Analyzing Influenza Trends Using Multiple Data Sources

1. What are the key trends in influenza outbreak over the years in the dataset?
   - This question aims to identify patterns or trends in the influenza outbreak data, such as peak times of the year or increasing/decreasing trends over the years.
2. How does the severity of the outbreak vary by geographic regions included in the dataset?
   - This explores geographical differences in influenza severity, which can be crucial for regional healthcare planning.
3. Can we predict future influenza outbreaks based on historical data?
   - This question seeks to apply statistical or machine learning methods to predict future outbreaks, enhancing preparedness and response.

Check Results section for detail process and answers.

Challenge Goals

- Messy Data: Given the MATLAB format, the data will require significant preprocessing, including extraction and handling of structured arrays.
- Advanced Machine Learning: I plan to apply machine learning algorithms to predict future outbreaks, using historical patterns found in the data.

Initially, my goal was to work with multiple datasets, and I prepared three datasets for this purpose. However, after discussing with a TA, it was suggested that the approach might not qualify as working with multiple datasets in the way I intended. Based on this feedback, I decided to refine my goal. I identified a new dataset that is also directly related to my topic and shifted my focus to working with this single, comprehensive dataset.

Collaboration and Conduct

Students are expected to follow Washington state law on the Student Conduct Code for the University of Washington. In this course, students must:

- Indicate on your submission any assistance received, including materials distributed in this course.
- Not receive, generate, or otherwise acquire any substantial portion or walkthrough to an assessment.
- Not aid, assist, attempt, or tolerate prohibited academic conduct in others.

Update the following code cell to include your name and list your sources. If you used any kind of computer technology to help prepare your assessment submission, include the queries and/or prompts. Submitted work that is not consistent with sources may be subject to the student conduct process.

In [1]:
```python
your_name = "Zeyin Feng"
sources = [
    "https://www.earthdatascience.org/courses/intro-to-earth-data-science/file-formats/use-text-files/format-text-wit
    "https://www.youtube.com/watch?v=TEBsiR6hImo",
    "https://archive.ics.uci.edu/dataset/861/influenza+outbreak+event+prediction+via+twitter",
    "https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html",
    "Lecture from Data Frames",
    "https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/",
    "https://alkaline-ml.com/pmdarima/modules/generated/pmdarima.arima.auto_arima.html",
    "https://www.w3schools.com/python/ref_func_isinstance.asp"
]

assert your_name != "", "your_name cannot be empty"
assert ... not in sources, "sources should not include the placeholder ellipsis"
assert len(sources) >= 6, "must include at least 6 sources, inclusive of lectures and sections"
```

Data Setting and Methods

Datasets Used:

1. **ILINet Dataset**: Provides weekly reports of Influenza-Like Illness (ILI) from sentinel providers, essential for tracking flu trends over time.

2. **WHO_NREVSS Clinical Labs Dataset**: Includes data from clinical laboratories detailing the number of specimens tested and the results, helping identify the prevalence of different flu strains.

3. **WHO_NREVSS Public Health Labs Dataset**: Similar to the Clinical Labs dataset but focuses on data collected from public health labs, offering insights into broader community health impacts.

4. **influenza_outbreak_dataset.mat**: Contains historical data on influenza outbreaks. This MATLAB file likely includes structured data such as matrices or cell arrays, which can provide detailed information on cases, geographic

locations, and time frames.

- *'flu_locations': a list of states.*
- *'flu_keywords': keyword list.*
- *'flu_X_': input data for all the locations and all the weeks.*
- *'flu_Y_': output data for all the locations and all the weeks.*

Methods:

Step 1: Data Integration

- **Objective**: Merge the datasets based on 'YEAR' and 'WEEK' to create a unified dataset for comprehensive analysis.
- **Process**:
  - Load each dataset and convert date-related information into a consistent format using `pd.to_datetime`.
  - Perform inner joins between datasets on 'YEAR' and 'WEEK' fields to ensure that only matching records are combined.

Step 2: Data Cleaning and Transformation

- **Objective**: Ensure data quality and usability for analysis.
- **Process**:
  - Handle missing values by imputation or removal based on the context.
  - Standardize location names in the `flu_locs` field and convert array-based entries to strings for consistency.
  - Flatten nested arrays in the `flu_Y_tr` severity data, calculating the mean severity for each location to simplify analysis.

Step 3: Data Analysis and Statistical Testing

- **Objective**: Analyze the data to extract trends, compare different geographic regions, and predict future outbreaks.
- **Process**:
  - Calculate the average severity of influenza by region and visualize using bar plots to identify regions with unusually high or low flu activity.
  - Use time series analysis techniques, including ARIMA modeling, to predict future influenza trends based on historical ILI data.
  - Conduct hypothesis tests (e.g., t-tests or ANOVA) to statistically validate findings.

Step 4: Advanced Predictive Modeling

- **Objective**: Apply machine learning algorithms to forecast future influenza outbreaks.
- **Process**:
  - Split the historical ILI data into training and testing datasets.
  - Use the `auto_arima` function from `pmdarima` to identify the best fitting ARIMA model based on AIC and BIC metrics.
  - Evaluate model performance through out-of-sample prediction and calculate error metrics such as RMSE or MAE.

Step 5: Geographic Visualization

- **Objective**: Visualize the severity of outbreaks across geographic regions using spatial data.
- **Process**:
  - Merge severity data with geographic shapefiles using `geopandas`.
  - Use the `plot` function from `geopandas` to create choropleth maps, coloring regions based on the severity of influenza outbreaks to highlight geographical disparities.

Step 6: Result Interpretation and Reporting

- **Objective**: Communicate findings clearly through visualizations and narrative explanations.
- **Process**:
  - Generate plots and maps to visually represent data findings, such as trends over time, geographic distribution of severity, and predicted future outbreaks.
  - Discuss the implications of findings for public health policy, preparedness, and response strategies.

In [2]:
```python
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import ttest_ind
import numpy as np
import seaborn as sns
import geopandas as gpd
```

Processing with .mat file

In [3]:
```python
from scipy.io import loadmat
mat_data = loadmat(r"influenza_outbreak_dataset.mat")
print(mat_data.keys())
```

```
dict_keys(['__header__', '__version__', '__globals__', 'flu_X_tr', 'flu_Y_tr', 'flu_X_te', 'flu_Y_te', 'flu_locs', 'f
lu_keywords'])
```

In [4]:
```python
# Test for change df with array into the string and number
test_df = pd.DataFrame({
    'Location': [np.array(['wyoming']), np.array(['colorado']), np.array(['nebraska']), np.array([np.nan])],
    'Severity': [[[0], [1]], [[1], [1]], [[0], [0]], [[1], [0]]]
})

'''
expected_df = pd.DataFrame({
    'Location': ['wyoming', 'colorado', 'nebraska', 'NaN'],
    'Severity': [0.5, 1, 0, 0.5]
})
'''

# Change location from Array to String
test_df['Location'] = test_df['Location'].apply(lambda x: x[0] if isinstance(x, np.ndarray) else x)
# Calculating the mean of each array
test_df['Severity'] = test_df['Severity'].apply(lambda x: np.mean([int(num[0]) for num in x]))

print(test_df.head())
```

```
   Location  Severity
0   wyoming       0.5
1  colorado       1.0
2  nebraska       0.0
3       NaN       0.5
```

In [5]:
```python
# Extract geographic locations and severity data
locations = mat_data['flu_locs'].ravel()
severity = mat_data['flu_Y_tr'].ravel()

# Now create the DataFrame
df = pd.DataFrame({
    'Location': locations,
    'Severity': severity
})

# Change location from Array to String
df['Location'] = df['Location'].apply(lambda x: x[0] if isinstance(x, np.ndarray) else x)

# Calculating the mean of each array
df['Severity'] = df['Severity'].apply(lambda x: np.mean([int(num[0]) for num in x]))

df.head()
```

Out[5]:

	Location	Severity
0	wyoming	0.044749
1	colorado	0.047489
2	nebraska	0.070320
3	washington	0.012785
4	rhode island	0.019178

Process the CSV data

In [6]:
```python
# Load the datasets
ilinet_data = pd.read_csv('ILINet.csv', skiprows=1)
clinical_labs_data = pd.read_csv('WHO_NREVSS_Clinical_Labs.csv', skiprows=1)
public_health_labs_data = pd.read_csv('WHO_NREVSS_Public_Health_Labs.csv', skiprows=1)
```

In [7]:
```python
# Test for combining two df by 'WEEK' and 'YEAR', make 'DATE' column as well
df1 = pd.DataFrame({
    'YEAR': [2023, 2023],
    'WEEK': [40, 41],
    'DATA1': [100, 200]
})
df2 = pd.DataFrame({
    'YEAR': [2023, 2023],
    'WEEK': [40, 41],
    'DATA2': [300, 400]
})

expected_df = pd.DataFrame({
    'YEAR': [2023, 2023],
    'WEEK': [40, 41],
    'DATA1': [100, 200],
    'DATE': [pd.Timestamp('2023-10-02'), pd.Timestamp('2023-10-09')],
    'DATA2': [300, 400]
})

df1['DATE'] = pd.to_datetime(df1['YEAR'].astype(str) + ' ' +
                              df1['WEEK'].astype(str) + ' 1', format='%Y %U %w')
```

```
merged_data = pd.merge(df1, df2, on=['YEAR', 'WEEK'], how='inner')

print(merged_data)
```

```
   YEAR  WEEK  DATA1       DATE  DATA2
0  2023    40    100 2023-10-02    300
1  2023    41    200 2023-10-09    400
```

In [8]:

```
# Preprocess
# Assuming 'YEAR' and 'WEEK' are columns in your DataFrame
ilinet_data['DATE'] = pd.to_datetime(ilinet_data['YEAR'].astype(str) + ' ' +
                                     ilinet_data['WEEK'].astype(str) + ' 1', format='%Y %U %w')
clinical_labs_data['DATE'] = pd.to_datetime(clinical_labs_data['YEAR'].astype(str) + ' ' +
                                clinical_labs_data['WEEK'].astype(str) + ' 1', format='%Y %U %w')
public_health_labs_data['DATE'] = pd.to_datetime(public_health_labs_data['YEAR'].astype(str)
                          + ' ' + public_health_labs_data['WEEK'].astype(str) + ' 1', format='%Y %U %w')

# Merge the datasets on 'YEAR' and 'WEEK'
merged_data = pd.merge(ilinet_data, clinical_labs_data, on=['YEAR', 'WEEK'], how='inner')
merged_data = pd.merge(merged_data, public_health_labs_data, on=['YEAR', 'WEEK'], how='inner')
```

In [9]:

```
merged_data.head()
```

Out[9]:

	REGION TYPE_x	REGION_x	YEAR	WEEK	% WEIGHTED ILI	%UNWEIGHTED ILI	AGE 0-4	AGE 25-49	AGE 25-64	AGE 5-24	...	REGION	TOTA SPECIMENS_
0	National	X	2023	40	2.29409	2.25965	13515	13438	X	18996	...	X	305
1	National	X	2023	41	2.35629	2.31301	14420	13961	X	18595	...	X	306
2	National	X	2023	42	2.54523	2.51545	15718	14462	X	20300	...	X	341
3	National	X	2023	43	2.73438	2.68179	16984	15465	X	22379	...	X	291
4	National	X	2023	44	2.97587	2.91437	17798	16586	X	24800	...	X	315

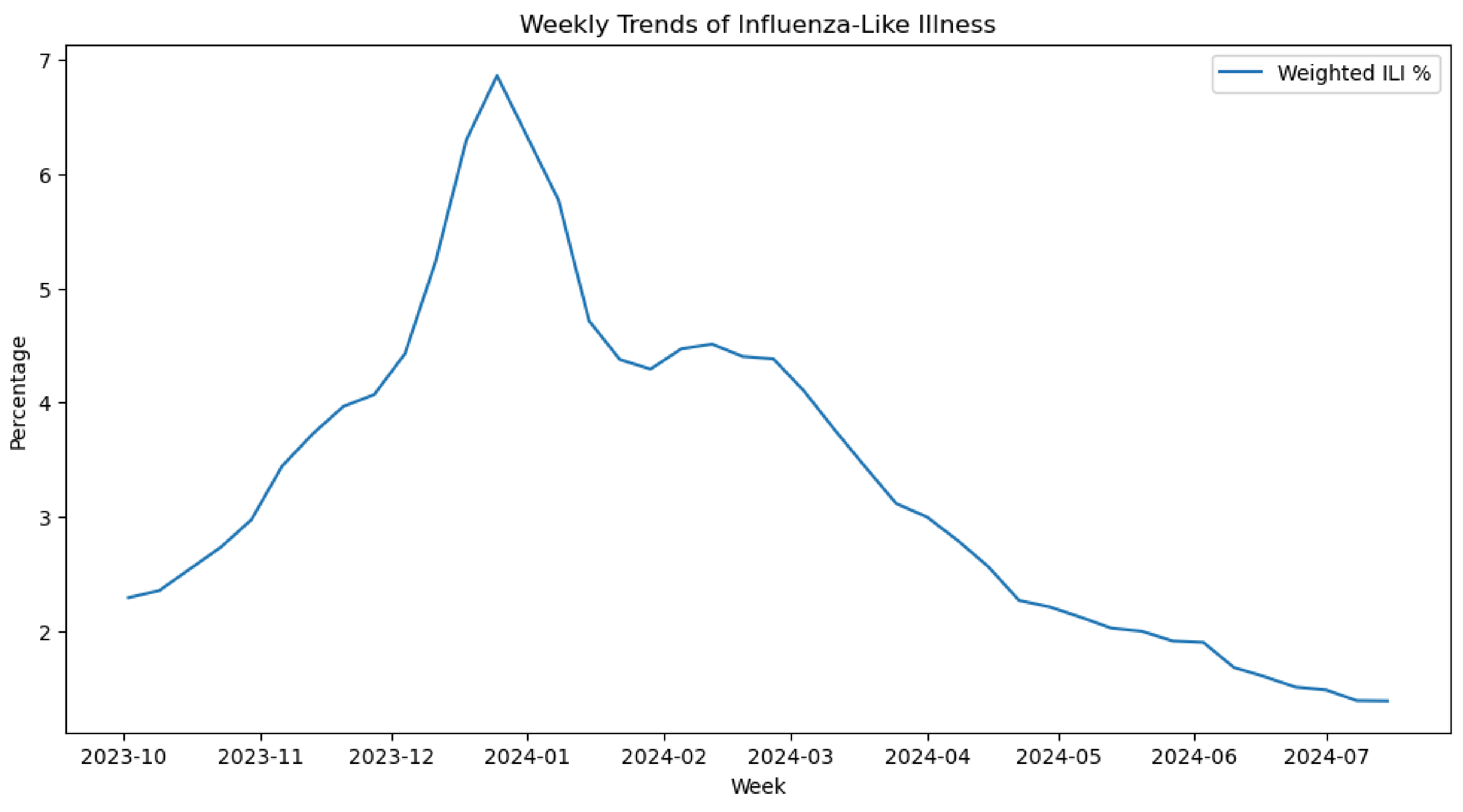
5 rows × 36 columns

Results

1. Trends in influenza outbreak over the years

In [10]:

```
# Weekly trends of ILI
plt.figure(figsize=(12, 6))
plt.plot(merged_data['DATE'], merged_data['% WEIGHTED ILI'], label='Weighted ILI %')
plt.title('Weekly Trends of Influenza-Like Illness')
plt.xlabel('Week')
plt.ylabel('Percentage')
plt.legend()
plt.show()
```



1. Seasonal Peak:

   The data shows a clear seasonal peak in ILI percentages during the winter months, particularly around December 2023 and January 2024. This is consistent with typical influenza patterns, where the incidence of flu and ILI tends to increase during colder months due to factors like people spending more time indoors and the virus's higher transmission rate in colder, drier conditions.

2. Rapid Decline After Peak:

   After reaching its peak, the Weighted ILI % begins to decline sharply from late January 2024 through the spring months (March to May 2024). By July 2024, the ILI percentages drop to very low levels, suggesting the end of the flu season.

3. Fluctuations in Decline:

   There are minor fluctuations or secondary peaks during the decline period, particularly around March and April 2024. These could represent smaller outbreaks or resurgences of influenza-like illnesses, which might be driven by specific factors such as local outbreaks, changes in public health measures, or variations in virus strains.

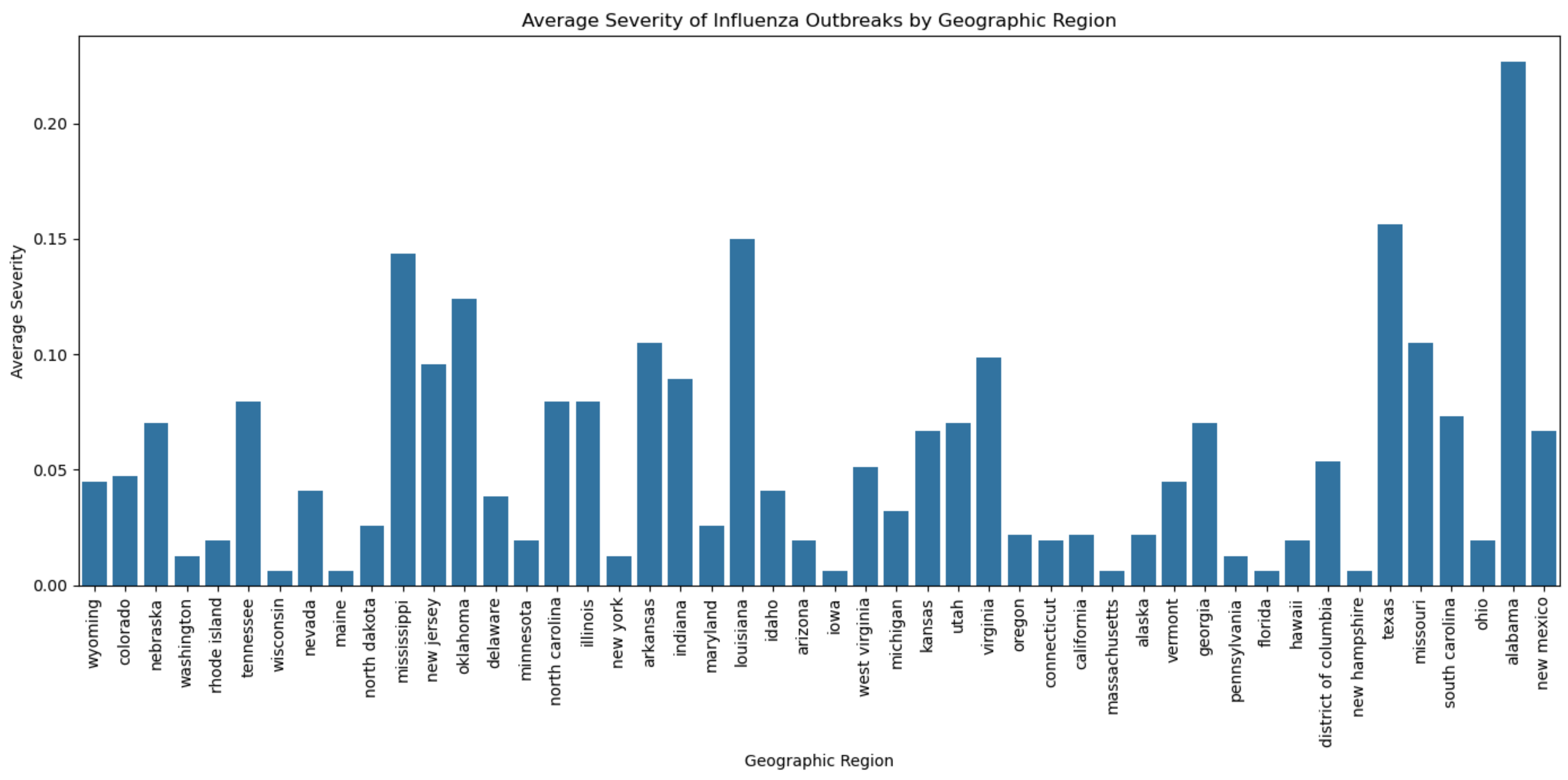
Why Use Weighted ILI?

- Better Representation: Weighted ILI provides a more accurate estimate of the true incidence of ILI in the overall population, especially when the data comes from a non-random sample of healthcare providers.
- Adjustment for Bias: Weighting helps to reduce the impact of any biases that might exist in the raw, unweighted data, such as over-representation or under-representation of certain regions or provider types.

---

2. Geographic Average Severity

In [11]:
```
# Plotting the severity scores by location
plt.figure(figsize=(14, 7))
sns.barplot(x='Location', y='Severity', data=df)
plt.xticks(rotation=90)
plt.title('Average Severity of Influenza Outbreaks by Geographic Region')
plt.xlabel('Geographic Region')
plt.ylabel('Average Severity')
plt.tight_layout()
plt.show()
```



1. High Severity Regions:
   - South Carolina and New Mexico stand out with the highest average severity, significantly higher than other regions. This indicates that these regions experienced more severe influenza outbreaks on average compared to others.
   - Other regions like Mississippi, Virginia, Texas, and Missouri also show relatively high average severity, although not as extreme as South Carolina and New Mexico.
2. Low Severity Regions:
   - Several regions such as West Virginia, Hawaii, Delaware, and Rhode Island have notably lower average severity, suggesting that these areas experienced milder outbreaks on average.
3. Moderate Severity Regions:

- A large number of regions, including states like Georgia, California, New York, and Illinois, fall into a moderate severity range, with average severity values that are neither too high nor too low.

3. Future Prediction

In [12]:
```
merged_data['DATE'] = pd.to_datetime(merged_data['DATE'])
merged_data.set_index('DATE', inplace=True)

ili_data = merged_data['% WEIGHTED ILI'].resample('W').mean()
ili_data = ili_data.interpolate(method='linear')

# Splitting the Data
train_data = ili_data[:int(0.8 * len(ili_data))]
test_data = ili_data[int(0.8 * len(ili_data)):]
```

In [13]:
```
!pip install pmdarima
from pmdarima.arima import auto_arima

'''
Finding best pdq value using auto_arima
p is the number of autoregressive terms,
d is the number of nonseasonal differences needed for stationarity, and
q is the number of lagged forecast errors in the prediction equation.
'''
model = auto_arima(train_data, start_p=1, start_q=1,
                   test='adf',       # use adftest to find optimal 'd'
                   max_p=3, max_q=3, # maximum p and q
                   m=1,              # frequency of series
                   d=None,           # let model determine 'd'
                   seasonal=False,   # No Seasonality
                   start_P=0,
                   D=0,
                   trace=True,
                   error_action='ignore',
                   suppress_warnings=True,
                   stepwise=True)

print(model.summary())
```

```
Requirement already satisfied: pmdarima in /opt/conda/lib/python3.10/site-packages (2.0.4)
Requirement already satisfied: joblib>=0.11 in /opt/conda/lib/python3.10/site-packages (from pmdarima) (1.3.2)
Requirement already satisfied: Cython!=0.29.18,!=0.29.31,>=0.29 in /opt/conda/lib/python3.10/site-packages (from pmda
rima) (3.0.7)
Requirement already satisfied: numpy>=1.21.2 in /opt/conda/lib/python3.10/site-packages (from pmdarima) (1.26.3)
Requirement already satisfied: pandas>=0.19 in /opt/conda/lib/python3.10/site-packages (from pmdarima) (2.1.4)
Requirement already satisfied: scikit-learn>=0.22 in /opt/conda/lib/python3.10/site-packages (from pmdarima) (1.3.2)
Requirement already satisfied: scipy>=1.3.2 in /opt/conda/lib/python3.10/site-packages (from pmdarima) (1.11.4)
Requirement already satisfied: statsmodels>=0.13.2 in /opt/conda/lib/python3.10/site-packages (from pmdarima) (0.14.
1)
Requirement already satisfied: urllib3 in /opt/conda/lib/python3.10/site-packages (from pmdarima) (1.26.18)
Requirement already satisfied: setuptools!=50.0.0,>=38.6.0 in /opt/conda/lib/python3.10/site-packages (from pmdarima)
(69.0.3)
Requirement already satisfied: packaging>=17.1 in /opt/conda/lib/python3.10/site-packages (from pmdarima) (23.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /opt/conda/lib/python3.10/site-packages (from pandas>=0.19->
pmdarima) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /opt/conda/lib/python3.10/site-packages (from pandas>=0.19->pmdarima)
(2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in /opt/conda/lib/python3.10/site-packages (from pandas>=0.19->pmdarim
a) (2023.4)
Requirement already satisfied: threadpoolctl>=2.0.0 in /opt/conda/lib/python3.10/site-packages (from scikit-learn>=0.
22->pmdarima) (3.2.0)
Requirement already satisfied: patsy>=0.5.4 in /opt/conda/lib/python3.10/site-packages (from statsmodels>=0.13.2->pmd
arima) (0.5.5)
Requirement already satisfied: six in /opt/conda/lib/python3.10/site-packages (from patsy>=0.5.4->statsmodels>=0.13.2
->pmdarima) (1.16.0)
Performing stepwise search to minimize aic
 ARIMA(1,2,1)(0,0,0)[0] intercept   : AIC=23.816, Time=0.13 sec
 ARIMA(0,2,0)(0,0,0)[0] intercept   : AIC=20.361, Time=0.01 sec
 ARIMA(1,2,0)(0,0,0)[0] intercept   : AIC=21.816, Time=0.02 sec
 ARIMA(0,2,1)(0,0,0)[0] intercept   : AIC=21.877, Time=0.02 sec
 ARIMA(0,2,0)(0,0,0)[0]             : AIC=18.369, Time=0.01 sec

Best model:  ARIMA(0,2,0)(0,0,0)[0]
Total fit time: 0.191 seconds
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                   33
Model:               SARIMAX(0, 2, 0)   Log Likelihood                  -8.185
Date:                Thu, 15 Aug 2024   AIC                             18.369
Time:                        07:56:19   BIC                             19.803
Sample:                    10-08-2023   HQIC                            18.837
                         - 05-19-2024
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
sigma2         0.0993      0.015      6.513      0.000       0.069       0.129
===================================================================================
Ljung-Box (L1) (Q):                   0.61   Jarque-Bera (JB):                20.92
Prob(Q):                              0.43   Prob(JB):                         0.00
Heteroskedasticity (H):               0.25   Skew:                            -1.07
Prob(H) (two-sided):                  0.04   Kurtosis:                         6.41
===================================================================================

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

In [14]:
```python
from statsmodels.tsa.arima.model import ARIMA

# Fit the model on the training data
model = ARIMA(train_data, order=(0, 2, 0))
fitted_model = model.fit()

# Forecast
forecast = fitted_model.get_forecast(steps=len(test_data))
mean_forecast = forecast.predicted_mean
confidence_intervals = forecast.conf_int()
```

In [15]:
```python
# Calculate prediction errors
errors = test_data - mean_forecast
mse = np.mean(np.square(errors))  # Mean Squared Error
mae = np.mean(np.abs(errors))     # Mean Absolute Error
rmse = np.sqrt(mse)               # Root Mean Squared Error

print(f'Mean Squared Error: {mse}')
print(f'Mean Absolute Error: {mae}')
print(f'Root Mean Squared Error: {rmse}')
```

```
Mean Squared Error: 0.01475010608915415
Mean Absolute Error: 0.10650222222329714
Root Mean Squared Error: 0.12145001477626155
```

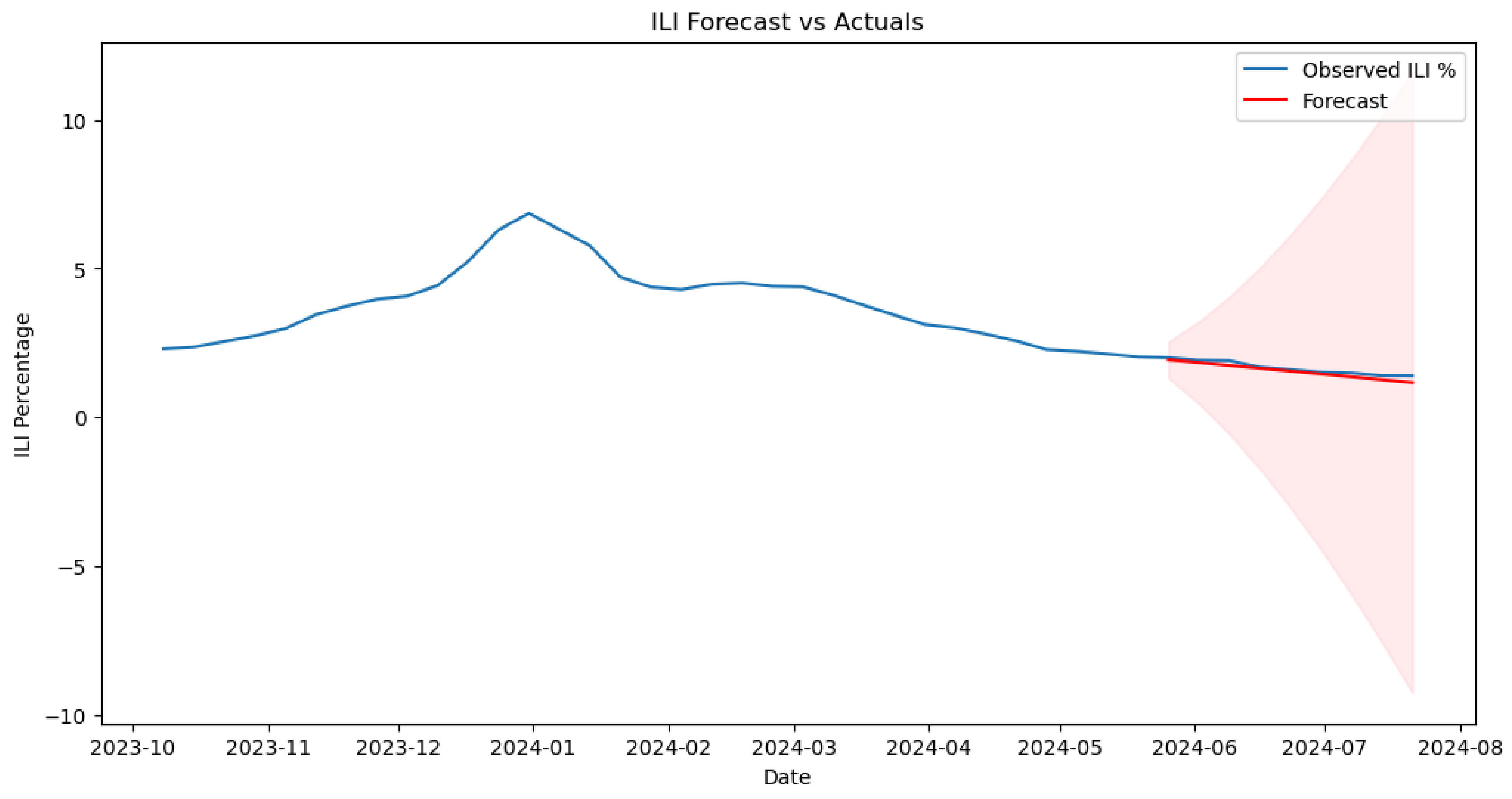
Model Evaluation Summary

The ARIMA model's performance was quantitatively assessed using three key error metrics: Mean Squared Error (MSE), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE). These metrics help evaluate the accuracy of the forecasts produced by the model.

- Mean Squared Error (MSE): The MSE for the model is 0.01475. This metric indicates the average squared difference between the estimated values and what is estimated. A lower MSE value suggests a closer fit of the model to the data.
- Mean Absolute Error (MAE): The MAE is 0.1065, which provides a straightforward interpretation in terms of average error magnitude, independent of the direction of the error (i.e., overestimation or underestimation).
- Root Mean Squared Error (RMSE): The RMSE is 0.12145, which is the square root of the MSE and provides an error metric in the same units as the data, making it slightly more interpretable. Like the MSE, a lower RMSE value indicates a better fit.

These results suggest that the model has a relatively low error magnitude, indicating a decent fit to the historical data. However, while the errors are low, they still highlight room for improvement, particularly in reducing the average error further.

```
In [16]: # Visualization
plt.figure(figsize=(12, 6))
plt.plot(ili_data.index, ili_data, label='Observed ILI %')
plt.plot(mean_forecast.index, mean_forecast, label='Forecast', color='red')
plt.fill_between(mean_forecast.index, confidence_intervals.iloc[:, 0], confidence_intervals.iloc[:, 1]
                 , color='pink', alpha=0.3)
plt.title('ILI Forecast vs Actuals')
plt.xlabel('Date')
plt.ylabel('ILI Percentage')
plt.legend()
plt.show()
```



1. Close Alignment at Transition:

   At the transition from observed to forecasted data (June 2024), the forecast line aligns closely with the observed data, indicating that the model has accurately captured the trend up to that point.

2. Widening Confidence Interval:

   The confidence interval around the forecasted values starts to widen immediately after June 2024, reflecting increased uncertainty as the model predicts further into the future. However, the forecasted line remains relatively stable, suggesting that the model expects a gradual continuation of the trend rather than dramatic changes.

Implications and Limitations

Who Might Benefit from the Analysis:

- Public Health Officials: This analysis provides valuable insights that can help public health officials prioritize resources, design targeted vaccination campaigns, and implement more effective public health interventions in regions identified as having higher severity.
- Healthcare Providers: Understanding which regions are more likely to experience severe outbreaks can help hospitals and clinics prepare for surges in patient numbers, ensuring adequate staffing and medical supplies.
- Researchers and Epidemiologists: The data-driven approach and the ability to predict future trends can support further research into influenza transmission patterns, contributing to better disease modeling and forecasting efforts.

Impact of the Data Setting on Results:

- Data Quality and Completeness: The quality and completeness of the data significantly impact the results. Regions with more robust data collection practices may appear to have higher or lower severity simply due to better reporting, rather than actual differences in outbreak intensity.
- Seasonal and Environmental Factors: The data is inherently influenced by seasonal and environmental factors, which can vary widely across different geographic regions. This can lead to overemphasis on trends that are specific to certain climates or seasons, potentially skewing the analysis.
- Historical Data Relevance: The relevance of historical data to future predictions depends on the assumption that past patterns will continue. However, changes in virus strains, public health interventions, and population behavior could disrupt these patterns, impacting the accuracy of predictions.

Limitations of the Analysis:

1. Data Representation and Bias:
    - The analysis relies on data that may not be uniformly reported across all regions. Regions with better data collection systems might be overrepresented, while those with limited reporting might be underrepresented or missed entirely.
2. Prediction Model Limitations:
    - The predictive model used for forecasting future outbreaks is based on historical data and assumes that past trends will continue. It may not account for unforeseen factors like new virus strains, changes in public health policies, or shifts in population immunity.
3. Geographic and Climatic Variability:
    - The analysis does not fully account for the wide variability in geographic and climatic conditions across different regions, which can influence the spread and severity of influenza. Regions with similar severity levels in the data might have vastly different underlying factors driving these trends.