PREDICIVE ANALYTICS AND Statistical Analysis

```
#Data transformation libraries
import pandas as pd
import io
from prophet import Prophet
from google.colab import files
uploaded = files.upload()
import numpy as np

#Data visualisation libraries
import matplotlib.pyplot as plt

#ANOVA table
import scipy.stats as stats
import statsmodels.api as sm
from statsmodels.formula.api import ols
```

Choose files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving Weekly Count1.csv to Weekly Count1.csv
```

```
df1 = pd.read_csv(io.BytesIO(uploaded['Weekly Count1.csv']))
```

```
df1.dropna()
```

	Jurisdiction_code	Jurisdiction	Week_end_date	State Abbreviation	Year	Week	Cause_grp	Num_of_death	Cause_Subgroup	Time_Period	Suppress	Note	Avg_Nu

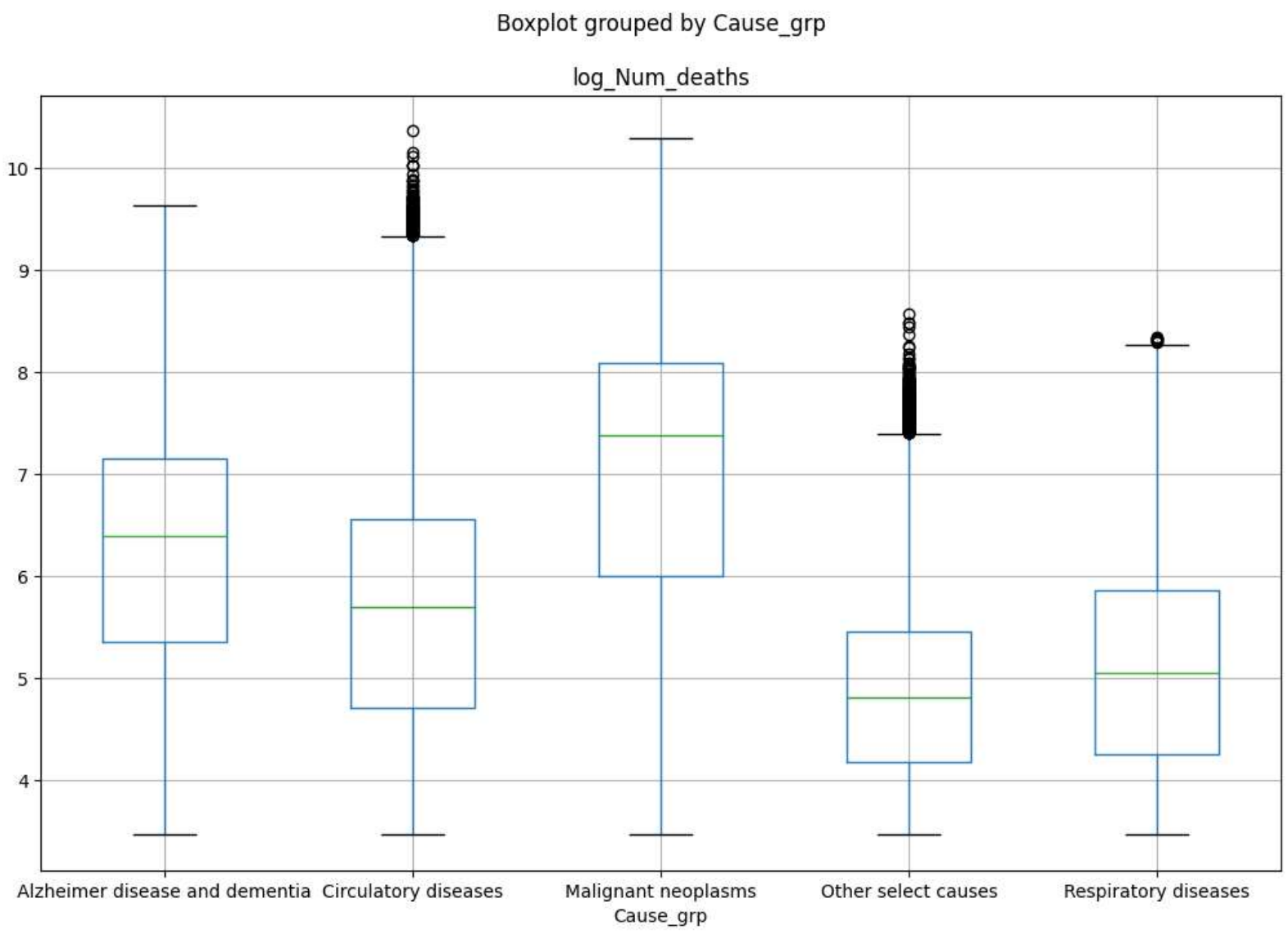
```
df1.head()
df1['log_Num_deaths'] = np.log2(df1['Num_of_death'])
```

ANOVA Implementation

```
# stats f_oneway functions takes the groups as input and returns ANOVA F and p value
fvalue, pvalue = stats.f_oneway(df1['Cause_grp_cod'],df1['log_Num_deaths'])
print(fvalue, pvalue)

df1.boxplot('log_Num_deaths', by='Cause_grp', figsize=(12, 8))

# Ordinary Least Squares (OLS) model
model = ols('Cause_grp_cod ~ C(log_Num_deaths)', data=df1).fit()
anova_table = sm.stats.anova_lm(model, typ=2)
anova_table
```

```
212768.9660780993 0.0
```

	sum_sq	df	F	PR(>F)
C(log_Num_deaths)	18799.249227	1019.0	11.271287	0.0
Residual	179341.378141	109569.0	NaN	NaN



ARIMA

```
df_arima = df1.rename(columns={'Week_end_date': 'ds', 'log_Num_deaths':'y'})
df_arima.head()
```

	Jurisdiction_code	Jurisdiction	ds	State Abbreviation	Year	Week	Cause_grp	Num_of_death	Cause_Subgroup	Time_Period	Suppress	Note	Avg_Num_(
0	1	Alabama	05-01-2019	AL	2019	1	Alzheimer disease and dementia	115	Alzheimer disease and dementia	2015-2019	NaN	NaN	
1	1	Alabama	04-01-2020	AL	2020	1	Alzheimer disease and dementia	102	Alzheimer disease and dementia	2020	NaN	NaN	
2	1	Alabama	09-01-2021	AL	2021	1	Alzheimer disease and dementia	122	Alzheimer disease and dementia	2021	NaN	NaN	
3	1	Alabama	08-01-2022	AL	2022	1	Alzheimer disease and dementia	100	Alzheimer disease and dementia	2022	NaN	Data in recent weeks are incomplete. Only 60% ...	
4	1	Alabama	07-01-2023	AL	2023	1	Alzheimer disease and dementia	113	Alzheimer disease and dementia	2023	NaN	Data in recent weeks are incomplete. Only 60% ...	

Double-click (or enter) to edit

```
grouped = df_arima.groupby('Cause_grp_cod')
for g in grouped.groups:
    group = grouped.get_group(g)
    m = Prophet()
    m.fit(group)
    future = m.make_future_dataframe(periods=365)
    forecast = m.predict(future)
    print(forecast.tail())
```

```
/usr/local/lib/python3.10/dist-packages/prophet/forecaster.py:1124: UserWarning: Parsing dates in DD/MM/YYYY format when dayfirst=False (the default) w
  self.history_dates = pd.to_datetime(pd.Series(df['ds'].unique(), name='ds')).sort_values()
/usr/local/lib/python3.10/dist-packages/prophet/forecaster.py:271: UserWarning: Parsing dates in DD/MM/YYYY format when dayfirst=False (the default) wa
  df['ds'] = pd.to_datetime(df['ds'])
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
DEBUG:cmdstanpy:input tempfile: /tmp/tmp6a0q7y5b/wim8hxxw.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmp6a0q7y5b/ex7y7i5l.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_model/prophet_model.bin', 'random', 'seed=62450', 'data', 'file=/t
15:43:38 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
15:43:38 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
/usr/local/lib/python3.10/dist-packages/prophet/forecaster.py:1124: UserWarning: Parsing dates in DD/MM/YYYY format when dayfirst=False (the default) w
  self.history_dates = pd.to_datetime(pd.Series(df['ds'].unique(), name='ds')).sort_values()
/usr/local/lib/python3.10/dist-packages/prophet/forecaster.py:271: UserWarning: Parsing dates in DD/MM/YYYY format when dayfirst=False (the default) wa
  df['ds'] = pd.to_datetime(df['ds'])
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
           ds     trend  yhat_lower  yhat_upper  trend_lower  trend_upper  \
572 2024-06-26  6.370398    4.724410    8.068993     6.360735     6.381516
573 2024-06-27  6.370510    4.655373    7.993179     6.360815     6.381692
574 2024-06-28  6.370622    4.797137    7.951655     6.360895     6.381868
575 2024-06-29  6.370735    4.693732    7.997542     6.360963     6.382043
576 2024-06-30  6.370847    4.620219    7.966485     6.361034     6.382185

     additive_terms  additive_terms_lower  additive_terms_upper    weekly  \
572       -0.003361             -0.003361             -0.003361  0.048435
573       -0.092845             -0.092845             -0.092845 -0.041753
574       -0.030379             -0.030379             -0.030379  0.020095
575       -0.039199             -0.039199             -0.039199  0.010717
576       -0.105572             -0.105572             -0.105572 -0.056183

     weekly_lower  weekly_upper    yearly  yearly_lower  yearly_upper  \
572      0.048435      0.048435 -0.051796     -0.051796     -0.051796
573     -0.041753     -0.041753 -0.051091     -0.051091     -0.051091
574      0.020095      0.020095 -0.050474     -0.050474     -0.050474
575      0.010717      0.010717 -0.049916     -0.049916     -0.049916
576     -0.056183     -0.056183 -0.049389     -0.049389     -0.049389

     multiplicative_terms  multiplicative_terms_lower  \
572                   0.0                         0.0
573                   0.0                         0.0
574                   0.0                         0.0
575                   0.0                         0.0
576                   0.0                         0.0

     multiplicative_terms_upper      yhat
572                         0.0  6.367037
573                         0.0  6.277665
574                         0.0  6.340243
575                         0.0  6.331535
576                         0.0  6.265275
DEBUG:cmdstanpy:input tempfile: /tmp/tmp6a0q7y5b/rcuoeoat.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmp6a0q7y5b/2g160p9l.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
```

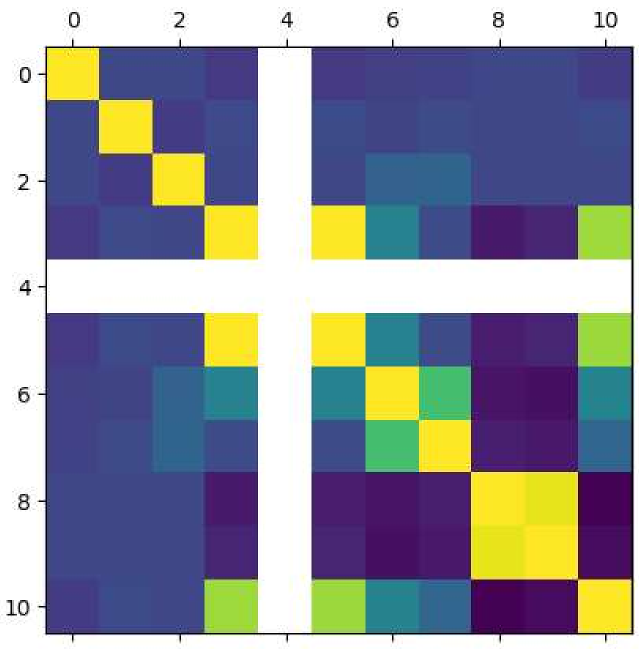
```
import pandas as pd
import statsmodels.api as sm
import math

# Load the data into a Pandas DataFrame
#df = pd.read_csv("Weekly Count1.csv")
```

```
# Set the 'Date' column as the index of the DataFrame
df_arima = df_arima.set_index("ds")

# Create a log-transformed variable 'log_total_death'
df_arima["log_total_death"] = df_arima["Num_of_death"].apply(lambda x: math.log(x))

# Identify the ARIMA model with up to 20 lags
model = sm.tsa.arima.ARIMA(df_arima["log_total_death"], order=(1,0,1))
results = model.fit()
print(results.summary())
```

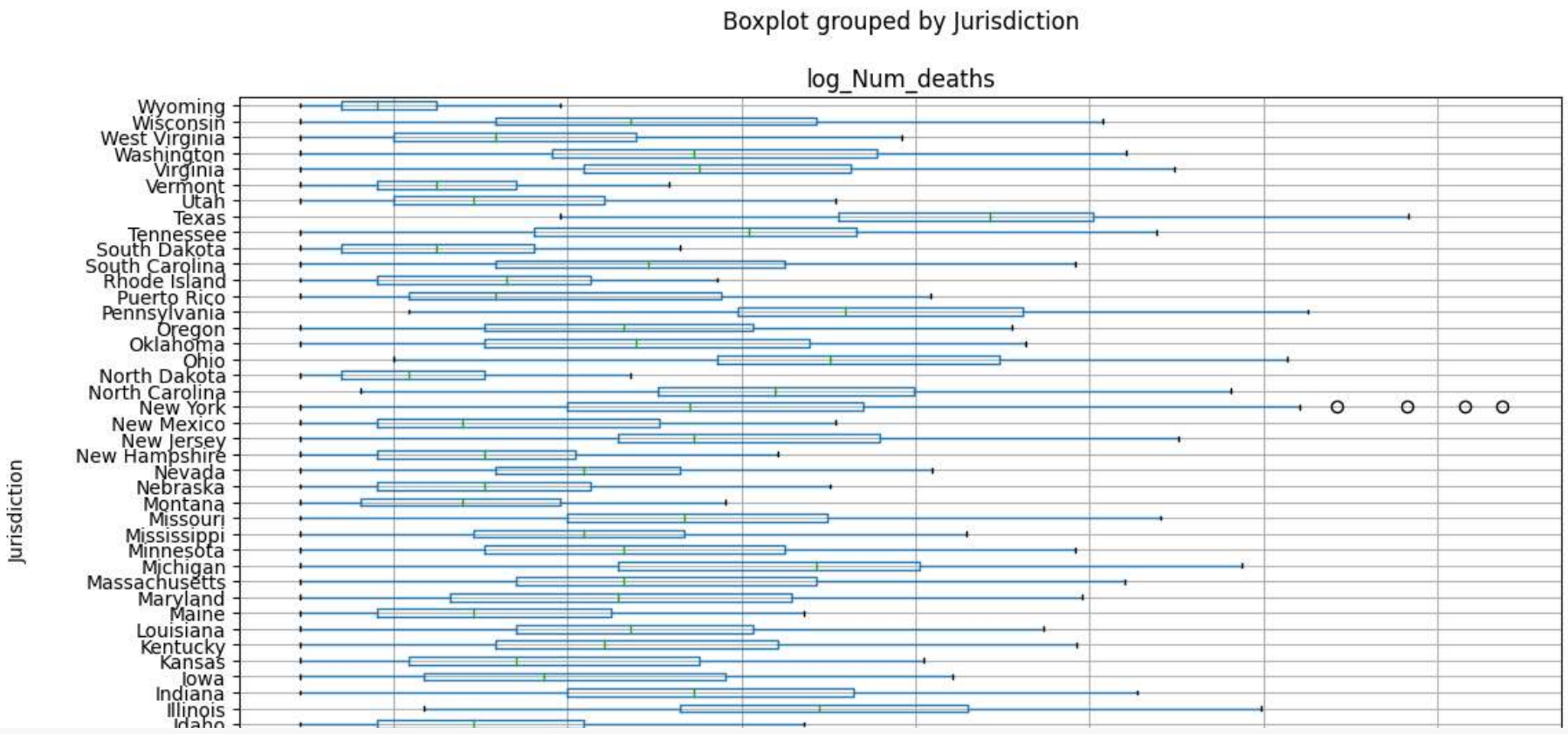
```
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:557: UserWarning: Parsing dates in DD/MM/YYYY format when dayfirst=False (the de
  _index = to_datetime(index)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning: A date index has been provided, but it has no associated freq
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning: A date index has been provided, but it is not monotonic and s
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:557: UserWarning: Parsing dates in DD/MM/YYYY format when dayfirst=False (the de
  _index = to_datetime(index)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning: A date index has been provided, but it has no associated freq
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning: A date index has been provided, but it is not monotonic and s
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:557: UserWarning: Parsing dates in DD/MM/YYYY format when dayfirst=False (the de
  _index = to_datetime(index)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning: A date index has been provided, but it has no associated freq
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning: A date index has been provided, but it is not monotonic and s
  self._init_dates(dates, freq)
                               SARIMAX Results
==============================================================================
Dep. Variable:        log_total_death   No. Observations:               110589
Model:                 ARIMA(1, 0, 1)   Log Likelihood                4452.579
Date:                Sat, 20 May 2023   AIC                          -8897.159
Time:                        15:44:28   BIC                          -8858.705
Sample:                             0   HQIC                         -8885.541
                           - 110589
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          3.9049      0.054     72.254      0.000       3.799       4.011
ar.L1          0.9946      0.000   2620.770      0.000       0.994       0.995
ma.L1         -0.6079      0.001   -588.593      0.000      -0.610      -0.606
sigma2         0.0540   9.48e-05    569.516      0.000       0.054       0.054
===================================================================================
Ljung-Box (L1) (Q):                  10.50   Jarque-Bera (JB):            878286.61
Prob(Q):                              0.00   Prob(JB):                         0.00
Heteroskedasticity (H):               1.16   Skew:                             0.01
Prob(H) (two-sided):                  0.00   Kurtosis:                        16.81
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

```
from pandas import datetime
from matplotlib import pyplot
from pandas.plotting import autocorrelation_plot

# df1.plot()
# pyplot.show()

plt.matshow(df1.corr())
plt.show()
```

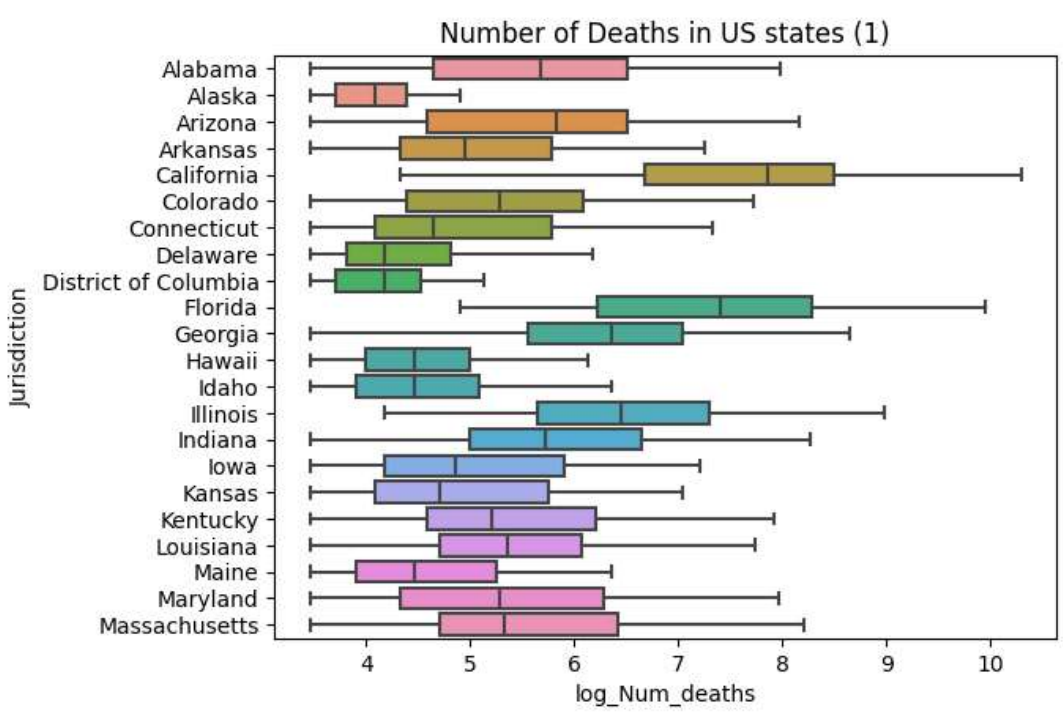
```
<ipython-input-9-dd8083cccc42>:1: FutureWarning: The pandas.datetime class is deprecated and will be removed from pandas in a future version. Import fr
  from pandas import datetime
<ipython-input-9-dd8083cccc42>:8: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will defaul
  plt.matshow(df1.corr())
```



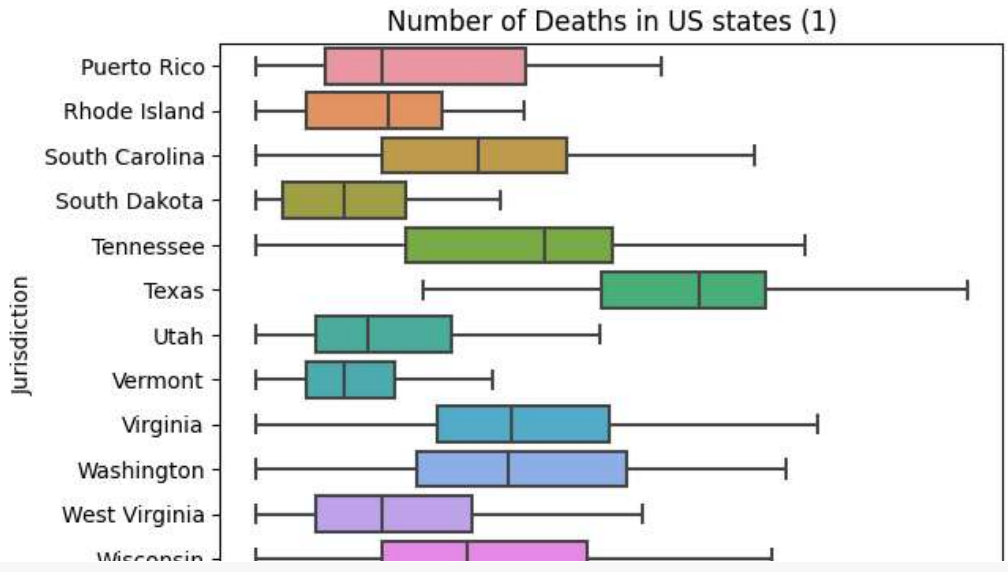

```
df1.boxplot('log_Num_deaths', by='Jurisdiction', figsize=(12, 8), vert=False)
plt.show()
```




```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Load the data into a Pandas DataFrame
#df = pd.read_csv("Weekly Count1.csv")

# Filter the data for the specified states
states = ['Alabama', 'Alaska', 'Arizona', 'Arkansas', 'California', 'Colorado', 'Connecticut',
          'Delaware', 'District of Columbia', 'Florida', 'Georgia', 'Hawaii', 'Idaho', 'Illinois', 'Indiana',
          'Iowa', 'Kansas', 'Kentucky', 'Louisiana', 'Maine', 'Maryland', 'Massachusetts']
df_filtered = df1.loc[df1['Jurisdiction'].isin(states)]

# Create a horizontal box plot of log_Num_deaths by Jurisdiction
sns.boxplot(data=df_filtered, x='log_Num_deaths', y='Jurisdiction')

# Set the title of the plot
plt.title('Number of Deaths in US states (1)')

# Display the plot
plt.show()
```




```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Load the data into a Pandas DataFrame
#df = pd.read_csv("Weekly Count1.csv")

# Filter the data for the specified states
states = ['Rhode Island', 'South Carolina', 'South Dakota', 'Tennessee',
'Texas', 'Utah', 'Vermont', 'Virginia', 'Washington', 'West Virginia', 'Wisconsin',
'Wyoming', 'Puerto Rico']
df_filtered = df1.loc[df1['Jurisdiction'].isin(states)]

# Create a horizontal box plot of log_Num_deaths by Jurisdiction
sns.boxplot(data=df_filtered, x='log_Num_deaths', y='Jurisdiction')

# Set the title of the plot
plt.title('Number of Deaths in US states (1)')

# Display the plot
plt.show()
```




```
def freq_one_variable(dataset,var1):
    f=dataset[var1].value_counts(dropna=False)
    p=dataset[var1].value_counts(dropna=False, normalize=True)
    df=pd.concat([f,p], axis=1, keys=['frequency', 'percent'])
    df["cumfrequency"] = df["frequency"].cumsum()
    df["cumpercent"] = df["percent"].cumsum()
    return df

freq_one_variable(df1,"Cause_grp")
```

	frequency	percent	cumfrequency	cumpercent
Circulatory diseases	46267	0.418369	46267	0.418369
Respiratory diseases	22470	0.203185	68737	0.621554
Other select causes	20154	0.182242	88891	0.803796
Malignant neoplasms	11228	0.101529	100119	0.905325
Alzheimer disease and dementia	10470	0.094675	110589	1.000000

```
datax = df1['Cause_grp'].value_counts().sort_index()

# Create a dataframe
datay = pd.DataFrame({
  'state': datax.index,
  'Frequency': datax.values,
  'Percent': ((datax.values/datax.values.sum())*100).round(2),
  'Cumulative Frequency': datax.values.cumsum(),
  'Cumulative Percent': ((datax.values.cumsum()/datax.values.sum())*100)\
.round(2)
})
```

```
pd.crosstab(df1["Cause_grp"],df1["Cause_Subgroup"], normalize='all')
```

Cause_Subgroup	Alzheimer disease and dementia	Cerebrovascular diseases	Chronic lower respiratory disease	Diabetes	Heart failure	Hypertensive diseases	Influenza and pneumonia	Ischemic heart disease	Malignant neoplasms	Other diseases of the circulatory system	Other diseases of the respiratory system	Rena failur
Cause_grp												
Alzheimer disease and dementia	0.094675	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
Circulatory diseases	0.000000	0.085144	0.000000	0.000000	0.068913	0.076545	0.000000	0.098889	0.000000	0.088879	0.000000	0.000
Malignant neoplasms	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.101529	0.000000	0.000000	0.000
Other select causes	0.000000	0.000000	0.000000	0.076997	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.056
Respiratory diseases	0.000000	0.000000	0.085244	0.000000	0.000000	0.000000	0.053423	0.000000	0.000000	0.000000	0.064518	0.000

```
df2 = df1.groupby(['State Abbreviation','Jurisdiction'])['Num_of_death'].sum().reset_index().rename(columns={'State Abbreviation':'State_code'})

high_mort = df2[df2.Num_of_death > 321980]
avg_mort = df2[df2.Num_of_death.between(182090,321980)]
less_mort = df2[df2.Num_of_death.between(105668,182090)]

df2.sort_values(by='Num_of_death', ascending=False)
```

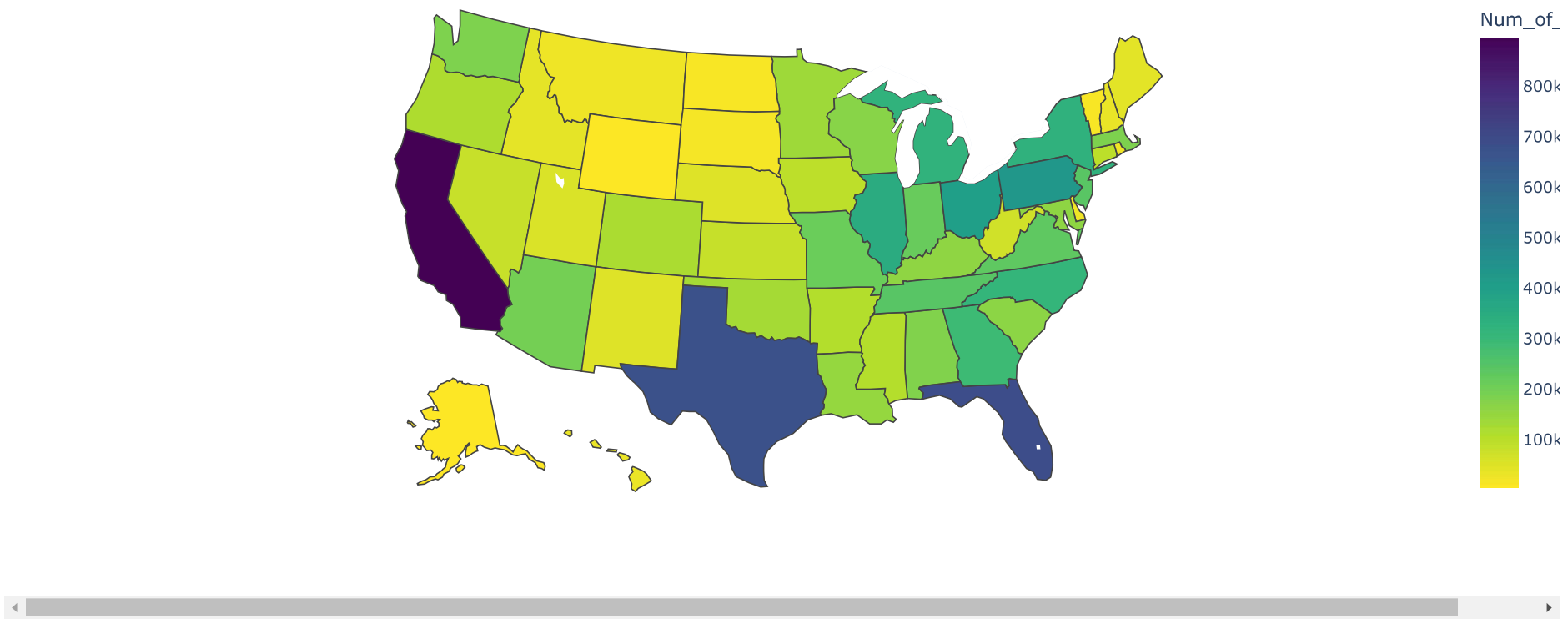
	State_code	Jurisdiction	Num_of_death
4	CA	California	895327
9	FL	Florida	686287
44	TX	Texas	672748
38	PA	Pennsylvania	424303
35	OH	Ohio	395029
14	IL	Illinois	347667
34	NY	New York	324878
22	MI	Michigan	321990
27	NC	North Carolina	308279
10	GA	Georgia	288230
43	TN	Tennessee	241627
31	NJ	New Jersey	239889
46	VA	Virginia	229966
15	IN	Indiana	213847
24	MO	Missouri	209350
3	AZ	Arizona	194160
19	MA	Massachusetts	182094
48	WA	Washington	180838
52	YC	New York	178797
1	AL	Alabama	176631
49	WI	Wisconsin	166917
41	SC	South Carolina	161111
20	MD	Maryland	157876
17	KY	Kentucky	156799
18	LA	Louisiana	149004
23	MN	Minnesota	135975
36	OK	Oklahoma	126050
5	CO	Colorado	117850
37	OR	Oregon	114023
2	AR	Arkansas	106629
25	MS	Mississippi	105668
6	CT	Connecticut	95173
39	PR	Puerto Rico	94210
12	IA	Iowa	93519
16	KS	Kansas	81163
33	NV	Nevada	79607
50	WV	West Virginia	67621
45	UT	Utah	53383
32	NM	New Mexico	48364
29	NE	Nebraska	47681
21	ME	Maine	41399
13	ID	Idaho	38656
30	NH	New Hampshire	31653

```
df2.head(10)
```

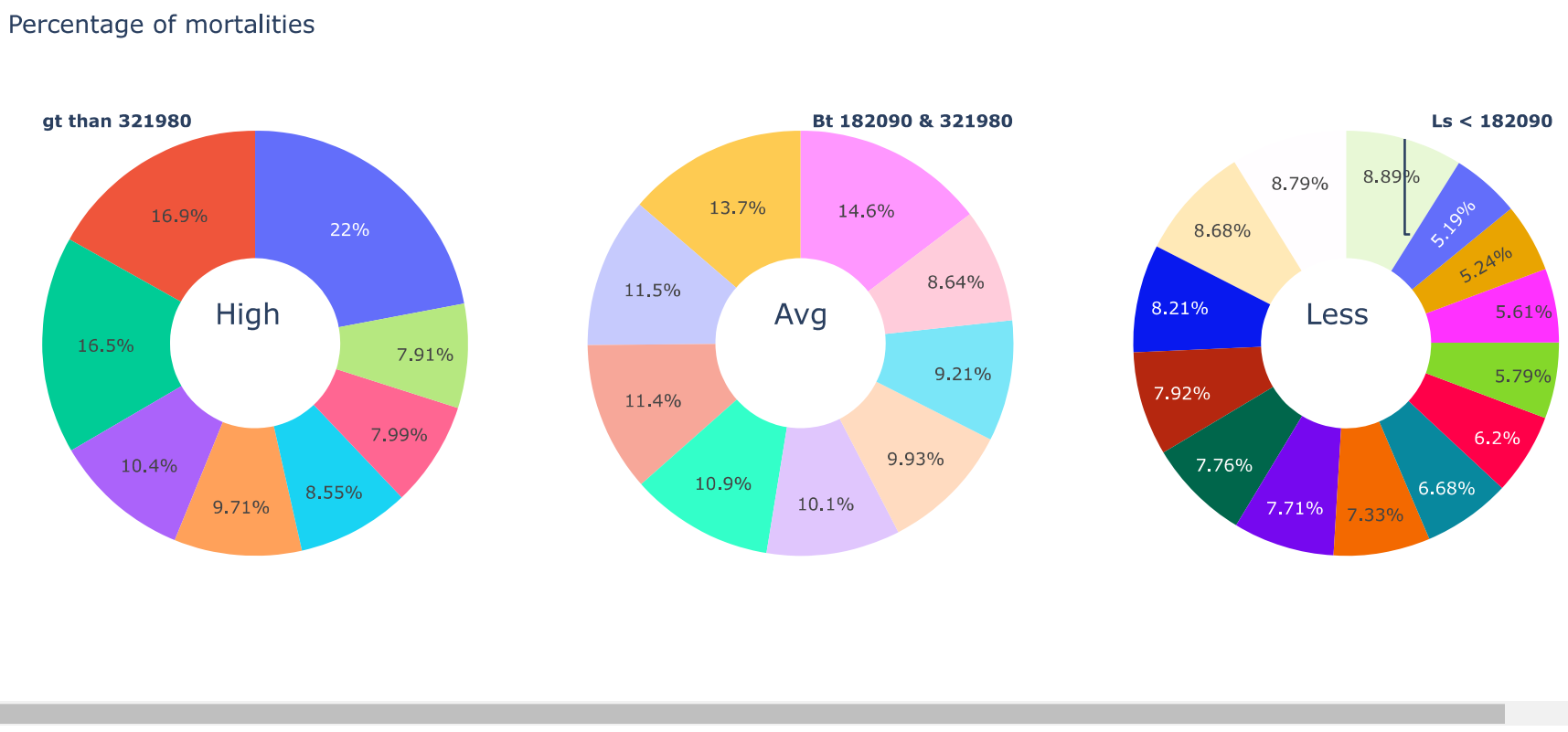
	State_code	Jurisdiction	Num_of_death
0	AK	Alaska	5303
1	AL	Alabama	176631
2	AR	Arkansas	106629
3	AZ	Arizona	194160
4	CA	California	895327
5	CO	Colorado	117850
6	CT	Connecticut	95173
7	DC	District of Columbia	9189
8	DE	Delaware	22566
9	FL	Florida	686287

```
import plotly.express as px
fig = px.choropleth(df2,
                    locations='State_code',
                    locationmode="USA-states",
                    scope="usa",
                    color='Num_of_death',
                    color_continuous_scale="Viridis_r",
```

```
      )
fig.show()
```




```
import plotly.graph_objects as go
from plotly.subplots import make_subplots

# Create subplots: use 'domain' type for Pie subplot
fig = make_subplots(rows=1, cols=3,subplot_titles=("Plot 1", "Plot 2", "Plot 3", "Plot 4"), specs=[[{'type':'domain'}, {'type':'domain'}, {'type':'domain'}]])
fig.add_trace(go.Pie(labels=high_mort['State_code'], values=high_mort['Num_of_death'], name="High Mortality",title=f"<b>gt than 321980</b>", titleposition='top
              1, 1)
fig.add_trace(go.Pie(labels=avg_mort['State_code'], values=avg_mort['Num_of_death'], name="Average Mortality",title=f"<b>Bt 182090 & 321980</b>", titleposition
              1, 2)
fig.add_trace(go.Pie(labels=less_mort['State_code'], values=less_mort['Num_of_death'], name="less Mortality",title=f"<b>Ls < 182090 </b>", titleposition='top r
              1, 3)

# Use `hole` to create a donut-like pie chart
fig.update_traces(hole=.4, hoverinfo="label+percent+name")

fig.update_layout(
    title_text="Percentage of mortalities",

    # Add annotations in the center of the donut pies.
    annotations=[dict(text='High', x=0.14, y=0.5, font_size=20, showarrow=False),
                 dict(text='Avg', x=0.50, y=0.5, font_size=20, showarrow=False),
                 dict(text='Less', x=0.85, y=0.5, font_size=20, showarrow=False)])
fig.show()
```



ANOVA Implementation

```
# stats f_oneway functions takes the groups as input and returns ANOVA F and p value
fvalue, pvalue = stats.f_oneway(df1['Cause_grp_cod'],df1['log_Num_deaths'])
print(fvalue, pvalue)
# 17.492810457516338 2.639241146210922e-05

df1.boxplot('log_Num_deaths', by='Cause_grp', figsize=(1, 8))

# Ordinary Least Squares (OLS) model
model = ols('Cause_grp_cod ~ C(log_Num_deaths)', data=df1).fit()
anova_table = sm.stats.anova_lm(model, typ=2)
anova_table
```

ARIMA

```
df_arima = df1.rename(columns={'Week_end_date': 'ds', 'log_Num_deaths':'y'})
df_arima.head()
```

	Jurisdiction_code	Jurisdiction	ds	State Abbreviation	Year	Week	Cause_grp	Num_of_death	Cause_Subgroup	Time_Period	Suppress	Note	Avg_Num_of_I
0	1	Alabama	05-01-2019	AL	2019	1	Alzheimer disease and dementia	115	Alzheimer disease and dementia	2015-2019	NaN	NaN	
1	1	Alabama	04-01-2020	AL	2020	1	Alzheimer disease and dementia	102	Alzheimer disease and dementia	2020	NaN	NaN	
2	1	Alabama	09-01-2021	AL	2021	1	Alzheimer disease and dementia	122	Alzheimer disease and dementia	2021	NaN	NaN	
3	1	Alabama	08-01-2022	AL	2022	1	Alzheimer disease and dementia	100	Alzheimer disease and dementia	2022	NaN	Data in recent weeks are incomplete. Only 60% ...	
4	1	Alabama	07-01-2023	AL	2023	1	Alzheimer disease and dementia	113	Alzheimer disease and dementia	2023	NaN	Data in recent weeks are incomplete. Only 60% ...	

```
grouped = df_arima.groupby('Cause_grp_cod')
for g in grouped.groups:
    group = grouped.get_group(g)
    m = Prophet()
    m.fit(group)
    future = m.make_future_dataframe(periods=365)
    forecast = m.predict(future)
    print(forecast.tail())
```

```
/usr/local/lib/python3.10/dist-packages/prophet/forecaster.py:1124: UserWarning:

Parsing dates in DD/MM/YYYY format when dayfirst=False (the default) was specified. This may lead to inconsistently parsed dates! Specify a format to e

/usr/local/lib/python3.10/dist-packages/prophet/forecaster.py:271: UserWarning:

Parsing dates in DD/MM/YYYY format when dayfirst=False (the default) was specified. This may lead to inconsistently parsed dates! Specify a format to e

INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
DEBUG:cmdstanpy:input tempfile: /tmp/tmp6a0q7y5b/den6hey9.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmp6a0q7y5b/yujm7sjp.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_model/prophet_model.bin', 'random', 'seed=6143', 'data', 'file=/tm
15:47:52 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
15:47:52 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
/usr/local/lib/python3.10/dist-packages/prophet/forecaster.py:1124: UserWarning:

Parsing dates in DD/MM/YYYY format when dayfirst=False (the default) was specified. This may lead to inconsistently parsed dates! Specify a format to e

/usr/local/lib/python3.10/dist-packages/prophet/forecaster.py:271: UserWarning:

Parsing dates in DD/MM/YYYY format when dayfirst=False (the default) was specified. This may lead to inconsistently parsed dates! Specify a format to e

INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
            ds     trend  yhat_lower  yhat_upper  trend_lower  trend_upper  \
572 2024-06-26  6.370398    4.691668    8.090279     6.359483     6.381081
573 2024-06-27  6.370510    4.685354    8.003597     6.359556     6.381255
574 2024-06-28  6.370622    4.655606    7.871918     6.359628     6.381428
575 2024-06-29  6.370735    4.738924    7.964804     6.359702     6.381598
576 2024-06-30  6.370847    4.678033    7.822910     6.359775     6.381763

     additive_terms  additive_terms_lower  additive_terms_upper    weekly  \
572       -0.003361             -0.003361             -0.003361  0.048435
573       -0.092845             -0.092845             -0.092845 -0.041753
574       -0.030379             -0.030379             -0.030379  0.020095
575       -0.039199             -0.039199             -0.039199  0.010717
576       -0.105572             -0.105572             -0.105572 -0.056183

     weekly_lower  weekly_upper    yearly  yearly_lower  yearly_upper  \
572      0.048435      0.048435 -0.051796     -0.051796     -0.051796
573     -0.041753     -0.041753 -0.051091     -0.051091     -0.051091
574      0.020095      0.020095 -0.050474     -0.050474     -0.050474
575      0.010717      0.010717 -0.049916     -0.049916     -0.049916
576     -0.056183     -0.056183 -0.049389     -0.049389     -0.049389

     multiplicative_terms  multiplicative_terms_lower  \
572                   0.0                         0.0
573                   0.0                         0.0
574                   0.0                         0.0
575                   0.0                         0.0
576                   0.0                         0.0

     multiplicative_terms_upper      yhat
572                         0.0  6.367037
```

```
import pandas as pd
import statsmodels.api as sm
import math

# Load the data into a Pandas DataFrame
#df = pd.read_csv("Weekly Count1.csv")
```

```
# Set the 'Date' column as the index of the DataFrame
df_arima = df_arima.set_index("ds")

# Create a log-transformed variable 'log_total_death'
df_arima["log_total_death"] = df_arima["Num_of_death"].apply(lambda x: math.log(x))

# Identify the ARIMA model with up to 20 lags
model = sm.tsa.arima.ARIMA(df_arima["log_total_death"], order=(1,0,1))
results = model.fit()
print(results.summary())
```

```
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:557: UserWarning:

Parsing dates in DD/MM/YYYY format when dayfirst=False (the default) was specified. This may lead to inconsistently parsed dates! Specify a format to e

/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information and so will be ignored when e.g. forecasting.

/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning:

A date index has been provided, but it is not monotonic and so will be ignored when e.g. forecasting.

/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:557: UserWarning:

Parsing dates in DD/MM/YYYY format when dayfirst=False (the default) was specified. This may lead to inconsistently parsed dates! Specify a format to e

/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information and so will be ignored when e.g. forecasting.

/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning:

A date index has been provided, but it is not monotonic and so will be ignored when e.g. forecasting.

/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:557: UserWarning:

Parsing dates in DD/MM/YYYY format when dayfirst=False (the default) was specified. This may lead to inconsistently parsed dates! Specify a format to e

/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning:

A date index has been provided, but it has no associated frequency information and so will be ignored when e.g. forecasting.

/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning:

A date index has been provided, but it is not monotonic and so will be ignored when e.g. forecasting.

                               SARIMAX Results
==============================================================================
Dep. Variable:        log_total_death   No. Observations:               110589
Model:                 ARIMA(1, 0, 1)   Log Likelihood                4452.579
Date:                Sat, 20 May 2023   AIC                          -8897.159
Time:                        15:48:38   BIC                          -8858.705
Sample:                             0   HQIC                         -8885.541
                             - 110589
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          3.9049      0.054     72.254      0.000       3.799       4.011
ar.L1          0.9946      0.000   2620.770      0.000       0.994       0.995
ma.L1         -0.6079      0.001   -588.593      0.000      -0.610      -0.606
sigma2         0.0540   9.48e-05    569.516      0.000       0.054       0.054
===================================================================================
Ljung-Box (L1) (Q):                  10.50   Jarque-Bera (JB):           878286.61
Prob(Q):                              0.00   Prob(JB):                        0.00
Heteroskedasticity (H):               1.16   Skew:                            0.01
Prob(H) (two-sided):                  0.00   Kurtosis:                       16.81
```