HEL-8048-1 24V Advanced data analysis and visualization using programming
Final exam project.
Hamid Taghipourbibalan, Ph.D. student at McCutcheon Lab, IPS, Helsefak, UiT.
htbibalan@uit.no,

Click here to access GitHub repository of this project.

Click here to navigate to the html version of the project script (download the file hel8048_project.html).

Next page you will find the PDF version of the project script.

Hi!

Should you have any questions about the script, feel free to contact me at htbibalan@uit.no 😉

In [2]:
```
# importing some libraries that I will use in this script
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
from matplotlib.lines import Line2D
from sklearn.linear_model import LinearRegression
import numpy as np
import matplotlib.ticker as ticker
import pycountry
import re
from collections import Counter
import geopandas as gpd
# codes below adjust the number of rows and columns displayed by Pandas, remove
# pd.options.display.max_rows= 40000
# pd.options.display.max_columns= 6

# The line below defines the pathway to store plots which will be used throughou
fig_folder = "../plots//"

# changing the font used for plots
plt.rcParams['font.family'] = 'cambria'
```

In [4]:
```
# The line below gets the CSV file which is stored in the "data" folder
csv="../data/trump_insult_tweets_2014_to_2021.csv"
# This line reads the CSV file called in the previous line
df_trump=pd.read_csv(csv)
# This line displays the content of the CSV file stored in the object " df_trump
df_trump

# Uncomment the lines below if you want to see the length of the dataset or a ra
#len(df_trump)
#df_trump.sample(10)
```

Out[4]:

	Unnamed: 0	date	target	insult	tweet
0	1	2014-10-09	thomas-frieden	fool	Can you believe this fool, Dr. Thomas Frieden ...
1	2	2014-10-09	thomas-frieden	DOPE	Can you believe this fool, Dr. Thomas Frieden ...
2	3	2015-06-16	politicians	all talk and no action	Big time in U.S. today - MAKE AMERICA GREAT AG...
3	4	2015-06-24	ben-cardin	It's politicians like Cardin that have destroy...	Politician @SenatorCardin didn't like that I s...
4	5	2015-06-24	neil-young	total hypocrite	For the nonbeliever, here is a photo of @Neily...
...	...	...	...	...	...
10355	10356	2021-01-06	2020-election	Many States want to decertify the mistake they...	If Vice President @Mike_Pence comes through fo...
10356	10357	2021-01-06	2020-election	based on irregularities and fraud, plus corrup...	States want to correct their votes, which they...
10357	10358	2021-01-06	2020-election	Our Election Process is worse than that of thi...	They just happened to find 50,000 ballots late...
10358	10359	2021-01-06	2020-election	a FRAUD	The States want to redo their votes. They foun...
10359	10360	2021-01-06	chuck-todd	Sleepy Eyes, Sad to watch!	Sleepy Eyes Chuck Todd is so happy with the fa...

10360 rows × 5 columns

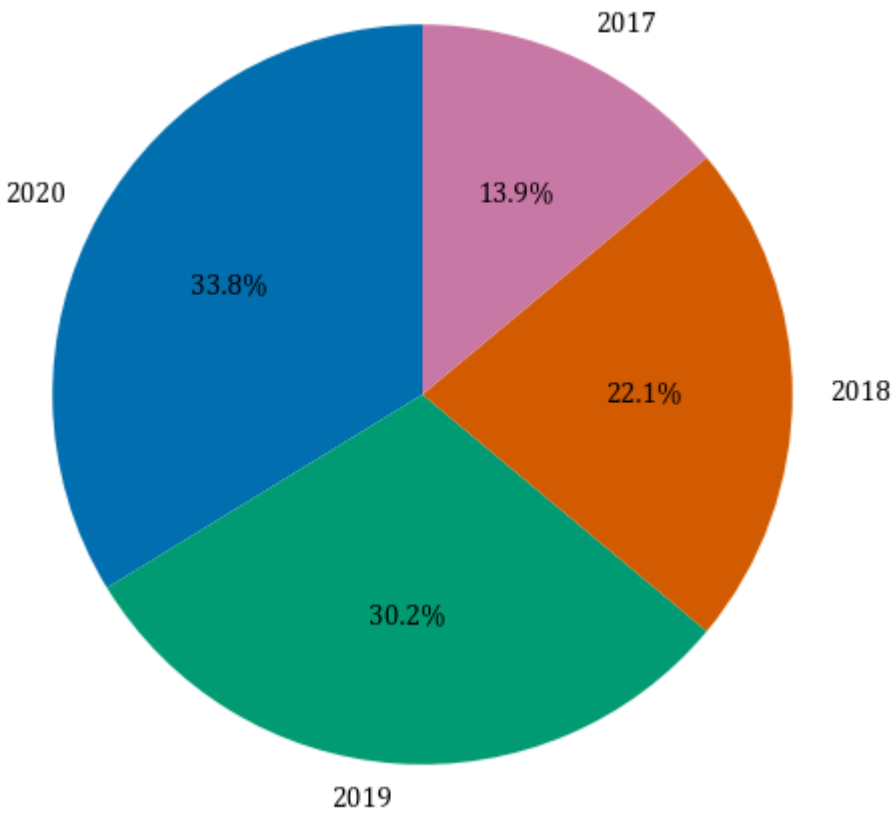
In the cell below, I am rearranging and adjusting the data frame (df_trump)

In [5]:
```
#this line removes the column "Unnamed:0" from the df_trump and only keeps the i
df_trump.drop(df_trump.columns[0], axis= 1, inplace= True)

# Since I want to attribute dates to the day of the week I am using the to_datet
df_trump['date'] = pd.to_datetime(df_trump['date'])
df_trump['day_of_week'] = df_trump['date'].dt.day_name()

# Next I am going to rearrange the columns in a way that date and day_of_week be
new_column_order = ['date', 'day_of_week', 'target', 'insult', 'tweet']
df_trump = df_trump[new_column_order]

# The date column in the df_trump was not a date.time format, so I am converting
#df_trump['date'] = pd.to_datetime(df_trump['date'])
```

```
# I am filtering the data to the years Donald Trump was in office, from Early 20
df_trump = df_trump[(df_trump['date'] >= '2017-01-01') & (df_trump['date'] <= '2
```

Now that I have sorted the dataframe and filtered it out I try to extrapolate some patterns and plot them

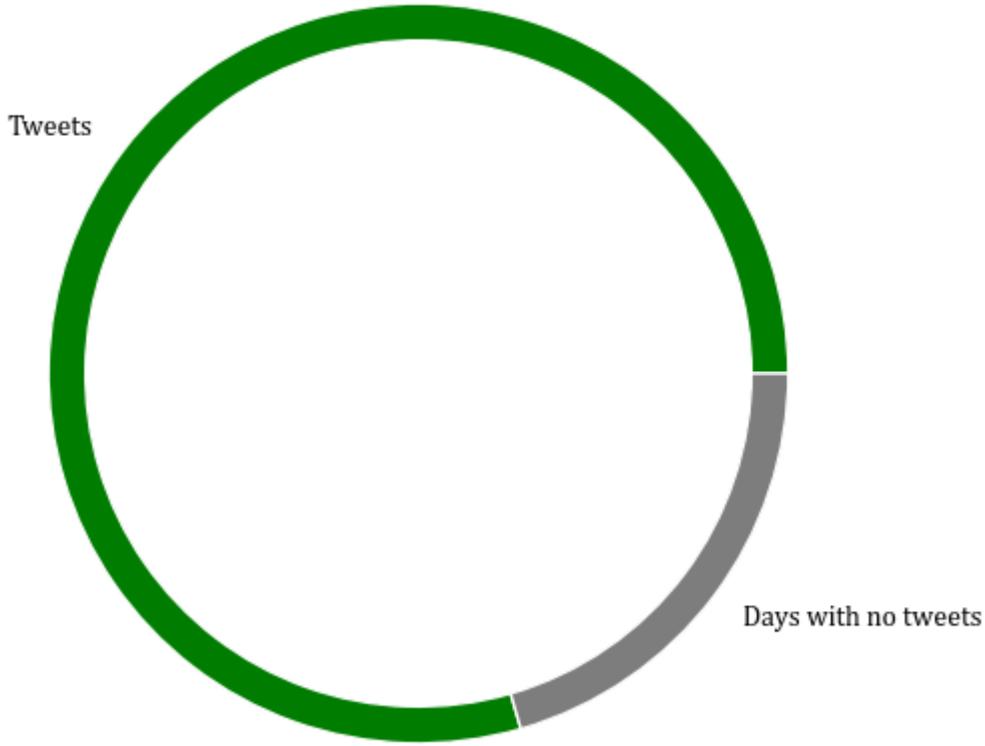
In [24]:
```
#Percentage of tweets per year across the four years in office
#  get the total number of tweets in each year and sort in ascending order

tweets_per_year = df_trump['date'].dt.year.value_counts().sort_index(ascending=F

#here making a pie plot with color-blindness friendly
plt.figure(figsize=(8,6))
plt.pie(tweets_per_year, labels=tweets_per_year.index, autopct='%1.1f%%', starta
plt.title("")
#plt.savefig(fig_folder + "pieplot_tweets.png")

plt.show()
```



In [22]:
```
# finding  dates Trump did not send any tweet at all
all_dates = pd.date_range(start='2017-01-01', end='2020-12-31')
trump_dates = df_trump['date'].dt.date.unique()
missing_dates = all_dates[~all_dates.isin(trump_dates)]
missing_dates
```

```
# using len to find how many days did Trump not send any tweet at all how many d
len(all_dates)
len(trump_dates),
len(missing_dates)


# make a doughnut of the data in this cell (a hollow pie plot!)
plt.figure(figsize=(8,6))
plt.pie([len(trump_dates), len(missing_dates)], labels=['Tweets', 'Days with no
plt.title('')
plt.show()
```

```
C:\Users\hta031\AppData\Local\Temp\ipykernel_50492\1134967701.py:4: FutureWarnin
g: The behavior of 'isin' with dtype=datetime64[ns] and castable values (e.g. str
ings) is deprecated. In a future version, these will not be considered matching b
y isin. Explicitly cast to the appropriate dtype before calling isin instead.
  missing_dates = all_dates[~all_dates.isin(trump_dates)]
```

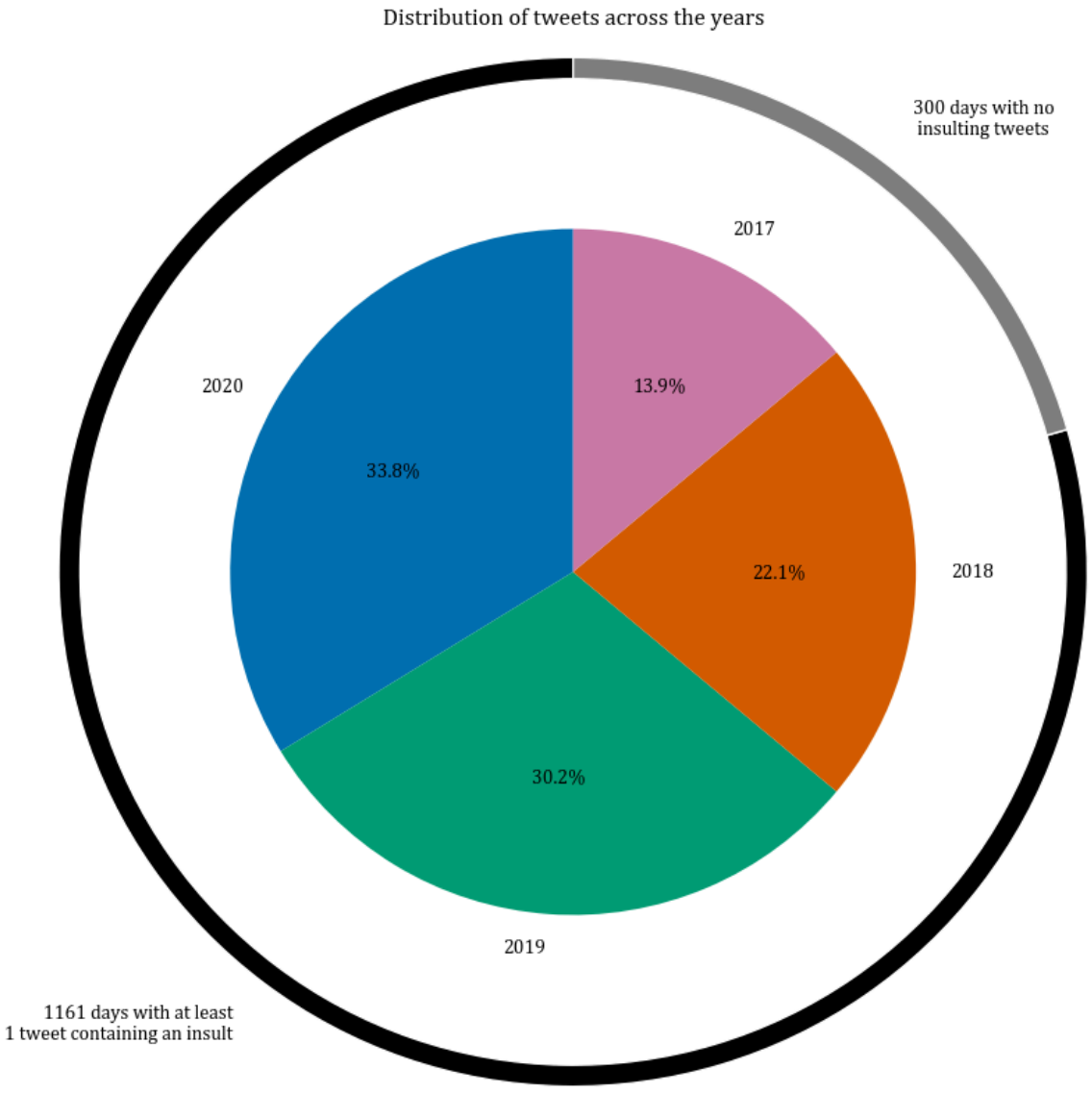


In the cell below making an overlay of the pie plot and doughnut plots

In [25]:
```
tweets_per_year = df_trump['date'].dt.year.value_counts().sort_index(ascending=F
all_dates = pd.date_range(start='2017-01-01', end='2020-12-31')
trump_dates = df_trump['date'].dt.date.unique()
missing_dates = all_dates[~all_dates.isin(trump_dates)]
# Data for the doughnut plot
len(all_dates)
total_days = 1461
```

```
len(missing_dates)
days_with_no_tweets = 300
days_with_tweets = total_days - days_with_no_tweets  # Calculated days with at l

# Setting up the figure
plt.figure(figsize=(10, 10))
# Making the pie plot for the percentage of tweets per year, positioned centrall
plt.pie(tweets_per_year, labels=tweets_per_year.index, autopct='%1.1f%%', starta
        colors=['#0072B2', '#009E73', '#D55E00', '#CC79A7'])
# Adding the doughnut plot around the pie plot with a larger diameter but thinne
plt.pie([days_with_tweets, days_with_no_tweets], radius=1.2, startangle=90,
        colors=['black', 'grey'], wedgeprops=dict(width=0.05, edgecolor='w'),lab

plt.title('Distribution of tweets across the years')
#plt.savefig(fig_folder + "pie_doughnut.png")
plt.show()
```

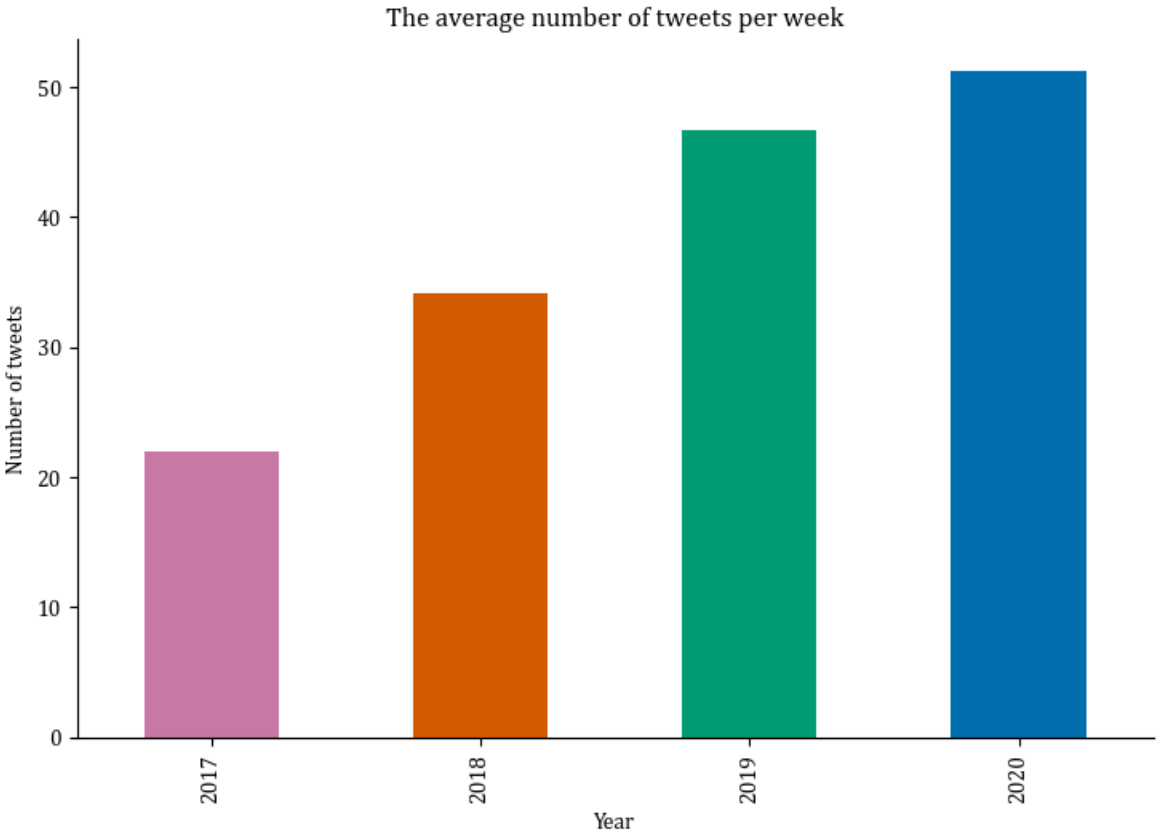
```
C:\Users\hta031\AppData\Local\Temp\ipykernel_50492\3228383670.py:4: FutureWarnin
g: The behavior of 'isin' with dtype=datetime64[ns] and castable values (e.g. str
ings) is deprecated. In a future version, these will not be considered matching b
y isin. Explicitly cast to the appropriate dtype before calling isin instead.
  missing_dates = all_dates[~all_dates.isin(trump_dates)]
```



In [26]:
```
# code below gets the average number of tweets per week per year
df_trump['week'] = df_trump['date'].dt.isocalendar().week
```

```
df_trump['year'] = df_trump['date'].dt.year
df_week = df_trump.groupby(['year', 'week']).size().reset_index(name='number_of_
df_week = df_week.groupby('year')['number_of_tweets'].mean().reset_index()

# Plotting
plt.figure(figsize=(10,10))
ax = df_week.plot(x='year', y='number_of_tweets', kind='bar',width = 0.5, color=

plt.xlabel('Year')
plt.ylabel('Number of tweets')
plt.title('The average number of tweets per week')
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
#ax.legend().remove()
plt.tight_layout(pad=3)
#plt.savefig(fig_folder + "av_tweets_per_week.png")
```

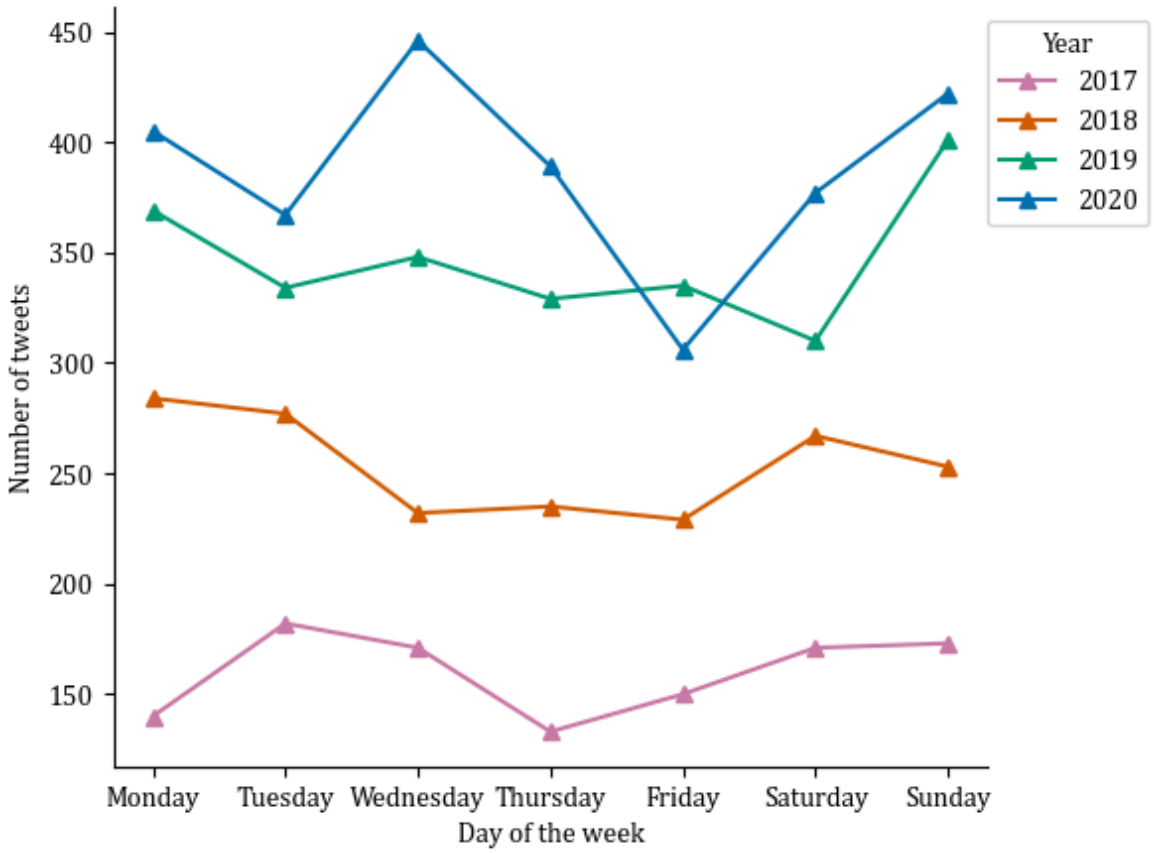
```
<Figure size 1000x1000 with 0 Axes>
```



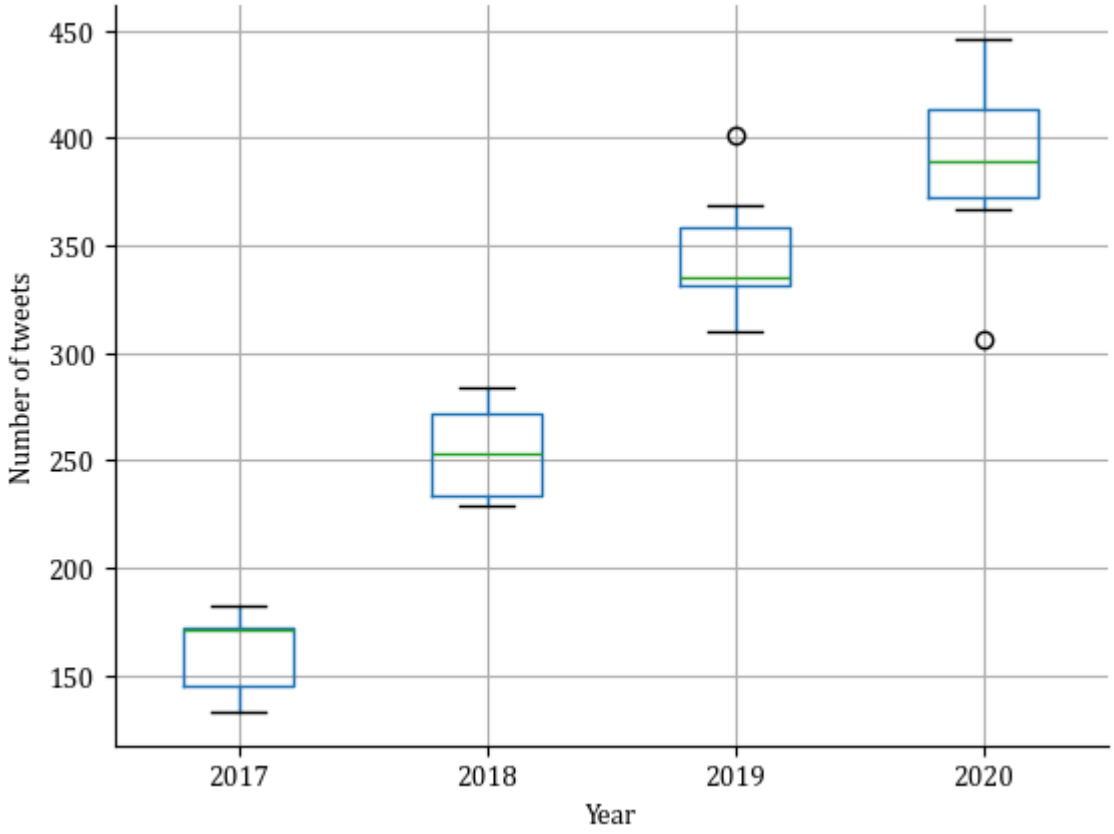
In [27]:

```
# getting the average number of tweets per day of the week per year
tweets_per_day_of_week = df_trump.groupby(['day_of_week', df_trump['date'].dt.ye
tweets_per_day_of_week = tweets_per_day_of_week.reindex(['Monday', 'Tuesday', 'W
plt.figure(figsize=(8,6))
ax = tweets_per_day_of_week.plot(kind='line', marker='^', color=['#CC79A7', '#D5
plt.title('')
plt.xlabel('Day of the week')
plt.ylabel('Number of tweets')
plt.legend( tweets_per_day_of_week.columns, title='Year', loc='upper right', bbo

ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
plt.tight_layout(pad=2)
#plt.savefig(fig_folder + "trend_per_day_per_year.png")
plt.show()
```

```
<Figure size 800x600 with 0 Axes>
```

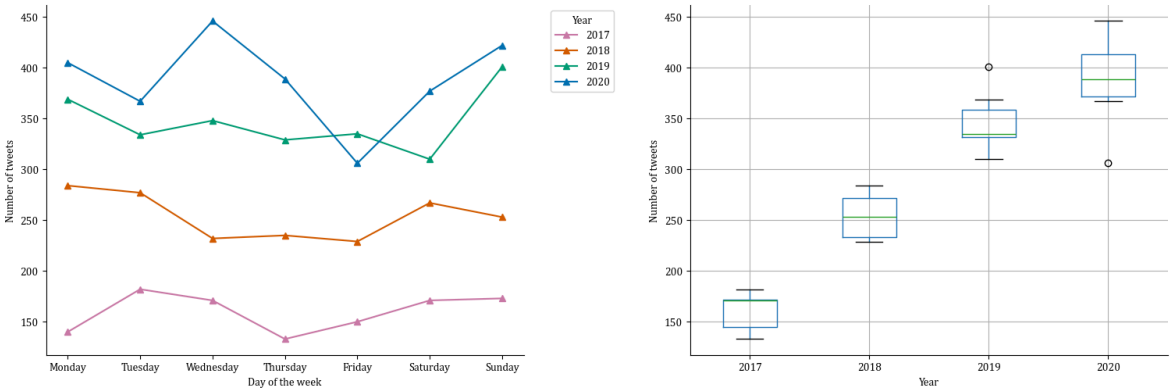


In [28]:
```
# code below is generating a box plot of the Number of tweets per day of the wee
tweets_per_day_of_week = df_trump.groupby(['day_of_week', df_trump['date'].dt.ye
tweets_per_day_of_week = tweets_per_day_of_week.reindex(['Monday', 'Tuesday', 'W
ax = tweets_per_day_of_week.boxplot()
plt.title('')
plt.xlabel('Year')
plt.ylabel('Number of tweets')
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
#plt.savefig(fig_folder + "boxplot_tweets_per_year.png")

plt.show()
```

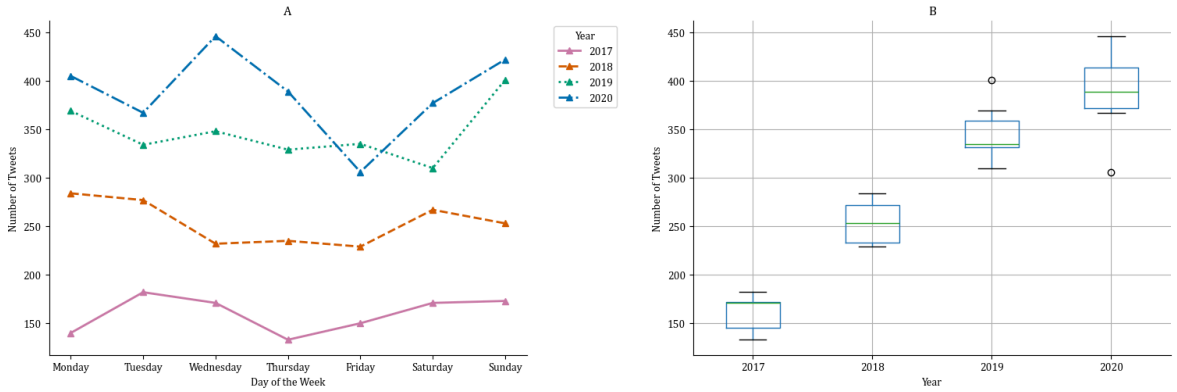


In [30]:

```
# Here I will combine the two plots made in the previous cells into one single f
tweets_per_day_of_week = df_trump.groupby(['day_of_week', df_trump['date'].dt.ye
tweets_per_day_of_week = tweets_per_day_of_week.reindex(['Monday', 'Tuesday', 'W

# plotting  - creating a figure and two subplots side by side
fig, ax = plt.subplots(1, 2, figsize=(16, 6))

# Plot 1: Line plot
tweets_per_day_of_week.plot(kind='line', marker='^', color=['#CC79A7', '#D55E00'
ax[0].set_title('')
ax[0].set_xlabel('Day of the week')
ax[0].set_ylabel('Number of tweets')
ax[0].legend(tweets_per_day_of_week.columns, title='Year', loc='upper right', bb
ax[0].spines['top'].set_visible(False)
ax[0].spines['right'].set_visible(False)

# Plot 2: Boxplot
tweets_per_day_of_week.boxplot(ax=ax[1])
ax[1].set_title('')
ax[1].set_xlabel('Year')
ax[1].set_ylabel('Number of tweets')
ax[1].spines['top'].set_visible(False)
ax[1].spines['right'].set_visible(False)
plt.tight_layout(pad=4)
#plt.savefig(fig_folder + "combined_plots.png")

plt.show()

# The number of tweets posted on each year in total
# tweets_per_year = df_trump['date'].dt.year.value_counts().sort_index(ascending
# tweets_per_year
# tweets_per_day_of_week.idxmax()
```

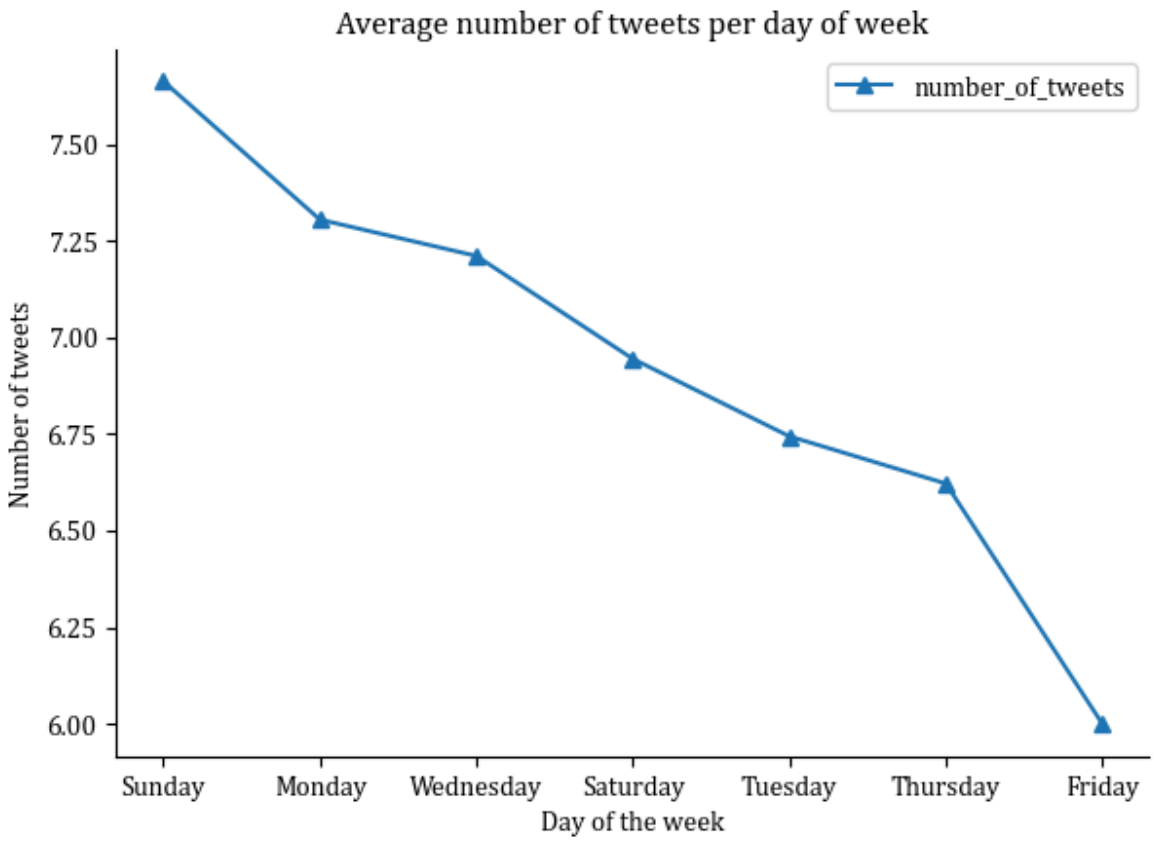


In [33]:
```
# I realized that the previous plot is not color-blindness friendly, therefore i
tweets_per_day_of_week = df_trump.groupby(['day_of_week', df_trump['date'].dt.ye
tweets_per_day_of_week = tweets_per_day_of_week.reindex(['Monday', 'Tuesday', 'W


# plotting:
fig, ax = plt.subplots(1, 2, figsize=(16, 6))
# First define line styles and colors for clarity and visibility
line_styles = ['-', '--', ':', '-.']
colors = ['#CC79A7', '#D55E00', '#009E73', '#0072B2']
# Second, ensure that each line is plotted individually with its style
for i, (column, line_style, color) in enumerate(zip(tweets_per_day_of_week.colum
    ax[0].plot(tweets_per_day_of_week.index, tweets_per_day_of_week[column], lin

ax[0].set_title('A')
ax[0].set_xlabel('Day of the Week')
ax[0].set_ylabel('Number of Tweets')
ax[0].legend(title='Year', loc='upper right', bbox_to_anchor=(1.2, 1))
ax[0].spines['top'].set_visible(False)
ax[0].spines['right'].set_visible(False)

# box plot remains the same as previous cell
tweets_per_day_of_week.boxplot(ax=ax[1])
ax[1].set_title('B')
ax[1].set_xlabel('Year')
ax[1].set_ylabel('Number of Tweets')
ax[1].spines['top'].set_visible(False)
ax[1].spines['right'].set_visible(False)
plt.tight_layout(pad=4)
#plt.savefig(fig_folder + "combined_plots_line.png")
plt.show()

#The number of tweets posted on each year in total
# tweets_per_year = df_trump['date'].dt.year.value_counts().sort_index(ascending
# tweets_per_year
# tweets_per_day_of_week.idxmax()
```



In [34]:
```
# Here plotting the average number of tweets per each individual day for each ye
df_day_of_week_per_year = df_trump.groupby(['date', 'day_of_week']).size().reset
df_day_of_week_per_year.columns = ['date', 'day_of_week', 'number_of_tweets']
df_day_of_week_per_year = df_day_of_week_per_year.groupby('day_of_week')['number
df_day_of_week_per_year = df_day_of_week_per_year.sort_values(by='number_of_twee
df_day_of_week_per_year


# making a bar plot
plt.figure(figsize=(10,10))

ax = df_day_of_week_per_year.plot(kind="line", marker = "^",  x='day_of_week', y
plt.xlabel('Day of the week')
plt.ylabel('Number of tweets')
plt.title('Average number of tweets per day of week')
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
plt.tight_layout(pad=2)
#plt.savefig(fig_folder + "av_tweet_per_day_of_week.png")
plt.show()
```

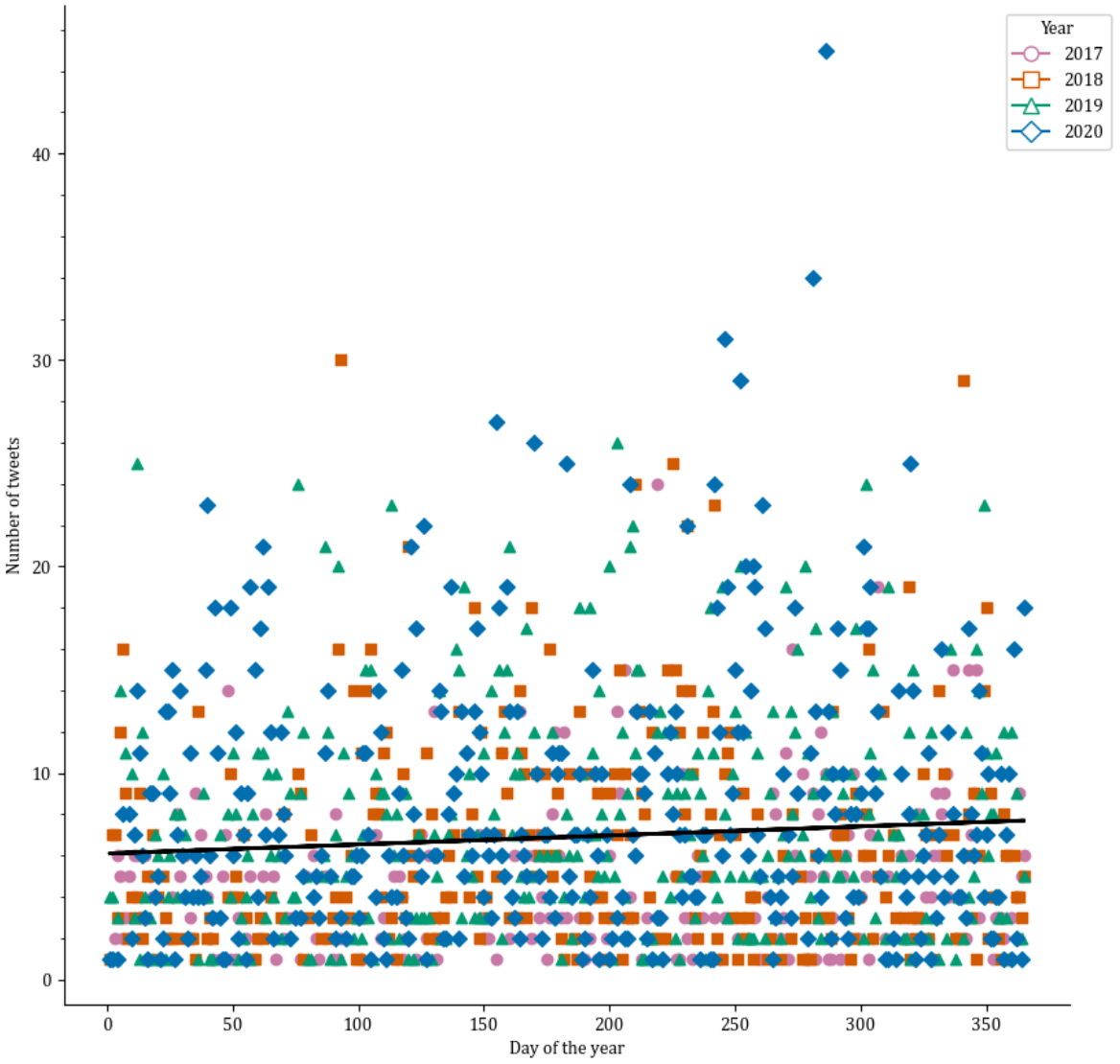
```
<Figure size 1000x1000 with 0 Axes>
```



In [41]:

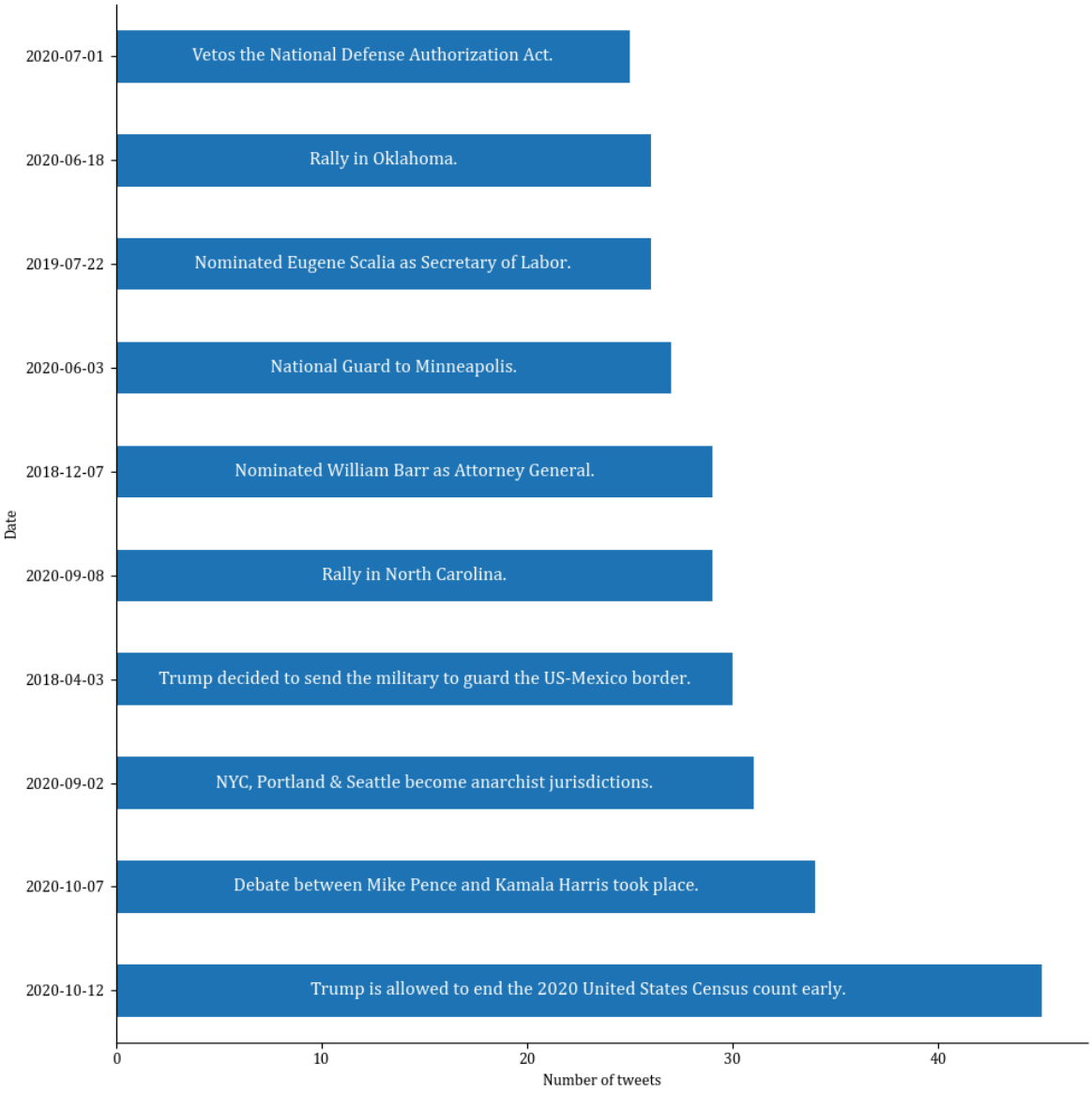
```
# trying to see if a regression model can show me a prediction of which period o
tweets_per_day = df_trump['date'].value_counts().sort_index()
# create a dataframe with the number of tweets per day
df_tweets_per_day = pd.DataFrame({'date': tweets_per_day.index, 'tweets': tweets
# create a new column with the day of the year
df_tweets_per_day['day_of_year'] = df_tweets_per_day['date'].dt.dayofyear
# fit a linear regression model to the data
model = LinearRegression()
model.fit(df_tweets_per_day[['day_of_year']], df_tweets_per_day['tweets'])
# Make predictions with the model
predictions = model.predict(df_tweets_per_day[['day_of_year']])
# Allocate a color to data points of each year
colors = np.array(['#CC79A7', '#D55E00', '#009E73', '#0072B2'])
years = np.sort(df_tweets_per_day['date'].dt.year.unique())
# using markers because the colors are not helpful for a color-blindness-friendl
markers = ['o', 's', '^', 'D', '*', 'x']
plt.figure(figsize=(10, 10))
for i, year in enumerate(years):
    yearly_data = df_tweets_per_day[df_tweets_per_day['date'].dt.year == year]
    plt.scatter(yearly_data['day_of_year'], yearly_data['tweets'], color=colors[

plt.plot(df_tweets_per_day['day_of_year'], predictions, color='black', linestyle

plt.xlabel('Day of the year')
plt.ylabel('Number of tweets')
plt.title('')
ax = plt.gca()
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.yaxis.set_major_locator(ticker.AutoLocator())
ax.yaxis.set_minor_locator(ticker.AutoMinorLocator())

#Here creating a custom legend made by chatGPT
```

```
legend_handles = [Line2D([0], [0], marker=markers[i], color=colors[i], markerfac
plt.legend(handles=legend_handles, title="Year", bbox_to_anchor=(1.05, 1))

#plt.savefig(fig_folder + "regres_markerstyle.png")

plt.show()
```

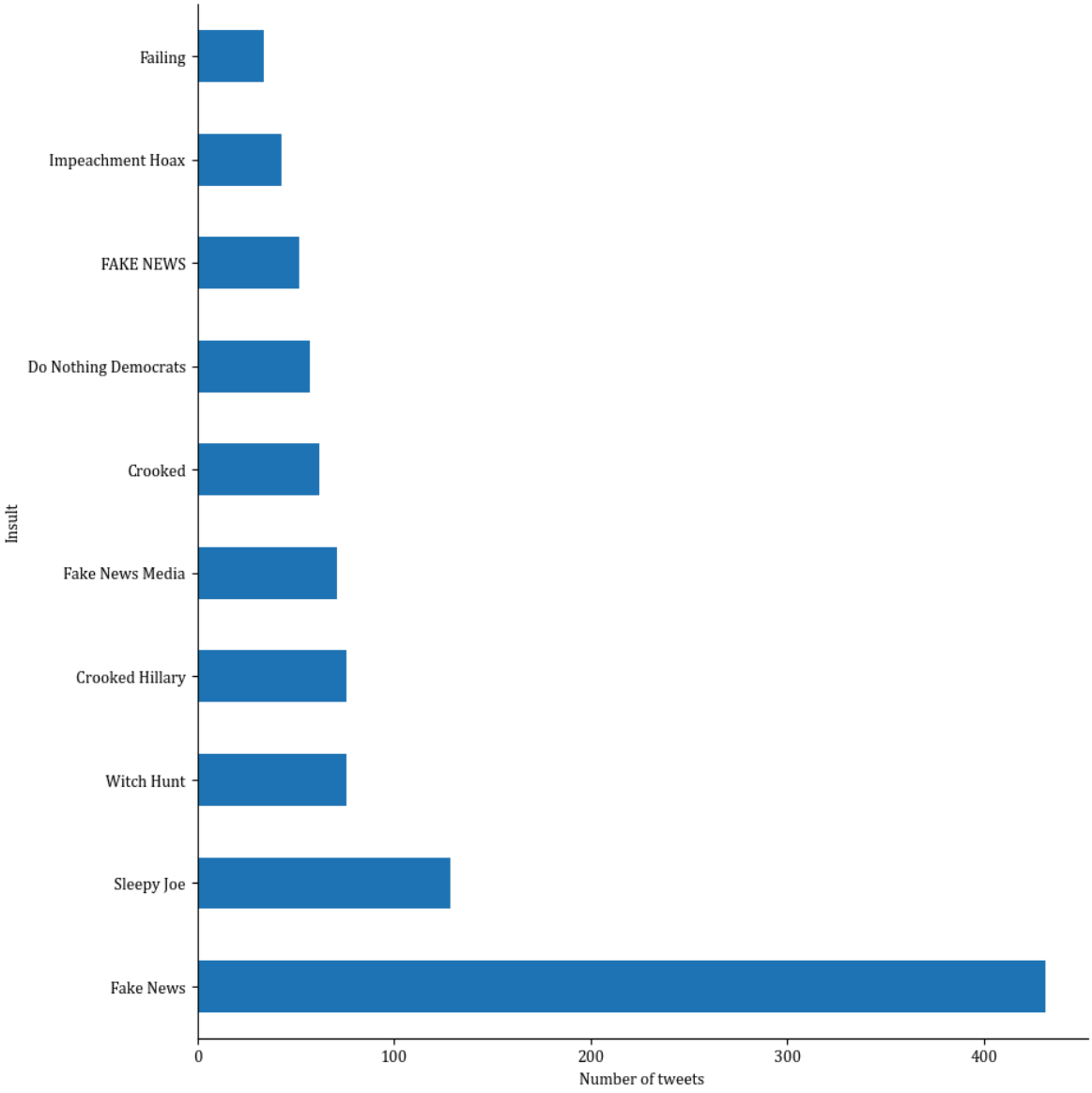


In [50]:
```
# plotting the top 10 dates with most tweets and see which major event happened


# first get the dates with the most tweets top 10
top_10_dates = df_trump['date'].value_counts().head(10).reset_index()
top_10_dates.columns = ['date', 'number_of_tweets']
top_10_dates
# find what major event happened on the top 10 dates with the most tweets in thi
major_events = [
            "Trump is allowed to end the 2020 United States Census count early.",
           "Debate between Mike Pence and Kamala Harris took place.",
           "NYC, Portland & Seattle become anarchist jurisdictions.",
           "Trump decided to send the military to guard the US-Mexico border.",
           "Rally in North Carolina.",
           "Nominated William Barr as Attorney General.",
           "National Guard to Minneapolis.",
           "Nominated Eugene Scalia as Secretary of Labor.",
           "Rally in Oklahoma.",
           "Vetos the National Defense Authorization Act."]
```

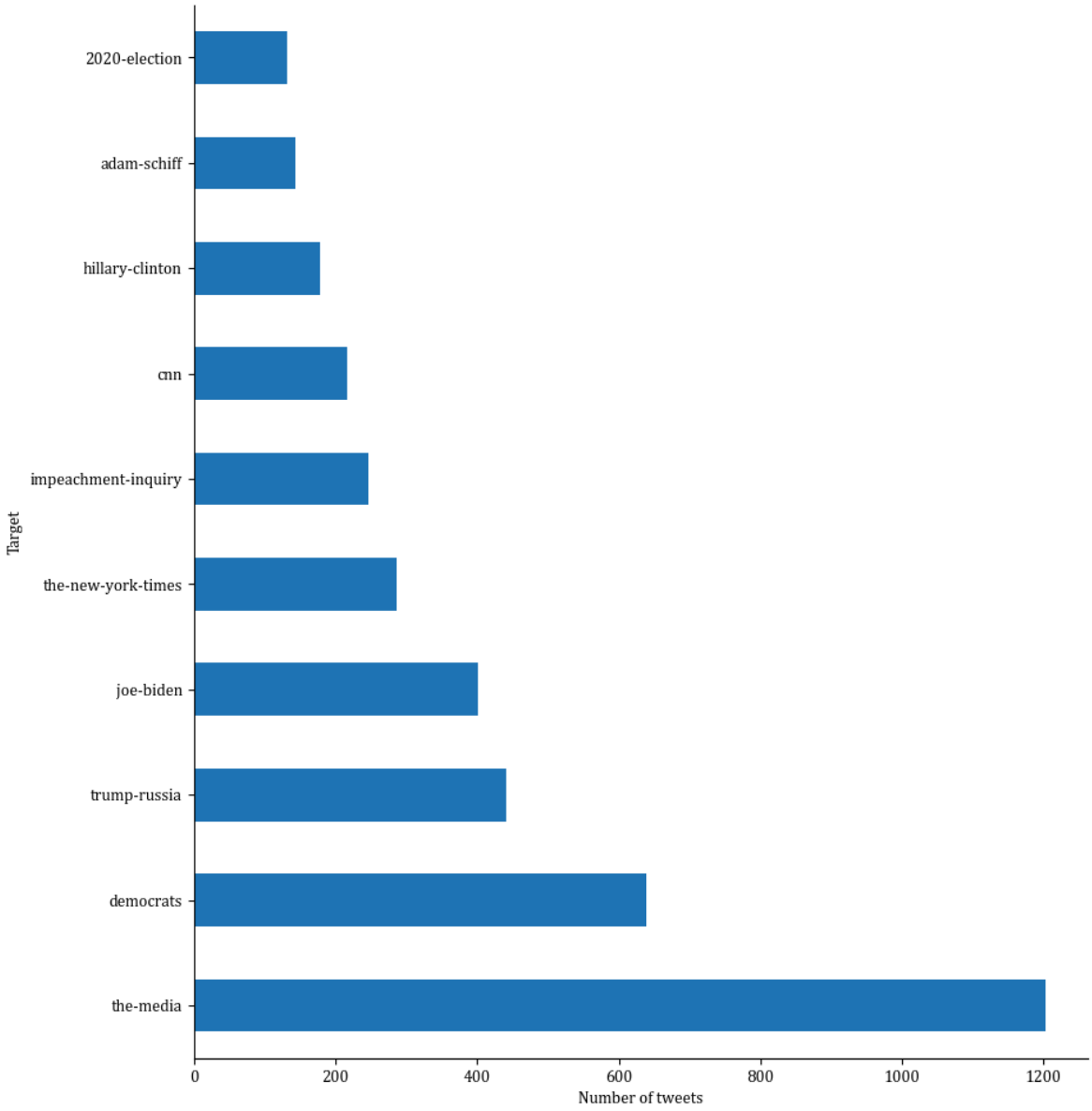
```
plt.figure(figsize=(10, 10))  # Adjusted for better visibility of bars and text
ax = df_trump['date'].value_counts().head(10).plot(kind='barh')
plt.xlabel('Number of tweets')
plt.ylabel('Date')
plt.title('')

# Since we're plotting horizontally, adjust y-tick labels to match the top 10 da
ax.set_yticklabels(top_10_dates['date'].dt.strftime('%Y-%m-%d'), rotation=0)

# Add event labels in the middle of the bars with increased font size
for i, (index, row) in enumerate(top_10_dates.iterrows()):
    # Calculate the midpoint of the bar's length
    midpoint = row['number_of_tweets'] / 2
    # Place the text in the middle of the bar with a larger font size
    ax.text(midpoint, i, f"{major_events[i]}", va='center', ha='center', fontsiz

ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)


#plt.savefig(fig_folder + "top10dates_events.png")

plt.tight_layout()
plt.show()
```



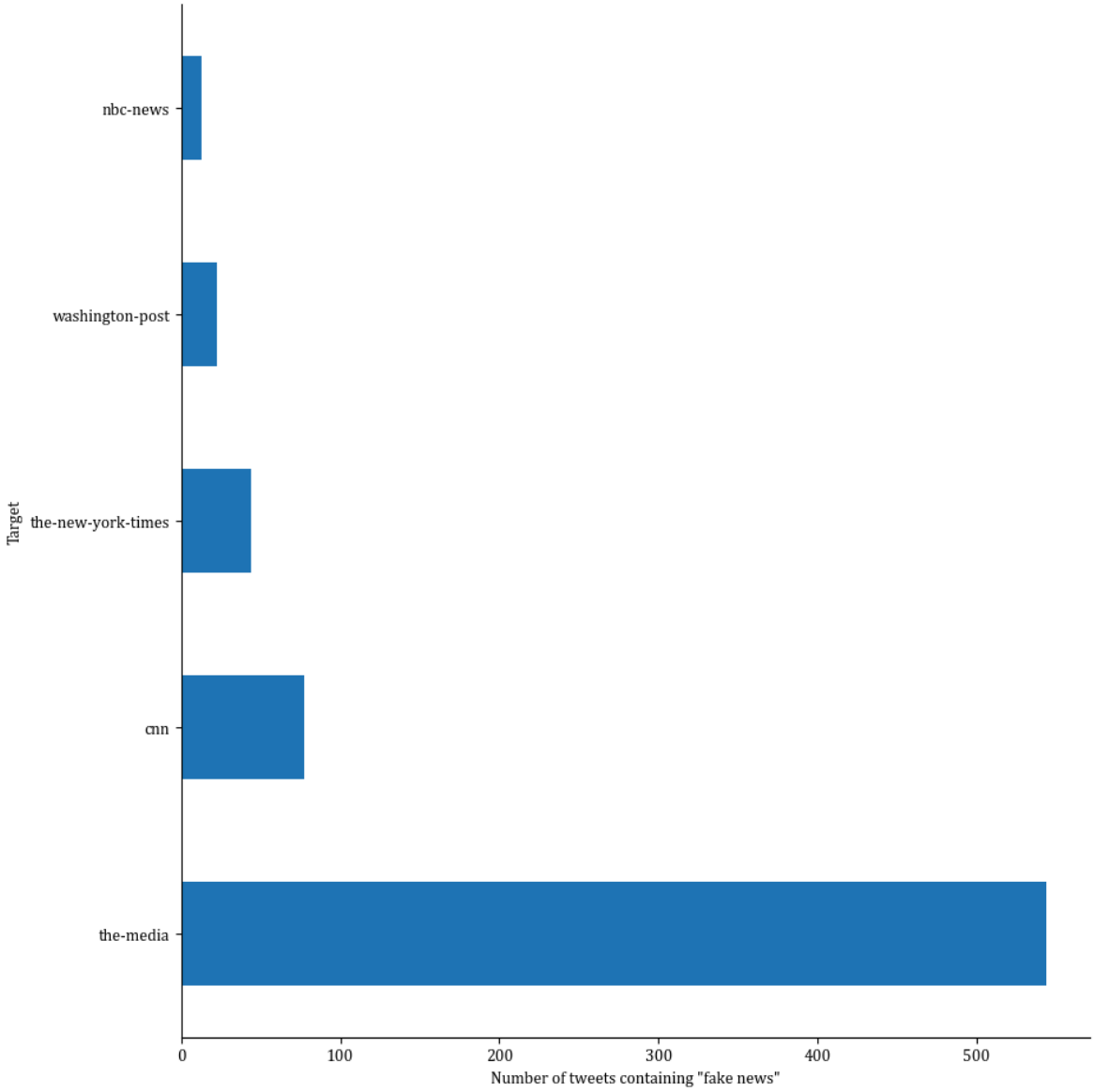
In [45]:
```
# fidning which insults repeated the most
# in df_trump, get the number of each insult , here plotting the top 10 insults
insults = df_trump['insult'].value_counts()
insults
plt.figure(figsize=(10,10))
ax = df_trump['insult'].value_counts().head(10).plot(kind="barh")

plt.xlabel('Number of tweets')
plt.ylabel('Insult')
plt.title('')

ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
plt.tight_layout(pad=3.0)
#plt.savefig(fig_folder + "top_10_insults.png")
plt.show()

# the data shows an interesting psychological effect, "Fake news" comes in diffe
#this might also be attributed to some level of anger
```



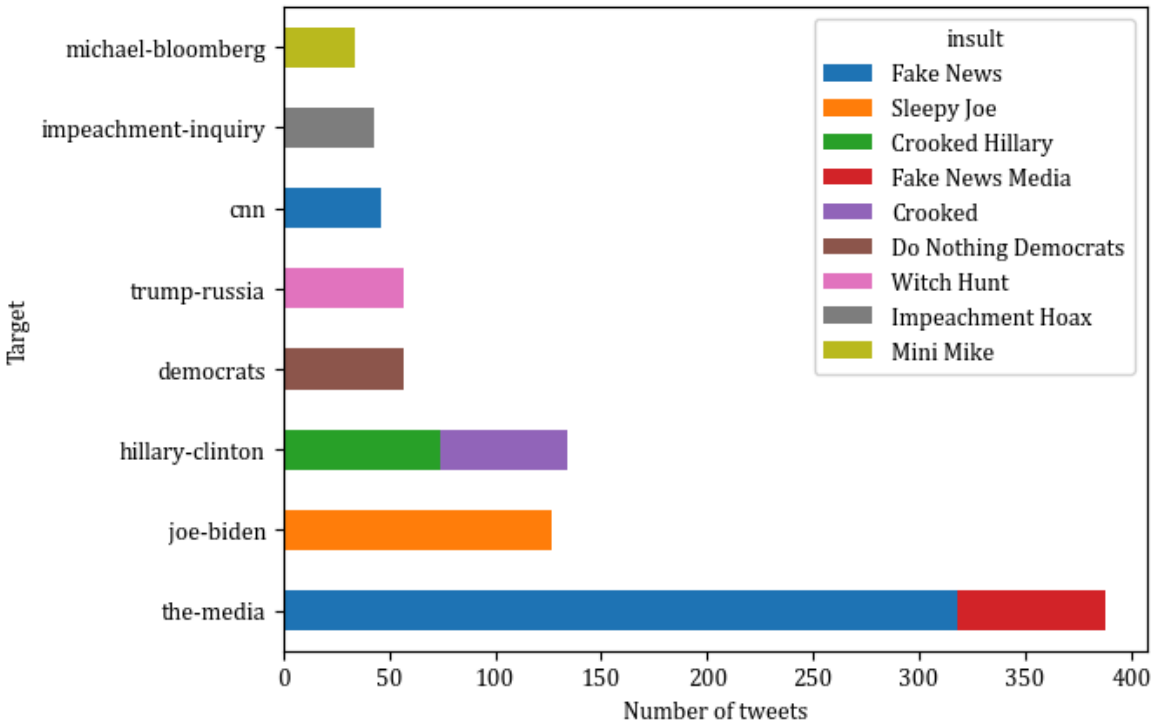
In [46]:
```
# in df_trump, get the number of each target, which are the top 10 targets of Tr
targets = df_trump['target'].value_counts()
targets

# make a barplot of the top 10 targets
plt.figure(figsize=(10,10))
ax = df_trump["target"].value_counts().head(10).plot(kind= "barh")

ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)

plt.tight_layout(pad=3.0)
plt.xlabel('Number of tweets')
plt.ylabel('Target')
plt.title('')
#plt.savefig(fig_folder + "top_10_targets.png")
plt.show()
```



In [47]:
```
# find how many times insults containing the terms but not exactly "fake news" a
fake_news = df_trump[df_trump['insult'].str.contains('fake news', case=False)]
fake_news['target'].value_counts().head(5)
# make a plot showing the number of tweets with "fake news" and their correspond
plt.figure(figsize=(10,10))
ax = fake_news['target'].value_counts().head(5).plot(kind='barh')
plt.xlabel('Number of tweets containing "fake news"')
plt.ylabel('Target')
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
plt.tight_layout(pad=3.0)
#plt.savefig(fig_folder + "which_media.png")
plt.title('')
plt.show()
```



In [48]:
```
# find another way to show the correlation between target and insult
df_trump.groupby(['target', 'insult']).size().sort_values(ascending=False).head(
plt.xlabel('Number of tweets')
plt.ylabel('Target')
plt.title('')
plt.show()

# not using this plot in the results
```



In [21]:

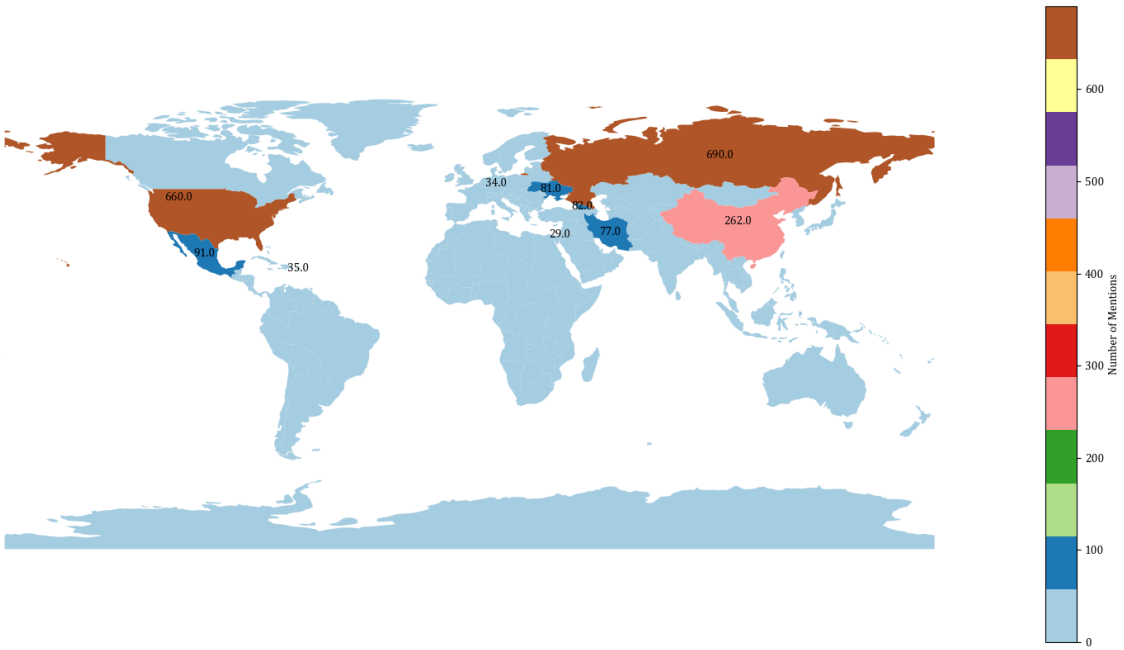
```
# With great help from ChatGPT and Copilot, creating a map showing which countri


#generate a basic list of country names from pycountry using a dictionary comper
country_names = {country.name.lower(): country.name for country in pycountry.cou

# Update the list with common names and abbreviations for specific countries - s
country_aliases = {
    **country_names,
    'iran': 'Iran',
    'us': 'United States',
    'usa': 'United States',
    'united states': 'United States',
    'uk': 'United Kingdom',
    'united kingdom': 'United Kingdom',
    'russia': 'Russia',
    # Add other common names or abbreviations for Russia if necessary
}

# Initialize a counter for the country mentions
country_mentions = Counter()

# Function to preprocess text and search for country mentions
def preprocess_and_search(text, aliases):
    # Normalize the text to lowercase and replace separators
    text = re.sub(r'[-_]', ' ', text.lower())
    for alias, official_name in aliases.items():
        if re.search(r'\b' + re.escape(alias) + r'\b', text):
            country_mentions[official_name] += 1

# Iterate over each row in the DataFrame
for _, row in df_trump.iterrows():
    # Concatenate text from relevant columns, handling NaN values
    combined_text = ' '.join(str(row[col]) for col in ['tweet', 'insult', 'targe
    preprocess_and_search(combined_text, country_aliases)

# Convert the counter to a DataFrame for easier analysis and visualization
```

```
df_country_mentions = pd.DataFrame(country_mentions.items(), columns=['Country',

#print(df_country_mentions)
```

In [22]:

```
# Load the GeoPandas world dataset
world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))

# pycountry detects specific name styles, this is just an adjustment to the name
country_name_adjustments = {
    'United States': 'United States of America',
    'UK': 'United Kingdom',
    }

# Apply the adjustments to the 'Country' column in your DataFrame
df_country_mentions['Country'] = df_country_mentions['Country'].replace(country_

# Merge the GeoDataFrame with the mentions DataFrame
world = world.merge(df_country_mentions, how='left', left_on='name', right_on='C

# Fill NaN values in the 'Mentions' column with 0
world['Mentions'] = world['Mentions'].fillna(0)

# Get the top 10 countries by mentions
top_countries = world.nlargest(10, 'Mentions')

# Plot the map
# here I am just trying to see which color pallet is the best, of course only th
colormap = 'viridis'
colormap = "plasma"
colormap = "Pastel1"
colormap = "Paired"
colormap = "Set3"
colormap = "Pastel1"
colormap = 'PiYG'
colormap = 'Set3'
colormap = "Accent"
colormap = "Paired"
fig, ax = plt.subplots(1, figsize=(20, 10))
world.plot(column='Mentions', ax=ax, legend=True,
           legend_kwds={'label': "Number of Mentions", 'orientation': "vertical"
           cmap=colormap,
           missing_kwds={"color": "lightgrey", "edgecolor": "black", "hatch": "/

# Annotate the top 10 countries with their mention counts
for idx, row in top_countries.iterrows():
    # Some countries may be too small to display properly, adjust the positionin
    plt.annotate(text=row['Mentions'], xy=(row['geometry'].centroid.x, row['geom
                 ha='center', va='center', fontsize=10, color='black')

plt.title('', fontsize=15)
ax.set_axis_off()
#plt.savefig(fig_folder + "map_pair.png")
plt.show()
```

```
C:\Users\hta031\AppData\Local\Temp\ipykernel_20864\1391050826.py:2: FutureWarnin
g: The geopandas.dataset module is deprecated and will be removed in GeoPandas 1.
0. You can get the original 'naturalearth_lowres' data from https://www.naturalea
rthdata.com/downloads/110m-cultural-vectors/.
  world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))
```



In [20]:
```
# # the pycountry package seems not able to detect all the mentions of countries

# # number of the term "iran" in df_trump "tweet", "target" and "insult" columns
# iran = df_trump['tweet'].str.contains('iran', case=False).sum() + df_trump['ta
# iran

# # number of the term "US" in df_trump "tweet", "target" and "insult" columns a
# US = df_trump['tweet'].str.contains('US', case=False).sum() + df_trump['target
# US

# # number of the term "Europe" in df_trump "tweet", "target" and "insult" colum
# europe = df_trump['tweet'].str.contains('europe', case=False).sum() + df_trump
# europe

# # make a bar plot of iran, US and europe
# plt.bar(['Iran', 'US', 'Europe'], [iran, US, europe])
# plt.xlabel('Term')
# plt.ylabel('Number of tweets')
# plt.title('')
# plt.show()
```

THIS IS THE END OF THIS SCRIPT - THANKS! 😊