DATAVIZ COVASIM Web Application development

Github Repo: https://github.com/ranga519/COVASIM_Viz
Github Page:https://ranga519.github.io/COVASIM_Viz/

Objective: To create a web front-end for COVASIM agent-based simulator using D3.js JavaScript library for visualizing simulation elements.

The web interface of IDM COVASIM web application uses Plotly python library to visualize simulation graphs(Check the IDM COVASIM description below). There are several reasons for using D3.js over Plotly

- D3.js is a low-level library that gives more control over aspects of the visualization. Since we have specific design requirements and need to create custom visualizations, D3.js is a better choice as it has control over the SVG elements
- D3.js has a large and active community of examples, tutorials, and plugins.
- D3.js can be easily integrated with other JavaScript libraries and frameworks.

Plotting simulation graphs using D3.js with a JSON file:

The COVASIM model provides an inbuilt option for the simulation object(sim) to print the simulation results called sim.to_json().

Example code (python)

```
import covasim as cv
cv.options(jupyter=True, verbose=0)

pars = dict(
        pop_size  = 337,
        start_day = '2020-12-01',
        end_day   = '2020-12-03',
        beta = 5
)
sim = cv.Sim(pars=pars, datafile='covid_county_df_San_Diego.csv')
sim.run()
sim.to_json('results.json')
```

Results dictionary will be in the following form

```
{
   "results": {
     "date": ["2020-12-01",
     "2020-12-02",
     "2020-12-03"],
     "cum_infectious": [340.0,
     351.0,
     359.0]
   }
}
```

We can easily plot the graphs in D3.js using static data provided in the json file. Check the GitHub page above for current visualizations and JavaScript code .

However, it is important that we create more dynamic visualizations, which could show the progression of the simulation in real-time. In order to do that, it is essential that we look into the original COVASIM web application, and understand how it has been designed.

Covasim web app is created using Flask and SciriśWeb. SciriśWeb is an extension of Sciris python library that allows us to build Python webpps and supports vue.js for frontend. Flask is a web application framework that is used to establish the server.

Intializing ScirisWeb app (python)

```
import scirisweb as sw
# Creating an app
app = sw.ScirisApp(__name__, name="Covasim")
flask_app = app.flask_app
```

Once the Sciris flask app is initiated, JavaScript can be used to initiate a VUE instance as shown below. Here, the app object is added to the Vue instance (it is important that we use the same name that was used in creating Sciris app with a #).

Furthermore, responses from the functions that are defined in the python script can be called in the vue instance using sciris.rpc(), as shown below.

Creating VUE instance(javascript)

```
var vm = new Vue({
   el: '#app',

   components: {
      'plotly-chart': PlotlyChart,
   },
async runSim() {
 #Awaiting response from scris app
const response = await sciris.rpc('run_sim', undefined, kwargs);}
})
```

When we define a function in python script we should use the @app.register_RPC() header to call that function through the js file. The following code snippet shows the defining run_sim() function which is called through js front end after starting the simulation.

Defining functions for ScirisWeb app (python)

```
 @app.register_RPC()
def run_sim(sim_pars=None, epi_pars=None, int_pars=None, datafile=None,
show_animation=False, n_days=90, location=None, verbose=True, die=die):
   ''' Create, run, and plot everything '''
   errs = []
   try:
      web_pars = parse_parameters(sim_pars=sim_pars, epi_pars=epi_pars,
int_pars=int_pars, n_days=n_days, location=location, verbose=verbose, errs=errs,
die=die)
      if verbose:
         print('Input parameters:')
         print(web_pars)
   except Exception as E:
      errs.append(log_err('Parameter conversion failed!', E))
      if die: raise
```

IDM COVASIM Web App Description

Covasim web app has several components:

Web UI Components: The Covasim web UI consists of static files index.html and a SciriisWeb/Flask app in cova_app.py.

Local Testing: For local testing, we can run the app via "./launch_flask" or "python cova_app.py."

Deployment: Deployment involves using nginx to serve static files and gunicorn to run the Flask app.

Requirements to launch: Install nginx (sudo apt install nginx) and gunicorn (pip install gunicorn).

Nginx Setup:

Nginx Configuration Example:

```
nginx
server {
    listen 8188; #Change port accrodingly
    server_name localhost;
    location / {
        root /home/my_username/covasim_webapp/covasim_webapp;
    }
    location /api {
        proxy_pass http://127.0.0.1:8097/;
    }
}
```

Gunicorn Setup: Run launch_gunicorn, optionally setting the number of workers (e.g., ./launch_gunicorn --workers=32).

Local Development: For local development, use the --reload flag with the gunicorn command to reload the site automatically.

We can Adjust paths, filenames, and ports based on our requirements.

Cova_app.js file breakdown:

Vue Instance (vm)

- This is the main Vue instance representing the entire application.
- VUE - Vue. js is a reactive front-end framework which allows to declaratively render data to the DOM (Document Object Model)
- Data properties for various aspects of the application state contained in this, including methods for handling user interactions, computed properties for dynamic data, and lifecycle hooks.

PlotlyChart Component(javascript)

```
const PlotlyChart = {
   props: ['graph'],
   render(h) {
      return h('div', {
         attrs: {
            id: this.graph.id,
         }
      });
   },
   mounted() {
      // Code executed after the component is inserted into the DOM
   },
   updated() {
      // Code executed after a data update causes a re-render
   }
};
```

- This is a Vue component for rendering Plotly charts. It takes a prop called 'graph' and renders a div with the specified ID.
- The mounted and updated lifecycle hooks are used to interact with the Plotly library. When the component is mounted or updated, it parses JSON data from the prop, makes it responsive, and updates the Plotly chart.
- This part should be discarded or amended as we indent to draw graphs using d3.js

interventionTableConfig Object

- This object defines configurations for different intervention scenarios related to disease modeling. Each scenario has a form title, tooltip, and fields specifying parameters such as start day, end day, and effectiveness.
- The object contains several scenarios like social distancing, school closures, symptomatic testing, and contact tracing.

Functions (copyright_year, generate_upload_file_handler)

- copyright_year: Returns a string representing the copyright year range.
- generate_upload_file_handler: Returns a function that handles file uploads. It sends the uploaded file to a server via a fetch request and calls the provided callbacks on success or error.

Lifecycle Hooks, Methods, and Computed Properties

- created: Invoked when the Vue instance is created. It initializes various aspects of the application.
- methods: Defines various methods for interacting with the application state, such as adding/deleting interventions, resizing panels, getting version information, running simulations, etc.
- computed: Contains computed properties that derive values based on other properties, like whether the "Run" button should be disabled (isRunDisabled) and whether debugging is enabled (is_debug).

HTML Element

- The Vue instance is attached to an HTML element with the ID 'app,' and it uses various HTML elements and attributes to render the user interface.

<u>notes:</u>

- Button definitions are defined in Cova_app.py not in the js file(line 115 in cova_app.py)
- !Nginx - Web server for the COVASIM website. - Not needed in our case

References:

Kerr, C. C., Stuart, R. M., Mistry, D., Abeysuriya, R. G., Rosenfeld, K., Hart, G. R., Núñez, R. C., Cohen, J. A., Selvaraj, P., Hagedorn, B., George, L., Jastrzębski, M., Izzo, A. S., Fowler, G., Palmer, A., Delport, D., Scott, N., Kelly, S. L., Bennette, C. S., Wagner, B. G., … Klein, D. J. (2021). Covasim: An agent-based model of COVID-19 dynamics and interventions. *PLoS computational biology*, *17*(7), e1009149. https://doi.org/10.1371/journal.pcbi.1009149

InstituteforDiseaseModeling. (n.d.). *Institutefordiseasemodeling/covasim_webapp: Webapp for covasim*. GitHub. https://github.com/InstituteforDiseaseModeling/covasim_webapp

Sciris. (n.d.). *Sciris/scirisweb: Scirisweb python tools*. GitHub. https://github.com/sciris/scirisweb

Learn vue with Evan You. Introduction | Vue.js. (n.d.). https://vuejs.org/guide/introduction.html#:~:text=on%20VueMastery.com-,What%20is%20Vue%3F,be%20they%20simple%20or%20complex.