6. Covid-19 Analysis

- IBM CLOUD WATSON STUDIO

In [0]:

```
pip install plotly
```

```
Python interpreter will be restarted.
Collecting plotly
  Downloading plotly-4.12.0-py2.py3-none-any.whl (13.1 MB)
Requirement already satisfied: six in /databricks/python3/lib/python3.7/site-pac
kages (from plotly) (1.14.0)
Collecting retrying>=1.3.3
  Downloading retrying-1.3.3.tar.gz (10 kB)
Building wheels for collected packages: retrying
  Building wheel for retrying (setup.py): started
  Building wheel for retrying (setup.py): finished with status 'done'
  Created wheel for retrying: filename=retrying-1.3.3-py3-none-any.whl size=1143
0 sha256=528640eb6b7c1d585a2b8396b17ffb379afd31f68708f1cea0cd98d3125c9cfc
  Stored in directory: /root/.cache/pip/wheels/f9/8d/8d/f6af3f7f9eea3553bc2fe6d5
3e4b287dad18b06a861ac56ddf
Successfully built retrying
Installing collected packages: retrying, plotly
Successfully installed plotly-4.12.0 retrying-1.3.3
Python interpreter will be restarted.
```

In [0]:

```
pip install jinja2
```

```
Python interpreter will be restarted.
Collecting jinja2
  Downloading Jinja2-2.11.2-py2.py3-none-any.whl (125 kB)
Collecting MarkupSafe>=0.23
  Downloading MarkupSafe-1.1.1-cp37-cp37m-manylinux1_x86_64.whl (27 kB)
Installing collected packages: MarkupSafe, jinja2
Successfully installed MarkupSafe-1.1.1 jinja2-2.11.2
Python interpreter will be restarted.
```

In [0]:

```
import seaborn as sns
import plotly.express as px
%matplotlib inline
import pandas as pd
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt

df = pd.read_csv("https://pandemicdatalake.blob.core.windows.net/public/raw/covid-19/ec
dc_cases/latest/ECDCCases.csv")
df.head(1000)
```

	dateRep	day	month	year	cases	deaths	countriesAndTerritories	geold	countryter
0	08/11/2020	8	11	2020	126	6	Afghanistan	AF	
1	07/11/2020	7	11	2020	58	2	Afghanistan	AF	
2	06/11/2020	6	11	2020	40	0	Afghanistan	AF	
3	05/11/2020	5	11	2020	121	6	Afghanistan	AF	
4	04/11/2020	4	11	2020	86	4	Afghanistan	AF	
...	...	...	...	...	...	...	...	...	
995	24/06/2020	24	6	2020	0	0	Andorra	AD	
996	23/06/2020	23	6	2020	0	0	Andorra	AD	
997	22/06/2020	22	6	2020	0	0	Andorra	AD	
998	21/06/2020	21	6	2020	0	0	Andorra	AD	
999	20/06/2020	20	6	2020	0	0	Andorra	AD	

1000 rows × 12 columns

In [0]:

```
df.dtypes
```

```
Out[2]: dateRep                                                      object
day                                                          int64
month                                                        int64
year                                                         int64
cases                                                        int64
deaths                                                       int64
countriesAndTerritories                                     object
geoId                                                       object
countryterritoryCode                                        object
popData2019                                                float64
continentExp                                                object
Cumulative_number_for_14_days_of_COVID-19_cases_per_100000    float64
dtype: object
```

In [0]:

```
df_1 = pd.read_csv("https://pandemicdatalake.blob.core.windows.net/public/raw/covid-19/
ecdc_cases/latest/ECDCCases.csv")
df_1 = spark.createDataFrame(df_1)
```

In [0]:

```
df_1.write.mode("overwrite").saveAsTable("covidtable")
```

In [0]:

```
df.set_index('dateRep',inplace=True)
```

In [0]:

```
cv19_countries_day = df.groupby(by=['dateRep','countriesAndTerritories']).sum()[['case
s','deaths']]

Total_confirmed = cv19_countries_day.groupby('dateRep').sum()[['cases','deaths']].sum()
['cases']
Total_deaths = cv19_countries_day.groupby('dateRep').sum()[['cases','deaths']].sum()['d
eaths']

dicc = {'TotalConfirmed' : Total_confirmed, 'TotalDeaths' : Total_deaths, 'DeathRate' :
round((Total_deaths/Total_confirmed)*100,2)}
total = pd.DataFrame(dicc,index=['Counter'])[['TotalConfirmed','TotalDeaths','DeathRat
e']]


total.style.set_properties(**{
    'background-color': 'white',
    'font-size': '20pt',
    'color' : 'red'
})
```

	TotalConfirmed	TotalDeaths	DeathRate
Counter	49945364	1250275	2.500000

In [0]:

```
total= spark.createDataFrame(total)
```

In [0]:

```
total.write.mode("overwrite").saveAsTable("Total_Covid_numbers")
```

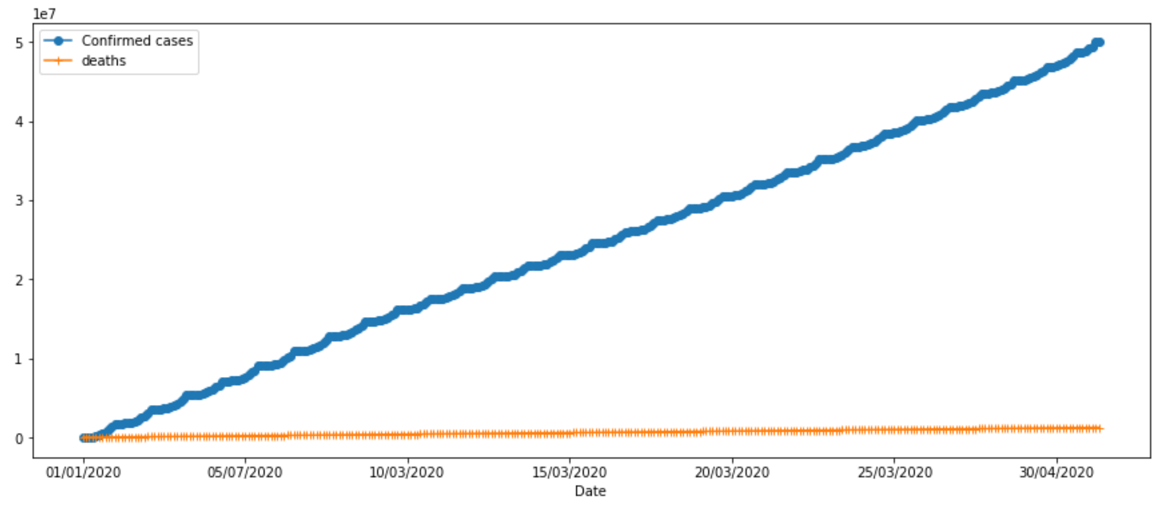
Visualization for total deaths and confirmed cases with timeline

In [0]:

```
covid19_total = df[['countriesAndTerritories','cases']].groupby(by='countriesAndTerrito
ries').sum().sort_values(by='cases',ascending=False)
covid19_total.columns=['cases']
covid19_total_d = df[['countriesAndTerritories','deaths']].groupby(by='countriesAndTerr
itories').sum().sort_values(by='deaths',ascending=False)
covid19_total_d.columns=['deaths']
```

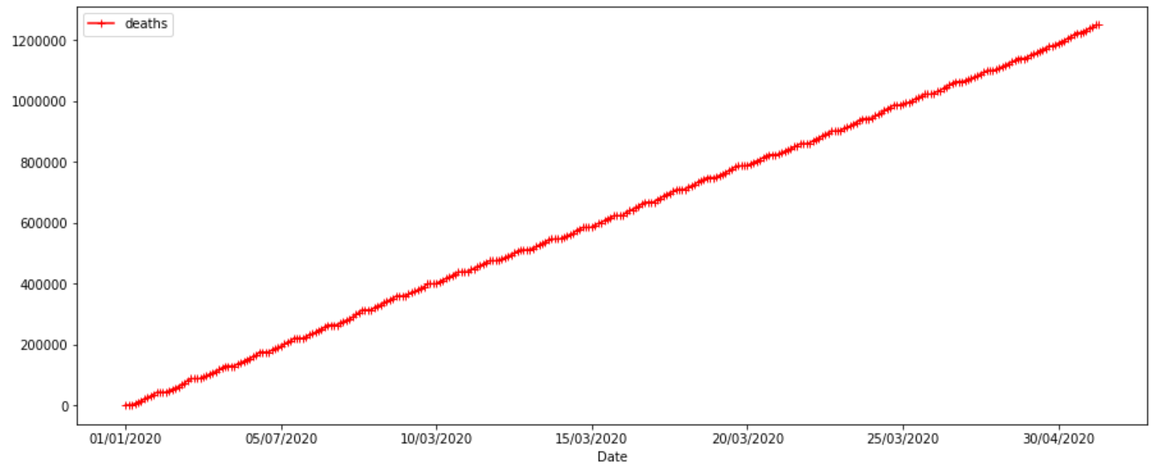
In [0]:

```
cv19_countries_day = df.groupby(by=['dateRep']).sum()[['cases','deaths']]
cv19_countries_day['cases'].cumsum().plot(figsize=(15,6),label='Confirmed cases',marker
='o')
cv19_countries_day['deaths'].cumsum().plot(label="deaths",marker='+')
plt.legend()
plt.xlabel('Date')
plt.show()

#cv19_countries_day.sort_values(by='DateRep',ascending = False)
```



In [0]:

```
cv19_countries_day['deaths'].cumsum().plot(figsize=(15,6),color='red',label="deaths",ma
rker='+')
plt.legend()
plt.xlabel('Date')
plt.show()
```



Now below we will create a table with with only the most latetst data of numbers for cases and deaths in respect to countries.

In [0]:

```
Todaynumbers=df.groupby('countriesAndTerritories').first().filter(['cases','deaths'])
```

In [0]:

```
Todaynumbers = spark.createDataFrame(Todaynumbers)
```

In [0]:

```
Todaynumbers.write.mode("overwrite").saveAsTable("Today_Covid_numbers")
```

In [0]:

```
#Top 10 cases
t10 = pd.concat([covid19_total.head(10),covid19_total_d.head(10)],axis=1,sort=False).he
ad(10)
```

Fatality Rate

In [0]:

```
def highlight_max_yellow(s):
    is_max = s == s.max()
    return ['background-color: yellow' if v else '' for v in is_max]

def highlight_max(data, color='yellow'):
    attr = 'background-color: {}'.format(color)
    if data.ndim == 1:  # Series from .apply(axis=0) or axis=1
        is_max = data == data.max()
        return [attr if v else '' for v in is_max]
    else:  # from .apply(axis=None)
        is_max = data == data.max().max()
        return pd.DataFrame(np.where(is_max, attr, ''),
                            index=data.index, columns=data.columns)

def highlight_max_all(s):
    is_max = s == s.max()
    return ['background-color: #f59d71' if v else '' for v in is_max]


def highlight_min(data):
    color_min= '#b5f5d4' #green
    attr = 'background-color: {}'.format(color_min)

    if data.ndim == 1:  # Series from .apply(axis=0) or axis=1
        is_min = data == data.min()
        return [attr if v else '' for v in is_min]
    else:
        is_min = data.groupby(level=0).transform('min') == data
        return pd.DataFrame(np.where(is_min, attr, ''),
                            index=data.index, columns=data.columns)
```

In [0]:

```
t10['DeathRatio'] = round((t10['deaths'] / t10['cases']) *100,2)
t10.sort_values(by='DeathRatio',ascending = False)
t10f = t10[['DeathRatio']].sort_values(by='DeathRatio',ascending=False).dropna()
t10f.style.apply(highlight_max, color='red', axis=None)
```

	DeathRatio
Mexico	9.860000
United_Kingdom	4.170000
Spain	2.920000
Brazil	2.880000
United_States_of_America	2.400000
France	2.300000
India	1.480000

In [0]:

```
t10f = spark.createDataFrame(t10f)
```

In [0]:

```
t10f.write.mode("overwrite").saveAsTable("fatility_covid")
```

CHANGE COVID RATE

In [0]:

```
covid19_change_global = cv19_countries_day.cumsum()
covid19_change_global[['Cases Day','Deaths Day']] = cv19_countries_day[['cases','death
s']]
covid19_change_global = covid19_change_global.pct_change(1)
covid19_change_global = covid19_change_global.sort_values(by='dateRep',ascending=False)
covid19_change_global = covid19_change_global.replace([np.inf, -np.inf], np.nan)
covid19_change_global = covid19_change_global.fillna(0)
covid19_change_global = round(covid19_change_global*100,2)
covid19_change_global = covid19_change_global.reset_index()

covid19_change_global_d = cv19_countries_day.cumsum()
covid19_change_global_d[['Cases Day','Deaths Day']] = cv19_countries_day[['cases','deat
hs']]
covid19_change_global_d = covid19_change_global_d.pct_change(1)
covid19_change_global_d = covid19_change_global_d.sort_values(by='dateRep',ascending=Fa
lse)
covid19_change_global_d = covid19_change_global_d.replace([np.inf, -np.inf], np.nan)
covid19_change_global_d = covid19_change_global_d.fillna(0)
covid19_change_global_d = round(covid19_change_global_d*100,2)
covid19_change_global_d = covid19_change_global_d.reset_index()
```

In [0]:

```
px.bar(data_frame=covid19_change_global,x=covid19_change_global['dateRep'],y=covid19_ch
ange_global['cases'], \
       color='cases',
       labels={'Cases':'Date','Deaths':'% change'},
       title='Cases: Global change percentage per day')
```

In [0]:

```
px.bar(data_frame=covid19_change_global_d,x=covid19_change_global_d['dateRep'],y=covid1
9_change_global_d['deaths'], \
       color='deaths',
       labels={'DateRep':'Date','Deaths':'% change'},
       title='Deaths: Global change percentage per day')
```

In [0]:

```
#Calculate change by country for the 15 first

Impacted_countries = df[['countriesAndTerritories','cases']].sort_values(by=['dateRep',
'cases'],ascending=False).head(15)['countriesAndTerritories']
Impacted_countries

top_impact = pd.DataFrame()

for country in Impacted_countries:
    top_impact[country] = df[df['countriesAndTerritories']==country]['cases']


top_impact = top_impact.reset_index().sort_values(by='dateRep',ascending=True) #true


#Normalize

#top_impact_norm = top_impact/top_impact.iloc[0] * 100
```

(Cases) Growth by time period (%) C(t)=C(t−1)∗Δ

In [0]:

```
#Australia 0 cases correction (mean between days 25.03 and 27.03)
#top_impact['Australia'][1] = 671.5

growth_impact_day = top_impact.set_index('dateRep').pct_change(1).reset_index().sort_va
lues(by='dateRep',ascending=False)
growth_impact_day = round(growth_impact_day.set_index('dateRep')*100,2).head(10)
growth_impact_day.style.apply(highlight_max_all).apply(highlight_min)
```

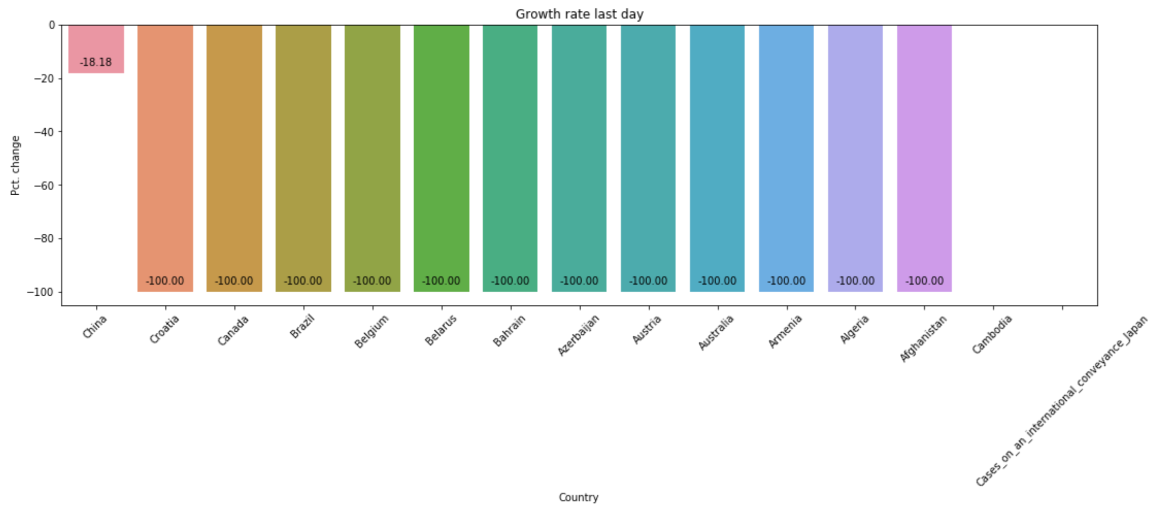
	China	Afghanistan	Algeria	Armenia	Australia	Austria
dateRep						
31/12/2019	-18.180000	-100.000000	-100.000000	-100.000000	-100.000000	-100.000000
31/10/2020	3.120000	726.320000	-12.360000	1820.160000	-89.430000	1765.840000
31/08/2020	-88.410000	-73.240000	-39.530000	-64.970000	-82.940000	72.390000
31/07/2020	5420.000000	-91.800000	352.630000	-0.280000	5908.330000	270.450000
31/05/2020	-94.050000	3107.410000	82.190000	610.000000	-97.410000	-94.530000
31/03/2020	-95.760000	inf	inf	inf	46300.000000	inf
31/01/2020	7820.000000	-100.000000	-100.000000	-100.000000	-93.330000	-100.000000
30/10/2020	8.700000	720.000000	97.420000	633.330000	-21.050000	496.260000
30/09/2020	-14.810000	400.000000	-59.100000	86.860000	-80.810000	314.940000
30/08/2020	-87.890000	inf	-38.270000	-32.430000	-64.390000	6.100000

In [0]:

```
import seaborn as sns
```

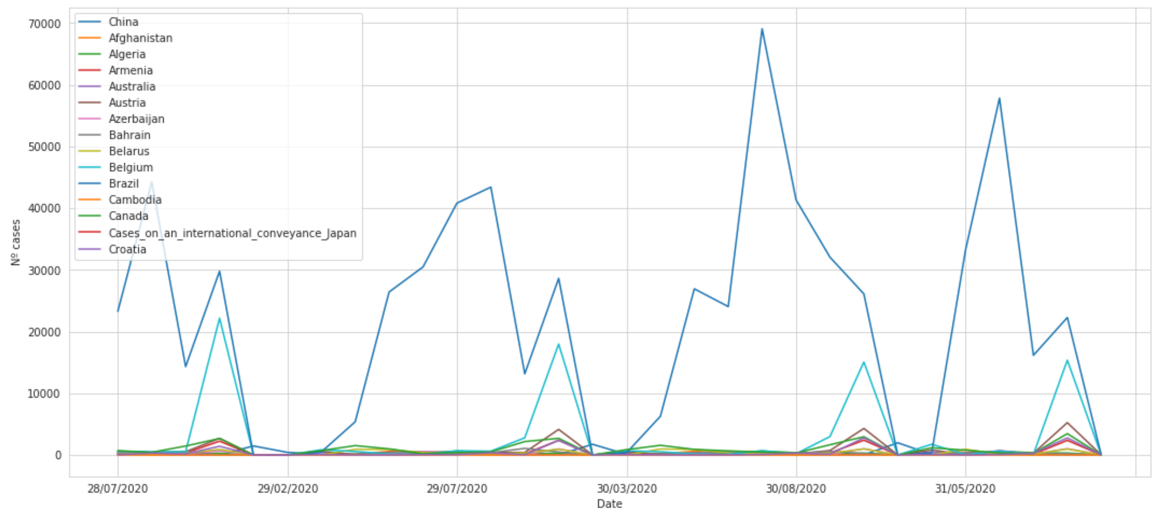
In [0]:

```
last_gi = growth_impact_day.iloc[0].sort_values(ascending = False)

last_gi = pd.DataFrame(data=[last_gi],index=[0],columns=last_gi.index)
plt.figure(figsize=(18,5))
splot = sns.barplot(x='index',y=0,data=last_gi.T.reset_index())
plt.title('Growth rate last day')
plt.ylabel('Pct. change')
plt.xlabel('Country')
plt.xticks(rotation=45)
for p in splot.patches:
    splot.annotate(format(p.get_height(), '.2f'), (p.get_x() + p.get_width() / 2., p.ge
t_height()), ha = 'center', va = 'center', xytext = (0, 10), textcoords = 'offset point
s')
plt.show()
```



In [0]:

```
sns.set_style('whitegrid')
top_impact.set_index('dateRep').tail(30).plot(figsize=(18,8))
plt.ylabel('Nº cases')
plt.xlabel('Date')
plt.show()
```



(Deaths) Growth per day time period (%) $D(t)=D(t-1)*\Delta$

In [0]:

```
Impacted_countries_d = df[['countriesAndTerritories','deaths']].sort_values(by=['death
s'],ascending=True).head(15)['countriesAndTerritories']

top_impact_d = pd.DataFrame()

for country in Impacted_countries_d:
    top_impact_d[country] = df[df['countriesAndTerritories']==country]['deaths']


top_impact_d = top_impact_d.reset_index().sort_values(by='dateRep',ascending=False)

#Normalize


#top_impact_norm = top_impact/top_impact.iloc[0] * 100
```

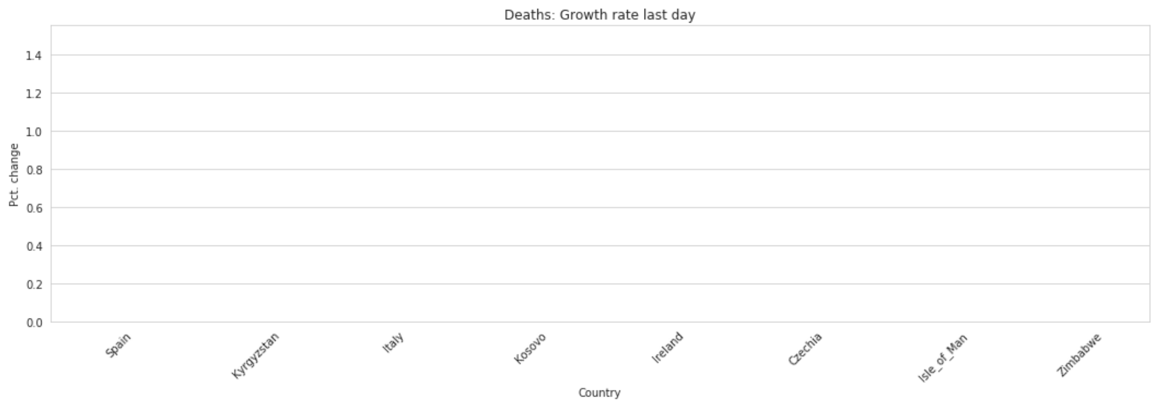
In [0]:

```
growth_impact_d_day = top_impact_d.set_index('dateRep').pct_change(1).reset_index().sor
t_values(by='dateRep',ascending=True)
growth_impact_d_day = round(growth_impact_d_day.set_index('dateRep')*100,2).head(10)
growth_impact_d_day.style.apply(highlight_max_all).apply(highlight_min)
```

	Spain	Kyrgyzstan	Italy	Kosovo	Ireland	Czechia	Isle
dateRep							
01/01/2020	nan	nan	nan	nan	nan	nan	
01/02/2020	nan	nan	-100.000000	nan	nan	nan	
01/03/2020	-100.000000	nan	-99.050000	nan	-100.000000	-100.000000	
01/04/2020	202.140000	nan	194.390000	-100.000000	-59.520000	-22.220000	
01/05/2020	inf	nan	280.000000	inf	inf	800.000000	
01/06/2020	-100.000000	-100.000000	226.090000	-100.000000	-100.000000	0.000000	
01/07/2020	inf	-73.680000	155.560000	-86.670000	inf	-66.670000	
01/08/2020	-100.000000	1800.000000	50.000000	66.670000	nan	200.000000	
01/09/2020	-68.130000	0.000000	-68.420000	inf	-100.000000	-94.740000	
01/10/2020	inf	-80.000000	-93.600000	-100.000000	-80.000000	-89.020000	

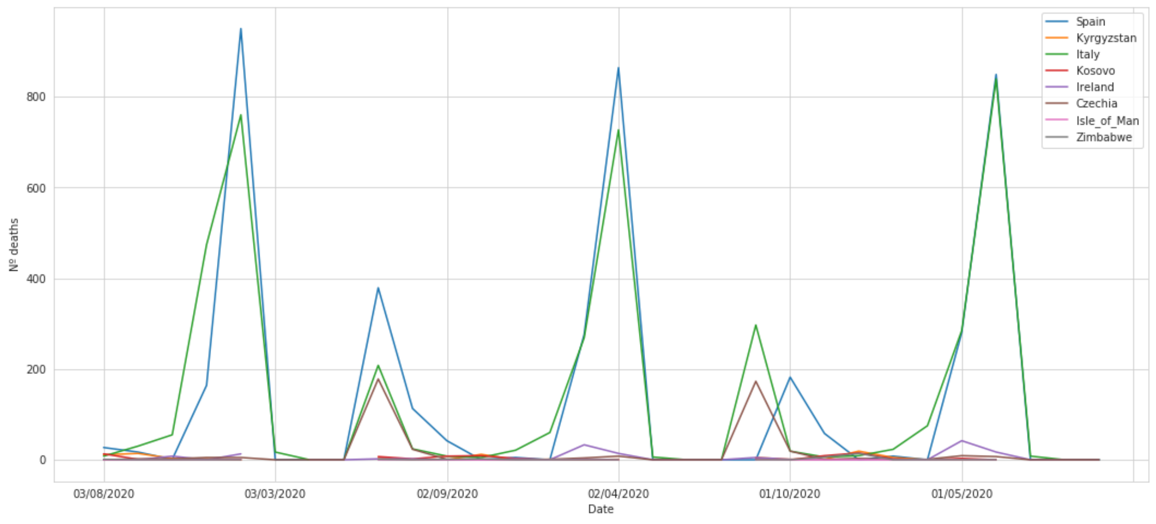
In [0]:

```
last_gi = growth_impact_d_day.iloc[0].sort_values(ascending = False)

last_gi = pd.DataFrame(data=[last_gi],index=[0],columns=last_gi.index)
plt.figure(figsize=(18,5))
splot = sns.barplot(x='index',y=0,data=last_gi.T.reset_index())
plt.title('Deaths: Growth rate last day')
plt.ylabel('Pct. change')
plt.xlabel('Country')
plt.xticks(rotation=45)
for p in splot.patches:
    splot.annotate(format(p.get_height(), '.2f'), (p.get_x() + p.get_width() / 2., p.ge
t_height()), ha = 'center', va = 'center', xytext = (0, 10), textcoords = 'offset point
s')
plt.show()
```



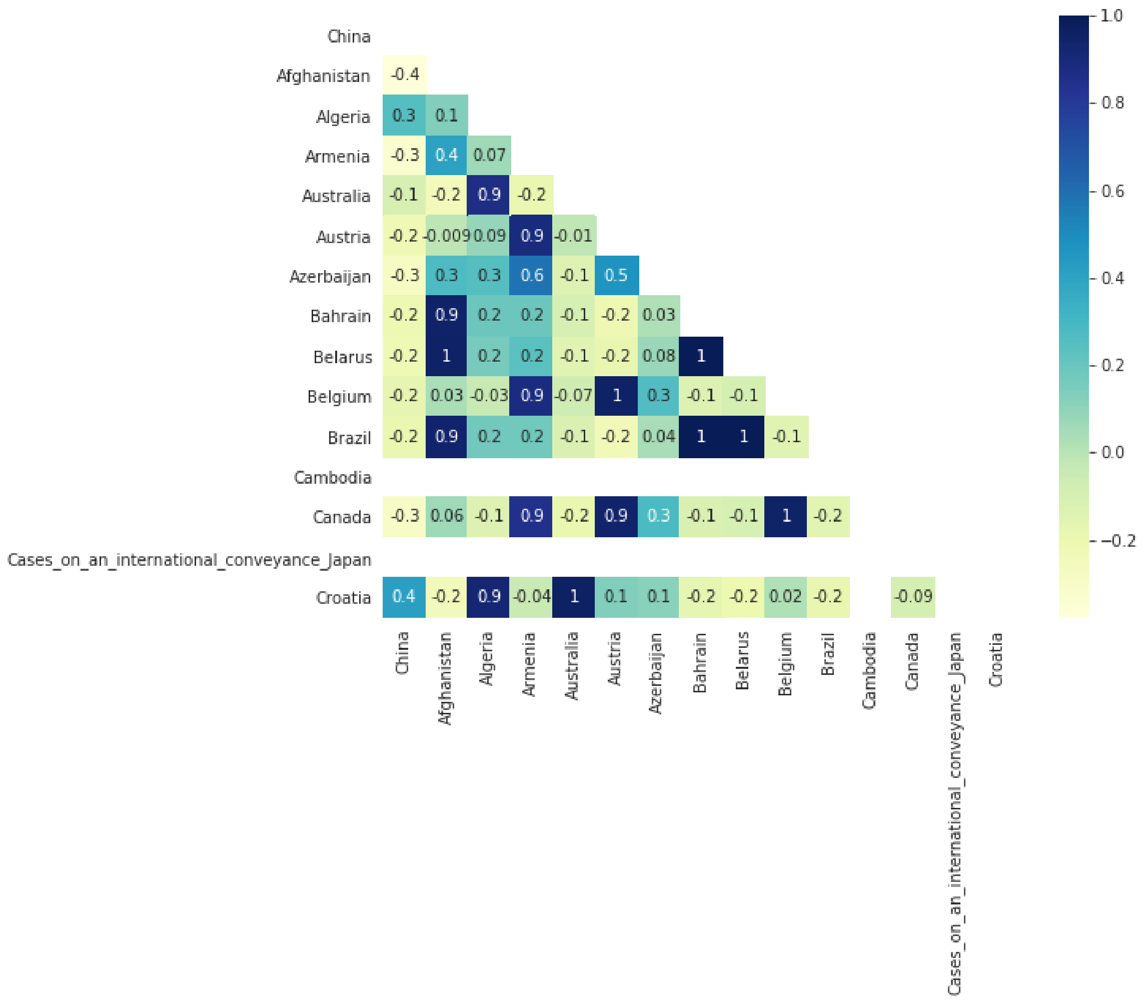
In [0]:

```
sns.set_style('whitegrid')
top_impact_d.set_index('dateRep').tail(30).plot(figsize=(18,8))
plt.ylabel('Nº deaths')
plt.xlabel('Date')
plt.show()
```



In [0]:

```
mask = np.zeros_like(growth_impact_day.corr())
mask[np.triu_indices_from(mask)] = True
plt.figure(figsize=(9,7))
sns.heatmap(growth_impact_day.corr(),mask=mask,cmap='YlGnBu', annot = True, fmt='.1g')
plt.show()
```

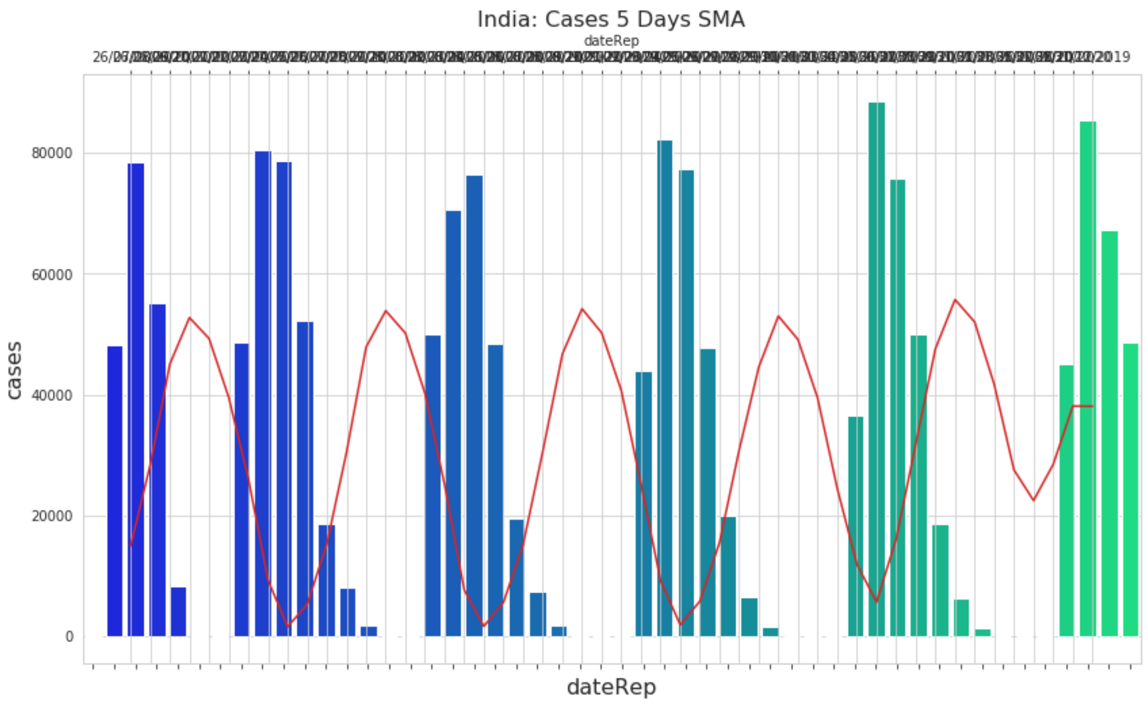


In [0]:

```
India_df = df[df['countriesAndTerritories']=='India']
Brazil_df = df[df['countriesAndTerritories']=='Brazil']
USA_df = df[df['countriesAndTerritories']=='United_States_of_America']

India_df = India_df.sort_values(by='dateRep',ascending=True)
India_df['Cases-5-days-SMA']=India_df['cases'].rolling(window=5).mean()
India_df['Deaths-5-days_SMA']=India_df['deaths'].rolling(window=5).mean()
India_df = India_df.sort_values(by='dateRep',ascending=False)

Brazil_df = Brazil_df.sort_values(by='dateRep',ascending=True)
Brazil_df['Cases-5-days-SMA']=Brazil_df['cases'].rolling(window=5).mean()
Brazil_df['Deaths-5-days_SMA']=Brazil_df['deaths'].rolling(window=5).mean()
Brazil_df = Brazil_df.sort_values(by='dateRep',ascending=False)

USA_df = USA_df.sort_values(by='dateRep',ascending=True)
USA_df['Cases-5-days-SMA']=USA_df['cases'].rolling(window=5).mean()
USA_df['Deaths-5-days_SMA']=USA_df['deaths'].rolling(window=5).mean()
USA_df =USA_df.sort_values(by='dateRep',ascending=False)
```

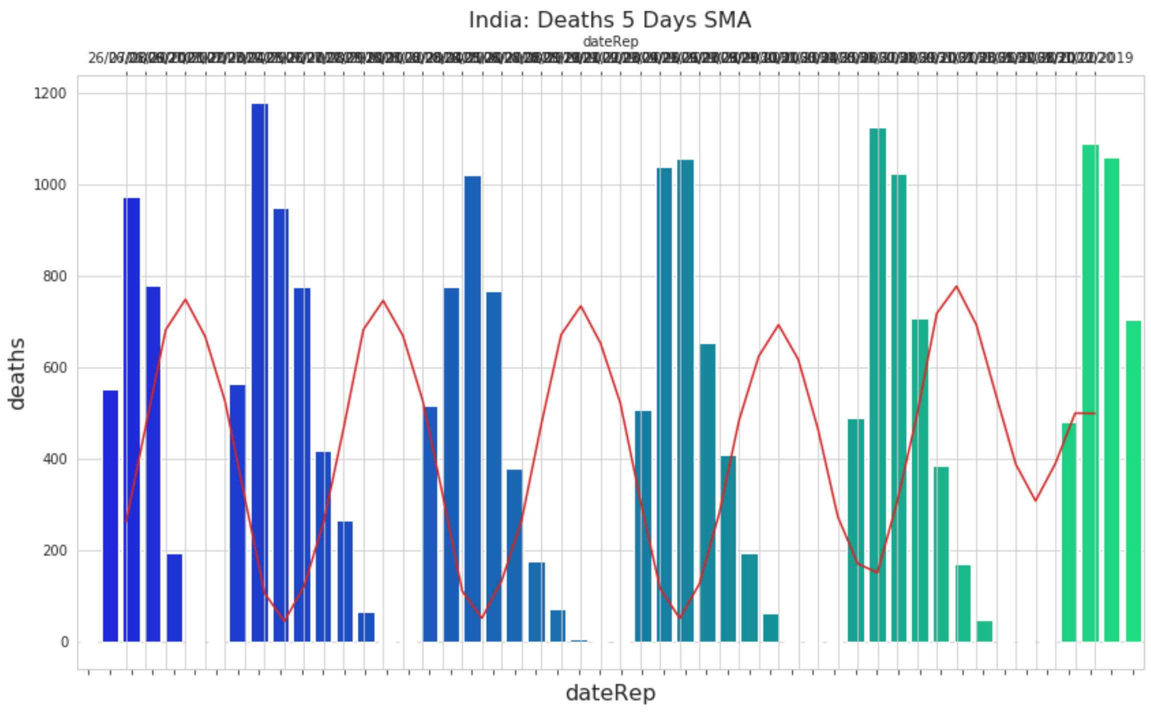
In [0]:

```
#Create combo chart
fig, ax1 = plt.subplots(figsize=(14,8))
color = 'tab:green'
#bar plot creation
ax1.set_title('India: Cases 5 Days SMA', fontsize=16)
ax1.set_xlabel('Date', fontsize=16)
ax1.set_ylabel('Cases', fontsize=16)
ax1 = sns.barplot(x='dateRep', y='cases', data = India_df.reset_index()[:50], palette=
'winter')

ax1.set_xticklabels(
    ax1.get_xticklabels(minor=True),
    rotation=45,
    horizontalalignment='right',
    fontweight='light',
    fontsize='x-large'
)
ax1.tick_params(axis='y')

#specify we want to share the same x-axis
ax2 = ax1.twiny()
color = 'tab:red'
#line plot creation
#ax2.set_ylabel('5 days SMA', fontsize=16)
ax2 = sns.lineplot(x='dateRep', y='Cases-5-days-SMA', data=India_df.reset_index()[:50],
color=color)
ax2.tick_params(axis='y', color=color)
plt.show()
```



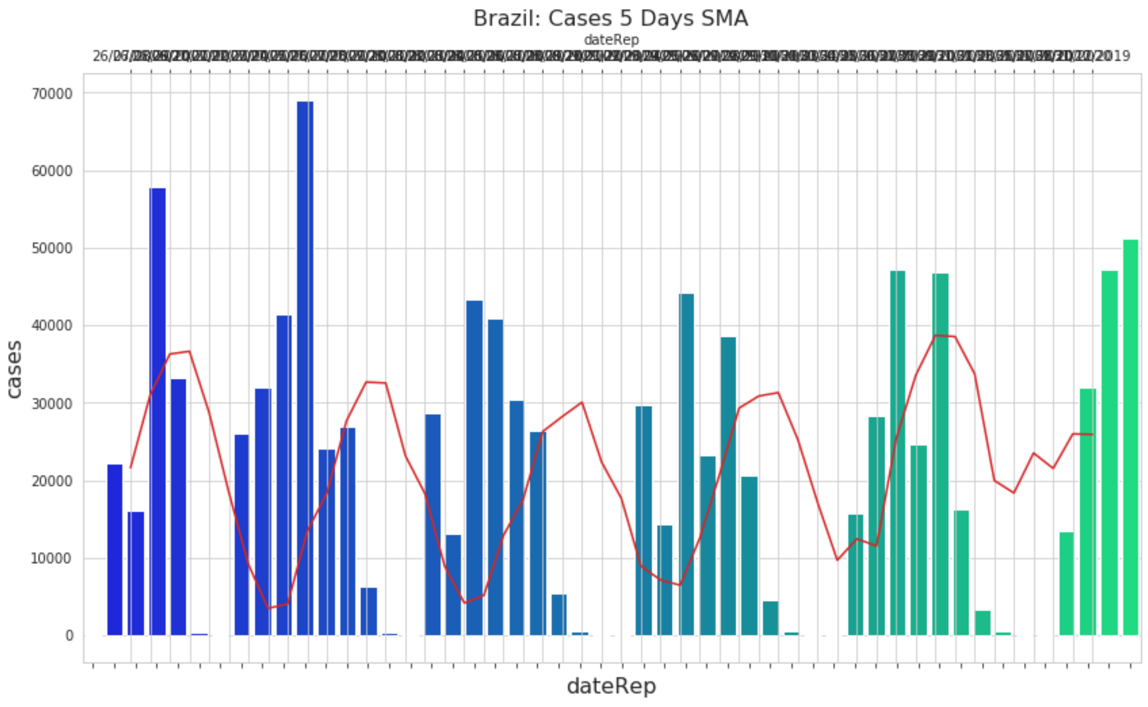
In [0]:

```
fig, ax1 = plt.subplots(figsize=(14,8))
color = 'tab:green'
#bar plot creation
ax1.set_title('India: Deaths 5 Days SMA', fontsize=16)
ax1.set_xlabel('Date', fontsize=16)
ax1.set_ylabel('deaths', fontsize=16)
ax1 = sns.barplot(x='dateRep', y='deaths', data = India_df.reset_index()[:50], palette=
'winter')

ax1.set_xticklabels(
    ax1.get_xticklabels(minor=True),
    rotation=45,
    horizontalalignment='right',
    fontweight='light',
    fontsize='x-large'
)
ax1.tick_params(axis='y')
#specify we want to share the same x-axis
ax2 = ax1.twiny()
color = 'tab:red'
#line plot creation
ax2.set_ylabel('5 days SMA', fontsize=16)
ax2 = sns.lineplot(x='dateRep', y='Deaths-5-days_SMA', data = India_df.reset_index()[:5
0], color=color)
ax2.tick_params(axis='y', color=color)
#show plot
plt.show()
```



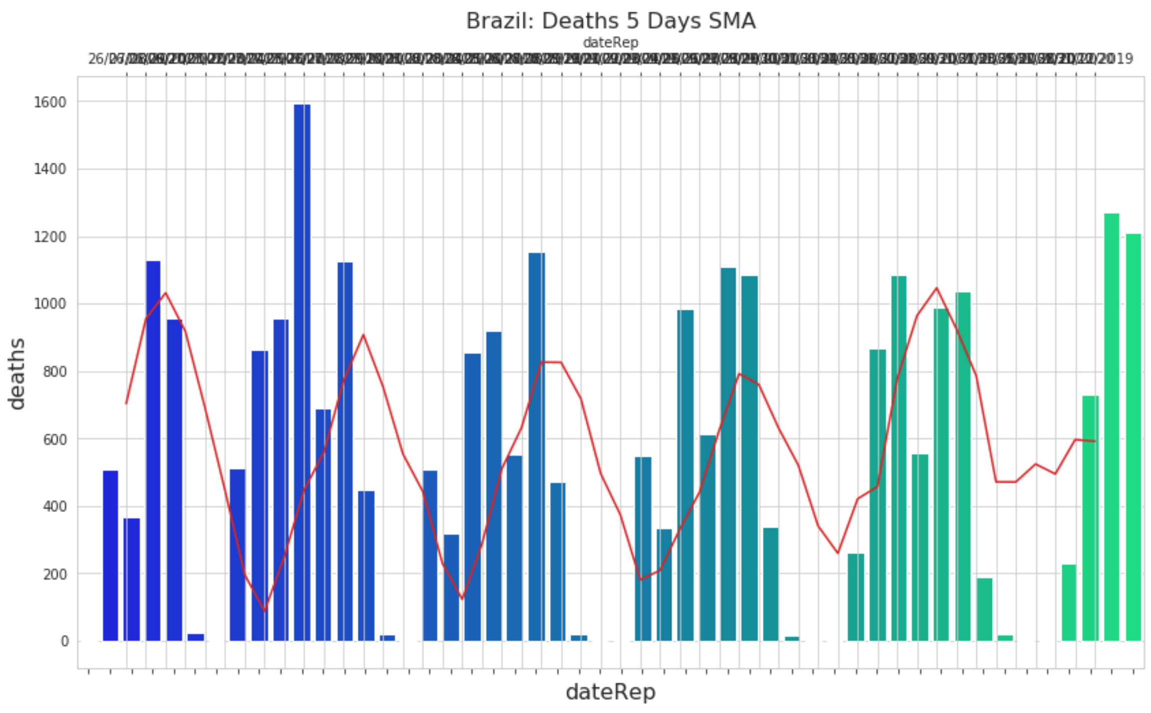
In [0]:

```
#Create combo chart
fig, ax1 = plt.subplots(figsize=(14,8))
color = 'tab:green'
#bar plot creation
ax1.set_title('Brazil: Cases 5 Days SMA', fontsize=16)
ax1.set_xlabel('Date', fontsize=16)
ax1.set_ylabel('Cases', fontsize=16)
ax1 = sns.barplot(x='dateRep', y='cases', data = Brazil_df.reset_index()[:50], palette=
'winter')

ax1.set_xticklabels(
    ax1.get_xticklabels(minor=True),
    rotation=45,
    horizontalalignment='right',
    fontweight='light',
    fontsize='x-large'
)
ax1.tick_params(axis='y')

#specify we want to share the same x-axis
ax2 = ax1.twiny()
color = 'tab:red'
#line plot creation
#ax2.set_ylabel('5 days SMA', fontsize=16)
ax2 = sns.lineplot(x='dateRep', y='Cases-5-days-SMA', data=Brazil_df.reset_index()[:50
], color=color)
ax2.tick_params(axis='y', color=color)
plt.show()
```



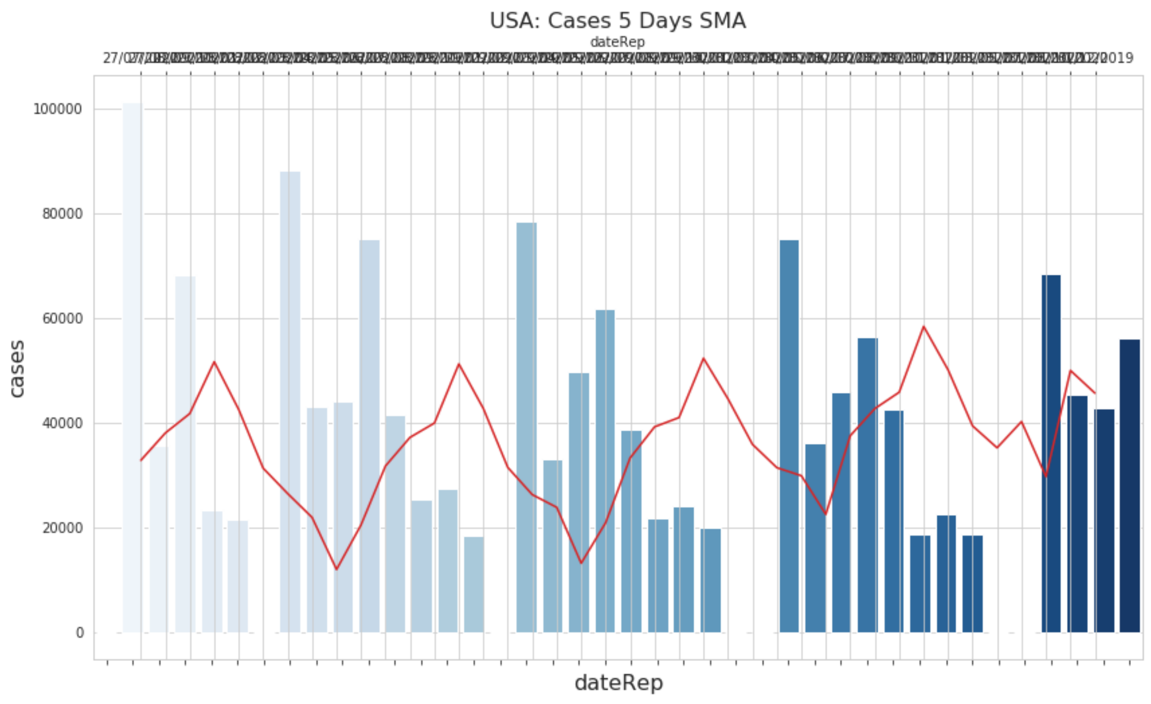
In [0]:

```
fig, ax1 = plt.subplots(figsize=(14,8))
color = 'tab:green'
#bar plot creation
ax1.set_title('Brazil: Deaths 5 Days SMA', fontsize=16)
ax1.set_xlabel('Date', fontsize=16)
ax1.set_ylabel('deaths', fontsize=16)
ax1 = sns.barplot(x='dateRep', y='deaths', data = Brazil_df.reset_index()[:50], palette
='winter')

ax1.set_xticklabels(
    ax1.get_xticklabels(minor=True),
    rotation=45,
    horizontalalignment='right',
    fontweight='light',
    fontsize='x-large'
)
ax1.tick_params(axis='y')
#specify we want to share the same x-axis
ax2 = ax1.twiny()
color = 'tab:red'
#line plot creation
ax2.set_ylabel('5 days SMA', fontsize=16)
ax2 = sns.lineplot(x='dateRep', y='Deaths-5-days_SMA', data = Brazil_df.reset_index()[:
50], color=color)
ax2.tick_params(axis='y', color=color)
#show plot
plt.show()
```



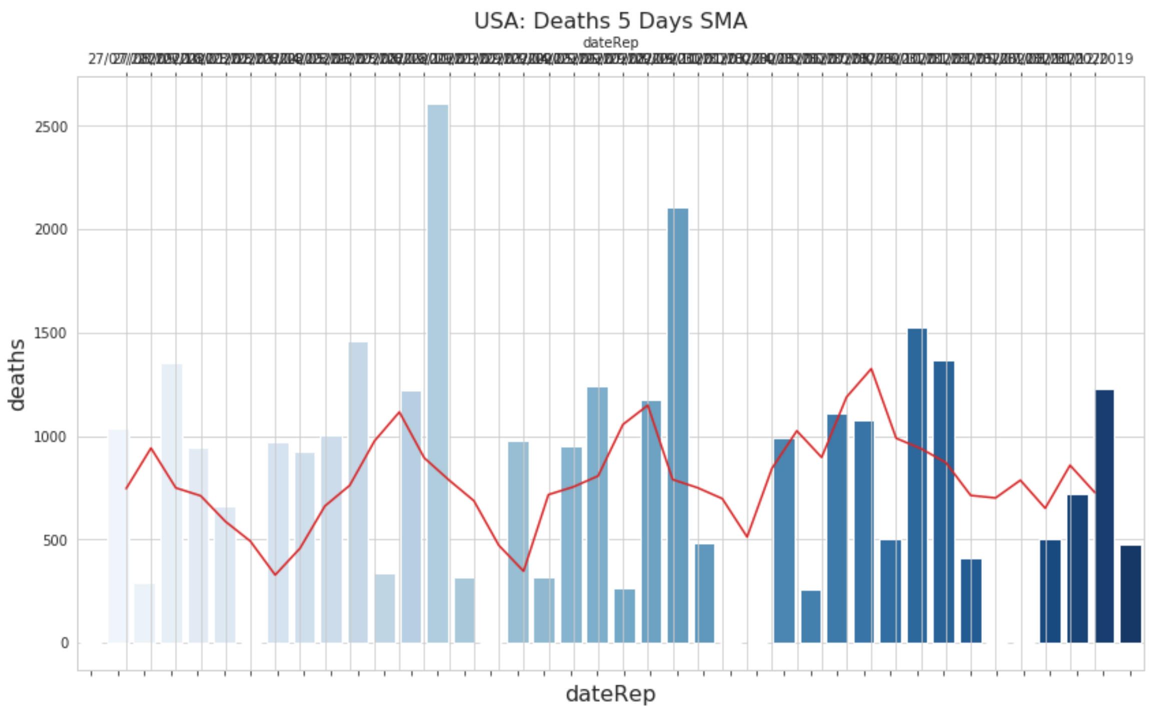
In [0]:

```
#Create combo chart
fig, ax1 = plt.subplots(figsize=(14,8))
color = 'tab:green'
#bar plot creation
ax1.set_title('USA: Cases 5 Days SMA', fontsize=16)
ax1.set_xlabel('Date', fontsize=16)
ax1.set_ylabel('Cases', fontsize=16)
ax1 = sns.barplot(x='dateRep', y='cases', data = USA_df.reset_index()[:40], palette='Bl
ues')

ax1.set_xticklabels(
    ax1.get_xticklabels(minor=True),
    rotation=45,
    horizontalalignment='right',
    fontweight='light',
    fontsize='x-large'
)
ax1.tick_params(axis='y')
#specify we want to share the same x-axis
ax2 = ax1.twiny()
color = 'tab:red'
#line plot creation
ax2.set_ylabel('5 days SMA', fontsize=16)
ax2 = sns.lineplot(x='dateRep', y='Cases-5-days-SMA', data = USA_df.reset_index()[:40],
color=color)
ax2.tick_params(axis='y', color=color)
#show plot
plt.show()
```



In [0]:

```python
#Create combo chart
fig, ax1 = plt.subplots(figsize=(14,8))
color = 'tab:green'
#bar plot creation
ax1.set_title('USA: Deaths 5 Days SMA', fontsize=16)
ax1.set_xlabel('Date', fontsize=16)
ax1.set_ylabel('Deaths', fontsize=16)
ax1 = sns.barplot(x='dateRep', y='deaths', data = USA_df.reset_index()[:40], palette='B
lues')

ax1.set_xticklabels(
    ax1.get_xticklabels(minor=True),
    rotation=45,
    horizontalalignment='right',
    fontweight='light',
    fontsize='x-large'
)
ax1.tick_params(axis='y')
#specify we want to share the same x-axis
ax2 = ax1.twiny()
color = 'tab:red'
#line plot creation
ax2.set_ylabel('5 days SMA', fontsize=16)
ax2 = sns.lineplot(x='dateRep', y='Deaths-5-days_SMA', data = USA_df.reset_index()[:40
], color=color)
ax2.tick_params(axis='y', color=color)
#show plot
plt.show()
```



In [0]:

```python
India_df =spark.createDataFrame(India_df)
USA_df =spark.createDataFrame(USA_df)
Brazil_df =spark.createDataFrame(Brazil_df)
```

In [0]:

```
India_df.write.mode("overwrite").saveAsTable("India_Covid")
USA_df.write.mode("overwrite").saveAsTable("USA_Covid")
Brazil_df.write.mode("overwrite").saveAsTable("Brazil_Covid")
```

Top Countries

In [0]:

```
import plotly.graph_objects as go
import plotly.express as px
import matplotlib.pyplot as plt

df.loc[: , ['countriesAndTerritories', 'cases', 'deaths']].groupby(['countriesAndTerritories']).max().sort_values(by='cases',ascending=False).reset_index()[:15].style.background_gradient(cmap='rainbow')
```

	countriesAndTerritories	cases	deaths
0	United_States_of_America	130623	4928
1	India	97894	2003
2	France	86852	2004
3	Brazil	69074	1595
4	Spain	55019	1623
5	Italy	39809	971
6	Chile	36179	1057
7	Poland	27875	445
8	United_Kingdom	26687	1224
9	Germany	23399	315
10	Belgium	22176	321
11	Switzerland	21842	93
12	Russia	20582	389
13	Kazakhstan	19246	324
14	Argentina	18326	3351

In [0]:

```
top_countries = df.loc[: , ['countriesAndTerritories', 'cases', 'deaths']].groupby(['countriesAndTerritories']).max().sort_values(by='cases',ascending=False).reset_index()
```

In [0]:

```
display(top_countries)
```

countriesAndTerritories	cases	deaths
United_States_of_America	130623	4928
India	97894	2003
France	86852	2004
Brazil	69074	1595
Spain	55019	1623
Italy	39809	971
Chile	36179	1057
Poland	27875	445
United_Kingdom	26687	1224
Germany	23399	315
Belgium	22176	321

In [0]:

```
top_countries =spark.createDataFrame(top_countries)
```

In [0]:

```
top_countries.write.mode("overwrite").saveAsTable("top_countries")
```

Saving main dataframe into a table.

AFRICA

In [0]:

```
africa_df = df[df['continentExp']=='Africa']
```

In [0]:

```
display(africa_df)
```

day	month	year	cases	deaths	countriesAndTerritories	geoId	countryterritoryCode	po
8	11	2020	581	12	Algeria	DZ	DZA	4
7	11	2020	1273	25	Algeria	DZ	DZA	4
6	11	2020	0	0	Algeria	DZ	DZA	4
5	11	2020	548	10	Algeria	DZ	DZA	4
4	11	2020	405	9	Algeria	DZ	DZA	4
3	11	2020	302	7	Algeria	DZ	DZA	4
2	11	2020	330	9	Algeria	DZ	DZA	4
1	11	2020	291	8	Algeria	DZ	DZA	4
31	10	2020	319	7	Algeria	DZ	DZA	4

In [0]:

```
africa_df = spark.createDataFrame(africa_df)
```

In [0]:

```
africa_df.write.mode("overwrite").saveAsTable("Africa_covidtable")
```

ASIA

In [0]:

```
asia_df = df[df['continentExp']=='Asia']
display(asia_df)
```

day	month	year	cases	deaths	countriesAndTerritories	geoId	countryterritoryCode	p
8	11	2020	126	6	Afghanistan	AF	AFG	3
7	11	2020	58	2	Afghanistan	AF	AFG	3
6	11	2020	40	0	Afghanistan	AF	AFG	3
5	11	2020	121	6	Afghanistan	AF	AFG	3
4	11	2020	86	4	Afghanistan	AF	AFG	3
3	11	2020	95	3	Afghanistan	AF	AFG	3
2	11	2020	132	5	Afghanistan	AF	AFG	3
1	11	2020	76	0	Afghanistan	AF	AFG	3
31	10	2020	157	4	Afghanistan	AF	AFG	3

In [0]:

```
asia_df = spark.createDataFrame(asia_df)
```

In [0]:

```
asia_df.write.mode("overwrite").saveAsTable("Asia_covidtable")
```

Europe

In [0]:

```
europe_df = df[df['continentExp']=='Europe']
display(europe_df)
```

day	month	year	cases	deaths	countriesAndTerritories	geoId	countryterritoryCode	pc
8	11	2020	495	8	Albania	AL	ALB	
7	11	2020	489	6	Albania	AL	ALB	
6	11	2020	421	7	Albania	AL	ALB	
5	11	2020	396	4	Albania	AL	ALB	
4	11	2020	381	5	Albania	AL	ALB	
3	11	2020	321	9	Albania	AL	ALB	
2	11	2020	327	9	Albania	AL	ALB	
1	11	2020	241	7	Albania	AL	ALB	
31	10	2020	319	3	Albania	AL	ALB	

In [0]:

```
europe_df = spark.createDataFrame(europe_df)
```

In [0]:

```
europe_df.write.mode("overwrite").saveAsTable("Europe_covidtable")
```

Oceania

In [0]:

```
Oceania_df = df[df['continentExp']=='Oceania']
display(Oceania_df)
```

day	month	year	cases	deaths	countriesAndTerritories	geoId	countryterritoryCode	po
8	11	2020	7	0	Australia	AU	AUS	
7	11	2020	12	0	Australia	AU	AUS	
6	11	2020	11	0	Australia	AU	AUS	
5	11	2020	12	0	Australia	AU	AUS	
4	11	2020	8	0	Australia	AU	AUS	
3	11	2020	7	0	Australia	AU	AUS	
2	11	2020	5	0	Australia	AU	AUS	
1	11	2020	8	0	Australia	AU	AUS	
31	10	2020	13	0	Australia	AU	AUS	

In [0]:

```
Oceania_df = spark.createDataFrame(Oceania_df)
```

In [0]:

```
Oceania_df.write.mode("overwrite").saveAsTable("Oceania_covidtable")
```

America

In [0]:

```
America_df = df[df['continentExp']=='America']
display(America_df)
```

day	month	year	cases	deaths	countriesAndTerritories	geoId	countryterritoryCode	po
8	11	2020	0	0	Anguilla	AI	AIA	
7	11	2020	0	0	Anguilla	AI	AIA	
6	11	2020	0	0	Anguilla	AI	AIA	
5	11	2020	0	0	Anguilla	AI	AIA	
4	11	2020	0	0	Anguilla	AI	AIA	
3	11	2020	0	0	Anguilla	AI	AIA	
2	11	2020	0	0	Anguilla	AI	AIA	
1	11	2020	0	0	Anguilla	AI	AIA	
31	10	2020	0	0	Anguilla	AI	AIA	

In [0]:

```
America_df = spark.createDataFrame(America_df)
```

In [0]:

```
America_df.write.mode("overwrite").saveAsTable("America_covidtable")
```

In [4]:

```
print("VINITA VERMA - 06917704418" )
print("MCA -5B")
```

```
VINITA VERMA - 06917704418
MCA -5B
```

In [ ]:

```
```

In [ ]:

```
```