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\AllVAERSDataCSVS"):
    for fname in files:


        if "VAERSVAX" in fname:
            try:

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

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

datasetvax = pd.concat(dfList2)
```

STATS ON THE DATA

2352562 unique VAERS IDs [**datasetvax**['VAERS_ID'].nunique()]

2856247 total number of VAERS Ids [**datasetvax**['VAERS_ID'].count()]

503685 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 2049804 unique VAERS IDs and 2049804 VAERS IDs in total. So now we just have one of each VAERS ID. This means that we have 2049804 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. If we are doing a statistical analysis, those vaccines that have fewer than 100 reports will generate unreliable scores, so should be dropped. I haven't done this in the current analysis.

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

COVID19	1391183	FLUN4	3815	RV	184
VARZOS	98254	HEPAB	3504	DTPPVHBHPB	150
FLU3	71582	DTAPIPV	3367	MU	121
HEP	42877	DTP	2899	DTAPH	117
PPV	42262	FLUN(H1N1)	2607	MER	107
VARCEL	40604	FLUA3	2437	CHOL	107
HPV4	36590	DTAPIPVHIB	2414	PER	69
FLU4	29085	TTOX	2383	ADEN_4_7	61
MMR	26953	LYME	2188	MM	56
FLUX	23054	FLUX(H1N1)	2150	DPP	53
TDAP	20383	IPV	2134	TDAPIPV	50
UNK	15168	FLUR4	2081	EBZR	44
PNC13	13973	HPVX	2043	DTPIPV	37
HPV9	12957	YF	1553	TBE	34
HEPA	12618	MEN	1543	PNC15	30
TD	11715	FLUA4	1467	CEE	30
RV5	11673	SMALLMNK	1227	DTPHEP	28
DTAP	10456	DT	1221	DTIPV	26
MMRV	9443	DTAPHEPBIP	1138	JEVX	25
MNQ	8620	FLUC3	1083	DTPPHIB	21
MENB	8030	DTPHIB	836	HBPV	20
FLU(H1N1)	7735	RUB	825	PLAGUE	20
COVID19-2	6641	MEA	522	PNC10	20
HIBV	6520	PNC20	465	DTPIHI	19
ANTH	6148	RVX	448	DTOX	18
FLUN3	5269	HBHEPB	338	HEPATYP	16
FLUC4	5037	JEV	290	MUR	9
RAB	4596	BCG	284	MENHIB	6
HPV2	4580	FLUR3	225	H5N1	4
PNC	4210	DF	217	MNC	4
RV1	4062	OPV	192	SSEV	3
TYP	3908	JEV1	191	MNQHIB	3
SMALL	3851	6VAX-F	188		

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=[]

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

        if "VAERSSYMPTOMS" in fname:
            try:

                frameY = pd.read_csv(r"C:\Users\User\Downloads\AllVAERSDataCSVS\\" + 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)
```

REMOVAL OF NULL VALUES FROM SYMPTOM1 COLUMN

There are a total count of 15682620 records in **datasetsymptoms** found by counting the VAERS-IDs. However this is c omprised of comprised of 9959381 symptoms and 5723239 null values.

- 15682620 symptom records
- 9959381 symptoms
- 5723239 nulls

These null values were removed leaving –

- 9959381 symptom records where there were
- 9959381 symptoms
- 2352303 VAERS-IDs

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

These are the symptoms for 2352303 unique VAERS-IDs, which closely corresponds to the original number of VAERS-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	318381
Headache	277176
SARS-CoV-2 test	227363
Fatigue	223902
Pain	199188
COVID-19	189505
Chills	168428
Nausea	160436
Dizziness	155965
Pain in extremity	155492
Injection site erythema	118251
Myalgia	118157
Injection site pain	118016
Rash	111443
Dyspnoea	108718
Arthralgia	106531
No adverse event	106449
Vomiting	90657

MERGING

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

- 8685997 records
- 2049647 unique (VAERS_ID)
- 16575 unique (SYMPTOM1)
- 98 unique vaccines (VAX_TYPE)
- averaging 4.237 symptoms per report (VAERS_ID).

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

The VAXSYM dataset is the raw data, listing every symptom and its associated vaccine. It can be downloaded here, as a csv file. Unfortunately it has 8.7 million rows so cannot be opened in excel (Excel has a limit of 1 million rows). How ever, you will be able to read it with Python into a Jupyter Notebook, and carry out any analysis there.

https://howbad.info/vaxsym.zip (67.3 Mb)

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.

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 useful for seeing the frequency of occurrence of each symptom for each vaccine – so you can see how many datapoints analysis is based upon. This file can be opened I Excel.

https://howbad.info/fsym.zip (0.35 Mb)

CONVERTING RAW COUNTS TO A PRR RATIO

The PRR ratio is calculated by

1. Counting the frequency of symptom S for vaccine V
2. Dividing this by the total symptom count for vaccine V

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

   [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

```
df1 = fsym.apply(lambda x: x/sum(x), axis=0)
df2 = fsym.apply(lambda x: sum(x) - x, axis = 1)
df3 = fsym.apply(lambda x: 8685997 - sum(x), axis=0)
dfA = df1.div(df2.values)
dfB = pd.DataFrame(dfA.values*df3.values, columns=fsym.columns, index=fsym.index)
```

```
dfB.to_csv(r"C:\Users\User\Downloads\all-sym-vax-safety2.csv")
```

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 -

https://howbad.info/grouped-by-vaccine.zip (1.6 Mb)

TRANSPOSING DATA

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

```
tdf.to_csv(r"C:\Users\User\Downloads\prr-ratios-all-vaccines.csv")
```

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 16575 columns, it may not fully load into Excel.

You can download this dataset here – https://howbad.info/grouped-by-symptoms.zip (1.2 Mb)

GENERATING RANDOM SAMPLES FOR DIFFERENT VACCINES FOR COMPARISON OF SYMPTOMS

```
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