Allison Roeser

Week 1 Assignment: COVID-19 Visualizations

Prevalence: As of April 9, 2020, there were 1.48 million COVID-19 infections across 205 countries and territories. Eight countries have more than 50,000 cases (Figure 1), which represents 76% of worldwide cases. The United States alone contains 29% of total cases (Figure 2). Seventeen countries have more than 10,000 cases and account for 88% of worldwide cases. Figure 3 contains a tree map of countries with 10,000 or more cases. Figures 4-6 contain density plots and scatter plots for population and case count before and after scaling. Standard Scaling transforms the mean to 0 and the standard deviation to 1. Min Max Scaling is used to transform the data to a specified range of -1 to 1. These plots illustrate that population and case count are both highly skewed right.

***Infection Rates*:** The International Conveyance Japan has an especially low population (3000) and high number of cases (696), leading to an infection rate of 232 per thousand (Figure 7). With this outlier removed, the mean infection rate is 1.23 per thousand with an interquartile range of 0.01 to 0.32 infections per thousand (Figure 8). For countries with greater than 10,000 cases, the mean remains 1.23 infections per thousand and the interquartile range increases to 0.52 to 1.47 (Figure 9). Larger countries are experiencing higher infection rates, likely because of availability of testing. Of the eight counties with more than 50,000 cases, Spain has the highest infection rate at 3.14 per thousand, and China has the lowest at 0.06 per thousand (Figure 10). Figure 11 shows a density plot of infection rate grouped by population size. Figure 12 and 13 display the data after scaling. At all population levels, the data is skewed right with a few countries having very high infection rates.

Cumulative Cases by Country: Figure 14 provides a line chart illustrating how the cumulative number of cases has grown since 12/31/19 for the eight countries with the most cases. China experienced its peak rate of case growth on 2/13/20 with 15,141 new cases. Since mid-February, the growth in new cases coming out of China has slowed dramatically. In contrast, the other seven countries experienced their largest exponential growth swings beginning in late March and continuing through

to the time of the most current data (April 9, 2020). These counties may have not yet reached their peaks. Figure 15 displays a more refined date range of 3/1/20 to 4/7/20. The graph illustrates how China's case count growth had plateaued by 3/1/20. The United States has become the leader in case count with US case numbers increasing at a much faster rate than the other seven top countries. A log of cumulative case count provides an additional view showing the strong disparity between the Chinese growth curve and those for the other countries (Figure 16). The logarithmic curves for the non-Chinese countries follow a strongly similar pattern with the beginning of strong exponential growth starting at the same time point for each country. COVID-19 became a global pandemic when at the time it took hold in these countries.

***Death Rate*:** While the curve for cumulative number of cases is similar across countries, the curves for daily death rates varies significantly between countries. Figures 17 & 18 display the daily death rates (per million people) for the eight top countries. The peak daily death rates in France, Spain, Italy, and UK were more than double those in other countries. Strikingly, the daily death rate curve for China is nearly flat when plotted against other countries' daily death rate curves. in total, China has only recorded 45.3 deaths per million people while Spain has 311.5 deaths per million and Italy has 292.4 deaths per million. (Figure 19).

***Likelihood of Death After Infection*:** The percentage of cases that result in death for all 205 countries is displayed in Figure 20. While most countries have low rates of death (under 5%), there are some outliers with death rates up to 25%. For countries with greater than 10,000 cases, the mean death rate per infection is 3.8% with and interquartile range of 0.3% to 4.9%. The death rate among the top eight countries varies significantly. France and Italy have rates of 13.2% and 12.9% respectively while Germany's death rate is only 1.9%. The variance in death rate can partially be attributed to where each individual country is in the outbreak progression. Germany at an earlier stage so its death rate will likely increase with time. As with infection rate, the death rate was grouped by population and scaled. The data was again right skewed by high-value outliers. (Figure 23-25)

Identified Cases vs. Actual Cases: The data provided and analyzed is only as good as the testing for the virus. The true prevalence of infected individuals is likely significantly larger than these visualizations convey. Asymptomatic carriers are under-tested and under-represented.

References

Dzindo, I. Feature Scaling in Python. (May 23, 2018). Medium. Retrieved from https://medium.com/@ian.dzindo01/feature-scaling-in-python-a59cc72147c1

Holtz, Y. #124 Spaghetti Plot. (n.d.). The Python Graph Gallery. Retrieved from: https://python-graph-gallery.com/124-spaghetti-plot/

Holtz, Y. #200 Basic Treemap with Python. (n.d.). The Python Graph Gallery. Retrieved from: https://python-graph-gallery.com/200-basic-treemap-with-python/

Appendix

Figure 1: Bar Chart of total case count for top 8 countries

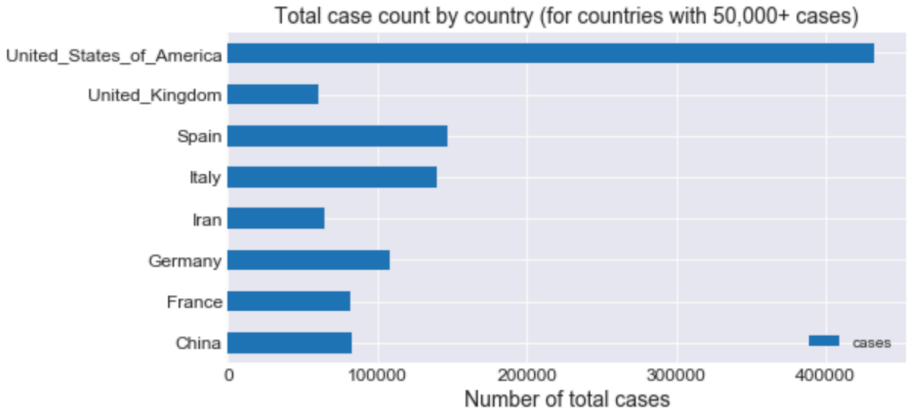


Figure 2: Bar Chart of percent of worldwide cases by country

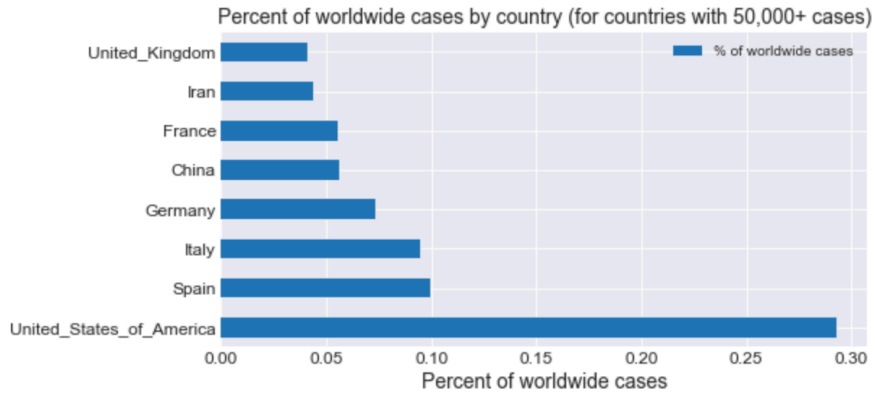


Figure 3: Tree map of case counts for countries with > 10,000 total cases (sized by case count) (Holtz n.d.)

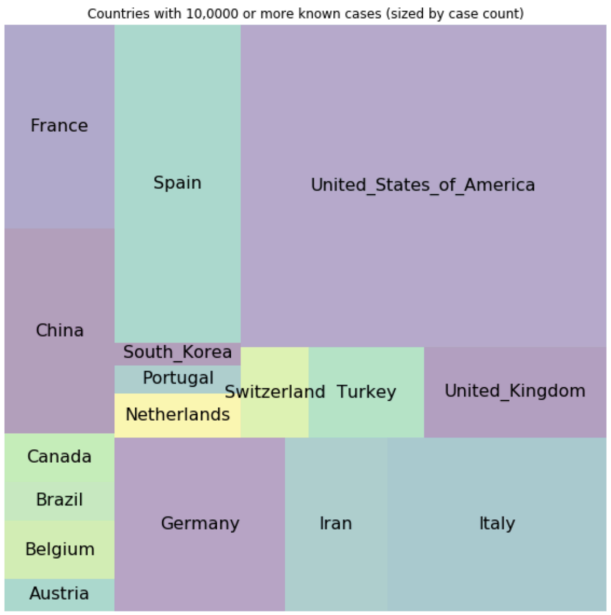


Figure 4: Population and Number of Cases Before Scaling

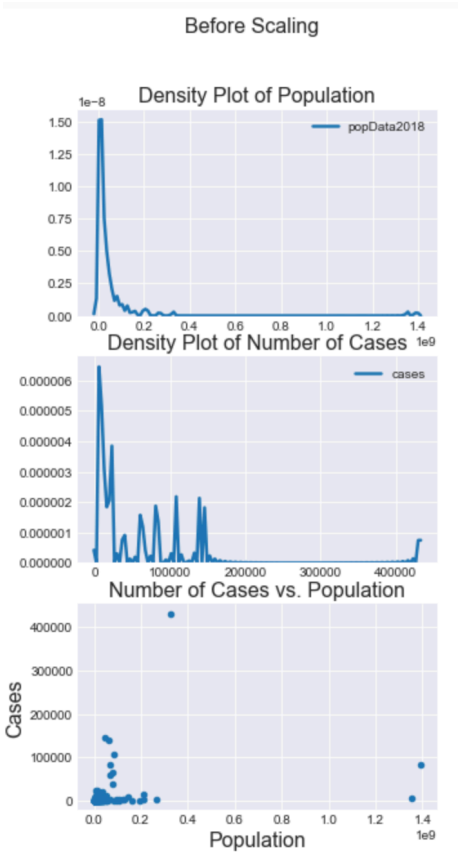


Figure 5: Population and Number of Cases After Standard Scaling

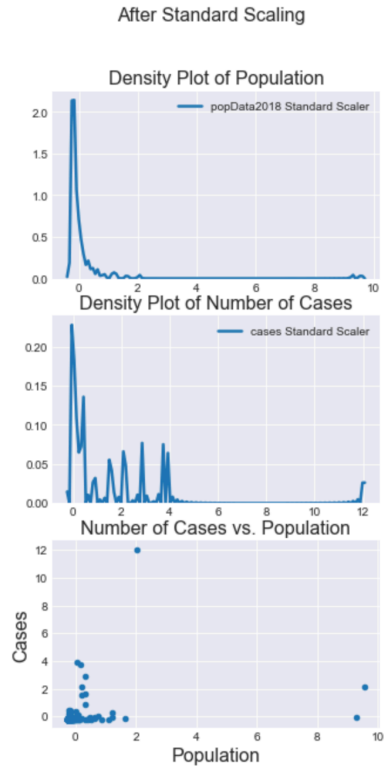


Figure 6: Population and Number of Cases After Min Max Scaling

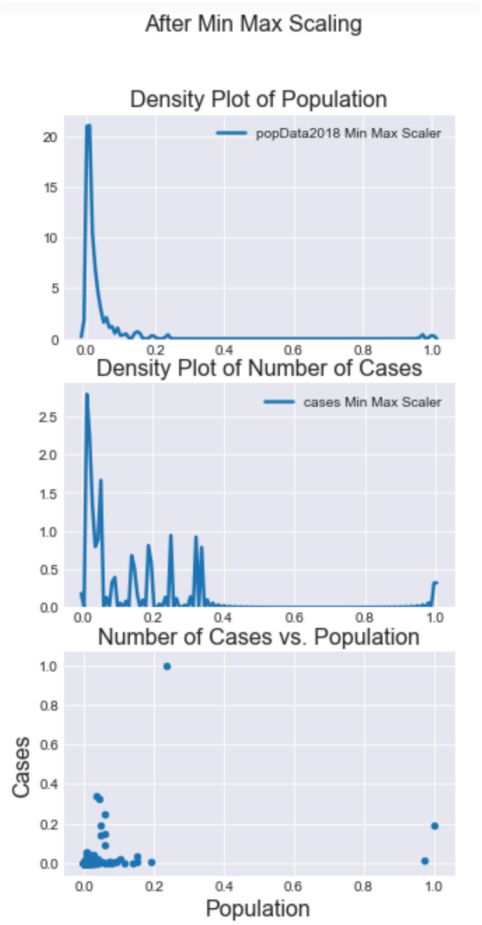


Figure 7: Box plot of Infection rate per thousand people

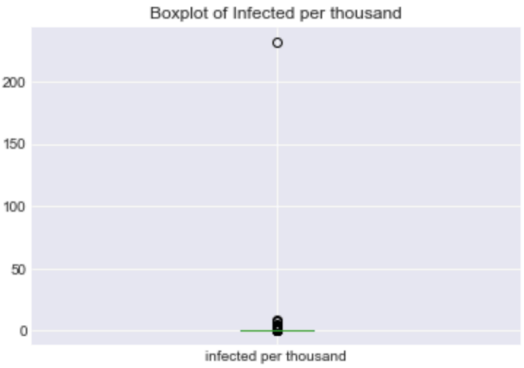


Figure 8: Boxplot of infection rate per thousand people (with outlier removed)

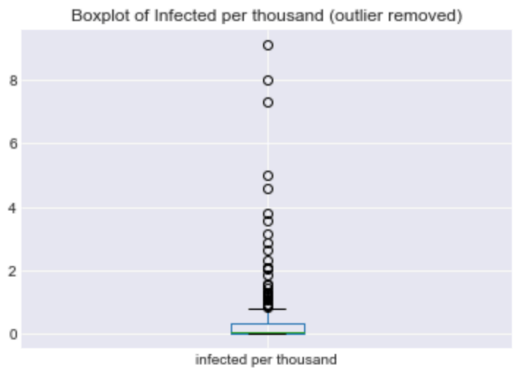


Figure 9: Boxplot of infection rate per thousand (for countries with more than 10,000 cases)

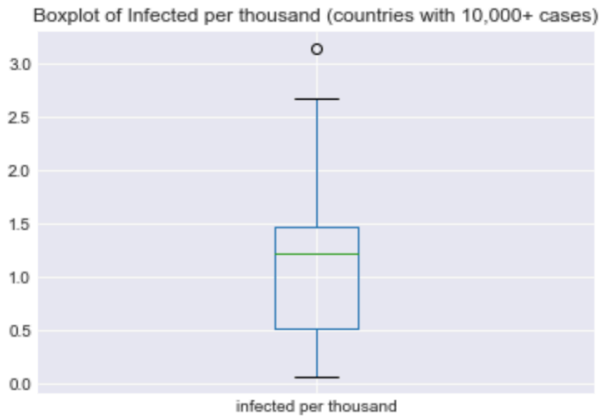


Figure 10: Bar Chart of Infection rate per thousand people for Top 8 countries

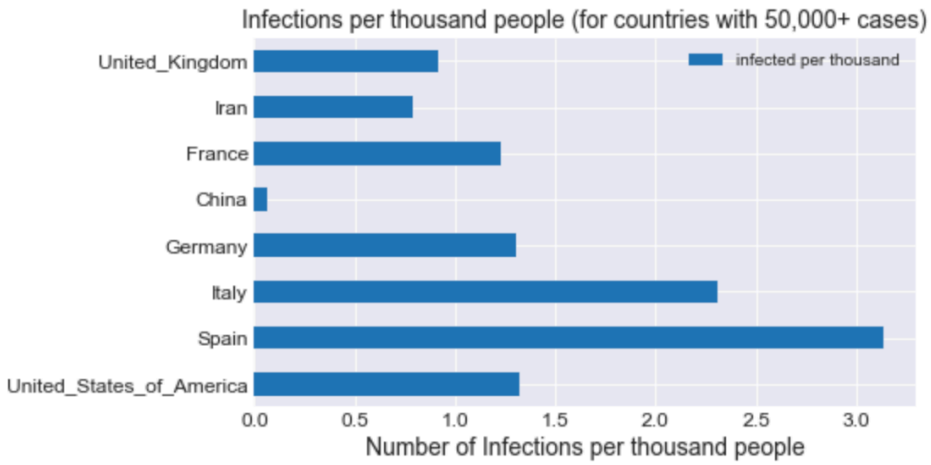


Figure 11: Infection rates per thousand grouped by population size (Dzindo 2018)

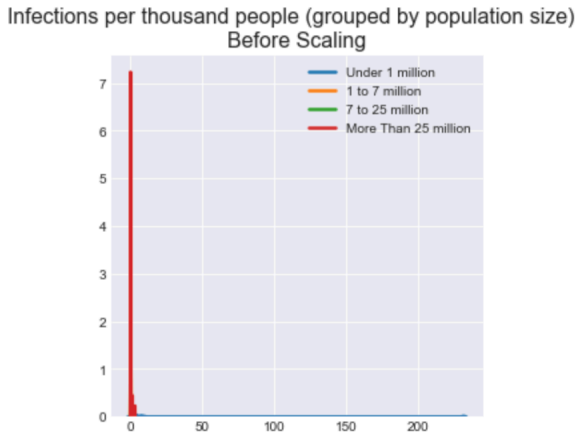


Figure 12: Infection rates per thousand grouped by population size (After Standard Scaling) (Dzindo 2018)

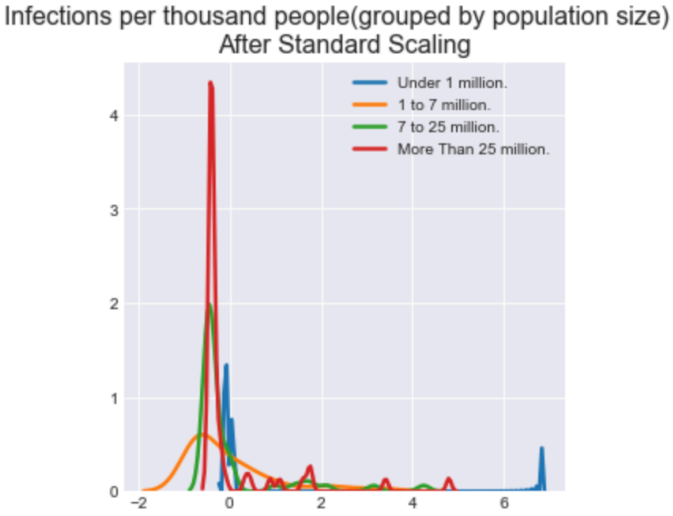


Figure 13: Infection rates per thousand grouped by population size (After Min Max Scaling) (Dzindo 2018)

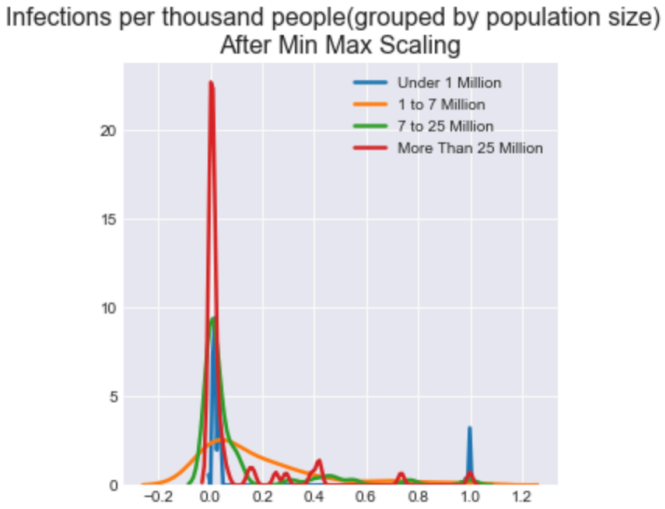


Figure 14: Cumulative case counts from 12/31/19 to 4/9/20 for Top 8 countries (50,000+ cases) (Holtz n.d.)

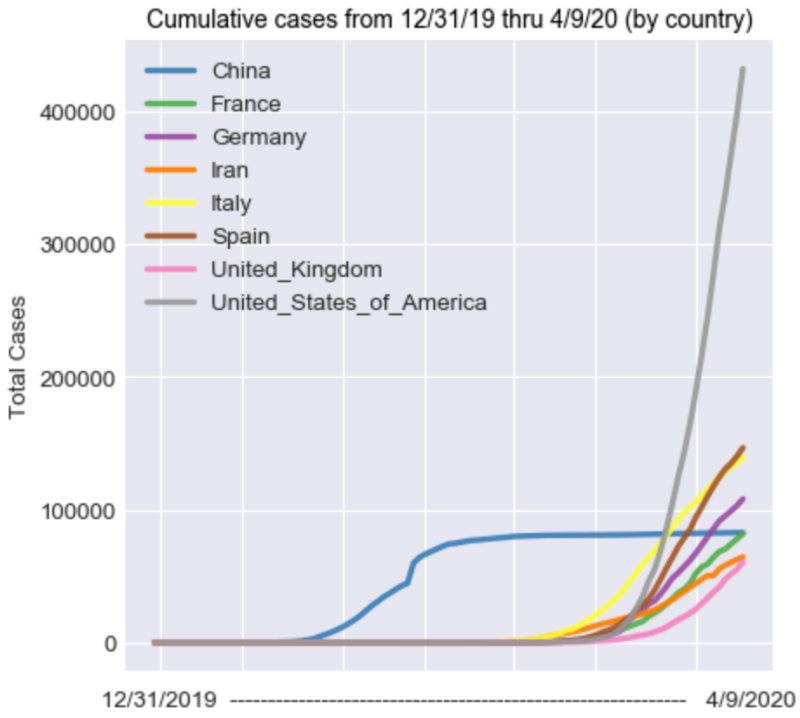


Figure 15: Cumulative case counts from 3/1/20 to 4/9/20 for Top 8 countries (50,000+ cases) (Holtz n.d.)

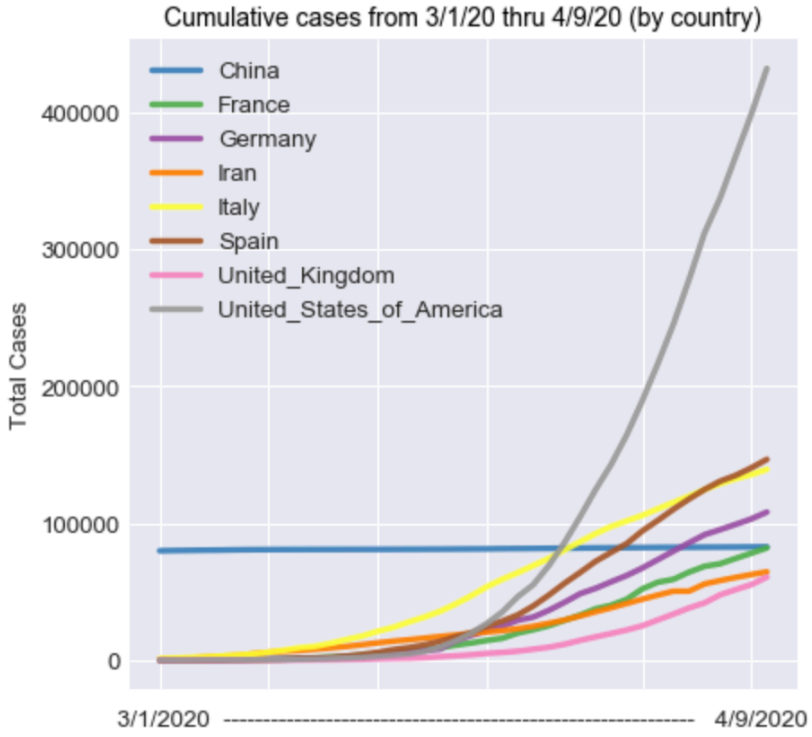


Figure 16: Log of cumulative cases from 12/31/19 to 4/9/20 for Top 8 countries (50,000+ cases) (Holtz n.d.)

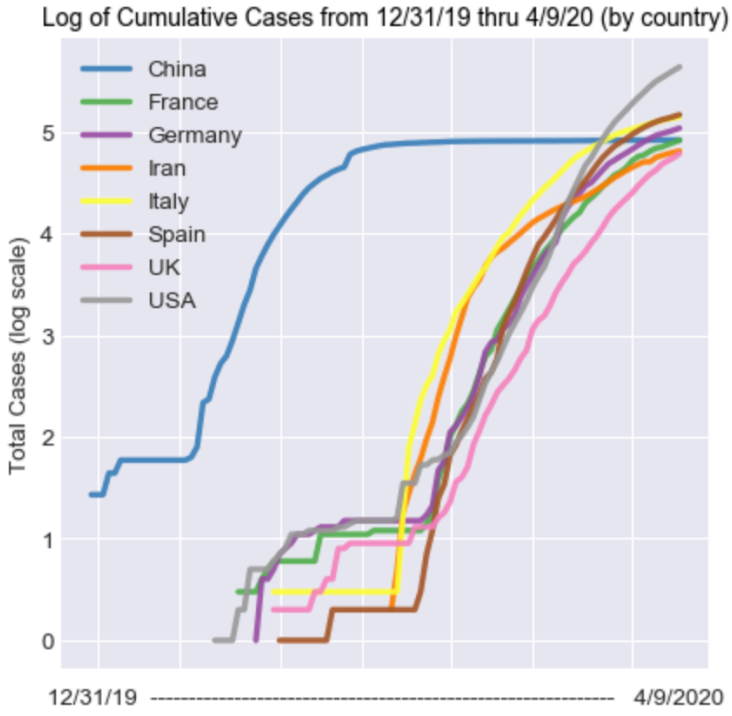


Figure 17: Daily deaths per million from 12/31/19 to 4/9/20 for Top 8 countries (50,000+ cases) (Holtz n.d.)

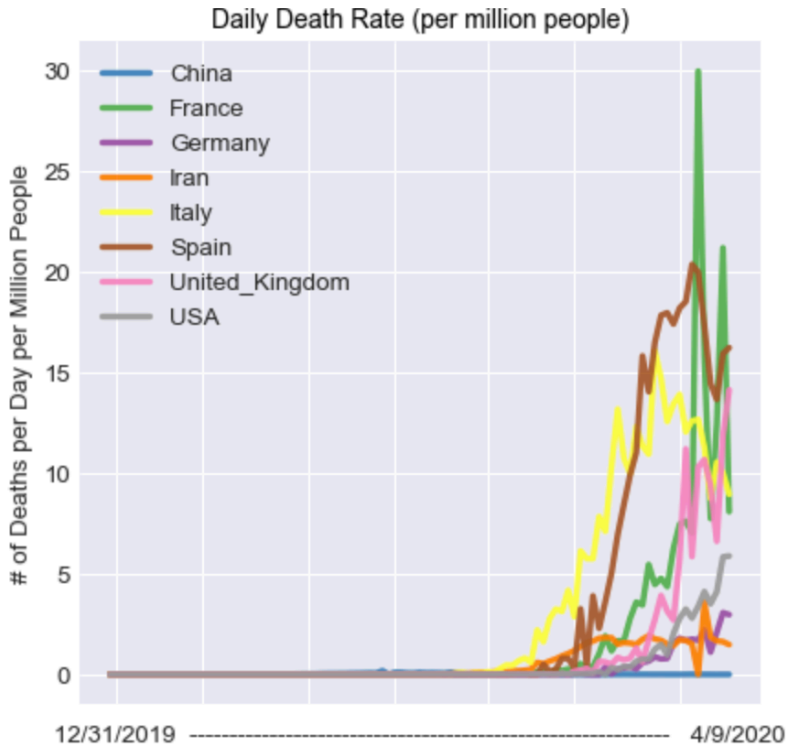


Figure 18: Daily deaths per million from 3/1/20 to 4/9/20 for Top 8 countries (50,000+ cases) (Holtz n.d.)

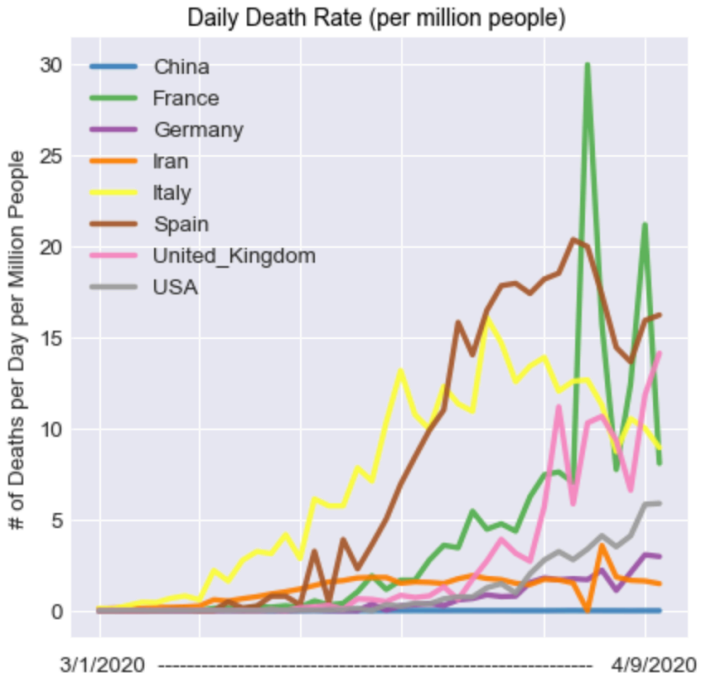


Figure 19: Bar chart of deaths per million people (for countries with 50,000 or more cases)

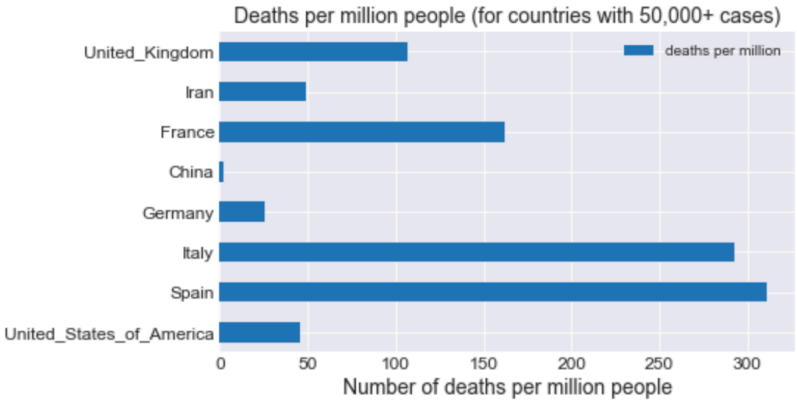


Figure 20: Boxplot of death rate per infection for all countries

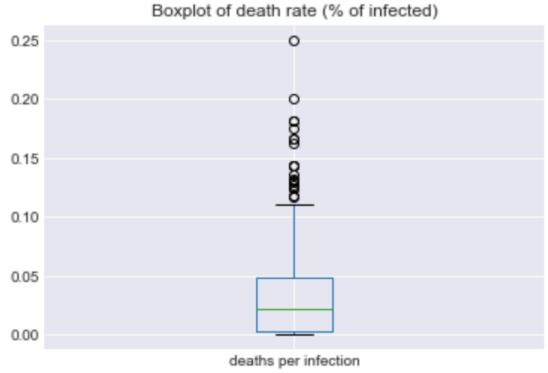


Figure 21: Boxplot of death rate per infection (for countries with 10,000 or more cases)

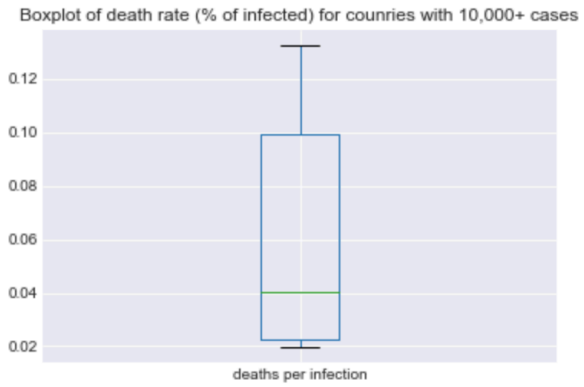


Figure 22: Bar chart of death rate per infection (for countries with 50,000 or more cases)

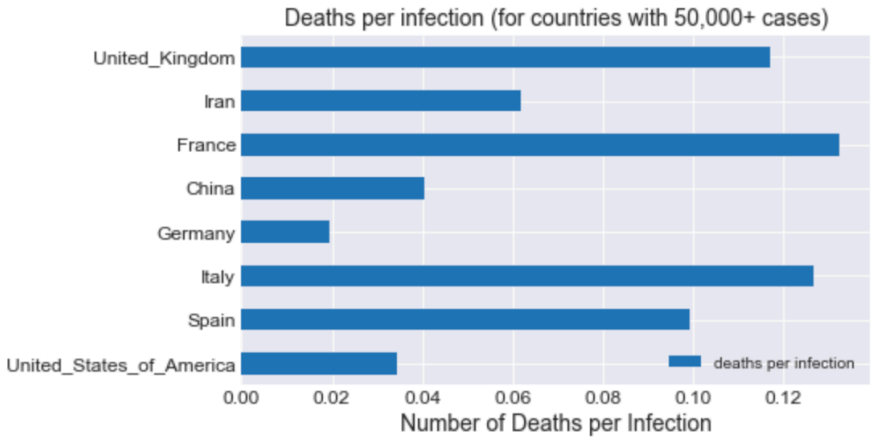


Figure 23: Death rate per infection grouped by population size (Dzindo 2018)

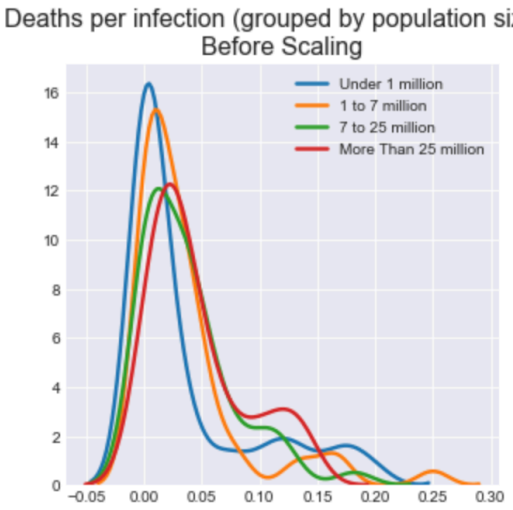


Figure 24: Death rate per infection grouped by population size (After Standard Scaling) (Dzindo 2018)

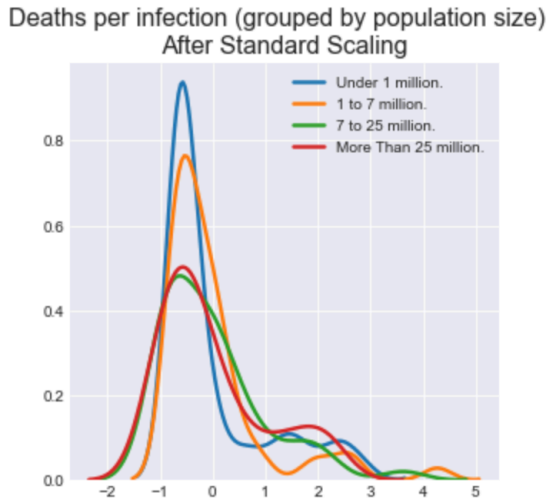
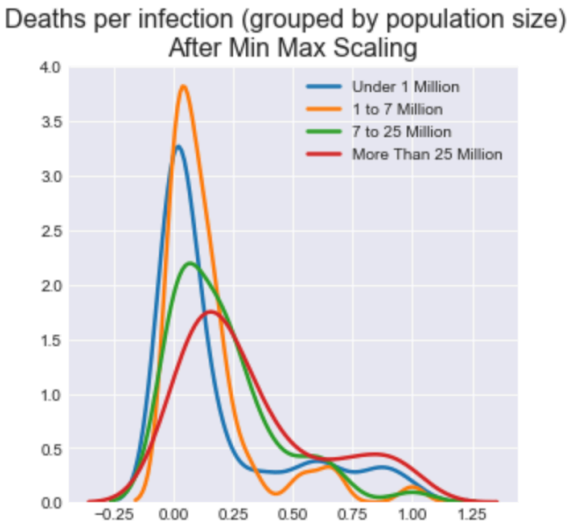


Figure 25: Death rate per infection grouped by population size (After Min Max Scaling) (Dzindo 2018)



In [20]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import squarify
import seaborn as sns
import datetime
import numpy as np
import matplotlib as mpl
from matplotlib import cm
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
```

In [21]:
```
# import warnings filter
from warnings import simplefilter
# ignore all future warnings
simplefilter(action='ignore', category=FutureWarning)
```

In [22]:
```
# read in the csv file
corona = pd.read_csv("COVID-19-April-9")
```

In [23]:
```
# print the first five observations
corona.head()
```

Out[23]:

	dateRep	day	month	year	cases	deaths	countriesAndTerritories	geoId	countryterritoryCod
0	09/04/2020	9	4	2020	56	3	Afghanistan	AF	AF
1	08/04/2020	8	4	2020	30	4	Afghanistan	AF	AF
2	07/04/2020	7	4	2020	38	0	Afghanistan	AF	AF
3	06/04/2020	6	4	2020	29	2	Afghanistan	AF	AF
4	05/04/2020	5	4	2020	35	1	Afghanistan	AF	AF

In [24]:
```
# print the last five observations
corona.tail()
```

Out[24]:

	dateRep	day	month	year	cases	deaths	countriesAndTerritories	geoId	countryterritory
9712	25/03/2020	25	3	2020	0	0	Zimbabwe	ZW	
9713	24/03/2020	24	3	2020	0	1	Zimbabwe	ZW	
9714	23/03/2020	23	3	2020	0	0	Zimbabwe	ZW	
9715	22/03/2020	22	3	2020	1	0	Zimbabwe	ZW	
9716	21/03/2020	21	3	2020	1	0	Zimbabwe	ZW	

In [25]:
```
# Get information about the variables
corona.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9717 entries, 0 to 9716
Data columns (total 10 columns):
dateRep                    9717 non-null object
day                        9717 non-null int64
month                      9717 non-null int64
year                       9717 non-null int64
cases                      9717 non-null int64
deaths                     9717 non-null int64
countriesAndTerritories    9717 non-null object
geoId                      9691 non-null object
countryterritoryCode       9524 non-null object
popData2018                9569 non-null float64
dtypes: float64(1), int64(5), object(4)
memory usage: 759.3+ KB
```

In [26]:
```
# Get summary statistics for the numeric variables
corona.describe()
```

Out[26]:

	day	month	year	cases	deaths	popData2018
count	9717.000000	9717.000000	9717.000000	9717.000000	9717.000000	9.569000e+03
mean	15.411341	2.621385	2019.993105	151.983019	9.037357	6.375818e+07
std	9.374690	1.287272	0.082754	1184.675208	74.966368	2.005423e+08
min	1.000000	1.000000	2019.000000	-9.000000	0.000000	1.000000e+03
25%	7.000000	2.000000	2020.000000	0.000000	0.000000	3.449299e+06
50%	15.000000	3.000000	2020.000000	0.000000	0.000000	1.028176e+07
75%	24.000000	3.000000	2020.000000	14.000000	0.000000	4.222843e+07
max	31.000000	12.000000	2020.000000	34272.000000	2004.000000	1.392730e+09

In [27]:
```
# Convert dateRep to a datetime variable called "date"
corona["date"] = pd.to_datetime(corona['dateRep'], format='%d/%m/%Y')
corona.head()
```

Out[27]:

	dateRep	day	month	year	cases	deaths	countriesAndTerritories	geoId	countryterritoryCod
0	09/04/2020	9	4	2020	56	3	Afghanistan	AF	AF
1	08/04/2020	8	4	2020	30	4	Afghanistan	AF	AF
2	07/04/2020	7	4	2020	38	0	Afghanistan	AF	AF
3	06/04/2020	6	4	2020	29	2	Afghanistan	AF	AF
4	05/04/2020	5	4	2020	35	1	Afghanistan	AF	AF

In [28]:
```
# Create byCountry DataFrame by grouping rows by country to calculate total
byCountry = corona.groupby(["countriesAndTerritories"])[["cases"]].sum().re
byCountry
```

Out[28]:

	countriesAndTerritories	cases
0	Afghanistan	423
1	Albania	400
2	Algeria	1572
3	Andorra	564
4	Angola	19
...	...	...
200	Uzbekistan	555
201	Venezuela	167
202	Vietnam	251
203	Zambia	39
204	Zimbabwe	11

205 rows × 2 columns

In [29]:
```
# Create deaths DataFrame by grouping rows by country and calculate deaths
deaths = corona.groupby(["countriesAndTerritories"])[["deaths"]].sum().rese
deaths
```

Out[29]:

	countriesAndTerritories	deaths
0	Afghanistan	14
1	Albania	22
2	Algeria	205
3	Andorra	23
4	Angola	2
...	...	...
200	Uzbekistan	3
201	Venezuela	8
202	Vietnam	0
203	Zambia	1
204	Zimbabwe	2

205 rows × 2 columns

In [30]:
```
# Create popData DataFrame by grouping rows by country and population
popData = corona.groupby(["countriesAndTerritories"])[["popData2018"]].mean
popData
```

Out[30]:

	countriesAndTerritories	popData2018
0	Afghanistan	37172386.0
1	Albania	2866376.0
2	Algeria	42228429.0
3	Andorra	77006.0
4	Angola	30809762.0
...	...	...
200	Uzbekistan	32955400.0
201	Venezuela	28870195.0
202	Vietnam	95540395.0
203	Zambia	17351822.0
204	Zimbabwe	14439018.0

205 rows × 2 columns

In [31]:
```
# Merge popData and deaths DataFrames with byCountry DataFrame
byCountry = pd.merge(byCountry, popData, on="countriesAndTerritories")
byCountry = pd.merge(byCountry, deaths, on="countriesAndTerritories")
byCountry
```

Out[31]:

	countriesAndTerritories	cases	popData2018	deaths
0	Afghanistan	423	37172386.0	14
1	Albania	400	2866376.0	22
2	Algeria	1572	42228429.0	205
3	Andorra	564	77006.0	23
4	Angola	19	30809762.0	2
...	...	...	...	...
200	Uzbekistan	555	32955400.0	3
201	Venezuela	167	28870195.0	8
202	Vietnam	251	95540395.0	0
203	Zambia	39	17351822.0	1
204	Zimbabwe	11	14439018.0	2

205 rows × 4 columns

In [32]:
```
# Calculate number of people infected per million by country
# Calculate death rate per country
byCountry['infected per thousand'] = byCountry['cases']/byCountry['popData2
byCountry['deaths per infection'] = byCountry["deaths"]/ byCountry['cases']
byCountry
```

Out[32]:

	countriesAndTerritories	cases	popData2018	deaths	infected per thousand	deaths per infection
0	Afghanistan	423	37172386.0	14	0.011379	0.033097
1	Albania	400	2866376.0	22	0.139549	0.055000
2	Algeria	1572	42228429.0	205	0.037226	0.130407
3	Andorra	564	77006.0	23	7.324105	0.040780
4	Angola	19	30809762.0	2	0.000617	0.105263
...	...	...	...	...	...	...
200	Uzbekistan	555	32955400.0	3	0.016841	0.005405
201	Venezuela	167	28870195.0	8	0.005785	0.047904
202	Vietnam	251	95540395.0	0	0.002627	0.000000
203	Zambia	39	17351822.0	1	0.002248	0.025641
204	Zimbabwe	11	14439018.0	2	0.000762	0.181818

205 rows × 6 columns

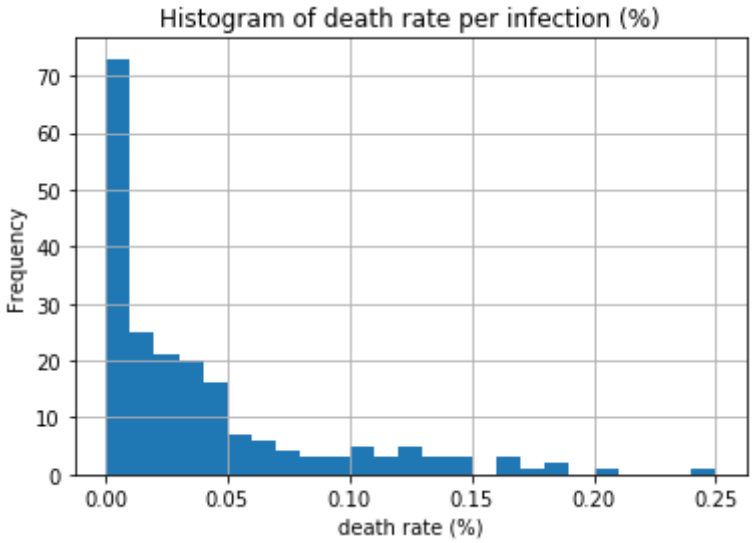
In [33]:
```
# Get summary statistics for byCountry DataFrame numerical variables
byCountry.describe()
```

Out[33]:

	cases	popData2018	deaths	infected per thousand	deaths per infection
count	205.000000	2.000000e+02	205.000000	200.000000	205.000000
mean	7203.995122	3.749471e+07	428.370732	1.661673	0.038344
std	35355.268965	1.420721e+08	2122.888806	16.415397	0.047518
min	1.000000	1.000000e+03	0.000000	0.000182	0.000000
25%	25.000000	1.267305e+06	1.000000	0.011425	0.003344
50%	263.000000	7.003150e+06	5.000000	0.076978	0.022222
75%	1623.000000	2.547777e+07	40.000000	0.329080	0.048544
max	432132.000000	1.392730e+09	17669.000000	232.000000	0.250000

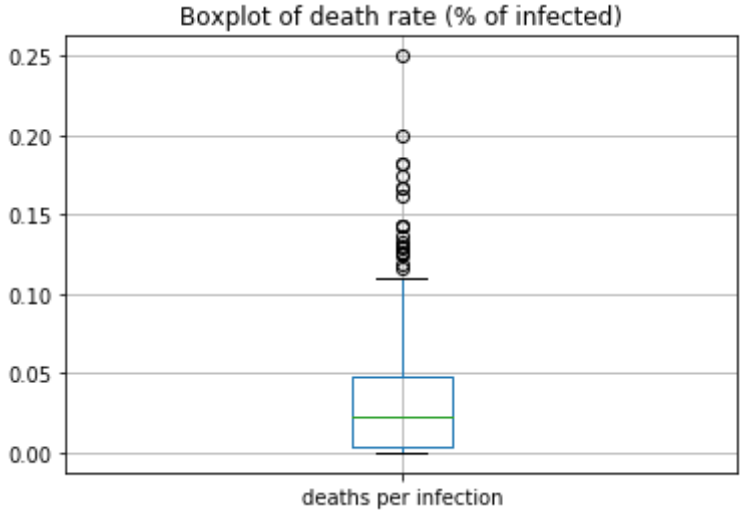
In [34]:
```
# Create histogram of "death rate" variable
byCountry.hist(column='deaths per infection', bins = 25)
plt.title('Histogram of death rate per infection (%)')
plt.ylabel('Frequency')
plt.xlabel('death rate (%)')
```

Out[34]:
```
Text(0.5, 0, 'death rate (%)')
```



In [35]:
```
# Create boxplot of "death rate" variable
byCountry.boxplot(column = "deaths per infection")
plt.title('Boxplot of death rate (% of infected)')
```

Out[35]:
```
Text(0.5, 1.0, 'Boxplot of death rate (% of infected)')
```



In [119]:
```
byCountry.describe()
```

Out[119]:

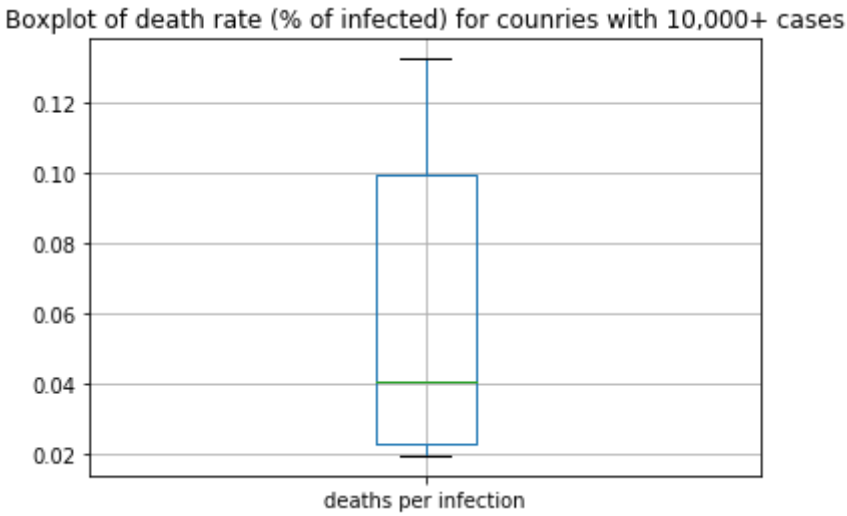
	cases	popData2018	deaths	infected per thousand	deaths per infection	popData2018 Standard Scaler	ca Stanc Sc
count	205.000000	2.000000e+02	205.000000	200.000000	205.000000	2.000000e+02	2.050000e
mean	7203.995122	3.749471e+07	428.370732	1.661673	0.038344	1.748601e-17	-2.12567
std	35355.268965	1.420721e+08	2122.888806	16.415397	0.047518	1.002509e+00	1.002448e
min	1.000000	1.000000e+03	0.000000	0.000182	0.000000	-2.645684e-01	-2.04230
25%	25.000000	1.267305e+06	1.000000	0.011425	0.003344	-2.556329e-01	-2.03550
50%	263.000000	7.003150e+06	5.000000	0.076978	0.022222	-2.151589e-01	-1.96802
75%	1623.000000	2.547777e+07	40.000000	0.329080	0.048544	-8.479565e-02	-1.58241
max	432132.000000	1.392730e+09	17669.000000	232.000000	0.250000	9.563002e+00	1.204822e

In [36]:
```
# Filter byCountry DataFrame to only countries with 10,000 or more total ca
over10000 = byCountry[byCountry['cases'] >= 10000]
over10000.shape
```

Out[36]:
```
(17, 6)
```

In [37]:
```
# Create boxplot of "death rate" variable
over10000.boxplot(column = "deaths per infection")
plt.title('Boxplot of death rate (% of infected) for counries with 10,000+
```

Out[37]:
```
Text(0.5, 1.0, 'Boxplot of death rate (% of infected) for counries with 1
0,000+ cases')
```



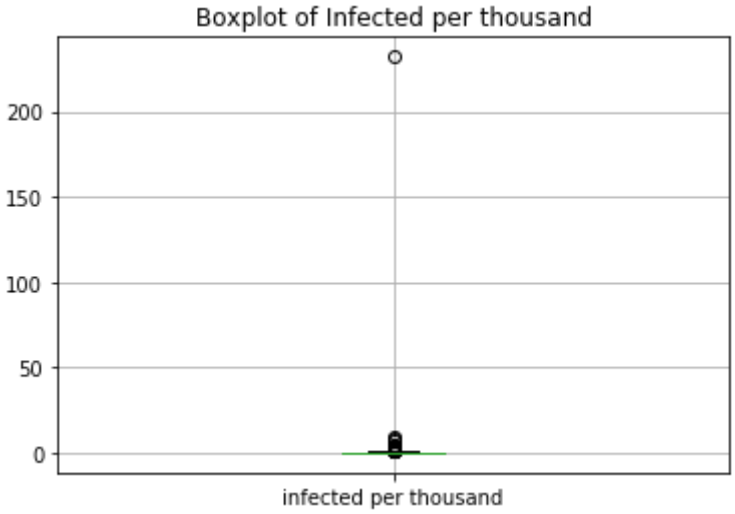
In [38]:
```
over10000.describe()
```

Out[38]:

	cases	popData2018	deaths	infected per thousand	deaths per infection
count	17.000000	1.700000e+01	17.000000	17.000000	17.000000
mean	76076.764706	1.507081e+08	4855.470588	1.233762	0.060641
std	102008.730412	3.301803e+08	5890.416996	0.892923	0.042452
min	10423.000000	8.516543e+06	204.000000	0.059502	0.019473
25%	19274.000000	1.723102e+07	705.000000	0.520092	0.022569
50%	38226.000000	6.043128e+07	2240.000000	1.224830	0.040292
75%	82870.000000	8.231972e+07	7097.000000	1.465915	0.099223
max	432132.000000	1.392730e+09	17669.000000	3.139517	0.132471

In [39]:
```
# Create boxplot of "infected per million" variable
byCountry.boxplot(column = "infected per thousand")
plt.title('Boxplot of Infected per thousand')
```

Out[39]:
```
Text(0.5, 1.0, 'Boxplot of Infected per thousand')
```



In [40]:
```
# Filter byCountry DataFrame to identify outlier with very high "infected p
outlier = byCountry[byCountry['infected per thousand'] > 100]
outlier
```

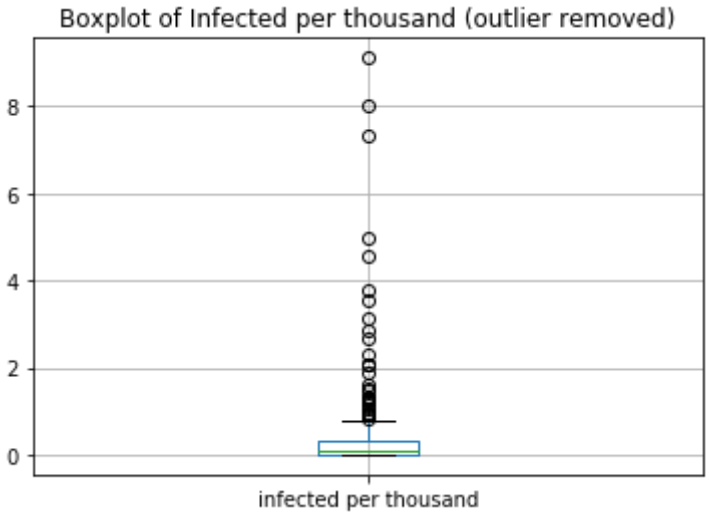
Out[40]:

	countriesAndTerritories	cases	popData2018	deaths	infected per thousand	deaths per infection
37	Cases_on_an_international_conveyance_Japan	696	3000.0	7	232.0	0.010057

In [41]:
```
# Filter byCountry DataFrame to remove outlier with very high "infected per
outlierRemoved = byCountry[byCountry['infected per thousand'] < 100]
```

In [42]:
```
# Create boxplot of "infected per thousand" variable with outlier removed
outlierRemoved.boxplot(column = "infected per thousand")
plt.title('Boxplot of Infected per thousand (outlier removed)')
```

Out[42]:
```
Text(0.5, 1.0, 'Boxplot of Infected per thousand (outlier removed)')
```



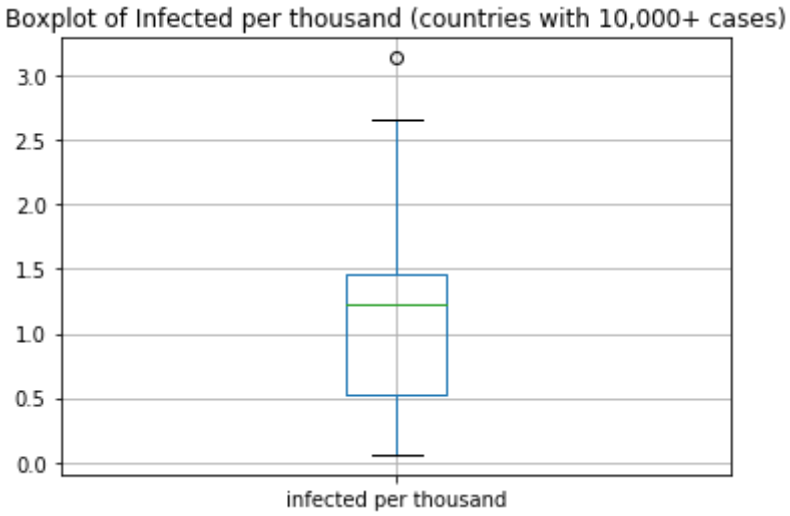
In [43]:
```
outlierRemoved.describe()
```

Out[43]:

	cases	popData2018	deaths	infected per thousand	deaths per infection
count	199.000000	1.990000e+02	199.000000	199.000000	199.000000
mean	7390.793970	3.768311e+07	440.753769	0.504194	0.039356
std	35868.667541	1.424054e+08	2153.582222	1.233432	0.047850
min	1.000000	1.000000e+03	0.000000	0.000182	0.000000
25%	26.500000	1.288473e+06	1.000000	0.011410	0.003705
50%	264.000000	7.024216e+06	5.000000	0.076970	0.022998
75%	1645.500000	2.573930e+07	40.500000	0.327673	0.049426
max	432132.000000	1.392730e+09	17669.000000	9.116472	0.250000

In [44]:
```
# Create boxplot of "infected per thousand" variable for countries with 10,
over10000.boxplot(column = "infected per thousand")
plt.title('Boxplot of Infected per thousand (countries with 10,000+ cases)'
```

Out[44]:
```
Text(0.5, 1.0, 'Boxplot of Infected per thousand (countries with 10,000+
cases)')
```



In [45]:
```
over10000["infected per thousand"].describe()
```

Out[45]:
```
count    17.000000
mean      1.233762
std       0.892923
min       0.059502
25%       0.520092
50%       1.224830
75%       1.465915
max       3.139517
Name: infected per thousand, dtype: float64
```

In [46]:
```
# Filter byCountry DataFrame to only countries with 50,000 or more total ca
over50000 = byCountry[byCountry['cases'] >= 50000]
over50000
```

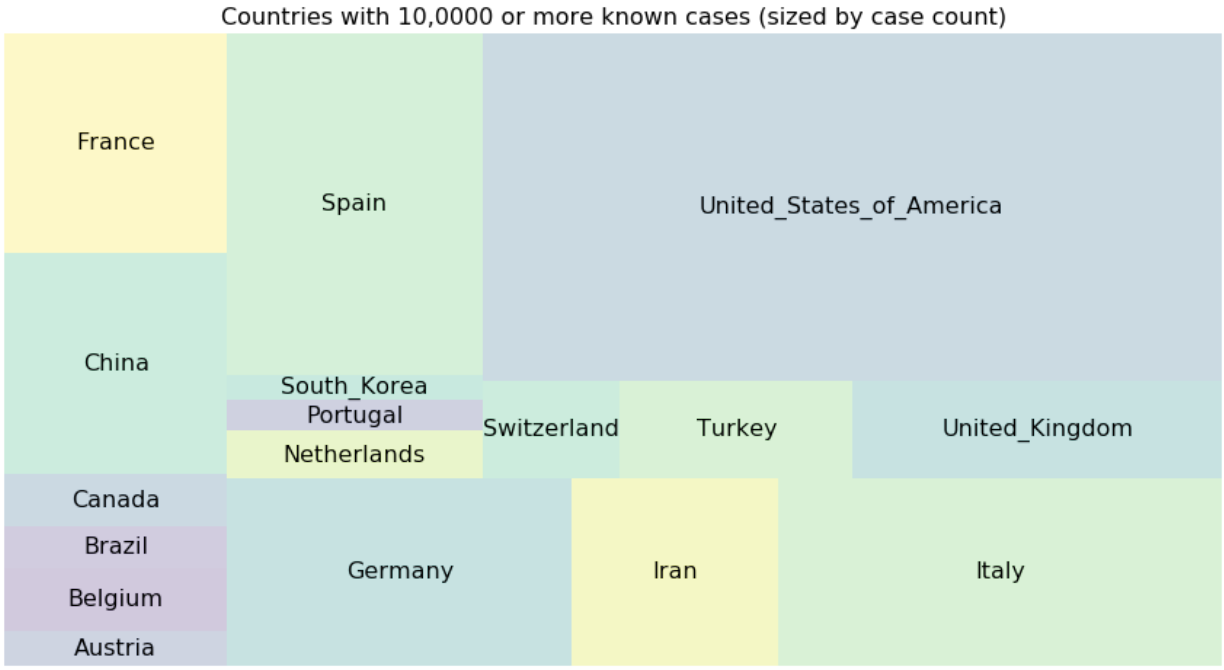
Out[46]:

	countriesAndTerritories	cases	popData2018	deaths	infected per thousand	deaths per infection
42	China	82870	1.392730e+09	3339	0.059502	0.040292
69	France	82048	6.698724e+07	10869	1.224830	0.132471
74	Germany	108202	8.292792e+07	2107	1.304772	0.019473
93	Iran	64586	8.180027e+07	3993	0.789557	0.061825
98	Italy	139422	6.043128e+07	17669	2.307116	0.126730
177	Spain	146690	4.672375e+07	14555	3.139517	0.099223
195	United_Kingdom	60733	6.648899e+07	7097	0.913429	0.116856
198	United_States_of_America	432132	3.271674e+08	14817	1.320828	0.034288

In [47]:
```
# List of countries with 50,000 or more total cases
over50000list = over50000["countriesAndTerritories"].unique()
over50000list
```

Out[47]:
```
array(['China', 'France', 'Germany', 'Iran', 'Italy', 'Spain',
       'United_Kingdom', 'United_States_of_America'], dtype=object)
```

In [48]:
```
#Holtz, Y. #200 Basic Treemap with Python. (n.d.). The Python Graph Gallery
    #https://python-graph-gallery.com/200-basic-treemap-with-python/


# Create Tree map for all countries with 10,000 or more total cases
plt.figure(figsize=(15,8))
plt.title("Countries with 10,0000 or more known cases (sized by case count)
squarify.plot(sizes=over10000['cases'], label=over10000['countriesAndTerrit
plt.axis('off')
plt.show()
```



In [49]:
```
# Calculate the total number of cases worldwide
totalCases = byCountry["cases"].sum()
totalCases
```

Out[49]: 1476819

In [50]:
```
# Filter byCountry DataFrame to only countries with fewer than 10,000 total
under10000 = byCountry[byCountry['cases'] < 10000]
```

In [51]:
```
# Filter byCountry DataFrame to only countries with more than 10,000 total
over10000 = byCountry[byCountry['cases'] >= 10000]
```

In [52]:
```
# Calculate the total number of cases for all countries with >= 10,000 case
totalOver10000 = over10000["cases"].sum()
totalOver10000
```

Out[52]: 1293305

In [53]:
```
# Calculate the total number of cases for all countries with < 10,000 cases
totalUnder10000 = under10000["cases"].sum()
totalUnder10000
```

Out[53]: 183514

In [54]:
```
# Calculate the total number of cases for all countries with > 50,000 cases
totalOver50000 = over50000["cases"].sum()
totalOver50000
```

Out[54]: 1116683

In [55]:
```
# Calculate % of worldwide cases that are from countries with > 50,000 case
Over50000PercentOfTotal = totalOver50000/totalCases
Over50000PercentOfTotal
```

Out[55]: 0.7561407322088896

In [56]:
```
# Calculate % of worldwide cases that are NOT from countries with > 50,000
allOthersPercentOfTotal = 1 - Over50000PercentOfTotal
allOthersPercentOfTotal
```

Out[56]: 0.24385926779111045

In [57]:
```
# Calculate % of worldwide cases that are from countries with > 50,000 case
Over10000PercentOfTotal = totalOver10000/totalCases
Over10000PercentOfTotal
```

Out[57]: 0.875736972506448

In [58]:
```
# Calculate % of worldwide cases per country
over50000['% of worldwide cases'] = over50000['cases']/totalCases
```

```
/Users/allisonroeser/opt/anaconda3/lib/python3.7/site-packages/ipykernel_
launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy)
```

In [59]:
```
# Sort values by descending % of cases
over50000.sort_values(by=['% of worldwide cases'], inplace=True, ascending
over50000
```

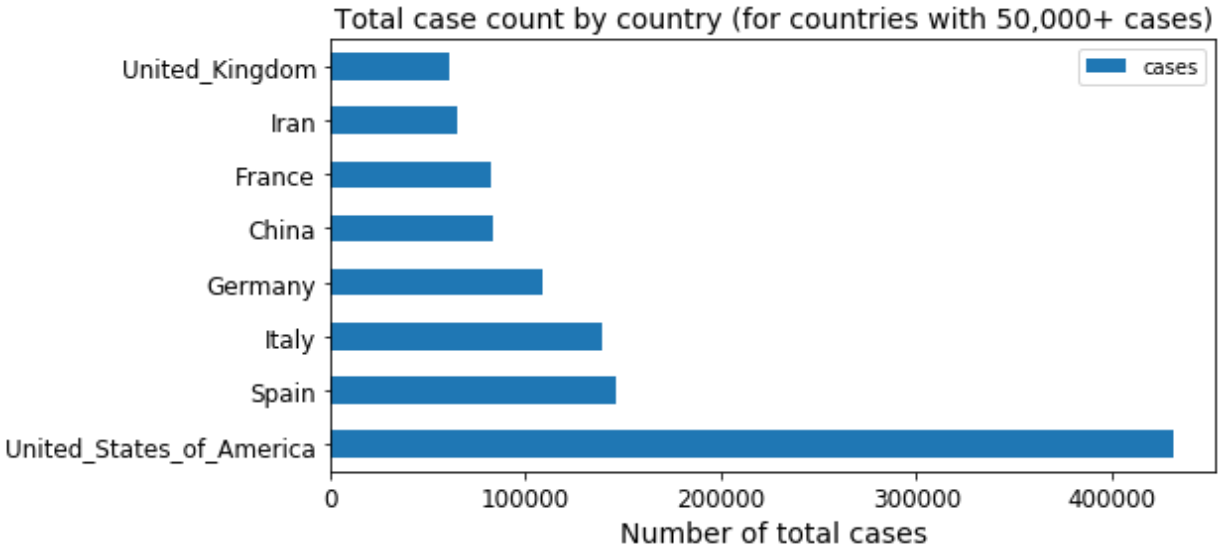
```
/Users/allisonroeser/opt/anaconda3/lib/python3.7/site-packages/ipykernel_
launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy)
```

Out[59]:

	countriesAndTerritories	cases	popData2018	deaths	infected per thousand	deaths per infection	% of worldwide cases
198	United_States_of_America	432132	3.271674e+08	14817	1.320828	0.034288	0.292610
177	Spain	146690	4.672375e+07	14555	3.139517	0.099223	0.099328
98	Italy	139422	6.043128e+07	17669	2.307116	0.126730	0.094407
74	Germany	108202	8.292792e+07	2107	1.304772	0.019473	0.073267
42	China	82870	1.392730e+09	3339	0.059502	0.040292	0.056114
69	France	82048	6.698724e+07	10869	1.224830	0.132471	0.055557
93	Iran	64586	8.180027e+07	3993	0.789557	0.061825	0.043733
195	United_Kingdom	60733	6.648899e+07	7097	0.913429	0.116856	0.041124

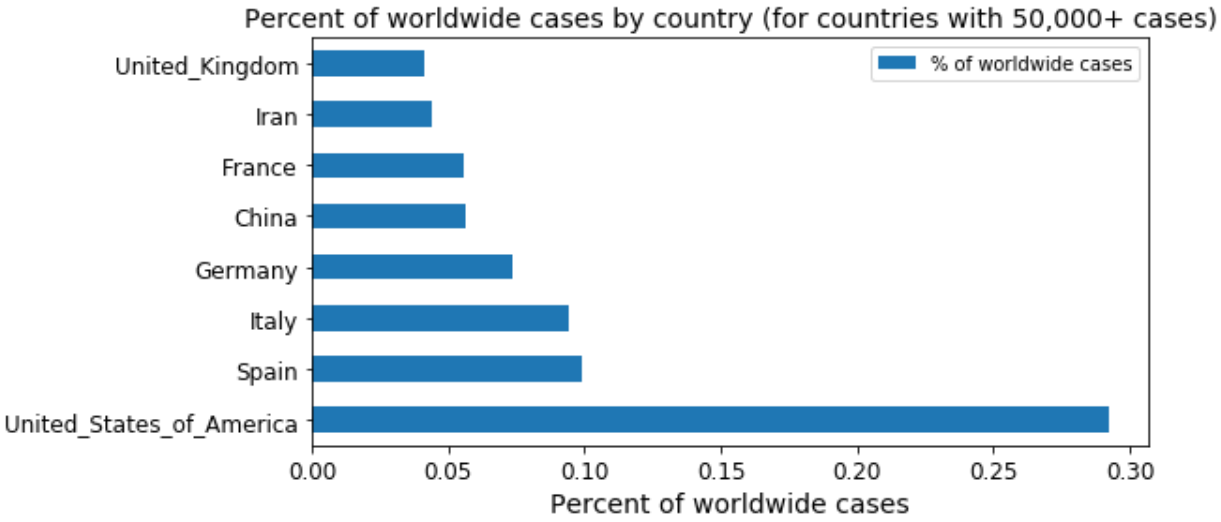
In [60]:
```
# Bar plot of total cases by country (for countries with > 50,000 total cas
ax = over50000.plot.barh(x = "countriesAndTerritories", y = "cases", figsiz
plt.title("Total case count by country (for countries with 50,000+ cases)",
plt.xlabel("Number of total cases", fontsize = 14)
plt.ylabel("")
```

Out[60]:
```
Text(0, 0.5, '')
```



In [61]:
```
# Bar plot of total cases by country (for countries with > 50,000 total cas
ax = over50000.plot.barh(x = "countriesAndTerritories", y = "% of worldwide
plt.title("Percent of worldwide cases by country (for countries with 50,000
plt.xlabel("Percent of worldwide cases", fontsize = 14)
plt.ylabel("")
```

Out[61]:
```
Text(0, 0.5, '')
```



In [62]:
```
over50000["deaths per million"] = over50000["deaths"]/ over50000["popData20
over50000
```

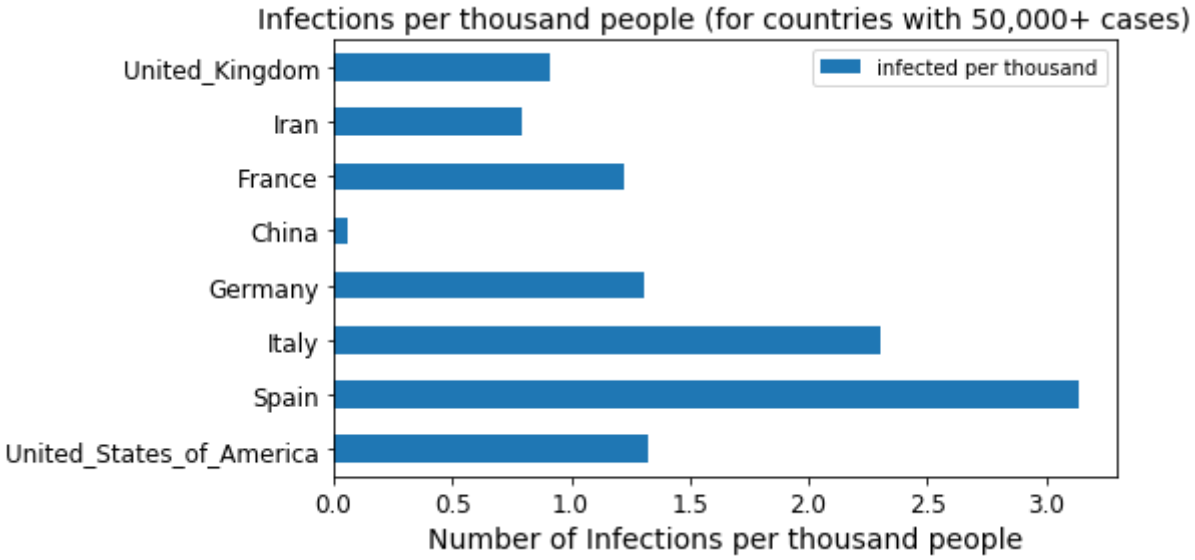
```
/Users/allisonroeser/opt/anaconda3/lib/python3.7/site-packages/ipykernel_
launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy)
  """Entry point for launching an IPython kernel.
```

Out[62]:

	countriesAndTerritories	cases	popData2018	deaths	infected per thousand	deaths per infection	% of worldwide cases	death m
198	United_States_of_America	432132	3.271674e+08	14817	1.320828	0.034288	0.292610	45.28
177	Spain	146690	4.672375e+07	14555	3.139517	0.099223	0.099328	311.51
98	Italy	139422	6.043128e+07	17669	2.307116	0.126730	0.094407	292.38
74	Germany	108202	8.292792e+07	2107	1.304772	0.019473	0.073267	25.40
42	China	82870	1.392730e+09	3339	0.059502	0.040292	0.056114	2.39
69	France	82048	6.698724e+07	10869	1.224830	0.132471	0.055557	162.25
93	Iran	64586	8.180027e+07	3993	0.789557	0.061825	0.043733	48.81
195	United_Kingdom	60733	6.648899e+07	7097	0.913429	0.116856	0.041124	106.73

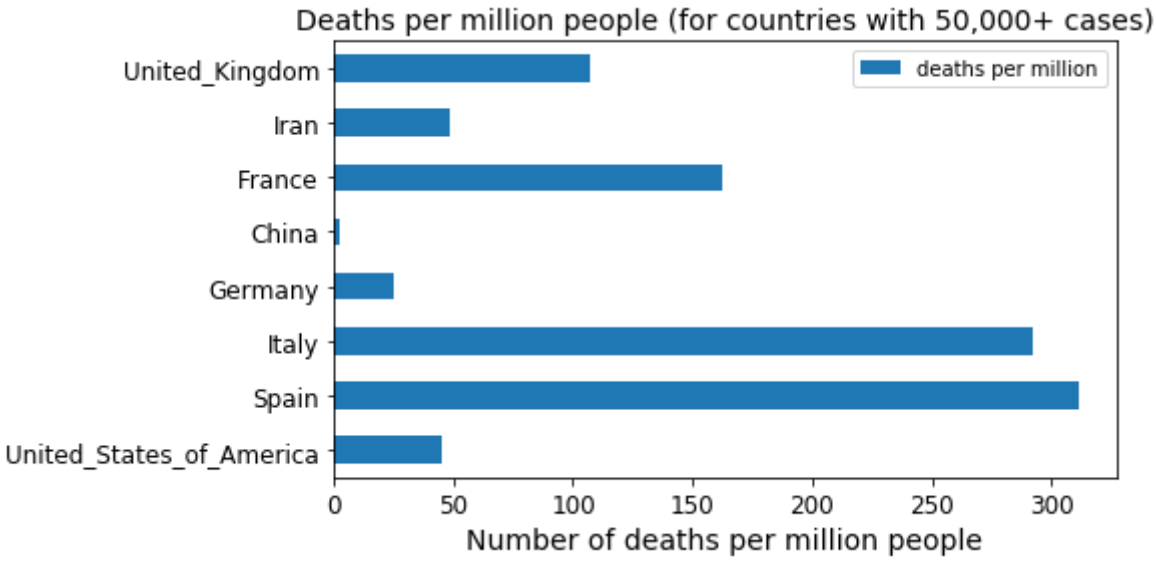
In [63]:
```
# Bar plot of # of infections per million by country (for countries with >
ax = over50000.plot.barh(x = "countriesAndTerritories", y = "infected per t
plt.title("Infections per thousand people (for countries with 50,000+ cases
plt.xlabel("Number of Infections per thousand people", fontsize = 14)
plt.ylabel("")
```

Out[63]:
```
Text(0, 0.5, '')
```



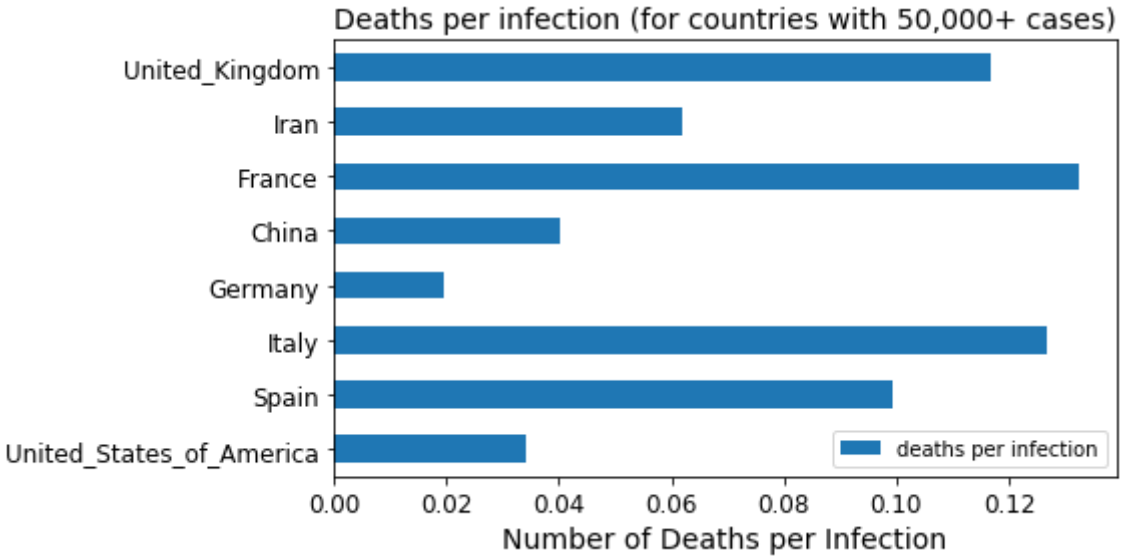
In [64]:
```
# Bar plot of # of infections per million by country (for countries with >
ax = over50000.plot.barh(x = "countriesAndTerritories", y = "deaths per mil
plt.title("Deaths per million people (for countries with 50,000+ cases)", f
plt.xlabel("Number of deaths per million people", fontsize = 14)
plt.ylabel("")
```

Out[64]:
```
Text(0, 0.5, '')
```



In [65]:
```
# Bar plot of death rate by country (for countries with > 50,000 total case
ax = over50000.plot.barh(x = "countriesAndTerritories", y = "deaths per inf
plt.title("Deaths per infection (for countries with 50,000+ cases)", fontsi
plt.xlabel("Number of Deaths per Infection", fontsize = 14)
plt.ylabel("")
```

Out[65]: Text(0, 0.5, '')



In [121]:
```
over50000
```

Out[121]:

	countriesAndTerritories	cases	popData2018	deaths	infected per thousand	deaths per infection	% of worldwide cases	death m
198	United_States_of_America	432132	3.271674e+08	14817	1.320828	0.034288	0.292610	45.28
177	Spain	146690	4.672375e+07	14555	3.139517	0.099223	0.099328	311.51
98	Italy	139422	6.043128e+07	17669	2.307116	0.126730	0.094407	292.38
74	Germany	108202	8.292792e+07	2107	1.304772	0.019473	0.073267	25.40
42	China	82870	1.392730e+09	3339	0.059502	0.040292	0.056114	2.39
69	France	82048	6.698724e+07	10869	1.224830	0.132471	0.055557	162.25
93	Iran	64586	8.180027e+07	3993	0.789557	0.061825	0.043733	48.81
195	United_Kingdom	60733	6.648899e+07	7097	0.913429	0.116856	0.041124	106.73

In [66]:
```
# Pivot the original corona DataFrame on the countriesAndTerritories column
casesPivot = pd.DataFrame(data = corona, columns = ["date", "countriesAndTe
casesPivot = casesPivot.pivot(index = "date", columns ="countriesAndTerrito
casesPivot.head()
```

Out[66]:

countriesAndTerritories	date	Afghanistan	Albania	Algeria	Andorra	Angola	Anguilla	Antigua_a
0	2019-12-31	0.0	NaN	0.0	NaN	NaN	NaN	
1	2020-01-01	0.0	NaN	0.0	NaN	NaN	NaN	
2	2020-01-02	0.0	NaN	0.0	NaN	NaN	NaN	
3	2020-01-03	0.0	NaN	0.0	NaN	NaN	NaN	
4	2020-01-04	0.0	NaN	0.0	NaN	NaN	NaN	

5 rows × 206 columns

In [67]:
```
# Filter the pivoted DataFrame to only include countries with > 50,000 case
mostCases = pd.DataFrame(data = casesPivot, columns = over50000list)
mostCases["date"] = casesPivot["date"]
mostCases
```

Out[67]:

	China	France	Germany	Iran	Italy	Spain	United_Kingdom	United_States_of_America	
0	27.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2
3	17.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2
...	...	...	...	...	...	...	...	...	
96	48.0	4267.0	5936.0	5275.0	4805.0	7026.0	3735.0	34272.0	2
97	67.0	1873.0	3677.0	2483.0	4316.0	6023.0	5903.0	25398.0	2
98	56.0	3912.0	3834.0	2274.0	3599.0	4273.0	3802.0	30561.0	2
99	86.0	3777.0	4003.0	2089.0	3039.0	5478.0	3634.0	30613.0	2
100	86.0	3881.0	4974.0	1997.0	3836.0	6180.0	5491.0	33323.0	2

101 rows × 9 columns

In [68]:
```
# Create a DataFrame with cumulative sum by country from the mostCases Data
mostCases_cumSum = pd.DataFrame(mostCases)
for country in over50000list:
    mostCases_cumSum[str(country)] = mostCases[country].cumsum()
mostCases_cumSum
```

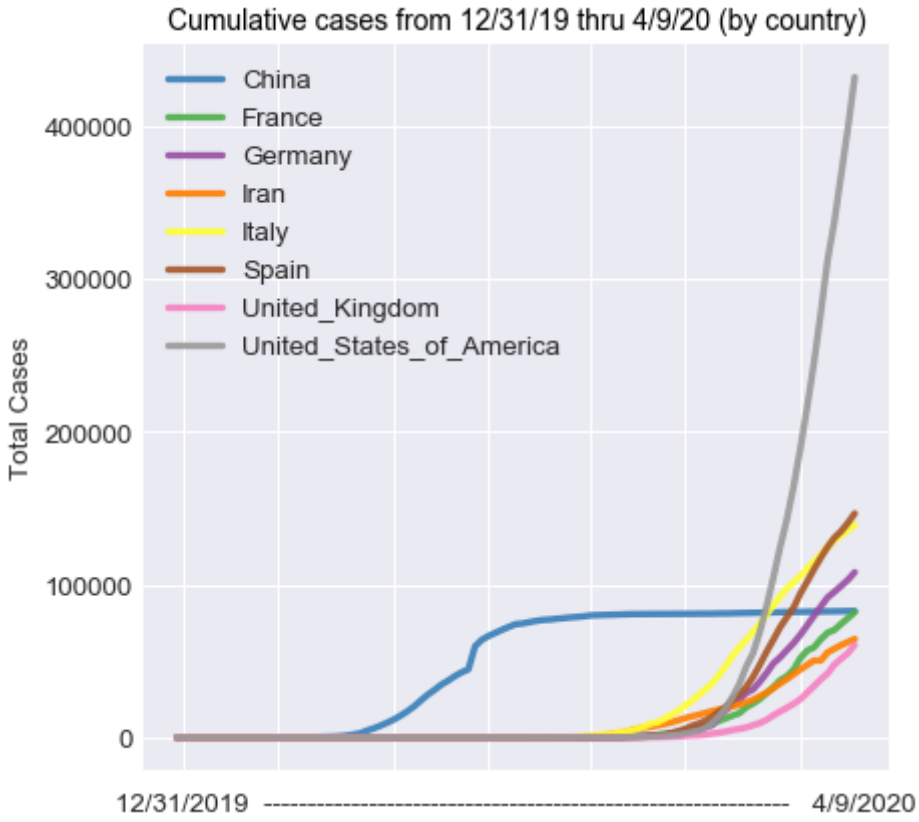
Out[68]:

	China	France	Germany	Iran	Italy	Spain	United_Kingdom	United_States_of_An
0	27.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	27.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	27.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	44.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	44.0	0.0	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	
96	82575.0	68605.0	91714.0	55743.0	124632.0	124736.0	41903.0	312
97	82642.0	70478.0	95391.0	58226.0	128948.0	130759.0	47806.0	337
98	82698.0	74390.0	99225.0	60500.0	132547.0	135032.0	51608.0	368
99	82784.0	78167.0	103228.0	62589.0	135586.0	140510.0	55242.0	398
100	82870.0	82048.0	108202.0	64586.0	139422.0	146690.0	60733.0	432

101 rows × 9 columns

In [69]:
```
# Create a graph of Cumulative Cases from 12/31/19 thru 4/9/20 (by country)
#Holtz, Y. #124 Spaghetti Plot. (n.d.). The Python Graph Gallery. Retrieved
#   https://python-graph-gallery.com/124-spaghetti-plot/


# style
plt.style.use('seaborn-darkgrid')
my_dpi=96
plt.figure(figsize=(480/my_dpi, 480/my_dpi), dpi=my_dpi)

# create a color palette
palette = plt.get_cmap('Set1')

# multiple line plot
num=0
for column in mostCases_cumSum.drop('date', axis=1):
    num+=1
    plt.plot(mostCases_cumSum['date'], mostCases_cumSum[column], marker='',

# Add legend
plt.legend(loc=2, ncol=1)

# Add titles
plt.title("Cumulative cases from 12/31/19 thru 4/9/20 (by country)", loc='c
plt.xlabel("12/31/2019  ----------------------------------------------------
plt.ylabel("Total Cases")

plt.xticks(color='w', fontsize=1)
```

Out[69]:
```
(array([737425., 737439., 737456., 737470., 737485., 737499., 737516.]),
 <a list of 7 Text xticklabel objects>)
```



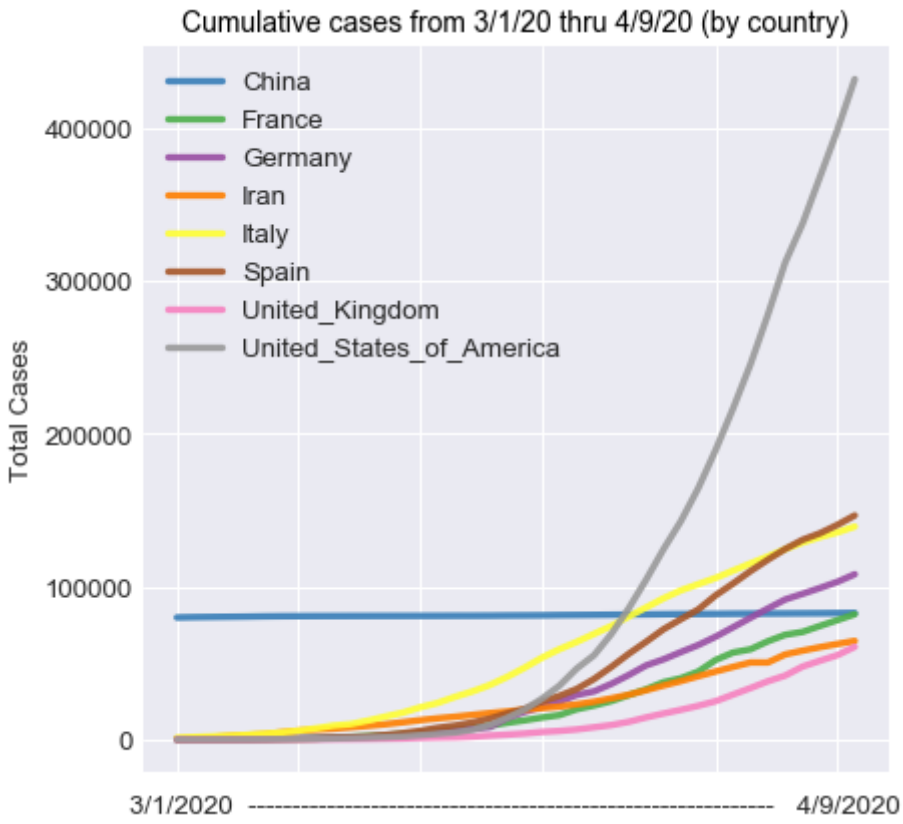
In [70]:
```
# Create a new DataFrame for cumulative cases filtered to dates between 3/1
date = datetime.datetime(2020, 3, 1)
mostCases_Zoom = pd.DataFrame(mostCases_cumSum)
mostCases_Zoom = mostCases[mostCases_cumSum['date'] >= date]
mostCases_Zoom.head()
```

Out[70]:

	China	France	Germany	Iran	Italy	Spain	United_Kingdom	United_States_of_America	
61	79929.0	100.0	111.0	593.0	1128.0	66.0	23.0	69.0	2 0
62	80134.0	130.0	129.0	978.0	1689.0	83.0	36.0	89.0	2 0
63	80261.0	178.0	157.0	1501.0	2036.0	114.0	40.0	103.0	2 0
64	80380.0	212.0	196.0	2336.0	2502.0	151.0	51.0	125.0	2 0
65	80497.0	285.0	262.0	2922.0	3089.0	200.0	85.0	159.0	2 0

In [71]:
```
# Create a graph of Cumulative Cases from 3/1/20 thru 4/8/20 (by country)
#Holtz, Y. #124 Spaghetti Plot. (n.d.). The Python Graph Gallery. Retrieved
#   https://python-graph-gallery.com/124-spaghetti-plot/


# style
plt.style.use('seaborn-darkgrid')
my_dpi=96
plt.figure(figsize=(480/my_dpi, 480/my_dpi), dpi=my_dpi)

# create a color palette
palette = plt.get_cmap('Set1')

# multiple line plot
num=0
for column in mostCases_Zoom.drop('date', axis=1):
    num+=1
    plt.plot(mostCases_Zoom['date'], mostCases_Zoom[column], marker='', col

# Add Legend
plt.legend(loc=2, ncol=1)

# Add titles
plt.title("Cumulative cases from 3/1/20 thru 4/9/20 (by country)", loc='cen
plt.xlabel("3/1/2020  ----------------------------------------------------
plt.ylabel("Total Cases")

plt.xticks(color='w', fontsize=1)
```

Out[71]:
```
(array([737485., 737492., 737499., 737506., 737516., 737523.]),
 <a list of 6 Text xticklabel objects>)
```


Cumulative cases from 3/1/20 thru 4/9/20 (by country)
China
France
Germany
Iran
Italy
Spain
United_Kingdom
United_States_of_America
Total Cases
400000
300000
200000
100000
0
3/1/2020
4/9/2020

In [72]:
```
# Create a DataFrame of log of cumulative cases using the MostCases_cum Dat
mostCaseslog = pd.DataFrame(mostCases_cumSum)
for country in over50000list:
    mostCaseslog[str(country)] = np.log10(mostCases_cumSum[country].replace
mostCaseslog
```

Out[72]:

	China	France	Germany	Iran	Italy	Spain	United_Kingdom	United_States_of
0	1.431364	NaN	NaN	NaN	NaN	NaN	NaN	
1	1.431364	NaN	NaN	NaN	NaN	NaN	NaN	
2	1.431364	NaN	NaN	NaN	NaN	NaN	NaN	
3	1.643453	NaN	NaN	NaN	NaN	NaN	NaN	
4	1.643453	NaN	NaN	NaN	NaN	NaN	NaN	
...	...	...	...	...	...	...	...	
96	4.916849	4.836356	4.962436	4.746190	5.095630	5.095992	4.622245	
97	4.917201	4.848054	4.979507	4.765117	5.110415	5.116472	4.679482	
98	4.917495	4.871515	4.996621	4.781755	5.122370	5.130437	4.712717	
99	4.917946	4.893023	5.013798	4.796498	5.132215	5.147707	4.742269	
100	4.918397	4.914068	5.034235	4.810138	5.144331	5.166401	4.783425	

101 rows × 9 columns

In [73]:
```
# Rename 'United_States_of_America' column to 'USA' (takes up less space or
# Rename 'United_Kingdom' column to 'UK' (takes up less space on graph lege
mostCaseslog.rename(columns = {'United_States_of_America':'USA'}, inplace =
mostCaseslog.rename(columns = {'United_Kingdom':'UK'}, inplace = True)
mostCaseslog
```

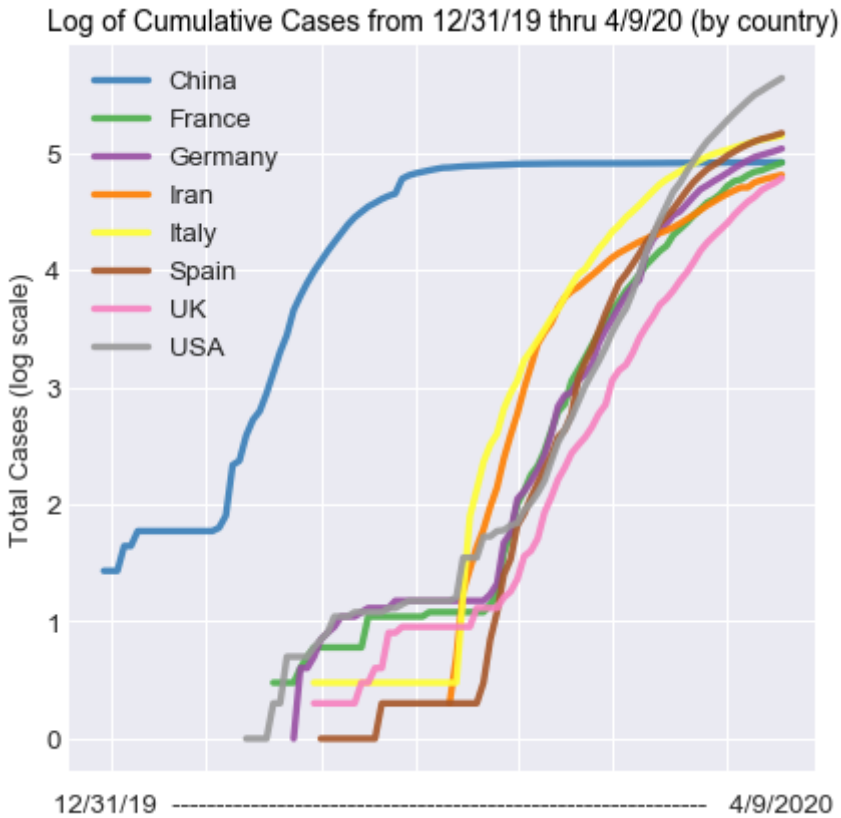
Out[73]:

	China	France	Germany	Iran	Italy	Spain	UK	USA	date
0	1.431364	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2019-12-31
1	1.431364	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2020-01-01
2	1.431364	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2020-01-02
3	1.643453	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2020-01-03
4	1.643453	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2020-01-04
...	...	...	...	...	...	...	...	...	...
96	4.916849	4.836356	4.962436	4.746190	5.095630	5.095992	4.622245	5.494484	2020-04-05
97	4.917201	4.848054	4.979507	4.765117	5.110415	5.116472	4.679482	5.528447	2020-04-06
98	4.917495	4.871515	4.996621	4.781755	5.122370	5.130437	4.712717	5.566079	2020-04-07
99	4.917946	4.893023	5.013798	4.796498	5.132215	5.147707	4.742269	5.600765	2020-04-08
100	4.918397	4.914068	5.034235	4.810138	5.144331	5.166401	4.783425	5.635616	2020-04-09

101 rows × 9 columns

In [115]:
```
# Create a graph of Log of Cumulative Cases from 12/31/19 thru 4/9/20 (by c

# style
plt.style.use('seaborn-darkgrid')
my_dpi=96
plt.figure(figsize=(480/my_dpi, 480/my_dpi), dpi=my_dpi)

# create a color palette
palette = plt.get_cmap('Set1')

# multiple line plot
num=0
for column in mostCaseslog.drop('date', axis=1):
    num+=1
    plt.plot(mostCaseslog['date'], mostCaseslog[column], marker='', color=p

# Add legend
plt.legend(loc=2, ncol=1)

# Add titles
plt.title("Log of Cumulative Cases from 12/31/19 thru 4/9/20 (by country)",
plt.xlabel("12/31/19  ----------------------------------------------------
plt.ylabel("Total Cases (log scale)")

plt.xticks(color='w', fontsize=1)
```

Out[115]:
```
(array([737425., 737439., 737456., 737470., 737485., 737499., 737516.]),
 <a list of 7 Text xticklabel objects>)
```



In [75]:
```
# Calculate the largest daily increase in total cases per country
biggestIncrease = pd.DataFrame(corona)
biggestIncrease = corona.groupby(["countriesAndTerritories"])[["cases"]].ma
biggestIncrease
```

Out[75]:

	countriesAndTerritories	cases
0	Afghanistan	56
1	Albania	29
2	Algeria	314
3	Andorra	43
4	Angola	4
...	...	...
200	Uzbekistan	107
201	Venezuela	48
202	Vietnam	54
203	Zambia	12
204	Zimbabwe	2

205 rows × 2 columns

In [76]:
```
# Determine China's largest daily increase in cases
ChinaMaxIncrease = biggestIncrease[biggestIncrease['countriesAndTerritories
ChinaMaxIncrease
```

Out[76]:

	countriesAndTerritories	cases
42	China	15141

In [77]:
```
# Determine which date the max increase in cases occurred for China
# China reached the peak on 2/13/20
ChinaMaxIncrease = corona[corona['cases'] == 15141]
ChinaMaxIncrease
```

Out[77]:

	dateRep	day	month	year	cases	deaths	countriesAndTerritories	geoId	countryterritory(
1942	13/02/2020	13	2	2020	15141	254	China	CN	

https://towardsdatascience.com/preprocessing-with-sklearn-a-complete-and-comprehensive-guide-670cb98fcfb9 (https://towardsdatascience.com/preprocessing-with-sklearn-a-complete-and-comprehensive-guide-670cb98fcfb9)

https://towardsdatascience.com/preprocessing-with-sklearn-a-complete-and-comprehensive-guide-670cb98fcfb9 (https://towardsdatascience.com/preprocessing-with-sklearn-a-complete-

and-comprehensive-guide-670cb98fcfb9)

Death Rate Anlysis

Daily Death Rate per Million People by Country

In [78]:
```
corona["daily death rate per million"] = corona["deaths"] / corona["popData
```

In [79]:
```
# Pivot the original corona DataFrame on the countriesAndTerritories columr
deathsPivot = pd.DataFrame(data = corona, columns = ["date", "countriesAndT
deathsPivot = deathsPivot.pivot(index = "date", columns ="countriesAndTerri
deathsPivot.tail()
```

Out[79]:

countriesAndTerritories	date	Afghanistan	Albania	Algeria	Andorra	Angola	Anguilla	Antigua
96	2020-04-05	0.026902	0.697745	1.112994	12.986001	0.0	NaN	
97	2020-04-06	0.053803	0.697745	0.520976	12.986001	0.0	NaN	
98	2020-04-07	0.000000	0.348873	0.497295	38.958003	0.0	NaN	
99	2020-04-08	0.107607	0.000000	0.497295	12.986001	0.0	NaN	
100	2020-04-09	0.080705	0.000000	0.260488	12.986001	0.0	NaN	

5 rows × 206 columns

In [80]:
```
# Filter the pivoted DataFrame to only include countries with > 50,000 case
dailyDeathRate = pd.DataFrame(data = deathsPivot, columns = over50000list)
dailyDeathRate["date"] = deathsPivot["date"]
# Rename 'United_States_of_America' column to 'USA' (takes up less space on
dailyDeathRate.rename(columns = {'United_States_of_America':'USA'}, inplace
dailyDeathRate
```

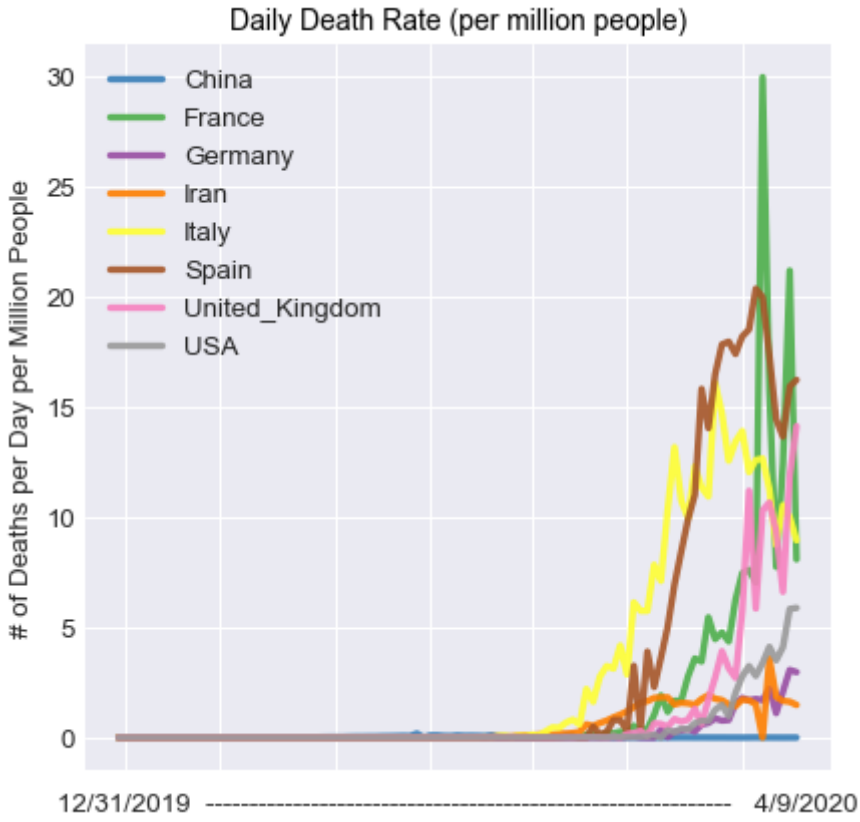
Out[80]:

	China	France	Germany	Iran	Italy	Spain	United_Kingdom	USA	da
0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	20 12
1	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	20 01
2	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	20 01
3	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	20 01
4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	20 01
...	...	...	...	...	...	...	...	...	
96	0.002154	15.719411	2.218794	3.569670	11.268998	17.314535	10.648379	4.107988	20 04
97	0.001436	7.732816	1.109397	1.845960	8.720649	14.425212	9.339892	3.502794	20 04
98	0.000000	12.435203	2.086149	1.662586	10.524350	13.633324	6.602597	4.101875	20 04
99	0.001436	21.153281	3.062901	1.625912	9.994823	15.901977	11.821506	5.825763	20 04
100	0.001436	8.076164	2.966431	1.479213	8.935769	16.201611	14.107599	5.874668	20 04

101 rows × 9 columns

In [116]:
```
# Create a graph of Daily Death Rates per Million people from 12/31/19 thru
#Holtz, Y. #124 Spaghetti Plot. (n.d.). The Python Graph Gallery. Retrieved
#   https://python-graph-gallery.com/124-spaghetti-plot/


# style
plt.style.use('seaborn-darkgrid')
my_dpi=96
plt.figure(figsize=(480/my_dpi, 480/my_dpi), dpi=my_dpi)

# create a color palette
palette = plt.get_cmap('Set1')

# multiple line plot
num=0
for column in dailyDeathRate.drop('date', axis=1):
    num+=1
    plt.plot(dailyDeathRate['date'], dailyDeathRate[column], marker='', col

# Add legend
plt.legend(loc=2, ncol=1)

# Add titles
plt.title("Daily Death Rate (per million people)", loc='center', fontsize=1
plt.xlabel("12/31/2019  ----------------------------------------------------
plt.ylabel("# of Deaths per Day per Million People")

plt.xticks(color='w', fontsize=1)
```

Out[116]:
```
(array([737425., 737439., 737456., 737470., 737485., 737499., 737516.]),
 <a list of 7 Text xticklabel objects>)
```



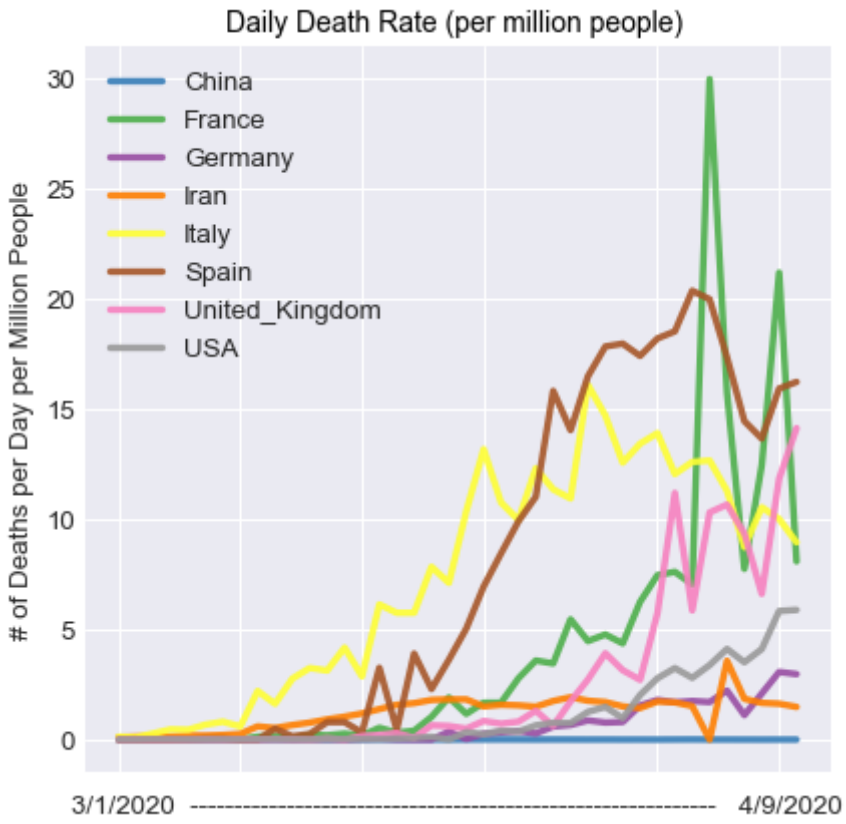
In [82]:
```
# Create a new DataFrame for Daily Death Rates filtered to dates between 3/
date = datetime.datetime(2020, 3, 1)
dailyDeathRate_Zoom = pd.DataFrame(dailyDeathRate)
dailyDeathRate_Zoom = dailyDeathRate[dailyDeathRate['date'] >= date]
dailyDeathRate_Zoom.head()
```

Out[82]:

	China	France	Germany	Iran	Italy	Spain	United_Kingdom	USA	date
61	0.025130	0.000000	0.0	0.110024	0.132382	0.000000	0.0	0.003057	2020-03-01
62	0.030157	0.000000	0.0	0.134474	0.099286	0.000000	0.0	0.003057	2020-03-02
63	0.022976	0.014928	0.0	0.146699	0.281311	0.000000	0.0	0.012226	2020-03-03
64	0.026567	0.014928	0.0	0.134474	0.463336	0.000000	0.0	0.009170	2020-03-04
65	0.022258	0.000000	0.0	0.183373	0.446788	0.021402	0.0	0.006113	2020-03-05

In [117]:
```
# Create a graph of Daily Death Rates per Million people from 3/1/20 thru 4

# style
plt.style.use('seaborn-darkgrid')
my_dpi=96
plt.figure(figsize=(480/my_dpi, 480/my_dpi), dpi=my_dpi)

# create a color palette
palette = plt.get_cmap('Set1')

# multiple line plot
num=0
for column in dailyDeathRate_Zoom.drop('date', axis=1):
    num+=1
    plt.plot(dailyDeathRate_Zoom['date'], dailyDeathRate_Zoom[column], mark

# Add legend
plt.legend(loc=2, ncol=1)

# Add titles
plt.title("Daily Death Rate (per million people) ", loc='center', fontsize=
plt.xlabel("3/1/2020  ----------------------------------------------------
plt.ylabel("# of Deaths per Day per Million People")

plt.xticks(color='w', fontsize=1)
```

Out[117]:
```
(array([737485., 737492., 737499., 737506., 737516., 737523.]),
 <a list of 6 Text xticklabel objects>)
```



Average Death Rates by population groupings

In [84]:
```
over50000
```

Out[84]:

	countriesAndTerritories	cases	popData2018	deaths	infected per thousand	deaths per infection	% of worldwide cases	death m
198	United_States_of_America	432132	3.271674e+08	14817	1.320828	0.034288	0.292610	45.28
177	Spain	146690	4.672375e+07	14555	3.139517	0.099223	0.099328	311.51
98	Italy	139422	6.043128e+07	17669	2.307116	0.126730	0.094407	292.38
74	Germany	108202	8.292792e+07	2107	1.304772	0.019473	0.073267	25.40
42	China	82870	1.392730e+09	3339	0.059502	0.040292	0.056114	2.39
69	France	82048	6.698724e+07	10869	1.224830	0.132471	0.055557	162.25
93	Iran	64586	8.180027e+07	3993	0.789557	0.061825	0.043733	48.81
195	United_Kingdom	60733	6.648899e+07	7097	0.913429	0.116856	0.041124	106.73

In [85]:
```
#Determine quartile values for population data
byCountry.describe()
```

Out[85]:

	cases	popData2018	deaths	infected per thousand	deaths per infection
count	205.000000	2.000000e+02	205.000000	200.000000	205.000000
mean	7203.995122	3.749471e+07	428.370732	1.661673	0.038344
std	35355.268965	1.420721e+08	2122.888806	16.415397	0.047518
min	1.000000	1.000000e+03	0.000000	0.000182	0.000000
25%	25.000000	1.267305e+06	1.000000	0.011425	0.003344
50%	263.000000	7.003150e+06	5.000000	0.076978	0.022222
75%	1623.000000	2.547777e+07	40.000000	0.329080	0.048544
max	432132.000000	1.392730e+09	17669.000000	232.000000	0.250000

In [86]:
```
# Create a new DataFrame for countries with population <= 1,000,000 (0-25th
deaths1stQ = byCountry[byCountry['popData2018'] <=  1000000]
deaths1stQ["Under 1 million"] = deaths1stQ["deaths per infection"]
deaths1stQ.shape
```

```
/Users/allisonroeser/opt/anaconda3/lib/python3.7/site-packages/ipykernel_
launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy)
  This is separate from the ipykernel package so we can avoid doing impor
ts until
```

Out[86]:
```
(48, 7)
```

In [87]:
```
# Create a new DataFrame for countries with population <= 1,000,000 (0-25th
infections1stQ = byCountry[byCountry['popData2018'] <=  1000000]
infections1stQ["Under 1 million"] = deaths1stQ["infected per thousand"]
infections1stQ.head()
```

```
/Users/allisonroeser/opt/anaconda3/lib/python3.7/site-packages/ipykernel_
launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy)
  This is separate from the ipykernel package so we can avoid doing impor
ts until
```

Out[87]:

	countriesAndTerritories	cases	popData2018	deaths	infected per thousand	deaths per infection	Under 1 million
3	Andorra	564	77006.0	23	7.324105	0.040780	7.324105
6	Antigua_and_Barbuda	15	96286.0	0	0.155786	0.000000	0.155786
9	Aruba	77	105845.0	0	0.727479	0.000000	0.727479
13	Bahamas	40	385640.0	7	0.103724	0.175000	0.103724
16	Barbados	63	286641.0	3	0.219787	0.047619	0.219787

In [88]:
```
# Create a new DataFrame for countries with population <= 7,000,000 & > 1,0
deaths2ndQ = byCountry[(byCountry['popData2018'] <= 7000000) & (byCountry['
deaths2ndQ["1 to 7 million"] = deaths2ndQ["deaths per infection"]
deaths2ndQ.shape
```

```
/Users/allisonroeser/opt/anaconda3/lib/python3.7/site-packages/ipykernel_
launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy)
  This is separate from the ipykernel package so we can avoid doing impor
ts until
```

Out[88]:
```
(52, 7)
```

In [89]:
```
# Create a new DataFrame for countries with population <= 7,000,000 & > 1,0
infections2ndQ = byCountry[(byCountry['popData2018'] <= 7000000) & (byCount
infections2ndQ["1 to 7 million"] = infections2ndQ["infected per thousand"]
infections2ndQ.head()
```

```
/Users/allisonroeser/opt/anaconda3/lib/python3.7/site-packages/ipykernel_
launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy)
  This is separate from the ipykernel package so we can avoid doing impor
ts until
```

Out[89]:

	countriesAndTerritories	cases	popData2018	deaths	infected per thousand	deaths per infection	1 to 7 million
1	Albania	400	2866376.0	22	0.139549	0.055000	0.139549
8	Armenia	881	2951776.0	9	0.298464	0.010216	0.298464
14	Bahrain	823	1569439.0	5	0.524391	0.006075	0.524391
25	Bosnia_and_Herzegovina	816	3323929.0	35	0.245493	0.042892	0.245493
26	Botswana	7	2254126.0	1	0.003105	0.142857	0.003105

In [90]:
```
# Create a new DataFrame for countries with population <= 25,000,000 & > 7,
deaths3rdQ = byCountry[(byCountry['popData2018'] <= 25000000) & (byCountry[
deaths3rdQ["7 to 25 million"] = deaths3rdQ["deaths per infection"]
deaths3rdQ.shape
```

```
/Users/allisonroeser/opt/anaconda3/lib/python3.7/site-packages/ipykernel_
launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy)
  This is separate from the ipykernel package so we can avoid doing impor
ts until
```

Out[90]:
```
(48, 7)
```

In [91]:
```
# Create a new DataFrame for countries with population <= 25,000,000 & > 7,
infections3rdQ = byCountry[(byCountry['popData2018'] <= 25000000) & (byCoun
infections3rdQ["7 to 25 million"] = deaths3rdQ["infected per thousand"]
infections3rdQ.describe()
```

```
/Users/allisonroeser/opt/anaconda3/lib/python3.7/site-packages/ipykernel_
launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy)
  This is separate from the ipykernel package so we can avoid doing impor
ts until
```

Out[91]:

	cases	popData2018	deaths	infected per thousand	deaths per infection	7 to 25 million
count	48.000000	4.800000e+01	48.000000	48.000000	48.000000	48.000000
mean	3055.750000	1.352108e+07	158.854167	0.270381	0.037494	0.270381
std	5957.008304	4.736906e+06	465.538672	0.570116	0.040293	0.570116
min	2.000000	7.024216e+06	0.000000	0.000182	0.000000	0.000182
25%	36.000000	9.898947e+06	1.000000	0.002386	0.008035	0.002386
50%	350.500000	1.145356e+07	7.500000	0.022107	0.029980	0.022107
75%	2248.000000	1.723521e+07	54.000000	0.209520	0.051966	0.209520
max	23403.000000	2.499237e+07	2248.000000	2.666575	0.181818	2.666575

In [92]:
```
# Create a new DataFrame for countries with population > 25,000,000  (75th-
deathsTopQ = byCountry[byCountry['popData2018'] > 25000000]
deathsTopQ["More Than 25 million"] = deathsTopQ["deaths per infection"]
deathsTopQ.head()
```

```
/Users/allisonroeser/opt/anaconda3/lib/python3.7/site-packages/ipykernel_
launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy)
  This is separate from the ipykernel package so we can avoid doing impor
ts until
```

Out[92]:

	countriesAndTerritories	cases	popData2018	deaths	infected per thousand	deaths per infection	More Than 25 million
0	Afghanistan	423	37172386.0	14	0.011379	0.033097	0.033097
2	Algeria	1572	42228429.0	205	0.037226	0.130407	0.130407
4	Angola	19	30809762.0	2	0.000617	0.105263	0.105263
7	Argentina	1795	44494502.0	65	0.040342	0.036212	0.036212
15	Bangladesh	218	161356039.0	20	0.001351	0.091743	0.091743

In [93]:
```
# Create a new DataFrame for countries with population > 25,000,000 (75th-1
infectionsTopQ = byCountry[byCountry['popData2018'] > 25000000]
infectionsTopQ["More Than 25 million"] = deathsTopQ["infected per thousand"
infectionsTopQ.describe()
```

```
/Users/allisonroeser/opt/anaconda3/lib/python3.7/site-packages/ipykernel_
launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy)
  This is separate from the ipykernel package so we can avoid doing impor
ts until
```

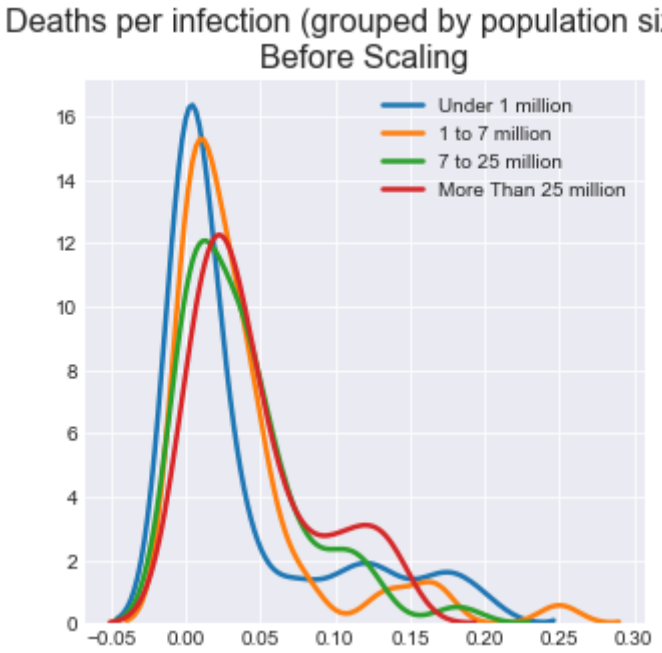
Out[93]:

	cases	popData2018	deaths	infected per thousand	deaths per infection	More Than 25 million
count	52.000000	5.200000e+01	52.000000	52.000000	52.000000	52.000000
mean	24413.519231	1.277709e+08	1516.346154	0.258012	0.045650	0.258012
std	67496.201128	2.596468e+08	4024.127945	0.606185	0.041368	0.606185
min	9.000000	2.506923e+07	0.000000	0.000320	0.000000	0.000320
25%	242.750000	3.680143e+07	6.000000	0.003527	0.015480	0.003527
50%	1820.000000	5.267183e+07	63.000000	0.031603	0.032136	0.031603
75%	6468.500000	1.004807e+08	204.250000	0.097913	0.062875	0.097913
max	432132.000000	1.392730e+09	17669.000000	3.139517	0.142857	3.139517

In [94]:
```
fig, (ax1) = plt.subplots(ncols=1, figsize=(5, 5))
fig.suptitle('Deaths per infection (grouped by population size)', fontsize
ax1.set_title('Before Scaling', fontsize = 16)
ax1.legend(fontsize='x-large', title_fontsize='40')
sns.kdeplot(deaths1stQ['Under 1 million'], ax=ax1, linewidth=2.5)
sns.kdeplot(deaths2ndQ['1 to 7 million'], ax=ax1, linewidth=2.5)
sns.kdeplot(deaths3rdQ['7 to 25 million'], ax=ax1, linewidth=2.5)
sns.kdeplot(deathsTopQ['More Than 25 million'], ax=ax1, linewidth=2.5)
```

```
No handles with labels found to put in legend.
```

Out[94]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x130fd9210>
```



In [95]:
```
byCountry
```

Out[95]:

	countriesAndTerritories	cases	popData2018	deaths	infected per thousand	deaths per infection
0	Afghanistan	423	37172386.0	14	0.011379	0.033097
1	Albania	400	2866376.0	22	0.139549	0.055000
2	Algeria	1572	42228429.0	205	0.037226	0.130407
3	Andorra	564	77006.0	23	7.324105	0.040780
4	Angola	19	30809762.0	2	0.000617	0.105263
...	...	...	...	...	...	...
200	Uzbekistan	555	32955400.0	3	0.016841	0.005405
201	Venezuela	167	28870195.0	8	0.005785	0.047904
202	Vietnam	251	95540395.0	0	0.002627	0.000000
203	Zambia	39	17351822.0	1	0.002248	0.025641
204	Zimbabwe	11	14439018.0	2	0.000762	0.181818

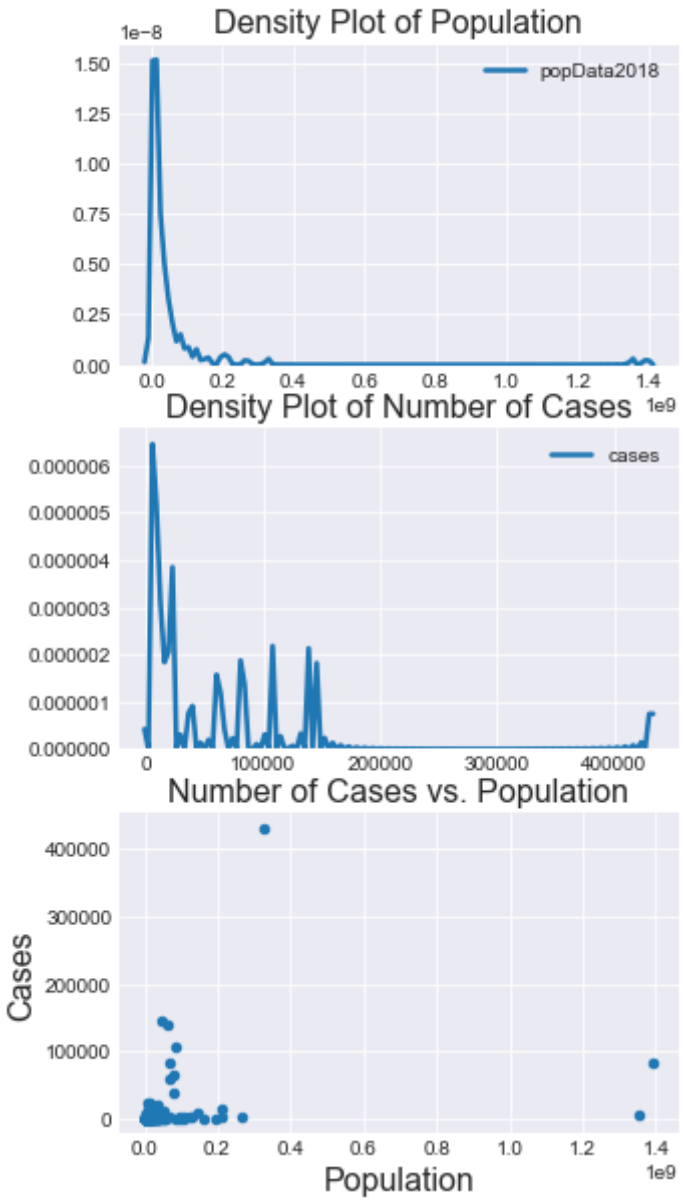
205 rows × 6 columns

In [96]:
```
fig, (ax1, ax2, ax3) = plt.subplots(nrows=3, figsize=(5, 10))
fig.suptitle('Before Scaling', fontsize = 16)
ax1.set_title('Density Plot of Population', fontsize = 16)
ax2.set_title('Density Plot of Number of Cases', fontsize = 16)
ax3.set_title('Number of Cases vs. Population', fontsize = 16)
ax1.legend(fontsize='x-large', title_fontsize='40')
sns.kdeplot(byCountry['popData2018'], ax=ax1, linewidth=2.5)
sns.kdeplot(byCountry['cases'], ax=ax2, linewidth=2.5)
byCountry.plot.scatter('popData2018', "cases", ax=ax3)
ax3.set_xlabel('Population', fontsize = 16)
ax3.set_ylabel('Cases', fontsize = 16)
```

```
No handles with labels found to put in legend.
/Users/allisonroeser/opt/anaconda3/lib/python3.7/site-packages/statsmodel
s/nonparametric/kde.py:447: RuntimeWarning: invalid value encountered in
greater
  X = X[np.logical_and(X > clip[0], X < clip[1])] # won't work for two co
lumns.
/Users/allisonroeser/opt/anaconda3/lib/python3.7/site-packages/statsmodel
s/nonparametric/kde.py:447: RuntimeWarning: invalid value encountered in
less
  X = X[np.logical_and(X > clip[0], X < clip[1])] # won't work for two co
lumns.
```

Out[96]:
```
Text(0, 0.5, 'Cases')
```

Before Scaling

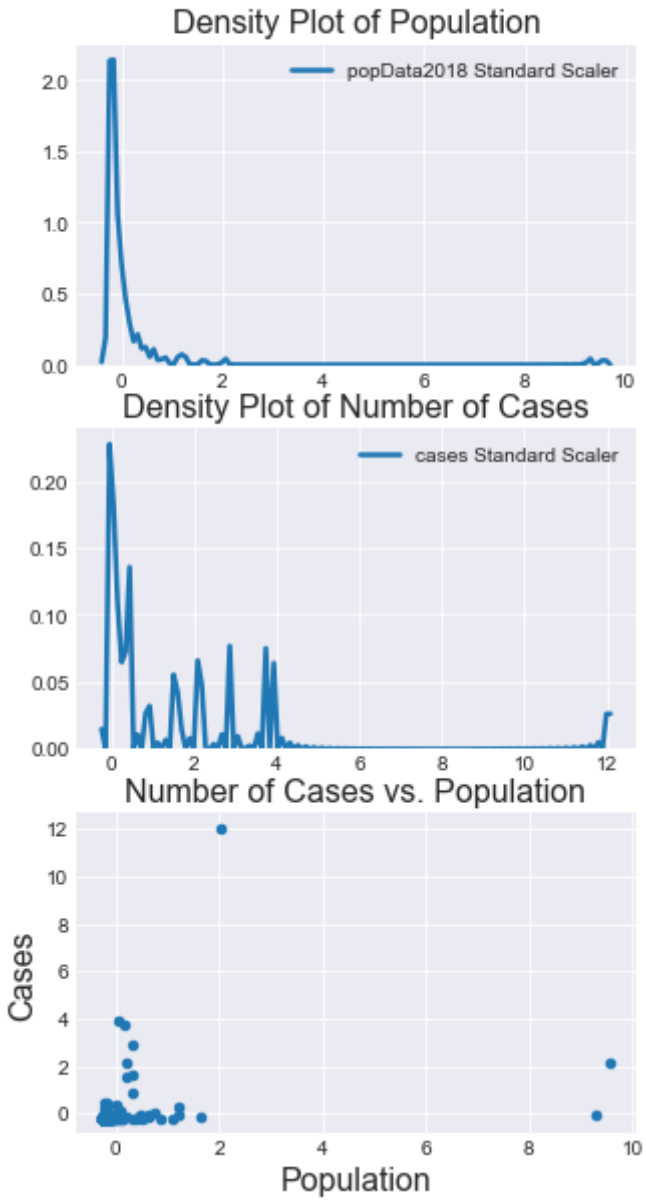


In [97]:
```
fig, (ax1, ax2, ax3) = plt.subplots(nrows=3, figsize=(5, 10))
scaler = StandardScaler()
byCountry["popData2018 Standard Scaler"] = scaler.fit_transform(byCountry["
byCountry["cases Standard Scaler"] = scaler.fit_transform(byCountry["cases"
fig.suptitle('After Standard Scaling', fontsize = 16)
ax1.set_title('Density Plot of Population', fontsize = 16)
ax2.set_title('Density Plot of Number of Cases', fontsize = 16)
ax3.set_title('Number of Cases vs. Population', fontsize = 16)
ax1.legend(fontsize='x-large', title_fontsize='40')
sns.kdeplot(byCountry['popData2018 Standard Scaler'], ax=ax1, linewidth=2.5
sns.kdeplot(byCountry['cases Standard Scaler'], ax=ax2, linewidth=2.5)
byCountry.plot.scatter('popData2018 Standard Scaler', "cases Standard Scale
ax3.set_xlabel('Population', fontsize = 16)
ax3.set_ylabel('Cases', fontsize = 16)
```

```
No handles with labels found to put in legend.
/Users/allisonroeser/opt/anaconda3/lib/python3.7/site-packages/statsmodel
s/nonparametric/kde.py:447: RuntimeWarning: invalid value encountered in
greater
  X = X[np.logical_and(X > clip[0], X < clip[1])] # won't work for two co
lumns.
/Users/allisonroeser/opt/anaconda3/lib/python3.7/site-packages/statsmodel
s/nonparametric/kde.py:447: RuntimeWarning: invalid value encountered in
less
  X = X[np.logical_and(X > clip[0], X < clip[1])] # won't work for two co
lumns.
```

Out[97]:
```
Text(0, 0.5, 'Cases')
```

After Standard Scaling

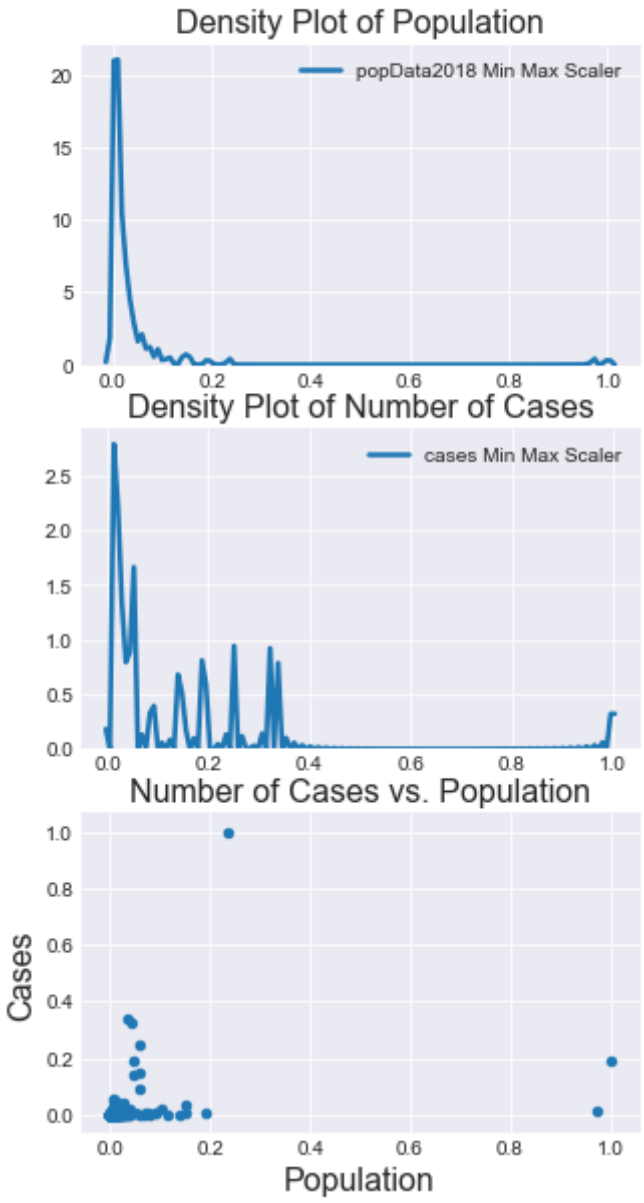


In [114]:
```
fig, (ax1, ax2, ax3) = plt.subplots(nrows=3, figsize=(5, 10))
scaler = MinMaxScaler()
byCountry["popData2018 Min Max Scaler"] = scaler.fit_transform(byCountry["p
byCountry["cases Min Max Scaler"] = scaler.fit_transform(byCountry["cases"]
fig.suptitle('After Min Max Scaling', fontsize = 16)
ax1.set_title('Density Plot of Population', fontsize = 16)
ax3.set_title('Number of Cases vs. Population', fontsize = 16)
ax2.set_title('Density Plot of Number of Cases', fontsize = 16)
ax1.legend(fontsize='x-large', title_fontsize='40')
sns.kdeplot(byCountry['popData2018 Min Max Scaler'], ax=ax1, linewidth=2.5)
sns.kdeplot(byCountry['cases Min Max Scaler'], ax=ax2, linewidth=2.5)
byCountry.plot.scatter('popData2018 Min Max Scaler', "cases Min Max Scaler"
ax3.set_xlabel('Population', fontsize = 16)
ax3.set_ylabel('Cases', fontsize = 16)
```

```
No handles with labels found to put in legend.
/Users/allisonroeser/opt/anaconda3/lib/python3.7/site-packages/statsmodel
s/nonparametric/kde.py:447: RuntimeWarning: invalid value encountered in
greater
  X = X[np.logical_and(X > clip[0], X < clip[1])] # won't work for two co
lumns.
/Users/allisonroeser/opt/anaconda3/lib/python3.7/site-packages/statsmodel
s/nonparametric/kde.py:447: RuntimeWarning: invalid value encountered in
less
  X = X[np.logical_and(X > clip[0], X < clip[1])] # won't work for two co
lumns.
```

Out[114]:
```
Text(0, 0.5, 'Cases')
```

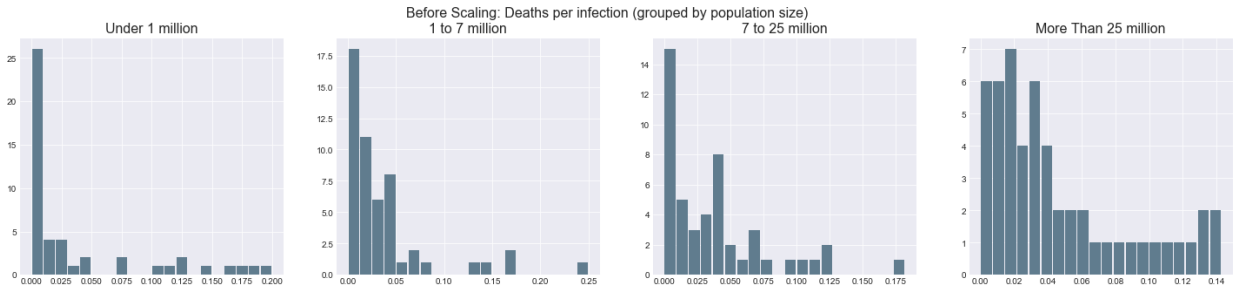
After Min Max Scaling



In [99]:
```
fig, (ax1, ax2, ax3, ax4) = plt.subplots(ncols=4, figsize=(25, 5))
fig.suptitle('Before Scaling: Deaths per infection (grouped by population s
ax1.set_title('Under 1 million', fontsize = 16)
ax2.set_title('1 to 7 million', fontsize = 16)
ax3.set_title('7 to 25 million', fontsize = 16)
ax4.set_title('More Than 25 million', fontsize = 16)
ax1.legend(fontsize='x-large', title_fontsize='40')
deaths1stQ['Under 1 million'].hist(grid=True, bins=20, rwidth=0.9,color='#6
deaths2ndQ['1 to 7 million'].hist(grid=True, bins=20, rwidth=0.9,color='#60
deaths3rdQ['7 to 25 million'].hist(grid=True, bins=20, rwidth=0.9,color='#6
deathsTopQ['More Than 25 million'].hist(grid=True, bins=20, rwidth=0.9,colo
```

```
No handles with labels found to put in legend.
```

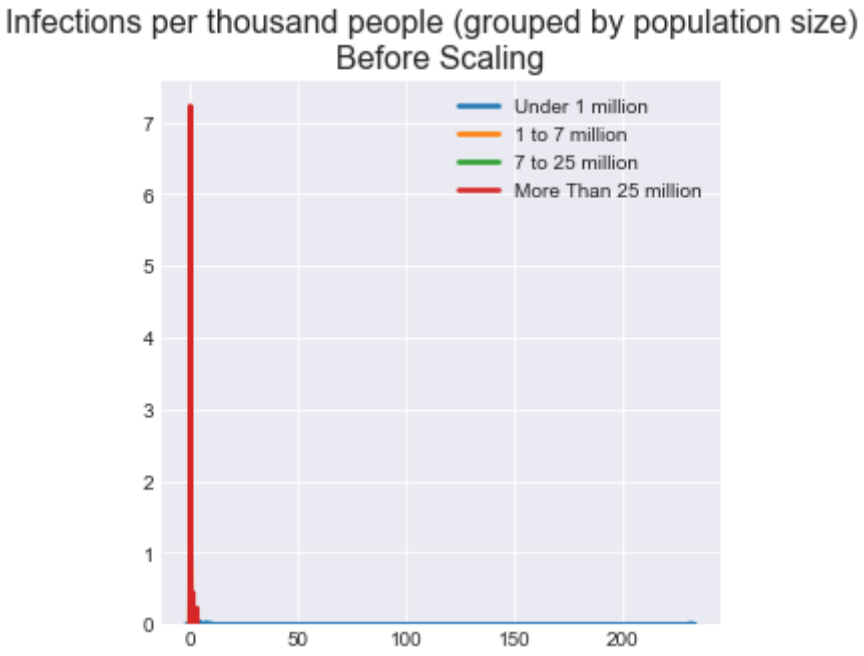
Out[99]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1317f6410>
```



In [100]:
```
#Dzindo, I. Feature Scaling in Python. (May 23, 2018). Medium. Retrieved fr
#https://medium.com/@ian.dzindo01/feature-scaling-in-python-a59cc72147c1


fig, (ax1) = plt.subplots(ncols=1, figsize=(5, 5))
fig.suptitle('Infections per thousand people (grouped by population size)',
ax1.set_title('Before Scaling', fontsize = 16)
ax1.legend(fontsize='x-large', title_fontsize='40')
sns.kdeplot(infections1stQ['Under 1 million'], ax=ax1, linewidth=2.5)
sns.kdeplot(infections2ndQ['1 to 7 million'], ax=ax1, linewidth=2.5)
sns.kdeplot(infections3rdQ['7 to 25 million'], ax=ax1, linewidth=2.5)
sns.kdeplot(infectionsTopQ['More Than 25 million'], ax=ax1, linewidth=2.5)
```

```
No handles with labels found to put in legend.
```

Out[100]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x131c26610>
```

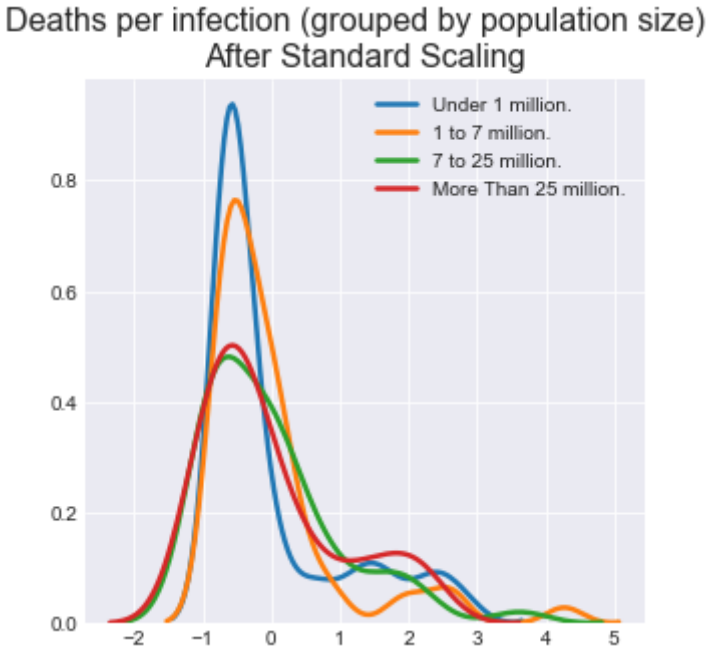


In [101]:
```
#Dzindo, I. Feature Scaling in Python. (May 23, 2018). Medium. Retrieved fr
#https://medium.com/@ian.dzindo01/feature-scaling-in-python-a59cc72147c1

fig, (ax2) = plt.subplots(ncols=1, figsize=(5, 5))
fig.suptitle('Deaths per infection (grouped by population size)', fontsize
ax1.set_title('After Standard Scaling', fontsize = 16)
ax1.legend(fontsize='x-large', title_fontsize='40')

scaler = StandardScaler()
deaths1stQ["Under 1 million."] = scaler.fit_transform(deaths1stQ['Under 1 m
deaths2ndQ["1 to 7 million."] = scaler.fit_transform(deaths2ndQ['1 to 7 mil
deaths3rdQ["7 to 25 million."] = scaler.fit_transform(deaths3rdQ['7 to 25 m
deathsTopQ["More Than 25 million."] = scaler.fit_transform(deathsTopQ['More
ax2.set_title('After Standard Scaling', fontsize = 16)
sns.kdeplot(deaths1stQ['Under 1 million.'], ax=ax2, linewidth=2.5)
sns.kdeplot(deaths2ndQ['1 to 7 million.'], ax=ax2, linewidth=2.5)
sns.kdeplot(deaths3rdQ['7 to 25 million.'], ax=ax2, linewidth=2.5)
sns.kdeplot(deathsTopQ['More Than 25 million.'], ax=ax2, linewidth=2.5)
```

```
/Users/allisonroeser/opt/anaconda3/lib/python3.7/site-packages/ipykernel_
launcher.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy)
  import sys
/Users/allisonroeser/opt/anaconda3/lib/python3.7/site-packages/ipykernel_
launcher.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy)

/Users/allisonroeser/opt/anaconda3/lib/python3.7/site-packages/ipykernel_
launcher.py:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy)
  if __name__ == '__main__':
/Users/allisonroeser/opt/anaconda3/lib/python3.7/site-packages/ipykernel_
launcher.py:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http://
```

```
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  # Remove the CWD from sys.path while we load stuff.
```

Out[101]: <matplotlib.axes._subplots.AxesSubplot at 0x131eb73d0>

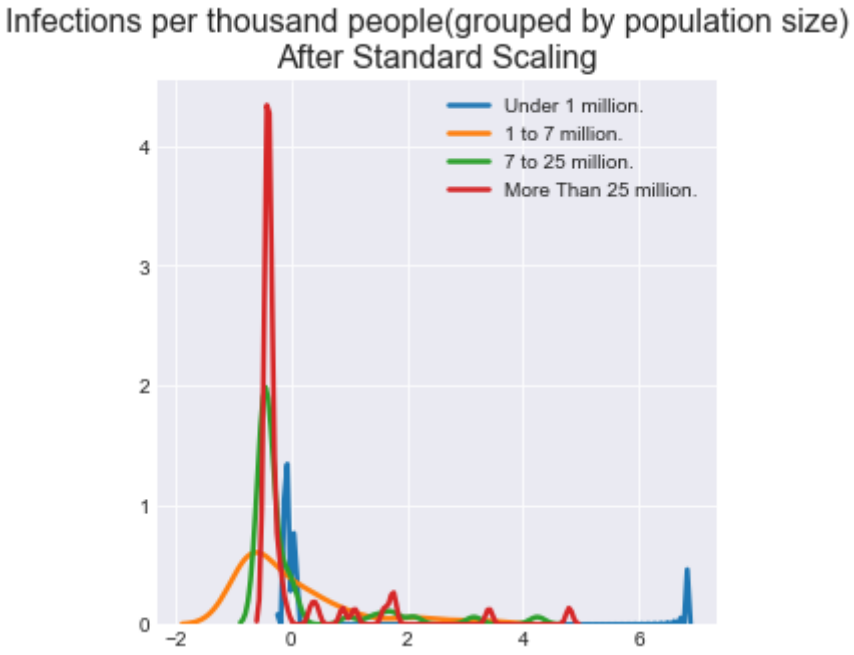


In [112]:
```
indo, I. Feature Scaling in Python. (May 23, 2018). Medium. Retrieved from
tps://medium.com/@ian.dzindo01/feature-scaling-in-python-a59cc72147c1

, (ax2) = plt.subplots(ncols=1, figsize=(5, 5))
.suptitle('Infections per thousand people(grouped by population size)', font
.set_title('After Standard Scaling', fontsize = 16)
.legend(fontsize='x-large', title_fontsize='40')

ler = StandardScaler()
ections1stQ["Under 1 million."] = scaler.fit_transform(infections1stQ['Under
ections2ndQ["1 to 7 million."] = scaler.fit_transform(infections2ndQ['1 to 7
ections3rdQ["7 to 25 million."] = scaler.fit_transform(infections3rdQ['7 to
ectionsTopQ["More Than 25 million."] = scaler.fit_transform(infectionsTopQ['
.set_title('After Standard Scaling', fontsize = 16)
.kdeplot(infections1stQ['Under 1 million.'], ax=ax2, linewidth=2.5)
.kdeplot(infections2ndQ['1 to 7 million.'], ax=ax2, linewidth=2.5)
.kdeplot(infections3rdQ['7 to 25 million.'], ax=ax2, linewidth=2.5)
.kdeplot(infectionsTopQ['More Than 25 million.'], ax=ax2, linewidth=2.5)
```

```
/Users/allisonroeser/opt/anaconda3/lib/python3.7/site-packages/ipykernel_
launcher.py:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy)
  # Remove the CWD from sys.path while we load stuff.
/Users/allisonroeser/opt/anaconda3/lib/python3.7/site-packages/ipykernel_
launcher.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy)
  # This is added back by InteractiveShellApp.init_path()
/Users/allisonroeser/opt/anaconda3/lib/python3.7/site-packages/ipykernel_
launcher.py:12: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy)
  if sys.path[0] == '':
/Users/allisonroeser/opt/anaconda3/lib/python3.7/site-packages/ipykernel_
launcher.py:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
```

```
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy)
  del sys.path[0]
```

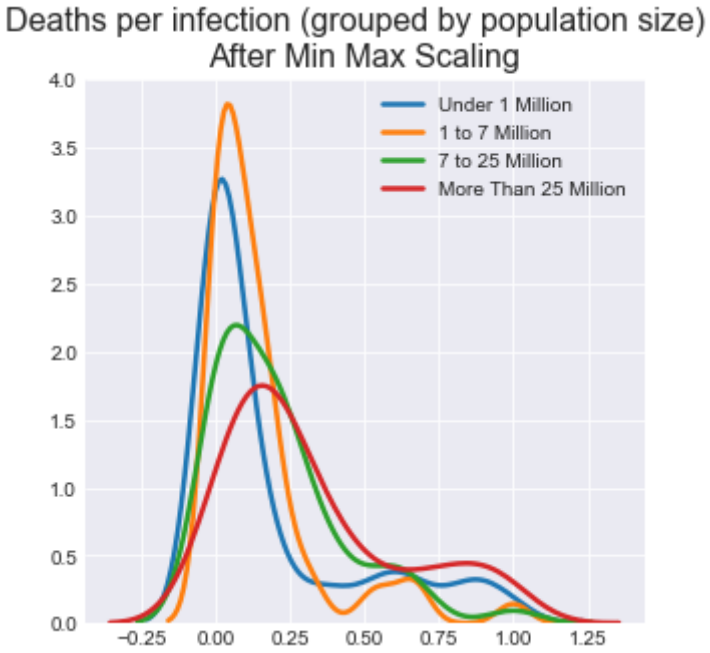
Out[112]: <matplotlib.axes._subplots.AxesSubplot at 0x12fbcf690>



In [103]:
```
fig, (ax3) = plt.subplots(ncols=1, figsize=(5, 5))
fig.suptitle('Deaths per infection (grouped by population size)', fontsize
ax1.set_title('After Standard Scaling', fontsize = 16)
ax1.legend(fontsize='x-large', title_fontsize='40')

scaler = MinMaxScaler()
deaths1stQ["Under 1 Million"] = scaler.fit_transform(deaths1stQ['Under 1 mi
deaths2ndQ["1 to 7 Million"] = scaler.fit_transform(deaths2ndQ['1 to 7 mill
deaths3rdQ["7 to 25 Million"] = scaler.fit_transform(deaths3rdQ['7 to 25 mi
deathsTopQ["More Than 25 Million"] = scaler.fit_transform(deathsTopQ['More
ax3.set_title('After Min Max Scaling', fontsize = 16)
sns.kdeplot(deaths1stQ['Under 1 Million'], ax=ax3, linewidth=2.5)
sns.kdeplot(deaths2ndQ['1 to 7 Million'], ax=ax3, linewidth=2.5)
sns.kdeplot(deaths3rdQ['7 to 25 Million'], ax=ax3, linewidth=2.5)
sns.kdeplot(deathsTopQ['More Than 25 Million'], ax=ax3, linewidth=2.5)
```

```
/Users/allisonroeser/opt/anaconda3/lib/python3.7/site-packages/ipykernel_
launcher.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy)
  import sys
/Users/allisonroeser/opt/anaconda3/lib/python3.7/site-packages/ipykernel_
launcher.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy)

/Users/allisonroeser/opt/anaconda3/lib/python3.7/site-packages/ipykernel_
launcher.py:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy)
  if __name__ == '__main__':
/Users/allisonroeser/opt/anaconda3/lib/python3.7/site-packages/ipykernel_
launcher.py:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy)
  # Remove the CWD from sys.path while we load stuff.
```

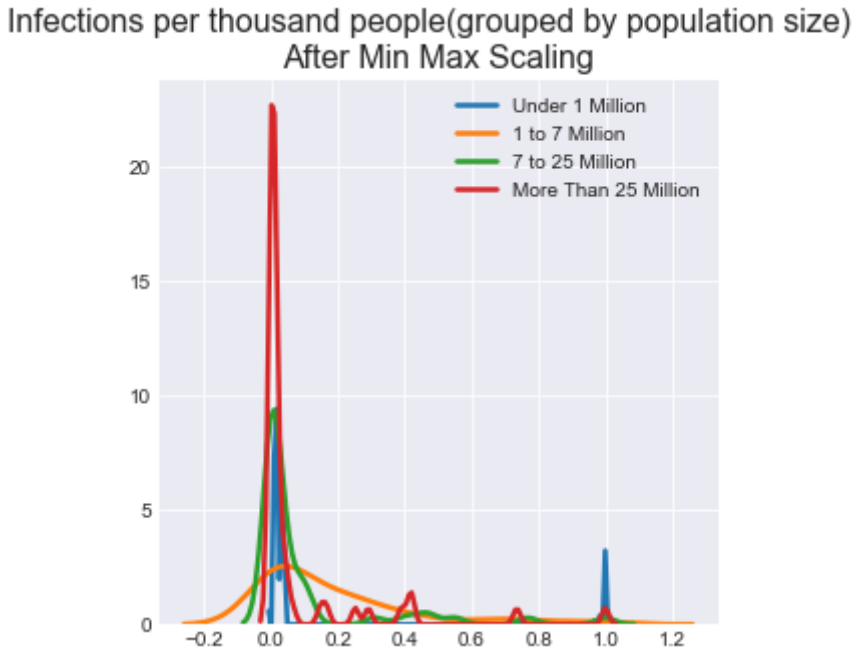
Out[103]: <matplotlib.axes._subplots.AxesSubplot at 0x132131c10>



In [104]:
```
fig, (ax3) = plt.subplots(ncols=1, figsize=(5, 5))
fig.suptitle('Infections per thousand people(grouped by population size)',
ax1.set_title('After Standard Scaling', fontsize = 16)
ax1.legend(fontsize='x-large', title_fontsize='40')

scaler = MinMaxScaler()
infections1stQ["Under 1 Million"] = scaler.fit_transform(infections1stQ['Un
infections2ndQ["1 to 7 Million"] = scaler.fit_transform(infections2ndQ['1 t
infections3rdQ["7 to 25 Million"] = scaler.fit_transform(infections3rdQ['7
infectionsTopQ["More Than 25 Million"] = scaler.fit_transform(infectionsTop
ax3.set_title('After Min Max Scaling', fontsize = 16)
sns.kdeplot(infections1stQ['Under 1 Million'], ax=ax3, linewidth=2.5)
sns.kdeplot(infections2ndQ['1 to 7 Million'], ax=ax3, linewidth=2.5)
sns.kdeplot(infections3rdQ['7 to 25 Million'], ax=ax3, linewidth=2.5)
sns.kdeplot(infectionsTopQ['More Than 25 Million'], ax=ax3, linewidth=2.5)
```

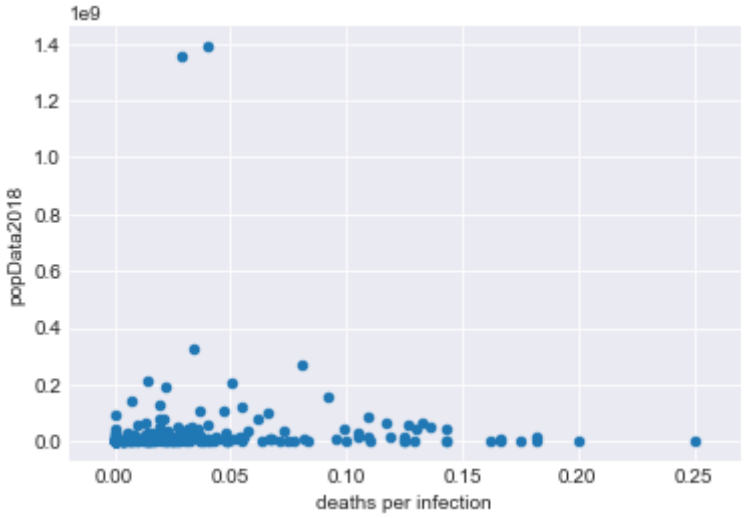
```
/Users/allisonroeser/opt/anaconda3/lib/python3.7/site-packages/ipykernel_
launcher.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy)
  import sys
/Users/allisonroeser/opt/anaconda3/lib/python3.7/site-packages/ipykernel_
launcher.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy)

/Users/allisonroeser/opt/anaconda3/lib/python3.7/site-packages/ipykernel_
launcher.py:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy)
  if __name__ == '__main__':
/Users/allisonroeser/opt/anaconda3/lib/python3.7/site-packages/ipykernel_
launcher.py:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy)
  # Remove the CWD from sys.path while we load stuff.
```

Out[104]: <matplotlib.axes._subplots.AxesSubplot at 0x1322cb910>



In [105]:
```
byCountry.plot.scatter(x = "deaths per infection", y = "popData2018")
```

Out[105]: <matplotlib.axes._subplots.AxesSubplot at 0x13230f290>



In [ ]:
```
```

In [106]:
```
scaler = StandardScaler()
byCountry["deathsStandardScaler"] = scaler.fit_transform(byCountry["deaths
```