ModViz: A Modular and Extensible Architecture for Drill-Down and Visualization of Complex Data

David Rademacher, Jacob Valdez, Endrit Memeti, Kunal Samant, Abhishek Santra and Sharma Chakravarthy

IT Laboratory & CSE Department, UT Arlington
{david.rademacher, jacob.valdez2, endrit.memeti, kunalnitin.samant, abhishek.santra}@mavs.uta.edu and sharma@cse.uta.edu

Abstract. Analysis of data sets that may be changing often or in real-time, consists of at least three important synchronized components: **i)** figuring out what to infer (objectives), **ii)** analysis or computation of those objectives, and **iii)** understanding of the results which may require drill-down and/or visualization. There is considerable research on the first two of the above components whereas understanding actionable inferences through visualization has not been addressed properly. Visualization is an important step towards both understanding (especially by non-experts) and inferring the actions that need to be taken. As an example, for Covid-19, knowing regions (say, at the county or state level) that have seen a spike or are prone to a spike in the near future may warrant additional actions with respect to gatherings, business opening hours, etc. This paper focuses on a modular and extensible architecture for visualization of base as well as analyzed data.

This paper proposes a modular architecture of a dashboard for user interaction, visualization management, and support for complex analysis of base data. The contributions of this paper are: i) extensibility of the architecture providing flexibility to add additional analysis, visualizations, and user interactions without changing the workflow, ii) decoupling of the functional modules to ease and speed up development by different groups, and iii) supporting concurrent users and addressing efficiency issues for display response time. This paper uses Multilayer Networks (or MLNs) for analysis.

To showcase the above, we present the architecture of a visualization dashboard, termed `CoWiz++` (for `Co`vid `Wiz`ard), and elaborate on how web-based user interaction and display components are interfaced seamlessly with the back-end modules.

1 Motivation

Since early 2020, when the Covid-19 cases were first reported in the US, the virus has spread to all 3141 US counties[1] in all states at different rates. As the hunt for a vaccine was launched, the number of cases has grown and leveled off based on the actions taken by different counties and states. Lack of a national policy and lack of synchronization between state and federal mandates have resulted in undesirable situations as compared to other coordinated efforts in other parts of the world.

[1]We focus on the USA as we have more accurate data for that although the pandemic is worldwide! Any country can be analyzed by swapping the data sets and with minor changes, such as prefectures in Japan instead of states.

From a data collection viewpoint, a number of sources provide features associated with a confirmed report, such as infected case, hospitalization, death, or recovery making this data set complex with diverse entity (person, county), feature (case, hospitalization, vaccination, ...), and relationship (similarity in cases, hospitalizations, vaccinations, ...) types.

Currently, many visualizations are used to plot the *peak, dip, and moving averages or colored maps* of Covid data, **without much analysis on the base data or inclusion of associated data** [5,6,1,7,12,11]. In other words, most of these focus on the visualization of base data using simple statistical computations. However, for a comprehensive understanding of the spread of the pandemic (or any data for that matter), there is a need to *analyse and compare the effects of different events (mask requirement, social distancing, etc.) and demographics, in multiple geographical regions across different time periods.*

Broadly, visualizations for a data set can be classified into:

I. Visualization of **Base Data**: There is very little *analysis* involved in this visualization. Visualization includes primarily statistical information. Attributes and visualization alternatives can be selected by the end-user. Temporal ranges, animation, and other visualization parameters can also be chosen. Some examples of this Category I objectives are:

- **(A1)** Did the vaccination drive increase confidence among people to take more road trips?
- **(A2)** In states with low per capita income, how testing progressed? Was there a surge in the number of cases?
- **(A3)** Has the death rate reduced in countries where most of the population has received all vaccine doses? What about countries where the vaccination drive is slow?

II. Visualization of **analyzed data**: *Explicit analyses* are performed on base and associated data prior to visualization. Various alternate visualizations may be produced for the analysed results and drilled-down details of results.

Typically a model is used for analysis and objectives computed using that model. Some examples of this Category II objectives are:

- **(A4)** In which regions was vaccination most effective? That is, how have geographical regions with maximum (and minimum) rise in cases shifted between the periods pre and post the beginning of the vaccination drive?
- **(A5)** Which regions got significantly affected due to long weekends/holidays (such as, Thanksgiving, New Year Celebration, Spring Break, Labor Day, ...)? What precautions need to be taken for future events? The inverse can be computed which may be very helpful as well.

Currently available online dashboards/visualizations primarily address parts of category I discussed above. For example, JHU (Johns Hopkins University) dashboard [5] shows a lot of base data and shows some of them also on a US map with circles indicating the numbers to get a relative understanding. Similarly, the WHO (World Health Organization) dashboard [11] shows base data for the world and a clickable map to show some base data for that country. For Covid data, most dashboards focus either on reporting and/or visualizing daily cases on maps ([12,1,11,2]) or generating time series plots and statistical information ([5,7,6]).

However, for category II, there is a need to **model** the base data which is dependent on the semantics of the data set. As an example, for Covid data, analysis is based on counties/states. We need to model *entities and relationships* in order to *analyze* and understand the data set from *multiple perspectives*. The result needs to be *visualized* to maximize understanding. In this paper, we use the Covid-19 data set as well as related information, such as population, average per capita income, education level etc. The focus is on an interactive dashboard architecture that is **modular, flexible, provides good response time, and supports both categories I and II above**.

For the analysis part, this dashboard uses the widely-popular Entity-Relationship (ER) model and its conversion to **Multilayer Networks**[2] **or MLNs** [20]. MLNs can handle multiple entities, relationships and features. Informally, MLNs are layers of networks where *each layer is a simple graph and captures the semantics of a (or a subset of) feature of an entity type*. The layers can also be connected. Moreover, an efficient **decoupling-based approach** proposed and used in [25,26,31] is used to analyze specified objectives.

The contributions of this paper are:

- An **interactive web-based dashboard** [3] for visualizing base and analyzed data using parameters.
- A **modular** architecture to minimize interaction between the modules to facilitate **development and optimization** of the system by multiple groups with different skill sets.
- **Extensibility** of each module to add analysis, visualization, interaction/display and/or optimization alternatives with minimal effort.
- **Multiple visualizations** of base and analyzed results.
- Use of **multilayer network for modeling** and performing analysis underneath.
- Guaranteeing **consistency** while providing good response time for **a large number of concurrent users**.

This paper is organized as follows. Section 2 discusses related work. Section 3 details the architecture of the dashboard in terms of its modules.

Section 4 presents base and objective-based analysis visualizations for the Covid-19 data set. Conclusions are in Section 5.

2 Related Work

Currently available online dashboards address category I and focus on reporting and visualizing daily cases on maps ([12,1,11,2]) or time series plots and statistical modeling ([5,7,6]). They are more focused on visualizing the base daily data. In contrast, drill-down of analysis of results is critical especially for complex data which has both structure and semantics. For example, it is not sufficient to know the identities of objects in a *community* (e.g., similar counties), but also additional details of the objects (e.g., population, per capita income etc.) Similarly, for a *centrality hub* or a

[2]A Multilayer Network is a set of networks (each network termed a layer) where nodes within a layer are connected by intra-layer edges and nodes between two layers can be optionally connected using inter-layer edges.

[3]Dashboard [24]: `https://itlab.uta.edu/cowiz/`, Youtube Videos: `https://youtu.be/4vJ56FYBSCg`,`https://youtu.be/V_wOQeyIB5s`. **Readers are encouraged to play with the dashboard and watch the videos.**

frequent substructure. As MLNs are being used as the data model, it is imperative to know the objects across layers and their intra- and inter-connections [19]. From a computation/efficiency perspective, minimal information is used for analysis and the drill-down phase is used to expand upon to the desired extent. Existing MLN algorithms, especially the decoupling-based ones, make it easier to perform drill-down without any additional mappings back and forth for recreating the structure [31,25]. The schema generation also separates information needed for drill-down (Relations) and information needed for analysis (MLNs) from the same Enhanced Entity Relationship (EER) diagram [20].

Visualization is not new and there exists a wide variety of tools for visualizing both base data, results, and drilled-down information in multiple ways [1,5,7]. Our focus, in this paper, is to make use of available tools in the best way possible and not propose new ones. For example, we have experimented with a wide variety of tools including, maps, individual graph and community visualization, animation of features in different ways, hovering to highlight data, and real-time data fetching and display, based on user input from a menu. The main contribution is our architecture with a common back end to drive different user interaction and visualization front ends. We have also paid attention to efficiency at the back end by caching pre-generated results and use of an efficient data structure for lookup.

Community detection algorithms have been extended to MLNs for identifying tightly knit groups of nodes based on different feature combinations ([17,32,22,18].) Algorithms based on matrix factorization [16], cluster expansion [21], Bayesian probabilistic models [33], regression [15] and spectral optimization of the modularity function based on the supra-adjacency representation [35] have been developed. Further, methods have been developed to determine *centrality measures* to identify highly influential entities [29,34]. However, all these approaches *analyze a MLN by reducing it to a simple graph* either by aggregating all (or a subset of) layers or by considering the entire MLN as a whole, thus leading to loss of semantics as the entity and feature type information is lost.

3 Modular Dashboard Architecture

As part of research on big data analytics (using graphs and multilayer networks), the need for drill-down and visualization of results for understanding and ground truth verification has been emphasized. The results of *aggregate analysis* as compared to statistics, require more details (or drill-down). For example when a community of counties are computed or centrality nodes (cities) are identified, it is important to understand the related information such as population density, per capita income, education level, etc. This was further exacerbated by the fact that the data sets we deal with have multiple types of entities, features, and relationships. So, drill-down and visualization of analyzed data along with additional details became pronounced.

To clearly understand Covid data analysis results, it was important not only to drill-down, but also to visualize the data set and analysis results in multiple ways combining different aspects of the data set. For example, it was useful to visualize new cases in multiple states on a daily/weekly basis to see how they were changing. This could be done for multiple features, such as deaths, hospitalizations, etc. We also wanted to visualize similar regions in the country that had same/similar increase/decrease in new cases over the same time period. This would be very useful in understanding the effects of certain measures taken (e.g., masking, lockdown, social distancing) in different parts of the country. This essentially involved processing the same data under the categories I and II indicated above. This is also true for other data sets.

As we tried to develop a dashboard for Covid-19 visualization, we realized that the skill sets needed for analysis was significantly *different* from those needed for visualization/user-interaction. Analysis required a much deeper understanding of the knowledge discovery process including modeling of the data, coming up with objectives and computing them efficiently. On the other hand, visualization required a deeper understanding of the packages that can be used based on what and how we wanted to display. The client module needed yet another different set of skills in terms of layout, menu design, Java Script, HTML and CSS. It seemed natural that these could be developed by different individuals or groups with appropriate skills if the dashboard can be **modularized along these functional components**. This primarily motivated our architecture shown in Figure 1.

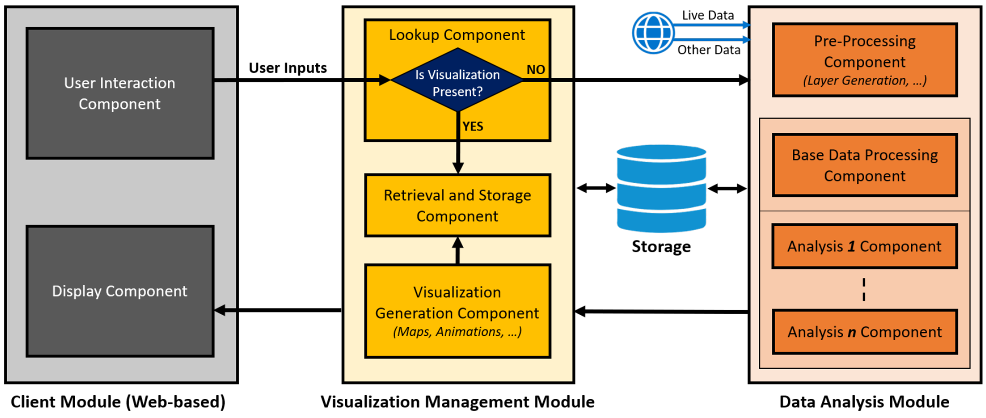
User Interaction Component
Display Component
User Inputs
Lookup Component
Is Visualization Present?
NO
YES
Retrieval and Storage Component
Visualization Generation Component (Maps, Animations, ...)
Live Data
Other Data
Storage
Pre-Processing Component (Layer Generation, ...)
Base Data Processing Component
Analysis 1 Component
Analysis n Component
Client Module (Web-based)
Visualization Management Module
Data Analysis Module

Fig. 1: Modular `CoWiz++` Dashboard Architecture

The second thing we noticed was that most of the currently available visualization dashboards seem to be *application and analysis specific*. That is, if the data set description and application objectives change over a period of time, then the entire system has to be re-built. Although there is likely to be a separation between the client and back end module, having a single back end module seemed to defeat extensibility in addition to modularity. This would create bottlenecks for progress making the development process quite inefficient. So, the requirement of **extensibility at the module level** was born out of this observation. This will also allow applying **different optimization strategies at the module level**.

Finally, **ability to visualize the same data in multiple ways** is extremely important from an understanding perspective. For example, one may want to visualize Covid cases/deaths/hospitalizations as a temporally animated graph for different states. One may also want to see the same data to make decisions by comparing geographical regions using MLN analysis [27]. Multiple visualizations and analysis capability in CoWiz++ follows directly from the extensibility aspect of the architecture. Currently, we support two visualization (one from each category above) as part of the visualization management module and multiple analysis (base and MLN) in the analysis module. We plan on adding more to each category.

3.1 Components of the Modular Architecture

Our proposed architecture and its components shown in Figure 1 have been designed to support the above observations: modularity with *minimal interaction* between the modules and extensibility within *each module*. Data is transferred between modules using file handles for efficiency as all modules are running on the same machine. These two, when incorporated properly, facilitate re-use of code within each module and the development of modules independently (by different groups) from one another for different applications. This is one of the major contributions of this paper, where we introduce 3 decoupled modules, each optimized for a specific functionality: **i) a web-based client**, **ii) visualization management**, and **iii) data analysis** modules. This architecture permits the optimization of each component, independently by separate groups with appropriate skill sets resulting in a flexible, extensible and efficient dashboard. In this paper, we show how the different modules interact and how a mix and match of analysis and visualization can be achieved. Also, note the minimal interaction between the modules (mainly parameters, file handles, etc.) As large number of files are used/generated by the two back end modules, a persistent storage is needed to store them.

There is a need for a closer synchronization between the client module and the back end visualization management module. For this to work correctly, the first step was to identify a web framework that can support these two modules, synergistically. The other considerations were: seamless communication, ease of use, availability of detailed documentation and strong open-source community for future development and extensions. Support for web deployment for increased portability was important.

	Minimalistic	**Language**	**Plotly Compatibility**	**Documentation Available**	**Flexibility and Control**
Flask [4]	**Yes**	**Python**	**Yes**	**Extensive**	**High**
Django [3]	No	Python	Limited	Extensive	Low
Vaadin [9]	No	Java	No	Limited	Low

Table 1: Web Framework Alternative and Feature Comparison

Table 1 lists the features of the widely used web frameworks that we considered. The **python-based web framework Flask** was chosen over Django and Vaadin, mainly due to its minimalistic, interactive, flexible and extensible characteristics. Flask satisfied all our requirements as shown in the table. Moreover, visualization tools like Plotly are supported exhaustively by Flask, which is not supported by others. Most importantly, as compared to others, it gives maximum flexibility and control due to granular tuning for customisation and makes no assumptions about how data is stored, thus becoming a viable choice for a wider spectrum of applications. Below, we describe each module emphasizing the modularity and extensibility aspects.

This modular approach allows **independent parallel collaboration** in **development, debugging and optimization** of every component. Any new user interaction component, data source, data model/structure, analysis algorithm and visualization technique can be added easily to a specific module, thus supporting **efficient extensibility**. Every module has various capabilities (compartmentalized through packages or sub-modules) which are **flexibly** utilised based on the requirements of the application. And most importantly, this **robust underlying system** can be readily used for different applications without major modifications. The following sections will talk in detail about these modules, in specific to the interactive COVID-19 data analysis and visualization. In Section 4, we show all these modules, together, seamlessly fulfill the goal - from accepting user inputs to analysing to displaying the results visually.

3.2 Interaction and Display (Client Module)

Each analysis and visualization uses a specific set of inputs given by the user. The client module is responsible for presenting an unambiguous, clear, and simple user interface for collecting those parameters. Once the parameters and display types are identified, this module can be implemented independently and the collected parameters are passed. The inputs can be in the form of ranges (dates, latitude-longitude, times, ...), lists and sets (features, items, ...) or filtering options. The various elements of this component are supported using HTML and CSS.

The other task of this module is to display the visualization generated by the other two modules, for the input parameters, typically in the form of an html file that is displayed using the *iframe component* which lets you load external URL elements (including other web pages) in your project within an iframe. An inline frame (iframe) is a HTML element that loads another HTML page within the document. In some cases, interaction with the HTML canvas element may be required to generate and display the visualization to enhance efficiency. In addition to displaying visualizations, this component is also responsible for tickers and other relevant information (part of display type.) For example, the visualization of top 10 Covid-19 news articles and the latest cumulative number of cases and deaths is achieved through *scrollable or moving tickers*, implemented using *JavaScript and AJAX scripts* and the *marquee component.* Note, this is based on the input and is done in real-time.

3.3 Visualization Management Module (Dashboard Back End)

Functionally, this is an important module, detailed in Figure 2, that handles several tasks: **i) visualization generation** – using either base data, or computed results – from the analysis module, **ii) reusing the generated visualization** using an efficient data structure[4], and **iii) looking up whether the visualization exists** for a given set of parameters and display type to *avoid re-generation and speedup response-time.* As can be seen in Figure 2, there are two separate visualization generation components, a hash and cache component for quick lookup and storage. Additional visualization generation modules can be easily added. This module interacts with the other two modules and the storage.

Since analysis and generation of the visualization accounts for most of the response time, we have used two known techniques for improving response time: **i) materialization of previous analysis results** – widely used in DBMSs to trade off computation with space and **ii) efficient hash-based lookup** to identify whether a **materialized visualization exists**. The first check in this module is to find the presence of the display file generated earlier. As there are hundreds of thousands of

[4]Currently, an in-memory hash table is used for quick lookup. If this hash table size exceeds available memory, this can be changed to a disk-based alternative (extendible hash or B+ tree) **without affecting any other module**. In this case, disk-based, pre-fetching and/or other buffer management strategies can be used to improve response time. Separate hash tables are used for different visualizations for scalability. Also, hash tables are written as binary objects and reloaded avoiding re-construction time.

possible user input combinations, avoiding collisions in hashing is important. If the display is present, it is used. If not, the parameters are sent to the analysis module to generate computed results so this module can generate the visualization after that. **This approach has shown significant improvement and has reduced the average response time from 15 seconds to 3 seconds (80% improvement)** for map visualizations (instrumented and averaged over several interactions.) This module uses packages from Python, R and Tableau to provide diverse types of interactive graphical visualizations. Two visualizations are currently supported by this module and are discussed briefly below. Also, note that the display as well as the ticker information is based on user input. Currently, after multiple user interactions from 20+ countries (as per Google analytics), 4000+ analysis result files and 1000+ map visualization files are present. The dashboard has been operational and publicly available for more than a year (getting more than 4500 hits from 20+ countries.) Due to the extensible architecture, we have been able to add new data (e.g., on vaccinations) with very little effort and with different developers.

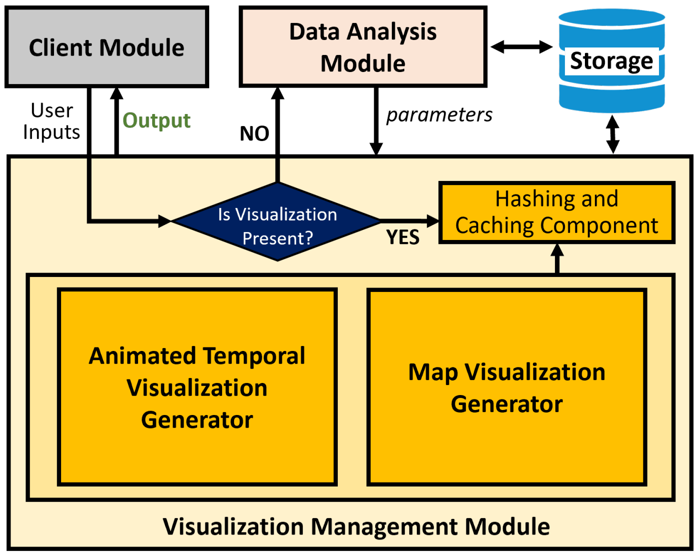
Client Module
Data Analysis Module
Storage
User Inputs
Output
NO
parameters
Is Visualization Present?
YES
Hashing and Caching Component
Animated Temporal Visualization Generator
Map Visualization Generator
Visualization Management Module

Fig. 2: Visualization Management Module Details

Support for Multiple Concurrent Users: The flask app generates a separate thread for each concurrent user. In order to maintain consistency in the back-end in the presence of multiple concurrent users, concepts of *multi-threading* have been used. *Critical sections* have been identified and *write-write* conflicting threads (that is, multiple threads with ***same*** parameters for which *visualization does **not exist***) have been properly synchronized so that the *consistency of generated visualization is guaranteed.*

Animated Temporal Visualizations: This is an example of visualization used for category I objectives discussed earlier. Based on the temporal requirements of **(A1) - (A3)**, the change in 2 selected features for up to 5 US states are compared by generating *2 side-by-side synchronized animated timeline plots* with a scrollable bar that the users can drag in either direction across the entire timeline. In the plots, the *per day* (or *per period*) values are synchronously plotted for each feature corresponding to the states (or countries) selected. In each plot the y-axis corresponds to one of the feature's values and x-axis to the timeline. The visualizations for these objectives are shown in Figures 6, 7 and 8 in Section 4.

Two implementation alternatives help showcase the extensibility of the system. Initially, the visualizations were generated using Python's popular *Plotly library and R language*, where two separate plots were displayed side-by-side by embedding in a single plotly subplot, to implement the *synchronised animated timeline with a slider.* This visualization was *stored* as an *html file*, and *sent over the network* to the client module for display. However, this alternative had *two drawbacks* - **(i)** The embedding of two feature-wise plots in the plotly subplot to enable the synchronization requirement led to *large processing time*, and **(ii)** The generated and stored html visualization files were *large* (approx. 20+ MB for 5 states with just 3 months of data points), thus taking a hit at the network I/O and page loading time. These issues led to a *high response time.*

To address these issues, we applied an optimization technique where *generation of the temporal visualizations have been shifted to the client side.* In this case, based on the user inputs (states and features) received in Data Analysis Module (Section 3.4), the required data points to be plotted are *fetched* from the data set and *stored* as an object. The visualization module sends over this generated object to the Client module. On the client side, these data points are used by the *customized* JavaScript classes, that have been written for interacting with the *native HTML canvas element*, to generate the required side-by-side line graphs with a synchronized timeline. This **optimization strategy** removed the overhead processing time to generate the plotly plots and the network I/O and loading time for receiving and displaying the heavy html files. Thus, **improving the average response time by about two orders of magnitude (from 5 minutes to less than 5 seconds for 5 states and 2 features)**. This also showcases that due to the modular and extensibility feature of the architecture, separate optimizations can be applied for different visualization needs. Moreover, apart from US states, support has been extended to **Indian States and World Countries**, as well. This support can be easily extended to additional countries, subject to data availability, due to the **parameterized approach followed in the module implementations**.

Map-Based Visualizations: As part of the requirements of category II analysis objectives **(A4)** and **(A5)**, communities are generated, where the counties are clustered based on similar change in a feature (in this case, similar change in *new cases*). Each county in the community is displayed on a colored US map based on the *severity* of changes in Covid cases reported in its assigned community which corresponds to a range - from SPIKE (as **red**) to BIG DIP (as **green**). The FIPS (Federal Processing Information Standards) codes of the US counties present in the community allocation file generated by Data Analysis Module (Section 3.4) are used by the *choropleth_mapbox() function of Python's plotly* library to generate colored counties on the US map with *pan and zoom capability* enabled. Moreover, the census information available as part of this file is used to generate the *hover text* for counties. The generated US map for a community file is stored as an *html file.* This visualization for objectives **(A4)** and **(A5)** is shown in Figures 9 and 10 in Section 4, respectively.

3.4 Data Analysis Module (Dashboard Back-End)

The analysis module is another key module of the architecture. This module contains all aspects of a particular analysis using the same base data. We have chosen to showcase the multilayer analysis for this dashboard. This can be any other analysis, such as relational database analysis using SQL or multi-dimensional analysis supported by data warehouses. In fact, multiple analysis modules can co-exist and feed the results into the same visualization management module.

It supports several components that are important for different aspects of data analysis: *i) extraction of relevant data* from external sources, *ii) pre-processing* of extracted (or downloaded) data, and *iii) generation of results* for both base and analysis alternatives. It is more or less agnostic to visualization except to generate information needed for visualization, but does not even know how they are visualized. All three components are extensible in their own right and only rely on the user input passed from client module through the visualization management module. This module interacts with the persistent storage for both input data and output generated. This module generates output for visualization. Of course, base data preparation is much simpler than the other one.

Extraction/Downloading and Pre-processing Component: Components of this module are responsible for the real-time extraction of data from identified web sources (e.g., New York Times, WHO, CDC), update of data that changes periodically – once a day/week/month (e.g., Covid data in our dashboard) using *cron jobs.* For example, when date ranges are input by the user as part of the menu, that information is used to extract information *only for those periods*, pre-processed (cleaned, filtered, sorted/ranked), and prepared for visualization. All pre-processing of data extracted from real-time sources as well as base data used for analysis are done in this component. Examples from the current dashboard are - **(i) Period Specific Top 10 Covid-19 Articles**: For category II, 2 periods are provided. From the set of New York Times articles, a subset of top 10 most relevant Covid-19 news articles for the *latest* period specified by the user are filtered using keywords, sorted in reverse chronological order and the top-k are chosen for display, and **(ii) Latest Cumulative Case and Death Count for Selected US States and/or Countries**: For category I, the latest total number of cases and deaths for the user specified US states (along with, US and World) are filtered out from the consistent clean WHO and CDC extractions.

Complex Analysis Component(s): Any analysis for the two categories discussed earlier is supported by this module. It can be *as simple as* fetching only required base data points or generating moving averages to be plotted to *as complex as* generating required models (graphs, MLNs, ...) and detecting network communities, centralities, patterns, frequent structures and so on.

For **(A1)** - **(A3)** in category I, just the *fetching and storing* of data points for the selected states and features is required. However, the category II involves modeling, computation of objectives, and drill-down before visualization. For understanding the *effect of vaccination drives and holiday breaks and on Covid cases in the US*, the formulated objectives **(A4)** and **(A5)** (stated earlier) are to be analyzed on the Covid data set using the Multilayer Network (MLN) model[5].

For **(A4)** and **(A5)**, *geographical regions* need to be analyzed across two periods for similar Covid spread. MLN layers are created using US counties as nodes and

[5]Any analysis approach and associated model can be used. We are using the Multilayer Network (MLN) model proposed in [25,28] for this dashboard. This also validates our assertion of the applicability of MLN model for complex analysis of real-world applications.

connecting them if the change of feature (e.g., new cases (shown in Figure 3), deaths, hospitalizations etc.) across the two periods is similar (using slabs of percentages.) Community detection algorithms (e.g., Louvain [13], Infomap [14], etc.) on the generated individual MLN layers for detecting communities that will correspond to geographical regions showing similar change in the feature. Any user-selected feature can be used for this purpose. The communities generated are categorized based on the severity of change in Covid cases - from spike in cases (>100% increase) to big dip (100% decrease). This generated *community allocation file* is enriched by adding the US census data like *population density per sq. mile, median household and percentage of high school graduates* for each county. The results of **(A4)** and **(A5)** are shown in Section 4.

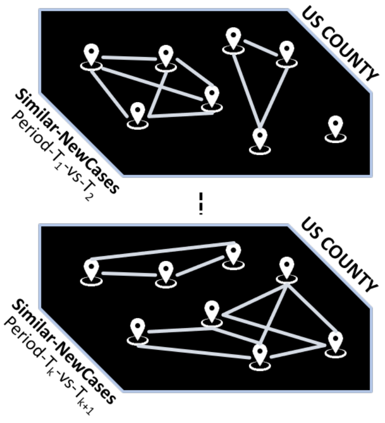

Fig. 3: MLN for Category II Obj.

The analysis module only needs to know the information used by the visualization management module and not the actual visualization *type*. This information is known to the client module for each type of user interaction and is passed on to other modules.

4 User-Interaction and Visualization of COVID-19 Data

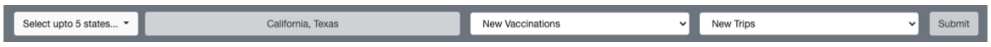

Fig. 4: Input Panel for Category I Objectives *(with inputs for Fig. 6 visualization)*

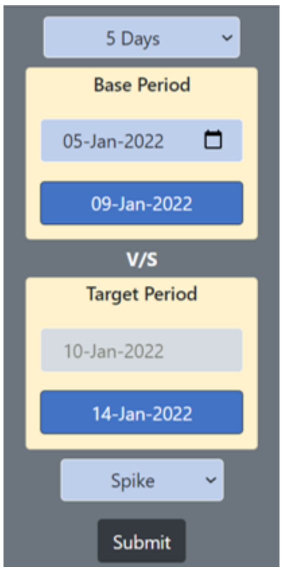

Fig. 5: Input Panel for Category II Objectives *(with inputs for Fig. 10 (b) visualization)*

The `CoWiZ++` dashboard is hosted on an Nginx Web Server 1.20.1 on Linux machine. It is supported on all major web browsers. For the best user experience, screen sizes above 1200 pixels are recommended.

The homepage of the current dashboard supports the two different types of analyses and visualizations. Figures 4 and 5 show sample user-interaction screens (with input), for Categories I and II, respectively. Category I objectives are currently supported for *World Countries*, the *US States* and the *India States*. Category II objectives are supported for *US counties*.

Here we discuss how the current dashboard has been used to address the analysis objectives from **(A1)** to **(A5)** based on *different periods*.

Vaccination vs. Road trips in US States **(A1)**: For *understanding correlation between vaccination and people taking road trips outside their homes*, the user-interaction component shown in Figure 4 is used. Figure 6 shows the completely rendered *snapshot* of the animated timeline depicting the correspondence between the *number of new vaccinations* and *number of new trips* undertaken by people in two of the *largest US states by population density* - **California**

and Texas, till April 2021. The plots reveal something interesting. **In Texas, new trips rose disproportionately to the vaccine whereas in California, that is not the case.** This conforms to our understanding of the way these two states have handled Covid. *Note the difference in scale between the two animations.*

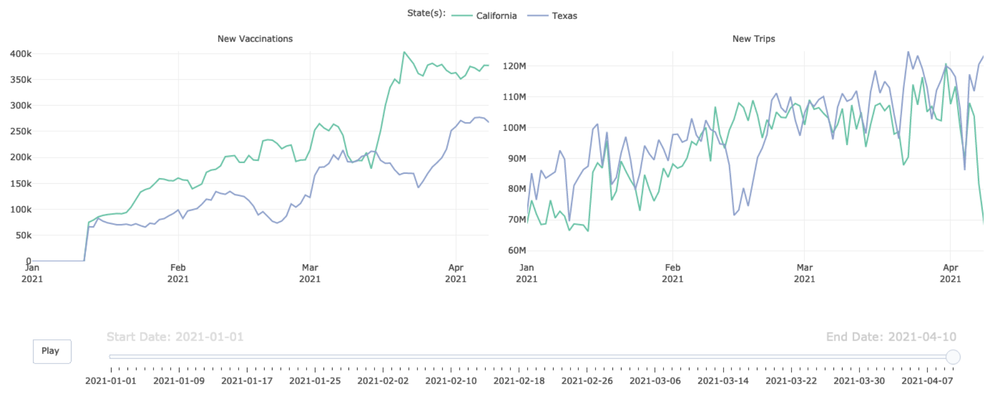

Fig. 6: **(A1)** Vaccinations vs. Road Travel Trend in 2 Populous US States California and Texas

New Covid Cases vs. Testing in US States **(A2)**: Testing for Covid is important according to CDC and should be continued independently of the new cases. For understanding whether this is the case, we considered **West Virginia** as the input for the paper, one of the **low per capita income** states. Figure 7 shows these animated plots side-by-side till April 2021. For an unknown reason, testing seems to follow new cases instead of staying constant. This seems to give the impression that **the ones that are being tested are mainly the ones coming with symptoms whereas general population is not likely being tested**. For a state-level decision maker, this can be useful as an important piece of information discovered through the visualization tool.

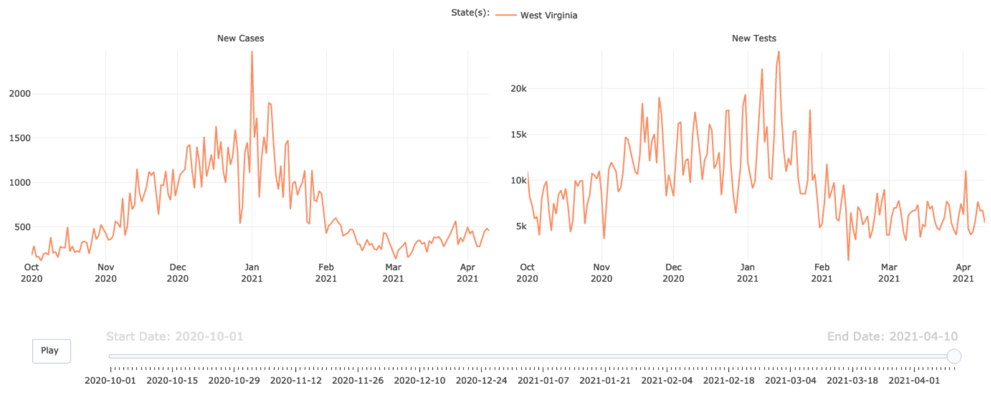

Fig. 7: **(A2)** New Cases vs. New Tests in Low Per Capita Income US State West Virginia

Fully Vaccinated Population vs. New Deaths in World Countries **(A3)**: Medical bodies across the globe have advised getting a large percentage of the population fully vaccinated at a faster rate in order to avoid serious Covid cases, and potentially achieve herd immunity. In order to understand this correlation, the snapshot in Figure 8 till June 2021 has been presented to illustrate the effect of getting a higher percentage of the population fully vaccinated. **Israel and the US** are two of the countries, where most of the people are **fully vaccinated** (**59.35% and 40.86%** , respectively, as of June 3, 2021.) Over time, with the rising vaccinations, the new Covid deaths have decreased in these countries. However, **in India where just 3.19% of the population was fully vaccinated until then, the number of new deaths had been on a rise and was more than the other better vaccinated nations**. Moreover, it is observed from *India's vaccination slope that the rate of second vaccine dose had also decreased since mid-May 2021.* Similar animations can be re-produced for different countries through the dashboard. Such **disparity in vaccination rates among countries is a cause of concern and a major roadblock in attaining global herd immunity against Covid-19** [10].

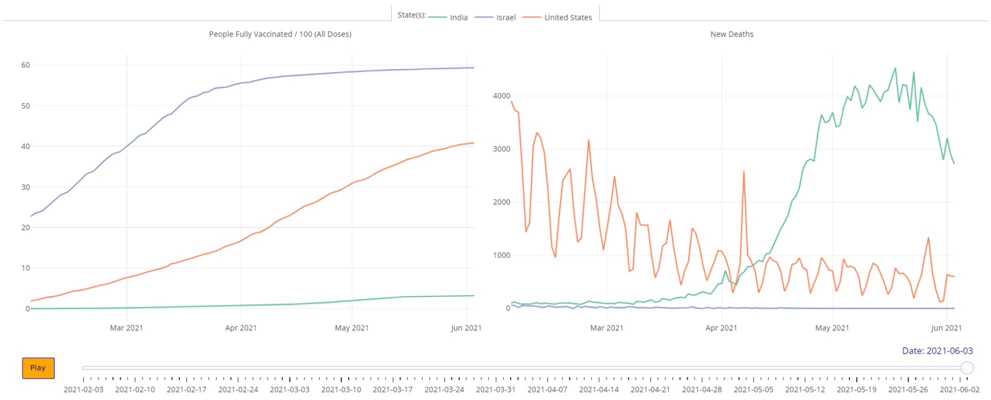

Fig. 8: **(A3)** Percentage of People Fully Vaccinated vs. New Deaths in India, Israel, and USA

Vaccination Drive Effect in the US **(A4)**: Here we visualize how the geographical regions with decline in daily confirmed cases shift in month-apart 3-day periods pre and post the *Vaccination Drive*. The vaccination drive in the US began from **December 14, 2020** [8]. For the *pre* vaccination drive layer, the 3-day intervals considered were Sep 20 to Sep 22 vs. Oct 21 to Oct 23 in 2020. For the *post* vaccination drive layer, the 3-day intervals were Jan 20 to Jan 22 vs. Feb 21 to Feb 23 in 2021. The *community* (groups of counties) results have been drilled-down from the individual layers and the ones displaying a downward trend have been visualized in Figure 9. This visualization clearly shows how the **vaccination drive became one of the reasons that led to controlling the spread of COVID across US**. This fact is also verified from independent sources that say how the administration of the vaccine has led to a *decline in severe cases, hospitalizations and deaths* in the US (and many other countries) even with newer variants around [30,23].

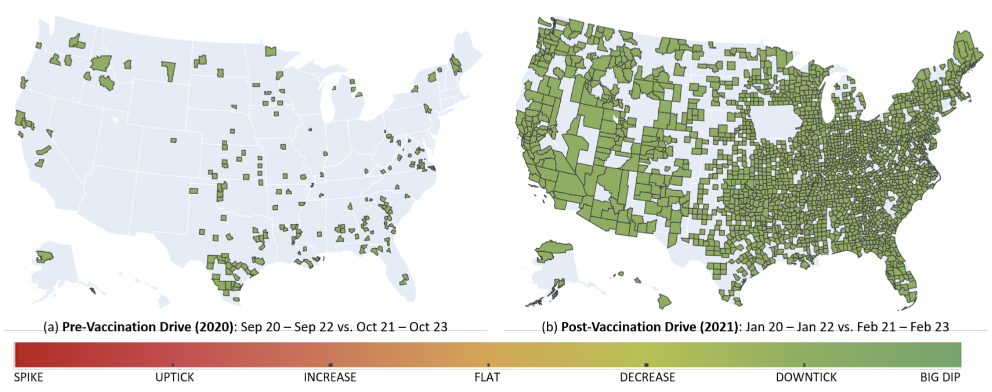
(a) Pre-Vaccination Drive (2020): Sep 20 – Sep 22 vs. Oct 21 – Oct 23
(b) Post-Vaccination Drive (2021): Jan 20 – Jan 22 vs. Feb 21 – Feb 23
SPIKE
UPTICK
INCREASE
FLAT
DECREASE
DOWNTICK
BIG DIP

Fig. 9: **(A4)** BIG DIP due to Vaccination Drive in the US

2022 New Year Holiday Break Effect in the US **(A5)**: For this category II visualization, we use the dashboard front-end shown in Figure 5 to first find out the geographical regions where a rise in daily confirmed cases was observed between two consecutive 5-day intervals *prior to the 2022 new year holiday break* - Dec 13 to Dec 17 vs. Dec 18 to Dec 22. Similar consecutive 5-day periods were chosen *post the 2022 new year holiday break* - Jan 5 to Jan 9 vs. Jan 10 to Jan 14. The drill-down results have been visualized in Figure 10 that show how **after the new year holiday break there was a spike in the number of daily cases in counties across the US as compared to pre winter holiday break.** For example, the *San Diego County in California showed a surge of more than 300% in the number of new cases post the new year break*, as illustrated in the zoomed in display in Figure 10 (b). Various reports attributed this massive surge to the widespread travel to popular tourist destinations during the break leading to **crowds and non-adherence to social distancing norms** at the time when the Omicron Covid variant was becoming dominant!

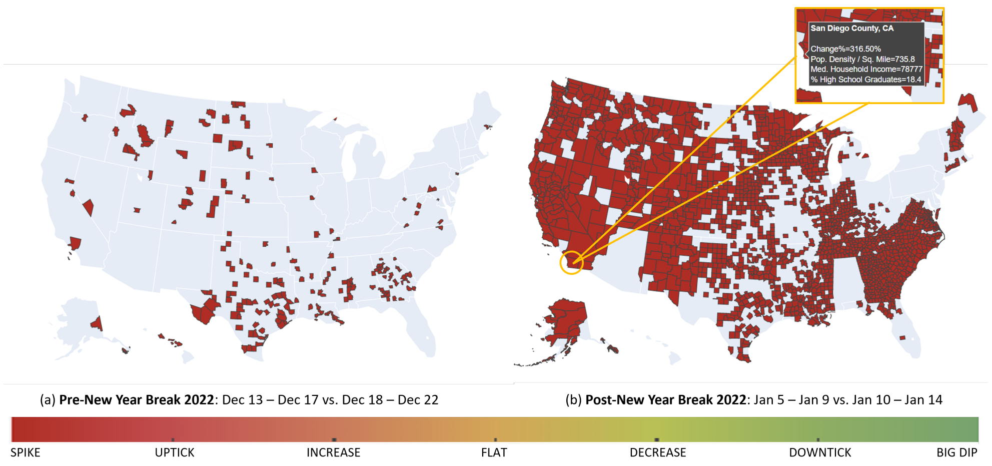
San Diego County, CA
Change%=316.50%
Pop. Density / Sq. Mile=735.8
Med. Household Income=78777
% High School Graduates=18.4
(a) Pre-New Year Break 2022: Dec 13 – Dec 17 vs. Dec 18 – Dec 22
(b) Post-New Year Break 2022: Jan 5 – Jan 9 vs. Jan 10 – Jan 14
SPIKE
UPTICK
INCREASE
FLAT
DECREASE
DOWNTICK
BIG DIP

Fig. 10: **(A5)** SPIKE in cases due to the 2022 New Year Holiday Break

5 Conclusions

In this paper, we have presented a modular dashboard architecture to visualize base data and complex *analysis* results meaningfully, based on input parameters interactively. In addition, we have enhanced it with display of relevant (top news articles from NYT for the period of interest) and real-time data (WHO statistics as they become available) extracted from multiple sources. The architecture and modularity, based on functionality, provide *flexibility* (of development and optimization), *extensibility* (of visualizations, analysis, and data sets), *consistency for multiple concurrent users*, and *efficiency* (for response time). Each component within a module is parameterized making it easier to replace data sets for similar visualization or change visualization for same data set.

Future work includes adding additional base data, other analysis options, and hierarchical visualizations for country and further into states. Other extensions to support multiple users efficiently and good throughput for large number of users are underway.

Acknowledgements

This work has been partly supported by NSF Grant CCF-1955798 and CNS-2120393.

References

1. The centre for disease control covid dashboard. `https://covid.cdc.gov/covid-data-tracker/`.
2. Covid-19 surveillance dashboard by univ. of virginia. `https://nssac.bii.virginia.edu/covid-19/dashboard/`.
3. Django web framework. `https://www.djangoproject.com/`.
4. Flask web framework. `https://palletsprojects.com/p/flask/`.
5. Johns hopkins university covid dashboard. `https://www.arcgis.com/apps/opsdashboard/index.html#/bda7594740fd40299423467b48e9ecf6`.
6. The new york times covid dashboard. `https://www.nytimes.com/interactive/2020/us/coronavirus-us-cases.html`.
7. The university of washington covid dashboard. `https://hgis.uw.edu/virus/`.
8. Us administers 1st doses of pfizer coronavirus vaccine. `https://abcnews.go.com/US/story?id=74703018`.
9. Vaadin web framework. `https://vaadin.com/`.
10. What would it take to vaccinate the world against covid? `https://www.nytimes.com/2021/05/15/world/americas/covid-vaccine-patent-biden.html`.
11. The world health organization covid dashboard. `https://covid19.who.int/`.
12. Worldometer covid statistics. `https://www.worldometers.info/coronavirus/country/us/`.
13. Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of community hierarchies in large networks. *CoRR*, abs/0803.0476, 2008.
14. Ludvig Bohlin, Daniel Edler, Andrea Lancichinei, and Martin Rosvall. Community detection and visualization of networks with the map equation framework. 2014.
15. Deng Cai, Zheng Shao, Xiaofei He, Xifeng Yan, and Jiawei Han. Mining hidden community in heterogeneous social networks. In *Proceedings of the 3rd international workshop on Link discovery*, pages 58–65. ACM, 2005.
16. Xiaowen Dong, Pascal Frossard, Pierre Vandergheynst, and Nikolai Nefedov. Clustering with multi-layer graphs: A spectral perspective. *IEEE Transactions on Signal Processing*, 60(11):5820–5831, 2012.

17. S. Fortunato and C. Castellano. Community structure in graphs. In *Ency. of Complexity and Systems Science*, pages 1141–1163. 2009.
18. Jungeun Kim and Jae-Gil Lee. Community detection in multi-layer graphs: A survey. *SIGMOD Record*, 44(3):37–48, 2015.
19. M. Kivelä, A. Arenas, M. Barthelemy, J. P. Gleeson, Y. Moreno, and M. A. Porter. Multilayer networks. *CoRR*, abs/1309.7233, 2013.
20. Kanthi Komar, Abhishek Santra, Sanjukta Bhowmick, and Sharma Chakravarthy. Eer→mln: Eer approach for modeling, mapping, and analyzing complex data using multilayer networks (mlns). In *Int. Conf. on Conceptual Modeling, ER 2020*.
21. Huajing Li, Zaiqing Nie, Wang-Chien Lee, Lee Giles, and Ji-Rong Wen. Scalable community discovery on textual data with relations. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 1203–1212. ACM, 2008.
22. Matteo Magnani, Obaida Hanteer, Roberto Interdonato, Luca Rossi, and Andrea Tagarelli. Community detection in multiplex networks. *ACM Comput. Surv.*, 54(3):48:1–48:35, 2021.
23. Ehud Rinott. Reduction in covid-19 patients requiring mechanical ventilation following implementation of a national covid-19 vaccination program—israel, december 2020–february 2021. *MMWR. Morbidity and mortality weekly report*, 70, 2021.
24. Kunal Samant, Endrit Memeti, Abhishek Santra, Enamul Karim, and Sharma Chakravarthy. Cowiz: Interactive covid-19 visualization based on multilayer network analysis. In *ICDE 2021*. `https://itlab.uta.edu/cowiz/` and `https://www.youtube.com/watch?v=4vJ56FYBSCg`.
25. A. Santra, S. Bhowmick, and S. Chakravarthy. Efficient community re-creation in multilayer networks using boolean operations. In *Int. Conf. on Computational Science, Zurich, Switzerland*, pages 58–67, 2017.
26. A. Santra, S. Bhowmick, and S. Chakravarthy. Hubify: Efficient estimation of central entities across multiplex layer compositions. In *IEEE ICDM Workshops*, 2017.
27. Abhishek Santra. *Analysis of Complex Data Sets Using Multilayer Networks: A Decoupling-based Framework*. PhD thesis, The University of Texas at Arlington, July 2020.
28. Abhishek Santra, Kanthi Sannappa Komar, Sanjukta Bhowmick, and Sharma Chakravarthy. A new community definition for multilayer networks and a novel approach for its efficient computation. *arXiv preprint arXiv:2004.09625*, 2020.
29. Albert Solé-Ribalta, Manlio De Domenico, Sergio Gómez, and Alex Arenas. Centrality rankings in multiplex networks. In *Procds. of 2014 ACM conf. on Web science*, pages 149–155. ACM, 2014.
30. Reis Thebault. Four reasons experts say coronavirus cases are dropping in the united states. `https://www.washingtonpost.com/health/2021/02/14/why-coronavirus-cases-are-dropping/`.
31. Xuan-Son Vu, Abhishek Santra, Sharma Chakravarthy, and Lili Jiang. Generic multilayer network data analysis with the fusion of content and structure. In *CICLing 2019, La Rochelle, France*, 2019.
32. Zhige Xin. *Community Detection in Social Networks*. PhD thesis, University of California, Davis, 2018.
33. Zhiqiang Xu, Yiping Ke, Yi Wang, Hong Cheng, and James Cheng. A model-based approach to attributed graph clustering. In *Proceedings of the 2012 ACM SIGMOD international conference on management of data*, pages 505–516. ACM, 2012.
34. Qianyi Zhan, Jiawei Zhang, Senzhang Wang, S Yu Philip, and Junyuan Xie. Influence maximization across partially aligned heterogenous social networks. In *PAKDD (1)*, pages 58–69, 2015.
35. Han Zhang, Chang-Dong Wang, Jian-Huang Lai, and S Yu Philip. Modularity in complex multilayer networks with multiple aspects: a static perspective. In *Applied Informatics*, volume 4, page 7. Springer Berlin Heidelberg, 2017.