IFQ509 Assignment 1

Aaron Bachmann, Stephen Whebell

May 30, 2021

1 Introduction

COVID-19 has changed the world in an unprecedented way following its global spread throughout 2020. The availability and analysis of robust data from very early in the pandemic was crucial to understanding how it spread and planning both policy and health interventions.

Thorughout this notebook we will explore and analyse international case data from early in the COVID-19 pandemic. The analysis is presented with full code used to permit easy following of the methods used. The data used is made available publicly by the Johns Hopkins Coronavirus Resource Center.

This document includes selected code cells only, please see the attached .ipynb file for the complete python code for analysis and visualisation.

[1]:
```python
### Import the packages required for analysis and visualisation

import pandas as pd
import numpy as np

from datetime import datetime, timedelta

import matplotlib as mpl
mpl.rcParams['mpl_toolkits.legacy_colorbar'] = False
import matplotlib.pyplot as plt
from matplotlib.ticker import ScalarFormatter
import matplotlib.dates as mdates
import matplotlib.patches as mpatches
from mpl_toolkits.axes_grid1 import AxesGrid

import seaborn as sns

import cartopy.crs as ccrs
import cartopy.feature as cfeature
from cartopy.io import shapereader
from cartopy.mpl.geoaxes import GeoAxes
```

2 Data manipulation and cleaning

The data is available in two formats - one presented as a time series (with the potential to be updated in the same format for future data) and the other presented as seperate CSV files for each day of data. It is more efficient to load the time series data, rather than scraping the available files a directory for new dates as they become available.

The time series CSVs are seperated into three different files for confirmed COVID-19 cases, COVID-19 deaths and recovered cases. The data is presented as a cumulative total for each day. In addition to the case data, there are supplemental files containing information about country populations and abbreviated country codes.

[2]:
```
### Load the data

confirmed = pd.read_csv('data/time_series_19-covid-Confirmed.csv')
deaths = pd.read_csv('data/time_series_19-covid-Deaths.csv')
recovered = pd.read_csv('data/time_series_19-covid-Recovered.csv')

populations = pd.read_csv('data/population-figures-by-country-csv_csv.csv')

country_codes = pd.read_csv('data/country_codes.csv')
```

Examining the structure of our data; we have a wide form table with daily reported case numbers for each country, some divided by region. Each of tables are structured in the same way.

At this stage, we are only interested in examining data on a country, rather than state/territory level. We can store the lat/long specified in a separate dataframe for each country. A wide format dataframe is not the best structure for analysing our data, we can transpose the DF and then convert the dates to a time series index.

This method is not dependent on the number of dates in the dataset and will continue to work as it is updated, facillitating future analysis as more data becomes available.

[3]:
```
# Pull the coordinates for each country and keep in a separate dataframe

coords = confirmed[['Country/Region', 'Lat', 'Long']].groupby('Country/Region').mean()
# A couple of colonial countries aren't amenable to aggregating their coordinates...
coords.loc['United Kingdom']['Lat'] = confirmed[confirmed['Province/State'] == 'United Kingdom']['Lat']
coords.loc['United Kingdom']['Long'] = confirmed[confirmed['Province/State'] == 'United Kingdom']['Long']
coords.loc['Netherlands']['Lat'] = confirmed[confirmed['Province/State'] == 'Netherlands']['Lat']
coords.loc['Netherlands']['Long'] = confirmed[confirmed['Province/State'] == 'Netherlands']['Long']

# Use groupby to sum the number of cases for countries that are listed with more than one territory, then␣
↪drop the lat/long columns
# .T to transpose from wide to long

confirmed = confirmed.groupby('Country/Region').sum().drop(['Lat', 'Long'], axis=1).T
recovered = recovered.groupby('Country/Region').sum().drop(['Lat', 'Long'], axis=1).T
deaths = deaths.groupby('Country/Region').sum().drop(['Lat', 'Long'], axis=1).T

# Convert the indices to datetime
confirmed.index = pd.to_datetime(confirmed.index)
recovered.index = pd.to_datetime(recovered.index)
deaths.index = pd.to_datetime(deaths.index)
```

Next, we can convert our dataset into a single multi-index dataframe for ease of access.

[4]:
```
covid_cases = pd.concat([confirmed, recovered, deaths], axis=1, keys=['confirmed', 'recovered', 'deaths']).
↪sort_index(axis=1)
```

Finally, we need to deal with any missing values. As we can see below, there are currently no missing values. This may not always be the case. Given that we are dealing with time series data, interpolation is a reasonable strategy. We would not expect the number of cases to significantly deviate from a relatively linear pattern between two known datapoints.

[5]:
```
num_missing = covid_cases.isna().to_numpy().sum()

print(f"There are {num_missing} missing datapoints.")
```

```
There are 0 missing datapoints.
```

[6]:
```
# Fill up to 5 missing values between two known values

covid_cases.interpolate(method='time', axis=0, limit=5, limit_area = 'inside', inplace=True)
```

3 Descriptive analysis and initial data exploration

Initial analysis of our dataset should focus on understanding what it describes (i.e. what time period, how many countries, what datapoints are available) before embarking on our exploratory data analysis.

[7]:
```
## Set the most recent date in the dataset, this will change as more dates/data is added

most_recent_date = covid_cases.index[-1]
first_date = covid_cases.index[0]

total_countries = (covid_cases.loc[most_recent_date]['confirmed']).shape[0]

print(f"The data available on reported COVID cases extends from {first_date.date()} to {most_recent_date.
 ↪date()}.")
print(f"A total of {total_countries} countries have reported data on COVID cases.")
```

```
The data available on reported COVID cases extends from 2020-01-22 to
2020-03-19.
A total of 155 countries have reported data on COVID cases.
```

3.1 How many countries have reported at least 10 cases?

[8]:
```
## Select confirmed cases for all countries and the most recent date in the dateime index, this will work␣
 ↪even when more dates are added
## Check to see if there are more than 10 cases (forms boolean series), then add them together

no_countries_over10 = (covid_cases.loc[most_recent_date]['confirmed'] > 10).sum()

print(f"{no_countries_over10} countries have reported more than 10 cases out of a total of␣
 ↪{total_countries} countries reporting data.")
```

```
110 countries have reported more than 10 cases out of a total of 155 countries
reporting data.
```

3.2 What are the five countries with the highest number of active cases?

Active cases are an important distinction from the pure number of cases diagnoised - they describe the number of patients who are currently unwell with COVID-19 and at risk of hospitalisation or even death. Additionally, as cases recover or die, the number of active cases is able to provide insights into how well a particular countries outbreak is controlled.

[9]:
```
## Add a separate multi-indexed column for active cases for each country, this is easiest done with stack/
 ↪unstack

covid_cases = covid_cases.stack()
covid_cases['active'] = np.nan
covid_cases = covid_cases.unstack()

## Calculate active cases by active = confirmed - (recovered + deaths)
covid_cases['active'] = covid_cases['confirmed'] - (covid_cases['recovered'] + covid_cases['deaths'])
```

[10]:
```
print("The five countries with the highest number of active cases are:")
for count, i in enumerate(covid_cases.loc[most_recent_date]['active'].T.sort_values(ascending=False)[:5].
 ↪iteritems()):
    print(f'{count+1}. {i[0]} with {i[1]} active cases.')
```

```
The five countries with the highest number of active cases are:
1. Italy with 33190 active cases.
2. Spain with 16026 active cases.
3. Germany with 15163 active cases.
4. US with 13477 active cases.
5. Iran with 11413 active cases.
```

3.3 What is the current rate of increase in the total number of cases, based on the last week of data?

[11]:
```python
# Calculate the weekly rate of change over the last week of data

week_change = covid_cases['confirmed'].iloc[-7:].sum(axis=1)[6] - covid_cases['confirmed'].iloc[-7:].
↪sum(axis=1)[0]

print(f"In the last week of data, the number of confirmed cases has increased globally by {week_change}.")
```

```
In the last week of data, the number of confirmed cases has increased globally
by 97515.
```

4 Data normalisation

Normalisation of data between countries is important to enable meaningful analysis as not all countries have the same population or began their COVID-19 outbreak at the same time.

We can normalise the number of cases per million population in a country, we can do this using the population data provided in the archive.

Additionally, we can examine timing of the outbreak in various countries by counting the days since they first reached 10 confirmed cases.

[12]:
```python
### Add the required columns to out dataframe

covid_cases = covid_cases.stack()
covid_cases['confirmed_per_million'] = np.nan
covid_cases['recovered_per_million'] = np.nan
covid_cases['deaths_per_million'] = np.nan
covid_cases['active_per_million'] = np.nan
covid_cases['days_since_ten'] = np.nan
covid_cases = covid_cases.unstack()
```

[13]:
```python
# Not every country as their populaton reported at the same time (see eritrea in csv), although every␣
↪country in the current dataset has a 2016 population recorded
# Not ever country (e.g. Cruise Ship and Holy See) have a population
# Our loop needs to be able to handle this

country_list = covid_cases['active'].columns

for i in country_list:

    # Calculate cases per million for each country
    try:
        code = country_codes[country_codes.Country == i]['Country_Code'].iloc[0]
        country_populations = populations[populations.Country_Code == code].T
        millions = country_populations.loc[country_populations.last_valid_index()].item() / 1000000
    except Exception as e:
        print(e)
        print("Couldn't find population for " + i + " filling with NaN")
        millions = np.nan

    covid_cases.loc[:, ('confirmed_per_million', i)] = covid_cases.loc[:, ('confirmed', i)] / millions
    covid_cases.loc[:, ('recovered_per_million', i)] = covid_cases.loc[:, ('recovered', i)] / millions
    covid_cases.loc[:, ('deaths_per_million', i)] = covid_cases.loc[:, ('deaths', i)] / millions
    covid_cases.loc[:, ('active_per_million', i)] = covid_cases.loc[:, ('active', i)] / millions

    # Now calculate the days since 10 confirmed cases were reported

    mask = covid_cases['confirmed'][i] >= 10 # Create a boolean mask
    idx = next(iter(mask.index[mask]), np.nan) # Get the index (date) of the first true value in the boolean␣
↪mask
    if pd.isnull(idx):
        covid_cases.loc[:,('days_since_ten', i)] = np.nan # If a country never reached 10 cases␣
↪days_since_ten = np.nan
    else:
        covid_cases.loc[:,('days_since_ten', i)] = (covid_cases['confirmed'][i].index - idx).days #␣
↪Subtract the datetime index from all the dates to give us "day since"
```

```
None
Couldn't find population for Cruise Ship filling with NaN
None
Couldn't find population for Holy See filling with NaN
```

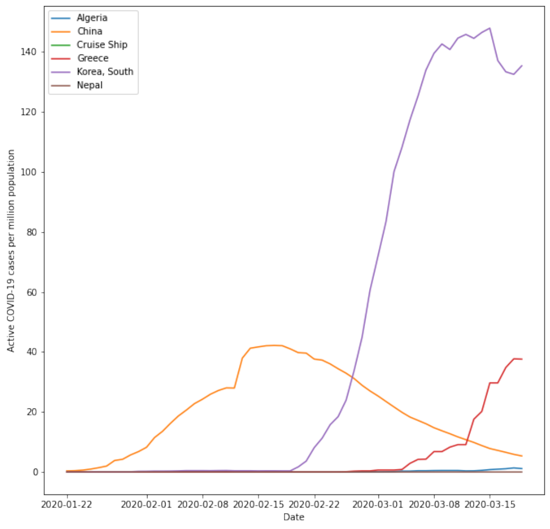
5 Further data exploration

5.1 Which countries appear to be past the peak of their local outbreak?

[14]:
```
# List the countires had their highest daily active cases prior to the end date of the data

past_peak = []

for i in country_list:
    if covid_cases['active'][i][::-1].idxmax() < most_recent_date: # Find the date that the highest active␣
↪cases occurred on and check if it was before the latest date
        print(i)
        past_peak.append(i)
```

```
Algeria
China
Cruise Ship
Greece
Korea, South
Nepal
```

We can further examine these countries visually to see which are ”past the peak”.



Following visual inspection:

- China and South Korea appear to be definitively passed their peak

- Greece may have passed their peak, but it’s difficult to definitively say

- Algeria and Nepal both have small case numbers, which make it difficult to interpret the peak

- Cruise Ships are not a country

5.2 What can you say about how long it takes for the outbreak to peak?

We can only confidently identify the peak with reasonable confidence in China, South Korea and Greece.

[16]:
```
peaked = ['China', 'Korea, South', 'Greece']

days_to_peak = []

for i in peaked:
    peak_idx = covid_cases['active'][i][::-1].idxmax()
    days_to_peak.append(covid_cases.loc[peak_idx, ('days_since_ten', i)])

mean_days_to_peak = np.mean(days_to_peak)
std_days_to_peak = np.std(days_to_peak)

print(f"Of the three countries to have peaked the peak occured {np.round(mean_days_to_peak, 2)} ± {np.
↪round(std_days_to_peak, 2)} days after reaching 10 confirmed cases.")
```

```
Of the three countries to have peaked the peak occured 27.67 ± 12.71 days after
reaching 10 confirmed cases.
```

5.3 Based on the avilable data, can you estimate how long it takes a patient to recover? Does this vary by region or country? How confident can you be about these results?

As we do not have individual case level data, it is difficult to confidently estimate the time to recovery. We can however confidently identify when a country first reported a COVID case and when it first reported a recovered case. Using this data, we can roughly estimate the time to recovery of the first case. There are significant confounders with this approach - recovered cases may not have been reported promptly at the start of the pandemic, the first recovered case may not have actually been the first confirmed case, the first confirmed case may date of reporting may not reflect the actual date of diagnosis, and it does not take into account patients who die. A more robust analysis could aim to approximate matching between recoveries, deaths and confirmed cases throughout the time series. To achieve a firm estimate however, individual case level data would be required.

[17]:
```
first_recovery = pd.DataFrame(index = covid_cases.confirmed.columns, columns=['time_to_first_recovery'])

for country in covid_cases.confirmed.columns:
    temp_conf = covid_cases.confirmed[country].replace(0, np.nan)
    temp_recov = covid_cases.recovered[country].replace(0, np.nan)
    if temp_conf.loc[temp_conf.first_valid_index()] < 10: # Skip countries that had significant cases prior␣
↪to data availability
        first_reported = temp_conf.first_valid_index()
        first_recovered = temp_recov.first_valid_index()

        if pd.notnull(first_recovered):
            first_recovery.loc[country, 'time_to_first_recovery'] = (first_recovered - first_reported).days


first_recovery.time_to_first_recovery = first_recovery.time_to_first_recovery.astype('float')
```

79 countries had reported their first case(s) during this dataset and subsequently reported a recovery. The mean time to recovery (for the first confirmed case) was 12.6 days, with a range of of 4 to 38 days and a standard deviation of 5.27 days. This indicates that there is significant variation between time to recovery between countries. Given the confounders listed above, the certainty about this estimation is low. It is even lower given the mean time to recovery is lower than most definitions of recovery (14 days without an intervening hospitalisation in the absence of a negative rt-PCR test).

5.4 Mitigation strategies are aimed at 'flattening' the outbreak, to reduce the strain on the health system. We discuss this in more detail in the concluding remarks below. For countries or regions that are far enough into their local outbreak, consider the number of active cases relative to the number of confirmed cases. Can you say anything about the effectiveness of their mitigation strategies?

Early in the reporting of COVID-19 cases, we would expect the ratio of active cases to confirmed cases to be equal (1:1), then, as patients recover or die this ratio will gradually decline. If no new cases were identified then this ratio will eventually drop to zero. Conversely, if there is a high burden of new active cases identified this ratio will remain relatively high. Examining the change in this ratio over time may provide insights into the relative effectiveness of a countries mitigation strategies. More effective mitigation will result in a greater and more rapid reduction active cases compared to confirmed cases.

As all countries will start with an active:confirmed ratio of 1 and there is a degree of noise in the reporting of recoveries and deaths, we will examine this ratio based on a 7-day rolling average.

It is also important to examine the rate of increase in number of confirmed cases to provide context to the ratio of active to confirmed cases.

[19]:
```
# Add column for active / confirmed ratio + rolling average

covid_cases = covid_cases.stack()
covid_cases['active_confirmed_ratio'] = np.nan
covid_cases['active_confirmed_ratio_7day_rolling'] = np.nan
covid_cases = covid_cases.unstack()

# Calculate the ratio of active cases per confirmed cases
covid_cases['active_confirmed_ratio'] = covid_cases['active'] / covid_cases['confirmed']
covid_cases['active_confirmed_ratio_7day_rolling'] = covid_cases['active_confirmed_ratio'].rolling(7).mean()
```

To attempt to gauge the effectiveness of mitigation strategies, there needs to have been enough elapsed time for them to have an impact and enough overall confirmed cases the show a reasonable effect size. Accounting for this, we have selected countries which have reported at least 30 days of data since their 10th case and have had at least 250 cases overall.

[20]:
```
# Create a list of countries that have 30 days of data since 10 cases and at least 250 reported cases
at_least_30_days = []

for country in covid_cases['days_since_ten'].columns:
    if covid_cases.loc[most_recent_date, ('confirmed', country)].item() > 250:
        if covid_cases.loc[most_recent_date, ('days_since_ten', country)].item() > 30 and country !=␣
↪'Cruise Ship':
            at_least_30_days.append(country)

print(f"There are {len(at_least_30_days)} countries with 30 days of data after their 10th case and more␣
↪than 250 total cases reported.")
```

```
There are 10 countries with 30 days of data after their 10th case and more than
250 total cases reported.
```

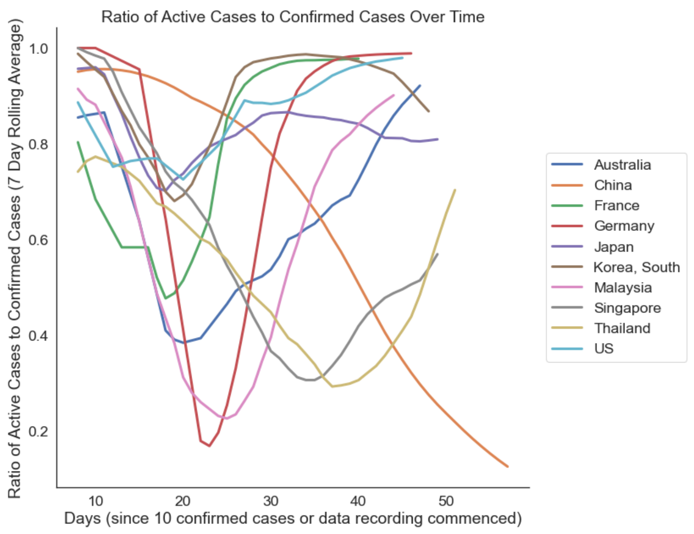
We can now compare the most recent 7-day rolling average ratio of active to confirmed cases for the countries identified.

[21]:
```
# Examine the change in this ratio over the 20 days since 10 cases
# As we are looking at the 7 day rolling mean, the first value will begin on day 7 following the first 10␣
↪cases

ratio_df = pd.DataFrame(index=at_least_30_days, columns = ['latest_ratio'])

for country in at_least_30_days:
    country_df = covid_cases.loc[:, (slice(None), country)].droplevel(axis=1,level=1).
↪set_index('days_since_ten')
    most_recent_ratio = country_df['active_confirmed_ratio_7day_rolling'].iloc[-1]
    ratio_df.loc[country, 'latest_ratio'] = most_recent_ratio

print("\nThe three countries with the most suppression of active cases compared to confirmed cases are (by␣
↪most recent 7 day average ratio only):")
for count, i in enumerate(ratio_df['latest_ratio'].sort_values(ascending=True)[:3].iteritems()):
    current = covid_cases.loc[most_recent_date, ('confirmed_per_million', i[0])]
    past_week = covid_cases.loc[(most_recent_date - timedelta(days=7)), ('confirmed_per_million', i[0])]
    change = np.round(current - past_week,2)
    print(f'{count+1}. {i[0]} with ratio of {np.round(i[1],2)} active cases to confirmed cases and an␣
↪increase of {change} per million cases in the last week.')

print("\nThe three countries with the least suppression of active cases compared to confirmed cases are (by␣
↪by most recent 7 day average ratio only):")
for count, i in enumerate(ratio_df['latest_ratio'].sort_values(ascending=False)[:3].iteritems()):
    current = covid_cases.loc[most_recent_date, ('confirmed_per_million', i[0])]
    past_week = covid_cases.loc[(most_recent_date - timedelta(days=7)), ('confirmed_per_million', i[0])]
    change = np.round(current - past_week,2)
    print(f'{count+1}. {i[0]} with ratio of {np.round(i[1],2)} active cases to confirmed cases and an␣
↪increase of {change} per million cases in the last week.')
```

```
The three countries with the most suppression of active cases compared to
confirmed cases are (by most recent 7 day average ratio only):
1. China with ratio of 0.12 active cases to confirmed cases and an increase of
0.16 per million cases in the last week.
2. Singapore with ratio of 0.57 active cases to confirmed cases and an increase
of 29.78 per million cases in the last week.
3. Thailand with ratio of 0.7 active cases to confirmed cases and an increase of
2.93 per million cases in the last week.

The three countries with the least suppression of active cases compared to
confirmed cases are (by by most recent 7 day average ratio only):
1. Germany with ratio of 0.99 active cases to confirmed cases and an increase of
160.18 per million cases in the last week.
2. US with ratio of 0.98 active cases to confirmed cases and an increase of
37.18 per million cases in the last week.
3. France with ratio of 0.98 active cases to confirmed cases and an increase of
129.41 per million cases in the last week.
```

There appears to be a drastic differences in the degree of suppresion achieved between the countries identified, despite all being at least 30 days into their outbreak and the ratio being independent of population. Adding validity to the ratio on active:confirmed cases as a marker of outbreak suppression, countries with a lower ratio appear to have a markedly lower rate of increase in number of cases. However, it is important to note, that a single point in time measurement may not be the most accurate representation of this measure.

To explore this further, we need to examine the trends over time visually to enable more effective comparison between countries.



Examining the visual trend in the ratio of active to confirmed cases over time provides greater insight into the trajectory of each country. While some countries have achieved good levels of suppression to date, they are now experiencing a rise in active:confirmed cases, which may indicate mitigation measures are failing (e.g. Singapore and Thailand). Other countries have achieved a more sustained and gradual downward trend in their ratio (e.g. Japan and China). A caveat to the interpretation of this data is that most countries have not yet reached their peak and as such it is difficult to comment on mitigation strategies prolonging time to peak or reducing peak magnitude. It must also be noted, that data on China's outbreak is reported well after their first 10 cases.

Overall, it appears that a lower ratio of active:confirmed cases is indicative of more effective mitigation strategies and a slower rise in confirmed COVID-19 cases.

5.5 In epidemiology, the case fatality rate (CFR) is the ratio of deaths from a certain disease to the total number of people diagnosed with this disease. The formula is straightforward once an epidemic has ended. However, while an epidemic is still ongoing—as is the case with the COVID-19 outbreak—this formula can be misleading if, at the time of analysis, the outcome is unknown for a non-negligible proportion of patients. One alternative is to estimate CFR as deaths / (deaths + recovered). What kind of assumptions is that making? If you use this formula, what range of values do you get? Does this vary over time?

Utilising the formula deaths / (deaths + recovered) to estimate the case fatality rate (CFR) attempts to compensate for the unknown end point of cases by using only cases that have "completed" their illness (i.e. died or recovered). There are multiple assumptions made when using this method:

1. Cases die and recover at the same rate

- This is rarely the case. As such, if cases die faster than they recover the CFR will be overestimated. Conversely, if cases generally recover faster than they die, it will be underestimated.

2. Reporting of recoveries and deaths is consistent

- Not all countries regularly report recovered, but most report deaths, which may overestimate the CFR

3. Definition of recovery is consistent

- Not all countries define "recovery" in the same manner

- Two weeks post positive PCR without hospitalisation vs repeat negative PCR testing

4. The population tested does not vary over time

- Early in the pandemic many countries only carried out "targetted" testing of patients with risk factors or those on hospital, rather than widespread community testing. This may result in the cases being detected early in the pandemic being more severe (more likely to be hospitalised) and more likely to die. While cases detected after the commencement of widespread testing are less likely to have been detected in hospitalised patients, thus reducing the incidence of mortality.

To attempt to maintain a degree of robustness, we will only analyse countries that have reported at least 50 confirmed COVID-19 cases.

[24]:
```
# Make a list of all countries that have more than 50 cases to date

at_least_50 = []

for country in covid_cases['confirmed'].columns:
    if covid_cases.loc[most_recent_date, ('confirmed', country)].item() > 50:
        at_least_50.append(country)
```

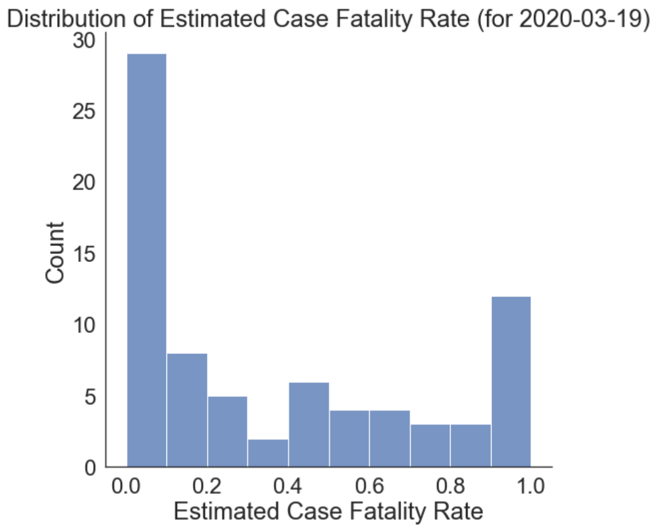
[25]:
```
total_reporting = covid_cases['cfr_estimate'][at_least_50].iloc[-1].notna().sum()

print(f"As of the most recently reported data there are {total_reporting} countries with >50 confirmed␣
↪cases in which the estimated CFR can be calculated.")
```

```
As of the most recently reported data there are 76 countries with >50 confirmed
cases in which the estimated CFR can be calculated.
```

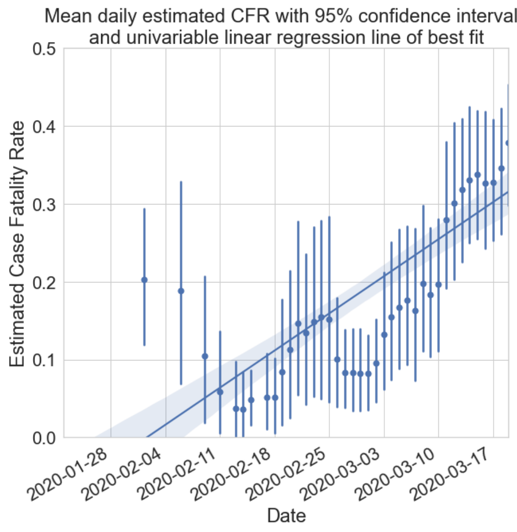
We can review the distribution of the estimated CFR (by country identified above) with summary statistics as well as visually using a histogram.

[26]:
```
# Summary statistics of the estimated CFR

covid_cases['cfr_estimate'][at_least_50].iloc[-1].describe()
```

[26]:
```
count    76.000000
mean      0.358562
std       0.374759
min       0.000000
25%       0.000000
50%       0.205263
75%       0.683611
max       1.000000
Name: 2020-03-19 00:00:00, dtype: float64
```

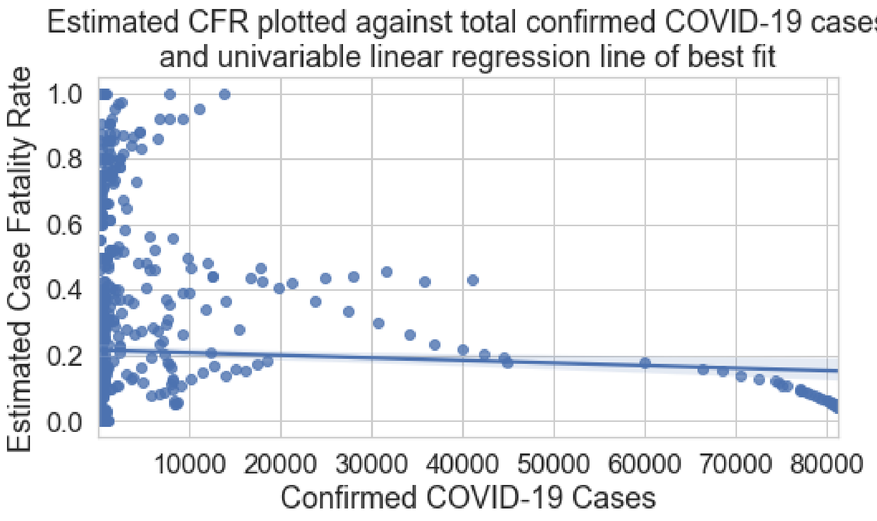


Examining the most recent date in the dataset provided, we can see a mean estimated CFR of 35.9% with a broad distribution and many countries reporting values of 0% or 100%. This is considerably higher than the currently reported case fatality rate, which ranges between 0.5% and 5% globally (ref: https://coronavirus.jhu.edu/data/mortality). This is likely indicative of this method violating the first two assumptions mentioned above. Many countries have not reported any deaths compared to recoveries, while other have reported only deaths and no recoveries. In reality, this is unlikely to be the case. This is potentially related to differing lag times in reporting recovering and deaths and potentially differing definitions for recovery reporting by country. As the data analysed is early in the global COVID-19 pandemic, a large proportion of the cases are likely to have been detected in hospitalised patients, significantly increasing their risk of mortality. To further investigate this, it is important to examine the estimated CFR over time.



There is significant temporal change in the average estimated CFR, with a trend toward increase over time. Explanations for this include early reporting of deaths (out of proportion with recovered cases) and the detection of more severely unwell cases due to the early timing of the dataset during the pandemic.

5.6 With a disease like COVID-19 where the vast majority of cases are mild or even asymptomatic, the number of confirmed cases is going to be highly dependent on the testing strategy. Do you see any relationship between the number of cases and your estimated CFR values?



Examining the plot above, we can see that with fewer total confirmed cases reported the spread of the estimated CFR is very broad. However, with increasing confirmed cases reported we see a trend toward reducing estimated CFR. This is likely due to greater detection of mild and potentially asymptomatic cases, rather than the most unwell patients who are hospitalised with COVID-19.

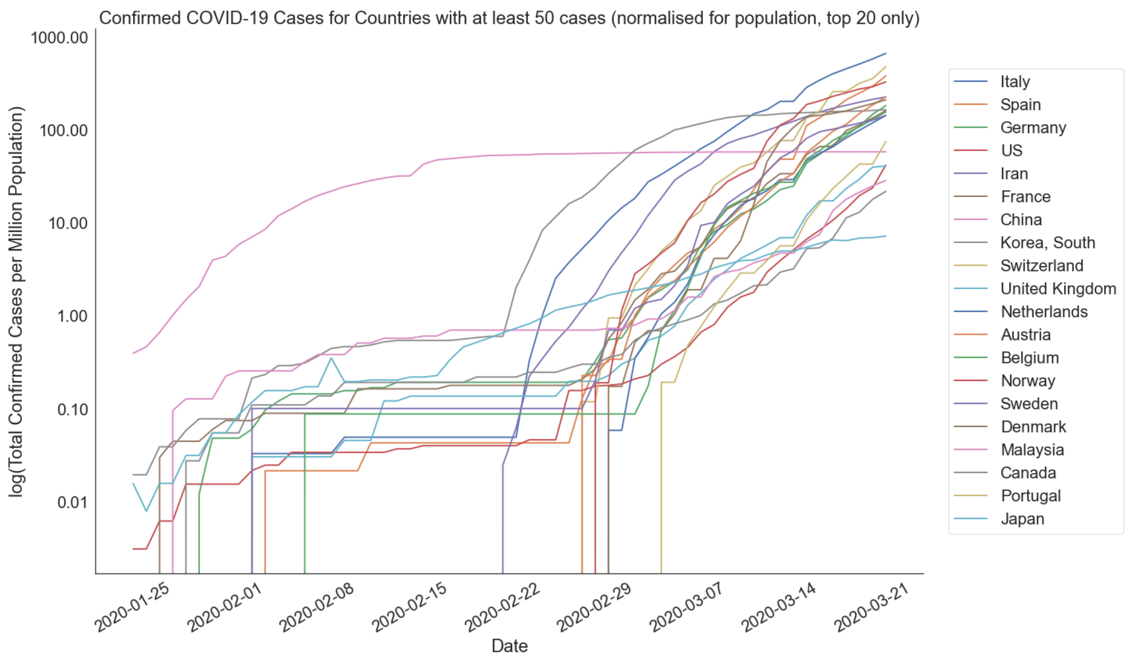
6 Data visualisation

6.1 For countries with at least 50 confirmed cases, plot the progression of the virus starting from day 0. Be mindful of the best ways to visualise this—normalised data, linear or log scale, etc.? Also pay attention to the specific parameters you use for creating your plots (remembering that default values are rarely the best choice).

There are a large number of countries which have reported more than 50 cases to date. Plotting them together results in a cluttered plot, to limit this only 20 countries with the highest number of confirmed cases are visualised.. In order to aid in interpretability between countries, the number of confirmed cases has been represented per million population (for each country). Given the significant variance in number of cases reported, this has been logarithmically scaled.

[30]:
```python
from matplotlib.ticker import FormatStrFormatter
# Make a list of all countries that have more than 50 cases to date

at_least_50 = []

for country in covid_cases['confirmed'].columns:
    if covid_cases.loc[most_recent_date, ('confirmed', country)].item() > 50:
        at_least_50.append(country)

top_20 = covid_cases.loc[most_recent_date]['active'][at_least_50].T.sort_values(ascending=False)[:20].index

# Plot each country that has more than 50 cases to date
```

```
sns.set(style='white')
sns.set_context('notebook', font_scale=2)

fig, ax = plt.subplots(figsize=(20,15))

for country in top_20:
    # Temporary DF of only confirmed cases per million for each country
    to_plot = covid_cases.loc[:,(['confirmed_per_million'], country)].swaplevel(axis=1).droplevel('Country/
↪Region', axis=1)
    ax.plot(to_plot, linewidth=2, label=country)

# Log scale for the y axis (makes it more interpretable between countries early in the pandemic)
ax.set_yscale('log')
ax.yaxis.set_major_formatter(FormatStrFormatter('%.2f'))

# Title and labels
ax.set_ylabel("log(Total Confirmed Cases per Million Population)")
ax.set_xlabel("Date")

ax.set_title("Confirmed COVID-19 Cases for Countries with at least 50 cases (normalised for population, top␣
↪20 only)")

# Legend
ax.legend(ncol=1, loc='center left', bbox_to_anchor=(1.02, 0.5))

# Major ticks every 7 Days.
fmt_week = mdates.DayLocator(interval=7)
ax.xaxis.set_major_locator(fmt_week)

#Make the date ticks look right
fig.autofmt_xdate()

# Aesthetics
sns.despine()

plt.show()
```



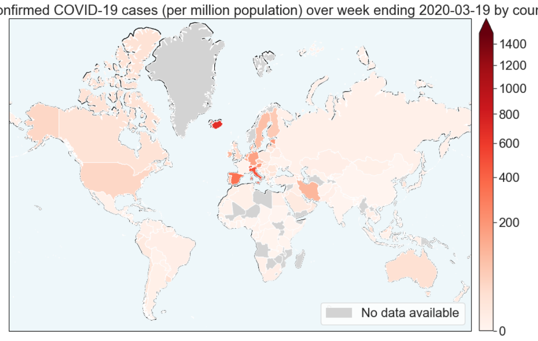
6.2 Optional: For each of the last five weeks, create a global map showing the rate of increase in the number of confirmed cases over that week. Create similar maps for the rate of increase in the number of active cases.

Utilising freely available geographic data, we are able to visualise change in confirmed COVID numbers on a map. This enables regional comparison of the rate of COVID growth. Plotting several sequential charts also enables temporal comparisons.

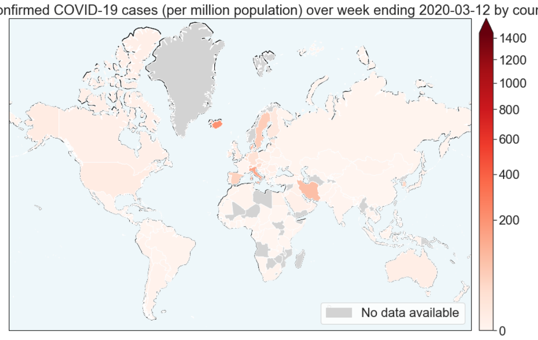
[31]:
```
projection = ccrs.Mercator()
axes_class = (GeoAxes,
              dict(map_projection=projection))

fig = plt.figure(figsize=(15,60))

# Create and axes grid for each of the 5 maps (one for each week of data)
axgr = AxesGrid(fig, 111, axes_class=axes_class,
                nrows_ncols=(5, 1),
                axes_pad=0.6,
                cbar_location='right',
                cbar_mode='each',
                cbar_pad=0.2,
                cbar_size='3%',
                label_mode='')  # note the empty label_mode

# Download the shape/polygon data for each country from the natural earth database
shpfilename = shapereader.natural_earth(resolution='50m',
                                        category='cultural',
                                        name='admin_0_countries')
reader = shapereader.Reader(shpfilename)

# Iterate through each subplot
for i, (ax, cax) in enumerate(zip(axgr, axgr.cbar_axes)):

    # Generate the list of country data (unfortunately this seems to need to be generated each iteration)
    countries = reader.records()

    # Add the base map features
    ax.add_feature(cfeature.COASTLINE)
    ax.add_feature(cfeature.BORDERS, linestyle=':', alpha=1)
    ax.add_feature(cfeature.OCEAN, facecolor=("lightblue"), alpha=0.2)
    ax.set_extent([-180,180,-60,86], crs=ccrs.PlateCarree())
    ax.coastlines()

    #Create a DF for the week being plotted to store the absolute change in cases
    week_change = pd.DataFrame(index=country_codes.Country_Code.unique(), columns=['Country',␣
↪'week_change'])

    # Calculate the change in cases for the week being plotted
    for country in covid_cases['confirmed'].columns:
        code = country_codes[country_codes.Country == country].Country_Code.iloc[0]
        week_end = most_recent_date - timedelta(days=(7*i))
        week_start = week_end - timedelta(days=7)
        week_end_confirmed = covid_cases.loc[week_end, ('confirmed_per_million', country)].item()
        week_start_confirmed = covid_cases.loc[week_start, ('confirmed_per_million', country)].item()
        week_change.loc[code, 'week_change'] = (week_end_confirmed - week_start_confirmed)

    # Set the normalisation function for the week change data and add a column of normalised values
    # This will allow us to plot each countries colour in a reproducible way and have a scaled colourbar on␣
↪the side
    norm = mpl.colors.PowerNorm(vmin=np.nanmin(week_change.week_change), vmax=np.nanmax(week_change.
↪week_change), gamma=0.5)
    week_change['week_change_norm'] = norm(week_change.week_change)

    # Choose your colourmap
    cmap = mpl.cm.get_cmap('Reds')
    ax.set_title(f"Change in confirmed COVID-19 cases (per million population) over week ending {week_end.
↪date()} by country")

    # Iterate through each country and if we have data for it, add the geometry to the map and colour it␣
↪appropriately
    for country in countries:
        if country_codes.Country_Code.str.contains(country.attributes['ISO_A3_EH']).any():
            code = country.attributes['ISO_A3_EH']
            if pd.notna(week_change.loc[code, 'week_change']):
                rgba = cmap(week_change.loc[code,'week_change_norm'])
                ax.add_geometries([country.geometry], ccrs.PlateCarree(), facecolor=rgba)
            else: # If null value in week change then fill country with gray
                ax.add_geometries([country.geometry], ccrs.PlateCarree(), facecolor='lightgrey')
        else: # If no matching country code then fill country with gray
            ax.add_geometries([country.geometry], ccrs.PlateCarree(), facecolor='lightgrey')

    # Scale our colourbar appropriately for each week of data
    sm = plt.cm.ScalarMappable(cmap='Reds', norm=norm)
    sm._A = []
    cb = cax.colorbar(sm, extend='max')

    # Add a legend demonstrating that gray means no data
    grey_patch = mpatches.Patch(color='lightgrey', label='No data available')
    ax.legend(handles=[grey_patch], loc='lower right')

plt.show()
```

Utilising similar code to above, we are also able to examine the change in active cases regionally and temporally. Of note, active cases have the potential for negative change which is reflected on each subplots colourbar. Notably, this plot could be improved with better colour grading for falling active cases and colourbar scaling across weeks.

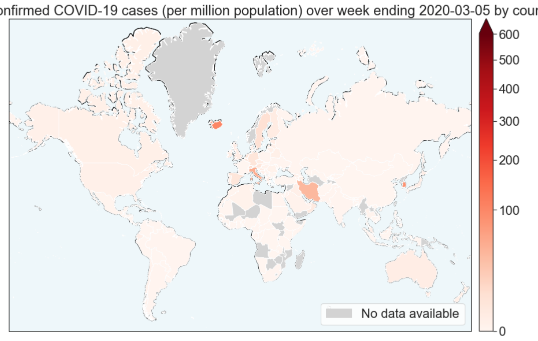
Change in confirmed COVID-19 cases (per million population) over week ending 2020-03-19 by country



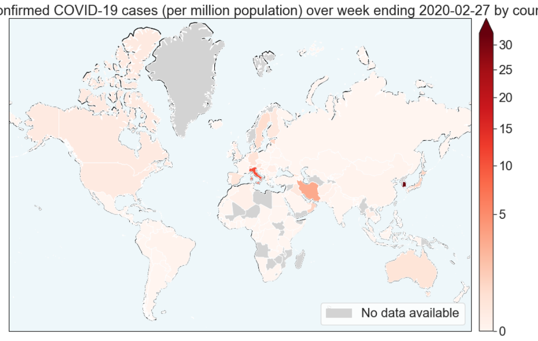
Change in confirmed COVID-19 cases (per million population) over week ending 2020-03-12 by country



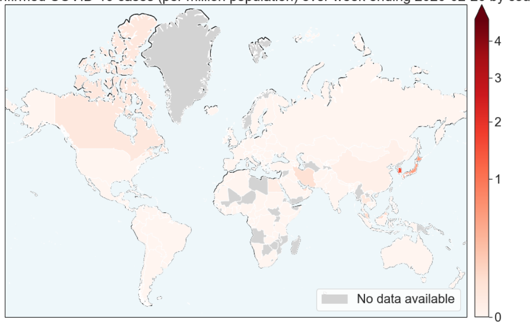
Change in confirmed COVID-19 cases (per million population) over week ending 2020-03-05 by country



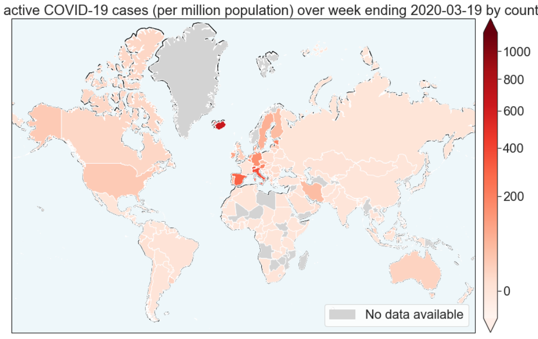
Change in confirmed COVID-19 cases (per million population) over week ending 2020-02-27 by country



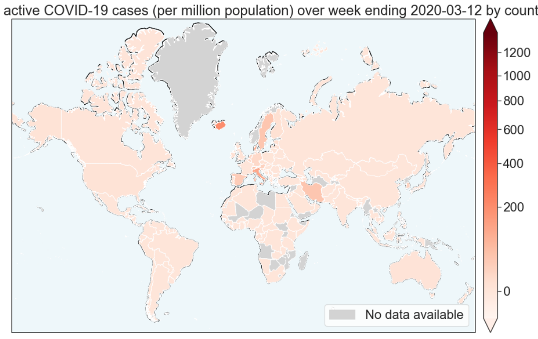
Change in confirmed COVID-19 cases (per million population) over week ending 2020-02-20 by country



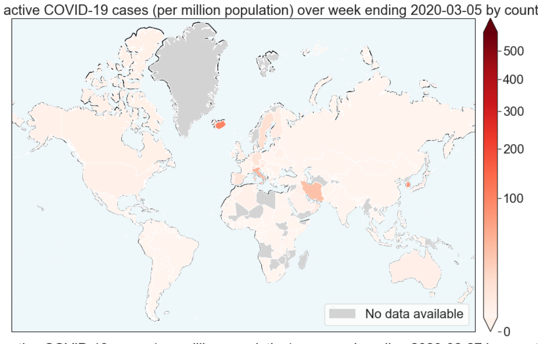
Change in active COVID-19 cases (per million population) over week ending 2020-03-19 by country



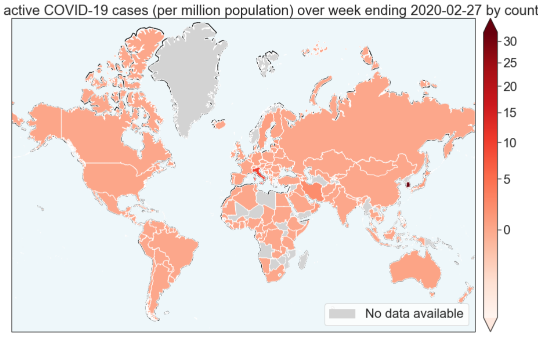
Change in active COVID-19 cases (per million population) over week ending 2020-03-12 by country



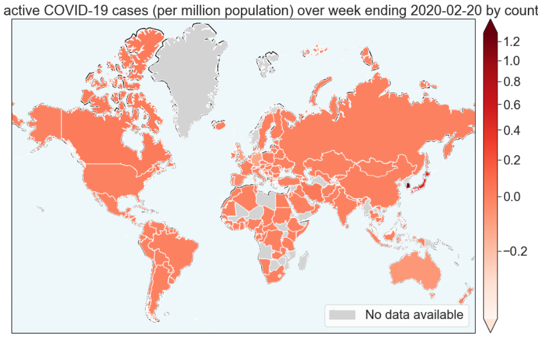
Change in active COVID-19 cases (per million population) over week ending 2020-03-05 by country



Change in active COVID-19 cases (per million population) over week ending 2020-02-27 by country

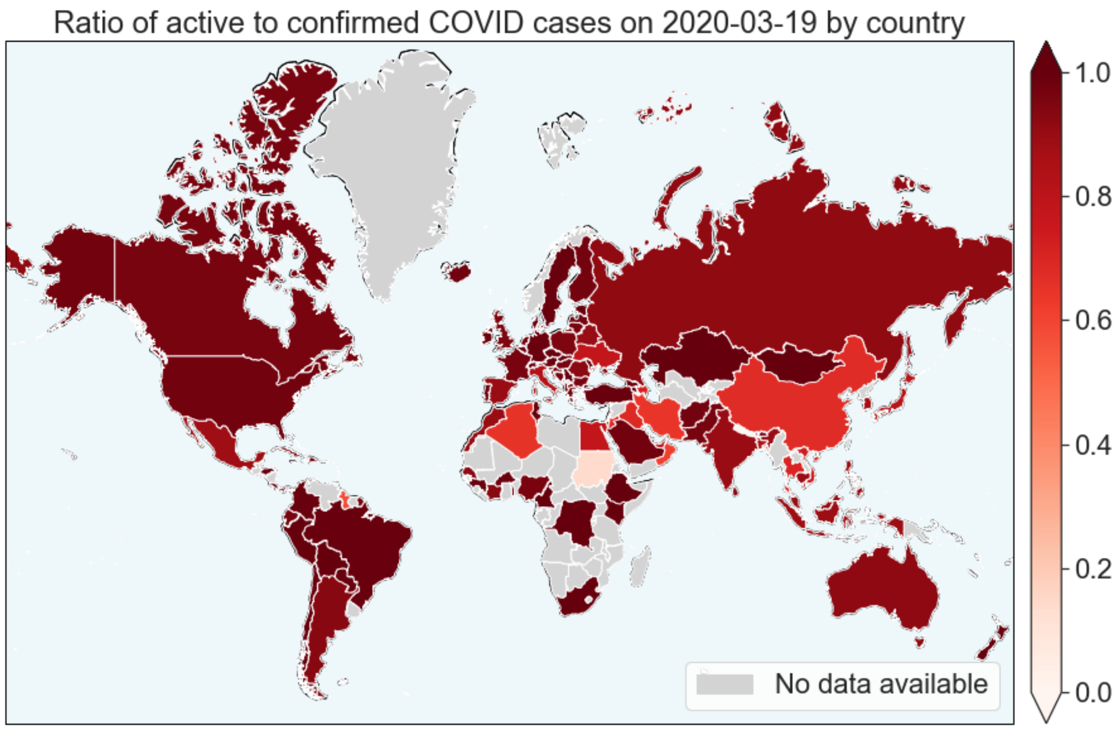


Change in active COVID-19 cases (per million population) over week ending 2020-02-20 by country



6.3 Optional: Create a map to visualise the results of your 'mitigation' analysis (fourth question of the previous section).

We are also able to visualise how effectively a country is suppressing the growth in active cases (mitigating their outbreak) by demonstrating the the active:confirmed ratio for our most recent day of data by country. This enables a global overview of how effectively different countries and regions are dealin with their outbreak.



Ratio of active to confirmed COVID cases on 2020-03-19 by country
No data available
1.0
0.8
0.6
0.4
0.2
0.0

7 Concluding remarks

With publicly available data, we have been able to conduct a thorough exploratory analysis of the start of the COVID-19 pandemic. In addition to this, we have been able to estimate key parameters of the disease (time to recovery, time to peak) as well as countries response to it (active:confirmed ratio and mitigation effects). In addition to this, our analysis has been structured in a manner that it can be continually repeated as new data becomes available. It is also robust to missing data.