arcovid19

Release 0.6b

Jun 01, 2020

Contents:

Libs is a package to use the COV-19 data analysis tool. It has a module written in python called arcovid19 dedicated to the analysis of all real cases of COVID-19 in Argentina.

Arcovid19 is made up of two classes CasesFrame and CasesPlot, these tools are used to report the analysis of COVID-19 data in Argentina from tables and visualizations. Furthermore, the CasesFrame class inherits and adds functionalities to the DataFrame class of Pandas.

CHAPTER 1

License

arcovid19 is under The BSD-3 License

The BSD 3-clause license allows you almost unlimited freedom with the software so long as you include the BSD copyright and license notice in it (found in Fulltext).

CHAPTER 2

Citation

Please acknowledge arcovid19 in any research report or publication that requires citation of any author's work. Our suggested acknowledgment is:

> The authors acknowledge the arcovid19 project that contributed to the research reported here. <https://github.com/ivco19/libs/>

ABOUT THE DATA

Please cite:

> Luczywo, N. A., Daza, V., Koraj, M., Dominguez, M., Lares, M., Paz, D. J., Quiroga, R., Rios, M. E. D. L., Sánchez, B. O., Stasyszyn, F., & Cabral, J. B. (2020). Infecciones de COVID-19 en Argentina. Unpublished. https://doi.org/10.13140/RG.2.2.22519.78246

BibText:

```
@misc{https://doi.org/10.13140/rg.2.2.22519.78246,
    doi = {10.13140/RG.2.2.22519.78246},
    url = {http://rgdoi.net/10.13140/RG.2.2.22519.78246},
    author = {
        Luczywo,  Nadia Ayelen and Daza,  Vanessa and Koraj,  Mauricio and
        Dominguez,  Mariano and Lares,  Marcelo and Paz,  Dante Javier and
        Quiroga,  Rodrigo and Rios,  Martín Emilio De Los and
        Sánchez,  Bruno Orlando and Stasyszyn,  Federico and
        Cabral,  Juan Bautista},
    language = {es},
    title = {Infecciones de COVID-19 en Argentina},
    publisher = {Unpublished},
    year = {2020}
}
```

CHAPTER 3

Raw Data

- Viewer
- CSV
- XLSX

CHAPTER 4

Contact:

For bugs or question please contact

- **Juan B. Cabral:** jbcabral@unc.edu.ar
- **Bruno Sánchez:** bruno.sanchez@duke.edu
- **Vanessa Daza:** vanessa.daza@unc.edu.ar

CHAPTER 5

Contents:

5.1 Installation

This is the recommended way to install arcovid-libs.

5.1.1 Installing with pip

Make sure that the Python interpreter can load arcovid-libs code. The most convenient way to do this is to use virtualenv, virtualenvwrapper, and pip.

After setting up and activating the virtualenv, run the following command:

```
$ pip install arcovid19
```

That should be it all.

5.1.2 Installing the development version

If you'd like to be able to update your corral code occasionally with the latest bug fixes and improvements, follow these instructions:

Make sure that you have Git installed and that you can run its commands from a shell. (Enter *git help* at a shell prompt to test this.)

Check out arcovid-libs main development branch like so:

```
$ git clone https://github.com/ivco19/libs.git
```

This will create a directory libs in your current directory.

5.2 Command line interface (CLI)

After install arcovid19 you also have a command line app to download the data as csv.

```
$ arcovid19 --help
Usage: arcovid19 [OPTIONS]

Retrieve and store the database as an as CSV file.

Options:
--url=STR    str The url for the excel table to parse. Default is ivco19 team table.␣
↪(default: https://github.com/ivco19/libs/raw/master/databases/cases.xlsx)
--out=STR    PATH (default=stdout) The output path to the CSV file. If it's not␣
↪provided the data is printed in the stdout.
--nocached   If you want to ignore the local cache or retrieve a new value.

Other actions:
-h, --help   Show the help
```

5.3 Cases Tutorial

This section contains a step-by-step tutorial with examples for using the **arcovid19** tools.

In part 1 of the tutorial, we explained loading data using arcovid19 and some of the methods and attributes inherited from the DataFrame class. Part 2 shows the implementation of the *Cases Frame* class methods on the data basis. Lastly, in part 3 of this tutorial, we share the visualization of the COVID-19 database using the tools provided by the *Plot Cases* class from arcovid19.

5.3.1 Part 1 - Load cases

The first step is to import **arcovid19**, then we load the COVID-19 database from Argentina (CASES_URL). As a result, a table is obtained that presents a multiple pandas index, with the following hierarchy:

level 0: cod_provincia - Argentina states

level 1: cod_status - Four states of disease patients (**R** = Recovered, **C** = Confirmed, **A** = Active, **D** = Dead)

[1]:
```
import arcovid19 as ac
```

[2]:
```
cases = ac.load_cases()
```

[3]:
```
cases.head(4)
```

[3]:
```
                          provincia_status              Pcia_status  \
cod_provincia cod_status
CABA          C                    CABA_C  CABA Casos Confirmados
              R                    CABA_R  CABA Casos Recuperados
              D                    CABA_D      CABA Casos Muertos
              A                    CABA_A       CABA Casos Activos

                           2020-03-03 00:00:00  2020-03-04 00:00:00  \
cod_provincia cod_status
CABA          C                            1.0                  1.0
              R                            0.0                  0.0
```

(continues on next page)

(continued from previous page)

```
              D                          0.0                  0.0
              A                          1.0                  1.0

                           2020-03-05 00:00:00  2020-03-06 00:00:00  \
cod_provincia cod_status
CABA          C                          1.0                  5.0
              R                          0.0                  0.0
              D                          0.0                  0.0
              A                          1.0                  5.0

                           2020-03-07 00:00:00  2020-03-08 00:00:00  \
cod_provincia cod_status
CABA          C                          6.0                  8.0
              R                          0.0                  0.0
              D                          1.0                  1.0
              A                          5.0                  7.0

                           2020-03-09 00:00:00  2020-03-10 00:00:00  ...  \
cod_provincia cod_status                                             ...
CABA          C                          9.0                 10.0  ...
              R                          0.0                  0.0  ...
              D                          1.0                  1.0  ...
              A                          8.0                  9.0  ...

                           2020-05-19 00:00:00  2020-05-20 00:00:00  \
cod_provincia cod_status
CABA          C                       3566.0               3823.0
              R                       1139.0               1139.0
              D                        135.0                137.0
              A                       2292.0               2547.0

                           2020-05-21 00:00:00  2020-05-22 00:00:00  \
cod_provincia cod_status
CABA          C                       4202.0               4606.0
              R                       1139.0               1139.0
              D                        139.0                146.0
              A                       2924.0               3321.0

                           2020-05-23 00:00:00  2020-05-24 00:00:00  \
cod_provincia cod_status
CABA          C                       5006.0               5500.0
              R                       1139.0               1678.0
              D                        152.0                155.0
              A                       3715.0               3667.0

                           2020-05-25 00:00:00  2020-05-26 00:00:00  \
cod_provincia cod_status
CABA          C                       5875.0               6202.0
              R                       1678.0               1678.0
              D                        163.0                168.0
              A                       4034.0               4356.0

                           2020-05-27 00:00:00  2020-05-28 00:00:00
cod_provincia cod_status
CABA          C                       6564.0               6989.0
              R                       1678.0               1963.0
              D                        172.0                175.0
```

(continues on next page)

(continued from previous page)

```
              A                          4714.0               4851.0

[4 rows x 89 columns]
```

[4]:
```
cases.shape
```

[4]:
```
(101, 89)
```

[5]:
```
cases.describe()
```

[5]:
```
       2020-03-03  2020-03-04  2020-03-05  2020-03-06  2020-03-07  2020-03-08  \
count  100.000000  101.000000  101.000000  101.000000  101.000000  101.000000
mean     0.040000    0.039604    0.089109    0.346535    0.357673    0.478548
std      0.196946    0.196000    0.349257    1.359674    1.432113    1.941601
min      0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
25%      0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
50%      0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
75%      0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
max      1.000000    1.000000    2.000000    8.000000    9.000000   12.000000

       2020-03-09  2020-03-10  2020-03-11  2020-03-12  ...   2020-05-19  \
count  101.000000  101.000000  101.000000  101.000000  ...   101.000000
mean     0.677393    0.753640    0.832725    1.232438  ...   348.871805
std      2.603514    2.926856    3.252652    4.689155  ...  1176.302303
min      0.000000    0.000000    0.000000    0.000000  ...     0.000000
25%      0.000000    0.000000    0.000000    0.000000  ...     2.000000
50%      0.000000    0.000000    0.000000    0.000000  ...    14.000000
75%      0.000000    0.000000    0.000000    0.000000  ...    89.000000
max     17.000000   19.000000   21.000000   31.000000  ...  8809.000000

        2020-05-20   2020-05-21    2020-05-22    2020-05-23    2020-05-24  \
count   101.000000   101.000000   101.000000    101.000000    101.000000
mean    367.644097   393.307622   421.743290    449.624417    478.258056
std    1248.414401  1348.455749  1460.412128   1572.673190   1664.493107
min       0.000000     0.000000     0.000000      0.000000      0.000000
25%       2.000000     2.000000     2.000000      2.000000      2.000000
50%      14.000000    15.000000    16.000000     16.000000     16.000000
75%      89.000000    90.000000    90.000000     90.000000     90.000000
max    9283.000000  9931.000000  10649.000000  11353.000000  12076.000000

         2020-05-25    2020-05-26    2020-05-27    2020-05-28
count    101.000000    101.000000    101.000000    101.000000
mean     500.119264    523.881659    551.802508    582.257972
std     1752.146024   1823.126460   1933.166375   2034.023400
min        0.000000      0.000000      0.000000      0.000000
25%        2.000000      2.000000      3.000000      3.000000
50%       17.000000     17.000000     17.000000     14.000000
75%       90.000000     90.000000     89.000000    110.000000
max    12628.000000  13228.000000  13933.000000  14702.000000

[8 rows x 87 columns]
```

Remember that the *cases* object has the properties of a DataFrame.

5.3.2 Part 2 - Functions

After loading the database we can use some of the functions of the class object *Cases Frame*.

[1]:
```
import arcovid19 as ac
```

[2]:
```
cases = ac.load_cases()
```

Using the method *last_growth_rate* and specifying the *province* argument of the function, we obtain the latest growth rate available for the whole country or by province.

If provincia = None, the method will yield the value of Argentina's last growth rate.

[3]:
```
cases.last_growth_rate(provincia=None)
```

[3]:
```
0.05519270795952047
```

If you want to get the latest growth rate for any particular province, you must replace None for the full name of the province.

[4]:
```
cases.last_growth_rate(provincia='córdoba')
```

```
/home/docs/checkouts/readthedocs.org/user_builds/arcovid19/envs/latest/lib/python3.7/
↪site-packages/arcovid19-0.6b0-py3.7.egg/arcovid19/cases.py:731: RuntimeWarning:␣
↪divide by zero encountered in true_divide
  growth_rate = np.array((I_n / I_n_1) - 1)
/home/docs/checkouts/readthedocs.org/user_builds/arcovid19/envs/latest/lib/python3.7/
↪site-packages/arcovid19-0.6b0-py3.7.egg/arcovid19/cases.py:731: RuntimeWarning:␣
↪invalid value encountered in true_divide
  growth_rate = np.array((I_n / I_n_1) - 1)
```

[4]:
```
-0.002178649237472796
```

In the case that you want to estimate the growth rate in the period where we have data by province or country, you can use the *grateful_full_period* method.

Again, if the province argument equals None, the series shown corresponds to Argentina.

[5]:
```
cases.grate_full_period(provincia=None).head(5)
```

[5]:
```
2020-03-04           0
2020-03-05           1
2020-03-06           3
2020-03-07       0.125
2020-03-08    0.333333
Name: (ARG, growth_rate_C), dtype: object
```

[6]:
```
cases.grate_full_period(provincia='córdoba').head(5)
```

[6]:
```
2020-03-04    NaN
2020-03-05    NaN
2020-03-06    NaN
2020-03-07    0.0
2020-03-08    0.0
dtype: float64
```

In the case that you want the observations by date, that is, not cumulative, you can use the *restore_time_serie* method as shown in the next cell.

[7]:
```
cases.restore_time_serie().head(4)
```

[7]:
```
                         provincia_status              Pcia_status  \
cod_provincia cod_status
CABA          C                   CABA_C  CABA Casos Confirmados
              R                   CABA_R  CABA Casos Recuperados
              D                   CABA_D      CABA Casos Muertos
              A                   CABA_A       CABA Casos Activos

                         2020-03-03 00:00:00  2020-03-04 00:00:00  \
cod_provincia cod_status
CABA          C                          1.0                  0.0
              R                          0.0                  0.0
              D                          0.0                  0.0
              A                          1.0                  0.0

                         2020-03-05 00:00:00  2020-03-06 00:00:00  \
cod_provincia cod_status
CABA          C                          0.0                  4.0
              R                          0.0                  0.0
              D                          0.0                  0.0
              A                          0.0                  4.0

                         2020-03-07 00:00:00  2020-03-08 00:00:00  \
cod_provincia cod_status
CABA          C                          1.0                  2.0
              R                          0.0                  0.0
              D                          1.0                  0.0
              A                          0.0                  2.0

                         2020-03-09 00:00:00  2020-03-10 00:00:00  ...  \
cod_provincia cod_status                                             ...
CABA          C                          1.0                  1.0  ...
              R                          0.0                  0.0  ...
              D                          0.0                  0.0  ...
              A                          1.0                  1.0  ...

                         2020-05-19 00:00:00  2020-05-20 00:00:00  \
cod_provincia cod_status
CABA          C                        224.0                257.0
              R                        262.0                  0.0
              D                          5.0                  2.0
              A                        -43.0                255.0

                         2020-05-21 00:00:00  2020-05-22 00:00:00  \
cod_provincia cod_status
CABA          C                        379.0                404.0
              R                          0.0                  0.0
              D                          2.0                  7.0
              A                        377.0                397.0

                         2020-05-23 00:00:00  2020-05-24 00:00:00  \
cod_provincia cod_status
CABA          C                        400.0                494.0
              R                          0.0                539.0
              D                          6.0                  3.0
              A                        394.0                -48.0
```

(continues on next page)

(continued from previous page)

```
                          2020-05-25 00:00:00  2020-05-26 00:00:00  \
cod_provincia cod_status
CABA          C                         375.0                327.0
              R                           0.0                  0.0
              D                           8.0                  5.0
              A                         367.0                322.0

                          2020-05-27 00:00:00  2020-05-28 00:00:00
cod_provincia cod_status
CABA          C                         362.0                425.0
              R                           0.0                285.0
              D                           4.0                  3.0
              A                         358.0                137.0

[4 rows x 89 columns]
```

5.3.3 Part 3 - Visualization of cases

In this part of the tutorial you can find some of the functions of the * Cases Plot * class, in particular this class inherits the properties of the Matplotlib class.

As usual the first step will be to import the arcovid19 module.

[1]:
```
import arcovid19 as ac
import matplotlib.pyplot as plt
```

The next cell performs a cache cleanup, as arcovid19 saves a one-hour chache for every request made to the online database. With this action, we make sure that we are using an updated database.

[2]:
```
ac.CACHE.clear()
```

[2]:
```
2
```

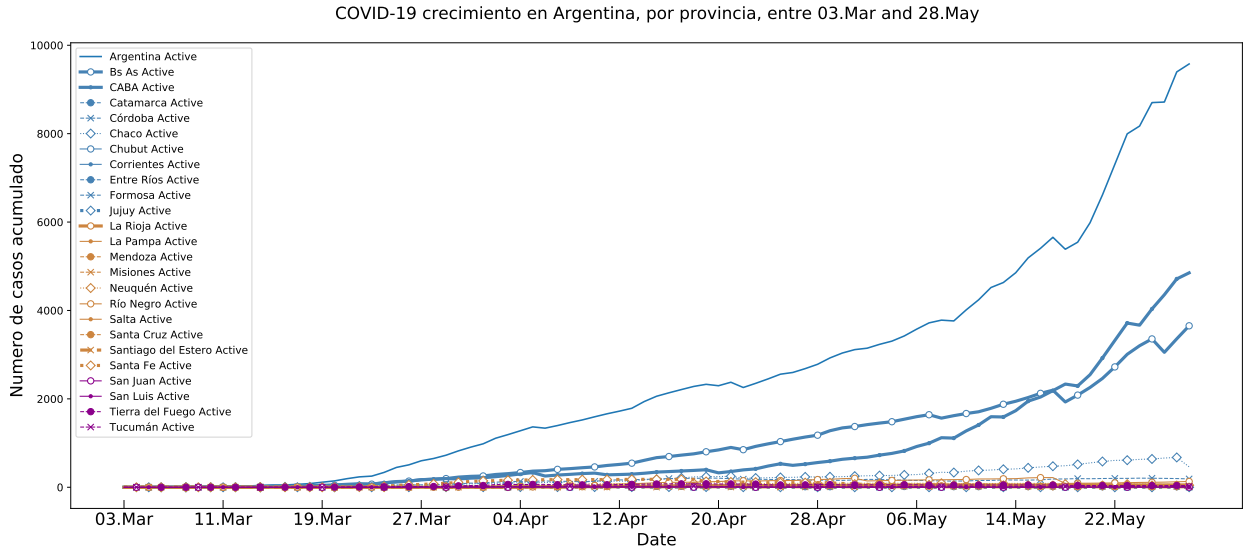
We loaded the COVID-19 database from Argentina.

[3]:
```
cases = ac.load_cases()
```

The *plot* method shows the trend to date of each of the provinces along with that of Argentina. By default, it shows confirmed cases, but you can also choose to view active, recovered and dead cases.

[4]:
```
fig, ax = plt.subplots(figsize=(20, 8))
cases.plot(ax=ax, confirmed=False, active=True, recovered=False, deceased=False)
ax.legend(loc=2)
```

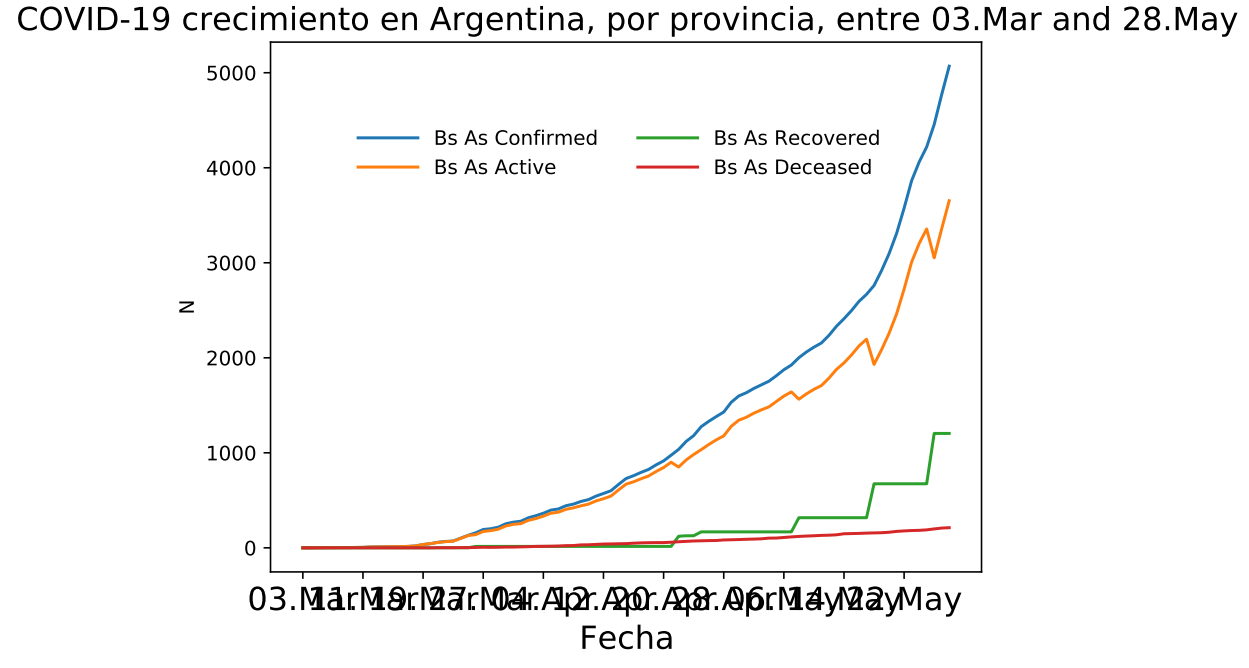
[4]:
```
<matplotlib.legend.Legend at 0x7f611f3a0080>
```



Then if you are interested in a particular province, you just have to add *grate_full_periode* and specify the name of the province. If you do not enter the name of any province the tool will show you the cases of Argentina.

[5]:
```
ax = cases.plot.grate_full_period('Bs As', confirmed=True, active=True,␣
↪recovered=True, deceased=True)
```

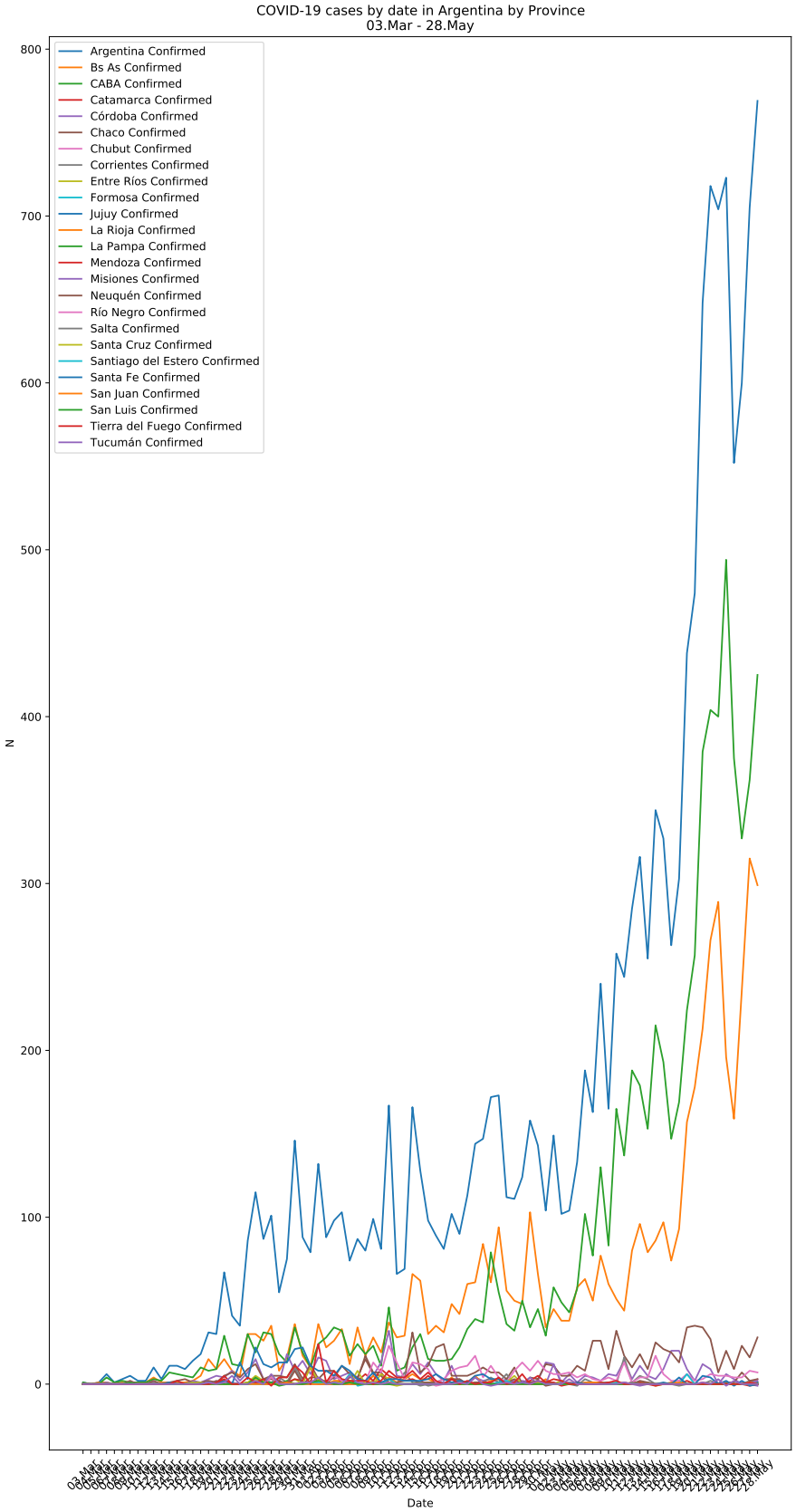
```
/home/docs/checkouts/readthedocs.org/user_builds/arcovid19/envs/latest/lib/python3.7/
↪site-packages/ipykernel_launcher.py:1: DeprecationWarning: Call to deprecated␣
↪method grate_full_period. (use growth_provincia instead) -- Deprecated since␣
↪version 0.5.
  """Entry point for launching an IPython kernel.
```



Also, if you want to observe the behavior of observations by date, you can use the following graphical representations.

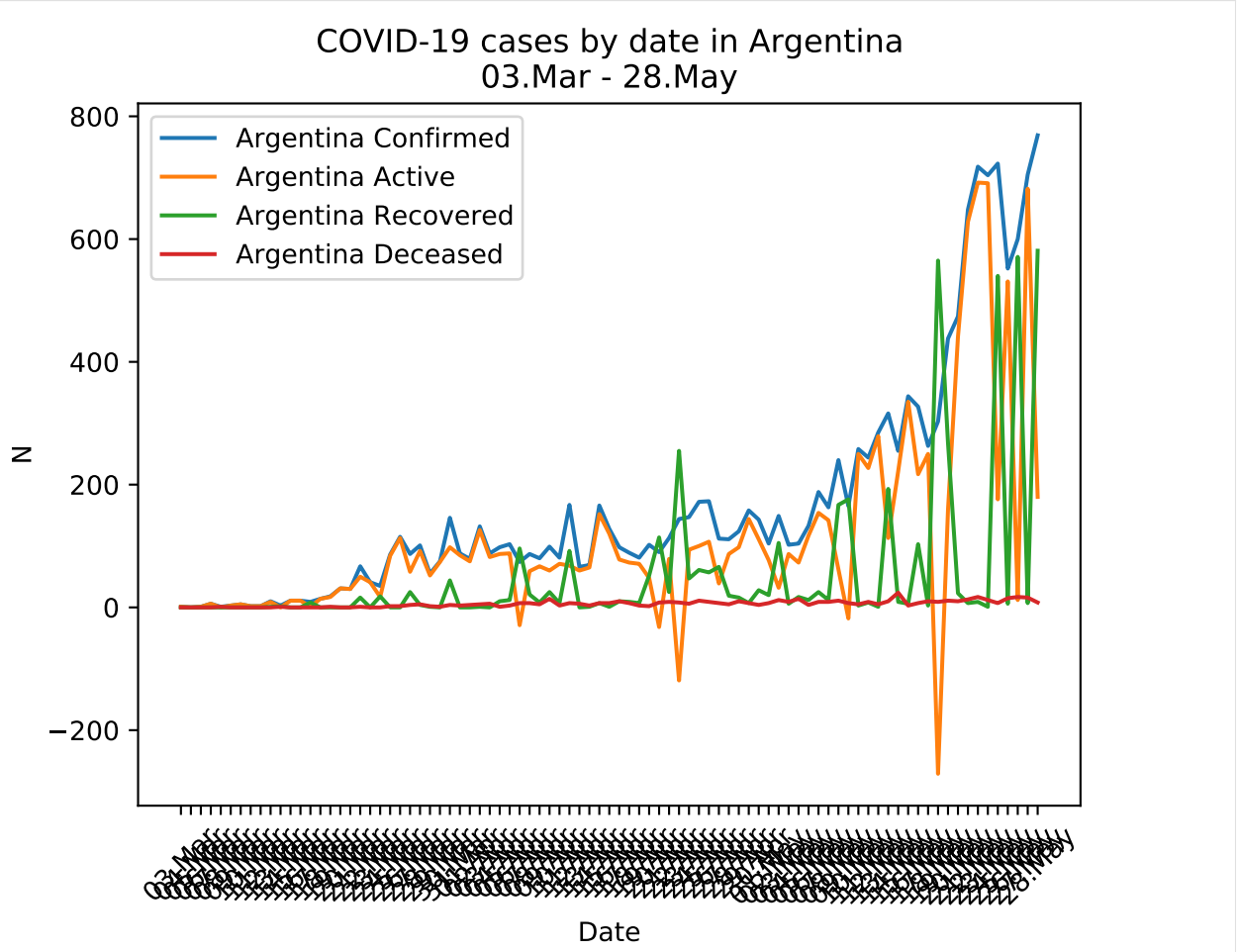
[6]:
```
cases.plot.time_serie_all()
```

[6]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f611f5475c0>
```


COVID-19 cases by date in Argentina by Province
03.Mar - 28.May
Argentina Confirmed
Bs As Confirmed
CABA Confirmed
Catamarca Confirmed
Córdoba Confirmed
Chaco Confirmed
Chubut Confirmed
Corrientes Confirmed
Entre Ríos Confirmed
Formosa Confirmed
Jujuy Confirmed
La Rioja Confirmed
La Pampa Confirmed
Mendoza Confirmed
Misiones Confirmed
Neuquén Confirmed
Río Negro Confirmed
Salta Confirmed
Santa Cruz Confirmed
Santiago del Estero Confirmed
Santa Fe Confirmed
San Juan Confirmed
San Luis Confirmed
Tierra del Fuego Confirmed
Tucumán Confirmed
800
700
600
500
400
300
200
100
0
N
Date

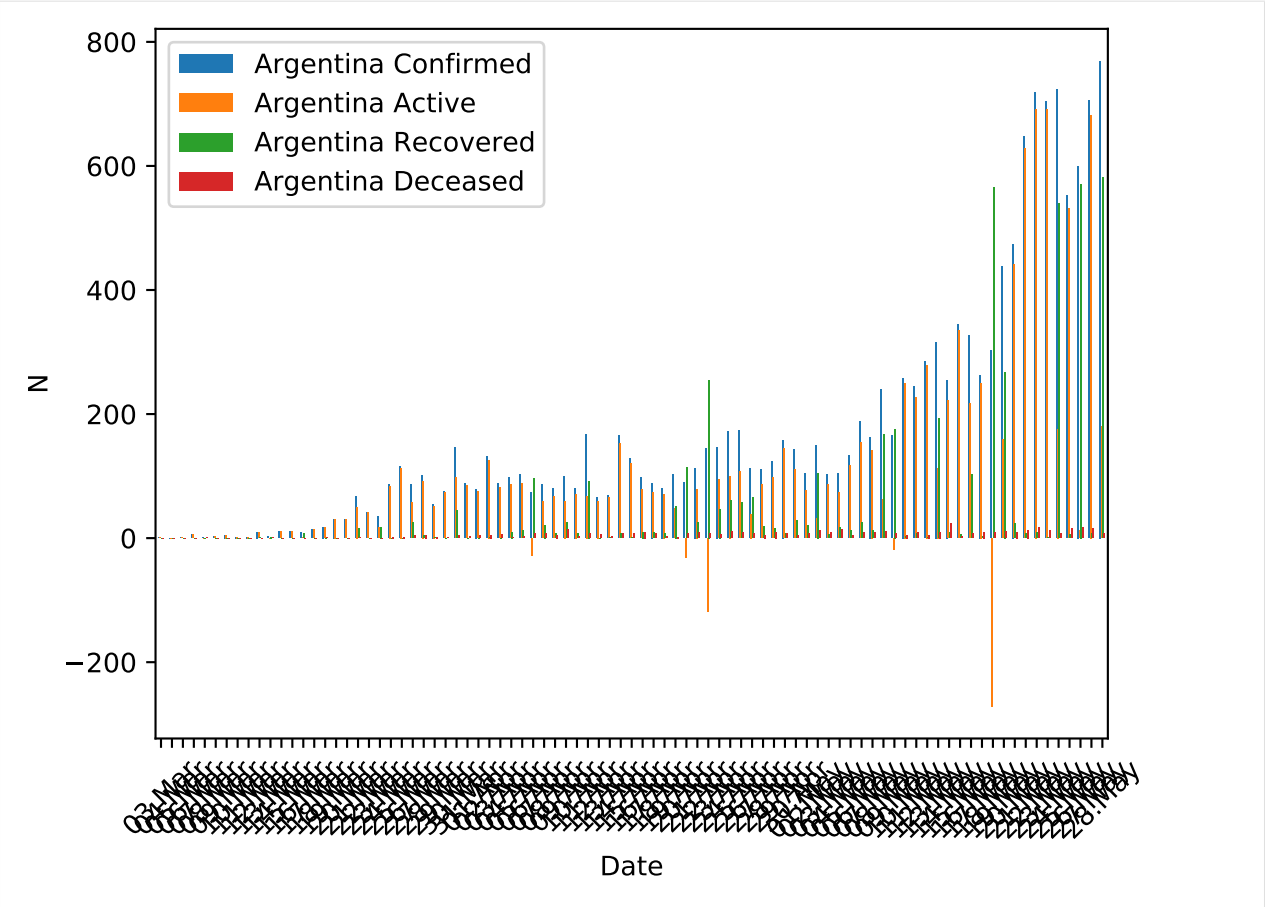
[7]:
```
cases.plot.time_serie()
```

[7]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f611f725208>
```



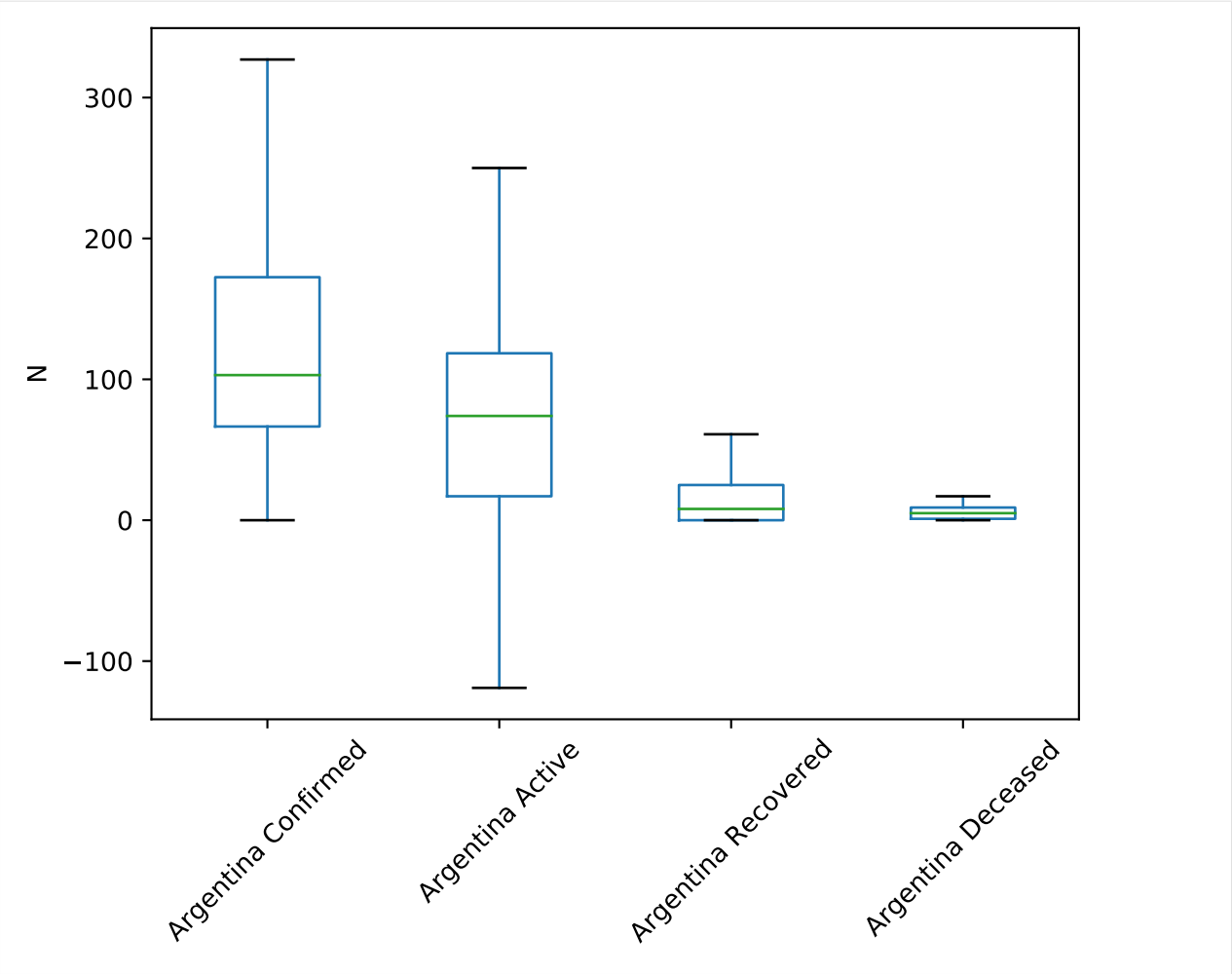
[8]:
```
cases.plot.barplot()
```

[8]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f611d0152e8>
```



[9]:
```
cases.plot.boxplot(showfliers=False)
```

[9]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f611cbedeb8>
```



Finally, at the following link part 3 you will find an updated guide on March 28, 2020, of part 3 of the tutorial.

5.4 Web Tutorial

Arcovid19 presents a tool for the online visualization of epidemiological models.

The simplest way to execute this application is simply to run

> **Warning:** This is a test server, and should not be used in a production environment for any reason.

```
$ arcovid19 webserver
* Serving Flask app "arcovid19.web" (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
* Restarting with inotify reloader
```

(continues on next page)

(continued from previous page)

```
* Debugger is active!
* Debugger PIN: XXX
```

This launches a **local** application which you can access through the url `http://localhost:5000`.

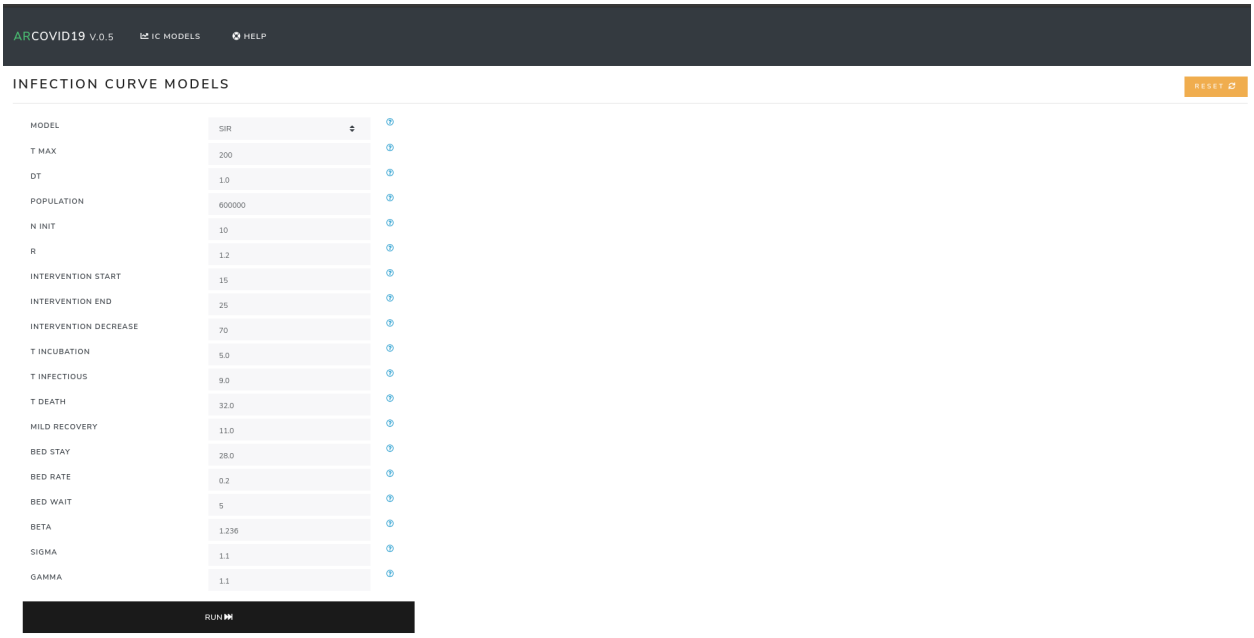


Fig. 1: Default view of arcovid19 webclient in version 0.5.

If for some reason it is necessary to launch the app in another *IP* or *port*, this can be specified with the options `--host` and `--port` respectively. For example, if you want to serve the local network on port *8000* the command would be

```
$ arcovid19 webserver --host 0.0.0.0 --port 8000
* Serving Flask app "arcovid19.web" (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: on
* Running on http://0.0.0.0:8000/ (Press CTRL+C to quit)
* Restarting with inotify reloader
* Debugger is active!
* Debugger PIN: 242-079-243
```

This would allow anyone connected to the same local network as the computer where the web application is started to access the website via the server's IP and port *8000*.

Note: For more web server options you can execute the command `arcovid19 webserver --help`.

5.4.1 Cambiando de idioma

So far arcovid19 web only has two languages implemented.

1. `en` - Ingles (Activated by default)

2. `es` - Español.

To activate the alternative language, you must assign a environment variable. called `ARCOVID19_DEFAULT_LOCALE`.

This is done with the command

```
$ export ARCOVID19_DEFAULT_LOCALE=es;
```

After that, it is simply a matter of launching the application with `arcovid19 werbserver`.

5.4.2 Deployment

To run arcovid19 webserver in a production environment at least 2 environment variables must be configured:

- `ARCOVID19_DEBUG=false`
- `ARCOVID19_SECRET_KEY=a-lot-of-random-chars`

ARCOVID19_SECRET_KEY has to be a contiguous random string of random values, you can, for example, press keys without any sense.

If you want to configure the language to Spanish, you must export the environment variable

- `ARCOVID19_DEFAULT_LOCALE=es`

Deployment can be done with any available method to Flask, and the wsgi application is available with the function `arcovid19.web.create_app()`.

For example, to launch the application with Gunicorn can be done with the command

```
$ gunicorn 'arcovid19.web:create_app()'
```

In addition, the repository is already configured with the files *requirements.txt* and *Procfile* to work in Heroku.

An online version of the webapp is available in here.

5.5 API

5.5.1 arcovid19 package

Module contents

Utilities to access different Argentina-Related databases of COVID-19 data from the Arcovid19 group.

Subpackages

arcovid19.web package

Submodules

arcovid19.web.bp module

Core functionalities for arcovid19

arcovid19.web.bp.**wavid19 = <flask.blueprints.Blueprint object>**
Flask blueprint with the views implemented in arcovid19.

arcovid19.web.forms module

WTF Forms for arcovid19

class arcovid19.web.forms.**InfectionCurveForm**(*formdata=<object object>*, ***kwargs*)
Bases: flask_wtf.form.FlaskForm

N_init = <UnboundField(IntegerField, (l'N Init',), {'description': l'Number of initial

R = <UnboundField(FloatField, (l'R',), {'description': l'Reproduction number.', 'defau

bed_rate = <UnboundField(FloatField, (l'Bed Rate',), {'description': l'Hospitalization

bed_stay = <UnboundField(FloatField, (l'Bed Stay',), {'description': l'Length of hospi

bed_wait = <UnboundField(IntegerField, (l'Bed Wait',), {'description': l'Time from firs

beta = <UnboundField(FloatField, (l'Beta',), {'description': l'SEIR Model beta ($\beta

dt = <UnboundField(FloatField, (l'Dt',), {'description': l'Time step [days].', 'defaul

gamma = <UnboundField(FloatField, (l'Gamma',), {'description': l'SEIR Model gamma ($\g

intervention_decrease = <UnboundField(FloatField, (l'Intervention Decrease',), {'descr

intervention_end = <UnboundField(IntegerField, (l'Intervention End',), {'description':

intervention_start = <UnboundField(IntegerField, (l'Intervention Start',), {'descripti

mild_recovery = <UnboundField(FloatField, (l'Mild Recovery',), {'description': l'Recov

model = <UnboundField(SelectField, (l'Model',), {'choices': [('do_SIR', l'SIR'), ('do_

population = <UnboundField(IntegerField, (l'Population',), {'description': l'Populatio

sigma = <UnboundField(FloatField, (l'Sigma',), {'description': l'SEIR Model sigma ($\s

t_death = <UnboundField(FloatField, (l'T Death',), {'description': l'Time from end of

t_incubation = <UnboundField(FloatField, (l'T Incubation',), {'description': l'Length

t_infectious = <UnboundField(FloatField, (l'T Infectious',), {'description': l'None',

t_max = <UnboundField(IntegerField, (l'T Max',), {'description': l'Time range [days].'

Submodules

arcovid19.core module

Core functionalities for arcovid19

class arcovid19.core.**Frame**(*df*, *extra=NOTHING*)
Bases: object

plot_cls

class arcovid19.core.**Plotter**(*frame*)
Bases: object

default_plot_name_method

arcovid19.cache module

Cache abstractions for arcovid19

arcovid19.cache.**DEFAULT_CACHE_DIR = '/home/docs/arcovid19_data/_cache_'**
Default cache location, (default=~/arcovid_19_data/_cache_)

arcovid19.cache.**CACHE = <diskcache.core.Cache object>**
Default cache instance

arcovid19.cache.**CACHE_EXPIRE = 3600**
Time to expire of every load_cases call in seconds

arcovid19.cache.**from_cache**(*tag*, *function*, *force=False*, **args*, ***kwargs*)
Simple cache orchestration.

Parameters

- **tag** (*str*) – Normally every function call the cache with their own tag. We sugest "module.function" or "module.Class.function"
- **function** (*callable*) – The function to be cached
- **force** (*bool (default=False)*) – If the vale of the cache must be ignored and re-execute the function.
- **and kwargs** (*args*) – All the parameters needed to execute the function.

arcovid19.cases module

Utilities to Utility function to parse all the actual cases of the COVID-19 in Argentina.

arcovid19.cases.**CODE_TO_POVINCIA = {'BA': 'Bs As', 'CABA': 'CABA', 'CAT': 'Catamarca', 'CB**
List of Argentina provinces

arcovid19.cases.**D0 = datetime.datetime(2020, 3, 11, 0, 0)**
Pandemia Start 2020-03-11

arcovid19.cases.**Q1 = datetime.datetime(2020, 3, 20, 0, 0)**
Argentine quarantine starts 2020-03-20

class arcovid19.cases.**CasesPlot**(*frame*)
Bases: *arcovid19.core.Plotter*

barplot(*provincia=None*, *confirmed=True*, *active=True*, *recovered=True*, *deceased=True*, *ax=None*, ***kwargs*)

boxplot(*provincia=None*, *confirmed=True*, *active=True*, *recovered=True*, *deceased=True*, *ax=None*, ***kwargs*)

curva_epi_pais(*ax=None*, *argentina=True*, *exclude=None*, *log=False*, *norm=False*, *paint=None*, *count_days=None*, ***kwargs*)
method: full_period_normalized()

This function plots the time series, similar to grate_full_period_all, but including a second axis and comments about the start of quarantine

opciones para paint: pandemia, cuarentena

opciones para count_days: pandemia, cuarentena

curva_epi_provincia(**args*, ***kwargs*)

default_plot_name_method = 'curva_epi_pais'

grate_full_period(**args*, ***kwargs*)

grate_full_period_all(**args*, ***kwargs*)

growth_provincia(*provincia=None*, *confirmed=True*, *active=True*, *recovered=True*, *deceased=True*, *ax=None*, *log=False*, *norm=False*, ***kwargs*)

time_serie(*provincia=None*, *confirmed=True*, *active=True*, *recovered=True*, *deceased=True*, *ax=None*, ***kwargs*)

time_serie_all(*ax=None*, *argentina=True*, *exclude=None*, ***kwargs*)

class arcovid19.cases.**CasesFrame**(*df*, *extra=NOTHING*)

Bases: *arcovid19.core.Frame*

Wrapper around the *load_cases()* table.

This class adds functionalities around the dataframe.

dates

Returns the dates for which we have data.

Useful to use as time column (row) list for wide (long) format.

get_provincia_name_code(*provincia*)

Resolve and validate the name and code of a given provincia name or code.

grate_full_period(*provincia=None*)

Estimates growth rate for the period where we have data

last_growth_rate(*provincia=None*)

Returns the last available growth rate for the whole country if provincia is None, or for only the named region.

plot_cls

alias of *CasesPlot*

restore_time_serie()

Retrieve a new pandas.DataFrame but with observations by Date.

tot_cases

Returns latest value of total confirmed cases

arcovid19.cases.**load_cases**(*cases_url='https://github.com/ivco19/libs/raw/master/databases/cases.xlsx'*, *areas_pop_url='https://github.com/ivco19/libs/raw/master/databases/extra/arg_provs.dat'*, *force=False*)

Utility function to parse all the actual cases of the COVID-19 in Argentina.

Parameters

- **cases_url** (*str*) – The url for the excel table to parse. Default is ivco19 team table.
- **areas_pop_url** (*str*) – The url for the csv population table to parse. Default is ivco19 team table.
- **force** (*bool (default=False)*) – If you want to ignore the local cache and retrieve a new value.

Returns

CasesFrame – It features a pandas multi index, with the following hierarchy:

- level 0: cod_provincia - Argentina states
- level 1: cod_status - Four states of disease patients (R, C, A, D)

Return type Pandas-DataFrame like object with all the arcovid19 datatabase.

arcovid19.models module

Utilities to Utility function to parse all the actual cases of the COVID-19 in Argentina.

exception arcovid19.models.**NodeNotFoundError**
Bases: KeyError

If a node is not found inside a graph

class arcovid19.models.**Node**(*id*, *value*, *outgoing=NOTHING*, *incoming=NOTHING*)
Bases: object

This class is used to create and manipulated nodes.

add_neighbor(*neighbor*, *names*, *values*)

be_neighbor(*neighbor*, *names*, *values*)

get_connections()

get_weight(*neighbor*)

class arcovid19.models.**Graph**(*vert_dict=NOTHING*)
Bases: object

This class is used to create and manipulated graphs.

It makes a heavy use of the node class A graph is made of nodes and edges. This class allows to store a value for each node and different "weights" for each edge. Also, edges are directed.

Example

```
>>> g = Graph()
>>> for i, inode in enumerate(['A','B','C','D']):
...     print(i)
...     g.add_node(inode, 0)
```

```
>>> nms = ['x', 'y']
>>> g.add_edge('A', 'B', nms, [1, 100])
>>> g.add_edge('A', 'C', nms, [2, 200])
>>> g.add_edge('B', 'D', nms, [3, 300])
>>> g.add_edge('D', 'B', nms, [4, 400])
>>> g.add_edge('D', 'C', nms, [5, 500])
>>> g.add_edge('C', 'C', nms, [6, 600])
```

A node can be connected to itself. >>> g.add_edge('B', 'B', nms, [333, 333]) >>> g.show() ...

vert_dict
a dict containing the vertices

Type dict

add_edge(*frm*, *to*, *names=None*, *values=0*)
Link two nodes inside a graph.

Notes

Does not verify if edge already exists

Raises NodeNotFound: – If one of the node are not tregistered in the graph

add_node(*nnode*, *value*)
Adds a new node to a graph.

The node must have a value.

format_connections()

format_weights()

get_edge(*frm*, *to*, *field*)

get_node(*n*)
Retrieve a node if exists.

Parameters **n** (*str*) –

Returns **node**

Return type a node object

get_node_ids()

get_node_value(*n*)
Returns the value of the node.

Parameters **n** (*str*) –

Returns **value**

Return type float

get_nodes_from(*nnode*)

get_nodes_to(*nnode*)

node_activation(*nnode*, *key*)

node_upgrade(*nnode*, *key*)

resume_connections()

resume_weights()

set_node(*n*, *value*)
Updates the Node value. The node must exists inside the graph.

Parameters

- **n** (*str*) – The ID or name of the node
- **value** (*float*) – The value to be assigned to the node

class arcovid19.models.**InfectionCurve**(*population: int = 600000*, *N_init: int = 10*, *R: float = 1.2*, *intervention_start: int = 15*, *intervention_end: int = 25*, *intervention_decrease: float = 70*, *t_incubation: float = 5.0*, *t_infectious: float = 9.0*, *t_death: float = 32.0*, *mild_recovery: float = 11.0*, *bed_stay: float = 28.0*, *bed_rate: float = 0.2*, *bed_wait: int = 5*, *beta: float = 1.236*, *sigma: float = 1.1*, *gamma: float = 1.1*)

Bases: object

MArce documentame me siento sola.

Parameters

- **population** (*int (default=600000)*) – Population.
- **N_init** (*int (default=10)*) – Number of initial infections.

- **R** (*float (default=1.2)*) – Reproduction number.
- **intervention_start** (*int (default=15)*) – Start intervention days.
- **intervention_end** (*int (default=25)*) – End intervention days.
- **intervention_decrease** (*float (default=70)*) – Decrease in transmission for intervention, percentage (0-100) 100 means total isolation.
- **t_incubation** (*float (default=5.)*) – Length of incubation period.
- **t_infectiou** (*= (default=9.)*) – Duration patient is infectious.
- **t_death** (*float (default=32.)*) – Time from end of incubation to death.
- **mild_recovery** (*float (default=0.2)*) – Recovery time for mild (not severnot severe) cases, days hospitalization rate (fraction).
- **bed_stay** (*float (default=28.)*) – Length of hospital stay, days.
- **bed_rate** (*float (default=0.2)*) – Hospitalization rate (fraction).
- **bed_wait** (*int (default=5)*) – Time from first synthoms to hospitalization (days).
- **beta** (*float (default=1.236)*) – SEIR Model beta ($beta$).
- **sigma** (*float (default=1.1)*) – SEIR Model sigma ($sigma$).
- **gamma** (*float (default=1.1)*) – SEIR Model gamma ($gamma$).

References

"Stochastic SIR model with Python," epirecipes. [Online]. Available: https://tinyurl.com/y8zwvfk4. [Accessed: 09-May-2020].

do_SEIR(*t_max=200*, *dt=1.0*)
This function implements a SEIR model without vital dynamics under the assumption of a closed population. Recovered individuals become immune for ever. ref.: https://www.idmod.org/docs/hiv/model-seir.html

Parameters

- **t_max** (*int (default=200)*) – Time range [days].
- **dt** (*float (default=1.)*) – Time step [days].

Returns value

Return type Time series for S, E, I and R

do_SEIRF(*t_max=200*, *dt=1.0*)
Documentame MARCE

Parameters

- **t_max** (*int (default=200)*) – Time range [days].
- **dt** (*float (default=1.)*) – Time step [days].

Returns value

Return type Time series for S, E, I, R and F

do_SIR(*t_max=200*, *dt=1.0*)
This function implements a SIR model without vital dynamics under the assumption of a closed population.

Recovered individuals become immune for ever. In this model exposed individuals become instantly infectious, i.e., there is no latency period like in the SEIR model.

Parameters

- **t_max** (*int (default=200)*) – Time range [days].
- **dt** (*float (default=1.)*) – Time step [days].

Returns value

Return type Time series for S, I and R

arcovid19.models.**load_infection_curve**(***kwargs*)
Return a new instance of infection curve.

arcovid19.cli module

Command line interfaces to access different Argentina-Related databases of COVID-19 data from the Arcovid19 group.

arcovid19.cli.**main**()
Run the arcovid19 command line interface.

arcovid19.cli.**cases**(***, *url='https://github.com/ivco19/libs/raw/master/databases/cases.xlsx'*, *force=False*, *out=None*)
Retrieve and store the cases database in CSV format.

url: str The url for the excel table to parse. Default is ivco19 team table.

out: PATH (default=stdout) The output path to the CSV file. If it's not provided the data is printed in the stdout.

force: If you want to ignore the local cache or retrieve a new value.

arcovid19.cli.**webserver**(***, *host=None*, *port=None*, *nodebug=False*, *reload=False*, *load_dotenv=True*)
Run a development server for arcovid19 utilities.

host: str the hostname to listen on. Set this to '0.0.0.0' to have the server available externally as well. Defaults to '127.0.0.1' or the host in the SERVER_NAME config variable if present.

port: int the port of the webserver. Defaults to 5000 or the port defined in the SERVER_NAME config variable if present.

nodebug: bool if given, disable debug mode.

reload: bool If its True any change of the code will restart the webserver.

load_dotenv: Load the nearest '.env' and '.flaskenv' files to set environment variables. Will also change the working directory to the directory containing the first file found.

Python Module Index

a

Index

I

L

M

N

P

Q

R

S

T

V

W