




Tutorial

Set Operations in Python for Translational Medicine

Yoshiyasu Takefuji 

Data Science Department, Musashino University, Tokyo 1358181, Japan; takefuji@keio.jp or y-take@musashino-u.ac.jp

Abstract: This is the world's first tutorial article on Python programming on set operations for beginners and practitioners in translational medicine or medicine in general. This tutorial will allow researchers to demonstrate and showcase their tools on PyPI packages around the world. Via the PyPI packaging, a Python application with a single source code can run on Windows, MacOS, and Linux operating systems. In addition to the PyPI packaging, the reproducibility and quality of the source code must be guaranteed. This paper shows how to publish the Python application in Code Ocean after the PyPI packaging. Code Ocean is used in IEEE, Springer, and Elsevier for software reproducibility validation. First, programmers must understand how to scrape a dataset over the Internet. Second, the dataset files must be read in Python. Third, a program must be built to compute the target values using set operations. Fourth, the Python program must be converted to the PyPI package. Finally, the PyPI package is uploaded. Code Ocean plays a key role in publishing validation for software reproducibility. This paper depicts a vaers executable package as an example for calculating the number of deaths due to COVID-19 vaccines. Calculations were based on gender (male and female), age group, and vaccine group (Moderna, Pfizer, and Novartis), respectively.

Keywords: translational medicine; PyPI package; Python program; dataset; Code Ocean; VAERS



Citation: Takefuji, Y. Set Operations in Python for Translational Medicine. *Int. J. Transl. Med.* **2022**, *2*, 174–185. <https://doi.org/10.3390/ijtm2020015>

Academic Editor: Simone Brogi

Received: 17 March 2022

Accepted: 27 April 2022

Published: 29 April 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the author. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

First, scientists in translational medicine must understand how to use Google search engine. You may be surprised that depending on browsers, the searched result may be different. There are two types in keyword searches: word keyword search and phrase keyword search. In a phrase keyword search, quotation marks indicate the ordered set of words. For example, “set operations” is composed of two words, i.e., set and operations, where set must be the first word and operations should be the second word after set.

An exhaustive search for articles containing the two phrases “vaccine safety” and “set operations” revealed only three articles over the Internet [1–3]. Jacques et al. showed how to use set operations for breast cancer analysis where the dataset is only composed of 285 instances [1]. Lu et al. did not show set operations at all for their analysis where the phrase of “set operations” was included in their references [2]. Barry DeVille et al. published a SAS book that briefly introduced set operations using VAERS datasets with a Statistical Analysis System (SAS) [3]. However, there was no detailed explanation on set operations by just showing graphic results with SAS. Since SAS needs a proprietary license, it is not open-source programming. To the best of our knowledge, there is no tutorial on set operations with open-source programming for vaccine safety. This paper's role with open-source programming in Python will be critical for translational medicine to deal with large datasets.

The author has published a tutorial paper on the PyPI packaging for translational medicine [4]. However, the significant contribution of this paper lies in that the previous tutorial did not include software reproducibility and set operations for efficient computing with large datasets. This paper details the calculations on set operations used in translational medicine. Set operations are used for calculating adverse effects on deaths due to COVID-19 using VAERS datasets [5].

There are many articles on the efficacy of vaccines, but few articles on adverse effects with vaccines. Writing this tutorial on set operations with open-source Python for translational medicine is motivated by four reasons: (1) we need to show that efficient computation, such as set operations in Python, is crucial for manipulating large datasets such as VAERS with 748,230 instances; (2) the computational complexity should be understood for accelerating computation; (3) there is no tutorial analysis on the extensive adverse effects of COVID-19 vaccines; and (4) PyPI packaging and software reproducibility are essential for scientists in translational medicine for maximum software dissemination to the world.

This paper presents a data analysis with set operations. The computational time complexity is depending on the structure of nested loops and the size of individual loops in algorithms or programs. For example, if your program has a single loop, the size of the loop determines the computational time complexity. In Python, the computational time complexity for a single-for-loop is determined by the number of instances (n), which is called Big O Notation $O(n)$:

```
for i in range(len(instances)):
```

In double-nested loops or triple-nested loops, the time complexity can be expressed with $O(n^2)$ and $O(n^3)$, respectively. With set operations, the double-nested loops, the triple-nested loops, and other loops can be converted to $O(n)$. Therefore, this paper introduces set operations to significantly reduce the time complexity.

For example, when calculating the number of deaths with mixing Pfizer and Moderna vaccine, with $O(n)$ time complexity, the number of deaths can be generated with set intersection.

In datasets, the number of instances is equivalent to the number of patients. In other words, the unique patient IDs can be used and shared in set operations in multiple datasets. Patient IDs are unique and shared in three VAERS datasets.

The number of Pfizer-death-patients deathPFIZER set can be simply calculated by intersecting the deathIDs and PFIZERIDs sets. Similarly, the number of Moderna-death-patients can be computed by intersecting the deaths-set and Moderna-set. Therefore, patient deaths from mixing the Pfizer and Moderna vaccines can be calculated by intersecting the Pfizer-death-patients-set and Moderna-death-patients-set. However, we do not know if Pfizer is the first vaccine. In other words, there are Pfizer-Moderna-death-patients and Moderna-Pfizer-death-patients. The time complexity in the above calculations is with $O(n)$.

The maleIDs and femaleIDs sets can be similarly generated with $O(n)$ for gender class set operations. All features, such as types of vaccines, gender class (male or female), death or alive (non-death), and ages, can be simply computed in this manner with set operations with $O(n)$. In other words, the computation time with set operations is drastically reduced from $O(n^3)$ or $O(n^2)$ to $O(n)$.

The advantage of PyPI is that it allows vaers to run on Windows, MacOS, and Linux operating systems, without being aware of operating systems as long as Python is installed on the system. This advantageous feature of PyPI is that it can maximize the open-source dissemination of software to the world.

This paper introduces Code Ocean for the reproducibility of software codes after showing the PyPI packaging. Code Ocean is the de facto service provider for software reproducibility.

In traditional software development, programmers must write a program from scratch. With the rapid progress of open-source software, programmers must choose the right libraries from depositories and glue them with minimum effort. The selected libraries and packages are available to the public and can be installed by a simple pip terminal-line command [6]. In other words, programmers must be familiar with the bash command in the terminal.

In this tutorial, we will follow the order of the execution of the commands in the bash shell based on reverse engineering. There is no significant difference between Windows, MacOS, and Linux operating systems.

This paper depicts a vaers executable package [7] as an example for calculating adverse effects on the number of deaths due to COVID-19 by gender and age group against the Moderna [8] and Pfizer [9] vaccines. The vaers method is currently under review.

First, programmers must understand how to scrape a dataset over the Internet. The executable vaers use the VAERS datasets. VAERS stands for Vaccine Adverse Event Reporting System. VAERS is a national early warning system to detect possible safety problems in US-licensed vaccines. VAERS is not designed to determine if a vaccine caused a health problem, but it is especially useful for detecting unusual or unexpected patterns of adverse event reporting that might indicate a possible safety problem with a vaccine.

Second, the dataset file must be read in Python. VAERS is composed of three csv files: 2021VAERSDATA.csv, 2021VAERSSYMPTOMS.csv, and 2021VAERSVAX.csv. In vaers.py, 2021VAERSDATA.csv and 2021VAERSVAX.csv are used. csv stands for comma-separated-value.

Third, a program is built to compute the target values using set operations. This paper shows how to calculate adverse events of death by sex and age group for each of the Novartis [10], Moderna, and Pfizer vaccines.

Fourth, the Python program is converted to the PyPI package with three files: setup.py, vaers.py, and README.md. The README.md file can be created using the GitHub site. Therefore, you need to create a new account on the GitHub site.

Finally, the PyPI package is uploaded using a twine command. In order to upload a PyPI package, you need to have an account on the pypi.org site.

In order to use and run a Python program, you must choose a proper installation package, miniconda, depending on your operating system from the following site:

<https://docs.conda.io/en/latest/miniconda.html> (accessed on 16 March 2022)

For Windows, double-click on the file, Miniconda3-py38_4.11.0-Windows-x86_64.exe. Python3.8 is recommended in this paper. For MacOS, the file, Miniconda3-py38_4.11.0-MacOSX-x86_64.sh, should be installed by the following terminal command: zsh or bash [11,12]:

```
$ zsh Miniconda3-py38_4.11.0-MacOSX-x86_64.sh
```

or

```
$ bash Miniconda3-py38_4.11.0-MacOSX-x86_64.sh
```

For Linux, download Miniconda3-py38_4.11.0-Linux-x86_64.sh and run the following command:

```
$ bash Miniconda3-py38_4.11.0-Linux-x86_64.sh
```

For Windows users, you have two options of Miniconda: one on Windows 11 or 10 and the other on Windows Subsystem for Linux (WSL). WSL is a compatibility layer for running Linux binary executables (in ELF format) natively on Windows 11 or 10. WSL has not been completed yet, but you are allowed to use binary executables on Windows from the WSL command line.

From here onwards, there is no difference between all operating systems. You should be familiar with conda and pip command with options:

First, start a terminal command and update the Miniconda environment by the following command. The first (\$) is a prompt from the terminal, while the second (\$) is the dollar key.

```
$ conda update conda
```

Second, update the pip installation command. “-U” stands for update.

```
$ pip install -U pip
```

or

```
$ python -m pip install -U pip
```

In order to install pandas, for example, run the following command.

```
$ pip install -U pandas
```

or

```
$ conda install pandas
```

In order to know the Python version number,

```
$ python -V
```

```
Python 3.8.4
```

the “which” command can inform the location of Python.

```
$ which python
```

```
/home/takefuji/miniconda3/bin/python
```

If the library is not Python-related, install it by the apt command on WSL or brew on MacOS.

First, apt should be updated and upgraded on Linux or WSL on Windows.

```
$ sudo apt update
```

```
$ sudo apt upgrade
```

Then, you can install the necessary library. For example, “wget” is a library name. “sudo” is a superuser command.

```
$ sudo apt install wget
```

For MacOS users, you must install the brew command, then run the following command to install matplotlib library. matplotlib is a library name.

```
$ brew install matplotlib
```

In vaers, the wget command is needed.

In WSL and MacOS, you must install the X-Window. For Windows users, you should download VcXsrv Windows X Server exe file and install it. For Mac users, you should install XQuartz. Before running Python, you should start the X Server.

vaers was selected for this tutorial because there is no tutorial on Python set operations. Set operations are useful to calculate the adverse effects on death by gender (male and female), age group, and vaccine group (Moderna, Pfizer, and Novartis).

In traditional programming, the programmer must program the target software from scratch. In open-source programming, the right libraries must be chosen from depositories and the selected libraries are simply glued together with minimum effort. This is called rapid open-source prototyping. vaers.py was developed within a few hours.

In other words, the skills in open-source programming lie in selecting the right libraries from a variety of the existing libraries [13]. The more examples that are available in open-source libraries, the easier it is for users to create the desired code.

This tutorial was written based on our experience with 19 PyPI projects:

<https://pypi.org/user/takefuji/> (accessed on 16 March 2022)

2. Materials and Methods

This Section includes testing the Python environment, the PyPI package of three files (setup.py, README.md, vaers.py), how to upload a PyPI package, and how to run it.

2.1. Python Environment and How to Run Vaers

It is assumed that Python is ready to run on the terminal. We must make sure that the system has a pip command in the PATH variable by the following command. PATH is an environmental variable in Windows, WSL on Windows, MacOS, and Linux operating systems that tells the shell which directories to search for executable files.

If you would like to show the PATH, run the following command:

```
$ echo $PATH
```

```
$ which pip
```

Type the following command to install vaers on WSL on Windows, MacOS, or Linux operating systems.

```
$ pip install vaers
```

You may have several errors from the installation. Remember that pip command is not a fully automated command so that you may need to install the following library: pandas before installing vaers. Error messages can inform you what libraries are missing in the current Python environment.

vaers uses VAERS datasets. You can download VAERS datasets from the following site with Word Verification:

<https://vaers.hhs.gov/eSubDownload/index.jsp?fn=2021VAERSData.zip> (accessed on 16 March 2022)

2021VAERSData.zip (172.10 MB) is composed of three csv files. Unzip 2021VAERSData.zip file:

```
$ unzip 2021VAERSData.zip
```

Three csv files will be generated: 2021VAERSDATA.csv (636.80 MB), 2021VAERSSYMPTOMS.csv (77.18 MB), and 2021VAERSVAX.csv (56.95 MB).

If csv files are not available, the following comments will be printed in the terminal.

You need to download 2021VAERSData.zip from the following site:

<https://vaers.hhs.gov/eSubDownload/index.jsp?fn=2021VAERSData.zip> (accessed on 16 March 2022)

Additionally, unzip 2021VAERSData.zip, 2021VAERSDATA.csv, and 2021VAERSVAX.csv are needed.

2.2. PyPI Package

A PyPI package needs three files including README.md, setup.py, and vaers.py.

2.2.1. README.md

The README.md file can be easily prepared by using the GitHub site. You need to have an account on the GitHub site. When creating a new Repository, select “add a README file”. README.md will be created when you enter the necessary content of a new PyPI package. Remember that the image in GitHub should be linked to the global site image address, instead of the local address. Unless the image is linked to the global address link, the image will not be displayed on the PyPI site.

2.2.2. setup.py

The following is a template of setup.py file for creating an executable code. The shaded 10 lines should be changed for your PyPI package.

```
import setuptools
with open("README.md", "r", encoding="utf-8") as fh:
    long_description = fh.read()
setuptools.setup(
    name="vaers",
    version="0.0.3",
    author="yoshiyasu takefuji",
    author_email="takefuji@keio.jp",
    description="A package for adverse effects on death using VAERS",
    long_description=long_description,
    long_description_content_type="text/markdown",
    url="https://github.com/ytakefuji/safety_vaccine", (accessed on 16 March 2022)
    project_urls={
        "Bug Tracker": "https://github.com/ytakefuji/safety_vaccine",
    },
    classifiers=[
        "Programming Language :: Python :: 3",
        "License :: OSI Approved :: MIT License",
        "Operating System :: OS Independent",
    ],
    package_dir={"": "src"},
    py_modules=['vaers'],
    packages=setuptools.find_packages(where="src"),
    python_requires=">=3.8",
    entry_points = {
'console_scripts': [
```

```

'vaers = vaers:main'
]
},
)

```

2.2.3. vaers.py

vaers.py is used to calculate adverse effects or deaths due to COVID-19 associated with vaccines.

The vaers tar file is available at the following site:

<https://files.pythonhosted.org/packages/76/63/e4cf38acc1204600042bba3974837ef67891f76398412504dfb620839b6b/vaers-0.0.3.tar.gz> (accessed on 16 March 2022)

Expand the vaers tar file by the following command:

```

$ wget <above link>
$ tar xvf vaers-0.0.3.tar.gz
$ cd vaers-0.0.3/src
$ cat vaers.py

```

vaers.py is as follows, where shaded lines are a program indicating the location of the VAERS datasets for possible manual downloading by users.

```

import pandas as pd
import sys,os
if len(sys.argv)==2:
if os.path.exists(sys.argv [1]+'VAERSDATA.csv') and os.path.exists(sys.argv [1]+'VAERSVAX.csv'):
print(sys.argv [1]+" data will be used")
d=pd.read_csv(sys.argv [1]+'VAERSDATA.csv',low_memory=False,encoding='cp1252')
vax=pd.read_csv(sys.argv [1]+'VAERSVAX.csv',low_memory=False,encoding='cp1252')
else:
print("You need to download 2022VAERSData.zip from the following site:")
print("https://vaers.hhs.gov/eSubDownload/index.jsp?fn="+sys.argv [1]+"VAERS-
Data.zip") (accessed on 16 March 2022)
print("And unzip "+sys.argv [1]+"VAERSData.zip")
print(sys.argv [1]+"VAERSDATA.csv and "+sys.argv [1]+"VAERSVAX.csv are needed")
os._exit(0)
else:
if len(sys.argv)==1:
if os.path.exists('2021VAERSDATA.csv') and os.path.exists('2021VAERSVAX.csv'):
print("2021 data will be used")
d=pd.read_csv('2021VAERSDATA.csv', low_memory=False,encoding='cp1252')
vax=pd.read_csv('2021VAERSVAX.csv', low_memory=False,encoding='cp1252')
else:
print("You need to download 2021VAERSData.zip from the following site:")
print("https://vaers.hhs.gov/eSubDownload/index.jsp?fn=2021VAERSData.zip")
print("And unzip 2021VAERSData.zip")
print("2021VAERSDATA.csv and 2021VAERSVAX.csv are needed")
os._exit(0)

def main():
d['DIED'].fillna("N",inplace=True)
deathIDs=d.loc[d.DIED=='Y','VAERS_ID']
print("total instances: ",len(d['DIED']))
print("total deaths",len(deathIDs))

maleIDs=d.loc[d.SEX=="M",'VAERS_ID']
femaleIDs=d.loc[d.SEX=="F",'VAERS_ID']

```



```

NOVIDs=vax.loc[vax.VAX_MANU=="NOVARTIS VACCINES AND DIAGNOSTICS",
'VAERS_ID']
print("NOVIDs instances: ",len(NOVIDs))
deathNOV=set(deathIDs).intersection(NOVIDs)
print("NOVIDs deaths: ",len(deathNOV))
print("NOV death per instance",round(len(deathNOV)/len(NOVIDs),6))

MODERNAIDs=vax.loc[vax.VAX_MANU=="MODERNA",'VAERS_ID']
deathMODERNA=set(deathIDs).intersection(MODERNAIDs)

PFIZERIDs=vax.loc[vax.VAX_MANU=="PFIZER\BIONTECH",'VAERS_ID']
deathPFIZER=set(deathIDs).intersection(PFIZERIDs)

M_P=set(MODERNAIDs).intersection(PFIZERIDs)
M_Pdeath=set(deathMODERNA).intersection(deathPFIZER)
print('MODERNA+PFIZER:',len(M_P))
print('MODERNA+PFIZER death:',len(M_Pdeath))
print('MODERNA+PFIZER death per instance:',round(len(M_Pdeath)/len(M_P),6))

MODERNAIDs=set(MODERNAIDs).difference(M_P)
print("MODERNAIDs instances: ",len(MODERNAIDs))
deathMODERNA=set(deathIDs).intersection(MODERNAIDs)
print("MODERNA deaths",len(deathMODERNA))
print("MODERNA",round(len(deathMODERNA)/len(MODERNAIDs),6))
deathMODERNAmaleIDs=set(deathMODERNA).intersection(maleIDs)
print("deathMODERNAmaleIDs",len(deathMODERNAmaleIDs))
deathMODERNAfemaleIDs=set(deathMODERNA).intersection(femaleIDs)
print("deathMODERNAfemaleIDs",len(deathMODERNAfemaleIDs))
MODERNAfemaleIDs=set(MODERNAIDs).intersection(femaleIDs)
print("MODERNAfemaleIDs:",len(MODERNAfemaleIDs))
MODERNAmaleIDs=set(MODERNAIDs).intersection(maleIDs)
print("MODERNAmaleIDs:",len(MODERNAmaleIDs))
print("MODERNA female death",round(len(deathMODERNAfemaleIDs)/
len(MODERNAfemaleIDs),6))
print("MODERNA male death",round(len(deathMODERNAmaleIDs)/
len(MODERNAmaleIDs),6))

PFIZERIDs=set(PFIZERIDs).difference(M_P)
print("PFIZERIDs instances: ",len(PFIZERIDs))
deathPFIZER=set(deathIDs).intersection(PFIZERIDs)
print("PFIZER deaths",len(deathPFIZER))
print("PFIZER",round(len(deathPFIZER)/len(PFIZERIDs),6))
PFIZERfemaleIDs=set(PFIZERIDs).intersection(femaleIDs)
print("PFIZERfemaleIDs:",len(PFIZERfemaleIDs))
PFIZERmaleIDs=set(PFIZERIDs).intersection(maleIDs)
print("PFIZERmaleIDs:",len(MODERNAmaleIDs))
deathPFIZERmaleIDs=set(deathPFIZER).intersection(maleIDs)
deathPFIZERfemaleIDs=set(deathPFIZER).intersection(femaleIDs)
print("PFIZER female death",round(len(deathPFIZERfemaleIDs)/
len(PFIZERfemaleIDs),6))
print("PFIZER male death",round(len(deathPFIZERmaleIDs)/
len(PFIZERmaleIDs),6))

main()

```

The following Python program calculates deathIDs set mentioned in the Introduction Section.

```
deathIDs=d.loc[d.DIED=='Y','VAERS_ID']
```

Similarly, maleIDs and femaleIDs can be calculated by the following program, respectively.

```
maleIDs=d.loc[d.SEX=="M",'VAERS_ID']
```

```
femaleIDs=d.loc[d.SEX=="F",'VAERS_ID']
```

deathPFIZER can be computed by intersecting PFIZERIDs and deathIDs as follows.

```
PFIZERIDs=vax.loc[vax.VAX_MANU=="PFIZER\BIONTECH",'VAERS_ID']
```

```
deathPFIZER=set(deathIDs).intersection(PFIZERIDs)
```

2.3. How to Upload a PyPI Package

You need to generate three files (.whl file, .egg file, and .tar.gz file). Type the following commands:

```
$ python setup.py install
```

```
$ python setup.py sdist bdist_wheel
```

In order to upload three files (covidlag-0.0.7-py3-none-any.whl, covidlag-0.0.7-py3.8.egg, and covidlag-0.0.7.tar.gz), you need to install twine:

```
$ pip install twine
```

Before uploading three files to the PyPI site, you need to register at the following site:

<https://pypi.org/> (accessed on 16 March 2022)

The directory and files are as follows:

```
.
├── README.md
├── build
├── dist
├── setup.py
├── src
└── vaers.py
```

The following command can upload three files. The system will ask for a username and password.

```
$ twine upload dist/*
```

When you want to update the package, you must delete all files and directories in dist/* and build/* by the following command.

If you want to update the application, remove the old files:

```
$ rm -rf dist/* build/*
```

Then, repeat the same commands.

3. Discussion on Set Operations

This tutorial allows researchers to submit a new PyPI package and to showcase their skills on PyPI packages around the world. All that is required is to create three files, including vaers.py, setup.py, and README.md, by following instructions in the Materials and Methods Section. Before submitting the new package, you should test it on your local machine.

There are four set operations as shown in Figure 1: union, intersection, exclusive OR, and subtraction. In Python, the union set operation of set A and set B can be calculated by the following:

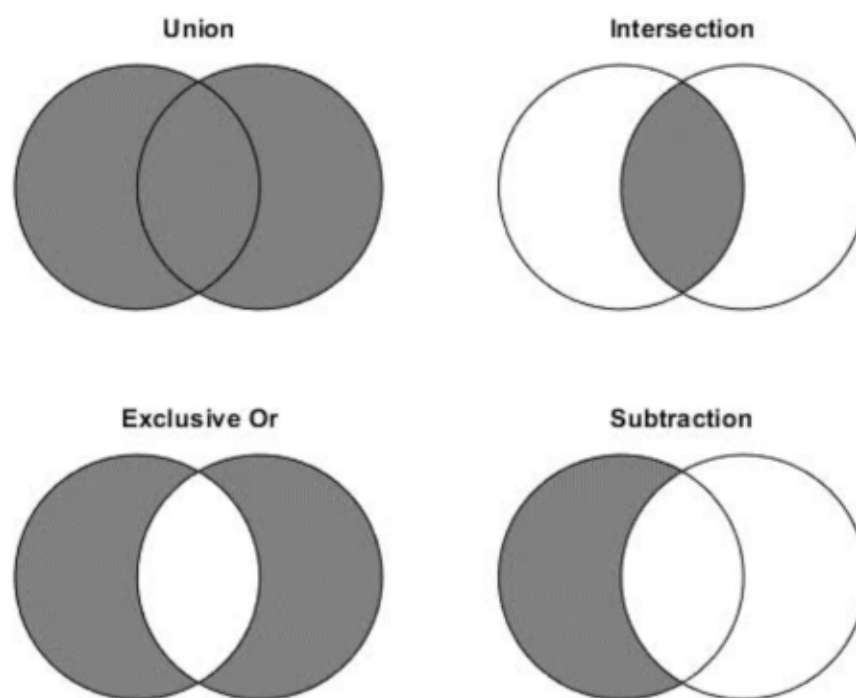


Figure 1. Four set operations.

```
set(A).union(B)
```

Similarly, the set intersection between A and B can be operated by:

```
set(A).intersection(B)
```

ExclusiveOR operation of A and B is calculated by:

```
set(A).symmetric_difference(B)
```

Subtraction operation of A and B is calculated by:

```
set(A).difference(B)
```

In the `vaers.py`, set intersection operations are used.

In `vaers.py`, the shaded lines from the first line before `def main()` are used for checking the existence of two files and, if two files exist, then they are read by `pd.read_csv` of pandas library.

```
d=pd.read_csv(sys.argv [1]+'VAERSDATA.csv',low_memory=False,encoding='cp1252')
```

```
vax=pd.read_csv(sys.argv [1]+'VAERSVAX.csv',low_memory=False,encoding='cp1252')
```

Three csv files use the common ID numbers so that `deathIDs` is a set of death IDs in the dataset. The following two lines calculate the number of total instances and the number of deaths. `d` is pandas data read from the '`VAERSDATA.csv`' file, while `vax` is pandas data read from '`VAERSVAX.csv`'.

```
d['DIED'].fillna("N",inplace=True)
```

```
deathIDs=d.loc[d.DIED=='Y','VAERS_ID']
```

There are two types in the `DIED` determinant: Y or N. Therefore, the number of the total instances is calculated by `len(d['DIED'])`, where `len` is length or size function in Python. `deathIDs` indicates the number of deaths where `d.DIED=='Y'`.

The pandas `.loc` function is convenient for enforcing the equal condition (`==`) in the dataset.

The gender of `SEX` determinant plays a key role in set operating.

```
maleIDs=d.loc[d.SEX=="M",'VAERS_ID']
```

```
femaleIDs=d.loc[d.SEX=="F",'VAERS_ID']
```

`maleIDs` indicates male IDs while `femaleIDs` indicate female IDs.

Novartis IDs can be calculated by:

```
NOVIDs=vax.loc[vax.VAX_MANU=="NOVARTIS VACCINES AND DIAGNOSTICS",  
'VAERS_ID']
```

where VAX_MANU determinant enforces “NOVARTIS VACCINES AND DIAGNOSTICS”.

The following three lines show the calculation of the intersection of two sets: Moderna and Pfizer IDs.

M_P indicates the intersection operation of two sets using the Moderna and Pfizer IDs.

```
MODERNAIDs=vax.loc[vax.VAX_MANU=="MODERNA",'VAERS_ID']
```

```
PFIZERIDs=vax.loc[vax.VAX_MANU=="PFIZER\BIONTECH",'VAERS_ID']
```

```
M_P=set(MODERNAIDs).intersection(PFIZERIDs)
```

M_Pdeath indicates the intersection of two sets: deathMODERNA and deathPFIZER.

```
M_Pdeath=set(deathMODERNA).intersection(deathPFIZER)
```

The following set operation indicates the intersection of two sets: deathPFIZER and femaleIDs. In other words, len(deathPFIZERfemaleIDs) indicates the number of female deaths due to the Pfizer vaccine.

```
deathPFIZERfemaleIDs=set(deathPFIZER).intersection(femaleIDs)
```

Set operations are not only useful for calculating the target sets for translational medicine, but also for efficient computing with converting $O(n^2)$ or $O(n^3)$ to $O(n)$ time complexity.

In order to run vaers, type the following command in the terminal. vaers will automatically start to calculate with set operations (Box 1).

Box 1. The result of vaers execution.

```
$ vaers
total instances: 748230
total deaths 10125
NOVIDs instances: 1475
NOVIDs deaths: 2
NOV death per instance 0.001356
MODERNA+PFIZER: 996
MODERNA+PFIZER death: 5
MODERNA+PFIZER death per instance: 0.00502
MODERNAIDs instances: 325993
MODERNA deaths 4071
MODERNA 0.012488
deathMODERNAfemaleIDs 2330
deathMODERNAfemaleIDs 1657
MODERNAfemaleIDs: 224687
MODERNAfemaleIDs: 87945
MODERNA female death 0.007375
MODERNA male death 0.026494
PFIZERIDs instances: 313773
PFIZER deaths 4488
PFIZER 0.014303
PFIZERfemaleIDs: 210111
PFIZERmaleIDs: 87945
PFIZER female death 0.009043
PFIZER male death 0.024402
```

4. Software Reproducibility via Code Ocean

Nowadays, refereed journals may ask authors to validate the reproducibility of the proposed software as a requirement. Code Ocean is one of the publishing reproducibility badges. You must create a new account on Code Ocean:

<https://codeocean.com/> (accessed on 16 March 2022)

The following steps are needed:

1. Click New Capsule.
2. Create Blank Capsule.
3. In Environment, click the necessary buttons. pick Python3.8.5 and pip3. Then, install pandas and vaers using the add button.

4. Click Start with Sample Files.
5. Fill the form of metadata.
6. Modify run.

This is an example run of vaers.

```
#!/usr/bin/env bash
set -ex
vaers 2022 >../results/result.txt
```

1. Click “Submit for publication” and check all items.
2. After receiving acceptance from Code Ocean, you can place the badge of the reproducibility qualification in the following form in any markdown document in the GitHub site:

`[![Open in Code Ocean](https://codeocean.com/codeocean-assets/badge/open-in-code-ocean.svg)](https://codeocean.com/capsule/xxxx/tree)` (accessed on 16 March 2022)

5. Conclusions

This is the world’s first tutorial on Python set operations for translational medicine or medicine in general. This tutorial allows researchers to build a new PyPI package and to showcase their skills on PyPI packages around the world. `vaers.py`, `setup.py`, and `README.md` were described in detail in this paper. Code Ocean was introduced for reproducibility requirements.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Jacques, G.M.; Barlow, J.; Rommel, R.; Kaufmann, A.; Rienti, M., Jr.; AvRuskin, G.; Rasul, J. Residential Mobility and Breast Cancer in Marin County, California, USA. *Int. J. Environ. Res. Public Health* **2014**, *11*, 271–295. [[CrossRef](#)] [[PubMed](#)]
2. Lu, J.; He, T.; Wei, G.; Wu, J.; Wei, C. Cumulative Prospect Theory: Performance Evaluation of Government Purchases of Home-Based Elderly-Care Services Using the Pythagorean 2-tuple Linguistic TODIM Method. *Int. J. Environ. Res. Public Health* **2020**, *17*, 1939. [[CrossRef](#)] [[PubMed](#)]
3. DeVillie, B.; Bawa, G.S. *Text as Data: Computational Methods of Understanding Written Expression Using SAS*; Wiley: Hoboken, NJ, USA, 2021; ISBN 978-1-119-48712-8.
4. Takefuji, Y. Python Programming in PyPI for Translational Medicine. *Int. J. Transl. Med.* **2021**, *1*, 323–331. [[CrossRef](#)]
5. Available online: <https://vaers.hhs.gov/data/datasets.html> (accessed on 16 March 2022).
6. Manzini, S.; Busnelli, M.; Colombo, A.; Franchi, E.; Grossano, P.; Chiesa, G. reString: An open-source Python software to perform automatic functional enrichment retrieval, results aggregation and data visualization. *Sci. Rep.* **2021**, *11*, 23458. [[CrossRef](#)] [[PubMed](#)]
7. Available online: <https://pypi.org/project/vaers/> (accessed on 16 March 2022).
8. Callaway, E. Pfizer COVID Vaccine Protects against Worrying Coronavirus Variants. *Nature* **2021**, *593*, 325–326. [[CrossRef](#)] [[PubMed](#)]
9. Gaviria, M.; Kilic, B. A network analysis of COVID-19 mRNA vaccine patents. *Nat. Biotechnol.* **2021**, *39*, 546–548. [[CrossRef](#)] [[PubMed](#)]
10. Irwin, A.; Nkengasong, J. What It Will Take to Vaccinate the World against COVID-19. *Nature* **2021**, *592*, 176–178. [[CrossRef](#)] [[PubMed](#)]
11. Perkel, J.M. Five Reasons Why Researchers Should Learn to Love the Command Line. *Nature* **2021**, *590*, 173–174. [[CrossRef](#)] [[PubMed](#)]

12. Reimann, H.; Hentschel, J.; Marek, J.; Huelnhagen, T.; Todiras, M.; Kox, S.; Waiczies, S.; Hodge, R.; Bader, M.; Pohlmann, A.; et al. Normothermic Mouse Functional MRI of Acute Focal Thermostimulation for Probing Nociception. *Sci. Rep.* **2016**, *6*, 17230. [[CrossRef](#)] [[PubMed](#)]
13. Pintacuda, G.; Lassen, F.H.; Hsu, Y.H.H.; Kim, A.; Martín, J.M.; Malolepsza, E.; Lim, J.K.; Fornelos, N.; Eggan, K.C.; Lage, K. Genoppi is an open-source software for robust and standardized integration of proteomic and genetic data. *Nat. Commun.* **2021**, *12*, 2580. [[CrossRef](#)] [[PubMed](#)]