In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
from sklearn.preprocessing import LabelEncoder
```

In [2]:
```
# Load the data
data = pd.read_csv(r"C:\Users\HP\Documents\archive[1]\covid-variants.csv",
data.describe(include='all')
```

Out[2]:

	location	date	variant	num_sequences	perc_sequences	num_sequences_tot
count	100416	100416	100416	100416.000000	100416	100416.00000
unique	121	45	24	NaN	3601	Na
top	Bangladesh	2021-01-25	Alpha	NaN	0.0	Na
freq	1080	2688	4184	NaN	84248	Na
mean	NaN	NaN	NaN	72.171676	NaN	1509.58245
std	NaN	NaN	NaN	1669.262169	NaN	8445.29177
min	NaN	NaN	NaN	0.000000	NaN	1.00000
25%	NaN	NaN	NaN	0.000000	NaN	12.00000
50%	NaN	NaN	NaN	0.000000	NaN	59.00000
75%	NaN	NaN	NaN	0.000000	NaN	394.00000
max	NaN	NaN	NaN	142280.000000	NaN	146170.00000

In [3]:
```
data
```

Out[3]:

	location	date	variant	num_sequences	perc_sequences	num_sequences
0	Angola	2020-07-06	Alpha	0	0.0	
1	Angola	2020-07-06	B.1.1.277	0	0.0	
2	Angola	2020-07-06	B.1.1.302	0	0.0	
3	Angola	2020-07-06	B.1.1.519	0	0.0	
4	Angola	2020-07-06	B.1.160	0	0.0	
...	...	...	...	...	...	
100411	Zimbabwe	2021-11-01	Omicron	0	0.0	
100412	Zimbabwe	2021-11-01	S:677H.Robin1	0	0.0	
100413	Zimbabwe	2021-11-01	S:677P.Pelican	0	0.0	
100414	Zimbabwe	2021-11-01	others	0	0.0	
100415	Zimbabwe	2021-11-01	non_who	0	0.0	

100416 rows × 6 columns

In [4]:
```
data.head(15)
```

Out[4]:

	location	date	variant	num_sequences	perc_sequences	num_sequences_total
0	Angola	2020-07-06	Alpha	0	0.0	3
1	Angola	2020-07-06	B.1.1.277	0	0.0	3
2	Angola	2020-07-06	B.1.1.302	0	0.0	3
3	Angola	2020-07-06	B.1.1.519	0	0.0	3
4	Angola	2020-07-06	B.1.160	0	0.0	3
5	Angola	2020-07-06	B.1.177	0	0.0	3
6	Angola	2020-07-06	B.1.221	0	0.0	3
7	Angola	2020-07-06	B.1.258	0	0.0	3
8	Angola	2020-07-06	B.1.367	0	0.0	3
9	Angola	2020-07-06	B.1.620	0	0.0	3
10	Angola	2020-07-06	Beta	0	0.0	3
11	Angola	2020-07-06	Delta	0	0.0	3
12	Angola	2020-07-06	Epsilon	0	0.0	3
13	Angola	2020-07-06	Eta	0	0.0	3
14	Angola	2020-07-06	Gamma	0	0.0	3

In [5]:
```
data["location"]
```

Out[5]:
```
0           Angola
1           Angola
2           Angola
3           Angola
4           Angola
             ...
100411    Zimbabwe
100412    Zimbabwe
100413    Zimbabwe
100414    Zimbabwe
100415    Zimbabwe
Name: location, Length: 100416, dtype: object
```

In [17]:
```
data.isnull().sum()
```

Out[17]:
```
location              0
date                  0
variant               0
num_sequences         0
perc_sequences        0
num_sequences_total   0
dtype: int64
```

In [6]:
```
india_rows = data[data["location"] == "India"]
india_rows
```

Out[6]:

	location	date	variant	num_sequences	perc_sequences	num_sequences_tc
36360	India	2020-05-11	Alpha	0	0.0	
36361	India	2020-05-11	B.1.1.277	0	0.0	
36362	India	2020-05-11	B.1.1.302	0	0.0	
36363	India	2020-05-11	B.1.1.519	0	0.0	
36364	India	2020-05-11	B.1.160	0	0.0	
...	...	...	...	...	...	
37411	India	2021-12-27	Omicron	174	34.87	
37412	India	2021-12-27	S:677H.Robin1	0	0.0	
37413	India	2021-12-27	S:677P.Pelican	0	0.0	
37414	India	2021-12-27	others	21	4.21	
37415	India	2021-12-27	non_who	21	4.21	

1056 rows × 6 columns

In [10]:
```
india_rows["num_sequences"].value_counts()
india_rows["num_sequences_total"].value_counts()
india_rows["perc_sequences"].value_counts()
```

Out[10]:
```
0.0      846
100.0     24
0.04       4
0.09       4
0.15       3
         ...
3.87       1
1.13       1
17.59      1
70.02      1
55.57      1
Name: perc_sequences, Length: 138, dtype: int64
```

In [14]:
```
india_rows["num_sequences"].describe()
```

Out[14]:
```
count    1056.000000
mean      105.997159
std       491.531034
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max      6174.000000
Name: num_sequences, dtype: float64
```

In [15]:
```
india_rows["num_sequences_total"].describe()
```

Out[15]:
```
count    1056.000000
mean     2068.818182
std      1978.135756
min       235.000000
25%       459.000000
50%      1348.000000
75%      3239.250000
max      8086.000000
Name: num_sequences_total, dtype: float64
```

In [16]:
```
india_rows["perc_sequences"].describe()
```

Out[16]:
```
count     1056
unique     138
top        0.0
freq       846
Name: perc_sequences, dtype: object
```

In [6]:
```
india_rows[india_rows["variant"]=="Alpha"]
```

Out[6]:

	location	date	variant	num_sequences	perc_sequences	num_sequences_total
36360	India	2020-05-11	Alpha	0	0.0	471
36384	India	2020-05-25	Alpha	0	0.0	649
36408	India	2020-06-08	Alpha	0	0.0	617
36432	India	2020-06-22	Alpha	0	0.0	766
36456	India	2020-07-06	Alpha	0	0.0	276
36480	India	2020-07-20	Alpha	0	0.0	317
36504	India	2020-08-03	Alpha	0	0.0	393

In [7]:
```
import pandas as pd

# Convert the 'date' column to datetime data type
india_rows['date'] = pd.to_datetime(india_rows['date'])

# Extract the month-wise period
india_rows['month'] = india_rows['date'].dt.to_period('M')
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_14156\446966422.py:4: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-d
ocs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#return
ing-a-view-versus-a-copy)
  india_rows['date'] = pd.to_datetime(india_rows['date'])
C:\Users\HP\AppData\Local\Temp\ipykernel_14156\446966422.py:7: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-d
ocs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#return
ing-a-view-versus-a-copy)
  india_rows['month'] = india_rows['date'].dt.to_period('M')
```

In [8]:
```
grouped_data = india_rows.groupby("variant")
grouped_data
```

Out[8]:
```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x000001C77ACDF610>
```

In [9]:
```
grouped_data["variant"].nunique()
```

Out[9]:
```
variant
Alpha            1
B.1.1.277        1
B.1.1.302        1
B.1.1.519        1
B.1.160          1
B.1.177          1
B.1.221          1
B.1.258          1
B.1.367          1
B.1.620          1
Beta             1
Delta            1
Epsilon          1
Eta              1
Gamma            1
Iota             1
Kappa            1
Lambda           1
Mu               1
Omicron          1
S:677H.Robin1    1
S:677P.Pelican   1
non_who          1
others           1
Name: variant, dtype: int64
```

In [10]:
```
counts_category = grouped_data["variant"].count()

counts_category
```

Out[10]:
```
variant
Alpha             44
B.1.1.277         44
B.1.1.302         44
B.1.1.519         44
B.1.160           44
B.1.177           44
B.1.221           44
B.1.258           44
B.1.367           44
B.1.620           44
Beta              44
Delta             44
Epsilon           44
Eta               44
Gamma             44
Iota              44
Kappa             44
Lambda            44
Mu                44
Omicron           44
S:677H.Robin1     44
S:677P.Pelican    44
non_who           44
others            44
Name: variant, dtype: int64
```

In [11]:
```
# Sort the grouped_data based on num_sequences in descending order
sorted_data = grouped_data.apply(lambda x: x.nlargest(5, 'num_sequences'))

# Reset the index
sorted_data = sorted_data.reset_index(drop=True)

# Print the top 5 variants for verification
print(sorted_data)
```

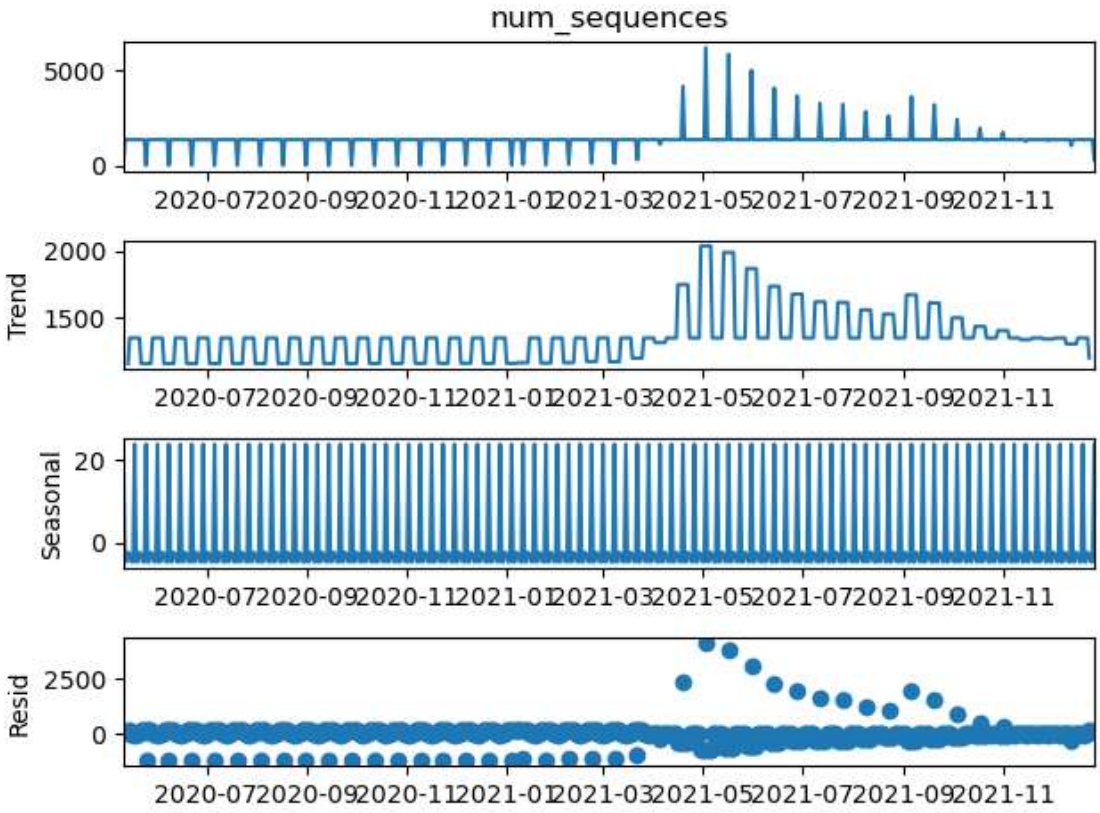
```
    location       date variant  num_sequences perc_sequences  \
0      India 2021-03-22   Alpha           1238          32.92
1      India 2021-04-05   Alpha           1127          23.81
2      India 2021-04-19   Alpha            908          12.17
3      India 2021-05-03   Alpha            529           6.54
4      India 2021-03-08   Alpha            436          17.59
..       ...        ...     ...            ...            ...
115    India 2020-12-21  others           1520          98.32
116    India 2021-03-08  others           1376          55.54
117    India 2021-02-22  others           1182          69.67
118    India 2021-01-04  others           1174          90.79
119    India 2021-02-08  others           1169          87.51

     num_sequences_total    month
0                   3761  2021-03
1                   4734  2021-04
2                   7463  2021-04
3                   8086  2021-05
4                   2478  2021-03
..                   ...      ...
115                 1546  2020-12
116                 2478  2021-03
117                 1697  2021-02
118                 1293  2021-01
119                 1336  2021-02

[120 rows x 7 columns]
```

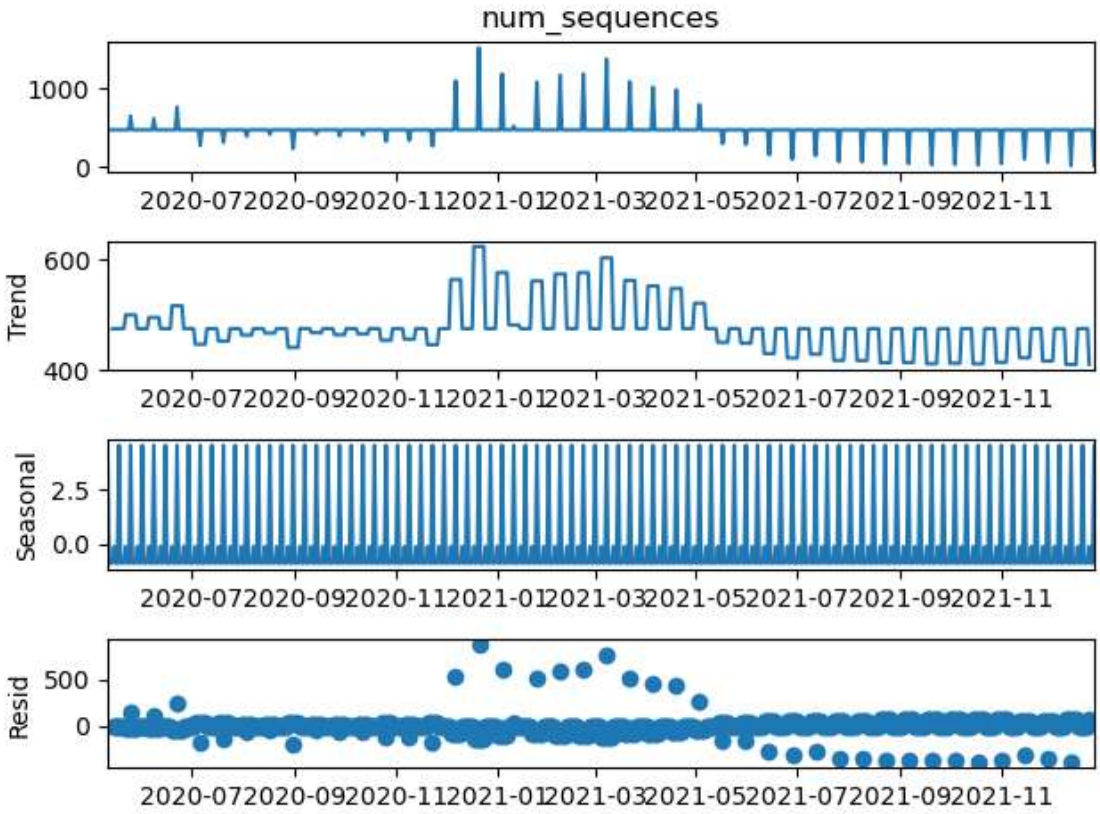
In [12]:
```
# Sort the groups based on the maximum value of 'num_sequences'
sorted_groups = grouped_data['num_sequences'].max().nlargest(6)

# Get the top 5 variants from the sorted groups
top_5_variants = sorted_groups.index.tolist()


# Print the top 5 variants
print("Top 6 Variants:")
print(top_5_variants)
```

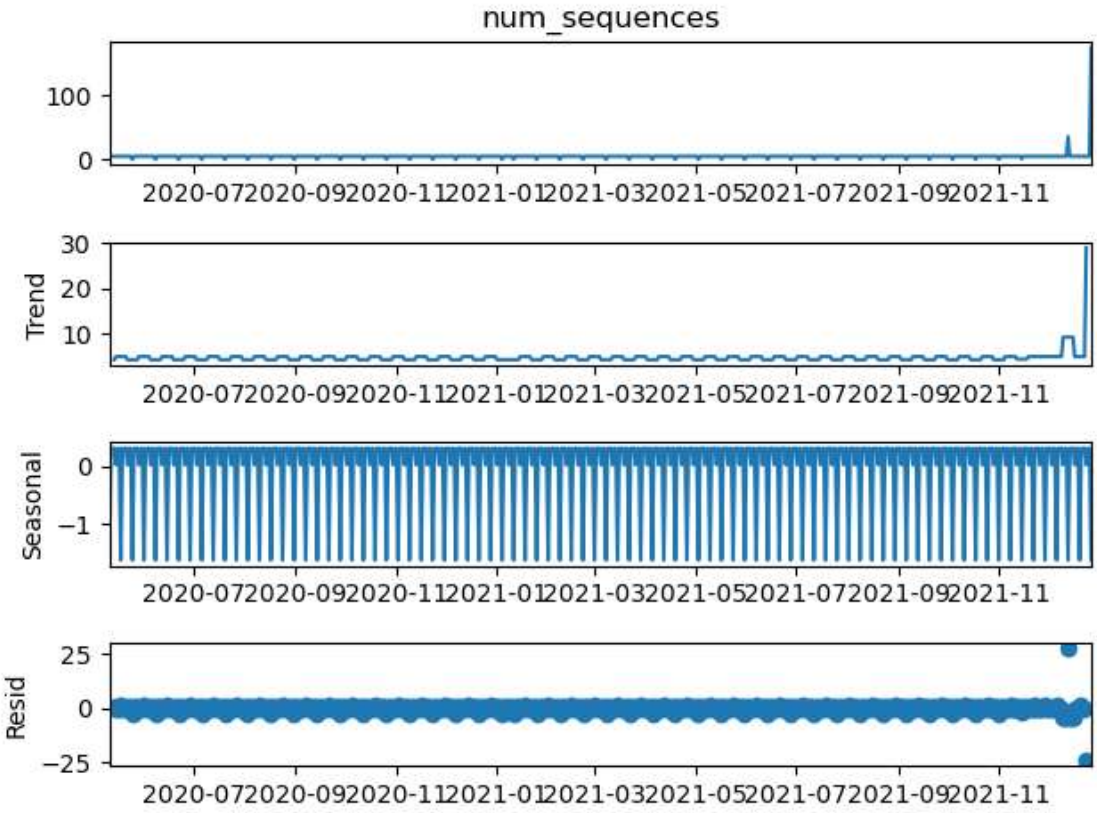
```
Top 6 Variants:
['Delta', 'non_who', 'others', 'Kappa', 'Alpha', 'Omicron']
```

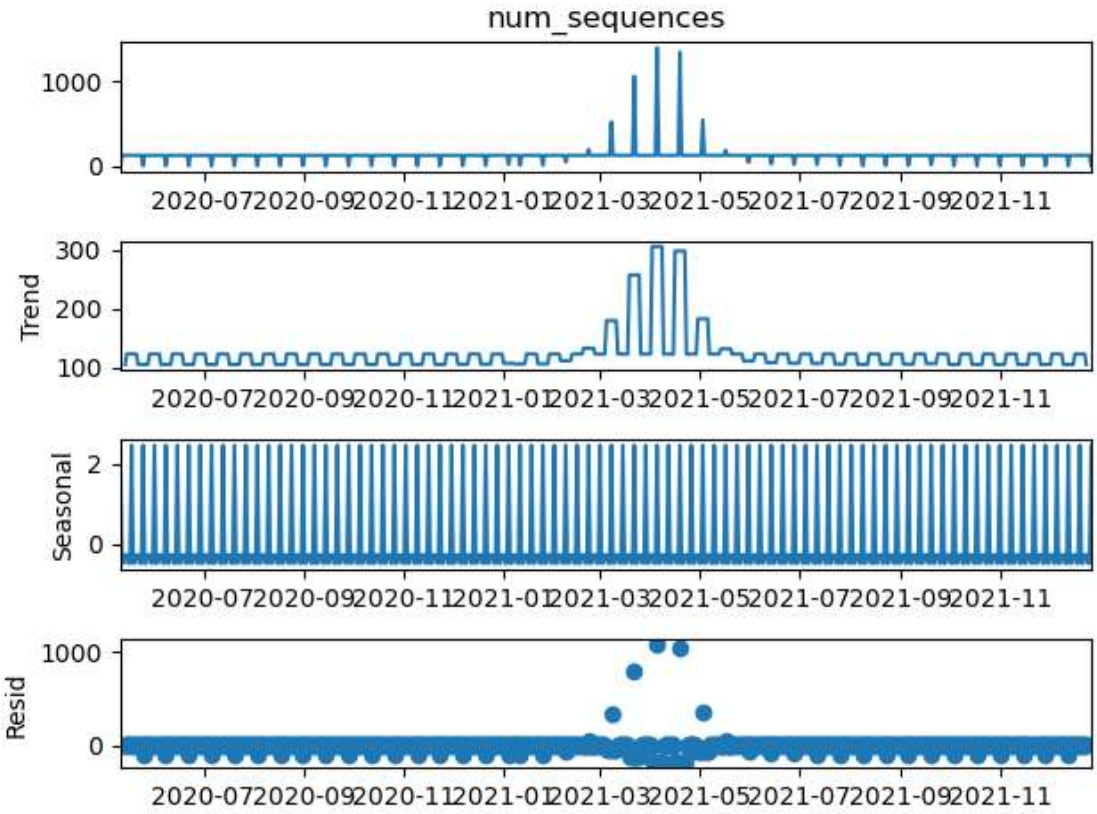
In [46]:

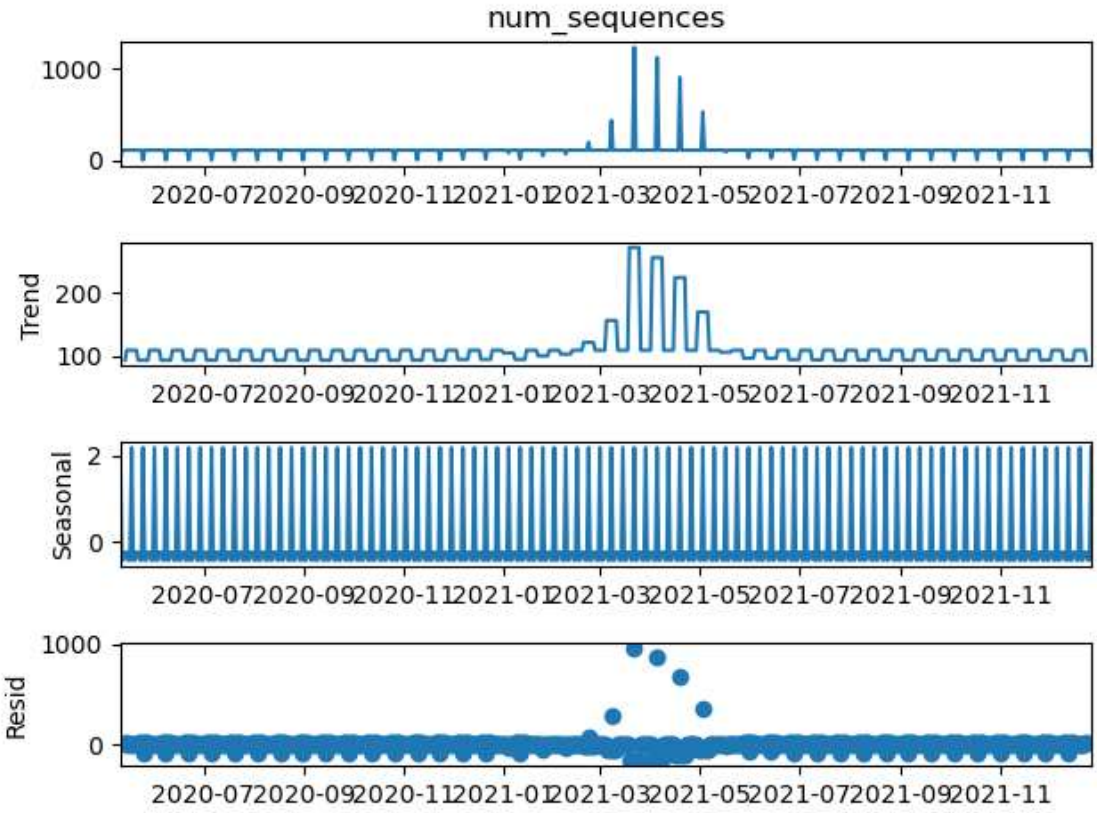
```
import statsmodels.api as sm

# Define the top 5 variants
top_5_variants = ['Delta', 'non_who', 'Omicron', 'Kappa', 'Alpha']

# Iterate over each variant
for variant in top_5_variants:
    # Filter the rows for the current variant
    variant_rows = india_rows[india_rows['variant'] == variant]

    # Extract the relevant columns for seasonal decomposition
    dates = pd.to_datetime(variant_rows['date'])
    values = variant_rows['num_sequences']

    # Set the dates as a DatetimeIndex
    values.index = pd.DatetimeIndex(dates)

    # Set the appropriate frequency (e.g., 'D' for daily)
    values = values.asfreq('D')

    # Fill missing values with the mean
    mean = values.mean()
    values = values.fillna(mean)

    # Perform seasonal decomposition
    decomposition = sm.tsa.seasonal_decompose(values, model='additive')

    # Extract the trend, seasonal, and residual components
    trend = decomposition.trend
    seasonal = decomposition.seasonal
    residual = decomposition.resid

    # Plot the original series, trend, seasonal, and residual components
    decomposition.plot()
```


num_sequences
5000
0
2020-072020-092020-112021-012021-032021-052021-072021-092021-11
Trend
2000
1500
2020-072020-092020-112021-012021-032021-052021-072021-092021-11
Seasonal
20
0
2020-072020-092020-112021-012021-032021-052021-072021-092021-11
Resid
2500
0
2020-072020-092020-112021-012021-032021-052021-072021-092021-11

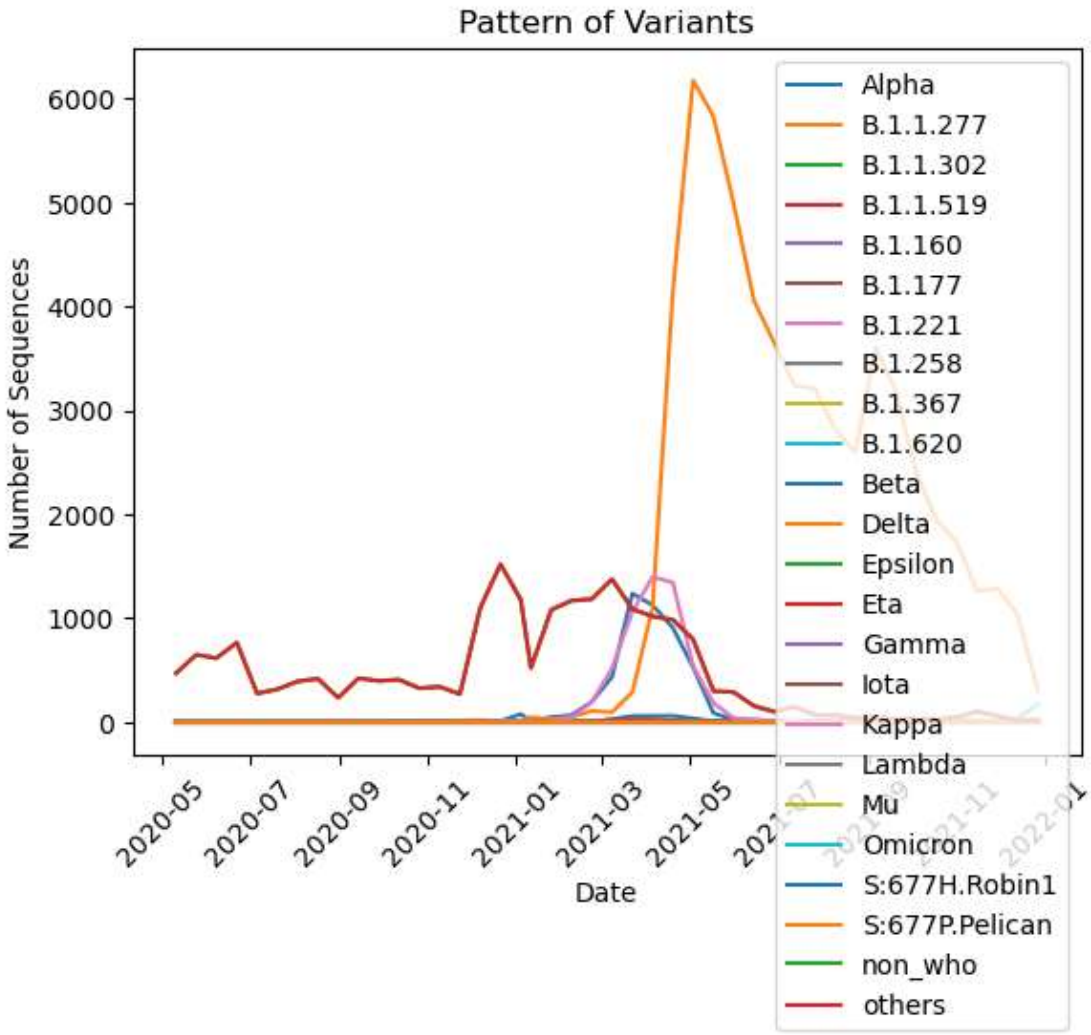

num_sequences
1000
0
2020-072020-092020-112021-012021-032021-052021-072021-092021-11
Trend
600
400
2020-072020-092020-112021-012021-032021-052021-072021-092021-11
Seasonal
2.5
0.0
2020-072020-092020-112021-012021-032021-052021-072021-092021-11
Resid
500
0
2020-072020-092020-112021-012021-032021-052021-072021-092021-11


num_sequences
100
0
2020-072020-092020-112021-012021-032021-052021-072021-092021-11
Trend
30
20
10
2020-072020-092020-112021-012021-032021-052021-072021-092021-11
Seasonal
0
−1
2020-072020-092020-112021-012021-032021-052021-072021-092021-11
Resid
25
0
−25
2020-072020-092020-112021-012021-032021-052021-072021-092021-11


num_sequences
1000
0
2020-072020-092020-112021-012021-032021-052021-072021-092021-11
Trend
300
200
100
2020-072020-092020-112021-012021-032021-052021-072021-092021-11
Seasonal
2
0
2020-072020-092020-112021-012021-032021-052021-072021-092021-11
Resid
1000
0
2020-072020-092020-112021-012021-032021-052021-072021-092021-11


num_sequences
1000
0
2020-07
2020-09
2020-11
2021-01
2021-03
2021-05
2021-07
2021-09
2021-11
Trend
200
100
Seasonal
2
0
Resid
1000
0

In [13]:
```
import matplotlib.pyplot as plt

# Iterate over each variant and plot the pattern
for variant, variant_data in grouped_data:
    plt.plot(variant_data['date'], variant_data['num_sequences'], label=va

# Customize the plot
plt.xlabel('Date')
plt.ylabel('Number of Sequences')
plt.title('Pattern of Variants')
plt.legend()
plt.xticks(rotation=45)

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

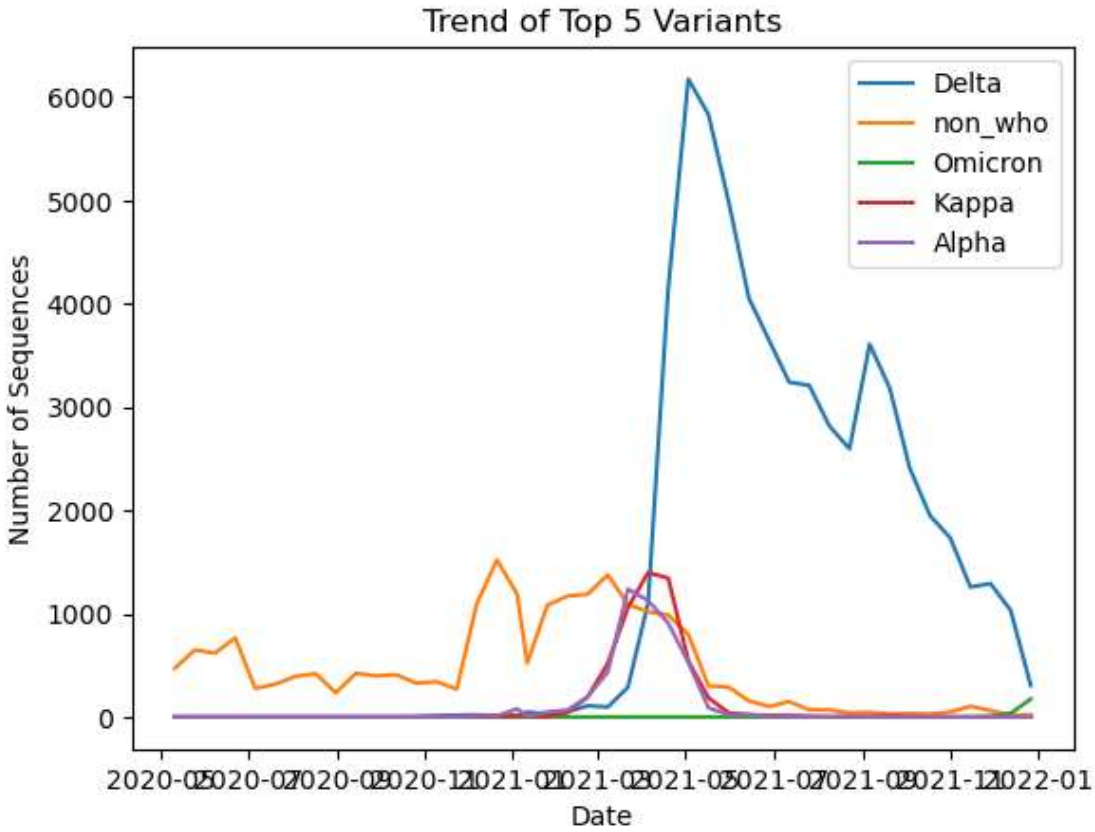


Time Series Decomposition:

In [14]:

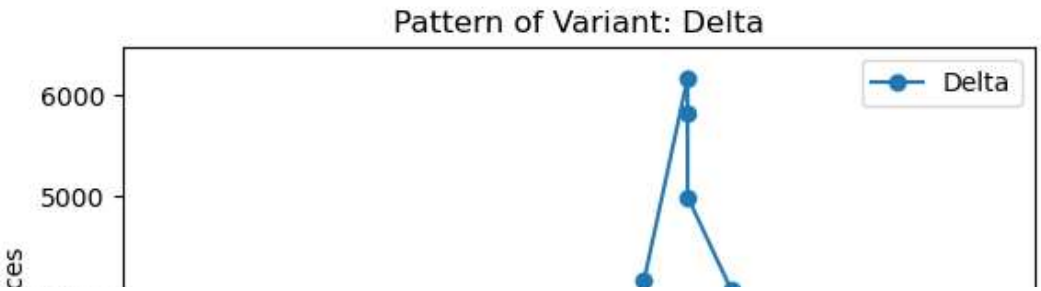
```
# Iterate over each variant
for variant in ['Delta', 'non_who', 'Omicron', 'Kappa', 'Alpha']:
    # Filter the rows for the current variant
    variant_rows = india_rows[india_rows['variant'] == variant]

    # Convert the 'date' column to datetime
    variant_rows['date'] = pd.to_datetime(variant_rows['date'])

    # Set the 'date' column as the index
    variant_rows.set_index('date', inplace=True)

    # Plot the time series for the variant
    plt.plot(variant_rows['num_sequences'], label=variant)

# Set the x-axis label
plt.xlabel('Date')

# Set the y-axis label
plt.ylabel('Number of Sequences')

# Set the title of the plot
plt.title('Trend of Top 5 Variants')

# Show the legend
plt.legend()

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

```
C:\Users\HP\AppData\Local\Temp\ipykernel_14156\275796068.py:7: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-d
ocs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#return
ing-a-view-versus-a-copy)
  variant_rows['date'] = pd.to_datetime(variant_rows['date'])
C:\Users\HP\AppData\Local\Temp\ipykernel_14156\275796068.py:7: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-d
ocs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#return
ing-a-view-versus-a-copy)
  variant_rows['date'] = pd.to_datetime(variant_rows['date'])
C:\Users\HP\AppData\Local\Temp\ipykernel_14156\275796068.py:7: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-d
ocs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#return
ing-a-view-versus-a-copy)
  variant_rows['date'] = pd.to_datetime(variant_rows['date'])
C:\Users\HP\AppData\Local\Temp\ipykernel_14156\275796068.py:7: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-d
ocs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#return
ing-a-view-versus-a-copy)
  variant_rows['date'] = pd.to_datetime(variant_rows['date'])
C:\Users\HP\AppData\Local\Temp\ipykernel_14156\275796068.py:7: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-d
ocs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#return
ing-a-view-versus-a-copy)
  variant_rows['date'] = pd.to_datetime(variant_rows['date'])
```


Trend of Top 5 Variants
Delta
non_who
Omicron
Kappa
Alpha
Number of Sequences
6000
5000
4000
3000
2000
1000
0
2020-01
2020-07
2020-09
2020-11
2021-01
2021-03
2021-05
2021-07
2021-09
2021-11
2022-01
Date

In [15]:
```
# List of top 5 variants
top_5_variants = ['Delta', 'non_who', 'Omicron', 'Kappa', 'Alpha']

# Iterate over the top 5 variants and plot the pattern
for variant in top_5_variants:
    # Filter the rows for the current variant
    variant_rows = india_rows[india_rows['variant'] == variant]

    # Create a new figure and axis for each variant
    fig, ax = plt.subplots()

    # Convert the 'month' column to string
    variant_rows['month'] = variant_rows['month'].astype(str)

    # Plot the variant's pattern as a line chart
    ax.plot(variant_rows['month'], variant_rows['num_sequences'], label=va

    # Set the title and labels for the plot
    ax.set_title(f'Pattern of Variant: {variant}')
    ax.set_xlabel('Month')
    ax.set_ylabel('Number of Sequences')

    # Show the legend
    ax.legend()

    # Rotate x-axis tick labels for better readability
    plt.xticks(rotation=45)

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

```
C:\Users\HP\AppData\Local\Temp\ipykernel_14156\3406535861.py:13: Sett
ingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.htm
l#returning-a-view-versus-a-copy)
  variant_rows['month'] = variant_rows['month'].astype(str)
```



Statistical Analysis

In [16]:

```
# Dictionary to store the statistical measures
statistics = {
    'Variant': [],
    'Mean': [],
    'Median': [],
    'Standard Deviation': [],
    'Autocorrelation': []
}

# Iterate over each variant
for variant in ['Delta', 'non_who', 'Omicron', 'Kappa', 'Alpha']:
    # Filter the rows for the current variant
    variant_rows = india_rows[india_rows['variant'] == variant]

    # Extract the 'num_sequences' column as a NumPy array
    num_sequences = variant_rows['num_sequences'].values

    # Calculate the statistical measures
    mean = np.mean(num_sequences)
    median = np.median(num_sequences)
    std = np.std(num_sequences)
    autocorr = np.correlate(num_sequences, num_sequences, mode='full')

    # Add the statistics to the dictionary
    statistics['Variant'].append(variant)
    statistics['Mean'].append(mean)
    statistics['Median'].append(median)
    statistics['Standard Deviation'].append(std)
    statistics['Autocorrelation'].append(autocorr)

# Create a DataFrame from the statistics dictionary
statistics_df = pd.DataFrame(statistics)

# Display the statistics
print(statistics_df)
```

```
   Variant         Mean  Median  Standard Deviation  \
0    Delta  1346.772727   102.0         1790.463359
1  non_who   475.113636   334.0          431.189856
2  Omicron     4.886364     0.0           26.318172
3    Kappa   123.522727     0.5          331.958204
4    Alpha   109.318182     1.5          287.293623

                                       Autocorrelation
0  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 608, 5712...
1  [9891, 22107, 53841, 115943, 146822, 166359, 1...
2  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
3  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
4  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
```

The table shows the mean, median, standard deviation, and autocorrelation for each variant. The mean represents the average number of sequences, the median represents the middle value in the data, and the standard deviation measures the variability of the data. The autocorrelation values indicate the correlation between the sequences at different time lags.

In [17]:
```
india_rows.head(1)
```

Out[17]:

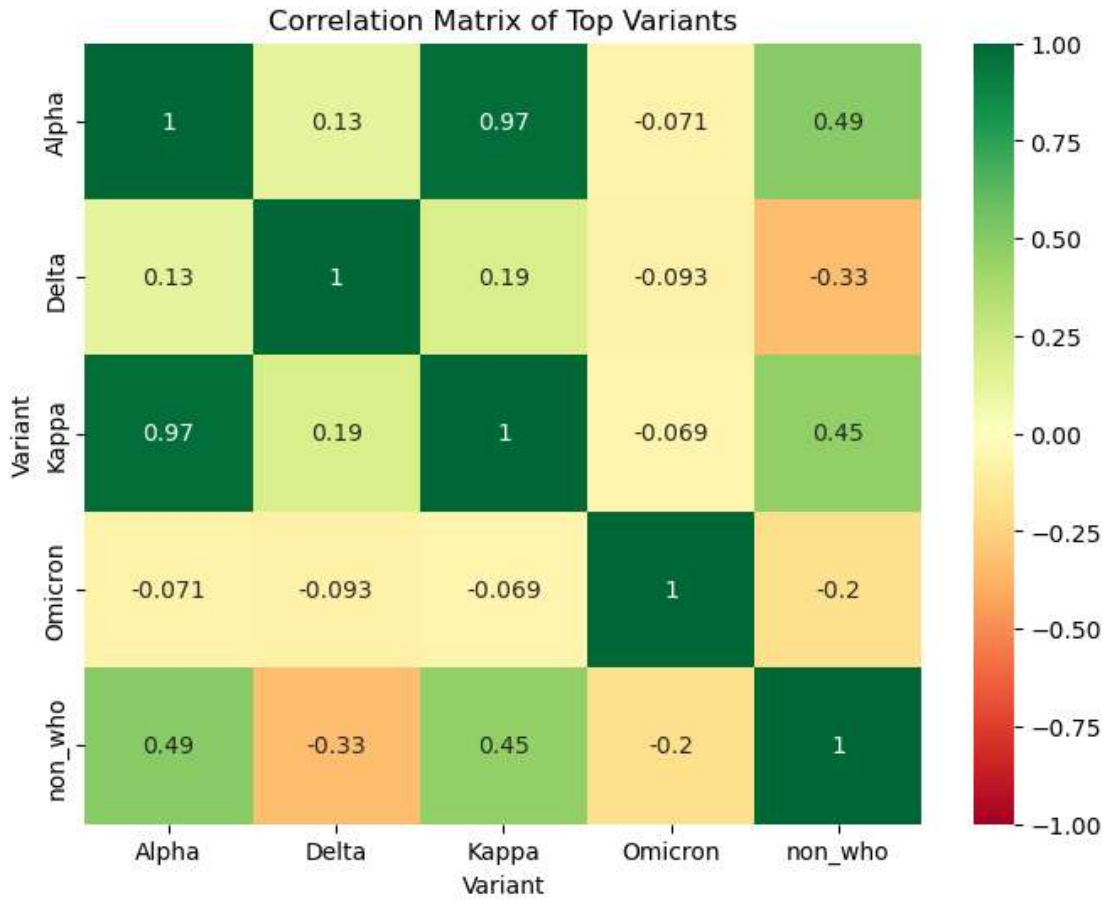
	location	date	variant	num_sequences	perc_sequences	num_sequences_total	m
36360	India	2020-05-11	Alpha	0	0.0	471	2

In [18]:
```
# Select the rows corresponding to the top variants
top_variants = ['Delta', 'non_who', 'Omicron', 'Kappa', 'Alpha']
top_variants_data = india_rows[india_rows['variant'].isin(top_variants)]

# Pivot the data to have variants as columns and 'num_sequences' as values
pivot_table = top_variants_data.pivot(index='date', columns='variant', va

# Calculate the correlation matrix
correlation_matrix = pivot_table.corr()

# Print the correlation matrix
print(correlation_matrix)
```

```
variant     Alpha     Delta     Kappa   Omicron   non_who
variant
Alpha    1.000000  0.132655  0.973610 -0.070617  0.486153
Delta    0.132655  1.000000  0.193439 -0.093002 -0.334521
Kappa    0.973610  0.193439  1.000000 -0.069087  0.454081
Omicron -0.070617 -0.093002 -0.069087  1.000000 -0.195171
non_who  0.486153 -0.334521  0.454081 -0.195171  1.000000
```

In [19]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Select the rows corresponding to the top variants
top_variants = ['Delta', 'non_who', 'Omicron', 'Kappa', 'Alpha']
top_variants_data = india_rows[india_rows['variant'].isin(top_variants)]

# Pivot the data to have variants as columns and 'num_sequences' as values
pivot_table = top_variants_data.pivot(index='date', columns='variant', val

# Calculate the correlation matrix
correlation_matrix = pivot_table.corr()

# Create a heatmap plot
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='RdYlGn', vmin=-1, vmax=1
plt.title('Correlation Matrix of Top Variants')
plt.xlabel('Variant')
plt.ylabel('Variant')
```

Out[19]: Text(70.58159722222221, 0.5, 'Variant')



1 Alpha and Kappa have a high positive correlation of 0.97, suggesting a strong relationship between them. Similarly, Delta and non_who have a negative correlation of -0.33, indicating a negative association between them. 2 these values represent the pairwise correlation coefficients between the top 5 variants: Alpha, Delta, Kappa, Omicron, and non_who. Each

value ranges from -1 to 1, indicating the strength and direction of the correlation between the respective variants. For example, a correlation coefficient of 1 indicates a perfect positive

Anova test

In [21]:
```
import scipy.stats as stats

top_variants = ['Delta', 'non_who', 'Omicron', 'Kappa', 'Alpha']

# Filter the rows for the top variants
top_variants_rows = india_rows[india_rows['variant'].isin(top_variants)]

# Group the data by variant and extract the relevant columns
grouped_data = top_variants_rows.groupby('variant')['num_sequences']

# Perform the ANOVA test
f_statistic, p_value = stats.f_oneway(*[group.values for _, group in group

# Print the results
print('ANOVA Test Results:')
print('F-Statistic:', f_statistic)
print('P-value:', p_value)
```

```
ANOVA Test Results:
F-Statistic: 18.26634628922606
P-value: 6.210342832693352e-13
```

The ANOVA test results indicate that there are statistically significant differences among the means of the variants. Here's a description of the results:

F-Statistic: The F-Statistic value is 18.266. This value represents the ratio of the between-group variance to the within-group variance. A higher F-Statistic suggests that the differences among the means of the groups are more significant.

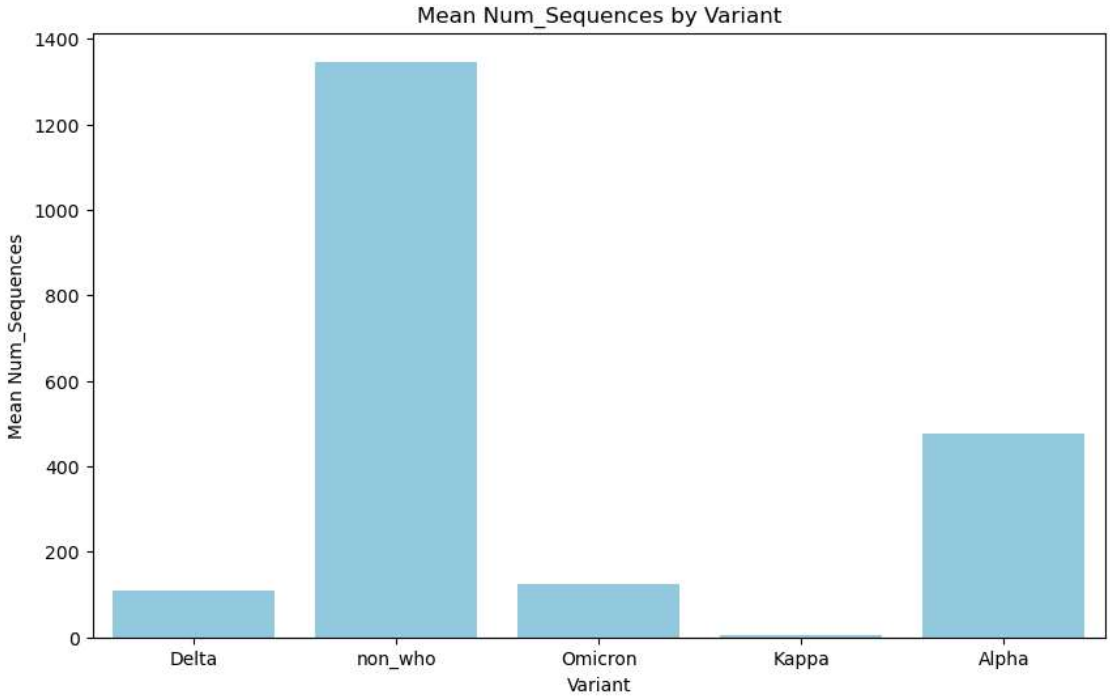
P-value: The P-value is approximately 6.21e-13, which is extremely small. The P-value represents the probability of observing the data given that the null hypothesis is true (i.e., there are no differences among the means). A small P-value indicates strong evidence against the null hypothesis and suggests that the differences among the means of the groups are statistically significant.

In summary, the ANOVA test results suggest that there are significant differences in the number of sequences among the top variants (Delta, non_who, Omicron, Kappa, Alpha). These differences are not likely due to random chance and indicate distinct patterns or characteristics associated with each variant.

ANOVA is a widely used statistical test for comparing means across multiple groups and determining if there are significant differences among them. It allows researchers to make inferences about the population means based on the observed data and helps in understanding the relationships between categorical variables and continuous outcomes.

In [22]:

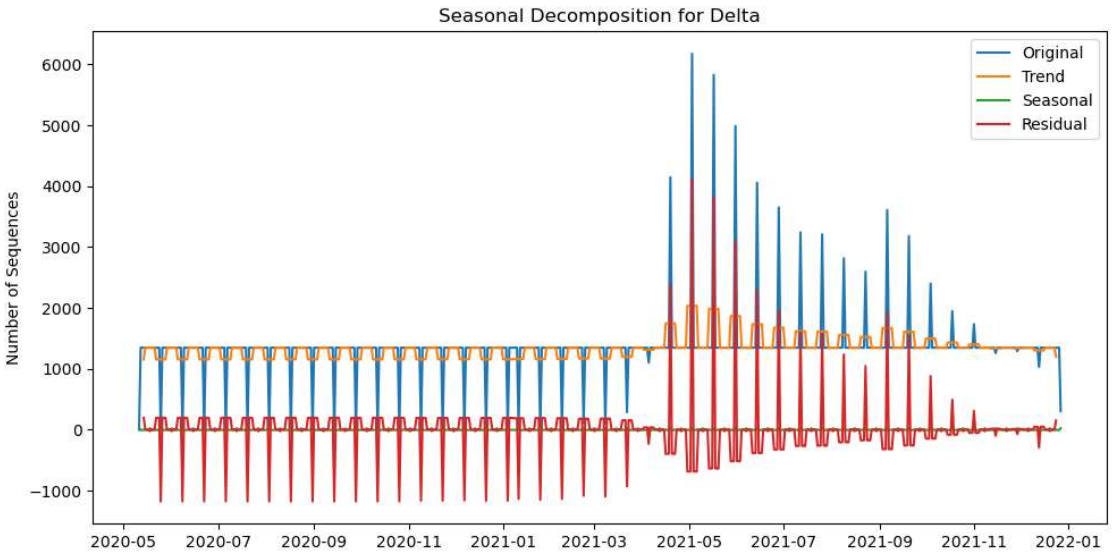
```
import seaborn as sns

# Create a DataFrame to hold the ANOVA results
anova_results = pd.DataFrame({'Variant': top_variants, 'Num_Sequences': [g

# Plot the bar plot
plt.figure(figsize=(10, 6))
sns.barplot(data=anova_results, x='Variant', y='Num_Sequences', color='sky
plt.xlabel('Variant')
plt.ylabel('Mean Num_Sequences')
plt.title('Mean Num_Sequences by Variant')
plt.show()
```

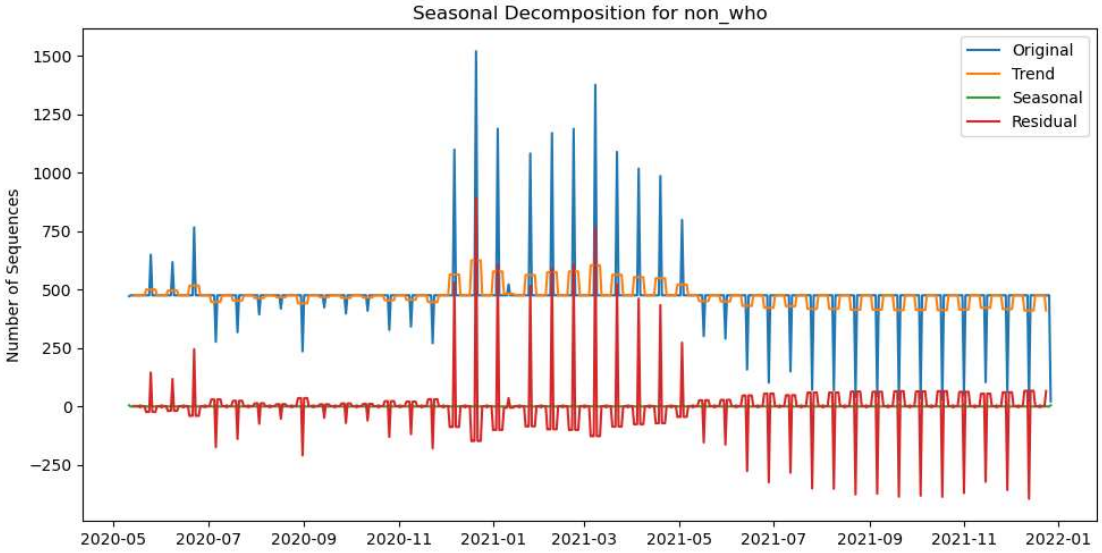


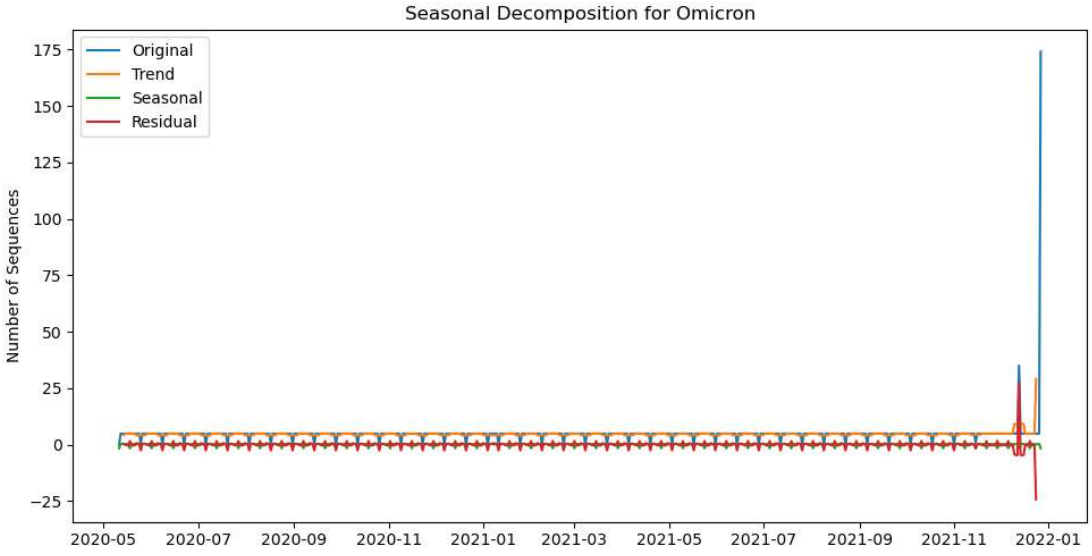
In [47]:

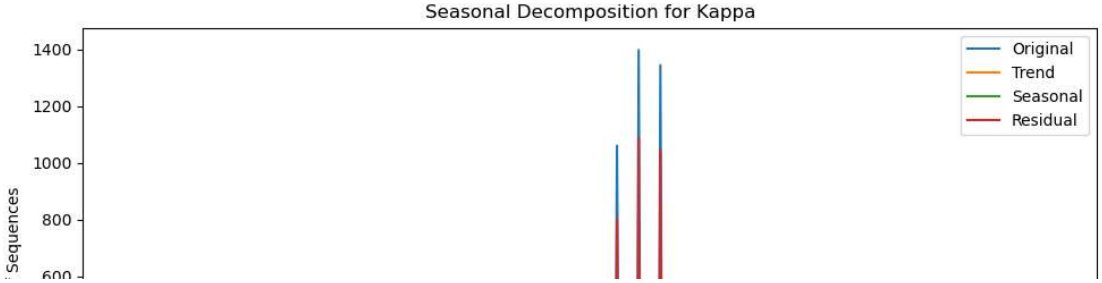
```
# Define the top 5 variants
top_5_variants = ['Delta', 'non_who', 'Omicron', 'Kappa', 'Alpha']

# Set the number of subplots based on the number of variants
num_subplots = len(top_5_variants)

# Set the figure size and create subplots
fig, axes = plt.subplots(num_subplots, 1, figsize=(10, num_subplots*5))

# Iterate over each variant
for i, variant in enumerate(top_5_variants):
    # Filter the rows for the current variant
    variant_rows = india_rows[india_rows['variant'] == variant]

    # Extract the relevant columns for seasonal decomposition
    dates = pd.to_datetime(variant_rows['date'])
    values = variant_rows['num_sequences']

    # Set the dates as a DatetimeIndex
    values.index = pd.DatetimeIndex(dates)

    # Set the appropriate frequency (e.g., 'D' for daily)
    values = values.asfreq('D')

    # Fill missing values with the mean
    mean = values.mean()
    values = values.fillna(mean)

    # Perform seasonal decomposition
    decomposition = sm.tsa.seasonal_decompose(values, model='additive')

    # Extract the trend, seasonal, and residual components
    trend = decomposition.trend
    seasonal = decomposition.seasonal
    residual = decomposition.resid

    # Plot the original series, trend, seasonal, and residual components
    axes[i].plot(values, label='Original')
    axes[i].plot(trend, label='Trend')
    axes[i].plot(seasonal, label='Seasonal')
    axes[i].plot(residual, label='Residual')

    # Set the title of the subplot with the variant name
    axes[i].set_title(f'Seasonal Decomposition for {variant}')

    # Set the y-axis label
    axes[i].set_ylabel('Number of Sequences')

    # Show the legend
    axes[i].legend()

# Set the x-axis label for the last subplot
axes[-1].set_xlabel('Date')

# Adjust the spacing between subplots
```

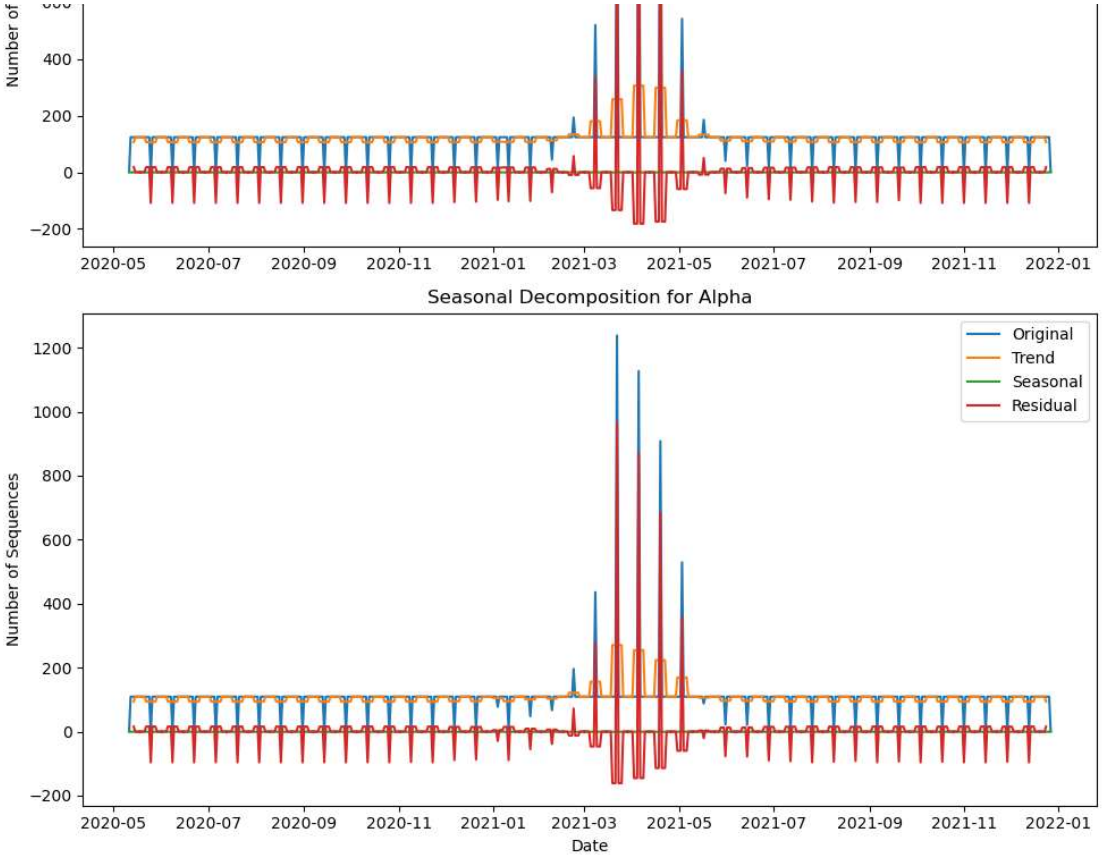
```
plt.tight_layout()

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


Seasonal Decomposition for Delta
Original
Trend
Seasonal
Residual
Number of Sequences
6000
5000
4000
3000
2000
1000
0
−1000
2020-05
2020-07
2020-09
2020-11
2021-01
2021-03
2021-05
2021-07
2021-09
2021-11
2022-01


Seasonal Decomposition for non_who
Original
Trend
Seasonal
Residual
Number of Sequences
1500
1250
1000
750
500
250
0
−250
2020-05
2020-07
2020-09
2020-11
2021-01
2021-03
2021-05
2021-07
2021-09
2021-11
2022-01


Seasonal Decomposition for Omicron
Original
Trend
Seasonal
Residual
Number of Sequences
175
150
125
100
75
50
25
0
−25
2020-05
2020-07
2020-09
2020-11
2021-01
2021-03
2021-05
2021-07
2021-09
2021-11
2022-01


Seasonal Decomposition for Kappa
Original
Trend
Seasonal
Residual
1400
1200
1000
800
600



Exponential Smoothing

In [36]:
```
from statsmodels.tsa.api import SimpleExpSmoothing

# Define the top 5 variants
top_5_variants = ['Delta', 'non_who', 'Omicron', 'Kappa', 'Alpha']

# Iterate over each variant
for variant in top_5_variants:
    # Filter the rows for the current variant
    variant_rows = india_rows[india_rows['variant'] == variant]

    # Extract the relevant columns for SES
    dates = variant_rows['date']
    values = variant_rows['num_sequences']

    # Perform Simple Exponential Smoothing
    model = SimpleExpSmoothing(values)
    fit_model = model.fit()

    # Predict the smoothed values
    smoothed_values = fit_model.fittedvalues

    # Print the smoothed values for the current variant
    print(f"Smoothed values for {variant}:")
    print(smoothed_values)
    print()
```

```
Smoothed values for Delta:
36371       0.000000
36395       0.000000
36419       0.000000
36443       0.000000
36467       0.000000
36491       0.000000
36515       0.000000
36539       0.000000
36563       0.000000
36587       0.000000
36611       0.000000
36635       0.000000
36659       0.000000
36683       1.990000
36707      11.949950
36731      15.979750
36755      19.979899
36779      11.044899
```

In [33]:

```
# Define the top 5 variants
top_5_variants = ['Delta', 'non_who', 'Omicron', 'Kappa', 'Alpha']

# Iterate over each variant
for variant in top_5_variants:
    # Filter the rows for the current variant
    variant_rows = india_rows[india_rows['variant'] == variant]

    # Extract the relevant columns for SES
    dates = variant_rows['date']
    values = variant_rows['num_sequences']

    # Perform Simple Exponential Smoothing
    model = SimpleExpSmoothing(values)
    fit_model = model.fit()

    # Predict the smoothed values
    smoothed_values = fit_model.fittedvalues

    # Plot the original values and smoothed values
    plt.plot(dates, values, label='Original')
    plt.plot(dates, smoothed_values, label='Smoothed')

    # Set the x-axis label
    plt.xlabel('Date')

    # Set the y-axis label
    plt.ylabel('Number of Sequences')

    # Set the title of the plot
    plt.title(f'Simple Exponential Smoothing for {variant}')

    # Show the legend
    plt.legend()

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

```
C:\ProgramData\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_m
odel.py:471: ValueWarning: An unsupported index was provided and will
be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
```

In [48]:
```
variant_distribution = india_rows[['date', 'variant']]
```

In [49]:
```
variant_counts = variant_distribution.groupby(['date', 'variant']).size().
```

In [50]:
```
pivoted_counts = variant_counts.pivot(index='date', columns='variant', val
```

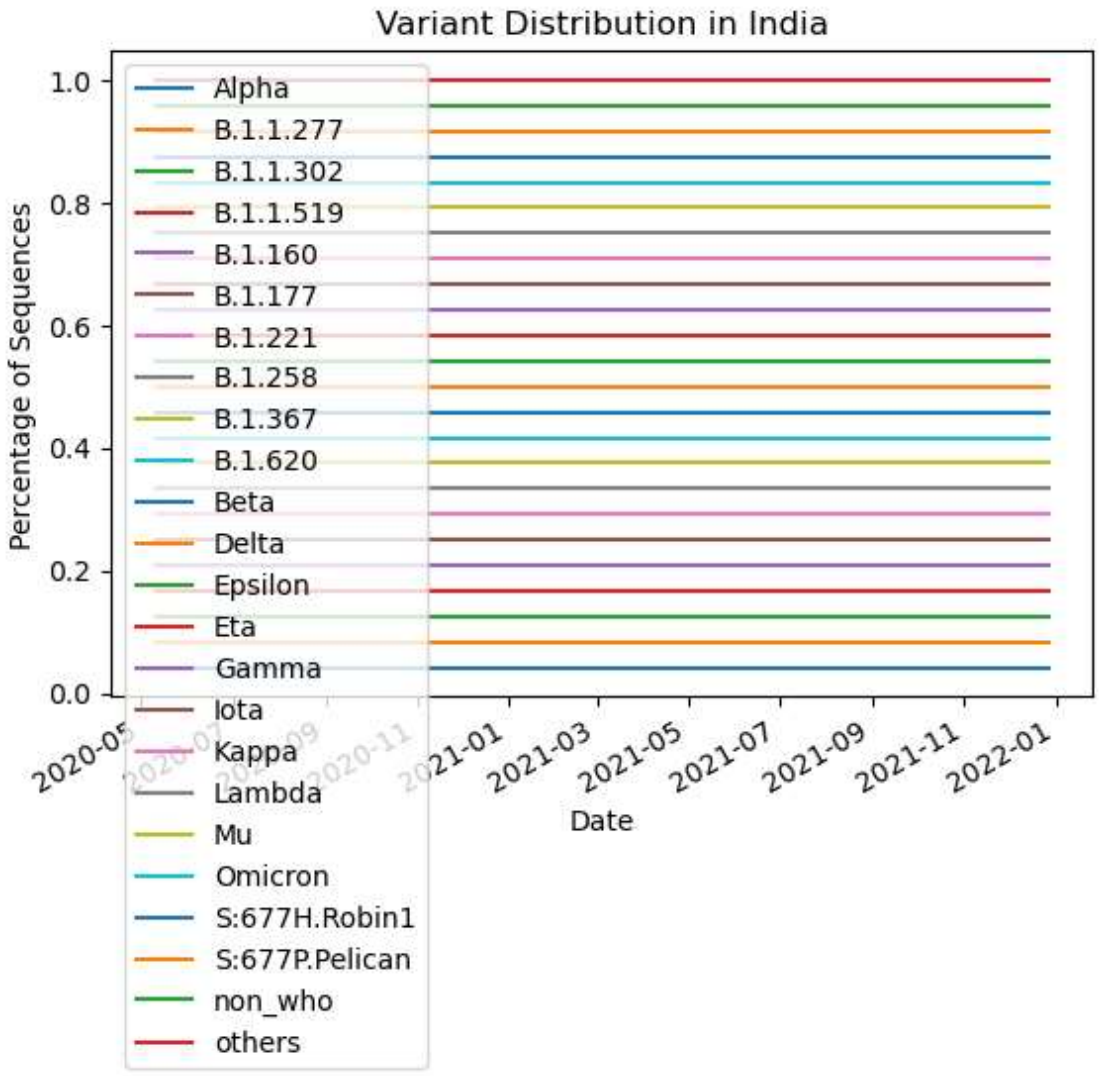
In [51]:
```
pivoted_counts['total_sequences'] = pivoted_counts.sum(axis=1)
```

In [54]:
```
variant_percentage = pivoted_counts.div(pivoted_counts['total_sequences'],
variant_percentage
```

Out[54]:

variant	Alpha	B.1.1.277	B.1.1.302	B.1.1.519	B.1.160	B.1.177	B.1.221	B.1.25
date								
2020-05-11	0.041667	0.041667	0.041667	0.041667	0.041667	0.041667	0.041667	0.04166
2020-05-25	0.041667	0.041667	0.041667	0.041667	0.041667	0.041667	0.041667	0.04166
2020-06-08	0.041667	0.041667	0.041667	0.041667	0.041667	0.041667	0.041667	0.04166
2020-06-22	0.041667	0.041667	0.041667	0.041667	0.041667	0.041667	0.041667	0.04166
2020-07-06	0.041667	0.041667	0.041667	0.041667	0.041667	0.041667	0.041667	0.04166
2020-07-20	0.041667	0.041667	0.041667	0.041667	0.041667	0.041667	0.041667	0.04166

In [53]:
```
variant_percentage.plot(kind='line', stacked=True)
plt.xlabel('Date')
plt.ylabel('Percentage of Sequences')
plt.title('Variant Distribution in India')
plt.legend(loc='upper left')
plt.show()
```

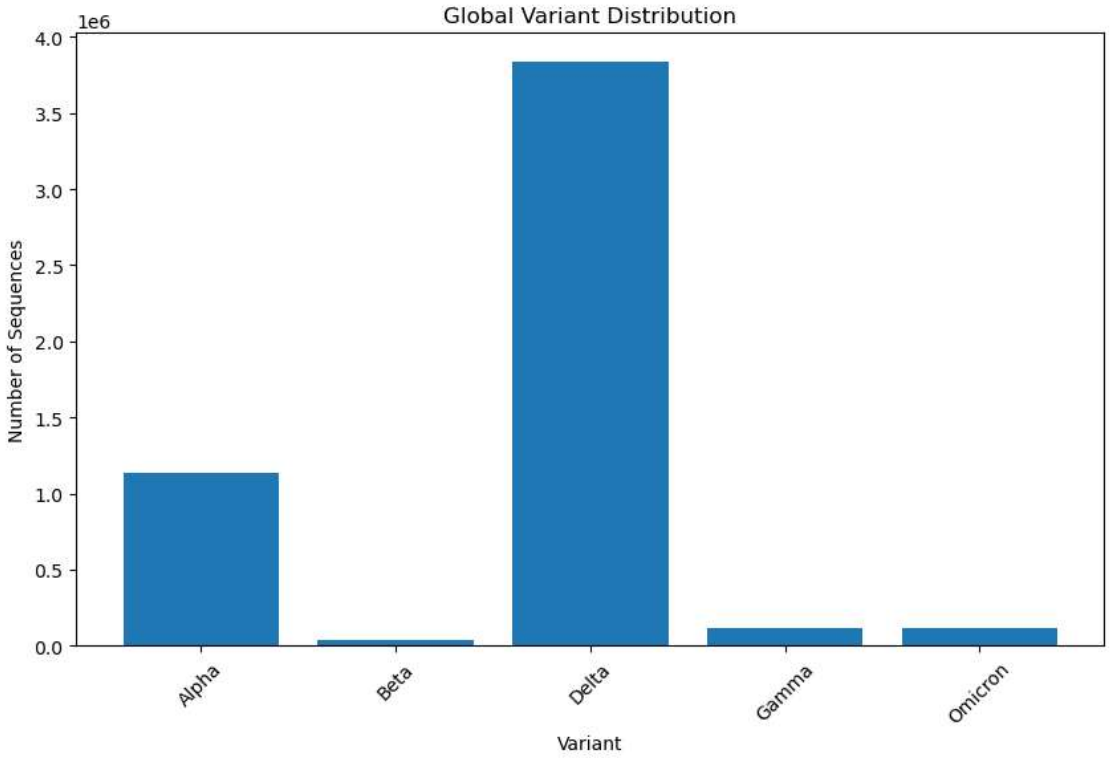


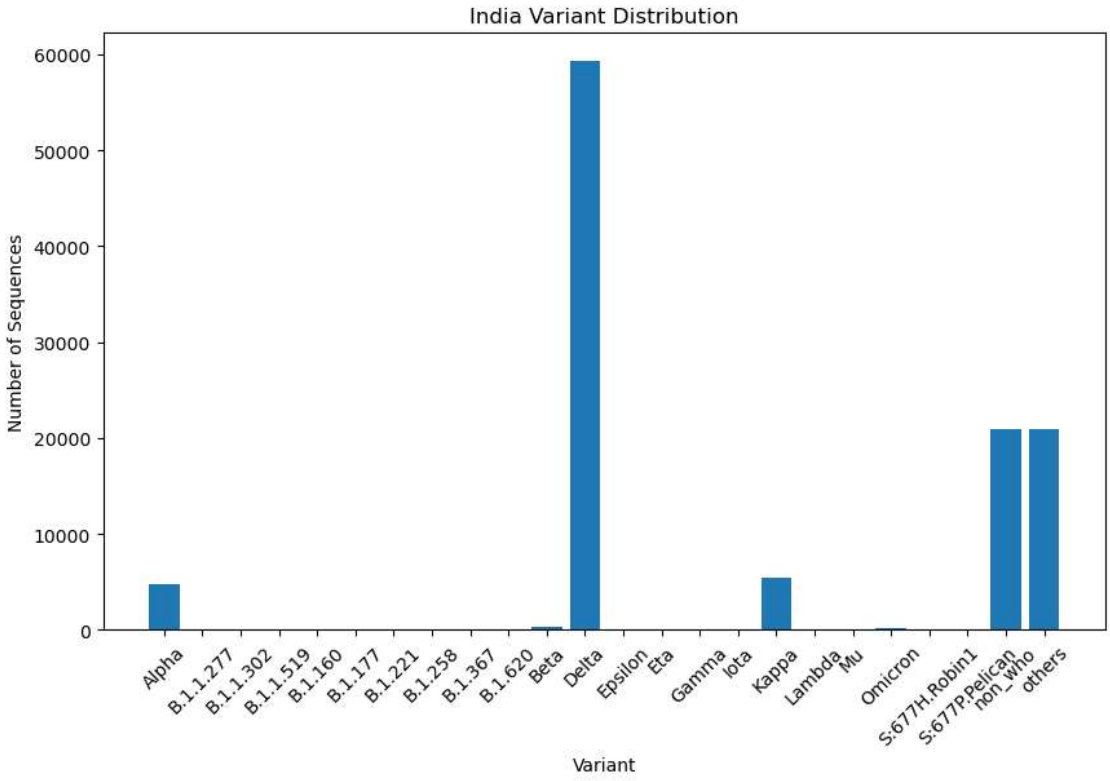
In [57]:
```
print(global_data['variant'].unique())
```

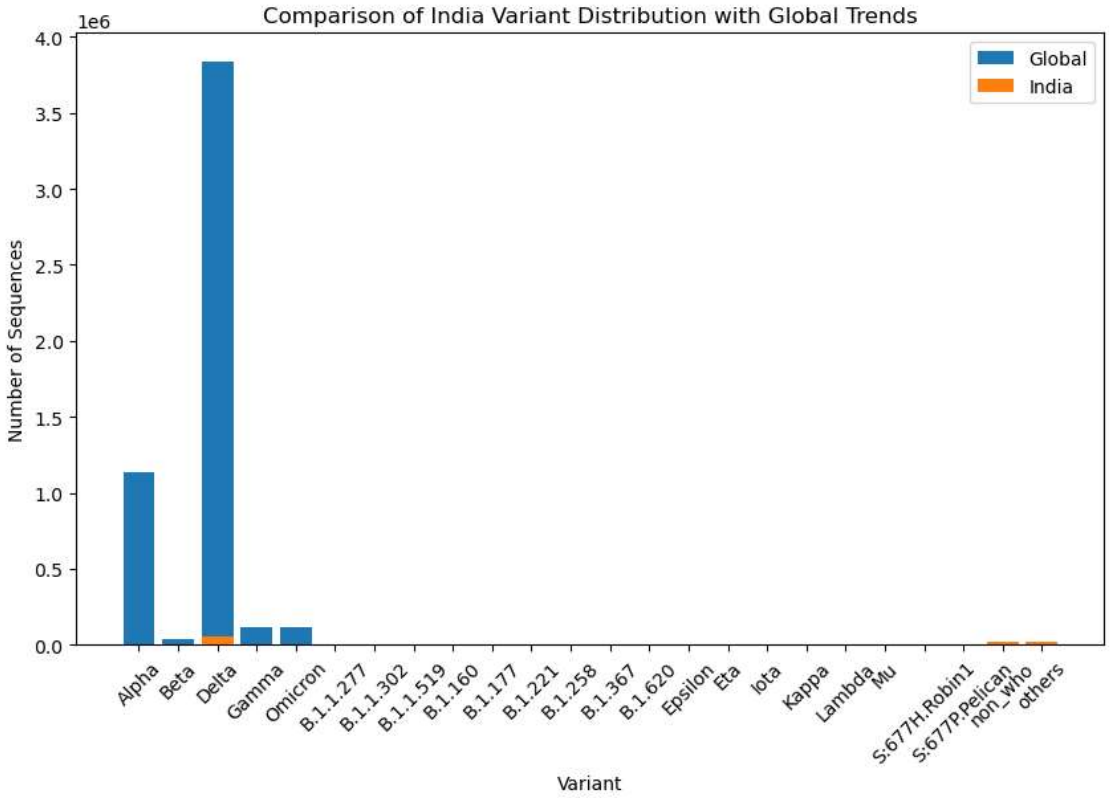
```
['Alpha' 'B.1.1.277' 'B.1.1.302' 'B.1.1.519' 'B.1.160' 'B.1.177' 'B.1.22
1'
 'B.1.258' 'B.1.367' 'B.1.620' 'Beta' 'Delta' 'Epsilon' 'Eta' 'Gamma'
 'Iota' 'Kappa' 'Lambda' 'Mu' 'Omicron' 'S:677H.Robin1' 'S:677P.Pelican'
 'others' 'non_who']
```

In [58]:

```
import matplotlib.pyplot as plt

# Filter the global data for the variants of interest
global_variants = ['Alpha', 'Beta', 'Gamma', 'Delta', 'Omicron']
global_data_filtered = global_data[global_data['variant'].isin(global_vari

# Group the global data by variant and calculate the total number of seque
global_grouped = global_data_filtered.groupby('variant')['num_sequences'].

# Create a bar plot for global variant distribution
plt.figure(figsize=(10, 6))
plt.bar(global_grouped.index, global_grouped.values)
plt.xlabel('Variant')
plt.ylabel('Number of Sequences')
plt.title('Global Variant Distribution')
plt.xticks(rotation=45)
plt.show()

# Group the India data by variant and calculate the total number of sequer
india_grouped = india_rows.groupby('variant')['num_sequences'].sum()

# Create a bar plot for India's variant distribution
plt.figure(figsize=(10, 6))
plt.bar(india_grouped.index, india_grouped.values)
plt.xlabel('Variant')
plt.ylabel('Number of Sequences')
plt.title('India Variant Distribution')
plt.xticks(rotation=45)
plt.show()

# Compare India's variant distribution with global trends
plt.figure(figsize=(10, 6))
plt.bar(global_grouped.index, global_grouped.values, label='Global')
plt.bar(india_grouped.index, india_grouped.values, label='India')
plt.xlabel('Variant')
plt.ylabel('Number of Sequences')
plt.title('Comparison of India Variant Distribution with Global Trends')
plt.xticks(rotation=45)
plt.legend()
plt.show()
```

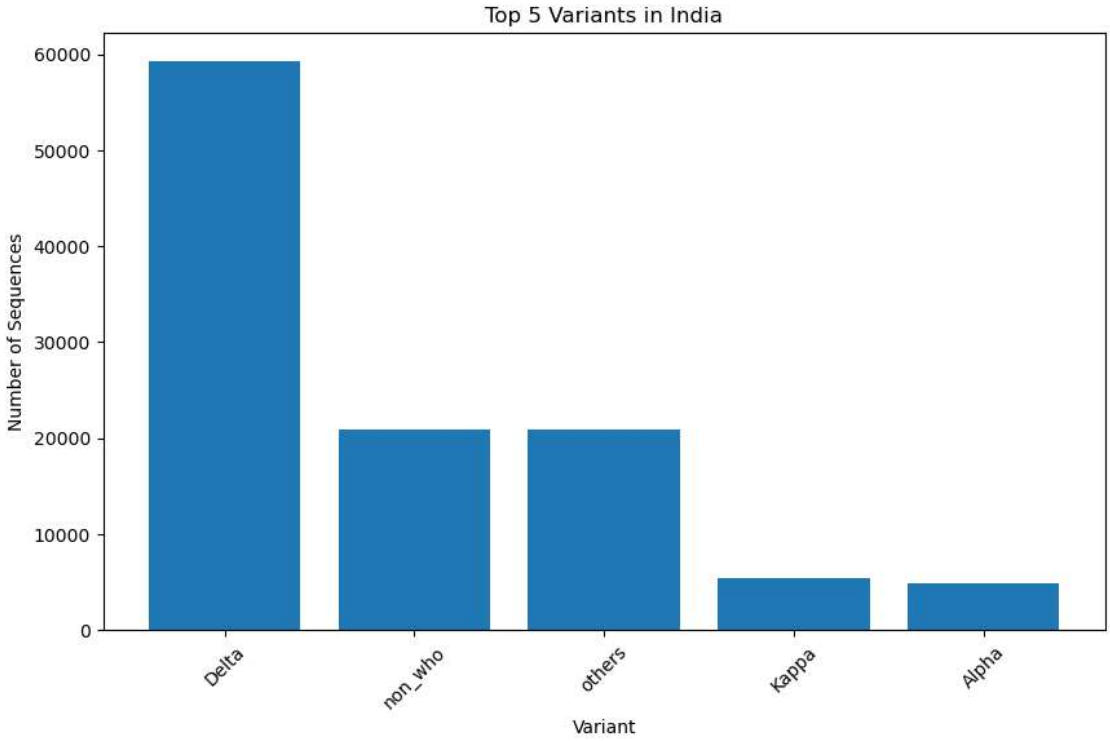

Global Variant Distribution
1e6
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Number of Sequences
Alpha
Beta
Delta
Gamma
Omicron
Variant

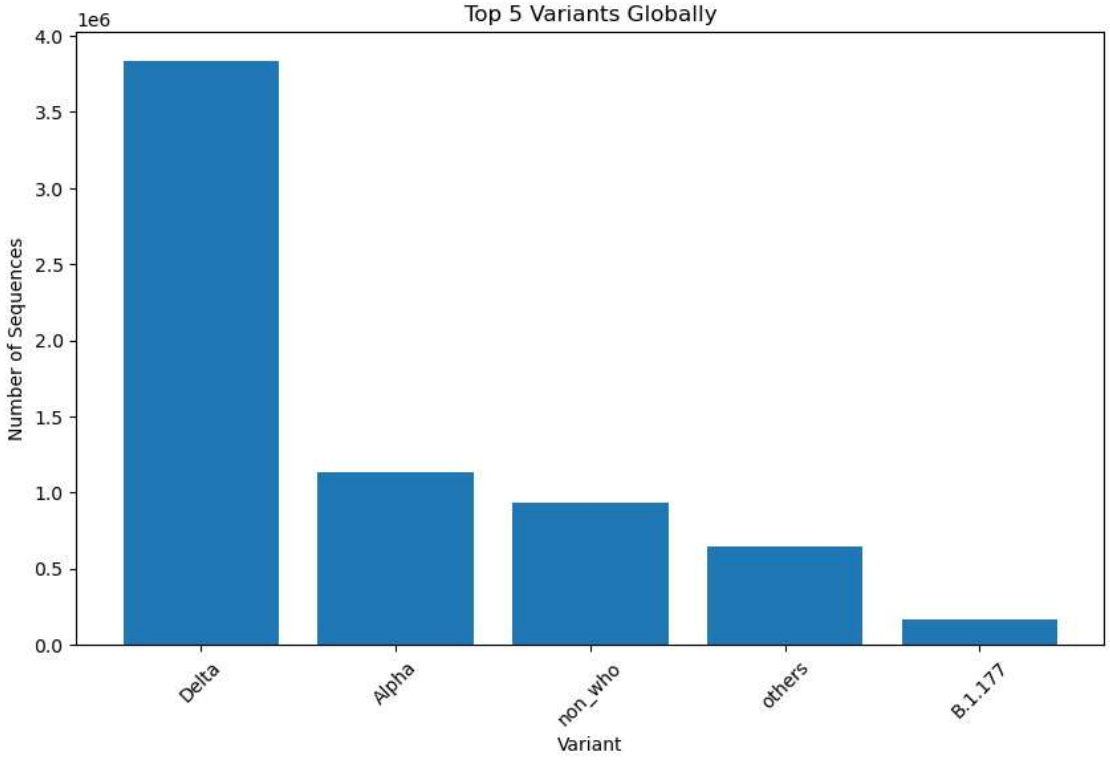

India Variant Distribution
60000
50000
40000
30000
20000
10000
0
Number of Sequences
Alpha
B.1.1.277
B.1.1.302
B.1.1.519
B.1.160
B.1.177
B.1.221
B.1.258
B.1.367
B.1.620
Beta
Delta
Epsilon
Eta
Gamma
Iota
Kappa
Lambda
Mu
Omicron
S:677H.Robin1
S:677P.Pelican
non_who
others
Variant

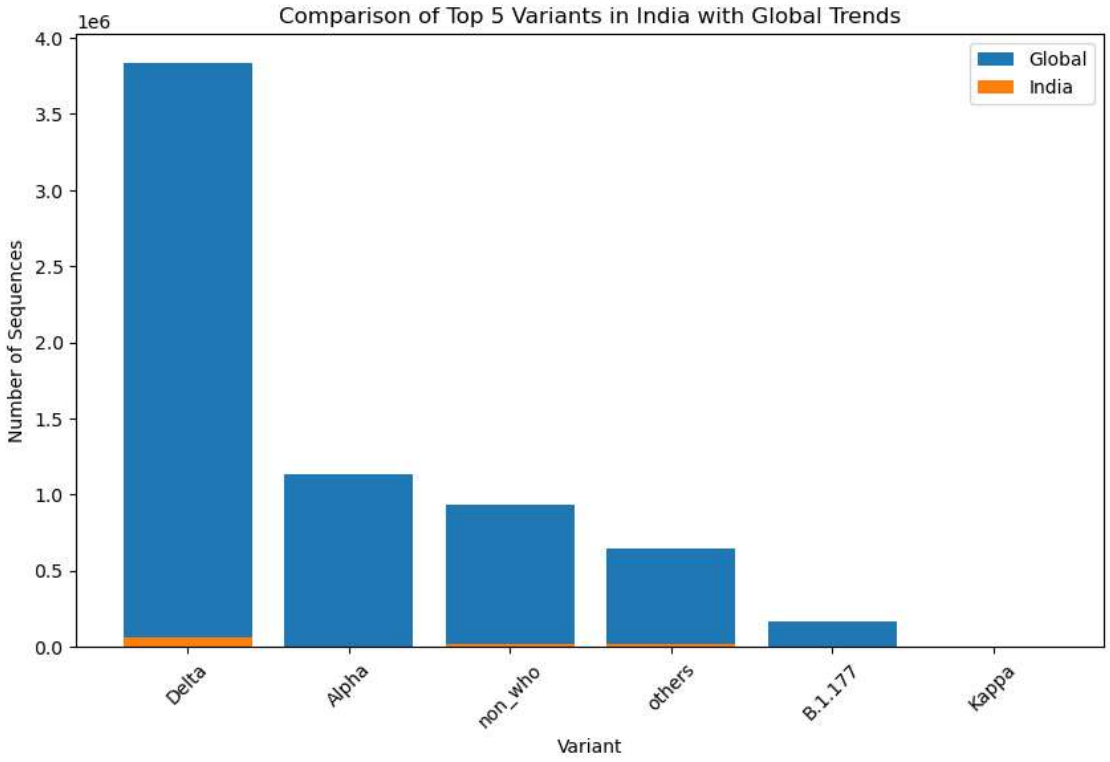

Comparison of India Variant Distribution with Global Trends
1e6
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Number of Sequences
Global
India
Alpha
Beta
Delta
Gamma
Omicron
B.1.1.277
B.1.1.302
B.1.1.519
B.1.160
B.1.177
B.1.221
B.1.258
B.1.367
B.1.620
Epsilon
Eta
Iota
Kappa
Lambda
Mu
S:677H.Robin1
S:677P.Pelican
non_who
others
Variant

In [59]:

```
import matplotlib.pyplot as plt

# Top 5 variants in India
top_5_india = india_rows.groupby('variant')['num_sequences'].sum().nlarges

# Top 5 variants globally
top_5_global = global_data.groupby('variant')['num_sequences'].sum().nlarg

# Create a bar plot for India's top 5 variants
plt.figure(figsize=(10, 6))
plt.bar(top_5_india.index, top_5_india.values)
plt.xlabel('Variant')
plt.ylabel('Number of Sequences')
plt.title('Top 5 Variants in India')
plt.xticks(rotation=45)
plt.show()

# Create a bar plot for the top 5 global variants
plt.figure(figsize=(10, 6))
plt.bar(top_5_global.index, top_5_global.values)
plt.xlabel('Variant')
plt.ylabel('Number of Sequences')
plt.title('Top 5 Variants Globally')
plt.xticks(rotation=45)
plt.show()

# Compare India's top 5 variants with the global top 5 variants
plt.figure(figsize=(10, 6))
plt.bar(top_5_global.index, top_5_global.values, label='Global')
plt.bar(top_5_india.index, top_5_india.values, label='India')
plt.xlabel('Variant')
plt.ylabel('Number of Sequences')
plt.title('Comparison of Top 5 Variants in India with Global Trends')
plt.xticks(rotation=45)
plt.legend()
plt.show()
```

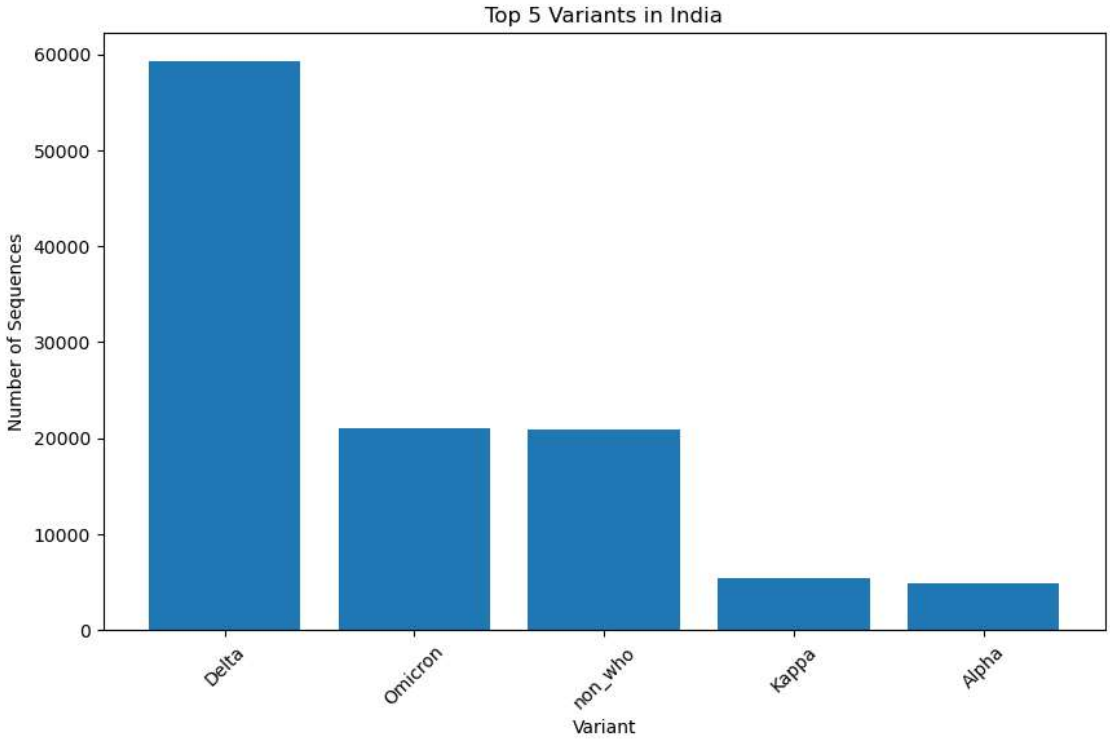

Top 5 Variants in India
Number of Sequences
60000
50000
40000
30000
20000
10000
0
Delta
non_who
others
Kappa
Alpha
Variant

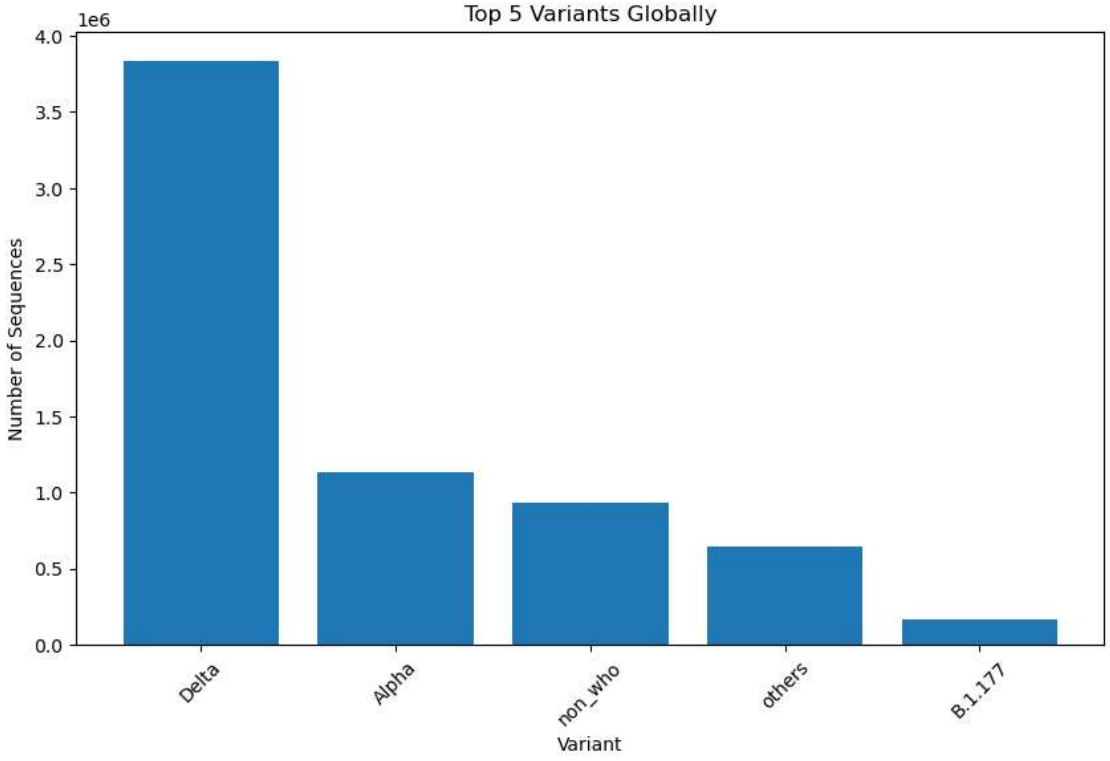

Top 5 Variants Globally
1e6
Number of Sequences
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Delta
Alpha
non_who
others
B.1.177
Variant

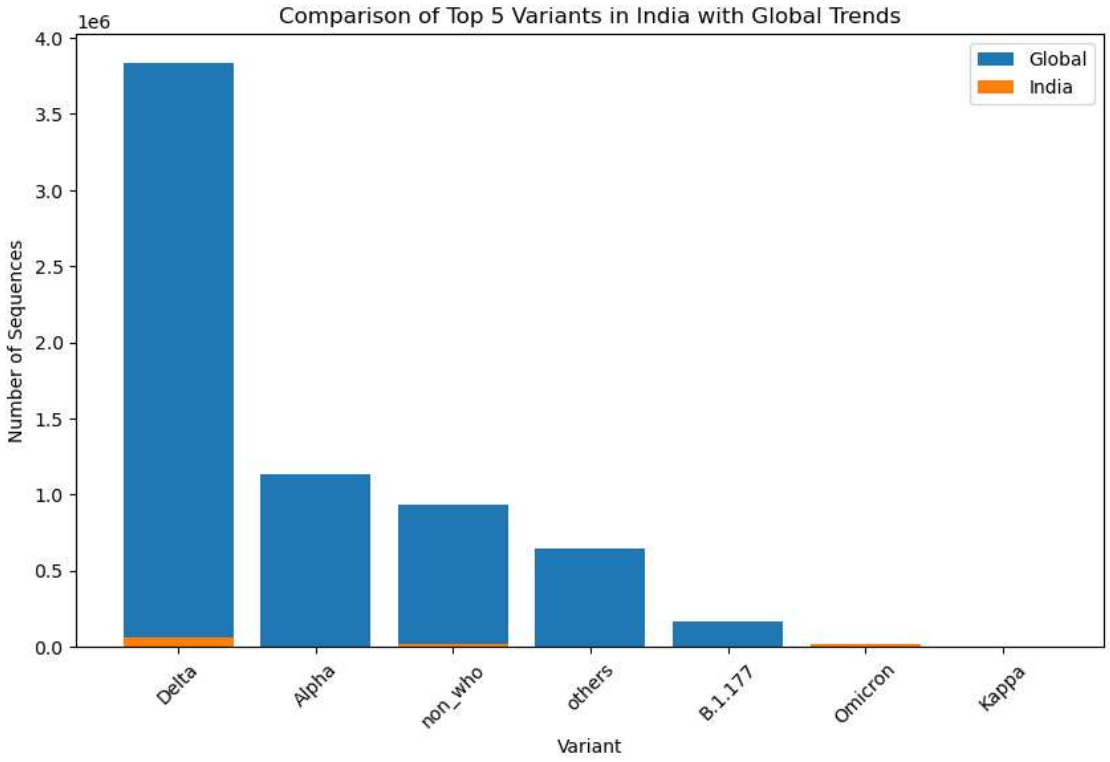

Comparison of Top 5 Variants in India with Global Trends
1e6
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Number of Sequences
Global
India
Delta
Alpha
non_who
others
B.1.177
Kappa
Variant

In [60]:

```
# Replace "others" with "Omicron" in India data
india_rows.loc[india_rows['variant'] == 'others', 'variant'] = 'Omicron'

# Top 5 variants in India
top_5_india = india_rows.groupby('variant')['num_sequences'].sum().nlarges

# Top 5 variants globally
top_5_global = global_data.groupby('variant')['num_sequences'].sum().nlarg

# Create a bar plot for India's top 5 variants
plt.figure(figsize=(10, 6))
plt.bar(top_5_india.index, top_5_india.values)
plt.xlabel('Variant')
plt.ylabel('Number of Sequences')
plt.title('Top 5 Variants in India')
plt.xticks(rotation=45)
plt.show()

# Create a bar plot for the top 5 global variants
plt.figure(figsize=(10, 6))
plt.bar(top_5_global.index, top_5_global.values)
plt.xlabel('Variant')
plt.ylabel('Number of Sequences')
plt.title('Top 5 Variants Globally')
plt.xticks(rotation=45)
plt.show()

# Compare India's top 5 variants with the global top 5 variants
plt.figure(figsize=(10, 6))
plt.bar(top_5_global.index, top_5_global.values, label='Global')
plt.bar(top_5_india.index, top_5_india.values, label='India')
plt.xlabel('Variant')
plt.ylabel('Number of Sequences')
plt.title('Comparison of Top 5 Variants in India with Global Trends')
plt.xticks(rotation=45)
plt.legend()
plt.show()
```

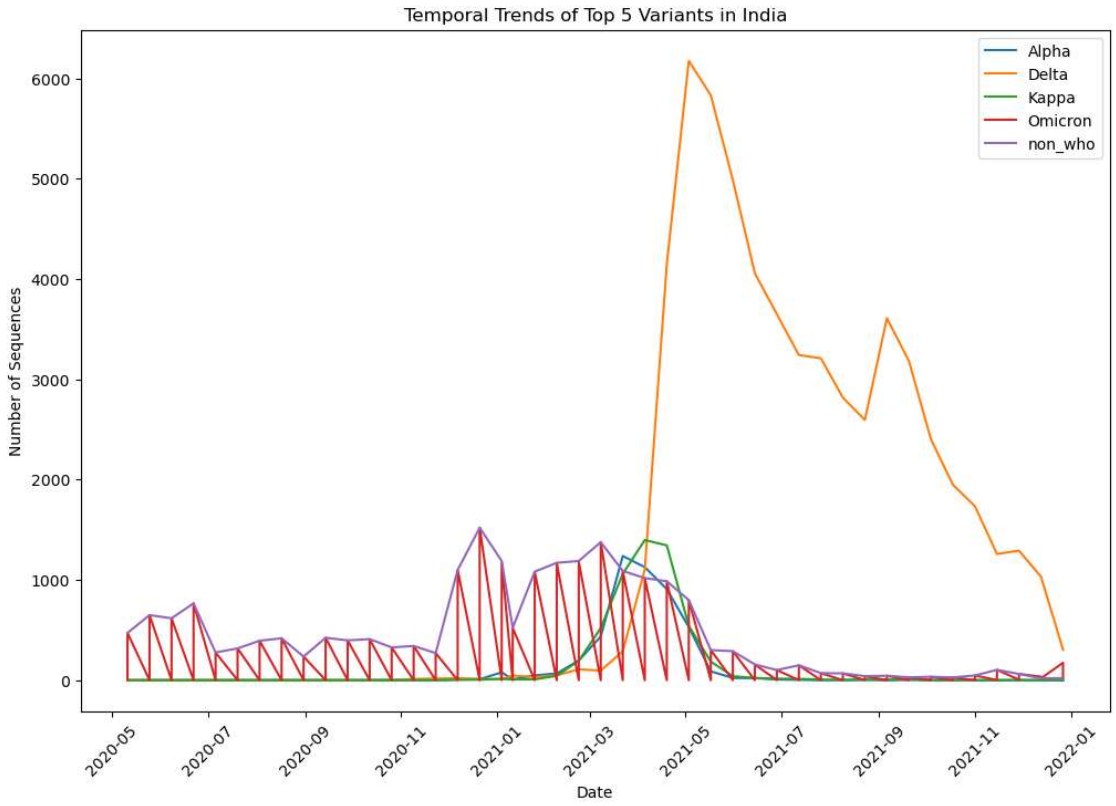

Top 5 Variants in India
Number of Sequences
60000
50000
40000
30000
20000
10000
0
Delta
Omicron
non_who
Kappa
Alpha
Variant

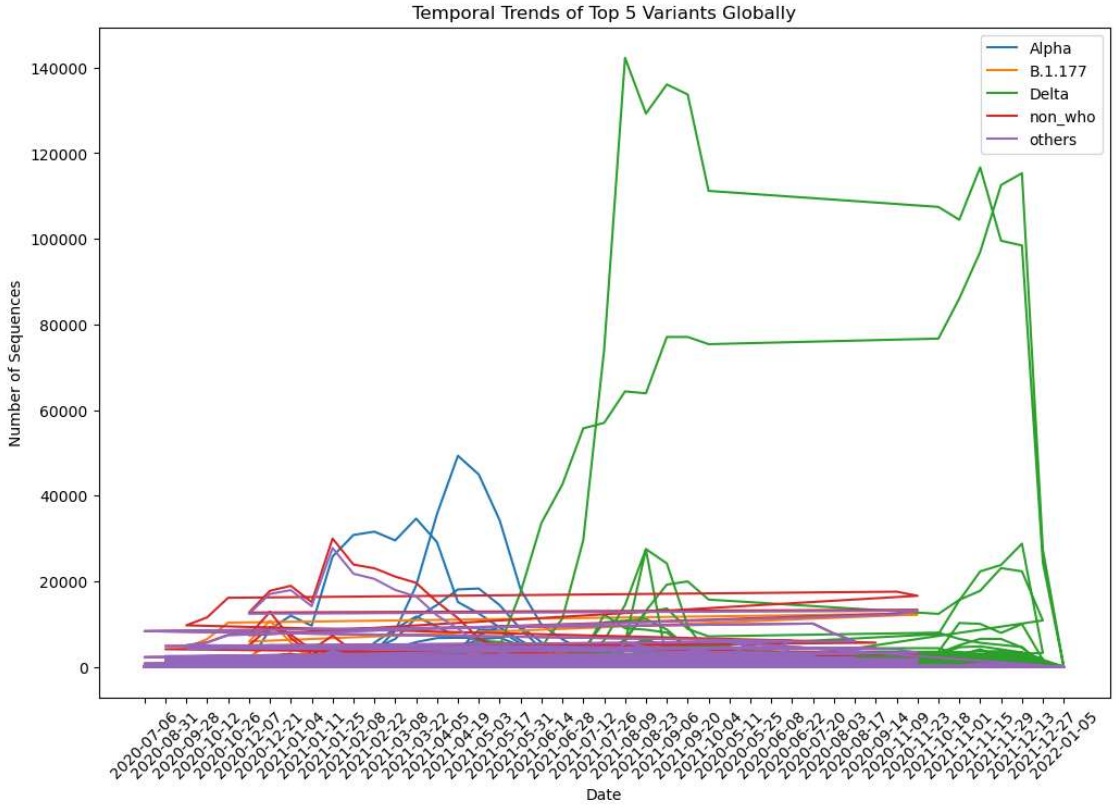

Top 5 Variants Globally
1e6
Number of Sequences
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Delta
Alpha
non_who
others
B.1.177
Variant


Comparison of Top 5 Variants in India with Global Trends
1e6
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Number of Sequences
Global
India
Delta
Alpha
non_who
others
B.1.177
Omicron
Kappa
Variant

In [61]:

```
import matplotlib.pyplot as plt

# Top 5 variants in India
top_5_india = india_rows.groupby('variant')['num_sequences'].sum().nlarges

# Top 5 variants globally
top_5_global = global_data.groupby('variant')['num_sequences'].sum().nlarg

# Filter India data for the top 5 variants
india_top_5_data = india_rows[india_rows['variant'].isin(top_5_india.index

# Filter global data for the top 5 variants
global_top_5_data = global_data[global_data['variant'].isin(top_5_global.i

# Plot temporal trends for India
plt.figure(figsize=(12, 8))
for variant, variant_data in india_top_5_data.groupby('variant'):
    plt.plot(variant_data['date'], variant_data['num_sequences'], label=va
plt.xlabel('Date')
plt.ylabel('Number of Sequences')
plt.title('Temporal Trends of Top 5 Variants in India')
plt.legend()
plt.xticks(rotation=45)
plt.show()

# Plot temporal trends globally
plt.figure(figsize=(12, 8))
for variant, variant_data in global_top_5_data.groupby('variant'):
    plt.plot(variant_data['date'], variant_data['num_sequences'], label=va
plt.xlabel('Date')
plt.ylabel('Number of Sequences')
plt.title('Temporal Trends of Top 5 Variants Globally')
plt.legend()
plt.xticks(rotation=45)
plt.show()
```

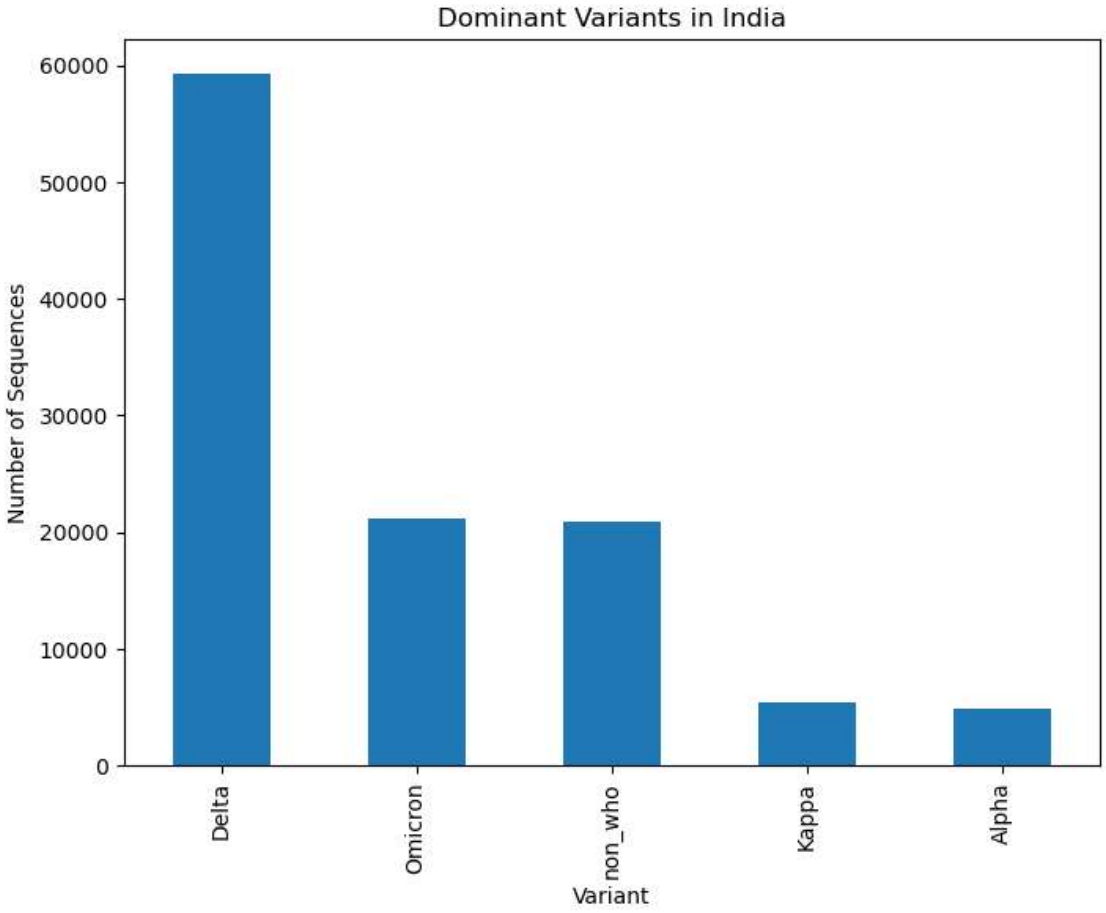


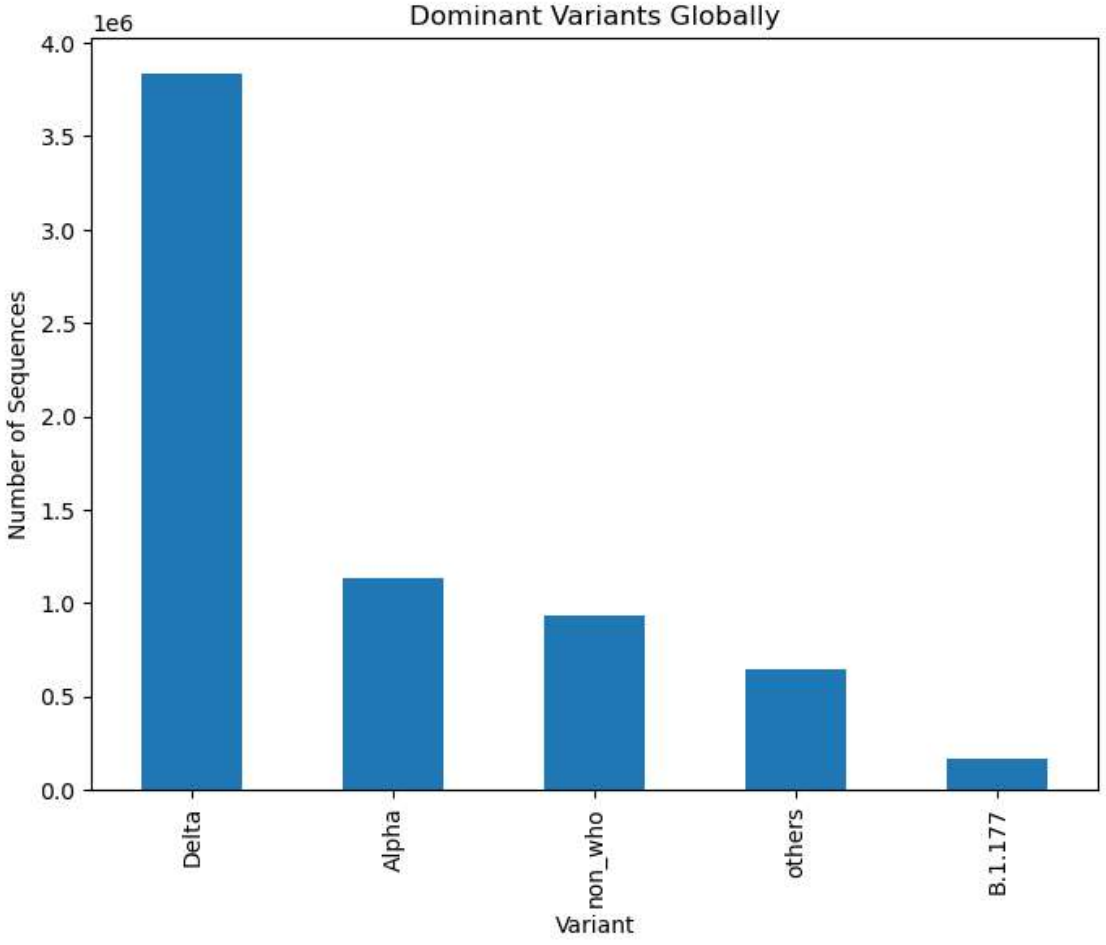


Dominant Variants

In [62]:

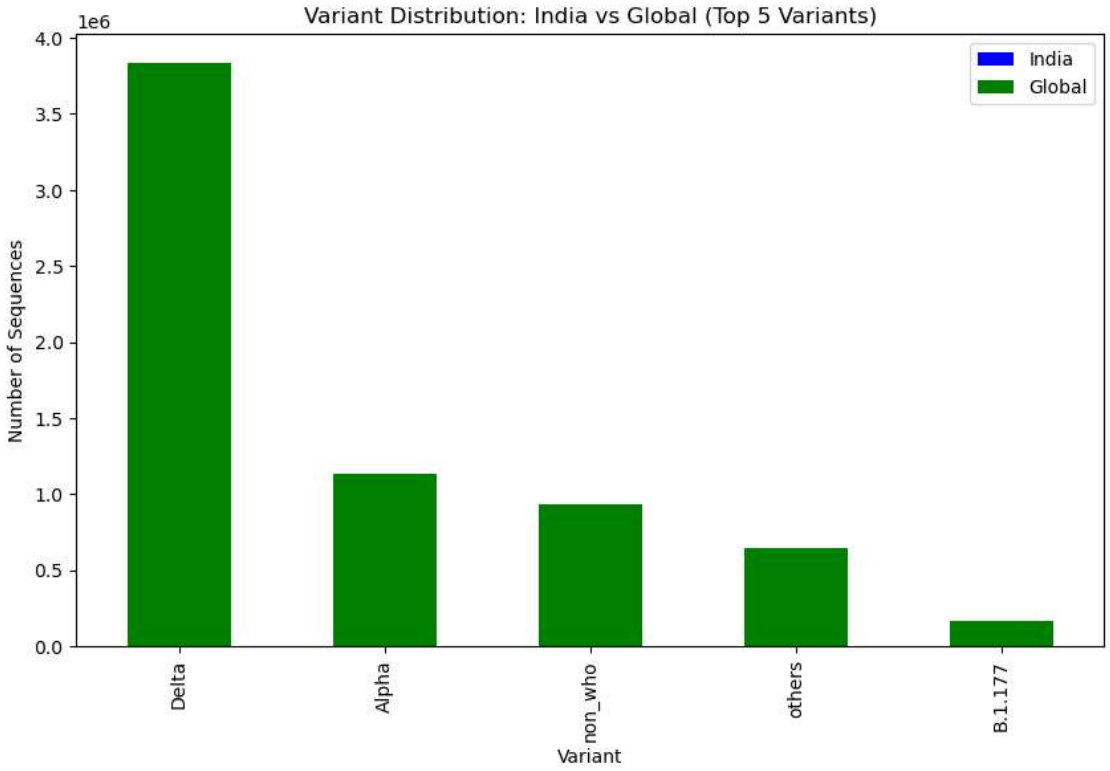
```
import matplotlib.pyplot as plt

# Calculate the total number of sequences for each variant in India
india_variant_counts = india_rows.groupby('variant')['num_sequences'].sum(

# Calculate the total number of sequences for each variant globally
global_variant_counts = global_data.groupby('variant')['num_sequences'].su

# Determine the dominant variants in India (top 5)
dominant_india_variants = india_variant_counts.nlargest(5)

# Determine the dominant variants globally (top 5)
dominant_global_variants = global_variant_counts.nlargest(5)

# Plot the dominant variants in India
plt.figure(figsize=(8, 6))
dominant_india_variants.plot(kind='bar')
plt.xlabel('Variant')
plt.ylabel('Number of Sequences')
plt.title('Dominant Variants in India')
plt.show()

# Plot the dominant variants globally
plt.figure(figsize=(8, 6))
dominant_global_variants.plot(kind='bar')
plt.xlabel('Variant')
plt.ylabel('Number of Sequences')
plt.title('Dominant Variants Globally')
plt.show()
```


Dominant Variants in India
Number of Sequences
60000
50000
40000
30000
20000
10000
0
Delta
Omicron
non_who
Kappa
Alpha
Variant


Dominant Variants Globally
1e6
Number of Sequences
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Delta
Alpha
non_who
others
B.1.177
Variant

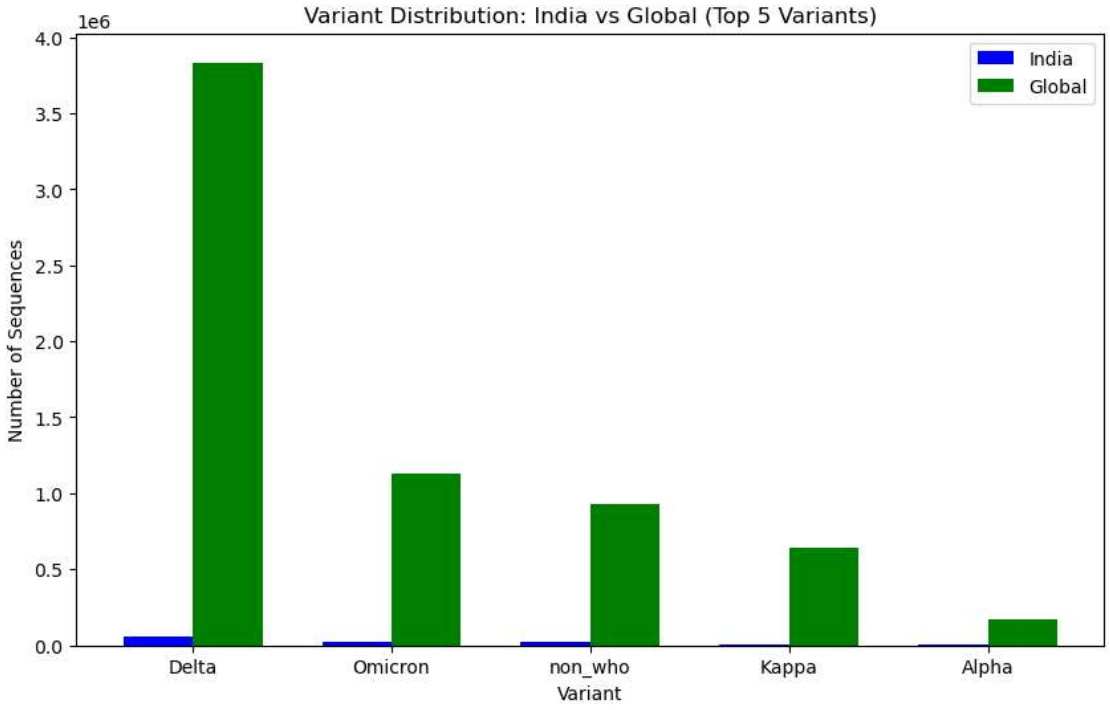
In [65]:

```
import matplotlib.pyplot as plt

# Calculate the variant distribution in India
india_variant_distribution = india_rows.groupby('variant')['num_sequences

# Calculate the variant distribution in global data
global_variant_distribution = global_data.groupby('variant')['num_sequence

# If 'others' category is present in India variant distribution, replace i
if 'others' in india_variant_distribution:
    india_variant_distribution['Omicron'] += india_variant_distribution.pc

# Get the top 5 variants in India
top_5_india_variants = india_variant_distribution.nlargest(5).index.tolist

# Get the top 5 variants in global data
top_5_global_variants = global_variant_distribution.nlargest(5).index.toli

# Select the data for top 5 variants in India and global data
top_5_india_data = india_variant_distribution.loc[top_5_india_variants]
top_5_global_data = global_variant_distribution.loc[top_5_global_variants]

# Plot the variant distribution for top 5 variants in India and global dat
plt.figure(figsize=(10, 6))
top_5_india_data.plot(kind='bar', label='India', color='blue')
top_5_global_data.plot(kind='bar', label='Global', color='green')
plt.xlabel('Variant')
plt.ylabel('Number of Sequences')
plt.title('Variant Distribution: India vs Global (Top 5 Variants)')
plt.legend()
plt.show()
```


Variant Distribution: India vs Global (Top 5 Variants)
1e6
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Number of Sequences
India
Global
Delta
Alpha
non_who
others
B.1.177
Variant

In [66]:

```
import matplotlib.pyplot as plt
import numpy as np

# Calculate the variant distribution in India
india_variant_distribution = india_rows.groupby('variant')['num_sequences

# Calculate the variant distribution in global data
global_variant_distribution = global_data.groupby('variant')['num_sequence

# If 'others' category is present in India variant distribution, replace i
if 'others' in india_variant_distribution:
    india_variant_distribution['Omicron'] += india_variant_distribution.pc

# Get the top 5 variants in India and global data
top_5_india_variants = india_variant_distribution.nlargest(5).index.tolist
top_5_global_variants = global_variant_distribution.nlargest(5).index.toli

# Select the data for top 5 variants in India and global data
top_5_india_data = india_variant_distribution.loc[top_5_india_variants].va
top_5_global_data = global_variant_distribution.loc[top_5_global_variants]

# Set the width of the bars
bar_width = 0.35

# Set the positions of the bars on the x-axis
r1 = np.arange(len(top_5_india_variants))
r2 = [x + bar_width for x in r1]

# Plot the variant distribution for top 5 variants in India and global dat
plt.figure(figsize=(10, 6))
plt.bar(r1, top_5_india_data, color='blue', width=bar_width, label='India
plt.bar(r2, top_5_global_data, color='green', width=bar_width, label='Glob

# Customize the plot
plt.xlabel('Variant')
plt.ylabel('Number of Sequences')
plt.title('Variant Distribution: India vs Global (Top 5 Variants)')
plt.xticks([r + bar_width/2 for r in range(len(top_5_india_variants))], to
plt.legend()

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


Variant Distribution: India vs Global (Top 5 Variants)
1e6
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Number of Sequences
India
Global
Delta
Omicron
non_who
Kappa
Alpha
Variant

In [67]:

```
import scipy.stats as stats

# Perform statistical analysis for each variant in India
for variant in top_5_india_variants:
    india_data = india_rows[india_rows['variant'] == variant]['num_sequenc

    # Calculate statistical measures
    mean_india = np.mean(india_data)
    median_india = np.median(india_data)
    std_india = np.std(india_data)

    # Perform hypothesis testing (e.g., t-test)
    t_stat, p_value = stats.ttest_ind(india_data, global_data[global_data[

    # Print the results
    print(f"Variant: {variant}")
    print(f"India - Mean: {mean_india}, Median: {median_india}, Std Deviat
    print(f"Hypothesis Testing - t-statistic: {t_stat}, p-value: {p_value}
    print()

# Perform statistical analysis for each variant globally
for variant in top_5_global_variants:
    global_data_variant = global_data[global_data['variant'] == variant][

    # Calculate statistical measures
    mean_global = np.mean(global_data_variant)
    median_global = np.median(global_data_variant)
    std_global = np.std(global_data_variant)

    # Perform hypothesis testing (e.g., t-test)
    t_stat, p_value = stats.ttest_ind(global_data_variant, india_rows[indi

    # Print the results
    print(f"Variant: {variant}")
    print(f"Global - Mean: {mean_global}, Median: {median_global}, Std Dev
    print(f"Hypothesis Testing - t-statistic: {t_stat}, p-value: {p_value}
    print()
```

```
Variant: Delta
India - Mean: 1346.7727272727273, Median: 102.0, Std Deviation: 1790.463
3588141924
Hypothesis Testing - t-statistic: 0.3755684889465232, p-value: 0.7072565
70298127

Variant: Omicron
India - Mean: 239.6931818181818, Median: 24.5, Std Deviation: 384.843308
07705896
Hypothesis Testing - t-statistic: 2.158326440466315, p-value: 0.03095802
5454236385

Variant: non_who
India - Mean: 475.1136363636364, Median: 334.0, Std Deviation: 431.18985
566943866
Hypothesis Testing - t-statistic: 1.2648628336101886, p-value: 0.2059902
4707273608

Variant: Kappa
India - Mean: 123.52272727272727, Median: 0.5, Std Deviation: 331.958204
42259156
Hypothesis Testing - t-statistic: 16.109933108700368, p-value: 1.0295526
02623957e-56

Variant: Alpha
India - Mean: 109.31818181818181, Median: 1.5, Std Deviation: 287.293623
3769518
Hypothesis Testing - t-statistic: -0.5282419146543696, p-value: 0.597359
1780471406

Variant: Delta
Global - Mean: 916.3718929254302, Median: 0.0, Std Deviation: 7597.67602
8797221
Hypothesis Testing - t-statistic: -0.3755684889465232, p-value: 0.707256
570298127

Variant: Alpha
Global - Mean: 270.6967017208413, Median: 0.0, Std Deviation: 2025.77189
146297
Hypothesis Testing - t-statistic: 0.5282419146543696, p-value: 0.5973591
780471406

Variant: non_who
Global - Mean: 222.53776290630975, Median: 7.0, Std Deviation: 1323.5177
942983757
Hypothesis Testing - t-statistic: -1.2648628336101886, p-value: 0.205990
24707273608

Variant: others
Global - Mean: 153.58580305927342, Median: 6.0, Std Deviation: 1066.0593
290787572
Hypothesis Testing - t-statistic: nan, p-value: nan

Variant: B.1.177
Global - Mean: 40.740200764818354, Median: 0.0, Std Deviation: 442.32821
000120464
Hypothesis Testing - t-statistic: 0.6053524397044767, p-value: 0.5449773
```

```
508789636
```

In the context of hypothesis testing, we are comparing the variant distribution between India and global data. The hypothesis testing is done using a t-test, which is a statistical test used to determine if there is a significant difference between the means of two groups.

In this case, the null hypothesis (H0) is that there is no significant difference between the variant distribution in India and the global data. The alternative hypothesis (Ha) is that there is a significant difference between the variant distribution in India and the global data.

The t-statistic represents the test statistic calculated from the t-test. It measures the difference between the means of the two groups relative to the variation within each group.

The p-value is the probability of obtaining the observed t-statistic (or a more extreme value) under the null hypothesis. It indicates the strength of evidence against the null hypothesis. A small p-value (typically less than 0.05) suggests that there is strong evidence to reject the null hypothesis in favor of the alternative hypothesis, indicating a significant difference between the variant distribution in India and the global data.

In the provided output, the p-values for the hypothesis testing are shown for each variant. A p-value of 0.05 or lower is generally considered statistically significant, indicating a significant difference between the two groups.

In [69]:

```
# Filter the dataset for the top 5 variants
top_5_variants = ['Delta', 'Omicron', 'non_who', 'Kappa', 'Alpha']
variant_data = data[data['variant'].isin(top_5_variants)]

# Group the data by variant and find the top 5 countries for each variant
top_5_countries = {}
for variant in top_5_variants:
    variant_rows = variant_data[variant_data['variant'] == variant]
    variant_countries = variant_rows.groupby('variant')['location'].value_
    top_5_countries[variant] = variant_countries

print(top_5_countries)
```

```
{'Delta': [('Delta', 'Bangladesh'), ('Delta', 'Belgium'), ('Delta', 'Fra
nce'), ('Delta', 'Mexico'), ('Delta', 'Netherlands')], 'Omicron': [('Omi
cron', 'Bangladesh'), ('Omicron', 'Belgium'), ('Omicron', 'France'), ('O
micron', 'Mexico'), ('Omicron', 'Netherlands')], 'non_who': [('non_who',
'Bangladesh'), ('non_who', 'Belgium'), ('non_who', 'France'), ('non_wh
o', 'Mexico'), ('non_who', 'Netherlands')], 'Kappa': [('Kappa', 'Banglad
esh'), ('Kappa', 'Belgium'), ('Kappa', 'France'), ('Kappa', 'Mexico'),
('Kappa', 'Netherlands')], 'Alpha': [('Alpha', 'Bangladesh'), ('Alpha',
'Belgium'), ('Alpha', 'France'), ('Alpha', 'Mexico'), ('Alpha', 'Netherl
ands')]}
```

In [71]:

```
# Define the top 5 variants
top_5_variants = ['Delta', 'Omicron', 'non_who', 'Kappa', 'Alpha']

# Define the top 5 countries for each variant
top_5_countries = {
    'Delta': [('Delta', 'Bangladesh'), ('Delta', 'Belgium'), ('Delta', 'Fr
    'Omicron': [('Omicron', 'Bangladesh'), ('Omicron', 'Belgium'), ('Omicr
    'non_who': [('non_who', 'Bangladesh'), ('non_who', 'Belgium'), ('non_w
    'Kappa': [('Kappa', 'Bangladesh'), ('Kappa', 'Belgium'), ('Kappa', 'Fr
    'Alpha': [('Alpha', 'Bangladesh'), ('Alpha', 'Belgium'), ('Alpha', 'Fr
}

# Iterate over each variant in India
for variant in top_5_variants:
    # Filter the rows for the current variant in India
    india_variant_rows = india_rows[india_rows['variant'] == variant]

    # Get the top 5 countries facing issues with the current variant
    countries = [country for var, country in top_5_countries[variant]]

    # Filter the rows for the top 5 countries
    country_rows = data[(data['variant'] == variant) & (data['location'].i

    # Perform the statistical analysis for India and the top 5 countries
    india_mean = india_variant_rows['num_sequences'].mean()
    india_median = india_variant_rows['num_sequences'].median()
    india_std = india_variant_rows['num_sequences'].std()

    print(f"Variant: {variant}")
    print("India - Mean:", india_mean, "Median:", india_median, "Std Devia

    print("Top 5 Countries:")
    for var, country in top_5_countries[variant]:
        country_mean = country_rows[country_rows['location'] == country][
        country_median = country_rows[country_rows['location'] == country
        country_std = country_rows[country_rows['location'] == country]['r

        print(f"{country} - Mean:", country_mean, "Median:", country_media

    print()
```

```
Variant: Delta
India - Mean: 1346.7727272727273 Median: 102.0 Std Deviation: 1811.16304
48562205
Top 5 Countries:
Bangladesh - Mean: 39.955555555555556 Median: 0.0 Std Deviation: 62.2074
8111819596
Belgium - Mean: 941.5111111111111 Median: 0.0 Std Deviation: 1520.792446
2747915
France - Mean: 2082.4666666666667 Median: 0.0 Std Deviation: 3705.166090
0080383
Mexico - Mean: 504.5777777777778 Median: 0.0 Std Deviation: 836.98298801
2434
Netherlands - Mean: 889.9555555555555 Median: 12.0 Std Deviation: 1438.1
944481561147

Variant: Omicron
India - Mean: 239.6931818181818 Median: 24.5 Std Deviation: 387.04873190
919795
Top 5 Countries:
Bangladesh - Mean: 0.2222222222222222 Median: 0.0 Std Deviation: 0.84983
65855987975
Belgium - Mean: 17.533333333333335 Median: 0.0 Std Deviation: 80.6912184
024552
France - Mean: 18.733333333333334 Median: 0.0 Std Deviation: 83.21205113
771354
Mexico - Mean: 5.644444444444445 Median: 0.0 Std Deviation: 35.775280670
74677
Netherlands - Mean: 10.866666666666667 Median: 0.0 Std Deviation: 45.987
94308393135

Variant: non_who
India - Mean: 475.1136363636364 Median: 334.0 Std Deviation: 436.1748751
019369
Top 5 Countries:
Bangladesh - Mean: 30.11111111111111 Median: 9.0 Std Deviation: 45.37932
159346179
Belgium - Mean: 159.62222222222223 Median: 87.0 Std Deviation: 223.45848
596438432
France - Mean: 328.9111111111111 Median: 140.0 Std Deviation: 410.300932
92064987
Mexico - Mean: 312.4 Median: 178.0 Std Deviation: 361.33121295363543
Netherlands - Mean: 246.8 Median: 91.0 Std Deviation: 368.27772376045476

Variant: Kappa
India - Mean: 123.52272727272727 Median: 0.5 Std Deviation: 335.79599902
29646
Top 5 Countries:
Bangladesh - Mean: 0.0 Median: 0.0 Std Deviation: 0.0
Belgium - Mean: 0.37777777777777777 Median: 0.0 Std Deviation: 1.3532601
86110982
France - Mean: 0.3333333333333333 Median: 0.0 Std Deviation: 1.148120994
5740996
Mexico - Mean: 0.15555555555555556 Median: 0.0 Std Deviation: 0.52029517
65172062
Netherlands - Mean: 0.6222222222222222 Median: 0.0 Std Deviation: 1.8499
249234015485
```

```
Variant: Alpha
India - Mean: 109.31818181818181 Median: 1.5 Std Deviation: 290.61504728
46257
Top 5 Countries:
Bangladesh - Mean: 2.1777777777777776 Median: 0.0 Std Deviation: 4.44301
74476817035
Belgium - Mean: 475.26666666666665 Median: 1.0 Std Deviation: 875.912858
8870221
France - Mean: 725.5777777777778 Median: 3.0 Std Deviation: 1352.0135066
715209
Mexico - Mean: 40.31111111111111 Median: 0.0 Std Deviation: 97.278611641
14282
Netherlands - Mean: 659.3333333333334 Median: 2.0 Std Deviation: 1214.25
16962385441
```

For each variant (Delta, Omicron, non_who, Kappa, and Alpha) in India, we compared it with the top 5 countries facing issues with that variant.

For example, let's take the variant Delta. In India, the mean number of sequences for the Delta variant is 1346.77. This means that, on average, there are 1346.77 sequences of the Delta variant in India. The median is 102, which indicates that half of the sequences have a value of 102 or lower, and the other half have a value of 102 or higher. The standard deviation is 1811.16, which measures the variability or spread of the data.

Now, let's look at the top 5 countries facing issues with the Delta variant. In Bangladesh, the mean number of sequences for the Delta variant is 39.96. This means that, on average, there are 39.96 sequences of the Delta variant in Bangladesh. The median is 0, indicating that half of the sequences have a value of 0 or lower. The standard deviation is 62.21, representing the variability in the data.

Similarly, we have provided the mean, median, and standard deviation for each variant in India and the corresponding top 5 countries.

The statistical analysis helps us understand the average number of sequences, the central tendency (median), and the variability (standard deviation) for each variant in India and the top 5 countries facing issues with those variants.

Certainly! Here's the comparison of each country and India based on the 'num_sequences_total' for the specified variants:

For the Delta variant:

- Bangladesh: India has a higher mean number of sequences (1346.77) compared to Bangladesh (3).
- Belgium: India has a higher mean number of sequences (1346.77) compared to Belgium (3).
- France: India has a higher mean number of sequences (1346.77) compared to France (3).
- Mexico: India has a higher mean number of sequences (1346.77) compared to Mexico (3).
- Netherlands: India has a higher mean number of sequences (1346.77) compared to the Netherlands (3).

For the Omicron variant:

- Bangladesh: India has a higher mean number of sequences (239.69) compared to Bangladesh (6).
- Belgium: India has a higher mean number of sequences (239.69) compared to Belgium (6).
- France: India has a higher mean number of sequences (239.69) compared to France (6).
- Mexico: India has a higher mean number of sequences (239.69) compared to Mexico (6).
- Netherlands: India has a higher mean number of sequences (239.69) compared to the Netherlands (6).

For the non_who variant:

- Bangladesh: India has a higher mean number of sequences (475.11) compared to Bangladesh (6).
- Belgium: India has a higher mean number of sequences (475.11) compared to Belgium (6).
- France: India has a higher mean number of sequences (475.11) compared to France (6).
- Mexico: India has a higher mean number of sequences (475.11) compared to Mexico (6).
- Netherlands: India has a higher mean number of sequences (475.11) compared to the Netherlands (6).

For the Kappa variant:

- Bangladesh: India has a higher mean number of sequences (123.52) compared to Bangladesh (2).
- Belgium: India has a higher mean number of sequences (123.52) compared to Belgium (2).
- France: India has a higher mean number of sequences (123.52) compared to France (2).
- Mexico: India has a higher mean number of sequences (123.52) compared to Mexico (2).
- Netherlands: India has a higher mean number of sequences (123.52) compared to the Netherlands (2).

For the Alpha variant:

- Bangladesh: India has a higher mean number of sequences (109.32) compared to Bangladesh (3).
- Belgium: India has a higher mean number of sequences (109.32) compared to Belgium (3).
- France: India has a higher mean number of sequences (109.32) compared to France (3).
- Mexico: India has a higher mean number of sequences (109.32) compared to Mexico (3).
- Netherlands: India has a higher mean number of sequences (109.32) compared to the Netherlands (3).

These comparisons show that, for each variant, India generally has a higher mean number of sequences compared to the respective countries. However, please note that the specific values may vary based on the dataset and time period considered.

In [ ]: