

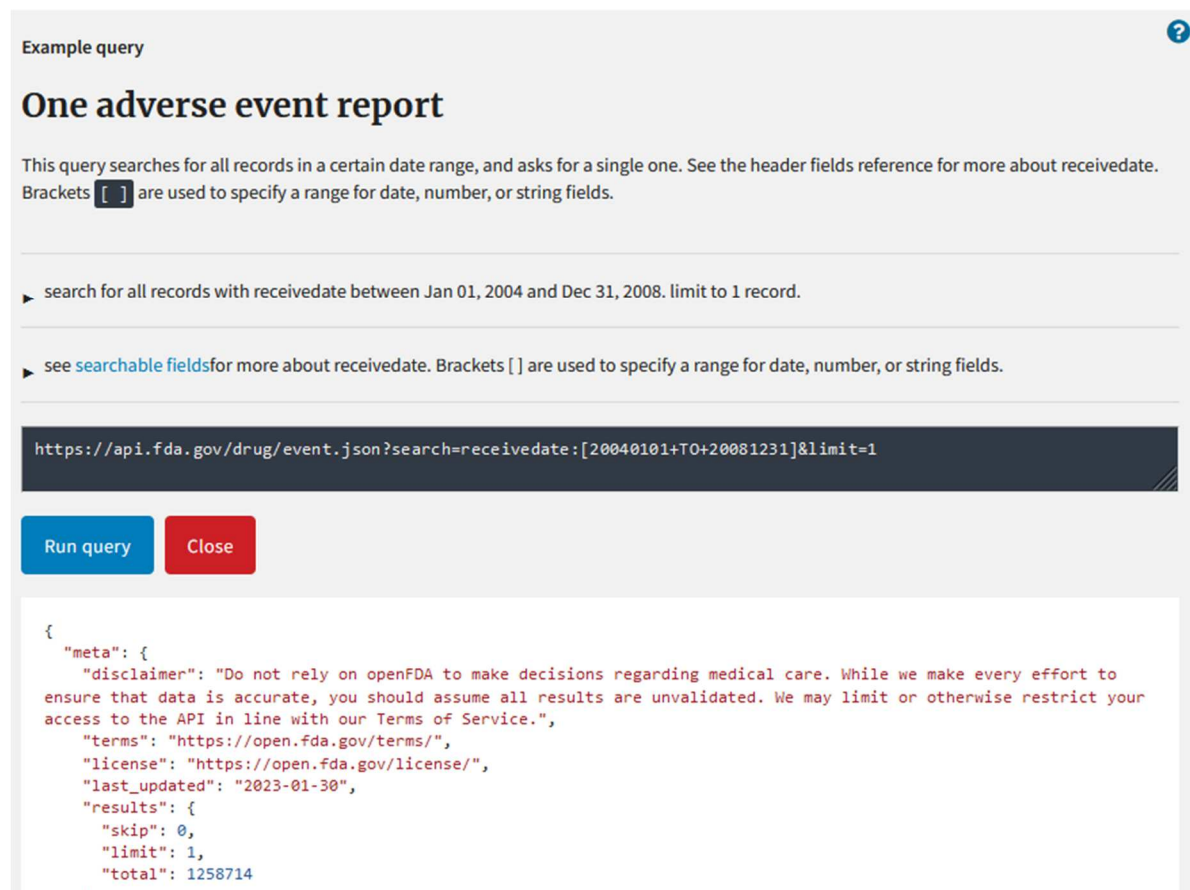
**PharmaSUG 2023 - Paper PO-143**  
**Real Time Analytical Reporting Using OpenFDA**  
**Shubhranshu Dutta, University of Rochester**

## ABSTRACT

One of the main challenges during the drug development process is knowing how a drug in a clinical trial interacts/might interact with other concomitant medications taken by a patient. The FDA has enabled access to real-world data via OpenFDA. With the use of OpenFDA APIs - specifically by creating daily refreshed data reports - we can track the emergence/development of common adverse reactions or drug interactions along with the severity and seriousness of such adverse events across various patients. I will be pulling data from OpenFDA APIs and creating reports that reflect real time data from the OpenFDA database for the purpose of analysis. I will also be exploring the metadata and discussing the interactive charts provided by OpenFDA to help with the right query selection from the database. Utilizing tools such as R libraries, Excel pivot charts and pivot tables, and converting JSON files into data for creating customized reports, I will also be discussing a way of automating reports and notifications to enable a faster alert system for clinical investigators, ensuring greater patient safety.

## INTRODUCTION

OpenFDA has provided many references on how to run a query within their database, few examples are listed below:



The screenshot shows a web interface for an example query. At the top, it says "Example query" with a help icon. The main heading is "One adverse event report". Below this, a description states: "This query searches for all records in a certain date range, and asks for a single one. See the header fields reference for more about receivedate. Brackets [ ] are used to specify a range for date, number, or string fields." There are two bullet points: "search for all records with receivedate between Jan 01, 2004 and Dec 31, 2008. limit to 1 record." and "see searchable fields for more about receivedate. Brackets [ ] are used to specify a range for date, number, or string fields." A dark box contains the API URL: `https://api.fda.gov/drug/event.json?search=receivedate:[20040101+TO+20081231]&limit=1`. Below the URL are two buttons: "Run query" (blue) and "Close" (red). At the bottom, a JSON response is shown, including a disclaimer, terms, license, last updated date, and a results object with skip, limit, and total values.

Figure 1.0: Some of the example API Query calls: from OpenFDA where searchable field is receivedate, in the above query a date range is provided.

Reference: "Example-Api-Queries." open.fda.gov, <https://open.fda.gov/apis/drug/event/example-api-queries/>.

## Adverse drug event reports since 2004

This is the openFDA API endpoint for adverse drug events. An adverse event is submitted to the FDA to report any undesirable experience associated with the use of a drug, including serious drug side effects, product use errors, product quality problems, and therapeutic failures.

Reporting of adverse events by healthcare professionals and consumers is voluntary in the United States. Increases in the total number of adverse events are likely caused by improved reporting, News, enforcement actions, and other phenomena can also spur reporting.

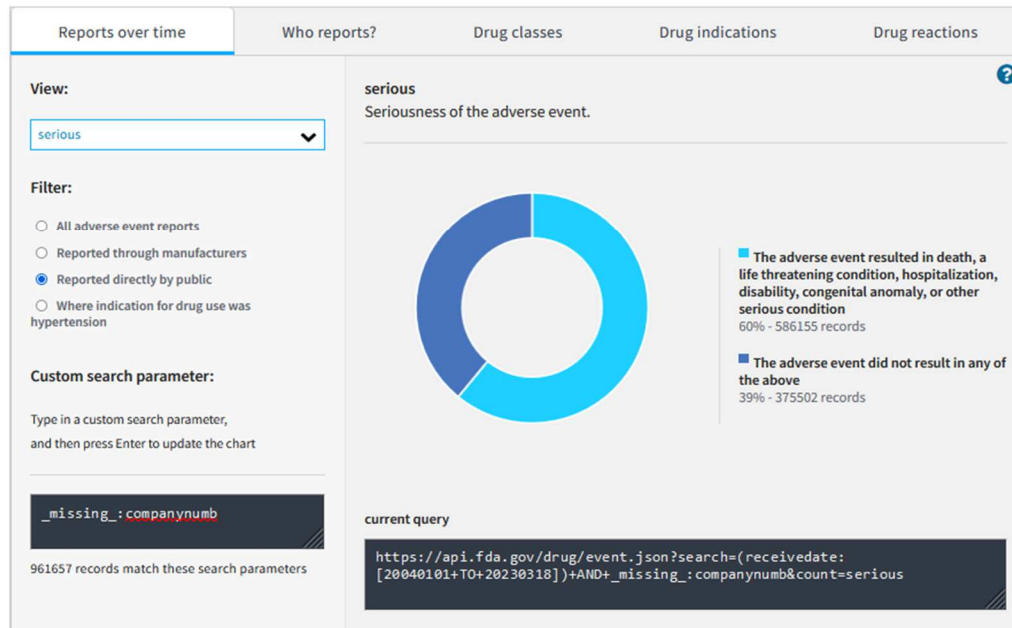


Figure 1.1: Some of the example API Query calls: from OpenFDA where searchable field is serious, in the above query a date range is provided

Reference: “Example-API-Queries.” Open.fda.gov, <https://open.fda.gov/apis/drug/event/example-api-queries/>.

## UNDERSTANDING API CALLS

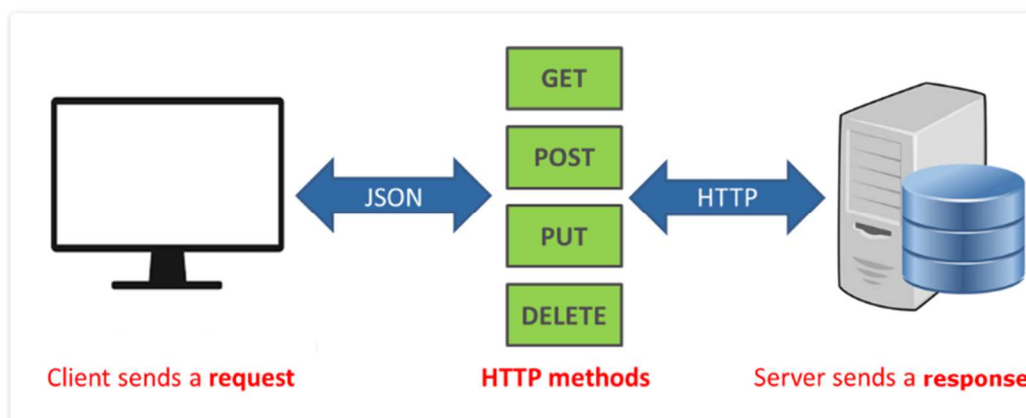


Figure 2.0: API calls are based on a set of commands where the client receives a response from the server or backend database based on HTTP methods – GET, POST, PUT, DELETE

Reference: Jecrespom, Por, and Jecrespom. “Arquitectura API.” Aprendiendo Arduino, <https://aprendiendoarduino.wordpress.com/tag/arquitectura-api/>.

## UNDERSTANDING DATA STRUCTURES USING POSTMAN

Postman is an API platform for building and using APIs. Postman simplifies each step of the API lifecycle and streamlines collaboration so you can create better APIs faster.

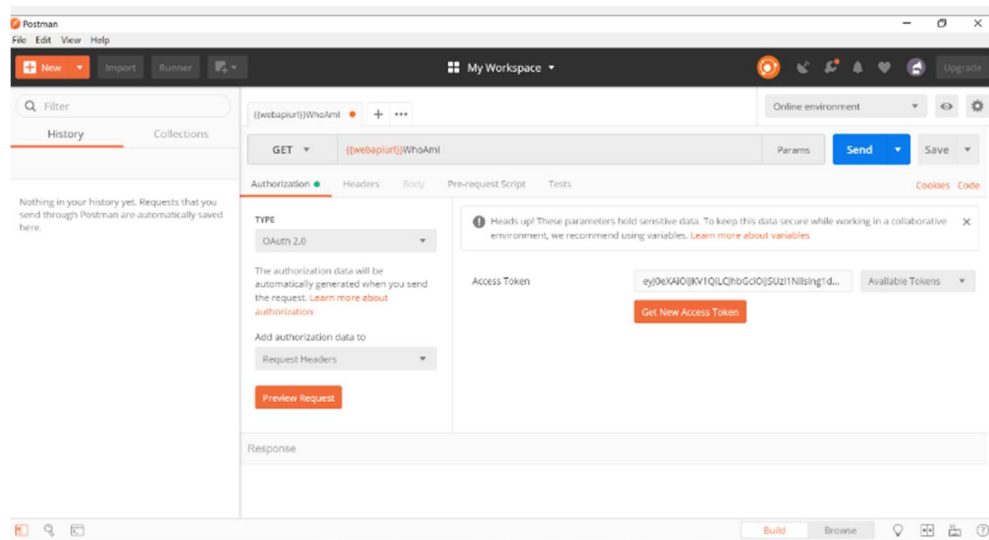


Figure 3.0: The Postman API Platform

Reference: “What Is Postman? Postman Api Platform.” Postman API Platform, <https://www.postman.com/product/what-is-postman/>.

## SAMPLE API RESPONSE

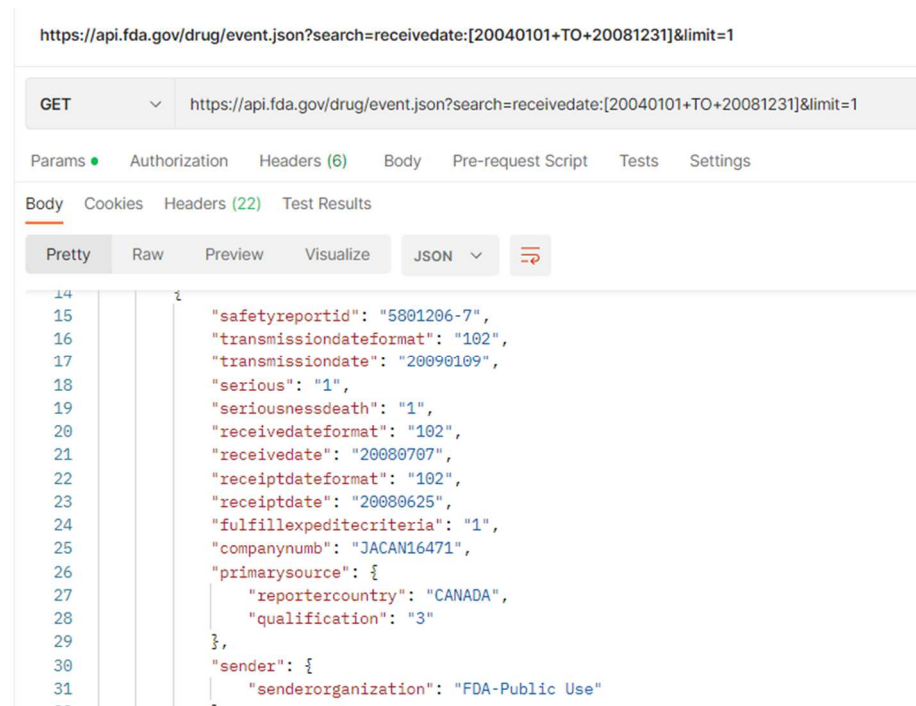


Figure 4.0: Sample API response in Postman

The above response displays the metadata and all values in a single record in Postman.

## REPORTING FROM API RESPONSES

The following programming approach was adopted with R as the main programming language.

### DEFINE PACKAGES AND LIBRARIES

Install the following packages in R and define libraries:

```
library(jsonlite)
library(tidyr)
library(dplyr)
library(openxlsx)
library(lubridate)
```

### DEFINE API DESTINATION

```
Url <- "https://api.fda.gov/drug/event.json?search=receivedate:[20040101+TO+20231231]&limit=100"
response <-
fromJSON("https://api.fda.gov/drug/event.json?search=receivedate:[20040101+TO+20231231]&limit=100")
df <- as.data.frame(response)
#Create a sequence number column to add as placeholder:
df <- df %>% mutate(seq_num = row_number())
```

### DATA WRANGLING - NORMALIZE/READ NESTED COLUMNS

In the OpenFDA API call, the Result column has nested columns for Patient level reaction and drug name. We would now unnest these columns. This will create two separate datasets which we will merge to get the complete unnested data.

```
df_normalized_reaction <- df %>% unnest(result.patient)
colnames(df_normalized_reaction)
df_reaction <- df_normalized_reaction %>% unnest(reaction)
colnames(df_reaction)
```

```
> colnames(df_reaction)
 [1] "meta.disclaimer"           "meta.terms"
 [3] "meta.license"             "meta.last_updated"
 [5] "meta.results.skip"        "meta.results.limit"
 [7] "meta.results.total"       "results.safetyreportid"
 [9] "results.transmissiondateformat" "results.transmissiondate"
[11] "results.serious"          "results.seriousnessdeath"
[13] "results.receivedateformat" "results.receivedate"
[15] "results.receiptdateformat" "results.receiptdate"
[17] "results.fulfillepeditecriteria" "results.companynumb"
[19] "results.primarysource"    "results.sender"
[21] "results.receiver"        "patientonsetage"
[23] "patientonsetageunit"     "patientsex"
[25] "patientdeath"            "reactionmeddrapt"
[27] "reactionmeddraversionpt" "reactionoutcome"
[29] "drug"                     "patientagegroup"
[31] "summary"                  "patientweight"
[33] "results.safetyreportversion" "results.primarysourcecountry"
[35] "results.reporttype"       "results.seriousnessdisabling"
[37] "results.duplicate"        "results.reportduplicate"
[39] "results.seriousnessother" "results.occurcountry"
[41] "results.seriousnesshospitalization" "results.seriousnesslifethreatening"
[43] "seq_num"
```

Figure 5.0: Unnested columns in Results.Patient.reaction

```

head(df_reaction,1)
#Write columns into external file to validate/check nested columns
df_r <- data.frame(df_reaction)
#write.xlsx(df_r, file="test.xlsx")
head(df_r,1)
colnames(df_r)
df_drug<- df_normalized_reaction %>% unnest(drug)
df_drug <- data.frame(df_drug)
colnames(df_drug)
> colnames(df_drug)
 [1] "meta.disclaimer"           "meta.terms"
 [3] "meta.license"             "meta.last_updated"
 [5] "meta.results.skip"        "meta.results.limit"
 [7] "meta.results.total"       "results.safetyreportid"
 [9] "results.transmissiondateformat" "results.transmissiondate"
[11] "results.serious"          "results.seriousnessdeath"
[13] "results.receivedateformat" "results.receivedate"
[15] "results.receiptdateformat" "results.receiptdate"
[17] "results.fulfillpeditecriteria" "results.companynumb"
[19] "results.primarysource"    "results.sender"
[21] "results.receiver"         "patientonsetage"
[23] "patientonsetageunit"      "patientsex"
[25] "patientdeath"             "reaction"
[27] "drugcharacterization"      "medicinalproduct"
[29] "drugauthorizationnumb"     "drugadministrationroute"
[31] "drugindication"           "drugbatchnumb"
[33] "drugstructuredosagenumb"   "drugstructuredosageunit"
[35] "drugdosagetext"           "drugstartdateformat"
[37] "drugstartdate"            "openfda"
[39] "drugrecurreadministration" "drugadditional"
[41] "actiondrug"               "drugdosageform"
[43] "activesubstance"          "drugenddateformat"
[45] "drugenddate"              "drugseparatedosagenumb"
[47] "drugintervaldosageunitnumb" "drugintervaldosagedefinition"
[49] "drugcumulativedosagenumb" "drugcumulativedosageunit"
[51] "patientagegroup"          "summary"
[53] "patientweight"            "results.safetyreportversion"
[55] "results.primarysourcecountry" "results.reporttype"
[57] "results.seriousnessdisabling" "results.duplicate"
[59] "results.reportduplicate"    "results.seriousnessother"
[61] "results.occurcountry"       "results.seriousnesshospitalization"
[63] "results.seriousnesslifethreatening" "seq_num"

```

Figure 6.0: Unnested columns in Results.Patient.Drug

Merge all Data sets having unnested columns:

```

merged_df <- merge(df_drug,df_reaction, by = "seq_num")
colnames(merged_df)
my_df <- data.frame(merged_df)
head(my_df, 3)

```

## REPORT CREATION

We will filter data based on all serious drug related adverse events to generate report:

```
filtered_df <- my_df[my_df$result.serious.x == 1, ]
```

Install packages and create a placeholder for today's date:

```
today <- Sys.Date()
```

```
install.packages("lubridate")
```

```
library(lubridate)
```

```
start_of_month <- as.Date(paste0(format(Sys.Date(), "%Y-%m"), "-01"))
```

```
end_of_month <- as.Date(paste0(format(Sys.Date(), "%Y-%m"), "-31"))
```

```
monthly_report <- filter_df %>% filter(meta.last_updated.x >= start_of_month, meta.last_updated.x <= end_of_month)
```

Report generated as HTML using the

```
---
```

```
title: "Monthly Report"
```

```
author: "Organization Name"
```

```
date: "r format(Sys.Date(), "%B %d, %Y")"
```

```
output:
```

```
html_document:
```

```
keep_md: true
```

```
---
```

Generate monthly report based on

```
monthly_report$medicinalproduct
```

```
monthly_report$FIGURE.company.number.x
```

```
monthly_report
```

## OPENFDA DATA IN EXCEL SPREADSHEET

OpenFDA data can also be downloaded and opened in Excel spreadsheets along with the nested columns and analytical charts/tables may be prepared.

AA	AB	AC	AD
Column1.patient.drug.drugadministrationroute	Column1.patient.drug.drugindication	Column1.safetyreportversion	Column1.primarysourcecountry
048		2	US
048		2	US
	ARTHRITIS	1	US
	ARTHRITIS	1	US
	ARTHRITIS	1	US
048	SYSTEMIC LUPUS ERYTHEMATOSUS	2	BR
065		3	JP
065		3	JP
065		3	JP
065		3	JP
	IMMUNOSUPPRESSION	1	CN
	IMMUNOSUPPRESSION	1	CN
	IMMUNOSUPPRESSION	1	CN

Figure 7.0: Normalized columns in \*.xlsx format

1	Column1.safetyreportid	Count of Column1.seriousnessdeath	Column Labels	Grand Total
2	Column1.transmissiondateformat	Row Labels	2	28
3	Column1.transmissiondate	ABBVIE		28
4	Column1.serious	3TC		
5	Column1.seriousnessdeath	ABILIFY		
6	Column1.receivedateformat	ACCUZIDE	1	1
7	Column1.receivedate	ACETAMINOPHEN		
8	Column1.receiptdateformat	ACTONEL		
9	Column1.receiptdate	ACTRAPID	1	1
10	Column1.fulfillexpeditecriteria	ADVAIR HFA		
11	Column1.companynumb	ADVIL		
12	Column1.primarysource.reportercountry	ALCOHOL		
13	Column1.primarysource.qualification	ALENDRONIC ACID		
14	Column1.sender.senderorganization	ALLOPURINOL	1	1
15	Column1.sender.sendertype	ALPRAZOLAM		
16	Column1.receiver.receiverstype	ALVEDON		
17	Column1.receiver.receiverorganization	AMBROXOL HYDROCHLORIDE		
18	Column1.patient.patientonsetage	AMLODIPINE		
19	Column1.patient.patientonsetageunit	AMOXYCILLIN		
20	Column1.patient.patientsex	ANALGETIC DRUG		
21	Column1.patient.patientdeath.patientdeathdateformat	ANGIOS (CALCIUM)		
22	Column1.patient.patientdeath.patientdeathdate	ANTI TUBERCULOSIS DRUG (NOT SPECIFIED)		
23	Column1.patient.reaction.reactionmeddrapt	ANTIBIOTICS		
24	Column1.patient.drug.drugcharacterization	ANTIHISTAMINES		
25	Column1.patient.drug.medicinalproduct			
26	Column1.patient.drug.drugauthorizationnumb			
27	Column1.patient.drug.drugadministrationroute			
28	Column1.patient.drug.drugindication			
29	Column1.safetyreportversion			
30	Column1.primarysourcecountry			
31	Column1.reporttype			
32	Column1.seriousnessdisabling			
33	Column1.duplicate			
34	Column1.reportduplicate.duplicatesource			
35	Column1.reportduplicate.duplicatenumb			

Figure 7.2: Pivot Chart of serious adverse events by drug name and manufacturer name

Column1.term	Column1.count
WATSON LABS	736
MYLAN	400
TEVA	306
SANDOZ	300
STRIDES PHARMA	248
ANI PHARMS	229
HIKMA	194
CHARTWELL RX	186
AUROBINDO PHARMA	180
IVAX SUB TEVA PHARMS	174
AUROBINDO PHARMA LTD	169
SUN PHARM INDUSTRIES	161
BARR	156
LUPIN LTD	155
RISING	142
ZYDUS PHARMS	138
PUREPAC PHARM	126
AMNEAL PHARMS	124
ROXANE	123

Figure 7.1: List of column names

Figure 7.3: Record counts

## CONCLUSION

Reporting with public APIs first requires an understanding of the API structure, followed by data manipulation to unnest any hidden columns. This may be done easily with R and other tools such as Python that have custom functions to read JSON API's. POSTMAN is a valuable tool that may serve as a guide in this process. Daily reporting with R may be done programmatically within R and also using CRON jobs too. This would ensure data parameter/filtered reporting.

## REFERENCES

1. Jecrespom, Por, and Jecrespom. “Arquitectura API.” *Aprendiendo Arduino*, <https://aprendiendoarduino.wordpress.com/tag/arquitectura-api/>.
2. “Example-Api-Queries.” *Open.fda.gov*, <https://open.fda.gov/apis/drug/event/example-api-queries/>.
3. What Is Postman? Postman Api Platform.” *Postman API Platform*, <https://www.postman.com/product/what-is-postman/>.

## RECOMMENDED READING

- Kass-Hout, Taha A, et al. “OpenFDA: An Innovative Platform Providing Access to a Wealth of FDA's Publicly Available Data.” *Journal of the American Medical Informatics Association: JAMIA*, U.S. National Library of Medicine, May 2016, <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4901374/>.
- Lee, Jaemin. “Creating Reports with R Markdown.” *Medium*, Towards Data Science, 22 June 2020, <https://towardsdatascience.com/creating-reports-with-r-markdown-c6031ecdd65c>.

## CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Shubhranshu Dutta  
University of Rochester  
Email: [shudutt@gmail.com](mailto:shudutt@gmail.com)

Any brand and product names are trademarks of their respective companies.