TECHNICAL DETAILS OF METHODOLOGY

By Craig Paardekooper

VACCINES

DATA SOURCE AND FILES

[VAERS Nov 11th Downloadable files (vaersaware.com)]

READING THE VAERSVAX FILES

```
import pandas as pd
import matplotlib.pyplot as plt
import folium
import os, re
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import normalize
from IPython.display import IFrame
from sklearn.cluster import AgglomerativeClustering
import scipy.cluster.hierarchy as shc
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

```
dfList2=[]


for root,dirs,files in os.walk(r"C:\Users\User\Downloads\AllVAERSDataCSV2223"):
    for fname in files:


        if "VAERSVAX" in fname:
            try:

                frame = pd.read_csv(r"C:\Users\User\Downloads\AllVAERSDataCSV2223\\" + fname, encoding='windows-1252')
                frame1 = frame[['VAERS_ID','VAX_TYPE']]

                dfList2.append(frame1)
            except:
                print(fname)

datasetvax = pd.concat(dfList2)


datasetvax['VAERS_ID'].count()
```

STATS ON THE DATA

2,462,044 unique VAERS IDs [**datasetvax**['VAERS_ID'].nunique()]

2,987,454 total number of VAERS Ids [**datasetvax**['VAERS_ID'].count()]

525,410 are VAERS Ids are duplicated

REMOVING DUPLICATES

Why are they duplicated? Because they represent two or more different vaccines that a person had at the same time. Taking multiple medicines makes it hard to attribute adverse effects to a particular medicine, so these records are removed. When they are dropped, we have 2,144,669 unique VAERS IDs and 2,144,669 VAERS IDs in total. So now we just have one of each VAERS ID. This means that we have 2,144,669 records where each person took one vaccine and had one set of adverse symptoms. This is ideal for pharmacovigilance.

```
datasetvax = datasetvax.drop_duplicates(subset=['VAERS_ID'],keep=False)
datasetvax['VAERS_ID'].count()
```

[This could be further refined selecting those on no medications at the time of vaccination, and those with no pre-existing illnesses – which would require looking at the VAERSDATA table]

REMOVING VACCINES WITH TOO FEW REPORTS

Here are the number of reports for each vaccine in VAERSVAX. When we do the statistical analysis, those vaccines that have fewer reports will generate a much wider standard deviation and confidence interval.

[**datasetvax**['VAX_TYPE'].value_counts()]

COVID19	1433820	HIBV	6981	FLUA4	1929
VARZOS	105176	ANTH	6185	SMALLMNK	1753
FLU3	71603	FLUC4	5335	YF	1665
HEP	43913	FLUN3	5270	MEN	1626
PPV	42677	RAB	4748	DTAPHEPBIP	1291
VARCEL	41320	HPV2	4581	PNC20	1236
HPV4	36734	PNC	4210	DT	1229
COVID19-2	33261	RV1	4174	FLUC3	1123
FLU4	31183	TYP	3995	DTPHIB	836
MMR	27457	DTAPIPV	3987	RUB	825
FLUX	23726	SMALL	3860	MEA	522
TDAP	21321	FLUN4	3859	RSV	482
UNK	16676	HEPAB	3758	RVX	464
PNC13	14155	DTP	2902	HBHEPB	338
HPV9	13916	FLUN(H1N1)	2607	JEV	290
HEPA	13599	DTAPIPVHIB	2566	BCG	290
RV5	11949	FLUA3	2437	DTPPVHBHPB	286
TD	11878	TTOX	2394	FLUR3	241
DTAP	10903	IPV	2347	DF	219
MMRV	10135	FLUR4	2320	JEV1	197
MNQ	9425	LYME	2189	OPV	192
MENB	8542	FLUX(H1N1)	2150	6VAX-F	189
FLU(H1N1)	7735	HPVX	2045	RV	184

PNC15	123	EBZR	44	HBPV	20
MU	121	DTPIPV	37	DTPIHI	19
DTAPH	117	TBE	36	DTOX	18
CHOL	108	CEE	30	HEPATYP	16
MER	107	DTPHEP	28	MUR	9
PER	69	DTIPV	26	MENHIB	6
ADEN_4_7	65	JEVX	25	H5N1	4
MM	56	DTPPHIB	21	MNC	4
DPP	53	PLAGUE	20	SSEV	3
TDAPIPV	50	PNC10	20	MNQHIB	3

SYMPTOMS

There are 5 symptoms columns in VAERS – labelled SYMPTOM1, SYMPTOM2, SYMPTOM3, SYMPTOM4, SYMPTOM5. There can be more than one row of symptoms for each VAERS ID

READING THE VAERS SYMPTOMS FILES

```
dfList3=[]
#C:\Users\User\Downloads\AllVAERSDataCSVS
for root,dirs,files in os.walk(r"C:\Users\User\Downloads\AllVAERSDataCSV2223"):
    for fname in files:

        if "VAERSSYMPTOMS" in fname:
            try:

                frameY = pd.read_csv(r"C:\Users\User\Downloads\AllVAERSDataCSV2223\\" + fname, encoding='windows-1252')

                frame1 = frameY[['VAERS_ID','SYMPTOM1']]
                frame2 = frameY[['VAERS_ID','SYMPTOM2']]
                frame3 = frameY[['VAERS_ID','SYMPTOM3']]
                frame4 = frameY[['VAERS_ID','SYMPTOM4']]
                frame5 = frameY[['VAERS_ID','SYMPTOM5']]


                frame2 = frame2.rename(columns={'SYMPTOM2': 'SYMPTOM1'})
                frame3 = frame3.rename(columns={'SYMPTOM3': 'SYMPTOM1'})
                frame4 = frame4.rename(columns={'SYMPTOM4': 'SYMPTOM1'})
                frame5 = frame5.rename(columns={'SYMPTOM5': 'SYMPTOM1'})
                concatenated = pd.concat([frame1, frame2, frame3, frame4, frame5])
                dfList3.append(concatenated)

            except:
                print(fname)
datasetsymptoms = pd.concat(dfList3)

datasetsymptoms = datasetsymptoms.dropna(how='any')

datasetsymptoms['VAERS_ID'].count()
```

REMOVAL OF NULL VALUES FROM SYMPTOM1 COLUMN

There are a total count of 16,393,005 records in **datasetsymptoms** found by counting the VAERS-IDs. However this is comprised of comprised of 10,361,637 symptoms and 6,031,368 null values.

- 16,393,005 symptom records
- 10,361,637 symptoms
- 6,031,368 nulls

These null values were removed leaving –

- 10,361,637 symptom records where there were
- 10,361,637 symptoms
- 17,428 unique symptoms
- 2,461,784 unique VAERS-IDs

```
datasetsymptoms = datasetsymptoms.dropna(how='any')
```

These are the symptoms for 2,461,784 unique VAERS-IDs, which closely corresponds to the original number of VAER S-IDs in the VAERSVAX table before duplicates were removed.

COUNTING THE FREQUENCY OF EACH SYMPTOM AND EXPORTING TO A DATAFRAME

```
countdf = datasetsymptoms['SYMPTOM1'].value_counts().rename_axis('unique_values').reset_index(name='counts')
```

unique_values	counts
Pyrexia	325258
Headache	283134
SARS-CoV-2 test	230767
Fatigue	230416
COVID-19	207079
Pain	205379
Chills	171740
Nausea	163297
Pain in extremity	159258
Dizziness	159018
No adverse event	123536
Injection site pain	120538
Injection site erythema	120165
Myalgia	120112
Rash	113874
Dyspnoea	112684
Arthralgia	108883
Vomiting	92563

MERGING

The **datsetvax** table was then merged with the **datasetsymptoms** table on the common field of VAERS-ID, so we end up with -

- 9,020,372 records
- 2,144,512 unique (VAERS_ID)
- 16849 unique (SYMPTOM1)
- 99 unique vaccines (VAX_TYPE)
- averaging 4.2 symptoms per report (VAERS_ID).

```
vaxsym = datasetvax.merge(datasetsymptoms[['VAERS_ID', 'SYMPTOM1']])
```

The VAXSYM dataset is the raw data, listing a count of every symptom for each vaccine. It has this format –

	VAERS_ID	VAX_TYPE	SYMPTOM1
0	25001	DTP	Agitation
1	25004	OPV	Chills
2	25004	OPV	Dermatitis contact
3	25004	OPV	Oedema genital
4	25004	OPV	Pelvic pain
5	25005	TD	Arthritis
6	25005	TD	Injection site oedema
7	25005	TD	Injection site reaction
8	25007	TD	Injection site inflammation
9	25007	TD	Injection site reaction
10	25008	TD	Injection site inflammation

It can be downloaded here, as a csv file called “raw.csv” Unfortunately it has 9 million rows so cannot be opened in e xcel (Excel has a limit of 1 million rows). However, you will be able to read it with Python, into a Jupyter Notebook, a nd carry out any analysis there.

https://www.howbad.info/safety-signal-datasets3.zip

SYMPTOMS PER RECORD

It is interesting to look at number of symptoms recorded for each vaccine-type and divide this by the number of VAE RS_IDs for that type to get the average number of symptoms per record. It is hypothesised that some vaccines may h ave a greater number of symptoms per record because their toxicity is more intense, more distributed or more persi stent. A more toxic vaccine would produce a greater incidence of the same symptom, whilst a more bio-distributed v accine would produce a greater variety of symptoms throughout the body. A more persistent vaccine would produce a incidence of adverse effects spread over a longer period.

This result can then be compared with the number of symptoms per record for other vaccines to get a PRR score.

The raw data for vaccines may show that a greater number of symptoms are reported for each COVID report compar ed to other vaccines. This could be because the vaccine was causing more ailments because –

1. it’s bio distribution was greater ,
2. it’s permeation of cells was greater,

3. it’s duration of effect more prolonged and
4. it’s effects more intense owing to the biologically active nature of the spike antigen which is a proven toxin.

In comparison a short lived, local, non-toxic , biologically inactive antigen would be expected to generate fewer symp toms.

What is the maximum and minimum number of symptoms recorded per record, what is the average and standard de viation ?

GROUPING BY PIVOT TABLE

```
fsym = vaxsym.pivot_table(index='SYMPTOM1', columns='VAX_TYPE',
aggfunc=len, fill_value=0)

fsym.to_csv(r"C:\Users\User\Downloads\vaccine-symptoms2.csv")
```

The FSYM dataset is a pivot table, so it takes the single column of symptoms from the raw dataset and groups them by vaccine. This is very useful for seeing the frequency of occurrence of each symptom for each vaccine – so you can see how many datapoints analysis is based upon. You can download it as a CSV file called “A.csv”. A transposed versi on is also available called “tA.csv”. “A.csv” has 16849 rows and 99 columns, so it can be opened in Excel.

https://www.howbad.info/safety-signal-datasets3.zip

CONVERTING RAW COUNTS TO A PRR RATIO

The PRR ratio is calculated by

1. Counting the frequency of symptom S for vaccine V (a)
2. Dividing this by the total symptom count for vaccine V (a + b)

   [This gives the % of symptoms for vaccine V that are symptom S]
3. Counting the frequency of symptom S for all other vaccines (c)
4. Dividing this by the total symptom count for all other vaccines (c + d)

   [This gives the % of symptoms for all other vaccines that are symptom S]
5. PRR = % of symptoms for vaccine V that are symptom S divided by % of symptoms for all other vaccines that are symptom S

```
import numpy as np

#CALCULATING PRR

df1 = fsym.apply(lambda x: x/sum(x), axis=0)  # a / a + b
df2 = fsym.apply(lambda x: sum(x) - x, axis = 1) # c
df3 = fsym.apply(lambda x: 9020372 - sum(x), axis=0) # c + d
dfA = df1.div(df2.values)
dfB = pd.DataFrame(dfA.values*df3.values, columns=fsym.columns, index=fsym.index) # PRR
```

```
A = fsym
AB = fsym.apply(lambda x: sum(x), axis=0)
C = df2
CD = df3
PRR = dfB
```

Each vaccine is shown as a separate column, and each row is a separate symptom. You can use this dataset to quickly see which symptoms are disproportionately associated with each vaccine.

You can download the dataset here, as a file called “PRR.csv”.
https://www.howbad.info/safety-signal-datasets3.zip

The 4 numbers used to calculate the PRR ratio, namely A, A + B, C and C + D, are each available as a csv file here - https://www.howbad.info/safety-signal-datasets3.zip, so you can spot check the calculations.

A	“A.csv”
A + B	“AB.csv”
C	“C.csv”
C + D	“CD.csv”

A transposed version of each of these files is also available

A	“tA.csv”
A + B	“tAB.csv”
C	“tC.csv”
C + D	“tCD.csv”

TRANSPOSING DATA

```
tdf = dfB.T
tdf.head()
```

This dataset has a separate column for each symptom, and a separate row for each vaccine. It is useful for quickly seeing which vaccines are worst for any chosen symptom, since you can sort each column by PRR ratio.

Please note that because this dataset has 16,849 columns, it may not fully load into Excel – but will truncate the sym ptom columns after “V”, but the file is perfectly useful for the remaining columns.

You can download this dataset here – https://www.howbad.info/safety-signal-datasets3.zip, as a file called “tPRR.csv”

CALCULATING LOWER LIMIT OF THE 95% CONFIDENCE INTERVAL

```
import numpy as np

#CALCULATING PRR

df1 = fsym.apply(lambda x: x/sum(x), axis=0)  # a / a + b
df2 = fsym.apply(lambda x: sum(x) - x, axis = 1) # c
df3 = fsym.apply(lambda x: 9020372 - sum(x), axis=0) # c + d
dfA = df1.div(df2.values)
dfB = pd.DataFrame(dfA.values*df3.values, columns=fsym.columns, index=fsym.index) # PRR
```

```
A = fsym
AB = fsym.apply(lambda x: sum(x), axis=0)
C = df2
CD = df3
PRR = dfB
```

```
InvA = 1/A
InvC = 1/C
InvAB = 1/AB
InvCD = 1/CD


Ssquared = InvA + InvC - InvAB - InvCD
S = np.sqrt(Ssquared)
LCI = dfB/np.exp(1.96*S) # lower CI 95%
expo = np.exp(1.96*S)

UCI = PRR.values * expo
```

```
tA = A.T
tAB = AB.T
tC = C.T
tCD = CD.T
tPRR = PRR.T
tS = S.T
tLCI = LCI.T
tUCI = UCI.T
```

```
fsym.to_csv(r"C:\Users\User\Downloads\NumbersNew\raw.csv")
A.to_csv(r"C:\Users\User\Downloads\NumbersNew\A.csv")
AB.to_csv(r"C:\Users\User\Downloads\NumbersNew\AB.csv")
C.to_csv(r"C:\Users\User\Downloads\NumbersNew\C.csv")
CD.to_csv(r"C:\Users\User\Downloads\NumbersNew\CD.csv")
PRR.to_csv(r"C:\Users\User\Downloads\NumbersNew\PRR.csv")
S.to_csv(r"C:\Users\User\Downloads\NumbersNew\S.csv")
LCI.to_csv(r"C:\Users\User\Downloads\NumbersNew\LCI.csv")
UCI.to_csv(r"C:\Users\User\Downloads\NumbersNew\UCI.csv")
```

```
tA.to_csv(r"C:\Users\User\Downloads\t-NumbersNew\tA.csv")
tAB.to_csv(r"C:\Users\User\Downloads\t-NumbersNew\tAB.csv")
tC.to_csv(r"C:\Users\User\Downloads\t-NumbersNew\tC.csv")
tCD.to_csv(r"C:\Users\User\Downloads\t-NumbersNew\tCD.csv")
tPRR.to_csv(r"C:\Users\User\Downloads\t-NumbersNew\tPRR.csv")
tS.to_csv(r"C:\Users\User\Downloads\t-NumbersNew\tS.csv")
tLCI.to_csv(r"C:\Users\User\Downloads\t-NumbersNew\tLCI.csv")
tUCI.to_csv(r"C:\Users\User\Downloads\t-NumbersNew\tUCI.csv")
```

The standard deviation for each symptom and each vaccine is found in the file called “S.csv”, and its transposition in the file called “tS.csv”

The file called “LCI.csv” shows lower limit of the 95% confidence interval of the PRR for each symptom and for each vaccine. The transposed version of this file is called “tLCI.csv”

The file called “UCI.csv” shows upper limit of the 95% confidence interval of the PRR for each symptom and for each vaccine. The transposed version of this file is called “tUCI.csv”

FILTERING THE LCI DATAFRAMES

Finally, I wanted to filter the LCI data-frames to remove entries where the number of reported incidents was less than 10, and where the LCI values were less than 2.

This was done for the purpose of removing noise caused by values that were not statistically significant. The result was an LCI dataset where strong patterns were more discernible.

```
dfX = A.applymap(lambda x: 1 if x >=10 else 0)
dfX.head(200)
```

```
dfXX = pd.DataFrame(LCI.values*dfX.values, columns=LCI.columns, index=LCI.index)
dfXX.head(200)
dfXXX = dfXX.applymap(lambda x: x if x >=2 else 0)
tdfXXX = dfXXX.T
dfXXX.to_csv(r"C:\Users\User\Downloads\NumbersNew\filtered.csv")
tdfXXX.to_csv(r"C:\Users\User\Downloads\NumbersNew\tfiltered.csv")
```

The resulting files – filtered.csv and tfiltered.csv, are the data used for Safety Signal search engine.

You can download these CSV files here – https://www.howbad.info/safety-signal-datasets3.zip

GENERATING RANDOM SAMPLES FOR DIFFERENT VACCINES FOR COMPARISON OF SYMPTOMS

The number of symptom records for each vaccine is shown below – the total symptom count comes to 9,020,372.

COVID19	6452217	LYME	13252	OPV	677
VARZOS	365754	TYP	12634	DTPPVHBHPB	583
FLU3	269177	DTAPIPV	11466	MER	472
PPV	163717	FLUN4	10956	RV	459
HPV4	157025	FLUX(H1N1)	9854	DTAPH	394
HEP	156741	HPVX	9816	MU	359
COVID19-2	118050	DTP	9409	PNC15	353
FLU4	114504	DTAPIPVHIB	9395	CHOL	345
FLUX	105434	FLUA3	9178	TDAPIPV	326
VARCEL	103508	TTOX	8990	DTIPV	281
MMR	95204	FLUN(H1N1)	8709	PER	256
TDAP	89169	IPV	8397	DTPIPV	253
UNK	61811	YF	7810	MM	247
PNC13	56511	MEN	6835	TBE	238
TD	45393	FLUR4	6777	ADEN_4_7	226
HPV9	40958	FLUA4	5994	DPP	220
HEPA	37513	PNC20	5253	EBZR	191
DTAP	37304	SMALLMNK	4931	CEE	178
MENB	32541	DT	4372	DTPHEP	128
RV5	30921	FLUC3	3712	DTOX	117
FLU(H1N1)	30404	DTAPHEPBIP	3581	DTPPHIB	106
MNQ	30304	RUB	3060	JEVX	102
HPV2	29976	DTPHIB	2445	PLAGUE	97
ANTH	29721	MEA	2076	PNC10	86
MMRV	23436	RVX	2011	HEPATYP	77
RV1	20802	DF	1888	DTPIHI	73
HIBV	20204	RSV	1748	HBPV	53
RAB	19566	BCG	1721	H5N1	26
FLUC4	19235	JEV	998	MUR	23
SMALL	16786	FLUR3	884	MENHIB	22
HEPAB	16729	6VAX-F	879	MNQHIB	16
FLUN3	16471	JEV1	824	SSEV	16
PNC	15616	HBHEPB	801	MNC	14

When one vaccine has a high PRR for a particular symptom, we can check to see if that high PRR is consistent for eve ry sample of that vaccine.

In the code shown below, I am comparing samples of COVID19 vaccine with samples of FLU3 vaccine. The smples of both contain exactly the same number of symptom records – 20,000 in each – taken at random from the total numb er of records for that vaccine.

```
Covid = vaxsym.loc[vaxsym['VAX_TYPE'] == 'COVID19']
Flu = vaxsym.loc[vaxsym['VAX_TYPE'] == 'FLU3']

for i in range(0,100):
    Cov = Covid.sample(n = 20000)
    Fov = Flu.sample(n= 20000)
    p = Cov.loc[Cov['SYMPTOM1'] == 'Thrombosis']
    q = Fov.loc[Fov['SYMPTOM1'] == 'Thrombosis']
    num = p['VAERS_ID'].count()/q['VAERS_ID'].count()
    print('{:4.2f}'.format(num) , "   Counts = " , p['VAERS_ID'].count(), " | " , q['VAERS_ID'].count())
```

FULL CODE FILE

Here you can find the complete Python code that was used to carry out this analysis.

Python Code for Vaccine Analysis