Installing the program:

Downloadable zip file can be found from: https://github.com/PetKosTuni/Epidemic-model-of-COVID/releases after the release of this program. Depending on errors either use python3 or python.

Windows and PowerShell (Python3 and pip already installed):

git clone git@github.com:PetKosTuni/Epidemic-model-of-COVID.git

(https url: https://github.com/PetKosTuni/Epidemic-model-of-COVID.git)

cd Epidemic-model-of-COVID

pip install virtualenv

python -m venv venv

.\venv\Scripts\activate

cd code

pip install -r requirements.txt

How to operate the program, program flow:

The user has to first move to the code directory: cd code

The command line would look like this …Epidemic-model-of-COVID\code>

The program works by running a command in the command line (examples below) for first validation.py, and after it has finished, for generate_predictions.py.

The format for the command is 'python validation.py/generate_predictions.py (inputs)'. The inputs are first selected from the changeable inputs list: --{input} {wanted input}. **MAKE SURE THAT THE INPUTS ARE VALID.** For example, don't put end date before start date**.** The program will crash otherwise. Data has to exist between start date and mid date.

When generating predictions, the user should only switch the script filename from validation.py to generate_predictions.py keeping the same input arguments. The inputs for both validation.py and generate_predictions.py must be the same.

When running the files, csv-results are generated in directories beginning with *val_results* and *pred_results*. Plots are generated in directories beginning with *figure* in a pdf form.

Example commands for generating predictions:

Example 1:

Create predictions for all states using NYtimes dataset with end date 2021-07-07 and validation end date 2021-07-14.

Step 1: python validation.py --END_DATE 2021-07-07 --VAL_END_DATE 2021-07-14 --dataset NYtimes --level state

Step 2: python generate_predictions.py --END_DATE 2021-07-07 --VAL_END_DATE 2021-07-14 --dataset NYtimes --level state

Example 2:

Create predictions for Orange county in California using custom dataset located in file path 'data/custom_dataset.csv'. Same dates are used as in example 1.

Step 1: python validation.py --END_DATE 2021-07-07 --VAL_END_DATE 2021-07-14 --dataset CUSTOM_DATASET --dataset_filepath 'data/custom_dataset.csv' --level county --state California --county Orange

Step 2: python generate_predictions.py --END_DATE 2021-07-07 --VAL_END_DATE 2021-07-14 --dataset CUSTOM_DATASET --dataset_filepath 'data/custom_dataset.csv' --level county --state California --county Orange

Examples with pictures on page 5.

Changeable inputs:

All dates in format yyyy-mm-dd. The inputs are the same for both files. Values must be given for END_DATE and VAL_END_DATE as other arguments can use default values.

--START_DATE

The date when the learning for the prediction starts. As default the starting date is taken from prediction_data.py file. **Some single county validations dont work without giving an early start date**, since the start date gets set later than the get-function end date (due to the counties not having enough data?).

--MID_DATE

The date when the second learning period starts. Must be at least 3 weeks before END_DATE and must be set by user if START_DATE is set by user. As 'default' the date is taken from prediction_data.py file.

--RESURGE_DATE

A date used when going through US regions. As 'default' the date is taken from prediction_data.py file. Should be between MID_DATE and END_DATE.

--END_DATE

The date when the final learning period ends. Must always be set by the user.

--VAL_END_DATE

The date when the prediction starts. Must always be set by the user. Should also be at least a week after END_DATE

--level

The level on which the program operates (nation/state/county). By default, the parameter is state.

--state

Validation/prediction for one specific state (level should be set as state).

--nation

Validation/prediction for one specific nation (level should be set as nation).

--county

Validation/prediction for one specific county (level should be set as county and county --state should also be specified)

--dataset

The user can choose which dataset to use. default = "NYtimes", if CUSTOM_DATASET is chosen, the user has to also input the filepath of the dataset (csv). Otherwise, if not NYtimes or CUSTOM_dataset. the dataset JHU is chosen.

--dataset_filepath

If a custom dataset has been chosen, by inputting dataset as CUSTOM_DATASET, the filepath of the dataset csv-file has to be put in the form 'data/...'.

--pop_in

The amount of people joining the susceptible population. As default determined by the region between 0.0002 and 0.1.

--bias

The bias of the learning model. As default determined by the region studied between 0.1 and 0.5.

--pred_range

Range for prediction in days (default: 100 days).

If wanted, you can add your own changeable inputs by adding them in to the args-list in start of the two main files (validation/generate_predictions).

Prediction_data.py

The file contains many hardcoded values that can be changed if wanted, such as the number of valid countries that can be included in a custom dataset.

The prediction_data.py contains hard-coded values that cause restrictions / have to be changed if new rules are wanted for datasets! For example, currently 26 countries MUST BE IN THE custom dataset, since the code checks the country names from this file. You probably should add wanted countries to this file if they are included in a custom dataset, or remove them, if they are not included in the dataset. There may be more of these kind of restrictions, so reading the code thoroughly may be beneficial. For example, the US county Lassan does not work with the code for some reason (relic from earlier developers).

Custom datasets:

The datasets should be in csv-form and added to the data/ directory.

If a new dataset is wanted to be implemented as an csv, the The **DATASET_template should be able to create useful training data, if it includes confirmed cases, deaths** (and recovery amounts if wanted, but we don't think these are used anywhere). **The names for these columns may be slightly different in different dataset.csv-files**, so they can be changed in either the csv-files or by changing the hard coded column names in the prediction_data.py-file. This template dataset class should fit the generic use case for a dataset, where the dataset has the following (or similar columns):

```
custom_dataset_columns = ['date', 'country', 'state', 'county', 'cases', 'deaths', 'recoveries']
```

```
date,county,fips,cases,deaths
2020-01-21,Orange,53,1,0
```

The dataset will pick a column and the values corresponding it. HOWEVER, **the template datasets have different restrictions depending on the level used**. For example, if the level is state, and there is county data, there would be multiple of the same state for one date. This would crash the program (perhaps because the array is simply too big in relation to some expected value?). In short, **with the DATASET_template, for state and nation levels, use one date per state/nation. County data will also require an additional state column which to match the county to.** It would also be wise to separate county/state/nation data in to their own csv-files, even though the template tries to check the correct level from the file. If using a single region to validate, these restrictions may not be so strict.

Try to also avoid false names for regions, since the populations are collected from certain csv-files, which have hard coded state, county and nation population data.

You can also add your own datasets as their own class with their own functions in the data.py, by using the other datasets as example, if the DATASET_template is not working satisfactorily, or you want to add other than csv-files with other kinds of columns. You also would have to add the datasets to the validation/generate_predictions-files as options in the data = ... rows (like where DATASET_template is used):

```
if args.dataset == "NYtimes":
    data = NYTimes(level='counties')
elif args.dataset == "CUSTOM_DATASET":
    data = DATASET_template(args.dataset_filepath, pdata.custom_dataset_columns, level = 'counties')
else:
    data = JHU_US(level='counties')
```

Examples with pictures:

Example 3:

Create predictions for Argentina (using JHU dataset as default) with end date 2021-07-07 and validation end date 2021-07-14. Prediction range is set as 300 days.

Running the following commands results in the graph below.

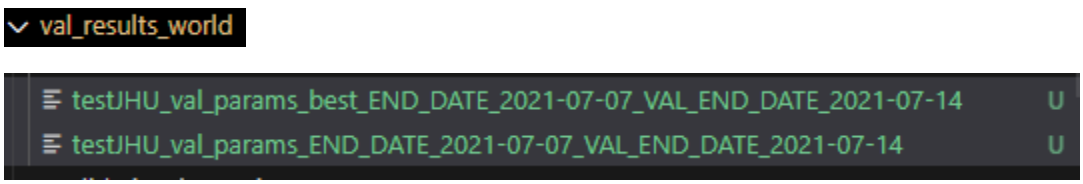
Step 1: cd code

```
PS T:\BACKUP\Koulu\Epidemic-model-of-COVID> cd code
PS T:\BACKUP\Koulu\Epidemic-model-of-COVID\code>
```

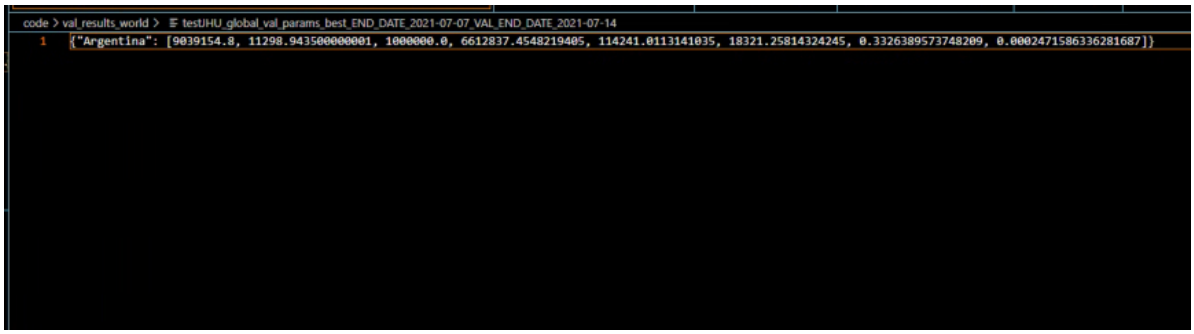
Step 2: python validation.py --START_DATE 2020-04-06 --MID_DATE 2021-05-07 --END_DATE 2021-07-07 --VAL_END_DATE 2021-07-14 --level nation --nation Argentina --pred_range 300

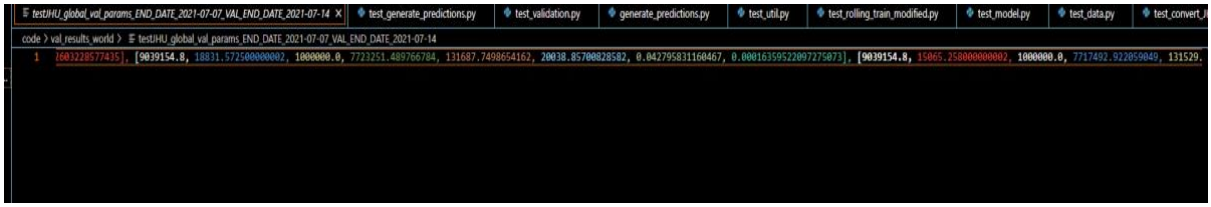
```
PS T:\BACKUP\Koulu\Epidemic-model-of-COVID\code> python validation.py --START_DATE
 2020-04-06 --MID_DATE 2021-05-07  --END_DATE 2021-07-07 --VAL_END_DATE 2021-07-14
  --level nation --nation Argentina --pred_range 300
```

Validation results are saved in code/val_results_world directory.



Validation result files are shown below.



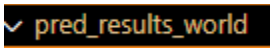


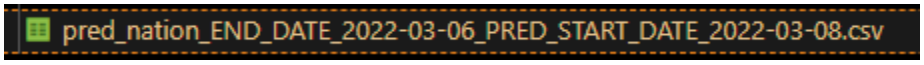
Step 3: python generate_predictions.py --START_DATE 2020-04-06 --MID_DATE 2021-05-07 --END_DATE 2021-07-07 --VAL_END_DATE 2021-07-14 --level nation --nation Argentina --pred_range 300

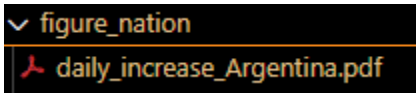
```
PS T:\BACKUP\Koulu\Epidemic-model-of-COVID\code> python generate_predictions.py
 --START_DATE 2020-04-06 --MID_DATE 2021-05-07  --END_DATE 2021-07-07 --VAL_END
_DATE 2021-07-14  --level nation --nation Argentina --pred_range 300
```

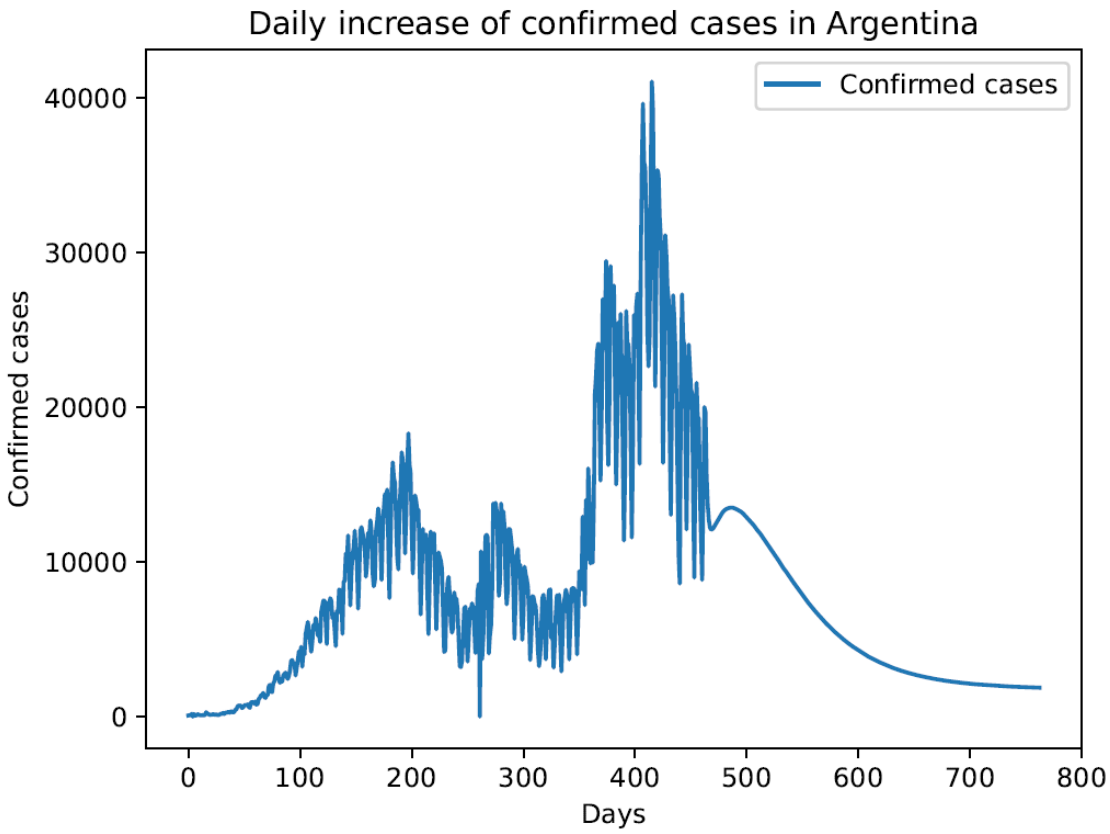
Prediction results are saved in code/pred_results_world directory.

The plots generated are saved in code/figure_nation directory.



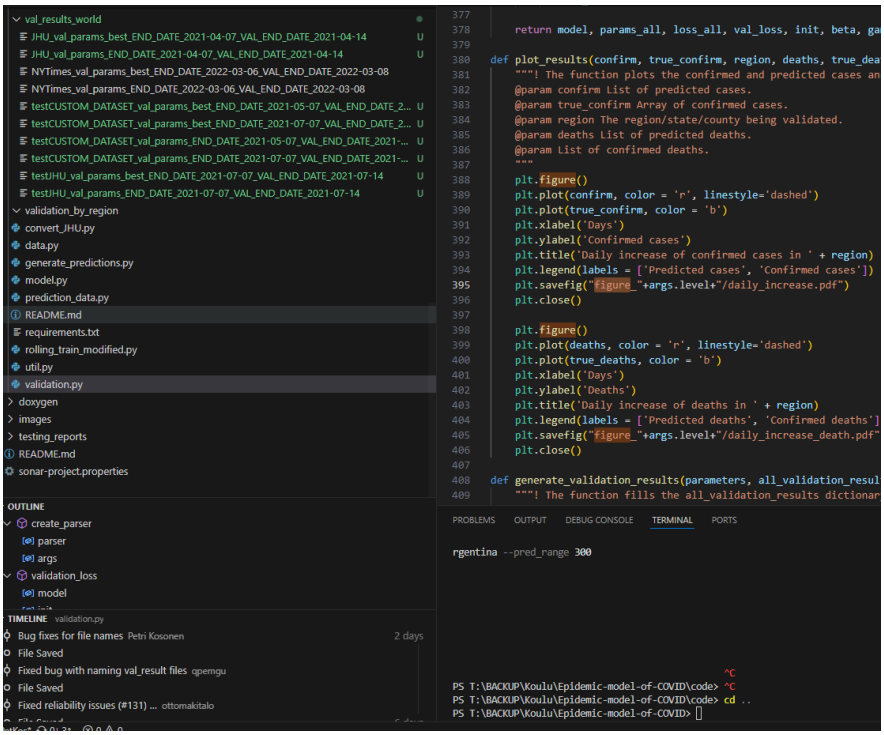






Example 4:

This example will be thorough. First, we start in our terminal of choice once the program has been installed with the instructions from the start of the file. In Visual Studio Code the view should look like this after opening the program folder. The terminal is in the lower right corner.



First we need to move to the correct directory, code.

```
PS T:\BACKUP\Koulu\Epidemic-model-of-COVID> cd code
PS T:\BACKUP\Koulu\Epidemic-model-of-COVID\code>
```

Before we can get the prediction results for the date range and region, we need to generate validation files. We have to decide the dataset, the dates for which the validation occurs, and the level of validation (county/state/nation). For this case we choose California, US on a state level with the END_DATE 2021-07-07 and VAL_END_DATE 2021-07-14. Rest of the inputs will be default values. The dataset will default to NYtimes. This will be a default run of the program, with only the mandatory inputs. If using the CUSTOM_dataset, remember to add the filepath and remember the restrictions written earlier.

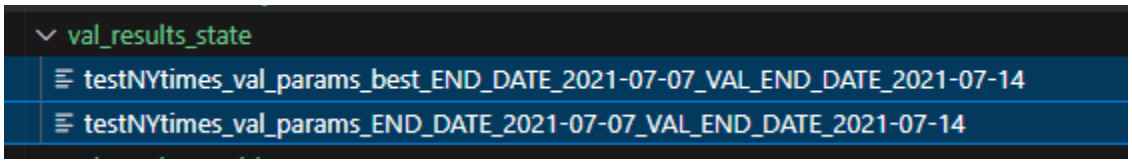
The command to generate the wanted validation files would look like this (single line will do):

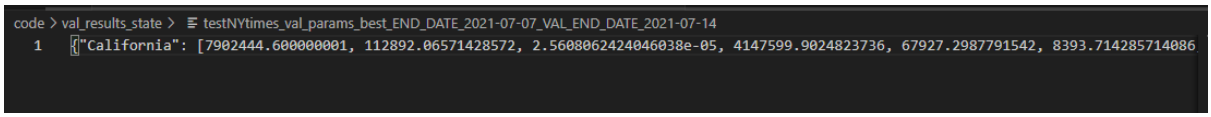
```
PS T:\BACKUP\Koulu\Epidemic-model-of-COVID\code> python validation.py
--END_DATE 2021-07-07 --VAL_END_DATE 2021-07-14 --state California
```

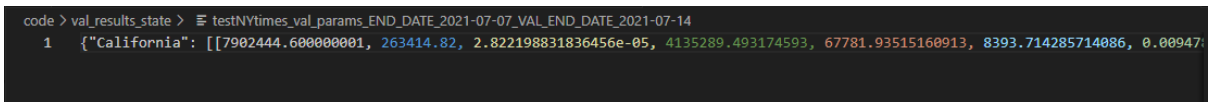
The terminal will start printing information. Once the total confirmed cases and deaths are printed, validation files have been created.

```
Total confirmed cases:  4147599.9024823736  Total deaths:  67927.2987791542
PS T:\BACKUP\Koulu\Epidemic-model-of-COVID\code>
```

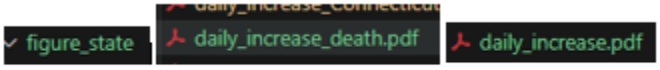
Since our validation level was at state level, the validation files were written into /val_results_state, in csv-form:

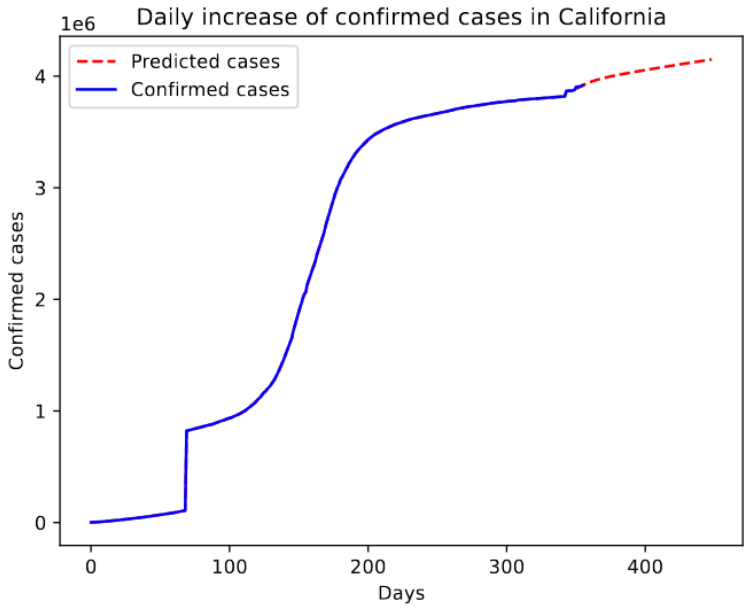


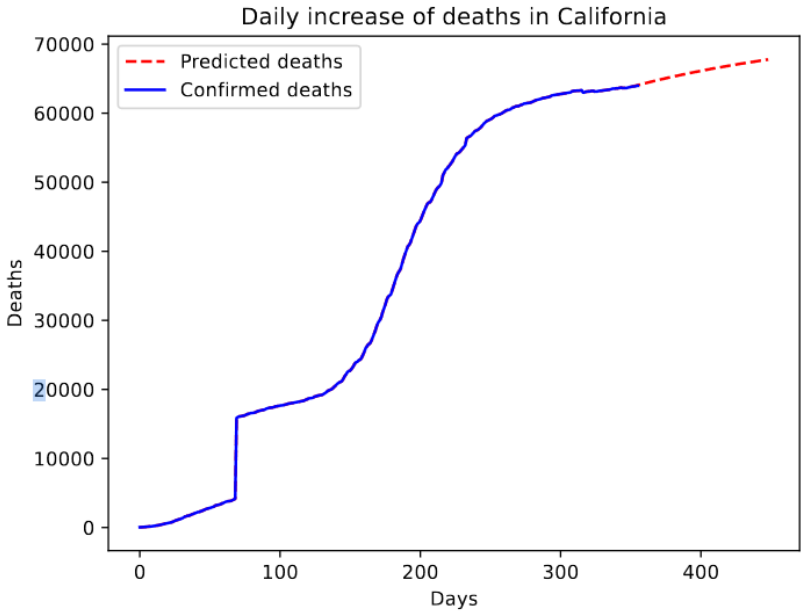




The validation process will also plot the latest region validated in the correct /figure_... folder, in our case in /figure_state. The plots are always called daily_increase_death.pdf and daily_increase.pdf:







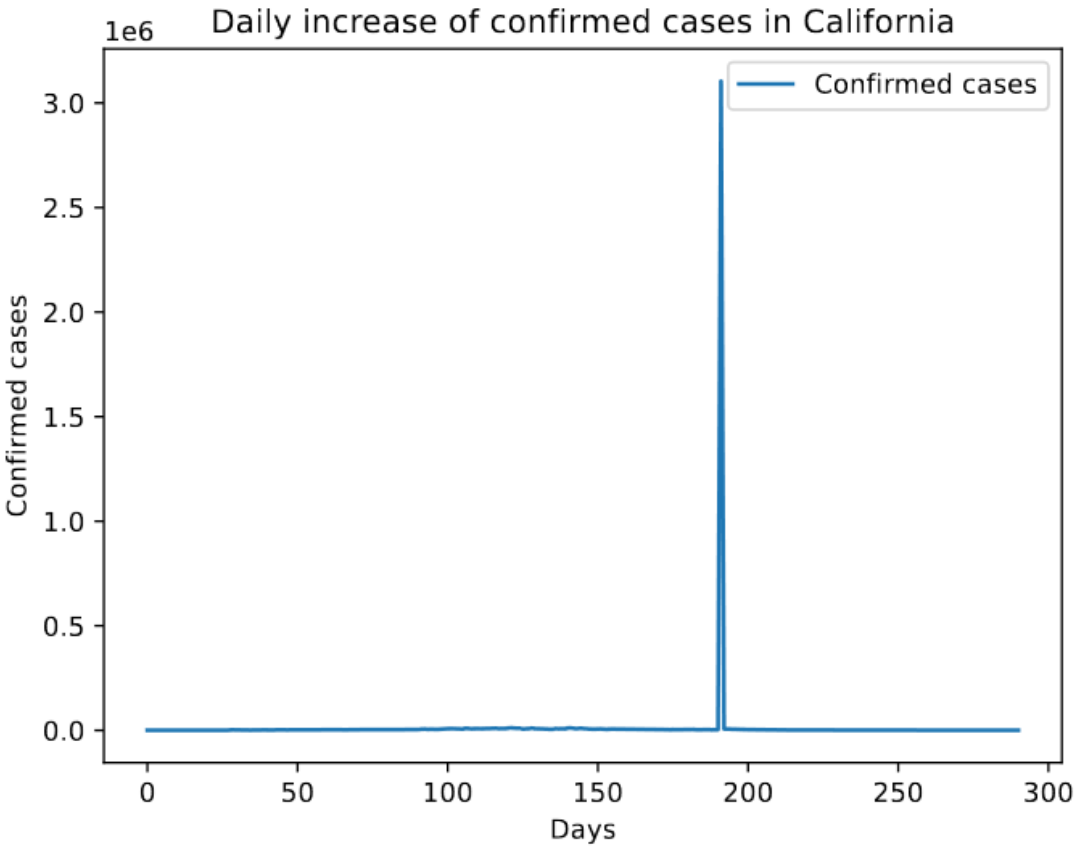
Now we can get the wanted prediction results. The command should be the exact same, expect for the file name. The command to generate the wanted prediction files looks like this (single line will do):

```
PS T:\BACKUP\Koulu\Epidemic-model-of-COVID\code> python generate_predictions.py
 --END_DATE 2021-07-07 --VAL_END_DATE 2021-07-14 --state California
```

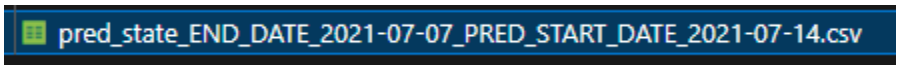
Once the terminal has printed the following information

```
region:  California  training loss:  [0.009866025960771942, 0.00079612604486260
47, 0.033021082650520145] [0.006372791178703813, 0.02882654860777848]  maximum
death cases:  69203  maximum confirmed cases:  4056809
```

the prediction results are generated. The program has plotted the Californian predictions in /figure_state, with the name daily_increase_California.pdf. The plot has a spike (see notes section):



The actual results are saved in csv-form in > pred_results_state :



Notes:

rolling_train_modified.py has some unreachable code in the loss train function for unknown reason. You can try fixing the code if it makes sense and makes better predictions.

validation.py and generation_predictions.py have resurge_date if-statements, that cause spikes in the graph results. The if statements and the resurge dates could just be removed to fix this problem. Currently this problem only exists for US counties/states.

If removed files or functionalities are required, the old code from the original project is still available.

The tutorial and video demo will be included in the handoff email and the root path of the repository. The /code-folder will include another read-me that also includes some of the information from the tutorial.

The test folder includes a readme to run the unit tests.