COVID - 19

In [1]:
```
#!pip install -q pycountry
#!pip install gensim
#!pip install textblob
#!pip install wordcloud
#!pip install plotly
```

Importing Necessary Modules

In [74]:
```
from scipy import stats
import warnings
warnings.filterwarnings("ignore")

import math
import numpy as np
import scipy as sp
import pandas as pd

import pycountry
from sklearn import metrics
from sklearn.utils import shuffle
from gensim.models import Word2Vec
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA

import nltk
from textblob import TextBlob
from wordcloud import WordCloud
from nltk.corpus import wordnet
from nltk.corpus import stopwords
from nltk import WordNetLemmatizer
from nltk.stem import WordNetLemmatizer
from nltk.sentiment.vader import SentimentIntensityAnalyzer

import random
import networkx as nx
from pandas import Timestamp

import requests
from IPython.display import HTML
```

In [75]:
```
import seaborn as sns
from tqdm import tqdm
import matplotlib.cm as cm
import matplotlib.pyplot as plt

import plotly.express as px
import plotly.graph_objects as go
import plotly.figure_factory as ff
from plotly.subplots import make_subplots
```

In [4]:
```
tqdm.pandas()
np.random.seed(0)
%env PYTHONHASHSEED=0

import warnings
warnings.filterwarnings("ignore")
```

```
env: PYTHONHASHSEED=0
```

Data Loading

In [5]:
```
biorxiv_df = pd.read_csv('D:/GSUCoursework/BigDataExp/Datasets/cleaned csv/biorxiv_clean.csv')
pmc_df = pd.read_csv('D:/GSUCoursework/BigDataExp/Datasets/cleaned csv/clean_pmc.csv')
comm_use_df = pd.read_csv('D:/GSUCoursework/BigDataExp/Datasets/cleaned csv/clean_comm_use.csv')
noncomm_use_df = pd.read_csv('D:/GSUCoursework/BigDataExp/Datasets/cleaned csv/clean_noncomm_use.csv')
```

Data Stiching

In [6]:
```
papers_df = pd.concat([pmc_df,
                       biorxiv_df,
                       comm_use_df,
                       noncomm_use_df], axis=0).reset_index(drop=True)
```

In [7]:
```
papers_df
```

Out[7]:

	paper_id	title	authors	affiliations	abstract	t
0	14572a7a9b3e92b960d92d9755979eb94c448bb5	Immune Parameters of Dry Cows Fed Mannan Oligo...	S T Franklin, M C Newman, K E Newman, K I Meek	S T Franklin (University of Kentucky, 40546-02...	Abstract\n\nThe objective of this study was to...	INTRODUCTION\n\n peripartu rient period is
1	bb790e8366da63c4f5e2d64fa7bbd5673b93063c	Discontinuous Transcription or RNA Processing ...	Beate Schwer, Paolo Vista, Jan C Vos, Hendrik ...	Beate Schwer, Paolo Vista, Jan C Vos, Hendrik ...	NaN	Discontinuous\n\nTranscrip or RNA Proces
2	24f204ce5a1a4d752dc9ea7525082d225caed8b3	NaN	NaN	NaN	NaN	Letter to the Editor\n\n non-contact handh
3	ab78a42c688ac199a2d5669e42ee4c39ff0df2b8	A real-time convective PCR machine in a capill...	Yi-Fan Hsieh, Da-Sheng Lee, Ping-Hei Chen, Sha...	Yi-Fan Hsieh (National Taiwan University, 106,...	Abstract\n\nThis research reports the design, ...	Introduction\n\nMullis e developed the p
4	31105078a2953217223699d09c6a80d0f5edfdf6	Infecciones virales graves en pacientes inmuno...	A Díaz, R Zaragoza, R Granada, M Salavert	A Díaz (Hospital Universitario Virgen del Rocí...	Abstract\n\nRecibido el 21 de diciembre de 201...	\n\nFurthermore, ventila associated pneumc
...	...	...	...	...	...	
29310	3b8b2a835cfa770c6cef711d52e1f1e869d8aca8	Tumor-Treating Fields Induce RAW264.7 Macropha...	Jeong-In Park, Kyung-Hee Song, Seung-Youn Jung...	Jeong-In Park, Kyung-Hee Song, Seung-Youn Jung...	Abstract\n\nObjective: Tumor-treating fields a...	Introduction\n\nTumor-trea fields (TTFs

	paper_id	title	authors	affiliations	abstract	t
29311	500e585afac01e4b3847aa138db86b04352484c5	Efficacy and safety of Chou-Ling-Dan granules ...	Jiayang He, Zhengtu Li, Wanyi Huang, Wenda Gua...	Jiayang He, Zhengtu Li, Wanyi Huang, Wenda Gua...	NaN	\n\nIntroduction Chou-L Dan (CLD) (Laggera
29312	87f178ed1bcc5a747200ccd4adf42d883afd9fb7	Complete Genome Sequences of the SARS-CoV: the...	Shengli Bi, E'de Qin, Zuyuan Xu, Wei Li,...	Shengli Bi, E'de Qin (Chinese Academy of...	Abstract\n\nBeijing has been one of the epicen...	\n\nAccumulated numbe probable cases and
29313	5d4f1f02d0e731966dddd635e8fa7bfdc169d3b9	Case Report A Rare Case of Autoimmune Polyglan...	Ryan Kenneth Smith, Beaumont Health, Peter M G...	Ryan Kenneth Smith, Beaumont Health, Peter M G...	Abstract\n\nAdrenal insufficiency is a rare, p...	Introduction\n\nAdre insufficiency is dec
29314	c64c4f6efb9878bcc31980393c199d997805132a	Factors influencing Dipylidium sp. infection i...	Marion L East, Christoph Kurze, Kerstin Wilhel...	Marion L East (Leibniz Institute for Zoo and W...	Abstract\n\nWe provide the first genetic seque...	Introduction\n\nThe adult f of Dipylidium

29315 rows × 9 columns

In [8]:
```
papers_df['authors']
```

Out[8]:
```
0           S T Franklin, M C Newman, K E Newman, K I Meek
1        Beate Schwer, Paolo Vista, Jan C Vos, Hendrik ...
2                                                      NaN
3        Yi-Fan Hsieh, Da-Sheng Lee, Ping-Hei Chen, Sha...
4                A Díaz, R Zaragoza, R Granada, M Salavert
                               ...
29310    Jeong-In Park, Kyung-Hee Song, Seung-Youn Jung...
29311    Jiayang He, Zhengtu Li, Wanyi Huang, Wenda Gua...
29312    Shengli Bi,  E&amp;apos;de Qin, Zuyuan Xu, Wei Li,...
29313    Ryan Kenneth Smith, Beaumont Health, Peter M G...
29314    Marion L East, Christoph Kurze, Kerstin Wilhel...
Name: authors, Length: 29315, dtype: object
```

In [9]:
```
full_table = pd.read_csv('D:/GSUCoursework/BigDataExp/Datasets/Covid_19_clean/Covid_19_latest/covid_19_clean_complete.csv')
```

In [10]:
```
full_table.head(10)
```

Out[10]:

	Province/State	Country/Region	Lat	Long	Date	Confirmed	Deaths	Recovered
0	NaN	Afghanistan	33.0000	65.0000	1/22/20	0	0	0
1	NaN	Albania	41.1533	20.1683	1/22/20	0	0	0
2	NaN	Algeria	28.0339	1.6596	1/22/20	0	0	0
3	NaN	Andorra	42.5063	1.5218	1/22/20	0	0	0
4	NaN	Angola	-11.2027	17.8739	1/22/20	0	0	0
5	NaN	Antigua and Barbuda	17.0608	-61.7964	1/22/20	0	0	0
6	NaN	Argentina	-38.4161	-63.6167	1/22/20	0	0	0
7	NaN	Armenia	40.0691	45.0382	1/22/20	0	0	0
8	Australian Capital Territory	Australia	-35.4735	149.0124	1/22/20	0	0	0
9	New South Wales	Australia	-33.8688	151.2093	1/22/20	0	0	0

Data Cleaning

In [11]:
```
# Converting Date column to datetime datatype

full_table.dtypes
```

Out[11]:
```
Province/State     object
Country/Region     object
Lat               float64
Long              float64
Date               object
Confirmed           int64
Deaths              int64
Recovered           int64
dtype: object
```

In [12]:
```
full_table['Date'] = pd.to_datetime(full_table['Date'])
```

In [13]:
```
full_table.dtypes
```

Out[13]:
```
Province/State            object
Country/Region            object
Lat                      float64
Long                     float64
Date              datetime64[ns]
Confirmed                  int64
Deaths                     int64
Recovered                  int64
dtype: object
```

In [14]:
```
# Checking for null values

full_table.isna().sum()
```

Out[14]:
```
Province/State    15834
Country/Region        0
Lat                   0
Long                  0
Date                  0
Confirmed             0
Deaths                0
Recovered             0
dtype: int64
```

In [15]:
```
# filling missing values

full_table['Province/State'] = full_table['Province/State'].fillna('')
```

In [16]:
```
full_table.isna().sum()
```

Out[16]:
```
Province/State    0
Country/Region    0
Lat               0
Long              0
Date              0
Confirmed         0
Deaths            0
Recovered         0
dtype: int64
```

In [17]:
```
# replacing Mainland china with just China

full_table['Country/Region'] = full_table['Country/Region'].replace('Mainland China', 'China')
```

In [18]:
```
# Creating a new column 'Active' which will represent all the present active cases

full_table['Active'] = full_table['Confirmed'] - full_table['Deaths'] - full_table['Recovered']
```

In [19]:
```
full_table.head(10)
```

Out[19]:

	Province/State	Country/Region	Lat	Long	Date	Confirmed	Deaths	Recovered	Active
0		Afghanistan	33.0000	65.0000	2020-01-22	0	0	0	0
1		Albania	41.1533	20.1683	2020-01-22	0	0	0	0
2		Algeria	28.0339	1.6596	2020-01-22	0	0	0	0
3		Andorra	42.5063	1.5218	2020-01-22	0	0	0	0
4		Angola	-11.2027	17.8739	2020-01-22	0	0	0	0
5		Antigua and Barbuda	17.0608	-61.7964	2020-01-22	0	0	0	0
6		Argentina	-38.4161	-63.6167	2020-01-22	0	0	0	0
7		Armenia	40.0691	45.0382	2020-01-22	0	0	0	0
8	Australian Capital Territory	Australia	-35.4735	149.0124	2020-01-22	0	0	0	0
9	New South Wales	Australia	-33.8688	151.2093	2020-01-22	0	0	0	0

Data Preprocessing

In [20]:
```
# Cases in the ships

ship = full_table[full_table['Province/State'].str.contains('Grand Princess')|full_table['Country/Region'].st
r.contains('Cruise Ship')]
```

In [21]:
```
# Let us seperate out only China data into variable China and all other countries_Region into variable row

# china and the row
china = full_table[full_table['Country/Region']=='China']
row = full_table[full_table['Country/Region']!='China']
```

In [22]:
```
# latest cases

full_latest = full_table[full_table['Date'] == max(full_table['Date'])].reset_index()
china_latest = full_latest[full_latest['Country/Region']=='China']
row_latest = full_latest[full_latest['Country/Region']!='China']
```

In [23]:
```
# latest condensed

full_latest_grouped = full_latest.groupby('Country/Region')['Confirmed', 'Deaths', 'Recovered', 'Active'].sum
().reset_index()
china_latest_grouped = china_latest.groupby('Province/State')['Confirmed', 'Deaths', 'Recovered', 'Active'].s
um().reset_index()
row_latest_grouped = row_latest.groupby('Country/Region')['Confirmed', 'Deaths', 'Recovered', 'Active'].sum()
.reset_index()
```

Exploratory Data Analysis

1) Symptoms

Source: [https://en.wikipedia.org/wiki/Coronavirus_disease_2019 (https://en.wikipedia.org/wiki/Coronavirus_disease_2019)](https://en.wikipedia.org/wiki/Coronavirus_disease_2019)

In [24]:
```
symptoms={'symptom':['Fever',
        'Dry cough',
        'Fatigue',
        'Sputum production',
        'Shortness of breath',
        'Muscle pain',
        'Sore throat',
        'Headache',
        'Chills',
        'Nausea or vomiting',
        'Nasal congestion',
        'Diarrhoea',
        'Haemoptysis',
        'Conjunctival congestion'],'percentage':[87.9,67.7,38.1,33.4,18.6,14.8,13.9,13.6,11.4,5.0,4.8,3.7,0.9
,0.8]}

symptoms=pd.DataFrame(data=symptoms,index=range(14))
symptoms
```

Out[24]:

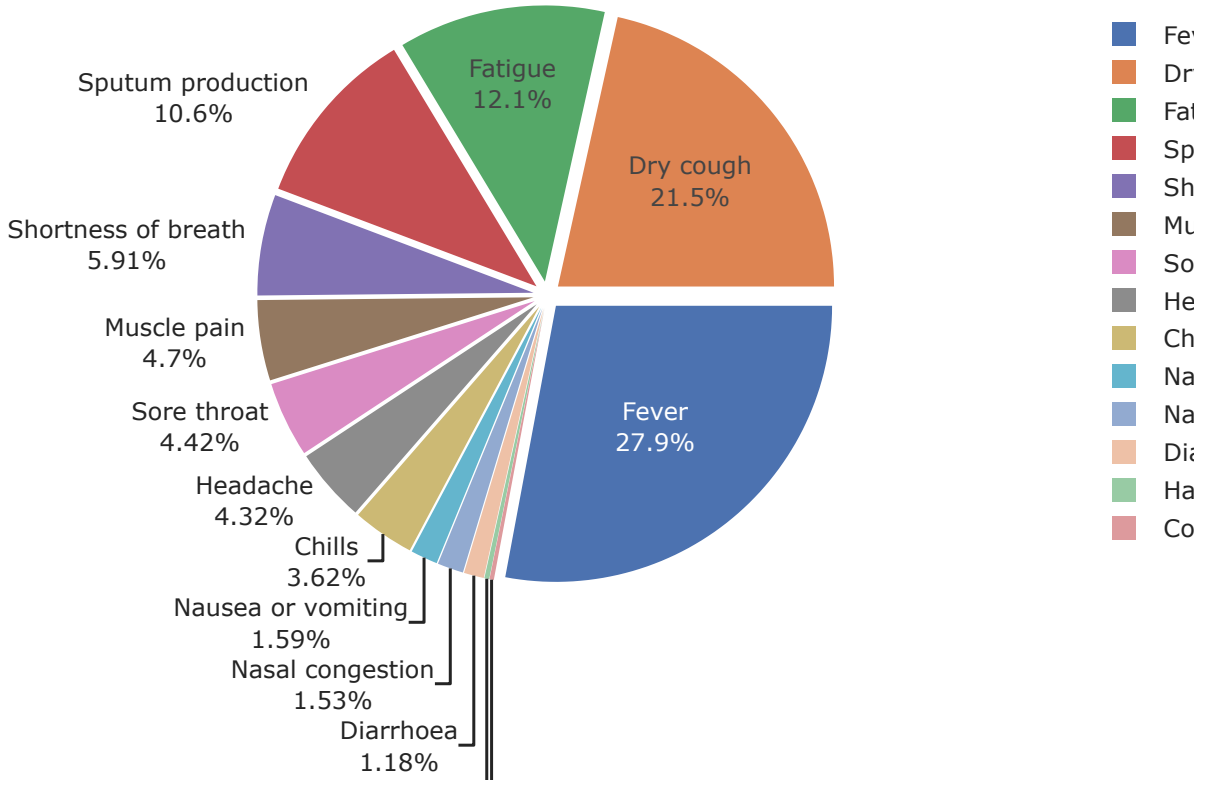
	symptom	percentage
0	Fever	87.9
1	Dry cough	67.7
2	Fatigue	38.1
3	Sputum production	33.4
4	Shortness of breath	18.6
5	Muscle pain	14.8
6	Sore throat	13.9
7	Headache	13.6
8	Chills	11.4
9	Nausea or vomiting	5.0
10	Nasal congestion	4.8
11	Diarrhoea	3.7
12	Haemoptysis	0.9
13	Conjunctival congestion	0.8

In [25]:
```
# Tree Plot

fig = px.treemap(symptoms, path=['symptom'], values='percentage',
                 color='percentage', hover_data=['symptom'],
                 color_continuous_scale='Rainbow')
fig.show()
```

In [26]:
```
# Pie Plot

fig = px.pie(symptoms,
             values="percentage",
             names="symptom",
             template="seaborn")
fig.update_traces(rotation=90, pull=0.05, textinfo="percent+label")
fig.show()
```



2)Current Situation

In [27]:
```
# Creating a consolidated table , which gives the country wise total defined cases

temp = full_table.groupby(['Country/Region', 'Province/State'])['Confirmed', 'Deaths', 'Recovered', 'Active']
.max()
temp = full_table.groupby('Date')['Confirmed', 'Deaths', 'Recovered', 'Active'].sum().reset_index()
temp = temp[temp['Date']==max(temp['Date'])].reset_index(drop=True)
temp.style.background_gradient(cmap='Pastel1')
```

Out[27]:

	Date	Confirmed	Deaths	Recovered	Active
0	2020-04-17 00:00:00	2240187	153821	557790	1528576

There are more recovered cases than deaths at this point of time

Countries which have been affected by the Coronavirus(2019-nCoV)till now

In [28]:
```
countries = full_table['Country/Region'].unique().tolist()
print(countries)

print("\nTotal countries affected by virus: ",len(countries))
```

```
['Afghanistan', 'Albania', 'Algeria', 'Andorra', 'Angola', 'Antigua and Barbuda', 'Argentina', 'Armenia', 'Au
stralia', 'Austria', 'Azerbaijan', 'Bahamas', 'Bahrain', 'Bangladesh', 'Barbados', 'Belarus', 'Belgium', 'Ben
in', 'Bhutan', 'Bolivia', 'Bosnia and Herzegovina', 'Brazil', 'Brunei', 'Bulgaria', 'Burkina Faso', 'Cabo Ver
de', 'Cambodia', 'Cameroon', 'Canada', 'Central African Republic', 'Chad', 'Chile', 'China', 'Colombia', 'Con
go (Brazzaville)', 'Congo (Kinshasa)', 'Costa Rica', "Cote d'Ivoire", 'Croatia', 'Diamond Princess', 'Cuba',
'Cyprus', 'Czechia', 'Denmark', 'Djibouti', 'Dominican Republic', 'Ecuador', 'Egypt', 'El Salvador', 'Equator
ial Guinea', 'Eritrea', 'Estonia', 'Eswatini', 'Ethiopia', 'Fiji', 'Finland', 'France', 'Gabon', 'Gambia', 'G
eorgia', 'Germany', 'Ghana', 'Greece', 'Guatemala', 'Guinea', 'Guyana', 'Haiti', 'Holy See', 'Honduras', 'Hun
gary', 'Iceland', 'India', 'Indonesia', 'Iran', 'Iraq', 'Ireland', 'Israel', 'Italy', 'Jamaica', 'Japan', 'Jo
rdan', 'Kazakhstan', 'Kenya', 'South Korea', 'Kuwait', 'Kyrgyzstan', 'Latvia', 'Lebanon', 'Liberia', 'Liechte
nstein', 'Lithuania', 'Luxembourg', 'Madagascar', 'Malaysia', 'Maldives', 'Malta', 'Mauritania', 'Mauritius',
'Mexico', 'Moldova', 'Monaco', 'Mongolia', 'Montenegro', 'Morocco', 'Namibia', 'Nepal', 'Netherlands', 'New Z
ealand', 'Nicaragua', 'Niger', 'Nigeria', 'North Macedonia', 'Norway', 'Oman', 'Pakistan', 'Panama', 'Papua N
ew Guinea', 'Paraguay', 'Peru', 'Philippines', 'Poland', 'Portugal', 'Qatar', 'Romania', 'Russia', 'Rwanda',
'Saint Lucia', 'Saint Vincent and the Grenadines', 'San Marino', 'Saudi Arabia', 'Senegal', 'Serbia', 'Seyche
lles', 'Singapore', 'Slovakia', 'Slovenia', 'Somalia', 'South Africa', 'Spain', 'Sri Lanka', 'Sudan', 'Surina
me', 'Sweden', 'Switzerland', 'Taiwan*', 'Tanzania', 'Thailand', 'Togo', 'Trinidad and Tobago', 'Tunisia', 'T
urkey', 'Uganda', 'Ukraine', 'United Arab Emirates', 'United Kingdom', 'Uruguay', 'US', 'Uzbekistan', 'Venezu
ela', 'Vietnam', 'Zambia', 'Zimbabwe', 'Dominica', 'Grenada', 'Mozambique', 'Syria', 'Timor-Leste', 'Belize',
'Laos', 'Libya', 'West Bank and Gaza', 'Guinea-Bissau', 'Mali', 'Saint Kitts and Nevis', 'Kosovo', 'Burma',
'MS Zaandam', 'Botswana', 'Burundi', 'Sierra Leone', 'Malawi', 'South Sudan', 'Western Sahara', 'Sao Tome and
Principe', 'Yemen']

Total countries affected by virus:  185
```

For visualizations, we are using Plotly Python Open Source Graphing Library. Plotly is another great Python visualization tool that's capable of handling geographical, scientific, statistical, and financial data. Plotly has several advantages over matplotlib. One of the main advantages is that only a few lines of codes are necessary to create aesthetically pleasing, interactive plots. The interactivity also offers a number of advantages over static matplotlib plots:¶

1) Saves time when initially exploring your dataset

2) Makes it easy to modify and export your plot

3) Offers a more ornate visualization, which is well-suited for conveying the important insights hidden within your dataset

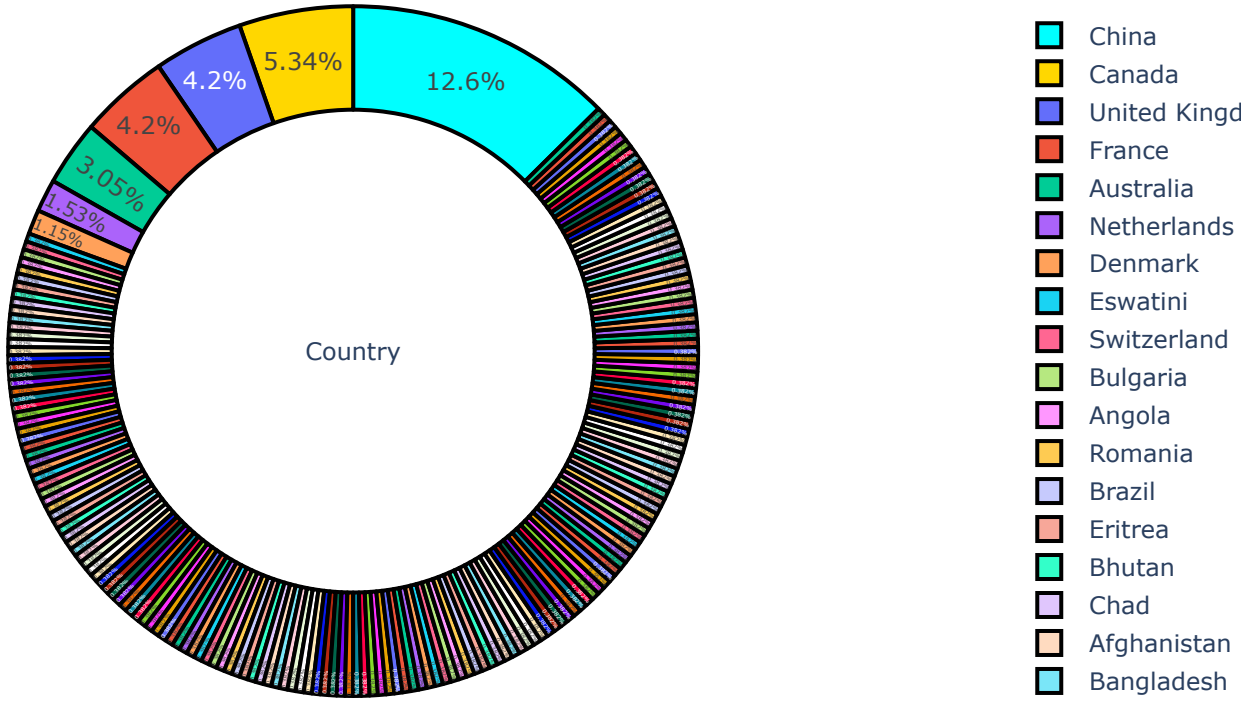
Pie Plot

In [29]:
```
'''A Function To Plot Pie Plot using Plotly'''

def pie_plot(cnt_srs, colors, title):
    labels=cnt_srs.index
    values=cnt_srs.values
    trace = go.Pie(labels=labels,
                   values=values,
                   title=title,
                   hoverinfo='percent+value',
                   textinfo='percent',
                   textposition='inside',
                   hole=0.7,
                   showlegend=True,
                   marker=dict(colors=colors,
                               line=dict(color='#000000',
                                         width=2),
                              )
                  )
    return trace
```

In [30]:
```
'''Plotly visualization'''
import plotly.offline as py
from plotly.offline import iplot, init_notebook_mode
import plotly.graph_objs as go
py.init_notebook_mode(connected = True) # Required to use plotly offline in jupyter notebook
```

In [31]:
```
py.iplot([pie_plot(full_table['Country/Region'].value_counts(), ['cyan', 'gold'], 'Country')])
```

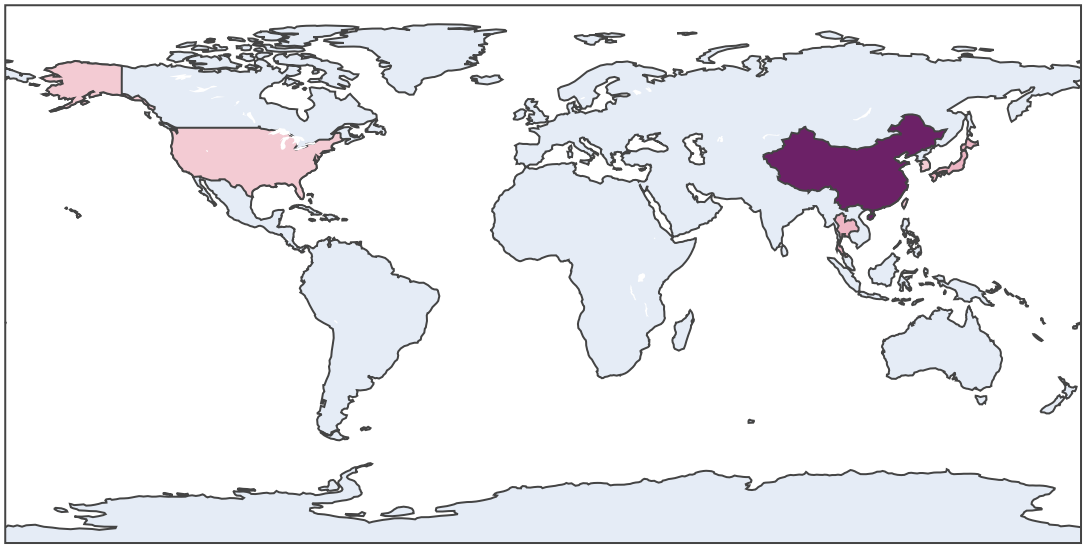


In [32]:
```
full_grouped = full_table.groupby(['Date', 'Country/Region'])['Confirmed', 'Deaths', 'Recovered', 'Active'].sum().reset_index()
```

In [33]:
```
# Over the time

fig = px.choropleth(full_grouped, locations="Country/Region", locationmode='country names', color=np.log(full
_grouped["Confirmed"]),
                    hover_name="Country/Region", animation_frame=full_grouped["Date"].dt.strftime('%Y-%m-%d'
),
                    title='Cases over time', color_continuous_scale=px.colors.sequential.Magenta)
fig.update(layout_coloraxis_showscale=False)
fig.show()
```

Cases over time



animation_frame=2020-01-22

Top 20 countries having most confirmed cases

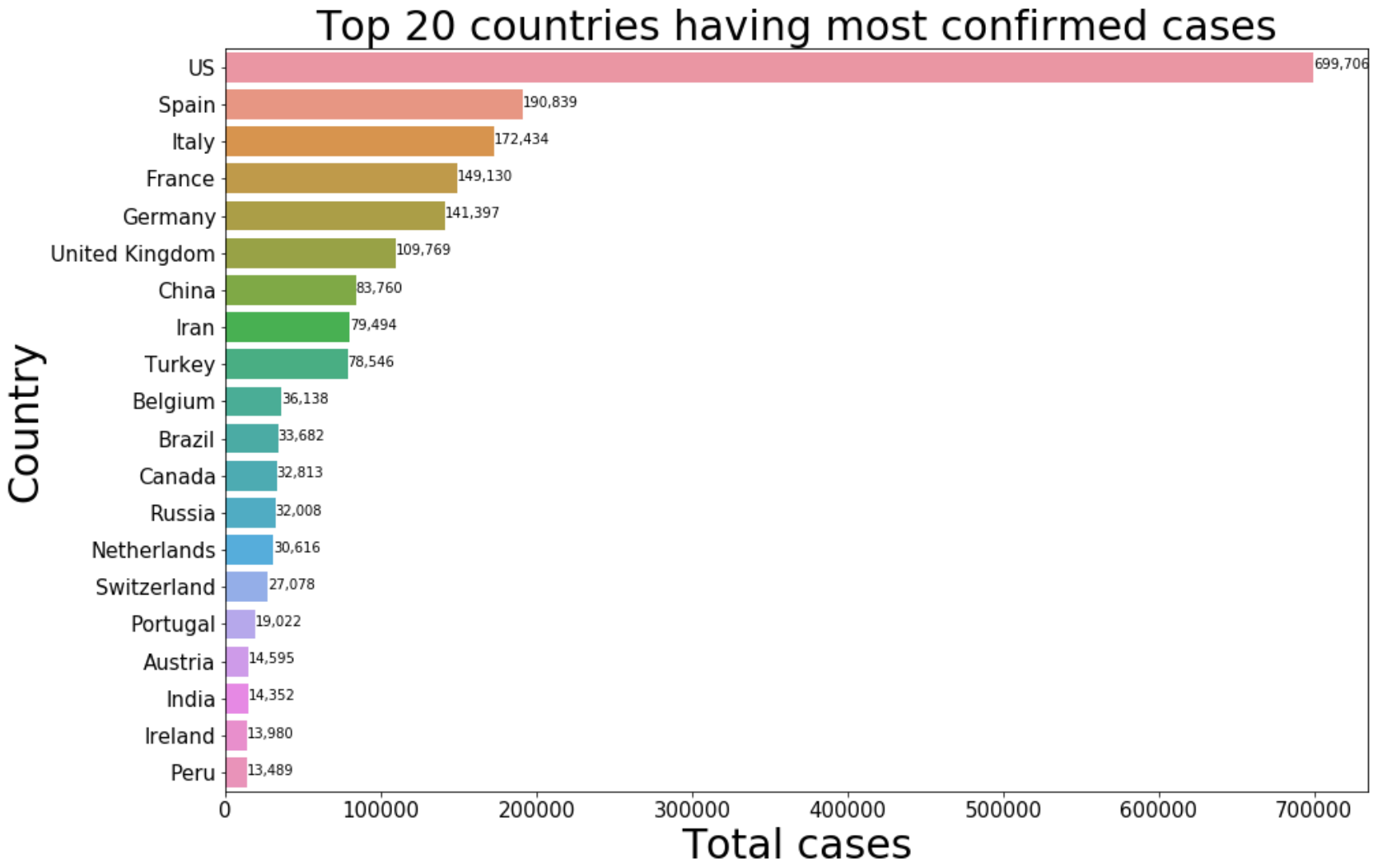
In [34]:
```
top = full_table[full_table['Date'] == full_table['Date'].max()]
top_casualities = top.groupby(by = 'Country/Region')['Confirmed'].sum().sort_values(ascending = False).head(20).reset_index()
top_casualities
```

Out[34]:

	Country/Region	Confirmed
0	US	699706
1	Spain	190839
2	Italy	172434
3	France	149130
4	Germany	141397
5	United Kingdom	109769
6	China	83760
7	Iran	79494
8	Turkey	78546
9	Belgium	36138
10	Brazil	33682
11	Canada	32813
12	Russia	32008
13	Netherlands	30616
14	Switzerland	27078
15	Portugal	19022
16	Austria	14595
17	India	14352
18	Ireland	13980
19	Peru	13489

In [35]:
```
plt.figure(figsize= (15,10))
plt.xticks(fontsize = 15)
plt.yticks(fontsize = 15)
plt.xlabel("Total cases",fontsize = 30)
plt.ylabel('Country',fontsize = 30)
plt.title("Top 20 countries having most confirmed cases" , fontsize = 30)
ax = sns.barplot(x = top_casualities['Confirmed'], y = top_casualities['Country/Region'])
for i, (value, name) in enumerate(zip(top_casualities['Confirmed'],top_casualities['Country/Region'])):
    ax.text(value, i-.05, f'{value:,.0f}',  size=10, ha='left',  va='center')
ax.set(xlabel='Total cases', ylabel='Country')
```

Out[35]: [Text(0, 0.5, 'Country'), Text(0.5, 0, 'Total cases')]



Observations :

1) China was leading this from many days, but now they are controlling the pandemic spread.

2) The number of confirmed cases are on a high in the US, Italy, Spain, and France.

3) But the number of cases in the third world countries is less.

Top 20 countries having most active cases

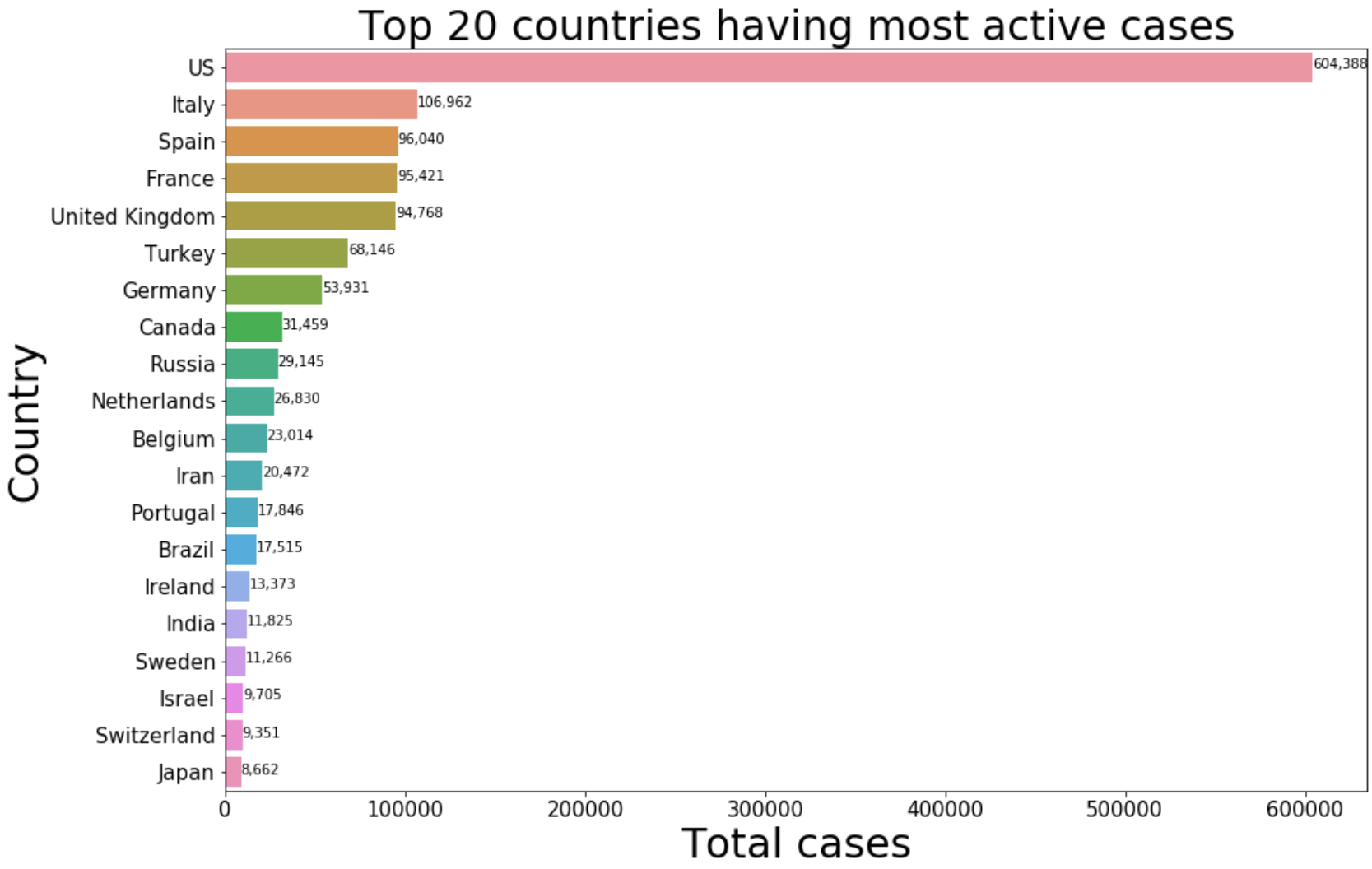
In [36]:
```
top_actives = top.groupby(by = 'Country/Region')['Active'].sum().sort_values(ascending = False).head(20).reset_index()
top_actives
```

Out[36]:

	Country/Region	Active
0	US	604388
1	Italy	106962
2	Spain	96040
3	France	95421
4	United Kingdom	94768
5	Turkey	68146
6	Germany	53931
7	Canada	31459
8	Russia	29145
9	Netherlands	26830
10	Belgium	23014
11	Iran	20472
12	Portugal	17846
13	Brazil	17515
14	Ireland	13373
15	India	11825
16	Sweden	11266
17	Israel	9705
18	Switzerland	9351
19	Japan	8662

In [37]:
```
plt.figure(figsize= (15,10))
plt.xticks(fontsize = 15)
plt.yticks(fontsize = 15)
plt.xlabel("Total cases",fontsize = 30)
plt.ylabel('Country',fontsize = 30)
plt.title("Top 20 countries having most active cases" , fontsize = 30)
ax = sns.barplot(x = top_actives['Active'], y = top_actives['Country/Region'])
for i, (value, name) in enumerate(zip(top_actives['Active'], top_actives['Country/Region'])):
    ax.text(value, i-.05, f'{value:,.0f}',  size=10, ha='left',  va='center')
ax.set(xlabel='Total cases', ylabel='Country')
```

Out[37]: [Text(0, 0.5, 'Country'), Text(0.5, 0, 'Total cases')]



Observations :

1) As the covid-19 testing is increasing, The active number of cases is also increasing day by day.

2) The number of active cases is on a high in the US, Italy, Spain, and France.

Top 20 countries having most deaths

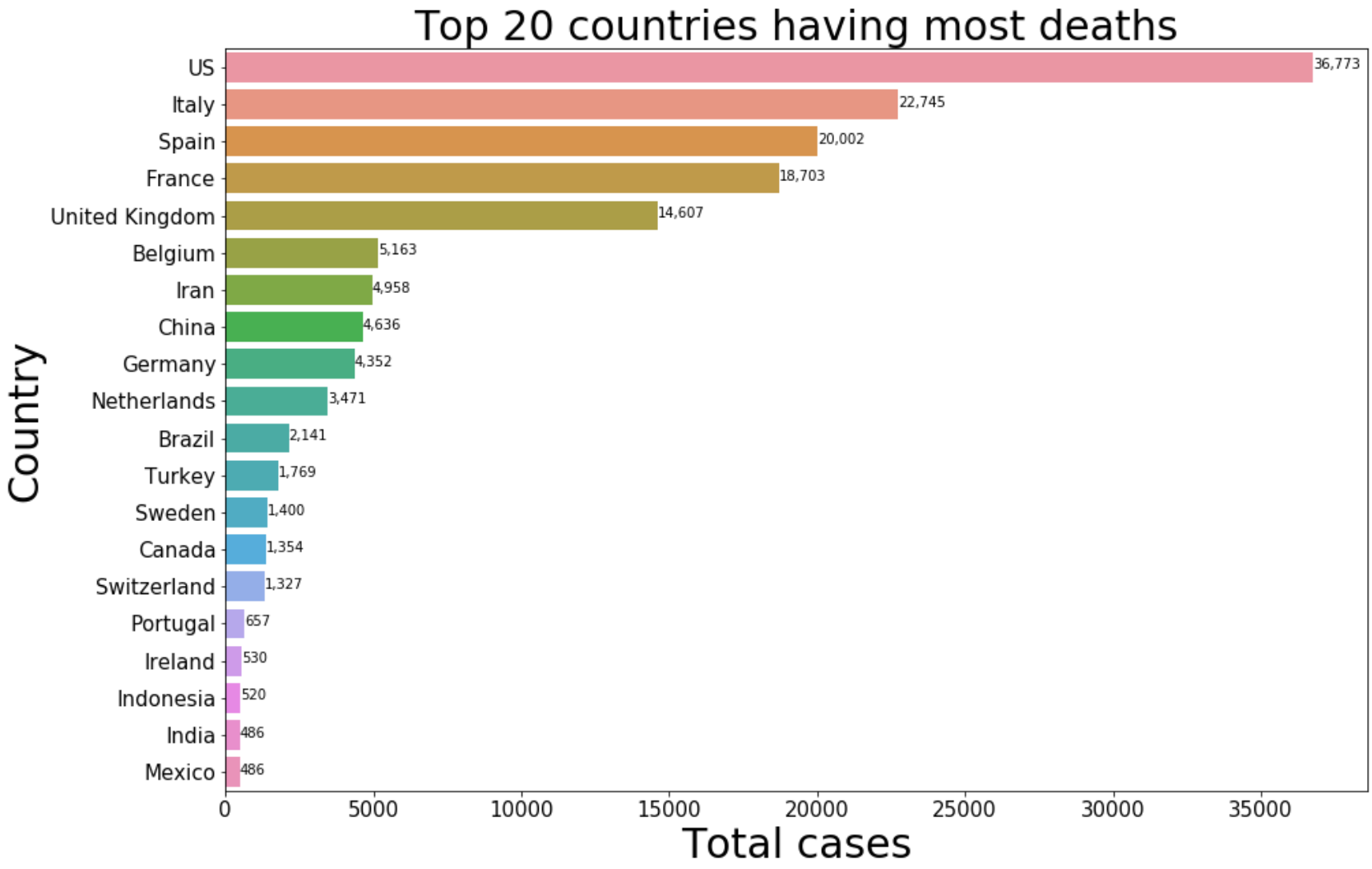
In [38]:
```
top_deaths = top.groupby(by = 'Country/Region')['Deaths'].sum().sort_values(ascending = False).head(20).reset
_index()
top_deaths
```

Out[38]:

	Country/Region	Deaths
0	US	36773
1	Italy	22745
2	Spain	20002
3	France	18703
4	United Kingdom	14607
5	Belgium	5163
6	Iran	4958
7	China	4636
8	Germany	4352
9	Netherlands	3471
10	Brazil	2141
11	Turkey	1769
12	Sweden	1400
13	Canada	1354
14	Switzerland	1327
15	Portugal	657
16	Ireland	530
17	Indonesia	520
18	India	486
19	Mexico	486

In [39]:
```
plt.figure(figsize= (15,10))
plt.xticks(fontsize = 15)
plt.yticks(fontsize = 15)
plt.xlabel("Total cases",fontsize = 30)
plt.ylabel('Country',fontsize = 30)
plt.title("Top 20 countries having most deaths" , fontsize = 30)
ax = sns.barplot(x = top_deaths['Deaths'], y = top_deaths['Country/Region'])
for i, (value, name) in enumerate(zip(top_deaths['Deaths'],top_deaths['Country/Region'])):
    ax.text(value, i-.05, f'{value:,.0f}',  size=10, ha='left',  va='center')
ax.set(xlabel='Total cases', ylabel='Country')
```

Out[39]: [Text(0, 0.5, 'Country'), Text(0.5, 0, 'Total cases')]



Observations :

1) Even though Italy has the 2nd best healthcare system according to the WHO, they haven't been able to tackle the pandemic problem effectively.

2) China even having so many confirmed cases was able to decrease the number of deaths

3) The number of deaths is also on a rise, especially in Italy, Spain, and France

Top 20 countries having most recovered cases

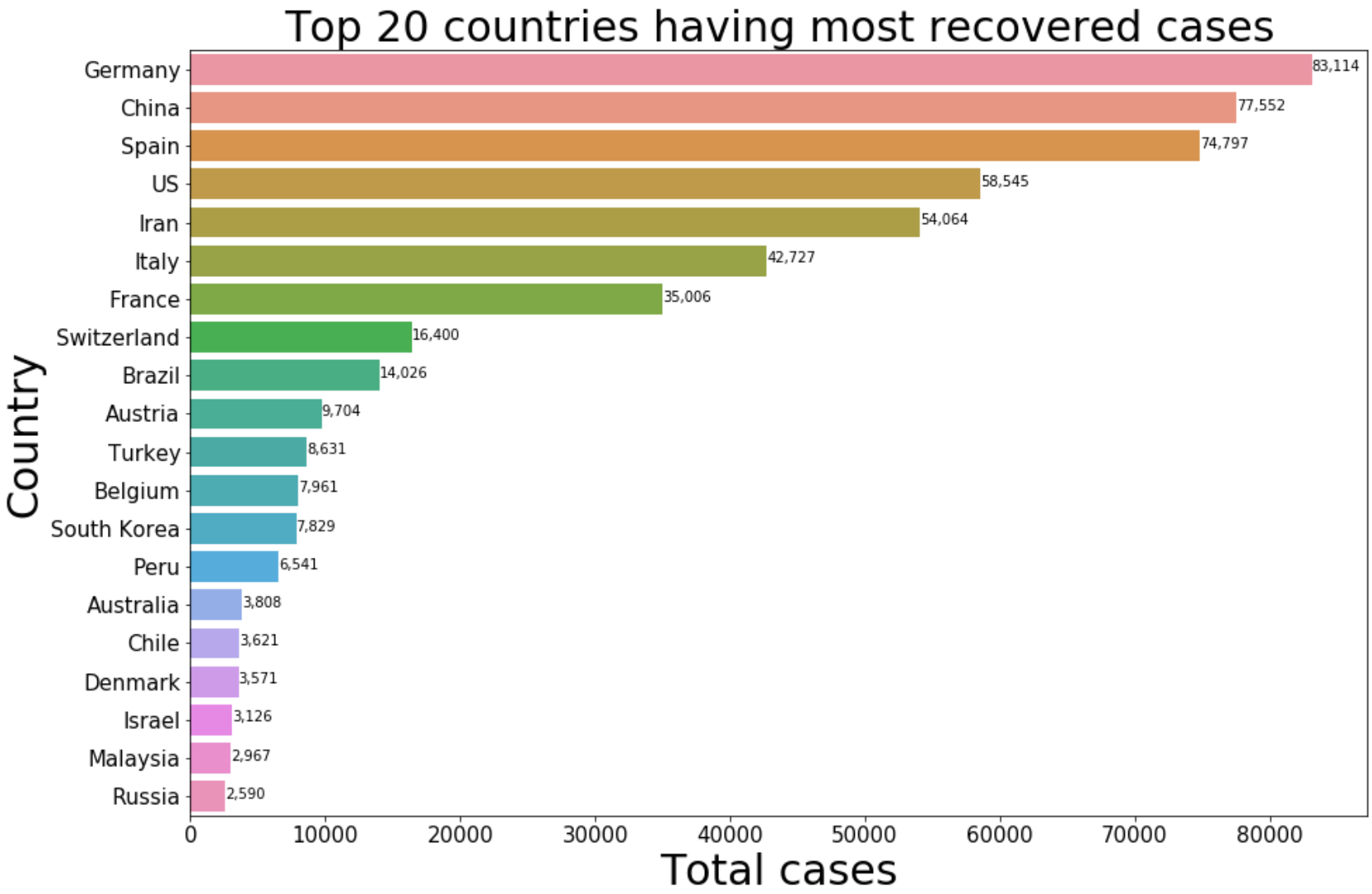
In [40]:
```
top_recovered = top.groupby(by = 'Country/Region')['Recovered'].sum().sort_values(ascending = False).head(20)
.reset_index()
top_recovered
```

Out[40]:

	Country/Region	Recovered
0	Germany	83114
1	China	77552
2	Spain	74797
3	US	58545
4	Iran	54064
5	Italy	42727
6	France	35006
7	Switzerland	16400
8	Brazil	14026
9	Austria	9704
10	Turkey	8631
11	Belgium	7961
12	South Korea	7829
13	Peru	6541
14	Australia	3808
15	Chile	3621
16	Denmark	3571
17	Israel	3126
18	Malaysia	2967
19	Russia	2590

In [41]:
```
plt.figure(figsize= (15,10))
plt.xticks(fontsize = 15)
plt.yticks(fontsize = 15)
plt.xlabel("Total cases",fontsize = 30)
plt.ylabel('Country',fontsize = 30)
plt.title("Top 20 countries having most recovered cases" , fontsize = 30)
ax = sns.barplot(x = top_recovered['Recovered'], y = top_recovered['Country/Region'])
for i, (value, name) in enumerate(zip(top_recovered['Recovered'],top_recovered['Country/Region'])):
    ax.text(value, i-.05, f'{value:,.0f}',  size=10, ha='left',  va='center')
ax.set(xlabel='Total cases', ylabel='Country')
```

Out[41]: [Text(0, 0.5, 'Country'), Text(0.5, 0, 'Total cases')]



1) By far China was leading in the number of recoveries even though having a huge number of confirmed cases, but recently Germany has surpassed China in terms of most recovered cases. No wonder Germany has best healthcare facilities

2) Spain, US, Italy and Iran are also doing a good job.

3) We have to pump up these numbers for a promising future!

3)Finding ways to contain COVID-19

Now, we will look at the evolution of the virus in different countries and look at what strategies could be used to contain COVID-19.

The current situation (as of April 17th, 2020)

First, we will look at the current situation in five countries: Italy, China, US, Iran, and South Korea. (as of April 17th, 2020)

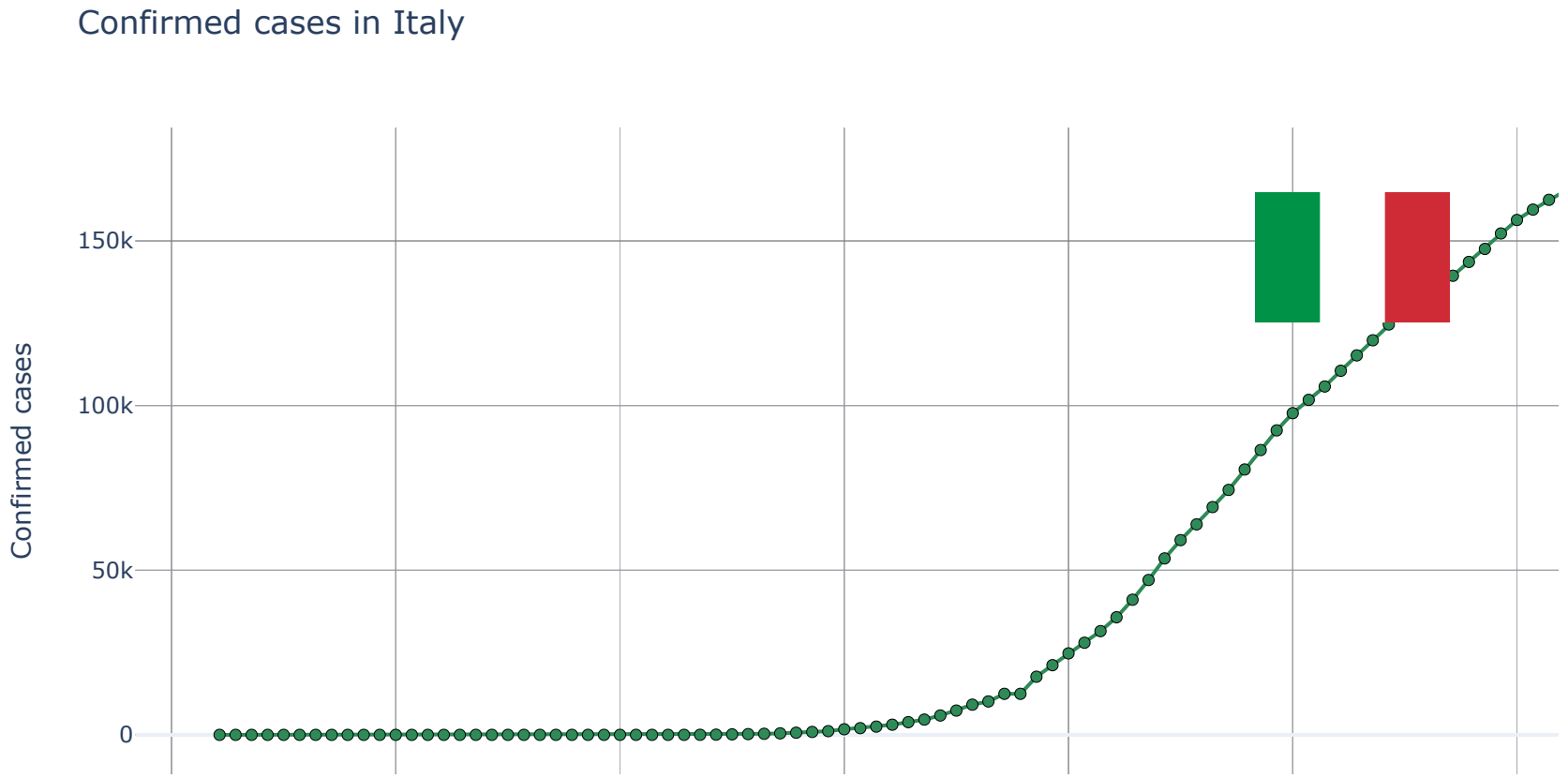
In [42]:
```
tbl = full_table.sort_values(by=["Country/Region", "Date"]).reset_index(drop=True)
tbl["Country"] = tbl["Country/Region"]
conts = sorted(list(set(tbl["Country"])))
dates = sorted(list(set(tbl["Date"])))

confirmed = []
for idx in range(len(conts)):
    confirmed.append(tbl.query('Country == "{}"'.format(conts[idx])).groupby("Date").sum()["Confirmed"].value
s)
confirmed = np.array(confirmed)
```

In [43]:
```
def visualize_country(fig, cont, image_link, colors, step, xcor, ycor, done=True, multiple=False, sizex=0.78,
sizey=0.2):
    if not done:
        showlegend = True
    else:
        showlegend = False
    for idx, color in enumerate(colors):
        fig.add_trace(go.Scatter(x=dates, y=confirmed[conts.index(cont)]-step*idx, showlegend=showlegend,
                    mode='lines+markers', name=cont,
                         marker=dict(color=colors[idx], line=dict(color='rgb(0, 0, 0)', width=0.5))))
    fig.add_layout_image(
        dict(
            source=image_link,
            xref="paper", yref="paper",
            x=xcor, y=ycor,
            sizex=sizex, sizey=sizey,
            xanchor="right", yanchor="bottom")
        )
    title = "Confirmed cases in {}".format(cont) if done else "Confirmed cases"
    if multiple: title = "Confirmed cases"
    fig.update_layout(xaxis_title="Date", yaxis_title="Confirmed cases", title=title, template="plotly_white"
, paper_bgcolor="#f0f0f0")
    if done:
        fig.show()
```

Italy

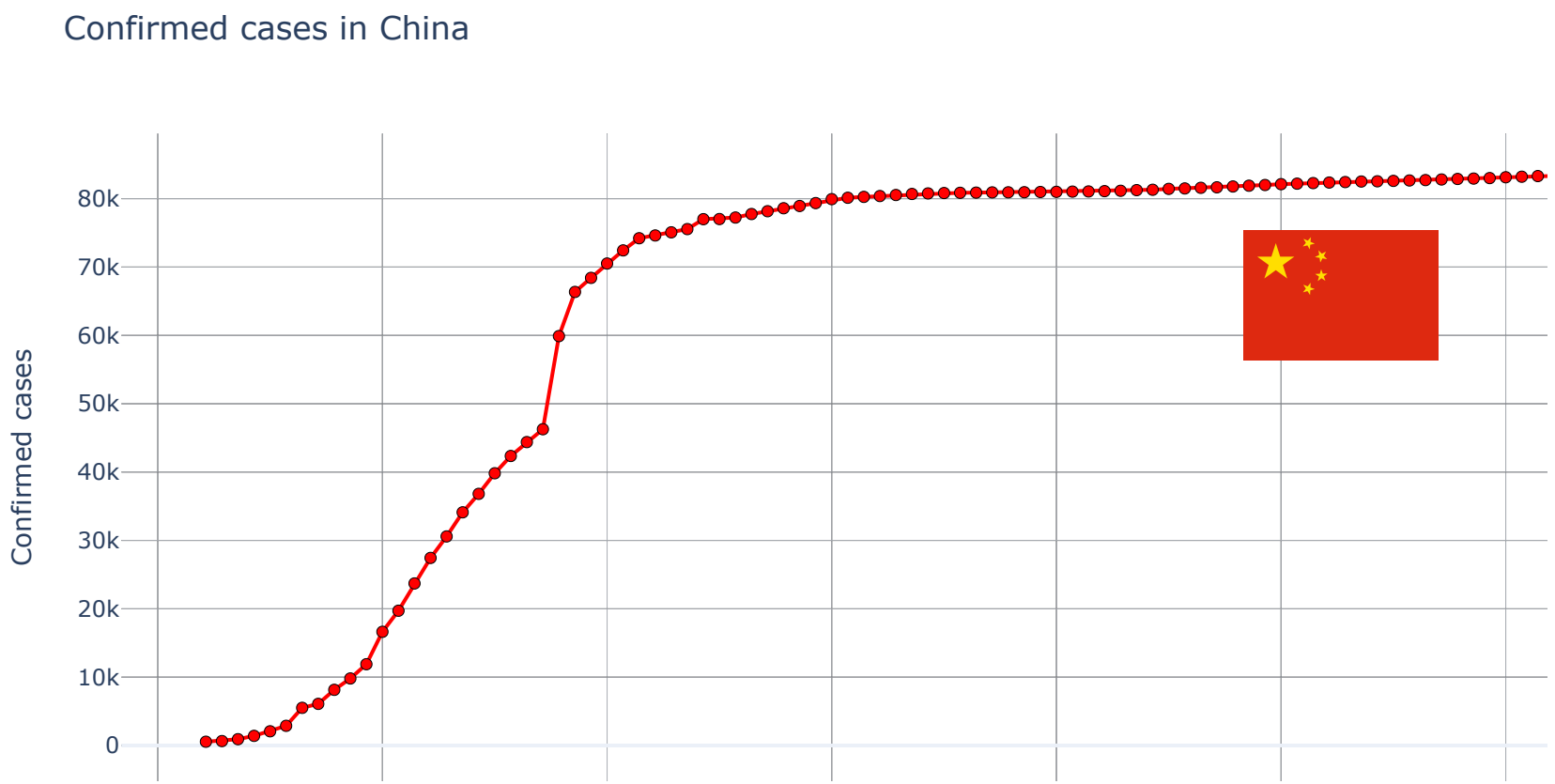
In [44]:
```
fig = go.Figure()
visualize_country(fig, "Italy", "https://upload.wikimedia.org/wikipedia/en/0/03/Flag_of_Italy.svg", colors=[
"seagreen"], step=400, xcor=0.85, ycor=0.7)
```



The current epidemic is in a very bad state right now. The number of cases are growing everyday. The entire nation is under lockdown due to the massive number of new cases being reported everyday. The mortality rate is also very high in Italy due to the large elderly population. There are currently close to 172K confirmed cases in Italy.

China

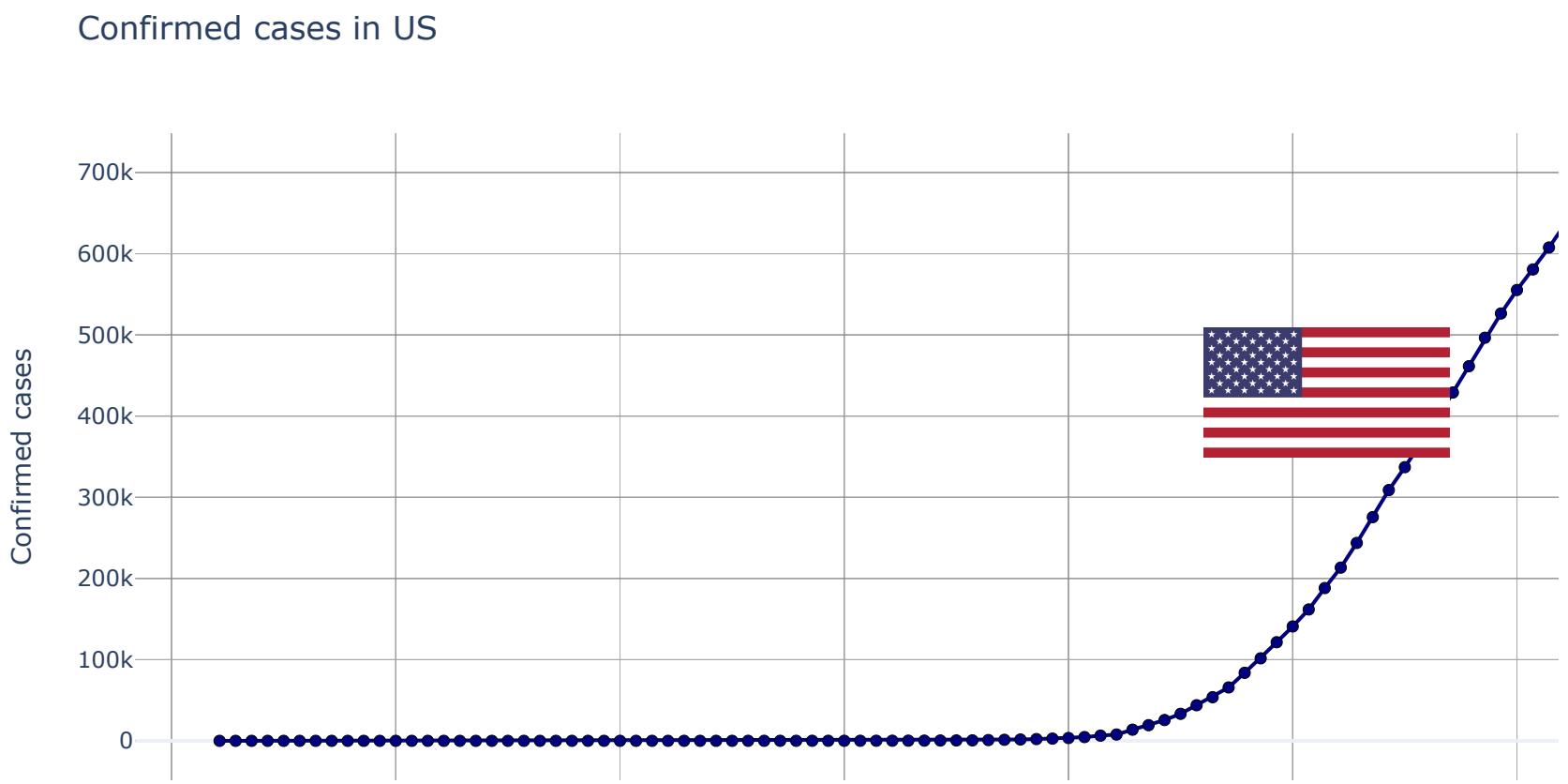
In [45]:
```
fig = go.Figure()
visualize_country(fig, "China", "https://upload.wikimedia.org/wikipedia/commons/f/fa/Flag_of_the_People%27s_Republic_of_China.svg", colors=["red"], step=1000, xcor=0.85, ycor=0.65)
```



The initial epidemic in China was spreading very fast, with new cases in the thousands and new deaths in the hundreds everyday. But through a series of measures, including community and industry lockdown throughout China, they have been able to reduce community transmission and "flatten the curve". On March 18th 2020, China reported 0 new cases. They successfully implemented measures at the right time to mitigate the spread of the virus.

USA

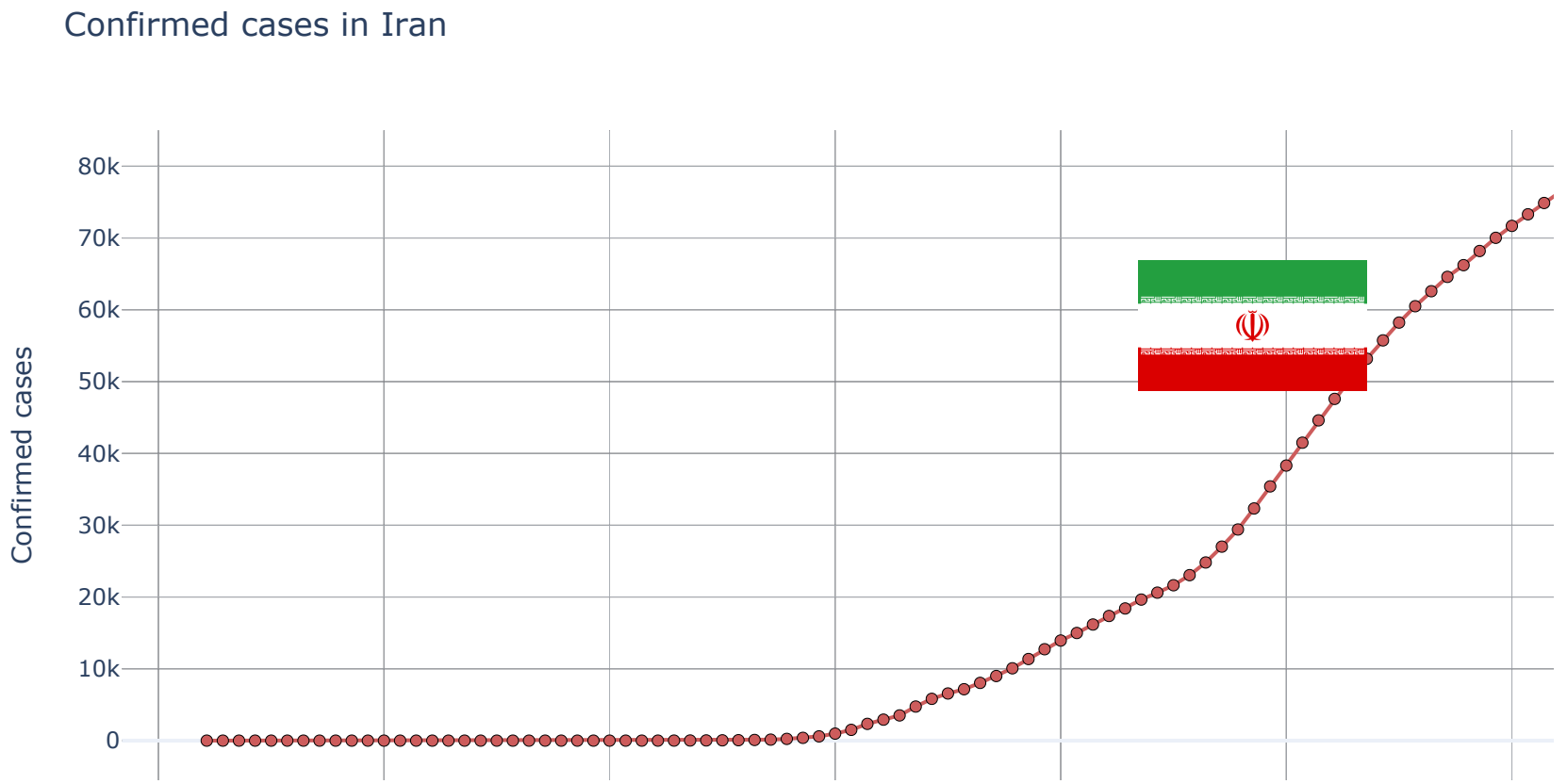
In [46]:
```
fig = go.Figure()
visualize_country(fig, "US", "https://upload.wikimedia.org/wikipedia/en/a/a4/Flag_of_the_United_States.svg",
colors=["navy"], step=60, xcor=0.85, ycor=0.5)
```



The situation in the US is also difficult at the time of writing. A delay in mass-scale testing, travel lockdown, and social distancing has resulted in a lot of community transmission. As of April 17th, there are currently close to 700,000 confirmed cases in the US, but the actual number may be more.

Iran

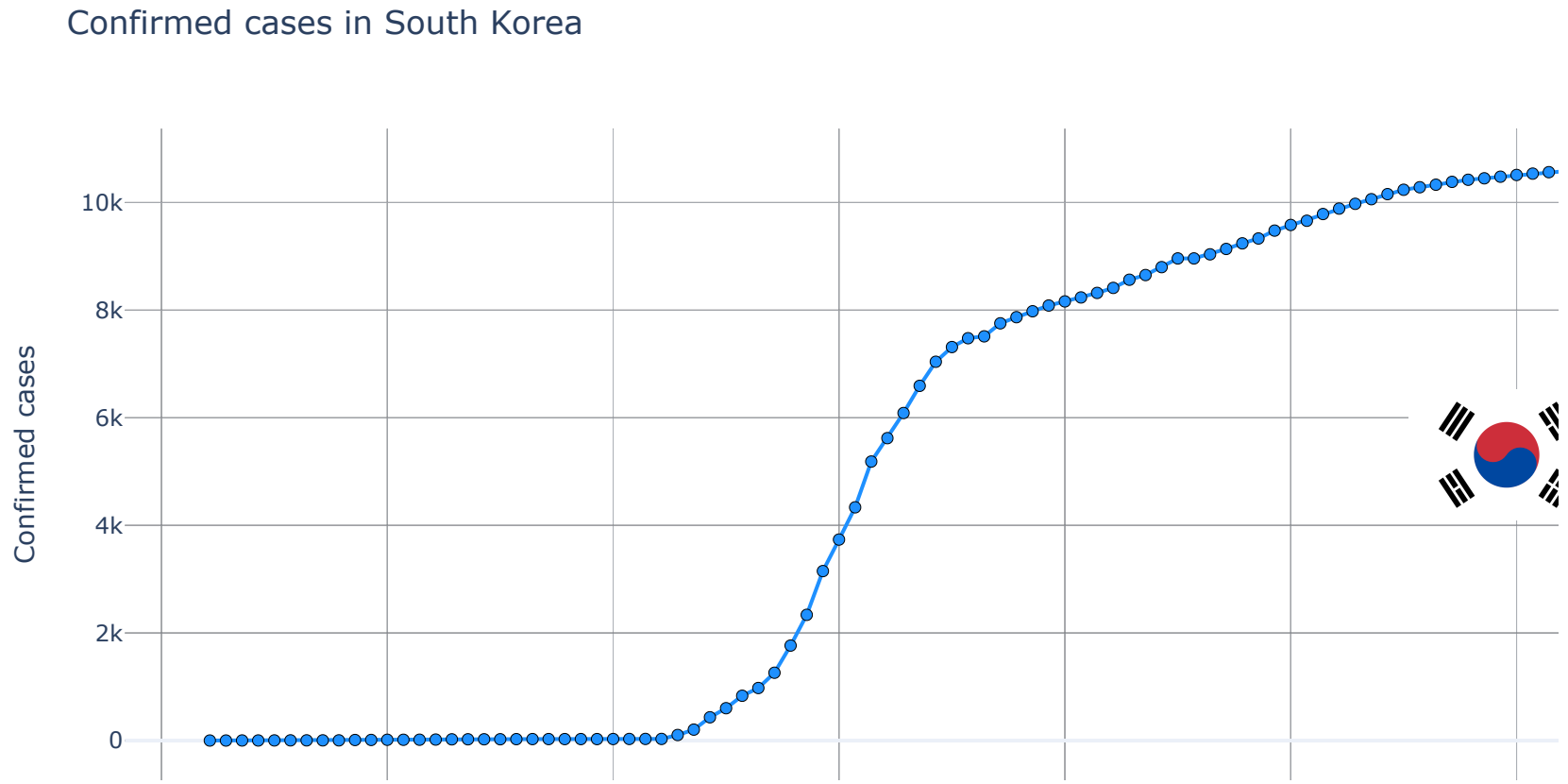
In [47]:
```
fig = go.Figure()
visualize_country(fig, "Iran", "https://upload.wikimedia.org/wikipedia/commons/c/ca/Flag_of_Iran.svg", colors
=["indianred"], step=175, xcor=0.8, ycor=0.6)
```



Iran is also going through a terrible epidemic at the moment, and a shortage in healthcare and testing equipment is making matters worse. There are currently close to 80,000 confirmed cases in Iran.

South Korea

In [48]:
```
fig = go.Figure()
visualize_country(fig, "South Korea", "https://upload.wikimedia.org/wikipedia/commons/0/09/Flag_of_South_Korea.svg", colors=["dodgerblue"], step=80, xcor=0.95, ycor=0.4)
```



South Korea had a large initial burst in cases, but over time, they have been able to successfully mitigate the spread of the virus and reduce community transmission through a series of smart policies. Since South Korea did not have the capacity to lockdown the entire country (like China), they relied on

1)mass testing

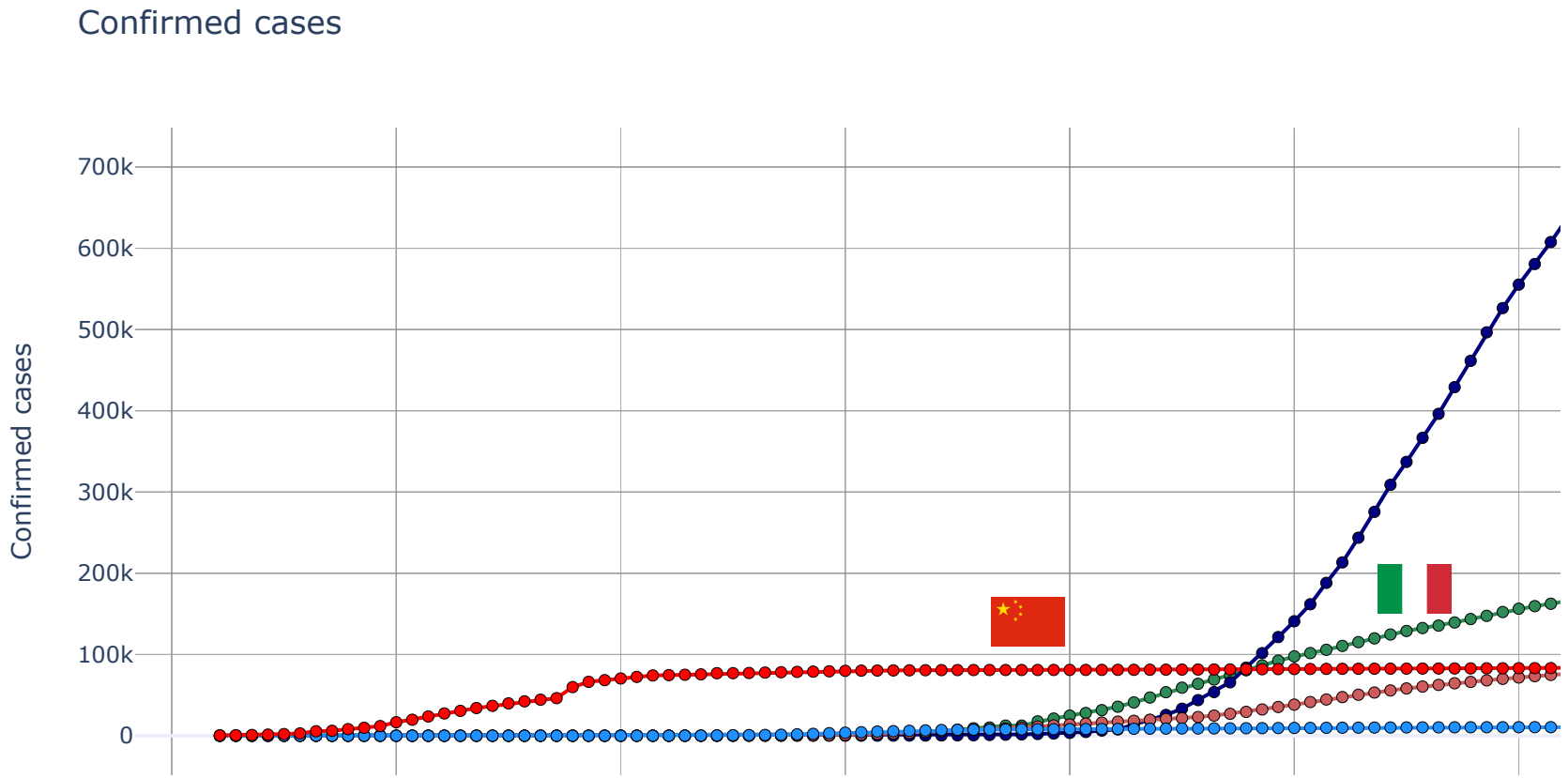
2)GPS-based quarantine tracking to mitigate the virus.

Social distancing combined with thousands of tests everyday has reduced the number of new cases dramatically over the last few days.

The app, developed by the Ministry of the Interior and Safety, allows those who have been ordered not to leave home to stay in contact with case workers and report on their progress. It will also use GPS to keep track of their location to make sure they are not breaking their quarantine

Comparing all 5 countries together

In [49]:
```
fig = go.Figure()
visualize_country(fig, "Italy", "https://upload.wikimedia.org/wikipedia/en/0/03/Flag_of_Italy.svg", colors=[
"seagreen"], step=400, xcor=0.85, ycor=0.25, sizex=0.15, sizey=0.075, done=False)
visualize_country(fig, "US", "https://upload.wikimedia.org/wikipedia/en/a/a4/Flag_of_the_United_States.svg",
colors=["navy"], step=60, xcor=0.980, ycor=0.8, sizex=0.1, sizey=0.065, done=False)
visualize_country(fig, "Iran", "https://upload.wikimedia.org/wikipedia/commons/c/ca/Flag_of_Iran.svg", colors
=["indianred"], step=175, xcor=0.999, ycor=0.15, sizex=0.1, sizey=0.065, done=False)
visualize_country(fig, "South Korea", "https://upload.wikimedia.org/wikipedia/commons/0/09/Flag_of_South_Kore
a.svg", colors=["dodgerblue"], step=80, xcor=0.99, ycor=0.05, sizex=0.15, sizey=0.075, done=False)
fig.update_layout(showlegend=False)
visualize_country(fig, "China", "https://upload.wikimedia.org/wikipedia/commons/f/fa/Flag_of_the_People%27s_R
epublic_of_China.svg", colors=["red"], step=1000, xcor=0.6, ycor=0.2, sizex=0.15, sizey=0.075, multiple=True)
```



When we see the number of new cases in all 5 countries together, we can see the which countries have been able to contain the virus so far (South Korea and China), and which ones have not (Iran, Italy, and US).

What can we learn from China and South Korea?

What strategies China adopted to contain Covid-19

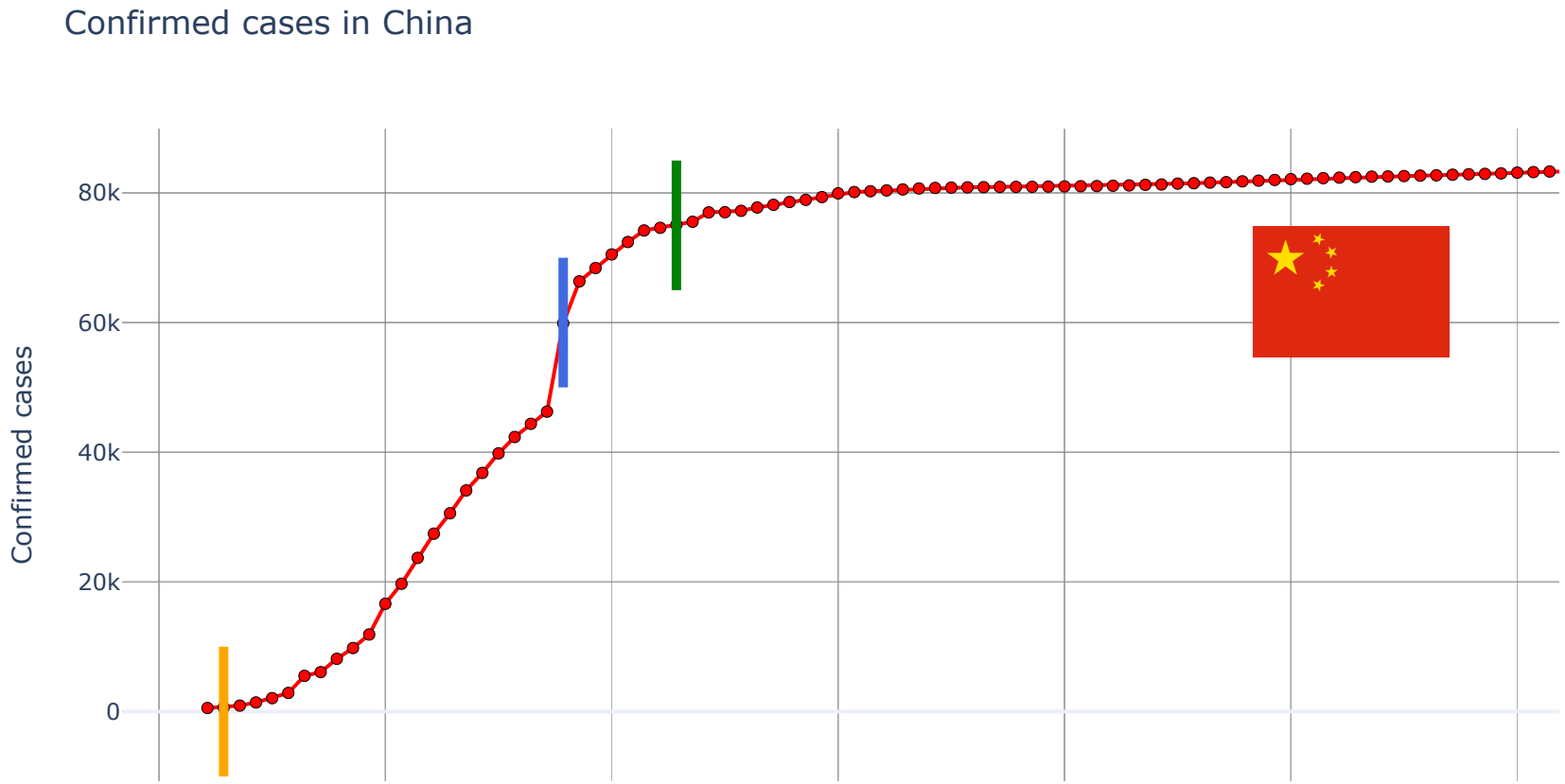
Lets see the measure taken by China to contain the virus

1) Wuhan was locked down on Jan 23rd 2020

2) Factories were closed across China on Feb 13th 2020

3) Complete (total) lockdown was imposed across China on Feb 20th 2020

In [50]:
```
fig = go.Figure()
visualize_country(fig, "China", "https://upload.wikimedia.org/wikipedia/commons/f/fa/Flag_of_the_People%27s_Republic_of_China.svg", colors=["red"], step=1000, xcor=0.85, ycor=0.65, done=False)
fig.add_shape(
        dict(
            type="line",
            x0=Timestamp('2020-02-13 00:00:00'),
            y0=50000,
            x1=Timestamp('2020-02-13 00:00:00'),
            y1=70000,
            line=dict(
                color="RoyalBlue",
                width=5
            )
))
fig.add_shape(
        dict(
            type="line",
            x0=Timestamp('2020-02-20 00:00:00'),
            y0=65000,
            x1=Timestamp('2020-02-20 00:00:00'),
            y1=85000,
            line=dict(
                color="Green",
                width=5
            )
))
fig.add_shape(
        dict(
            type="line",
            x0=Timestamp('2020-01-23 00:00:00'),
            y0=-10000,
            x1=Timestamp('2020-01-23 00:00:00'),
            y1=10000,
            line=dict(
                color="Orange",
                width=5
            )
))
fig.update_layout(title="Confirmed cases in China", showlegend=False)
fig.show()
```



We have plotted the number of new cases everyday in China above. The orange represents when Wuhan was locked down, the blue represents when factories were closed across China, and the green represents when complete (total) lockdown was imposed across China. Notice how the curve starts to flatten after the complete lockdown is imposed. Complete lockdown helps reduce community transmission and mitigate the virus.

Conclusion: China relied on complete lockdown to contain the virus.

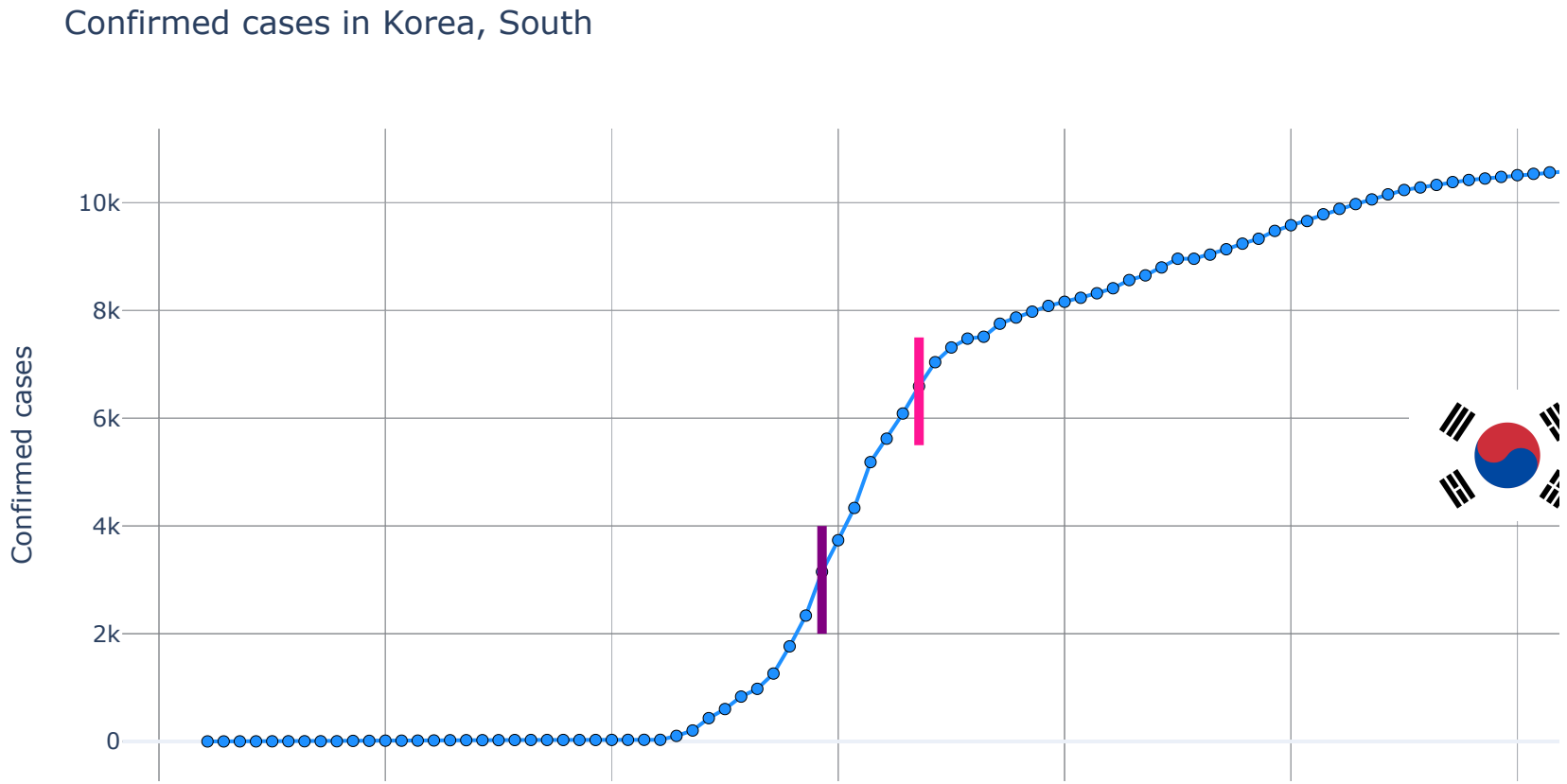
What strategies South Korea adopted to contain Covid-19

Measures taken by South Korea to contain the virus:

1) South Korea ramped up testing on February 29th

2) Government introduced a new GPS-enabled quarantine tracking app on March 6th

In [51]:
```
fig = go.Figure()
visualize_country(fig, "South Korea", "https://upload.wikimedia.org/wikipedia/commons/0/09/Flag_of_South_Kore
a.svg", colors=["dodgerblue"], step=80, xcor=0.95, ycor=0.4, done=False)
fig.add_shape(
        dict(
            type="line",
            x0=Timestamp('2020-02-29 00:00:00'),
            y0=2000,
            x1=Timestamp('2020-02-29 00:00:00'),
            y1=4000,
            line=dict(
                color="purple",
                width=5
            )
))
fig.add_shape(
        dict(
            type="line",
            x0=Timestamp('2020-03-06 00:00:00'),
            y0=5500,
            x1=Timestamp('2020-03-06 00:00:00'),
            y1=7500,
            line=dict(
                color="deeppink",
                width=5
            )
))
fig.update_layout(title="Confirmed cases in Korea, South", showlegend=False)
fig.show()
```



We have plotted the number of new cases everyday in South Korea above. The purple represents when South Korea ramped up testing, and the pink represents the when a new GPS-enabled quarantine tracking app was deployed by the South government. These two measures have together worked to reduce community transmission and flatten curve towards the end of the first week of March.

Conclusion: South Korea relied on mass testing and technology to contain the virus.

4) Finding cures for COVID-19

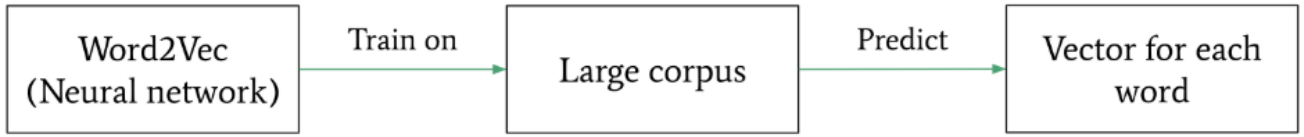
Now, we will leverage the power of unsupervised machine learning to try and find possible cures (medicines and drugs) to COVID-19.

Unsupervised NLP and Word2Vec

Unsupervised NLP involves the analysis of unlabeled language data. Certain techniques can be used to derive insights from a large corpus of text. One such method is called Word2Vec. Word2Vec is a neural network architecture trained on thousands of sentences of text. After training, the neural network finds the optimal vector representation of each word in the corpus. These vectors are meant to reflect the meaning of the word. Words with similar meanings have similar vectors.

In [52]:
```
from IPython.display import Image
Image(filename = 'D:/GSUCoursework/BigDataExp/Datasets/Pictures/Word2Vec_Basic.PNG', width = 700, height = 800)
```

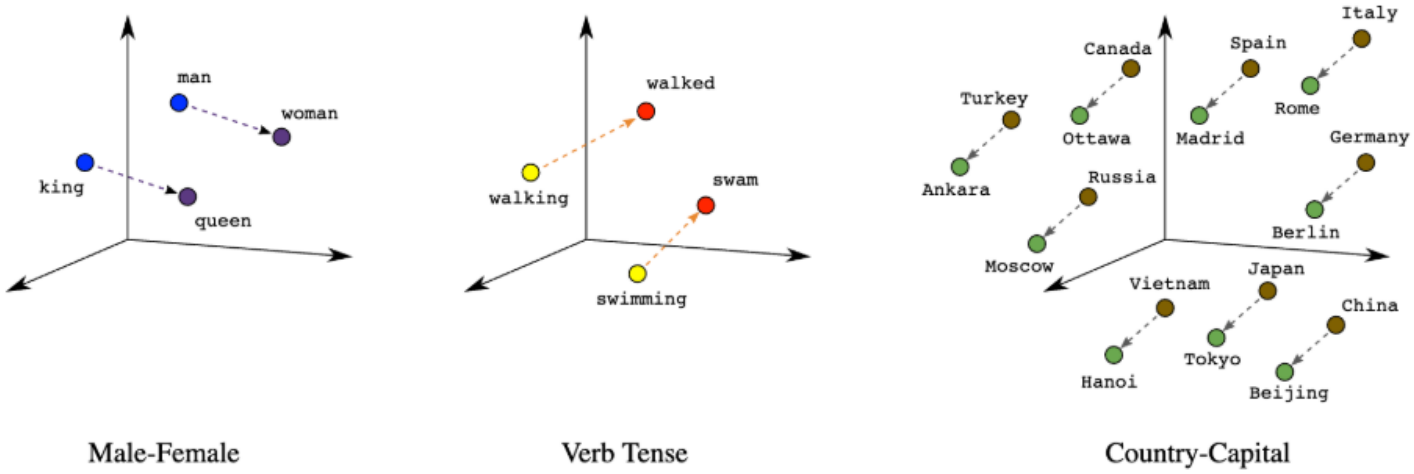
Out[52]:



As I stated earlier, each word is associated with a vector. Amazingly, these vectors can also encode relationships and analogies between words. The diagram below iillustrates some examples of linear vector relationships representing the relationships between words.

In [53]:
```
from IPython.display import Image
Image(filename = 'D:/GSUCoursework/BigDataExp/Datasets/Pictures/Wrod2Vec_Relationships.PNG', width = 800, hei
ght = 700)
```

Out[53]:



In the above image, we can see that word vectors can reflect relationships such as "King is to Queen as Man is to Woman" or "Italy is to Rome" as "Germany is to Berlin". These vectors can be also be used to find unknown relationships between words. These unknown relationships may help us find latent knowledge in research papers and find drugs that can possibly cure COVID_19!

Using Word2Vec to find cures

We can take advantage of these intricate relationships between word vectors to find cures for COVID-19. The steps are as follows:

Step 1 - Find common related to the study of COVID-19, such as "infection", "CoV", "viral", etc.

Step 2 - Find the words with lowest Euclidean distance to these words (most similar words).

Step 3 - Finally, find the words most similar to these words (second order similarity). These words will hopefully contain potential COVID-19 cures.

Note that the similarity between two Word2Vec vectors is calculated using the formula below (where u and v are the word vectors).

In [54]:
```
from IPython.display import Image
Image(filename = 'D:/GSUCoursework/BigDataExp/Datasets/Pictures/Word2Vec_Formula.PNG', width = 500, height = 500)
```

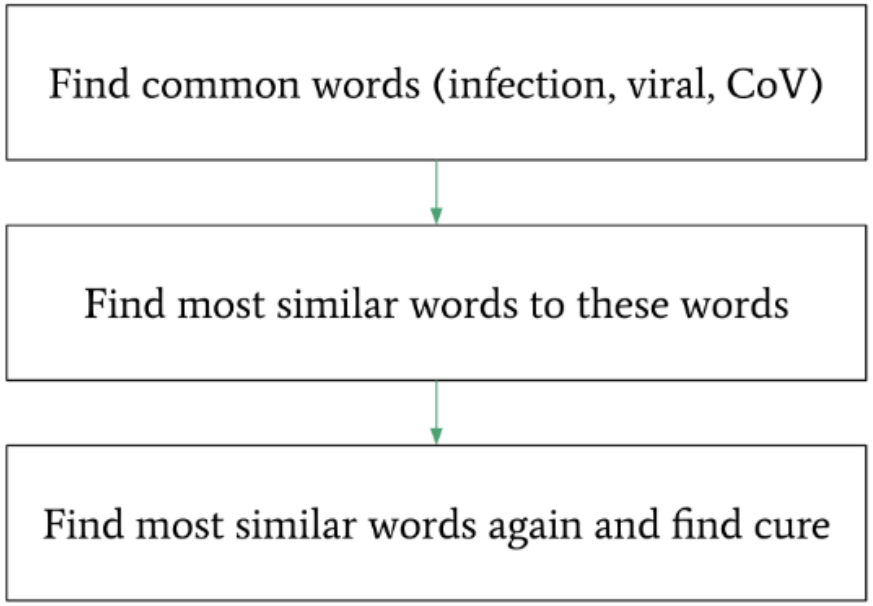
Out[54]:

$$Similarity = ||u - v||_2^2 = \sqrt{\sum_{i=1}^{n}(u_i - v_i)^2}$$

The entire process can be summarized with the flowchart below. (the same steps as given above)

In [55]:
```
from IPython.display import Image
Image(filename = 'D:/GSUCoursework/BigDataExp/Datasets/Pictures/NLP_three_Steps.PNG', width = 500, height = 5
00)
```

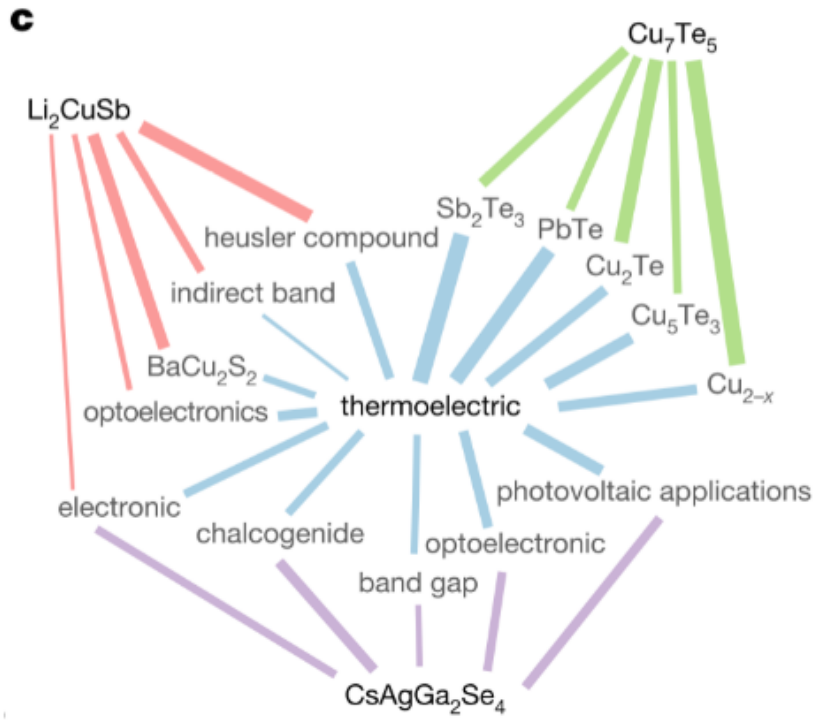
Out[55]:



The approach detailed above is actually inspired by a research paper called "Unsupervised word embeddings capture latent knowledge from materials science literature", where the authors find new materials with desirable properties (such as thermoelectricity) solely based on a large corpus materials science literature. These materials were never used for these purposes before, but they outperform old materials by a large margin. I hope to emulate the same method to look for COVID-19 cures. The diagram below illustrates what the authors did in their research.

In [56]:
```
from IPython.display import Image
Image(filename = 'D:/GSUCoursework/BigDataExp/Datasets/Pictures/Thermoelectricity.PNG', width = 500, height = 500)
```

Out[56]:



In the diagram above, we can see that the authors found two levels of words similar to "thermoelectric" in a heirarchical manner. The second order similar words contained compounds like Li2CuSb, Cu7Te5, and CsAgGa2Se4, which turned out to be very good thermoelectric materials in real life.

First, we need to find the most common words in the corpus to continue our analysis. To find the most common words, we can create a word cloud

Word cloud of abstracts

In [57]:
```
def nonan(x):
    if type(x) == str:
        return x.replace("\n", "")
    else:
        return ""

text = ' '.join([nonan(abstract) for abstract in papers_df["abstract"]])
wordcloud = WordCloud(max_font_size=None, background_color='white', collocations=False,
                      width=1200, height=1000).generate(text)
fig = px.imshow(wordcloud)
fig.update_layout(title_text='Common words in abstracts')
```

Common words in abstracts

From the above word cloud, we can see that "infection", "cell", "virus", and "protein" are among the most common words in COVID-19 research paper abstracts. These words will form our "keyword" list.

Train Word2Vec Model

In [87]:
```
def nltk_tag_to_wordnet_tag(nltk_tag):
    if nltk_tag.startswith('J'):
        return wordnet.ADJ

    elif nltk_tag.startswith('V'):
        return wordnet.VERB

    elif nltk_tag.startswith('N'):
        return wordnet.NOUN

    elif nltk_tag.startswith('R'):
        return wordnet.ADV

    else:
        return None

def lemmatize_sentence(sentence):
    nltk_tagged = nltk.pos_tag(nltk.word_tokenize(sentence))
    wordnet_tagged = map(lambda x: (x[0], nltk_tag_to_wordnet_tag(x[1])), nltk_tagged)
    lemmatized_sentence = []

    for word, tag in wordnet_tagged:
        if tag is None:
            lemmatized_sentence.append(word)
        else:
            lemmatized_sentence.append(lemmatizer.lemmatize(word, tag))

    return " ".join(lemmatized_sentence)

def clean_text(abstract):
    abstract = abstract.replace(". ", " ").replace(", ", " ").replace("! ", " ")\
                       .replace("? ", " ").replace(": ", " ").replace("; ", " ")\
                       .replace("( ", " ").replace(") ", " ").replace("| ", " ").replace("/ ", " ")
    if "." in abstract or "," in abstract or "!" in abstract or "?" in abstract or ":" in abstract or ";" in
abstract or "(" in abstract or ")" in abstract or "|" in abstract or "/" in abstract:
        abstract = abstract.replace(".", " ").replace(",", " ").replace("!", " ")\
                           .replace("?", " ").replace(":", " ").replace(";", " ")\
                           .replace("(", " ").replace(")", " ").replace("|", " ").replace("/", " ")
    abstract = abstract.replace("  ", " ")

    for word in list(set(stopwords.words("english"))):
        abstract = abstract.replace(" " + word + " ", " ")
```

```
    return lemmatize_sentence(abstract).lower()

def get_similar_words(word, num):
    vec = model_wv_df[word].T
    distances = np.linalg.norm(model_wv_df.subtract(model_wv_df[word],
                                                    axis=0).values, axis=0)

    indices = np.argsort(distances)
    top_distances = distances[indices[1:num+1]]
    top_words = model_wv_vocab[indices[1:num+1]]
    return top_words

def visualize_word_list(color, word):
    top_words = get_similar_words(word, num=6)
    relevant_words = [get_similar_words(word, num=8) for word in top_words]
    fig = make_subplots(rows=3, cols=2, subplot_titles=tuple(top_words), vertical_spacing=0.05)
    for idx, word_list in enumerate(relevant_words):
        words = [word for word in word_list if word in model_wv_vocab]
        X = model_wv_df[words].T
        pca = PCA(n_components=2)
        result = pca.fit_transform(X)
        df = pd.DataFrame(result, columns=["Component 1", "Component 2"])
        df["Word"] = word_list
        word_emb = df[["Component 1", "Component 2"]].loc[0]
        df["Distance"] = np.sqrt((df["Component 1"] - word_emb[0])**2 + (df["Component 2"] - word_emb[1])**2)
        plot = px.scatter(df, x="Component 1", y="Component 2", text="Word", color="Distance", color_continuo
us_scale=color, size="Distance")
        plot.layout.title = top_words[idx]
        plot.update_traces(textposition='top center')
        plot.layout.xaxis.autorange = True
        plot.data[0].marker.line.width = 1
        plot.data[0].marker.line.color = 'rgb(0, 0, 0)'
        fig.add_trace(plot.data[0], row=(idx//2)+1, col=(idx%2)+1)
    fig.layout.coloraxis.showscale = False
    fig.update_layout(height=1400, title_text="2D PCA of words related to {}".format(word), paper_bgcolor="#f
0f0f0", template="plotly_white")
    return fig

def visualize_word(color, word):
    top_words = get_similar_words(word, num=20)
    words = [word for word in top_words if word in model_wv_vocab]
    X = model_wv_df[words].T
```

```
    pca = PCA(n_components=2)
    result = pca.fit_transform(X)
    df = pd.DataFrame(result, columns=["Component 1", "Component 2"])
    df["Word"] = top_words
    if word == "antimalarial":
        df = df.query("Word != 'anti-malarial' and Word != 'anthelmintic'")
    if word == "doxorubicin":
        df = df.query("Word != 'anti-rotavirus'")
    word_emb = df[["Component 1", "Component 2"]].loc[0]
    df["Distance"] = np.sqrt((df["Component 1"] - word_emb[0])**2 + (df["Component 2"] - word_emb[1])**2)
    fig = px.scatter(df, x="Component 1", y="Component 2", text="Word", color="Distance", color_continuous_sc
ale=color, size="Distance")
    fig.layout.title = word
    fig.update_traces(textposition='top center')
    fig.layout.xaxis.autorange = True
    fig.layout.coloraxis.showscale = True
    fig.data[0].marker.line.width = 1
    fig.data[0].marker.line.color = 'rgb(0, 0, 0)'
    fig.update_layout(height=800, title_text="2D PCA of words related to {}".format(word), template="plotly_w
hite", paper_bgcolor="#f0f0f0")
    fig.show()
```

In [88]:
```
import nltk
!pip install nltk
nltk.download('averaged_perceptron_tagger')
nltk.download('wordnet')
```

```
Requirement already satisfied: nltk in c:\users\abhij\anaconda3\lib\site-packages (3.4.5)
Requirement already satisfied: six in c:\users\abhij\anaconda3\lib\site-packages (from nltk) (1.14.0)

[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\abhij\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\abhij\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Out[88]:
```
True
```

In [89]:
```
lemmatizer = WordNetLemmatizer()

def get_words(abstract):
    return clean_text(nonan(abstract)).split(" ")
```

In [ ]:
```
words = papers_df["abstract"].progress_apply(get_words)
model = Word2Vec(words, size=200, sg=1, min_count=1, window=8, hs=0, negative=15, workers=1)
```

In [91]:
```
words
```

Out[91]:
```
0        [abstractthe, objective, study, evaluate, bene...
1                                                       []
2                                                       []
3        [abstractthis, research, report, design, analy...
4        [abstractrecibido, el, 21, de, diciembre, de, ...
                               ...
29310    [abstractobjective, tumor-treating, field, cur...
29311                                                   []
29312    [abstractbeijing, one, epicenter, attack, seve...
29313    [abstractadrenal, insufficiency, rare, potenti...
29314    [abstractwe, provide, first, genetic, sequence...
Name: abstract, Length: 29315, dtype: object
```

Cures

In [92]:
```
from sklearn.decomposition import PCA
import pandas as pd
import numpy as np
from gensim.models import Word2Vec
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import warnings
warnings.filterwarnings("ignore")
```

Visualize most similar words to the keywords

In [93]:
```
# Most similar words to Coronavirus
model.wv.most_similar('coronavirus', topn=20)
```

Out[93]:
```
[('coronaviruses', 0.7745884656906128),
 ('cov', 0.748910665512085),
 ('-cov', 0.7360830903053284),
 ('syndrome-cov', 0.7300434112548828),
 ('ß-cov', 0.7133302688598633),
 ('sars-hcov', 0.7062472701072693),
 ('beta-coronavirus', 0.7039520740509033),
 ('syndrome-coronavirus', 0.7034022808074951),
 ('middle-east', 0.6998024582862854),
 ('abstractsars-cov', 0.6977753043174744),
 ('dcov', 0.6943050026893616),
 ('-coronavirus', 0.6911903619766235),
 ('abstractcases', 0.6898853182792664),
 ('severe-acute', 0.6896257996559143),
 ('jhm-cov', 0.6892759799957275),
 ('sars-coronavirus', 0.6885084509849548),
 ('corona', 0.687771737575531),
 ('syndrome-like', 0.6873286366462708),
 ('coronavirus-229e', 0.6872488260269165),
 ('emc2012', 0.6853234767913818)]
```

In [94]:
```
# lets print all the similar words to our keywords generated from word cloud
keywords = ["infection", "cell", "protein", "virus",\
            "disease", "respiratory", "influenza", "viral",\
            "rna", "patient", "pathogen", "human", "medicine",\
            "cov", "antiviral"]

print("Most similar words to keywords")
print("")

#top_words_list = []
for word in keywords:
    top_words = model.wv.most_similar(word, topn=20)
    print(word + ":")
    for idx, top_word in enumerate(top_words):
        print(str(idx+1) + ". " + top_word[0])
        #top_words_list.append(top_word[0])
    print("")
```

```
Most similar words to keywords

infection:
1. oligosymptomatic
2. predisposes
3. immuno-competent
4. postviral
5. nonatopic
6. investigations
7. hmpv-and
8. infection-induced
9. experimentally-induced
10. re-infection
11. advisable
12. early-onset
13. lowered
14. ntabs
15. misdirect
16. non-polio
17. virusassociated
18. care-acquired
19. speciwc
20. infection-related

cell:
1. h1299
2. polarized
3. ceils
4. ags
5. monocyte-derived
6. huvec
7. ebk
8. raji
9. fibroblastic
10. huvecs
11. ebv-infected
12. co-cultures
13. fibroblasts
14. cd58
15. multipotent
16. proliferate
17. cocultures
18. u937
```

```
19. lm-k
20. hep3b

protein:
1. proteins
2. ezrin
3. unglycosylated
4. s100
5. pp60
6. nonglycosylated
7. ns3a
8. 140k
9. calnexin
10. periplasmic
11. colocalizes
12. nsp1␣
13. nanovesicles
14. ha3g
15. sars_cov
16. alphaviral
17. encapsidates
18. multi-functional
19. histidine-tagged
20. cytoplasmically

virus:
1. viruses
2. corona-and
3. paramyxo
4. viral
5. rubulavirus
6. pneumoviridae
7. notoriously
8. anatid
9. bav
10. fluv
11. vesiviruses
12. hastvs
13. rna-viruses
14. adeno-
15. calici
16. paramyxoviruses
17. lyssaviruses
```

```
18. paramyxovirus-1
19. pneumovirus
20. enterotropic

disease:
1. diseases
2. abstractdisease
3. repercussion
4. multi-factorial
5. nondomestic
6. mycosis
7. cloven-hoofed
8. non-specifi
9. incurable
10. rids
11. mets
12. human-only
13. counter-measures
14. co-ordinated
15. malignancies
16. schistosomiasis
17. recurs
18. tame
19. dread
20. livestock-related

respiratory:
1. alrtis
2. upper-respiratory
3. lrt
4. saris
5. nonpneumonic
6. alrti
7. hpivs
8. mono-infections
9. throat-swab
10. key-words
11. -hku1
12. troublesome
13. adenovirures
14. 2003-2004
15. respiratory-tract
16. underdiagnosed
```

```
17. lower-respiratory
18. nvri
19. pneumonia-like
20. wheezy

influenza:
1. h1n1
2. h3n2
3. pdm09
4. flu
5. uenza
6. swine-origin
7. h7n7
8. s-oiv
9. h2n2
10. h1n1pdm09
11. 2009-2010
12. infl
13. h1n1-2009
14. pdm
15. iavs
16. noninfluenza
17. h7n2
18. ph1n1
19. influenzas
20. pdm2009

viral:
1. virus
2. postviral
3. under-studied
4. single-and
5. speciwc
6. virusassociated
7. nodavirus
8. host-viral
9. wtc
10. diagnosing
11. non-polio
12. aetiologic
13. trypanosomes
14. cbdvgs
15. aberrantly
```

16. establishes
17. presumptive
18. co-
19. virally-induced
20. rtc-proximal

rna:
1. abstractrna
2. plus-strand
3. rnas
4. negative-strand
5. ssrna
6. tombusviruses
7. grna
8. negative-sense
9. templates
10. encapsidated
11. vrna
12. minus-strand
13. rna-induced
14. positive-strand
15. rna-rna
16. tbsv
17. sgmrnas
18. genomic-sized
19. encapsidation
20. pirna

patient:
1. patients
2. fob
3. ili-score
4. intubated
5. alrips
6. misdiagnosed
7. tonsillectomy
8. npae
9. admitted
10. non-hcws
11. nippv
12. neutropenic
13. non-ipf
14. arf

```
15. post-hct
16. ilds
17. 5119
18. oliguria
19. dialysis
20. esrd

pathogen:
1. pathogens
2. food-and
3. aetiologic
4. microbe
5. stis
6. microbes
7. abstractdiseases
8. legionellosis
9. uncultivable
10. food-borne
11. eopm
12. abstractbacterial
13. slow-growing
14. culture-dependent
15. trichomonosis
16. rarity
17. protozoal
18. molecular-based
19. pseudomallei
20. coevolved

human:
1. abstracthuman
2. hev4
3. comprising
4. hev3
5. 229e-based
6. bat-borne
7. quantifi
8. fvs
9. insectivore
10. infectivities
11. hev-7
12. species-specificity
13. zoonotically
```

```
14. lyssaviruses
15. weaponize
16. hpvs
17. sars-hcov
18. human-only
19. summaryhuman
20. translatability

medicine:
1. folk
2. medica
3. kampo
4. biomedicine
5. materia
6. tcm
7. formulas
8. strobilanthes
9. tcms
10. chm
11. ayurveda
12. herbal
13. multiflorum
14. fortune
15. botany
16. houttuynia
17. abstracttraditional
18. ailment
19. listing
20. magnolia

cov:
1. covs
2. coronaviruses
3. -cov
4. coronavirus
5. sars-cov
6. beta-coronavirus
7. sl-cov
8. bat-sars-cov
9. betacoronaviruses
10. syndrome-cov
11. nsp14-exon
12. sars-covs
```

```
13. abstractsars-cov
14. syndrome-like
15. beta-covs
16. mers-like
17. alphacov
18. merbecovirus
19. β-coronaviruses
20. sars-hcov

antiviral:
1. anti-viral
2. anti-coronavirus
3. anti-hbv
4. broad-spectrum
5. anti-hsv
6. chemotherapeutics
7. anti-ev71
8. anti-pathogenic
9. host-targeting
10. host-targeted
11. ntz
12. anti-hcv
13. antineoplastic
14. anticoronaviral
15. anti-virus
16. anti-zika
17. antagonists
18. hydroxychloroquine
19. antileishmanial
20. anti-iav
```

In the cell above, We have printed the most similar words to the 15 keywords (based on Euclidean distance). These words will form the next batch of words, which we will analyze to find cures to COVID-19.

In [95]:
```
# Lets see the vectors generated by model for the keywords that we got from wordcloud like coronavirus,infect
ion etc.
model['coronavirus'].shape
```

Out[95]:
```
(200,)
```

We can see, we have 200D vectors for the keyword 'coronavirus'

In [96]:
```
model['coronavirus']
```

Out[96]:
```
array([ 0.4615617 , -0.00901044, -0.4101654 ,  0.29175815,  0.3929378 ,
        0.27475485,  0.28804126, -0.04705918, -0.53638697, -0.11424761,
        0.02004503,  0.3806448 , -0.29235843,  0.2861658 ,  0.28292102,
        0.36630425, -0.01463398, -0.2862839 , -0.0521614 , -0.2245927 ,
       -0.15859687,  0.2891471 , -0.22210464, -0.24581833, -0.47125393,
        0.34812787, -0.10417574,  0.29509473, -0.23984113, -0.42109954,
        0.33525   , -0.20193085,  0.19189946,  0.0104987 ,  0.2865111 ,
       -0.24246925,  0.39607388, -0.07957052,  0.15215124,  0.1762481 ,
       -0.26825967, -0.14957179, -0.11931015, -0.6759713 ,  0.44224176,
       -0.17991503, -0.5768627 , -0.07067036,  0.55670524,  0.5764254 ,
        0.06325256,  0.25018764, -0.00749143,  0.55658907,  0.1302035 ,
        0.6509206 , -0.6369911 , -0.20705236, -0.21272177, -0.52980393,
       -0.2873021 ,  0.27615988, -0.22033885, -0.420798  , -0.41515946,
       -0.22784467,  0.22035483, -0.17420399,  0.19480374,  0.35217726,
       -0.19949743, -0.23806378, -0.2153483 ,  0.08181147,  0.20679344,
        0.12508734, -0.24883856,  0.52669466, -0.07846528,  0.32778242,
       -0.04849954, -0.26007307,  0.22915034,  0.06007564, -0.4618628 ,
       -0.24958049,  0.06937788,  0.11094405,  0.5558324 , -0.1464262 ,
        0.18962353, -0.46009126, -0.1785799 ,  0.25489113,  0.571201  ,
       -0.60449296, -0.22241552, -0.27032036, -0.38023394,  0.04200021,
       -0.53445965,  0.23382574,  0.05117855, -0.0741206 ,  0.21111956,
        0.17474285, -0.16982159,  0.18500401,  0.04202879,  0.17105407,
       -0.00383776,  0.07952704,  0.4897644 ,  0.46903092,  0.1879889 ,
       -0.11279085,  0.35299936, -0.23764935, -0.16527484, -0.19251518,
        0.05740533, -0.4002057 ,  0.09747632, -0.02672693, -0.32757542,
        0.0753654 ,  0.46628964,  0.12950629, -0.39880332, -0.5567139 ,
       -0.05273085,  0.14834179, -0.23797077, -0.1546192 ,  0.27031192,
       -0.66266054,  0.43217868,  0.23905987,  0.19884826, -0.23050146,
        0.34197032,  0.36666083, -0.6640435 ,  0.07922222, -0.1256697 ,
        0.03946539,  0.29394016, -0.21275914,  0.27326205,  0.6307207 ,
       -0.19593684, -0.44605565,  0.16142714, -0.18417056, -0.05744379,
        0.38213304,  0.35325113,  0.1971504 ,  0.04667094, -0.5363332 ,
        0.25794673,  0.01293749,  0.42905885, -0.34240943, -0.5512193 ,
        0.34277108,  0.08903651, -0.02765084, -0.14056784, -0.12053715,
       -0.30569726,  0.04673026,  0.03332438, -0.08568349,  0.6243508 ,
        0.71236956, -0.08842807, -0.06144186,  0.44791502,  0.24972683,
        0.13526598,  0.30942723,  0.07162203, -0.15377204, -0.25804567,
       -0.16307144,  0.58303386,  0.44848675,  0.54181063,  0.6674111 ,
        0.20934169, -0.08000875,  0.14227243,  0.34921077,  0.11151724,
        0.4457569 ,  0.22119084,  0.16633248, -0.0357211 , -0.28250337],
      dtype=float32)
```

In [97]:
```
# Similarly for infection, we have 200D vectors
model['infection'].shape
```

Out[97]:
```
(200,)
```

In [98]:
```
# Passing all the keywords in variable X and then checking the shape
words = [word for word in keywords]
X = model[words]
X.shape
```

Out[98]:
```
(15, 200)
```

PCA

PCA is a dimensionality reduction method which takes vectors with several dimensions and compresses it into a smaller vector (with 2 or 3 dimensions) while preserving most of the information in the original vector (using some linear algebra). PCA makes visualization easier while dealing with high-dimensional data, such as Word2Vec vectors.

2-D PCA of keyword vectors

In [99]:
```
from IPython.display import Image
Image(filename = 'D:/GSUCoursework/BigDataExp/Datasets/Pictures/PCA.PNG', width = 600, height = 600)
```

Out[99]:

In [100]:
```
# Principal Component Analysis
pca = PCA(n_components=2)
result = pca.fit_transform(X)
df = pd.DataFrame(result, columns=["Component 1", "Component 2"])
df
```

Out[100]:

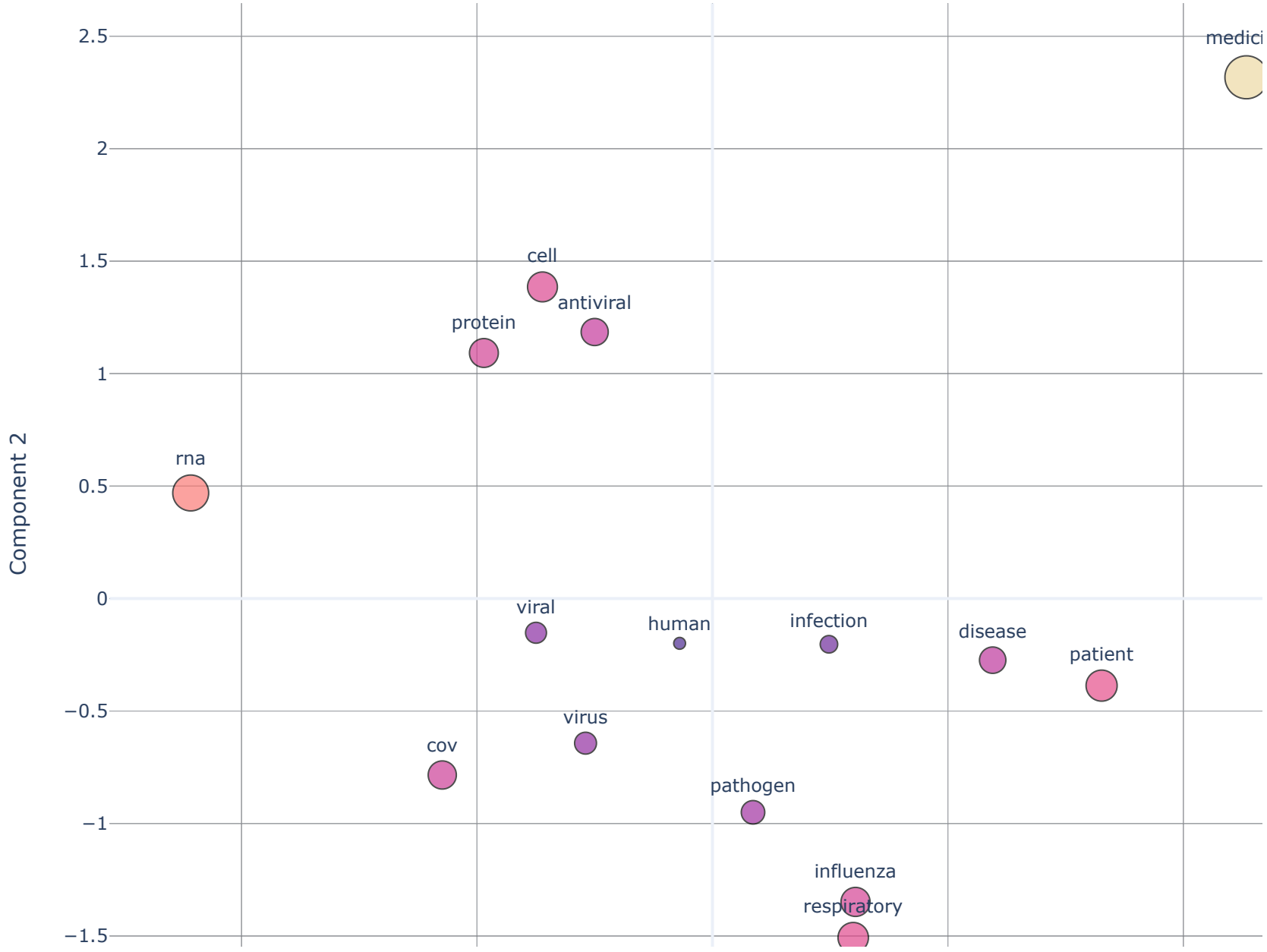
	Component 1	Component 2
0	0.494838	-0.203209
1	-0.721916	1.385460
2	-0.970664	1.091723
3	-0.538951	-0.642886
4	1.190258	-0.273905
5	0.597928	-1.507458
6	0.607677	-1.348558
7	-0.749352	-0.152106
8	-2.215995	0.469415
9	1.652938	-0.387069
10	0.172366	-0.950249
11	-0.139359	-0.199037
12	2.267911	2.317308
13	-1.147549	-0.784348
14	-0.500134	1.184921

In [101]:
```
# z1 and z2 Components for the keywords
df["Word"] = words
df["Distance"] = np.sqrt(df["Component 1"]**2 + df["Component 2"]**2)
df
```

Out[101]:

	Component 1	Component 2	Word	Distance
0	0.494838	-0.203209	infection	0.534938
1	-0.721916	1.385460	cell	1.562262
2	-0.970664	1.091723	protein	1.460838
3	-0.538951	-0.642886	virus	0.838910
4	1.190258	-0.273905	disease	1.221368
5	0.597928	-1.507458	respiratory	1.621712
6	0.607677	-1.348558	influenza	1.479148
7	-0.749352	-0.152106	viral	0.764634
8	-2.215995	0.469415	rna	2.265167
9	1.652938	-0.387069	patient	1.697654
10	0.172366	-0.950249	pathogen	0.965756
11	-0.139359	-0.199037	human	0.242975
12	2.267911	2.317308	medicine	3.242428
13	-1.147549	-0.784348	cov	1.389989
14	-0.500134	1.184921	antiviral	1.286146

In [102]:
```
fig = px.scatter(df, x="Component 1", y="Component 2", text="Word", color="Distance", color_continuous_scale=
"agsunset",size="Distance")
fig.update_traces(textposition='top center')
fig.layout.xaxis.autorange = True
fig.data[0].marker.line.width = 1
fig.data[0].marker.line.color = 'rgb(0, 0, 0)'
fig.update_layout(height=800, title_text="2D PCA of Word2Vec embeddings", template="plotly_white", paper_bgco
lor="#f0f0f0")
fig.show()
```


2D PCA of Word2Vec embeddings
Component 2
2.5
2
1.5
1
0.5
0
−0.5
−1
−1.5
medici
cell
antiviral
protein
rna
viral
human
infection
disease
patient
virus
cov
pathogen
influenza
respiratory

In the above plot, we can see the 2D PCA of the keywords' vectors.

1)The words "virus", "viral", and "CoV" form a cluster in the bottom-right part of the plot, indicating that they have similar meanings. This makes sense because CoV is a virus.

2)The words "medicine" and "patient" are both on the far left end of the image because these words are used together very frequently.

3)The "pathogen", "influenza", and "respiratory" form a cluster in the bottom-left part of the plot, indicating that they have similar meanings. This makes sense because influenza is a repsiratory disease.

These abstract linguistic relationships are successfully represented by word vectors.

Let's pick a specific keyword

In [103]:
```
# My Keywords
keywords
```

Out[103]:
```
['infection',
 'cell',
 'protein',
 'virus',
 'disease',
 'respiratory',
 'influenza',
 'viral',
 'rna',
 'patient',
 'pathogen',
 'human',
 'medicine',
 'cov',
 'antiviral']
```

To test results: 2D PCA of words similar to Cov and disease

Keyword: Cov

In [104]:
```
keyword = 'cov'
similar_words = model.wv.most_similar(keyword, topn=20)
df_similar_words = pd.DataFrame(similar_words, columns = ['word', 'dist'])
words = [word for word in df_similar_words['word'].tolist()]
X = model[words]
result = pca.fit_transform(X)
df = pd.DataFrame(result, columns=["Component 1", "Component 2"])
df["Word"] = df_similar_words['word']
word_emb = df[["Component 1", "Component 2"]].loc[0]
df["Distance"] = np.sqrt((df["Component 1"] - word_emb[0])**2 + (df["Component 2"] - word_emb[1])**2)
fig = px.scatter(df[2:], x="Component 1", y="Component 2", text="Word", color="Distance", color_continuous_sc
ale="viridis",size="Distance")
fig.update_traces(textposition='top center')
fig.layout.xaxis.autorange = True
fig.data[0].marker.line.width = 1
fig.data[0].marker.line.color = 'rgb(0, 0, 0)'
fig.update_layout(height=800, title_text="2D PCA of words related to {}".format(keyword), template="plotly_wh
ite", paper_bgcolor="#f0f0f0")
fig.show()
```


2D PCA of words related to cov
Component 2
1.5
1
0.5
0
−0.5
−1
coronav
sars-cov
-cov
syndrome-cov
abstractsars-cov
syndrome-like
nsp14-exon
sars-hcov
beta-coronavirus
sl-cov
sars-covs
alphacov
mers-like
betacoronaviruses

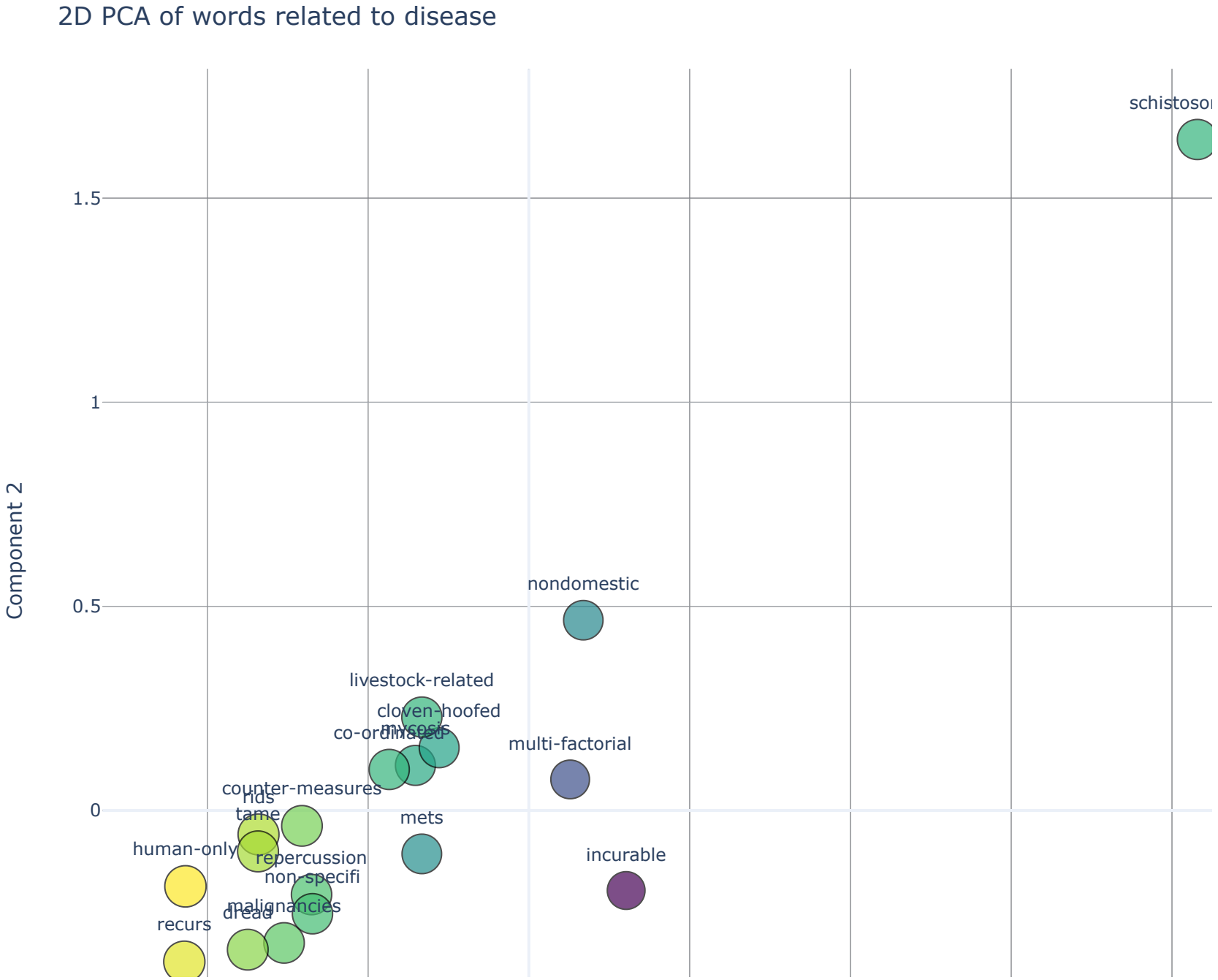
We have plotted the 2D PCA of the words most similar to CoV (stands for CoronaVirus) above.

We can see few words like "coronavirus", "SARS-CoV", and "coronaviral" which are almost synonymal with CoV. These words are surprisingly very close to "CoV" in the vector space.

We can also see a clear cluster in the bottom-left corner of the plot, and these words are also closely linked with the word "CoV".

Keyword: disease

In [105]:
```
keyword = 'disease'
similar_words = model.wv.most_similar(keyword, topn=20)
df_similar_words = pd.DataFrame(similar_words, columns = ['word', 'dist'])
words = [word for word in df_similar_words['word'].tolist()]
X = model[words]
result = pca.fit_transform(X)
df = pd.DataFrame(result, columns=["Component 1", "Component 2"])
df["Word"] = df_similar_words['word']
word_emb = df[["Component 1", "Component 2"]].loc[0]
df["Distance"] = np.sqrt((df["Component 1"] - word_emb[0])**2 + (df["Component 2"] - word_emb[1])**2)
fig = px.scatter(df[2:], x="Component 1", y="Component 2", text="Word", color="Distance", color_continuous_sc
ale="viridis",size="Distance")
fig.update_traces(textposition='top center')
fig.layout.xaxis.autorange = True
fig.data[0].marker.line.width = 1
fig.data[0].marker.line.color = 'rgb(0, 0, 0)'
fig.update_layout(height=800, title_text="2D PCA of words related to {}".format(keyword), template="plotly_wh
ite", paper_bgcolor="#f0f0f0")
fig.show()
```


2D PCA of words related to disease
Component 2
1.5
1
0.5
0
schistoso
nondomestic
livestock-related
cloven-hoofed
co-ordinated
multi-factorial
counter-measures
tame
mets
human-only
repercussion
incurable
non-specifi
malignancies
dread
recurs

We can see few words like "echinococcus", "mets", and "echinococcus", these words are closely linked with the our keyword "disease"

echinococcus is a granulosus disease, it is a tapeworm infection that affects the liver, lungs, brain, and other organs.

mets stands for Metastatic is a bone disease that results in spinal tumor

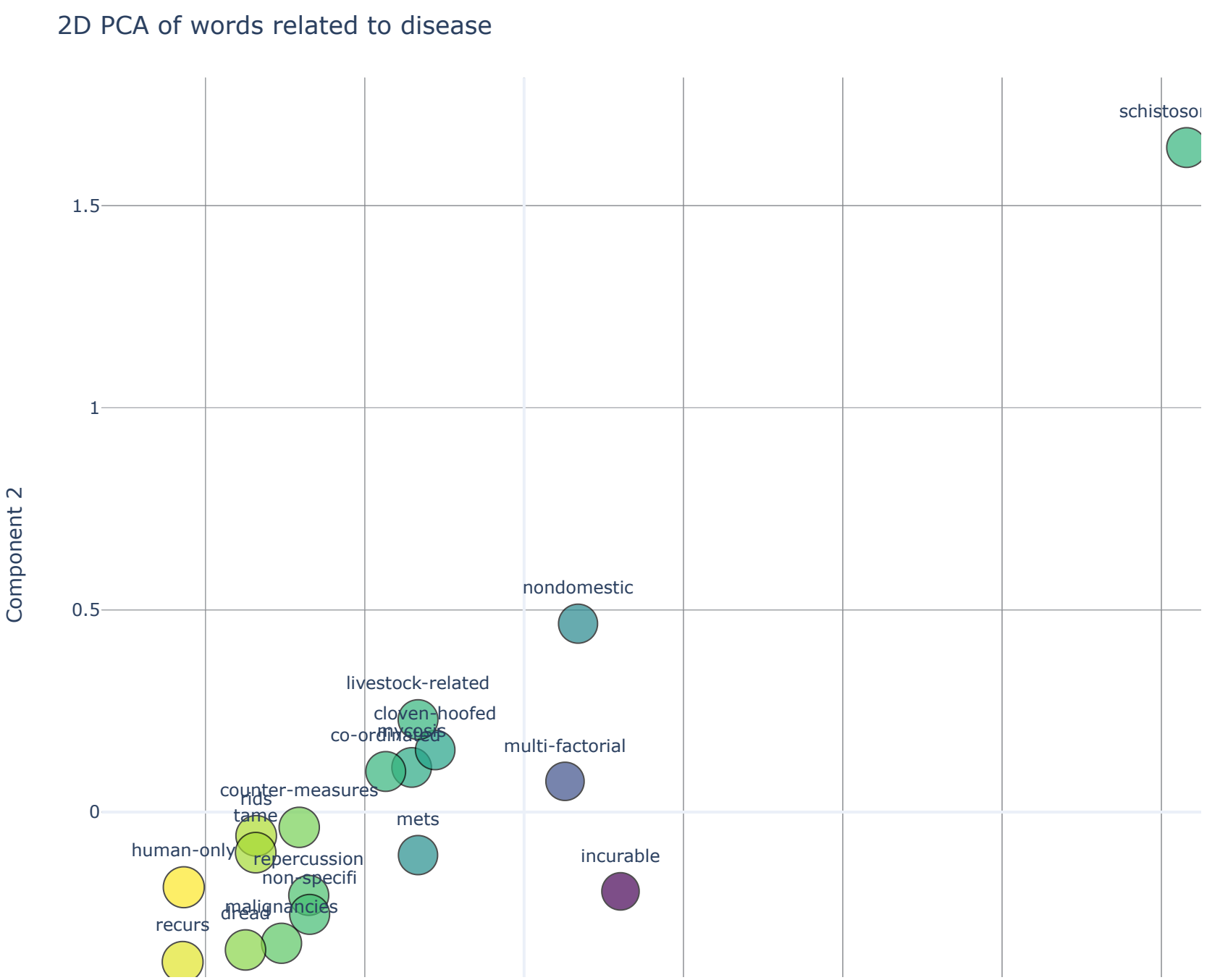
Ailment disease is a particular abnormal condition that negatively affects the structure or function of all or part of any living organism

Lets find the cure now

Second-order word similarities Now, we will look at the words similar to the words found above (second order similarity) to hopefully, find potential cures for COVID-19.

In [106]:
```
# 2D PCA of words related to antiviral

words = [word for word in df_similar_words['word'].tolist()]
X = model[words]
result = pca.fit_transform(X)
df = pd.DataFrame(result, columns=["Component 1", "Component 2"])
df["Word"] = df_similar_words['word']
word_emb = df[["Component 1", "Component 2"]].loc[0]
df["Distance"] = np.sqrt((df["Component 1"] - word_emb[0])**2 + (df["Component 2"] - word_emb[1])**2)
fig = px.scatter(df[2:], x="Component 1", y="Component 2", text="Word", color="Distance", color_continuous_sc
ale="viridis",size="Distance")
fig.update_traces(textposition='top center')
fig.layout.xaxis.autorange = True
fig.data[0].marker.line.width = 1
fig.data[0].marker.line.color = 'rgb(0, 0, 0)'
fig.update_layout(height=800, title_text="2D PCA of words related to {}".format(keyword), template="plotly_wh
ite", paper_bgcolor="#f0f0f0")
fig.show()
```


2D PCA of words related to disease
Component 2
1.5
1
0.5
0
schistoso
nondomestic
livestock-related
cloven-hoofed
co-ordinated
multi-factorial
counter-measures
rids
tame
mets
human-only
repercussion
incurable
non-specifi
malignancies
dread
recurs

We have plotted the 2D PCA of the words most similar to antiviral above. We can see a lot of different types of antivirals and other drugs in the plot, such as "saracatinib", an anti-malarial and anti-HIV drug. The list also includes "antiparasitic", "ant-HBV", and "anti-EV71".

In [107]:
```
# Lets see the most top 20 similar words for my keyword 'antiviral'
keyword = 'antiviral'
similar_words = model.wv.most_similar(keyword, topn=20)
df_similar_words = pd.DataFrame(similar_words, columns = ['word', 'dist'])
df_similar_words
```

Out[107]:

	word	dist
0	anti-viral	0.797267
1	anti-coronavirus	0.705209
2	anti-hbv	0.704857
3	broad-spectrum	0.704462
4	anti-hsv	0.698696
5	chemotherapeutics	0.694655
6	anti-ev71	0.694023
7	anti-pathogenic	0.692182
8	host-targeting	0.689157
9	host-targeted	0.686824
10	ntz	0.683107
11	anti-hcv	0.683041
12	antineoplastic	0.682346
13	anticoronaviral	0.681896
14	anti-virus	0.681344
15	anti-zika	0.681271
16	antagonists	0.681149
17	hydroxychloroquine	0.680798
18	antileishmanial	0.680618
19	anti-iav	0.680596

Second-order word similarities

In [108]:
```
keyword = 'gemcitabine'
similar_words = model.wv.most_similar(keyword, topn=20)
df_similar_words = pd.DataFrame(similar_words, columns = ['word', 'dist'])
words = [word for word in df_similar_words['word'].tolist()]
X = model[words]
result = pca.fit_transform(X)
df = pd.DataFrame(result, columns=["Component 1", "Component 2"])
df["Word"] = df_similar_words['word']
word_emb = df[["Component 1", "Component 2"]].loc[0]
df["Distance"] = np.sqrt((df["Component 1"] - word_emb[0])**2 + (df["Component 2"] - word_emb[1])**2)
fig = px.scatter(df[2:], x="Component 1", y="Component 2", text="Word", color="Distance", color_continuous_sc
ale="viridis",size="Distance")
fig.update_traces(textposition='top center')
fig.layout.xaxis.autorange = True
fig.data[0].marker.line.width = 1
fig.data[0].marker.line.color = 'rgb(0, 0, 0)'
fig.update_layout(height=800, title_text="2D PCA of words related to {}".format(keyword), template="plotly_wh
ite", paper_bgcolor="#f0f0f0")
fig.show()
```


2D PCA of words related to gemcitabine
Component 2
0.6
0.4
0.2
0
−0.2
−0.4
rg3
anti-parasitic
anti-aging
chemopreventive
antiinflammation
antitumour
anti-diabetic
synergic
tet
antineoplastic
immunosuppressant
saraca
anti-tubercular
xav939
antitubercular
antiprotozoal
t-705
ce-cyd

In [109]:
```
keyword = 'daas'
similar_words = model.wv.most_similar(keyword, topn=20)
df_similar_words = pd.DataFrame(similar_words, columns = ['word', 'dist'])
words = [word for word in df_similar_words['word'].tolist()]
X = model[words]
result = pca.fit_transform(X)
df = pd.DataFrame(result, columns=["Component 1", "Component 2"])
df["Word"] = df_similar_words['word']
word_emb = df[["Component 1", "Component 2"]].loc[0]
df["Distance"] = np.sqrt((df["Component 1"] - word_emb[0])**2 + (df["Component 2"] - word_emb[1])**2)
fig = px.scatter(df[2:], x="Component 1", y="Component 2", text="Word", color="Distance", color_continuous_sc
ale="viridis",size="Distance")
fig.update_traces(textposition='top center')
fig.layout.xaxis.autorange = True
fig.data[0].marker.line.width = 1
fig.data[0].marker.line.color = 'rgb(0, 0, 0)'
fig.update_layout(height=800, title_text="2D PCA of words related to {}".format(keyword), template="plotly_wh
ite", paper_bgcolor="#f0f0f0")
fig.show()
```


2D PCA of words related to daas
Component 2
0.8
0.6
0.4
0.2
0
−0.2
−0.4
resistance-associated
acv
foscarnet
lamivu
anti-hsv
antiherpetic
antimycobacterial
interferon-based
antileishmanial
abreva
hta
hostdirected
anti-mammarenavirus
host-targeting
exon-skipping
drug-resistance
anti-malarial
host-targeted

We can see some amazing patterns in the plots above. We see certain drugs and chemicals that keep repeating, including "anti-malarial", "hydroxychloroquine", and "doxorubicin". It is amazing that these drugs have actually been successfully applied on COVID-19 patients across the world. There are cases of anti-malarial drugs working for COVID-19!

Takeaways

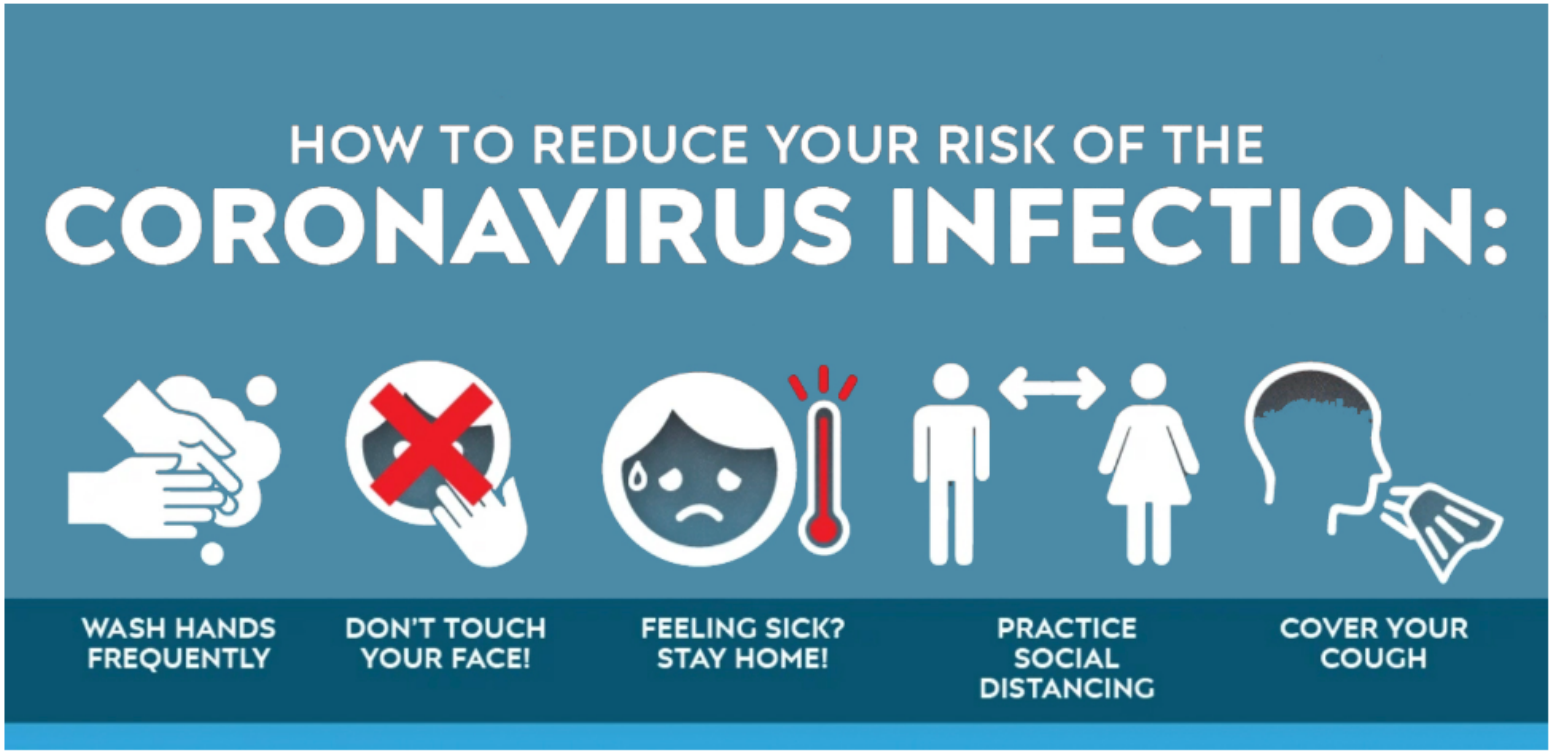
Several antimalarial drugs such as hydroxychloroquine might be potential drugs to cure COVID-19. Antimalarial drugs have been successfully tested on COVID-19 patients in certain countries.

The best ways to control the virus is mass testing, partial or complete lockdown, and use of technology (good examples are China and South Korea).

Conclusion

In [110]:
```
from IPython.display import Image
Image(filename = 'D:/GSUCoursework/BigDataExp/Datasets/Pictures/ReduceRiskCovid19.PNG', width = 800, height = 600)
```

Out[110]:



To avoid the critical situation people are suggested to do following things

Avoid contact with people who are sick.

Avoid touching your eyes, nose, and mouth.

Stay home when you are sick.

Cover your cough or sneeze with a tissue, then throw the tissue in the trash.

Clean and disinfect frequently touched objects and surfaces using a regular household

Wash your hands often with soap and water, especially after going to the bathroom; before eating; and after blowing your nose, coughing, or sneezing. If soap and water are not readily available, use an alcohol-based hand sanitizer.