VAERity 1.0RC3

User Manual

(Work in Progress)

Contents

What is VAERity?

VAERity is a free, open source tool to graphically explore the VAERS data set.

VAERS stands for Vaccine Adverse Event Reporting System. It is a database run by the US CDC via HHS (Health and Human Services) to record adverse reactions to vaccines.

The original rationale behind VAERity was to make the data available in a more accessible format, one that allows data insights to be extracted in a way that can drive decision making processes or contribute to a comprehensive risk assessment.

In order to do this, the limitations of the data set need to be understood. Actions to reach this understanding include awareness of the standard operating procedures of those maintaining the data, as well as knowing when parts of the data are missing, and the implications of this data gap on the graphical representation.

License and scope

VAERity is free and open source software, written in **Python**. The release candidate for 1.0RC1 has taken just over two months to reach a stable version which is portable and feature rich enough to cater to most types of query which would be helpful to investigators in bypassing the limitations of the CDC WONDER system as well as investigating data inconsistencies, of which there are many.

The software has been developed using my own funds, initially as a means of demonstrating the author’s competence in big data, Python and all things technical.

Although the program is reasonably robust, the scope of unit testing has been limited due to the RAD development methodology and practical considerations regarding the urgency of having such a tool available.

Thus, the program has been released under a **Creative Commons 3.0 non commercial share-alike license**, and the author disclaims, to the fullest extent possible by law, any and all liability arising from the use of this program.

The usual warnings also apply to the data extracted through use of this program. Beware of attributing causation where only correlation is shown, be mindful of the scope of the data quality as described in the Data Insights graph (see later in the manual), as well as the source of the reports in the database. Seek external data to verify any hypotheses arrived at through perusing the data insights, and consider the larger picture before making any risk assessment.

What is VAERity?

The data insights in this program are meant to provide visibility over publicly accessible data in a format that can drive discussion, but are not intended as medical advice. Speak to your doctor, GP or health care professional if you are concerned by any of the data you find in VAERS, and ultimately, be aware of your own responsibility in any actions you take as a result.

The goal of this program is, as stated, to “Uncover truth in data”, to allow data to speak for itself, not to put forward a political or cultural view. Indeed, as you begin your journey through the data in VAERS, you may find the scope of the data uncovers aspects of the pandemic that have not been treated objectively by either the mainstream or alternative media.

It is my sincere hope that this program can enable an objective conversation between vaccinated and unvaccinated individuals, encourage free and independent thought, and enable investigation that would otherwise have been impossible due to the limitations of the tool set.

Finally, if you appreciate this program or want it to continue to be developed and supported into the future, please consider donating. Links to donate will be provided at the end of the manual.

Installation

How you install the program will depend on how you received it.

If you downloaded this program from sourceforge, the ZIP file is a portable version of the program. Ideally, you will want write access to at least the resources folder, if you want to upload your own start up images. You may also want write access to the styles folder if you want to save your custom styles to a common folder for other users to access.

You can extract the folder to any writeable location and create a shortcut on your desktop to vaerity.exe

Although the program will prompt you for the location of your Vaers data files, if you have difficulty navigating to and selecting the database folder, you can also specify the location yourself. Briefly:

1) Right click the rootwindow.ini file in the extracted folder and select Copy. Alternatively, you can use Control-C to copy the file.

2) Select your Search bar in Windows 10 or 11 and type in %APPDATA% and click on the folder that appears in the search results.

3) Paste the file into this folder and rename it to vaerity.ini

4) Right click the file and open it in your favourite text editor. At the end of the file, change the line beginning with vaersfolder to reference the folder you've extracted the Vaers data files to.

Downloading the VAERS data files

The VAERS data files can be downloaded from https://vaers.hhs.gov/data/datasets.html

The "All Years Data" may or may not include the Non Domestic data (foreign reports). If this concerns you, download the Non Domestic as well. All files should be extracted to the same folder. Make sure that your VAERS folder is writeable by the user the program runs as, as VAERity will attempt to convert the CSV files into HDF5 (typically once only).

If you later want to update the data files, please see the section on maintenance.

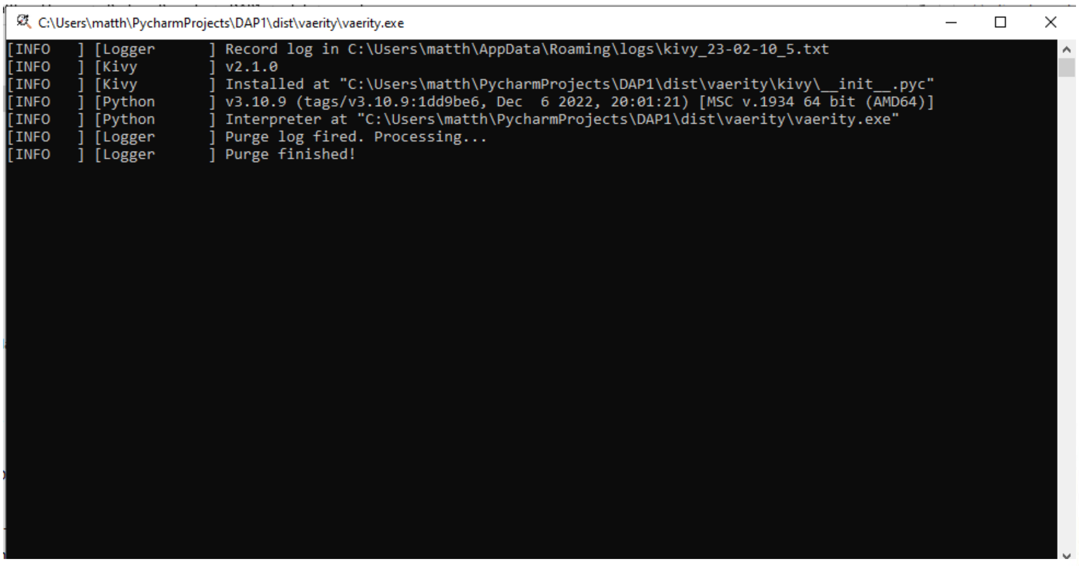
Installation

(This page is left blank for future insertion of the USB drive install procedure)

Running the program

The program can be started by double clicking its icon:

The first thing to pop up will be a console window.



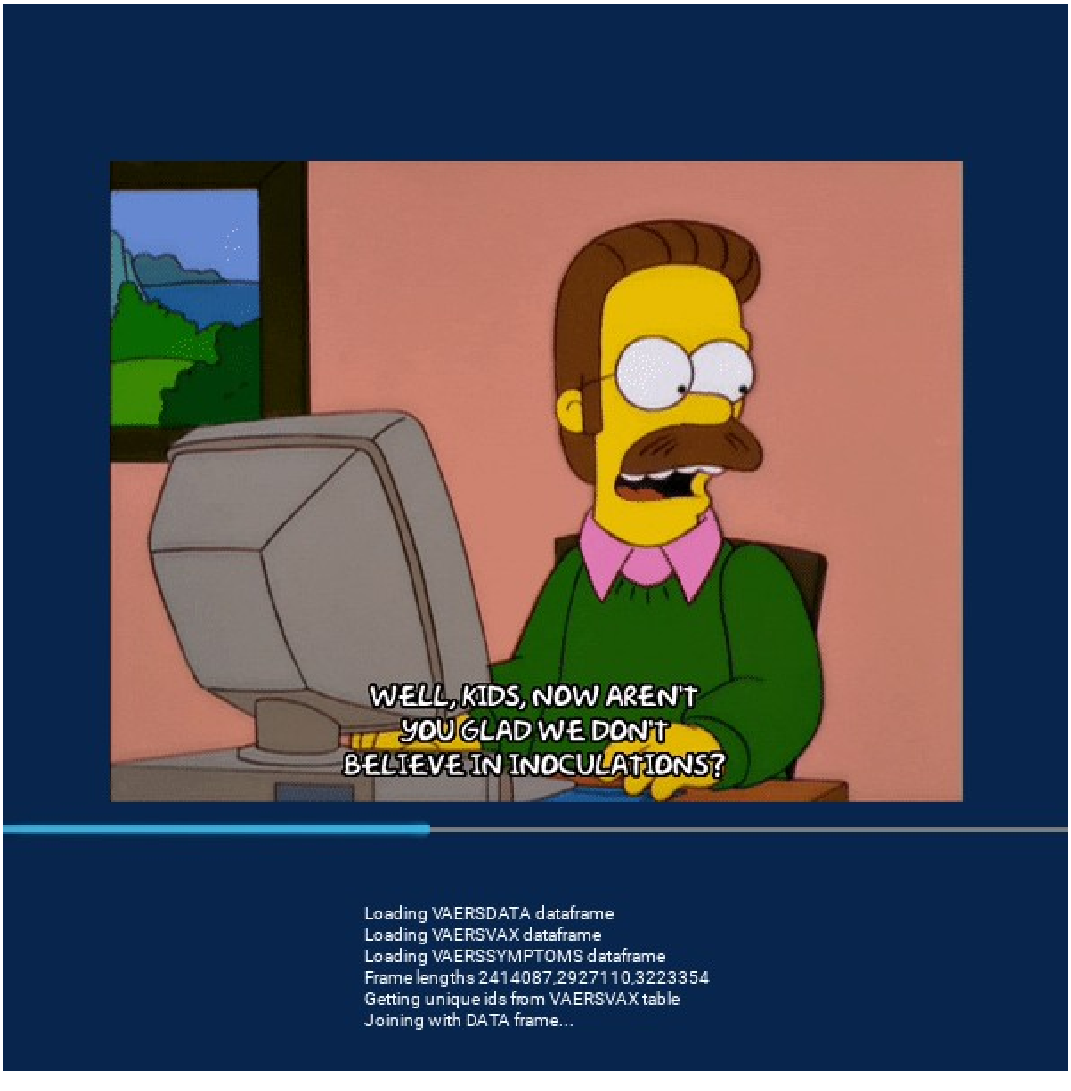
```
[INFO   ] [Logger      ] Record log in C:\Users\matth\AppData\Roaming\logs\kivy_23-02-10_5.txt
[INFO   ] [Kivy        ] v2.1.0
[INFO   ] [Kivy        ] Installed at "C:\Users\matth\PycharmProjects\DAP1\dist\vaerity\kivy\__init__.pyc"
[INFO   ] [Python      ] v3.10.9 (tags/v3.10.9:1dd9be6, Dec  6 2022, 20:01:21) [MSC v.1934 64 bit (AMD64)]
[INFO   ] [Python      ] Interpreter at "C:\Users\matth\PycharmProjects\DAP1\dist\vaerity\vaerity.exe"
[INFO   ] [Logger      ] Purge log fired. Processing...
[INFO   ] [Logger      ] Purge finished!
```

It may take several seconds for the start screen to also appear. The console window is mainly useful during the startup phase, as it will display informational messages, including messages indicating the progress of conversion of the VAERS data files to their proper format for querying.

The window manager/GUI prints a lot of information messages to the screen during operation of the program. These can be ignored unless you happen to be troubleshooting problems with the operation of the program.

Running the program

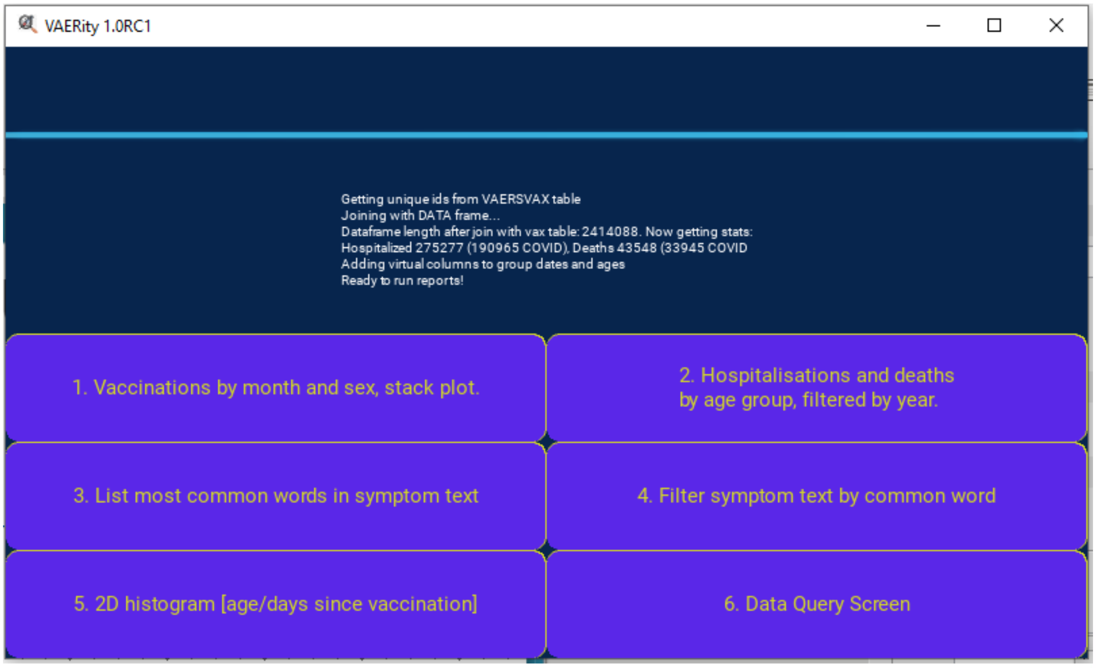
The startup screen prints information messages about the progress of loading the data set:


WELL, KIDS, NOW AREN'T YOU GLAD WE DON'T BELIEVE IN INOCULATIONS?

```
Loading VAERSDATA dataframe
Loading VAERSVAX dataframe
Loading VAERSSYMPTOMS dataframe
Frame lengths 2414087,2927110,3223354
Getting unique ids from VAERSVAX table
Joining with DATA frame...
```

Running the program

After the data set is loaded, this borderless window will disappear and turn into this:


VAERity 1.0RC1
1. Vaccinations by month and sex, stack plot.
2. Hospitalisations and deaths by age group, filtered by year.
3. List most common words in symptom text
4. Filter symptom text by common word
5. 2D histogram [age/days since vaccination]
6. Data Query Screen

```
Getting unique ids from VAERSVAX table
Joining with DATA frame...
Dataframe length after join with vax table: 2414088. Now getting stats:
Hospitalized 275277 (190965 COVID), Deaths 43548 (33945 COVID
Adding virtual columns to group dates and ages
Ready to run reports!
```

If, at any time during startup, the database thread crashes, the program may continue to wait indefinitely for the database to be loaded. If this happens, simply exit the program by selecting that startup window and pressing the Windows hotkey Alt+F4 to exit the program.

Most of your time using this program will be spent on the Data Query Screen. The other buttons are data set specific and will be described in the following pages.

Button Actions

1. Vaccinations by month and sex, stack plot.

Despite its name, this is a graph not of vaccinations but incident reports. The reports are grouped according to the date they were received, ie **RECVDATE**, as other date fields contain significant missing data. For instance, the report date, **RPT_DATE**, contains *1.7 million records* for which the date was not filled in, for a data set of just under 2.4 million records. This kind of reporting makes such fields useless for aggregation. The following list shows a typical output for the number of NA records for different date columns in the database:

```
TODAYS_DATE: 113558
VAX_DATE: 259062
ONSET_DATE: 321697
RPT_DATE: 1774032
```

For a dataset of around 2.4 million records, this still constitutes a significant amount of missing data.   This brief explanation hopefully gives the reader an inkling of some of the data quality issues (which are common in real world data sets and not restricted to VAERS)

This graph, and all graphs on the start screen, are **not COVID specific**, but relate to the **entire VAERS data set**. To view COVID specific data insights, use the **Data Query Screen** and select **Stats** on the data you are interested in.

You can also **View Raw Data** to look more closely at the statistics for a given period or gender. The data can then be exported via **Save CSV** for use in your favourite spreadsheet or data analysis software.

Button Actions

2. Hospitalisations and deaths by age group, filtered by year

This is one of the most basic graphs provided. It shows the percentage of deaths for different age brackets. The age brackets are defined in 20 year increments. The main insight this graph provides is how many people did not provide their age, or for whom an age was unable to be calculated. It also shows the prevalence of age in who has been most negatively impacted by vaccines in general.

This graph, and all graphs on the start screen, are **not COVID specific**, but relate to the **entire VAERS data set**. To view COVID specific data insights, use the **Data Query Screen** and select **Stats** on the data you are interested in.

You can also **View Raw Data** to look more closely at the statistics. The data can then be exported via **Save CSV** for use in your favourite spreadsheet or data analysis software.

Button Actions

3. List most common words in symptom text.
4. Filter symptom text by common word.

WARNING WARNING WARNING

Unless you are interested in the most common *words* (not symptoms) across the entire dataset, I **do not** recommend clicking on option 3. This, and to some extent #4, by common word, are of more interest on a purely academic level.

These options will **hammer** your **CPU** (the first for around two and a half minutes on average, the second for all of one minute) while it aggregates the symptoms across the entire database, the first time you use it. After this, the symptom frequency data is stored internally

These graphs, like all graphs on the front page, are again **not COVID specific**. Additionally, it is only for the data from the SYMPTOMS table, it does not include data from the symptom text field or other raw data fields within the DATA table.

These graphs were precursors to the more advanced capabilities available on the **Data Query Screen**.

Button Actions

5. 2D histogram (age/days since vaccination)

This gives you a snapshot of the **AGE_YRS** field vs the **NUMDAYS** field. It is useful primarily to demonstrate a data quality issue.

This graph, like all graphs on the front page, is **not COVID specific**.

The Z shape that you see here shows a number of entries in the data set whose age and days of onset have a linear relationship. These tend to be vaccines had at birth which later in life are assumed to have caused some problem.

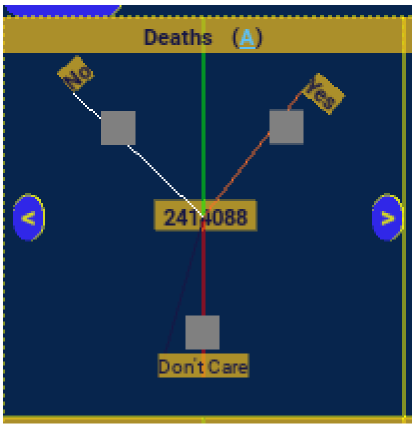
However, as you will see as you explore the data set, these appear in places where the correlation displayed is physically impossible. For example, days of onset 100 years for a COVID19 vaccine, or patients who have given their date of birth as the date of vaccination, or other data problems (such as the upper bar on the Z). This is included for the sake of completeness as it may be an interesting investigation for someone interested in the data purity of the VAERS data set.

Data Query Screen

6. Data Query Screen

This is the main screen you will be spending your time on, along with the subsequent two.

Most of the ten patient outcome fields come first, These are, briefly: Deaths, hospitalisations, emergency room visits, Disabled, Life threatening, Office/Clinic visit, and Birth Defect.

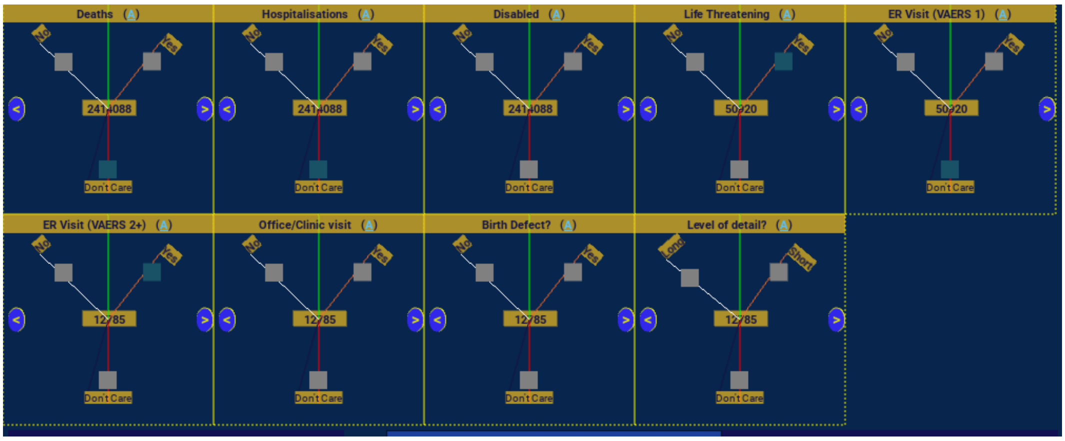


These patient outcome selectors are nicknamed “flux capacitors”, not only due to how they look, but because they let you get a glimpse of the result before you run your query, by seeing a count of the records as you widen or narrow by outcomes.

When you select one of the three values, the third is always a neutral pass through such as Don’t Care, and the upper two represent a binary choice. As such, internally in the software the top two are referred to as Mad Dog (right) and Density (left). The in between is Calvin Klein ;-)

Data Query Screen

To take it out of the realm of abstract thought into reality, here is an example filter as of 11/02/2003:



You can see if you zoom in that the number of records gradually decreases from left to right and top to bottom as choices are made. By the end, we have 12785 records that are both life threatening and necessitated an ER visit, that were recorded in the new VAERS 2 system.

We could switch with ER (VAERS 1) to see records of ER visits prior to the cutover, which would result in 6484 life threatening prior to VAERS 2 being introduced. You will note upon searching when VAERS 2 was brought in that there are a disproportionate number of ER visits recorded in VAERS 2 as compared to the prior ~30 years of ER data. Indeed, we find a spike in the “all cause” ER data, peaking in May of 2021, and gradually tapering off, unfortunately not to pre-pandemic levels.

The investigation of the reasons for this surge are left as an exercise for the reader if they are curious. The same increase is seen in Urticaria as a symptom, a type of hives, for example. The sudden spike occurs within one month. However, the incidents by vaccine type should clue you in that there is something interesting happening with regards to ER visits that goes beyond the typical guesses as to the rise in incident reports.

Data Query Screen

Having set a challenge, lets move on to the freeform text fields.

There are six of these, each of which corresponds to one of the free text fields documenting the adverse event. Each of them is fairly self explanatory by name: allergies, other medications (other than the vaccine), lab data, pre-existing conditions (patient history), existing (current illnesses), and the primary narrative or “symptom text”.

As this is a more resource intensive operation, you must check the box next to the text field to confirm that you want to search it. This then performs a case sensitive regular expression search with the query using pandas str_contains. If you want to ignore case, include (?i) at the start of the search term.

You cannot include a new line or apostrophe in the text (unless you escape it). If you want to search for multiple words or partial words, enclose them in brackets and separate them by the pipe character.

This functionality allows us to make some very interesting discoveries. For example, we can identify that for ten of the reports from the so called “batch of death”, EM0477, the patient was on fentanyl as one of their reported medications. Of the 601 patients with this medication reported, 40 died – about 6.66%. This is not statistically insignificant The question could then be asked, given the under-reporting of the other medications field, could this be a medical risk factor contributing to adverse interactions that is not being identified, both due to the typical limitations of data exploration and other reasons, such as the potential stigma this medication carries causing it to be under reported?

Data Query Screen

Without speculating further on this, let us move on to the ranged text fields and spinners.

First of all the spinners. You can choose a particular gender, vaccine type, manufacturer or dose number. You currently can't select multiple values. Sorry about that. If you select a value, the checkbox will automatically be selected next to the spinner. The reason for the checkbox is as described above for the raw text fields. String comparisons take time.

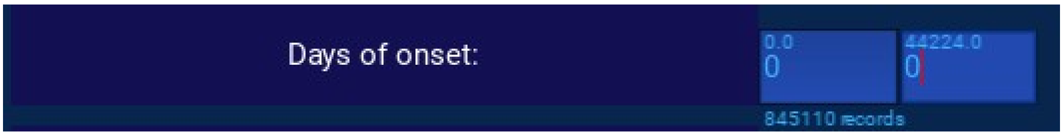
Ranged Text Fields

These share a characteristic in common with the flux capacitors: they support dynamic query. However, these ranged text fields were the first dynamic query fields to be implemented in the program, and as such they do not directly interface with the previous dynamic query fields. This will be fixed in a future version of the program.

The result set size that you see upon entering an upper or lower bound for the field therefore is the total records in the entire data set that match that range, not the total matching records with the outcomes chosen at the top of the form.

Data Query Screen

Each of Days of onset, days hospitalised, and the two age fields, have a range within which the values fall. These minimum and maximum values for the field in the data set are displayed at the top of the text field like so:


Days of onset:
0.0
0
44224.0
0
845110 records

Here we see that the maximum value for number of days between vaccination and onset of the adverse event is 44,224 days. This is because some vaccines are given at birth, and the adverse event occurs in the person’s later life.

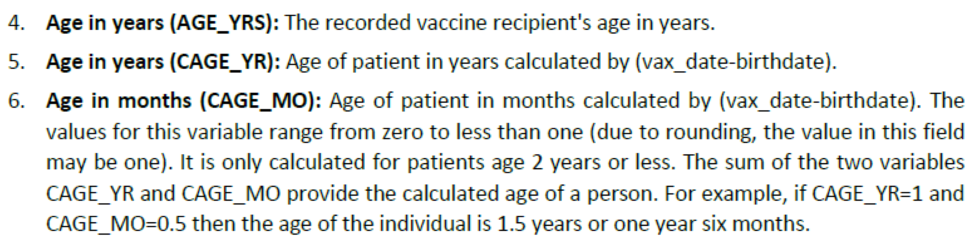
Thus the maximum days of onset is the age of the oldest person in the database: 120-121 years.

The above example also shows that there are 845,110 entries in the database whose onset was zero, ie immediate.

Data Query Screen

The Age Fields

The two age fields are outlined in the VAERS documentation like this:



4. **Age in years (AGE_YRS):** The recorded vaccine recipient's age in years.
5. **Age in years (CAGE_YR):** Age of patient in years calculated by (vax_date-birthdate).
6. **Age in months (CAGE_MO):** Age of patient in months calculated by (vax_date-birthdate). The values for this variable range from zero to less than one (due to rounding, the value in this field may be one). It is only calculated for patients age 2 years or less. The sum of the two variables CAGE_YR and CAGE_MO provide the calculated age of a person. For example, if CAGE_YR=1 and CAGE_MO=0.5 then the age of the individual is 1.5 years or one year six months.

However, in practice the fields do not always act in this way, and can be counter intuitive. The best suggestion I can give is to look at the Age distribution graph for patients with a high number of days of onset 10000 or more. You will note that the second age field, where specified, has about half the data set indicating age as an infant. However, owing to the number of different vaccines and ages at vaccination, as well as human error, the CAGE_YR for some of these 610 records (between30 and 35 records or about 5%) are unequally distributed between age 2 and 50. Finally, the remaining over two hundred records do not provide CAGE_YR, presumably because the date of vaccination was not known.

Further, if you look at a more typical data set, such as the full COVID19 data set, you will see in the data insights graph less than fifty percent omit AGE_YRS, but over seventy percent omits CAGE_YR.

Most of the time, unless you are doing a specialised query that demands it, you probably want AGE_YR. To avoid confusion, in the release candidate we have labelled these age fields to identify them as they are shown in the VAERS Data Use Guide.

Data Query Screen

Vax Lot and State

The vax lot, also known as the batch number, is a common way of classifying batches of vaccine made at the same time which may have been distributed to multiple different states.

For our purposes, as data researchers, it provides a quality control to identify if possible quality control factors related to a specific batch of vaccines may have contributed to a series of adverse events. For example, if there is an abnormally high death rate or rate of emergency room visits for a specific batch, this could indicate a temperature control issue, or a manufacturing problem.

In conjunction with State, this could be used to track possible temperature control issues that only impacted specific delivery routes. This at least is the theory.

However, in practice, these two fields suffer from data quality issues that complicate the investigation process. Indeed, for the conscientious researcher, you will be able to identify an entire block of records spanning over a year where no State information was recorded for specific vaccines.

Moreover, there are keying errors which commonly occur with the vaccine lot numbers. It is not uncommon for a reporter to key the name of the manufacturer, or the month relating to receipt of the batch, as a part of the lot number. The lot number will be keyed in upper, lower, or mixed case according to the whim of the reporter.

This is why the batch number has been made a dynamic query field with optional regular expression search. If the checkbox next to the batch number field is checked, the text you type will be folded to lower case and used as a case insensitive regular expression match against the lower case version of the batch field. If the box is not checked, the comparison will still be case insensitive, but the string is matched literally.

Data Query Screen

Vax Lot and State (continued)

Unlike batch number, state is not treated as a regular expression but as an ordinary string literal, matched without concern for case by folding both the text and the field to upper case. This is because state has been restricted in the VAERS form to limit it to a two character code.

Special to VAERS is the so called Non Domestic vaccine reports, marked by the State being set to FR (foreign). We will come back to this data set when we discuss the graph screen.

Despite there only being 52 US states, there are 61 unique values for state in the full COVID19 data set. We find clear keying errors at the end of this list such as QW, MH, XB and PW, none of which are real US states or classifiers.

In summary, be mindful of these issues as you carry out your investigations. The batch field being matchable as a regular expression helps by enabling you to match a subset of the batch number, or multiple batch patterns, but it does not magically fix keying errors such as keyboard mashing where adjacent keys are hit. Only by closely examining the results of your query will you be able to track down these problems and get a truly accurate picture of a batch's history. For more information, review the data quality notes in the later pages of this manual.

Data Query Screen

Symptom regex, date range, and VAERS IDs.

Date range works as follows: If and only if the field contains a string of the format YYYYmmdd-YYYYmmdd, the internally calculated (from RECVDATE) pdate column is used to identify records with a received date falling between the two.

For example entering 20200101-20221231 will select events falling within the years of 2020-2022. The dates act as a bound, in other words the comparison is less/greater than or equal to.

If a value is entered into VAERS IDs, it should be either a single integer number, or a series of integer numbers separated by the comma. Otherwise the field will be noted but ignored, and the remaining filters applied instead.

If the VAERS IDs can be interpreted as a series of numbers, VAERity will attempt to return **those reports only** as the data set. If the ids do not correspond to a report in the database, it will not return a record.

The utility of this function will be obvious to anyone who has worked with VAERS in the past.

On the next page we will examine the final field we have not yet touched on, one of the more powerful functions on the data query page: the symptom regex.

Data Query Screen

Symptom regex

When completing a VAERS report, reporters have the ability to select from an extremely comprehensive list of symptoms defined by the MedDRA dictionary. The full specification is outlined at https://www.meddra.org/

Each report can have an almost unlimited number of symptoms attached to it, spread amongst five fields labelled SYMPTOM1 through SYMPTOM5 in the VAERS symptom table.

There can be more than one line per VAERS id in the symptoms table, so a report with 25 symptoms may be spread across 5 or more records.

The purpose of symptom regex is to be able to match these multiple symptoms against your existing query as defined by all other fields other than VAERS IDs. For example you might search for (urticaria|pyrexia|dysopnea) to view all reports matching these terms.

The text you type will be treated as a case insensitive regular expression match via str_contains and the field and your text will be folded to lower case for comparison.

Example queries might be monkeypox (53 records), cardi (to match myocarditis, pericarditis, myocardial infarction, etc), or accident (13692 records)

Data Query Screen

The term **“accident”** is in itself a very interesting study in how terms are defined in MedDRA. There is not only the rather euphemistic “cerebrovascular accident”, but also “accidental exposure to product”, accidental overdose or underdose, and even “road traffic accident” (594 occurrences as of writing this). As one person quipped, I don’t know what’s worse, that its reported or that its a common enough side effect to have its own MedDRA term.

This is the very last field processed in order during the query process, just like VAERS IDs is the first. The filtering is done by extracting all records that match from the symptoms table, creating a sorted list of reports, and then doing a take on the identified records which match this list in the already filtered result set. This is done with some very fancy code to ensure it is done as efficiently as possible.

The symptoms that appear in conjunction with your search term will be of interest. For example, death is reported with road traffic accident in 27 of those 594 reports. Loss of consciousness is reported in over 150 of the reports. Seizure and impaired driving ability are also reported, along with several other symptoms that would clearly contribute to road traffic accidents. Over 250 of these reports are for the COVID19 vaccine, where the vaccine was specified. By combining a search for a specific symptom with the insights delivered in the two pages of provided graphs which may be further customized, clear hypotheses may be arrived at for further investigation by examining things such as the peak weekly occurrences of ER visits or deaths, manufacturers and vaccine types involved, and the days of onset (where provided).

Data Query Screen

SPLTTYPE / Report number field

NEW! In 1.0RC3

This was added as one of the final pre-release features to RC3 after I discovered the full impact of the removal of symptom text and data in this report number field from the UK and EU records. It is one of the more useful fields for forensic analysis.

Although it is not well known, prior to the VAERS purge of November 2022, extensive reporting was carried out by countries such as the UK and European countries such as France, Romania and so on. The first two characters of the SPLTTYPE field in the CSV, where the report code is not a special one such as those starting WAES, is the country code.

By filtering on this field, you can view all reports for a specific country. Using this in conjunction with the by time patient outcome graphs allows you to see not only patient outcomes throughout the pandemic by country, but also to see the prevalence of reports without a patient outcome by country. These absence of outcome entries seem to typically occur when manufacturers or reporters batch large numbers of separately recorded incidents, and often contain minimal information to classify the severity of the outcome or even when the event occurred in some instances.

Thus the quality of reporting by country can be compared, and any country specific anomalies can be addressed. One such anomaly is the prevalence in the UK of reports, presumably under the Yellow Card System, of a high number of individuals reported disabled, higher even than the number of hospitalisations.

Although it could be argued that now that future European and UK entries will not have any narrative or report number, including this field is a little too late, this is not true. Copies of the prior data still exist and can be downloaded through the appropriate channels, and integrated with new data via the Merge Export functionality. Moreover, many countries continue to report in VAERS including Australia, Japan and others.

Data Quality Concerns

Data Quality Concerns

Before moving on to the record viewer and customizable graph screen, it will be apropos to consider the common data quality issues that may be encountered in querying VAERS, as well as potential incorrect assumptions that may be made about the data. This will help you to avoid pitfalls such as attributing more significance to a data point than is appropriate, which can occur in data which has a high prevalence of missing data.

The first thing to understand is that a VAERS report is not a static thing. After it is reported, it may be altered by staff responsible for maintaining data integrity in the course of an investigation, or even after the investigation is complete. A common change to records appears to be the removal of the report itself in the SYMPTOM_TEXT field. It is unclear whether this is a standard operating procedure to safeguard patient confidentiality/privacy, a way of designating completed investigations, or something completely different. One confirmed reason for doing this as identified in November 2022 is in response to requests by international regulators.

Another common concerning data quality issue that you may encounter are large batches of reports with no patient outcomes or symptom text which only contain symptoms and minimal other fields. This is so prevalent in certain date ranges that the ability to filter graphs by absence of outcome was added to provide visibility and insight over what was going on.

There are many possible causes for missing or poor quality data, including outsourcing of reporting to third parties, a confusional state in the patient at the time of reporting, and more.

Data Quality Concerns

Data Quality Concerns (continued)

Common fields of concern are age related fields, date of vaccination or days of onset, the batch number or the state. Data can be missing or incorrect (for example, keying errors)

However, other fields may have errors as well. For example, some batches, which are typically manufacturer specific, appear to span manufacturers. A Pfizer batch may be occasionally listed as Moderna on a report, or vice versa. A patient may have gone to a vaccination hub and have no idea what they got. These are practical facts of life that need to be considered when interpreting the data.

To make things easier for the researcher, we have provided tools such as the Data Insights by Percent Filled graph, and the ability to filter some graphs by lack of an outcome, as well as the ability to remove outliers from graphs either by setting limits for each axis, or by filtering out specific terms using a regular expression filter.

Additionally, to ensure that this does not remain too abstract, I will examine the batch EK5730 next to show the range of keying or data entry problems that can occur.

Data Quality Concerns

Data Quality Concerns - EK5730

Here are some of the variants for EK5730 (note some of these are not returned by simply running a regular expression on the correct batch)

- EK5u30 (note how this is a clear keying error as the two keys are diagonally next to each other)
- There is case difference in entry (hence why both exact and inexact match use case insensitivity) eg ek5730
- There is a single record EK5700 that is possibly a misreading of EK5730 (as typically a batch will occur multiple times in the dataset, not just once, through sheer probability)
- There is EK573, leaving off the zero
- There is prefixes or suffixes with the name of the manufacturer or the word LOT, or even just a single hash character
- There is EK57030, with an extra zero added in the middle
- There is EK57 (space) 30
- There is EK5830 and EK5630, which may or may not be keying errors.
- There is "EKSF30 OR EK573", as if the person entering the report figured this was a multiple choice thing. Note that 57 may look like SF on paper if written in longhand.
- Sometimes all of these problems combine into one, as in "PFIZER LOT EK57", or even "PF-19EK5730"

Therefore you may need to search several partial batch numbers in order to identify all data quality errors, or use the work of someone who has come before you, such as howbadismybatch.

The Record Viewer

The Record Viewer.

Once you press the submit button on your query, the records will be collated. Depending on the complexity of your query, this may take anything between a few seconds, up to two minutes. Some aspects of queries are cached, meaning a query on the symptom text field may be faster the second time around after this field has been accessed.

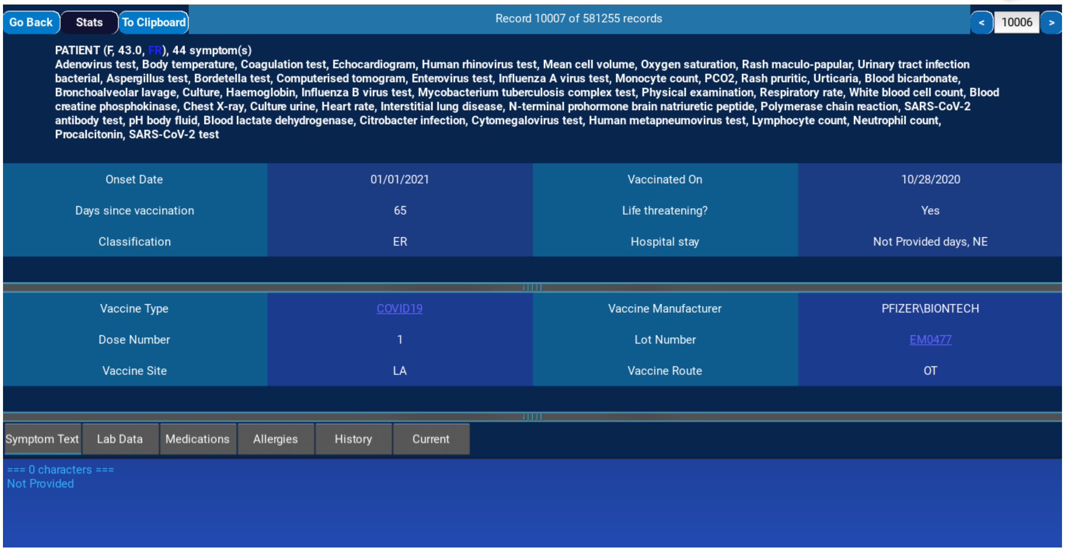
Once the records are collated, they are transformed into a pandas Data frame. For the non technical readers, all this means is that speed on the graph screen and record viewer are compromised slightly in order to allow a richer query interface and better handling of dates/categorical data.

On a laptop with no more than 8GB RAM, most of which is already used by the operating system and other running programs, this solution allows loading of at least one and a half million records without significant system degradation. Compared to the hard limit of 10,000 records imposed by WONDER due to the web based and distributed nature of that system, the flexibility of what can be done with the data is impressive.

In the next pages we will explore the basic functionality before moving on to the best part of VAERity functionality - the graphs.

The Record Viewer

The Record Viewer.



PATIENT (F, 43.0, FR), 44 symptom(s)
Adenovirus test, Body temperature, Coagulation test, Echocardiogram, Human rhinovirus test, Mean cell volume, Oxygen saturation, Rash maculo-papular, Urinary tract infection bacterial, Aspergillus test, Bordetella test, Computerised tomogram, Enterovirus test, Influenza A virus test, Monocyte count, PCO2, Rash pruritic, Urticaria, Blood bicarbonate, Bronchoalveolar lavage, Culture, Haemoglobin, Influenza B virus test, Mycobacterium tuberculosis complex test, Physical examination, Respiratory rate, White blood cell count, Blood creatine phosphokinase, Chest X-ray, Culture urine, Heart rate, Interstitial lung disease, N-terminal prohormone brain natriuretic peptide, Polymerase chain reaction, SARS-CoV-2 antibody test, pH body fluid, Blood lactate dehydrogenase, Citrobacter infection, Cytomegalovirus test, Human metapneumovirus test, Lymphocyte count, Neutrophil count, Procalcitonin, SARS-CoV-2 test

Onset Date	01/01/2021	Vaccinated On	10/28/2020
Days since vaccination	65	Life threatening?	Yes
Classification	ER	Hospital stay	Not Provided days, NE

Vaccine Type	COVID19	Vaccine Manufacturer	PFIZER\BIONTECH
Dose Number	1	Lot Number	EM0477
Vaccine Site	LA	Vaccine Route	OT

```
=== 0 characters ===
Not Provided
```

As pictured, the record view provides a snapshot of each report in the database include the symptoms, patient age and state, and narrative, along with all pertinent information summarised into two tables, one relating to the patient themselves and their outcomes, the other relating to the vaccine taken itself.

To get some useful information that provides an overview of the data range, you can press Ctrl+Alt+H (or select Analyse Dataset) within the record viewer to get the minimum and maximum patient age, maximum hospitalisation, maximum days of onset, top symptom count, and a list of all VAERS IDs with more than 50 symptoms (and how many that is). Note that this window may take up to half a minute to load depending on the size of your dataset. You will also see how many records by percentage do not have an outcome filled in.

You can copy up to around 900 of these VAERS IDs to your clipboard from the text field in a format that the data query screen will parse, if you are interested in exploring these types of record. Although most records you will view fall within on average up to 45 symptoms and as few as None, there are a certain percentage in the larger data sets that will have a more comprehensive list of terms, often indicating the type of tests being performed in conjunction with the common symptoms.

The Record Viewer

The Record Viewer (continued).

The reason this function has been provided is that these larger symptom sets, some without narrative, a few with, are a unique classification of report in and of themselves which is only provided in this fashion by specific types of reporter.

As such, the graphical overview of the symptom set generated by looking at these irregular reports (which can reach up to 210 symptoms and beyond) can be quite informative, providing a snap shot of the types of tests GPs are conducting for their patients presenting with symptoms post vaccination, as well as their results.

An example would be the top symptoms for all deaths,   On these 178 entries with more than 50 symptoms, we find for example endotracheal intubation, dyspnoea, intensive care; and, interestingly enough, haemoglobin decreased, and symptoms indicating kidney injury (such as creatinine and urea markers in the blood along with kidney injury as a symptom) and liver injury tests (Aspartate aminotransferase increased).

As you go down the list of top 30 or even up to 50 symptoms and beyond, a picture can be built up of the most common causes and modalities of death, which may inform options for optimising patient care and preventing future deaths. Additionally, the final feature added to 1.0RC3 is the ability to specify a range rather than just top 30/50 etc. This means you can effectively page through the whole of the symptom list on data sets where there is a lot of symptoms, which helps to prevent significant symptoms from getting lost in the signal.

The Record Viewer

The Record Viewer (continued).

Such data insights could be very helpful, for example, to private health insurance companies who could work with hospitals in analysing this data in such a way that hospitals can optimise their costs by purchasing the right drugs or medicines to prevent negative patient outcomes, as well as reducing risk in terms of payouts, as more life insurance payouts cost the insurers money. Making this data available at least provides the framework for this to be easily implemented if desired.

You can also copy the report data as text to the clipboard. Currently the rich text functionality is only partially implemented and will be completed in a future push. The current text output is a formatting of the data as html tables converted to text. In most cases the third party library which does the conversion gets it right, in some cases you may need to remove some unnecessary spacing/padding.

If you are outside of a textbox, the right and left arrow keys move between records, and entering a specific record number in the text box in the record header will jump to that record (zero indexed, so entering 9999 will show the 10000th record)

Each of the tabs at the bottom contains the text of one of the free form text fields VAERS defines, including allergies, other medications and so on. The text of any of these text boxes can be copied to the clipboard directly.

The Graph Screen

The Graph Screen.

Clicking the Stats Button in the Record Viewer will load the Graph Screen. The progress of loading the graphs is displayed in a popup window.

On a reasonable sized data set such as COVID19 FR with 580K records, the two pages of graphs are typically built in under one minute. The main bottlenecks are building or retrieving the list of top symptoms, converting dates not already converted from text, and aggregating histograms states and batches.

Once the graphs screen has loaded, you can close the popup window by clicking outside of it. The first page contains nine graphs, the second page five, for a total of fourteen graphs.

The following graphs are their own unique data insight or set and cannot be tweaked or repurposed:

- Data Insights By % Filled
- At a glance
- Reports by time
- Length of corpus vs Number of reports

The remaining graphs can be tweaked or repurposed. A graph can only be repurposed at present to its same graph type. There are top count graphs, histograms, and by time graphs.

The Graph Screen

The Graph Screen.

At present there are three histogram graphs on page one, and four top count graphs. On page two, there are two top count graphs and one other configurable graph.

The default configuration for page one is:

- Data Insights graph at top left
- Top 30 symptoms unfiltered for the whole record set (second graph from top on the left)
- Top Vaccine manufacturers (third graph from top on left)
- double histograms with 24 bins for age (age goes as high as 120, so this gives 5 years per bin) as the second graph from left at top
- ordinary histogram with X Bounds of 0,49 for length of hospitalisation as graph at top right
- A cumulative histogram of four subsets of the result set as the central graph by days of onset
- Top 40 states and Top 40 batches below that
- At a glance breakdown of the data set at far right second from top.

The Graph Screen

The Graph Screen.

Each of these types of graphs will now be described, along with its configuration options. A graph can be selected by pressing the Control key along with a number between 0 and 8, or selecting the graph directly from the drop down at the top of the page. The number to press for the hotkey corresponds to the number displayed next to the graph in the drop down.

Each graph can be configured, tweaked or repurposed by pressing Alt-R (re-purpose or reload).

Histograms

Histograms are about counting things: how many, and groupings.

To understand the patterns in data, histograms are invaluable. They are your primary tool. The Top count graphs are themselves a generalisation of a particular type of histogram.

Imagine you are trying to understand or decipher an ancient language you've never seen before. As you read, certain characters or letter patterns repeat. Things like Male and Female, or a symptom selected from a drop down, have repeating patterns. This is how we learn.

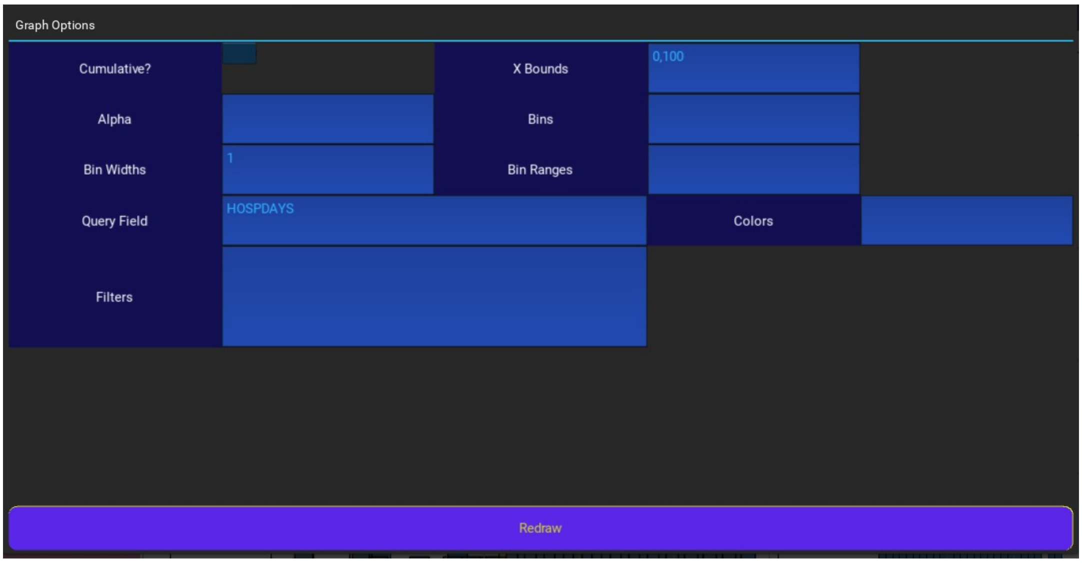
A dataset is like a language in that it has its most common patterns, like myocarditis, or dyspnoea, arranged in a syntax with patient outcomes etc, to form its own set of relationships that describes amd defines it. Each node or classification has its own weight, like a neural network. You understand the data by understanding the relationships between the different groups.

The Graph Screen

The Graph Screen (continued)

Although the word histogram comes from the Greek root for a mast, or indeed anything stood upright, in data science a histogram can be displayed in several different ways: with vertical or horizontal bars, cumulatively, with multiple histograms sharing the same x axis or even stacked on top of one another. There can also be different ways of grouping the bars.

These aspects of histograms are controlled through the Graph Options screen by pressing Ctrl+R.



Graph Options

Cumulative?		X Bounds	0,100
Alpha		Bins	
Bin Widths	1	Bin Ranges	
Query Field	HOSPDAYS	Colors	
Filters			

Checking the cumulative box will make this histogram cumulative along its primary (x) axis.

The bins, bin widths and bin ranges options control how data is grouped.

The Graph Screen

The Graph Screen (continued)

Generally speaking, you will either set a bin width for a histogram (eg 1, 2, 3.5), or you will set the bins and optionally the bin range. The range restricts or removes outliers. This is useful particularly with NUMDAYS and HOSPDAYS, as the largest values for these columns are in the tens of thousands for the first of these, to a thousand or slightly over for the second.

For example, on the cumulative by days onset graph, the binwidth is set explicitly to 1 with a range of 0-365 but setting X bounds of 0-60. This means each bar will represent one day, and 60 days of data will be displayed, but 365 will be plotted. This means that the X bounds can be changed later on to reveal more data without re-plotting the graph. Anything beyond 365 days of onset will not be displayed or counted.

Values on the graph options screen are generally separated by commas if you are specifying multiple values, with the exception of the bin ranges, which are themselves comma separated lists, which are separated by semicolons, and the filters which are separated by new lines.

If you’re only plotting one histogram, you only need to specify one value per option.

The Graph Screen

The Graph Screen (continued)

The Filters, if used, filter the dataset further. This is primarily useful to plot multiple histograms for different outcomes, sex, or state etc. For the technically minded, these filters are nothing more or less than input to a pandas query.

For the non technically minded, if you use the filters, the names of the fields you can select are generally on the data insights graph on page 1. You can use all the typical operators: == (equals), less than, greater than, etc: < <= >= >.
Here is an example web page which gives some examples of the format of different filters.

By using the same query field with different filters, the different histograms are visually displayed one after another on top of each other and coloured according to their respective colours and transparency (or *alpha value*). When doing this, ordering of the histograms and the alphas can be critical. I usually recommend using a high alpha value and placing the smallest result sets last, so that they are plotted over the top of the previous, larger bars.

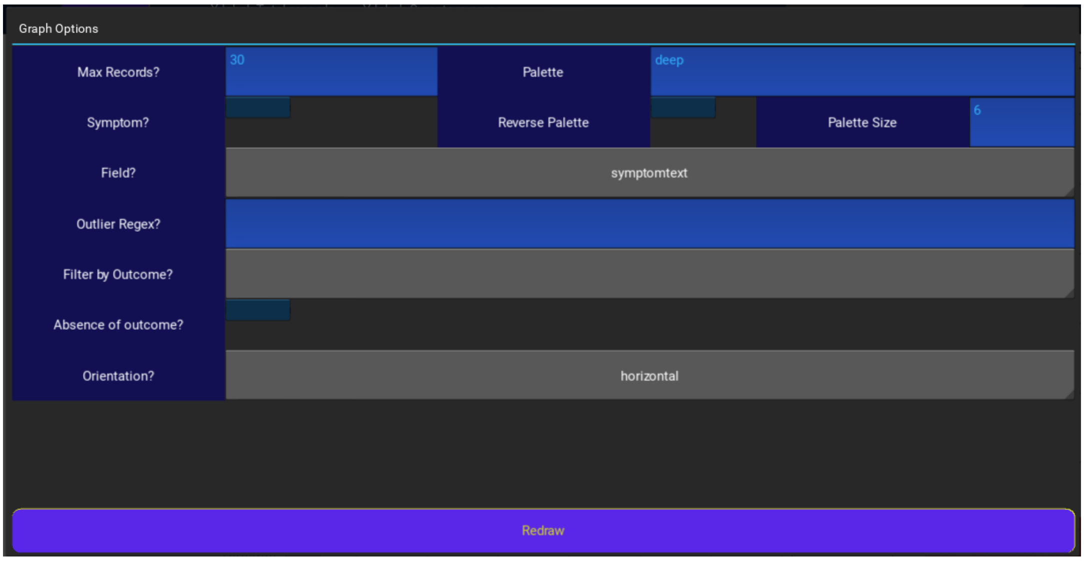
Colors

Colors supported are outlined here (the list is fairly comprehensive)

The Graph Screen

The Graph Screen (continued)

Top Count graphs



Graph Options

Max Records?	30	Palette	deep		
Symptom?		Reverse Palette		Palette Size	6
Field?	symptomtext				
Outlier Regex?					
Filter by Outcome?					
Absence of outcome?					
Orientation?	horizontal				

These graphs display up to *Max Records* records of the data set field in the *Field* dropdown, and the total records for each value.

Many fields are somewhat categorical (meaning there are limited values that it can be set to), with the exception of text fields such as the patient narrative, date fields, and continuous values such as age (floating point). Anything that is a spinner (sex, manufacturer, vaccine type, dose number) is at least pseudo-categorical.

Instead of individual colours, the colours of the bars of the top count graph are defined by the entries in the color palette selected. The color palettes are defined here and here, for the technically minded, any palette descriptor string supported by seaborn or matplotlib can be used (including the cubehelix palettes). You can reverse the default order of the color palette, and specify the number of entries in the color palette. On continuous color palettes which fade to white or very pale colors, I typically recommend selecting a number slightly more than your maximum records, to ensure the color of the final bars is not completely transparent.

The Graph Screen

The Graph Screen (continued)

You can plot the graph horizontally or vertically by selecting an orientation.

If you don’t care about certain values on a graph such as COVID19, Pain, Fatigue etc on the Symptom graph, you can remove them using the Outlier Regex. For the technically minded, this is passed to str_contains for non symptom graphs, or to re.match for symptom based graphs. The match by default is case sensitive, if this bothers you, you can use a non matching group (?i) to turn off case sensitivity.

Finally, you can also filter by patient outcome, or even by the absence of an outcome. The absence of outcome option is typically used to get a handle on the nature of a specific type of data quality issue that is encountered in the data, particularly in 2022. Large numbers of records without patient outcomes containing only symptoms and limited other fields exist in the second half of 2022, and may also exist elsewhere in the data set. This allows these records to be identified so they can be queried for what information they **do** contain.

New in 1.0RC3 is the ability to specify a range such as 50-100. If this is specified, please note that the start index is zero based. This means the final bar on a graph with range 50-100 will be record 99, which is is the record with index 99, and the 50 records displayed will be the 51st through 100th records. It is important to know this so that you don’t specify, for example, 1-50, which would display the 49 records starting with record #2.

This range option gives you the ability to page through symptoms, batches and so on and not lose your signal in the noise.

The Graph Screen

The Graph Screen (continued)

Patient Outcomes By Time

There is only one graph with this type configured by default, as the fifth graph on page two (selected using Ctrl+4 when the page is selected). There are four options you can select to change how this is displayed.

By default, the graph appears as a non cumulative, unstacked set of lines representing the number of reports received each week for each patient outcome. However you can choose to represent this number of reports as a percentage of the size of the entire data set.

For the full COVID19 FR data set, you can expect to see the standard peak around 0.65% per week at the end of August 2021 for hospitalisations, with the remaining lines straddling 0.1 percentage points from 0.05 to 0.15, the lowest always being the most serious outcome of death. Deaths drops below 100 and finds its baseline not long after April of 2022, with a mean of about 50 per week being reported, after having reached a peak of around 350 per week during the height of the pandemic.

You can also stack the lines or make the lines cumulative. These options were originally provided to investigate the number of entries without an outcome, which has now been added in 1.0RC3 as a fourth and final option. This line of absence of outcome turns out to be very important due to reporting processes followed by manufacturers and/or regulatory authorities in different countries, which includes batching large numbers of reports to VAERS which then get processed with the same RECVDATE. These reports frequently have large amounts of missing data and often are nothing more than a list of symptoms a patient has reported, sometimes not even having days of onset.

The Graph Screen

The Graph Screen (continued)

Patient Outcomes By Time (continued)

This becomes clearer when we examine the At a glance graph. The hospitalisations for COVID19 FR peak out at around 115000, the disabled at around 45-48K, clinic visits at 40K, ER at 35K and life threatening conditions at around 22K. Deaths comes in at around 18 or 19K, with birth defects being negligible. We end up with a total of around 280K records with an outcome. 280K out of a total record set of around 580K records.

The record set seems split around 50/50 between no outcome and an outcome. In fact, there is precisely 52.18% without any outcome (including recovered patients as an outcome) in COVID19 FR. This is an incredibly large data quality issue. When over half your reports do not specify how the patient ended up, even down to dead or alive (DIED vs RECOVD), this goes beyond simple negligence on the part of the reporter. I would suggest that an investigation of the causes of this data problem are of critical importance to ensuring public confidence in the data set providing meaningful data.

To help with this investigation therefore, the absence of outcome line has been added to this graph effective from version 1.0RC3.

One observation arising from adding the ability to filter by absence of outcome on the top count graphs is that for the COVID19 data set, at current date (14/02/2023) there are 188 deaths recorded using MedDRA term death that do not have DIED as an outcome. It is also possible to observe, in the filtered symptom graph, the existence of very serious symptoms such as Bells palsy which would ordinarily be classified as disabling. Other conditions such as myocarditis or pericarditis also appear without an outcome, and these are often life threatening conditions.

The Graph Screen

The Graph Screen (continued)

Patient Outcomes By Time (continued)

It is difficult to accurately assess the impact of this data quality issue. In the COVID19 data set (full, not foreign) there is 51 percent of records without an outcome. However in Australia's data set (all records with SPLTTYPE starting with AU), this goes up as high as approximately 75 percent! By comparing the relative size of the line in comparison to the key outcomes, or by examining the Analyse Dataset output which shows the overall percentage, the quality of different countries reporting data can be quantified, a fact that may help drive enquiries around accountability in the countries in question!

I recommend reviewing both, as the reporting quality can vary over time. France's data just before the purge is particularly interesting.

The Graph Screen

The Graph Screen (continued)

Data Insights By % Filled

This graph is perhaps one of the most useful and underused graphs in the program. It simply shows the number of missing or na values as defined by the pandas function isnull().

For some fields, it is normal for there to be significant missing data. For example, patient outcome fields and their corresponding day fields. Briefly: DIED, HOSPITAL, DISABLE, ER_ED_VISIT, OFC_VISIT will be missing if this outcome did not apply to the patient.

Similarly, if a patient was not hospitalised, their days of hospitalisation can also be assumed to be null or missing. Therefore for such fields, the line simply shows an approximate proportion of records that **do not** have that outcome.

By comparing the length of the lines for the corresponding fields, data quality issues can be picked up: for example, comparing HOSPITAL and HOSPDAYS, or DIED and DATEDIED. If the non outcome field is shorter than the outcome field, it implies missing data, as the ideal outcome would be a date of death or number of days hospitalised for each patient with the outcome.

Age fields are particularly vulnerable to missing data and I highly recommend checking these fields and HOSPDAYS if histograms for reasonably sized data sets look chunky. I also recommend viewing the line for VAX_LOT in conjunction with the Top batches graph. In fact for any graph you are interpreting results for, it is useful to first check how well it has been filled in to avoid an inaccurate interpretation. This can be particularly problematic for histograms, where missing data might cause you to conclude an age correlation or a hospitalisation trend where none exists.

The data on this graph also provides you with knowledge about what fields are available to select from. Any field in lower case is a calculated field added by the program, such as osdate, weekdate (the start of a week as defined by the date minus the number of days defined by dayofweek) and monthdate (first day of the corresponding month). All of these are based off RECVDATE. Similarly, country_code is the first two characters of SPLTTYPE for use in filters or top count graphs.

The Graph Screen

The Graph Screen (continued)

Data Insights By % Filled

For the remaining fields such as the free text fields, the percent of these missing gives a baseline of how comprehensive the reports in the given data set are. A field may also be not filled in for other, trivial reasons, such as ER_VISIT being not filled in on a data set containing only VAERS 2 reports.

When looking at the other graphs that specify count of things, I suggest getting used to glancing back at this graph as a matter of course. For example COVID19 FR has almost 120K hospitalisations, but these have HOSPDAYS poorly filled out with all but a fraction showing as missing values. As a result, you see some jagged spikes as artefacts on the histograms and low counts and graph area compared to the total records.

Length of corpus vs number of reports

As the name suggests, this gives histograms of the lengths of the reports in the data set. It is useful in order to provide more insight into how comprehensive the reports are beyond simply filled in or not. This shows the distribution and in conjunction with the graph manipulation commands (to be discussed next) it can be used to find outliers.

It is particularly handy to spot problems or data features which can be identified by multiple peaks in the histograms.

The Graph Screen

The Graph Screen (continued)

Graph manipulation

VAERity comes with a series of commands to make working with the large amounts of data these graphs can generate.

One common thing people will want to do is adjusting the upper and lower bounds of the graph, or the left and right bounds. For numeric graphs, you can do this by typing a number on your keyboard, any floating point or integer number, and then press Ctrl+L to set the left bound, Ctrl+R to set the right bound, Ctrl+U to set the upper bound, or Ctrl+D (**mnemonic**: down) to set the lower bound. This lets you zoom in on part of a graph if you do not care for instance about the huge spike at zero or this or that periodic spike in the data.

For time based graphs, you can type a date stamp such as 20230218 for Feb 18, 2023, and press Ctrl+T. This converts your time stamp to a format that the graphing software can understand, and then you can use the Ctrl+L or Ctrl+R mentioned above.

This comes in particularly handy on screen number two, since the 30 years prior to the pandemic often squishes the key data we are interested in into a few centimetres of mashed up multi colored garbage. These bounds are not set automatically and the design decision behind this is that second guessing the user or setting fixed bounds should be avoided where possible, particularly as this program will be expanded in the future to other health data sets.

The Graph Screen

The Graph Screen (continued)

Graph manipulation (continued)

For graphs with categorical data along an axis, such as states or batches, you should bear in mind that matplotlib treats these graphs as a series of rectangles with the first starting at zero.

If at any time while typing your number you make a mistake, you can press Ctrl+C to drop what you have typed and start again. The number you are typing appears at the top of the screen to the left of the graph dropdown.

For vertical graphs where the bars are upright rather than left to right, you can use the Ctrl+[and Ctrl+] keys to just nudge a graph’s left or right bound. The first of these takes the right bound in, the second moves the left bound. The method used attempts to identify the next or previous value on the x axis. I do not recommend using these options on horizontal graphs for obvious reasons.

The numeric keys with Ctrl will select which graph you are operating on. The number corresponds to the number next to the graph in the drop down menu at the top of the page. You need to select a graph before changing its bounds with the commands listed above.

For the selected graph, you can also click on the word Reset next to an axis at the top of the page. For the X axis, this will attempt to reset its bounds based on the minimum and maximum values originally plotted. For the Y axis, it will attempt to calculate new bounds automatically based on the tallest bar currently on screen, plus a small percentage.

Finally, you can use page up, page down, the arrow keys, and Ctrl+Left or Ctrl+Right to move around the selected graph. Ctrl+Alt+N moves through the pages of graphs, of which there are currently two. It may take several seconds to load the next graph, so be patient.

The Graph Screen

The Graph Screen (continued)

Graph manipulation (continued)

At any point while on the Graph Screen, you can select to Save Figure, which will save the currently displayed graph as a png file. Currently only png format is supported and selecting a different file extension will not affect the data saved. The graph can be saved anywhere on the same drive that VAERity is installed on.

Final tips for the graph screen: Make full use of the ability to re-purpose graphs. If you are not interested in the graph by Manufacturer, re-purpose it as a second symptom graph, or to display top countries in the data set filtered by outcome.

Also make full use of the filter options available to you. These options have been given to improve efficiency so you don't have to continuously revise your query for the data set you are viewing. So for instance, if you are viewing the full COVID19 FR data set, you might filter by deaths and select country code to see which countries have reported the highest death toll that has been ascribed to adverse events. If you have only one state in the data set such as FR, re-use that graph for something else, and so on.

If future development is supported by the community, a key development goal is to allow these default graphs to be changed/personalised at a program level by editing the XML file describing the data set (and possibly at a per user level). I also expect to provide a blank page which can be subdivided for graphs however the user desires, using a gridspec.

Merge Export

Other Functionality

In light of knowledge of the deep impact that the November 2022 VAERS purge had on data scientists studying the VAERS data set, the Merge Export command has been added to 1.0RC3. This function does something very simple but useful: it treats existing records in the data set as static. Here's how it works:

1) After clicking on the option, select which data file you are working with – data, symptoms or vax. You must run an export for each file in the set you are doing this for. Typically this will be the Non Domestic files.

2) Select which drive your data files are on.

3) Select the first file, which is the one whose existing entries you want to remain unchanged. Select the second file, which is of the same type, containing possibly the same records as the first plus new records.

4) Navigate to the destination folder, type in the name of the file you want to save the output to (typically the same as the first two), and press Dest to set the destination file.

5) When you press Merge, the system will read in the CSV in question as a Vaex data frame, identify the VAERS IDs in the file, and copy the original to your destination name. It will then filter the second file to only records that do not match the first file, and append these filtered records to the destination file. The progress of this operation appears in the status bar of the main window.

6) After you have done this for all three files, create a folder with the other data files for years that are not changing, and copy your merged files into that folder. If there are any corresponding hdf5 or yaml files, delete them and then edit your %APPDATA\vaerity.ini config file to point the program at the new folder and restart the program.

Styles

Styles

With only a few exceptions, VAERity’s colours are completely configurable. Fonts and font sizes will be made configurable in the next major release.

The default styles are configurable through the config screen which can be accessed using Ctrl+F1. However due to the number of colours and elements, this interface can be confusing.

If you just want to change a particular color for the duration of the run of the program, select Ctrl+Alt+C. This will bring up a popup window with a dropdown and a color wheel, with an update button at the bottom. The name of what you are changing along with its current colour are displayed. The colour should update dynamically on the screen when you press Update, if that item is currently on screen.

You can save the current styles to disk to load in other copies of the program, or for a later run. To do this, or to load an existing predefined or saved style, press Ctrl+Alt+S. Currently available are a bright theme; Alice Springs, a warm theme; and Cool Runnings, a cool theme.

Hotkeys

Hotkeys

The following list is not exhaustive. Some internal hotkeys exist that will not be discussed.

General:

- Alt+F4: Close Program
- Arrow keys, Page Up, Page Down: Navigates when in a scrollable form or graph. Left and Right arrows switch between records on the record viewer when not inside a text field.

Record Viewer:

- Ctrl+Alt+H – Analyse Dataset (Top Honkers List)
- Ctrl+Alt+D – Save/Analyse Database State Change (will be removed in future version). Not documented.

Graph Screen:

- Ctrl+U: Set upper bound on selected graph to Arg
- Ctrl+D: Set lower bound on selected graph to Arg
- Ctrl+L: Set left bound on selected graph to Arg
- Ctrl+R: Set right bound on selected graph to Arg
- Ctrl+[: Attempt to narrow the right bound to the next smallest x value recorded on the graph.
- Ctrl+]:Attempt to narrow the left bound to the next largest x value recorded on the graph.
- Number keys (including dash and fullstop): Enter Arg
- Ctrl+C: Cancel entered Arg
- Ctrl+T: Assume Arg is a timestamp in the form YYYYmmdd and convert it to a format that can be used to set the left or right bound of a by time graph.
- Ctrl+0-9: Select graph/subplot.
- Ctrl+Alt+N: Next graph (wraps to first)
- Alt+R: Select graph options / re-purpose graph

Hotkeys

Hotkeys (continued)

Style related:

- Ctrl+Alt+C: Change style in program
- Ctrl+Alt+S: Load or save style.
- Ctrl+F1: Configure styles (and other program options)

Debugging:

- Ctrl+Alt+M: Dumps a sorted list of loaded namespaces to the file modules.txt in the program folder. Only used if debugging a problem with the executable/portable version generally.

Known Bugs

Known bugs

The following bugs have been observed and will be fixed in a future push if development of the program continues to be supported by the community:

- The program may become non responsive or crash after some but not all conversions of new VAERS data files. When this occurs, killing the program and re-starting typically resolves the issue immediately.

- Using the Ctrl+square brackets on some graphs not intended to be used with that function may cause a crash or unexpected behavior.

- No checks are done for how much free memory exists before loading a data set. Thus, the amount of RAM on your machine will determine whether the program can load the entire dataset eg 2.4 million records or not. Note that the only sanity check is that if no change occurs to the size of the result set, the data query screen will not load the entire data set into pandas. Similarly, the program will not attempt to load a data set with zero records. However, this only becomes a problem if you have only 8GB of RAM and try to load more than one and a half million records.

- No checks are made to ask the user before overwriting a file when saving a CSV or a PNG.

- There are some known isolated issues related to styling flux capacitors, particularly when making dynamic changes to the color scheme, which will be addressed in the next release.

Future Development

Future Development

The initial development of this program was driven by my own up-skilling in attempts to develop a portfolio of my demonstrated experience in data analysis and software development. It also turned into a labor of love to open VAERS to ordinary people with no data analysis experience, so that real conversations could be had about actual data, conversations about risk management and cost-benefit analysis, to supplement the existing conversations about other warning signals.

In doing so it became exceedingly clear that gross data issues which can nevertheless be worked around to some extent existed in the data set due to the reporting processes used in different countries, and the reporting processes used by the diverse manufacturers.

These problems have been compounded as the result of the VAERS purge of November 2022 and it is hoped that for those individuals who can obtain access to one of the publicly available copies of the (already previously publicly available) VAERS data prior to the purge, they can continue to update their data set into the future while maintaining the critical data used to extract missing fields from iteration to iteration, thanks to the Merge Export functionality in this program.

Moving into the future, my personal funds which were relied on during the development of version 1 of this program have been exhausted, so future development will have to rely on community support. It is my hope that enough interest exists in opening other data sets to the public, or improving the existing functionality with contributions from others, for us to continue to develop material to serve and inform the community. It's my intention if I get enough subscriptions or support to begin developing the next iteration using an Agile methodology with full transparency via a blog, with changes pushed to a branch on the Github. Additionally, anyone who pledges on a membership will have priority on new feature requests.

Future Development

Future Development (continued)

Beyond that, there are also the development ideas for pluggable data sets and declarative configurable graphs I outlined in the project’s README file on Github.

My goals for the program if community support extends to three months would be a complete interface for querying any arbitrary data set (primarily file based, but realistically any data set which vaex can manage and spool to pandas), and eventually a machine learning component to do natural language processing on free text within the data sets. This would, for example enable grouping based on a machine learnt understanding of the MedDRA terms and their structure, and then there are word clouds or cluster relationship diagrams that could further analyse the symptom text and other fields for relationships.

Send any ideas, feedback or bug reports via the Github at https://github.com/leprechaunt33/VAERity

Finally, if you want to help support further development or want to show some appreciation head on over to https://www.buymeacoffee.com/vaerity

Thank you, and enjoy your explorations!

--Tony Stark Lives, 19th February 2003.

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.
It makes use of the works of Mateus Machado Luna.