Data Collection and Preprocessing

In [1]:

```
import pandas as pd
df = pd.read_csv('covid_19_data.csv',parse_dates=['Last Update
'],index_col=0)
#parsing according to the last update datetime column
#serial number as index
df.rename(columns={'ObservationDate':'Date', 'Country/Region':
'Country'}, inplace=True)
#renaming certain columns
```

In [2]:

```
#used for Map later
df_confirmed = pd.read_csv("time_series_covid_19_confirmed.csv
")
df_confirmed.rename(columns={'Country/Region':'Country'}, inpl
ace=True)
df_confirmed = df_confirmed[["Province/State","Lat","Long","Co
untry"]]
df_temp = df.copy()
df_temp['Country'].replace({'Mainland China': 'China'}, inplac
e=True)
df_latlong = pd.merge(df_temp, df_confirmed, on=["Country", "P
rovince/State"])
```

Exploratory Data Analysis (EDA)

In [3]:

```
print("Rows : ",df.shape[0])
print("\nColumns : ",df.shape[1])
print("\nFeatures : ",df.columns.tolist())
print("\nMissing Values : ",df.isnull().sum().sum())
print("\nUnique values : \n", df.nunique())
```

```
Rows :  10984

Columns :  7

Features :  ['Date', 'Province/State', 'Country',
'Last Update', 'Confirmed', 'Deaths', 'Recovered']

Missing Values :  5132

Unique values :
 Date                71
Province/State      293
Country             215
Last Update        1814
Confirmed          1613
Deaths              352
Recovered           768
dtype: int64
```

In [4]:

```
df.info() #information about the datatypes
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10984 entries, 1 to 10984
Data columns (total 7 columns):
Date              10984 non-null object
Province/State    5852 non-null object
Country           10984 non-null object
Last Update       10984 non-null datetime64[ns]
Confirmed         10984 non-null float64
Deaths            10984 non-null float64
Recovered         10984 non-null float64
dtypes: datetime64[ns](1), float64(3), object(3)
memory usage: 686.5+ KB
```

In [5]:

```
print("Basic Statistics : \n",df.describe()) #metrics
```

```
Basic Statistics :
            Confirmed         Deaths      Recovered
count   10984.000000  10984.000000  10984.000000
mean     1050.407502     43.769938    298.845776
std      6524.120490    426.967500   3051.738866
min         0.000000      0.000000      0.000000
25%         3.000000      0.000000      0.000000
50%        33.000000      0.000000      0.000000
75%       245.000000      2.000000     15.000000
max    110574.000000  13155.000000  63326.000000
```

In [6]:

```
print("Earliest Cases : \n",df.head())
```

```
Earliest Cases :
            Date Province/State         Country
Last Update  Confirmed  \
SNo
1    01/22/2020          Anhui  Mainland China 202
0-01-22 17:00:00        1.0
2    01/22/2020        Beijing  Mainland China 202
0-01-22 17:00:00       14.0
3    01/22/2020      Chongqing  Mainland China 202
0-01-22 17:00:00        6.0
4    01/22/2020         Fujian  Mainland China 202
0-01-22 17:00:00        1.0
5    01/22/2020          Gansu  Mainland China 202
0-01-22 17:00:00        0.0

     Deaths  Recovered
SNo
1       0.0        0.0
2       0.0        0.0
3       0.0        0.0
4       0.0        0.0
5       0.0        0.0
```

In [7]:

```
print("Latest Cases : \n",df.tail())
```

```
Latest Cases :
               Date Province/State         Country
Last Update  \
SNo
10980  04/01/2020        Wyoming              US 2
020-04-01 22:04:58
10981  04/01/2020       Xinjiang  Mainland China 2
020-04-01 22:04:58
10982  04/01/2020          Yukon           Canada 2
020-04-01 22:04:58
10983  04/01/2020         Yunnan  Mainland China 2
020-04-01 22:04:58
10984  04/01/2020       Zhejiang  Mainland China 2
020-04-01 22:04:58

       Confirmed  Deaths  Recovered
SNo
10980      130.0     0.0        0.0
10981       76.0     3.0       73.0
10982        5.0     0.0        0.0
10983      182.0     2.0      172.0
10984     1257.0     1.0     1226.0
```

In [8]:

```
print("\n\t\t Date-wise number of cases in each category\n")
df.groupby('Date').sum()
#total number of Confirmed cases , Deaths and Recovery per day
.
```

```
                Date-wise number of cases in each
category
```

Out[8]:

Date	Confirmed	Deaths	Recovered
01/22/2020	555.0	17.0	28.0
01/23/2020	653.0	18.0	30.0
01/24/2020	941.0	26.0	36.0

01/25/2020	1438.0	42.0	39.0
01/26/2020	2118.0	56.0	52.0
01/27/2020	2927.0	82.0	61.0
01/28/2020	5578.0	131.0	107.0
01/29/2020	6165.0	133.0	126.0
01/30/2020	8235.0	171.0	143.0
01/31/2020	9925.0	213.0	222.0
02/01/2020	12038.0	259.0	284.0
02/02/2020	16787.0	362.0	472.0
02/03/2020	19881.0	426.0	623.0
02/04/2020	23892.0	492.0	852.0
02/05/2020	27636.0	564.0	1124.0
02/06/2020	30818.0	634.0	1487.0
02/07/2020	34392.0	719.0	2011.0
02/08/2020	37121.0	806.0	2616.0
02/09/2020	40151.0	906.0	3244.0
02/10/2020	42763.0	1013.0	3946.0
02/11/2020	44803.0	1113.0	4683.0
02/12/2020	45222.0	1118.0	5150.0
02/13/2020	60370.0	1371.0	6295.0
02/14/2020	66887.0	1523.0	8058.0
02/15/2020	69032.0	1666.0	9395.0
02/16/2020	71226.0	1770.0	10865.0
02/17/2020	73260.0	1868.0	12583.0
02/18/2020	75138.0	2007.0	14352.0
02/19/2020	75641.0	2122.0	16121.0
02/20/2020	76199.0	2247.0	18177.0
...	...	...	...
03/03/2020	92844.0	3160.0	48229.0
03/04/2020	95124.0	3254.0	51171.0

03/05/2020	97886.0	3348.0	53797.0
03/06/2020	101800.0	3460.0	55866.0
03/07/2020	105836.0	3558.0	58359.0
03/08/2020	109835.0	3803.0	60695.0
03/09/2020	113582.0	3996.0	62512.0
03/10/2020	118582.0	4262.0	64404.0
03/11/2020	125865.0	4615.0	67003.0
03/12/2020	128343.0	4720.0	68324.0
03/13/2020	145193.0	5404.0	70251.0
03/14/2020	156099.0	5819.0	72624.0
03/15/2020	167447.0	6440.0	76034.0
03/16/2020	181546.0	7126.0	78088.0
03/17/2020	197168.0	7905.0	80840.0
03/18/2020	214915.0	8733.0	83313.0
03/19/2020	242713.0	9867.0	84962.0
03/20/2020	272167.0	11299.0	87403.0
03/21/2020	304528.0	12973.0	91676.0
03/22/2020	337020.0	14623.0	97243.0
03/23/2020	378287.0	16497.0	100958.0
03/24/2020	417966.0	18615.0	107705.0
03/25/2020	467594.0	21181.0	113770.0
03/26/2020	529591.0	23970.0	122150.0
03/27/2020	593291.0	27198.0	130915.0
03/28/2020	660706.0	30652.0	139415.0
03/29/2020	720117.0	33925.0	149082.0
03/30/2020	782365.0	37582.0	164566.0
03/31/2020	857487.0	42107.0	178034.0
04/01/2020	932605.0	46809.0	193177.0

71 rows × 3 columns

In [9]:

```
print("\n\t\tMaximum number of Confirmed,Deaths and Recovered Cases\n")
df1 = df.groupby(['Country', 'Province/State'])['Confirmed', 'Deaths', 'Recovered'].max()
df1 = df.groupby('Date')['Confirmed', 'Deaths', 'Recovered'].sum().reset_index()
df1 = df1[df1['Date']==max(df1['Date'])].reset_index(drop=True)
df1.style.background_gradient(cmap='Pastel1')
```

```
                Maximum number of Confirmed,Deaths and Recovered Cases
```

Out[9]:

	Date	Confirmed	Deaths	Recovered
0	04/01/2020	932605	46809	193177

In [10]:

```
print("\n\t\tWorld View - Country wise\n")
df_grouped = df.groupby('Country')['Confirmed', 'Deaths', 'Recovered'].sum().reset_index()
df2 = df_grouped.sort_values(by='Confirmed', ascending=False)
df2 = df2.reset_index(drop=True)
df2.style.background_gradient(cmap='summer')
#Maximum number of cases in China followed by Italy and Iran
```

```
                World View - Country wise
```

Out[10]:

	Country	Confirmed	Deaths	Recovered
0	Mainland China	4.2578e+06	148955	2.45914e+06
1	Italy	1.32166e+06	129684	162971
2	US	1.30082e+06	22960	27926
3	Spain	846650	63950	125911
4	Germany	666629	4796	87128
5	Iran	575157	37407	188316

6	France	467914	25386	64099
7	South Korea	266905	2847	60094
8	UK	195591	11479	2320
9	Switzerland	172905	3054	12034
10	Netherlands	115697	7012	1066
11	Belgium	102999	3973	11338
12	Austria	95530	772	4319
13	Turkey	74455	1253	1007
14	Canada	61818	673	5255
15	Norway	56212	278	101
16	Portugal	55170	979	391
17	Sweden	48865	1184	321
18	Brazil	43968	1216	417
19	Australia	41009	212	2967
20	Israel	40948	120	1310
21	Denmark	36004	646	1479
22	Japan	32603	914	6455
23	Malaysia	31111	309	4557
24	Czech Republic	29492	149	229
25	Ireland	26756	381	80
26	Others	26228	189	5003
27	Chile	20694	66	817
28	Luxembourg	19529	191	350
29	Ecuador	19217	516	142
30	Poland	18778	256	165
31	Pakistan	17705	153	471
32	Romania	17202	406	1735
33	Thailand	16277	84	2816
34	Russia	15051	78	795
35	Finland	14865	86	212
36	Greece	14824	389	547

36	Greece	14824	389	547
37	Singapore	14124	28	4784
38	Saudi Arabia	13986	55	858
39	Philippines	13846	793	436
40	Indonesia	13648	1146	675
41	Iceland	12520	29	1221
42	India	12336	284	1012
43	South Africa	11553	17	239
44	Qatar	11128	7	572
45	Hong Kong	10391	146	2828
46	Slovenia	9450	98	83
47	Peru	9025	180	912
48	Bahrain	8495	45	3383
49	Panama	8466	163	36
50	Mexico	8141	137	192
51	Estonia	8132	19	168
52	Egypt	7935	390	1611
53	Argentina	7548	186	1171
54	Dominican Republic	7424	277	39
55	Croatia	7385	42	428
56	Colombia	7197	85	148
57	Iraq	7097	561	1664
58	Serbia	6925	99	41
59	United Arab Emirates	6609	40	963
60	Diamond Princess	5696	81	4790
61	Algeria	5675	382	655
62	Lebanon	5660	124	300
63	Taiwan	4874	68	779
64	Armenia	4670	16	236
65	Kuwait	4642	0	725
66	Morocco	4412	231	143

67	New Zealand	4347	4	424
68	Lithuania	4192	55	36
69	Hungary	4037	139	333
70	Bulgaria	3998	81	112
71	Ukraine	3858	109	53
72	Slovakia	3840	5	49
73	Latvia	3723	0	23
74	San Marino	3700	344	107
75	Costa Rica	3362	26	33
76	Bosnia and Herzegovina	3281	63	91
77	Andorra	3177	53	47
78	Vietnam	3148	0	881
79	Uruguay	3041	5	82
80	Tunisia	2899	76	24
81	North Macedonia	2827	51	66
82	Moldova	2751	26	76
83	Jordan	2699	20	160
84	Albania	2491	106	308
85	Cyprus	2409	50	134
86	Kazakhstan	2315	13	112
87	Azerbaijan	2274	42	238
88	Burkina Faso	2183	102	207
89	Oman	1932	2	394
90	Malta	1895	0	37
91	Brunei	1819	5	221
92	Sri Lanka	1651	9	144
93	Senegal	1634	1	230
94	Belarus	1630	3	400
95	Afghanistan	1462	33	31
96	Venezuela	1439	14	303

97	Georgia	1377	0	155
98	Cambodia	1368	0	181
99	Ghana	1341	43	71
100	Cameroon	1300	30	39
101	Uzbekistan	1243	11	43
102	Ivory Coast	1242	4	48
103	Cuba	1232	34	39
104	Nigeria	1019	13	48
105	Mauritius	1002	32	0
106	Honduras	990	30	15
107	Macau	926	0	418
108	Trinidad and Tobago	847	21	7
109	Montenegro	824	12	0
110	Bolivia	797	18	1
111	Liechtenstein	762	0	0
112	West Bank and Gaza	751	7	124
113	Congo (Kinshasa)	751	50	13
114	Kyrgyzstan	698	0	9
115	Kosovo	687	7	20
116	Rwanda	661	0	0
117	Paraguay	615	29	6
118	Bangladesh	600	55	158
119	Monaco	489	4	13
120	Kenya	451	7	10
121	Jamaica	426	16	34
122	Guatemala	372	17	66
123	Madagascar	328	0	0
124	Togo	314	7	29
125	Maldives	308	0	99
126	Barbados	289	0	0
127	Uganda	255	0	0

128	Ethiopia	232	0	10
129	Zambia	230	0	0
130	Niger	193	16	0
131	El Salvador	188	2	0
132	Guadeloupe	187	0	0
133	Tanzania	183	2	6
134	Martinique	172	6	0
135	Mongolia	164	0	6
136	Guyana	159	23	0
137	Djibouti	158	0	0
138	Equatorial Guinea	145	0	2
139	Mali	137	8	0
140	Reunion	137	0	0
141	Guinea	134	0	0
142	Seychelles	121	0	0
143	French Guiana	117	0	6
144	Congo (Brazzaville)	116	0	0
145	Haiti	112	0	4
146	Namibia	111	0	18
147	Bahamas	107	1	9
148	Gabon	104	13	0
149	Suriname	98	0	0
150	Nepal	97	0	50
151	Dominica	91	0	0
152	Palestine	86	0	0
153	Eswatini	84	0	0
154	Benin	81	0	2
155	Eritrea	80	0	0
156	Saint Lucia	79	0	6
157	Burma	70	2	0

158	Antigua and Barbuda	68	0	0
159	Mozambique	66	0	0
160	Syria	62	7	0
161	Zimbabwe	61	10	0
162	Grenada	61	0	0
163	Sudan	60	23	3
164	Laos	60	0	0
165	Holy See	60	0	0
166	Angola	56	8	2
167	Cabo Verde	55	9	0
168	Mauritania	52	3	8
169	Fiji	52	0	0
170	Bhutan	50	0	0
171	Liberia	47	0	0
172	Central African Republic	44	0	0
173	Libya	43	0	1
174	Chad	42	0	0
175	Nicaragua	38	6	0
176	Guinea-Bissau	35	0	0
177	Somalia	34	0	2
178	Gambia	34	10	2
179	Saint Kitts and Nevis	33	0	0
180	occupied Palestinian territory	25	0	0
181	Mayotte	21	0	0
182	Republic of Ireland	21	0	0
183	Belize	21	0	0
184	Aruba	19	0	0
185	Saint Vincent and the Grenadines	19	0	5
186	Saint Barthelemy	17	0	0
187	MS Zaandam	17	2	0

188	Papua New Guinea	13	0	0
189	Timor-Leste	11	0	0
190	Botswana	11	2	0
191	Bahamas, The	10	0	0
192	Faroe Islands	10	0	0
193	Gibraltar	7	0	1
194	Guam	6	0	0
195	Jersey	6	0	0
196	Vatican City	4	0	0
197	Burundi	4	0	0
198	Gambia, The	4	0	0
199	Cayman Islands	3	0	0
200	Sierra Leone	3	0	0
201	Greenland	3	0	0
202	Guernsey	3	0	0
203	The Bahamas	3	0	0
204	Puerto Rico	3	0	0
205	Curacao	2	0	0
206	('St. Martin',)	2	0	0
207	St. Martin	2	0	0
208	East Timor	1	0	0
209	Channel Islands	1	0	0
210	The Gambia	1	0	0
211	Cape Verde	1	0	0
212	North Ireland	1	0	0
213	Republic of the Congo	1	0	0
214	Azerbaijan	1	0	0

In [11]:

```
print("\n\t\tChina \n")
China = df.query('Country=="Mainland China"').groupby("Last Update")[['Confirmed', 'Deaths', 'Recovered']].sum().reset_index()
China
#Exploring data of china
```

China

Out[11]:

	Last Update	Confirmed	Deaths	Recovered
0	2020-01-22 17:00:00	547.0	17.0	28.0
1	2020-01-23 17:00:00	639.0	18.0	30.0
2	2020-01-24 17:00:00	916.0	26.0	36.0
3	2020-01-25 17:00:00	1399.0	42.0	39.0
4	2020-01-26 16:00:00	2062.0	56.0	49.0
5	2020-01-27 23:59:00	2863.0	82.0	58.0
6	2020-01-28 23:00:00	5494.0	131.0	101.0
7	2020-01-29 19:30:00	6070.0	133.0	120.0
8	2020-01-30 16:00:00	8124.0	171.0	135.0
9	2020-01-31 15:20:00	29.0	0.0	2.0
10	2020-01-31 23:59:00	9783.0	213.0	214.0
11	2020-02-01 01:52:00	1176.0	4.0	19.0
12	2020-02-01 01:52:40	10.0	0.0	0.0
13	2020-02-01 02:13:00	26.0	0.0	0.0
14	2020-02-01 05:37:00	295.0	0.0	1.0
15	2020-02-01 06:05:00	169.0	1.0	10.0
16	2020-02-01 07:51:00	206.0	0.0	3.0
17	2020-02-01 08:43:00	309.0	2.0	4.0
18	2020-02-01 09:17:00	168.0	1.0	9.0
19	2020-02-01 10:33:00	80.0	2.0	2.0

20	2020-02-01 10:53:00	599.0	0.0	21.0
21	2020-02-01 11:03:00	389.0	0.0	8.0
22	2020-02-01 11:53:00	7153.0	249.0	168.0
23	2020-02-01 13:33:00	297.0	0.0	5.0
24	2020-02-01 14:03:00	202.0	0.0	6.0
25	2020-02-01 14:23:00	535.0	0.0	14.0
26	2020-02-01 15:23:00	64.0	0.0	1.0
27	2020-02-01 15:43:00	81.0	0.0	0.0
28	2020-02-01 15:53:00	93.0	0.0	2.0
29	2020-02-02 00:23:13	111.0	0.0	2.0
...	...	...	...	...
765	2020-03-19 23:43:03	411.0	9.0	399.0
766	2020-03-20 00:13:24	134.0	2.0	98.0
767	2020-03-20 00:43:02	879.0	7.0	826.0
768	2020-03-20 01:13:16	1009.0	10.0	985.0
769	2020-03-20 01:43:03	793.0	7.0	780.0
770	2020-03-20 02:13:46	968.0	26.0	926.0
771	2020-03-20 02:43:09	1533.0	2.0	1514.0
772	2020-03-20 06:43:05	491.0	8.0	390.0
773	2020-03-20 07:43:02	67800.0	3133.0	58382.0
774	2020-03-20 09:13:30	1395.0	8.0	1323.0
775	2020-03-21 00:43:02	764.0	7.0	749.0
776	2020-03-21 01:13:10	303.0	1.0	295.0
777	2020-03-21 01:13:11	542.0	3.0	536.0
778	2020-03-21 01:43:03	248.0	3.0	239.0
779	2020-03-21 03:13:13	134.0	2.0	113.0
780	2020-03-21 03:43:03	1236.0	1.0	1219.0
781	2020-03-21 04:43:06	380.0	3.0	327.0
782	2020-03-21 05:13:04	504.0	8.0	396.0
783	2020-03-21 10:13:08	67800.0	3139.0	58946.0
784	2020-03-21 12:43:08	1400.0	8.0	1325.0

785	2020-03-23 23:23:20	81116.0	3270.0	72709.0
786	2020-03-24 23:41:50	81180.0	3277.0	73169.0
787	2020-03-25 23:37:49	81221.0	3281.0	73661.0
788	2020-03-26 23:53:24	81298.0	3287.0	74061.0
789	2020-03-27 23:27:48	81345.0	3292.0	74600.0
790	2020-03-28 23:11:06	81401.0	3295.0	74978.0
791	2020-03-29 23:14:06	81444.0	3300.0	75460.0
792	2020-03-30 22:58:55	81478.0	3304.0	75790.0
793	2020-03-31 23:49:27	81524.0	3305.0	76068.0
794	2020-04-01 22:04:58	81555.0	3312.0	76248.0

795 rows × 4 columns

In [12]:

```
print("\n\t\tCountry Wise - Sorted(Alphabetically) order\n")
df.groupby("Country")[['Confirmed', 'Deaths', 'Recovered']].sum().reset_index()
```

```
                Country Wise - Sorted(Alphabetical
ly) order
```

Out[12]:

	Country	Confirmed	Deaths	Recovered
0	Azerbaijan	1.0	0.0	0.0
1	('St. Martin',)	2.0	0.0	0.0
2	Afghanistan	1462.0	33.0	31.0
3	Albania	2491.0	106.0	308.0
4	Algeria	5675.0	382.0	655.0
5	Andorra	3177.0	53.0	47.0
6	Angola	56.0	8.0	2.0
7	Antigua and Barbuda	68.0	0.0	0.0
8	Argentina	7548.0	186.0	1171.0
9	Armenia	4670.0	16.0	236.0

9	Armenia	4070.0	16.0	230.0
10	Aruba	19.0	0.0	0.0
11	Australia	41009.0	212.0	2967.0
12	Austria	95530.0	772.0	4319.0
13	Azerbaijan	2274.0	42.0	238.0
14	Bahamas	107.0	1.0	9.0
15	Bahamas, The	10.0	0.0	0.0
16	Bahrain	8495.0	45.0	3383.0
17	Bangladesh	600.0	55.0	158.0
18	Barbados	289.0	0.0	0.0
19	Belarus	1630.0	3.0	400.0
20	Belgium	102999.0	3973.0	11338.0
21	Belize	21.0	0.0	0.0
22	Benin	81.0	0.0	2.0
23	Bhutan	50.0	0.0	0.0
24	Bolivia	797.0	18.0	1.0
25	Bosnia and Herzegovina	3281.0	63.0	91.0
26	Botswana	11.0	2.0	0.0
27	Brazil	43968.0	1216.0	417.0
28	Brunei	1819.0	5.0	221.0
29	Bulgaria	3998.0	81.0	112.0
...	...	...	...	...
185	St. Martin	2.0	0.0	0.0
186	Sudan	60.0	23.0	3.0
187	Suriname	98.0	0.0	0.0
188	Sweden	48865.0	1184.0	321.0
189	Switzerland	172905.0	3054.0	12034.0
190	Syria	62.0	7.0	0.0
191	Taiwan	4874.0	68.0	779.0
192	Tanzania	183.0	2.0	6.0
193	Thailand	16277.0	84.0	2816.0

194	The Bahamas	3.0	0.0	0.0
195	The Gambia	1.0	0.0	0.0
196	Timor-Leste	11.0	0.0	0.0
197	Togo	314.0	7.0	29.0
198	Trinidad and Tobago	847.0	21.0	7.0
199	Tunisia	2899.0	76.0	24.0
200	Turkey	74455.0	1253.0	1007.0
201	UK	195591.0	11479.0	2320.0
202	US	1300824.0	22960.0	27926.0
203	Uganda	255.0	0.0	0.0
204	Ukraine	3858.0	109.0	53.0
205	United Arab Emirates	6609.0	40.0	963.0
206	Uruguay	3041.0	5.0	82.0
207	Uzbekistan	1243.0	11.0	43.0
208	Vatican City	4.0	0.0	0.0
209	Venezuela	1439.0	14.0	303.0
210	Vietnam	3148.0	0.0	881.0
211	West Bank and Gaza	751.0	7.0	124.0
212	Zambia	230.0	0.0	0.0
213	Zimbabwe	61.0	10.0	0.0
214	occupied Palestinian territory	25.0	0.0	0.0

215 rows × 4 columns

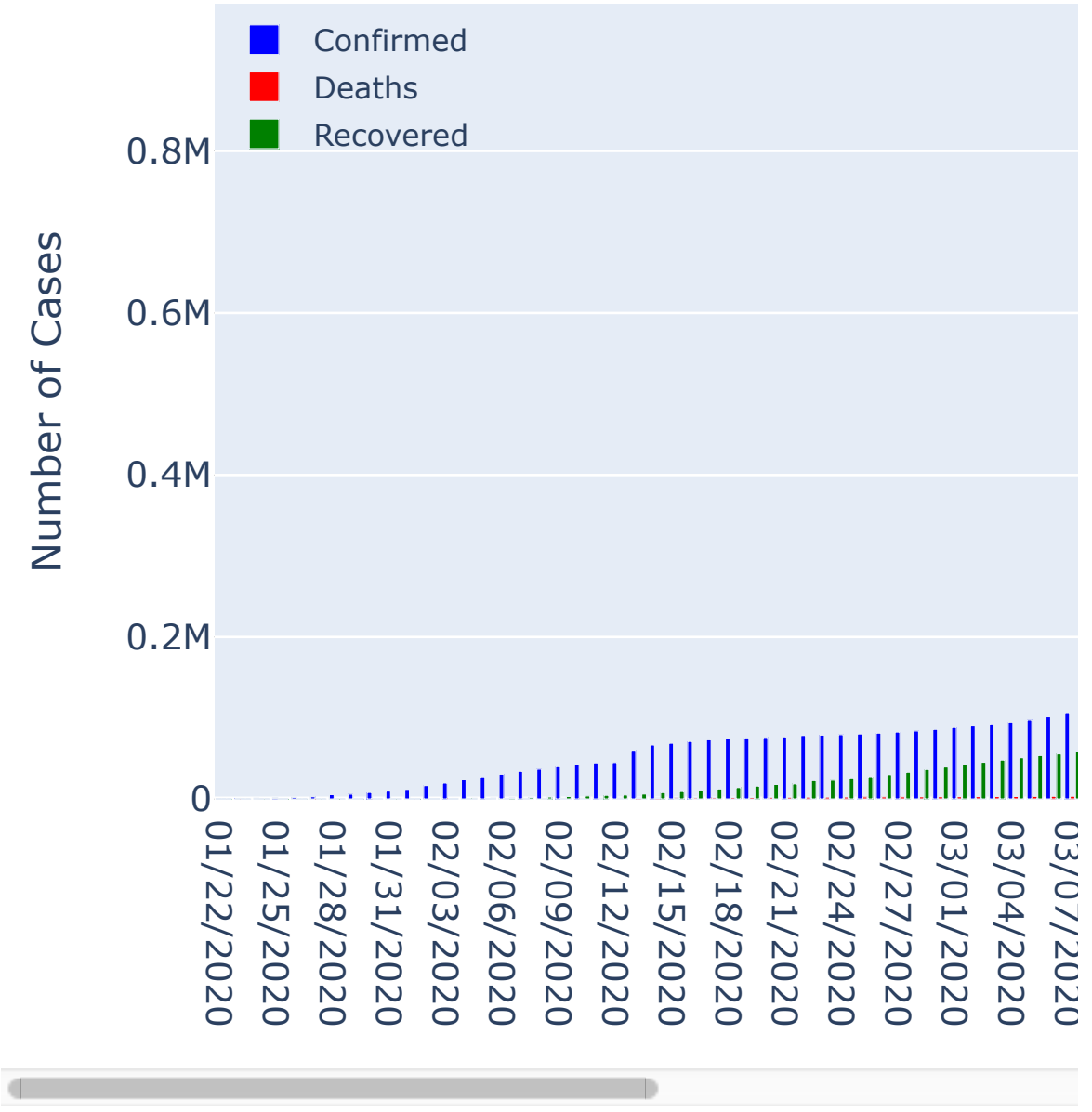
Visualisation

In [13]:

```
confirmed = df.groupby('Date').sum()['Confirmed'].reset_index(
)
deaths = df.groupby('Date').sum()['Deaths'].reset_index()
recovered = df.groupby('Date').sum()['Recovered'].reset_index(
)
```

In [14]:

```
import plotly.graph_objects as go
fig = go.Figure()
fig.add_trace(go.Bar(x=confirmed['Date'],
                y=confirmed['Confirmed'],
                name='Confirmed',
                marker_color='blue'
                ))
fig.add_trace(go.Bar(x=deaths['Date'],
                y=deaths['Deaths'],
                name='Deaths',
                marker_color='Red'
                ))
fig.add_trace(go.Bar(x=recovered['Date'],
                y=recovered['Recovered'],
                name='Recovered',
                marker_color='Green'
                ))

fig.update_layout(title='Worldwide Corona Virus Cases - Confir
med, Deaths, Recovered (Bar Chart)',
    xaxis_tickfont_size=14,
    yaxis=dict(
        title='Number of Cases',
        titlefont_size=16,
        tickfont_size=14,
    ),
    legend=dict(
        x=0,
        y=1.0,
        bgcolor='rgba(255, 255, 255, 0)',
        bordercolor='rgba(255, 255, 255, 0)'
    ),
    barmode='group',
    bargap=0.15, # gap between bars of adjacent location coord
inates.
    bargroupgap=0.1 # gap between bars of the same location co
ordinate.
)
fig.show()
#steep rise in confirmed cases around 12th March 2020. Reason
could be more tests and results
#available
```



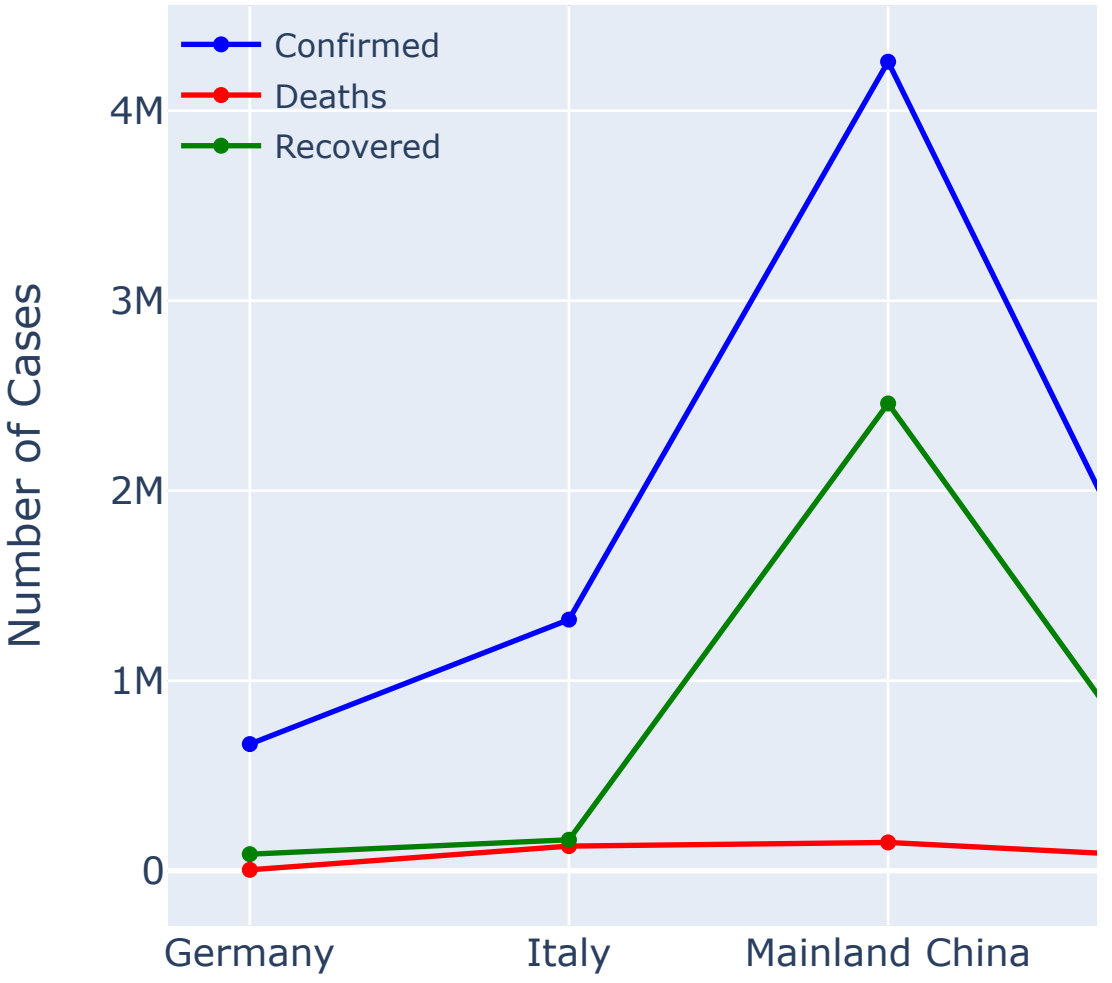
In [15]:

```
n = 5
c_lrgst = df_grouped.Confirmed.nlargest(n)
d_lrgst = df_grouped.Deaths.nlargest(n)
r_lrgst = df.Recovered.nlargest(n)

top_5 = df_grouped.query('Confirmed in @c_lrgst')
```

In [16]:

```
fig = go.Figure()
fig.add_trace(go.Scatter(x=top_5['Country'],
                         y=top_5['Confirmed'],
                         mode='lines+markers',
                         name='Confirmed',
                         line=dict(color='blue', width=2)
                        ))
fig.add_trace(go.Scatter(x=top_5['Country'],
                         y=top_5['Deaths'],
                         mode='lines+markers',
                         name='Deaths',
                         line=dict(color='Red', width=2)
                        ))
fig.add_trace(go.Scatter(x=top_5['Country'],
                         y=top_5['Recovered'],
                         mode='lines+markers',
                         name='Recovered',
                         line=dict(color='Green', width=2)
                        ))
fig.update_layout(
    title='Top 5 Countries with Corona Virus Cases (Line Chart
)',
    xaxis_tickfont_size=14,
    yaxis=dict(
        title='Number of Cases',
        titlefont_size=16,
        tickfont_size=14,
    ),
    legend=dict(
        x=0,
        y=1.0,
        bgcolor='rgba(255, 255, 255, 0)',
        bordercolor='rgba(255, 255, 255, 0)'
    )
)
fig.show()
#China cases far more than the other top 5 countries
```

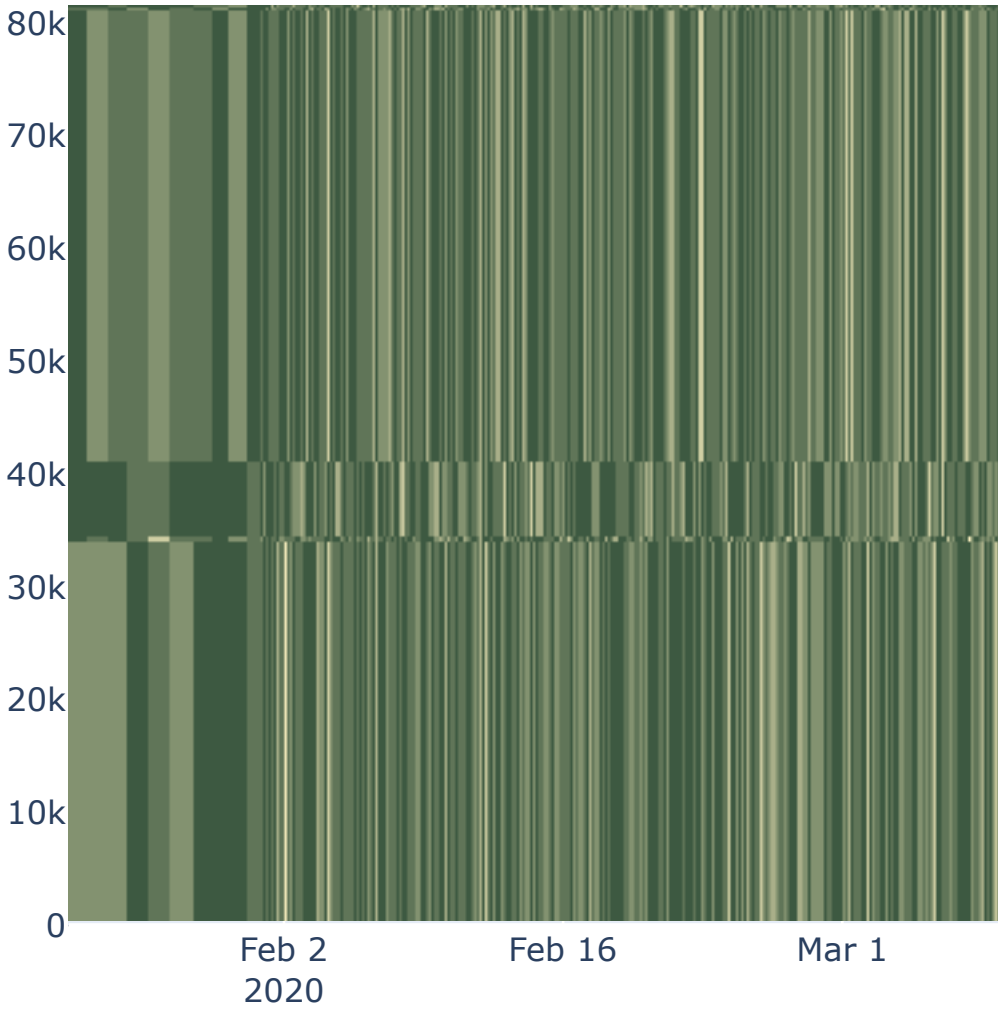
Top 5 Countries with Corona Virus Cases (Line



In [17]:

```
import plotly.graph_objects as go
import numpy as np
z = np.random.poisson(size=(len(China['Confirmed']), len(China
['Last Update'])))

fig = go.Figure(data=go.Heatmap(
        z=z,
        x=China['Last Update'],
        y=China['Confirmed'],
        colorscale='fall'))

fig.update_layout(
    title='China - Number of Cases per day',
    xaxis_nticks=5)

fig.show()

#Cases are still increasing . No notion of decrease till now.
```

China - Number of Cases per day



In [18]:

```
import plotly.express as px
fig = px.density_mapbox(df_latlong,
                        lat="Lat",
                        lon="Long",
                        hover_name="Province/State",
                        hover_data=["Confirmed","Deaths","Recovered"],
                        animation_frame="Date",
                        color_continuous_scale="Portland",
                        radius=7,
                        zoom=1,height=700)
fig.update_layout(title='Worldwide Corona Virus Cases Time Lapse - Confirmed, Deaths, Recovered',
                  font=dict(family="Courier New, monospace",
                            size=14,
                            color="#7f7f7f")
                 )
fig.update_layout(mapbox_style="open-street-map", mapbox_center_lon=0)
fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0}) #tight layout - right , top, left , below


fig.show()
#trend of spread and at what rate. Huge jump in March
```

Forecasting / Prediction

Prophet Algorithm

Worldwide - Confirmed Cases

In [19]:

```
from fbprophet import Prophet
```

In [20]:

```
#Fixing data for forecasting
confirmed = df.groupby('Date').sum()['Confirmed'].reset_index(
)
confirmed.columns = ['ds','y']
confirmed['ds'] = pd.to_datetime(confirmed['ds'])
confirmed['cap']= 10000000
confirmed.head()
```

Out[20]:

	ds	y	cap
0	2020-01-22	555.0	10000000
1	2020-01-23	653.0	10000000
2	2020-01-24	941.0	10000000
3	2020-01-25	1438.0	10000000
4	2020-01-26	2118.0	10000000

In [21]:

```
m = Prophet(interval_width=0.95,yearly_seasonality=True,daily_
seasonality=True,growth='logistic')
m.fit(confirmed)
future = m.make_future_dataframe(periods=61)
future.tail()
```

Out[21]:

	ds
127	2020-05-28
128	2020-05-29
129	2020-05-30
130	2020-05-31
131	2020-06-01

In [22]:

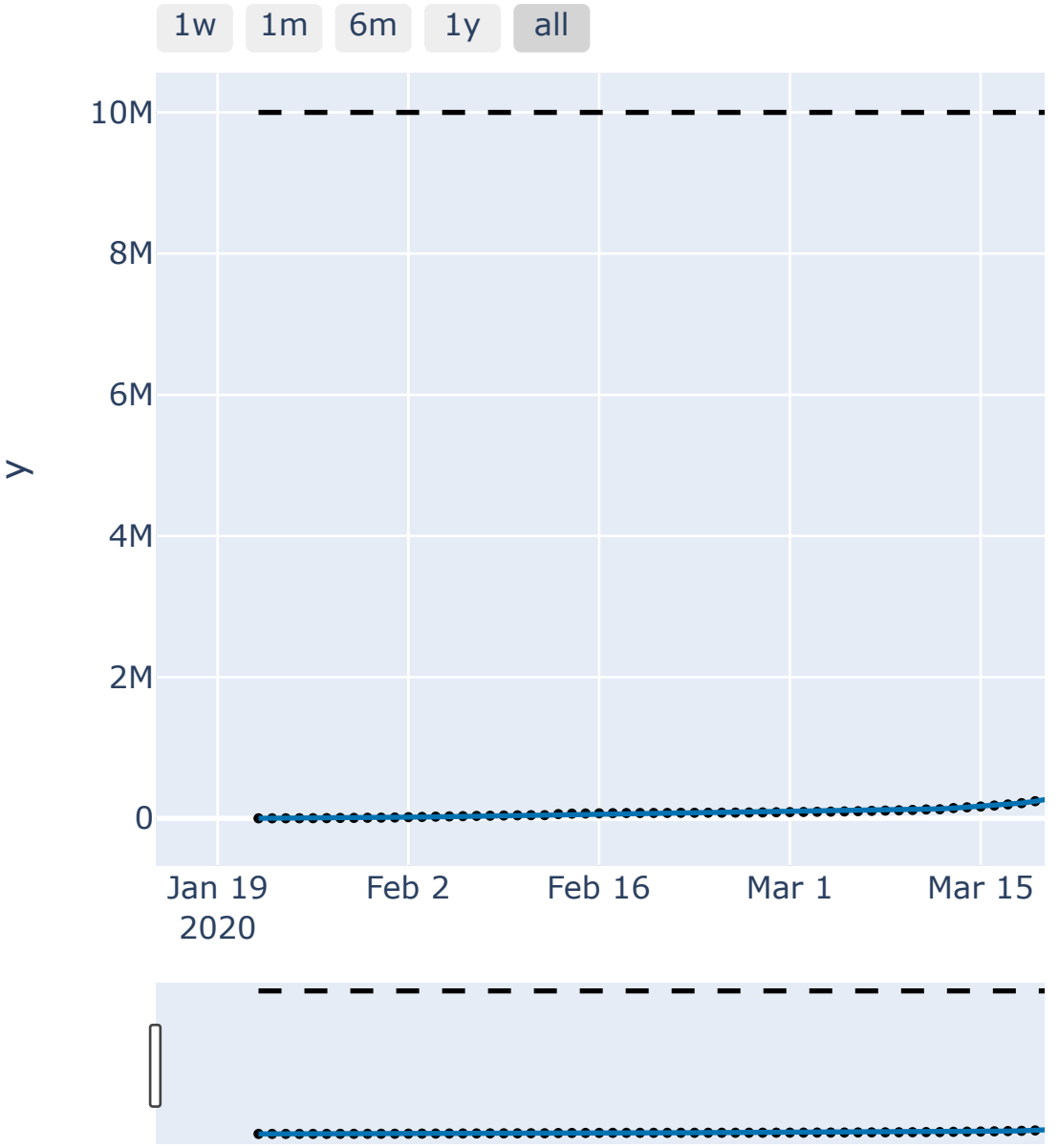
```
future['cap']=10000000
forecast = m.predict(future)
forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
```

Out[22]:

	ds	yhat	yhat_lower	yhat_upper
127	2020-05-28	7.927155e+06	7.922538e+06	7.931943e+06
128	2020-05-29	7.460269e+06	7.455638e+06	7.465080e+06
129	2020-05-30	6.959954e+06	6.955363e+06	6.964956e+06
130	2020-05-31	6.436252e+06	6.431948e+06	6.440879e+06
131	2020-06-01	5.901951e+06	5.897467e+06	5.906621e+06

In [23]:

```
from fbprophet.plot import plot_plotly
import plotly.offline as py
py.init_notebook_mode()

fig = plot_plotly(m, forecast)  # This returns a plotly Figure
py.iplot(fig)
```

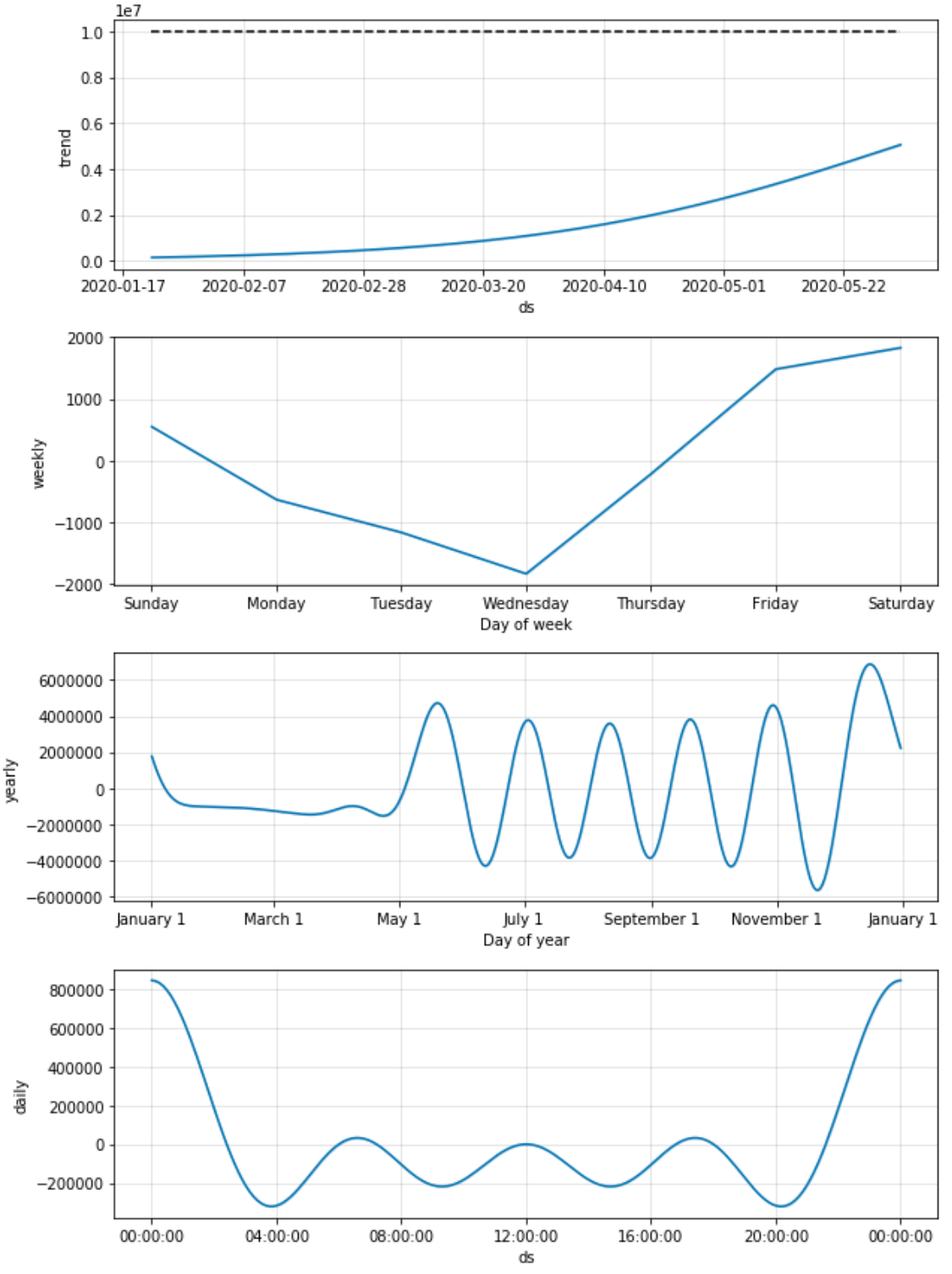


In [24]:

```
#help(Prophet)
```

In [25]:

```
m.plot_components(forecast);
```



Worldwide - Deaths

In [26]:

```
#fixing data
deaths.columns = ['ds','y']
deaths['ds'] = pd.to_datetime(deaths['ds'])
```

In [27]:

```
m = Prophet(interval_width=0.95,yearly_seasonality=True,daily_
seasonality=True)
m.fit(deaths)
future = m.make_future_dataframe(periods=61)
future.tail()
```

Out[27]:

	ds
127	2020-05-28
128	2020-05-29
129	2020-05-30
130	2020-05-31
131	2020-06-01

In [28]:

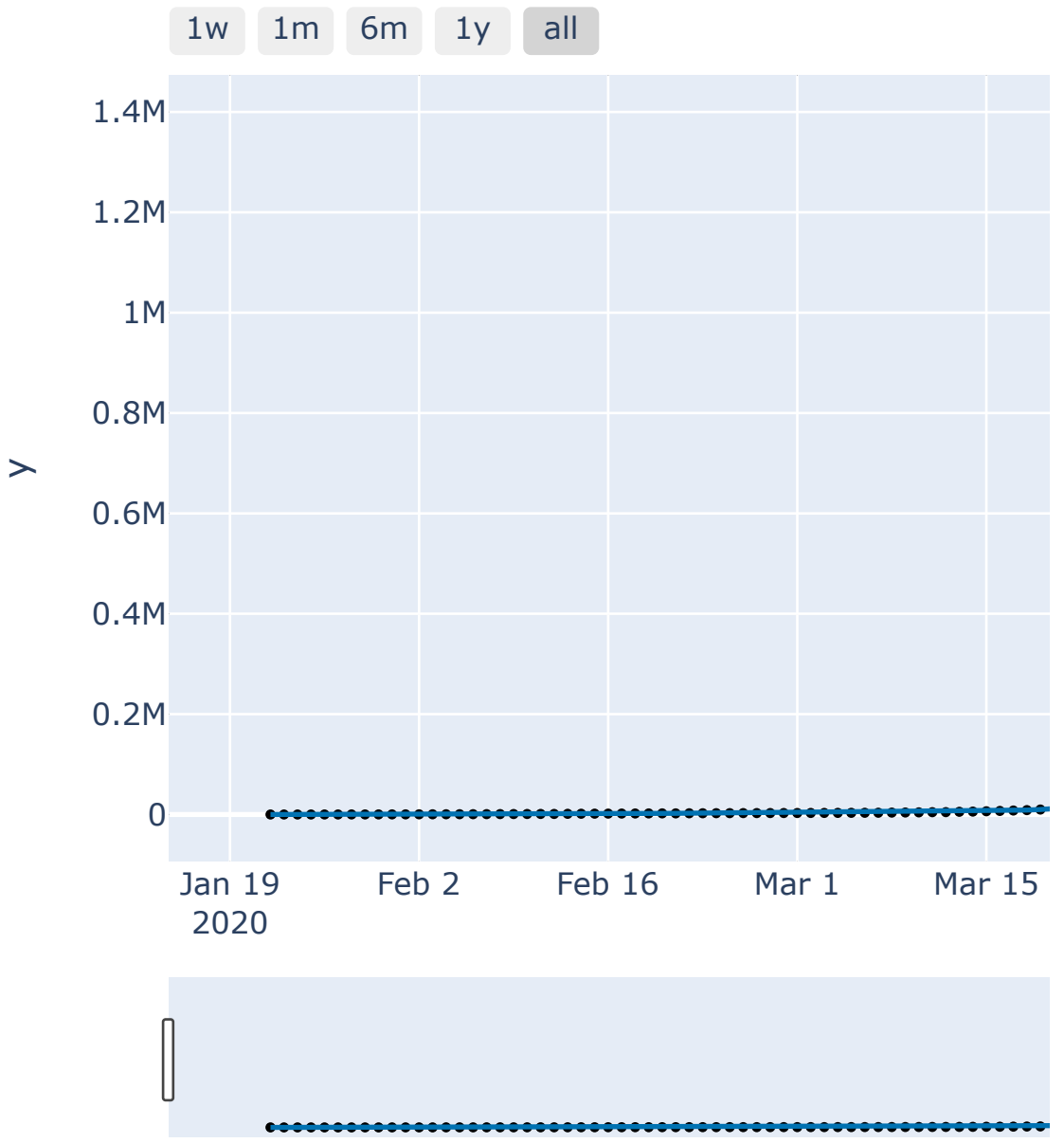
```
forecast = m.predict(future)
forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
```

Out[28]:

	ds	yhat	yhat_lower	yhat_upper
127	2020-05-28	776561.240581	775895.515981	777189.346960
128	2020-05-29	671867.985595	671183.206533	672508.301321
129	2020-05-30	563038.907552	562365.880648	563708.982770
130	2020-05-31	451423.990187	450733.211881	452111.922118
131	2020-06-01	338589.685631	337893.222230	339286.849017

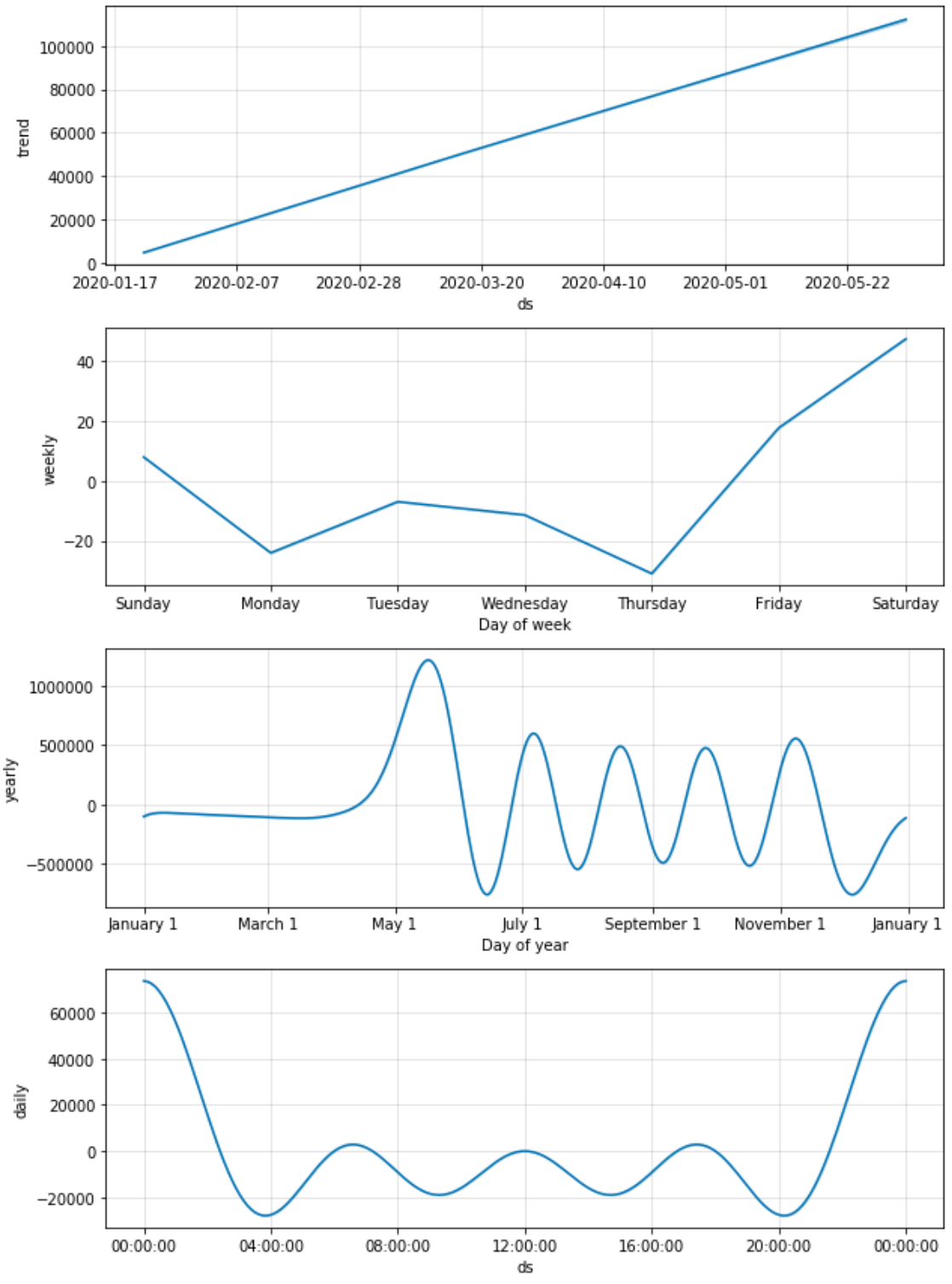
In [29]:

```
fig = plot_plotly(m, forecast)  # This returns a plotly Figure
py.iplot(fig)
```



In [30]:

```
m.plot_components(forecast);
```



Worldwide - Recovery

In [31]:

```
#fixing data
recovered.columns = ['ds','y']
recovered['ds'] = pd.to_datetime(recovered['ds'])
```

In [32]:

```
m = Prophet(interval_width=0.95,yearly_seasonality=True,daily_
seasonality=True)
m.fit(recovered)
future = m.make_future_dataframe(periods=61)
future.tail()
```

Out[32]:

	ds
127	2020-05-28
128	2020-05-29
129	2020-05-30
130	2020-05-31
131	2020-06-01

In [33]:

```
forecast = m.predict(future)
forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
```

Out[33]:

	ds	yhat	yhat_lower	yhat_upper
127	2020-05-28	108136.812296	86368.647714	131299.693304
128	2020-05-29	-681.653939	-23153.594715	22644.557341
129	2020-05-30	-98949.064529	-122050.561299	-74749.389746
130	2020-05-31	-185763.354863	-209296.789229	-161218.084032
131	2020-06-01	-259535.376280	-284033.169796	-234544.161261

In [34]:

```
fig = plot_plotly(m, forecast)  # This returns a plotly Figure
py.iplot(fig)
```

In [35]:

```
m.plot_components(forecast);
```