Bivariate Regression

Adjusting for the variance in level of fidelity between the primary dataset- containing attacks on healthcare workers and infrastructure- and the supporting dataset- containing Russian personnel losses- a daily total was created containing wrapped healthcare workers killed in attacks. Using this as an index, a simple bivariate regression was performed to describe the relationships between Russian deaths and deaths resulting from attacks on healthcare workers, Russian deaths dependent variable. The results of that regression are below:

OLS Regression Results

Dep. Variable:	daily_increase	R-squared:	0.011
Model:	OLS Adj.	R-squared:	0.008
Method:	Least Squares	F-statistic:	3.171
Date:	Fri, 05 May 2023	Prob (F-statistic):	0.0761
Time:	19:33:42	Log-Likelihood:	-1969.9
No. Observations:	275	AIC:	3944.
Df Residuals:	273	BIC:	3951.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	405.9708	19.733	20.573	0.000	367.122	444.820
totaldeaths	33.7563	18.957	1.781	0.076	-3.563	71.076

Omnibus:	209.427	Durbin-Watson:	1.167
Prob(Omnibus):	0.000	Jarque-Bera (JB):	4411.915
Skew:	2.800	Prob(JB):	0.00
Kurtosis:	21.806	Cond. No.	1.35

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

These results provide some insight into the relationship between the dependent variable "daily_increase," a daily measure of Russian KIA, and the independent variable and "totaldeaths," a daily measure of healthcare workers killed in attacks.

Examining R-squared, the R-squared value of 0.011 indicates that only a small portion (1.1%) of the variation in the daily_increase can be explained by the totaldeaths variable. This means that **the totaldeaths variable has limited predictive power over daily_increase.**

For Adjusted R-squared, The adjusted R-squared value of 0.008 takes into account the degrees of freedom and penalizes the model for the number of predictors. It provides a slightly lower value than the R-squared and suggests that the totaldeaths variable may not significantly improve the model's fit.

For the F-statistic, the value of 3.171 and the corresponding p-value of 0.0761 indicate that **the overall regression model is not statistically significant at the conventional significance level of 0.05. This suggests that the relationship between totaldeaths and daily_increase may not be significant.**

The coefficients reveal the constant term to be 405.9708, indicating the expected value of daily_increase (Russian KIA) when the totaldeaths (Healthcare workers killed) variable is zero. **The coefficient for the totaldeaths variable is 33.7563, suggesting that, on average, an increase of one unit in totaldeaths is associated with an increase of 33.7563 units in daily_increase. However, since the p-value for this coefficient is 0.076, it is not statistically significant at the conventional significance level.**

Interpreting the p-values associated with the coefficients can provide a measure of their statistical significance, though in this case they are also unfortunately not significant. The p-value for the totaldeaths coefficient is 0.076, which is greater than 0.05. **This suggests that there is insufficient evidence to reject the null hypothesis that the coefficient is equal to zero.**

The Omnibus and Jarque-Bera tests reveal that the residuals in this case deviate significantly from a normal distribution.

The Durbin-Watson statistic of 1.167 measures the *autocorrelation* of the residuals. A value close to 2 suggests no autocorrelation. In this case, the value is less than 2, indicating the presence of positive autocorrelation.

The condition number of 1.35 suggests that there may be multicollinearity in the model.

Overall, looking at the results of this regression, the relationship between Healthcare workers killed and daily Russian KIA appears to not be statistically significant or meaningful. The low R-squared and the lack of significance in the coefficients suggest that there may be other variables that have a stronger influence on the Russian KIA.

T-Test

A T-Test was peformed between the two variables, and the results are shown below:

T-test result:
t-statistic: 21.902439984650314
p-value: 7.403475411434446e-77

T-tests provide information about the statistical significance of the difference between the "totaldeaths" and "daily_increase" variables. The t-statistic of 21.902439984650314 suggests a significant difference between the means of "totaldeaths" and "daily_increase" variables, due ton one being consistently very low (healthcare workers killed) and the other showing more variation, but being consistently quite high.

The p-value shows us the probability of observing such a significant difference (or more extreme) between the groups if there were no true difference in the population. In this case, the p-value is 7.403475411434446e-77, which is an extremely small value close to zero. This means that the difference is highly unlikely to have occurred by chance alone, and that the means are very different (as we know). Therefore, we can conclude that there is a statistically significant difference between the average values of "totaldeaths" and "daily_increase".

Correlation

A correlation was calculated between daily_increase (Russian KIA) and two combined variables representing total workers effected and total infrastructure effected. Total workers effected was calculated as ‘Health Workers Killed' + 'Health Workers Kidnapped' + Health Workers Arrested + Health Workers Injured' +'Health Workers Assaulted’. Total infrastructure effected was calculated as 'Infrastructure: Hospital'+'Infrastructure: Health Transport'+'Infrastructure: Other'. The results are below:

Correlation between daily increase and total workers effected: -0.007055612123944261
Correlation between daily increase and total infrastructure effected: -0.04895629684547883

Correlation between daily increase and total workers affected of -0.007 suggests a very weak negative correlation between daily increase and the total number of workers affected. **This means that there is almost no linear relationship between these two variables.** In other words, as the number of workers affected increases, the daily increase does not consistently decrease or increase.

Correlation between daily increase and total infrastructure affected of -0.049 suggests a weak negative correlation between daily increase and the total number of infrastructure affected. **This indicates a slight tendency for the**

daily increase to be lower when there is a higher number of infrastructure affected, but the relationship is weak.

In both cases, the correlations are close to zero, indicating that there is no strong linear relationship between daily increase and the total number of workers or infrastructure affected.

Implications of Findings

Based on the interpretation and analysis contained within this report, it is apparent that Russian personnel and equipment losses in the War in Ukraine do not drive Russian attacks on Ukrainian healthcare workers and infrastructure.

Further, the hypothesis that Russian attacks on protected targets follow events of high battlefield losses is not supported by statistical evidence in the datasets used. No attempt can be made to quantify how long the typical time lag between losses and attack is. The severity of the attack on healthcare workers and infrastructure does not appear to be scaled to the severity of the loss, as the relationship is weak between Russian KIA and Healthcare workers killed.

Some limitations of the analysis were that the level of fidelity between the datasets was not symmetrical. This frustrated fusion and enrichment of the primary dataset. Ideally, in order to perform such analysis at a high level, the same fidelity of data would be present for all types of attacks recorded. This information exists, and is recorded in real time by battlefield operating systems and sensors, but is not available open source for use in academic pursuits.

Some additional relevant variables might be meteorological data, as this has a significant effect on battlefield operations through restricted mobility and preclusion of air operations. This would be simple to obtain and enrich the primary dataset with, and is what I would do next as a future avenue of inquiry if I were to continue to develop this research topic.

Conclusion

While it is disheartening that I did not find a relationship between these two variables, it is important to note that my study was limited to the available data and may not capture the full complexity of the situation. The absence of a significant correlation suggests that factors other than Russian KIA may be driving the number of attacks on healthcare workers.

On the bright side, this exercise has sparked my interest in continued research and investigation into the dynamics of the conflict and its impact on various aspects of society, including healthcare. Protecting healthcare workers and ensuring their safety is crucial for maintaining the well-being of the population and providing essential medical services in conflict zones during times of strife.

I express my hope that the conflict in Ukraine will be resolved soon, and that peace and stability will be restored. It is my sincere wish that the full territorial sovereignty of Ukraine is restored, allowing for the rebuilding of the affected regions and the establishment of a safe and stable environment for all individuals, including healthcare workers.

Слава Україні, Героям Слава, Смерть Ворогу!

Citations

1 Ariel Cohen Dr. and Robert E. Hamilton Colonel, *The Russian Military and the Georgia War: Lessons and Implications* (US Army War College Press, 2011),https://press.armywarcollege.edu/monographs/576

2 Asal, V., Phillips, B. J., Rethemeyer, R. K., Simonelli, C., & Young, J. K. (2019). Carrots, Sticks, and Insurgent Targeting of Civilians. Journal of Conflict Resolution, 63(7), 1710–1735. https://doi.org/10.1177/0022002718789748

3 Cookman, Liz. "The Kids Aren't Alright: Kyiv says more than 16,000 Ukrainian children have been taken to Russia." Foreign Policy, 17 APR 23. https://foreignpolicy.com/2023/04/17/russia-ukraine-children-kidnapping-war-crimes-icc/

4 Doctor, Austin & Willingham, John. (2020). Foreign Fighters, Rebel Command Structure, and Civilian Targeting in Civil War. Terrorism and Political Violence. 34. 1-19. 10.1080/09546553.2020.1763320.

5 Garrett A. Cavaliere, Reem Alfalasi, Gregory N. Jasani, Gregory R. Ciottone, and Benjamin J. Lawner. Terrorist Attacks Against Healthcare Facilities: A Review. Health Security. Oct 2021.546-550.http://doi.org/10.1089/hs.2021.0004

6 Ivanuik, Petro. "2022 Ukraine Russian War." Kaggle, 2023, https://www.kaggle.com/datasets/piterfm/2022-ukraine-russian-war.

7 "The Geneva Convention of 12 August 1949" (PDF, Library of Congress). LOC.gov.

8 "UCDP Data for Ukraine." Humanitarian Data Exchange, United Nations Office for the Coordination of Humanitarian Affairs, 2023, https://data.humdata.org/dataset/ucdp-data-for-ukraine.

9 Wong CH, Chen CY-T. Ambulances under siege in Syria. BMJ Glob Health2018;3:e001003. doi:10.1136/bmjgh-2018-001003

Code

```
# # Kevin Ryan, JHU Spring '23

# # Unleashing Open Data with Python

# # Final Project - Attacks on Healthcare Workers and
Infrstructure in Ukraine



# In[ ]:




import pandas as pd



# read the Excel file into a pandas dataframe

df = pd.read_excel("2022-2023-ukraine-attacks-on-health-
care-incident-data.xlsx")



# display the first 5 rows of the dataframe to verify it
was read in correctly

print(df.head())




# ### Translate Event Descriptions from Ukrainian to
English



# In[ ]:
```

```
from googletrans import Translator


# Create a translator object

translator = Translator()


# Define a function to translate from Ukrainian to English

def translate_text(text):

    return translator.translate(text, src='uk',
dest='en').text


# Apply the function adn make a new column

df['eventDescriptionEnglishLang'] =
df['eventDescriptionUkranianLang'].apply(translate_text)


# Save the translated dataframe to a new CSV file

df.to_csv("translated_data.csv", index=False)




# ### Read in the CSV and Setup


# In[87]:




import matplotlib.pyplot as plt
```

```
df = pd.read_csv("translated_data1.csv")



# ### Force Numeric and Date columns to read properly


# In[88]:



# Convert the selected columns to numeric


numeric_cols = ['categoryHealthFacilitiesDamagedDestroyed',
'Infrastructure: Hospital',

                'Infrastructure: Health Transport',
'Infrastructure: Other',

                'HealthWorkersAttack: Health Building',
'HealthWorkersAttack: No Information',

                'HealthWorkersAttack: Everyday Activities',
'HealthWorkersAttack: Outside Health Facility']


df[numeric_cols] = df[numeric_cols].apply(pd.to_numeric,
errors='coerce')


numeric_cols = ['Number of Attacks on Health Facilities
Reporting Destruction',

                'Number of Attacks on Health Facilities
Reporting Damaged',

                'Health Workers Killed', 'Health Workers
Kidnapped',
```

```
                'Health Workers Arrested', 'Health Workers
Injured',

                'Health Workers Assaulted']


df[numeric_cols] = df[numeric_cols].apply(pd.to_numeric,
errors='coerce')




# In[89]:




# Convert ISODate to date format, drop bad columns
(Ukrainian text explanation instead of date)


df = df.apply(lambda col: pd.to_datetime(col,
errors='coerce') if col.name == 'ISODate' else col)


df = df.dropna(subset=['ISODate'])


# Reset the index

df = df.reset_index(drop=True)




# In[90]:
```

```
# Describe df


df.describe()




# ### Plot types of attack on workers over time


# In[20]:




# Plot line graph

fig, ax = plt.subplots(figsize=(10, 6))

ax.plot(df['ISODate'], df['Health Workers Killed'],
label='Killed')

ax.plot(df['ISODate'], df['Health Workers Kidnapped'],
label='Kidnapped')

ax.plot(df['ISODate'], df['Health Workers Arrested'],
label='Arrested')

ax.plot(df['ISODate'], df['Health Workers Injured'],
label='Injured')

ax.plot(df['ISODate'], df['Health Workers Assaulted'],
label='Assaulted')


# Fix graph

ax.set_xlabel('Date')

ax.set_ylabel('Number of Health Workers')
```

```
ax.set_title('Health Workers Affected by Attacks Over
Time')

ax.legend()



plt.show()




# ### Plot infrastructure facilities affected by attacks
over time



# In[21]:




from matplotlib.dates import MonthLocator, DateFormatter



# Create plot

fig, ax = plt.subplots(figsize=(10, 6))

ax.plot(df['ISODate'], df['Infrastructure: Hospital'],
label='Hospital')

ax.plot(df['ISODate'], df['Infrastructure: Health
Transport'], label='Health Transport')

ax.plot(df['ISODate'], df['Infrastructure: Other'],
label='Other')



# Label graph

ax.set_xlabel('Date')
```

```
ax.set_ylabel('Number of Infrastructure Facilities
Affected')

ax.set_title('Infrastructure Facilities Affected by Attacks
Over Time')

ax.legend()



# Format x-axis to display only the months so it isn't
crowded

months = MonthLocator()

date_format = DateFormatter('%b %Y')

ax.xaxis.set_major_locator(months)

ax.xaxis.set_major_formatter(date_format)

plt.xticks(rotation=45, ha='right')



plt.show()



#LOL




# In[23]:




# Define bins for the histogram

bins = 20
```

```
# Create histogram for each facility type

fig, ax = plt.subplots(figsize=(10, 6))

for facility in ['Infrastructure: Hospital',
'Infrastructure: Health Transport', 'Infrastructure:
Other']:

    ax.hist(df[df[facility] > 0]['ISODate'], bins=bins,
alpha=0.7, label=facility)



# Label graph

ax.set_xlabel('Date')

ax.set_ylabel('Frequency')

ax.set_title('Histogram of Infrastructure Facilities
Affected by Attacks Over Time')

ax.legend()



plt.show()





# ### Plot attacks on a Map



# In[94]:




import folium
```

```
# Read in the data and drop any rows with missing latitude
or longitude data

df = df.dropna(subset=['latitude2', 'longitude2'])



# Create a map and center it

m = folium.Map(location=[df['latitude2'].median(),
df['longitude2'].median()], zoom_start=5)



# Loop

for col in ['Health Workers Killed', 'Health Workers
Kidnapped', 'Health Workers Arrested', 'Health Workers
Injured', 'Health Workers Assaulted']:

    df_filtered = df[df[col] > 0]

    for i, row in df_filtered.iterrows():

        popup_text = f"{row['eventDescription']} ({col}:
{row[col]})"

        folium.Marker([row['latitude2'],  row['longitude2']],
popup=popup_text).add_to(m)



# Show the map

m




# ### Plot Pie Charts for Perpetrators of attack, Weapons
used to attack



# In[37]:
```

```
### FIX

# Get the counts of each type of perpetrator

counts = df['Perpetrator of Attack'].value_counts()



# Create the pie chart

fig, ax = plt.subplots(figsize=(8, 8))

wedges, _, labels = ax.pie(counts.values,
labels=counts.index, autopct='%1.1f%%', startangle=90)



# title

ax.set_title("Perpetrators of Attacks on Health Workers")



# legend

ax.legend(wedges, labels, title="Perpetrator of Attack",
loc="center left", bbox_to_anchor=(1, 0, 0.5, 1))



# Show the chart

plt.show()




# In[38]:
```

```
### Fix

# Get the counts of each type of weapon used

counts = df['Weapon Used'].value_counts()



# Create the pie chart

fig, ax = plt.subplots(figsize=(8, 8))

wedges, _, labels = ax.pie(counts.values,
labels=counts.index, autopct='%1.1f%%', startangle=90,
labeldistance=1.1)



# title

ax.set_title("Weapons Used in Attacks on Health Workers")



# legend

ax.legend(wedges, labels, title="Weapon Used", loc="center
left", bbox_to_anchor=(1, 0, 0.5, 1))



# Show the chart

plt.show()




# In[49]:




# Read in Russian Personnel losses
```

```
df1 = pd.read_csv("Russia_losses_personnel.csv")

# Drop the non-useful column
df1 = df1.drop(columns=["personnel*"])

# Fix that it is a cumulative total
df1["daily_increase"] = df1["personnel"].diff()

# Force the date to be an ISODate
df1["date"] = pd.to_datetime(df1["date"])

print(df1)

# Summary statistics for df
print('Summary statistics for df:')
print(df.describe())

# Summary statistics for df1
print('\nSummary statistics for df1:')
print(df1.describe())


# In[44]:
```

```
# Create a new column for total workers effected

df['Total workers effected'] = df['Health Workers Killed']
+ df['Health Workers Kidnapped'] + df['Health Workers
Arrested'] + df['Health Workers Injured'] + df['Health
Workers Assaulted']



# Create a new column for total infrastructure effected

df['Total infrastructure effected'] = df[['Infrastructure:
Hospital', 'Infrastructure: Health Transport',
'Infrastructure: Other']].sum(axis=1)



# Merge the daily increase column from df1 into df based on
the date column

df = pd.merge(df, df1[['date', 'daily_increase']],
how='left', left_on='ISODate', right_on='date')



# Calculate the correlation between daily increase in
personnel losses and total workers/infrastructure effected

corr_workers = df['daily_increase'].corr(df['Total workers
effected'])

corr_infra = df['daily_increase'].corr(df['Total
infrastructure effected'])



print(f"Correlation between daily increase and total
workers effected: {corr_workers}")

print(f"Correlation between daily increase and total
infrastructure effected: {corr_infra}")
```

```
# ### A weak, negative correlation exists between Russian invaders killed in action (KIA) and Attacks on Healthcare workers (-0.007055612123944261) and Attacks on Healthcare Infrastructure (-0.04895629684547883).

#


# In[51]:




# Plot daily_increase over date

plt.plot(df1['date'], df1['daily_increase'])

plt.xlabel('Date')

plt.ylabel('Daily Russians KIA')

plt.title('Daily Russian Personnel Losses')

plt.show()



# In[53]:




# Plot cumulative increase over date

plt.plot(df1['date'], df1['personnel'])

plt.xlabel('Date')

plt.ylabel('Cumulative Russians KIA')

plt.title('Cumulative Total Russian Personnel Losses')
```

```
plt.show()



# In[101]:



import pandas as pd


df2 = pd.read_csv('russia_losses_equipment.csv')




# In[102]:




# Select the relevant columns

cumulative_columns = ['aircraft', 'helicopter', 'tank',
'APC', 'field artillery', 'MRL', 'military auto', 'fuel
tank', 'drone', 'naval ship', 'anti-aircraft warfare',
'special equipment', 'mobile SRBM system']



# Compute the daily increase for each column to fix
cumulative

for col in cumulative_columns:

    df2[f'{col}_daily_increase'] = df2[col].diff()



print(df2)
```

```
# In[62]:



import matplotlib.dates as mdates


fig, ax = plt.subplots(figsize=(22,6))


plt.plot(df2['date'], df2['aircraft_daily_increase'],
color='blue', label='Aircraft')

plt.plot(df2['date'], df2['helicopter_daily_increase'],
color='orange', label='Helicopter')

plt.plot(df2['date'], df2['tank_daily_increase'],
color='green', label='Tank')

plt.plot(df2['date'], df2['APC_daily_increase'],
color='red', label='APC')

plt.plot(df2['date'], df2['field artillery_daily_increase'],
color='purple', label='Field Artillery')

plt.plot(df2['date'], df2['MRL_daily_increase'],
color='brown', label='MRL')

plt.plot(df2['date'], df2['military auto_daily_increase'],
color='pink', label='Military Auto')

plt.plot(df2['date'], df2['fuel tank_daily_increase'],
color='gray', label='Fuel Tank')

plt.plot(df2['date'], df2['drone_daily_increase'],
color='black', label='Drone')

plt.plot(df2['date'], df2['naval ship_daily_increase'],
color='red', linestyle='dashed', label='Naval Ship')
```

```
plt.plot(df2['date'], df2['anti-aircraft
warfare_daily_increase'], color='blue', linestyle='dashed',
label='Anti-aircraft Warfare')

plt.plot(df2['date'], df2['special
equipment_daily_increase'], color='orange',
linestyle='dashed', label='Special Equipment')

plt.plot(df2['date'], df2['mobile SRBM
system_daily_increase'], color='green', linestyle='dashed',
label='Mobile SRBM System')



plt.legend(loc='upper left')

plt.xlabel('Date')

plt.ylabel('Daily Increase')

plt.title('Daily Increase in Military Equipment Losses')


# Format x-axis labels to show only the month

ax.xaxis.set_major_locator(mdates.MonthLocator(interval=1))

ax.xaxis.set_major_formatter(mdates.DateFormatter('%b %Y'))


plt.tight_layout()

plt.show()




# In[68]:




# Convert 'date' column in df2
```

```
df2['date'] = pd.to_datetime(df2['date'])


# Merge the three dataframes on 'date' column

df3 = pd.merge(df, df1, on='date')

df3 = pd.merge(df3, df2, on='date')


# Set the index to 'date'

df3.set_index('date', inplace=True)




# In[75]:




print(df3.shape)

print(df3.head)




# In[77]:




# Add a column to df3 called "Total_Attacks_Workers" that
is the sum of the daily totals of the columns you specified

df3['Total_Attacks_Workers'] = df[['Health Workers Killed',
'Health Workers Kidnapped', 'Health Workers Arrested',
'Health Workers Injured', 'Health Workers
Assaulted']].sum(axis=1)
```

In[86]:

```
import numpy as np

import statsmodels.api as sm


df3 = df3.dropna()  # drop rows with NaN values


X = sm.add_constant(df3['personnel'])  # add a constant
term

y = df3['total_attacks_workers']


model = sm.OLS(y, X).fit()  # fit the OLS model


print(model.summary())  # print the model summary
```

In[91]:

```
df1.describe()
```

```
# In[92]:


df2.describe()


# In[93]:


df2.iloc[:, :15].describe()


# In[96]:


# Get the counts of each type of perpetrator
counts = df['Perpetrator of Attack'].value_counts()

# Define color map
colors = plt.cm.tab20(np.linspace(0, 1, len(counts.index)))

# Create the pie chart
```

```
fig, ax = plt.subplots(figsize=(8, 8))

wedges, _ = ax.pie(counts.values, colors=colors,
startangle=90)



# title

ax.set_title("Perpetrators of Attacks on Health Workers")



# legend

ax.legend(wedges, counts.index, title="Perpetrator of
Attack", loc="center left", bbox_to_anchor=(1, 0, 0.5, 1))



# Show the chart

plt.show()





# In[97]:




# Get the counts of each type of weapon used

counts = df['Weapon Used'].value_counts()


# Define color map

colors = plt.cm.tab20(np.linspace(0, 1, len(counts.index)))
```

```
# Create the pie chart

fig, ax = plt.subplots(figsize=(8, 8))

wedges, _ = ax.pie(counts.values, colors=colors,
startangle=90, labeldistance=1.1)



# title

ax.set_title("Weapons Used in Attacks on Health Workers")



# legend

ax.legend(wedges, counts.index, title="Weapon Used",
loc="center left", bbox_to_anchor=(1, 0, 0.5, 1))



# Show the chart

plt.show()




# In[98]:




from tabulate import tabulate



# Get the counts and percentages of each type of
perpetrator

counts = df['Perpetrator of Attack'].value_counts()

percentages = counts / counts.sum() * 100
```

```
# Create a table with the counts and percentages

table = []

for i, index in enumerate(counts.index):

    table.append([index, counts[i],
f'{percentages[i]:.1f}%'])



# Print the table

print(tabulate(table, headers=['Perpetrator of Attack',
'Total', 'Percentage']))




# In[99]:




from tabulate import tabulate



# Get the counts and percentages of each type of weapon
used

counts = df['Weapon Used'].value_counts()

percentages = counts / counts.sum() * 100



# Create a table with the counts and percentages

table = []

for i, index in enumerate(counts.index):
```

```
    table.append([index, counts[i],
f'{percentages[i]:.1f}%'])


# Print the table

print(tabulate(table, headers=['Weapon Used', 'Total',
'Percentage']))




# In[108]:




import matplotlib.dates as mdates


# It never reads right

df2['date'] = pd.to_datetime(df2['date'])


# Clippity doooohh daaaaaahhhh

df2['APC_daily_increase'] =
df2['APC_daily_increase'].clip(lower=0)


# The longest

fig, ax = plt.subplots(figsize=(22,6))


plt.plot(df2['date'], df2['aircraft_daily_increase'],
color='blue', label='Aircraft')
```

```
plt.plot(df2['date'], df2['helicopter_daily_increase'],
color='orange', label='Helicopter')

plt.plot(df2['date'], df2['tank_daily_increase'],
color='green', label='Tank')

plt.plot(df2['date'], df2['APC_daily_increase'],
color='red', label='APC')

plt.plot(df2['date'], df2['field artillery_daily_increase'],
color='purple', label='Field Artillery')

plt.plot(df2['date'], df2['MRL_daily_increase'],
color='brown', label='MRL')

plt.plot(df2['date'], df2['military auto_daily_increase'],
color='pink', label='Military Auto')

plt.plot(df2['date'], df2['fuel tank_daily_increase'],
color='gray', label='Fuel Tank')

plt.plot(df2['date'], df2['drone_daily_increase'],
color='black', label='Drone')

plt.plot(df2['date'], df2['naval ship_daily_increase'],
color='red', linestyle='dashed', label='Naval Ship')

plt.plot(df2['date'], df2['anti-aircraft
warfare_daily_increase'], color='blue', linestyle='dashed',
label='Anti-aircraft Warfare')

plt.plot(df2['date'], df2['special
equipment_daily_increase'], color='orange',
linestyle='dashed', label='Special Equipment')

plt.plot(df2['date'], df2['mobile SRBM
system_daily_increase'], color='green', linestyle='dashed',
label='Mobile SRBM System')



plt.legend(loc='upper left')

plt.xlabel('Date')

plt.ylabel('Daily Increase')

plt.title('Daily Increase in Military Equipment Losses')
```

```
# Format x-axis labels to show only the month

ax.xaxis.set_major_locator(mdates.MonthLocator(interval=1))

ax.xaxis.set_major_formatter(mdates.DateFormatter('%b %Y'))


plt.tight_layout()

plt.show()




# In[109]:




import folium


dff = df.dropna(subset=['latitude2', 'longitude2',
'ISODate'])


# Define the color scale

color_scale = ['blue', 'green', 'orange', 'red']


# Create a map and center it

m = folium.Map(location=[dff['latitude2'].median(),
dff['longitude2'].median()], zoom_start=5)


# Loop
```

```
for col in ['Health Workers Killed', 'Health Workers
Kidnapped', 'Health Workers Arrested', 'Health Workers
Injured', 'Health Workers Assaulted']:

    dff_filtered = dff[dff[col] > 0]

    for i, row in dff_filtered.iterrows():

        popup_text = f"{row['eventDescription']} ({col}:
{row[col]})"

        folium.Marker([row['latitude2'],  row['longitude2']],
popup=popup_text,
icon=folium.Icon(color=color_scale[int(row['ISODate'][-
4:])%4])).add_to(m)



# Show the map

m




# In[112]:




## I struck out here, but I was trying to apply a color
scale

## across time to the map plots to depict the shift
eastward in combat operations



import pandas as pd

import folium

import matplotlib.pyplot as plt

import matplotlib.colors as mcolors
```

```
# Read in data and drop any rows with missing latitude or
longitude

df = df.dropna(subset=['latitude2', 'longitude2'])



# Create a map and center it

m = folium.Map(location=[df['latitude2'].median(),
df['longitude2'].median()], zoom_start=5)



# Define the starting and ending dates

start_date = pd.Timestamp('2022-02-01')

end_date = pd.Timestamp('2023-03-31')



# Convert Timestamp objects to numeric values

start_value = start_date.timestamp()

end_value = end_date.timestamp()



# Loop

for col in ['Health Workers Killed', 'Health Workers
Kidnapped', 'Health Workers Arrested', 'Health Workers
Injured', 'Health Workers Assaulted']:

    df_filtered = df[df[col] > 0]

    for i, row in df_filtered.iterrows():

        popup_text = f"{row['eventDescription']} ({col}:
{row[col]})"

        time_value = row['ISODate'].timestamp()
```

```
        color = mcolors.rgb2hex(plt.cm.RdYlBu((time_value -
start_value) / (end_value - start_value)))

        folium.Marker([row['latitude2'],  row['longitude2']],
popup=popup_text, icon=folium.Icon(color=color)).add_to(m)


# Show the map

m





# In[116]:




from datetime import datetime


# Read the CSV file

df_math = pd.read_csv("HCK math.csv")


# Make it read correctly by coercing date

df_math["groupid"] = pd.to_datetime(df_math["groupid"])


from scipy.stats import ttest_ind


# Selecting the relevant columns from df1 and df_math

df1_selected = df1[["date", "daily_increase"]]
```

```
df_math_selected = df_math[["groupid", "totaldeaths"]]


# Merging the selected dataframes on "groupid"

merged_df = pd.merge(df1_selected, df_math_selected,
left_on="date", right_on="groupid")


# Dropping rows with NaNs

merged_df = merged_df.dropna()


# Perform the T-test

result = ttest_ind(merged_df["daily_increase"],
merged_df["totaldeaths"])


# Print the result

print("T-test result:")

print("t-statistic:", result.statistic)

print("p-value:", result.pvalue)





# In[117]:



import statsmodels.api as sm
```

```
# Define the variables

X = merged_df['daily_increase']

y = merged_df['totaldeaths']


# Adding a constant to the IV

X = sm.add_constant(X)


# Creating and fitting the model

model = sm.OLS(y, X)

results = model.fit()


# Printing the results

print(results.summary())




# In[118]:




# Selecting the relevant columns from df1 and df

df1_selected = df1["daily_increase"]

df_selected =
df[["categoryHealthFacilitiesDamagedDestroyed",
"Infrastructure: Hospital", "Infrastructure: Health
Transport", "Infrastructure: Other", "HealthWorkersAttack:
Health Building", "HealthWorkersAttack: No Information",
"HealthWorkersAttack: Everyday Activities",
"HealthWorkersAttack: Outside Health Facility", "Number of
```

```
Attacks on Health Facilities Reporting Destruction",
"Number of Attacks on Health Facilities Reporting Damaged",
"Health Workers Killed", "Health Workers Kidnapped",
"Health Workers Arrested", "Health Workers Injured",
"Health Workers Assaulted"]]


##### Does not work because the indexes can't line up


# Something about adding a constant might help?

df_selected = sm.add_constant(df_selected)


# Fit the multivariate regression model

model = sm.OLS(df1_selected, df_selected)

results = model.fit()


# Print the summary

print(results.summary())




# In[120]:




# Define the variables

X = merged_df['totaldeaths']

y = merged_df['daily_increase']
```

```
# Adding a constant to the independent variable
X = sm.add_constant(X)

# Creating and fitting the model
model = sm.OLS(y, X)
results = model.fit()

# Print the results
print(results.summary())


# In[ ]:
```