Get up-to-date national and regional charts for the covid-19 emergency using python notebooks

Tufarini Emanuele

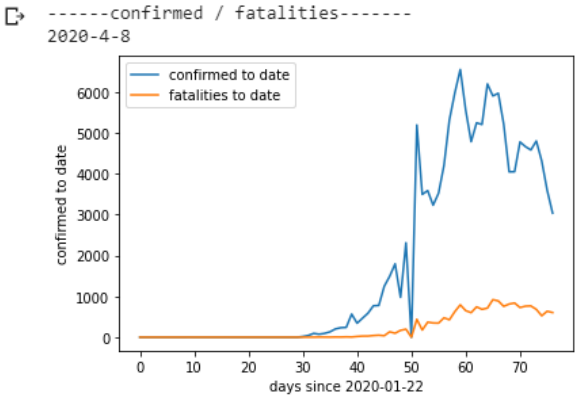
emanuele.tufarini@live.com

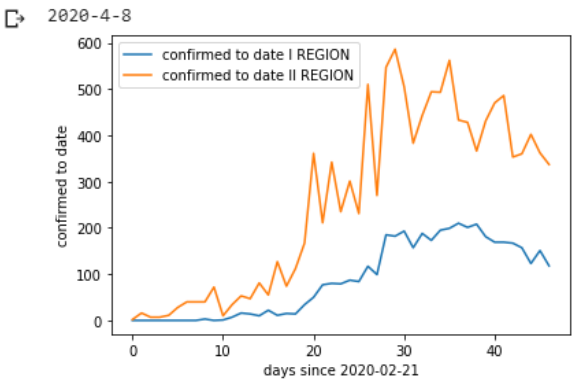
2020-04-11

Abstract

This study is intended to help in drawing the charts, constantly updated, based on data (confirmed cases and deaths) concerning the COVID-19 emergency. These programs can be used to get an overview of the trend of the curves.
The methods shown allow access to the data contained in Google BigQuery, a Google cloud data warehouse that allows you to store and retrieve data using an SQL environment. Google has recently made available to everyone a series of data concerning the COVID-19, through the COVID-19 Public Dataset Program, a freely accessible archive. This data source can be accessed through pre-existing libraries, developed for python, one of them is COVID19Py, it allows access to updated data on coronavirus cases (COVID-19, SARS-CoV-2) in a simple and fast way. Through this library you can get updated data on deaths and confirmed COVID-19 cases for several countries (including Italy).
In this study, python programs have been created that perform the listed operations for the individual regions (create charts of COVID-19 confirmed cases), analyzing data contained in an Excel file, these files are easy to create, update and share. The charts can be viewed using the matplotlib library, useful for creating static, animated and interactive views in python. Similar programs can also be used in other areas than COVID-19.
The data of Italy can also be analyzed through an excel file.
Notebooks are available at https://github.com/EmaTechnologist/COVID-19-national-and-regional-chart

Below are some charts obtained with these programs:





Keywords: covid-19, python, google colaboratory, google bigquery, covid19py, matplotlib, excel

1. GET AND MANAGE COVID-19 DATA THROUGH PYTHON NOTEBOOKS

Python notebooks have been created through Google Colaboratory (a web-based computing environment accessible for search purposes). These programs allow you to display curves that are always up to date by entering the desired nations or region.

In order to allow an equally efficient management for each region with regard to confirmed cases, a program has been created that analyzes an excel file, containing the data of all regions. This Excel file can be generated using data provided by the Italian Civil Protection or similar tables can be found by searching for 'daily COVID-19 cases in Italy by region'.

1.1 CHART BY NATION THROUGH NOTEBOOKS

First were installed the packages:

```
!pip install COVID19Py
!pip install matplotlib
```

The modules must be imported for program operation:

```
from matplotlib import pyplot as plt
import COVID19Py
import os
import re
```

You can set a function that records the date to store the day the search was performed:

```
# +1 italy time zone > str( int (data.hour + 1))

data = datetime.datetime.now()
data = (str(data.year) + "-" + str(data.month) + "-" + \
    str(data.day))
print (data)
```

Below is written a code to download confirmed cases data, the program will save these data in a text file, then they will be inserted in a list to be used to generate the chart.

The data comes from John Hopkins University.

You can change country by replacing the code "IT" with the code of the desired country (ISO 3166-1 alpha-2):

```
# confirmed curve program

covid19 = COVID19Py.COVID19(data_source="jhu")
# info from jhu Johns Hopkins University Center for Systems Science
# and Engineering

latest = covid19.getLatest()

location = covid19.getLocationByCountryCode("IT", timelines=True)

location = str (location)
```

```
p = (location.find (', 'timelines': {'confirmed': '))
p2 = (location.find (', 'deaths': {'latest':'))

f = open ("death_covid19.txt", "w")
f.write (location [p:p2])

f.close()

print ("------confirmed-------")

g = open ("death_covid19.txt", "r")
death_s = g.read()
death_s = str (death_s)

p3 = (death_s.find ("'timeline': "))

death_s = death_s[p3 + 12:]

g.close()

h = open ("death_covid19.txt", "w")
h.write (death_s)
h.close()

h = open ("death_covid19.txt", "r")
h = h.read()[1:-1]
h = re.split(": |,", h)
days = (h[0::2])
infected = (h[1::2])

days = [elem[:12] for elem in days]

days = [elem for elem in days if elem.strip()]

days = [item.replace('T', ') for item in days]
days = [item.replace("", ') for item in days]
days = [item.replace(" ", ') for item in days]

infected = [item.replace("}", ') for item in infected]

infected = list(map(int, infected))

n = 0
a = open ("infected_covid19_daily.txt", "w")
a.write (str (infected[0]) + "\n")
```

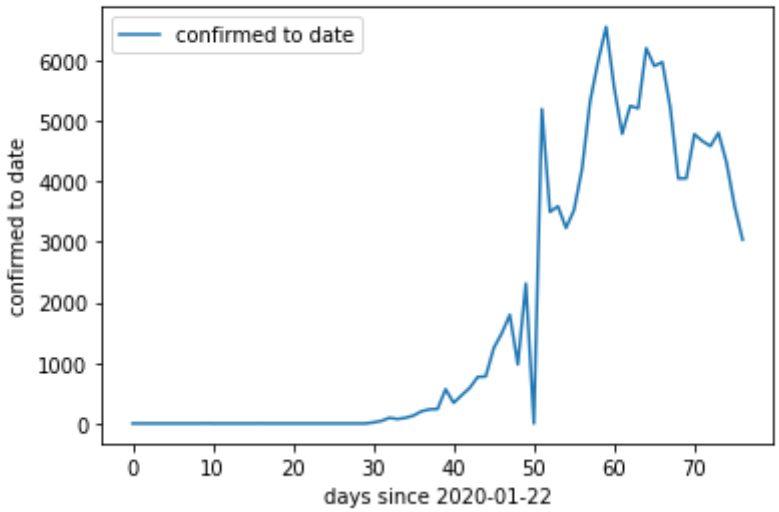
```
try:
  for e in infected:
    d_infected = infected[n + 1] - infected [n]
    n = n + 1
    a.write (str (d_infected) + "\n")
except: pass
a.close()

a = open ("infected_covid19_daily.txt", "r")

d_infected = a.read()
d_infected = list(map(int, d_infected.split("\n")[:-1]))
```

Through the matplotlib library can be traced the chart of confirmed cases, constantly updated:

```
print (data)

# GRAPH

plt.plot(d_infected, label="confirmed to date")
plt.ylabel("confirmed to date")
plt.xlabel("days since 2020-01-22")
plt.legend()

plt.show()
```

Below is the chart of confirmed cases for Italy:

```
------confirmed-------
2020-4-8
```



You can write a code to download death data:

```
covid19 = COVID19Py.COVID19(data_source="jhu")
# info from jhu Johns Hopkins University Center for Systems Science
# and Engineering

latest = covid19.getLatest()

location = covid19.getLocationByCountryCode("IT", timelines=True)

location = str (location)

p = (location.find ("'deaths': {"))
p2 = (location.find ("}, 'recovered': "))

f = open ("death_covid19.txt", "w")
f.write (location [p:p2])

f.close()

print ("------fatalities-------")

g = open ("death_covid19.txt", "r")
death_s = g.read()
death_s = str (death_s)

p3 = (death_s.find ("'timeline': "))

death_s = death_s[p3 + 12:]
g.close()

h = open ("death_covid19.txt", "w")
h.write (death_s)
h.close()

h = open ("death_covid19.txt", "r")
h = h.read()[1:-1]
h = re.split(": |,", h)
days = (h[0::2])
death = (h[1::2])

days = [elem[:12] for elem in days]

days = [elem for elem in days if elem.strip()]

days = [item.replace('T', '') for item in days]
days = [item.replace("'", '') for item in days]
days = [item.replace(" ", '') for item in days]
```

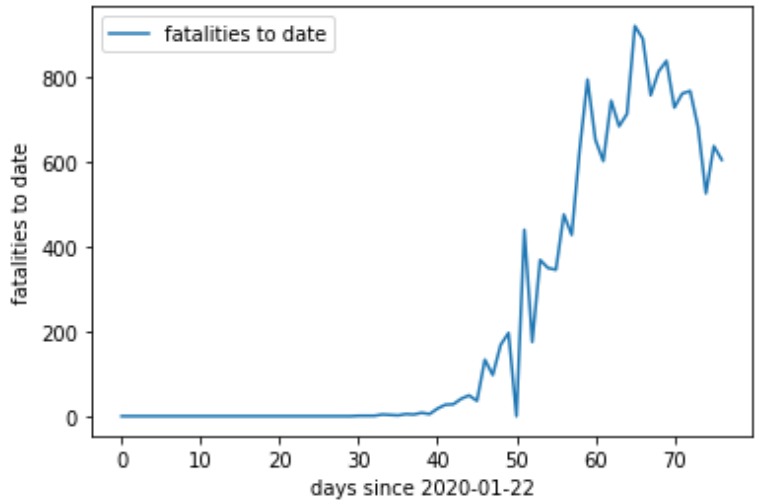
```
death = list(map(int, death))
n = 0
a = open ("death_covid19_daily.txt", "w")
a.write (str (death[0]) + "\n")

try:
  for e in death:
    d_death = death[n + 1] - death [n]
    n = n + 1
    a.write (str (d_death) + "\n")
except: pass

a.close()

a = open ("death_covid19_daily.txt", "r")

d_death = a.read()
d_death = list(map(int, d_death.split("\n")[:-1]))
```

As was done for the confirmed cases, you can chart the deaths:

```
print (data)

# GRAPH

plt.plot(d_death, label="fatalities to date")
plt.ylabel("fatalities to date")
plt.xlabel("days since 2020-01-22")
plt.legend()

plt.show()
```

Below is the chart of deaths for Italy:

```
------fatalities-------
2020-4-8
```



Using the codes shown you can get a chart showing both confirmed cases and deaths:

```
# fatality curve program

covid19 = COVID19Py.COVID19(data_source="jhu")
# info from jhu Johns Hopkins University Center for Systems Science
# and Engineering

latest = covid19.getLatest()

location = covid19.getLocationByCountryCode("IT", timelines=True)

location = str (location)

p = (location.find (", 'timelines': {'confirmed': "))
p2 = (location.find (", 'deaths': {'latest':"))

f = open ("death_covid19.txt", "w")
f.write (location [p:p2])

f.close()

g = open ("death_covid19.txt", "r")
death_s = g.read()
death_s = str (death_s)
```

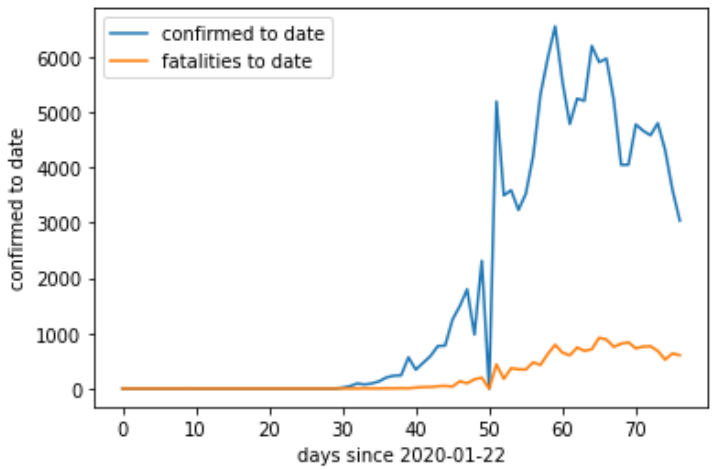
```
p3 = (death_s.find ("'timeline': "))

death_s = death_s[p3 + 12:]

g.close()

h = open ("death_covid19.txt", "w")
h.write (death_s)
h.close()

h = open ("death_covid19.txt", "r")
h = h.read()[1:-1]
h = re.split(": |,", h)
days = (h[0::2])
infected = (h[1::2])

days = [elem[:12] for elem in days]

days = [elem for elem in days if elem.strip()]

days = [item.replace('T', '') for item in days]
days = [item.replace('"', '') for item in days]
days = [item.replace(" ", '') for item in days]

infected = [item.replace("}", '') for item in infected]

infected = list(map(int, infected))

n = 0
a = open ("infected_covid19_daily.txt", "w")
a.write (str (infected[0]) + "\n")
try:
  for e in infected:
    d_infected = infected[n + 1] - infected [n]
    n = n + 1
    a.write (str (d_infected) + "\n")
except: pass
a.close()

a = open ("infected_covid19_daily.txt", "r")

d_infected = a.read()
d_infected = list(map(int, d_infected.split("\n")[:-1]))

covid19 = COVID19Py.COVID19(data_source="jhu")
# info from jhu - johns hopkins university
```

```
latest = covid19.getLatest()

location = covid19.getLocationByCountryCode("IT", timelines=True)

location = str (location)

p = (location.find ("'deaths': {"))
p2 = (location.find ("}, 'recovered': "))

f = open ("death_covid19.txt", "w")
f.write (location [p:p2])

f.close()

g = open ("death_covid19.txt", "r")
death_s = g.read()
death_s = str (death_s)

p3 = (death_s.find ("'timeline': "))

death_s = death_s[p3 + 12:]
g.close()

h = open ("death_covid19.txt", "w")
h.write (death_s)
h.close()

h = open ("death_covid19.txt", "r")
h = h.read()[1:-1]
h = re.split(": |,", h)
days = (h[0::2])
death = (h[1::2])


days = [elem[:12] for elem in days]

days = [elem for elem in days if elem.strip()]

days = [item.replace('T', '') for item in days]
days = [item.replace("'", '') for item in days]
days = [item.replace(" ", '') for item in days]

death = list(map(int, death))
n = 0
a = open ("death_covid19_daily.txt", "w")
a.write (str (death[0]) + "\n")
```

```
try:
  for e in death:
    d_death = death[n + 1] - death [n]
    n = n + 1
    a.write (str (d_death) + "\n")
except: pass
a.close()

a = open ("death_covid19_daily.txt", "r")

d_death = a.read()
d_death = list(map(int, d_death.split("\n")[:-1]))

print ("------confirmed / fatalities-------")

print (data)

# GRAPH

plt.plot(d_infected, label="confirmed to date")
plt.plot(d_death, label="fatalities to date")
plt.ylabel("confirmed to date")
plt.xlabel("days since 2020-01-22")
plt.legend()
plt.show()
```

Below is a chart showing both confirmed cases and deaths for Italy:

```
------confirmed / fatalities-------
2020-4-8
```



Finally we can remove unnecessary files:

```
os.remove('death_covid19.txt')
```

1.1.1 CHART BY REGION THROUGH PYTHON NOTEBOOKS

You can create a program that performs the same operations for the desired region. The program processes an excel file containing the data for each region. The data corresponds to the data released by the civil protection authorities. The excel file can be created manually or downloaded from different sites.

You can upload the excel file to your notebook using the code:

```
from google.colab import files
uploaded = files.upload()
```

First were installed the packages:

```
!pip install matplotlib
!pip install COVID19Py
!pip install pandas
!pip install xlrd
```

The modules must be imported for program operation:

```
from matplotlib import pyplot as plt
import pandas as pd
import datetime
import csv
import os
```

You can set a function that records the date to store the day the search was performed:

```
# +1 italy time zone > str( int (data.hour + 1))

data_day = datetime.datetime.now()
data_day = (str(data_day.year) + "-" + str(data_day.month) + "-" + \
    str(data_day.day))
```

The table used is shown below (white spaces are converted to 0). To use the program with other excel files you have to modify part of the code, or you can create a file that follows this scheme:

Date	North-West					North-East					Centre	
	VDA	LIG	PIE	LOM	VEN	TN	BZ	FVG	EMR	MAR	TOS	UMB
31/01/2020												
06/02/2020												
21/02/2020					15 (1) 2							
22/02/2020				1 (1) 40	16				2			
23/02/2020				2 (1) 57	7				7			
24/02/2020				(4) 61	7				9			
25/02/2020		1		(2) 67	(1) 11		1		8		2	
26/02/2020		10		(1) 65	28				(1) 21	1		
27/02/2020		8		1 (5) 98	40				50	2		
28/02/2020				9 (3) 128	40				(1) 48	3	6	
29/02/2020		23		(6) 84	40				(2) 72	5	3	
01/03/2020		−17		38 (1) 369	72				6 (4) 68	14	2	
02/03/2020		−3		2 (14) 270	10				3 (3) 50	(1) 10		
03/03/2020		(1) 2		5 (17) 266	(1) 34	4			4 (7) 85	(1) 26	6	
04/03/2020			2	26 (18) 300	(3) 53	1			5 (4) 124	(2) 23	19	
05/03/2020		2 (2) 2	(2) 26	(25) 431	(4) 47	2			3 (8) 154	40	23	
06/03/2020		5	4 (2) 35	(37) 361	(2) 81	3	3		10 (7) 172	35	18	
07/03/2020		1 (1) 19	(1) 64	(19) 808	(1) 55	4	5		11 (11) 140	(2) 48	34	
08/03/2020		1 (2) 27		153 (113) 769	(5) 127	9		(1) 15	(8) 170	(1) 65	53	
09/03/2020		6 (1) 31	(8) −10	(66) 1,280	(2) 74	10			36 (14) 206	(3) 51	(1) 42	
10/03/2020		2 (1) 32	(4) 103	(135) 322	(6) 112	19	29 (2) 23		(15) 147	(3) 71	56	
11/03/2020	(1) 3		53 (4) 48	(149)	(3) 167	25	37 (3) 10		(28) 206	(5) 85	56	

At this point you must create a program that takes the excel file as input, processes it and enters the confirmed cases data of the region into a text file. This data will be inserted in a list to create the chart with matplotlib:

```
# I REGION
#
# 0 DATA | 1 VDA | 2 LIG | 3 PIE | 4 LOM | 5 VEN | 6 TN | 7 BZ | 8 FVG | 9 EMR
# 10 MAR | 11 TOS | 12 UMB | 13 LAZ | 14 ABR | 15 MOL | 16 CAM | 17 BAS | 18 PUG
# 19 CAL | 20 SIC | 21 SAR
# https://en.m.wikipedia.org/wiki/Template:2019%E2%80%9320_coronavirus_
# pandemic_data/Italy_medical_cases
#
# insert region
region = 13

# insert file name
name = r'confirmed_region'
extension = '.xlsx'
read_file = pd.read_excel (name + extension)
name = name.lower()


read_file.to_csv (name + '.csv', index = None, header=True)
# delete first line of csv
lines = open(name + '.csv', 'r').readlines()
del lines[0:4]
```

```
open(name + '.csv', 'w').writelines(lines)

# process the csv

name = 'confirmed_region.csv'

with open(name, 'r') as infile, open('new_' + name, 'w') as outfile:
  writer = csv.writer(outfile)
  newrow_list = ["date","confirmed",]
  writer.writerow(newrow_list)

  reader = csv.reader(infile)
  for row in reader:
    newrow_list[0] = row[0]
    newrow_list[1] = row[region]

    if row[region] == '':
      newrow_list[1] = '0'

    writer.writerow(newrow_list)

    confirmed_region = open ('confirmed_region.txt','w')

    confirmed_region.write(newrow_list[0])

# convert csv in txt - contains data

csv_file = 'new_' + name
txt_file = 'data.txt'
with open(txt_file, "w") as my_output_file:
    with open(csv_file, "r") as my_input_file:
        [ my_output_file.write(" ".join(row)+'\n') \
         for row in csv.reader(my_input_file)]
    my_output_file.close()

with open('data.txt', 'r', encoding='utf-8') as inFile,\
     open('new_data.txt', 'w', encoding='utf-8') as outFile:
    for line in inFile:
        if line.strip():
            outFile.write(line)

lines = open('new_data' + '.txt', 'r').readlines()
del lines[-6:]
open('new_data' + '.txt', 'w').writelines(lines)

filein = open ("new_data.txt","r")
```

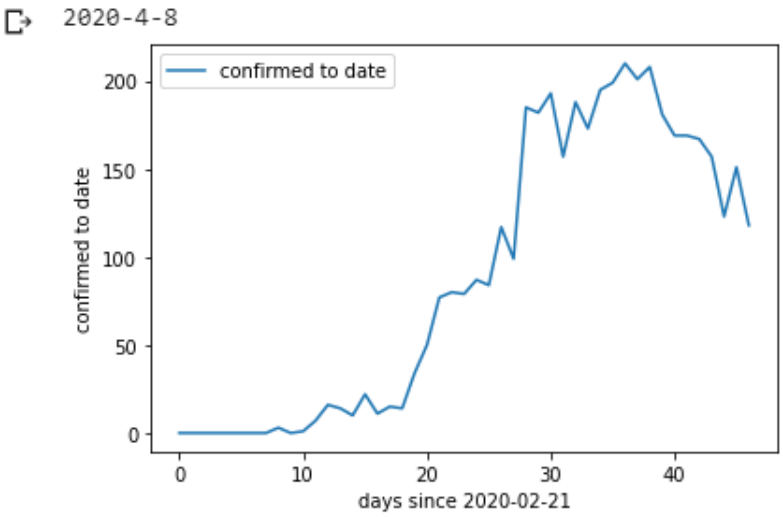
```
file_data = open ("only_data.txt","w")
file_confirmed = open ("only_confirmed.txt","w")

for l in filein:
  data = (l[:10])
  file_data.write(data.strip() + "\n")

  confirmed = (l[20:])
  file_confirmed.write(confirmed.strip() + "\n")

file_data.close()
file_confirmed.close()

# delete the first line
with open("only_data.txt", 'r') as fin:
  data = fin.read().splitlines(True)
with open("only_data.txt", 'w') as fout:
  fout.writelines(data[1:])


with open("only_confirmed.txt", 'r') as fin:
  data = fin.read().splitlines(True)
with open("only_confirmed.txt", 'w') as fout:
  fout.writelines(data[1:])

List = open("only_confirmed.txt").readlines()
List = [x.replace('\n', '') for x in List]

with open("only_confirmed.txt", 'w') as fin:

  for l in List:
    p = ") "
    if p in l:
      fin.write(l[l.index(p)+2:] + "\n")
    else:
      fin.write(l + "\n")

List = open("only_confirmed.txt").readlines()
List = [x.replace('\n', '') for x in List]
List = [x.replace(',', '') for x in List]
List = [x.replace('–', '-') for x in List]
# convert the elements in the list to int

for i in range(0, len(List)):
  List[i] = int(List[i])
```

```
List_data = open("only_data.txt").readlines()
List_data = [x.replace('\n', '') for x in List_data]
```

At this point you can create the chart for the desired region:

```
print (data_day)

# graph by region

plt.plot(List, label="confirmed to date")
plt.ylabel("confirmed to date")
plt.xlabel("days since 2020-02-21")
plt.legend()

plt.show()
```

The chart of the Lazio region is shown below:



You can write the same code to get the data of a second region:

```
# II REGION
#
# 0 DATA | 1 VDA | 2 LIG | 3 PIE | 4 LOM | 5 VEN | 6 TN | 7 BZ | 8 FVG | 9 EMR
# 10 MAR | 11 TOS | 12 UMB | 13 LAZ | 14 ABR | 15 MOL | 16 CAM | 17 BAS | 18 PUG
# 19 CAL | 20 SIC | 21 SAR
#
# insert region
region = 5


# insert file name
```

```
name = r'confirmed_region'
extension = '.xlsx'
read_file = pd.read_excel (name + extension)
name = name.lower()


read_file.to_csv (name + '.csv', index = None, header=True)
# delete first line of csv
lines = open(name + '.csv', 'r').readlines()
del lines[0:4]
open(name + '.csv', 'w').writelines(lines)

# process the csv

name = 'confirmed_region.csv'

with open(name, 'r') as infile, open('new_' + name, 'w') as outfile:
  writer = csv.writer(outfile)
  newrow_list = ["date","confirmed",]
  writer.writerow(newrow_list)

  reader = csv.reader(infile)
  for row in reader:
    newrow_list[0] = row[0]
    newrow_list[1] = row[region]

    if row[region] == '':
      newrow_list[1] = '0'

    writer.writerow(newrow_list)

    confirmed_region = open ('confirmed_region.txt','w')

    confirmed_region.write(newrow_list[0])

# convert csv in txt - contains data

csv_file = 'new_' + name
txt_file = 'data.txt'
with open(txt_file, "w") as my_output_file:
    with open(csv_file, "r") as my_input_file:
        [ my_output_file.write(" ".join(row)+'\n') \
         for row in csv.reader(my_input_file)]
    my_output_file.close()

with open('data.txt', 'r', encoding='utf-8') as inFile,\
     open('new_data.txt', 'w', encoding='utf-8') as outFile:
```

```
    for line in inFile:
        if line.strip():
            outFile.write(line)

lines = open('new_data' + '.txt', 'r').readlines()
del lines[-6:]
open('new_data' + '.txt', 'w').writelines(lines)

filein = open ("new_data.txt","r")

file_data = open ("only_data.txt","w")
file_confirmed = open ("only_confirmed.txt","w")

for l in filein:
  data = (l[:10])
  file_data.write(data.strip() + "\n")

  confirmed = (l[20:])
  file_confirmed.write(confirmed.strip() + "\n")


file_data.close()
file_confirmed.close()

# delete the first line
with open("only_data.txt", 'r') as fin:
  data = fin.read().splitlines(True)
with open("only_data.txt", 'w') as fout:
  fout.writelines(data[1:])


with open("only_confirmed.txt", 'r') as fin:
  data = fin.read().splitlines(True)
with open("only_confirmed.txt", 'w') as fout:
  fout.writelines(data[1:])

List_2 = open("only_confirmed.txt").readlines()
List_2 = [x.replace('\n', '') for x in List_2]

with open("only_confirmed.txt", 'w') as fin:

  for l in List_2:
    p = ") "
    if p in l:
      fin.write(l[l.index(p)+2:] + "\n")
    else:
      fin.write(l + "\n")
```

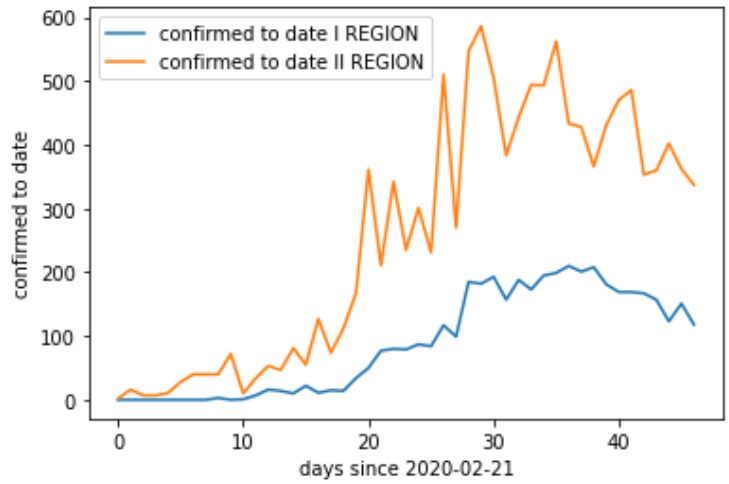
```
List_2 = open("only_confirmed.txt").readlines()
List_2 = [x.replace('\n', '') for x in List_2]
List_2 = [x.replace(',', '') for x in List_2]
# convert the elements in the list to int

for i in range(0, len(List_2)):
  List_2[i] = int(List_2[i])

List_data = open("only_data.txt").readlines()
List_data = [x.replace('\n', '') for x in List_data]
```

You can build a chart that shows the data of both regions through the matplotlib library:

```
print (data_day)

# comparison by region

plt.plot(List, label="confirmed to date I REGION")
plt.plot(List_2, label="confirmed to date II REGION")
plt.ylabel("confirmed to date")
plt.xlabel("days since 2020-02-21")
plt.legend()

plt.show()
```

Below is the chart containing the data for the Lazio region (blue) and the Veneto region (orange):

```
2020-4-8
```



Finally we can remove unnecessary files:

```
os.remove('confirmed_region.csv')
os.remove('new_confirmed_region.csv')
```

```
os.remove('confirmed_region.txt')
os.remove('data.txt')
os.remove('new_data.txt')
```

1.1.2 CHART BY NATION THROUGH PYTHON NOTEBOOKS

You can create updated charts for Italy from an excel file like this:

ITALY

n	dateRep	day	month	year	cases	deaths	countri	geoId	countr	popData2018
103	31/12/2019	31	12	2019	0	0	Italy	IT	ITA	60431283
102	01/01/2020	1	1	2020	0	0	Italy	IT	ITA	60431283
101	02/01/2020	2	1	2020	0	0	Italy	IT	ITA	60431283
100	03/01/2020	3	1	2020	0	0	Italy	IT	ITA	60431283
99	04/01/2020	4	1	2020	0	0	Italy	IT	ITA	60431283
98	05/01/2020	5	1	2020	0	0	Italy	IT	ITA	60431283
97	06/01/2020	6	1	2020	0	0	Italy	IT	ITA	60431283
96	07/01/2020	7	1	2020	0	0	Italy	IT	ITA	60431283
95	08/01/2020	8	1	2020	0	0	Italy	IT	ITA	60431283
94	09/01/2020	9	1	2020	0	0	Italy	IT	ITA	60431283
93	10/01/2020	10	1	2020	0	0	Italy	IT	ITA	60431283
92	11/01/2020	11	1	2020	0	0	Italy	IT	ITA	60431283
91	12/01/2020	12	1	2020	0	0	Italy	IT	ITA	60431283
90	13/01/2020	13	1	2020	0	0	Italy	IT	ITA	60431283
89	14/01/2020	14	1	2020	0	0	Italy	IT	ITA	60431283
88	15/01/2020	15	1	2020	0	0	Italy	IT	ITA	60431283
87	16/01/2020	16	1	2020	0	0	Italy	IT	ITA	60431283

This excel file was created from the downloadable file at the following link (contains data for different countries):
https://data.europa.eu/euodp/en/data/dataset/covid-19-coronavirus-data/resource/55e8f966-d5c8-438e-85bc-c7a5a26f4863

You can change the code used for the regions to get the chart of Italy's cases:

```
# CASES
#
#
# https://data.europa.eu/euodp/en/data/dataset/covid-19-coronavirus-data
# /resource/55e8f966-d5c8-438e-85bc-c7a5a26f4863
#
#

# insert info
info = 5

# insert file name
name = r'cases_italy'
extension = '.xlsx'
read_file = pd.read_excel (name + extension)
name = name.lower()


read_file.to_csv (name + '.csv', index = None, header=True)
# delete first line of csv
```

```
lines = open(name + '.csv', 'r').readlines()
del lines[0:4]
open(name + '.csv', 'w').writelines(lines)

#
#
# process the csv
#
#

name = 'cases_italy.csv'

with open(name, 'r') as infile, open('new_' + name, 'w') as outfile:
  writer = csv.writer(outfile)
  newrow_list = ["date","confirmed",]
  writer.writerow(newrow_list)

################################################################################

  reader = csv.reader(infile)
  for row in reader:
    newrow_list[0] = row[1]
    newrow_list[1] = row[info]

    if row[info] == '':
      newrow_list[1] = '0'

    writer.writerow(newrow_list)

    cases_italy = open ('cases_italy.txt','w')

    cases_italy.write(newrow_list[0])

#
#
# convert csv in txt - contains data
#
#

csv_file = 'new_' + name
txt_file = 'data.txt'
with open(txt_file, "w") as my_output_file:
    with open(csv_file, "r") as my_input_file:
        [ my_output_file.write(" ".join(row)+'\n') \
         for row in csv.reader(my_input_file)]
    my_output_file.close()
```

```
with open('data.txt', 'r', encoding='utf-8') as inFile,\
     open('new_data.txt', 'w', encoding='utf-8') as outFile:
    for line in inFile:
        if line.strip():
            outFile.write(line)

lines = open('new_data' + '.txt', 'r').readlines()
open('new_data' + '.txt', 'w').writelines(lines)

filein = open ("new_data.txt","r")

file_data = open ("only_data.txt","w")
file_confirmed = open ("only_confirmed.txt","w")

for l in filein:
  data = (l[:10])
  file_data.write(data.strip() + "\n")

  confirmed = (l[20:])
  file_confirmed.write(confirmed.strip() + "\n")

file_data.close()
file_confirmed.close()

# delete the first line
with open("only_data.txt", 'r') as fin:
  data = fin.read().splitlines(True)
with open("only_data.txt", 'w') as fout:
  fout.writelines(data[1:])


with open("only_confirmed.txt", 'r') as fin:
  data = fin.read().splitlines(True)
with open("only_confirmed.txt", 'w') as fout:
  fout.writelines(data[1:])

List = open("only_confirmed.txt").readlines()
List = [x.replace('\n', '') for x in List]

with open("only_confirmed.txt", 'w') as fin:

  for l in List:
    p = ") "
    if p in l:
      fin.write(l[l.index(p)+2:] + "\n")
    else:
```

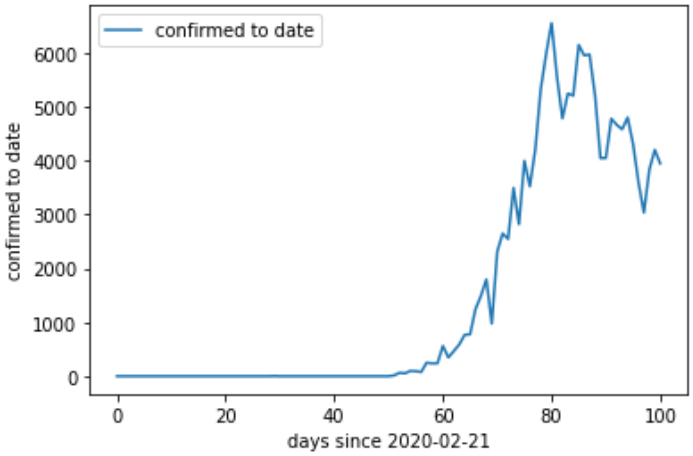
```
    fin.write(l + "\n")

List = open("only_confirmed.txt").readlines()
List = [x.replace('\n', '') for x in List]
List = [x.replace(',', '') for x in List]
List = [x.replace('–', '-') for x in List]
# convert the elements in the list to int

for i in range(0, len(List)):
  List[i] = int(List[i])

List_data = open("only_data.txt").readlines()
List_data = [x.replace('\n', '') for x in List_data]
```

Through the matplotlib library can be traced the chart of confirmed cases:

```
print (data_day)

#
#
# graph by info
#
#

plt.plot(List, label="confirmed to date")
plt.ylabel("confirmed to date")
plt.xlabel("days since 2020-02-21")
plt.legend()

plt.show()
```

Below is the chart of confirmed cases for Italy:

```
2020-4-11
```



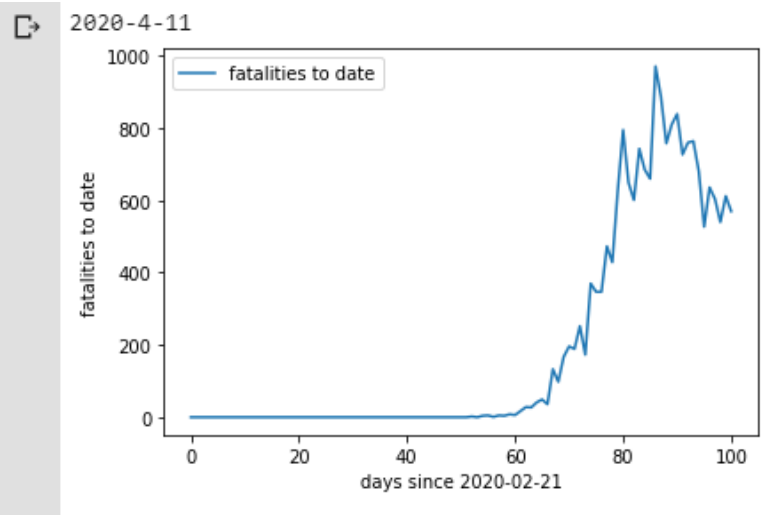
You can write a code to get death data:

```
# DEATHS
#
#
# https://data.europa.eu/euodp/en/data/dataset/covid-19-coronavirus-data
# /resource/55e8f966-d5c8-438e-85bc-c7a5a26f4863
#
#

# insert info
info = 6

# insert file name
name = r'cases_italy'
extension = '.xlsx'
read_file = pd.read_excel (name + extension)
name = name.lower()


read_file.to_csv (name + '.csv', index = None, header=True)
# delete first line of csv
lines = open(name + '.csv', 'r').readlines()
del lines[0:4]
open(name + '.csv', 'w').writelines(lines)

#
#
# process the csv
#
#

name = 'cases_italy.csv'

with open(name, 'r') as infile, open('new_' + name, 'w') as outfile:
  writer = csv.writer(outfile)
  newrow_list = ["date","confirmed",]
  writer.writerow(newrow_list)

###############################################################################

  reader = csv.reader(infile)
  for row in reader:
    newrow_list[0] = row[1]
    newrow_list[1] = row[info]

    if row[info] == '':
      newrow_list[1] = '0'
```

```
    writer.writerow(newrow_list)

    cases_italy = open ('cases_italy.txt','w')

    cases_italy.write(newrow_list[0])

#
#
# convert csv in txt - contains data
#
#

csv_file = 'new_' + name
txt_file = 'data.txt'
with open(txt_file, "w") as my_output_file:
    with open(csv_file, "r") as my_input_file:
        [ my_output_file.write(" ".join(row)+'\n') \
         for row in csv.reader(my_input_file)]
    my_output_file.close()

with open('data.txt', 'r', encoding='utf-8') as inFile,\
     open('new_data.txt', 'w', encoding='utf-8') as outFile:
    for line in inFile:
        if line.strip():
            outFile.write(line)

lines = open('new_data' + '.txt', 'r').readlines()
open('new_data' + '.txt', 'w').writelines(lines)

filein = open ("new_data.txt","r")

file_data = open ("only_data.txt","w")
file_confirmed = open ("only_confirmed.txt","w")

for l in filein:
  data = (l[:10])
  file_data.write(data.strip() + "\n")

  confirmed = (l[20:])
  file_confirmed.write(confirmed.strip() + "\n")

file_data.close()
file_confirmed.close()

# delete the first line
```

```
with open("only_data.txt", 'r') as fin:
  data = fin.read().splitlines(True)
with open("only_data.txt", 'w') as fout:
  fout.writelines(data[1:])


with open("only_confirmed.txt", 'r') as fin:
  data = fin.read().splitlines(True)
with open("only_confirmed.txt", 'w') as fout:
  fout.writelines(data[1:])

List2 = open("only_confirmed.txt").readlines()
List2 = [x.replace('\n', '') for x in List2]

with open("only_confirmed.txt", 'w') as fin:

  for l in List2:
    p = ") "
    if p in l:
      fin.write(l[l.index(p)+2:] + "\n")
    else:
      fin.write(l + "\n")

List2 = open("only_confirmed.txt").readlines()
List2 = [x.replace('\n', '') for x in List2]
List2 = [x.replace(',', '') for x in List2]
List2 = [x.replace('–', '-') for x in List2]
# convert the elements in the list to int

for i in range(0, len(List)):
  List2[i] = int(List2[i])

List_data = open("only_data.txt").readlines()
List_data = [x.replace('\n', '') for x in List_data]
```

Below is the chart of deaths for Italy:

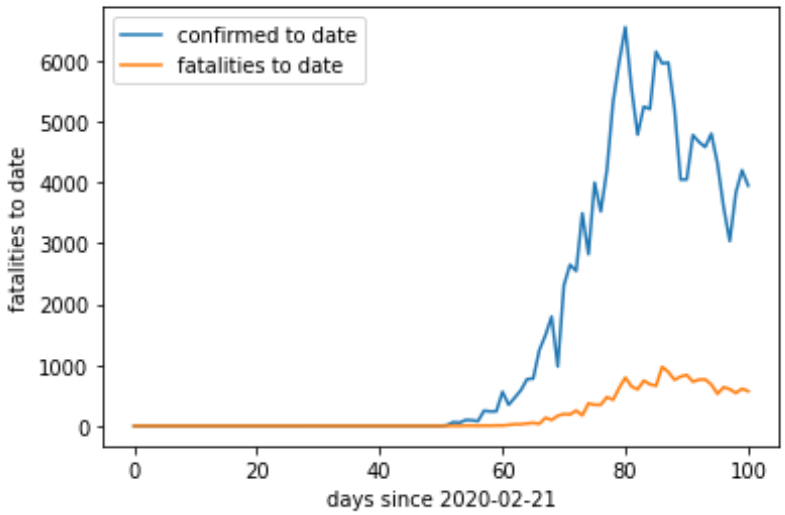


You can write a code to get both, cases and death data:

```
print (data_day)

#
#
# graph by info
#
#

plt.plot(List, label="confirmed to date")
plt.plot(List2, label="fatalities to date")

plt.ylabel("fatalities to date")
plt.xlabel("days since 2020-02-21")
plt.legend()

plt.show()
```

Below is a chart showing both confirmed cases and deaths for Italy:

```
2020-4-11
```



References

[1] https://www.python.org/
[2] https://colab.research.google.com/
[3] https://cloud.google.com/bigquery
[4] https://pypi.org/project/COVID19Py/
[5] https://matplotlib.org/