**You said:**

this is my code:

import pandas as pd

import plotly.express as px

import re

custom\_stopwords = set([

"i", "me", "my", "myself", "we", "our", "ours", "ourselves",

"you", "your", "yours", "yourself", "yourselves", "he", "him",

"his", "himself", "she", "her", "hers", "herself", "it", "its",

"itself","the", "they", "them", "their", "theirs", "themselves", "what",

"which", "who", "whom", "this", "that", "these", "those", "am", "is",

"are", "was", "were", "be", "been", "being", "have", "has", "had",

"having", "do", "does", "did", "doing", "a", "an", "the", "and", "but",

"if", "in", "or", "because", "as", "until", "while", "of", "at", "by", "for",

"with", "about", "against", "between", "into", "through", "during",

"before", "after", "above", "below", "to", "from", "up", "down", "in",

"out", "on", "off", "over", "under", "again", "further", "then", "once",

"here", "there", "when", "where", "why", "how", "all", "any", "both",

"each", "few", "more", "most", "other", "some", "such", "no", "nor", "not",

"only", "own", "same", "so", "than", "too", "very", "s", "t", "can", "will",

"just", "don", "should", "now", "al"

])

# Load CSV

df = pd.read\_csv("FASDH25-portfolio3/data/dataframes/title/title.csv") # Change this to your actual file path

df = df.dropna(subset=["title", "year", "month", "day" ])

df['date'] = pd.to\_datetime(df[['year', 'month', 'day']])

# Tokenize and clean titles

all\_words = []

for title in df["title"]:

tokens = re.findall(r'\b\w+\b', title.lower()) # Lowercase + remove punctuation

for word in tokens:

if word not in custom\_stopwords:

all\_words.append(word)

word\_series = pd.Series(all\_words)

word\_counts = word\_series.value\_counts().reset\_index()

word\_counts.columns = ["Word", "Count", "Date"]

# Optional: Show only the top N words

top\_n = 10

top\_words = word\_counts.head(top\_n)

# Plot using Plotly

fig = px.line(

top\_words,

x="date",

y="count",

title=f"Top {top\_n} 10 Most Frequent Words in Article Titles",

labels={"count": "frequency", "date": "date", "word": "word"},

color="Word",

markers=True

)

fig.show()

# Optional: Save to HTML file

fig.write\_html("Top\_Title\_Words\_LineChart.html")

i am getting this error:

Traceback (most recent call last):

File "C:\Users\ayesh\Downloads\test run barchart.py", line 40, in <module>

word\_counts.columns = ["Word", "Count", "Date"]

File "C:\Users\ayesh\AppData\Local\Programs\Python\Python313\Lib\site-packages\pandas\core\generic.py", line 6313, in \_\_setattr\_\_

return object.\_\_setattr\_\_(self, name, value)

File "properties.pyx", line 69, in pandas.\_libs.properties.AxisProperty.\_\_set\_\_

File "C:\Users\ayesh\AppData\Local\Programs\Python\Python313\Lib\site-packages\pandas\core\generic.py", line 814, in \_set\_axis

self.\_mgr.set\_axis(axis, labels)

File "C:\Users\ayesh\AppData\Local\Programs\Python\Python313\Lib\site-packages\pandas\core\internals\managers.py", line 238, in set\_axis

self.\_validate\_set\_axis(axis, new\_labels)

File "C:\Users\ayesh\AppData\Local\Programs\Python\Python313\Lib\site-packages\pandas\core\internals\base.py", line 98, in \_validate\_set\_axis

raise ValueError(

ValueError: Length mismatch: Expected axis has 2 elements, new values have 3 elements

The error happens here:

python

CopyEdit

word\_counts.columns = ["Word", "Count", "Date"]

because word\_counts only has **2 columns**, but you're trying to rename them to 3 columns.

**Why?**

You created word\_counts like this:

python

CopyEdit

word\_series = pd.Series(all\_words)

word\_counts = word\_series.value\_counts().reset\_index()

value\_counts() returns counts of each unique word, so after .reset\_index() you get a DataFrame with two columns:

* index (the word)
* 0 (the count)

So word\_counts has exactly 2 columns.

**How to fix it?**

Rename only two columns, like this:

python

CopyEdit

word\_counts.columns = ["Word", "Count"]

**But there’s a bigger issue: Your code loses the date information of each word!**

If you want to **plot how word frequencies change over time**, you need to keep track of the date for each word occurrence, like in the example I gave before with rows list that stores (date, word) pairs.

**So, here’s a fixed and working version of your code that keeps date info and plots the word frequency over time:**

python

CopyEdit

import pandas as pd

import plotly.express as px

import re

custom\_stopwords = set([

# your stopwords here...

])

# Load CSV and combine year/month/day into datetime

df = pd.read\_csv("FASDH25-portfolio3/data/dataframes/title/title.csv")

df = df.dropna(subset=["title", "year", "month", "day"])

df['date'] = pd.to\_datetime(df[['year', 'month', 'day']])

rows = []

# Loop through each title with date, tokenize words and filter stopwords

for date, title in zip(df['date'], df['title']):

tokens = re.findall(r'\b\w+\b', title.lower())

for word in tokens:

if word not in custom\_stopwords:

rows.append((date, word))

words\_df = pd.DataFrame(rows, columns=['date', 'word'])

# Count word occurrences by date

word\_counts\_per\_date = words\_df.groupby(['date', 'word']).size().reset\_index(name='count')

# Find top 10 most frequent words overall

top\_words = word\_counts\_per\_date.groupby('word')['count'].sum().sort\_values(ascending=False).head(10).index

# Filter counts to only top words

top\_word\_counts = word\_counts\_per\_date[word\_counts\_per\_date['word'].isin(top\_words)]

# Plot line graph of word frequency over time

fig = px.line(

top\_word\_counts,

x='date',

y='count',

color='word',

title="Top 10 Most Frequent Words in Article Titles Over Time",

labels={'count': 'Frequency', 'date': 'Date', 'word': 'Word'},

markers=True

)

fig.show()

**Summary:**

* Fix your column renaming by matching column count, **or better**
* Use the approach above that keeps date for every word occurrence, so you can track word frequency changes over time properly.

If you want, I can explain any part further!

**You said:**

what does zip do

**ChatGPT said:**

Great question!

zip is a built-in Python function that **pairs elements from two or more lists (or other iterables) together**, step by step, into tuples.

**How zip works:**

Suppose you have two lists:

python

CopyEdit

dates = ['2023-01-01', '2023-01-02', '2023-01-03']

titles = ['First article', 'Second article', 'Third article']

If you do:

python

CopyEdit

for d, t in zip(dates, titles):

print(d, t)

The output will be:

css

CopyEdit

2023-01-01 First article

2023-01-02 Second article

2023-01-03 Third article

It pairs the first element of dates with the first element of titles, then the second with the second, and so on — until one of the lists runs out of items.

**Why use it in your code?**

You want to go through your dataframe’s date and title columns **together** at the same time — so for each article, you know both the date it was published and the title text.

This way, when you extract words from the title, you can also attach the exact date to each word, which is necessary to track word usage over time.

**In short:**

* zip(df['date'], df['title']) creates pairs like (date, title) for every row in the dataframe.
* You can then loop through these pairs easily, like:

python

CopyEdit

for date, title in zip(df['date'], df['title']):

# process each title with its corresponding date

If you want, I can show you more examples or explain how zip can work with more than two lists!

**You said:**

what does .head(10) do

**ChatGPT said:**

Excellent question!

.head(10) is a **Pandas method** that returns the **first 10 rows** of a DataFrame or Series.

**Example:**

If you have a DataFrame like this:

python

CopyEdit

import pandas as pd

data = pd.Series([5, 3, 8, 6, 2, 9, 1, 7, 4, 0, 11, 13])

print(data.head(5))

The output will be:

go

CopyEdit

0 5

1 3

2 8

3 6

4 2

dtype: int64

Only the **first 5** values are shown — because of .head(5).

**In your code:**

python

CopyEdit

top\_words = word\_counts\_per\_date.groupby('word')['count'].sum().sort\_values(ascending=False).head(10).index

This means:

1. You're **grouping by word**, summing their counts across all dates.
2. Then you're **sorting** the words from most to least frequent.
3. Then .head(10) grabs the **top 10 most frequent words**.

So in simple terms:

👉 .head(10) gives you the "Top 10" of whatever you're working with.

Let me know if you'd like to see .tail() too — it does the opposite!

**You said:**

how do i make the graph less squished

**ChatGPT said:**

Great question — if your **line graph looks too squished**, it's usually because of one (or more) of these:

**✅ SOLUTIONS:**

**1. Resize the graph**

Use the width and height arguments in plotly.express.line():

python

CopyEdit

fig = px.line(

top\_word\_counts,

x='date',

y='count',

color='word',

title="Top 10 Most Frequent Words in Article Titles Over Time",

labels={'count': 'Frequency', 'date': 'Date', 'word': 'Word'},

markers=True,

width=1000, # wider

height=600 # taller

)

**2. Show fewer x-axis labels**

Too many dates can clutter the x-axis. Add:

python

CopyEdit

fig.update\_xaxes(tickangle=45, nticks=20) # Rotate ticks and reduce number

Or auto-fit:

python

CopyEdit

fig.update\_layout(xaxis=dict(tickformat="%b %Y", tickangle=45))

**3. Filter your data (Optional)**

You can limit the date range (e.g., only recent data):

python

CopyEdit

filtered = top\_word\_counts[top\_word\_counts['date'] >= '2024-01-01']

Then plot filtered instead of top\_word\_counts.

**🧠 Tip:**

You can **save the chart to HTML** with fig.write\_html("filename.html") and open it full-screen in your browser to make it even more readable.

Would you like me to update your full code to include these adjustments?

**You said:**

how do i improve the x axis for readability

**ChatGPT said:**

Improving **x-axis readability** in a Plotly line chart is all about avoiding overcrowding and making the labels easier to read.

Here are **practical ways** you can improve your x-axis in your graph:

**✅ 1. Rotate the x-axis labels**

python

CopyEdit

fig.update\_xaxes(tickangle=45)

This tilts the date labels to 45 degrees so they don’t overlap.

**✅ 2. Reduce the number of ticks**

python

CopyEdit

fig.update\_xaxes(nticks=10)

This will space the dates out more.

**✅ 3. Use a cleaner date format**

python

CopyEdit

fig.update\_xaxes(tickformat="%b %Y") # e.g., Jan 2024, Feb 2024

Other common formats:

* "%Y-%m-%d" → 2024-06-05
* "%b %d" → Jun 05
* "%b %Y" → Jun 2024

**✅ 4. Add more space below labels (if they overlap)**

python

CopyEdit

fig.update\_layout(margin=dict(b=100)) # adds space below the plot

**✅ 5. Make the graph wider**

python

CopyEdit

fig.update\_layout(width=1000)

**👇 Example (all combined):**

python

CopyEdit

fig.update\_layout(

width=1000,

height=600,

margin=dict(b=100),

)

fig.update\_xaxes(

tickangle=45,

tickformat="%b %Y",

nticks=12

)

This gives you a **wider chart**, **angled labels**, **cleaner date formatting**, and **less clutter**.

Let me know if you want help adding this directly into your code!

**You said:**

how do i modify this code to give me the most used words in article titles after october 7th

2023: import pandas as pd import plotly.express as px import re #add list of stopwords custom\_stopwords = set([ "i", "me", "my", "myself", "we", "our", "ours", "ourselves", "you", "your", "yours", "yourself", "yourselves", "he", "him", "his", "himself", "she", "her", "hers", "herself", "it", "its", "itself","the", "they", "them", "their", "theirs", "themselves", "what", "which", "who", "whom", "this", "that", "these", "those", "am", "is", "are", "was", "were", "be", "been", "being", "have", "has", "had", "having", "do", "does", "did", "day", "doing", "a", "an", "the", "and", "but", "if", "in", "or", "because", "as", "until", "while", "of", "at", "by", "for", "with", "about", "against", "between", "into", "through", "during", "before", "after", "above", "below", "to", "from", "up", "down", "in", "out", "on", "off", "over", "under", "again", "further", "then", "once", "here", "there", "when", "where", "why", "how", "all", "any", "both", "each", "few", "more", "most", "other", "some", "such", "no", "nor", "not", "only", "own", "same", "so", "than", "too", "very", "s", "t", "can", "will", "just", "don", "should", "now", "al" ]) # Load CSV and combine year/month/day into one column df = pd.read\_csv("FASDH25-portfolio3/data/dataframes/title/title.csv") df = df.dropna(subset=["title", "year", "month", "day"]) df['date'] = pd.to\_datetime(df[['year', 'month', 'day']]) rows = [] # Loop through each title with date, tokenize words and filter stopwords for date, title in zip(df['date'], df['title']): tokens = re.findall(r'\b\w+\b', title.lower()) for word in tokens: if word not in custom\_stopwords: rows.append((date, word)) words\_df = pd.DataFrame(rows, columns=['date', 'word']) # Count word occurrences by date word\_counts\_per\_date = words\_df.groupby(['date', 'word']).size().reset\_index(name='count') # Find top 10 most frequent words overall top\_words = word\_counts\_per\_date.groupby('word')['count'].sum().sort\_values(ascending=False).head(10).index # Filter counts to only top words top\_word\_counts = word\_counts\_per\_date[word\_counts\_per\_date['word'].isin(top\_words)] # Plot line graph of word frequency over time fig = px.line( top\_word\_counts, x='date', y='count', color='word', title="Top 10 Most Frequent Words in Article Titles Over Time", labels={'count': 'Frequency', 'date': 'Date', 'word': 'Word'}, markers=True ) # Improve x-axis labels for readability fig.update\_layout(xaxis\_tickangle=45) fig.show()

ChatGPT said:

To modify your code so it only analyzes article titles after October 7th, 2023, you just need to filter the DataFrame df before processing the titles.

Here's the modified version of your code with that filter added:

python

CopyEdit

import pandas as pd

import plotly.express as px

import re

# Add list of stopwords

custom\_stopwords = set([

"i", "me", "my", "myself", "we", "our", "ours", "ourselves",

"you", "your", "yours", "yourself", "yourselves", "he", "him",

"his", "himself", "she", "her", "hers", "herself", "it", "its",

"itself","the", "they", "them", "their", "theirs", "themselves", "what",

"which", "who", "whom", "this", "that", "these", "those", "am", "is",

"are", "was", "were", "be", "been", "being", "have", "has", "had",

"having", "do", "does", "did", "day", "doing", "a", "an", "the", "and", "but",

"if", "in", "or", "because", "as", "until", "while", "of", "at", "by", "for",

"with", "about", "against", "between", "into", "through", "during",

"before", "after", "above", "below", "to", "from", "up", "down", "in",

"out", "on", "off", "over", "under", "again", "further", "then", "once",

"here", "there", "when", "where", "why", "how", "all", "any", "both",

"each", "few", "more", "most", "other", "some", "such", "no", "nor", "not",

"only", "own", "same", "so", "than", "too", "very", "s", "t", "can", "will",

"just", "don", "should", "now", "al"

])

# Load CSV and combine year/month/day into one column

df = pd.read\_csv("FASDH25-portfolio3/data/dataframes/title/title.csv")

df = df.dropna(subset=["title", "year", "month", "day"])

df['date'] = pd.to\_datetime(df[['year', 'month', 'day']])

# Filter for dates after October 7, 2023

df = df[df['date'] > pd.Timestamp("2023-10-07")]

rows = []

# Loop through each title with date, tokenize words and filter stopwords

for date, title in zip(df['date'], df['title']):

tokens = re.findall(r'\b\w+\b', title.lower())

for word in tokens:

if word not in custom\_stopwords:

rows.append((date, word))

words\_df = pd.DataFrame(rows, columns=['date', 'word'])

# Count word occurrences by date

word\_counts\_per\_date = words\_df.groupby(['date', 'word']).size().reset\_index(name='count')

# Find top 10 most frequent words overall

top\_words = word\_counts\_per\_date.groupby('word')['count'].sum().sort\_values(ascending=False).head(10).index

# Filter counts to only top words

top\_word\_counts = word\_counts\_per\_date[word\_counts\_per\_date['word'].isin(top\_words)]

# Plot line graph of word frequency over time

fig = px.line(

top\_word\_counts,

x='date',

y='count',

color='word',

title="Top 10 Most Frequent Words in Article Titles After October 7, 2023",

labels={'count': 'Frequency', 'date': 'Date', 'word': 'Word'},

markers=True

)

# Improve x-axis labels for readability

fig.update\_layout(xaxis\_tickangle=45)

fig.show()

Key Change:

python

CopyEdit

df = df[df['date'] > pd.Timestamp("2023-10-07")]

This line filters the dataset to include only titles dated after October 7, 2023.

Let me know if you'd like to also include a word cloud or frequency table!

**You said:**

how do i make the frequency show on the bars of bar charts

**ChatGPT said:**

To show the **frequency values (counts)** on top of the bars in a Plotly bar chart, you already have the correct setup if you use the text='count' argument in px.bar. To make sure the text appears clearly on the bars, add text\_auto=True or explicitly set textposition='outside' or 'auto'.

Here's the improved version of your bar chart code with frequency labels:

python

Copy code

# Bar chart: Distribution of title lengths

title\_length\_counts = df['title\_length'].value\_counts().sort\_index().reset\_index()

title\_length\_counts.columns = ['title\_length', 'count']

fig2 = px.bar(

title\_length\_counts,

x='title\_length',

y='count',

title='Title Length Distribution',

labels={

'title\_length': 'Number of Words in Article Titles',

'count': 'Number of Titles'

},

text='count' # This displays frequency on bars

)

# Ensure the text shows properly

fig2.update\_traces(textposition='outside')

fig2.show()

✅ Now the frequency (count) will appear **above** each bar.

Let me know if you want to place the text **inside** the bars or change the font style!

**You said:**

how do i merge palestine and palestinians in my code so they don’t visualise as separate words

**ChatGPT said:**

Great question — if you want to treat **"Palestine"** and **"Palestinians"** as the same word in your analysis (so they appear as a single term in your visualisation), you can modify your preprocessing step to **normalize related words** into one.

**✅ How to merge "Palestine" and "Palestinians"**

You can do this inside the loop where you're appending (date, word) to rows. Just check for both and replace them with a common label, like "palestine".

**🔧 Modified part of your code:**

Replace this section:

python

CopyEdit

for word in tokens:

if word not in custom\_stopwords:

rows.append((date, word))

With this:

python

CopyEdit

for word in tokens:

if word in ["palestine", "palestinian", "palestinians"]:

word = "palestine" # normalize to one form

if word not in custom\_stopwords:

rows.append((date, word))

**🧠 What This Does:**

* It checks if a word is "palestine", "palestinian", or "palestinians".
* If it is, it **replaces it with "palestine"**, so they all count as one word in your frequency count and charts.
* You can use this same method for other variants too (like "israel" and "israeli" if needed).

Let me know if you want to apply this normalization to other word pairs or even automate it with a mapping dictionary!

Top of Form

Bottom of Form