Working With Data: MTHM501

Final Assessment

Declaration of AI Assistance: I have used OpenAI's ChatGPT tool in creating this report.

AI-supported/AI-integrated use is permitted in this assessment. I acknowledge the following uses of GenAI tools in this assessment:

1. I have used GenAI tools for developing ideas.
2. I have used GenAI tools to help me understand key theories and concepts.
3. I have used GenAI tools to identify trends and themes as part of my data analysis.
4. I have used GenAI tools to check and correct my code.
5. I have used GenAI tools to suggest a plan or structure for my assessment.
6. I have used GenAI tools to proofread and correct grammar or spelling errors.
7. I have used GenAI tools to give me feedback on a draft.

I declare that I have referenced use of GenAI outputs within my assessment in line with the University referencing guidelines.

To what extent can spatial and temporal patterns accurately classify whether a fire is a war-related fire in Ukraine?

Introduction:

The conflict between Ukraine and Russia escalated sharply on February 24, 2022, when Russia launched a full-scale invasion, intensifying hostilities that began with the 2014 annexation of Crimea and the rise of separatist movements in Eastern Ukraine. Since then, over 30,000 civilians have been killed or injured, with 3.7 million displaced internally and 6.5 million fleeing the country, marking one of Europe's most severe humanitarian crises. Beyond the human toll, widespread attacks on military and civilian areas have caused massive infrastructure damage. Hundreds of fires, many linked to the conflict, occur weekly. Accurate classification of these incidents is essential for emergency response, assessing the conflict's impact, and aiding long-term reconstruction.

Research Objectives

The primary objective of this analysis is to determine if spatial and temporal patterns can be used to classify fires in Ukraine as war-related. I will explore three sub questions:

1. Are certain regions or seasons more related with war-related fires?
2. Can factors like population density and sustained excess be reliable predictors for classifying fires as war-related?
3. Which patterns provide the most accurate and reliable predictions to assess a fire's classification?

Hypotheses

Null Hypothesis (H0): *Spatial and temporal factors do not influence the classification of fires as war-related.*
Alternative Hypothesis (H1): *Spatial and temporal factors are significant predictors of war-related fires.*

Report Structure

This report includes a section on hypothesis testing, followed by exploratory data analysis, detailed hierarchical model results, a discussion of key findings, implications and limitations, and a conclusion.

Data:

The primary dataset used for this analysis, 'ukraine_fires.csv,' was sourced from Kaggle and created by The Economist's Ukraine War-fire model. It includes over 60,000 observations with 30 parameters, capturing geospatial ('latitude', 'longitude'), temporal ('date', 'AQC time'), and contextual variables ('pop_density', 'sustain_excess'). Additionally, a GeoJSON file with regional mapping of Ukraine was merged, allowing for refined geospatial analysis.

Data Wrangling and Preprocessing

Can spatial, temporal, and contextual variables accurately predict whether a fire event is war-related in Ukraine? With this broader question in mind, I begun wrangling the data.

Data Cleaning: I began with tidyverse and dplyr packages for initial cleaning, using head(), str(), and summary() to assess the dataset. Using Amelia's missmap(), I identified that the 'city' parameter was entirely null, leading me to remove both 'city' and 'year' parameters, along with other irrelevant fields. I checked for duplicates and refined the dataset by removing eleven additional unnecessary parameters and renaming the remaining for clarity.

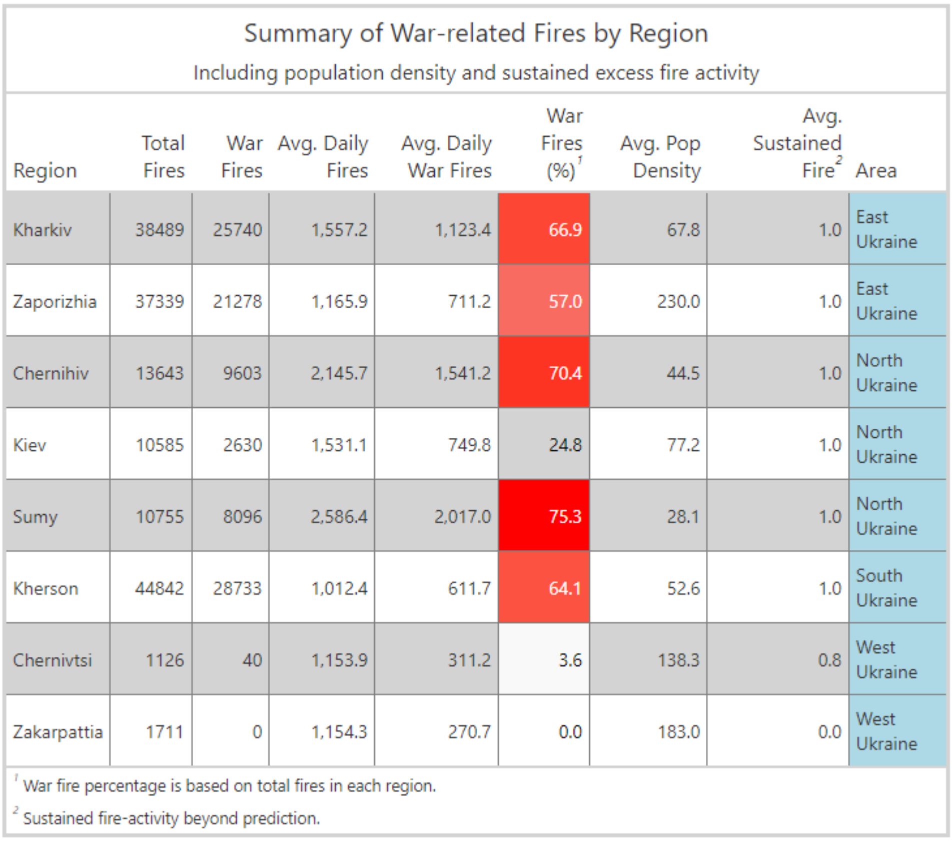
Data Formatting: To reformat dates, I applied lubridate, creating a 'season' variable by extracting the month and grouping the 'time_of_day' variable into morning, afternoon, evening, or night. I also converted the key categorical variable, 'war_fires,' into a factor to facilitate analysis.

Geospatial Context: Using the sf package, I prepared to merge the dataset with Ukraine's regional map. After validating the spatial variables and converting the dataset into an sf object, I merged the regions column and set region and season as factors to enable random effects modelling.

Exploratory Data Analysis

Summary Table of War-Fires by Region and Area

Table 1.



Summary of War-related Fires by Region

Including population density and sustained excess fire activity

Region	Total Fires	War Fires	Avg. Daily Fires	Avg. Daily War Fires	War Fires (%)[1]	Avg. Pop Density	Avg. Sustained Fire[2]	Area
Kharkiv	38489	25740	1,557.2	1,123.4	66.9	67.8	1.0	East Ukraine
Zaporizhia	37339	21278	1,165.9	711.2	57.0	230.0	1.0	East Ukraine
Chernihiv	13643	9603	2,145.7	1,541.2	70.4	44.5	1.0	North Ukraine
Kiev	10585	2630	1,531.1	749.8	24.8	77.2	1.0	North Ukraine
Sumy	10755	8096	2,586.4	2,017.0	75.3	28.1	1.0	North Ukraine
Kherson	44842	28733	1,012.4	611.7	64.1	52.6	1.0	South Ukraine
Chernivtsi	1126	40	1,153.9	311.2	3.6	138.3	0.8	West Ukraine
Zakarpattia	1711	0	1,154.3	270.7	0.0	183.0	0.0	West Ukraine

[1] War fire percentage is based on total fires in each region.

[2] Sustained fire-activity beyond prediction.

The "Summary of War-related Fires by Region" table reveals clear regional patterns in war-related fire activity. Sumy and Chernihiv show the highest concentrations (East and

Northern), with war fires comprising 75.3% and 70.4% of total fires, indicating intense conflict impact. In contrast, Kiev and Zaporizhia have lower proportions of war-related fires at 24.8% and 57%, suggesting varied conflict exposure. Western regions like Chernivtsi and Zakarpattia show minimal or no war-related fires, likely due to their distance from conflict zones.

These findings indicating spatial factors significantly affect the likelihood of a fire being a war-related fire. Data handling used dplyr for summarization and gt for table styling.

Spatial Visualisation of War-related Fires

Figure 1.

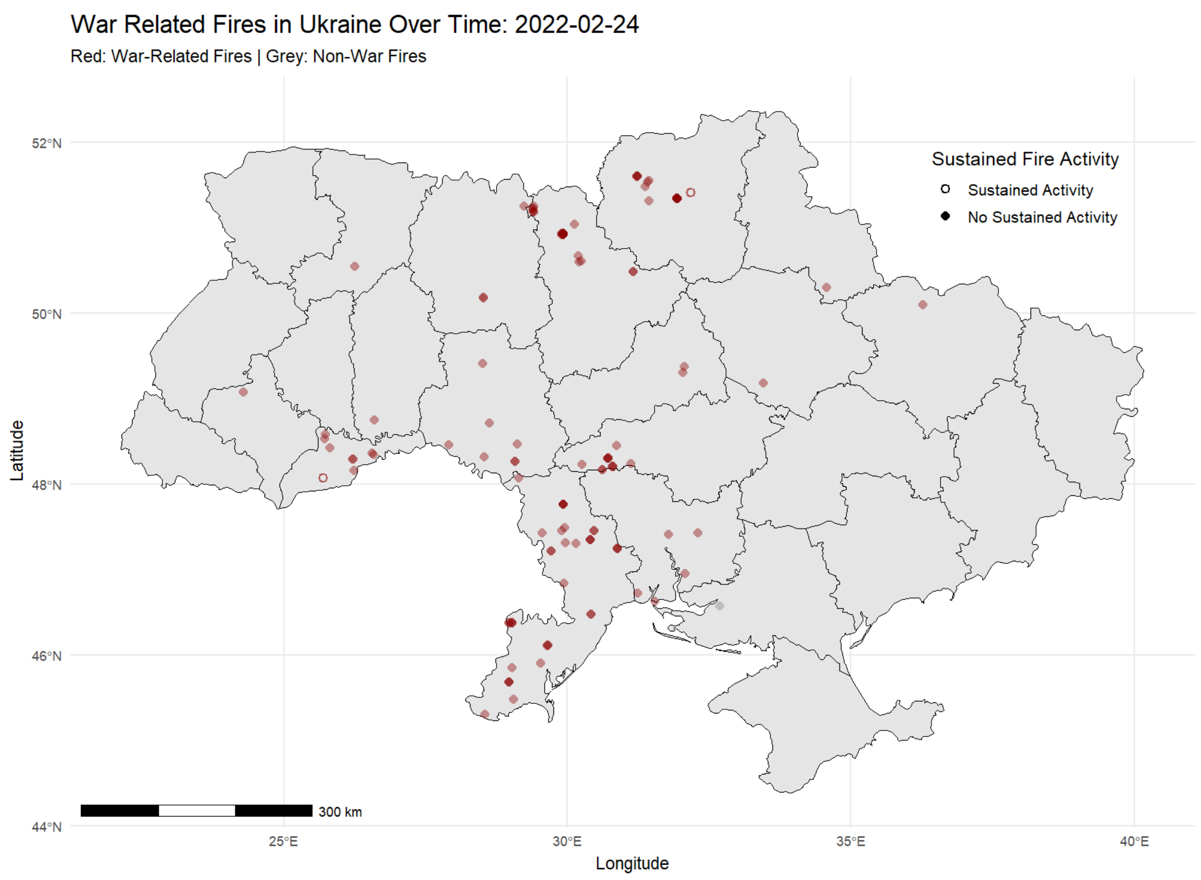


Figure 1(gif), generated using gganimate and ggplot2, depicts the distribution of war-related fires across Ukraine from the start of the war in February 2022 to October 2024. Each red mark represents a war-related fire event, while grey denotes non-war fires, allowing for clear visual differentiation. I used ggspatial to add a distance scale and ggshadow to mark past events. These packages combined allow enable advanced visualisation, and deeper insight.

As shown in Figure 1, we can see a high concentration of war-related fires around 100km from Russia's border, especially in Eastern Ukraine, with much lower activity as you move west. Temporally, marked surges in mid-2023 and throughout 2024 suggest recurring conflict intensity patterns. These spatial and temporal trends support rejecting the null hypothesis, confirming their significance in predicting war-related fires. The next analysis will cover the visible seasonal patterns. (Find the animated gif on my GitHub – linked in appendix)

Seasonal Trend Plot with War-related fires, Showing Peaks in Mid-2023 and Late-2024.

Figure 2.

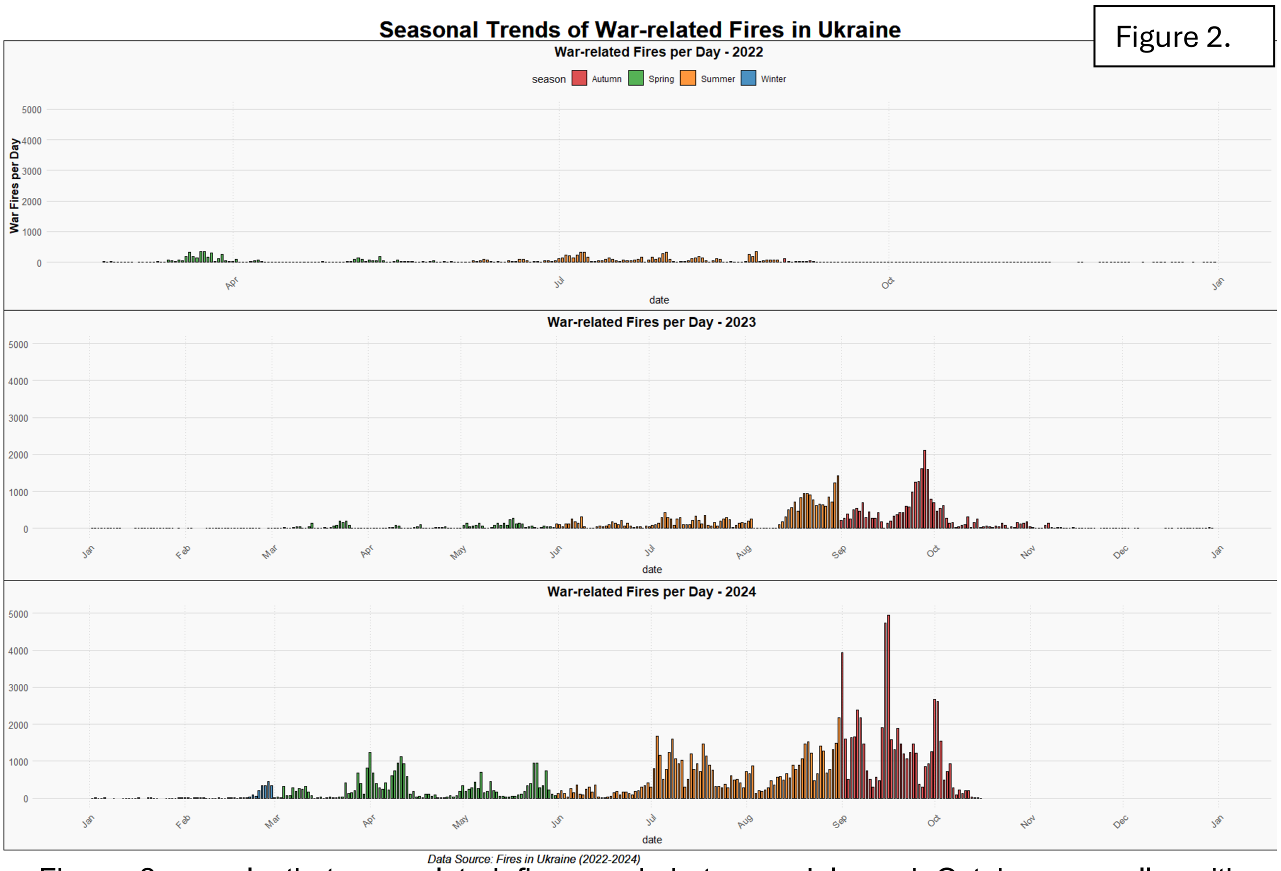


Figure 2 reveals that war-related fires peak between July and October annually, with noticeable increases each successive year. Reaching 2000 daily fires 2023, and 5000 2024. There is a consistent influx of war-related fires at the beginning of spring (April), suggesting higher likelihoods of conflict-related fire incidents during late Spring, Summer and Autum.. Another temporal pattern we see, the total amount of war-fires event is increasing year on year, indicating that as time passes, the probability of a fire being a war-related fire is higher.

The seasonal trends plot would support you to reject the null hypothesis as it illustrates how temporal factors, specifically seasonal shifts, significantly correlate with the frequency of war-related fires, indicating that certain seasons exhibit heightened conflict activity.

Figure 2 displays war-related fires per day over time, with distinctive colours representing each seasons. I adjusted the war_fire_per_day variable, using the summarise and mean function, then grouping it by date and season. Next, I extracted the yearly data from date with lubridate. I utilised ggplot2 with grid, gridExtra, and dplyr for structuring. Putting the plots together provide us with greater insight into temporal patterns.

Hypothesis Testing

Hierarchical Modelling

Hierarchical modelling will be used, as it allows us to account for region-specific and seasonal differences in fire activity, capturing the nested nature of spatial and temporal data. Since my outcome variable is binary, I decided to use Generalised Linear Mixed Models (GLMM) and specifically using the logistic regression link, R's lme4: :glmer, methods not covered in the course. As my outcome variable is binary, this improves the models fit, robustness and insights. The model includes *population density* and *sustained excess fire* as my fixed effects, with regional(25 levels), seasonal(4 levels), and coordinate-based clustering(106 levels) as random effects. I believe these this model, gives us a good approximation whilst staying relatively simple.

Model Setup:

I used `str()` to confirm variables were in the correct format to optimise the model. I converted random effect parameters to factors, and log-transformed, then standardised `population_density` to improve interpretability, model convergence, and reduce multicollinearity.

Model Specification

1. **Random Intercept Mode (RIM)**: *glmer(war_fire ~ population_density + sustained_excess_fires + (1 | region) + (1 | season) + (1 | id_big), data = TestData1, family = binomial)*

Assumes varying intercepts for regions and seasons.

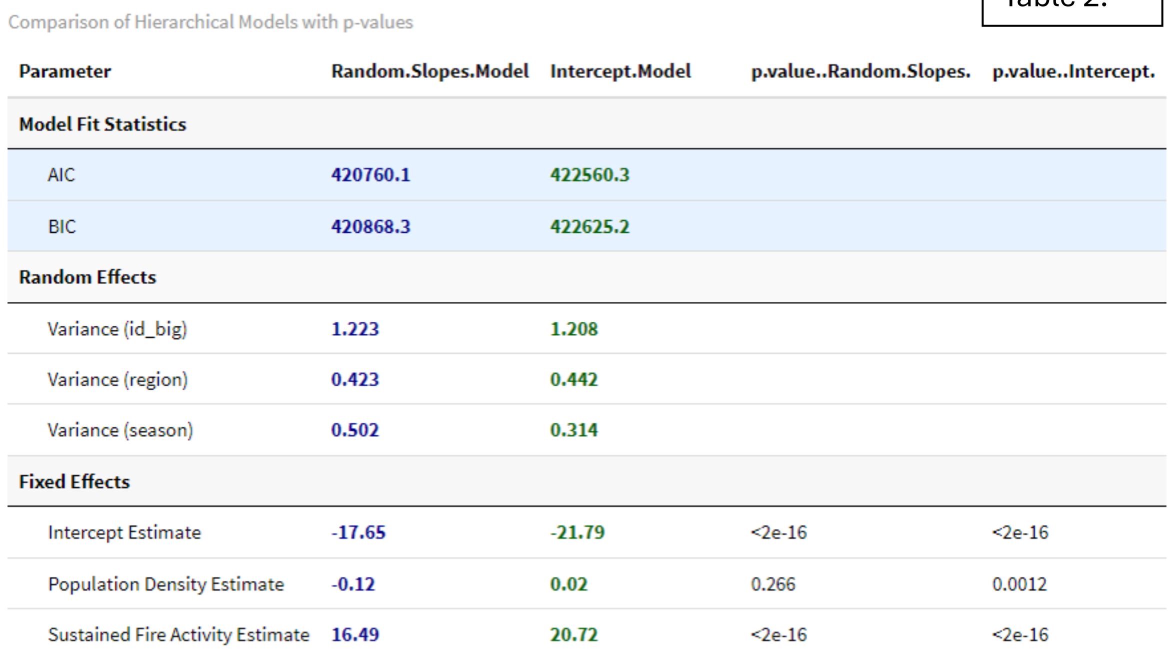
2. **Random Slopes Model (RSM)**: *glmer(war_fire ~ population_density + sustained_excess_fires + (1 + population_density | region) + (1 + sustained_excess_fires | season) + (1 | id_big), data = TestData1, family = binomial)*

Allows varying intercepts and slopes, accounting for variations in population density and sustained fire activity across regions and seasons. This model was expected to provide better insights given the anticipated regional and seasonal fluctuations in fire occurrences.

Results

Model Comparison and Analysis

Table 2.



Comparison of Hierarchical Models with p-values

Parameter	Random.Slopes.Model	Intercept.Model	p.value..Random.Slopes.	p.value..Intercept.
Model Fit Statistics				
AIC	420760.1	422560.3		
BIC	420868.3	422625.2		
Random Effects				
Variance (id_big)	1.223	1.208		
Variance (region)	0.423	0.442		
Variance (season)	0.502	0.314		
Fixed Effects				
Intercept Estimate	-17.65	-21.79	<2e-16	<2e-16
Population Density Estimate	-0.12	0.02	0.266	0.0012
Sustained Fire Activity Estimate	16.49	20.72	<2e-16	<2e-16

Table 2 shows that the RSM has a slightly lower AIC and BIC, indicating the model is a better fit. The Sustained Fire Activity Estimate remains highly significant across both models ($p < 2e-16$). Population Density is only significant in the RIM ($p = 0.0012$), suggesting that its predictive effect may vary when accounting for random slopes. We will cover the random effect variances later in the analysis.

Overall, the RSM provides a nuanced approach, supporting the alternative hypothesis that spatial and temporal factors significantly predict war-related fires. This table was formatted using knitr and kableExtra, and the modelling was conducted with lme4 in R.

Model Diagnostics

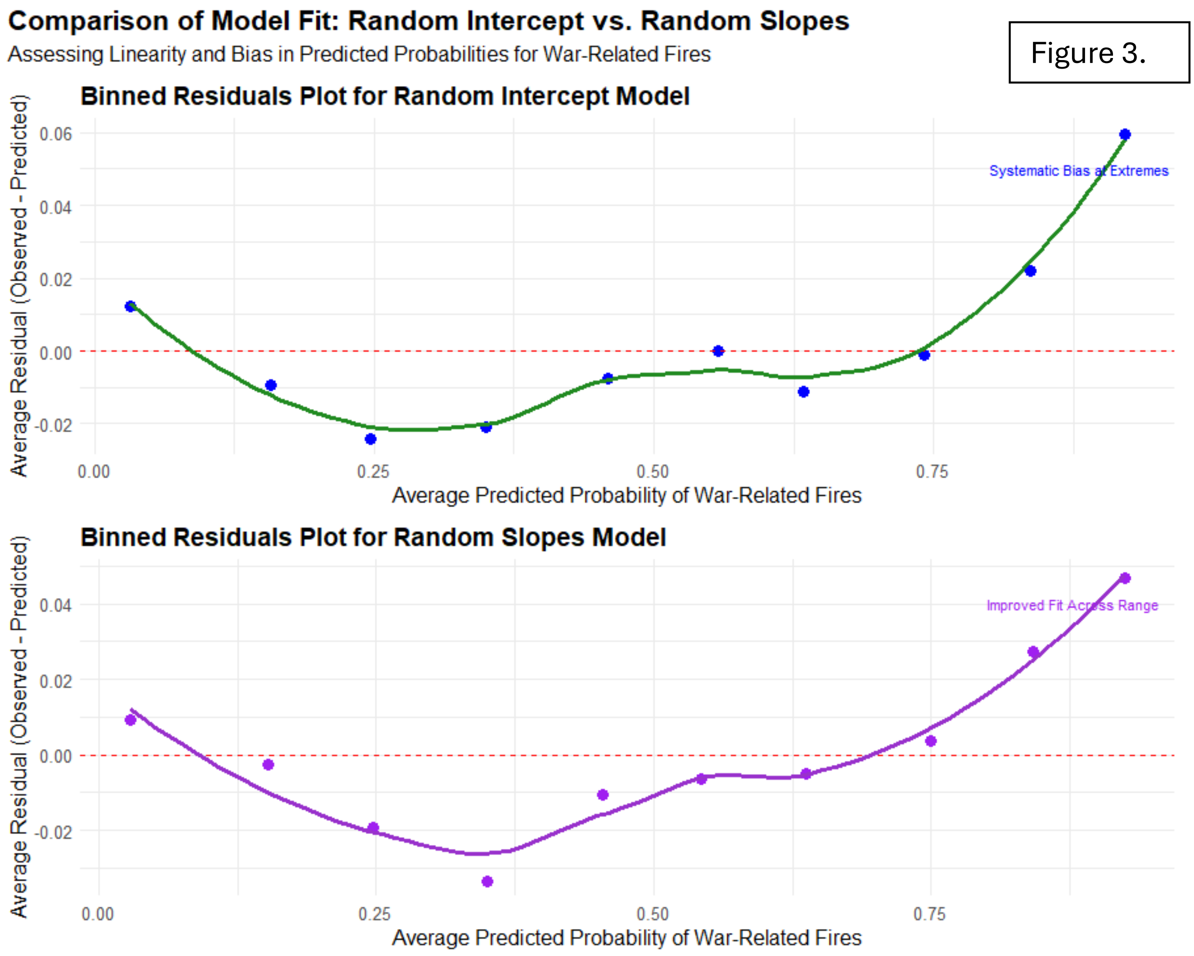


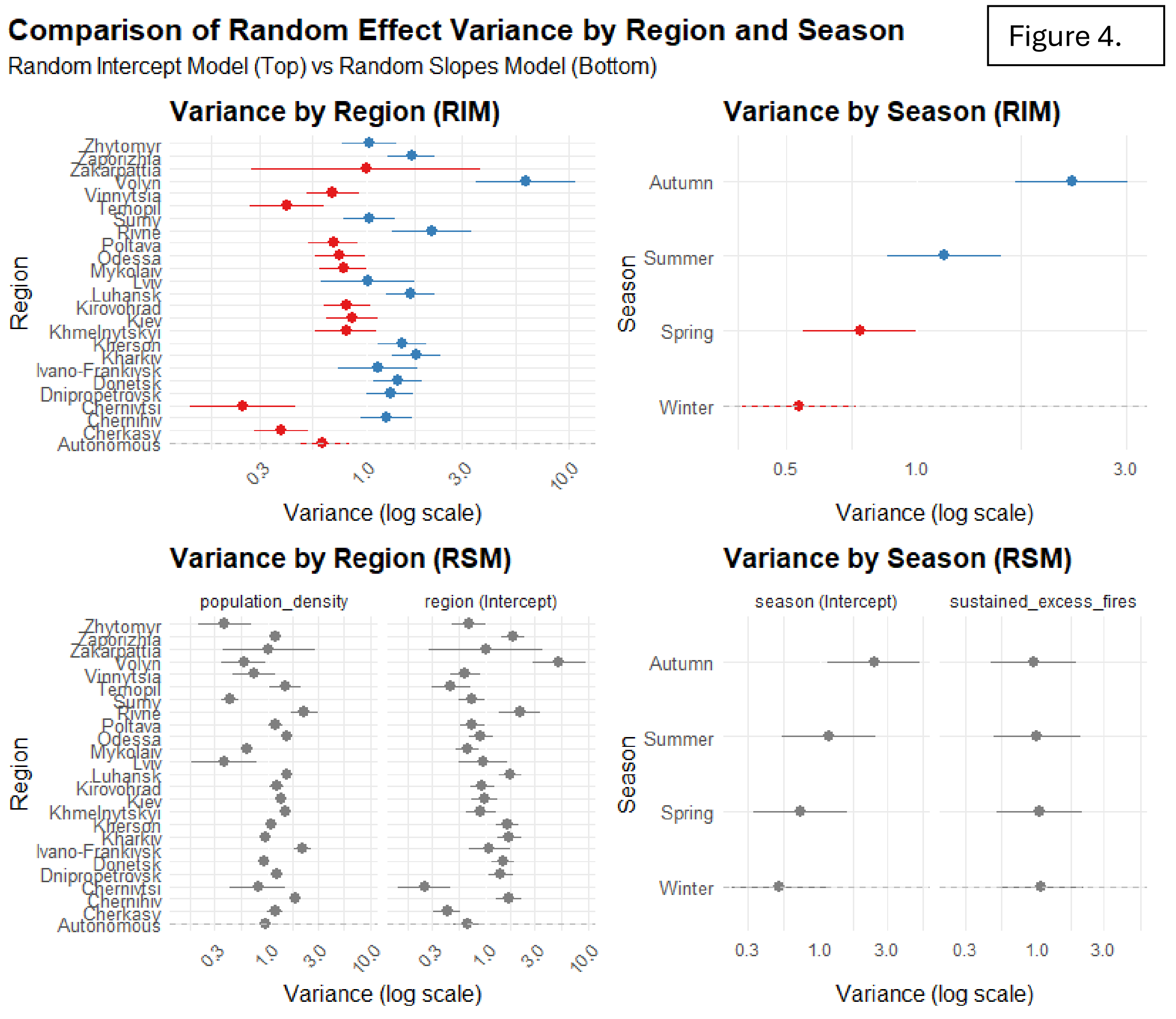
Figure 3.

The Binned Residuals Plots examines residual behaviour across different predicted probability ranges. This method is more suitable than a Q-Q plot or normality histogram, as my data is binary. A sample of 10,000 observations were used to streamline processing. In the RIM, the residuals exhibit a U-shaped pattern, especially at probability extremes, indicating systematic bias with underprediction or overprediction. In contrast, the RSM shows a flatter residual trend around zero in the midrange, suggesting a more accurate fit with fewer systematic deviations, supporting it as the better model for capturing variability across regions.

The Intraclass Correlation Coefficient (ICC) further reinforces this preference: the RIM has a lower ICC (0.113), while the RSM, with an ICC of 0.206, demonstrates slightly better reliability in accounting for regional differences in war-related fires. Figure 3 was created using ggplot2, organized with patchwork, and prepared with dplyr.

Model Insights

Figure 4.



ırithmic scale for clarity. This plot highlights how random effects vary by region and season, providing insights into the spatial and temporal variat

The Random Effect Variance Plot demonstrates substantial regional and seasonal variability in the effects of population density and sustained excess fires. High-density regions such as Zaporizhia exhibit greater variance, suggesting that areas with larger populations may be more strategically targeted or impacted by conflict. Seasonally, the RSM highlights variability in sustained fires activity, indicating that environmental conditions or shifts in conflict intensity influence fire classification differently across seasons.

While fixed effects reveal a general association between high population density, sustained excess fires, and an increased likelihood of war-related fire classification. The random effects underscore that these influences vary across specific regions and seasons. This variation supports the use of the RSM, which captures these nuanced patterns more effectively than the Random Intercept Model. This choice of model, emphasising localised effects, gives us greater insight into predicting spatial and temporal patterns, and therefore, should provide a higher degree of accuracy when classifying fires. These observable variances also indicate the presence of spatial and temporal patters, defining their relationship to war-related fires further. Figure 4 was created using sjPlot, ggplot2, patchwork, and stringr in R. I log'ed the x-axis, so differences in variance are clearer.

Model Suitability

The RSM proves to be the more accurate model for testing the hypothesis on spatial and temporal patterns in classifying fires as war-related. By allowing predictor effects to vary across spatial and temporal groups, the RSM outperforms the RIM on key diagnostic assumptions, including linearity, homoscedasticity, and normality of residuals. This flexibility reduces bias in fitted values and provides a closer approximation of normal residual distribution, resulting in a more accurate representation of the data.

Additionally, the anova() function confirms the RSM's superiority based on metrics such as AIC, BIC, log-likelihood, and deviance, with the Chi-squared test further supporting its improved fit. This enhanced performance makes the RSM a more robust model for capturing the spatial and temporal variability inherent in war-related fires, directly supporting the hypothesis that these factors are significant predictors.

Hypothesis Testing

Bonferroni Correction

To control for multiple comparisons, a Bonferroni correction adjusted the significance level to 0.0125 (initial $\alpha = 0.05$; $m = 4$ comparisons). This is crucial, as it prevents type 1 error arising from multiple comparisons, thus keeping the models statistical integrity.

Analysis

The likelihood ratio test (LRT), using R's lmtest package, comparing the null model (see code) with the RSM confirms that sustained fires activities is a highly significant predictor of war-related fire classification (Estimate = 16.4944, $p < 2.2e-16$). Population density does not show significance (Estimate = -0.1157, $p = 0.266$) at the adjusted level.

The model's random effects reveal notable variance across regions and seasons, illustrated in the Random Effect Variance Plot, underscore the uneven impact of population density and sustained excess fires across regions and seasons. The binned residuals plots support rejecting the null hypothesis by demonstrating improved model fit with the RSM, which accounts for regional and seasonal variability. The flatter residuals trend suggests fewer systematic errors, confirming spatial and temporal factors as significant predictors of war-related fires. This result aligns with AIC, BIC, and Chi-squared test results from the anova(), reinforcing the model's improved accuracy when these factors are considered.

Conclusion

The analysis supports rejecting the null hypothesis, confirming sustained excess fires as a significant factor in classifying war-related fires. Although population density was not significant, the random effects in the RSM highlight essential regional and seasonal variability. Therefore, the model confirm that spatial proximity to conflict zones and seasonal variations play critical roles in predicting and classifying war-related fires.

Figure 5.

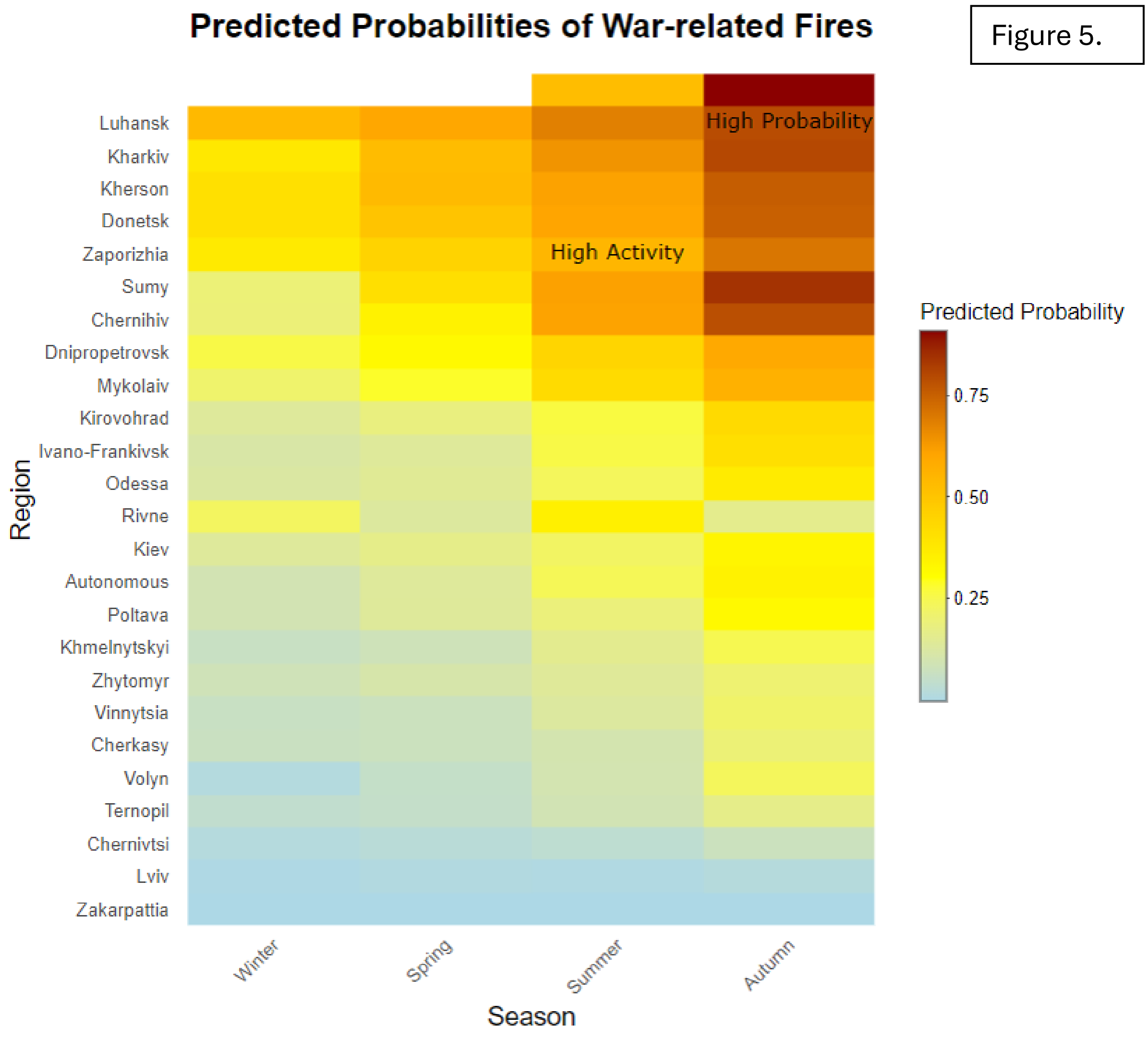


Figure 5 (interactive) visualises the predicted probabilities of war-related fires from the RSM across regions and seasons, organised into four areas of Ukraine (East, North, South, West)(top to bottom). Two distinct patterns appear, aligning with the research question on spatial and temporal patterns.

Firstly, a spatial pattern is clear, with eastern and northern regions consistently showing higher predicted probabilities of war-related fires (around 0.65), compared to western regions, which average a probability of 0.1. This confirms that war-related fires are more concentrated in specific areas, those closer to Russia, supporting the hypothesis that spatial patterns influence war-related fire classifications.

Secondly, a clear temporal pattern shows that the probability of war-related fires steadily increases from winter to autumn across most regions, showing seasonal factors may heighten conflict intensity as the year progresses. Together, these patterns affirm that both spatial and temporal factors are significant predictors. Thus, supporting us to reject the null hypothesis.

Created using ggplot2's geom_tile(), plotly, lme4, stringr, tidyr, and dplyr. I obtained the predicted probabilities using the predict() function from the lme4 package. (Find the interactive plot on my GitHub – linked in appendix)

Discussion:

The analyses confirm that spatial and temporal patterns are significant predictors of war-related fires in Ukraine. The Random Slopes Model provided the best fit, effectively capturing the regional and seasonal variations that simpler models could not. Higher probabilities of war-related fires in Eastern and Northern Ukraine suggest that proximity to conflict zones substantially increases the likelihood of such events. Seasonally, an upward trend from winter to autumn indicates that fire occurrences may align with specific environmental conditions or intensified conflict periods.

Implications

These findings have practical implications for conflict and environmental management. By identifying high-risk regions and seasons, policymakers and emergency responders can better allocate resources for fire prevention and response. The results provide a foundation for more targeted monitoring and intervention in areas and times most affected by conflict-related fires.

Limitations

The binary sustained fire activity variable limits insight by not capturing the intensity or duration of fires, which could refine our understanding of fire dynamics. Additionally, cloud cover in satellite imagery likely obscured some fires, leading to underreporting and potential bias in the data.

Future Research

Future studies could improve on these findings by refining the sustained fire activity variable to capture more detailed fire event data and incorporating conflict-specific and socio-economic data for more nuanced modelling. Using ground-based observations to complement satellite data could also enhance the reliability of war-related fire analysis, reducing the limitations posed by cloud cover.

Conclusion

In answer to the research question—"To what extent can spatial and temporal patterns accurately classify whether a fire is a war-related fire in Ukraine?" The study finds that these patterns can indeed serve as accurate classifiers, strongly rejecting the null hypothesis. Spatial and temporal factors, as modelled through the Random Slopes approach, provide significant predictive power in distinguishing war-related fires, particularly when accounting for regional proximity to conflict zones and seasonal variations. This research underscores the value of spatial and temporal data in conflict-related environmental analysis and highlights avenues for refining predictive capabilities in future studies.

Appendix

Link to animated plot and interactive heatmap - https://github.com/KetchupJL/university-projects?tab=readme-ov-file

https://github.com/KetchupJL

ChatGPT Prompts:

A range of different prompts I used:

1. Give me suggesting on how I can improve this code
2. Here is an error in my code " ", how can I fix it?
3. Give me some example themes/styles for my table/plot
4. How can I add a *specific feature* to my table/plot?
5. Can you improve my comments on my code
6. Is my code clear and reproducible?
7. Can you check my writing and suggest improvements?
8. With this information " ", can you help me structure how I go about completing " ".

In the next pages, I will paste my R code raw. At the start of each script, there's a briefly comment on what this script is.

Side note – I made a few extra graphics, you will see these in code. I will most likely put these on my GitHub project version as they don't fit on this (feel free to ignore them).

Data Wrangling:

```
# Load necessary libraries for data wrangling and visualisation
library(dplyr)       # Data manipulation
library(tidyverse)   # Data science tools
library(ggplot2)     # Data visualisation

# Load libraries for handling geo-spatial data
library(sf)          # Spatial data handling

# Load library for date/time manipulation
library(lubridate)   # Working with dates

# Load fire datasets
# Main fire dataset (includes both war and non-war fire data)
fires <- read.csv("C:/Users/james/OneDrive/Documents/Working With Data (MTHM501/Ukraine War-Fires - MTHM501
Project/Data/ukraine_fires.csv")

# War-fire dataset (contains only war-fire related variables)
war_fires <- read.csv("C:/Users/james/OneDrive/Documents/Working With Data (MTHM501/Ukraine War-Fires - MTHM501
Project/Data/ukraine_war_fires.csv")

# Inspecting the structure and summary of both datasets
head(fires)
str(fires)
summary(fires)

head(war_fires)
str(war_fires)
summary(war_fires)

# Checking for missing data in both datasets
colSums(is.na(fires))
colSums(is.na(war_fires))

# Visualising missing data
library(Amelia)
missmap(fires)  # This may take some time to load. All values for the 'city' variable are missing

# Remove the 'year' and 'city' columns due to incomplete data
fires_cleaned <- fires %>% select(-year, -city)

# Double-checking for any remaining missing values
colSums(is.na(fires_cleaned))

# Checking for duplicate rows (keeping them for now)
duplicate_rows <- fires_cleaned[duplicated(fires_cleaned), ]

# Identifying any completely empty rows (i.e., rows where all values are NA)
empty_rows <- fires_cleaned[rowSums(is.na(fires_cleaned)) == ncol(fires_cleaned), ]

# Check the structure of the dataset to ensure the columns are removed
str(fires_cleaned)


# Renaming columns for clarity and consistency
fires_cleaned <- fires_cleaned %>%
  rename(latitude = LATITUDE,
         longitude = LONGITUDE,
         time_of_day = ACQ_TIME,
         population_density = pop_density,
         urban_area = in_urban_area,
         excess_fires = excess_fire,
         sustained_excess_fires = sustained_excess)

# Removing unnecessary columns
fires_cleaned <- fires_cleaned %>%
  select(-time_of_year, -urban_area, -predicted_fire, -fire_in_window,
         -length_of_war_fire_area, -war_fire_restrictive,
         -in_ukraine_held_area, -fires_per_day_in_ukraine_held_area,
         -war_fires_per_day_in_ukraine_held_area,
         -fires_per_day_in_russia_held_area,
         -war_fires_per_day_in_russia_held_area)

# Converting the 'date' column to Date type
fires_cleaned$date <- as.Date(fires_cleaned$date)

# Converting categorical variable 'war_fire' to factors
fires_cleaned$war_fire <- as.factor(fires_cleaned$war_fire)

# Double-checking the column types
str(fires_cleaned)
```

```
# Simplifying the 'time_of_day' variable for better analysis by clustering into periods of the day
fires_cleaned <- fires_cleaned %>%
  mutate(hour = floor(time_of_day / 100),
         time_of_day = case_when(
           hour >= 6 & hour < 12 ~ "Morning",
           hour >= 12 & hour < 18 ~ "Afternoon",
           hour >= 18 & hour < 24 ~ "Evening",
           TRUE ~ "Night"
         ))

# Setting the levels
fires_cleaned$time_of_day <- factor(fires_cleaned$time_of_day,
                                  levels = c("Morning", "Afternoon", "Evening", "Night"))

# Adding spatial and temporal variables to enhance analysis

# Grouping fires by season using the 'lubridate' package for month extraction
fires_cleaned <- fires_cleaned %>%
  mutate(season = case_when(
    month(date) %in% c(12, 1, 2) ~ "Winter",
    month(date) %in% c(3, 4, 5) ~ "Spring",
    month(date) %in% c(6, 7, 8) ~ "Summer",
    month(date) %in% c(9, 10, 11) ~ "Autumn"
  ))

# Setting the levels
fires_cleaned$season <- factor(fires_cleaned$season,
                                  levels = c("Winter", "Spring", "Summer", "Autumn"))

# Loading Ukraine regional boundaries dataset for spatial analysis
sf_use_s2(FALSE)  # Disabling spherical geometry operations
regions <- st_read("C:/Users/james/OneDrive/Documents/Working With Data (MTHM501/Ukraine War-Fires - MTHM501 Project/Data/ukraine-
with-regions_1530.geojson")

# Ensuring the regions dataset is valid for spatial operations
regions <- st_make_valid(regions)

# Checking if lat/lon values are within valid ranges before transforming to 'sf'
fires_cleaned <- fires_cleaned %>%
  filter(between(latitude, -90, 90), between(longitude, -180, 180))

# Converting fire data to an sf object with latitude and longitude coordinates
fires_cleaned_sf <- fires_cleaned %>%
  st_as_sf(coords = c("longitude", "latitude"), crs = 4326)

# Spatial join to add regional information to the fires dataset
fires_with_regions <- st_join(fires_cleaned_sf, regions, join = st_intersects)

# Adding 'region_name' from the regions dataset to the fires dataset
fires_with_regions <- fires_with_regions %>%
  mutate(region_name = regions$region_name) %>%
  rename(region = name)

# Summarising total war fires per region
total_war_fires <- fires_with_regions %>%
  group_by(region) %>%
  summarize(total_war_fires = sum(war_fire))

# Converting 'region','season' and 'id_big'to factors for random effects modelling
fires_with_regions$region <- as.factor(fires_with_regions$region)
fires_with_regions$season <- as.factor(fires_with_regions$season)
fires_with_regions$id_big <- as.factor(fires_with_regions$id_big)

# Converting war_fire to numeric
fires_with_regions$war_fire <- as.numeric(as.character(fires_with_regions$war_fire))

# Saving the cleaned dataset
write.csv(fires_with_regions, "C:/Users/james/OneDrive/Documents/Working With Data (MTHM501/Ukraine War-Fires - MTHM501
Project/Data/fires_with_region.csv", row.names = FALSE)
```

Table 1. Summary table of war-fires by Region

```
# Load required packages
library(gt)
library(dplyr)

# Converting sf object to a regular data frame
TestData1 <- as.data.frame(fires_with_regions)

TestData1 <- TestData1 %>%
  mutate(region = str_extract(region, "^[^ ]+"))  # Extract the first word from Regions – Had “Oblast” at the start of every region

unique_regions <- unique(TestData1$region)
print(unique_regions)

# Creating sub-regions column
region_mapping <- c(
  "Autonomous" = "Crimea",
  "Cherkasy" = "Central Ukraine",
  "Chernihiv" = "North Ukraine",
  "Chernivtsi" = "West Ukraine",
  "Dnipropetrovsk" = "East Ukraine",
  "Donetsk" = "East Ukraine",
  "Ivano-Frankivsk" = "West Ukraine",
  "Kharkiv" = "East Ukraine",
  "Kherson" = "South Ukraine",
  "Khmelnytskyi" = "West Ukraine",
  "Kiev" = "North Ukraine",
  "Kirovohrad" = "Central Ukraine",
  "Luhansk" = "East Ukraine",
  "Lviv" = "West Ukraine",
  "Mykolaiv" = "South Ukraine",
  "Odessa" = "South Ukraine",
  "Poltava" = "Central Ukraine",
  "Rivne" = "West Ukraine",
  "Sumy" = "North Ukraine",
  "Ternopil" = "West Ukraine",
  "Vinnytsia" = "Central Ukraine",
  "Volyn" = "West Ukraine",
  "Zakarpattia" = "West Ukraine",
  "Zaporizhia" = "East Ukraine",
  "Zhytomyr" = "North Ukraine"
)

# Update the heatmap_data with new region names
TestData1 <- TestData1 %>%
  mutate(sub_region = recode(region, !!!region_mapping))  # Create new sub_region column

# Define the mapping for sub-regions and corresponding regions
region_mapping <- list(
  "North Ukraine" = c("Chernihiv", "Kiev", "Sumy", "Zhytomyr"),
  "South Ukraine" = c("Kherson", "Mykolaiv", "Odessa"),
  "East Ukraine" = c("Dnipropetrovsk", "Donetsk", "Kharkiv", "Luhansk", "Zaporizhia"),
  "West Ukraine" = c("Chernivtsi", "Cherkasy", "Ivano-Frankivsk", "Khmelnytskyi",
                     "Lviv", "Poltava", "Rivne", "Ternopil", "Vinnytsia",
                     "Volyn", "Zakarpattia")
)


# Define the selected regions
selected_regions <- c("Chernihiv", "Sumy", "Kharkiv", "Zaporizhia", "Kiev", "Kherson", "Chernivtsi", "Zakarpattia")

# Filter TestData1 for only the selected regions and exclude rows with NA in relevant columns
war_fires_summary <- TestData1 %>%
```

```r
  filter(region %in% selected_regions) %>%
  group_by(region) %>%
  summarise(
    total_fires = n(),
    war_fires = sum(war_fire, na.rm = TRUE),
    fires_per_day = mean(fires_per_day, na.rm = TRUE),
    war_fires_per_day = mean(war_fires_per_day, na.rm = TRUE),
    war_fire_percentage = (sum(war_fire, na.rm = TRUE) / n()) * 100,
    avg_pop_density = mean(population_density, na.rm = TRUE),
    avg_sustained_excess = mean(sustained_excess_fires, na.rm = TRUE),
    sub_region = first(sub_region)  # Use the first occurrence of sub_region for each region
  )

# Arrange by sub_region so that regions are automatically grouped by area in the table
war_fires_summary <- war_fires_summary %>%
  arrange(sub_region, region)

# Display the table using gt
war_fires_summary %>%
  gt() %>%

  # Add title and subtitle to the table
  tab_header(
    title = "Summary of War-related Fires by Region",
    subtitle = "Including population density and sustained excess fire activity"
  ) %>%

  # Format specific columns to have 1 decimal place
  fmt_number(
    columns = c(war_fire_percentage, avg_pop_density, avg_sustained_excess, fires_per_day, war_fires_per_day),
    decimals = 1
  ) %>%

  # Rename columns for better readability in the table
  cols_label(
    region = "Region",
    total_fires = "Total Fires",
    war_fires = "War Fires",
    war_fire_percentage = "War Fires (%)",
    avg_pop_density = "Avg. Pop Density",
    avg_sustained_excess = "Avg. Sustained Fire",
    fires_per_day = "Avg. Daily Fires",
    war_fires_per_day = "Avg. Daily War Fires",
    sub_region = "Area"
  ) %>%

  # Customize table options for font sizes and alignment
  tab_options(
    table.font.size = px(14),               # Font size for table content
    column_labels.font.size = px(16),       # Font size for column labels
    table.width = pct(100),                 # Set table width to 100% of available space
    heading.title.font.size = px(20),       # Font size for title
    heading.subtitle.font.size = px(16),    # Font size for subtitle
    heading.align = "center"                # Center-align the title and subtitle
  ) %>%

  # Apply row striping (alternate row coloring) with light gray color
  tab_style(
    style = cell_fill(color = "lightgray"),
    locations = cells_body(rows = seq(1, nrow(war_fires_summary), 2))  # Apply to every other row (odd rows)
  ) %>%

  # Add an outline around the entire table
  opt_table_outline() %>%
  # Apply color gradient to the "War Fires (%)" column for visual emphasis
  data_color(
    columns = vars(war_fire_percentage),
    colors = scales::col_numeric(
      palette = c("white", "lightgrey", "lightcoral", "red"),  # Color gradient from white to red
      domain = c(0, max(war_fires_summary$war_fire_percentage, na.rm = TRUE))  # Scale from 0 to max percentage
    )
  ) %>%

  # Add a bold vertical border to separate the "War Fires (%)" column
  tab_style(
    style = cell_borders(sides = "left", color = "black", weight = px(2)),
    locations = cells_body(columns = vars(war_fire_percentage))
  ) %>%

  # Highlight the "Area" column with a light blue background for visual separation
  tab_style(
    style = cell_fill(color = "lightblue"),
    locations = cells_body(columns = vars(sub_region))
  ) %>%

  # Set a light gray border around all table cells for a cleaner look
  tab_style(
    style = cell_borders(sides = "all", color = "gray", weight = px(1)),
    locations = cells_body()
  ) %>%

  # Add a footnote for the "War Fires (%)" column
  tab_footnote(
    footnote = "War fire percentage is based on total fires in each region.",
    locations = cells_column_labels(columns = vars(war_fire_percentage))
  ) %>%

  # Add a footnote for the "Avg. Sustained Fire" column
  tab_footnote(
    footnote = "Sustained fire-activity beyond prediction.",
    locations = cells_column_labels(columns = vars(avg_sustained_excess))
  )
```

Figure 1. Animated Geospatial Plot with Ukraine Regional Map

```
# Load Required Packages
library(ggplot2)
library(gganimate)
library(dplyr)
library(lubridate)
library(sf)
ukraine_map <- st_read("C:/Users/james/OneDrive/Documents/Working With Data (MTHM501/Ukraine War-Fires - MTHM501
Project/Data/ukraine-with-regions_1530.geojson") # Adding map data

GeospacialDataSet <- st_as_sf(fires_cleaned, coords = c("longitude", "latitude"), crs = 4326, remove = FALSE) # Turning
dataset in a sf

# Spatial join to add regional information to GeospacialDataSet
GeospacialDataSet <- st_join(GeospacialDataSet, regions, join = st_intersects)

# Adding 'region_name' from the regions dataset to GeospacialDataSet
GeospacialDataSet <- GeospacialDataSet %>%
  mutate(region_name = regions$region_name) %>%
  rename(region = name)


# Load necessary libraries
library(ggplot2)
library(gganimate)
library(ggshadow)

# Create base map with Ukraine's boundaries
base_map <- ggplot(data = ukraine_map) +
  geom_sf(fill = "gray90", color = "black") +
  theme_minimal() +
  labs(
    title = 'Fire Events Over Time in Ukraine',
    subtitle = 'War-related fires, Non-war fires, and Sustained Excess',
    x = "Longitude",
    y = "Latitude"
  )

library(ggspatial)
base_map <- base_map +
  annotation_scale(location = "bl")  # Scale bar at bottom left

# Add point data layer for fire events
animated_map <- base_map +
  geom_point(
    data = GeospacialDataSet,
    aes(
      x = longitude,
      y = latitude,
      color = factor(war_fire),
      shape = factor(sustained_excess_fires)
    ),
    size = 2,  # Reduced size
    stroke = 1,
    alpha = ifelse(GeospacialDataSet$war_fire == 0, 0.4, 1)  # Reduce alpha for non-war fires
  ) +
  scale_color_manual(
    values = c("1" = "#0073C2", "0" = "darkred"),  # Blue for war-related, Orange for non-war
    labels = c("0" = "Non-War Fire", "1" = "War-Related Fire"),
    name = "Fire Type"
  ) +
  scale_shape_manual(
    values = c("0" = 21, "1" = 16),  # 21 = filled circle, 16 = circle with border
    labels = c("0" = "Sustained Activity", "1" = "No Sustained Activity"),  # Labelling
    name = "Sustained Fire Activity"
  ) +
  labs(
    title = "Fires in Ukraine Over Time: {frame_time}",                              # Labels
    subtitle = "Blue: War-Related Fires | Orange: Non-War Fires",
    x = "Longitude", y = "Latitude"
  ) +
  theme_minimal(base_family = "Arial") +                              # Setting theme and legends
  theme(
    plot.title = element_text(size = 16, face = "bold"),
    legend.position = c(0.85, 0.85),
    legend.box = "vertical",
    legend.title = element_text(size = 12),
    legend.text = element_text(size = 10)
  ) +
  transition_time(date) +  # Animate over the 'date' variable
  shadow_mark(past = TRUE, alpha = 0.10) +  # Shadow mark for past events
  scale_color_manual(
    values = c("1" = "grey", "0" = "darkred"),  # Same as main points
    guide = "none" ) + # Suppress legend for shadow marks
  ease_aes('linear')

# To display or save the animation
animated_map
anim_save(
  filename = "ukraine_fire_animation.gif",
  animation = animated_map,
  width = 1600,         # Width in pixels
  height = 1200,        # Height in pixels
  end_pause = 15,       # Pause at the end of the animation for 15 frames
  res = 150             # Resolution in DPI
)
```

Figure 2. Seasonal Trend Plot with War-related over Time

```
#2.        Seasonal Trend Plot with War-related and Non-War-related Fires over time


# Creating new data frame, as I some variables need adjusting
fires_daily_agg <- fires_with_regions %>%
  group_by(date, season) %>%                          #group by season
  summarise(
    fires_per_day = mean(fires_per_day, na.rm = TRUE),
    war_fires_per_day = mean(war_fires_per_day, na.rm = TRUE),        # Getting daily war-fires variable
    .groups = 'drop'  # Prevent warnings about grouping
  )

fires_daily_agg <- fires_daily_agg %>%
  mutate(
    year = year(date),             # Extract the year from the date
    month = month(date, label = TRUE, abbr = TRUE))  # Extract month names

    # Create a seasonal indicator column
    fires_daily_agg <- fires_daily_agg %>%
      mutate(season = case_when(
        month(date) %in% c(12, 1, 2) ~ "Winter",
        month(date) %in% c(3, 4, 5) ~ "Spring",
        month(date) %in% c(6, 7, 8) ~ "Summer",
        month(date) %in% c(9, 10, 11) ~ "Autumn",
        TRUE ~ "Unknown"))

    write.csv(fires_daily_agg, "C:/Users/james/OneDrive/Documents/Working With Data (MTHM501/Ukraine War-Fires -
MTHM501 Project/Data/fires_daily_agg.csv", row.names = FALSE)

    # Load required libraries
    library(ggplot2)
    library(gridExtra)
    library(lubridate)
    library(dplyr)
    library(grid)

    # Define colors for seasons for enhanced clarity
    season_colors <- c("Winter" = "#1f77b4", "Spring" = "#2ca02c", "Summer" = "#ff7f0e", "Autumn" = "#d62728")

    # Ensure fires_daily_agg has seasons labeled appropriately
    fires_daily_agg <- fires_daily_agg %>%
      mutate(
        season = case_when(
          month(date) %in% c(12, 1, 2) ~ "Winter",
          month(date) %in% c(3, 4, 5) ~ "Spring",
          month(date) %in% c(6, 7, 8) ~ "Summer",
          month(date) %in% c(9, 10, 11) ~ "Autumn",
          TRUE ~ NA_character_
        )
      )

    # Create the first plot with the legend included
    plot_with_legend <- ggplot(fires_daily_agg %>% filter(year == 2022), aes(x = date, y = war_fires_per_day, fill = season)) +
      geom_bar(stat = "identity", width = 0.6, color = "black", alpha = 0.8) +
      scale_fill_manual(values = season_colors, guide = guide_legend(direction = "horizontal")) +  # Set legend to horizontal
      labs(subtitle = "War-related Fires per Day - 2022", y = "War Fires per Day") +
      ylim(0, 5000)+
      theme_minimal(base_family = "Arial") +                              #Themes and intricate formatting
      theme(
        plot.subtitle = element_text(hjust = 0.5, size = 14, face = "bold"),
        axis.text.x = element_text(angle = 45, hjust = 1, size = 10),
        axis.text.y = element_text(size = 10),
        axis.title.y = element_text(size = 12, face = "bold"),
        panel.grid.major.x = element_line(color = "grey80", linetype = "dotted", size = 0.5),
        panel.grid.major.y = element_line(color = "grey85"),
        panel.grid.minor = element_blank(),
        plot.background = element_rect(fill = "#f9f9f9"),
        legend.position = "top"  # Place the legend at the top
      )

    # Generate the remaining plots without the legend
    plot_list <- lapply(setdiff(unique(fires_daily_agg$year), 2022), function(y) {
      ggplot(fires_daily_agg %>% filter(year == y), aes(x = date, y = war_fires_per_day, fill = season)) +
        geom_bar(stat = "identity", width = 0.6, color = "black", alpha = 0.8, show.legend = FALSE) +  # Suppress legend here
        scale_fill_manual(values = season_colors, na.translate = FALSE) +
        labs(subtitle = paste("War-related Fires per Day -", y)) +
        scale_x_date(limits = c(as.Date(paste0(y, "-01-01")), as.Date(paste0(y, "-12-31"))), date_breaks = "1 month", date_labels =
"%b") +
        ylim(0, 5000) +
        theme_minimal(base_family = "Arial") +                          #Themes and intricate formatting
        theme(
          plot.subtitle = element_text(hjust = 0.5, size = 14, face = "bold"),
          axis.text.x = element_text(angle = 45, hjust = 1, size = 10),
          axis.text.y = element_text(size = 10),
          axis.title.y = element_blank(),
          panel.grid.major.x = element_line(color = "grey80", linetype = "dotted", size = 0.5),
          panel.grid.major.y = element_line(color = "grey85"),
          panel.grid.minor = element_blank(),
          plot.background = element_rect(fill = "#f9f9f9")
        )
    })

    # Arrange the plots vertically with the legend only in the first plot
    combined_plot <- arrangeGrob(
      grobs = c(list(plot_with_legend), plot_list),  # Include plot_with_legend as the first plot
      ncol = 1
    )

    # Add main title and data source text
    final_output <- grid.arrange(
      combined_plot,
      top = textGrob("Seasonal Trends of War-related Fires in Ukraine", gp = gpar(fontsize = 24, fontface = "bold", family =
"Arial")),
      bottom = textGrob("Data Source: Fires in Ukraine (2022-2024)", gp = gpar(fontsize = 12, fontface = "italic", family = "Arial"),
hjust = 1)
    )

    # Display the final output
    grid.draw(final_output)
```

Everything to do with Hierarchical Models, Hypothesis testing and diagnostics.

```
# HLM #############################
fires_with_region <- read.csv("C:/Users/james/OneDrive/Documents/Working With Data (MTHM501/Ukraine War-Fires - 
MTHM501 Project/Data/fires_with_region.csv")


# Check data structure
# Ensuring variable formats are compatible for hierarchical modeling.
str(fires_with_regions)

# Converting sf object to a regular data frame
TestData1 <- as.data.frame(fires_with_regions)

################# Standardising the population density parameter for the HLM

# Standardise population_density

# Apply log transformation (add a small constant to avoid log(0) if needed)
# Applying log transformation to reduce skewness in population density
TestData1$population_density <- log(TestData1$population_density + 1)
# Standardisation
TestData1$population_density <- scale(TestData1$population_density, center = TRUE, scale = TRUE)


# Visualising this odd distribution

hist(TestData1$population_density, main = "Distribution of Standardized Population Density", xlab = "Standardized Population 
Density")


write.csv(TestData1, "C:/Users/james/OneDrive/Documents/Working With Data (MTHM501/Ukraine War-Fires - MTHM501 
Project/Data/TestData1.csv", row.names = FALSE)


# Inspect the data frame
str(TestData1)

# Loading packages

require(lmerTest)
require(tidyverse)
require(readxl)
require(magrittr)
require(sjPlot)
require(sjmisc)
require(sjstats)
require(arm)
require(performance)


####################################################
# Fit the hierarchical logistic model - random intercept model
hlm_model_intercept <- glmer(war_fire ~ population_density + sustained_excess_fires +
                 (1 | region) + (1 | season) + (1 | id_big),
                data = TestData1, family = binomial)

# Display the model summary
summary(hlm_model_intercept)

##################################################
## Random slopes model

hlm_model_random_slopes <- glmer(war_fire ~ population_density + sustained_excess_fires +
                          (1 + population_density | region) +
                          (1 + sustained_excess_fires | season)+
                          (1 | id_big),
                        data = TestData1, family = binomial)

summary(hlm_model_random_slopes)
```

Table 2. Model Comparison

```
#######################################################################
######################## Analysis ####################################
## Summary stats

# Create the data for the table
library(knitr)
library(kableExtra)

# Streamlined data, focusing on key parameters and rounding values
model_comparison <- data.frame(
  Parameter = c("AIC", "BIC",
                "Variance (id_big)", "Variance (region)", "Variance (season)",
                "Intercept Estimate", "Population Density Estimate",
                "Sustained Fire Activity Estimate"),
  `Random Slopes Model` = c("420760.1", "420868.3",
                            "1.223", "0.423", "0.502",
                            "-17.65", "-0.12", "16.49"),
  `Intercept Model` = c("422560.3", "422625.2",
                        "1.208", "0.442", "0.314",
                        "-21.79", "0.02", "20.72"),
  `p-value (Random Slopes)` = c("", "",
                                "", "", "",
                                "<2e-16", "0.266", "<2e-16"),  # p-values for random slopes model
  `p-value (Intercept)` = c("", "",
                            "", "", "",
                            "<2e-16", "0.0012", "<2e-16")  # p-values for intercept model
)

# Create and format the table
kable(model_comparison, caption = "Comparison of Hierarchical Models with p-values") %>%
  kable_styling(full_width = FALSE, position = "center") %>%
  column_spec(2:3, width = "4cm") %>%  # Set column width for better layout
  column_spec(4:5, width = "3cm", color = "black") %>%  # p-values columns styling
  column_spec(2, bold = TRUE, color = "darkblue") %>%
  column_spec(3, bold = TRUE, color = "darkgreen") %>%
  row_spec(1:2, background = "#e6f2ff") %>%  # Highlight AIC/BIC rows for readability
  pack_rows("Model Fit Statistics", 1, 2, bold = TRUE, background = "#f7f7f7") %>%
  pack_rows("Random Effects", 3, 5, bold = TRUE, background = "#f9f9f9") %>%
  pack_rows("Fixed Effects", 6, 8, bold = TRUE, background = "#f7f7f7")



#######################################################################
######################################################################
#########################################################################
```

Figure 3. Binned Residuals Plot

```
# Model diagnostics

############## Binned Residuals Plot
# Load required libraries
library(ggplot2)
library(dplyr)
library(patchwork)  # For combining plots

# Set seed for reproducibility
# Ensuring that the sample selection below can be reproduced in future runs
set.seed(123)

# Random Intercept Model: Sample and prepare binned data
# Here, I sample 10,000 residuals and fitted values from the model to reduce computational load
sample_intercept <- sample(seq_along(residuals(hlm_model_intercept, type = "response")), 10000)
residuals_intercept <- residuals(hlm_model_intercept, type = "response")[sample_intercept]
fitted_intercept <- fitted(hlm_model_intercept)[sample_intercept]

# Creating a binned dataset to examine mean residuals within ranges of fitted values
binned_intercept <- data.frame(
  fitted_intercept = fitted_intercept,
  residuals_intercept = residuals_intercept
) %>%
  mutate(bins = cut(fitted_intercept, breaks = seq(0, 1, by = 0.1))) %>%  # Divide fitted values into 10 bins (0.1 intervals)
  group_by(bins) %>%
  summarize(
    mean_fitted_intercept = mean(fitted_intercept, na.rm = TRUE),        # Mean fitted probability within each bin
    mean_residuals_intercept = mean(residuals_intercept, na.rm = TRUE)    # Mean residual within each bin
  )

# Random Slopes Model: Sample and prepare binned data
# I repeat the sampling and binning process for the Random Slopes Model for a fair comparison
sample_slopes <- sample(seq_along(residuals(hlm_model_random_slopes, type = "response")), 10000)
residuals_slopes <- residuals(hlm_model_random_slopes, type = "response")[sample_slopes]
fitted_slopes <- fitted(hlm_model_random_slopes)[sample_slopes]

# Creating a binned dataset for the Random Slopes Model
binned_slopes <- data.frame(
  fitted_slopes = fitted_slopes,
  residuals_slopes = residuals_slopes
) %>%
  mutate(bins = cut(fitted_slopes, breaks = seq(0, 1, by = 0.1))) %>%  # Again, 10 bins (0.1 intervals)
  group_by(bins) %>%
  summarize(
    mean_fitted_slopes = mean(fitted_slopes, na.rm = TRUE),        # Mean fitted probability within each bin
    mean_residuals_slopes = mean(residuals_slopes, na.rm = TRUE)    # Mean residual within each bin
  )

# Improved plot for Random Intercept Model
# Creating a residual plot for the Random Intercept Model, with visual enhancements for clarity
RIM_Bin <- ggplot(binned_summary_intercept, aes(x = mean_fitted_intercept, y = mean_residuals_intercept)) +
  geom_point(color = "blue", size = 3) +  # Using larger blue points for visibility
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +  # Adding a reference line at zero for bias assessment
  geom_smooth(method = "loess", se = FALSE, color = "forestgreen", size = 1.2) +  # Adding a LOESS curve to show trend in
residuals
  labs(
    title = "Binned Residuals Plot for Random Intercept Model",  # Specific title for this model
    x = "Average Predicted Probability of War-Related Fires",     # X-axis shows fitted probability
    y = "Average Residual (Observed - Predicted)"                 # Y-axis shows mean residual
  ) +
  annotate("text", x = 0.8, y = 0.05, label = "Systematic Bias at Extremes", color = "blue", size = 3, hjust = 0) +  # Annotation
highlighting observed bias
  theme_minimal() +
  theme(
    plot.title = element_text(size = 14, face = "bold"),           # Formatting title for readability
    axis.title = element_text(size = 12),                          # Adjusting axis titles for readability
    plot.subtitle = element_text(size = 10),                       # Formatting subtitle
    axis.text = element_text(size = 10)                            # Adjusting axis text for clarity
  )

# Improved plot for Random Slopes Model
# Creating a similar residual plot for the Random Slopes Model
RSM_Bin <- ggplot(binned_summary_slopes, aes(x = mean_fitted_slopes, y = mean_residuals_slopes)) +
  geom_point(color = "purple", size = 3) +  # Larger purple points for consistency and visibility
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +  # Reference line at zero for assessing bias
  geom_smooth(method = "loess", se = FALSE, color = "darkorchid", size = 1.2) +  # LOESS curve for trend analysis in
residuals
  labs(
    title = "Binned Residuals Plot for Random Slopes Model",       # Specific title for this model
    x = "Average Predicted Probability of War-Related Fires",      # X-axis shows fitted probability
    y = "Average Residual (Observed - Predicted)"                  # Y-axis shows mean residual
  ) +
  annotate("text", x = 0.8, y = 0.04, label = "Improved Fit Across Range", color = "purple", size = 3, hjust = 0) +  # Annotation
noting better fit
  theme_minimal() +
  theme(
    plot.title = element_text(size = 14, face = "bold"),           # Formatting title for consistency
    axis.title = element_text(size = 12),                          # Adjusting axis titles for readability
    plot.subtitle = element_text(size = 10),                       # Formatting subtitle
    axis.text = element_text(size = 10)                            # Adjusting axis text for clarity
  )

# Combine the plots vertically with patchwork and add an overall title
# Using patchwork to stack the two residual plots vertically, making it easy to compare the models
combined_plot <- (RIM_Bin / RSM_Bin) +  # Stacking the plots
  plot_annotation(
    title = "Comparison of Model Fit: Random Intercept vs. Random Slopes",   # Overall title to contextualize the plot
    subtitle = "Assessing Linearity and Bias in Predicted Probabilities for War-Related Fires",  # Subtitle to clarify purpose
    theme = theme(
      plot.title = element_text(size = 16, face = "bold"),   # Emphasizing the overall title for readability
      plot.subtitle = element_text(size = 12)                # Making subtitle legible
    )
  )

# Display the combined plot for visualization
combined_plot  # Displaying the combined plot
```

Normality of Residuals Plot and Q-Q Plot (Not included, as data is binary/No space)

```
#######################
# Checking Normality of residuals
library(ggplot2)

# Plot for the Random Intercept Model
RIM_HIS <- ggplot(data.frame(residuals = residuals(hlm_model_intercept)), aes(x = residuals)) +
  geom_histogram(aes(y = ..density..), bins = 30, fill = "lightblue", color = "black") +  # Histogram with density scaling
  geom_density(color = "blue", size = 1) +  # Density curve for residuals
  stat_function(fun = dnorm, args = list(mean = mean(residuals(hlm_model_intercept)),
                                          sd = sd(residuals(hlm_model_intercept))),
                color = "red", linetype = "dashed") +  # Overlay normal curve for comparison
  labs(title = "Residuals Distribution of Random Intercept Model", x = "Residuals", y = "Density") +
  theme_minimal()

# Q-Q Plot for Random Intercept Model
RIM_QQ <- ggplot(data.frame(residuals = residuals(hlm_model_intercept)), aes(sample = residuals)) +
  stat_qq() +
  stat_qq_line(color = "red") +
  labs(title = "Q-Q Plot of Residuals for Random Intercept Model", x = "Theoretical Quantiles", y = "Sample Quantiles") +
  theme_minimal()

# Plot for the Random Slopes Model
RSM_HIS <- ggplot(data.frame(residuals = residuals(hlm_model_random_slopes)), aes(x = residuals)) +
  geom_histogram(aes(y = ..density..), bins = 30, fill = "lightblue", color = "black") +  # Histogram with density scaling
  geom_density(color = "blue", size = 1) +  # Density curve for residuals
  stat_function(fun = dnorm, args = list(mean = mean(residuals(hlm_model_random_slopes)),
                                          sd = sd(residuals(hlm_model_random_slopes))),
                color = "red", linetype = "dashed") +  # Overlay normal curve for comparison
  labs(title = "Residuals Distribution of Random Slopes Model", x = "Residuals", y = "Density") +
  theme_minimal()

# Q-Q Plot for Random Slopes Model
RSM_QQ <- ggplot(data.frame(residuals = residuals(hlm_model_random_slopes)), aes(sample = residuals)) +
  stat_qq() +
  stat_qq_line(color = "red") +
  labs(title = "Q-Q Plot of Residuals for Random Slopes Model", x = "Theoretical Quantiles", y = "Sample Quantiles") +
  theme_minimal()

RIM_HIS + RSM_HIS
RIM_QQ + RSM_QQ # May take a while to load

##########################################################################################
####################
##########################################################################################
##########################################################################################
```

Calculating ICC

```
# Calculating ICC

# For Random Intercept Model
icc(hlm_model_intercept)

# For Random Slopes Model
icc(hlm_model_random_slopes) # This takes too long to load.. so ill work it out manually...

# Extract variance components from the random slopes model
variance_components <- VarCorr(hlm_model_random_slopes)

# Calculate the random intercept and random slope variances
```

```
region_intercept_variance <- attr(variance_components$region, "stddev")[1]^2  # Variance of random intercept for 'region'
population_density_slope_variance <- attr(variance_components$region, "stddev")[2]^2  # Variance of random slope for
'population_density'

season_intercept_variance <- attr(variance_components$season, "stddev")[1]^2  # Variance of random intercept for 'season'

# Set the residual variance for logistic regression
residual_variance <- pi^2 / 3

# Calculate Total Variance
total_variance <- region_intercept_variance + population_density_slope_variance + season_intercept_variance +
residual_variance

# Calculate ICC
icc_RSM <- (region_intercept_variance + season_intercept_variance) / total_variance
icc_RSM

# Calculate ICC
icc_RSM <- intercept_variance / (intercept_variance + slope_variance + residual_variance)
icc_RSM



##################################################
```

Figure 4. Random Effect Variance Plot

```
# Random Effect Variance Plot
library(sjPlot)
library(ggplot2)
library(patchwork)
library(stringr)

# Helper function to get the first word of each region name
# This function extracts only the first word from the region name (for concise labeling on the plot)
get_first_word <- function(region_name) {
  str_extract(region_name, "^[^ ]+")
}

# Define custom colors for different random effects in the Random Slopes Model (RSM)
# Each effect has its own color for easier identification in the plot
colors <- c("population_density" = "blue", "region (Intercept)" = "green", "season (Intercept)" = "orange",
"sustained_excess_fires" = "purple")

# Random Effect Variance Plot for Region (Random Intercept Model - RIM)
RE_Region_In <- plot_model(hlm_model_intercept, type = "re")[[2]] +
  scale_y_log10(labels = scales::comma) +  # Apply log scale to the y-axis to better visualize variance differences
  scale_x_discrete(labels = function(x) sapply(x, get_first_word)) +  # Only display the first word of each region
  labs(title = "Variance by Region (RIM)", x = "Region", y = "Variance (log scale)") +  # Set plot titles and labels
  theme_minimal() +  # Use a minimal theme for a clean look
  theme(
    plot.title = element_text(size = 14, face = "bold"),
    axis.title.x = element_text(size = 12),
    axis.title.y = element_text(size = 12),
    axis.text.x = element_text(angle = 45, hjust = 1),  # Rotate x-axis text for readability
    axis.text.y = element_text(size = 10)
  ) +
  geom_vline(xintercept = 1, linetype = "dashed", color = "grey")  # Add a reference line at variance = 1 for context

# Random Effect Variance Plot for Season (Random Intercept Model - RIM)
RE_Season_In <- plot_model(hlm_model_intercept, type = "re")[[3]] +
  scale_y_log10(labels = scales::comma) +  # Log scale for y-axis (variance)
  labs(title = "Variance by Season (RIM)", x = "Season", y = "Variance (log scale)") +
  theme_minimal() +
  theme(
    plot.title = element_text(size = 14, face = "bold"),
    axis.title.x = element_text(size = 12),
    axis.title.y = element_text(size = 12),
    axis.text.y = element_text(size = 10)
  ) +
  geom_vline(xintercept = 1, linetype = "dashed", color = "grey")  # Reference line at variance = 1

# Combine the plots for the Random Intercept Model (RIM)
# This combines the region and season variance plots for the RIM, creating a single view of variance by both factors
RIM_Variance_Plots <- (RE_Region_In | RE_Season_In) +
  plot_annotation(title = "Random Effect Variance by Region and Season - Random Intercept Model")

# Random Effect Variance Plot for Region (Random Slopes Model - RSM)
RE_Region_Sl <- plot_model(hlm_model_random_slopes, type = "re")[[2]] +
  scale_y_log10(labels = scales::comma) +  # Log scale for variance on y-axis
  scale_x_discrete(labels = function(x) sapply(x, get_first_word)) +  # Only the first word of each region
  labs(title = "Variance by Region (RSM)", x = "Region", y = "Variance (log scale)") +
  theme_minimal() +
  theme(
    plot.title = element_text(size = 14, face = "bold"),
    axis.title.x = element_text(size = 12),
    axis.title.y = element_text(size = 12),
    axis.text.x = element_text(angle = 45, hjust = 1),
    axis.text.y = element_text(size = 10)
  ) +
  scale_color_manual(values = colors) +  # Assign custom colors to different random effects
  geom_vline(xintercept = 1, linetype = "dashed", color = "grey")  # Reference line at variance = 1

# Random Effect Variance Plot for Season (Random Slopes Model - RSM)
RE_Season_Sl <- plot_model(hlm_model_random_slopes, type = "re")[[3]] +
  scale_y_log10(labels = scales::comma) +  # Log scale for y-axis
  labs(title = "Variance by Season (RSM)", x = "Season", y = "Variance (log scale)") +
  theme_minimal() +
  theme(
    plot.title = element_text(size = 14, face = "bold"),
    axis.title.x = element_text(size = 12),
    axis.title.y = element_text(size = 12),
    axis.text.y = element_text(size = 10)
  ) +
  scale_color_manual(values = colors) +  # Apply the custom colors to make effects more distinguishable
  geom_vline(xintercept = 1, linetype = "dashed", color = "grey")  # Reference line at variance = 1

# Combine the Random Slopes Model (RSM) plots
# This combines the region and season variance plots for the RSM to give a complete view of random effect variability
RSM_Variance_Plots <- (RE_Region_Sl | RE_Season_Sl) +
  plot_annotation(title = "Random Effect Variance by Region and Season - Random Slopes Model")

# Final combined plot of RIM and RSM for comparison
# Here, I combine the RIM and RSM plots in a single view to compare the models' variance across regions and seasons
Final_Variance_Plot <- (RIM_Variance_Plots / RSM_Variance_Plots) +
  plot_annotation(
    title = "Comparison of Random Effect Variance by Region and Season",
    subtitle = "Random Intercept Model (Top) vs Random Slopes Model (Bottom)",
    caption = "Note: Variance is presented on a logarithmic scale for clarity. This plot highlights how random effects vary by
region and season, providing insights into the spatial and temporal variability of war-related fire classifications.",
    theme = theme(
      plot.title = element_text(size = 16, face = "bold"),
      plot.subtitle = element_text(size = 12),
      plot.caption = element_text(size = 10, hjust = 0.5)
    )
  )

# Display the final plot
# This produces the final combined plot showing random effect variances across both models for my report
Final_Variance_Plot
```

##
##
##

Extra Model Tests

```
# Comparing models

anova(hlm_model_intercept, hlm_model_random_slopes)

################################
# Conducting a Likely-hood ratio test
# Null model without random effects
model_null <- glm(war_fire ~ population_density + sustained_excess_fires,
                  data = TestData1, family = binomial)

summary(model_null)
# Likelihood ratio test
anova(model_null, hlm_model_random_slopes, test = "LRT")
```

Figure 5. Interactive Heatmap

```
#1. Heatmaps of Predicted Probabilities by Region and Season
# Load libraries
library(lme4)
library(dplyr)
library(ggplot2)
library(stringr)
library(tidyr)
library(ploty)

# Get predicted probabilities
TestData1$predicted_probability <- predict(hlm_model_random_slopes, newdata = TestData1, re.form = NULL,
allow.new.levels = TRUE, type = "response")

# Prepare data for heatmap
heatmap_data <- TestData1 %>%
  select( -geometry, -path) %>%
  group_by(region, season) %>%
  summarise(predicted_probability = mean(predicted_probability, na.rm = TRUE)) %>%
  ungroup()

# Editing Region names for aesthetic
heatmap_data <- heatmap_data %>%
  mutate(region = str_extract(region, "^[^ ]+"))  # Extract the first word

unique_regions <- unique(heatmap_data$region)
print(unique_regions)

# Creating sub-regions column
region_mapping <- c(
  "Autonomous" = "Crimea",
  "Cherkasy" = "Central Ukraine",
  "Chernihiv" = "North Ukraine",
  "Chernivtsi" = "West Ukraine",
  "Dnipropetrovsk" = "East Ukraine",
  "Donetsk" = "East Ukraine",
  "Ivano-Frankivsk" = "West Ukraine",
  "Kharkiv" = "East Ukraine",
  "Kherson" = "South Ukraine",
  "Khmelnytskyi" = "West Ukraine",
  "Kiev" = "North Ukraine",
  "Kirovohrad" = "Central Ukraine",
  "Luhansk" = "East Ukraine",
  "Lviv" = "West Ukraine",
  "Mykolaiv" = "South Ukraine",
  "Odessa" = "South Ukraine",
  "Poltava" = "Central Ukraine",
  "Rivne" = "West Ukraine",
  "Sumy" = "North Ukraine",
  "Ternopil" = "West Ukraine",
  "Vinnytsia" = "Central Ukraine",
  "Volyn" = "West Ukraine",
  "Zakarpattia" = "West Ukraine",
  "Zaporizhia" = "East Ukraine",
  "Zhytomyr" = "North Ukraine"
)

# Add sub-region information using case_when for multiple region matches
heatmap_data <- heatmap_data %>%
  mutate(sub_region = case_when(
    region %in% c("Chernihiv", "Kiev", "Sumy", "Zhytomyr") ~ "North Ukraine",
    region %in% c("Kherson", "Mykolaiv", "Odessa") ~ "South Ukraine",
    region %in% c("Dnipropetrovsk", "Donetsk", "Kharkiv", "Luhansk", "Zaporizhia") ~ "East Ukraine",
    region %in% c("Chernivtsi", "Cherkasy", "Ivano-Frankivsk", "Khmelnytskyi",
                  "Lviv", "Poltava", "Rivne", "Ternopil", "Vinnytsia",
                  "Volyn", "Zakarpattia") ~ "West Ukraine",
    region == "Autonomous" ~ "Crimea",
    region %in% c("Kirovohrad", "Poltava", "Vinnytsia") ~ "Central Ukraine",
    TRUE ~ NA_character_  # Assign NA if the region does not match any specified condition

# Step 1: Remove rows with NA in the predicted probability column
heatmap_data <- heatmap_data %>%
  filter(!is.na(predicted_probability)) %>%  # Exclude rows where predicted_probability is NA
  arrange(sub_region, predicted_probability)  # Order by sub-region and probability for a cleaner look

# Step 2: Create the heatmap plot with consistent formatting
heatmap_plot <- ggplot(heatmap_data, aes(x = season, y = reorder(region, predicted_probability), fill = predicted_probability)) +

  # Create tiles for each region-season combination and add detailed tooltips
  geom_tile(color = "white", aes(text = paste("Region:", region, "<br>",
                                              "Sub-region:", sub_region, "<br>",
                                              "Predicted Probability:", round(predicted_probability, 2)))) +

  # Enhanced colour gradient
  scale_fill_gradientn(colors = c("lightblue", "yellow", "orange", "darkred"), na.value = "grey50") +

  # Add titles, axis labels, and legend title
  labs(
    title = "Predicted Probabilities of War-related Fires",
    subtitle = "Seasonal average probability by region in Ukraine",
    x = "Season",
    y = "Region",
    fill = "Predicted Probability"
  ) +

  # Consistent theme settings for Arial font
  theme_minimal() +
  theme(
    plot.title = element_text(size = 18, face = "bold", hjust = 0.5, family = "Arial"),
    plot.subtitle = element_text(size = 14, hjust = 0.5, family = "Arial"),
    axis.title.x = element_text(size = 14, family = "Arial"),  # Increased font size for readability
    axis.title.y = element_text(size = 14, family = "Arial"),
    axis.text.x = element_text(angle = 45, hjust = 1, size = 10, family = "Arial"),
    axis.text.y = element_text(size = 10, face = "bold", family = "Arial"),
    legend.title = element_text(size = 12, face = "bold", family = "Arial"),
    legend.text = element_text(size = 10, family = "Arial"),
    legend.key.width = unit(0.5, "cm"),
    legend.key.height = unit(2, "cm"),
    legend.position = "right",
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank()
  ) +

  # Add annotations with larger font size for better visibility
  annotate("text", x = "Autumn", y = "Luhansk", label = "High Probability", color = "black", size = 4, hjust = 0, family = "Arial",
fontface = "bold") +
  annotate("text", x = "Summer", y = "Zaporizhia", label = "High Activity", color = "black", size = 4, hjust = 0, family = "Arial",
fontface = "bold")

# Convert ggplot to an interactive plotly plot
interactive_heatmap <- ggplotly(heatmap_plot, tooltip = "text")

# Display the interactive plot
interactive_heatmap
```

Correlation plot (not using)

```
install.packages("ggcorrplot")
library(ggcorrplot)
library(ggplot2)

# Data frame with relevent columns
cor_data <- fires_cleaned %>%
  select(war_fires_per_day, fires_per_day, population_density, pop_exact, sustained_excess_fires, war_fire, excess_fires,
latitude, longitude)

# Check the structure of your data
str(cor_data)

# Convert factor columns to numeric if necessary
cor_data <- cor_data %>%
  mutate(across(where(is.factor), as.numeric)) %>%
  mutate(across(where(is.character), as.numeric))

# Calculate the correlation matrix
cor_matrix <- round(cor(cor_data), 1)
head(cor_matrix[, 1:9])

# Using hierarchical clustering
ggcorrplot(cor_matrix, hc.order = TRUE, outline.color = "white")
ggcorrplot(cor_matrix)
```