PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
NÚCLEO DE EDUCAÇÃO A DISTÂNCIA
Pós-graduação *Lato Sensu* em Ciência de Dados e Big Data

Diego Rabelo Saraiva

BOT ANALYTICS – APLICAÇÃO PARA MONITORAMENTO E ANÁLISE DE DADOS DA COVID

Belo Horizonte
2021

Diego Rabelo Saraiva

BOT ANALYTICS – APLICAÇÃO PARA MONITORAMENTO E ANÁLISE DE DADOS DA COVID

Trabalho de Conclusão de Curso apresentado ao Curso de Especialização em Ciência de Dados e Big Data como requisito parcial à obtenção do título de especialista.

Belo Horizonte
2021

SUMÁRIO

1. Introdução

O projeto tem como objetivo, criar uma aplicação para a coleta D-1 de dados de covid e amostras do Twitter, e disponibiliza-lo para que qualquer pessoa possa replicar esse processo. O propósito principal é criar um Dashboard com um comparativo entre a pandemia do novo coronavírus em relação a epidemias históricas para mensurarmos alguns indicadores, como a média de mortalidade e share de casos por continentes e países. Em seguida, vamos explorar um pouco mais algumas estatísticas do novo coronavírus, como a regressão entre casos e mortes, e previsão de casos, mortes e vacinas utilizando a biblioteca Facebook Prophet. Por fim, realizaremos um estudo em tweets para visualizarmos a análise de sentimento com amostras do Twitter em D-1.

O trabalho foi estruturado nas seguintes partes:

1) Contextualização dos assuntos que serão tratados e o problema proposto;
2) Coleta e tratamento inicial dos datasets;
3) Limpeza e estruturação dos dados;
4) Análises exploratórias e insights identificados para o levantamento de informações relevantes, assim como aplicação de estatísticas e forecasts;
5) Aplicações de modelos de Machine Learning com Python;
6) Apresentação dos resultados através do Dashboard Google Data Studio;
7) Links e referências bibliográficas;

1.1. Contextualização

Entre o final do ano de 2019 e início do ano de 2020, recebemos os primeiros relatos do surgimento de um novo vírus. O novo coronavírus ou covid 19, é um vírus infeccioso com o surgimento dos primeiros casos na China, que em decorrer de poucos dias essa doença extremamente contagiosa, já estava pulverizada na sociedade global. No dia 11 de março de 2020, a Organização Mundial de Saúde (OMS), declarou o novo coronavírus como pandemia global, devido a transmissão já estar presente em diversos continentes, sendo transmitida de pessoa para pessoa em uma escala de tempo muito curta.

Porém este não foi o único caso de epidemia global. Durante algum tempo atrás tivemos casos de outros vírus transmissíveis, como o Ebola, H1N1 e Sars. Diante

deste contexto, precisamos acompanhar diariamente as estatísticas entre as epidemias, e a evolução dos casos, mortes e vacinações do novo coronavírus.

1.2. O problema proposto

O trabalho será dividido três partes:

1) Elaborar painel comparativo com as seguintes informações: Novo Coronavírus, Ebola, Sars e H1N1. O Dashboard deverá ser dividido em 3 painéis, contendo uma visão para visualizarmos a taxa de mortalidade por países e continentes, uma visão por share de casos e por fim uma visão com uma amostra D-1 do twitter com textos relacionados a COVID;

2) Criar uma arquitetura automatizada para que esses dados possam ser coletados diariamente e disponibilizados na nuvem para exibirmos no Dashboard em formato D-1, além de estarem disponíveis localmente para análise exploratória dos dados, sem consumir recursos na nuvem;

3) Com os dados armazenados e disponíveis em nosso servidor Docker, faremos uma análise exploratória dos dados coletados relacionados ao Covid19.

O desafio é gerarmos insights com os dados do passado, fixarmos o entendimento dos dados do presente através de correlações entre casos e mortes prevendo o cenário em até 60 dias, e gerar insigths através do comportamento atual da população (tweets).

O nosso dataset principal da Covid, tem origem do site Our World in Data (https://ourworldindata.org/), e os demais datasets utilizados para comparativo com as demais pandemias, foram extraídas do Kaggle.

Os dados da Covid-19 serão coletados diariamente através de uma URL, os dados das demais pandemias via arquivo .CSV, e os dados de tweets de amostra através da biblioteca python twint. Com esses dados em mãos, construiremos nosso dashboard e geraremos nossas análises exploratórias.

Período dos dados analisados: Covid: 01 de janeiro de 2020 à D-1 (Data D-1 da última execução da ETL), Ebola: 29 de agosto de 2014 à 23 de março de 2016, H1N1: 23 de maio de 2009 à 06 de julho de 2009, Sars: 17 de março de 2003 à 11 de julho de 2003. Os dados de tweets serão coletados no formato D-1, porém podem conter dados de dias anteriores que fizeram parte da construção desse modelo.

2. Coleta de Dados

A origem dos dados do projeto, está dividido em 5 datasets, sendo em dois repositórios diferentes.

O conjunto de dados deste projeto relacionados ao novo Coronavírus, está sendo obtido do github do projeto Our World in Data. Este projeto sem fins lucrativos, é baseado em um esforço colaborativo entre grupo de pesquisadores da Universidade de Oxford.

O dataset principal tem como objetivo extrair as informações reais sobre os principais índices de casos e mortes do novo coronavírus (Covid19). Está em um formato .csv, composto por aproximadamente cem mil registros.

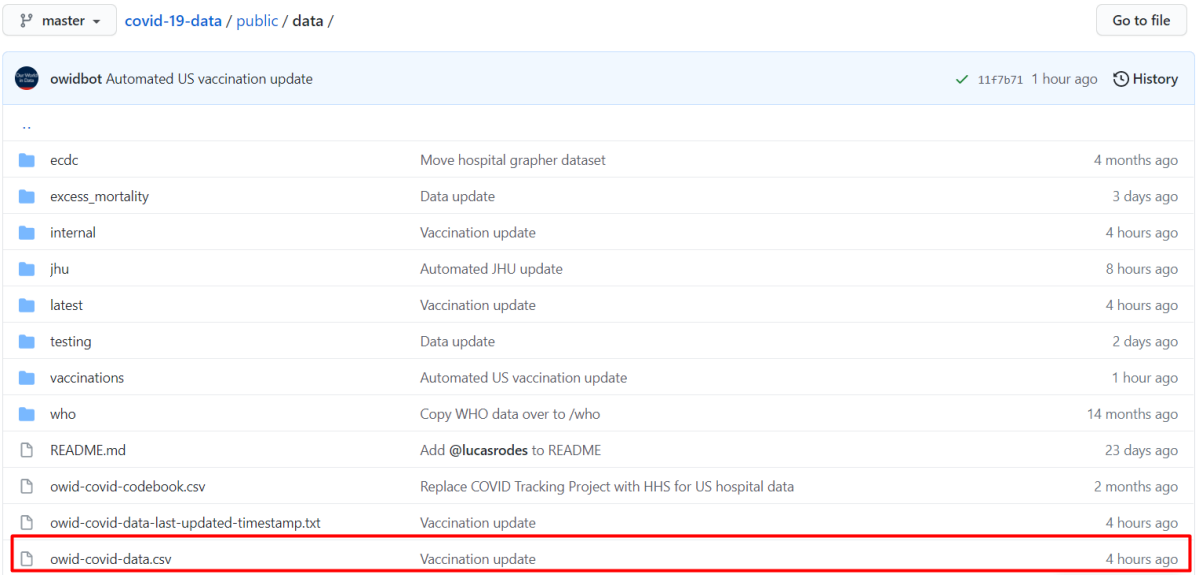


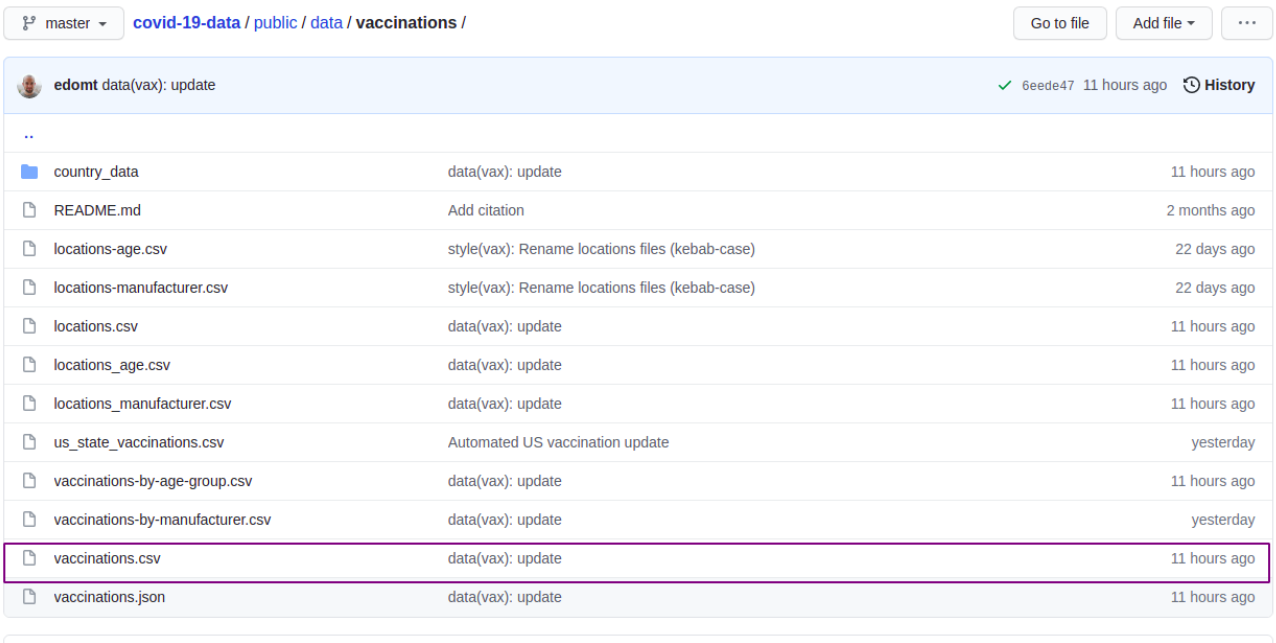
Figura 1 - Link para download do arquivo owid-covid-data.csv

Segue abaixo o dicionário do dataset obtido, apenas com os campos que vamos utilizar ao longo do projeto.

Dataset Covid19 (owid-covid-data.csv):

Nome da coluna/campo	Descrição	Tipo
id	Sigla do País de origem	string
date	Data da ocorrência	datetime
location	País de origem	string
continent	Continente de origem	string
new_cases	Novos casos	float64
new_deaths	Novas mortes	float64

Ainda relacionado ao novo coronavírus, também vamos extrair as informações sobre os índices de vacinações ao redor do mundo. O arquivo está no formato .csv:



Dataset Covid19 Vaccinations (vaccinations.csv):

Nome da coluna/campo	Descrição	Tipo
date	Data da ocorrência	datetime
iso_code	Sigla do País de origem	string
people_fully_vaccinated	Total de pessoas vacinadas	float64

Conforme informado anteriormente, trabalharemos com mais 3 datasets, que foram coletados do site Kaggle para coletar informações sobre o Ebola, H1N1 e Sars. Segue abaixo a imagem com link e os respectivos dicionários:

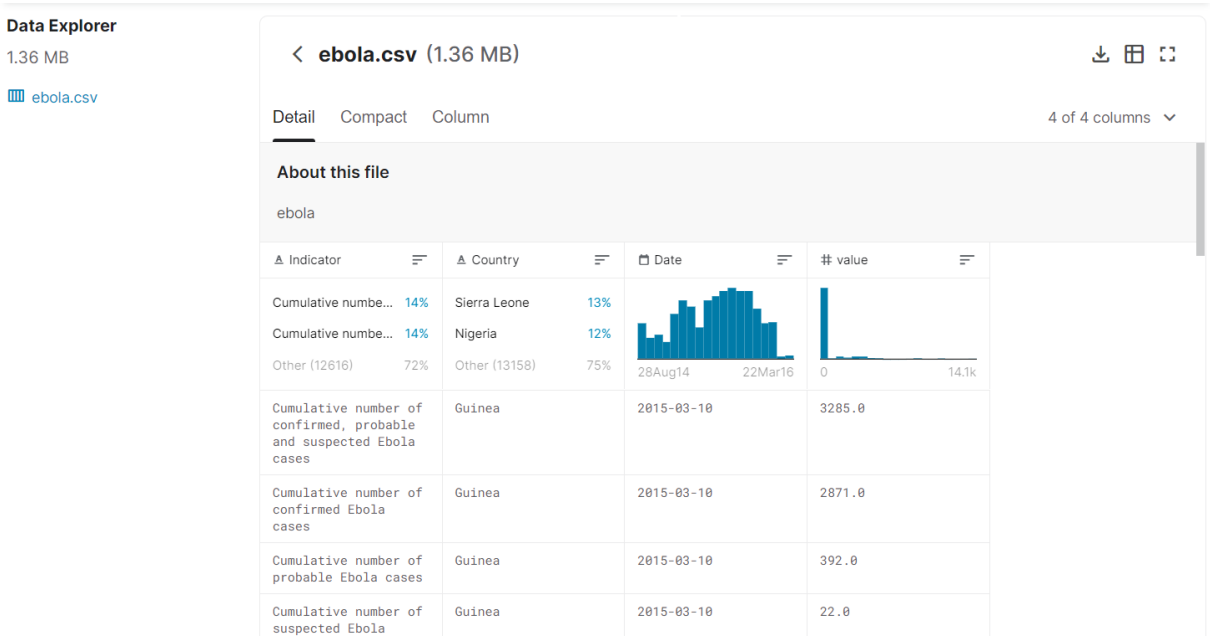


Figura 2 - Link para download do arquivo ebola.csv

Dataset Ebola (ebola.csv):

Nome da coluna/campo	Descrição	Tipo
Indicator	Indicador sobre casos confirmados e mortes	string
Country	País de origem	string
Date	Data da ocorrência	datetime
value	Quantitativo de acordo	float64

	com o campo indicador	

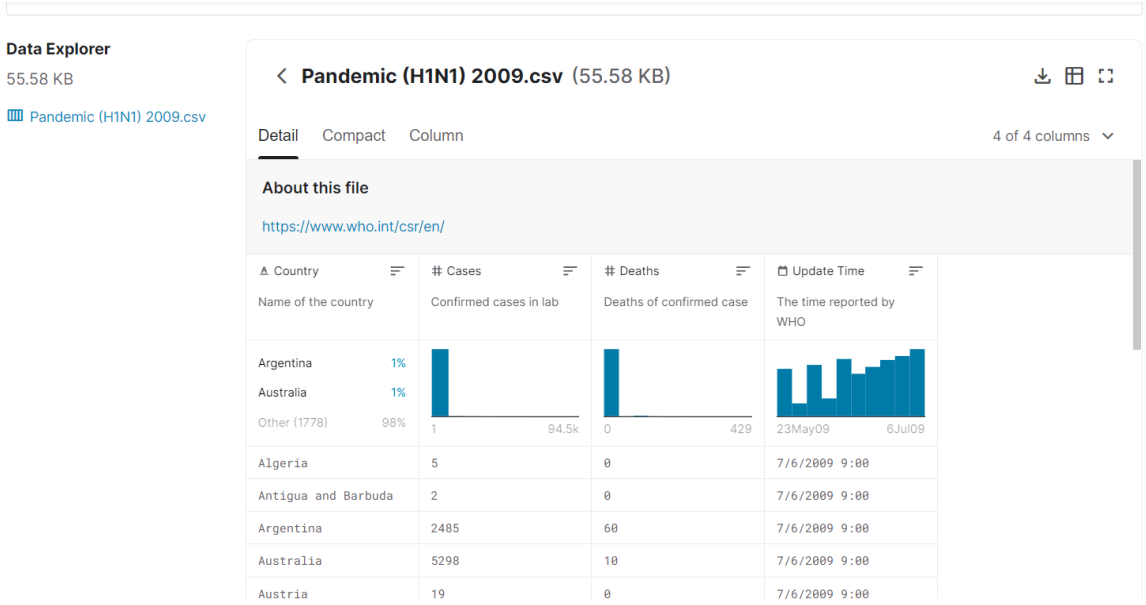


Figura 3 - Link para download do arquivo Pandemic (H1N1) 2009.csv

Dataset H1N1 (Pandemic (H1N1) 2009.csv):

Nome da coluna/campo	Descrição	Tipo
Country	País de origem	string
Cases	Quantidade de casos	float64
Deaths	Quantidade de mortes	float64
Update Time	Data da ocorrência	datetime

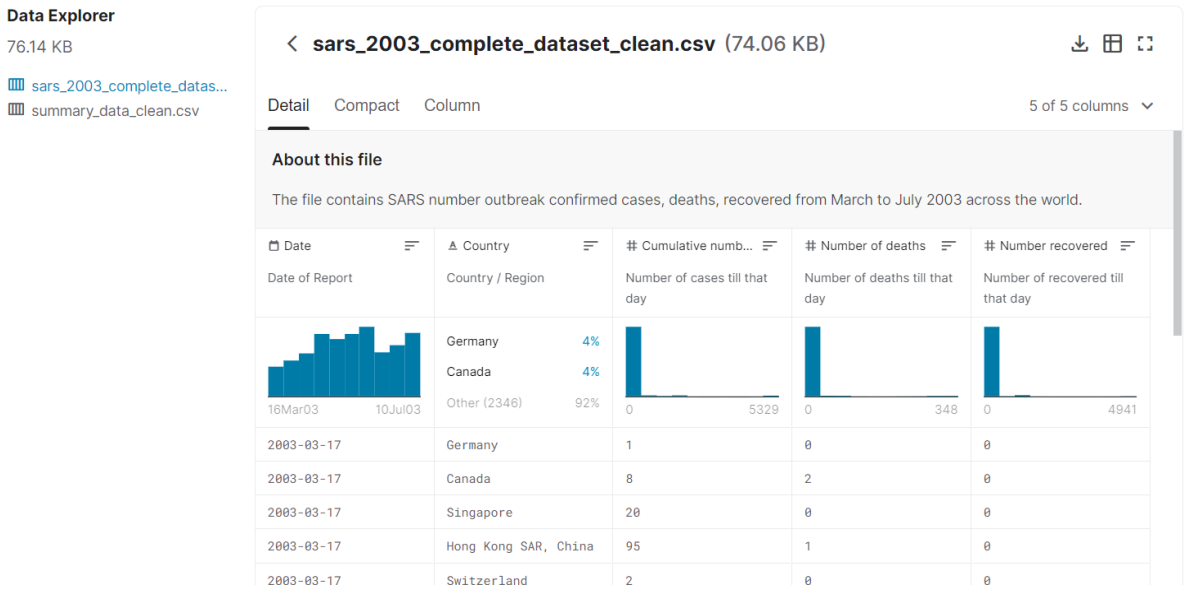


Figura 4 - [Link para download do arquivo sars_2003_complete_dataset_clean.csv]

Dataset SARS (sars_2003_complete_dataset_clean.csv):

Nome da coluna/campo	Descrição	Tipo
Date	Data da ocorrência	datetime
Country	País de origem	string
Cumulative number of case(s)	Quantidade de casos	float64
Number of deaths	Quantidade de mortes	float64

3. Processamento/Tratamento de Dados

- **Download do projeto:**

O projeto completo pode ser obtido através do GitHub, no link: https://github.com/diegorabelos/tccpucminasbotanalytics

- **Preparando o ambiente:**

Nosso ambiente foi definido na seguinte estrutura:

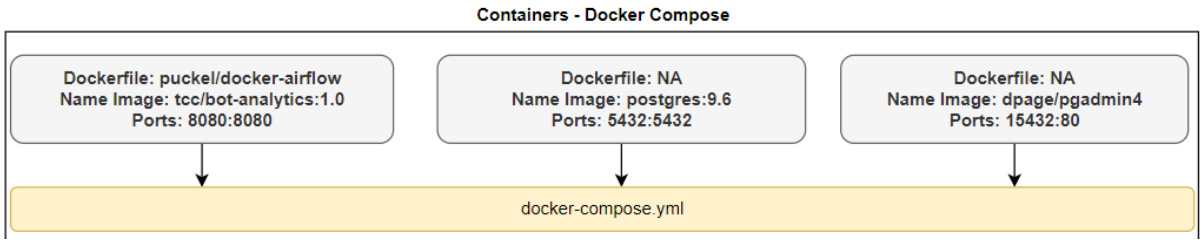


Figura 5 - Estrutura dos containers utilizado no projeto

Antes de partimos para a parte mais prática do projeto, é importante você estar utilizando uma máquina Linux (no projeto utilizei Ubuntu 18.04 LTS) para seguir o passo a passo, porém não é impeditivo, caso você esteja utilizando o Windows 10, poderá habilitar o recurso de subsistemas e habilitar no “Docker Desktop” a opção Use the WSL 2 based engine.
A instalação do Docker Engine e Docker Compose é obrigatória para executar a aplicação de ponta a ponta, não vou entrar em detalhes de como instalar, porém, vou informar abaixo duas documentações do site oficial, que utilizei como base:

Docker-engine: https://docs.docker.com/engine/install/ubuntu/

Docker-compose: https://docs.docker.com/compose/install/

Vamos precisar de três containers, sendo:

1. Imagem Airflow do Docker: puckel/docker-airflow. Essa imagem original foi modificada através do arquivo Dockerfile, onde contém os comandos Docker, conforme a imagem abaixo, para instalar os pacotes do arquivo requeriments.txt e alguns comandos pip install. Para criar a imagem modificada, você deve estar no diretório padrão: /usr/local/airflow/dags (Obs.: caso esteja utilizando Windows, ou queira utilizar outro diretório, basta mudar o caminho no docker-compose) e executar o comando no terminal para criar e subir a imagem: **$docker image build -t tcc/bot-analytics:1.0 .**

```
FROM puckel/docker-airflow

USER root

COPY requeriments.txt .

RUN pip install -r requeriments.txt
RUN pip install --upgrade pip
RUN pip install google-auth
RUN pip install pandas-gbq
RUN pip install openpyxl
RUN pip install xlrd
RUN pip install matplotlib
RUN pip install pandas-datareader
RUN pip install textblob
RUN pip install pycountry
```

Figura 6 - arquivo "Dockerfile"

2. Imagem postgres:9.6 do Docker: Servidor local, onde vamos armazenar os dados para explorarmos posteriormente.
3. Imagem dpage/pgadmin4: Interface do PGAdmin, para trabalharmos via browser.

Toda essa orquestração dos containers ficará na responsabilidade do arquivo **“docker-compose.yml”**. Ele está estruturado da seguinte forma:

```yaml
version: '2.1'
services:
    postgres:
        image: postgres:9.6
        restart: always
        environment:
            - POSTGRES_USER=airflow
            - POSTGRES_PASSWORD=airflow
            - POSTGRES_DB=airflow
        ports:
            - "5432:5432"

    pgadmin4:
        image: dpage/pgadmin4
        restart: always
        environment:
            - PGADMIN_DEFAULT_EMAIL=tcc@pucminas.com
            - PGADMIN_DEFAULT_PASSWORD=postgres2021
        ports:
            - "15432:80"
    webserver:
        image: tcc/bot-analytics:1.0
        restart: always
        depends_on:
            - postgres
        environment:
            - LOAD_EX=n
            - EXECUTOR=Local
            - AIRFLOW_HOME=/usr/local/airflow
        volumes:
            - /usr/local/airflow/dags:/usr/local/airflow/dags
        ports:
            - "8080:8080"
        command: webserver
        privileged: true
        healthcheck:
            test: ["CMD-SHELL", "[ -f /usr/local/airflow/airflow-webserver.pid ]"]
            interval: 30s
            timeout: 30s
            retries: 3
```

Figura 7 – arquivo “docker-compose.yml”

Para subirmos os containers, novamente, devemos estar via terminal no diretório /usr/local/airflow/dags, e executar o seguinte comando: **$ docker-compose -f docker-compose.yml up -d**

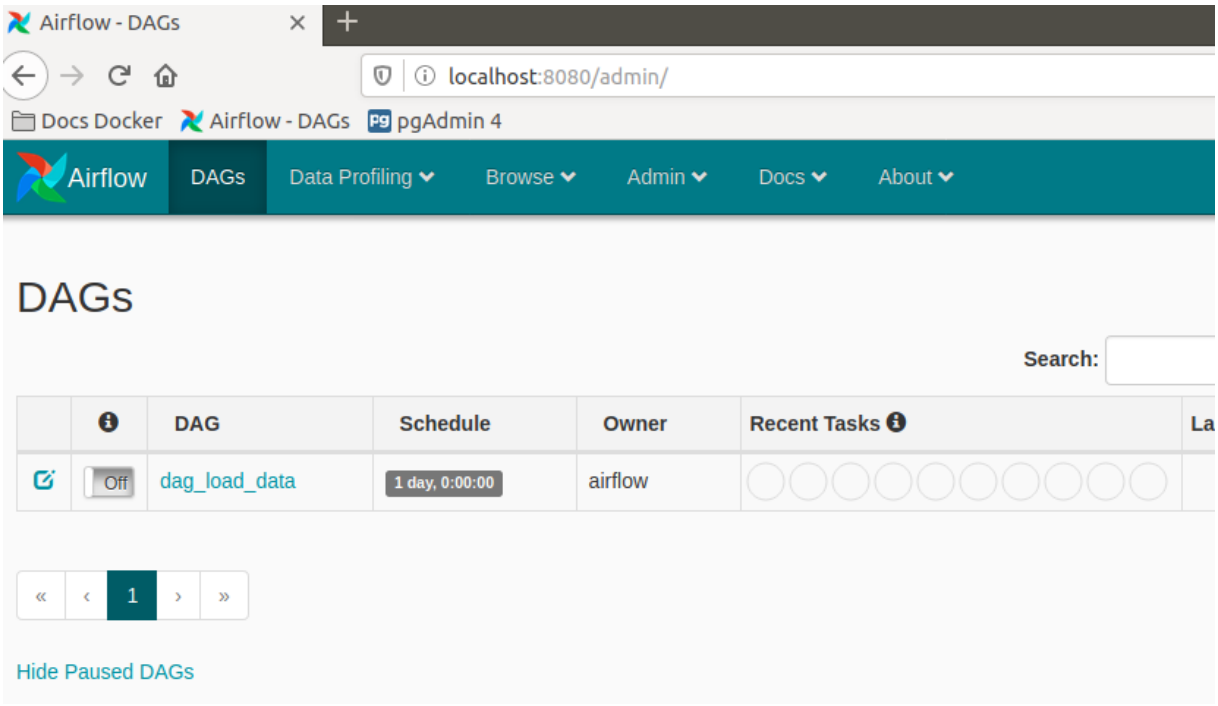
O resultado será similar a imagem abaixo:

```
root@tcc-VirtualBox:/usr/local/airflow/dags# docker-compose -f docker-compose.yml up -d
Creating network "dags_default" with the default driver
Creating dags_postgres_1 ... done
Creating dags_pgadmin4_1 ... done
Creating dags_webserver_1 ... done
root@tcc-VirtualBox:/usr/local/airflow/dags#
```

Figura 8 - Resultado da execução do comando docker-compose

Com isso, após alguns segundos, podemos acessar via browser o airflow e o pgadm, Segue os links:

- Airflow: http://localhost:8080/admin



- PGAdmin4: http://localhost:15432

  No caso do pgadmin, precisamos seguir algumas instruções para logar. O usuário e senha do login inicial, é o mesmo criado no docker-compose.yml (Valores das variáveis: PGADMIN_DEFAULT_EMAIL, PGADMIN_DEFAULT_PASSWORD) que foi criado apenas como exemplo, mas pode ser alterado. É uma autenticação apenas para acessar a interface do PGADMIN, podendo ser inclusive um e-mail fake, como fiz no meu exemplo.

  Ao acessar, é necessário criar uma conexão com o banco com os parâmetros que definimos no container do postgres, sendo:

  Host: dags_postgres_1

  Porta: 5432

  Username: airflow

Password: airflow

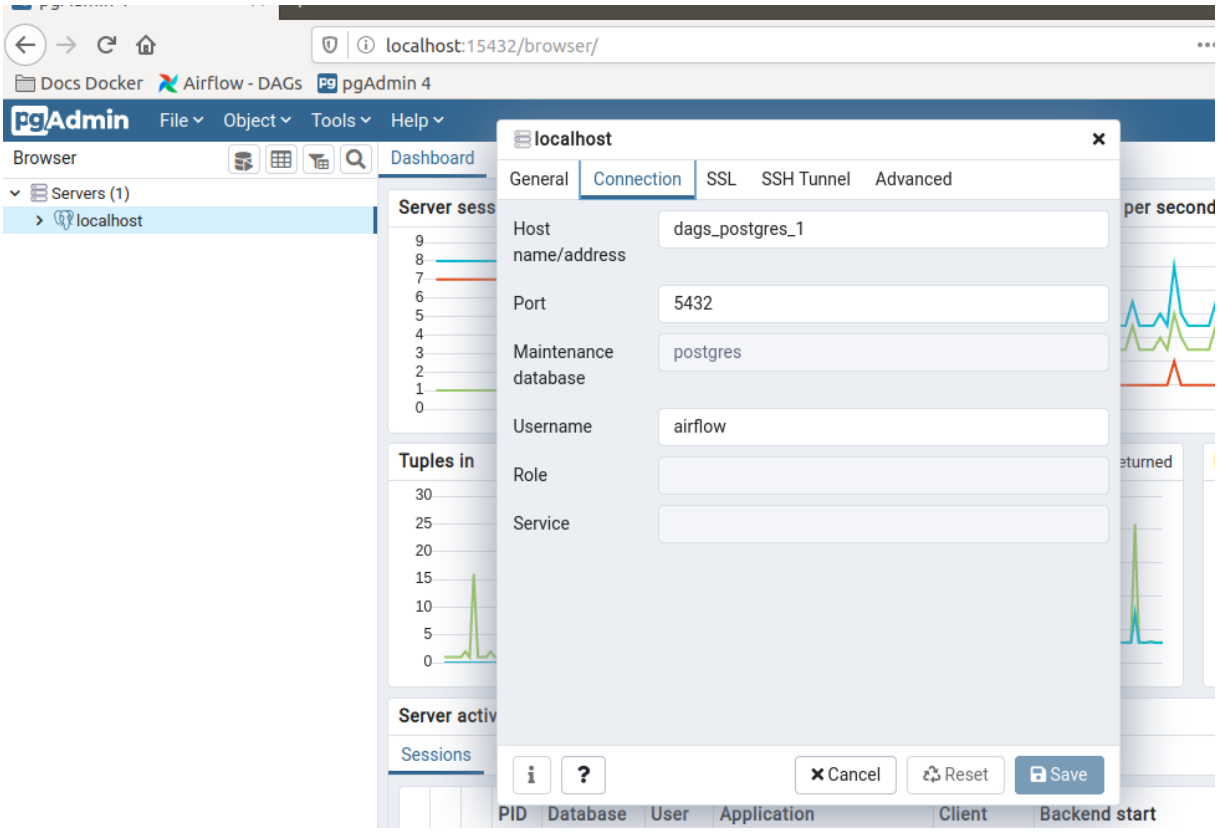


Figura 9 - Demonstração da conexão com o banco localhost

- **Entendendo os scripts e a DAG:**

Agora que subimos os containers, antes de irmos para execução do JOB, primeiro é necessário entender como foi construído os scripts.

Todos os scripts estão disponíveis na pasta “dags”. Podemos dividir da seguinte forma:

- **dag_load_data.py**:

```
from airflow import DAG
from airflow.operators import BashOperator
from airflow.operators.dummy_operator import DummyOperator
from datetime import datetime, timedelta
from configparser import RawConfigParser

date = datetime.now()

# Following are defaults which can be overridden later on
default_args = {
    'owner': 'airflow',
```

```
    'depends_on_past': False,
    'start_date': datetime(date.year, date.month, date.day),
    'email': ['airflow@airflow.com'],
    'email_on_failure': False,
    'email_on_retry': False,
    'retries': 1,
    'retry_delay': timedelta(minutes=5),
    'schedule_interval': '0 9 * * *',
}

dag = DAG('dag_load_data', default_args=default_args)

t1 = BashOperator(
    task_id='task_1',
    bash_command='python3 $AIRFLOW_HOME/dags/get_data_pandemics.py',
    dag=dag)

t2 = BashOperator(
    task_id='task_2',
    bash_command='python3 $AIRFLOW_HOME/dags/get_data_twitter.py',
    dag=dag)

t2.set_upstream(t1)
```

Script da DAG que o airflow utiliza para executar o JOB. São as instruções que devemos passar na ordem de execução dos jobs.

Sobre alguns parâmetros, vale destacar:

schedule_interval: indica que o processo será executado uma vez por dia às 09hrs

t1: É a primeira tarefa que pedimos para ser executada. Nesse caso, vamos carregar os dados do COVID, e também os dados das demais pandemias, contidas no script get_data_pandemics.py

t2: É a segunda tarefa, onde vamos carregar os dados com amostra do Twitter, contidas no script get_data_twitter.py

t2.set_upstream(t1): Aqui dizemos para o airflow, que a task2 (t2), deverá ser executada após a task1

- **get_data_pandemics.py**:

Vamos importas as bibliotecas e definimos o diretório;

```
import pandas as pd
import numpy as np
from google.oauth2 import service_account
import pandas_gbq
from datetime import *
import matplotlib.pyplot as plt
from google.oauth2 import service_account
import google.auth.compute_engine
from google.cloud import bigquery
import sqlalchemy

directory = '/usr/local/airflow/dags/'
```

df_covid:

Primeiro passo após importarmos as bibliotecas, é carregar e tratar os dados de covid. Os dados serão extraídos através de URL, e carregados em um Dataframe do pandas (df_covid). Criamos a coluna category, para separarmos as pandemias por categoria. Selecionamos apenas as colunas que utilizaremos no projeto e deletamos os dados que estão com o continente vazios. Percebemos abaixo que neste momento tivemos quase cem mil registros e 7 colunas no processo.

```
url_cases_deaths='https://raw.githubusercontent.com/owid/covid-19-
data/master/public/data/owid-covid-data.csv'
df_covid = pd.read_csv(url_cases_deaths,index_col=0,parse_dates=[0])
df_covid['category'] = 'covid'
df_covid['id'] = df_covid.index
df_covid = df_covid[['id','category','date','location','continent','new_ca
ses','new_deaths']]
df_covid = df_covid.rename(columns={"new_cases":"tt_cases","new_deaths":"t
t_deaths","location":"country"})
# Delete Continente NULL
df_covid_index_to_drop = df_covid[df_covid['continent'].isnull()].index
df_covid.drop(df_covid_index_to_drop , inplace=True)
```

Criamos o Dataframe df_country_covid, para coletarmos todos os continentes dos respectivos países. Utilizaremos essa variável para alimentarmos o continente dos demais datasets que serão apresentados logo abaixo, pois algumas não possuem continentes. Também criamos uma coluna com o contador dos dias. Essas variáveis são interessantes para sabermos por exemplo, a quantidade de dias que temos dados da covid, ou qual o pais ou continente com a origem da doença.

```
# Create dataframe to get continent
df_country_covid = df_covid[['country','continent']]
```

```
df_country_covid['country'] = df_covid['country'].str[:8]
df_country_covid.drop_duplicates(inplace=True)

# Gerar acumulado por dias
df_covid_date = df_covid['date'].dropna().drop_duplicates()
df_covid_date = pd.DataFrame(df_covid_date)
df_covid_date['contador_dia'] = 1
df_covid_date['contador_dia'] = df_covid_date['contador_dia'].cumsum()
df_covid = pd.merge(left=df_covid,right=df_covid_date,how='left',left_on='
date',right_on='date')

df_covid['date'] = pd.to_datetime(df_covid['date'])
```

Os próximos scripts vou demonstrar dentro do próprio console do Spyder, para entendermos os resultados:

A imagem abaixo, traz o total de registros de casos negativos:

```
In [5]: df_covid['date'] = pd.to_datetime(df_covid['date'])
   ...:
   ...: # tt cases negativos
   ...: df_covid[(df_covid.tt_cases < 0)].count()
   ...:
Out[5]:
id              67
category        67
date            67
country         67
continent       67
tt_cases        67
tt_deaths       61
contador_dia    67
dtype: int64
```

Total de registros de mortes negativos:

```
In [6]: df_covid[(df_covid.tt_deaths < 0)].count()
Out[6]:
id              101
category        101
date            101
country         101
continent       101
tt_cases        101
tt_deaths       101
contador_dia    101
dtype: int64
```

A primeira tratativa, é passar esses valores para duas novas variáveis, e criar uma coluna date_dif, onde vamos buscar a data de 7 dias anteriores. Com essa data, fazemos um merge ainda nessas novas variáveis com o nosso Dataframe original "df_covid", para preenchermos com os totais. E por fim, fazemos o merge entre o nosso Dataframe original, com esses valores dos dias que ficaram negativos. No último step da imagem abaixo, tratamos os valores NaN, e consideramos o valor 0, pois para esses casos são cenários que não houve mortes ou casos e o dataset estava trazendo NaN, ou invés de valor 0.

```
df_covid_cases_outliers = df_covid[(df_covid.tt_cases < 0)]
df_covid_deaths_outliers = df_covid[(df_covid.tt_deaths < 0)]

df_covid_cases_outliers['date_dif'] = df_covid_cases_outliers['date'] -
 timedelta(days=7)
df_covid_deaths_outliers['date_dif'] = df_covid_deaths_outliers['date'] -
 timedelta(days=7)

df_covid_cases_outliers = pd.merge(left=df_covid_cases_outliers,right=df_c
ovid[['date','country','tt_cases','tt_deaths']],how='left',left_on=['date_
dif','country'],right_on=['date','country'])
df_covid_cases_outliers['tt_cases_y'] = np.where(df_covid_cases_outliers['
tt_cases_y'] < 0,0,df_covid_cases_outliers['tt_cases_y'])
df_covid_cases_outliers['tt_deaths_y'] = np.where(df_covid_cases_outliers[
'tt_deaths_y'] < 0,0,df_covid_cases_outliers['tt_deaths_y'])

df_covid_deaths_outliers = pd.merge(left=df_covid_deaths_outliers,right=df
_covid[['date','country','tt_cases','tt_deaths']],how='left',left_on=['dat
e_dif','country'],right_on=['date','country'])
df_covid_deaths_outliers['tt_cases_y'] = np.where(df_covid_deaths_outliers
['tt_cases_y'] < 0,0,df_covid_deaths_outliers['tt_cases_y'])
df_covid_deaths_outliers['tt_deaths_y'] = np.where(df_covid_deaths_outlier
s['tt_deaths_y'] < 0,0,df_covid_deaths_outliers['tt_deaths_y'])

df_covid_outliers = pd.concat([df_covid_cases_outliers,df_covid_deaths_out
liers])

df_covid_outliers.drop_duplicates(inplace=True)

df_covid = pd.merge(left=df_covid,right=df_covid_outliers[['date_x','count
ry','tt_cases_y','tt_deaths_y']],how='left',left_on=['date','country'],rig
ht_on=['date_x','country'])
df_covid['tt_cases'] = np.where(df_covid['tt_cases'] < 0,df_covid['tt_case
s_y'],df_covid['tt_cases'])
df_covid['tt_deaths'] = np.where(df_covid['tt_deaths'] < 0,df_covid['tt_de
aths_y'],df_covid['tt_deaths'])
```

```
# tratando os nan
df_covid['tt_cases'] = np.where(df_covid['tt_cases'].isnull(),0,df_covid['
tt_cases'])
df_covid['tt_deaths'] = np.where(df_covid['tt_deaths'].isnull(),0,df_covid
['tt_deaths'])
```

Agora executando novamente o comando para validar os valores negativos, percebemos que foram corrigidos:

```
In [10]: df_covid[(df_covid.tt_cases < 0)].count()
Out[10]:
id              0
category        0
date            0
country         0
continent       0
tt_cases        0
tt_deaths       0
contador_dia    0
date_x          0
tt_cases_y      0
tt_deaths_y     0
dtype: int64
```

```
In [11]: df_covid[(df_covid.tt_deaths < 0)].count()
Out[11]:
id              0
category        0
date            0
country         0
continent       0
tt_cases        0
tt_deaths       0
contador_dia    0
date_x          0
tt_cases_y      0
tt_deaths_y     0
dtype: int64
```

E por fim, apenas deletamos as colunas que foram geradas no merge:

```
df_covid.drop(["date_x","tt_cases_y","tt_deaths_y"], axis=1, inplace=True)
```

df_vaccinations:

O dataset tem como objetivo trazer informações sobre a quantidade de pessoas vacinadas. Ela é extraída do mesmo diretório dos dados da covid, porém em uma URL diferente. Nesse caso, vamos carregar a variável df_vaccinations, formatar a data, validar se existe algum valor menor que 0, ou valores nulos na coluna que utilizaremos: “people_fully_vaccined”. E por fim vamos fazer um merge desses dados com o nosso Dataframe da Covid (df_covid):

```
## Get vaccinations
url_vaccinations='https://raw.githubusercontent.com/owid/covid-19-
data/master/public/data/vaccinations/vaccinations.csv'
df_vaccinations = pd.read_csv(url_vaccinations,index_col=0,parse_dates=[0]
)
df_vaccinations['date'] = pd.to_datetime(df_vaccinations['date'])

df_vaccinations[(df_vaccinations.people_fully_vaccinated < 0)].count()
df_vaccinations.isnull().any()
df_vaccinations['people_fully_vaccinated'] = np.where(df_vaccinations['peo
ple_fully_vaccinated'].isnull(),0,df_vaccinations['people_fully_vaccinated
'])

df_covid = pd.merge(left=df_covid,right=df_vaccinations[['date','iso_code'
,'people_fully_vaccinated']],how='left',left_on=['date','id'],right_on=['d
ate','iso_code'])
```

Para os demais datasets, utilizamos a mesma lógica. Vou destacar apenas alguns pontos que mudam entre a lógica que utilizei para o Covid, em relação aos demais datasets.

df_ebola:

O dataset “dataset_ebola.csv”, está disponível no diretório dags/data. Para o continent, atribuimos o valor nulo, porém tratamos na sequência utilizando a variável df_country_covid no merge. Foi necessário o ajuste no valor de dois países, que estão formatados errados, e daria problema para buscarmos os valores de continentes.

```
df_ebola = pd.read_csv(directory+'data/dataset_ebola.csv',sep=",")
#df_ebola['Indicator'].unique().tolist()
df_ebola['category'] = 'ebola'
df_ebola['continent'] = 'null'
df_ebola['tt_cases'] = df_ebola['value'].where(df_ebola['Indicator'] == 'C
umulative number of confirmed Ebola cases')
df_ebola['tt_deaths'] = df_ebola['value'].where(df_ebola['Indicator'] == '
Cumulative number of confirmed Ebola deaths')
df_ebola.columns=['indicator','country','date','value','category','contine
nt','tt_cases','tt_deaths']
df_ebola = df_ebola.groupby(['category','date','country','continent'])['tt
_cases','tt_deaths'].sum().reset_index()

# Alterar nomes de paises invalidos
df_ebola["country"] = df_ebola["country"].replace(['Liberia 2'],'Liberia')
df_ebola["country"] = df_ebola["country"].replace(['Guinea 2'],'Guinea')
df_ebola['country_merge'] = df_ebola['country'].str[:8]
```

As demais lógicas são exatamente a mesma utilizadas para o Dataframe df_covid, nesse caso não temos dados negativos, por isso não foi necessário a tratativa:

```
# merge para classificar continent
df_ebola = pd.merge(left=df_ebola,right=df_country_covid,how='left',left_o
n='country_merge',right_on='country')
df_ebola = df_ebola.rename(columns={"country_x":"country","continent_y":"c
ontinent"})
df_ebola = df_ebola[["category","date","country","continent","tt_cases","t
t_deaths"]]

# Gerar acumulado por dias
df_ebola_date = df_ebola['date'].dropna().drop_duplicates()
df_ebola_date = pd.DataFrame(df_ebola_date)
df_ebola_date['contador_dia'] = 1
df_ebola_date['contador_dia'] = df_ebola_date['contador_dia'].cumsum()
df_ebola = pd.merge(left=df_ebola,right=df_ebola_date,how='left',left_on='
date',right_on='date')

# tt cases negativos
df_ebola[(df_ebola.tt_cases < 0)].count()
# tt deaths negativos
df_ebola[(df_ebola.tt_deaths < 0)].count()
```

df_h1n1:

O dataset “dataset_h1n1.csv”, está disponível no diretório dags/data. Para o continent, atribuímos o valor nulo, porém tratamos na sequência utilizando a variável df_country_covid no merge. Foi necessário o ajuste no valor de três países, que estão formatados errados, e daria problema para buscarmos os valores de continentes. Também criamos a coluna country_merge, onde vamos selecionar a string dos países até 8 caracteres. Isso ocorre pois, exclusivamente nesse dataset, temos alguns países que contém valores com maior extensão, por exemplo: Ao invés de United States, esse dataset traz United States Of America. Com isso melhoramos nossa precisão.

```
df_h1n1 = pd.read_csv(directory+'data/dataset_H1N1.csv',sep=",",encoding='
unicode_escape')
df_h1n1['category'] = 'h1n1'
df_h1n1['continent'] = 'null'
df_h1n1.columns=['country','tt_cases','tt_deaths','date','category','conti
nent']
df_h1n1['date'] = pd.to_datetime(df_h1n1['date'])
df_h1n1['date'] = df_h1n1['date'].dt.strftime('%Y-%m-%d')

# Alterar nomes de paises invalidos
df_h1n1["country"] = df_h1n1["country"].replace(['Korea, Republic of'],'So
uth Korea')
df_h1n1["country"] = df_h1n1["country"].replace(['Viet Nam'],'Vietnam')
df_h1n1["country"] = df_h1n1["country"].replace(['Brunei Darussalam'],'Bru
nei')
df_h1n1["continent"] = df_h1n1["continent"].where(df_h1n1["country"] == "G
uatemala").replace('','North America')
df_h1n1['country_merge'] = df_h1n1['country'].str[:8]
df_h1n1 = df_h1n1[['category','date','country','country_merge','continent'
,'tt_cases','tt_deaths']]

# merge para classificar continent
df_h1n1 = pd.merge(left=df_h1n1,right=df_country_covid,how='left',left_on=
'country_merge',right_on='country')
df_h1n1 = df_h1n1.rename(columns={"country_x":"country","continent_y":"con
tinent"})
df_h1n1 = df_h1n1[["category","date","country","continent","tt_cases","tt_
deaths"]]
df_h1n1_index_to_drop = df_h1n1[df_h1n1['country']== "Grand Total"].index
df_h1n1.drop(df_h1n1_index_to_drop , inplace=True)
```

As demais lógicas são exatamente a mesma utilizadas para o Dataframe df_ebola:

```
# Gerar acumulado por dias
df_h1n1_date = df_h1n1['date'].dropna().drop_duplicates()
df_h1n1_date = pd.DataFrame(df_h1n1_date)
df_h1n1_date['contador_dia'] = 1
df_h1n1_date['contador_dia'] = df_h1n1_date['contador_dia'].cumsum()
df_h1n1 = pd.merge(left=df_h1n1,right=df_h1n1_date,how='left',left_on='dat
e',right_on='date')

# tt cases negativos
df_h1n1[(df_h1n1.tt_cases < 0)].count()
# tt deaths negativos
df_h1n1[(df_h1n1.tt_deaths < 0)].count()
```

df_sars:

O dataset “dataset_sars.csv”, está disponível no diretório dags/data. Para o continent, atribuímos o valor nulo, porém tratamos na sequência utilizando a variável df_country_covid no merge. Foi necessário o ajuste no valor de seis países, que estão formatados errados, e daria problema para buscarmos os valores de continentes.

```
df_sars = pd.read_csv(directory+'data/dataset_sars.csv',sep=",")
df_sars['category'] = 'sars'
df_sars['continent'] = 'null'
df_sars.columns=['date','country','tt_cases','tt_deaths','tt_recovery','ca
tegory','continent']

# Alterar nomes de paises invalidos
df_sars["country"] = df_sars["country"].replace(['Hong Kong SAR, China'],'
China')
df_sars["country"] = df_sars["country"].replace(['Taiwan, China'],'China')
df_sars["country"] = df_sars["country"].replace(['Macao SAR, China'],'Chin
a')
df_sars["country"] = df_sars["country"].replace(['Viet Nam'],'Vietnam')
df_sars["country"] = df_sars["country"].replace(['Russian Federation'],'Ru
ssia')
df_sars["country"] = df_sars["country"].replace(['Republic of Ireland'],'I
reland')
df_sars["country"] = df_sars["country"].replace(['Republic of Korea'],'Sou
th Korea')

df_sars['country_merge'] = df_sars['country'].str[:8]
df_sars = df_sars[['category','date','country','country_merge','continent'
,'tt_cases','tt_deaths']]
```

```
# merge para classificar continent
df_sars = pd.merge(how='left',left=df_sars,right=df_country_covid,left_on=
'country_merge',right_on='country')
df_sars = df_sars.rename(columns={"country_x":"country","continent_y":"con
tinent"})
df_sars = df_sars[["category","date","country","continent","tt_cases","tt_
deaths"]]
```

As demais lógicas são exatamente a mesma utilizadas para o Dataframe df_h1n1:

```
# Gerar acumulado por dias
df_sars_date = df_sars['date'].dropna().drop_duplicates()
df_sars_date = pd.DataFrame(df_sars_date)
df_sars_date['contador_dia'] = 1
df_sars_date['contador_dia'] = df_sars_date['contador_dia'].cumsum()
df_sars = pd.merge(left=df_sars,right=df_sars_date,how='left',left_on='dat
e',right_on='date')

# tt cases negativos
df_sars[(df_sars.tt_cases < 0)].count()
# tt deaths negativos
df_sars[(df_sars.tt_deaths < 0)].count()
```

E por fim, fazemos a junção dos 4 Dataframes que foram criados e tratados:

```
df_final = pd.concat([df_covid,df_ebola,df_h1n1,df_sars])
```

Apresentamos na tela os períodos coletados de todos os Dataframes:

```
print("Periodo de covid19 entre: " + str(df_covid['date'].min())[:10]+" e
"+str(df_covid['date'].max())[:10])
print("Periodo de ebola entre: " + str(df_ebola['date'].min())+" e "+str(d
f_ebola['date'].max()))
print("Periodo de h1n1 entre: " + str(df_h1n1['date'].min())+" e "+str(df_
h1n1['date'].max()))
print("Periodo de sars entre: " + str(df_sars['date'].min())+" e "+str(df_
sars['date'].max()))
```

Os próximos passos são referentes a carga da base.

Primeiro passo é inserirmos os dados no bigquery. Para isso é necessário ter uma conta no gmail, e criar um projeto no bigquery (https://console.cloud.google.com/bigquery). Após criar o projeto, é necessário gerar uma credencial de acesso (arquivo JSON), e subir no diretório dags/credentials. Eu publiquei um artigo em meu site, explicando detalhadamente como realizar esse o passo a passo para gerar a credencial: http://diegorabelo.com/pandas-bigquery-insert-data-in-bigquery/

Lembre-se de alterar a variável pk_json_input, informando a nomenclatura do arquivo JSON gerado pelo google, e o project_input, caso queira criar um outro nome para o projeto:

```
pk_json_input = directory+"credentials/tcc-pucminas-bot-analytics-462531da5c8e.json"

project_input = "tcc-pucminas-bot-analytics"

auth = service_account.Credentials.from_service_account_file(pk_json_input)

# após criar o account service, ir até o IAM e em permissoes do projeto adicionar Administrador do BigQuery

df_final.to_gbq(
    'bota-nalytics_log.tb_pandemic_data',credentials=auth, project_id=project_input, if_exists='replace',
)
```

E por fim, carregamos os dados no postgres local em que subimos no docker_compose.yml, lembrando que precisa ter iniciado os containers do Docker (página 13):

```
databa-se_connection_try1 = sqlalchemy.create_engine('postgresql://airflow:airflow@dags_postgres_1/postgres')
databa-se_connection_try2 = sqlalchemy.create_engine('postgresql://airflow:airflow@0.0.0.0/postgres')

#database_connection
```

```
try:
    df_final.to_sql(con=database_connection_try1,name='tb_pandemics',if_ex
ists='replace')
except:
    df_final.to_sql(con=database_connection_try2,name='tb_pandemics',if_ex
ists='replace')
```

- **get_data_twitter.py**:

Vamos importas as bibliotecas utilizadas no processo de coleta de tweets, e definir o diretório:

```
import pandas as pd
import nltk
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('vader_lexicon')
import re
import string
from textblob import TextBlob
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from sklearn.feature_extraction.text import CountVectorizer
from datetime import *
from google.oauth2 import service_account
import pandas_gbq
from google.cloud import bigquery
import twint
import sqlalchemy

directory = '/usr/local/airflow/dags/'
```

Definir a data de coleta, sendo D-1 (timedelta(1)):

```
date_since = datetime.now() - timedelta(1)
date_until = datetime.now()
date_since = datetime.strftime(date_since, '%Y-%m-%d')
date_until = datetime.strftime(date_until, '%Y-%m-%d')
```

Agora vamos coletar 1000 tweets, contendo a palavra Covid. Em seguida filtraremos apenas os tweets no idioma Inglês. Nesse exemplo, de 1000 tweets, tivemos 719 dentro desse idioma:

```
c = twint.Config()
c.Search = "Covid"
c.Since = date_since
c.Until = date_until
c.Limit = 1000
c.Popular_tweets = True
c.Pandas = True

twint.run.Search(c)

df_tweets = twint.storage.panda.Tweets_df

df_tweets = df_tweets.loc[df_tweets['language']=='en']
```

Agora com os tweets carregados no Dataframe df_tweets, vamos trabalhar com alguns tratamentos no texto. Primeiro passo é removermos do tweet original, as menções “@”, “#”, “:”,”RT”, “http://”,”https://”:

```
# Create a function to clean the tweets
def cleanTxt(text):
 text = re.sub('@[A-Za-z0-9]+', '', text)
 text = re.sub('#', '', text)
 text = re.sub(':', '', text)
 text = re.sub('RT[\s]+', '', text)
 text = re.sub('https?\/\/\S+', '', text)
 text = re.sub('https?:\/\/\S+', '', text)

 return text


# Clean the tweets
df_tweets['text'] = df_tweets['tweet'].apply(cleanTxt)
```

Calcular a análise de sentimentos nessa coluna text, em que realizamos a limpeza anteriormente:

```
#Calculating Negative, Positive, Neutral and Compound values
df_tweets[['polarity', 'subjectivity']] = df_tweets['text'].apply(lambda T
ext: pd.Series(TextBlob(Text).sentiment))
for index, row in df_tweets['text'].iteritems():
 score = SentimentIntensityAnalyzer().polarity_scores(row)
 neg = score['neg']
 neu = score['neu']
 pos = score['pos']
 comp = score['compound']
```

```
if neg > pos:
    df_tweets.loc[index, 'sentiment'] = "negative"
elif pos > neg:
    df_tweets.loc[index, 'sentiment'] = "positive"
else:
    df_tweets.loc[index, 'sentiment'] = "neutral"
    df_tweets.loc[index, 'neg'] = neg
    df_tweets.loc[index, 'neu'] = neu
    df_tweets.loc[index, 'pos'] = pos
    df_tweets.loc[index, 'compound'] = comp
```

Criei duas colunas para calcular a quantidade do tamanho do texto, e a quantidade de palavras. Isso nos ajuda a entender a performance da análise de sentimento, sendo que quanto maior o tweet, menor a chance de assertividade da classificação:

```
#Calculating tweet's lenght and word count
df_tweets['text_len'] = df_tweets['text'].astype(str).apply(len)
df_tweets['text_word_count'] = df_tweets['text'].apply(lambda x: len(str(x
).split()))
```

Segue um exemplo de como ficou o DataFrame até o momento:

text	polarity	subjectivity	sentiment	neg	neu	pos	compound	text_len	:ext_word_count
ll of these umbfucks c…	-0.23333333…	0.5333333333	negative	nan	nan	nan	nan	239	42
9 neumonia i…	-0.375	0.5	negative	nan	nan	nan	nan	87	16
$ You can y this bu…	0.5	0.5	positive	nan	nan	nan	nan	185	37
_Fury It's een postpo…	0	0	negative	nan	nan	nan	nan	64	10
3 Yes Kevi…	0.075	0.325	neutral	0.126	0.748	0.126	0	89	18
WalesOnlin…	0.35	0.55	neutral	0	1	0	0	169	25
l _free Lol Covid rel…	0.1371428571	0.6471428571	negative	nan	nan	nan	nan	122	20
9241 3ar : depends …	0.2083333333	0.5166666667	negative	nan	nan	nan	nan	112	22
But for nese test …	-0.20227272…	0.35	negative	nan	nan	nan	nan	241	46
ook I know nat techni…	0.2	0.5333333333	positive	nan	nan	nan	nan	185	35
5687573 elp, you'r…	5.551115123…	0.6666666667	negative	nan	nan	nan	nan	232	44
Being omophobic …	-0.55	1	positive	nan	nan	nan	nan	146	25
niversity f Ibadan d…	0	0	negative	nan	nan	nan	nan	61	9
hurch ounder say…	0	0.3	neutral	0	1	0	0	101	16
Covid kills ne out of …	-0.2	0	negative	nan	nan	nan	nan	124	23
Yep. We ave heaps …	0.25	0.5	negative	nan	nan	nan	nan	272	51
af 23 _News hildren we…	0	0.75	neutral	0	1	0	0	268	37
You didn't…	0	0.066666666…	neutral	0	1	0	0	168	23
3 COVID	0	0	neutral	0	1	0	0	66	13

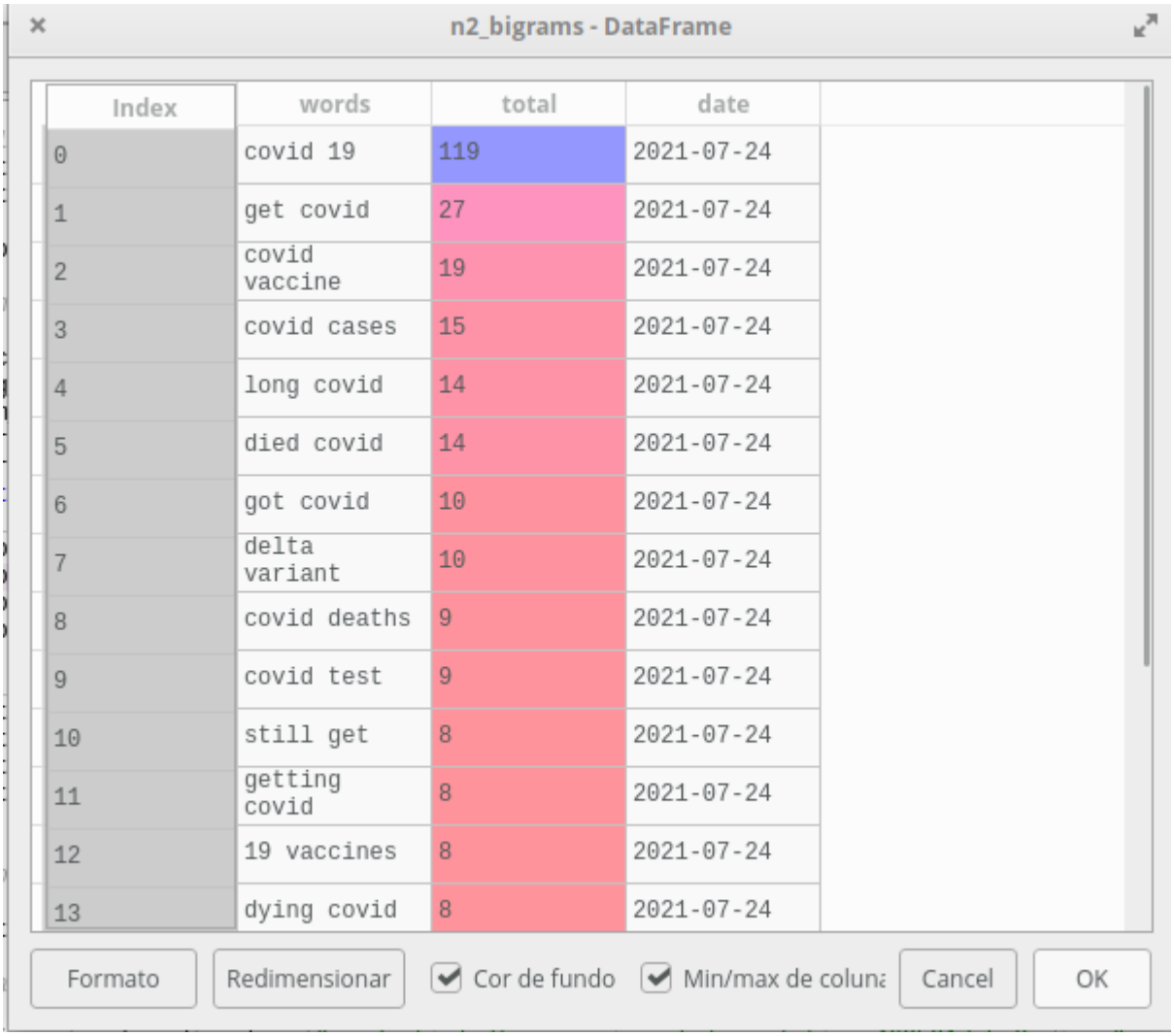
Agora, criamos uma nova variável para carregarmos todos os stopwords da língua inglesa, que será utilizada como referência para removermos artigos ou conjunções dos textos. Em seguida definimos dois novos Dataframes, sendo:

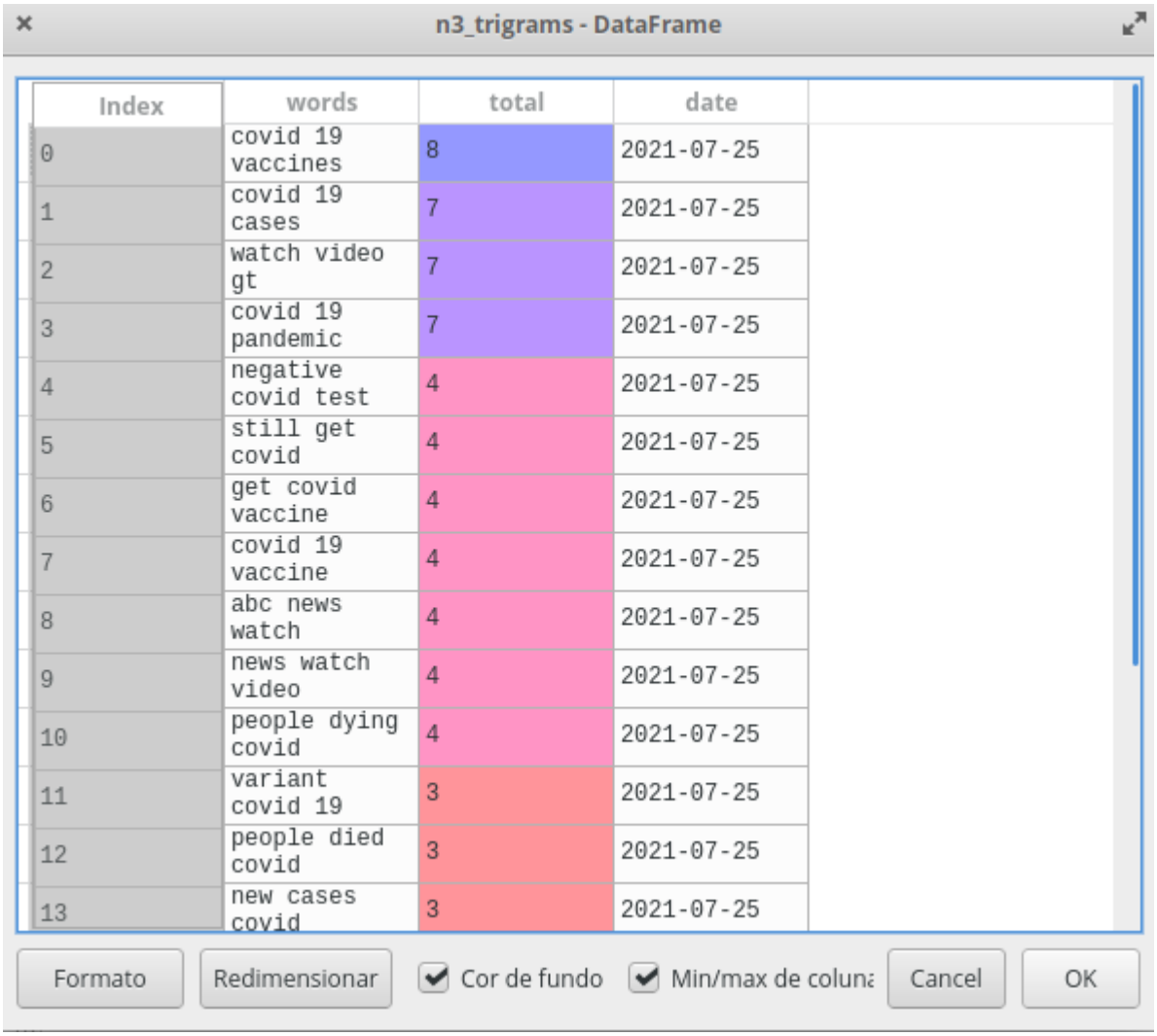
n2_bigrams: Armazenamos as top20 duas palavras mais comentadas

n3_trigrams: Armazenamos as top20 três palavras mais comentadas

```
stopword = nltk.corpus.stopwords.words('english')

#Function to ngram
def get_top_n_gram(corpus,ngram_range,n=None):
 vec = CountVectorizer(ngram_range=ngram_range,stop_words=stopword).fit(co
rpus)
 bag_of_words = vec.transform(corpus)
 sum_words = bag_of_words.sum(axis=0)
 words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.
items()]
 words_freq =sorted(words_freq, key = lambda x: x[1], reverse=True)
 return words_freq[:n]
#n2_bigram
n2_bigrams = get_top_n_gram(df_tweets['text'],(2,2),20)
n2_bigrams = pd.DataFrame(n2_bigrams)
n2_bigrams['date'] = date_since
n2_bigrams = n2_bigrams.rename(columns={0:"words",1:"total"})

#n3_trigram
n3_trigrams = get_top_n_gram(df_tweets['text'],(3,3),20)
n3_trigrams = pd.DataFrame(n3_trigrams)
n3_trigrams['date'] = date_until
n3_trigrams = n3_trigrams.rename(columns={0:"words",1:"total"})
```

Esse é o resultado nesse exemplo:


n2_bigrams - DataFrame
Index
words
total
date
0 covid 19 119 2021-07-24
1 get covid 27 2021-07-24
2 covid vaccine 19 2021-07-24
3 covid cases 15 2021-07-24
4 long covid 14 2021-07-24
5 died covid 14 2021-07-24
6 got covid 10 2021-07-24
7 delta variant 10 2021-07-24
8 covid deaths 9 2021-07-24
9 covid test 9 2021-07-24
10 still get 8 2021-07-24
11 getting covid 8 2021-07-24
12 19 vaccines 8 2021-07-24
13 dying covid 8 2021-07-24
Formato
Redimensionar
Cor de fundo
Min/max de colun
Cancel
OK



E por fim, selecionamos em nossa variável df_tweets, apenas as colunas em que precisamos subir em nossa tabela do banco de dados:

```
df_tweets = df_tweets[["date","tweet","text","sentiment"]]
```

Os próximos passos são referentes a carga da base. O processo alimentará três tabelas diferentes: tb_twitter_data, tb_twitter_bigrams e tb_twitter_trigrams

Primeiro passo é inserirmos os dados no bigquery. Reforçando novamente, é necessário ter uma conta no gmail, e criar um projeto no bigquery (https://console.cloud.google.com/bigquery). Após criar, é necessário gerar uma credencial de acesso (arquivo JSON), e subir no diretório dags/credentials. Eu

publiquei um artigo em meu site, explicando detalhadamente como gerar esse acesso: http://diegorabelo.com/pandas-bigquery-insert-data-in-bigquery/

Lembre de alterar a variável pk_json_input, informando a nomenclatura do arquivo JSON gerado pelo google, e o project_input, caso queira criar um outro nome para o projeto:

```
pk_json_input = directory+"credentials/tcc-pucminas-bot-analytics-
462531da5c8e.json"

project_input = "tcc-pucminas-bot-analytics"

auth = service_account.Credentials.from_service_account_file(pk_json_input
)

df_tweets.to_gbq(
    'bota-
nalytics_log.tb_twitter_data',credentials=auth, project_id=project_input,i
f_exists='append',
)

n2_bigrams.to_gbq(
    'bota-
nalytics_log.tb_twitter_bigrams',credentials=auth, project_id=project_inpu
t,if_exists='append',
)

n3_trigrams.to_gbq(
    'bota-
nalytics_log.tb_twitter_trigrams',credentials=auth, project_id=project_inp
ut,if_exists='append',
)
```

Agora carregamos as tabelas local do postgres, lembrando que precisa ter iniciado os containers do Docker (página 13):

```
databa-
se_connection_try1 = sqlalchemy.create_engine('postgresql://airflow:airflo
w@dags_postgres_1/postgres')
databa-
se_connection_try2 = sqlalchemy.create_engine('postgresql://airflow:airflo
w@0.0.0.0/postgres')

#database_connection
```

```
#df_final_teste = df_final.tail(1)
try:
    df_tweets.to_sql(con=database_connection_try1,name='tb_twitter_data',i
f_exists='append')
except:
    df_tweets.to_sql(con=database_connection_try2,name='tb_twitter_data',i
f_exists='append')

try:
    n2_bigrams.to_sql(con=database_connection_try1,name='tb_twitter_bigram
s',if_exists='append')
except:
    n2_bigrams.to_sql(con=database_connection_try2,name='tb_twitter_bigram
s',if_exists='append')

try:
    n3_trigrams.to_sql(con=database_connection_try1,name='tb_twitter_trigr
ams',if_exists='append')
except:
    n3_trigrams.to_sql(con=database_connection_try2,name='tb_twitter_trigr
ams',if_exists='append')
```

4. Análise e Exploração dos Dados

A análise exploratória dos dados está dividida em duas partes.

A primeira parte foi realizada através do script dags/covid19_analysis.py, e a outra parte está concentrada em um dashboard, em que vou disponibilizar os detalhes no tópico de apresentação.

Inicialmente vamos executar uma query no postgres local, e coletarmos as seguintes informações de covid: data, país, total de casos, total de mortes e total de vacinações:

```
import pandas as pd
import numpy as np
from datetime import *
import matplotlib.pyplot as plt
import sqlalchemy
from fbprophet import Prophet
from sklearn.linear_model import LinearRegression
import seaborn as sns
```

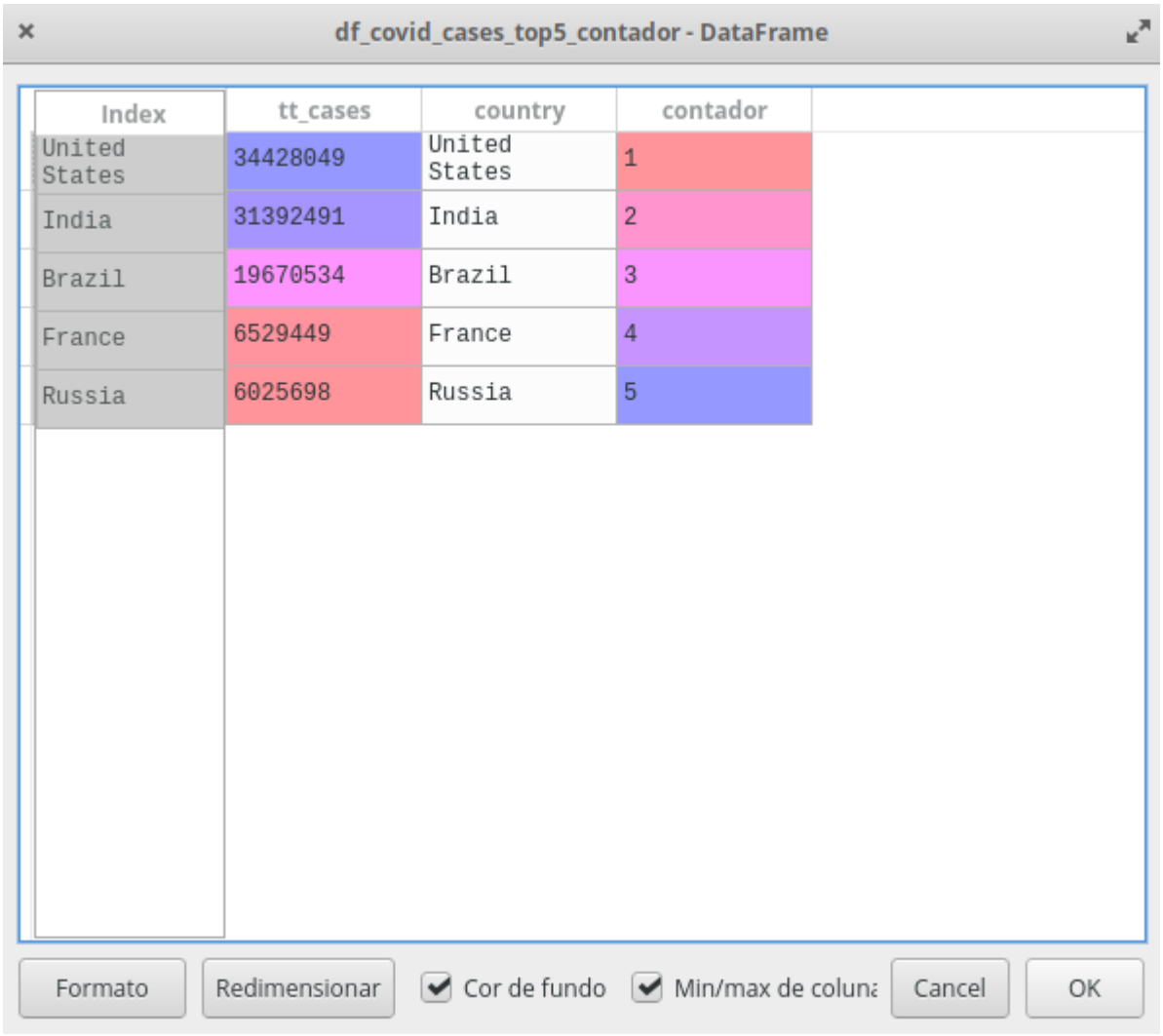
```
directory = '/usr/local/airflow/dags/img/'

#dags_postgres_1
databa-
se_connection = sqlalchemy.create_engine('postgresql://airflow:airflow@0.0.0.0
/postgres')

query = """

SELECT
    date,
    country,
    SUM(tt_cases) tt_cases,
    SUM(tt_deaths) tt_deaths,
    SUM(people_fully_vaccinated) people_fully_vaccinated
FROM tb_pandemics
WHERE category = 'covid'
GROUP BY 1,2
ORDER BY 1,2;

"""
```

df_covid - DataFrame

Index	date	country	tt_cases	tt_deaths	tt_vaccine
0	2020-01-01 00:00:00	Argentina	0	0	0
1	2020-01-01 00:00:00	Mexico	0	0	0
2	2020-01-01 00:00:00	Peru	0	0	0
3	2020-01-02 00:00:00	Argentina	0	0	0
4	2020-01-02 00:00:00	Mexico	0	0	0
5	2020-01-02 00:00:00	Peru	0	0	0
6	2020-01-03 00:00:00	Argentina	0	0	0
7	2020-01-03 00:00:00	Mexico	0	0	0
8	2020-01-03 00:00:00	Peru	0	0	0
9	2020-01-04 00:00:00	Argentina	0	0	0
10	2020-01-04 00:00:00	Mexico	0	0	0
11	2020-01-04 00:00:00	Peru	0	0	0
12	2020-01-04 00:00:00	Thailand	0	0	0
13	2020-01-05 00:00:00	Argentina	0	0	0
14	2020-01-05 00:00:00	Mexico	0	0	0
15	2020-01-05 00:00:00	Peru	0	0	0

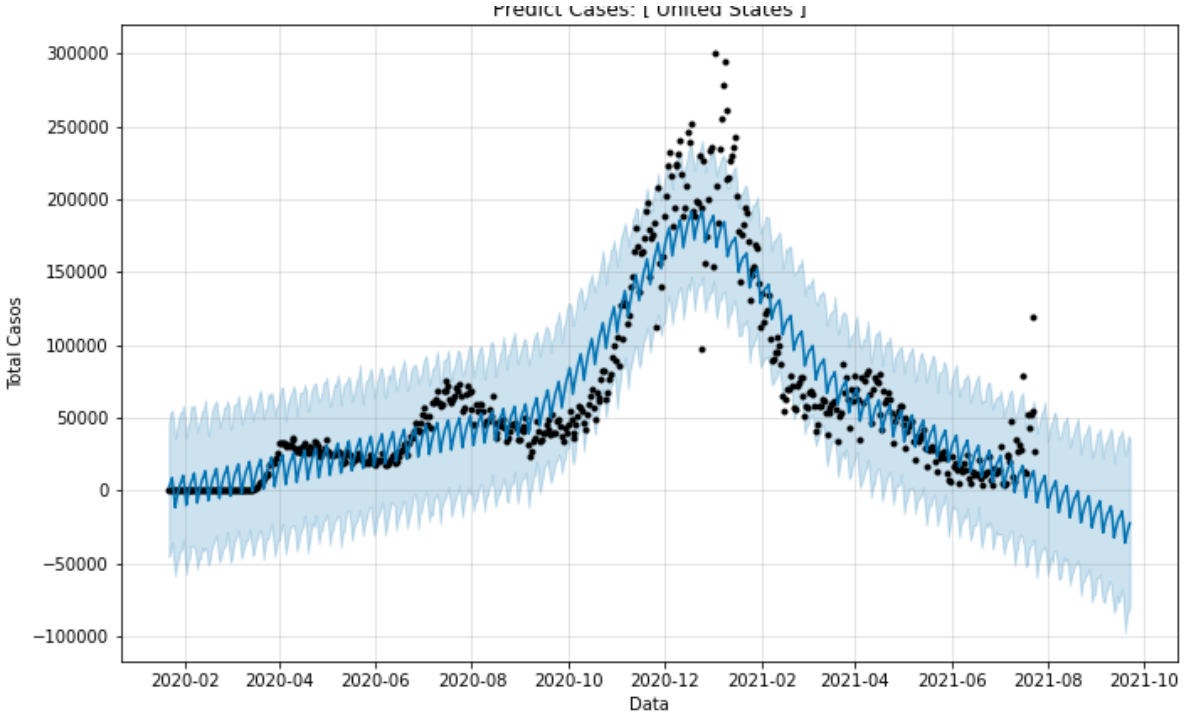
Formato | Redimensionar | Cor de fundo | Min/max de coluna | Cancel | OK

Agora vamos selecionar os cinco países com maior índice de casos:

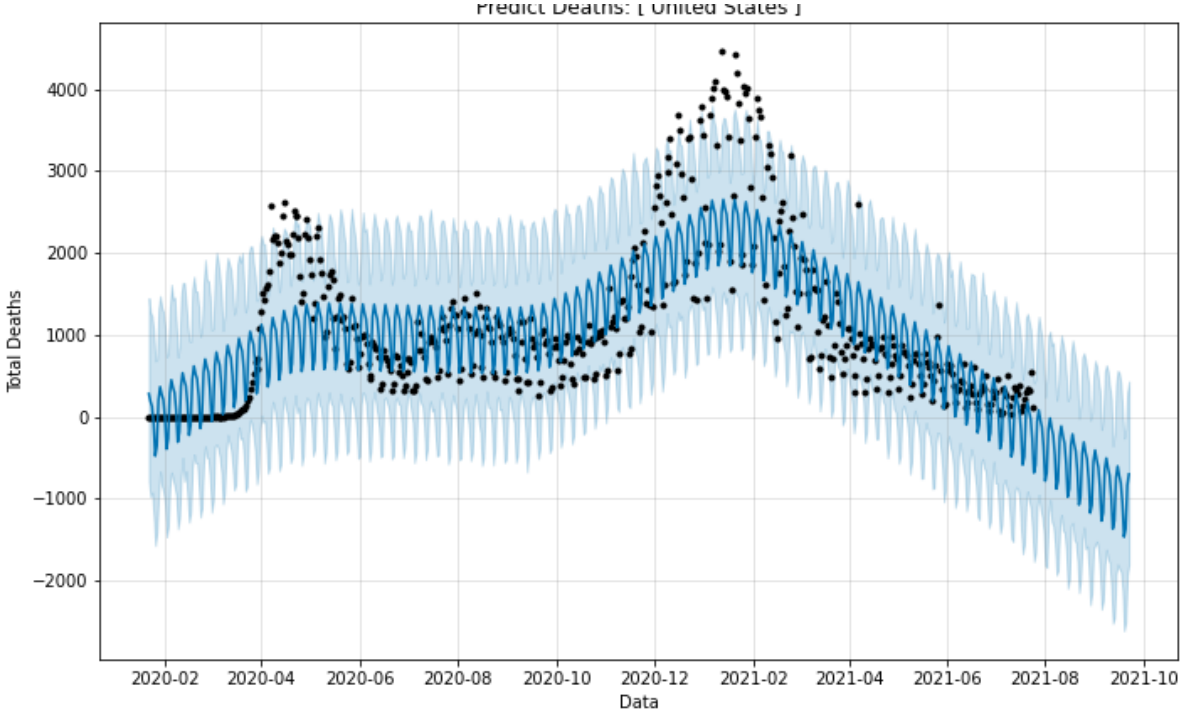


Agora o propósito é gerarmos um forecast de casos, mortes e vacinações, no período de 60 dias, utilizado o Prophet através de um laço de repetição FOR, e plotando uma imagem para cada país e para cada previsão, sendo que para um país vamos ter três gráficos (1 - previsões de casos, 2 - mortes e 3 - vacinações). Temos o seguinte resultado:

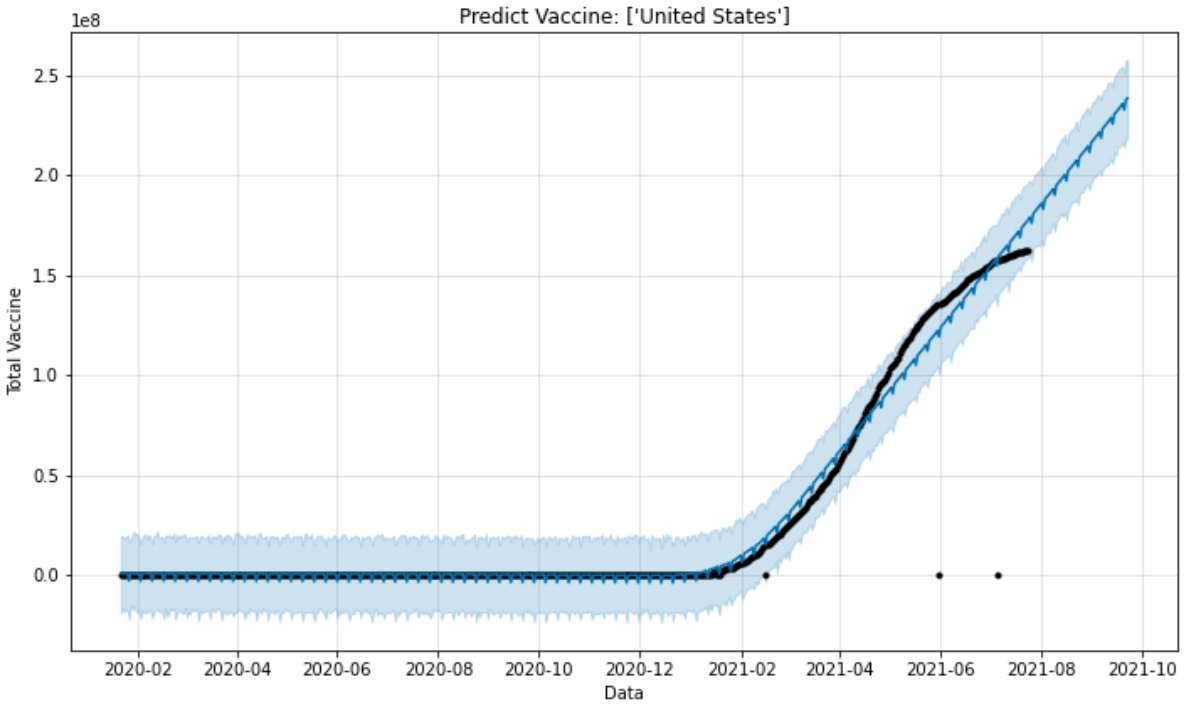
United States - Previsão de Casos 60 dias:



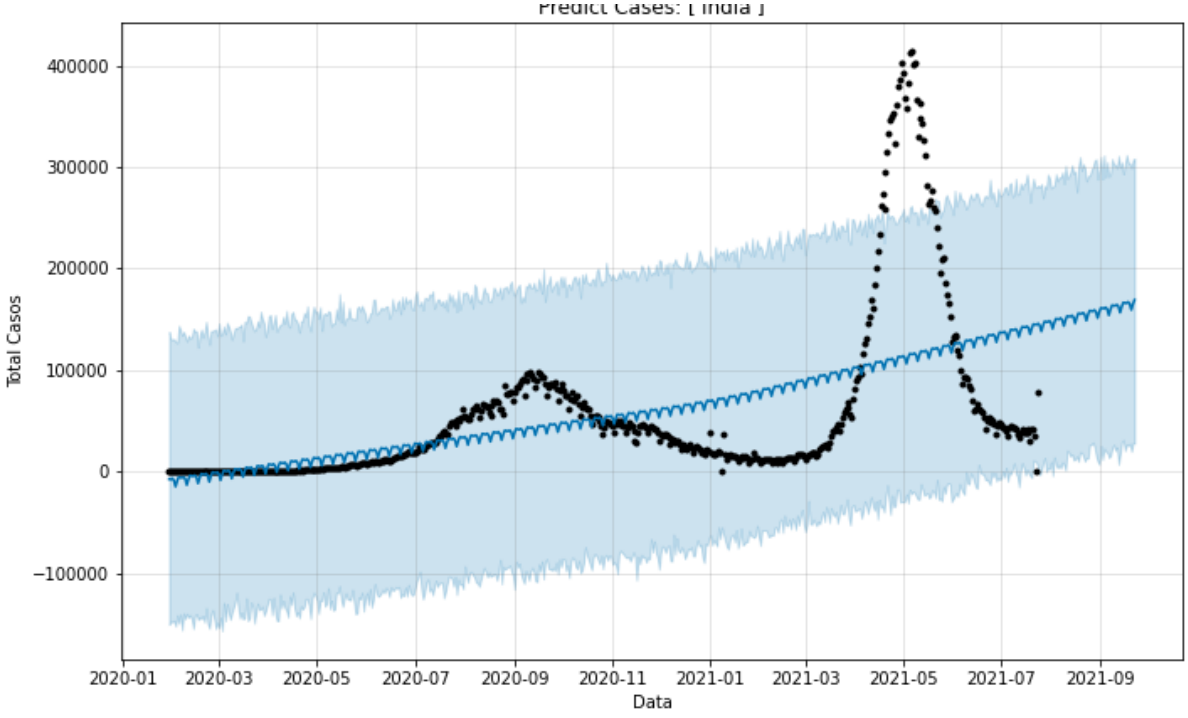
United States - Previsão de Mortes 60 dias:



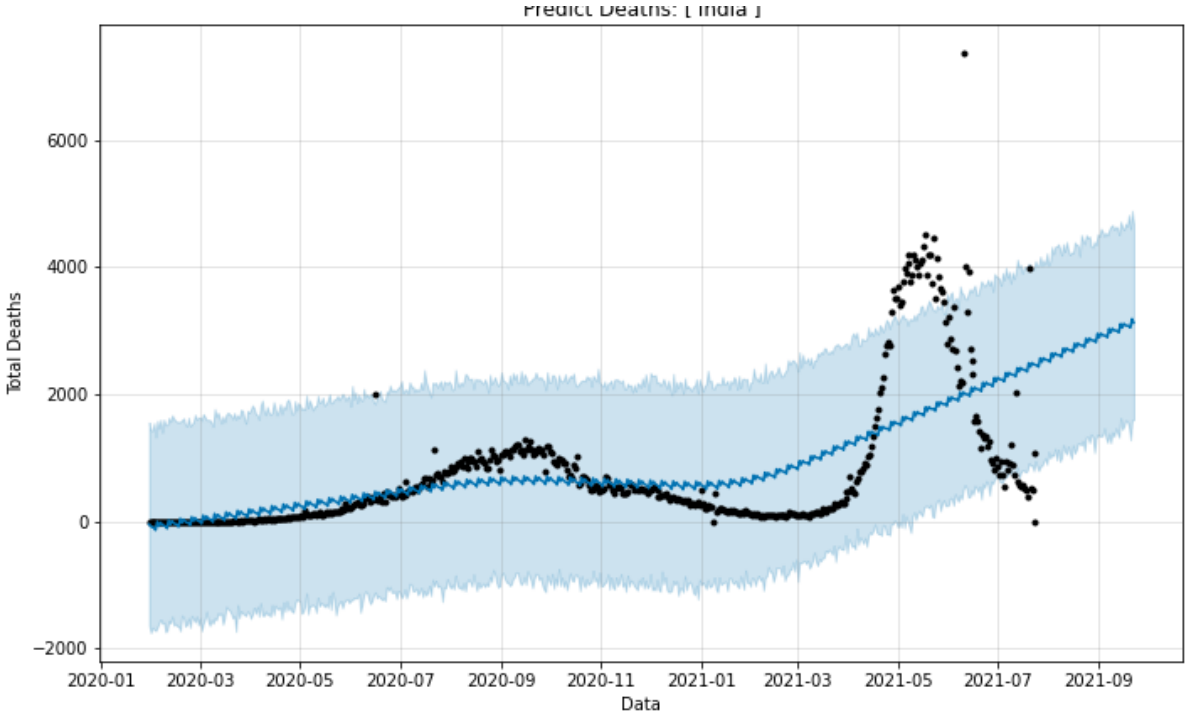
United States - Previsão de vacinações 60 dias:



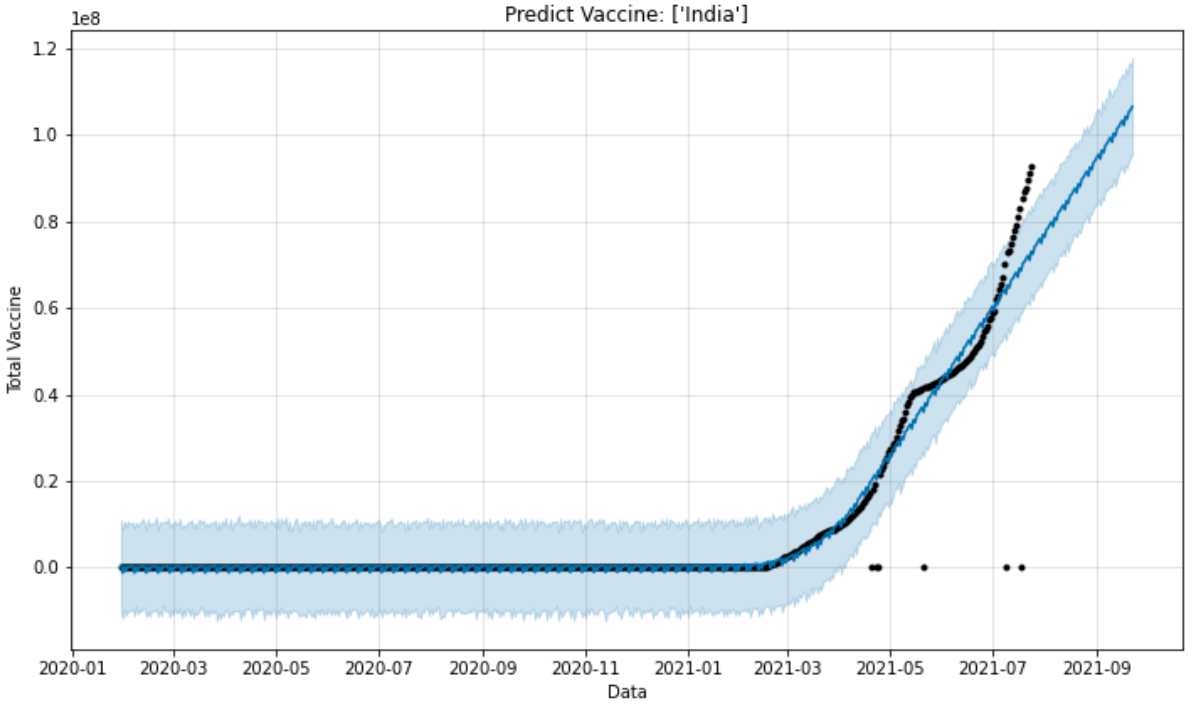
Índia - Previsão de Casos 60 dias:



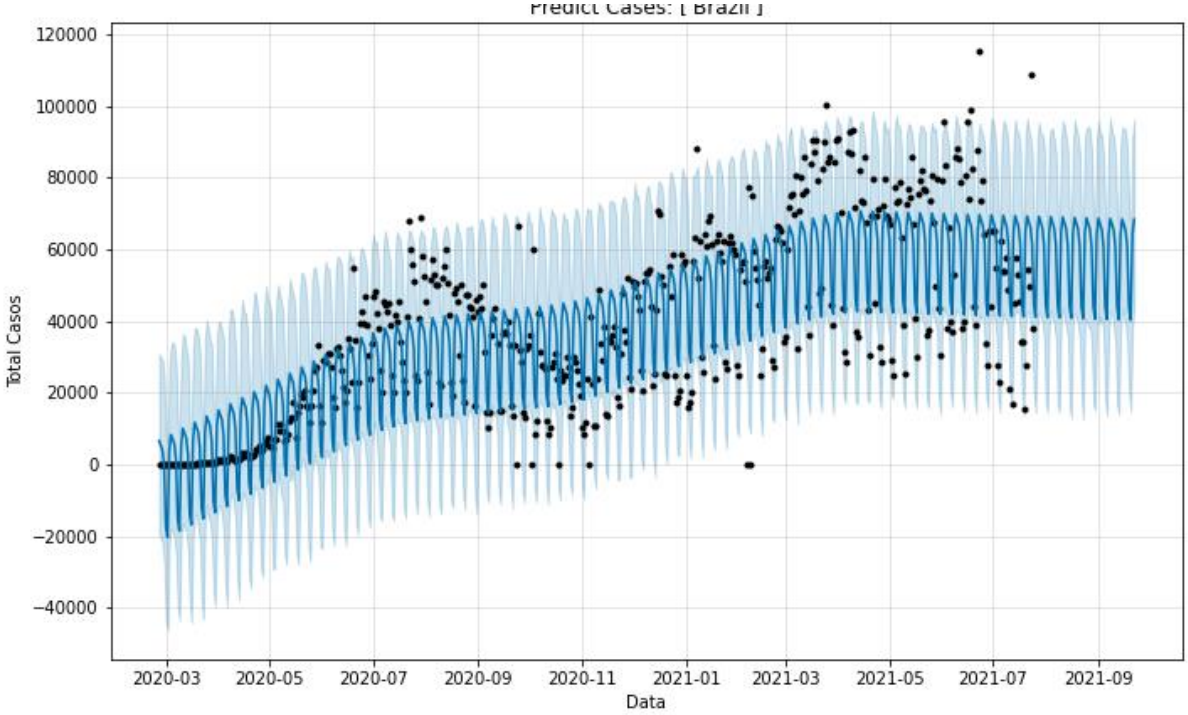
Índia - Previsão de mortes 60 dias:



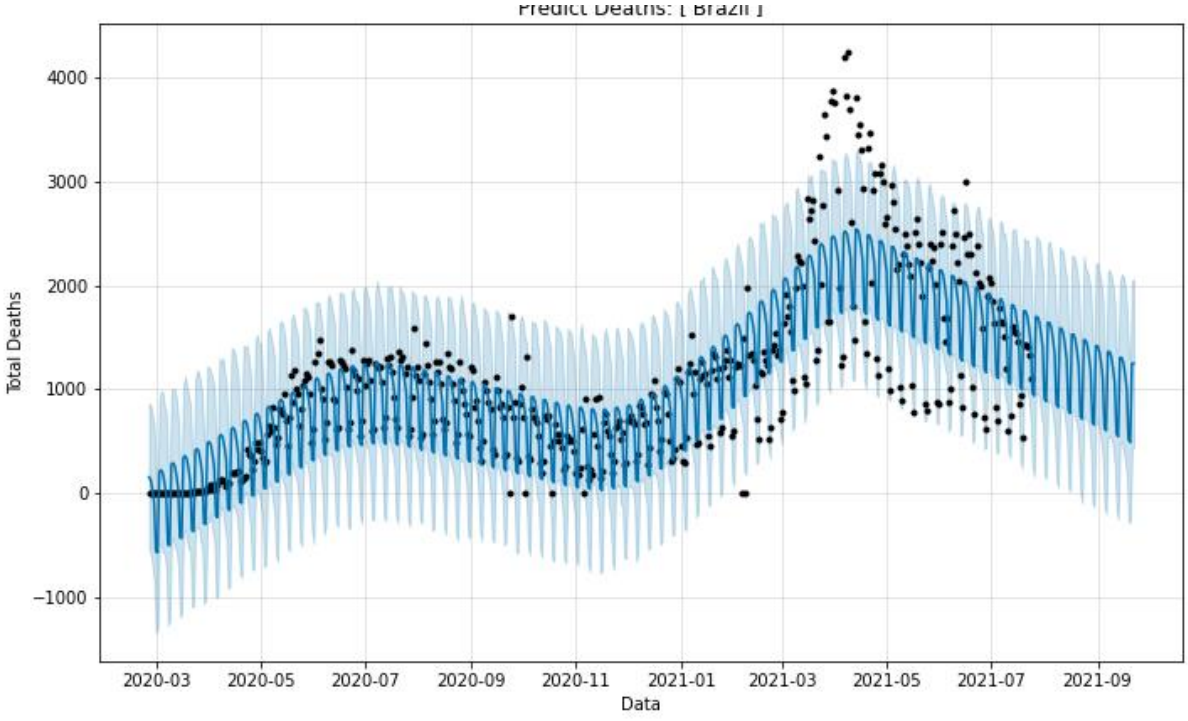
Índia - Previsão de vacinações 60 dias:



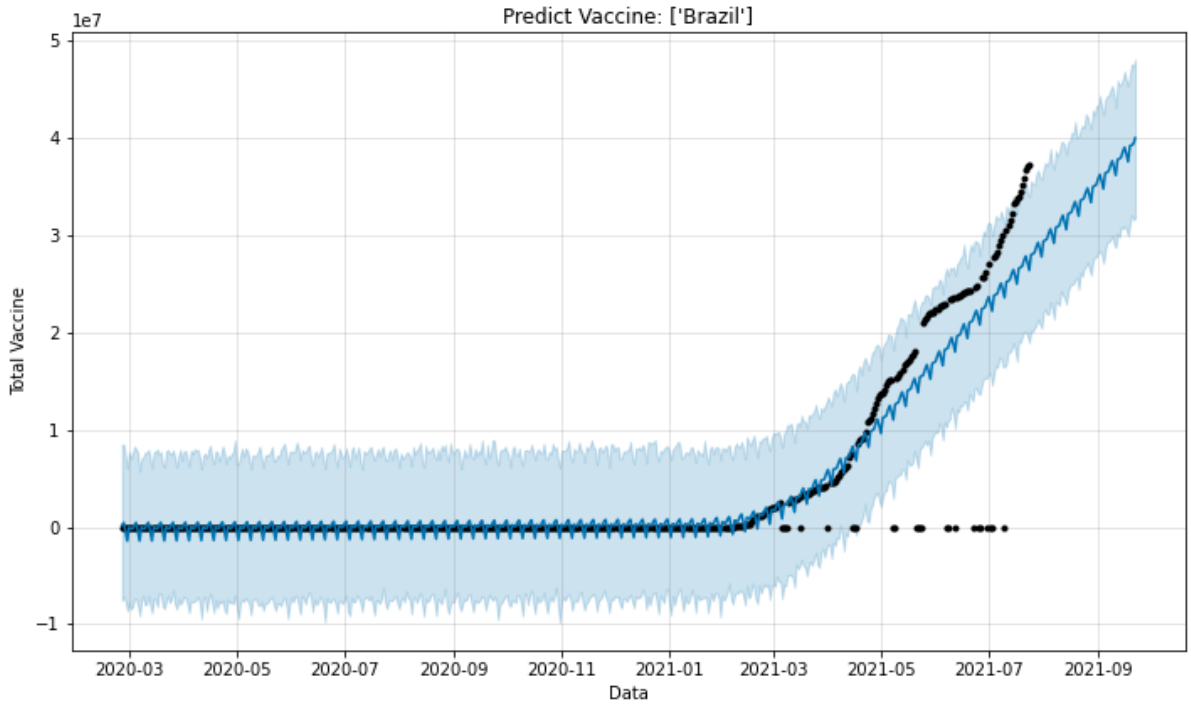
Brasil - Previsão de Casos 60 dias:



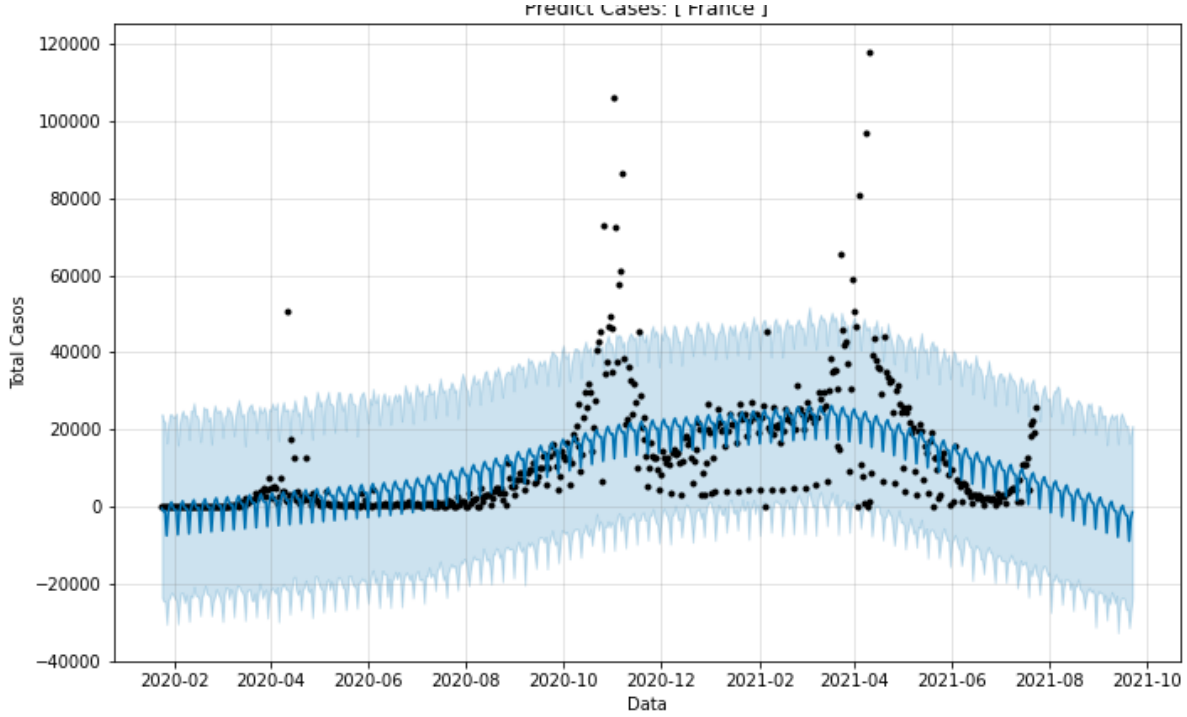
Brasil - Previsão de mortes 60 dias:



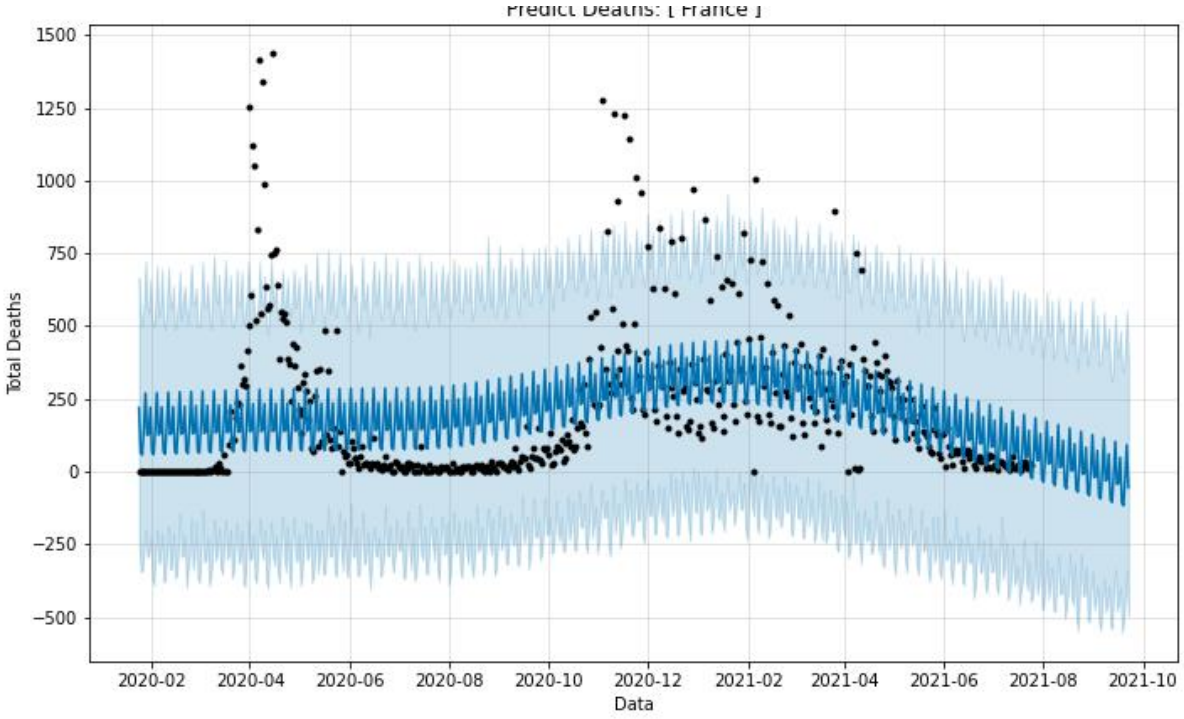
Brasil - Previsão de vacinações 60 dias:



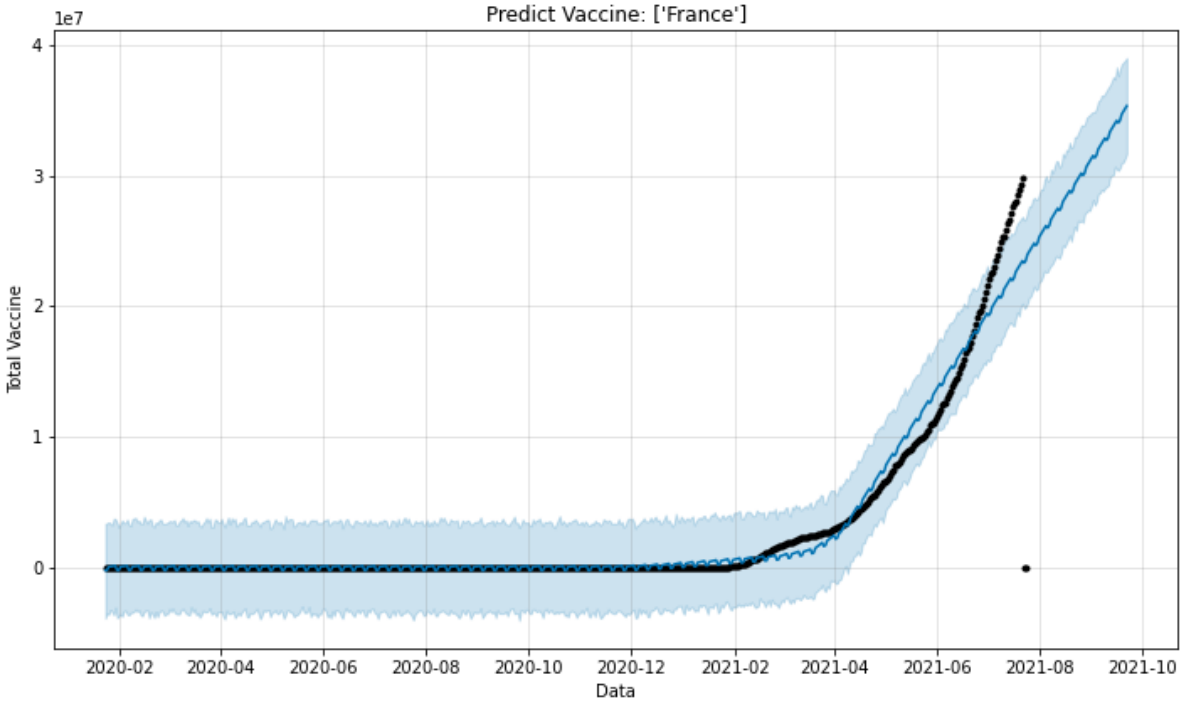
França - Previsão de Casos 60 dias:



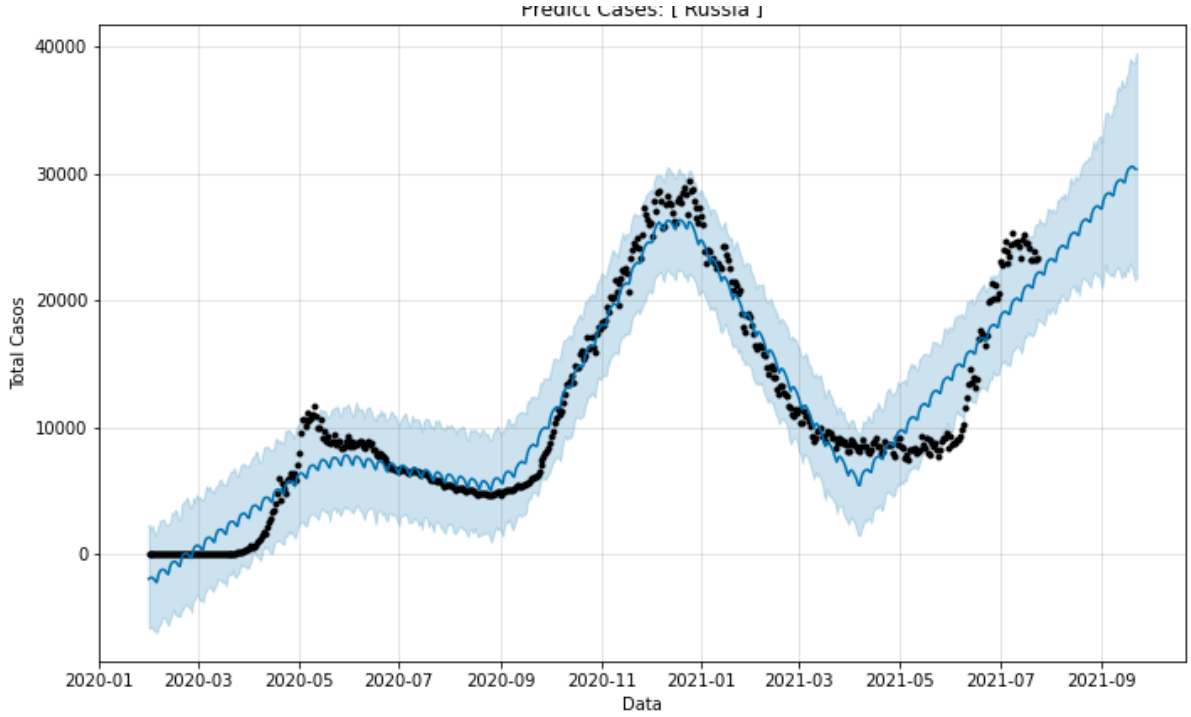
França - Previsão de mortes 60 dias:



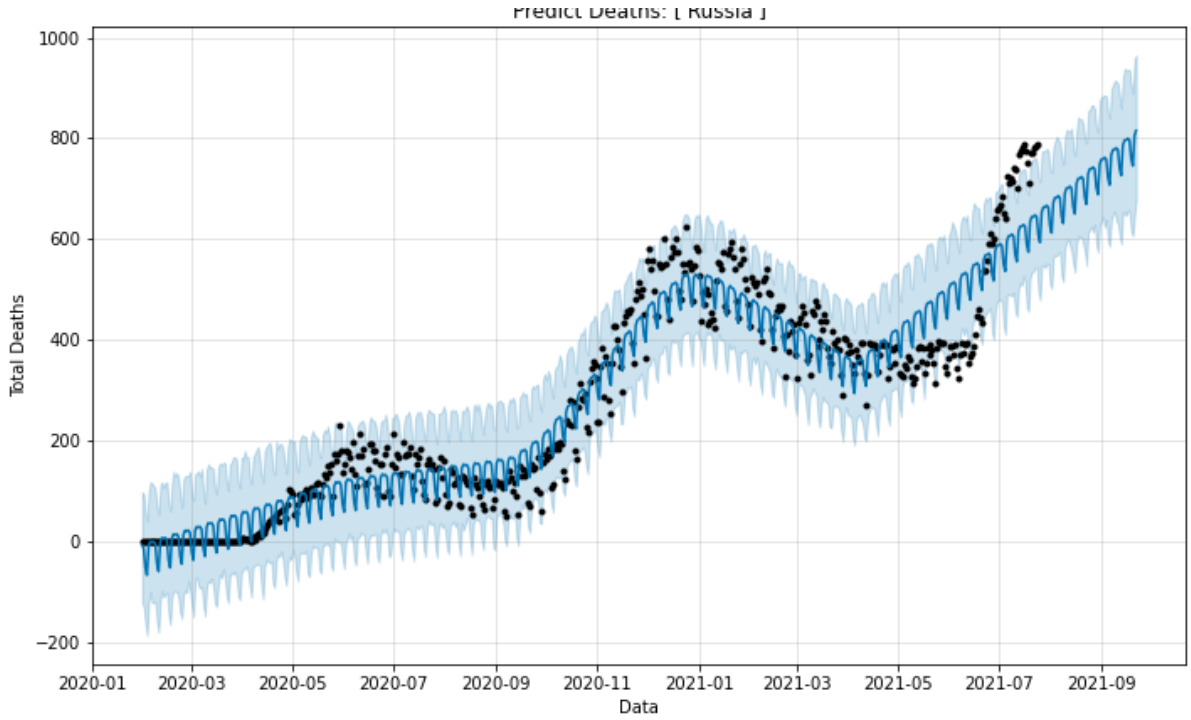
França - Previsão de vacinações 60 dias:



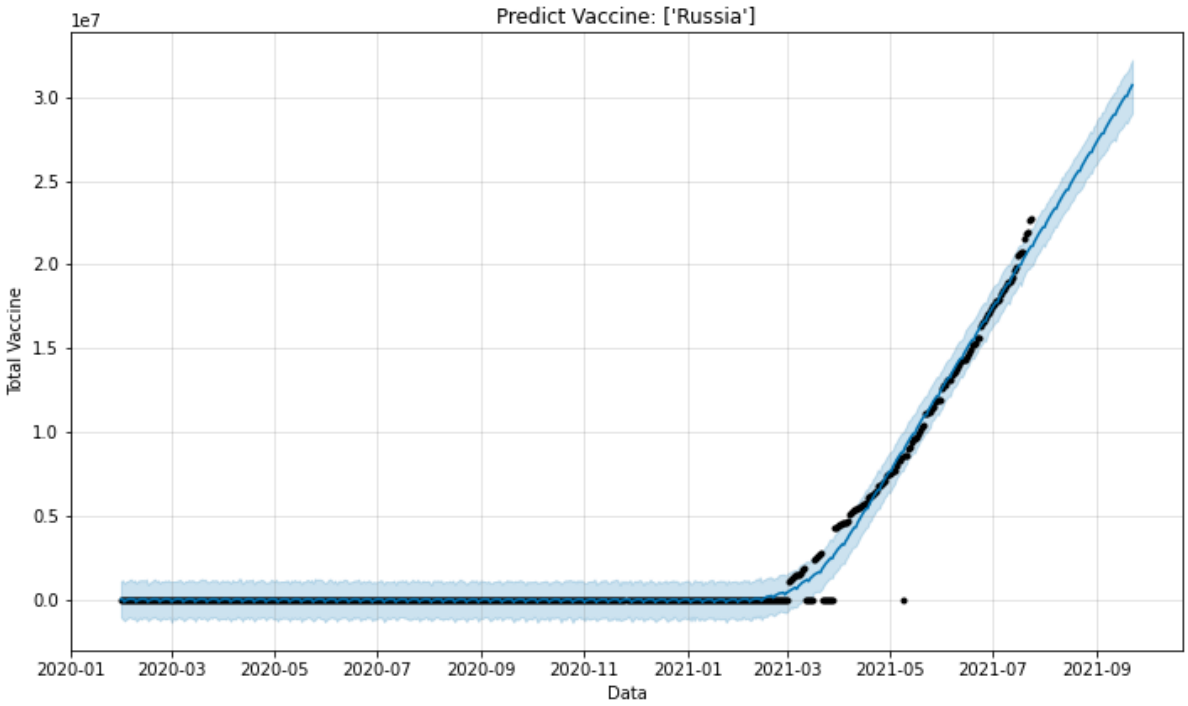
Rússia - Previsão de Casos 60 dias:



Rússia - Previsão de mortes 60 dias:



Rússia - Previsão de vacinações 60 dias:



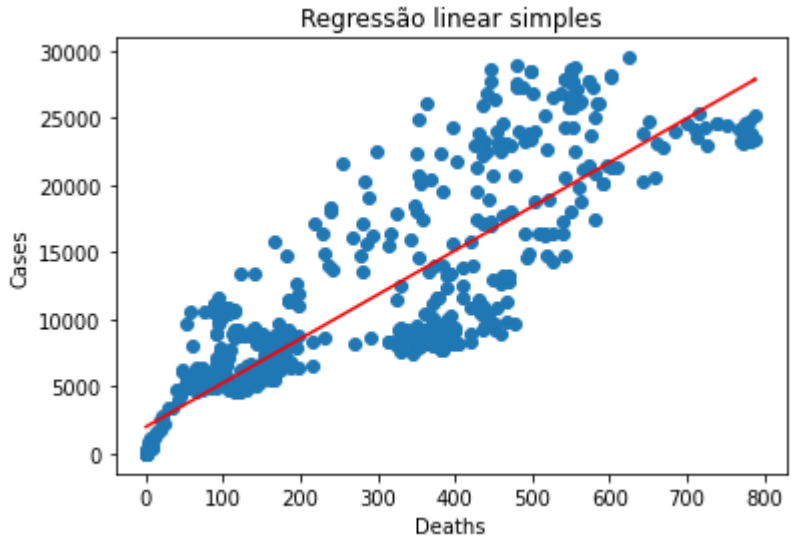
Através dessa previsão podemos perceber que dos cinco países com maior quantidade de casos, os países considerados de primeiro mundo, como Estados Unidos e França, vêm com uma tendência de queda em casos e mortes, por ter uma vacinação mais acelerada com a curva iniciando em fev/2021, algo que não ocorre com a Índia, por exemplo. Já o Brasil, tem uma tendência de queda na quantidade de mortes, porém a quantidade de casos ainda está estável, que poderia ser um alerta, pois ainda tem pessoas sendo infectadas, mesmo com a vacinação.

5. Criação de Modelos de Machine Learning

Agora que já conseguimos realizar um forecast de 60 dias com Prophet, vamos testar dois modelos de Machine Learning, utilizando a biblioteca Scikit-learn do Python. O script está localizado nesse diretório: dags/covid19_analysis.py

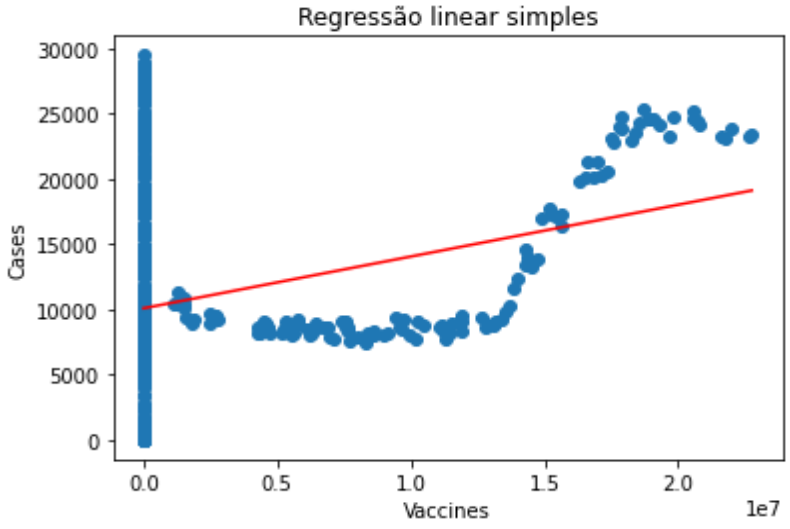
Regressão Linear – Quantidade de Casos x Quantidade de Mortes

O primeiro objetivo é analisarmos a correlação entre casos e mortes, e verificar se de fato, quanto maior a quantidade de casos, maior a quantidade de mortes. Para isso vamos utilizar a regressão linear simples:



Na regressão entre quantidade de casos x quantidade de mortes, podemos perceber que temos uma distribuição de tendência, entre quanto maior a quantidade de casos, maior a quantidade de mortes.

Regressão Linear – Quantidade de Casos x Quantidade de vacinas

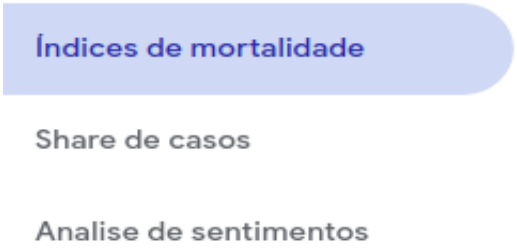


Já na regressão linear entre quantidade de casos x quantidade de vacinas, é perceptível que a linha de previsão está com leve tendencia de alta, isso ainda ocorre porque a vacinação ainda não está completamente distribuída a população até o momento, mas é notável que quanto maior a quantidade de vacina, maior a tendência da estabilização da linha.

6. Apresentação dos Resultados

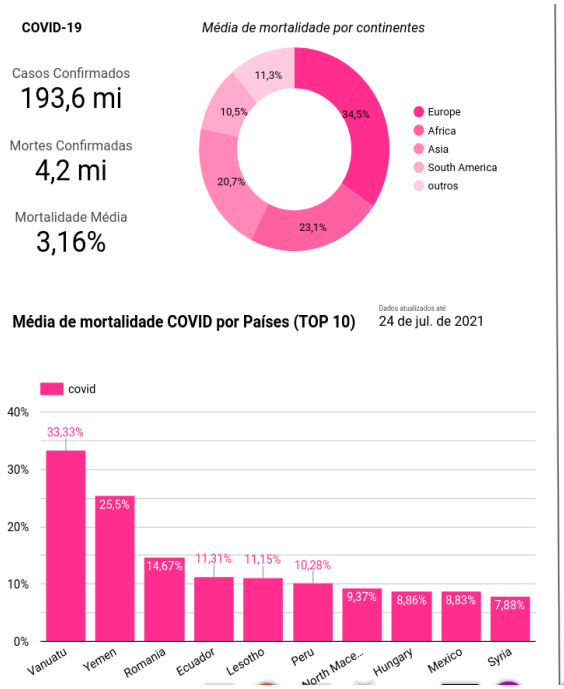
Neste trabalho optei por utilizar o dashboard Google DataStudio. É um dashboard totalmente interativo onde é possível trabalhar com filtros clicáveis, e ter uma sincronização quase que em tempo real com o bigquery.

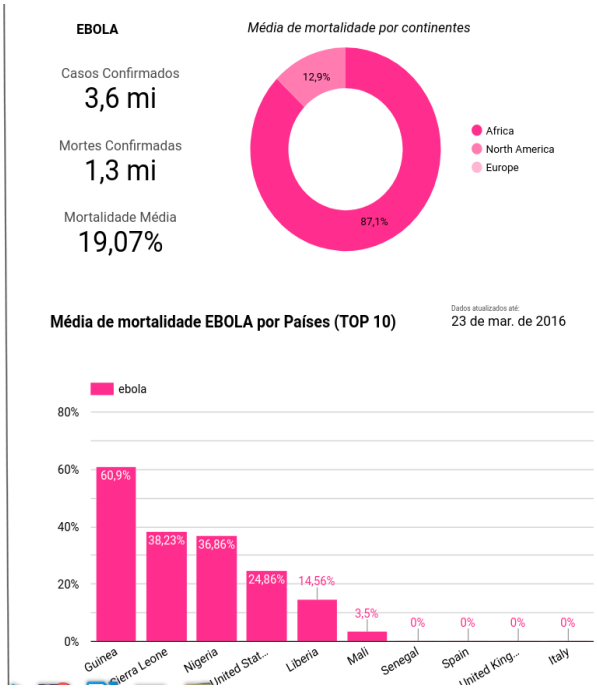
A apresentação está dividida em três painéis distintos:

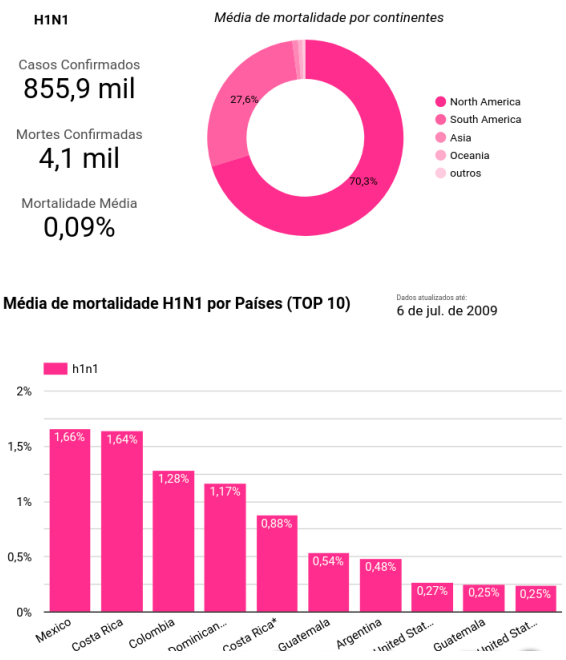


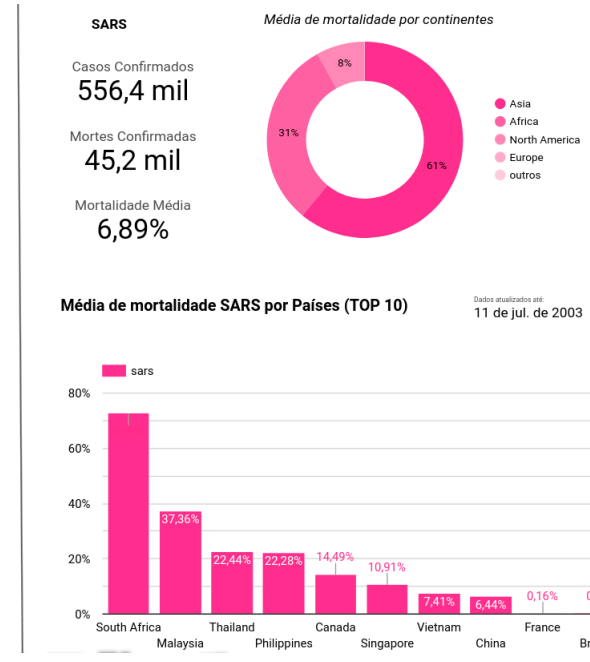
- **Índices de mortalidade - Ranking de mortalidade média por pandemias:**

Nesse primeiro painel, o objetivo é trazer um ranking de mortalidade das epidemias por países.





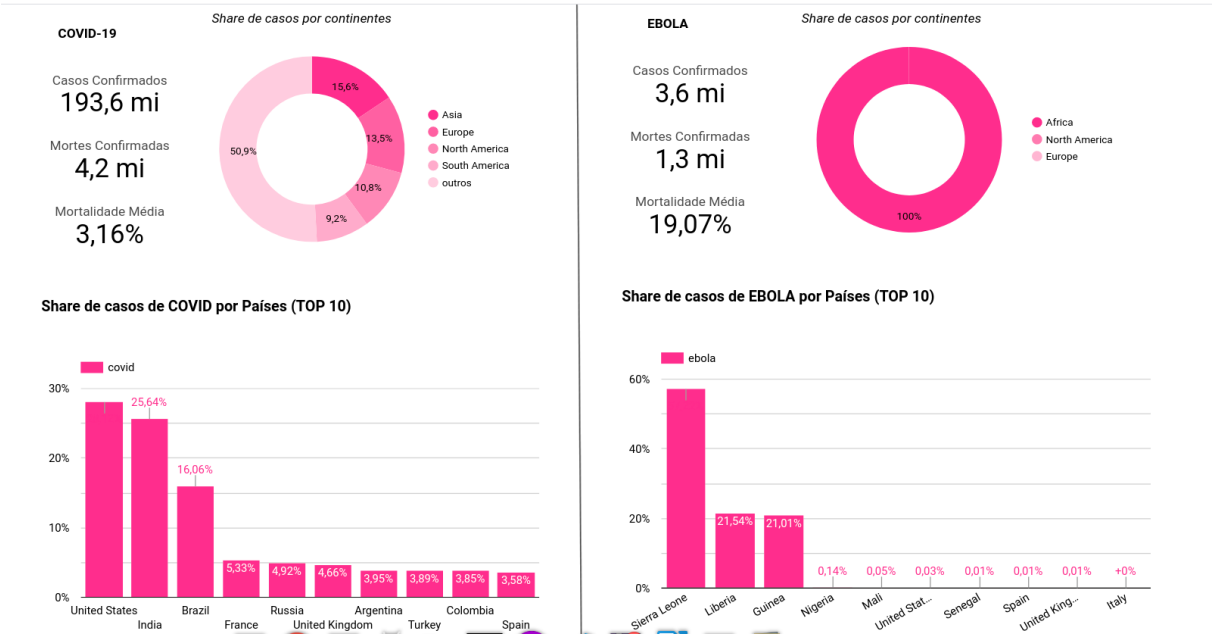


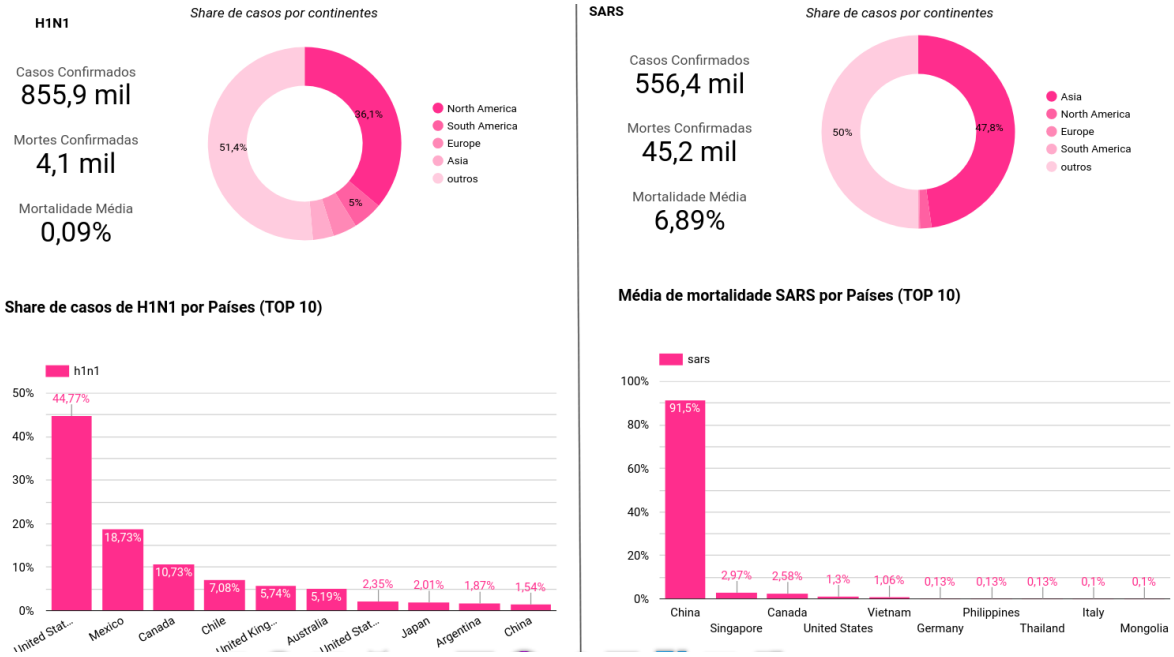


Podemos notar, que disparado o Covid-19 lidera com a maior quantidade de casos e consequentemente de mortes. Sendo os 5 países mais letais: Vanuatu, Yemen, Romênia, Equador e Lesotho. Porém não é o vírus mais letal se olharmos a proporção entre a quantidade de mortes em relação a quantidade de casos. Nesse aspecto o Ebola lidera com uma mortalidade média de 19,07%, seguido por Sars, Covid-19 e H1n1 respectivamente.

Em relação ao continente, no Covid-19 a Europa lidera com o maior índice de mortalidade, seguido por África e Asia. No Ebola, temos o continente Africano e América do Norte liderando no ranking.

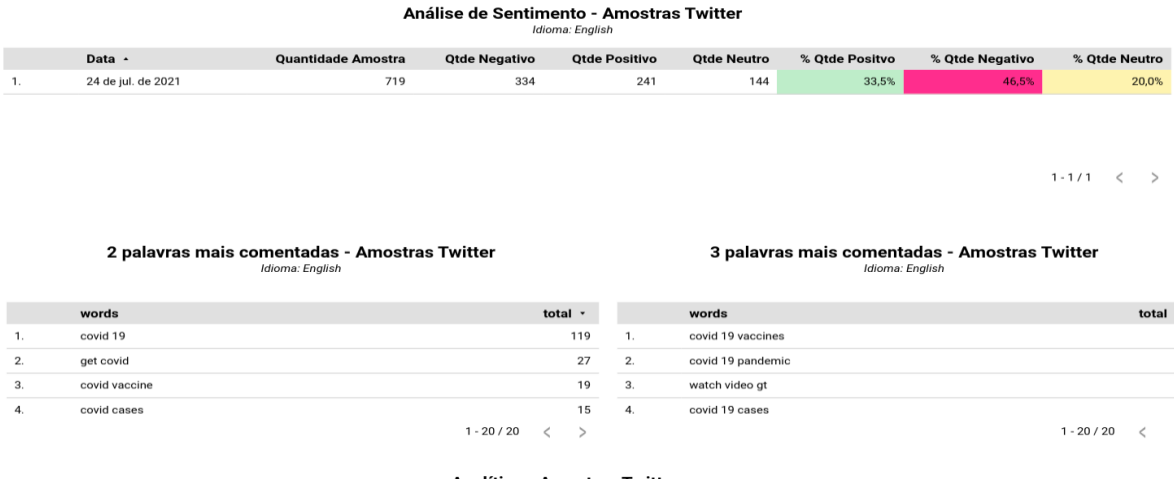
- **Share de casos – Share de casos por pandemias:**





No share de casos, podemos perceber que para todas as epidemias, a maior porcentagem é representada pelo continente Africano ou Oceania. No caso do covid é liderado pelo Estados Unidos, seguido por Índia, Brasil, França e Rússia. No Ebola, a maior representatividade está na Sierra Leone, Libéria e Guinea.

- **Análise de sentimentos:**



Na primeira parte do painel, é exibido um consolidado diário, com a quantidade de amostras, quantidade de tweets positivos, neutros e negativos.

Na parte central, temos dois componentes, onde temos as duas palavras mais comentadas do lado esquerdo, e as três palavras mais comentadas do lado direito.

Analítico - Amostras Twitter

Idioma: English

	date	text	sentiment
1.	24 de jul. de 2021 20:59:59	74 You can try this but it won't work. You can't say 97% of people have survived Covid because t...	positive
2.	24 de jul. de 2021 20:59:59	All of these dumbfucks coming in for covid that refused their shots. Go fuck yourselves. We will...	negative
3.	24 de jul. de 2021 20:59:59	60 Pneumonia is a symptom of covid. COUGHING IS NOT COVID! See how dumb that sounds?	negative
4.	24 de jul. de 2021 20:59:58	_Fury It's been postponed until October. Tyson fury got covid.	negative
5.	24 de jul. de 2021 20:59:57	23 Yes Kevin, of course ignoring it and going back to normal will make Covid go away 🤦‍♀️	neutral
6.	24 de jul. de 2021 20:59:56	70241 3ar It depends on how crowded the hospital is. The more Covid cases the less care and ...	negative
7.	24 de jul. de 2021 20:59:56	21 _free Lol 1 Covid related death in AUS this year get ya facts right before spreading crap. Get...	negative
8.	24 de jul. de 2021 20:59:56	(WalesOnline)Criticism comes in for Sajid Javid, who suggested people 'cowered' from Covid-1...	neutral
9.	24 de jul. de 2021 20:59:56	But for these test events you need to have been double jabbed or have carried out a negative co...	negative
10.	24 de jul. de 2021 20:59:55	26687573 Welp, you're wrong. About a year from now when the bad thing you think will happen ...	negative
11.	24 de jul. de 2021 20:59:55	Look I know that technically the Olympics probably shouldn't be running at all this year but it's s...	positive
12.	24 de jul. de 2021 20:59:55	Being homophobic is not a sin at all but being queer transsexual (fake or artificial woman) is a ...	positive
13.	24 de jul. de 2021 20:59:54	Church founder says vaccines are a personal choice after congregant refuses shot and dies of ...	neutral
14.	24 de jul. de 2021 20:59:54	University of Ibadan denies COVID-19 death in its premises	negative
15.	24 de jul. de 2021 20:59:53	Yep. We have heaps of AZ. Sydney should use it. People are going to get a vaccine or they are g...	negative
16.	24 de jul. de 2021 20:59:53	"Covid kills one out of 200 people," and has killed millions to date. "There are diseases that kill o...	negative
17.	24 de jul. de 2021 20:59:52	_af 23 _News Children were 1.3%-3.6% of total reported hospitalizations, and between 0.1%-1.9...	neutral
18.	24 de jul. de 2021 20:59:51	In Massachusetts, 70% of the new Covid cases are vaccinated people. But it is the unvaxed peo...	negative

E por fim é exibido um analítico com o Twitter e a sua respectiva classificação:

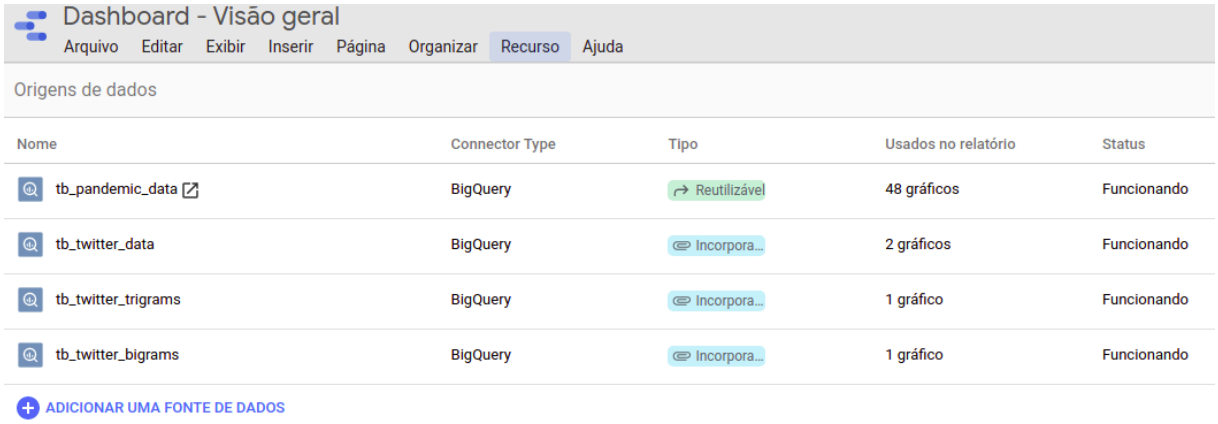
Com isso podemos perceber que de acordo com a amostra, por mais que temos avançado nas vacinações principalmente nos Estados Unidos, o sentimento negativo ainda é superior ao positivo.

6,4 Como acessar a apresentação:

A apresentação desse trabalho está disponível em um Dashboard publicado no Data Studio no link: https://datastudio.google.com/reporting/6e204ed2-6a25-496b-b03b-0edb34086605

Para replicar esse Dashboard basta ter seguido os passos anteriores, carregado as tabelas no Bigquery, e conectar o DataStudio da conta sua gmail.

No DataStudio deverá ser criado quatro conexões com o Bigquery conforme o exemplo abaixo, referente as quatro tabelas que foram carregadas anteriormente no passo de tratamento dos dados:



7. Links

Link Youtube: https://youtu.be/17pvejqF2sM

Link GitHub: https://github.com/diegorabelos/tccpucminasbotanalytics

REFERÊNCIAS

Origem da Covid, Jornal USP: https://jornal.usp.br/artigos/covid2-o-que-se-sabe-sobre-a-origem-da-doenca/#:~:text=Em%20dezembro%20de%202019%2C%20iniciou,vivos%20ou%20abatidos%20no%20local.

Iniciando o Airflow com Docker, Towards Data Science: https://towardsdatascience.com/getting-started-with-airflow-using-docker-cd8b44dbff98

Documentação da biblioteca utilizada para coleta de tweets, Twint: https://github.com/twintproject/twint/wiki