ZepplinTest

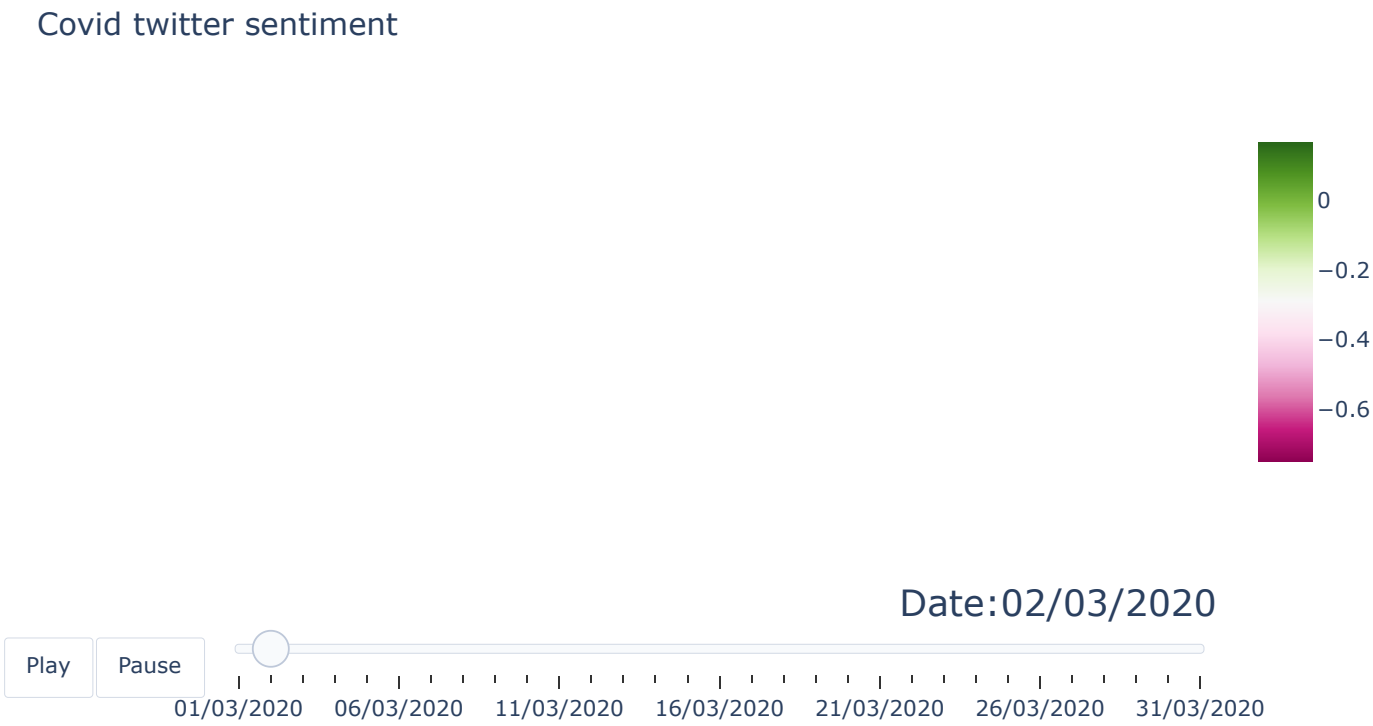
```
%sh pip install pandas
```

```
%sh pip install plotly
```

```
%pyspark

#########################################################################################################################################################################
#
#                PLOT CHOROPLETH MAP SHOWING VARIATION IN SENTIMENT BY COUNTRY OVER TIME
#
#########################################################################################################################################################################

import plotly
from plotly.graph_objs import Scatter, Layout, Choropleth, Frame
from pyspark import SparkConf
from pyspark.sql import SparkSession
from pyspark.sql.types import StringType
from pyspark.sql.functions import date_format

def create_spark_session():
    conf = SparkConf()
    spark = SparkSession.builder \
        .config(conf=conf) \
        .master("local[*]") \
        .appName("sentiment_analysis") \
        .enableHiveSupport()\
        .getOrCreate()

    return spark


def plot(plot_dic, height=800, width=800, **kwargs):
    kwargs['output_type'] = 'div'
    plot_str = plotly.offline.plot(plot_dic, **kwargs)
    print('%%angular <div style="height: %ipx; width: %spx"> %s </div>' % (height, width, plot_str))

spark_session = create_spark_session()
data = spark_session.sql("select datestring, countryterritorycode,  avg(avgsentiment) as sentiment from dailyCountryCovidSentiment group by countryterritorycode, dat
data_df = data.toPandas()
data_df = data_df.set_index(['datestring'])


fig_dict = {
```

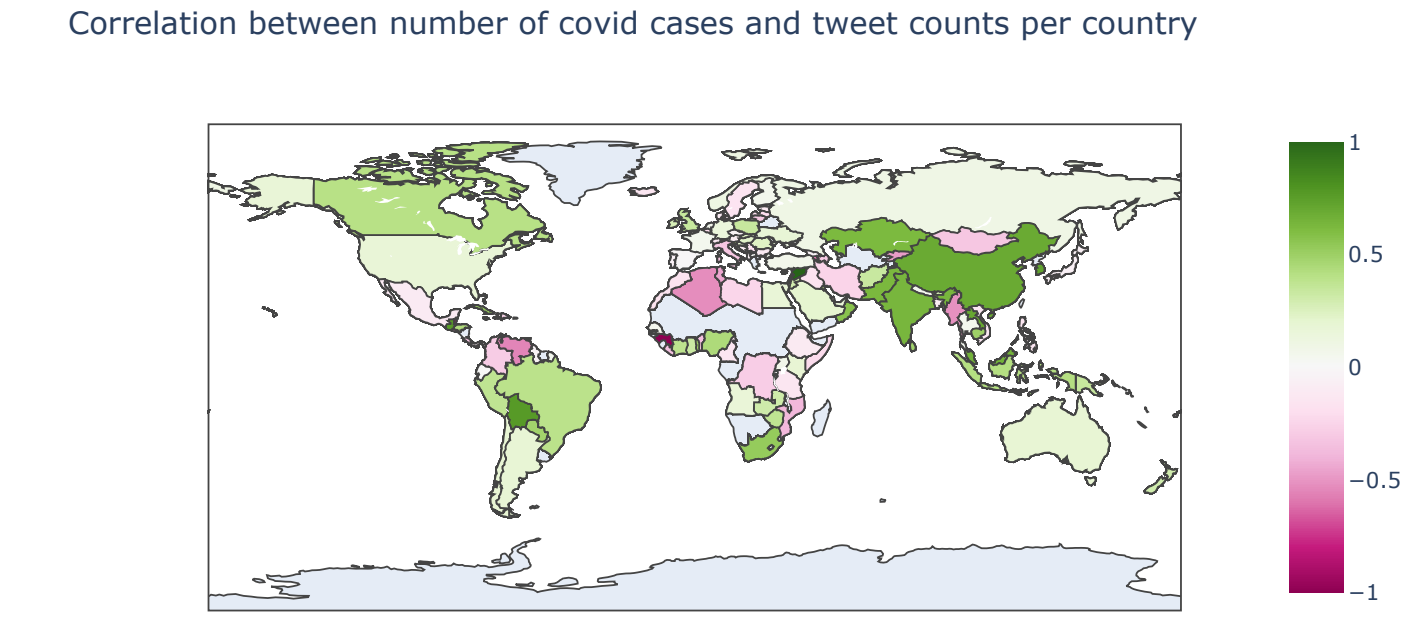
```
    "data": [],
    "layout": {},
    "frames": []
}

# fill in most of layout
fig_dict["layout"]["title"] = "Covid twitter sentiment"
fig_dict["layout"]["updatemenus"] = [
    {
        "buttons": [
            {
                "args": [None, {"frame": {"duration": 500, "redraw": True},
                                "fromcurrent": True, "transition": {"duration": 300,
                                                                    "easing": "quadratic-in-out"}}],
                "label": "Play",
                "method": "animate"
            },
            {
                "args": [[None], {"frame": {"duration": 0, "redraw": True},
                                  "mode": "immediate",
                                  "transition": {"duration": 0}}],
                "label": "Pause",
                "method": "animate"
            }
        ],
        "direction": "left",
        "pad": {"r": 10, "t": 87},
        "showactive": False,
        "type": "buttons",
        "x": 0.1,
        "xanchor": "right",
        "y": 0,
        "yanchor": "top"
    }
]

sliders_dict = {
    "active": 0,
    "yanchor": "top",
    "xanchor": "left",
    "currentvalue": {
        "font": {"size": 20},
        "prefix": "Date:",
        "visible": True,
        "xanchor": "right"
    },
    "transition": {"duration": 300, "easing": "cubic-in-out"},
    "pad": {"b": 10, "t": 50},
    "len": 0.9,
    "x": 0.1,
    "y": 0,
    "steps": []
}

# make data
```

```
fig_dict["data"].append(Choropleth(locations=data_df["countryterritorycode"].loc["01/03/2020"], # Spatial coordinates
                    z = data_df["sentiment"].loc["01/03/2020"]))

# make frames
for idx in sorted(data_df.index.unique()):
    frame = {"data":  [Choropleth(locations=data_df["countryterritorycode"].loc[idx],
             z=data_df["sentiment"].loc[idx])], "name": str(idx)}

    fig_dict["frames"].append(frame)
    slider_step = {"args": [
        [idx],
        {"frame": {"duration": 300, "redraw": True},
         "mode": "immediate",
         "transition": {"duration": 300}}
    ],
        "label": idx,
        "method": "animate"}
    sliders_dict["steps"].append(slider_step)


fig_dict["layout"]["sliders"] = [sliders_dict]


plot(fig_dict)
```

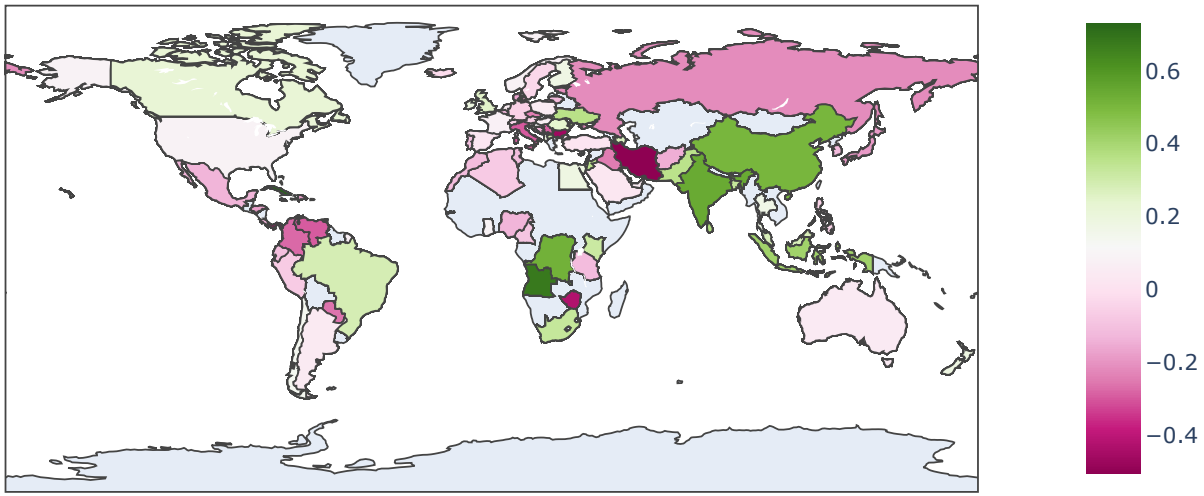


```
%pyspark

#################################################################################################################################################################
#
#               PLOT CORRELATION STATS FOR CASES VS TWEET COUNTS PER COUNTRY
#
#################################################################################################################################################################

from pyspark import SparkConf
from pyspark.sql import SparkSession
import plotly
from plotly.graph_objs import Scatter, Layout, Choropleth, Frame, Bar

def plot(plot_dic, height=800, width=800, **kwargs):
    kwargs['output_type'] = 'div'
    plot_str = plotly.offline.plot(plot_dic, **kwargs)
    print('%%angular <div style="height: %ipx; width: %spx"> %s </div>' %
          (height, width, plot_str))

def create_spark_session():
    conf = SparkConf()
    spark = SparkSession.builder \
        .config(conf=conf) \
        .master("local[*]") \
        .appName("sentiment_analysis") \
        .enableHiveSupport()\
        .getOrCreate()
    return spark

spark_session = create_spark_session()

country_sentiment_data = spark_session.sql("select t2.countryterritorycode, t1.cases_tweet_correlation from correlationdata t1, dailycountrycovidsentiment t2 where t
```

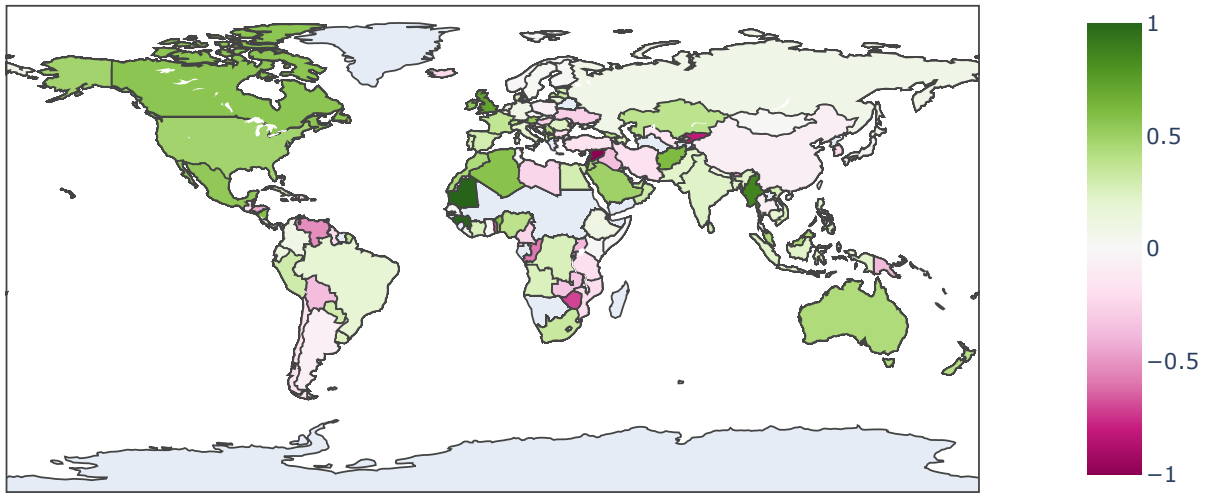
```
plot_data = country_sentiment_data.toPandas()

plot({
    "data": [
        Choropleth(locations=plot_data['countryterritorycode'], z=plot_data['cases_tweet_correlation'])
    ],
    "layout": Layout(
        title="Correlation between number of covid cases and tweet counts per country"
    )
})
```



Correlation between number of covid deaths and tweet counts per country

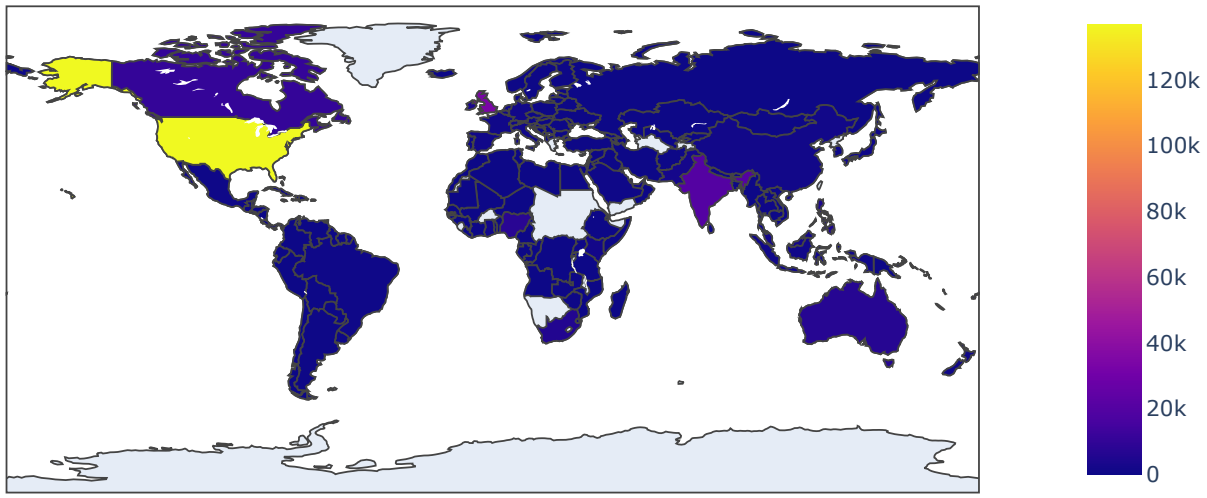


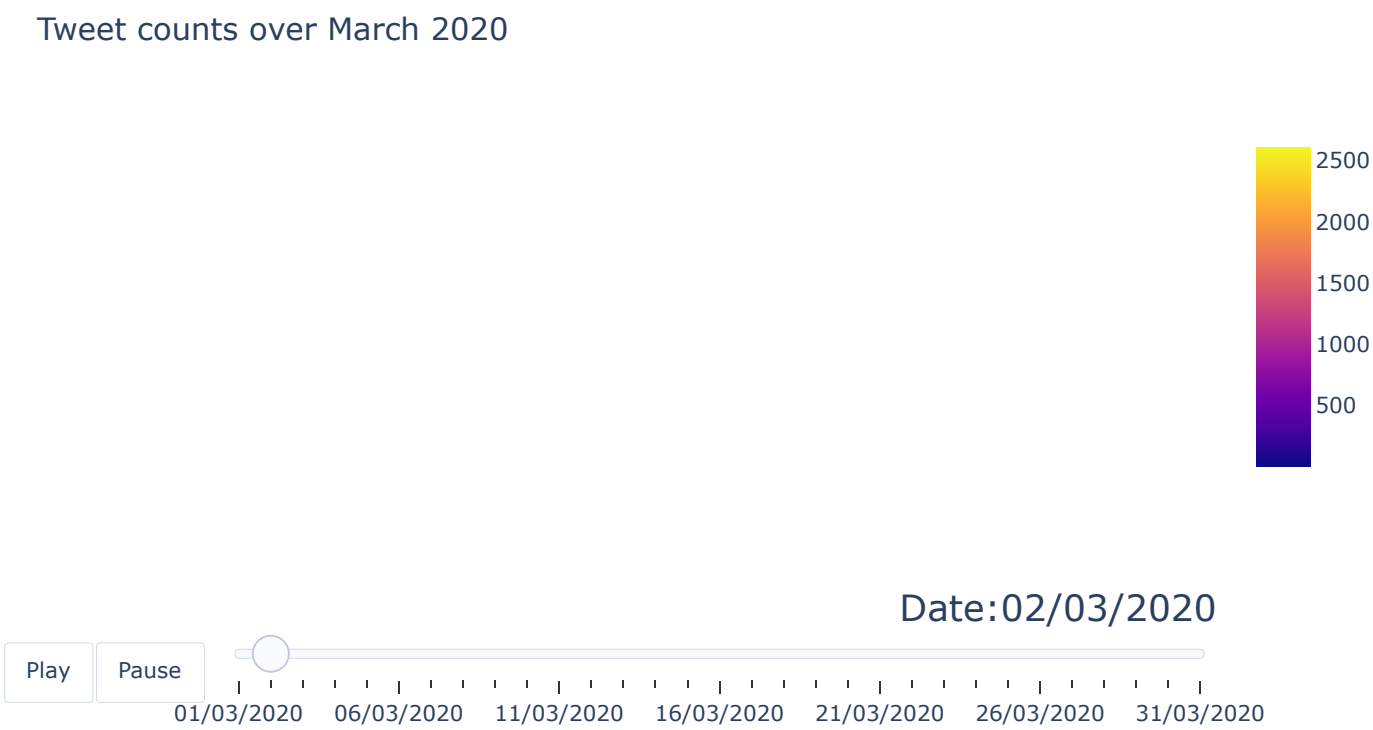
Correlation between number of covid cases and tweet sentiment per country



Correlation between number of covid deaths and tweet sentiment per country

Tweets per country

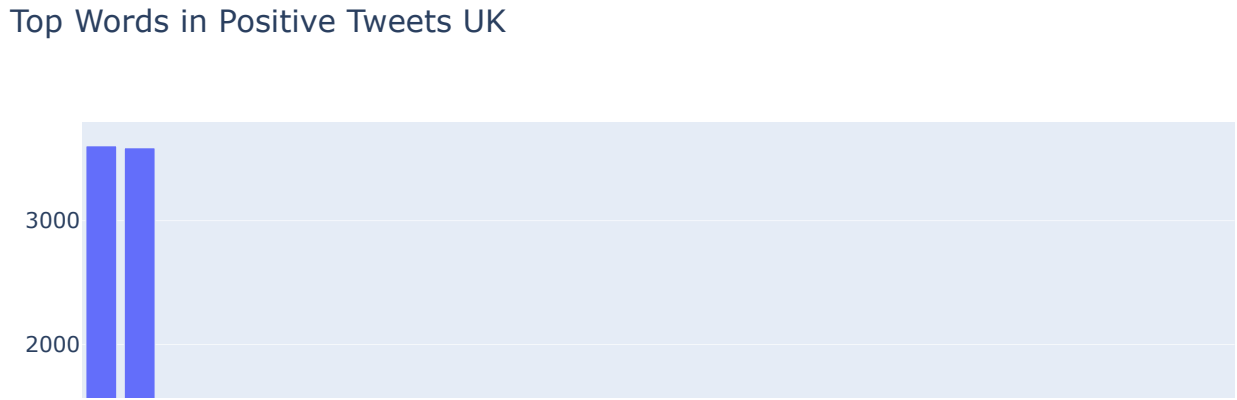


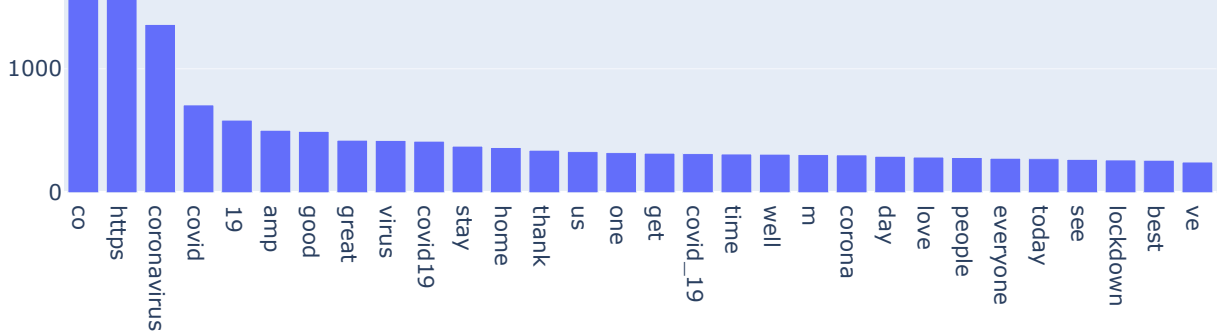

Tweet counts over March 2020
2500
2000
1500
1000
500
Date:02/03/2020
Play
Pause
01/03/2020
06/03/2020
11/03/2020
16/03/2020
21/03/2020
26/03/2020
31/03/2020

```
%pyspark

#################################################################################################################################################
#
```

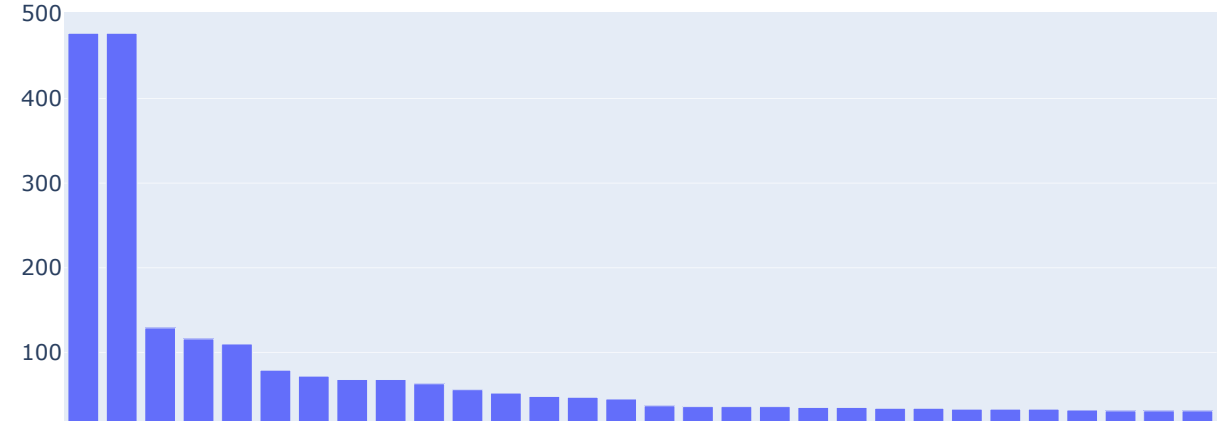
```
#               PLOTS SHOWING TOP WORDS PER POSITIVE TWEEKS UK & IRELAND
#
#########################################################################################################################################################################

from pyspark.ml.feature import StopWordsRemover
from pyspark.ml.feature import RegexTokenizer
from pyspark.sql.functions import date_format, to_date, col, explode, split, desc, udf, asc
from pyspark.sql.types import StringType, IntegerType
from pyspark.sql import SparkSession
from pyspark import SparkConf
from plotly.graph_objs import Scatter, Layout, Choropleth, Frame, Bar
import plotly

def create_spark_session():
    conf = SparkConf()
    spark = SparkSession.builder \
        .config(conf=conf) \
        .master("local[*]") \
        .appName("sentiment_analysis") \
        .enableHiveSupport()\
        .getOrCreate()
    return spark


def plot(plot_dic, height=800, width=800, **kwargs):
    kwargs['output_type'] = 'div'
    plot_str = plotly.offline.plot(plot_dic, **kwargs)
    print('%%angular <div style="height: %ipx; width: %spx"> %s </div>' %
          (height, width, plot_str))


spark_session = create_spark_session()
input_df = spark_session.sql(
    "select t2.full_text, t1.sentiment_label, t1.country  from sentimentsummary t1 join hydratedtweets t2 on t1.tweet_id = t2.tweet_id where t1.country in ('GB', 'IE
        ('positive')")

hydrated_tweets = spark_session.sql("select * from hydratedtweets")

tweets_to_hydrate = spark_session.sql("select t1.Tweet_ID from sentimentSummary t1 left join hydratedTweets t2 on t2.Tweet_ID = t1.Tweet_ID where t1.country in ('GB'
    IS NULL")

sentiment_data = spark_session.sql("select * from sentimentsummary where country in ('GB', 'IE')")

# print('Size input df')
# print(input_df.count())

# print('Hydrated tweet size')
# print(hydrated_tweets.count())

# print('UK & IE tweets to hydrate size')
# print(tweets_to_hydrate.count())

# print('UK & IE total tweets')
# print(sentiment_data.count())
```

```
regexTokenizer = RegexTokenizer(
    inputCol="full_text", outputCol="words", pattern="\\W")
countTokens = udf(lambda words: len(words), IntegerType())

regexTokenized = regexTokenizer.transform(input_df)

remover = StopWordsRemover(inputCol="words", outputCol="filtered")
filteredTokens = remover.transform(regexTokenized)
word_count_df = filteredTokens.withColumn("word", explode(
    col('filtered'))).groupBy('word', 'country').count().sort(asc('count'))
uk_word_count_df = word_count_df.filter(
    word_count_df.country.isin('GB')).sort(desc('count'))
irl_word_count_df = word_count_df.filter(
    word_count_df.country.isin('IE')).sort(desc('count'))

uk_df = uk_word_count_df.toPandas()
plot_uk_data = uk_df.head(30)

ie_df = irl_word_count_df.toPandas()
plot_ie_data = ie_df.head(30)

plot({
    "data": [
        Bar(x=plot_uk_data['word'], y=plot_uk_data['count'])
    ],
    "layout": Layout(
        title="Top Words in Positive Tweets UK"
    )
})

plot({
    "data": [
        Bar(x=plot_ie_data['word'], y=plot_ie_data['count'])
    ],
    "layout": Layout(
        title="Top Words in Positive Tweets Ireland"
    )
})
```

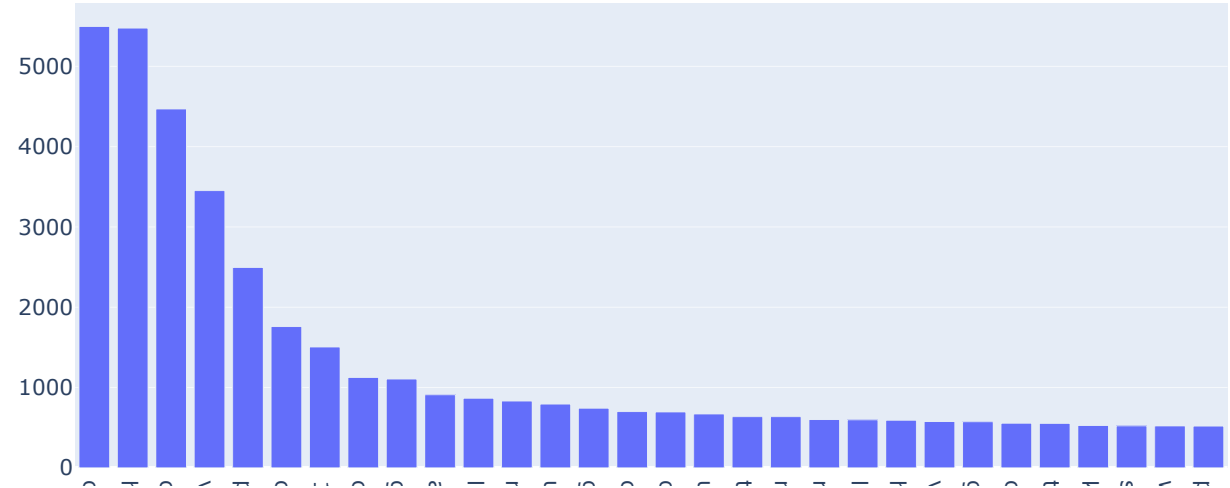




Top Words in Positive Tweets Ireland

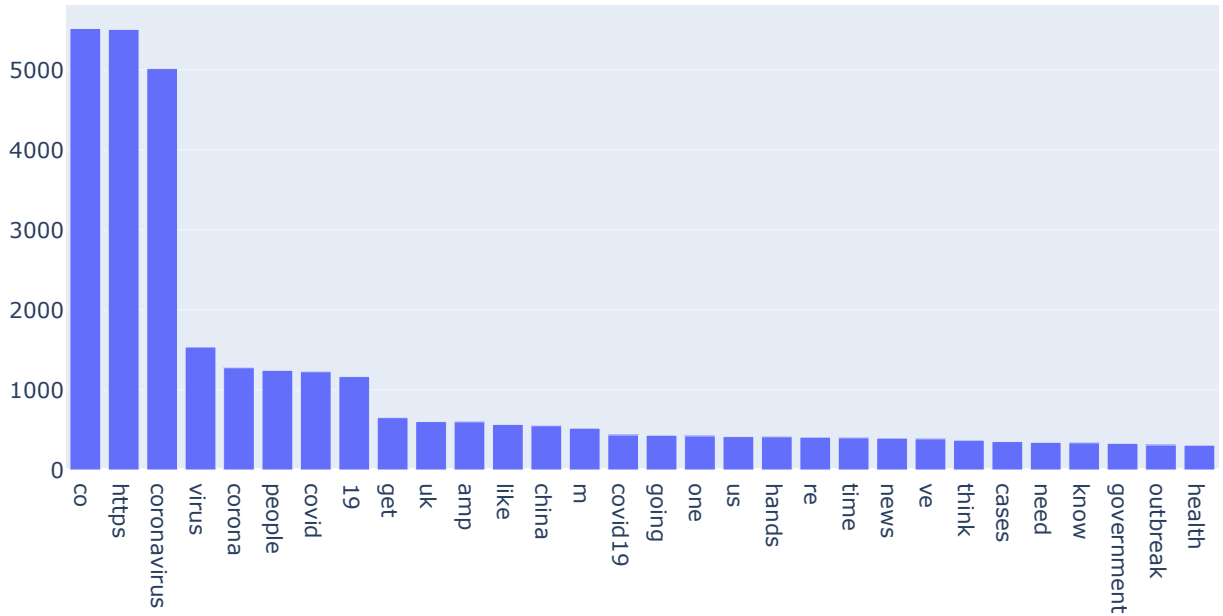


Top Words in Negative Tweets UK

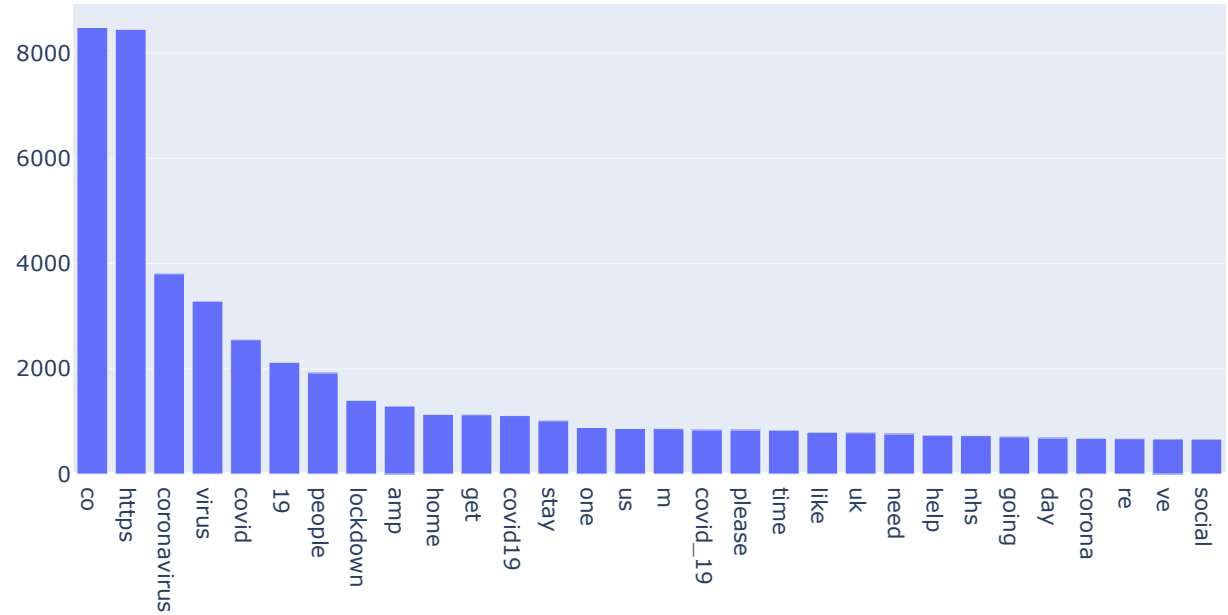


Top Words in Negative Tweets Ireland

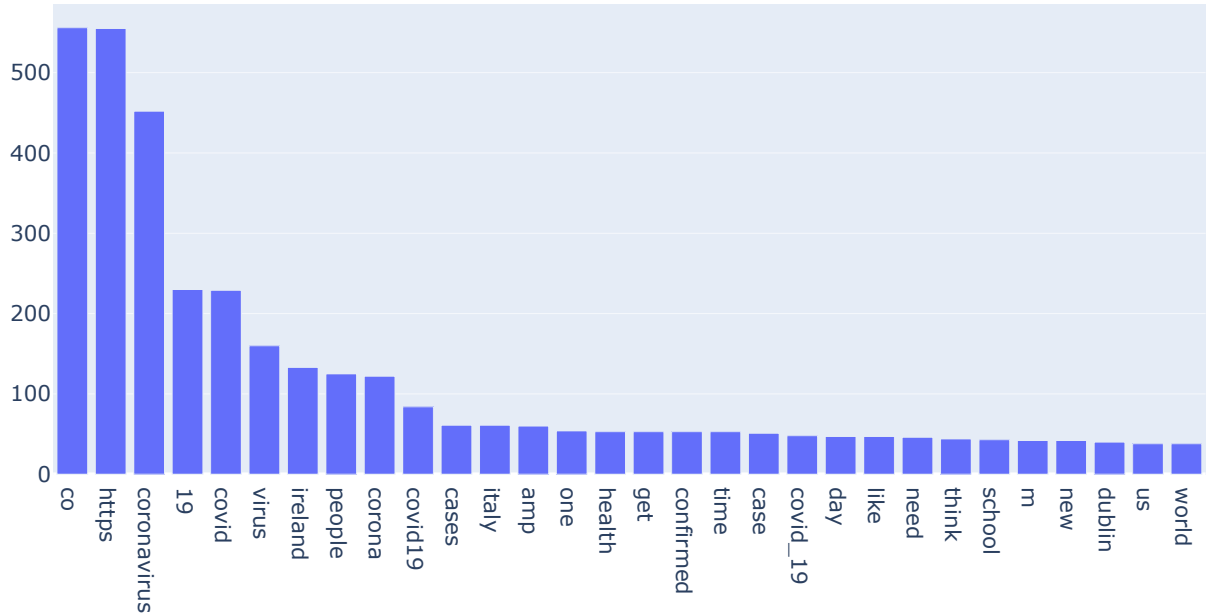
Top 30 Words Pre-lockdown UK



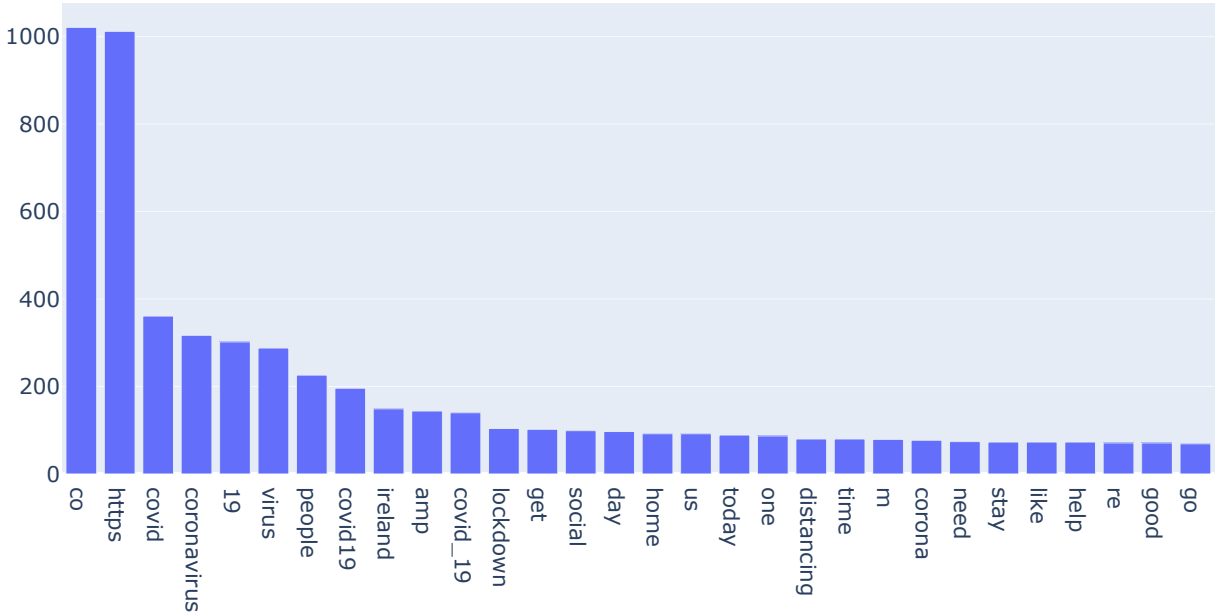
Top 30 Words Post-lockdown UK



Top 30 Words Pre-lockdown Ireland

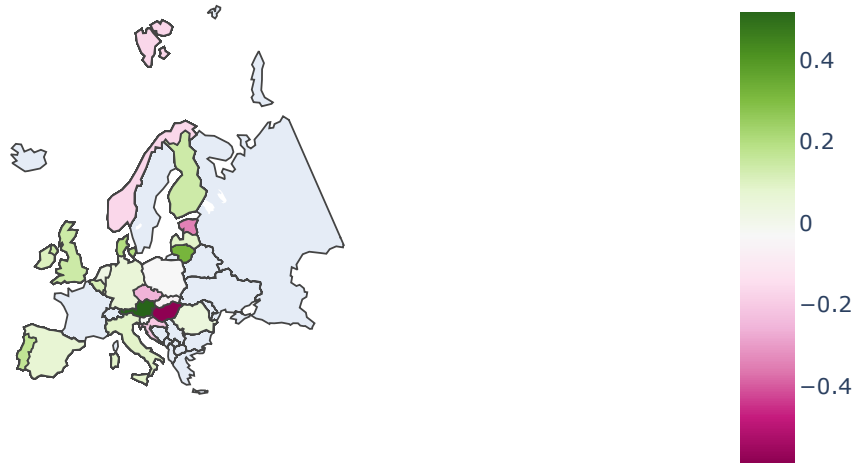


Top 30 Words Post-lockdown Ireland



```
%pyspark

from pyspark.sql.functions import date_format
from pyspark.sql.types import StringType
from pyspark.sql import SparkSession
from pyspark import SparkConf
from plotly.graph_objs import Scatter, Layout, Choropleth, Frame
import plotly

##########################################################################################################################################################
#
#               PLOT CHOROPLETH MAP SHOWING SENTIMENT DIFFERENT PRE- AND POST- RESPONSE
#
##########################################################################################################################################################


def create_spark_session():
    conf = SparkConf()
    spark = SparkSession.builder \
        .config(conf=conf) \
        .master("local[*]") \
        .appName("sentiment_analysis") \
        .enableHiveSupport()\
        .getOrCreate()

    return spark


def plot(plot_dic, height=800, width=800, **kwargs):
    kwargs['output_type'] = 'div'
    plot_str = plotly.offline.plot(plot_dic, **kwargs)
    print('%%angular <div style="height: %ipx; width: %spx"> %s </div>' %
          (height, width, plot_str))


spark_session = create_spark_session()
data = spark_session.sql(
    "select t1.country, t2.countryterritorycode,  t1.diff_sentiment from responseSentimentDifference t1 join dailyCountryCovidSentiment t2 on t1.country = t2.country
plot_data = data.toPandas()

plot({
    "data": [
        Choropleth(locations=plot_data['countryterritorycode'],
                   z=plot_data['diff_sentiment'])
```

```
    ],
    "layout": Layout(
        title="Sentiment difference after primary school closures",
        geo_scope="europe"
    )
})
```



```
%pyspark
```