descriptive_statistics

December 13, 2024

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Read data from the CSV file
df = pd.read_csv('../Data/raw_analyst_ratings.csv')
```

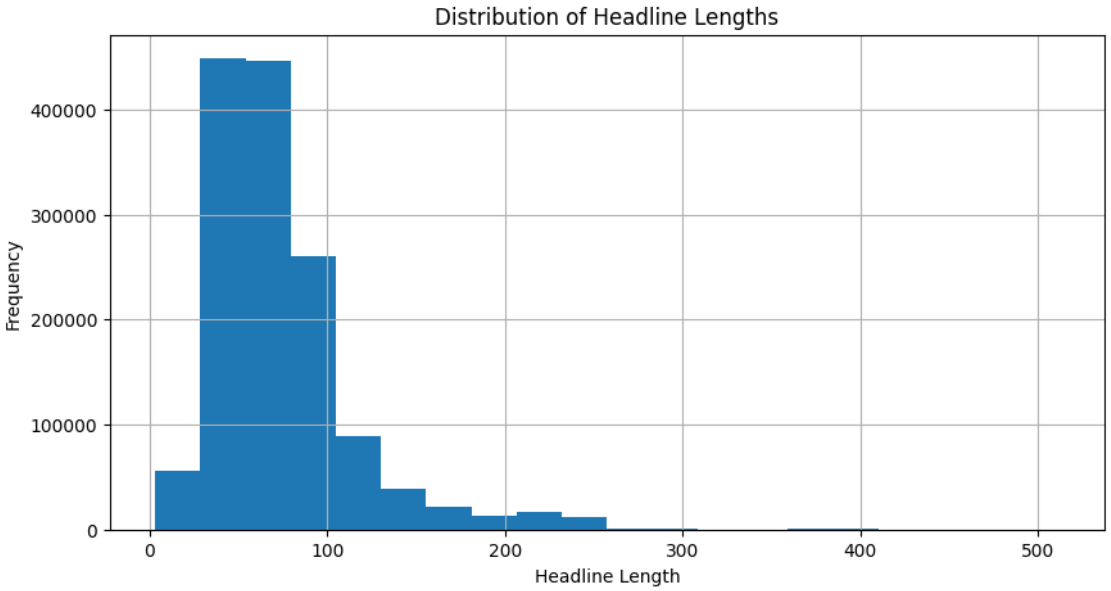
[ ]:
```
# Obtain basic statistics for textual lengths (like headline length)
df['headline_length'] = df['headline'].apply(len)
headline_length_stats = df['headline_length'].describe()

print("Basic Statistics for Headline Lengths:")
print(headline_length_stats)
```

[ ]:
```
# Count the number of articles per publisher
articles_per_publisher = df['publisher'].value_counts()

print("\nNumber of Articles per Publisher:")
print(articles_per_publisher)
```

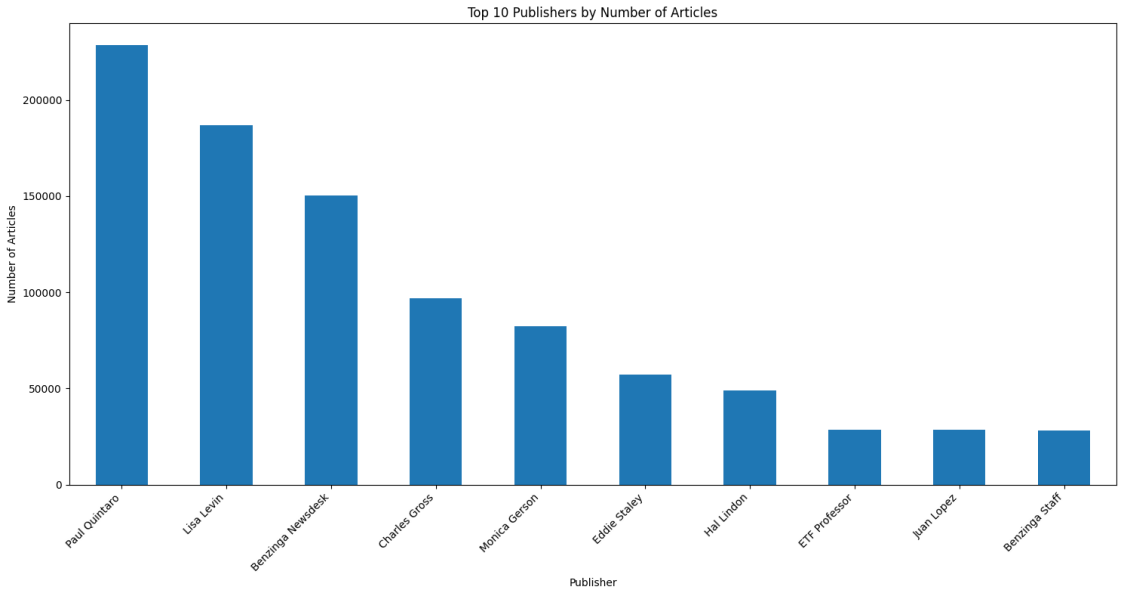
[5]:
```
# Analyze the publication dates to see trends over time
df['date'] = pd.to_datetime(df['date'], utc=True)
df['day_of_week'] = df['date'].dt.day_name()

articles_per_day = df['day_of_week'].value_counts()

print("\nNumber of Articles per Day of the Week:")
print(articles_per_day)
```

```
Number of Articles per Day of the Week:
Thursday     302619
Wednesday    300922
Tuesday      296505
Monday       265139
Friday       217918
Sunday        16466
Saturday       7759
Name: day_of_week, dtype: int64
```

[6]:
```
# Plot headline lengths
plt.figure(figsize=(10, 5))
df['headline_length'].hist(bins=20)
plt.title('Distribution of Headline Lengths')
plt.xlabel('Headline Length')
plt.ylabel('Frequency')
plt.show()
```

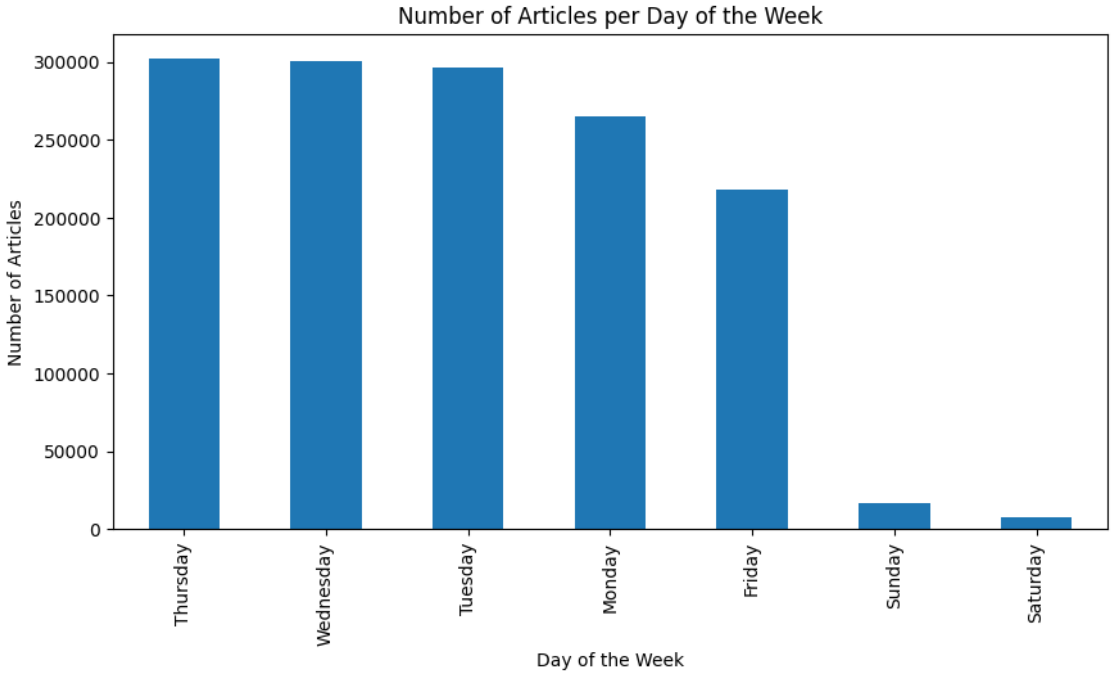


[10]:
```
# Plot articles per publisher (top 10)
top_n = 10  # Change this to display more or fewer publishers
top_publishers = articles_per_publisher.head(top_n)

plt.figure(figsize=(15, 8))
top_publishers.plot(kind='bar')
plt.title(f'Top {top_n} Publishers by Number of Articles')
plt.xlabel('Publisher')
plt.ylabel('Number of Articles')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



[13]:
```
import plotly.express as px

fig = px.bar(articles_per_publisher, title='Number of Articles per Publisher')
fig.update_layout(
    xaxis_title="Publisher",
    yaxis_title="Number of Articles",
    xaxis_tickangle=-45,
    title_x=0.5
)
fig.show()
```

[8]:
```
# Plot articles per day of the week
plt.figure(figsize=(10, 5))
articles_per_day.plot(kind='bar')
plt.title('Number of Articles per Day of the Week')
plt.xlabel('Day of the Week')
plt.ylabel('Number of Articles')
plt.show()
```


Number of Articles per Day of the Week
Number of Articles
300000
250000
200000
150000
100000
50000
0
Thursday
Wednesday
Tuesday
Monday
Friday
Sunday
Saturday
Day of the Week

text_analysis

December 13, 2024

[ ]:
```
import pandas as pd
from textblob import TextBlob
from sklearn.feature_extraction.text import CountVectorizer
import matplotlib.pyplot as plt
import seaborn as sns
import nltk
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
```

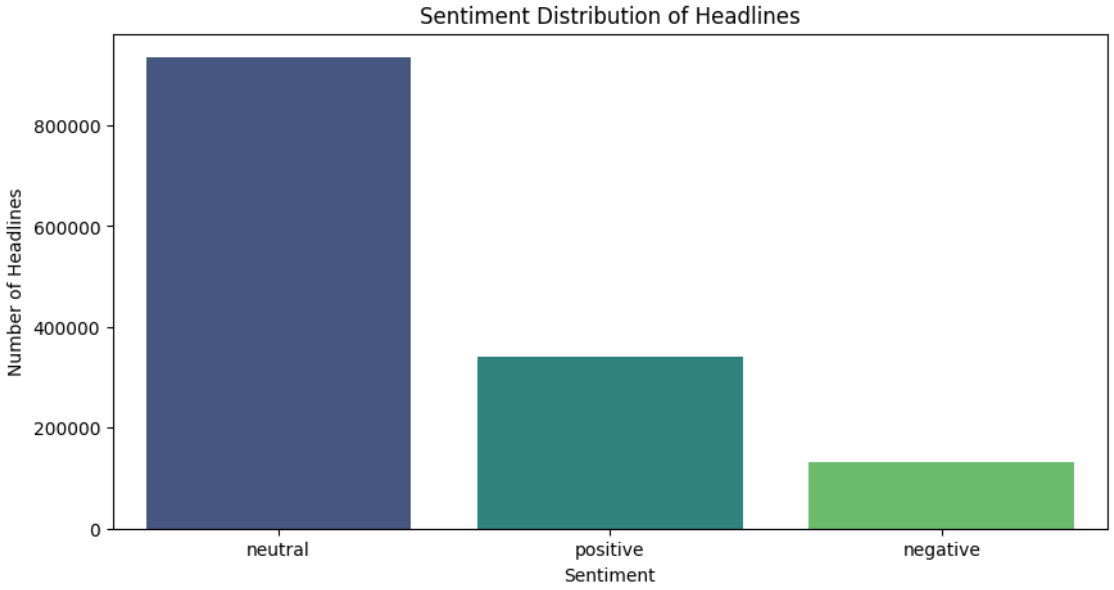
[ ]:
```
# Load the CSV file
df = pd.read_csv('../Data/raw_analyst_ratings.csv')

# Display the first few rows of the dataframe
df.head()
```

[5]:
```
# Function to get the sentiment of a headline
def get_sentiment(text):
    analysis = TextBlob(text)
    # Classify sentiment
    if analysis.sentiment.polarity > 0:
        return 'positive'
    elif analysis.sentiment.polarity < 0:
        return 'negative'
    else:
        return 'neutral'

# Apply the sentiment analysis function
df['sentiment'] = df['headline'].apply(get_sentiment)

# Display the sentiment counts
df['sentiment'].value_counts()
```

[5]:
```
neutral     934914
positive    341178
negative    131236
Name: sentiment, dtype: int64
```

[6]:
```
# Plot the sentiment distribution
plt.figure(figsize=(10, 5))
sns.countplot(x='sentiment', data=df, palette='viridis')
plt.title('Sentiment Distribution of Headlines')
plt.xlabel('Sentiment')
plt.ylabel('Number of Headlines')
plt.show()
```

```
/tmp/ipykernel_93475/2192442204.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.countplot(x='sentiment', data=df, palette='viridis')
```



[7]:
```
# Function to extract keywords
def get_top_n_words(corpus, n=None):
    vec = CountVectorizer(stop_words='english').fit(corpus)
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.
 ↪items()]
    words_freq = sorted(words_freq, key = lambda x: x[1], reverse=True)
    return words_freq[:n]

# Extract top 10 keywords
```

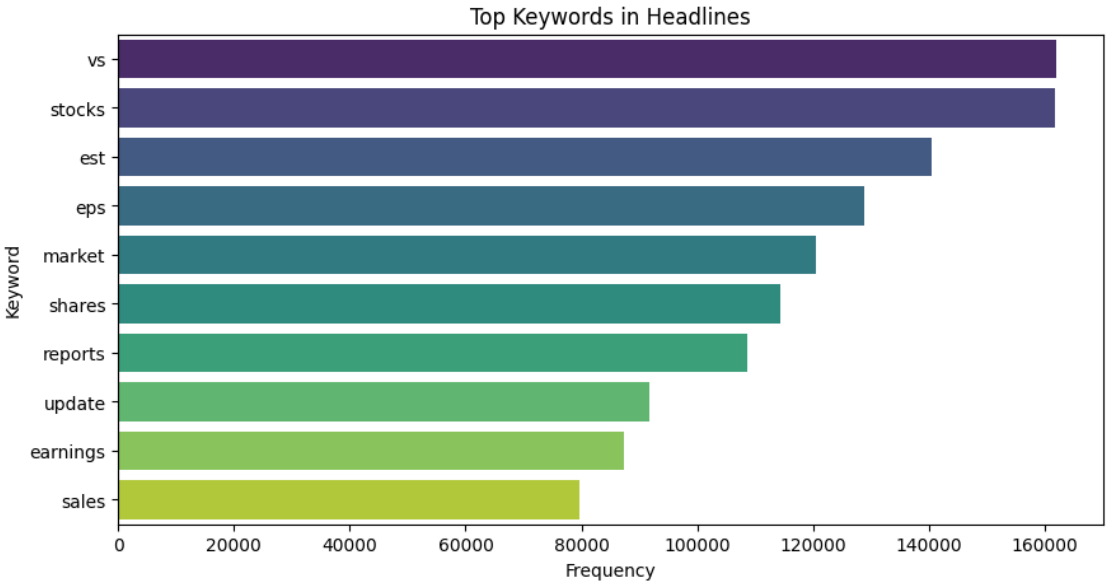
```
top_n_words = get_top_n_words(df['headline'], 10)

# Display the top keywords
top_n_words_df = pd.DataFrame(top_n_words, columns=['Keyword', 'Frequency'])
top_n_words_df
```

[7]:

	Keyword	Frequency
0	vs	162099
1	stocks	161776
2	est	140604
3	eps	128897
4	market	120558
5	shares	114313
6	reports	108710
7	update	91723
8	earnings	87399
9	sales	79645

[8]:
```
# Plot the top keywords
plt.figure(figsize=(10, 5))
sns.barplot(x='Frequency', y='Keyword', data=top_n_words_df, palette='viridis')
plt.title('Top Keywords in Headlines')
plt.xlabel('Frequency')
plt.ylabel('Keyword')
plt.show()
```

```
/tmp/ipykernel_93475/4023718166.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x='Frequency', y='Keyword', data=top_n_words_df,
palette='viridis')
```


Top Keywords in Headlines
vs
stocks
est
eps
market
shares
reports
update
earnings
sales
Keyword
0
20000
40000
60000
80000
100000
120000
140000
160000
Frequency

timeseries_analysis

December 13, 2024

[2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

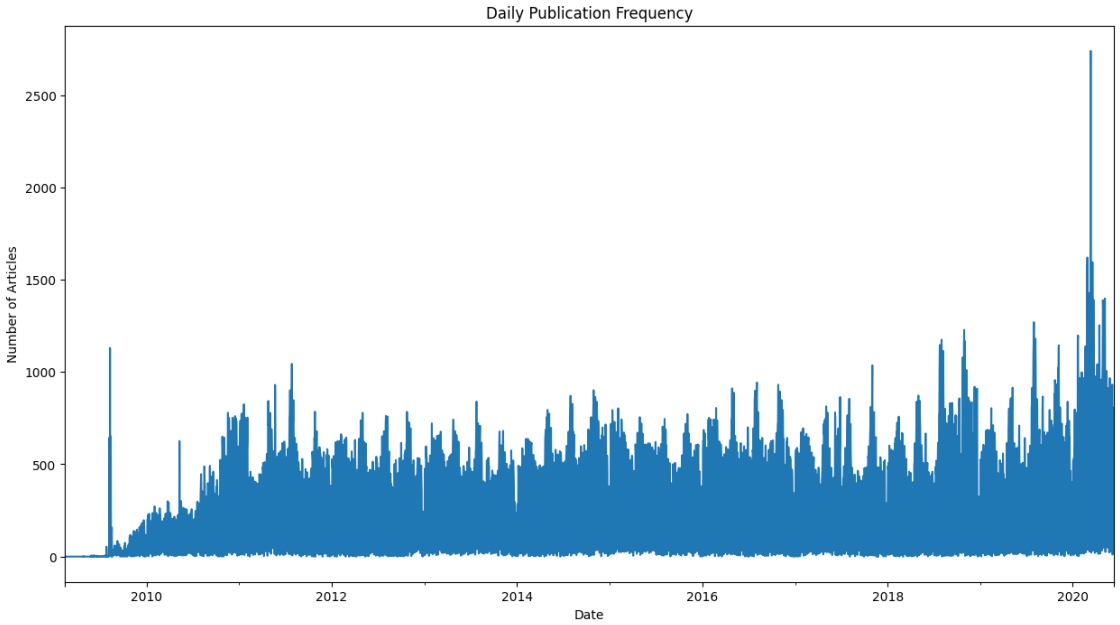
[4]:
```
# Load the CSV file
df = pd.read_csv('../Data/raw_analyst_ratings.csv')

# Display the first few rows of the dataframe
df.head()
```

[4]:
```
   Unnamed: 0                                          headline  \
0           0            Stocks That Hit 52-Week Highs On Friday
1           1         Stocks That Hit 52-Week Highs On Wednesday
2           2                       71 Biggest Movers From Friday
3           3       46 Stocks Moving In Friday's Mid-Day Session
4           4  B of A Securities Maintains Neutral on Agilent…

                                                 url          publisher  \
0  https://www.benzinga.com/news/20/06/16190091/s…  Benzinga Insights
1  https://www.benzinga.com/news/20/06/16170189/s…  Benzinga Insights
2  https://www.benzinga.com/news/20/05/16103463/7…         Lisa Levin
3  https://www.benzinga.com/news/20/05/16095921/4…         Lisa Levin
4  https://www.benzinga.com/news/20/05/16095304/b…         Vick Meyer

                        date stock
0  2020-06-05 10:30:54-04:00     A
1  2020-06-03 10:45:20-04:00     A
2  2020-05-26 04:30:07-04:00     A
3  2020-05-22 12:45:06-04:00     A
4  2020-05-22 11:38:59-04:00     A
```

[5]:
```
# Convert the date column to datetime
df['date'] = pd.to_datetime(df['date'], utc=True)

# Display the dataframe to ensure the date column is correctly converted
df.head()
```

[5]:
```
   Unnamed: 0                                          headline  \
0           0            Stocks That Hit 52-Week Highs On Friday
1           1         Stocks That Hit 52-Week Highs On Wednesday
2           2                      71 Biggest Movers From Friday
3           3       46 Stocks Moving In Friday's Mid-Day Session
4           4  B of A Securities Maintains Neutral on Agilent…

                                                 url          publisher  \
0  https://www.benzinga.com/news/20/06/16190091/s…  Benzinga Insights
1  https://www.benzinga.com/news/20/06/16170189/s…  Benzinga Insights
2  https://www.benzinga.com/news/20/05/16103463/7…         Lisa Levin
3  https://www.benzinga.com/news/20/05/16095921/4…         Lisa Levin
4  https://www.benzinga.com/news/20/05/16095304/b…         Vick Meyer

                       date stock
0 2020-06-05 14:30:54+00:00     A
1 2020-06-03 14:45:20+00:00     A
2 2020-05-26 08:30:07+00:00     A
3 2020-05-22 16:45:06+00:00     A
4 2020-05-22 15:38:59+00:00     A
```

[6]:
```python
# Set the date column as the index
df.set_index('date', inplace=True)

# Resample the data by day to get the count of articles per day
daily_publications = df['headline'].resample('D').count()

# Plot the publication frequency over time
plt.figure(figsize=(15, 8))
daily_publications.plot()
plt.title('Daily Publication Frequency')
plt.xlabel('Date')
plt.ylabel('Number of Articles')
plt.show()
```

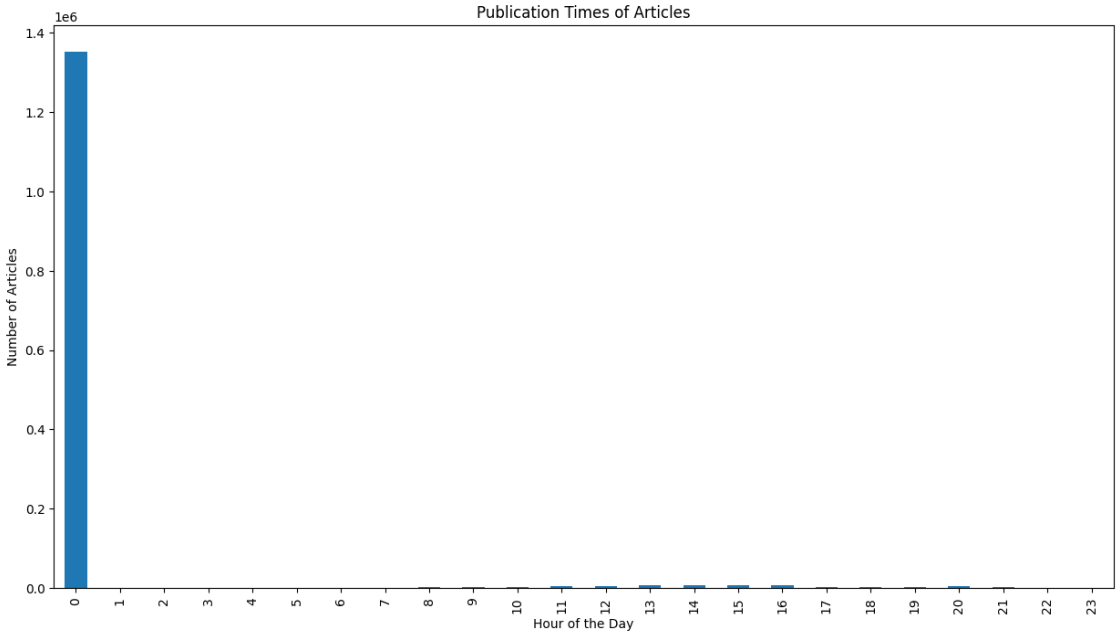


[7]:
```
# Identify dates with spikes in publication frequency
spike_threshold = daily_publications.mean() + 2 * daily_publications.std()
spike_dates = daily_publications[daily_publications > spike_threshold]

# Display the dates with spikes
spike_dates
```

[7]:
```
date
2009-08-10 00:00:00+00:00    1130
2011-05-23 00:00:00+00:00     930
2011-07-28 00:00:00+00:00    1044
2016-04-28 00:00:00+00:00     911
2016-08-04 00:00:00+00:00     943
                             ...
2020-05-08 00:00:00+00:00     927
2020-05-13 00:00:00+00:00    1005
2020-05-18 00:00:00+00:00     914
2020-05-26 00:00:00+00:00     967
2020-06-05 00:00:00+00:00     932
Name: headline, Length: 93, dtype: int64
```

[8]:
```
# Extract the hour from the datetime
df['hour'] = df.index.hour

# Count the number of articles published per hour
hourly_publications = df['hour'].value_counts().sort_index()
```

```
# Plot the publication times
plt.figure(figsize=(15, 8))
hourly_publications.plot(kind='bar')
plt.title('Publication Times of Articles')
plt.xlabel('Hour of the Day')
plt.ylabel('Number of Articles')
plt.show()
```



publisher_analysis

December 13, 2024

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import re
```

[2]:
```
# Load the CSV file
df = pd.read_csv('../Data/raw_analyst_ratings.csv')

# Display the first few rows of the dataframe
df.head()
```

[2]:
```
   Unnamed: 0                                          headline  \
0           0            Stocks That Hit 52-Week Highs On Friday
1           1         Stocks That Hit 52-Week Highs On Wednesday
2           2                       71 Biggest Movers From Friday
3           3       46 Stocks Moving In Friday's Mid-Day Session
4           4  B of A Securities Maintains Neutral on Agilent…

                                                 url          publisher  \
0  https://www.benzinga.com/news/20/06/16190091/s…  Benzinga Insights
1  https://www.benzinga.com/news/20/06/16170189/s…  Benzinga Insights
2  https://www.benzinga.com/news/20/05/16103463/7…         Lisa Levin
3  https://www.benzinga.com/news/20/05/16095921/4…         Lisa Levin
4  https://www.benzinga.com/news/20/05/16095304/b…         Vick Meyer

                       date stock
0 2020-06-05 10:30:54-04:00     A
1 2020-06-03 10:45:20-04:00     A
2 2020-05-26 04:30:07-04:00     A
3 2020-05-22 12:45:06-04:00     A
4 2020-05-22 11:38:59-04:00     A
```

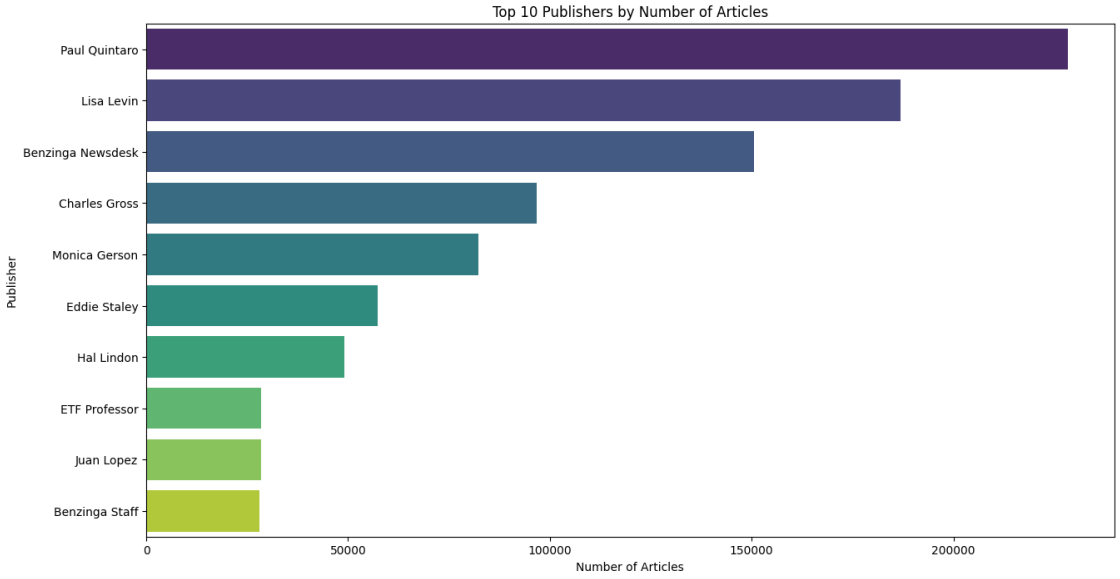
[3]:
```
# Count the number of articles per publisher
publisher_counts = df['publisher'].value_counts()

# Display the top publishers
top_publishers = publisher_counts.head(10)
```

```
top_publishers
```

[3]:
```
Paul Quintaro        228373
Lisa Levin           186979
Benzinga Newsdesk    150484
Charles Gross         96732
Monica Gerson         82380
Eddie Staley          57254
Hal Lindon            49047
ETF Professor         28489
Juan Lopez            28438
Benzinga Staff        28114
Name: publisher, dtype: int64
```

[4]:
```
# Plot the top publishers
plt.figure(figsize=(15, 8))
sns.barplot(x=top_publishers.values, y=top_publishers.index, palette='viridis')
plt.title('Top 10 Publishers by Number of Articles')
plt.xlabel('Number of Articles')
plt.ylabel('Publisher')
plt.show()
```

```
/tmp/ipykernel_97958/2296406934.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x=top_publishers.values, y=top_publishers.index,
palette='viridis')
```



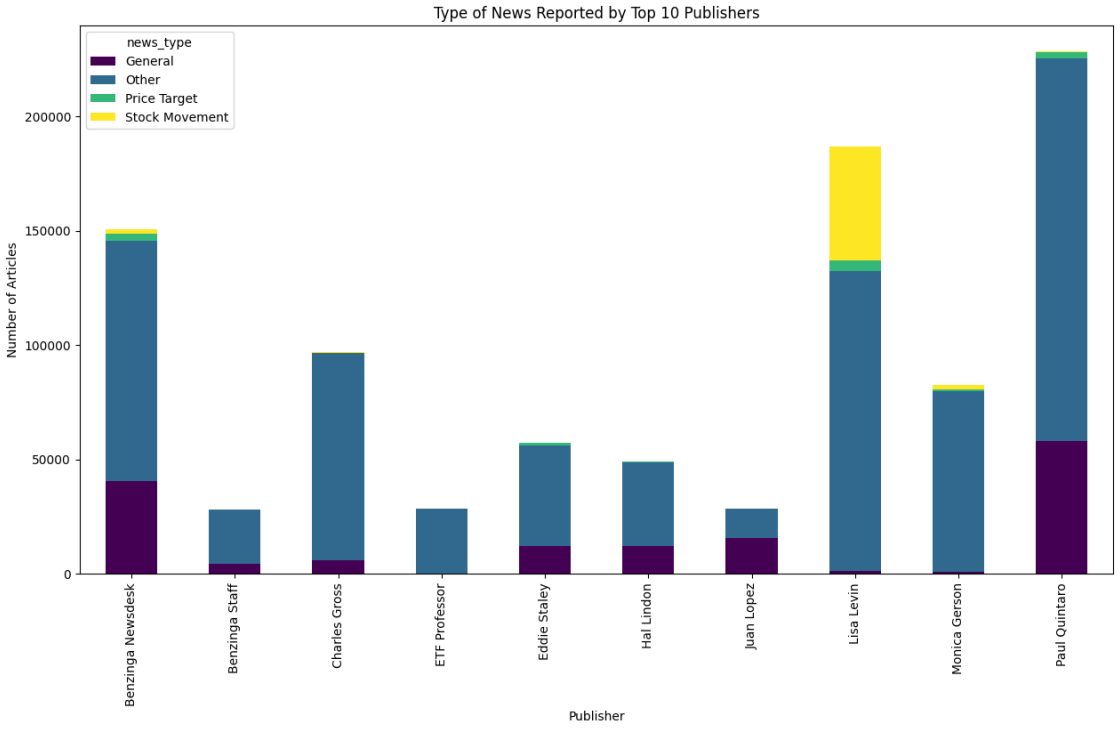
[5]:
```
# Analyze the type of news reported by top publishers
# For simplicity, we'll categorize news by keywords in headlines

# Define keywords for different types of news
news_types = {
    'Price Target': ['price target', 'raises price target', 'lowers price␣
 ↪target'],
    'Stock Movement': ['stocks moving', 'biggest movers'],
    'Earnings': ['Q2 EPS', 'Q2 earnings', 'EPS and sales results'],
    'General': ['trading higher', 'trading lower', 'reports', 'maintains']
}

# Function to categorize headlines based on keywords
def categorize_headline(headline):
    for category, keywords in news_types.items():
        if any(keyword in headline.lower() for keyword in keywords):
            return category
    return 'Other'

# Apply the function to categorize headlines
df['news_type'] = df['headline'].apply(categorize_headline)

# Analyze the type of news reported by top publishers
news_by_publisher = df[df['publisher'].isin(top_publishers.index)].
 ↪groupby(['publisher', 'news_type']).size().unstack().fillna(0)

news_by_publisher
```

[5]:

news_type	General	Other	Price Target	Stock Movement
publisher				
Benzinga Newsdesk	40296.0	105269.0	3023.0	1896.0
Benzinga Staff	4324.0	23531.0	259.0	0.0
Charles Gross	5880.0	90759.0	78.0	15.0
ETF Professor	54.0	28435.0	0.0	0.0
Eddie Staley	12046.0	43892.0	1316.0	0.0
Hal Lindon	12187.0	36410.0	439.0	11.0
Juan Lopez	15374.0	13064.0	0.0	0.0
Lisa Levin	1100.0	131122.0	4694.0	50063.0
Monica Gerson	747.0	79142.0	547.0	1944.0
Paul Quintaro	57951.0	167283.0	3029.0	110.0

[6]:
```
# Plot the type of news reported by top publishers
news_by_publisher.plot(kind='bar', stacked=True, figsize=(15, 8),␣
 ↪colormap='viridis')
```

```
plt.title('Type of News Reported by Top 10 Publishers')
plt.xlabel('Publisher')
plt.ylabel('Number of Articles')
plt.show()
```



[7]:
```
# Function to extract domain from email address
def extract_domain(email):
    match = re.search(r'@([\w\.-]+)', email)
    return match.group(1) if match else email

# Apply the function to extract domains
df['domain'] = df['publisher'].apply(extract_domain)

# Count the number of articles per domain
domain_counts = df['domain'].value_counts()

# Display the top domains
top_domains = domain_counts.head(10)
top_domains
```

[7]:
```
Paul Quintaro        228373
Lisa Levin           186979
Benzinga Newsdesk    150484
Charles Gross         96732
```

```
Monica Gerson          82380
Eddie Staley           57254
Hal Lindon             49047
ETF Professor          28489
Juan Lopez             28438
Benzinga Staff         28114
Name: domain, dtype: int64
```

[8]:
```
# Plot the top domains
plt.figure(figsize=(15, 8))
sns.barplot(x=top_domains.values, y=top_domains.index, palette='viridis')
plt.title('Top 10 Domains by Number of Articles')
plt.xlabel('Number of Articles')
plt.ylabel('Domain')
plt.show()
```

```
/tmp/ipykernel_97958/1133846947.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x=top_domains.values, y=top_domains.index, palette='viridis')
```