduce platform
- Execution engine optimized for in-memory computing, compared to Hadoop's disk-based engine
- Leverages existing data concepts: SQL-based query language and DataFrames (inspired from R language and Python's Pandas)
- Up to 100X faster than Hadoop in memory, 3X on disk
- Supports interactive data exploration from Python, R, Scala
- Largely compatible with Hadoop ecosystem tools and data
- Primary abstraction is Resilient Distributed Dataset (RDD), collection of elements partitioned across the nodes of a cluster



Docker Overview

- Challenge: Hadoop tools and Apache Spark require significant time to install and configure
  - Utilize numerous TCP ports for subcomponents to enable communication between worker and master processes
  - Difficult to run multiple worker processes if default ports are the same
  - Orchestration of components across multiple machines in a cluster is tedious
- Docker allows application isolation, provides OS-level virtualization
  - Similar to a VM but lightweight (uses Linux kernel namespaces and cgroups), uses existing OS resources instead of emulation host hardware
  - Uses the concept of containers to package a pre-configured application
  - Containers are versatile and can move between systems
- Docker repository of pre-built containers for a host of applications
  - Use existing repo images for Hadoop, Apache Spark, and iPython with PySpark for interactive analysis
  - Each application runs in an isolation container, using a virtual IP address
  - Containers communicate with each other (as well as the host) using standard networking protocols

Data Exploration- Challenges

- Filenames are mostly consistent (e.g. DEMO04Q1.TXT) but some are lowercase or the extension is lowercase (.txt)
- Data formats change at least once for each category (DEMO, DRUG, INDI, etc.)
  - FAERS added additional columns in these change
  - File header with column labels is thus variable and must be accommodated
  - In some categories the schema of the data is altered 3 or 4 times between Q1 2004 to Q1 2015
    - E.g. DRUG and REAC both undergo 3 schema alterations, DEMO 4
  - All categories changed formats in 2012Q4
  - We use the common fields and schema that exist for the whole range of data
- Since AEs are often user reported, many entries are incomplete or have missing data
- STAT and SIZE files are not regularly-formatted tabular files, but rather generated reports, thus are excluded from analysis

Data Exploration - Schemas

'DEMO' : "id event_dt mrf_dt fda_dt mfr_name age age_unit sex weight weight_unit”

'DRUG' : "id drug_seq role_code drugname dose dechal rechal exp_dt”

'INDI' : "id drug_seq indi_pt”

'OUTC' : "id outcome_code”

'REAC' : "id pt"

'RPSR' : "id report_code"

'THER' : "id drug_seq start_dt end_dt duration duration_unit"

Data Exploration - Code

```
# Dr. Nicholas Davis, 9-18-15
# Process FDA Adverse Event Reporting System (FAERS) data using Spark
from pyspark.sql import SQLContext
from pyspark.sql.types import *
from dateutil.parser import parse

sqlContext = SQLContext(sc)

# HDFS location of FAERS data files
hdfs_path = "hdfs://172.17.0.9:9000/user/root/faers"

# Each 'partition' of a category has a distinct header, thus must be read and
# grouped individually initially
categories = ['DEMO1', 'DEMO2', 'DEMO3', 'DEMO4', 'DRUG1', 'DRUG2', 'DRUG3',
            'INDI1', 'INDI2', 'OUTC1', 'OUTC2', 'REAC1', 'REAC2', 'REAC3',
            'RPSR1', 'RPSR2', 'THER1', 'THER2']

groupings = ['{DEMO04Q[1-4]*,DEMO05Q[1-2]*}', '{DEMO05Q[3-4]*,DEMO0[6-9]*,DEMO1[0-1]*,DEMO12Q[1-3]*}',
            '{DEMO12Q4*,DEMO13*,DEMO14Q[1-2]*}', '{DEMO14Q[3-4]*,DEMO15*}', '{DRUG0[4-9]*,DRUG1[0-1]*,DRUG12Q[1-3]*}',
            '{DRUG12Q4*,DRUG13*,DRUG14Q[1-2]*}', '{DRUG14Q[3-4]*,DRUG15*}', '{INDI0[4-9]*,INDI1[0-1]*,INDI12Q[1-3]*}',
            '{INDI12Q4*,INDI1[3-5]*}', '{OUTC0[4-9]*,OUTC1[0-1]*,OUTC12Q[1-3]*}', '{OUTC12Q4*,OUTC1[3-5]*}',
            '{REAC0[4-9]*,REAC1[0-1]*,REAC12Q[1-3]*}', '{REAC12Q4*,REAC13*,REAC14Q[1-2]*}', '{REAC14Q[3-4]*,REAC15*}',
            '{RPSR0[4-9]*,RPSR1[0-1]*,RPSR12Q[1-3]*}', '{RPSR12Q4*,RPSR1[3-5]*}', '{THER0[4-9]*,THER1[0-1]*,THER12Q[1-3]*}',
            '{THER12Q4*,THER1[3-5]*}']
```

Data Exploration - Code (cont.)

```
maps = {
    'DEMO1' : [0, 5, 6, 7, 10, 11, 12, 13, 15, 16],
    'DEMO2' : [0, 5, 6, 7, 10, 11, 12, 13, 15, 16],
    'DEMO3' : [0, 4, 5, 7, 10, 11, 12, 13, 15, 16],
    'DEMO4' : [0, 4, 5, 7, 11, 13, 14, 16, 18, 19],
    'DRUG1' : [0, 1, 2, 3, 6, 7, 8, 10],
    'DRUG2' : [0, 2, 3, 4, 7, 10, 11, 13],
    'DRUG3' : [0, 2, 3, 4, 8, 11, 12, 14],
    'INDI1' : [0, 1, 2],
    'INDI2' : [0, 2, 3],
    'OUTC1' : [0, 1],
    'OUTC2' : [0, 2],
    'REAC1' : [0, 1],
    'REAC2' : [0, 2],
    'REAC3' : [0, 2],
    'RPSR1' : [0, 1],
    'RPSR2' : [0, 2],
    'THER1' : [0, 1, 2, 3, 4, 5],
    'THER2' : [0, 2, 3, 4, 5, 6]
}

schemas = {
    'DEMO' : "id event_dt mrf_dt fda_dt mfr_name age age_unit sex weight weight_unit",
    'DRUG' : "id drug_seq role_code drugname dose dechal rechal exp_dt",
    'INDI' : "id drug_seq indi_pt",
    'OUTC' : "id outcome_code",
    'REAC' : "id pt",
    'RPSR' : "id report_code",
    'THER' : "id drug_seq start_dt end_dt duration duration_unit"
}
```

Data Exploration - Code (cont.)

```
scfiles = {}
scparts = {}
scelements = {}
scschema = {}
counts = []
for cat, grp in zip(categories, groupings):
    # Read file categories into Spark RDD
    tf = sc.textFile(hdfs_path + "/" + grp)
    # collect headers and remove from input
    header = tf.filter(lambda l: l.startswith("primaryid") or l.startswith("ISR"))
    header.collect()
    tf_nohdr = tf.subtract(header)
    # filter out lines with missing data, where number of fields < greatest field in map
    badlines = tf_nohdr.filter(lambda l: len(l.split("$")) < maps[cat][-1] + 1)
    badlines.collect()
    tf_good = tf_nohdr.subtract(badlines)
    scfiles[cat] = tf_good
    scparts[cat] = tf_good.map(lambda l: l.split("$"))
    scelements[cat] = scparts[cat].map(lambda p: (eval("".join(tuple("p[" + str(i) + "]," for i in maps[cat])))))
    # specify fields via previously defined maps/schemas
    fields = [StructField(field_name, DateType(), True) if "_dt" in field_name else
            StructField(field_name, StringType(), True) for field_name in schemas[cat[0:4]].split()]
    schema = StructType(fields)
    scschema[cat] = sqlContext.createDataFrame(scelements[cat], schema)
    counts.append(tf_nohdr.count())

# now combine partitions into original sources (DEMO, DRUG, etc.)
demo = scschema['DEMO1'].unionAll(scschema['DEMO2']).unionAll(scschema['DEMO3']).unionAll(scschema['DEMO4'])
drug = scschema['DRUG1'].unionAll(scschema['DRUG2']).unionAll(scschema['DRUG3'])
indi = scschema['INDI1'].unionAll(scschema['INDI2'])
outc = scschema['OUTC1'].unionAll(scschema['OUTC2'])
reac = scschema['REAC1'].unionAll(scschema['REAC2']).unionAll(scschema['REAC3'])
rpsr = scschema['RPSR1'].unionAll(scschema['RPSR2'])
ther = scschema['THER1'].unionAll(scschema['THER2'])
```

Examples

Count distinct drug name values and sort by count size

```
drug.groupby('drugname').count().sort('count', ascending = False).show()
```

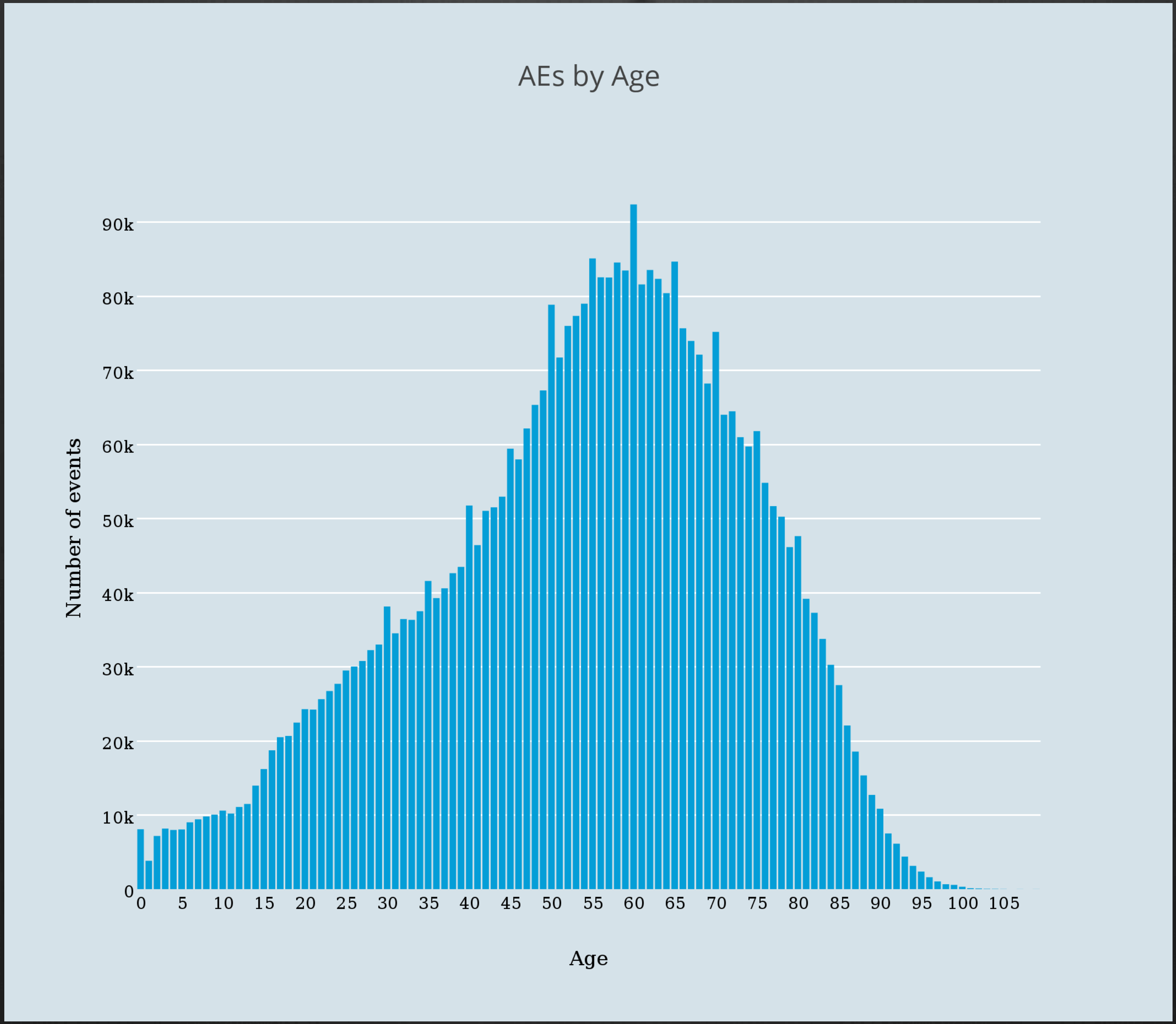
```
+------------+------+
|    drugname| count|
+------------+------+
|      HUMIRA|320548|
|     ASPIRIN|232669|
|      ENBREL|203476|
|      AVONEX|188123|
|     TYSABRI|173756|
|METHOTREXATE|158019|
|    REMICADE|156752|
|     LIPITOR|128664|
|    SEROQUEL|124765|
|      NEXIUM|123843|
```

Find most frequently occurring drugname and outcome code combinations

```
drug.join(outc, drug.id == outc.id).select(drug.drugname, outc.outcome_code).groupby(['drugname', 'outcome_code']).count().sort('count', ascending = False).limit(10).collect()
```

```
[Row(drugname=u'ASPIRIN', outcome_code=u'HO', count=108060),
 Row(drugname=u'REMICADE', outcome_code=u'OT', count=94816),
 Row(drugname=u'ASPIRIN', outcome_code=u'OT', count=91467),
 Row(drugname=u'AVONEX', outcome_code=u'HO', count=82472),
 Row(drugname=u'HUMIRA', outcome_code=u'HO', count=76989),
 Row(drugname=u'REMICADE', outcome_code=u'HO', count=71848),
 Row(drugname=u'HUMIRA', outcome_code=u'OT', count=70364),
 Row(drugname=u'VIOXX', outcome_code=u'HO', count=63845),
 Row(drugname=u'LIPITOR', outcome_code=u'OT', count=61579),
 Row(drugname=u'FOSAMAX', outcome_code=u'OT', count=61475)]
```

Examples


AEs by Age
Number of events
90k
80k
70k
60k
50k
40k
30k
20k
10k
0
0
5
10
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
100
105
Age

Conclusions

- ADRs and AEs worthy of exploration and further research, to help decrease associated morbidity, mortality and costs
- FAERS data set provides a convenient, open, and free resource for exploring trends in reported ADRs/AEs
- Distributed open source computational platforms (i.e. Hadoop and Apache Spark) provide a free, convenient mechanism to analyze large data sets
- Docker and pre-built Docker images can help alleviate some of the troublesome installation and configuration pains
- Time to analysis is reduced by leveraging these tools

References

- Beijer HJ, de Blaey CJ. Hospitalisations caused by adverse drug reactions (ADR): A meta-analysis of observational studies. *Pharmacy World and Science* Apr 2002; 24: 46 - 54.
- Classen DC, Pestotnik SL, Evans RS, Lloyd JF, Burke JP. Adverse drug events in hospitalized patients: Excess length of stay, extra costs, and attributable mortality. *Journal of the American Medical Association* Jan 22, 1997; 277: 301 - 306.
- Hakkarainen KM, Hedna K, Petzold M, Hägg S (2012) Percentage of Patients with Preventable Adverse Drug Reactions and Preventability of Adverse Drug Reactions – A Meta-Analysis. *PLoS ONE* 7(3): e33236. doi:10.1371/journal.pone.0033236
- Lazarou J, Pomeranz BH, Corey PN. Incidence of adverse drug reactions in hospitalized patients: A meta-analysis of prospective studies. *Journal of the American Medical Association* Apr 15, 1998; 279: 1200 - 1205.
- Patel P, Zed PJ. Drug-related visits to the emergency department: How big is the problem?. *Pharmacotherapy* Jul 2002; 22: 915 - 923.
- Pirmohamed M, James S, Meakin S, et al. Adverse drug reactions as cause of admission to hospital: prospective analysis of 18 820 patients. *BMJ: British Medical Journal*. 2004;329(7456):15-19.
- Zhong, W., Maradit-Kremers, H., St. Sauver, J. L., Yawn, B. P., Ebbert, J. O., Roger, V. L., ... Rocca, W. A. (2013, July 1). Age and sex patterns of drug prescribing in a defined American population. *Mayo Clinic Proceedings*. Mayo Clinic. Mayo Foundation for Medical Education and Research.

Questions?