Team 17 COVID-19 Project

By: Jesse Choi, Darren Leong, Nicole Liu, Sungwoo Noh

1. Scope

We aimed to develop an effective policy plan to control and minimize COVID-19 cases and deaths during the spring period in Caladan, a country with a population of 3.2 million. We looked at ten countries, with a growth rate of deaths below 1% and new cases below 3% on a 30-day rolling average, from a dataset that reported policy restrictiveness and implementation frequency. The goal was to find the most unrestrictive policies associated with the growth rate of deaths < 1% and new cases < 3% on a 30-day rolling average. With this objective, we applied Power BI visualizations to the data, giving us a basis for our analyses, which are elaborated on below.

2. Data exploration

We started with having raw data in three areas, SQL Server, Azure VM, and CosmosDB, which we extracted and loaded into Data Lake storage. Within this process, we cleaned the data by removing superfluous columns. We then transformed the raw data and put it into an ODS that we made. In the ODS, we changed the data to be more usable in the data lake, which helps keep the Azure Data Warehouse updated.

When designing the schema, we added another column labeled SID, which concatenated “Updated” and “ISO3” for all of the metrics data, and concatenated “Date” and “CountryCode,” for the policy data. This improved our schema since it eliminated our many-to-many relationships. From here, we could visualize and analyze the data.

3. Define and explain statistics (business and statistical understanding)

In Power BI, we compared the average growth rates of deaths and cases for each policy and plotted these trends against the average rates of all policies combined. This visual shows what these average growth rates were at what point in time, as well as the restrictiveness of each policy during said point in time. The diagram helped us determine which policies were in place during the lowest average growth rate of deaths and cases and how restrictive the policies were during that period.

On a business level, these statistics give us insight into reaching our goal: determining the most unrestrictive policies when the average growth rates were lowest, specifically looking at spring. With the ability to compare the rates in the spring of different years, we recommended effective and unrestrictive policies to formulate Caladan’s optimal policy plan.

4. Statistical implementation

We used advanced statistical analyses in our project, concentrating especially on correlation matrices, in order to comprehend the interdependencies between different elements affecting COVID-19 instances and the efficacy of policy. With the help of these matrices, we were able to determine the direction and intensity of correlations between a variety of independent variables, including the growth and mortality rates of cases. As our statistical analysis sections have shown, we have observed very strong associations between countries with lower case and death increase rates in Canada and New Zealand.

With this method, we were able to identify the strategies that worked best at various points in time—especially during the spring seasons we examined. By confirming that our predictors were suitable and preventing variation, the usage of correlation matrices helped us validate the assumptions of our multiple linear regression models and increase the robustness of our results. These matrices were more than just statistical tools; they were the foundation of our data-driven decision-making process, helping us to formulate the best possible policy approach for successfully controlling the pandemic in Caladan.

5. Policy recommendations

We examined the sum of deaths per population across several countries, observing notably low figures for Korea, Japan, New Zealand, and Sweden, contrasted with notable high figures for France, Italy and Great Britain. This left us to choose between Germany and Canada, and we ultimately selected Canada since their total population (38.93 mil) is much more similar to Caladan's (3.2 mil) than Germany's (83.8 mil). Additionally, we considered New Zealand, as its total population (5.124 mil) closely resembles that of Caladan.

When looking at the springs (March-May) of the years provided, we noted the restrictiveness of each policy when the average growth rates were lowest. We found the least restrictive policies by looking at the restrictiveness value of each policy and then normalizing the value. We concluded that policies C1 (school closing), C2 (workplace closing), C5 (closing public transport), and C6 (stay-at-home requirements) best fit the objective.

These policies had the lowest restrictiveness from March through May for 2020 and 2021. When the average growth rate of deaths is below 1% and ~0%, C2, C5, C6 are the most unrestrictive in all cases. When the average growth rate of cases is below 3% and ~ 0%, C6 best fits the objective; however, looking strictly at the growth rate of cases below 3%, C2 and C5 also fulfill the objective, though nowhere near as strong as C6.

Combining the two growth rates, the optimal policy plan includes C2, C5, and C6.

6. External data enhancements and further research

We also ran a linear regression for cases and deaths growth rates for the countries New Zealand and Canada. The results can be seen in the appendix below. Unfortunately, the regression coefficients do not reflect the recommendations very well. However, this may be due to the very high correlations between the policy variables that will prevent an observation of an independent effect of each policy on the target variable.

7. Appendix

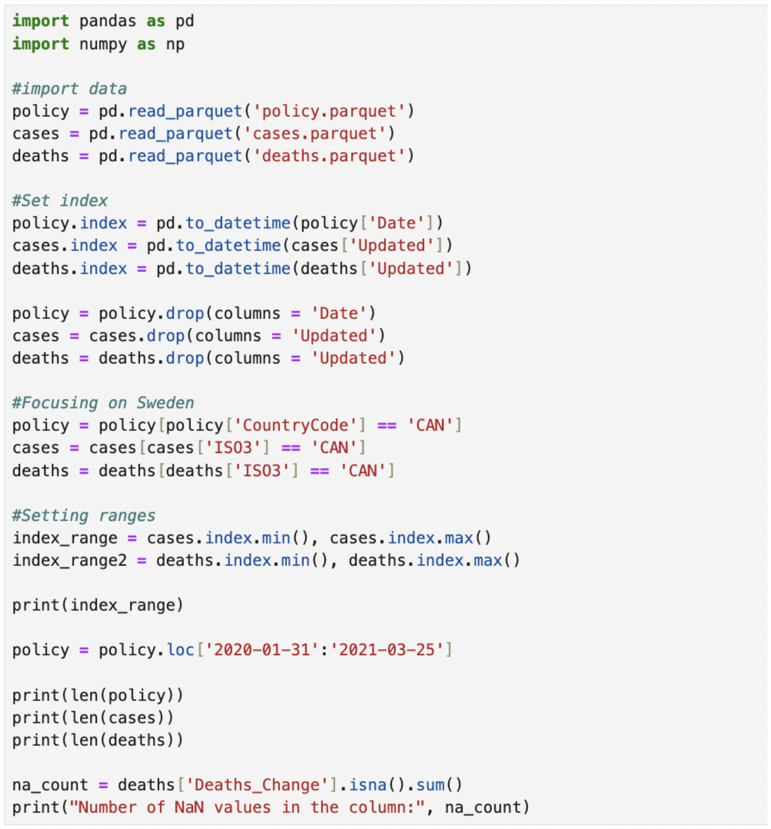
challenge #1: extracting & loading raw data
    VM & SQL server: Sungwoo Noh
    Cosmos DB: Nicole Liu
challenge #2: transforming raw data
    Dataflows & Pipeline: Nicole Liu

challenge #3: loading processed data

    External tables: Darren Leong

challenge #4: analyzing & visualizing data

    Schema: Jesse Choi, Darren Leong, Nicole Liu, Sungwoo Noh
    Correlation matrix: Darren Leong
    Data visualizations: Jesse Choi, Darren Leong, Nicole Liu, Sungwoo Noh
    Architecture: Nicole Liu
    Linear regression: Jesse Choi



```python
import pandas as pd
import numpy as np

#import data
policy = pd.read_parquet('policy.parquet')
cases = pd.read_parquet('cases.parquet')
deaths = pd.read_parquet('deaths.parquet')

#Set index
policy.index = pd.to_datetime(policy['Date'])
cases.index = pd.to_datetime(cases['Updated'])
deaths.index = pd.to_datetime(deaths['Updated'])

policy = policy.drop(columns = 'Date')
cases = cases.drop(columns = 'Updated')
deaths = deaths.drop(columns = 'Updated')

#Focusing on Sweden
policy = policy[policy['CountryCode'] == 'CAN']
cases = cases[cases['ISO3'] == 'CAN']
deaths = deaths[deaths['ISO3'] == 'CAN']

#Setting ranges
index_range = cases.index.min(), cases.index.max()
index_range2 = deaths.index.min(), deaths.index.max()

print(index_range)

policy = policy.loc['2020-01-31':'2021-03-25']

print(len(policy))
print(len(cases))
print(len(deaths))

na_count = deaths['Deaths_Change'].isna().sum()
print("Number of NaN values in the column:", na_count)
```

```python
#Data Cleaning

#Choosing columns
#Picked the most used policies, but decided to omit C2: Workspace Closing due to high correlation among other independent variables
policy_final = pd.DataFrame()

policy_final['C1'] = policy['C1_School_closing']
policy_final['C2'] = policy['C2_Workplace_closing']
policy_final['C3'] = policy['C3_Cancel_public_events']
policy_final['C4'] = policy['C4_Restrictions_on_gatherings']
policy_final['C5'] = policy['C5_Close_public_transport']
policy_final['C6'] = policy['C6_Stay_at_home_requirements']
policy_final['C7'] = policy['C7_Restrictions_on_internal_movement']
policy_final['C8'] = policy['C8_International_travel_controls']

policy_final = policy_final.drop(policy_final.index[:30]) #Removing first 30 observations to account for missing values in rolling average

#Checking for na values
policy_final.isna().value_counts()
```

```
C1     C2     C3     C4     C5     C6     C7     C8
False  False  False  False  False  False  False  False    390
Name: count, dtype: int64
```

```python
#make rolling average growth rate variables
cases_final = pd.DataFrame()
deaths_final = pd.DataFrame()

#Cases rolling average and growth rate
cases_final['30_day_rolling_average'] = cases['Confirmed_Change'].rolling(30).mean()
cases_final['cases_growth_rate'] = cases_final['30_day_rolling_average']/cases_final['30_day_rolling_average'].shift(1) - 1

#Deaths rolling average and grwoth rate
deaths_final['30_day_rolling_average'] = deaths['Deaths_Change'].rolling(30).mean()
deaths_final['deaths_growth_rate'] = deaths_final['30_day_rolling_average']/deaths_final['30_day_rolling_average'].shift(1) - 1

#Removing first 30 rows due to 0 values
cases_final = cases_final.drop(cases_final.index[:30]) #Removing first 30 observations to account for missing values in rolling average
deaths_final = deaths_final.drop(deaths_final.index[:30]) #Removing first 30 observations to account for missing values in rolling average

#To replace NaN values occuring due to denominator of 0
deaths_final['deaths_growth_rate'] = deaths_final['deaths_growth_rate'].fillna(0)

value_counts = deaths_final['deaths_growth_rate'].value_counts()

# Get the index (unique values) before sorting
original_index = value_counts.index

# Sort the counts in ascending order
counts_sorted = value_counts.sort_index(ascending=True)

value_counts = deaths_final['deaths_growth_rate'].value_counts()

#Replace the one infinite value
deaths_final['deaths_growth_rate'].loc['2020-03-11'] = 0
```

```python
# Making categorical to dummy variables

for col in policy_final.columns: # Changing non 0 values to 1
    for index, row in policy_final.iterrows():
        if row[col] != 0:
            policy_final.loc[index, col] = 1
```

```python
#Regression
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import sklearn.metrics as metrics

#Cases growth Rate
X = policy_final
y = cases_final['cases_growth_rate']

#Train test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=101)

#Reg, training
model = LinearRegression()
model.fit(X_train,y_train)

#Predict values
predictions = model.predict(X_test)

#Check accuracy
def regression_results(y_test, y_pred):

    # Regression metrics
    explained_variance=metrics.explained_variance_score(y_test, y_pred)
    mean_absolute_error=metrics.mean_absolute_error(y_test, y_pred)
    mse=metrics.mean_squared_error(y_test, y_pred)
    median_absolute_error=metrics.median_absolute_error(y_test, y_pred)
    r2=metrics.r2_score(y_test, y_pred)

    print('explained_variance: ', round(explained_variance,4))
    print('r2: ', round(r2,4))
    print('MAE: ', round(mean_absolute_error,4))
    print('MSE: ', round(mse,4))
    print('RMSE: ', round(np.sqrt(mse),4))
```

```python
#Cases growth rate
print("Regression Coefficients:", model.coef_)
regression_results(y_test, predictions)
```

```
Regression Coefficients: [-0.03609658  0.00817035 -0.1535177  -0.1535177  -0.08547517  0.00817035
 -0.0064673  -0.03609658]
explained_variance:  0.6861
r2:  0.675
MAE:  0.0343
MSE:  0.0019
RMSE:  0.0438
```

```python
#Deaths Growth Rate
X = policy_final
y = deaths_final['deaths_growth_rate']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=101)

model = LinearRegression()
model.fit(X_train,y_train)

predictions = model.predict(X_test)
```

```python
#Deaths growth rate
print("Regression Coefficients:", model.coef_)
regression_results(y_test, predictions)
```

```
Regression Coefficients: [-0.02537604 -0.03448393  0.16666667  0.16666667 -0.21263234 -0.03448393
  0.00758746 -0.02537604]
explained_variance:  0.5479
r2:  0.5439
MAE:  0.064
MSE:  0.0114
RMSE:  0.107
```