In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import ticker
import pycountry_convert as pc
import folium
import branca
from datetime import datetime, timedelta,date
from scipy.interpolate import make_interp_spline, BSpline
import plotly.express as px
import json, requests
import calmap

out = ""#+"output/"
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
#Resources directly obtained from Johns Hopkins github

# Retriving Dataset  master data  These two are specific to the city
df_confirmed = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_confirmed_global.csv')
df_deaths = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_deaths_global.csv')

# Depricated  webdata   specific to the country
# df_recovered = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_19-covid-Recovered.csv')
df_covid19 = pd.read_csv("https://raw.githubusercontent.com/CSSEGISandData/COVID-19/web-data/data/cases_country.csv")

#Data is a bit strange. The time is vertical, to be processed. It seems to be one more data per day.
df_table = pd.read_csv("https://raw.githubusercontent.com/CSSEGISandData/COVID-19/web-data/data/cases_time.csv",parse_dates=['Last_Update'])
```

In [3]:

```
df_confirmed = df_confirmed.rename(columns={"Province/State":"state","Country/Region": "country"})
df_deaths = df_deaths.rename(columns={"Province/State":"state","Country/Region": "country"})
df_covid19 = df_covid19.rename(columns={"Country_Region": "country"})
df_covid19["Active"] = df_covid19["Confirmed"]-df_covid19["Recovered"]-df_covid19["Deaths"]
```

In [4]:

```
# df
#             max_speed  shield
# cobra               1       2
# viper               4       5
# sidewinder          7       8

#  df.loc['cobra':'viper', 'max_speed']
#     cobra    1
#     viper    4


# Changing the conuntry names as required by pycountry_convert Lib
df_confirmed.loc[df_confirmed['country'] == "US", "country"] = "USA"
df_deaths.loc[df_deaths['country'] == "US", "country"] = "USA"
df_covid19.loc[df_covid19['country'] == "US", "country"] = "USA"
df_table.loc[df_table['Country_Region'] == "US", "Country_Region"] = "USA"
# df_recovered.loc[df_recovered['country'] == "US", "country"] = "USA"


df_confirmed.loc[df_confirmed['country'] == 'Korea, South', "country"] = 'South Korea'
df_deaths.loc[df_deaths['country'] == 'Korea, South', "country"] = 'South Korea'
df_covid19.loc[df_covid19['country'] == "Korea, South", "country"] = "South Korea"
df_table.loc[df_table['Country_Region'] == "Korea, South", "Country_Region"] = "South Korea"
# df_recovered.loc[df_recovered['country'] == 'Korea, South', "country"] = 'South Korea'

df_confirmed.loc[df_confirmed['country'] == 'Taiwan*', "country"] = 'Taiwan'
df_deaths.loc[df_deaths['country'] == 'Taiwan*', "country"] = 'Taiwan'
df_covid19.loc[df_covid19['country'] == "Taiwan*", "country"] = "Taiwan"
df_table.loc[df_table['Country_Region'] == "Taiwan*", "Country_Region"] = "Taiwan"
# df_recovered.loc[df_recovered['country'] == 'Taiwan*', "country"] = 'Taiwan'

df_confirmed.loc[df_confirmed['country'] == 'Congo (Kinshasa)', "country"] = 'Democratic Republic of the Congo'
df_deaths.loc[df_deaths['country'] == 'Congo (Kinshasa)', "country"] = 'Democratic Republic of the Congo'
df_covid19.loc[df_covid19['country'] == "Congo (Kinshasa)", "country"] = "Democratic Republic of the Congo"
df_table.loc[df_table['Country_Region'] == "Congo (Kinshasa)", "Country_Region"] = "Democratic Republic of the Congo"
# df_recovered.loc[df_recovered['country'] == 'Congo (Kinshasa)', "country"] = 'Democratic Republic of the Congo'

df_confirmed.loc[df_confirmed['country'] == "Cote d'Ivoire", "country"] = "Côte d'Ivoire"
df_deaths.loc[df_deaths['country'] == "Cote d'Ivoire", "country"] = "Côte d'Ivoire"
df_covid19.loc[df_covid19['country'] == "Cote d'Ivoire", "country"] = "Côte d'Ivoire"
df_table.loc[df_table['Country_Region'] == "Cote d'Ivoire", "Country_Region"] = "Côte d'Ivoire"
# df_recovered.loc[df_recovered['country'] == "Cote d'Ivoire", "country"] = "Côte d'Ivoire"

df_confirmed.loc[df_confirmed['country'] == "Reunion", "country"] = "Réunion"
df_deaths.loc[df_deaths['country'] == "Reunion", "country"] = "Réunion"
df_covid19.loc[df_covid19['country'] == "Reunion", "country"] = "Réunion"
df_table.loc[df_table['Country_Region'] == "Reunion", "Country_Region"] = "Réunion"
# df_recovered.loc[df_recovered['country'] == "Reunion", "country"] = "Réunion"

df_confirmed.loc[df_confirmed['country'] == 'Congo (Brazzaville)', "country"] = 'Republic of the Congo'
df_deaths.loc[df_deaths['country'] == 'Congo (Brazzaville)', "country"] = 'Republic of the Congo'
df_covid19.loc[df_covid19['country'] == "Congo (Brazzaville)", "country"] = "Republic of the Congo"
df_table.loc[df_table['Country_Region'] == "Congo (Brazzaville)", "Country_Region"] = "Republic of the Congo"
# df_recovered.loc[df_recovered['country'] == 'Congo (Brazzaville)', "country"] = 'Republic of the Congo'

df_confirmed.loc[df_confirmed['country'] == 'Bahamas, The', "country"] = 'Bahamas'
df_deaths.loc[df_deaths['country'] == 'Bahamas, The', "country"] = 'Bahamas'
df_covid19.loc[df_covid19['country'] == "Bahamas, The", "country"] = "Bahamas"
df_table.loc[df_table['Country_Region'] == "Bahamas, The", "Country_Region"] = "Bahamas"
# df_recovered.loc[df_recovered['country'] == 'Bahamas, The', "country"] = 'Bahamas'

df_confirmed.loc[df_confirmed['country'] == 'Gambia, The', "country"] = 'Gambia'
df_deaths.loc[df_deaths['country'] == 'Gambia, The', "country"] = 'Gambia'
df_covid19.loc[df_covid19['country'] == "Gambia, The", "country"] = "Gambia"
df_table.loc[df_table['Country_Region'] == "Gambia", "Country_Region"] = "Gambia"
# df_recovered.loc[df_recovered['country'] == 'Gambia, The', "country"] = 'Gambia'

# getting all countries
#Because there is a special usa data, so usa data in confirmed usa is used as a whole.
countries = np.asarray(df_confirmed["country"])
countries1 = np.asarray(df_covid19["country"])
# Continent_code to Continent_names
continents = {
    'NA': 'North America',
    'SA': 'South America',
    'AS': 'Asia',
    'OC': 'Australia',
    'AF': 'Africa',
    'EU' : 'Europe',
    'na' : 'Others'
}
```

In [5]:

```
#Add the corresponding mainland information to each data, using the ISO 3166-1 alpha-2 code
# Defininng Function for getting continent code for country.
#ISO 3166-1 alpha-2 codes are two-letter country codes defined in ISO 3166-1,
# part of the ISO 3166 standard[1] published by the International Organization for Standardization (ISO),
# to represent countries, dependent territories, and special areas of geographical interest.
# They are the most widely used of the country codes published by ISO (the others being alpha-3 and numeric),
# and are used most prominently for the Internet's country code top-level domains (with a few exceptions).
# They are also used as country identifiers extending the postal code when appropriate within the international postal system for paper mail,
# and has replaced the previous one consisting one-letter codes. They were first included as part of the ISO 3166 standard in its first editio
n in 1974.
def country_to_continent_code(country):
    try:
        return pc.country_alpha2_to_continent_code(pc.country_name_to_country_alpha2(country))
    except :
        return 'na'

#Collecting Continent Information
df_confirmed.insert(2,"continent", [continents[country_to_continent_code(country)] for country in countries[:]])
df_deaths.insert(2,"continent",  [continents[country_to_continent_code(country)] for country in countries[:]])
df_covid19.insert(1,"continent",  [continents[country_to_continent_code(country)] for country in countries1[:]])
df_table.insert(1,"continent",  [continents[country_to_continent_code(country)] for country in df_table["Country_Region"].values])
```

In [6]:

```
df_confirmed = df_confirmed.replace(np.nan, '', regex=True)
df_deaths = df_deaths.replace(np.nan, '', regex=True)
```

In [7]:

```
df_countries_cases = df_covid19.copy().drop(['Lat','Long_','continent','Last_Update'],axis =1)
df_countries_cases.index=df_countries_cases["country"]
df_countries_cases=df_countries_cases.drop(['country'],axis=1)

df_continents_cases = df_covid19.copy().drop(['Lat','Long_','country','Last_Update'],axis =1)
df_continents_cases = df_continents_cases.groupby(["continent"]).sum()

df_countries_cases.fillna(0,inplace=True)
df_continents_cases.fillna(0,inplace=True)
```

In [8]:

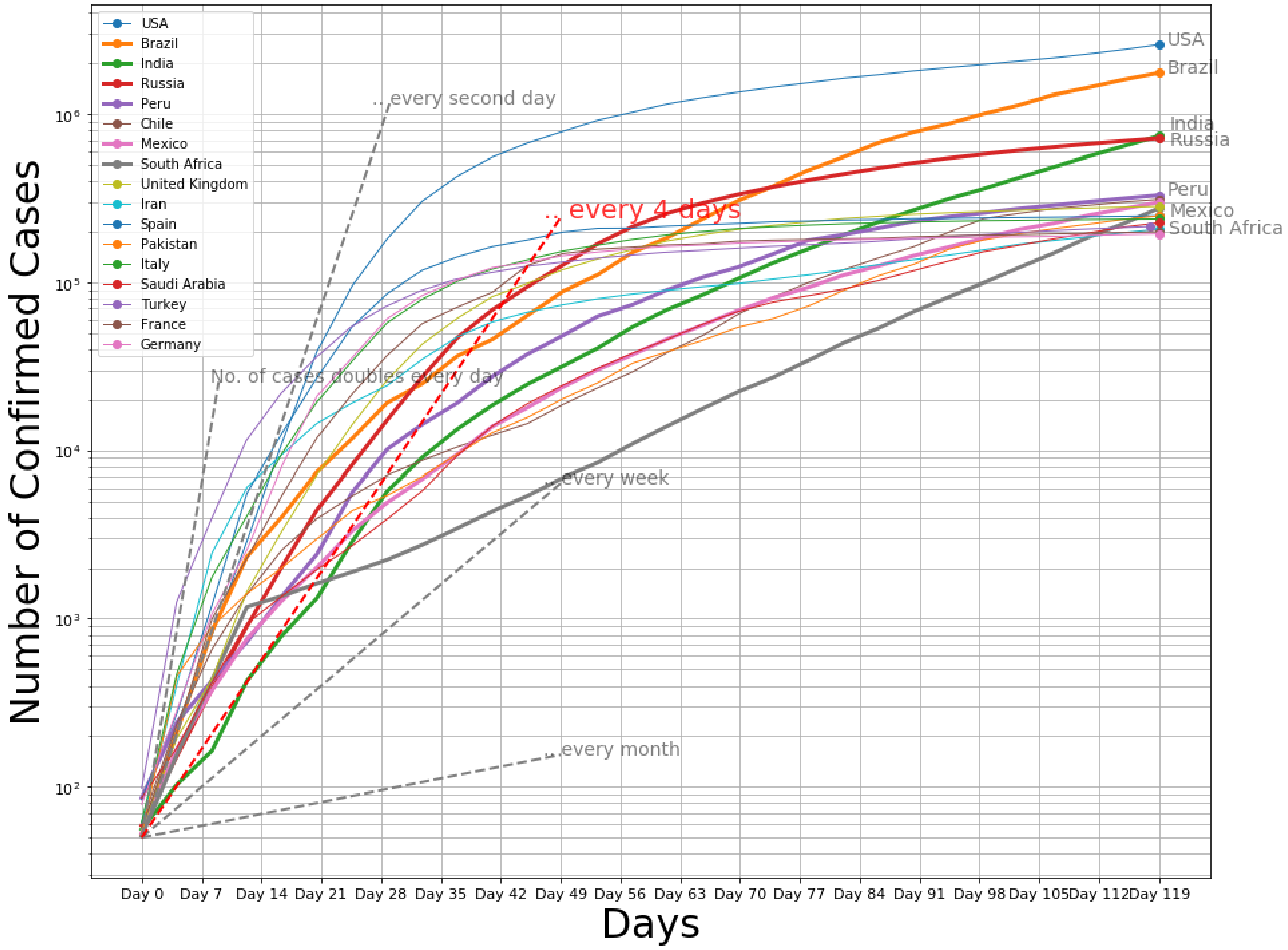
```
df_confirmed_glo=df_confirmed.drop(['continent','state',"Lat","Long"],axis=1)
```

In [9]:

```
df_confirmed_glo=df_confirmed_glo.groupby(["country"]).sum()
```

In [10]:

```
temp=df_confirmed_glo.sort_values(df_confirmed_glo.columns[-1],ascending=False).iloc[0:17].replace(np.nan,0)
List=["Brazil","Russia","India","Peru","South Africa","Mexico"]
List2=["India"]
threshold = 50
f = plt.figure(figsize=(15,12))
ax = f.add_subplot(111)
for i,country in enumerate(temp.index):
    days = 120
    t = temp.loc[temp.index== country].values[0]
    t = t[t>threshold][:days]

    date = np.arange(0,len(t[:days])) #0,1,2,3,4,5
    xnew = np.linspace(date.min(), date.max(), 30)
    spl = make_interp_spline(date, t, k=1)  # type: BSpline
    power_smooth = spl(xnew)
    if country in List:
        plt.plot(xnew,power_smooth,'-o',label = country,linewidth =3, markevery=[-1])
        if country == "India":
            plt.annotate(country,(xnew[-1]+1.2,power_smooth[-1]+70000),xycoords="data",fontsize=14,alpha = 0.5)
        elif country == "South Africa":
            plt.annotate(country,(xnew[-1]+1.2,power_smooth[-1]-80000),xycoords="data",fontsize=14,alpha = 0.5)
        elif country == "Russia":
            plt.annotate(country,(xnew[-1]+1.2,power_smooth[-1]-70000),xycoords="data",fontsize=14,alpha = 0.5)
        elif country == "Mexico":
            plt.annotate(country,(xnew[-1]+1.2,power_smooth[-1]-50000),xycoords="data",fontsize=14,alpha = 0.5)
        else:
            plt.annotate(country,(xnew[-1]+1,power_smooth[-1]),xycoords="data",fontsize=14,alpha = 0.5)
    else:
        if country == "USA":
            plt.annotate(country,(xnew[-1]+1,power_smooth[-1]),xycoords="data",fontsize=14,alpha = 0.5)
            plt.plot(xnew,power_smooth,'-o',label = country,linewidth =1, markevery=[-1])
        else:
            plt.plot(xnew,power_smooth,'-o',label = country,linewidth =1, markevery=[-1])
plt.tick_params(labelsize = 11)
plt.xticks(np.arange(0,days,7),[ "Day "+str(i) for i in range(days)][::7])

# Reference lines
x = np.arange(0,10)
y = 2**(x+np.log2(threshold))
plt.plot(x,y,"--",linewidth =2,color = "gray")
plt.annotate("No. of cases doubles every day",(x[-2],y[-1]),xycoords="data",fontsize=14,alpha = 0.5)

x = np.arange(0,int(days-90))
y = 2**(x/2+np.log2(threshold))
plt.plot(x,y,"--",linewidth =2,color = "gray")
plt.annotate(".. every second day",(x[-3],y[-1]),xycoords="data",fontsize=14,alpha = 0.5)

x = np.arange(0,int(days-70))
y = 2**(x/7+np.log2(threshold))
plt.plot(x,y,"--",linewidth =2,color = "gray")
plt.annotate(".. every week",(x[-3],y[-1]),xycoords="data",fontsize=14,alpha = 0.5)

x = np.arange(0,int(days-70))
y = 2**(x/30+np.log2(threshold))
plt.plot(x,y,"--",linewidth =2,color = "gray")
plt.annotate(".. every month",(x[-3],y[-1]),xycoords="data",fontsize=14,alpha = 0.5)


# India is following trend similar to doulbe the cases in 4 days but it may increase the rate
x = np.arange(0,int(days-70))
y = 2**(x/4+np.log2(threshold))
plt.plot(x,y,"--",linewidth =2,color = "Red")
plt.annotate(".. every 4 days",(x[-3],y[-1]),color="Red",xycoords="data",fontsize=20,alpha = 0.8)

# plot Params
plt.xlabel("Days",fontsize=30)
plt.ylabel("Number of Confirmed Cases",fontsize=30)
# plt.title("Trend Comparison of Different Country (confirmed) ",fontsize=22)
plt.legend(loc = "upper left")
plt.yscale("log")
plt.rcParams["font.family"] = "Times New Roman"
plt.grid(which="both")
plt.savefig(out+'C:/Users/Administrator/Desktop/Trend Comparison with country (confirmed).png',dpi=300)
plt.show()
```

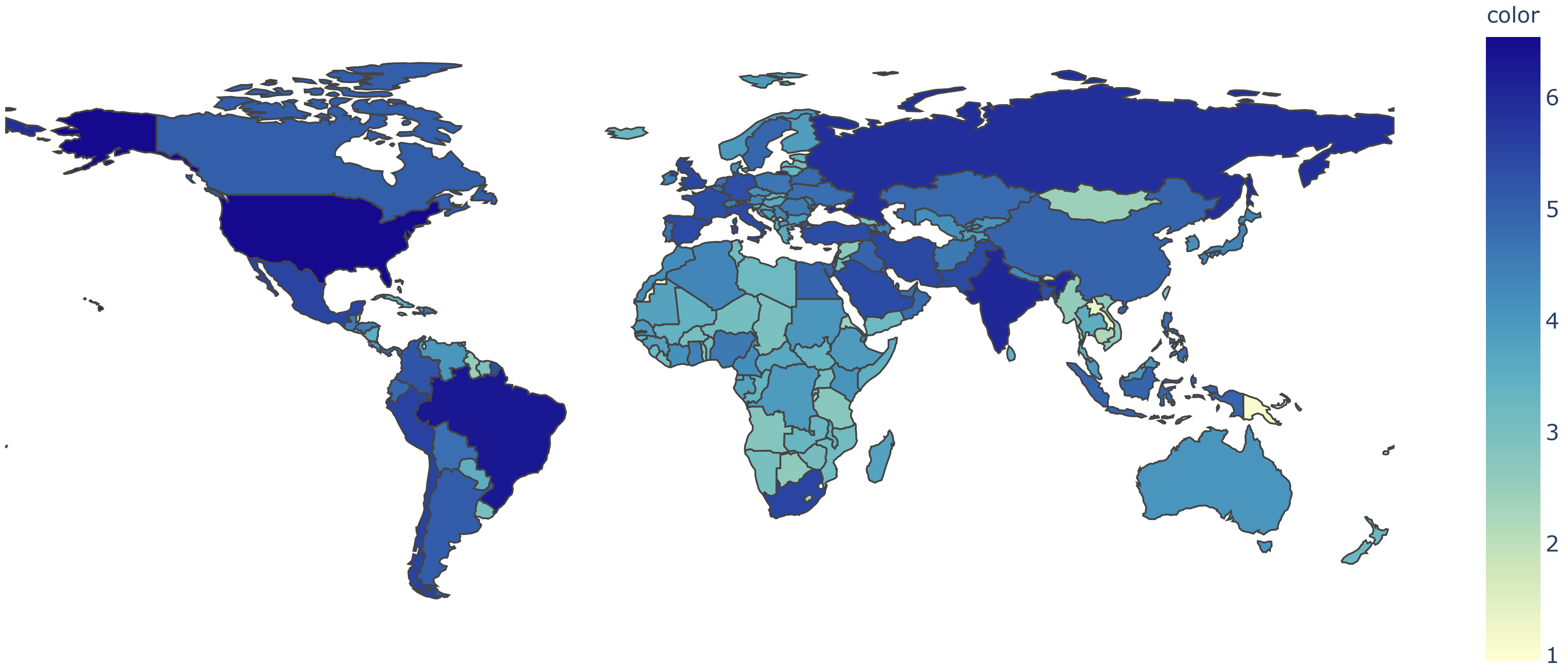


In [11]:
```
import plotly
```

In [12]:

```
# hover_nameS pecify the column name.
#Bold the value in the column directly above the content of the hover prompt;

# hover_data: Specify a list of column names.
#The values of all columns are displayed in the content of the hover prompt, below the x/y value.
#Only one data is displayed when the specified column is repeated with x/y;

temp_df = pd.DataFrame(df_countries_cases['Confirmed'])   #
temp_df = temp_df.reset_index()
fig = px.choropleth(temp_df, locations="country",
                    color=np.log10(temp_df["Confirmed"]), # lifeExp is a column of gapminder
                    hover_name="country", # column to add to hover information
                    hover_data=["Confirmed"],
                    color_continuous_scale=[[0,  'rgb(255, 254, 209)'], [0.143, 'rgb(207, 231, 185)'], [0.286, 'rgb(146, 203, 188)'], [0.4286,
'rgb(101, 180, 194)'], [0.5714, 'rgb(69, 144, 188)'],[1.0, 'rgb(21, 9, 141)']],locationmode="country names")
fig.update_geos(fitbounds="locations", visible=False)  #other places
fig.update_layout(title_text="Confirmed Cases Heat Map (Log Scale)")
# fig.update_coloraxes(colorbar_title="Confirmed Cases(Log Scale)",colorscale="Blues")
# plotly.io.orca.config.save()
# fig.to_image("Global Heat Map confirmed.png")
fig.show()
```

Confirmed Cases Heat Map (Log Scale)



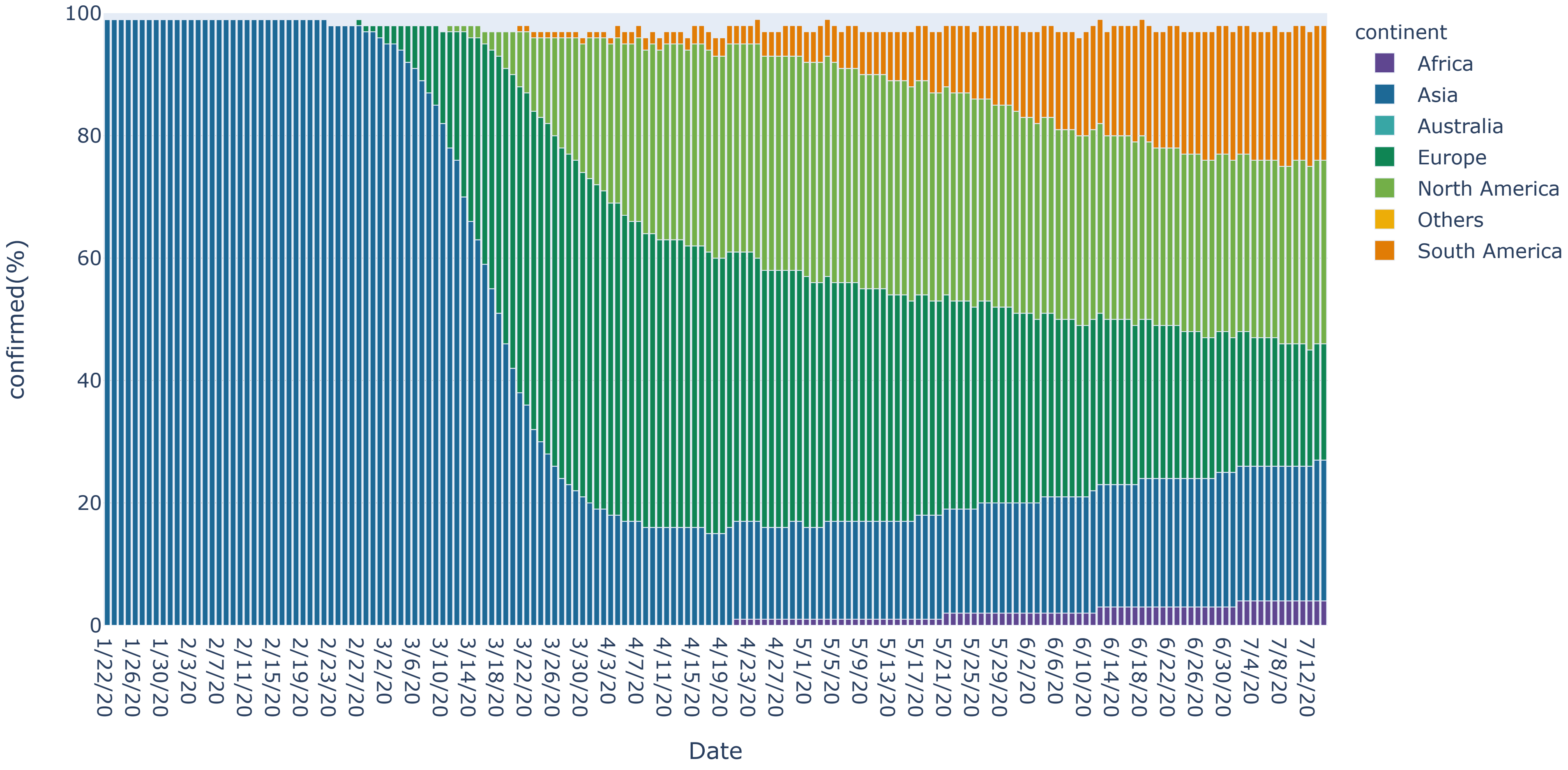
In [13]:

```
df_continents=df_confirmed.groupby(["continent"]).sum()
df_continents=df_continents.drop(['Lat','Long'],axis=1)
```

In [14]:

```
dfc=df_continents.copy()

for index,row in dfc.iteritems():
    sum=0
    for i in range(7):
        sum=sum+row[i]
    for i in range(7):
        row[i]=round(row[i]/sum,3)*100

result=pd.DataFrame(columns=('idx','Date','continent','confirmed(%)'))
list2=['Africa','Asia','Australia','Europe','North America','Others','South America']
h=0
for index,row in dfc.iteritems():
    for i in range(7):
        result=result.append(pd.DataFrame({'idx':[h],'Date':[index],'continent':[list2[i]],'confirmed(%)':[row[i]]}),ignore_index=True)
        h=h+1
```

In [15]:
```
import plotly.graph_objects as go


fig = px.bar(result, x='Date', y='confirmed(%)', color='continent',
             range_y=(0, 100),
             color_discrete_sequence=px.colors.qualitative.Prism)




fig.show()
```



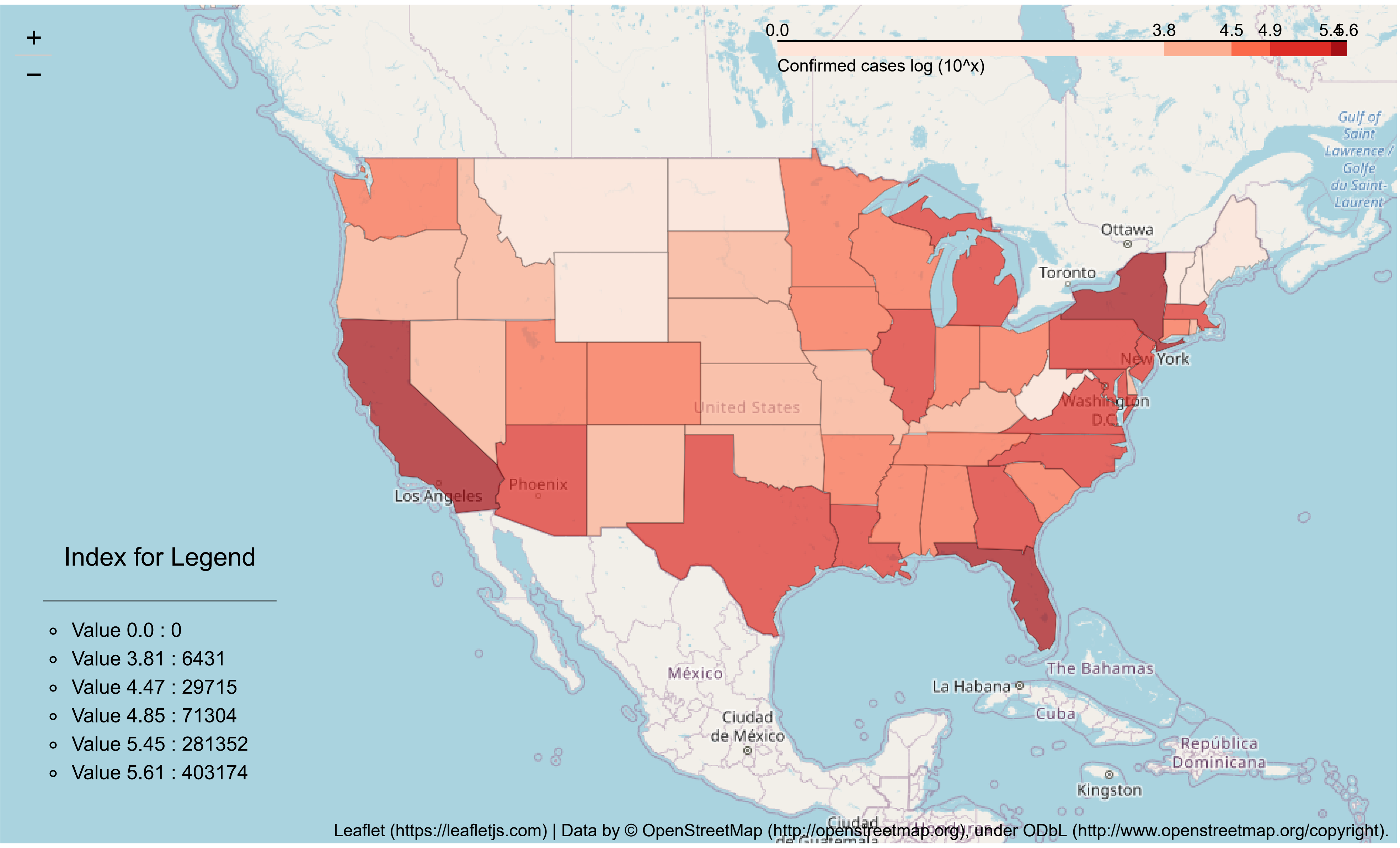
In [16]:
```
date_usa = datetime.strptime(df_confirmed.columns[-1],'%m/%d/%y').strftime("%m-%d-%Y")
df_temp = pd.read_csv("https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_daily_reports/"+date_
usa+".csv")
df_usa = df_temp.loc[df_temp["Country_Region"]== "US"]
df_usa = df_usa.rename(columns={"Admin2":"County"})
```

In [17]:
```
state_geo = requests.get('https://raw.githubusercontent.com/python-visualization/folium/master/examples/data/us-states.json').json()
county_geo = requests.get('https://raw.githubusercontent.com/python-visualization/folium/master/examples/data/us_counties_20m_topo.json').json
()
# county_geo
```

In [18]:

```
data_temp = df_usa.groupby(["Province_State"]).sum().reset_index().drop(["Lat","Long_"],axis=1)
data_temp["Confirmed_log"] = np.log10(data_temp["Confirmed"]+1)
bins = list(data_temp['Confirmed_log'].quantile([0, 0.25, 0.5, 0.75,0.95 ,1]))
m = folium.Map(location=[37, -102], zoom_start=4,max_zoom=6,min_zoom=3)

# Add the color for the chloropleth:
folium.Choropleth(
    geo_data=state_geo,
    name='choropleth',
    data = data_temp,
    columns=['Province_State', 'Confirmed_log'],
    key_on='feature.properties.name',
    fill_color='Reds',
    fill_opacity=0.7,
    line_opacity=0.2,
    bins = bins,
    reset=True,
    legend_name='Confirmed cases log (10^x)'
).add_to(m)
folium.LayerControl().add_to(m)

legend_html = "<div style='padding:10px;background-color:rgba(255,255,255,0.5);position:fixed;bottom:20px;left:20px;z-index:1000'>"
legend_html += "<div style='width:100%;text-align:center;'><h4>Index for Legend</h4></div><hr style='border-top-color: rgba(25,25,25,0.5);'>"
legend_html += "<ul style='margin:0;padding:0;color: #555;list-style-type:circle;align-item:left;padding-left:20px;padding-right:20px'>"
for i in bins:
    legend_html += "<li style='margin:0;padding:0;line-height: 0;'>Value "+str(np.round(i,2))+" : "+str(int(10**i)-1)+"</li><br>"
legend_html += "</ul></div>"
m.get_root().html.add_child(folium.Element(legend_html))
m
```

Out[18]:


+
−
0.0
3.8
4.5
4.9
5.4
5.6
Confirmed cases log (10^x)
Gulf of Saint Lawrence / Golfe du Saint-Laurent
Ottawa
Toronto
New York
Washington D.C.
United States
Phoenix
Los Angeles
México
Ciudad de México
La Habana
The Bahamas
Cuba
República Dominicana
Kingston
Ciudad de Guatemala
Leaflet (https://leafletjs.com) | Data by © OpenStreetMap (http://openstreetmap.org), under ODbL (http://www.openstreetmap.org/copyright).

Index for Legend

- Value 0.0 : 0
- Value 3.81 : 6431
- Value 4.47 : 29715
- Value 4.85 : 71304
- Value 5.45 : 281352
- Value 5.61 : 403174

In [19]:

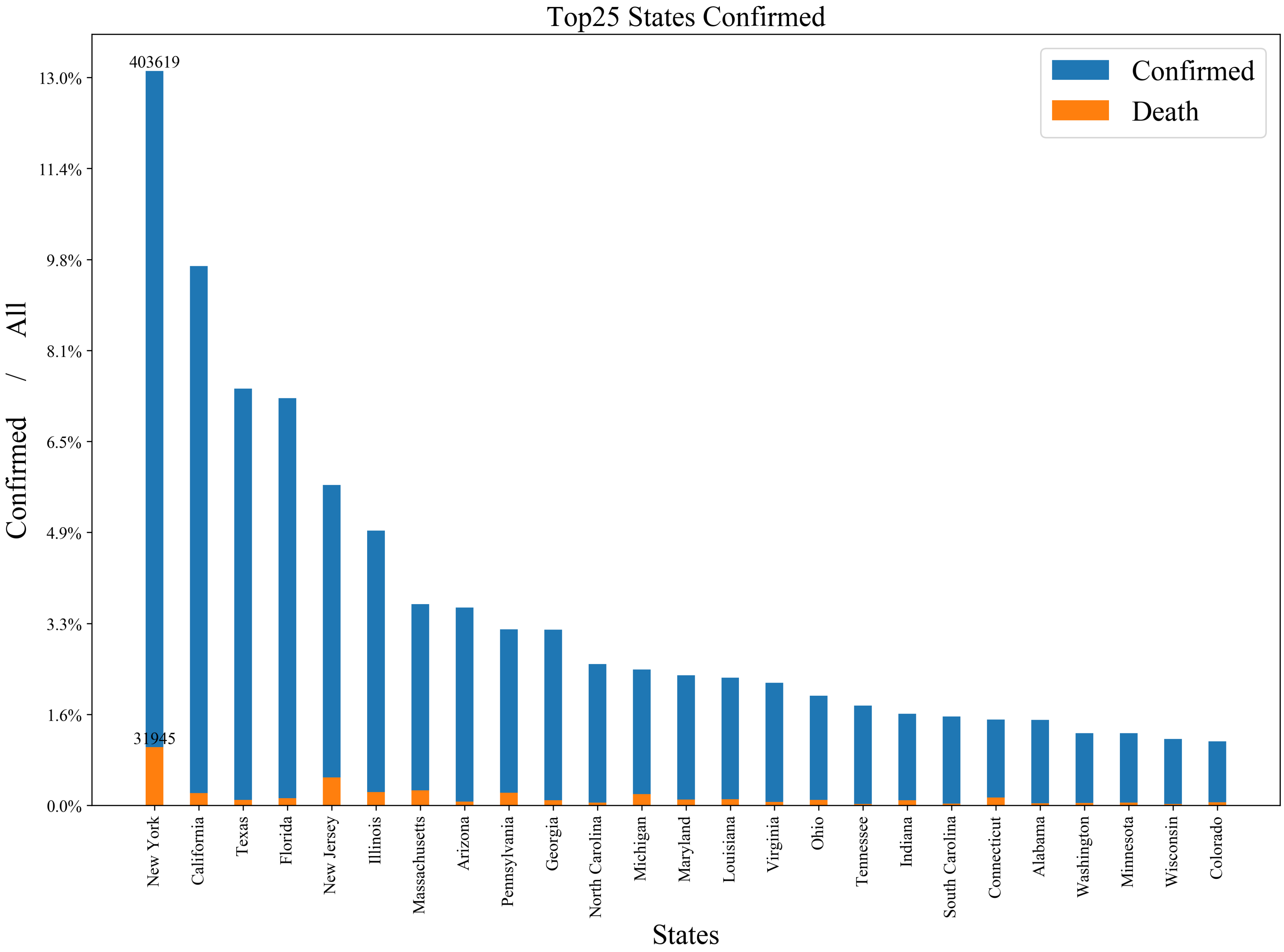
```
cnf, dth, rec, act = '#393e46', '#ff2e63', '#21bf73', '#fe9801'
data=pd.read_csv("us-states.csv")
data_1=data.loc[data['date'] == "2020-07-08"]
```

In [20]:

```
sorted=data_1.sort_values('cases',ascending=False)
total = sorted['cases'].sum()
test=sorted.head(25)# get top25 confirmed states
test['percentage']=(test.cases/3071571)
```

In [21]:

```
from matplotlib.ticker import PercentFormatter
import matplotlib.patches as mpatches
plt.rcParams['figure.figsize'] = (4, 3)
plt.rcParams['image.interpolation'] = 'nearest'
plt.rcParams['image.cmap'] = 'white'
plt.rcParams['figure.dpi'] = 300
k=plt.figure(figsize=(15,10))
k.set_facecolor('white')
x=np.arange(25)+1


y=np.array(list(test.cases))

y1=np.array(list(test.deaths))


xticks1=list(test.state)


plt.xticks(x,xticks1,size='large',rotation=90)




plt.xlabel('States',fontsize=20.0)

plt.ylabel('Confirmed     /     All',fontsize=20.0)

plt.title('Top25 States Confirmed',fontsize=20.0)





plt.text(x[0], y[0], '403619' , ha='center', va= 'bottom',fontsize=12.0)

plt.text(x[0], y1[0], '31945', ha='center', va= 'bottom',fontsize=12.0)

plt.bar(x, y, width = 0.4,label='Confirmed')
plt.bar(x, y1,width = 0.4, label='Death')

yaxis=plt.gca().yaxis.set_major_formatter(PercentFormatter(3071571))
plt.yticks(size='large')
plt.rcParams["font.family"] = "Times New Roman"
plt.legend(fontsize=20.0)
plt.savefig("2020final.png")



plt.show()
```



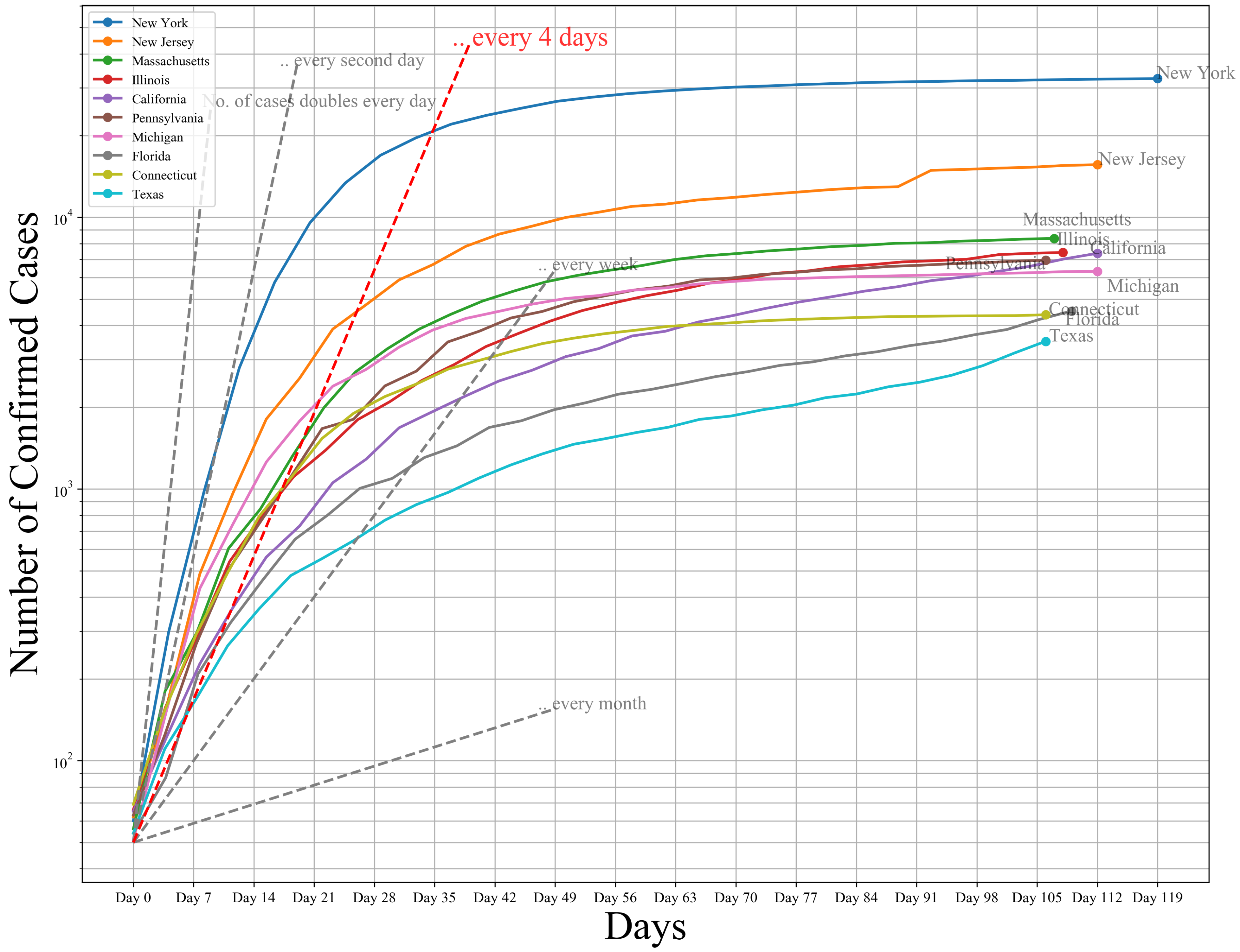
In [22]:
```
df_confirmed_us=pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_confirmed_US.csv')
df_deaths_us=pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_deaths_US.csv')
```

In [23]:
```
df_confirmed_us=df_confirmed_us.drop(["Admin2","UID","iso2","iso3","code3","FIPS","Country_Region","Lat","Long_"],axis=1)
df_deaths_us=df_deaths_us.drop(["Admin2","UID","iso2","iso3","code3","FIPS","Country_Region","Lat","Long_"],axis=1)
df_deaths_us=df_deaths_us.drop(["Population"],axis=1)
```

In [24]:
```
df_confirmed_us_province=df_confirmed_us.groupby(["Province_State"]).sum()
df_deaths_us_province=df_deaths_us.groupby(["Province_State"]).sum()
```

In [25]:

```
temp=df_deaths_us_province.sort_values(df_deaths_us_province.columns[-1],ascending=False).head(10).replace(np.nan,0)

threshold = 50
f = plt.figure(figsize=(15,12))
ax = f.add_subplot(111)
for i,country in enumerate(temp.index):
    days = 120
    t = temp.loc[temp.index== country].values[0]
    t = t[t>threshold][:days]

    date = np.arange(0,len(t[:days])) #0,1,2,3,4,5
    xnew = np.linspace(date.min(), date.max(), 30)
    spl = make_interp_spline(date, t, k=1)  # type: BSpline
    power_smooth = spl(xnew)
    plt.plot(xnew,power_smooth,'-o',label = country,linewidth =2, markevery=[-1])
    if country == "Massachusetts":
        plt.annotate(country,(xnew[-2],power_smooth[-1]+1000),xycoords="data",fontsize=14,alpha = 0.5)
    elif country == "California":
        plt.annotate(country,(xnew[-2]+3,power_smooth[-1]),xycoords="data",fontsize=14,alpha = 0.5)
    elif country == "Michigan":
        plt.annotate(country,(xnew[-2]+5,power_smooth[-1]-1000),xycoords="data",fontsize=14,alpha = 0.5)
    elif country == "Illinois":
        plt.annotate(country,(xnew[-2]+3,power_smooth[-1]+500),xycoords="data",fontsize=14,alpha = 0.5)
    elif country == "Pennsylvania":
        plt.annotate(country,(xnew[-2]-8,power_smooth[-1]-500),xycoords="data",fontsize=14,alpha = 0.5)
    elif country == "Florida":
        plt.annotate(country,(xnew[-2]+3,power_smooth[-1]-500),xycoords="data",fontsize=14,alpha = 0.5)
    else:
        plt.annotate(country,(xnew[-2]+4,power_smooth[-1]),xycoords="data",fontsize=14,alpha = 0.5)
plt.tick_params(labelsize = 11)
plt.xticks(np.arange(0,days,7),[ "Day "+str(i) for i in range(days)][::7])

# Reference lines
x = np.arange(0,10)
y = 2**(x+np.log2(threshold))
plt.plot(x,y,"--",linewidth =2,color = "gray")
plt.annotate("No. of cases doubles every day",(x[-2],y[-1]),xycoords="data",fontsize=14,alpha = 0.5)

x = np.arange(0,int(days-100))
y = 2**(x/2+np.log2(threshold))
plt.plot(x,y,"--",linewidth =2,color = "gray")
plt.annotate(".. every second day",(x[-3],y[-1]),xycoords="data",fontsize=14,alpha = 0.5)

x = np.arange(0,int(days-70))
y = 2**(x/7+np.log2(threshold))
plt.plot(x,y,"--",linewidth =2,color = "gray")
plt.annotate(".. every week",(x[-3],y[-1]),xycoords="data",fontsize=14,alpha = 0.5)

x = np.arange(0,int(days-70))
y = 2**(x/30+np.log2(threshold))
plt.plot(x,y,"--",linewidth =2,color = "gray")
plt.annotate(".. every month",(x[-3],y[-1]),xycoords="data",fontsize=14,alpha = 0.5)


# India is following trend similar to doulbe the cases in 4 days but it may increase the rate
x = np.arange(0,int(days-80))
y = 2**(x/4+np.log2(threshold))
plt.plot(x,y,"--",linewidth =2,color = "Red")
plt.annotate(".. every 4 days",(x[-3],y[-1]),color="Red",xycoords="data",fontsize=20,alpha = 0.8)

# plot Params
plt.xlabel("Days",fontsize=30)
plt.ylabel("Number of Confirmed Cases",fontsize=30)
# plt.title("Trend Comparison of Different States (confirmed) ",fontsize=22)
plt.legend(loc = "upper left")
plt.yscale("log")
plt.rcParams["font.family"] = "Times New Roman"
plt.grid(which="both")
plt.savefig(out+'C:/Users/Administrator/Desktop/Trend Comparison with states (confirmed).png',dpi=300)
plt.show()
```


New York
New Jersey
Massachusetts
Illinois
California
Pennsylvania
Michigan
Florida
Connecticut
Texas
.. every 4 days
.. every second day
No. of cases doubles every day
.. every week
.. every month
New York
New Jersey
Massachusetts
Illinois
California
Pennsylvania
Michigan
Connecticut
Florida
Texas
Number of Confirmed Cases
10^4
10^3
10^2
Day 0
Day 7
Day 14
Day 21
Day 28
Day 35
Day 42
Day 49
Day 56
Day 63
Day 70
Day 77
Day 84
Day 91
Day 98
Day 105
Day 112
Day 119
Days