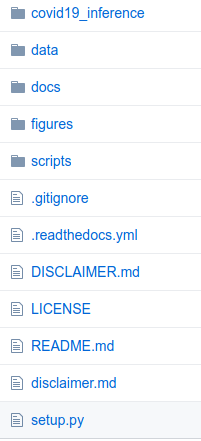
**A simple model to get started**

I take notes from the code released by the Priesemann group as [a Github project](https://github.com/Priesemann-Group/covid19_inference_forecast).

There, we can have the repository of the entire work:



It can be used by anyone following 3 alternative ways:

1/ “Clone the repository”

Go to the Github repository of the project and click on:



2/ “Install the module via pip” (e.g. in your laptop by using the terminal)



3/ “Run the notebooks directly in Google Colab”

How to copy and edit the module? Via the files of the repository?

We can start with one folder which contains a notebook and the data files.

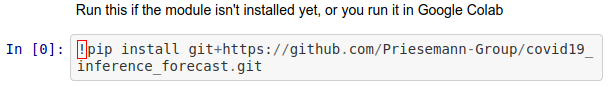
1. Fitting with the SIR model

The notebook file “[example\_script\_covid19\_inference.ipynb](https://github.com/Priesemann-Group/covid19_inference_forecast/blob/master/scripts/example_script_covid19_inference.ipynb) creates the SIR model with 3 changing points from the set of parameters {λi, ti, μ, D, σ,I0}, (i = 1,2,3).

“The layout and design of the plots has been simplified in order to make the code more readable and more easily adaptable to other problems.”

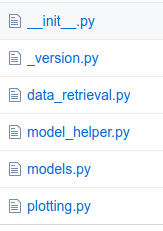
Structure of the file:

0/ Install the module (like the instruction above)



00/

The example code imports the folder “[covid19\_inference](https://github.com/Priesemann-Group/covid19_inference_forecast/tree/master/covid19_inference)” which contains



1/ Fit model

It estimates the set of parameters {λi, tj, μ, D, σ,I0}, (i = 0 to 3; j = 0 to 2) plus the durations over which the changes (labeled by j) are taking place, called transients at tj by using: a/ prior knowledge on tj,

1.1/ Parameters:

* **new\_cases\_obs** ([*list*](https://docs.python.org/3/library/stdtypes.html#list) *or array*) – Timeseries (day over day) of newly reported cases (not the total number)
* **change\_points\_list** (*list of dicts*) –  
   List of dictionaries, each corresponding to one change point.  
    
   Each dict can have the following key-value pairs. If a pair is not provided, the respective default is used.  
  + pr\_mean\_date\_begin\_transient : datetime.datetime, NO default
  + pr\_median\_lambda : number, same as default priors, below
  + pr\_sigma\_lambda : number, same as default priors, below
  + pr\_sigma\_date\_begin\_transient : number, 3
  + pr\_median\_transient\_len : number, 3
  + pr\_sigma\_transient\_len : number, 0.3
* **date\_begin\_simulation** ([*datetime.datetime*](https://docs.python.org/3/library/datetime.html#datetime.datetime)) – The begin of the simulation data
* **num\_days\_sim** (*integer*) – Number of days to forecast into the future
* **diff\_data\_sim** (*integer*) – Number of days that the simulation-begin predates the first data point in new\_cases\_obs. This is necessary so the model can fit the reporting delay. Set this parameter to a value larger than what you expect to find for the reporting delay.
* **N** (*number*) – The population size. For Germany, we used 83e6
* **priors\_dict** ([*dict*](https://docs.python.org/3/library/stdtypes.html#dict)) –  
   Dictionary of the prior assumptions  
    
   Possible key-value pairs (and default values) are:
  + pr\_beta\_I\_begin : number, default = 100
  + pr\_median\_lambda\_0 : number, default = 0.4
  + pr\_sigma\_lambda\_0 : number, default = 0.5
  + pr\_median\_mu : number, default = 1/8
  + pr\_sigma\_mu : number, default = 0.2
  + pr\_median\_delay : number, default = 8
  + pr\_sigma\_delay : number, default = 0.2
  + pr\_beta\_sigma\_obs : number, default = 10
  + week\_end\_days : tuple, default = (6,7)
  + pr\_mean\_weekend\_factor : number, default = 0.7
  + pr\_sigma\_weekend\_factor :number, default = 0.17
* **weekends\_modulated** ([*bool*](https://docs.python.org/3/library/functions.html#bool)) – Whether to add the prior that cases are less reported on week ends. Multiplies the new cases numbers on weekends by a number between 0 and 1, given by a prior beta distribution. The beta distribution is parametrised by pr\_mean\_weekend\_factor and pr\_sigma\_weekend\_factor
* **weekend\_modulation\_type** (*'step' or 'abs\_sine':*) – whether the weekends are modulated by a step function, which only multiplies the days given by week\_end\_days by the week\_end\_factor, or whether the whole week is modulated by an abs(sin(x)) function, with an offset with flat prior.

1.2/ Returns

2/ Plot

1. Another SIR model with change points

The notebook file “SIR\_with\_delay\_Germany\_3scenarios.ipynb”

General structure of a code:

Import

(try sometimes?)

Stats

Beta, mean, sigma,

Python: object, dict, list, …,

user-defined function

def \_SIR\_model(

S\_begin, I\_begin

pm.Model() [import pymc3 as pm]

I\_begin = pm.HalfCauchy(

pm.Lognormal(