In [722…

```
#Rather long list, apologies.
import nltk
nltk.download('punkt')
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
from nltk import stem
stemmer = stem.PorterStemmer()
from nltk import word_tokenize
nltk.download('stopwords')
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
import string
punct = list(string.punctuation)
punctuations = string.punctuation
from collections import Counter
import requests
import pandas as pd
import seaborn as sns
sns.set()
import matplotlib.pyplot as plt
!pip install PRAW
import numpy as np
import praw
import datetime
import time
import os
import plotly
import plotly.express as px
!pip install jupyterlab "ipywidgets>=7.5"
from nltk.corpus import wordnet as wn
import plotly.graph_objects as go
import csv
vad = pd.read_excel('/Users/louisvsbigmac/Documents/Office 365/xlsx/vad.xlsx', index_col = 0)
import gensim
import gensim.downloader as api
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import PCA
from sklearn.cluster import AgglomerativeClustering
from IPython.display import IFrame
import plotly.express as px
from sklearn.cluster import KMeans
from sklearn.cluster import AffinityPropagation
from scipy.spatial import distance
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.cm as cm
import spacy
from IPython.display import FileLink
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     /Users/louisvsbigmac/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/louisvsbigmac/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
Requirement already satisfied: PRAW in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (7.7.0)
Requirement already satisfied: prawcore<3,>=2.1 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (fro
m PRAW) (2.3.0)
Requirement already satisfied: websocket-client>=0.54.0 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packa
ges (from PRAW) (0.58.0)
Requirement already satisfied: update-checker>=0.18 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages
(from PRAW) (0.18.0)
Requirement already satisfied: requests<3.0,>=2.6.0 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages
(from prawcore<3,>=2.1->PRAW) (2.28.1)
Requirement already satisfied: six in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (from websocket-c
lient>=0.54.0->PRAW) (1.16.0)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages
(from requests<3.0,>=2.6.0->prawcore<3,>=2.1->PRAW) (1.26.11)
Requirement already satisfied: certifi>=2017.4.17 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (f
rom requests<3.0,>=2.6.0->prawcore<3,>=2.1->PRAW) (2022.9.24)
Requirement already satisfied: idna<4,>=2.5 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (from re
quests<3.0,>=2.6.0->prawcore<3,>=2.1->PRAW) (3.3)
Requirement already satisfied: charset-normalizer<3,>=2 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packa
ges (from requests<3.0,>=2.6.0->prawcore<3,>=2.1->PRAW) (2.0.4)
Requirement already satisfied: jupyterlab in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (3.4.4)
Requirement already satisfied: ipywidgets>=7.5 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (7.6.
5)
Requirement already satisfied: packaging in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (from jupyt
erlab) (21.3)
Requirement already satisfied: ipython in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (from jupyter
lab) (7.31.1)
Requirement already satisfied: jupyter-core in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (from ju
pyterlab) (4.11.1)
Requirement already satisfied: notebook<7 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (from jupy
terlab) (6.4.12)
Requirement already satisfied: tornado>=6.1.0 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (from
jupyterlab) (6.1)
Requirement already satisfied: jupyterlab-server~=2.10 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packag
es (from jupyterlab) (2.10.3)
Requirement already satisfied: nbclassic in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (from jupyt
erlab) (0.3.5)
Requirement already satisfied: jupyter-server~=1.16 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages
(from jupyterlab) (1.18.1)
Requirement already satisfied: jinja2>=2.1 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (from jup
yterlab) (2.11.3)
Requirement already satisfied: nbformat>=4.2.0 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (from
ipywidgets>=7.5) (5.5.0)
Requirement already satisfied: traitlets>=4.3.1 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (fro
m ipywidgets>=7.5) (5.1.1)
```

```
Requirement already satisfied: jupyterlab-widgets>=1.0.0 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-pack
ages (from ipywidgets>=7.5) (1.0.0)
Requirement already satisfied: ipykernel>=4.5.1 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (fro
m ipywidgets>=7.5) (6.15.2)
Requirement already satisfied: ipython-genutils~=0.2.0 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packag
es (from ipywidgets>=7.5) (0.2.0)
Requirement already satisfied: widgetsnbextension~=3.5.0 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-pack
ages (from ipywidgets>=7.5) (3.5.2)
Requirement already satisfied: jupyter-client>=6.1.12 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-package
s (from ipykernel>=4.5.1->ipywidgets>=7.5) (7.3.4)
Requirement already satisfied: appnope in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (from ipykern
el>=4.5.1->ipywidgets>=7.5) (0.1.2)
Requirement already satisfied: pyzmq>=17 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (from ipyke
rnel>=4.5.1->ipywidgets>=7.5) (23.2.0)
Requirement already satisfied: psutil in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (from ipykerne
l>=4.5.1->ipywidgets>=7.5) (5.9.0)
Requirement already satisfied: debugpy>=1.0 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (from ip
ykernel>=4.5.1->ipywidgets>=7.5) (1.5.1)
Requirement already satisfied: matplotlib-inline>=0.1 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-package
s (from ipykernel>=4.5.1->ipywidgets>=7.5) (0.1.6)
Requirement already satisfied: nest-asyncio in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (from ip
ykernel>=4.5.1->ipywidgets>=7.5) (1.5.5)
Requirement already satisfied: setuptools>=18.5 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (fro
m ipython->jupyterlab) (63.4.1)
Requirement already satisfied: decorator in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (from ipyth
on->jupyterlab) (5.1.1)
Requirement already satisfied: pexpect>4.3 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (from ipy
thon->jupyterlab) (4.8.0)
Requirement already satisfied: pickleshare in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (from ipy
thon->jupyterlab) (0.7.5)
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0 in /Users/louisvsbigmac/opt/anaconda3/lib/
python3.9/site-packages (from ipython->jupyterlab) (3.0.20)
Requirement already satisfied: pygments in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (from ipytho
n->jupyterlab) (2.11.2)
Requirement already satisfied: backcall in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (from ipytho
n->jupyterlab) (0.2.0)
Requirement already satisfied: jedi>=0.16 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (from ipyt
hon->jupyterlab) (0.18.1)
Requirement already satisfied: MarkupSafe>=0.23 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (fro
m jinja2>=2.1->jupyterlab) (2.0.1)
Requirement already satisfied: Send2Trash in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (from jupy
ter-server~=1.16->jupyterlab) (1.8.0)
Requirement already satisfied: websocket-client in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (fro
m jupyter-server~=1.16->jupyterlab) (0.58.0)
```

```
Requirement already satisfied: anyio<4,>=3.1.0 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (from
jupyter-server~=1.16->jupyterlab) (3.5.0)
Requirement already satisfied: prometheus-client in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (fr
om jupyter-server~=1.16->jupyterlab) (0.14.1)
Requirement already satisfied: terminado>=0.8.3 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (fro
m jupyter-server~=1.16->jupyterlab) (0.13.1)
Requirement already satisfied: nbconvert>=6.4.4 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (fro
m jupyter-server~=1.16->jupyterlab) (6.4.4)
Requirement already satisfied: argon2-cffi in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (from jup
yter-server~=1.16->jupyterlab) (21.3.0)
Requirement already satisfied: babel in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (from jupyterla
b-server~=2.10->jupyterlab) (2.9.1)
Requirement already satisfied: entrypoints>=0.2.2 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (f
rom jupyterlab-server~=2.10->jupyterlab) (0.4)
Requirement already satisfied: json5 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (from jupyterla
b-server~=2.10->jupyterlab) (0.9.6)
Requirement already satisfied: requests in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (from jupyte
rlab-server~=2.10->jupyterlab) (2.28.1)
Requirement already satisfied: jsonschema>=3.0.1 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (fr
om jupyterlab-server~=2.10->jupyterlab) (4.16.0)
Requirement already satisfied: fastjsonschema in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (from
nbformat>=4.2.0->ipywidgets>=7.5) (2.16.2)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packa
ges (from packaging->jupyterlab) (3.0.9)
Requirement already satisfied: idna>=2.8 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (from anyio
<4,>=3.1.0->jupyter-server~=1.16->jupyterlab) (3.3)
Requirement already satisfied: sniffio>=1.1 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (from an
yio<4,>=3.1.0->jupyter-server~=1.16->jupyterlab) (1.2.0)
Requirement already satisfied: parso<0.9.0,>=0.8.0 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages
(from jedi>=0.16->ipython->jupyterlab) (0.8.3)
Requirement already satisfied: attrs>=17.4.0 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (from j
sonschema>=3.0.1->jupyterlab-server~=2.10->jupyterlab) (21.4.0)
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in /Users/louisvsbigmac/opt/anaconda3/li
b/python3.9/site-packages (from jsonschema>=3.0.1->jupyterlab-server~=2.10->jupyterlab) (0.18.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-package
s (from jupyter-client>=6.1.12->ipykernel>=4.5.1->ipywidgets>=7.5) (2.8.2)
Requirement already satisfied: nbclient<0.6.0,>=0.5.0 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-package
s (from nbconvert>=6.4.4->jupyter-server~=1.16->jupyterlab) (0.5.13)
Requirement already satisfied: jupyterlab-pygments in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages
(from nbconvert>=6.4.4->jupyter-server~=1.16->jupyterlab) (0.1.2)
Requirement already satisfied: defusedxml in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (from nbco
nvert>=6.4.4->jupyter-server~=1.16->jupyterlab) (0.7.1)
Requirement already satisfied: beautifulsoup4 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (from
nbconvert>=6.4.4->jupyter-server~=1.16->jupyterlab) (4.11.1)
```

```
Requirement already satisfied: mistune<2,>=0.8.1 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (fr
om nbconvert>=6.4.4->jupyter-server~=1.16->jupyterlab) (0.8.4)
Requirement already satisfied: bleach in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (from nbconver
t>=6.4.4->jupyter-server~=1.16->jupyterlab) (4.1.0)
Requirement already satisfied: testpath in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (from nbconv
ert>=6.4.4->jupyter-server~=1.16->jupyterlab) (0.6.0)
Requirement already satisfied: pandocfilters>=1.4.1 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages
(from nbconvert>=6.4.4->jupyter-server~=1.16->jupyterlab) (1.5.0)
Requirement already satisfied: ptyprocess>=0.5 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (from
pexpect>4.3->ipython->jupyterlab) (0.7.0)
Requirement already satisfied: wcwidth in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (from prompt-
toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0->ipython->jupyterlab) (0.2.5)
Requirement already satisfied: argon2-cffi-bindings in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages
(from argon2-cffi->jupyter-server~=1.16->jupyterlab) (21.2.0)
Requirement already satisfied: pytz>=2015.7 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (from ba
bel->jupyterlab-server~=2.10->jupyterlab) (2022.1)
Requirement already satisfied: charset-normalizer<3,>=2 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packa
ges (from requests->jupyterlab-server~=2.10->jupyterlab) (2.0.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages
(from requests->jupyterlab-server~=2.10->jupyterlab) (1.26.11)
Requirement already satisfied: certifi>=2017.4.17 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (f
rom requests->jupyterlab-server~=2.10->jupyterlab) (2022.9.24)
Requirement already satisfied: six in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (from websocket-c
lient->jupyter-server~=1.16->jupyterlab) (1.16.0)
Requirement already satisfied: cffi>=1.0.1 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (from arg
on2-cffi-bindings->argon2-cffi->jupyter-server~=1.16->jupyterlab) (1.15.1)
Requirement already satisfied: soupsieve>1.2 in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (from b
eautifulsoup4->nbconvert>=6.4.4->jupyter-server~=1.16->jupyterlab) (2.3.1)
Requirement already satisfied: webencodings in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (from bl
each->nbconvert>=6.4.4->jupyter-server~=1.16->jupyterlab) (0.5.1)
Requirement already satisfied: pycparser in /Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages (from cffi>
=1.0.1->argon2-cffi-bindings->argon2-cffi->jupyter-server~=1.16->jupyterlab) (2.21)
```

In [723…

```
#NOTE:
#My YouTube API Key is confidential.
```

In [724…

```
#KEY:

#Right Wing UK News Outlets:
#DM = The Daily Mail
#DE = The Daily Express

#Centrist UK News Outlets:
```

```
#BBC = Exactly that
#FT = The Financial Times

#Left Wing UK News Outlets:
#TG = The Guardian
#TM = The Mirror
```

In [725…

```
#This enquiry analyses YouTube titles published by UK news agencies.
#The publishers were chosen for their positioning on the political spectrum (Oxford Royale Academy, 2022)).
#Content on YouTube has a tremendous influence upon influencing public opinion, and its content has been used to under
```

In [726…

```
#Polical compass of UK news outlets sourced from https://www.oxford-royale.com/articles/a-guide-to-british-newspapers/
```

In [727…

```
#Admin for YouTube API
API_KEY = "AIzaSyANsLb9I3d9sMCNq9Vy2R9dPKWqcuNmjc8"

#Right Wing
DM_CHANNEL_ID = "UCrFXs-7T3dCTv1aYmdArzFQ"
DE_CHANNEL_ID = "UCz8omIXCPabFEgblrC_VIfA"

#Centrist
BBC_CHANNEL_ID = "UC16niRr50-MSBwiO3YDb3RA"
FT_CHANNEL_ID = "UCoUxsWakJucWg46KW5RsvPw"

#Left Wing
TG_CHANNEL_ID = "UCIRYBXDze5krPDzAEOxFGVA"
TM_CHANNEL_ID = "UC3EmxrWVl7K8xBH_UVIWY4Q"
```

In [728…

```
#Data was garnered through YouTube’s API.
#My developer account limits me to 50 datapoints per account request.
#I narrowed my enquiry to the publishers’ most recent videos; this to get a better understanding of each publisher thr
#Once data was sourced from the API, errors present within the datasets (repeated, useless phrases such as #shorts) we
```

In [729…

```
#SECTION 1: RIGHT WING ANALYSIS.
```

In [730…

```
#DM
```

In [731…

```
#Getting/parsing Daily Mail's YouTube stats. Sorted by descending view count.
DM_url = "https://www.googleapis.com/youtube/v3/search?key="+API_KEY+"&channelId="+DM_CHANNEL_ID+"&part=snippet,id&ord
DM_response = requests.get(DM_url).json()
```

In [732…

```
#Obtaining Daily Mail's YouTube titles, joined together into long list.
DM_title_list = []
for video in DM_response ['items']:
    if video ['id']['kind'] == "youtube#video":
        DM_titles = video['snippet']['title']
        DM_titles = str(DM_titles).replace("&#39;"," ")
        DM_titles = str(DM_titles).replace("|"," ")
        DM_titles = DM_titles.encode('ascii', 'ignore')
        DM_titles = DM_titles.decode()
        DM_title_list.append(DM_titles)
```

In [733…

```
#Translating list of titles into string
DM_title_long_string = " ".join(DM_title_list)

#Cleaning up
DM_title_string_complete = DM_title_long_string.replace('Jungle Confidential', '')

#Record amount of punctuation in text
DM_num_punctuation = len([token for token in DM_title_string_complete if token in punctuations])

#Record amount of UPPER CASE TEXT
DM_num_uppercase = len([word for word in DM_title_string_complete if word.isupper()])
DM_PROP_UPPER = DM_num_uppercase / len(DM_title_string_complete)

#Tokenise words
DM_tokenised_words = word_tokenize(DM_title_string_complete)

#lowercase
DM_lc_tokens = [i.lower() for i in DM_tokenised_words]

#Remove punctuation
DM_lc_filtered_tokens = [word for word in DM_lc_tokens if word not in punctuations]

#Remove stop words
DM_lc_fully_filtered_tokens = [word for word in DM_lc_filtered_tokens if not word in stop_words]

#Lemmatize into tokens
DM_lemmatized_tokens = []
for token in DM_lc_fully_filtered_tokens:
    DM_lemmatized_tokens.append(lemmatizer.lemmatize(token))
```

In [734…

```
#Word counts
DM_word_counts = pd.Series(Counter(DM_lemmatized_tokens))
DM_word_counts.sort_values()

#DF of counts
DM_df = pd.DataFrame()
DM_df['DM_Lemma_Counts'] = DM_word_counts
```

In [735…

```
#Gather VAD data
DM_words = []
DM_emo = []

for i in DM_lemmatized_tokens:
    if i in vad.index:
        DM_emo.append(vad.loc[i])
        DM_words.append(i)
    else:
        pass

#DF charting VAD scores
DM_emo_df = pd.DataFrame(DM_emo, index = DM_words)
```

In [736…

```
#DE
```

In [737…

```
#Getting/parsing Daily Express's YouTube stats. Sorted by descending view count.
DE_url = "https://www.googleapis.com/youtube/v3/search?key="+API_KEY+"&channelId="+DE_CHANNEL_ID+"&part=snippet,id&ord
DE_response = requests.get(DE_url).json()
```

In [738…

```
#Obtaining The Daily Express's YouTube titles, joined together into long list.
DE_title_list = []
for video in DE_response ['items']:
    if video ['id']['kind'] == "youtube#video":
        DE_titles = video['snippet']['title']
        DE_titles = str(DE_titles).replace("&#39;"," ")
        DE_titles = str(DE_titles).replace("|"," ")
        DE_titles = str(DE_titles).replace("#shorts"," ")
        DE_titles = DE_titles.encode('ascii', 'ignore')
        DE_titles = DE_titles.decode()
        DE_title_list.append(DE_titles)
```

In [739…

```
#Translating list of titles into string
DE_title_string_complete = " ".join(DE_title_list)

#Record amount of punctuation in text
DE_num_punctuation = len([token for token in DE_title_string_complete if token in punctuations])

#Record amount of UPPER CASE TEXT
DE_num_uppercase = len([word for word in DE_title_string_complete if word.isupper()])
DE_PROP_UPPER = DE_num_uppercase / len(DE_title_string_complete)

#Tokenise words
DE_tokenised_words = word_tokenize(DE_title_string_complete)

#lowercase
DE_lc_tokens = [i.lower() for i in DE_tokenised_words]

#Remove punctuation
DE_lc_filtered_tokens = [word for word in DE_lc_tokens if word not in punctuations]

#Remove stop words
DE_lc_fully_filtered_tokens = [word for word in DE_lc_filtered_tokens if not word in stop_words]

#Lemmatize into tokens
DE_lemmatized_tokens = []
for token in DE_lc_fully_filtered_tokens:
    DE_lemmatized_tokens.append(lemmatizer.lemmatize(token))
```

In [740…

```
#Word counts
DE_word_counts = pd.Series(Counter(DE_lemmatized_tokens))
DE_word_counts.sort_values()

#DF of counts
DE_df = pd.DataFrame()
DE_df['DE_Lemma_Counts'] = DE_word_counts
```

In [741…

```
#Gather VAD data
DE_words = []
DE_emo = []

for i in DE_lemmatized_tokens:
    if i in vad.index:
        DE_emo.append(vad.loc[i])
        DE_words.append(i)
    else:
        pass

#DF charting VAD scores
DE_emo_df = pd.DataFrame(DE_emo, index = DE_words)
```

In [742…

```
#SECTION 2: CENTRIST ANALYSIS.
```

In [743…

```
#BBC
```

In [744…

```
#Getting/parsing BBC's YouTube stats. Sorted by descending view count.
BBC_url = "https://www.googleapis.com/youtube/v3/search?key="+API_KEY+"&channelId="+BBC_CHANNEL_ID+"&part=snippet,id&c
BBC_response = requests.get(BBC_url).json()
```

In [745…

```
#Obtaining BBC's YouTube titles, joined together into long list.
BBC_title_list = []
for video in BBC_response ['items']:
    if video ['id']['kind'] == "youtube#video":
        BBC_titles = video['snippet']['title']
        BBC_titles = str(BBC_titles).replace("&#39;"," ")
        BBC_titles = str(BBC_titles).replace("|"," ")
        BBC_titles = str(BBC_titles).replace("— BBC News"," ")
        BBC_titles = str(BBC_titles).replace("- BBC News"," ")
        BBC_titles = str(BBC_titles).replace("#shorts"," ")
        BBC_titles = BBC_titles.encode('ascii', 'ignore')
        BBC_titles = BBC_titles.decode()
        BBC_title_list.append(BBC_titles)
```

In [746…

```
#Translating list of titles into string
BBC_title_string_complete = " ".join(BBC_title_list)

#Record amount of punctuation in text
BBC_num_punctuation = len([token for token in BBC_title_string_complete if token in punctuations])

#Record amount of UPPER CASE TEXT
BBC_num_uppercase = len([word for word in BBC_title_string_complete if word.isupper()])
BBC_PROP_UPPER = BBC_num_uppercase / len(BBC_title_string_complete)

#Tokenise words
BBC_tokenised_words = word_tokenize(BBC_title_string_complete)
```

```
#lowercase
BBC_lc_tokens = [i.lower() for i in BBC_tokenised_words]

#Remove punctuation
BBC_lc_filtered_tokens = [word for word in BBC_lc_tokens if word not in punctuations]

#Remove stop words
BBC_lc_fully_filtered_tokens = [word for word in BBC_lc_filtered_tokens if not word in stop_words]

#Lemmatize into tokens
BBC_lemmatized_tokens = []
for token in BBC_lc_fully_filtered_tokens:
    BBC_lemmatized_tokens.append(lemmatizer.lemmatize(token))
```

In [747…

```
#Word counts
BBC_word_counts = pd.Series(Counter(BBC_lemmatized_tokens))
BBC_word_counts.sort_values()

#DF of counts
BBC_df = pd.DataFrame()
BBC_df['BBC_Lemma_Counts'] = BBC_word_counts
```

In [748…

```
#Gather VAD data
BBC_words = []
BBC_emo = []

for i in BBC_lemmatized_tokens:
    if i in vad.index:
        BBC_emo.append(vad.loc[i])
        BBC_words.append(i)
    else:
        pass

#DF charting VAD scores
BBC_emo_df = pd.DataFrame(BBC_emo, index = BBC_words)
```

In [749…

```
#FT
```

In [750…

```
#Getting/parsing The Financial Time's YouTube stats. Sorted by descending view count.
FT_url = "https://www.googleapis.com/youtube/v3/search?key="+API_KEY+"&channelId="+FT_CHANNEL_ID+"&part=snippet,id&ord
FT_response = requests.get(FT_url).json()
```

In [751…
```
#Obtaining FT's YouTube titles, joined together into long list.
FT_title_list = []
for video in FT_response ['items']:
    if video ['id']['kind'] == "youtube#video":
        FT_titles = video['snippet']['title']
        FT_titles = str(FT_titles).replace("&#39;"," ")
        FT_titles = str(FT_titles).replace("|"," ")
        FT_titles = str(FT_titles).replace("FT"," ")
        FT_titles = str(FT_titles).replace("#shorts"," ")
        FT_titles = FT_titles.encode('ascii', 'ignore')
        FT_titles = FT_titles.decode()
        FT_title_list.append(FT_titles)
```

In [752…
```
#Translating list of titles into string
FT_title_string_complete = " ".join(FT_title_list)

#Record amount of punctuation in text
FT_num_punctuation = len([token for token in FT_title_string_complete if token in punctuations])

#Record amount of UPPER CASE TEXT
FT_num_uppercase = len([word for word in FT_title_string_complete if word.isupper()])
FT_PROP_UPPER = FT_num_uppercase / len(FT_title_string_complete)

#Tokenise words
FT_tokenised_words = word_tokenize(FT_title_string_complete)

#lowercase
FT_lc_tokens = [i.lower() for i in FT_tokenised_words]

#Remove punctuation
FT_lc_filtered_tokens = [word for word in FT_lc_tokens if word not in punctuations]

#Remove stop words
FT_lc_fully_filtered_tokens = [word for word in FT_lc_filtered_tokens if not word in stop_words]

#Lemmatize into tokens
FT_lemmatized_tokens = []
for token in FT_lc_fully_filtered_tokens:
    FT_lemmatized_tokens.append(lemmatizer.lemmatize(token))
```

In [753…
```
#Word counts
FT_word_counts = pd.Series(Counter(FT_lemmatized_tokens))
```

```
FT_word_counts.sort_values()

#DF of counts
FT_df = pd.DataFrame()
FT_df['FT_Lemma_Counts'] = FT_word_counts
```

In [754…

```
#Gather VAD data
FT_words = []
FT_emo = []

for i in FT_lemmatized_tokens:
    if i in vad.index:
        FT_emo.append(vad.loc[i])
        FT_words.append(i)
    else:
        pass

#DF charting VAD scores
FT_emo_df = pd.DataFrame(FT_emo, index = FT_words)
```

In [755…

```
#SECTION 3: LEFT WING ANALYSIS.
```

In [756…

```
#TG
```

In [757…

```
#Getting/parsing The Guardian's YouTube stats. Sorted by descending view count.
TG_url = "https://www.googleapis.com/youtube/v3/search?key="+API_KEY+"&channelId="+TG_CHANNEL_ID+"&part=snippet,id&ord
TG_response = requests.get(TG_url).json()
```

In [758…

```
#Obtaining The Guardian's YouTube titles, joined together into long list.
TG_title_list = []
for video in TG_response ['items']:
    if video ['id']['kind'] == "youtube#video":
        TG_titles = video['snippet']['title']
        TG_titles = str(TG_titles).replace("&#39;"," ")
        TG_titles = str(TG_titles).replace("|"," ")
        TG_titles = str(TG_titles).replace("#shorts"," ")
        TG_titles = TG_titles.encode('ascii', 'ignore')
        TG_titles = TG_titles.decode()
        TG_title_list.append(TG_titles)
```

In [759…

```
#Translating list of titles into string
TG_title_long_string = " ".join(TG_title_list)
```

```
#Record amount of punctuation in text
TG_num_punctuation = len([token for token in TG_title_long_string if token in punctuations])

#Record amount of UPPER CASE TEXT
TG_num_uppercase = len([word for word in TG_title_long_string if word.isupper()])
TG_PROP_UPPER = TG_num_uppercase / len(TG_title_long_string)

#Tokenise words
TG_tokenised_words = word_tokenize(TG_title_long_string)

#lowercase
TG_lc_tokens = [i.lower() for i in TG_tokenised_words]

#Remove punctuation
TG_lc_filtered_tokens = [word for word in TG_lc_tokens if word not in punctuations]

#Remove stop words
TG_lc_fully_filtered_tokens = [word for word in TG_lc_filtered_tokens if not word in stop_words]

#Lemmatize into tokens
TG_lemmatized_tokens = []
for token in TG_lc_fully_filtered_tokens:
    TG_lemmatized_tokens.append(lemmatizer.lemmatize(token))
```

In [760…

```
#Word counts
TG_word_counts = pd.Series(Counter(TG_lemmatized_tokens))
TG_word_counts.sort_values()

#DF of counts
TG_df = pd.DataFrame()
TG_df['TG_Lemma_Counts'] = TG_word_counts
```

In [761…

```
#Gather VAD data
TG_words = []
TG_emo = []

for i in TG_lemmatized_tokens:
    if i in vad.index:
        TG_emo.append(vad.loc[i])
        TG_words.append(i)
    else:
        pass
```

```
#DF charting VAD scores
TG_emo_df = pd.DataFrame(TG_emo, index = TG_words)
```

In [762…

```
#TM
```

In [763…

```
#Getting/parsing The Mirror's YouTube stats. Sorted by descending view count.
TM_url = "https://www.googleapis.com/youtube/v3/search?key="+API_KEY+"&channelId="+TM_CHANNEL_ID+"&part=snippet,id&ord
TM_response = requests.get(TM_url).json()
```

In [764…

```
#Obtaining TM's YouTube titles, joined together into long list.
TM_title_list = []
for video in TM_response ['items']:
    if video ['id']['kind'] == "youtube#video":
        TM_titles = video['snippet']['title']
        TM_titles = str(TM_titles).replace("&#39;"," ")
        TM_titles = str(TM_titles).replace("|"," ")
        TM_titles = str(TM_titles).replace("#shorts"," ")
        TM_titles = TM_titles.encode('ascii', 'ignore')
        TM_titles = TM_titles.decode()
        TM_title_list.append(TM_titles)
```

In [765…

```
#Translating list of titles into string
TM_title_long_string = " ".join(TM_title_list)

#Record amount of punctuation in text
TM_num_punctuation = len([token for token in TM_title_long_string if token in punctuations])

#Record amount of UPPER CASE TEXT
TM_num_uppercase = len([word for word in TM_title_long_string if word.isupper()])
TM_PROP_UPPER = TM_num_uppercase / len(TM_title_long_string)

#Tokenise words
TM_tokenised_words = word_tokenize(TM_title_long_string)

#lowercase
TM_lc_tokens = [i.lower() for i in TM_tokenised_words]

#Remove punctuation
TM_lc_filtered_tokens = [word for word in TM_lc_tokens if word not in punctuations]

#Remove stop words
```

```
TM_lc_fully_filtered_tokens = [word for word in TM_lc_filtered_tokens if not word in stop_words]

#Lemmatize into tokens
TM_lemmatized_tokens = []
for token in TM_lc_fully_filtered_tokens:
    TM_lemmatized_tokens.append(lemmatizer.lemmatize(token))
```

In [766…

```
#Word counts
TM_word_counts = pd.Series(Counter(TM_lemmatized_tokens))
TM_word_counts.sort_values()

#DF of counts
TM_df = pd.DataFrame()
TM_df['TM_Lemma_Counts'] = TM_word_counts
```

In [767…

```
#Gather VAD data
TM_words = []
TM_emo = []

for i in TM_lemmatized_tokens:
    if i in vad.index:
        TM_emo.append(vad.loc[i])
        TM_words.append(i)
    else:
        pass

#DF charting VAD scores
TM_emo_df = pd.DataFrame(TM_emo, index = TM_words)
```

In [768…

```
#Shifting index across on them all
DM_emo_df = DM_emo_df.reset_index().rename(columns={'index': 'word'})
DM_emo_df['Publication'] = 'The Daily Mail'
DM_emo_df['Political Compass'] = 'Right-Wing'

DE_emo_df = DE_emo_df.reset_index().rename(columns={'index': 'word'})
DE_emo_df['Publication'] = 'The Daily Express'
DE_emo_df['Political Compass'] = 'Right-Wing'

BBC_emo_df = BBC_emo_df.reset_index().rename(columns={'index': 'word'})
BBC_emo_df['Publication'] = 'The BBC'
BBC_emo_df['Political Compass'] = 'Centrist'

FT_emo_df = FT_emo_df.reset_index().rename(columns={'index': 'word'})
```

```
FT_emo_df['Publication'] = 'The Financial Times'
FT_emo_df['Political Compass'] = 'Centrist'

TG_emo_df = TG_emo_df.reset_index().rename(columns={'index': 'word'})
TG_emo_df['Publication'] = 'The Guardian'
TG_emo_df['Political Compass'] = 'Left-Wing'

TM_emo_df = TM_emo_df.reset_index().rename(columns={'index': 'word'})
TM_emo_df['Publication'] = 'The Mirror'
TM_emo_df['Political Compass'] = 'Left-Wing'
```

In [769…

```
#Finally bringing them together
Mega_VAD_Df = pd.concat([DM_emo_df, DE_emo_df, BBC_emo_df, FT_emo_df, TG_emo_df, TM_emo_df], axis=0)
Mega_VAD_Df = Mega_VAD_Df.reset_index(drop=True)
```

In [770…

```
#Creating mega dataframe logging titles, political compass, and publisher.
#t was just a stand in for word temporary
tdf = pd.DataFrame()
tdf['Titles'] = DM_title_list
tdf.loc[:, "Publisher"] = "The Daily Mail"

ttdf = pd.DataFrame()
ttdf['Titles'] = DE_title_list
ttdf.loc[:, "Publisher"] = "The Daily Express"

Tdf = pd.concat([tdf, ttdf], axis=0)
Tdf.loc[:, "Political Compass"] = "Right-Wing"

tttdf = pd.DataFrame()
tttdf['Titles'] = BBC_title_list
tttdf.loc[:, "Publisher"] = "The BBC"

ttttdf = pd.DataFrame()
ttttdf['Titles'] = FT_title_list
ttttdf.loc[:, "Publisher"] = "The Financial Times"

TTdf = pd.concat([tttdf, ttttdf], axis=0)
TTdf.loc[:, "Political Compass"] = "Centrist"

tttttdf = pd.DataFrame()
tttttdf['Titles'] = TG_title_list
tttttdf.loc[:, "Publisher"] = "The Guardian"
```

```
tttttdf = pd.DataFrame()
tttttdf['Titles'] = TM_title_list
tttttdf.loc[:, "Publisher"] = "The Mirror"

TTTdf = pd.concat([ttttdf, tttttdf], axis=0)
TTTdf.loc[:, "Political Compass"] = "Left-Wing"
```

In [771…

```
#Finally bringing them together
Mega_Df = pd.concat([Tdf, TTdf, TTTdf], axis=0)
Combined_Titles_df = Mega_Df.reset_index(drop=True)
```

In [772…

```
#Handy combo list
Combined_Title_List = []
Combined_Title_List.extend(DM_title_list)
Combined_Title_List.extend(DE_title_list)
Combined_Title_List.extend(BBC_title_list)
Combined_Title_List.extend(FT_title_list)
Combined_Title_List.extend(TG_title_list)
Combined_Title_List.extend(TM_title_list)
```

In [773…

```
#SECTION 4: ANALYSIS OF VAD VALUES ACROSS PUBLICATIONS.
```

In [774…

```
#Calculating means of VAD scores
DM_avg_v = DM_emo_df['valence'].mean()
DM_avg_a = DM_emo_df['arousal'].mean()
DM_avg_d = DM_emo_df['dominance'].mean()

DE_avg_v = DE_emo_df['valence'].mean()
DE_avg_a = DE_emo_df['arousal'].mean()
DE_avg_d = DE_emo_df['dominance'].mean()

BBC_avg_v = BBC_emo_df['valence'].mean()
BBC_avg_a = BBC_emo_df['arousal'].mean()
BBC_avg_d = BBC_emo_df['dominance'].mean()

FT_avg_v = FT_emo_df['valence'].mean()
FT_avg_a = FT_emo_df['arousal'].mean()
FT_avg_d = FT_emo_df['dominance'].mean()

TG_avg_v = TG_emo_df['valence'].mean()
TG_avg_a = TG_emo_df['arousal'].mean()
TG_avg_d = TG_emo_df['dominance'].mean()
```

```
TM_avg_v = TM_emo_df['valence'].mean()
TM_avg_a = TM_emo_df['arousal'].mean()
TM_avg_d = TM_emo_df['dominance'].mean()
```

In [775…

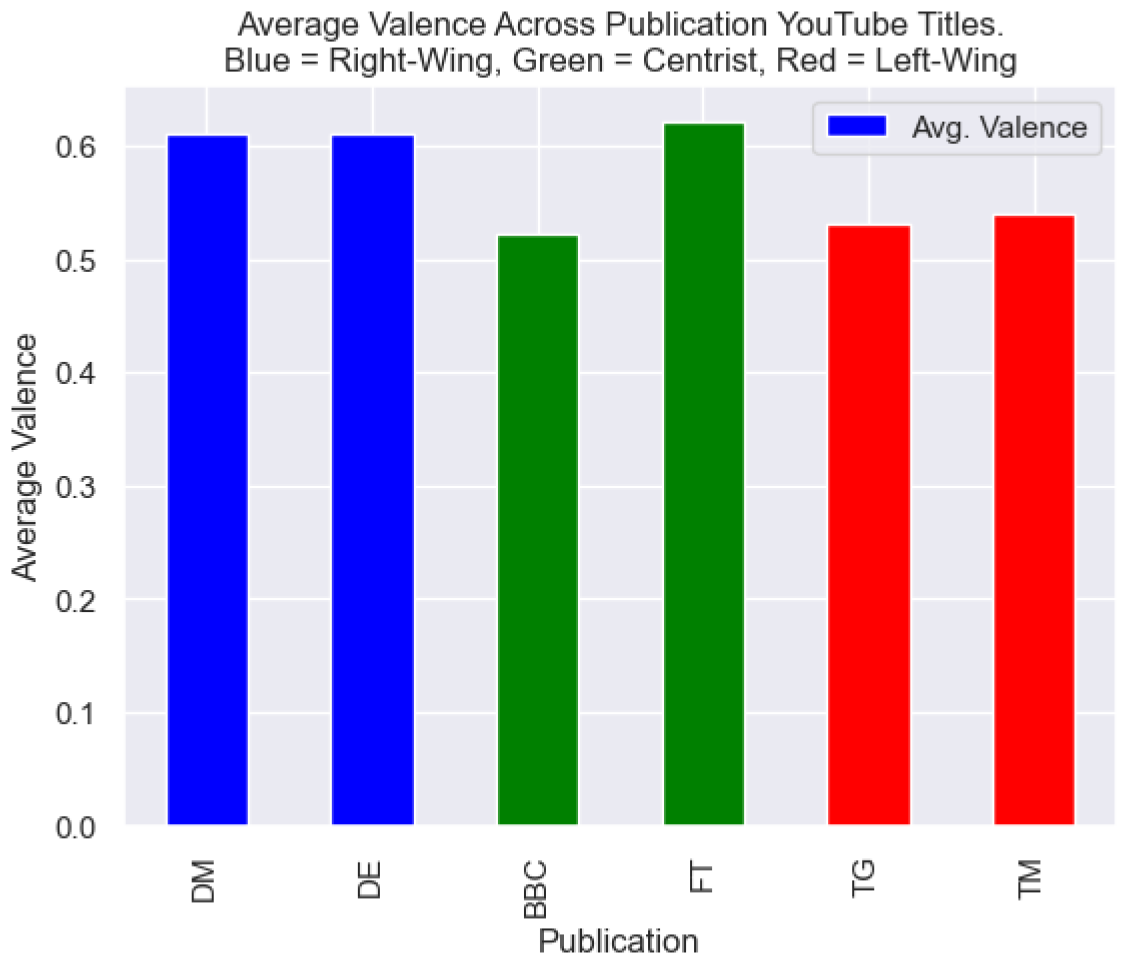
```
#Comparing means of VAD scores
Analysis_data = {'Publication': ['DM', 'DE', 'BBC','FT','TG','TM'],
        'Avg. Valence': [DM_avg_v, DE_avg_v, BBC_avg_v, FT_avg_v, TG_avg_v, TM_avg_v],
        'Avg. Arousal': [DM_avg_a, DE_avg_a, BBC_avg_a, FT_avg_a, TG_avg_a, TM_avg_a],
        'Avg. Dominance': [DM_avg_d, DE_avg_d, BBC_avg_d, FT_avg_d, TG_avg_d, TM_avg_d]}
comp_df = pd.DataFrame(Analysis_data)
```

In [776…

```
#Plotting average VAD values against publications.
colours = ['blue', 'blue', 'green', 'green', 'red', 'red']
comp_df.plot(x='Publication', y='Avg. Valence', kind='bar', color = colours)
plt.title('Average Valence Across Publication YouTube Titles.\nBlue = Right-Wing, Green = Centrist, Red = Left-Wing')
plt.xlabel('Publication')
plt.ylabel('Average Valence')
```

Out[776]:

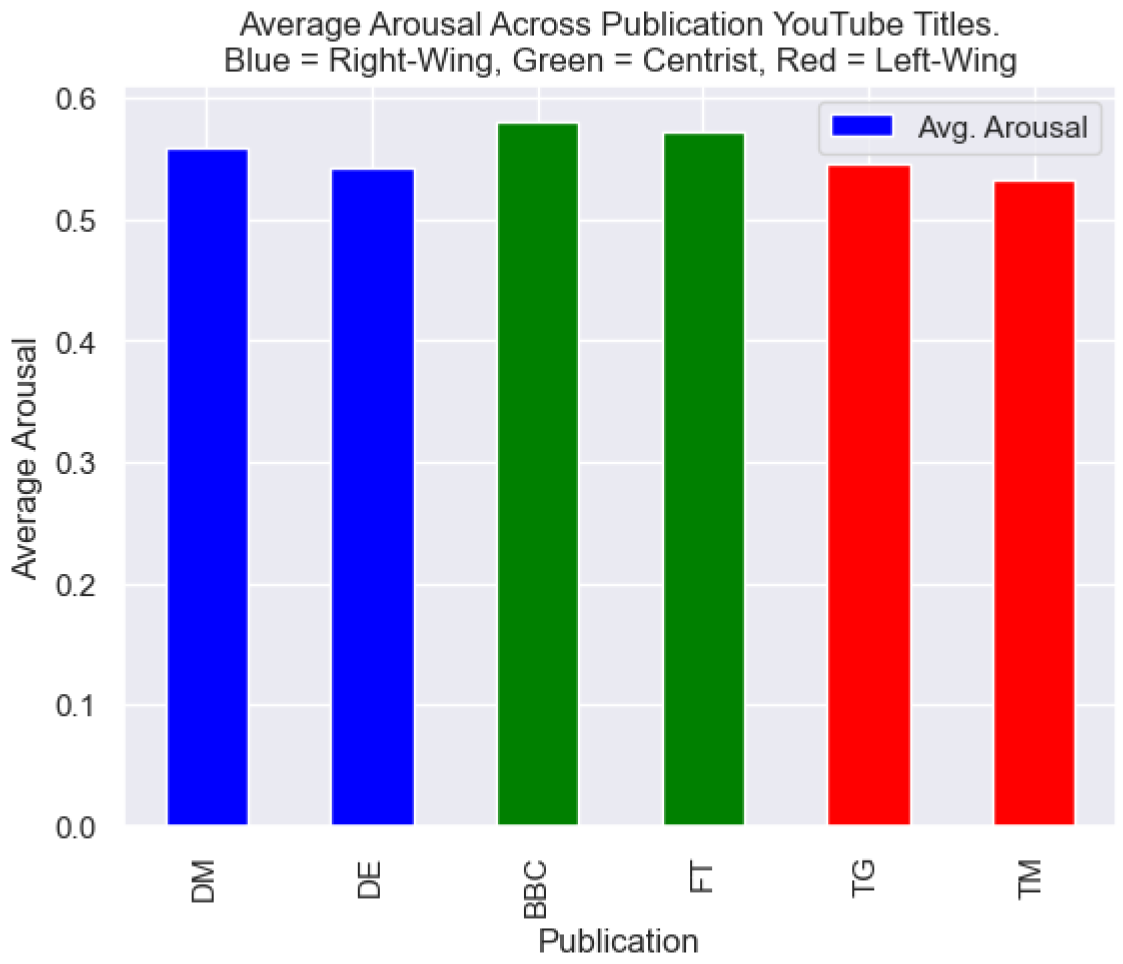
```
Text(0, 0.5, 'Average Valence')
```



```
In [777…  #The VAD model was successful in representing how a news' political leaning influenced the emotional makeup of their t
```

```
In [778…  #Right-wing publications, on average, use happier emotional language than centrist/left-wing publications.
          #I have no plausible reasoning as to why.
```

```
In [779…  comp_df.plot(x='Publication', y='Avg. Arousal', kind='bar', color = colours)
          plt.title('Average Arousal Across Publication YouTube Titles.\nBlue = Right-Wing, Green = Centrist, Red = Left-Wing')
          plt.xlabel('Publication')
          plt.ylabel('Average Arousal')

Out[779]: Text(0, 0.5, 'Average Arousal')
```



In [780…

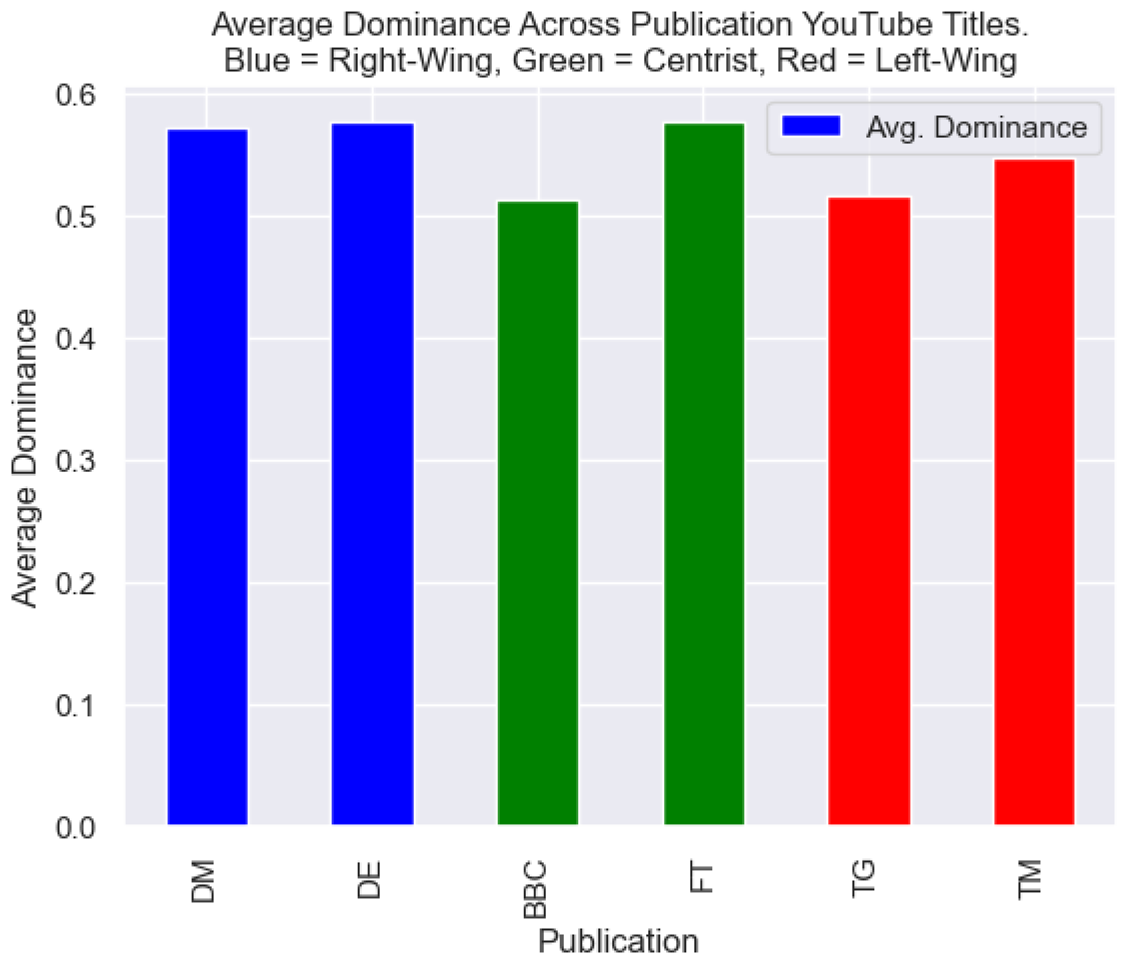
```
#All publications feature a high level of arousal.
#This is to be expected as news titles readily use arousing text to maintain reader attention.
```

In [781…

```
comp_df.plot(x='Publication', y='Avg. Dominance', kind='bar', color = colours)
plt.title('Average Dominance Across Publication YouTube Titles.\nBlue = Right-Wing, Green = Centrist, Red = Left-Wing'
plt.xlabel('Publication')
plt.ylabel('Average Dominance')
```

Out[781]:

```
Text(0, 0.5, 'Average Dominance')
```



In [782…

```
#Right-wing publications, on average, utilise dominating language more than other publications.
#I propose that dominating language instils security in those that read them.
#This security maybe attractive to the right-wing audience who aims to also perpetuate secure, conservative values.
```

In [783…

```
#Plotting TG and DM VAD values in 3D.
x1 = DM_emo_df['valence']
y1 = DM_emo_df['arousal']
z1 = DM_emo_df['dominance']

x2 = DE_emo_df['valence']
y2 = DE_emo_df['arousal']
z2 = DE_emo_df['dominance']
```

```
x3 = BBC_emo_df['valence']
y3 = BBC_emo_df['arousal']
z3 = BBC_emo_df['dominance']

x4 = FT_emo_df['valence']
y4 = FT_emo_df['arousal']
z4 = FT_emo_df['dominance']

x5 = TG_emo_df['valence']
y5 = TG_emo_df['arousal']
z5 = TG_emo_df['dominance']

x6 = TM_emo_df['valence']
y6 = TM_emo_df['arousal']
z6 = TM_emo_df['dominance']

trace1 = go.Scatter3d(x=x1, y=y1, z=z1, mode='markers', marker=dict(size=5, color='blue'), name='The Daily Mail')
trace2 = go.Scatter3d(x=x2,y=y2,z=z2,mode='markers',marker=dict(size=5,color='blue'),name='The Daily Express')
trace3 = go.Scatter3d(x=x3,y=y3,z=z3,mode='markers',marker=dict(size=5,color='green'),name='The BBC')
trace4 = go.Scatter3d(x=x4,y=y4,z=z4,mode='markers',marker=dict(size=5,color='green'),name='The Financial Times')
trace5 = go.Scatter3d(x=x5,y=y5,z=z5,mode='markers',marker=dict(size=5,color='red'),name='The Guardian')
trace6 = go.Scatter3d(x=x6,y=y6,z=z6,mode='markers',marker=dict(size=5,color='red'),name='The Mirror')

fig = go.Figure(data=[trace1, trace2, trace3, trace4, trace5, trace6])
fig.update_layout(title='Comparing VAD Scores Across Right-Wing, Centrist, and Left-Wing UK Publications.\nAchieved Th
fig.show()
```

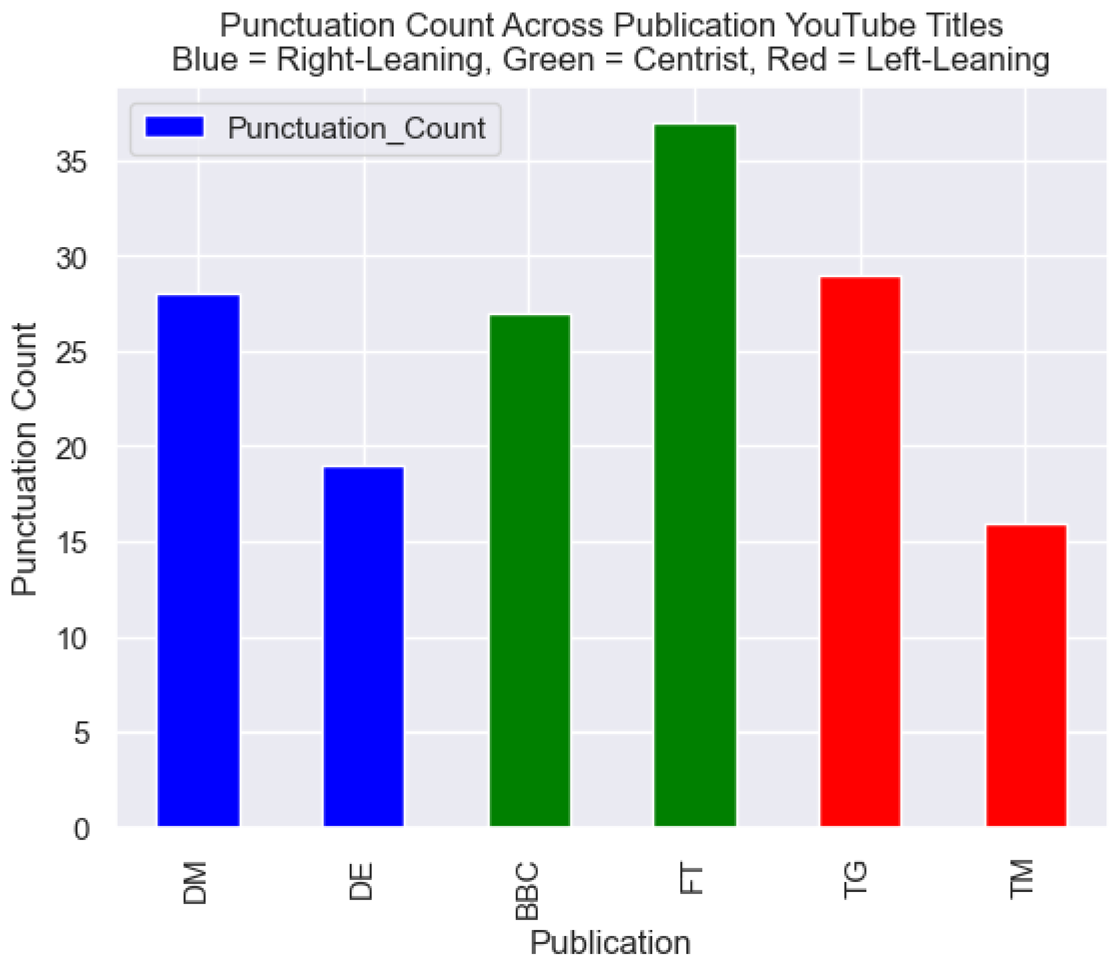
In [784…
```
#VAD data was modelled in 3D, highlighting how similar publishers were to each other.
#They all follow an identical distribution to normal VAD models (negative linear correlation between Dominance and Val
```

In [785…
```
#I did notice a missing patch in distribution between low arousal and dominance.
#This may also be a feature of common VAD models — I cannot confirm with the data available.
#Being in control is arousing, therefore, the inverse should be true.
```

In [786…
```
#SECTION 4: ANALYSIS OF PUNCTUATION ACROSS PUBLICATIONS.
```

In [787…
```
Punct_data = {'Publication': ['DM', 'DE', 'BBC', 'FT', 'TG', 'TM'],'Punctuation_Count': [DM_num_punctuation, DE_num_pu
Punct_df = pd.DataFrame(Punct_data)
Punct_df.plot(x='Publication', y='Punctuation_Count', kind='bar', color = colours)
plt.title('Punctuation Count Across Publication YouTube Titles\nBlue = Right-Leaning, Green = Centrist, Red = Left-Lea
plt.xlabel('Publication')
plt.ylabel('Punctuation Count')
```

Out[787]:
```
Text(0, 0.5, 'Punctuation Count')
```

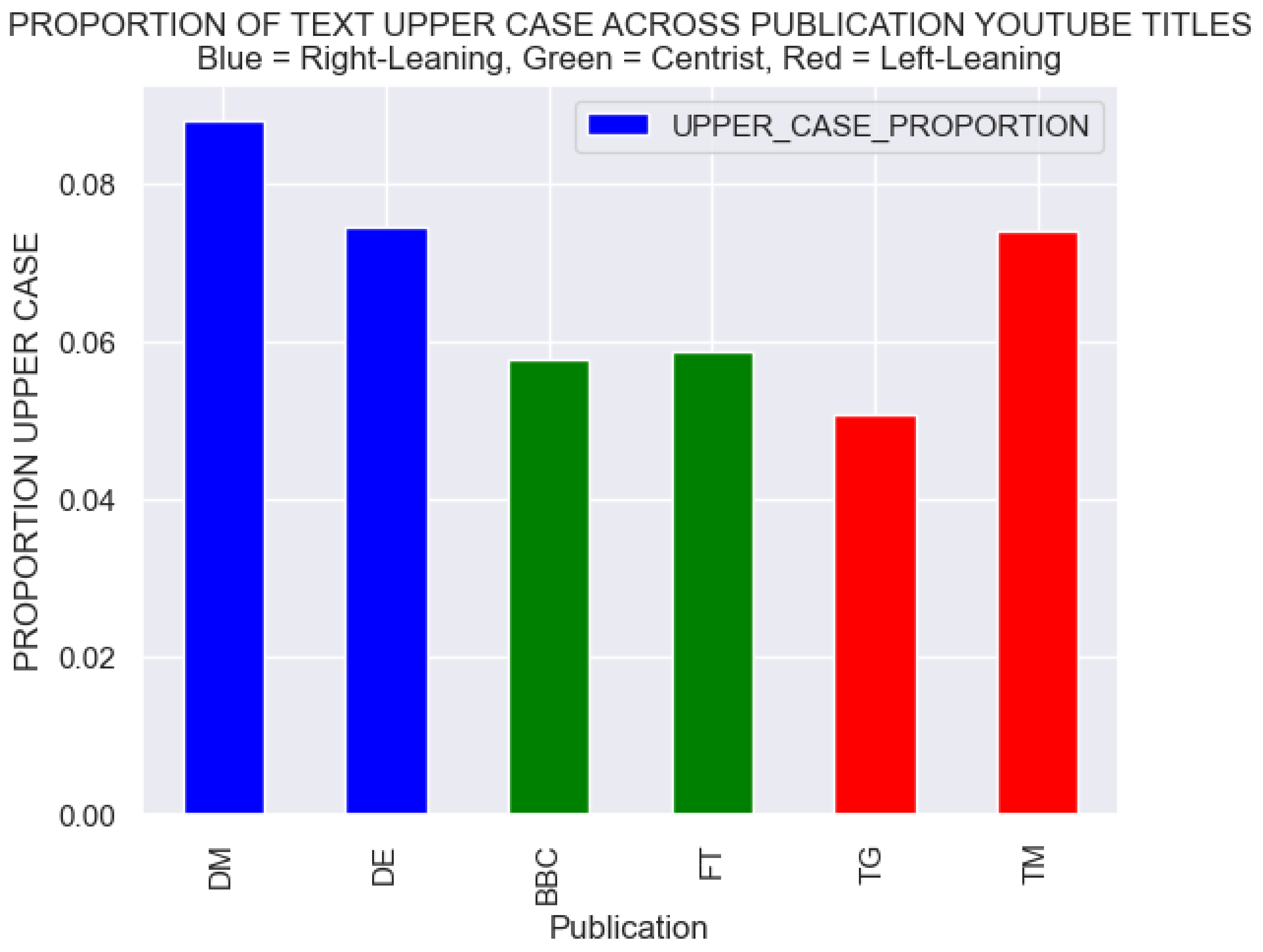


In [788…
```
#Centrist publishers use more punctuation, on average, than other publishers.
#I could not gain meaningful analysis from this chart as I was not specific enough about which types of punctuation we
```

```
In [789…  #SECTION 5: ANALYSIS OF CASE OF TEXT ACROSS PUBLICATIONS.
```

```
In [790…  UPPER_CASE_data = {'Publication': ['DM', 'DE', 'BBC', 'FT', 'TG', 'TM'],'UPPER_CASE_PROPORTION': [DM_PROP_UPPER, DE_PR
          UPPER_CASE_df = pd.DataFrame(UPPER_CASE_data)
          UPPER_CASE_df.plot(x='Publication', y='UPPER_CASE_PROPORTION', kind='bar', color = colours)
          plt.title('PROPORTION OF TEXT UPPER CASE ACROSS PUBLICATION YOUTUBE TITLES\nBlue = Right-Leaning, Green = Centrist, Re
          plt.xlabel('Publication')
          plt.ylabel('PROPORTION UPPER CASE')
```

```
Out[790]: Text(0, 0.5, 'PROPORTION UPPER CASE')
```

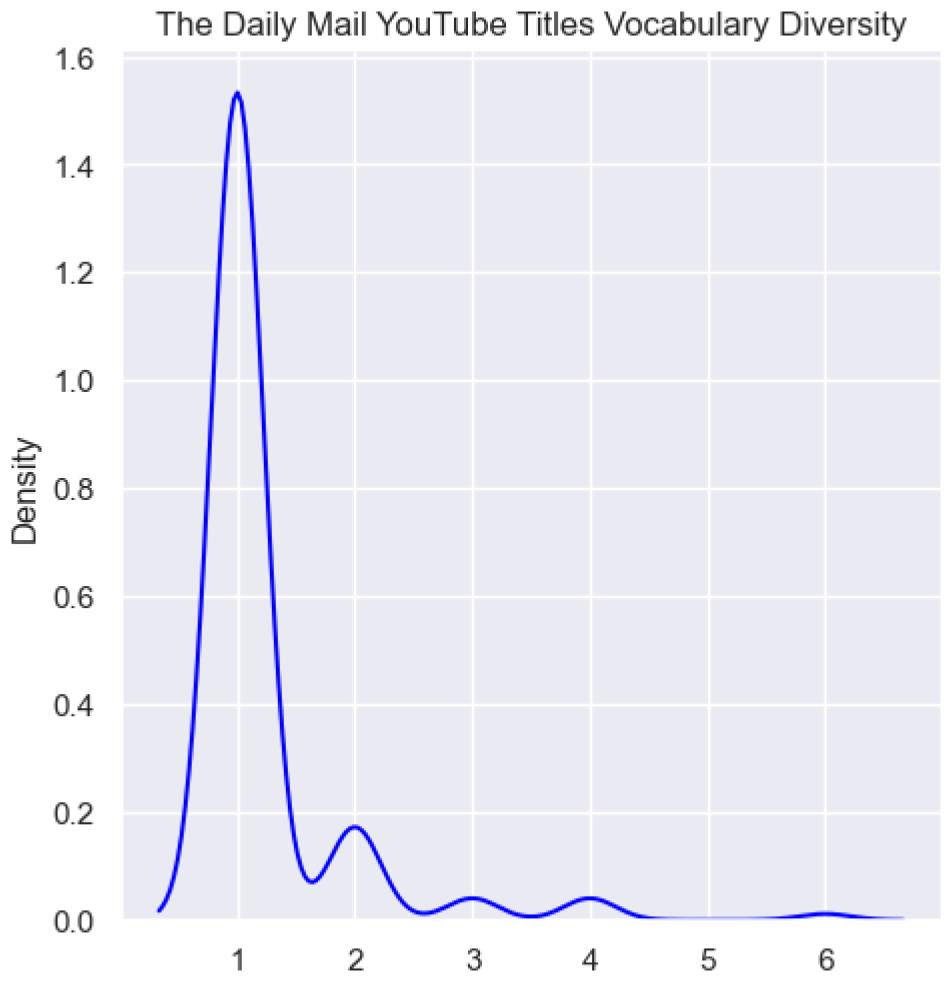


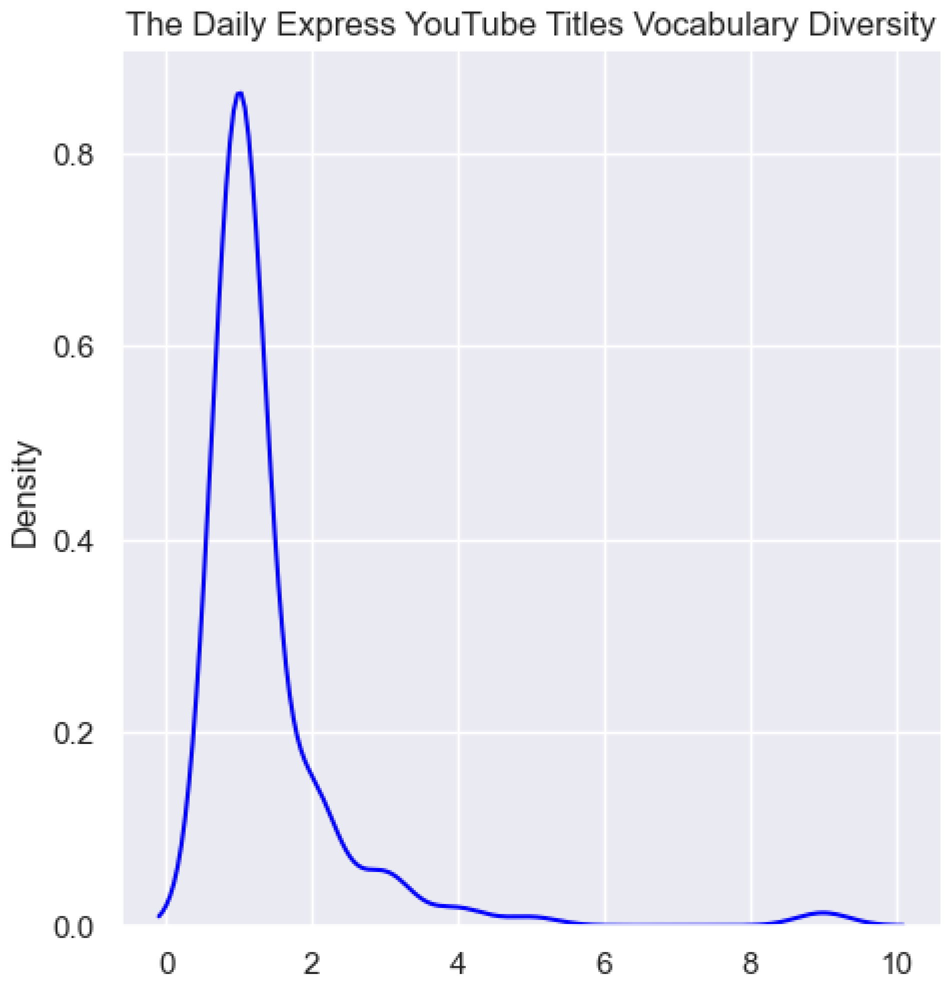
In [791…
```
#The right-wing uses more UPPER-CASE text, on average, than other publications.
#Upper case text increases the urgency of a headline, maintaining reader attention.
#This is not symbolic of the political leaning, more of the sensational nature of the title.
```

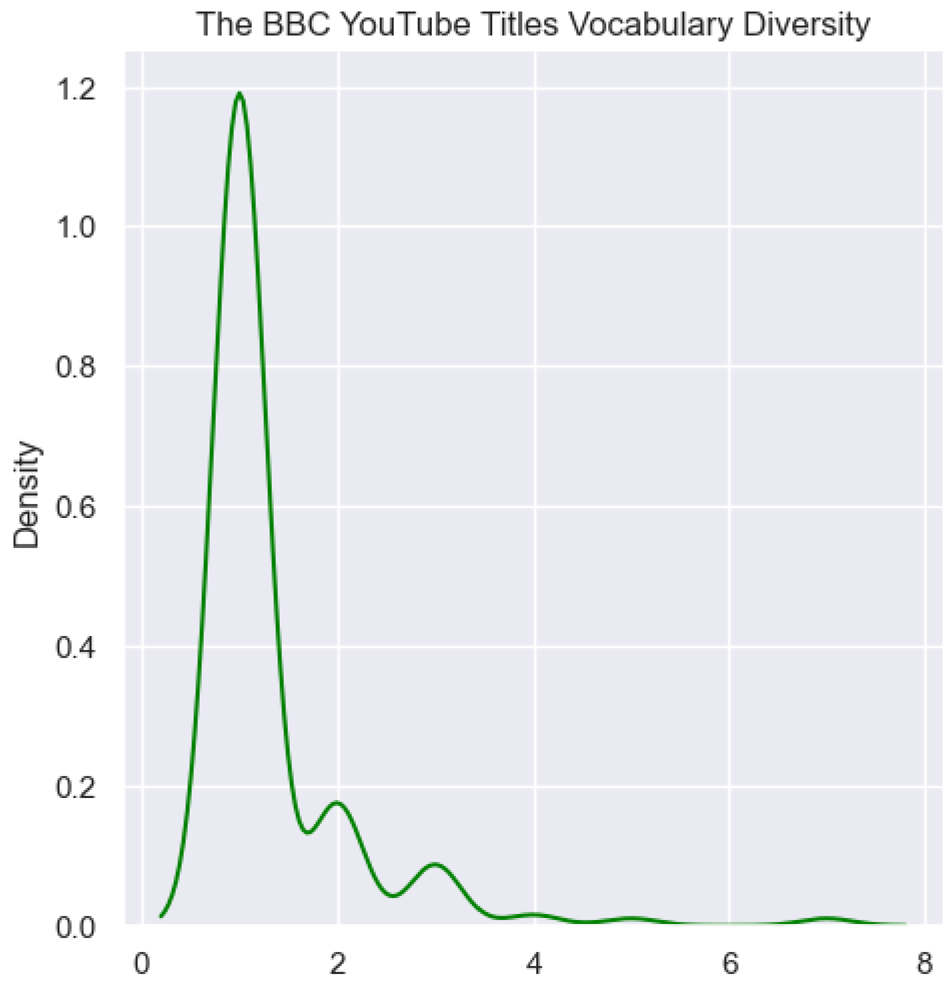
In [792…
```
#SECTION 6: ANALYSIS OF VOCABULARY RANGE ACROSS PUBLICATIONS.
```

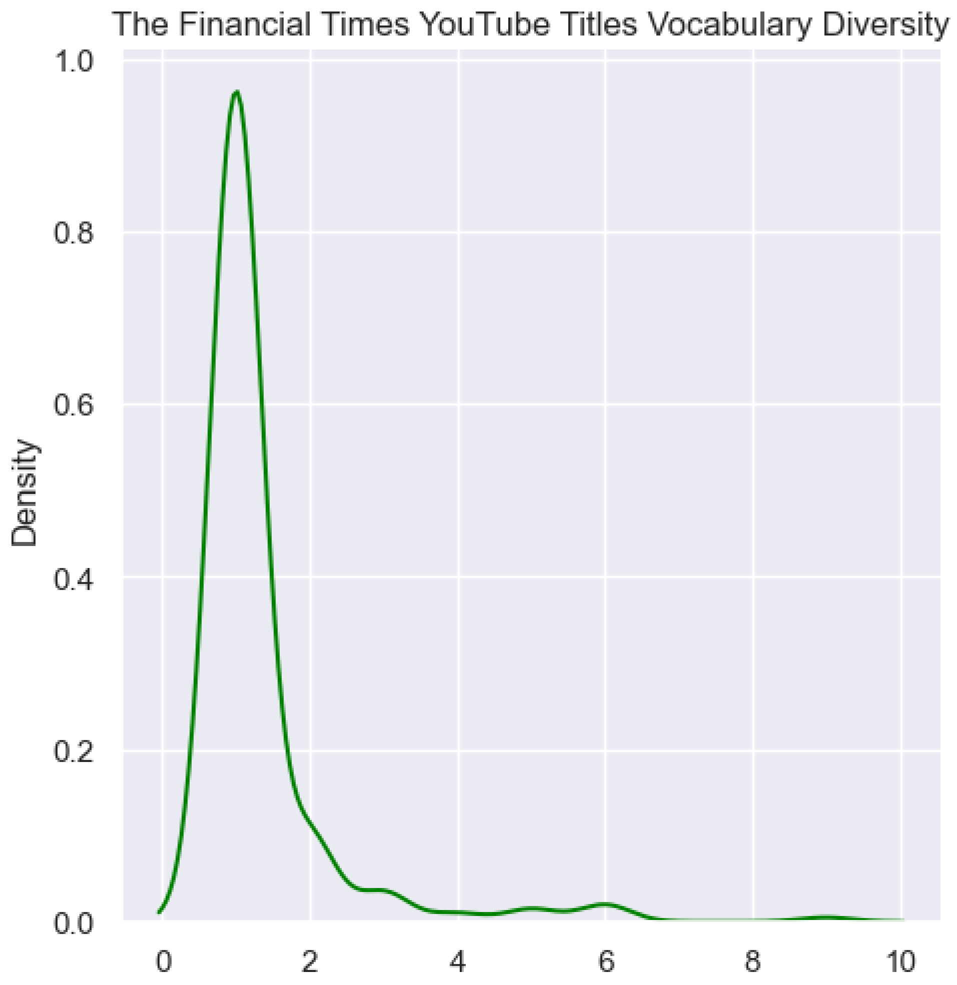
In [793…
```
DM_word_counts_plot = sns.displot(DM_word_counts, kind = 'kde', color = 'blue')
DM_word_counts_plot.set(title='The Daily Mail YouTube Titles Vocabulary Diversity')

DE_word_counts_plot = sns.displot(DE_word_counts, kind = 'kde', color = 'blue')
DE_word_counts_plot.set(title='The Daily Express YouTube Titles Vocabulary Diversity')

BBC_word_counts_plot = sns.displot(BBC_word_counts, kind = 'kde', color = 'green')
BBC_word_counts_plot.set(title='The BBC YouTube Titles Vocabulary Diversity')

FT_word_counts_plot = sns.displot(FT_word_counts, kind = 'kde', color = 'green')
FT_word_counts_plot.set(title='The Financial Times YouTube Titles Vocabulary Diversity')

TG_word_counts_plot = sns.displot(TG_word_counts, kind = 'kde', color = 'red')
TG_word_counts_plot.set(title='The Guardian YouTube Titles Vocabulary Diversity')

TM_word_counts_plot = sns.displot(TM_word_counts, kind = 'kde', color = 'red')
TM_word_counts_plot.set(title='The Mirror YouTube Titles Vocabulary Diversity')
```

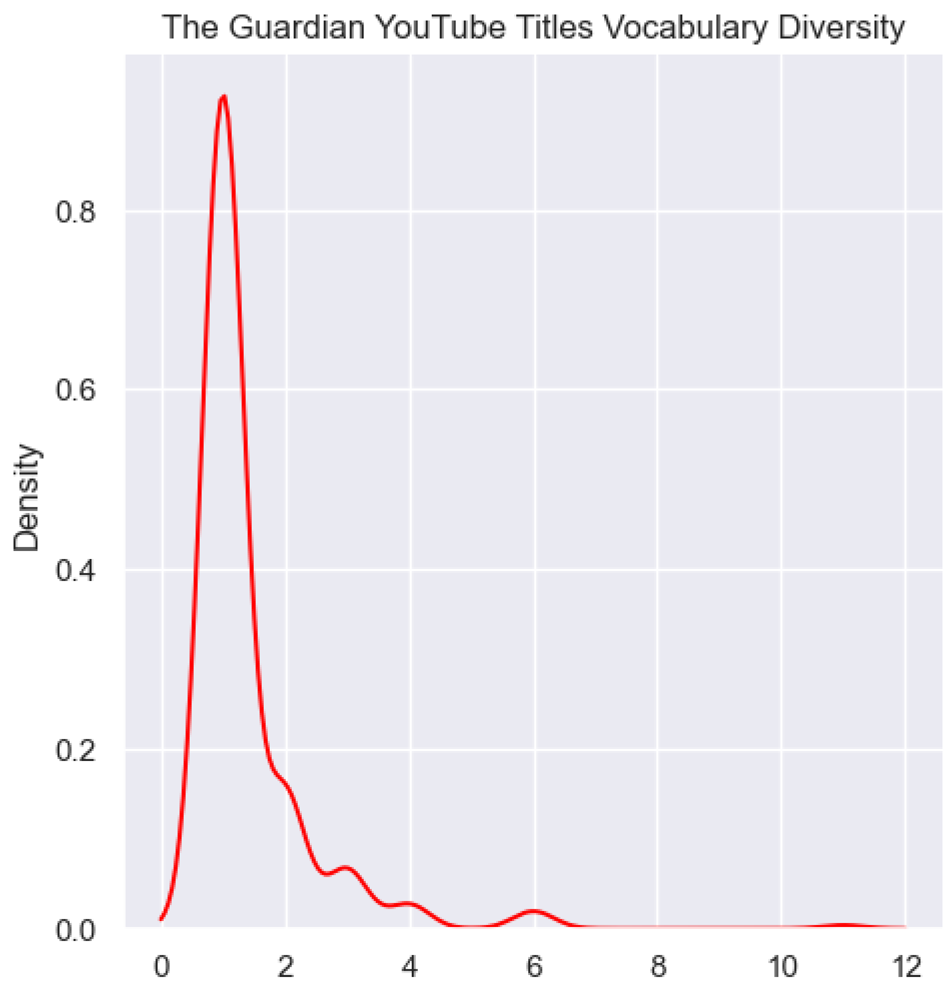
Out[793]:
```
<seaborn.axisgrid.FacetGrid at 0x7fb310d3acd0>
```

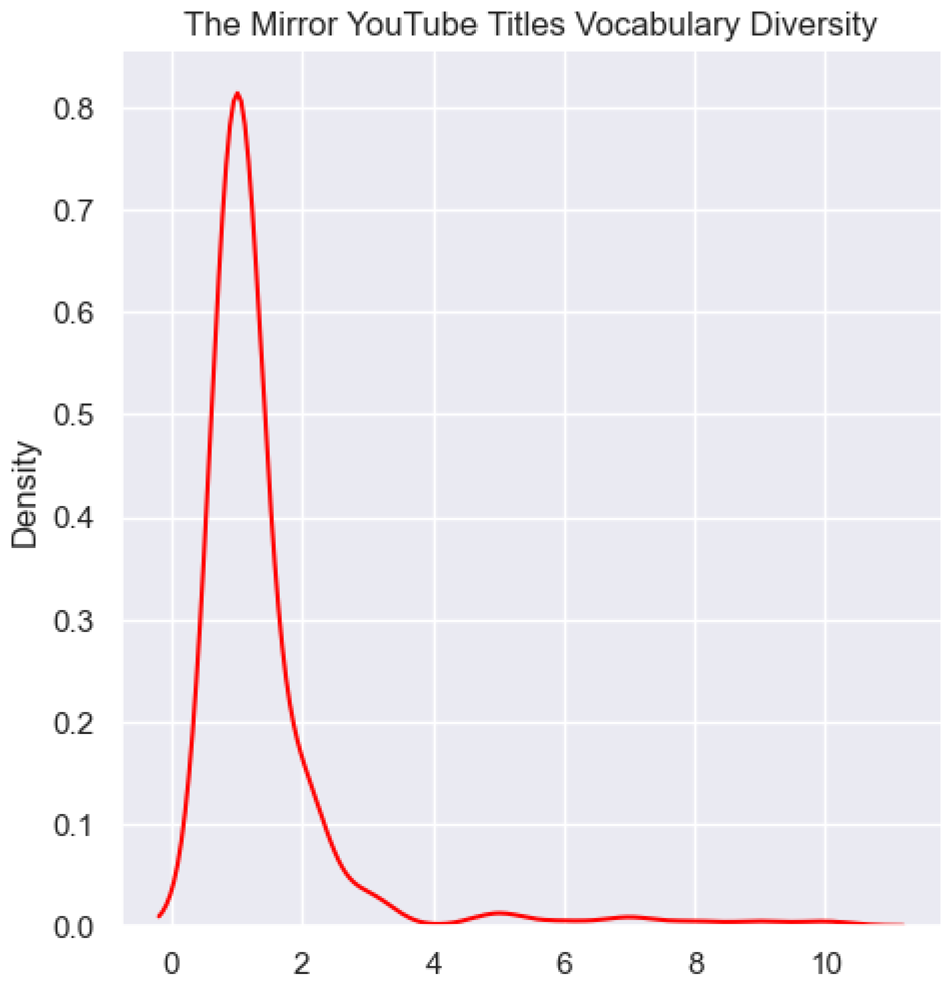

The Daily Mail YouTube Titles Vocabulary Diversity
Density
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
1
2
3
4
5
6


The Daily Express YouTube Titles Vocabulary Diversity
Density
0.0
0.2
0.4
0.6
0.8
0
2
4
6
8
10


The BBC YouTube Titles Vocabulary Diversity
Density
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8


The Financial Times YouTube Titles Vocabulary Diversity
Density
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10


The Guardian YouTube Titles Vocabulary Diversity
Density
0.0
0.2
0.4
0.6
0.8
0
2
4
6
8
10
12



```
In [794…  #I could not find any meaning from analysis into vocabulary range as graphs were too inconsistent in spread.
          #They did all, however, follow an expected zip’s law distribution.
```

```
In [795…  #SECTION 7: Word Embeddings
```

```
In [796…  #Formatting for W2V
          CT_w2v = [sentence.split() for sentence in Combined_Title_List]
          CT_model = gensim.models.Word2Vec(CT_w2v, min_count=1, vector_size = 30)
```

In [797…
```
#Sorting by most freqeunt words in vocab
CT_vocab = CT_model.wv.index_to_key

#Assiging vocab to its unique matrix values
CT_vectors = [CT_model.wv[i] for i in CT_vocab]

#Dataframe of matrix values
CT_df = pd.DataFrame(CT_vectors)
```

In [798…
```
#Perform PCA analysis on dataframe
CT_pca = PCA(n_components=3)
CT_PCA_result = CT_pca.fit_transform(CT_df)
CT_PCA_df = pd.DataFrame(CT_PCA_result, columns=['PC1', 'PC2', 'PC3'])
```

In [799…
```
#Callibrating first column back with original words
CT_PCA_complete_df = pd.concat([pd.Series(CT_vocab, name='word'), CT_PCA_df], axis=1)
```

In [800…
```
#Clustering
kmeans = KMeans(n_clusters=6).fit(CT_PCA_complete_df[['PC1', 'PC2', 'PC3']])
CT_PCA_complete_df['cluster'] = [str(i) for i in kmeans.labels_]
```

In [801…
```
#Visualising
WE_fig = px.scatter_3d(CT_PCA_complete_df, x='PC1', y='PC2', z='PC3', color='cluster',  hover_data = ['word'])
WE_fig.update_layout(title='YouTube Titles of Left/Centre/Right Wing UK news outlets – PCA analysis of Word Embedding'
WE_fig.update_traces(marker=dict(size = 8, line=dict(width=1, color='black')), selector=dict(mode='markers'))
WE_fig.show()
```

In [802…
```
#This model separates the plots into a cube-like arrangement, whereby each side of the cube is another cluster from PC
#The groupings are random in nature, with no common thread between any word groupings.
#This randomness within clusters was not due to how many clusters assigned to the model.
```

In [803…
```
#SECTION 8: Topic Analysis
```

In [804…
```
#Setting up Vectorizer
vectorizer = TfidfVectorizer(input = 'content', strip_accents = 'ascii', stop_words = 'english')
```

```
#Channeling combined publisher data through vectoriser
CT_V = vectorizer.fit_transform(Combined_Title_List)
CT_V = CT_V.todense().tolist()
```

In [805…

```
#Converting to distance matrix to handy df
CT_V_Df = pd.DataFrame(CT_V, columns=vectorizer.get_feature_names())
Distances = [[] for i in range(len(CT_V_Df))]
for i in range(len(CT_V_Df)):
    for j in range(len(CT_V_Df)):
        Distances[i].append(distance.cosine(CT_V_Df.iloc[i], CT_V_Df.iloc[j]))
Dist_CT_V_Df = pd.DataFrame(Distances, columns = CT_V_Df.index, index = CT_V_Df.index)
```

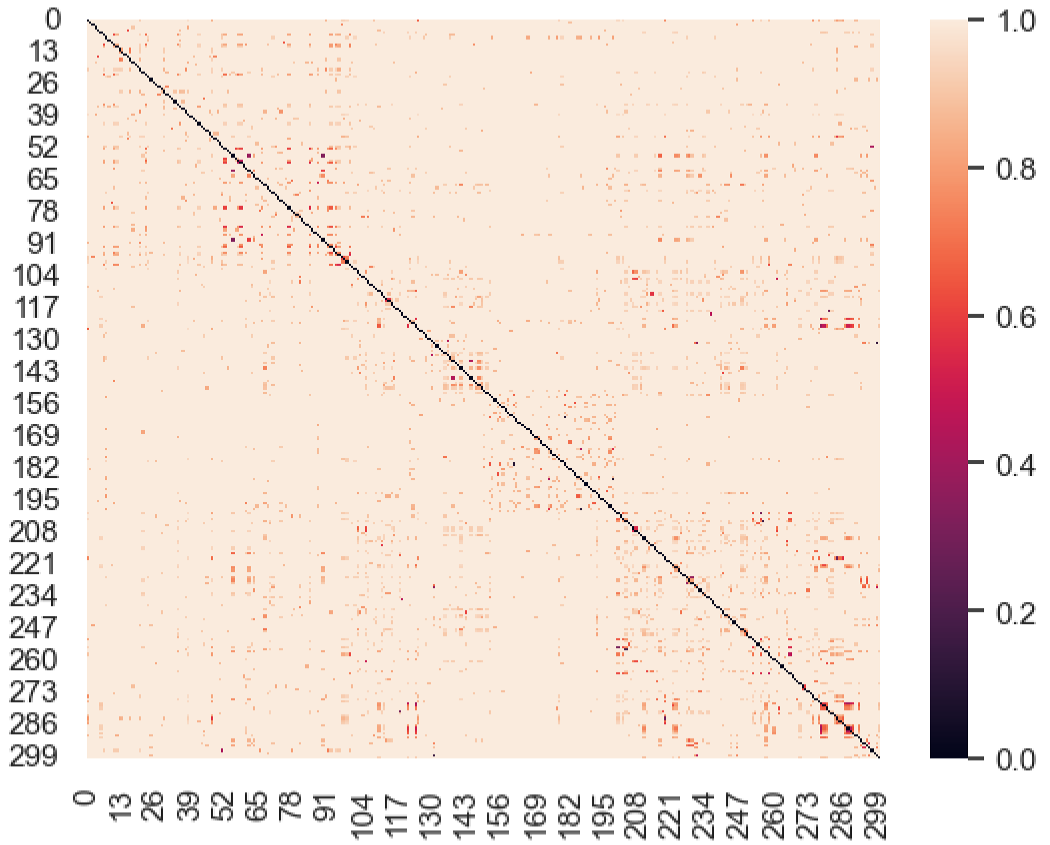
```
/Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages/sklearn/utils/deprecation.py:87: FutureWarning:

Function get_feature_names is deprecated; get_feature_names is deprecated in 1.0 and will be removed in 1.2. Please us
e get_feature_names_out instead.
```

In [806…

```
#Heatmap of dataframe
sns.heatmap(Dist_CT_V_Df)
```

Out[806]:

```
<AxesSubplot:>
```



In [807…

```
pca_1 = PCA(n_components = 3)
comps_1 = pca_1.fit_transform(Dist_CT_V_Df)
pc_df_1 = pd.DataFrame(data = comps_1, columns = ['PC1', 'PC2', 'PC3'])
```

In [808…

```
#Clustering
clustering = AgglomerativeClustering(n_clusters=5, linkage='ward').fit(Dist_CT_V_Df)
kmeans = KMeans(n_clusters=5, random_state=0).fit(Dist_CT_V_Df)
```

```
/Users/louisvsbigmac/opt/anaconda3/lib/python3.9/site-packages/scipy/cluster/hierarchy.py:834: ClusterWarning:

scipy.cluster: The symmetric non-negative hollow observation matrix looks suspiciously like an uncondensed distance ma
trix
```

In [809…
```
#DF for visualisation
TC_DF_All = pd.concat([Dist_CT_V_Df, pc_df_1], axis = 1)
```

In [810…
```
TC_DF_All['clusters_ag'] = [str(i) for i in clustering.labels_]
TC_DF_All['clusters_knn'] = [str(i) for i in kmeans.labels_]
TC_DF_All['Titles'] = Combined_Titles_df["Titles"]
TC_DF_All['Publisher'] = Combined_Titles_df["Publisher"]
TC_DF_All['Political Compass'] = Combined_Titles_df["Political Compass"]
```

In [811…
```
#Visualising
TA_fig = px.scatter_3d(TC_DF_All, x='PC1', y='PC2', z='PC3', color='clusters_ag', hover_data = ['Titles','Political Cc
TA_fig.update_traces(marker=dict(size = 8, line=dict(width=1, color='black')), selector=dict(mode='markers'))
TA_fig.update_layout(title='YouTube Titles of Left/Centre/Right Wing UK news outlets — Topic analysis')
TA_fig.show()
```

In [818…

```
#PCA/Topic analysis was not a fruitful endeavour.
#The clusters are indecipherable, apart from the observation that it clusters the titles in accordance with their poli
#Most Left/Right-Wing titles are dotted outside of the central blob.
#Centrist publications are practically all concentrated in the centre (Figure 12).
```

In [819…

```
#Conclusions
#I struggled to build on top of the meaningful analysis conducted in my earlier inquiry.
#Whilst I am confident that my methodology is correct in the code, the results didn’t tell me anything new, or that of
#I guess that I had a small source dataset, and that this is not befitted to larger PCA analysis, however, I am still
```

```
#I was also not logical in the methodology of my inquiry, several charts (punctuation/case of text) are more suited tc
#However, I am very pleased in the novel coding techniques that this project entailed, PCA has an outstanding potentia
```