```
# As usual, load needed packages and methods
from datascience import *
import matplotlib
matplotlib.use('Agg', warn=False)
%matplotlib inline
import matplotlib.pyplot as plots
plots.style.use('fivethirtyeight')
import numpy as np
import pandas as pd
np.set_printoptions(threshold=50)
import folium
```

```
/usr/local/lib/python3.7/dist-packages/datascience/tables.py:17: MatplotlibDeprecat
  matplotlib.use('agg', warn=False)
/usr/local/lib/python3.7/dist-packages/datascience/util.py:10: MatplotlibDeprecatio
  matplotlib.use('agg', warn=False)
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: MatplotlibDeprecati
  after removing the cwd from sys.path.
```

The Center for at Johns Hopkins University maintains a dashboard of coronavirus cases at https://www.arcgis.com/apps/opsdashboard/index.html#/bda7594740fd40299423467b48e9ecf6. The data behind this dashboard are available at a GitHub repository that is updated regularly. **It is extremely important to remember that this dataset, like all datasets, is only as good as the methodology used to compile it. I encourage you to reflect on the many reasons this particular dataset is not accurate.** (You don't have to include these reflections in your writeup.)

```
# Read data from gitHub repository maintained by Johns Hopkins researchers
path = 'https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_da
confirmed = Table.read_table(path+'confirmed_global.csv')
confirmed_US = Table.read_table(path+'confirmed_US.csv')
deaths = Table.read_table(path+'deaths_global.csv')
deaths_US = Table.read_table(path+'deaths_US.csv')
recovered = Table.read_table(path+'recovered_global.csv')

confirmed.show(20)
```

Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20
nan	Afghanistan	33.9391	67.71	0	0	0	0
nan	Albania	41.1533	20.1683	0	0	0	0
nan	Algeria	28.0339	1.6596	0	0	0	0
nan	Andorra	42.5063	1.5218	0	0	0	0
nan	Angola	-11.2027	17.8739	0	0	0	0
nan	Antigua and Barbuda	17.0608	-61.7964	0	0	0	0
nan	Argentina	-38.4161	-63.6167	0	0	0	0
nan	Armenia	40.0691	45.0382	0	0	0	0
Australian Capital Territory	Australia	-35.4735	149.012	0	0	0	0
New South Wales	Australia	-33.8688	151.209	0	0	0	0
Northern Territory	Australia	-12.4634	130.846	0	0	0	0

Part 1. Before looking at the data, address the following questions:

- Visit http://cidd.psu.edu/ and view some of the "AskCIDD" videos. **Insert a text box in which you describe two facts you did not know about coronavirus before watching the videos.**
- Read the article at https://medium.com/@noahhaber/flatten-the-curve-of-armchair-epidemiology-9aa8cf92d652. **Insert a text box in which you explain how this humorous article has serious implications for the ethical use of data science.**

You're welcome to write at whatever length you feel is appropriate to adequately address the questions. Roughly speaking, a sentence for each fact in part one and a few senteces for part two should be about right.

Facts: Moderna and Pfizer vaccines are mRNA vaccines and how they protect our bodies is by reconstructing the RNA in a safe way for our body to detect and fight COVID by an immunity response. An mRNA immunization conveys the steps for making a bacterial or viral protein to our cells. Our safe framework at that point reacts to these proteins and creates the apparatuses to respond to future contaminations with the pathogen. By providing genetic instructions, our body will hopefully be prepared for infectious diseases. Learning about how mRNA vaccines are constructed gives me a clearer understanding of what I am putting in my body and how it will protect citizens. Remdesivir, which was developed due to ebola, has been recently tested to see if it would be effective for covid-19. It is an ant-viral that stops the virus from spreading through a recipient's body. When testing this anti-viral, they split positive patients into two categories those who received Remdesivir and our placebo group (for an unbiased opinion). The anti-viral results shorten the duration of the sickness, lessened the severity, and reduced death rates. Prior to this video, I was not aware of Remdesivir and did not know such studies were being taken place.

Article: The article merges data science terminology with a humorous twist of the epidemic of 'bullshit.' Although technically, there is no firm measure of DKE, and the subject is highly subjective. Nonetheless, the author does incorporate graphs and suggestions to flatten the curve. Firstly there is no evidence to support that it affects men ages 24-36. Contrasting, the article discusses the exponential growth of

DKE-19 due to COVID-19, which the red trajectory can represent. The graph has also provided data under different scenarios to help the viewer visualize and compare the severity. But to reiterate, the article does not have reliable data to help support its claim and conduct the use of data science.

Part 2. Examine an case of exponential growth and use a least-squares line to estimate the growth rate

```
# Prepare the dataset
data = confirmed.to_df() # Convert to a dataframe object (in the "pandas" package)
data = pd.DataFrame(data.groupby("Country/Region").agg("sum"))
data = data.drop(columns = ["Lat", "Long"])
data = data.transpose()
data = Table.from_df(data) # Convert back to a Table object (in the "datascience" packag
data = data.with_column("days since 1/22/20", list(range(data.num_rows)))
data.show(10)
```

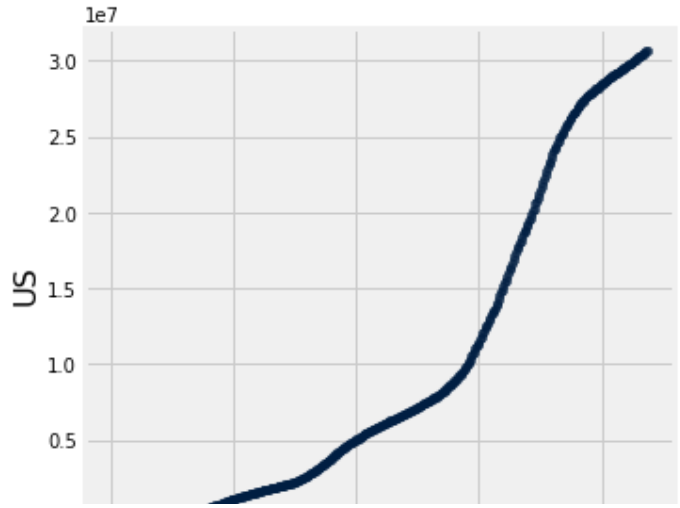
Afghanistan	Albania	Algeria	Andorra	Angola	Antigua and Barbuda	Argentina	Armenia	Australi
0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	

... (428 rows omitted)

```
# Examine the US column.  (You might want to look at various other countries too.)
# Notice that all US entries are nonzero, which is important for later when we take the 
# If there were zeros, we could either start with a later date or add a small value (lik
country = 'US'
data[country]
```

```
array([       1,        1,        2, ..., 30539868, 30609690, 30671844])
```

```
# Plot confirmed cases against time in days
data.scatter("days since 1/22/20", country)
```



The shape of the time series plot of confirmed cases above is characteristic of exponential growth. One way to express an exponential growth curve is

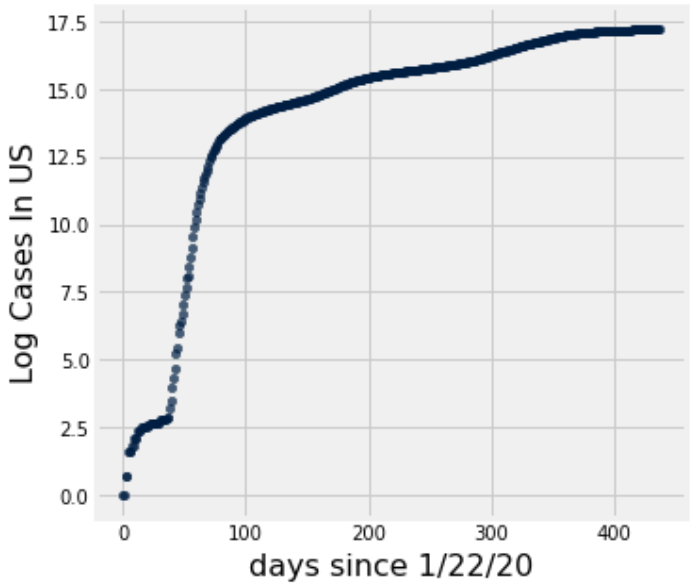
$$\text{cases at time } t = K \exp(rt),$$

where K is the number of cases at time $t = 0$ and r is the exponential rate of growth. If you take the natural logatirhm of both sides of this equation, you get

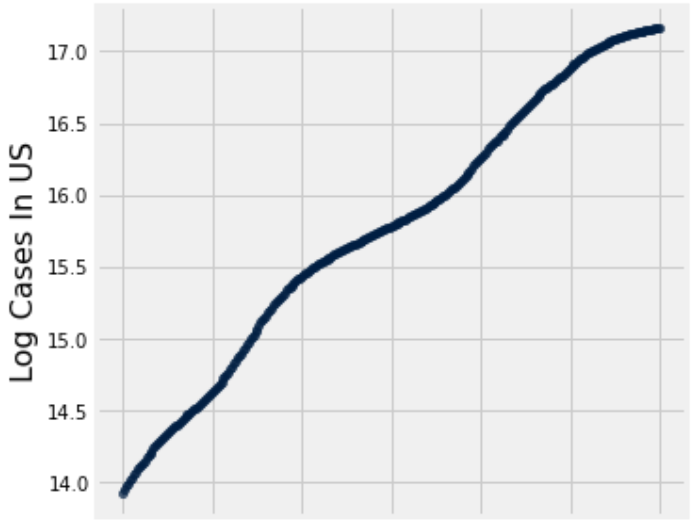
$$\log(\text{cases at time } t) = \log K + rt,$$

which is simply the equation for a line with slope equal to r and intercept equal to $\log K$. We'll exploit this fact by using a least-squares fit to these data to estimate the intercept and slope of the line relating $y = \log(\text{cases})$ to $x = \text{time}$.

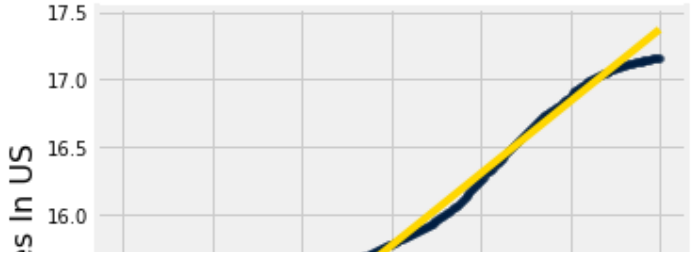
```
## Idea:  Use least squares as a way to fit a straight line (ignore the fact that regres
logCases = data.select('days since 1/22/20', country)
logCases = logCases.with_column('Log Cases In ' + country, np.log(logCases.column(countr
logCases.scatter("days since 1/22/20", 'Log Cases In ' + country)
```




```
# Focus on the period from day 100 to day 400
subset = logCases.where('days since 1/22/20', are.between(100, 401)) # went to 401, beca
subset.scatter("days since 1/22/20", 'Log Cases In ' + country)
```




```
# Use some of the functions defined in Chapter 15.
def standard_units(x):
    return (x - np.mean(x))/np.std(x)

def correlation(table, x, y):
    x_in_standard_units = standard_units(table.column(x))
    y_in_standard_units = standard_units(table.column(y))
    return np.mean(x_in_standard_units * y_in_standard_units)

def slope(table, x, y):
    r = correlation(table, x, y)
    return r * np.std(table.column(y))/np.std(table.column(x))

def intercept(table, x, y):
    a = slope(table, x, y)
    return np.mean(table.column(y)) -  a * np.mean(table.column(x))

def fit(table, x, y):
    a = slope(table, x, y)
    b = intercept(table, x, y)
    return a * table.column(x) + b

def scatter_fit(table, x, y):
    table.scatter(x, y, s=15)
    plots.plot(table.column(x), fit(table, x, y), lw=4, color='gold')
    plots.xlabel(x)
    plots.ylabel(y)



# What does the least squares line look like?
scatter_fit(subset, "days since 1/22/20", 'Log Cases In ' + country)
```



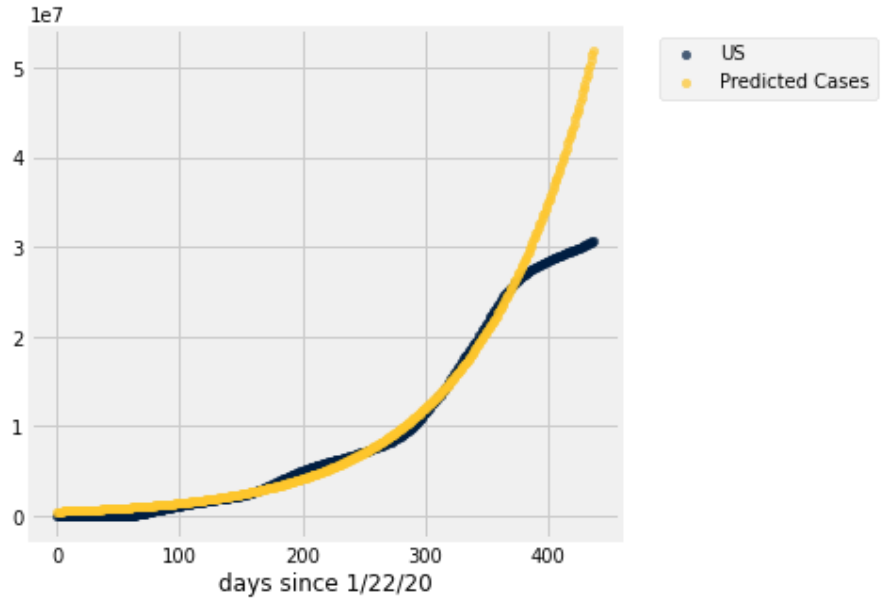

```
#Obtain the slope and intercept
r = slope(subset, "days since 1/22/20", 'Log Cases In ' + country)
logK = intercept(subset, "days since 1/22/20", 'Log Cases In ' + country)
(r, logK)
```

```
(0.010671721140202439, 13.10268836102415)
```

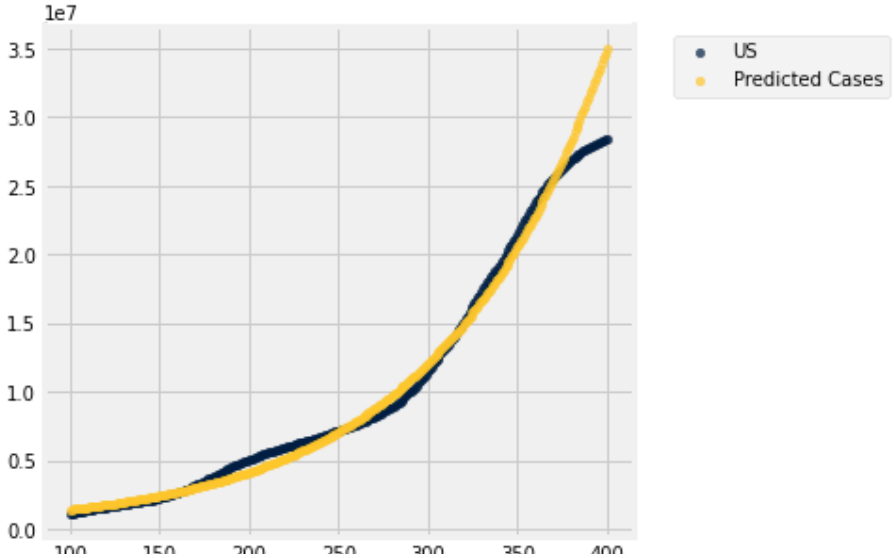
```
# Add a prediction column to the logCases table and the original data table
# For the logCases table, we can keep the predictions on the log scale
logCases = logCases.with_column('Log Predicted Cases', logK + r*logCases.column('days si
# For the data table, we exponentiate to give predictions on the original scale
data = data.with_column('Predicted Cases', np.exp(logK + r*logCases.column('days since 1
```

Next, we'll produce some visualizations of the observed and predicted values, both on the original scale and the log scale.

```
# If you invoke the "scatter" method using only a single column name, you'll get multiple
# of axes using the single column as the horizontal scale.  So we can select three colum
(data.select('days since 1/22/20', country, 'Predicted Cases')
     .scatter('days since 1/22/20'))
```




```
# We can focus on certain subplots to see better.
# Experiment with the 'are.between' limits here to see how things change:
(data.select('days since 1/22/20', country, 'Predicted Cases')
     .where('days since 1/22/20', are.between(100,401))
     .scatter('days since 1/22/20'))
```

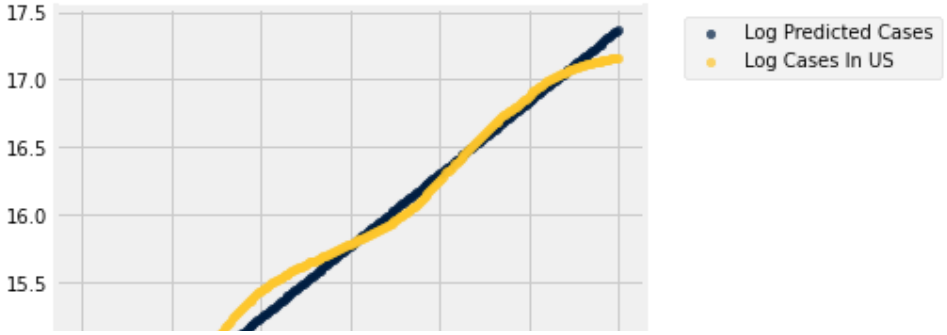


Questions

1. Does the exponential curve fit using the data between 100 and 400 days do a good job of capturing the **entire** dataset? Explain.
2. Make a similar scatterplot or scatterplots on the log scale using the `logCases` Table, that is, plot `Log Predicted Cases` and `Log Cases in US` from the `logCases` Table against `days since 1/22/20`. You can experiment with 'are.between' limits to zoom in on certian areas. What, if any, advantages or disadvantages are there to using the log scale here?

1. Yes, it provides an appropriate outline of the exponential curve (cases in the US related to the number of days since 1/22/20). It subsets our previous data to look at cases as the relationship is linear. Using the range 100-400, we already view the majority of the scatter plot hence capturing the entire dataset. Our graph captures the relationship pretty similar between predicted cases and actual cases. But it seems that around 400 days, the expected cases seem to take off more exponentially compared to actual cases. Apart from that, around day 160-250, we see actual cases greater than what our prediction was.
2. Comparing our log scatter plot with the data table scatter plot, we see our predicted line as much more linear, which clarifies our actual log cases in the US. It is obviously not a 'true line,' but it is somewhat plausible. The only disadvantage is that the log cases' shape in the US shifts between being greater than the predicted and less than. Around the 400 days, we see the log cases being less than the predicted count.

```
(logCases.select('days since 1/22/20', 'Log Predicted Cases', 'Log Cases In US')
     .where('days since 1/22/20', are.between(100,401))
     .scatter('days since 1/22/20'))
```



Part 3 Use the original 'confirmed' Table to produce a map visualization that resembles the one at https://coronavirus.jhu.edu/map.html.

```
# As a starting point, let's find out what the most recent day is
last_column_index = confirmed.num_columns - 1
confirmed.labels[last_column_index]
```

```
'4/3/21'
```

```
# In the code below, make sure to use the most recent date, found in the previous line.
# Can you change the code to make this automatic?
recent_confirmed = (confirmed.select('Province/State', 'Country/Region', 'Lat', 'Long',
                              .relabel('Province/State', 'Label1')
                              .relabel('Country/Region', 'Label2')
                              .relabel('4/3/21', 'Cases'))

recent_confirmed_US = (confirmed_US.select('Admin2', 'Province_State', 'Lat', 'Long_', '
                                    .relabel('Admin2', 'Label1')
                                    .relabel('Province_State', 'Label2')
                                    .relabel('Long_','Long')
                                    .relabel('4/3/21', 'Cases'))

(recent_confirmed.num_rows, recent_confirmed_US.num_rows) # How many rows in the two data
```

```
(274, 3342)
```

```
# Add all of the US case summaries to the overall table
for i in np.arange(recent_confirmed_US.num_rows):
  recent_confirmed = recent_confirmed.with_row(recent_confirmed_US.row(i))


# Get rid of all rows where the count is zero.  Again, make sure you're using the most re
recent_confirmed = recent_confirmed.where('Cases', are.above(0))
# Get rid of row for entire US country since we have added information on individual cit
recent_confirmed = recent_confirmed.where('Label2', are.not_equal_to('US'))


recent_confirmed.num_rows # How many rows are left in the combined dataset?
```

```
3514
```

```
recent_confirmed.show(5)
```

Label1	Label2	Lat	Long	Cases
nan	Afghanistan	33.9391	67.71	56595
nan	Albania	41.1533	20.1683	126183
nan	Algeria	28.0339	1.6596	117524
nan	Andorra	42.5063	1.5218	12174
nan	Angola	-11.2027	17.8739	22579

```
recent_confirmed.row(51)
```

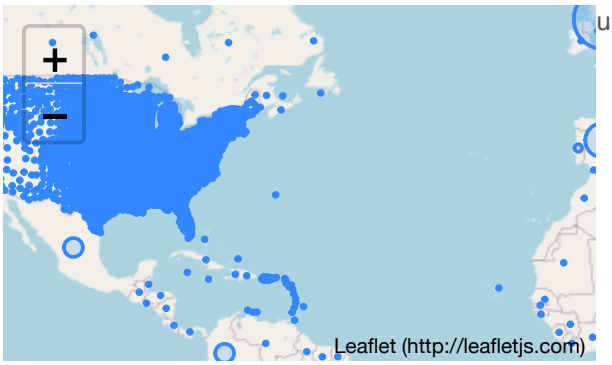
```
Row(Label1='Repatriated Travellers', Label2='Canada', Lat=nan, Long=nan, Cases=13)
```

```
recent_confirmed = recent_confirmed.where('Label1', are.not_containing('Repatriated Trave
```

Now adapt the folium package code you've used previously to produce your map. I recommend looking at the map produced in the bike sharing lab , Lab 5, for some guidance. Some potentially helpful ideas for using the 'Circle' method for folium:

- Using 'weight=0' will eliminate the border of the circles, which might make things easier to see.
- For circle radius, instead of the square root of number of cases, you might try the logarithm of number of cases (you can use 'np.log') since this results in a smaller range of circle sizes. You might play around with other ideas, such as defining a fixed set of circle radii according to some range of case numbers you define yourself (this appears to be what the Hopkins group has done).
- You will definitely need to experiment with the multiplier you use for your circle radii. Keep in mind that the radius units for folium.Circle are meters, and on a map of the world your circles will have to be thousands of meters wide to be visible.
- You can set 'fill_color' and 'fill_opacity' (the default of opacity is 20%) if you use 'fill=True', which I recommend.
- Don't forget you can include text for a popup with each marker. For instance, in the final line of code below I will add a column called "popup" that makes a simple string with location and case number information. Make sure to follow the bike sharing lab example if you want to use this column. This is not necessary, but would be a nice addition to your map.

```
CovidMap = folium.Map(location = [0, 0], width = '50%', height = '50%', zoom_start = 2)

multiplier = 2

for i in np.arange(recent_confirmed.num_rows):
  loc = [recent_confirmed.column('Lat')[i], recent_confirmed.column('Long')[i]]
  rad = recent_confirmed.column('Cases')[i] * multiplier/ 25
  folium.Circle(location = loc, radius = rad, weight = 2, fill=True, popup = text).add_t

CovidMap
```




```
# In case you want to use a 'popup' column in constructing your map, here is an example:
popup_text_column = make_array()
for i in np.arange(recent_confirmed.num_rows):
  text = (recent_confirmed.column('Label2')[i] + ': ' +
          str(recent_confirmed.column('Cases')[i]) + ' cases')
  if (recent_confirmed.column('Label1')[i] != 'nan'):
    text = recent_confirmed.column('Label1')[i] + ', ' + text
  popup_text_column = np.append(popup_text_column, text)

recent_confirmed = recent_confirmed.with_column('popup', popup_text_column)
recent_confirmed
```

Label1	Label2	Lat	Long	Cases	popup
nan	Afghanistan	33.9391	67.71	56384	Afghanistan: 56384 cases
nan	Albania	41.1533	20.1683	124723	Albania: 124723 cases
nan	Algeria	28.0339	1.6596	117061	Algeria: 117061 cases
nan	Andorra	42.5063	1.5218	11944	Andorra: 11944 cases
nan	Angola	-11.2027	17.8739	22182	Angola: 22182 cases
nan	Antigua and Barbuda	17.0608	-61.7964	1136	Antigua and Barbuda: 1136 cases
nan	Argentina	-38.4161	-63.6167	2332765	Argentina: 2332765 cases
nan	Armenia	40.0691	45.0382	191491	Armenia: 191491 cases
Australian Capital Territory	Australia	-35.4735	149.012	123	Australian Capital Territory, Australia: 123 cases
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	New South Wales,

✓ 7s completed at 7:20 PM