PharmaSUG 2016 - Paper IB14

Access OpenFDA: A Cloud Based Big Data Portal Using SAS®

Jie Roger Zhou, University of Bridgeport, Bridgeport, CT

James Sun, Insmed Inc, Bridgewater, NJ

Abstract

With the rise of data sharing and cloud data porta, big dada and analytics are gradually being adopted across pharmaceutical industry. It made first impact on part of business such as sales and marketing which data have readily available. Now big data analytic gradually reaching the core of pharma business- R&D. One such example is newly emerging big data OpenFDA. As FDA collects all new drug submissions and pharmacovigilance data, its importance should not be underestimated.

The paper briefly introduced OpenFDA, and its API when used in extracting data. We explore ways of handling such online data in JSON format with SAS data steps and procedures. We compare the strength of different methods and discuss some information

Keyword: OpenFDA, Big Data, Pharmacovigilance, JSON, PROC HTTP, PROC DS2, PROC JSON, PROC Groovy

Introduction

New integrated drug development model provides for more complex disciplines such as translational and evidence-based medicine. The industry is becoming more patient centric and realizing value of patient outcomes, improved safety and efficacy through better data insights. Such insights and intelligence which are extracted from big data can spur innovation, empower decision-making and enhance customer relationships. Now the big data processing capability is becoming critical for any drug company to stay as completive and successful.

The traditional pharmaceutical R&D model is driven by securing regulatory safety and efficacy and mostly managed within the boundary of company. For the last 10 years, we have seen tremendous changes in emphasis collaboration both internally and with the outside world. To gain a competitive edge and increase the efficiency, drug companies are now working with outsiders more than ever.

- **CROs**: We have seen growth of utilizing CROs in data analysis for all phases of clinical trials, and for both small and big drug companies. Other than CRO and other data service provider, nowadays we have seen more co-developing or co-marketing efforts which join forces of 2 pharmas together to push new drug to the market as soon as possible.
- **Health Professional and Patient**: With explosive development of social media, now drug companies can reach out to more care providers and patients with ease. They're also conducting sentiment analysis of online physician communities, electronic medical records, and consumer-generated media to flag potential safety issues. These data can then be used to shape strategy throughout the pipeline progression.
- **Insurance Companies:** in order to encourage share and analyze outcomes and claims data between payors and providers, pharmas are able to better position and promote their drugs and response to market changing conditions swiftly.
- **Public Healthcare Data Portals**: with fierce competitions and more "me-too" drugs crowded market, pharmas need to keep eyes on industry wide developments; cross leveraging the data shared by other players in the field is essential. One important source of insights and intelligence companies can gain is through analyses of some available public healthcare data portals.

Cloud Data

In supporting highly demanding collaborations, the traditional way of data accessing infrastructure is gradually becoming antiquated. Often the case on-premise hardware / data center solution can no longer face up to the challenges. Instead, cloud data along with cloud computing play an ever important role in supporting wider reach of collaborations. The main difference between a cloud and a data center is that a cloud is an off-premise form of computing that stores data on the Internet.

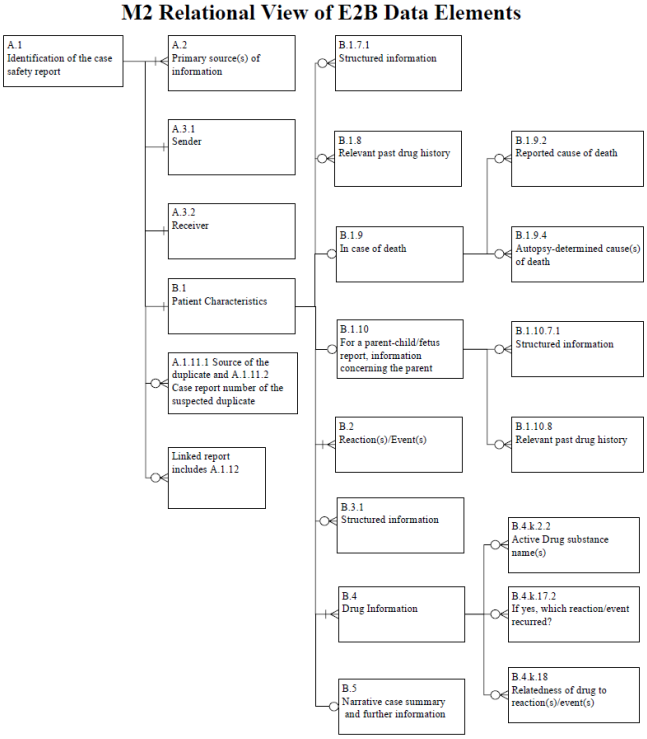
OpenFDA

OpenFDA is an initiative from Health Informatics at FDA. It provides researchers, consumers and health professionals, easy access to valuable FDA public data, making it simpler for them to use this data in their work to advance and promote the public health. The initiative leverages new technologies and methods such as cloud computing and open source software to unlock the tremendous public data and resources at FDA in a user-friendly way.

OpenFDA was launched in beta test mode on June 2, 2014. There are four parts each covers drug adverse events, medical device adverse events, drug labeling and recall information. Within one year since its launch, there have been more than 20 million API calls (more than half from outside the United States), 6000 registered users, 20,000 connected internet addresses, and dozens of new software (mobile or web) apps developed.

The data component of drug adverse events contains data from ***FDA Adverse Event Reporting System (FAERS***), a database of adverse event and medication error reports submitted to the agency. It covers publically releasable records from 2004-2015. The related information include serious drug side effects, product use errors, product quality problems and therapeutic failures for prescription and over-the-counter medicines.

The data fields of adverse event reports in OpenFDA are in compliance with the standard from International Conference on Harmonization: “Electronic Transmission of Individual Case Safety Reports Message Specification” (***ICH E2b/M2 v.2***). The following diagram depict the entity-relationship among each data field collected.



JSON Data File

When a user want access such AE data, he will use Application Program Interface (***API***) defined by OpenFDA to send request online. The API used is part of REST-API which is widely employed in online environment. Basically, everyone can submitted such request through web browser. The receiving data file from OpenFDA is in JSON format.

JSON stands for ***JavaScript Object Notation***. Because it is a "self-describing" and lightweight data format, JSON is widely used in data interchange. It is now becoming the most common data format, largely replacing XML, used in browser/server communication. ***XML***has been used to describe structured data and to serialize objects. Various XML-based protocols exist to represent the same kind of data structures as JSON for the same kind of data interchange purposes. Data can be encoded in XML by using tag pairs results in a much larger representation than JSON.

JSON supports datatype including integer and strings, as well as array. JSON syntax is derived from JavaScript object notation syntax, it has following features: data is in name/value pairs, data is separated by commas, curly braces hold objects, and square brackets hold arrays. Due to some complex structure like nested array in JSON file, typical SAS programming such as data step has hard time to parse JSON file some time.

In the following section I will show you three methods how we are going to import JSON file in different situations.

Example 1. Parse Simple JSON Data File with SCANOVER Option

As mentioned above, one can submit request to OpenFDA in a browser. By typing in following command in browser's URL address field:

```
https://api.fda.gov/drug/event.json?search=receivedate:[20150125+TO+20150201]&count=receivedate
```

Based on OpenFDA's API, what you requested is counts of AE reported from 1/25/2015 to 2/1/2016. You will see following JSON data in browser window. What we see is a simple JSON file which each data field separated by }.

To process such a simple JSON file saved from browser, we can used SAS data step's infile statement with SCANOVER option to parse character string ",}" through the whole JSON file. What we get is this simple dataset on left.

```
FILENAME OpenFDA "C:\OpenFDA_query.JSON";

DATA simple;
  INFILE openFDA. LRECL = 3456677 TRUNCOVER S
CANOVER dsd dlm=",}";
  INPUT
    @"'time":' ReportDate : $30.
    @"'count":' Count: $10.
run;
```

ReportDate	Count
20150125	273
20150126	4178
20150127	3651
20150128	3334

To save the step of saving JSON file from browser window, we can use PROC HTTP with 'GET' method to directly grab the JSON file within SAS.

```
filename outJSON "C:\ OpenFDA_query.JSON";

proc http method="get"
url='https://api.fda.gov/drug/event.json?search=receivedate:[20150125+TO+20150201]&count=receivedate'
out=outJSON;
run;
```

From this example we can see that SAS has default function can parse and import data in a very straightforward way. However, most of the JSON we get from REST-API may be more complex than this so we cannot directly use option.

Example 2. Process OpenFDA JSON Data with PROC GROOVY and Java Virtual Machine

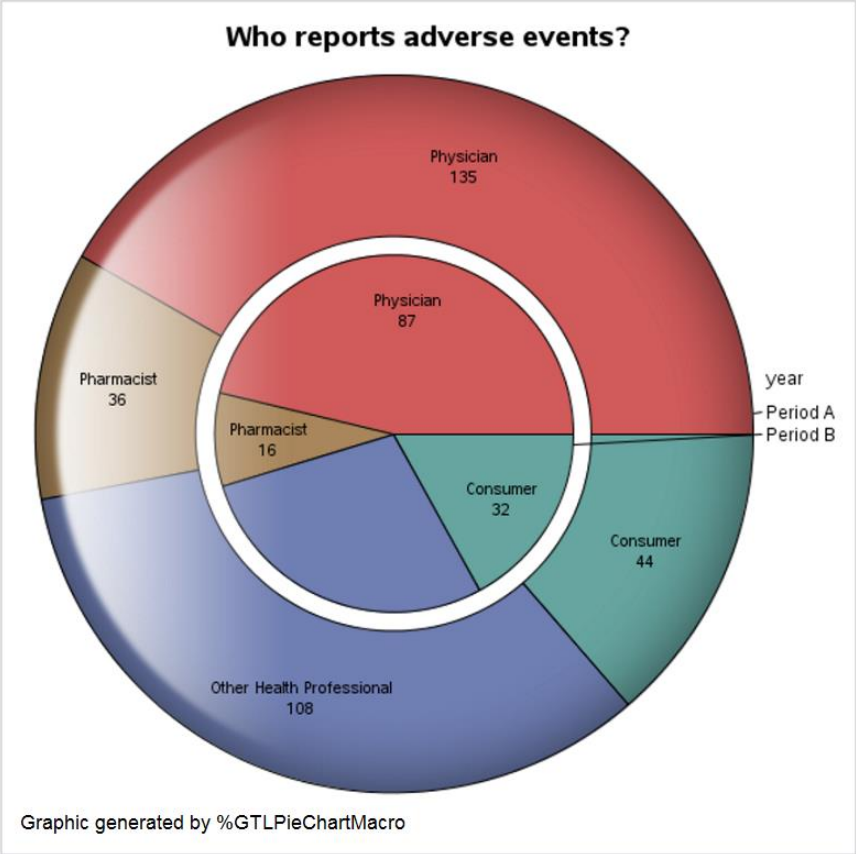
PROC GROOVY can be an alternative way to parse and import JSON file into SAS data set. Groovy is a dynamic language that runs on the Java Virtual Machine (JVM). PROC GROOVY enables SAS code to execute Groovy code on the JVM. One important advantage of using PROC GROOVY is it includes scripting capabilities. It is close to JSON script so it can read script smoothly. The disadvantage is that Groovy is language outside of SAS, thus you might need to learn some ABCs of Groovy. Plus, you have to make sure your Java Virtual Machine is working and some of java applets used in the process, such as groovy.jar and opencsv.jar, were downloaded in your local machine.

Here is the example SAS code:

```
%let JSON_OpenFDA_FILE=D:\OpenFDA.json;
%let CSV_OpenFDA_FILE=D:\ResponseContent.csv;
/* parse JSON and generate temporary CSV file */
 proc groovy
 classpath="C:\Program Files\groovy-2.4.6\embeddable\groovy-all-2.4.6.jar;D:\opencsv-3.7.jar";
  submit "&JSON_OpenFDA_FILE." "&CSV_TWEET_FILE.";
    import groovy.json.*
    import com.opencsv.CSVWriter
    def input = new File(args[0]).text
    def output = new JsonSlurper().parseText(input)
    def csvoutput = new FileWriter(args[1])
    CSVWriter writer = new CSVWriter(csvoutput);
    String[] header = new String[2];
    header[0] = "term";
    header[1] = "count";
    writer.writeNext(header);
    output.resluts.each {
      String[] content = new String[2];
      content[0] = it.primarysource.qualification.toString();
      content[1] = it.count.toString();
      writer.writeNext(content)
```

```
    }
    writer.close();
  endsubmit;
quit;
```

With a SAS macro for pie chart graphic (%GTLPieChartMacro), we can show the AE reported by primary source over two different time periods. You can spot the proportion of reports change over two periods



Brief Introduction of PROC DS2

PROC DS2 is a SAS proprietary programming language that is used for data manipulation and data modeling applications. It extends far beyond core features of SAS DATA step. It includes several features that are not available in the DATA step like: variable scoping, user-defined methods, ANSI SQL data types, user-defined packages, and programming structure elements.

PROC DS2 Components include Data Type, Variables, Constants, Expressions, Programming Blocks and Scope, Methods, and Packages. Its appearance is more in line with other programming language. With enhancement in SAS 9.4 TS1M3, PROC DS2 can be used to process complex JSON data files.

Structure of DS2 program:

DS2

- **PROC DS2;**
  - DATA <output dataset>;
    - ...Variable declarations…
    - …Retain/Keep/Drop Commands…
    - ...Method definitions…
    - …Data step commands, including SET Statement…
  - ENDDATA;
  - RUN;
  - …More data blocks…
- **QUIT;**

Traditional Data Step

- **DATA <output dataset>;**
  - …Retain/Keep/Drop Commands…
  - …Data step commands, including SET Statement…
- **RUN;**

You can see the difference between these two programs. The main features we will use in our example from DS2 differ from Data step are: variable scope, method and package. We can understand the scope concept in DS2 by macro global and local variable. Not like data step program only can have global variable DS2 can declare local or global variable from different scopes. This can help programmer handle variables easier and make methods more efficient.

Methods play very important role in DS2. Method concept come from object oriented principles. The main purpose of method is to encapsulate sequence of instructions which perform a specific task into modules this could make program reuse and maintenance easily. Also, method can improve the readability and understandability of modular programming and testing. You can define you own method in PROC DS2 and it can be invoke in current procedure.

Package is another powerful tool build in PROC DS2. DS2 package is a collection of methods and variables that can be used in the DS2 program. A DS2 package is similar to a class in object-oriented languages. There are two types of packages: Predefined Packages. These packages encapsulate common functionality that is useful to many customer solutions. Predefined packages are part of DS2. User-defined Packages. These are the packages that you create or that someone else created for reuse. In example 3 we will use two predefined packages HTTP package and JSON package. HTTP package has web request method similar to PROC HTTP so we can get data from openFDA API. JSON package includes a default parse method GETNEXTTOKEN(). Using this method we can read every token one by one.

Example 3. Process Complex OpenFDA JSON Data with PROC DS2

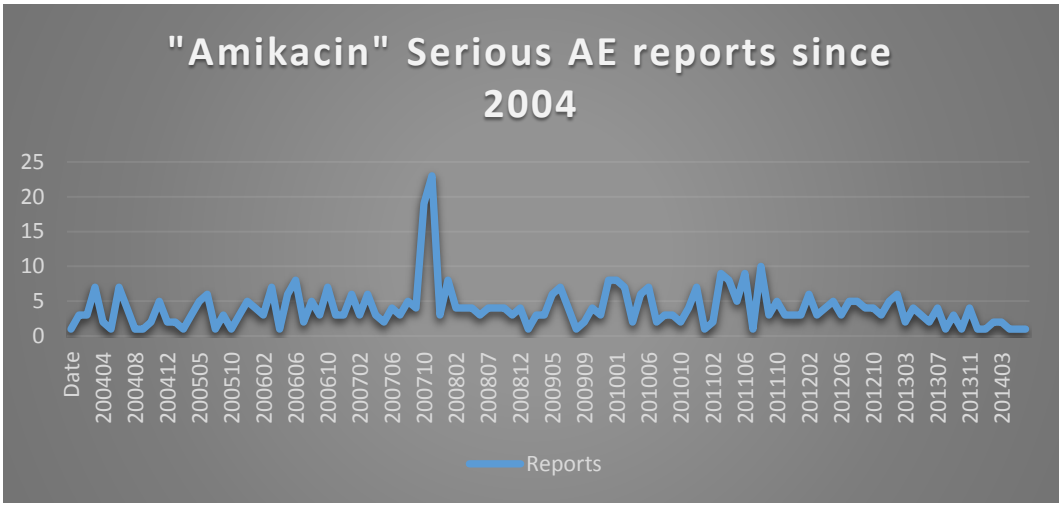
In this example, we want to display serious AE for drug AMICACIN reported overtime. The DS2 code below shows major steps of declare package and define methods.

```
proc ds2;
  data openFDA (overwrite=yes);
    /* Declare JSON package references */
  dcl package json j();
                        ……..
        method parser();
                        ……..
                                  /* iterate over all message entries */
              do while (rc=0);
                        j.getNextToken(rc, token, tokenType, parseFlags);
                        if (token eq 'time') then
                                  do;
                                            j.getNextToken(rc, token, tokenType, parseFlags);
                                            day=token;
                                  end;
                        if (token eq 'count') then
                                  do;
                                            j.getNextToken(rc, token, tokenType, parseFlags);
                                            count=token;
                                            output;
                                  end;
                end;
              return;
                        ……..
        method init();
              dcl package http webQuery();
                        ……..
```

```
            /* using get method to retrieve data */
            webQuery.createGetMethod(
                        'http://api.fda.gov/drug/event.json?search=' ||
                        'patient.drug.openfda.generic_name:%22amikacin%22+AND+serious:1+AND+' ||
                        'receivedate:[20040101+TO+20160101]&count=receivedate');
            /* execute the GET */
            webQuery.executeMethod();
            /* get the response and save it as a string */
            webQuery.getResponseBodyAsString(response, rc);

            rc = j.createParser( response );
            do while (rc = 0);
                        j.getNextToken( rc, token, tokenType, parseFlags);
            parser();
            end;
   end;
                        ……...
      enddata;
run;
quit;
```

The data retrieved from OpenFDA is presented in the following graphic:



Conclusion

We can used to retrieve cloud based clinical data stored in OpenFDA. With more related data portals pop up now and then, well positioned pharmaceutical companies are ready to harness the power of insights and intelligence across pharmaceutical R&D landscape.

Reference

1. SAS Institute, SAS(R) 9.4 DS2 Language Reference, Fifth Edition

2. OpenFDA, https://open.fda.gov

3. Russell Albright, Richard Foley, Listening to the Twitter Conversation , SAS Global Forum 2010

4. Murphy Choy, Kyong Jin Shim, Efficient extraction of JSON information in SAS® using the SCANOVER function, SAS Global Forum 2013

5. Mahdi About, A world of clinical data at your fingertips, PhUSE 2014

6. Isabel Litton, %GrabTweet: A SAS® Macro to Read JSON Formatted Tweets, WUSS 2013

7. Falko Schulz, How to import Twitter tweets in SAS DATA Step using OAuth 2 authentication style, BLOG

8. Chris Hemedinger, Using SAS DS2 to parse JSON, BLOG

Contact Information

Jie Roger Zhou,

University of Bridgeport

170 Lafayette Street, Bridgeport, CT 06604

zhoujie@my.bridgeport.edu, (203)551-4108