Supplement of

Fire Across Frontiers: Satellite-Based Investigation of Climate-Fire Interactions in the Middle East (West Asia)

Wan Ni Lin et al.

Correspondence to: Wan Ni Lin (wan_ni.lin@mgeo.lu.se)

Text S1 Determination of 10km unit based on Active fire hot spot test

To select an appropriate spatial resolution for the analysis, I compared results from spatial autocorrelation tests at 5 km, 10 km, and 20 km grid sizes. Using Moran's I with distance-based weights, I calculated Z-scores across a sequence of increasing distances. At the 5 km resolution, the Z-scores were very high but also highly sensitive to local variation, producing many small-scale clusters. At the 20 km resolution, Z-scores were much lower, and clustering patterns appeared overly generalized, losing finer spatial details. The 10 km resolution provided a balance between these extremes: it yielded consistently high Z-scores across multiple distance bands, while also capturing clear and interpretable clustering patterns without excessive noise. For this reason, I selected 10 km as the spatial study unit for subsequent analysis.

Figure S1.1 Spatial autocorrelation (Moran's I Z-scores) by distance at different grid resolutions (5 km)

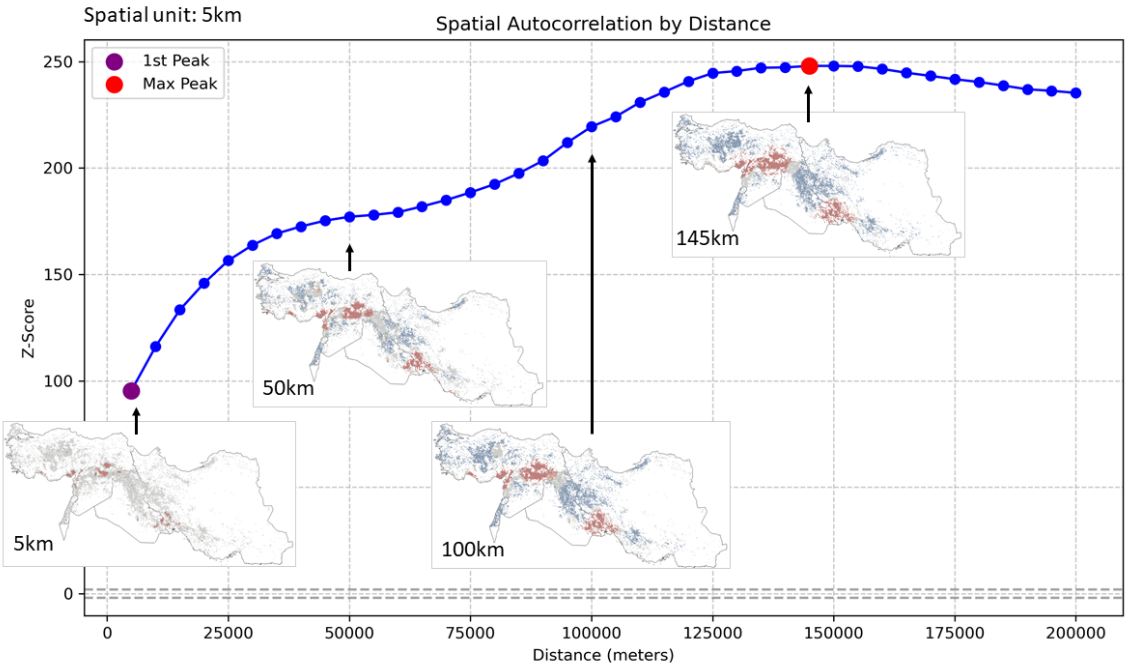


Figure S1.2 Spatial autocorrelation (Moran's I Z-scores) by distance at different grid resolutions (10 km)

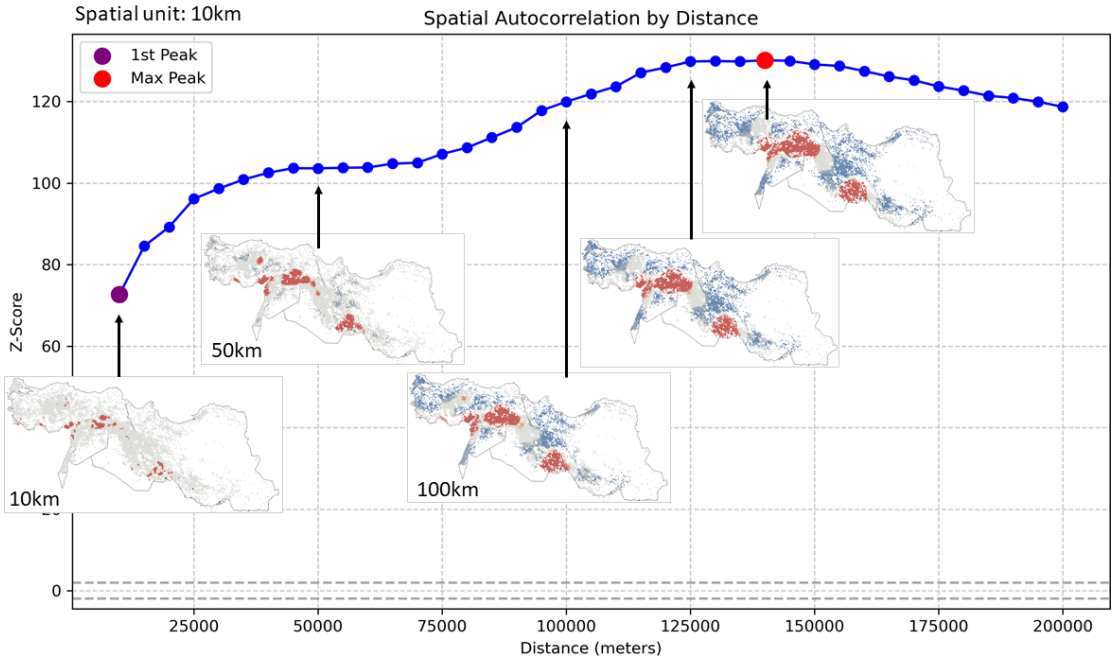


Figure S1.3 Spatial autocorrelation (Moran's I Z-scores) by distance at different grid resolutions (20 km)

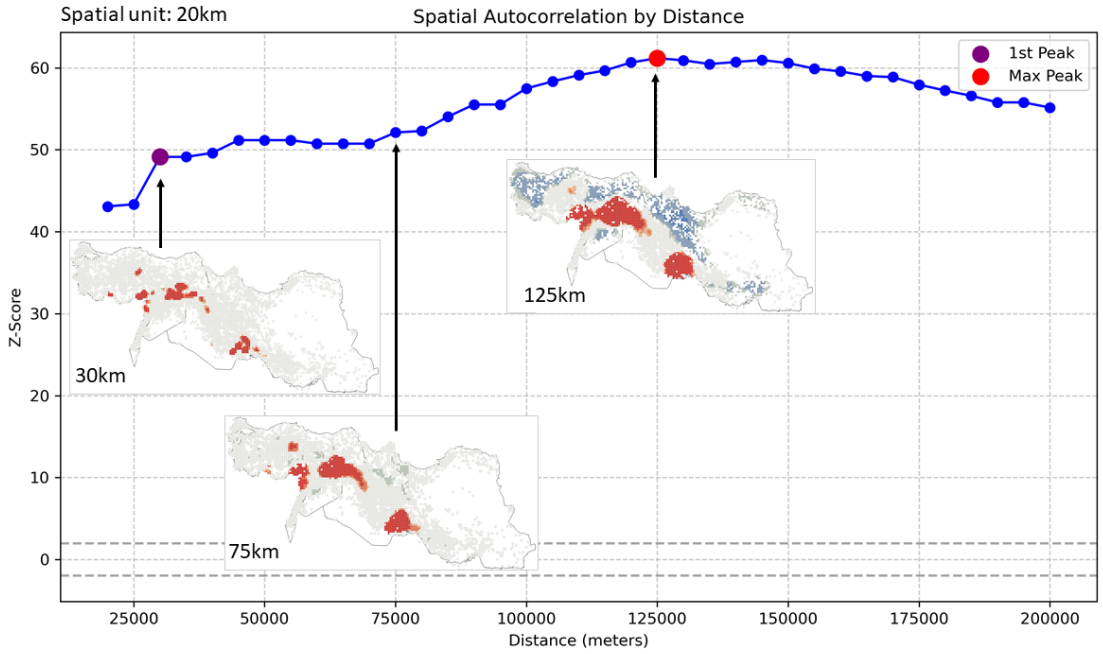


Table S1 Statistical Classification and definition for Emerging Hot spot pattern according to ArcGIS pro 3.6

Hot spot pattern	Symbol	definition
New Hot Spot		A location that is a statistically significant hot spot for the final time step and has never been a statistically significant hot spot before.
Consecutive Hot Spot		A location with a single uninterrupted run of at least two statistically significant hot spot bins in the final time-step intervals. The location has never been a statistically significant hot spot prior to the final hot spot run and less than 90 percent of all bins are statistically significant hot spots.
Intensifying Hot Spot		A location that has been a statistically significant hot spot for 90 percent of the time-step intervals, including the final time step. In addition, the intensity of clustering of high counts in each time step is increasing overall and that increase is statistically significant.
Persistent Hot Spot		A location that has been a statistically significant hot spot for 90 percent of the time-step intervals with no discernible trend in the intensity of clustering over time.
Diminishing Hot Spot		A location that has been a statistically significant hot spot for 90 percent of the time-step intervals, including the final time step. In addition, the intensity of clustering in each time step is decreasing overall and that decrease is statistically significant.
Sporadic Hot Spot		A statistically significant hot spot for the final time-step interval with a history of also being an on-again and off-again hot spot. Less than 90 percent of the time-step intervals have been statistically significant hot spots and none of the time-step intervals have been statistically significant cold spots.
Oscillating Hot Spot		A statistically significant hot spot for the final time-step interval that has a history of also being a statistically significant cold spot during a prior time step. Less than 90 percent of the time-step intervals have been statistically significant hot spots.
Historical Hot Spot		The most recent time period is not hot, but at least 90 percent of the time-step intervals have been statistically significant hot spots.

Resource: ArcGIS pro, ESRI

Table S2 Active fire statistic

The table shows the active fire count for each year in each region. The percentages represent each region's share of the total active fire counts over the entire 22-year period (unit: point count). Active fire count sum based on each year and each region between 2001 and 2022. The percentage entails the share of the active fire count in the region in the year

Year	Turkey	Syria	Iraq	Iran	Lebanon	Israel	West Bank	Gaza
2001	1202 (2%)	378 (4%)	366 (2%)	373 (1%)	3 (1%)	19 (3%)	3 (2%)	0
2002	1609 (3%)	450 (5%)	472 (2%)	1026 (4%)	25 (6%)	15 (2%)	8 (5%)	0
2003	1776 (3%)	641 (7%)	512 (2%)	1107 (4%)	7 (2%)	26 (4%)	6 (4%)	0
2004	1493 (3%)	334 (3%)	296 (1%)	894 (4%)	22 (5%)	21 (3%)	2 (1%)	0
2005	1773 (3%)	104 (1%)	252 (1%)	1083 (4%)	3 (1%)	36 (6%)	3 (2%)	0
2006	2290 (4%)	139 (1%)	267 (1%)	1195 (5%)	43 (11%)	40 (6%)	1 (1%)	0
2007	2263 (4%)	250 (3%)	457 (2%)	1703 (7%)	55 (13%)	23 (4%)	4 (3%)	0
2008	1368 (3%)	31 (0%)	889 (4%)	824 (3%)	8 (2%)	16 (3%)	10 (7%)	0
2009	5520 (11%)	252 (3%)	1208 (5%)	851 (3%)	15 (4%)	15 (2%)	1 (1%)	0
2010	3412 (7%)	444 (5%)	894 (4%)	2255 (9%)	34 (8%)	115 (18%)	5 (3%)	0
2011	4459 (9%)	209 (2%)	493 (2%)	883 (4%)	6 (1%)	18 (3%)	1 (1%)	0
2012	1534 (3%)	355 (4%)	465 (2%)	666 (3%)	15 (4%)	22 (3%)	3 (2%)	0
2013	2648 (5%)	656 (7%)	584 (3%)	1084 (4%)	29 (7%)	36 (6%)	4 (3%)	0
2014	1544 (3%)	265 (3%)	983 (4%)	1144 (5%)	10 (2%)	19 (3%)	0 (0%)	1(100%)
2015	4886 (9%)	723 (7%)	1979 (9%)	895 (4%)	14 (3%)	22 (3%)	8 (5%)	0
2016	3289 (6%)	443 (5%)	1206 (5%)	1411 (6%)	21 (5%)	30 (5%)	21 (14%)	0
2017	2184 (4%)	234 (2%)	719 (3%)	1206 (5%)	9 (2%)	17 (3%)	1 (1%)	0
2018	1614 (3%)	138 (1%)	1136 (5%)	924 (4%)	3 (1%)	23 (4%)	2 (1%)	0
2019	1292 (2%)	2128 (22%)	4522 (21%)	1650 (7%)	17 (4%)	43 (7%)	20 (14%)	0
2020	1848 (4%)	1408 (14%)	2253 (10%)	1503 (6%)	32 (8%)	29 (5%)	20 (14%)	0
2021	3227 (6%)	160 (2%)	1127 (5%)	1520 (6%)	29 (7%)	31 (5%)	20 (14%)	0
2022	846 (2%)	96 (1%)	915 (4%)	804 (3%)	8 (2%)	16 (3%)	4 (3%)	0
sum	52077	9838	21995	25001	408	632	147	1
average/ year	2367.4	447.18	999.77	1136.41	18.55	28.73	6.68	0.05

Figure S2 Distribution of the sum of the total active fire in the Middle East between 2001 and 2022. Unit: point counts

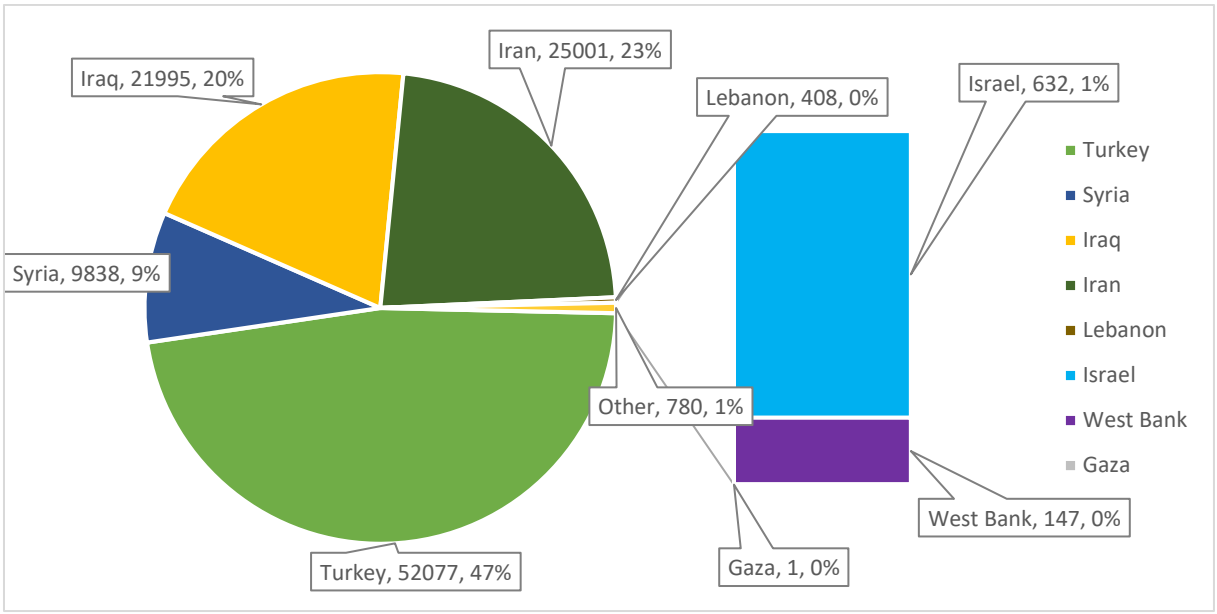


Table S3 Burned area statistic in the Middle East

The table shows the burned area for each year in each region. The percentages represent each region's share of the total burned area over the entire 22-year period (unit: km2). Burned area sum based on each year and each region between 2001 and 2022. The percentage entails the share of the burned area in the region in the year

Year	Turkey	Syria	Iraq	Iran	Lebanon	Israel	West Bank	Gaza
2001	5352.6 (5%)	802.9 (4%)	952.8 (2%)	638.4 (1%)	0.0 (0%)	6.5 (2%)	3.0 (6%)	0
2002	3824.5 (3%)	375.9 (2%)	1184.8 (2%)	3178.9 (6%)	0.2 (0%)	9.4 (3%)	5.7 (12%)	0
2003	3194.8 (3%)	533.5 (3%)	1216.1 (2%)	2324.2 (4%)	1.4 (2%)	18.4 (5%)	0.0 (0%)	0
2004	3974.6 (3%)	284.2 (2%)	527.5 (1%)	3112.0 (6%)	2.3 (3%)	15.9 (5%)	6.4 (14%)	0
2005	3602.3 (3%)	140.8 (1%)	473.1 (1%)	2870.3 (5%)	0.0 (0%)	21.8 (7%)	2.5 (5%)	0
2006	5730.0 (5%)	113.2 (1%)	551.5 (1%)	3252.4 (6%)	14.9 (19%)	20.5 (6%)	0.4 (1%)	0
2007	3870.2 (3%)	290.4 (2%)	963.8 (2%)	4173.2 (8%)	14.1 (18%)	19.7 (6%)	1.1 (2%)	0
2008	2369.4 (2%)	4.9 (0%)	1451.8 (3%)	2165.7 (4%)	0.0 (0%)	4.3 (1%)	1.6 (4%)	0
2009	12587 (11%)	308.5 (2%)	878.9 (2%)	1778.6 (3%)	0.0 (0%)	0.2 (0%)	0.0 (0%)	0
2010	7645.5 (7%)	474.3 (3%)	1915.8 (4%)	5582.1 (10%)	8.5 (11%)	54.5 (16%)	1.5 (3%)	0
2011	9378.7 (8%)	134.1 (1%)	730.6 (1%)	2709.9 (5%)	0.4 (1%)	6.2 (2%)	0.0 (0%)	0
2012	2562.3 (2%)	222.5 (1%)	456.1 (1%)	1048.1 (2%)	0.0 (0%)	19.1 (6%)	0.0 (0%)	0
2013	5953.3 (5%)	732.1 (4%)	885.3 (2%)	1988.9 (4%)	2.3 (3%)	43.1 (13%)	0.0 (0%)	0
2014	3463.4 (3%)	213.6 (1%)	1631.6 (3%)	2048.1 (4%)	0.0 (0%)	7.0 (2%)	0.0 (0%)	0
2015	9485.6 (8%)	1254.8 (7%)	3391.1 (7%)	1924.3 (4%)	0.9 (1%)	8.1 (2%)	8.1 (18%)	0
2016	7871.4 (7%)	301.6 (2%)	2895.8 (6%)	2467.6 (5%)	0.0 (0%)	24.4 (7%)	1.1 (2%)	0
2017	4566.3 (4%)	262.9 (1%)	1379.3 (3%)	2243.4 (4%)	1.5 (2%)	2.7 (1%)	4.1 (9%)	0
2018	3071.8 (3%)	112.6 (1%)	1384.3 (3%)	1595.3 (3%)	0.0 (0%)	1.1 (0%)	0.0 (0%)	0
2019	3650.8 (3%)	7913.6 (44%)	15395.6 (32%)	3176.0 (6%)	11.6 (15%)	17.3 (5%)	2.4 (5%)	0
2020	4642.7 (4%)	3159.2 (17%)	8109.5 (17%)	3366.4 (6%)	10.8 (14%)	9.9 (3%)	0.8 (2%)	0
2021	5802.1 (5%)	360.1 (2%)	1691.0 (3%)	1839.3 (3%)	9.1 (12%)	23.8 (7%)	7.2 (16%)	0
2022	2967.6 (3%)	133.2 (1%)	694.0 (1%)	938.6 (2%)	0.0 (0%)	0.6 (0%)	0.0 (0%)	0
sum	115566.7	18128.9	48760.2	54421.7	78.0	334.4	46.0	0
average/ year	5253.0	824.0	2216.4	2473.7	3.5	15.2	2.1	0

Figure S3 Distribution of the sum of the total burned area in the Middle East between 2001 and 2022. Unit: km2

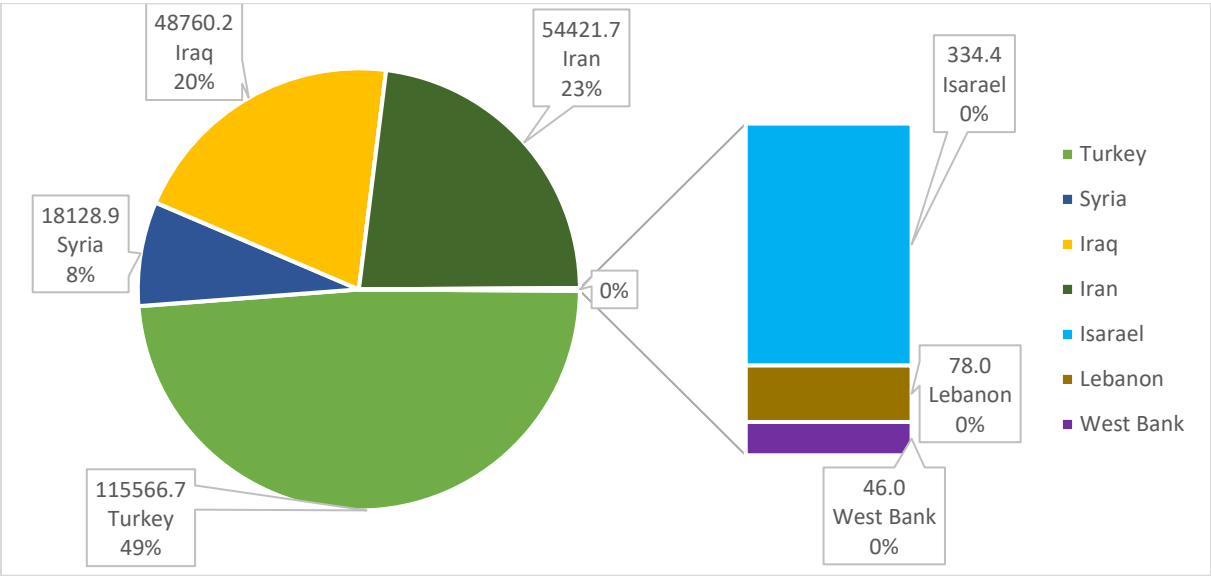
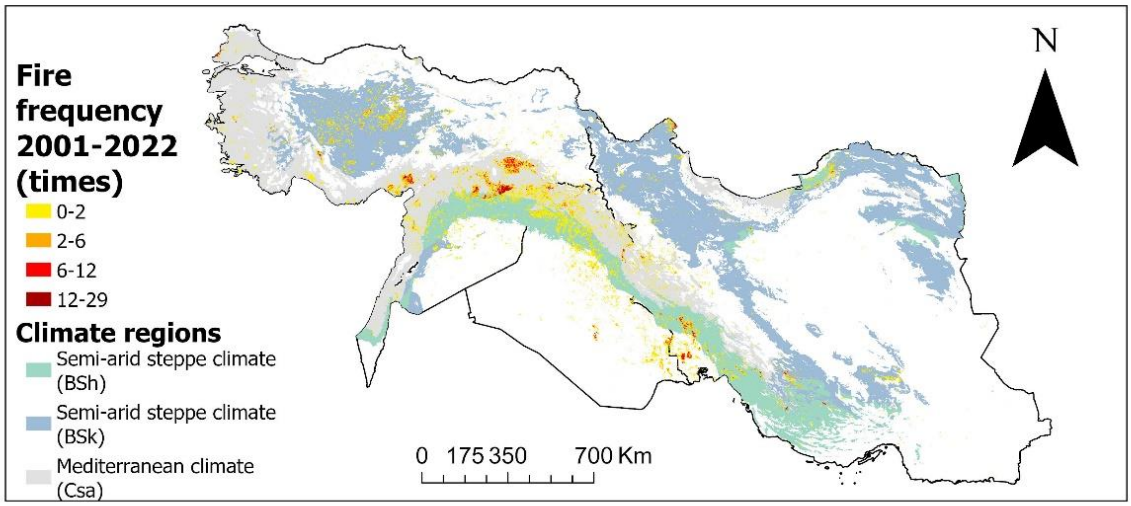


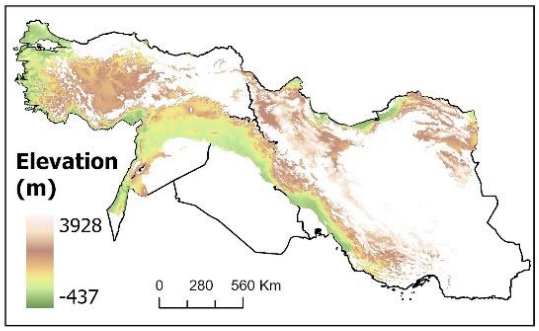
Figure S4 Spatial heterogeneity in Mediterranean climate region

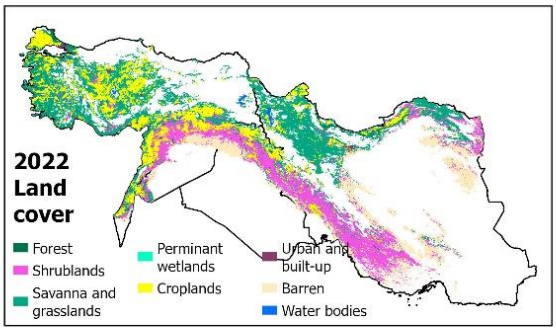
The variables included in this study are vegetation fire frequency (2001–2022), mean temperature during the 2022 dry season, mean temperature during the wet season, mean precipitation during the dry season, mean precipitation during the wet season, land cover (2020), and elevation (2020).

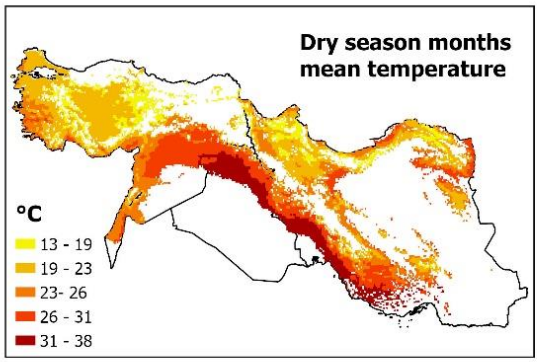
The dry season is defined as May to September, while the wet season is defined as October to April. Monthly temperature and precipitation data were obtained from the ERA5-Land dataset and aggregated to seasonal means. Elevation data were derived from ASTER GDEM. Land cover and fire frequency data were obtained from MODIS products.

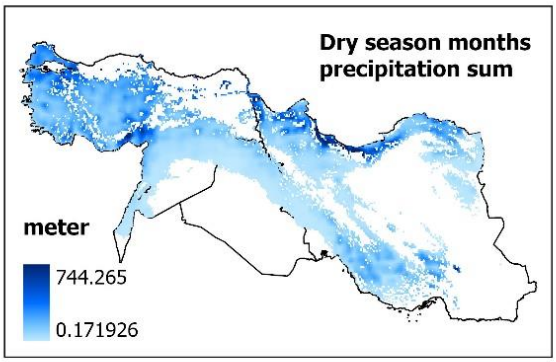
To define the climate zones, we used the historical Köppen–Geiger climate classification map (1991–2020) from Beck et al. (2023). From this dataset, we extracted the Mediterranean climate (Csa) and semi-arid steppe climates (BSh and BSk) to visualize the fire prone areas with diverse landscapes.

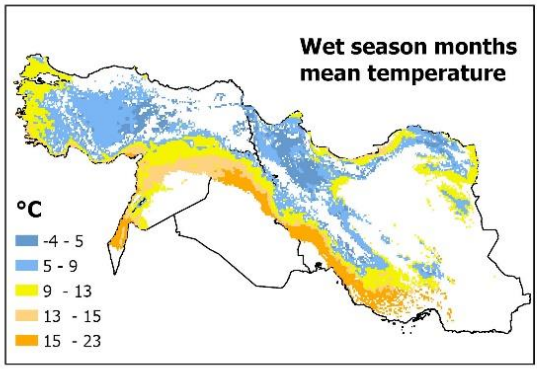

Fire frequency 2001-2022 (times)
0-2
2-6
6-12
12-29
Climate regions
Semi-arid steppe climate (BSh)
Semi-arid steppe climate (BSk)
Mediterranean climate (Csa)
N
0 175 350 700 Km

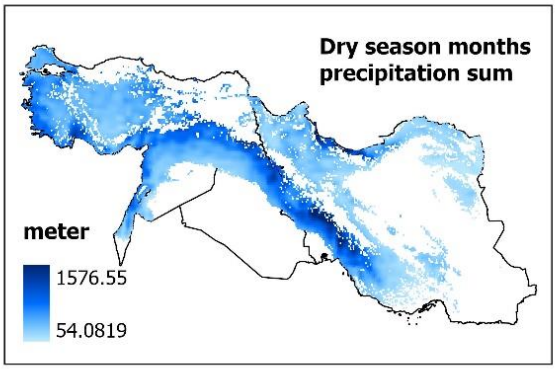

Elevation (m)
3928
-437
0 280 560 Km


2022 Land cover
Forest
Shrublands
Savanna and grasslands
Perminant wetlands
Croplands
Urban and built-up
Barren
Water bodies


Dry season months mean temperature
°C
13 - 19
19 - 23
23- 26
26 - 31
31 - 38


Dry season months precipitation sum
meter
744.265
0.171926


Wet season months mean temperature
°C
-4 - 5
5 - 9
9 - 13
13 - 15
15 - 23


Dry season months precipitation sum
meter
1576.55
54.0819

Code availability

Data analysis of this research article was conducted on R 4.3.3. The following sections provide codes for Mann Kendall analysis and Spearman correlation analysis. The Mann Kendall trend test original code was provided by the second author of the article, Abdulhakim M. Abdi. correlation analysis code

Mann Kendall trend test code

The following code starts from aggregating monthly burned area data to yearly data, and then conduct trend analysis.

```
## Load required packages
require(parallel)

require(trend)

require(zyp)

require(raster)

require(Kendall)


library(sf)

library(snow)

library(dplyr)

library(tidyr)
```

Part 1: Data preparation

```
# Define paths
stack_file <- file.path("BA10km", "BAStack_2001_2022_10km.tif")
pixel_grid <- st_read("BA10km/ME_grid_10km.shp")

# Standardize grid ID column name
names(pixel_grid)[names(pixel_grid) == "Id"] <- "grid_id"

# Create output directory
output_dir <- "BA10km_yearly2"
if (!dir.exists(output_dir)) {
  dir.create(output_dir)
}

# Load the original monthly stack
monthly_stack <- stack(stack_file)
print("Original monthly stack information:")

print(monthly_stack)

# Set up years
start_year <- 2001
end_year <- 2022
total_years <- end_year - start_year + 1
months_per_year <- 12
```

Part 2: Monthly burned area aggregation to yearly data

```
aggregate_to_yearly <- function(monthly_stack, start_year, total_years, mon
ths_per_year, output_dir) {
  yearly_stack_list <- list()

  for (year_idx in 1:total_years) {
    year <- start_year + year_idx - 1

    # Calculate start and end indices for this year
    start_idx <- (year_idx - 1) * months_per_year + 1
    end_idx <- year_idx * months_per_year

    # Check if we have all months for this year
    if (end_idx <= nlayers(monthly_stack)) {
      # Get the layers for the current year
      year_layers <- subset(monthly_stack, start_idx:end_idx)

      # Sum the monthly data to get yearly total
      yearly_sum <- calc(year_layers, fun = sum, na.rm = TRUE)

      # Set names for the yearly layer
      names(yearly_sum) <- paste0("BA_", year)

      # Save the yearly sum to disk
      output_file <- file.path(output_dir, paste0("BA_", year, ".tif"))
      writeRaster(yearly_sum, filename = output_file, format = "GTiff", ove
rwrite = TRUE)

      # Add to our list of yearly rasters
      yearly_stack_list[[year_idx]] <- yearly_sum

      cat("Processed year:", year, "\n")
    } else {
      cat("Skipping year", year, "- incomplete data\n")
    }
  }

  # Stack all the yearly rasters
  yearly_stack <- stack(yearly_stack_list)

  # Save the complete yearly stack
  writeRaster(yearly_stack, filename = file.path(output_dir, "BA_yearly_sta
ck.tif"),
              format = "GTiff", overwrite = TRUE)

  return(yearly_stack)
}

# Create the yearly stack
yearly_stack <- aggregate_to_yearly(monthly_stack, start_year, total_years,
 months_per_year, output_dir)

print("Yearly stack information:")
```

```r
print(yearly_stack)

# Ensure grid CRS matches raster CRS
if (st_crs(pixel_grid) != st_crs(yearly_stack)) {
  pixel_grid <- st_transform(pixel_grid, crs(yearly_stack))
}

print(paste("Loaded existing grid with", nrow(pixel_grid), "cells"))
```

Part 3: Mann Kendall trend analysis functions

```r
sigsen = function(x, p = 0.05, prewhitening = TRUE,
                  method = c("yuepilon", "zhang"), df = FALSE,...) {

  # if only one unique value exists in 'x', return NA
  if (length(unique(x)) == 1)
    return(NA)

  # with prewhitening
  if (prewhitening) {

    # try to compute pre-whitened mann-kendall trend test
    try(ss <- zyp::zyp.trend.vector(x, method = method[1]), silent = TRUE)

    # if previous computation fails, return NA
    if (!exists("ss")) {
      sig <- sen <- NA

      # else return sen slope and referring p value
    } else {

      id_sig <- grep("sig", names(ss))
      sig <- ss[id_sig]

      id_sen <- grep("trend$", names(ss))
      sen <- ss[id_sen]
    }

    # without prewhitening
  } else {

    snsl <- trend::sens.slope(x,conf.level=p)

    sig <- snsl$p.value
    sen <- snsl$estimates
  }

  # return data.frame
  if (df) {
    return(data.frame(sen = sen, p = sig))

    # reject value of slope if p >= 0.05
  } else {

    if (is.logical(sig) | is.logical(p)) {
```

```
      return(NA)
    } else {
      if (sig >= p) {
        return(NA)
        # keep value of slope if p < 0.05
      } else {
        return(sen)
      }
    }
  }
}

sigtau = function(x, p = 0.05, prewhitening = TRUE,
                  method = c("yuepilon", "zhang"), df = FALSE,...) {

  # if only one unique value exists in 'x', return NA
  if (length(unique(x)) == 1)
    return(NA)

  # with prewhitening
  if (prewhitening) {

    # try to compute pre-whitened mann-kendall trend test
    try(mkt <- zyp::zyp.trend.vector(x, method = method[1]), silent = TRUE)

    # if previous computation fails, return NA
    if (!exists("mkt")) {
      sig <- tau <- NA

      # else return kendall's tau and referring p value
    } else {

      id_sig <- grep("sig", names(mkt))
      sig <- mkt[id_sig]

      id_tau <- grep("tau", names(mkt))
      tau <- mkt[id_tau]
    }

    # without prewhitening
  } else {

    mk <- Kendall::MannKendall(x)

    sig <- mk$sl
    tau <- mk$tau
  }

  # return data.frame
  if (df) {
    return(data.frame(tau = tau, p = sig))

    # reject value of tau if p >= 0.05
  } else {
```

```
    if (is.logical(sig) | is.logical(p)) {
      return(NA)
    } else {
      if (sig >= p) {
        return(NA)
        # keep value of tau if p < 0.05
      } else {
        return(tau)
      }
    }
  }
}
```

Part 4: Run Mann Kendall function

```
# Set output file names
senslope_filename <- file.path(output_dir, "BA_10km_allY_senslope_p0.05.tif
")
tautrend_filename <- file.path(output_dir, "BA_10km_allY_kendalltau_p0.05.t
if")

# Initiate cluster for parallel computing
beginCluster(8)  # Adjust the number of cores based on your system

# Calculate Sen's slope
senslope <- raster::clusterR(
  x = yearly_stack,
  raster::calc,
  args = list(fun = sigsen),
  filename = senslope_filename,
  overwrite = TRUE
)

# Calculate Kendall's Tau
tautrend <- raster::clusterR(
  x = yearly_stack,
  raster::calc,
  args = list(fun = sigtau),
  filename = tautrend_filename,
  overwrite = TRUE
)

endCluster()

print("Mann-Kendall analysis completed")
```

Dataset construction (annual burned area and independent variables)

```
library(sf)
library(raster)
library(dplyr)
library(tidyr)
library(exactextractr)
library(stringr)
library(lubridate)
library(ggplot2)
```

Part1 Create annual burned area rasters

```
aggregate_to_yearly <- function(monthly_stack, start_year, total_years, mon
ths_per_year, output_dir) {
  yearly_stack_list <- list()

  for (year_idx in 1:total_years) {
    year <- start_year + year_idx - 1

    # Calculate start and end indices for this year
    start_idx <- (year_idx - 1) * months_per_year + 1
    end_idx <- year_idx * months_per_year

    # Check if we have all months for this year
    if (end_idx <= nlayers(monthly_stack)) {
      # Get the layers for the current year
      year_layers <- subset(monthly_stack, start_idx:end_idx)

      # Sum the monthly data to get yearly total (PIXEL-LEVEL AGGREGATION!)
      yearly_sum <- calc(year_layers, fun = sum, na.rm = TRUE)

      # Set names for the yearly layer
      names(yearly_sum) <- paste0("BA_", year)

      # Save the yearly sum to disk
      output_file <- file.path(output_dir, paste0("BA_", year, ".tif"))
      writeRaster(yearly_sum, filename = output_file, format = "GTiff", ove
rwrite = TRUE)

      # Add to our list of yearly rasters
      yearly_stack_list[[year_idx]] <- yearly_sum

      cat("Processed year:", year, "\n")
    } else {
      cat("Skipping year", year, "- incomplete data\n")
    }
  }

  # Stack all the yearly rasters
  yearly_stack <- stack(yearly_stack_list)

  # Save the complete yearly stack
  writeRaster(yearly_stack, filename = file.path(output_dir, "BA_yearly_sta
ck.tif"),
```

```
              format = "GTiff", overwrite = TRUE)

  return(yearly_stack)
}

# Set up parameters
start_year <- 2001
end_year <- 2022
total_years <- end_year - start_year + 1
months_per_year <- 12
output_dir <- "BA10km_yearly_CodeB


# Create output directory if it doesn't exist
if (!dir.exists(output_dir)) {
  dir.create(output_dir)
}

# Load monthly stack and create yearly rasters
cat("Loading monthly burned area stack...\n")
monthly_stack <- stack("BA10km/BAStack_2001_2022_10km.tif")
cat("Monthly stack layers:", nlayers(monthly_stack), "\n")

# Create yearly rasters using Code A's method
yearly_stack <- aggregate_to_yearly(monthly_stack, start_year, total_years,
 months_per_year, output_dir)
cat("Yearly stack created with", nlayers(yearly_stack), "layers\n\n")
```

Part 2 prepare spatial data

```
# Load vector data
grid <- st_read("GridData/ME_grid_10km.shp")
grid$FID <- 1:nrow(grid)

# Load the sample area
area <- st_read("ME_boundary_FAO.shp")
area_clean <- area %>%
  dplyr::select(-any_of(c("Shape_Leng", "STATUS", "geomType", "EXP1_YEAR",
"Shape_Area",
                          "type", "DISP_AREA", "XMax", "Xmin", "YMax", "Ymi
n", "STR1_YEAR"))) %>%
  st_as_sf()

# Intersect the grid with the sample area
grid_filtered <- st_intersection(grid, area_clean)

cat("Grid features after intersection:", nrow(grid_filtered), "\n")
```

Part 3 extract burned area

```
years <- 2001:2022

# Create matrix to store burned area data
ba_matrix <- matrix(NA, nrow = nrow(grid_filtered), ncol = length(years))
colnames(ba_matrix) <- years
```

```r
for (i in seq_along(years)) {
  year <- years[i]
  yearly_file <- file.path(output_dir, paste0("BA_", year, ".tif"))

  if (file.exists(yearly_file)) {
    cat("Processing year", year, "...\n")
    yearly_raster <- raster(yearly_file)

    # Extract using exact_extract with 'mean'
    yearly_values <- exact_extract(yearly_raster, grid_filtered, 'mean')
    ba_matrix[, i] <- yearly_values

    # Report statistics
    non_zero_count <- sum(yearly_values > 0, na.rm = TRUE)
    if (non_zero_count > 0) {
      cat("  Grids with burning:", non_zero_count,
          " | Max:", sprintf("%.4f", max(yearly_values, na.rm = TRUE)),
          " | Mean of burning grids:", sprintf("%.4f", mean(yearly_values[y
early_values > 0], na.rm = TRUE)), "\n")
    } else {
      cat("  No burning detected this year\n")
    }

  } else {
    cat("Warning: File not found:", yearly_file, "\n")
    ba_matrix[, i] <- rep(NA, nrow(grid_filtered))
  }
}

# Verify temporal variation
yearly_totals <- colSums(ba_matrix, na.rm = TRUE)
cat("\nTemporal variation verification:\n")
cat("Min yearly total:", sprintf("%.1f", min(yearly_totals)), "\n")
cat("Max yearly total:", sprintf("%.1f", max(yearly_totals)), "\n")
cat("Coefficient of variation:", sprintf("%.3f", sd(yearly_totals) / mean(y
early_totals)), "\n")

if (sd(yearly_totals) / mean(yearly_totals) > 0.05) {
  cat("SUCCESS: Good temporal variation in burned area!\n\n")
} else {
  cat("WARNING: Low temporal variation in burned area\n\n")
}
```

Part 4 Filter grids and extract coordinates

```r
# Filter grids with any burned area
has_burning <- apply(ba_matrix, 1, function(x) any(x > 0 & !is.na(x)))
valid_indices <- which(has_burning)
valid_grids <- grid_filtered[valid_indices, ]

cat("Grid filtering results:\n")
cat("- Original grids:", nrow(grid_filtered), "\n")
cat("- Grids with burning activity:", length(valid_indices), "\n")
cat("- Grids removed (never burned):", nrow(grid_filtered) - length(valid_i
ndices), "\n")
```

```r
cat("- Retention rate:", sprintf("%.1f%%", (length(valid_indices) / nrow(gr
id_filtered)) * 100), "\n\n")

# Extract coordinates for filtered grids
grid_centroids_final <- st_centroid(valid_grids)
grid_geographic_final <- st_transform(grid_centroids_final, crs = 4326)
grid_coords_geo_final <- st_coordinates(grid_geographic_final)
longitude_final <- grid_coords_geo_final[, "X"]
latitude_final <- grid_coords_geo_final[, "Y"]

cat("Coordinates extracted for", length(longitude_final), "filtered grids\n
")
cat("Longitude range:", sprintf("%.4f°E to %.4f°E", min(longitude_final), m
ax(longitude_final)), "\n")
cat("Latitude range:", sprintf("%.4f°N to %.4f°N", min(latitude_final), max
(latitude_final)), "\n\n")
```

Part 5 Extract other variables

```r
# Extract static variables for valid grids only
elevation_values <- exact_extract(raster("Elevation/ASTER_GDEM_30m_ME.tif
"), valid_grids, 'mean')
aspect_values <- exact_extract(raster("Aspect/ME_aspect.tif"), valid_grids,
 'mean')
slope_values <- exact_extract(raster("Slope/ME_slope.tif"), valid_grids, 'm
ean')
population_values <- exact_extract(raster("WorldPop_TopDown_constrained_202
0/ME_pop_TopDown_constrainedUN_2020p.tif"), valid_grids, 'mean')

# Extract land cover values (mode for each year)
land_cover_files <- list.files("MODISLCdata", pattern = "^LC[0-9]{4}_5km\\.
tif$", full.names = TRUE)
land_cover_matrix <- sapply(years, function(year) {
  file_pattern <- paste0("LC", year, "_5km.tif")
  lc_file <- land_cover_files[grepl(file_pattern, land_cover_files)]
  if (length(lc_file) > 0) {
    exact_extract(raster(lc_file), valid_grids, 'mode')
  } else {
    rep(NA, length(valid_indices))
  }
})

# Extract climate and ecoregion values (static)
climate_region_values <- exact_extract(raster("ClimateRegionData/ClimateReg
ion19912020_5km.tif"), valid_grids, 'mode')
eco_region_values <- exact_extract(raster("EcoregionData/Ecoregion5km.tif
"), valid_grids, 'mode')
```

Part 6 Extract SPEI03 (seasonal)

```r
# Function to extract seasonal SPEI statistics
extract_seasonal_spei <- function(target_year, grid_data) {
  # Wet season: Oct (target_year-1) to Apr (target_year)
  wet_months <- list(
    c(target_year-1, 10), c(target_year-1, 11), c(target_year-1, 12),
    c(target_year, 1), c(target_year, 2), c(target_year, 3), c(target_year,
```

```
 4)
  )

  # Dry season: May (target_year) to Sep (target_year)
  dry_months <- list(
    c(target_year, 5), c(target_year, 6), c(target_year, 7), c(target_year,
 8), c(target_year, 9)
  )

  # Extract SPEI values for wet season
  wet_values <- lapply(wet_months, function(month_year) {
    file_name <- sprintf("SPEI03_%04d_%02d.tif", month_year[1], month_year
[2])
    file_path <- file.path("SPEI03", "SPEI03p_tif", file_name)
    if (file.exists(file_path)) {
      exact_extract(raster(file_path), grid_data, 'mean')
    } else {
      rep(NA, nrow(grid_data))
    }
  })

  # Extract SPEI values for dry season
  dry_values <- lapply(dry_months, function(month_year) {
    file_name <- sprintf("SPEI03_%04d_%02d.tif", month_year[1], month_year
[2])
    file_path <- file.path("SPEI03", "SPEI03p_tif", file_name)
    if (file.exists(file_path)) {
      exact_extract(raster(file_path), grid_data, 'mean')
    } else {
      rep(NA, nrow(grid_data))
    }
  })

  # Convert to matrices
  wet_matrix <- do.call(cbind, wet_values)
  dry_matrix <- do.call(cbind, dry_values)

  # Calculate statistics
  spei_stats <- data.frame(
    spei_wet_mean = apply(wet_matrix, 1, mean, na.rm = TRUE),
    spei_wet_median = apply(wet_matrix, 1, median, na.rm = TRUE),
    spei_wet_min = apply(wet_matrix, 1, min, na.rm = TRUE),
    spei_wet_max = apply(wet_matrix, 1, max, na.rm = TRUE),
    spei_dry_mean = apply(dry_matrix, 1, mean, na.rm = TRUE),
    spei_dry_median = apply(dry_matrix, 1, median, na.rm = TRUE),
    spei_dry_min = apply(dry_matrix, 1, min, na.rm = TRUE),
    spei_dry_max = apply(dry_matrix, 1, max, na.rm = TRUE)
  )

  return(spei_stats)
}

# Extract SPEI values for all years
spei_list <- lapply(years, function(yr) {
```

```
  cat("Processing SPEI for year:", yr, "\n")
  extract_seasonal_spei(yr, valid_grids)
})

# Combine SPEI data
spei_combined <- do.call(rbind, spei_list)


spei_combined[] <- lapply(spei_combined, function(x) {
  x[is.infinite(x)] <- NA
  return(x)
})
```

Part 7 Extract SPEI12

```
spei12_list <- vector("list", length(years))
for (i in seq_along(years)) {
  current_year <- years[i]
  previous_year <- current_year - 1

  cat("Processing SPEI12 for year:", current_year, "(using Dec", previous_y
ear, ")\n")

  # File name for December of previous year
  file_name <- sprintf("SPEI12_%04d_%02d.tif", previous_year, 12)
  file_path <- file.path("SPEI12", "SPEI12p_tif", file_name)

  if (file.exists(file_path)) {
    spei12_values <- exact_extract(raster(file_path), valid_grids, 'mean')
  } else {
    cat("Warning: File not found:", file_path, "\n")
    spei12_values <- rep(NA, nrow(valid_grids))
  }

  spei12_list[[i]] <- spei12_values
}

# Combine SPEI12 data into a single vector
spei12_combined <- unlist(spei12_list)
```

Part 8 Extract NDVI (Annual NDVI SUM)

```
# Process NDVI from monthly mean files to create annual SUM (vegetation cap
acity)
process_ndvi_annual_sums <- function(years, grid_data, ndvi_dir = "NDVIdata
") {

  cat("Calculating annual SUM NDVI (vegetation capacity)...\n")

  ndvi_annual_list <- vector("list", length(years))

  for (i in seq_along(years)) {
    current_year <- years[i]
    cat("Processing SUM NDVI for year", current_year, "\n")

    # Create list of expected monthly files for this year
```

```
    monthly_files <- sprintf("%s/NDVI_mean%04d_%02d.tif", ndvi_dir, current
_year, 1:12)

    # Check which files actually exist
    existing_files <- monthly_files[file.exists(monthly_files)]

    if (length(existing_files) == 0) {
      cat("  No monthly files found for", current_year, "\n")
      ndvi_annual_list[[i]] <- rep(NA, nrow(grid_data))
      next
    }

    cat("  Found", length(existing_files), "out of 12 monthly files\n")

    # Extract values from each monthly file
    monthly_values_list <- list()

    for (j in seq_along(existing_files)) {
      monthly_raster <- raster(existing_files[j])
      monthly_values <- exact_extract(monthly_raster, grid_data, 'mean')
      monthly_values_scaled <- monthly_values * 0.0001  # Apply scale facto
r
      monthly_values_list[[j]] <- monthly_values_scaled
    }

    # Convert to matrix and calculate annual SUM
    monthly_matrix <- do.call(cbind, monthly_values_list)

    # Calculate annual SUM (vegetation capacity/productivity)
    annual_sum <- apply(monthly_matrix, 1, function(x) {
      if (sum(!is.na(x)) >= 6) {  # Require at least 6 months of data
        observed_sum <- sum(x, na.rm = TRUE)
        n_observed <- sum(!is.na(x))
        annual_total <- observed_sum * (12 / n_observed)  # Adjust for miss
ing months
        return(annual_total)
      } else {
        return(NA)
      }
    })

    # Report statistics
    valid_values <- annual_sum[!is.na(annual_sum)]
    if (length(valid_values) > 0) {
      cat("  Annual SUM NDVI range:", sprintf("%.2f to %.2f", min(valid_val
ues), max(valid_values)), "\n")
    }

    ndvi_annual_list[[i]] <- annual_sum
  }

  return(unlist(ndvi_annual_list))
}
```

```r
# Process NDVI for all years using SUM method
ndvi_combined <- process_ndvi_annual_sums(years, valid_grids)
```

Part 9 ROI ID

```r
# Create ROI IDs (Code B used full country boundary, so create single ROI)
roi_id_values_final <- rep("ME_Country", length(valid_indices))

cat("ROI ID assignment completed - all grids assigned to 'ME_Country'\n")
```

Part 10 Create final dataset

```r
final_data <- data.frame(
  grid_id = rep(valid_indices, each = length(years)),
  roi_id = rep(roi_id_values_final, each = length(years)),
  year = rep(years, times = length(valid_indices)),
  longitude = rep(longitude_final, each = length(years)),
  latitude = rep(latitude_final, each = length(years)),
  burned_area = as.vector(t(ba_matrix[valid_indices, ])),
  land_cover = as.vector(t(land_cover_matrix)),
  elevation = rep(elevation_values, each = length(years)),
  aspect = rep(aspect_values, each = length(years)),
  slope = rep(slope_values, each = length(years)),
  population = rep(population_values, each = length(years)),
  climate_region = rep(climate_region_values, each = length(years)),
  eco_region = rep(eco_region_values, each = length(years)),
  spei_combined,
  spei12 = spei12_combined,
  ndvi = ndvi_combined,
  stringsAsFactors = FALSE
)
```

Part 11 dataset validation

```r
# Check burned area unit
ba_sample <- final_data$burned_area[!is.na(final_data$burned_area)]
if (length(ba_sample) > 0) {
  cat("Burned area statistics (Code A method):\n")
  cat("  Min:", sprintf("%.0f", min(ba_sample)), "\n")
  cat("  Max:", sprintf("%.0f", max(ba_sample)), "\n")
  cat("  Mean:", sprintf("%.0f", mean(ba_sample)), "\n")

  # Check if values are in m² range
  if (mean(ba_sample) > 10000) {
    cat("  Values appear to be in m²\n")
  } else {
    cat("  Values appear to be in hectares\n")
  }
}

# Save the final dataset
write.csv(final_data, "year__dataset.csv", row.names = FALSE)
```

Spearman rank correlation (include and exclude crop)

Part 1 load and data cleaning

```
dataset <- read.csv("year__dataset.csv", row.names = NULL)

clean_dataNA_no0fire<- dataset %>%
  filter(!is.na(burned_area) & !is.na(population) & !is.na(elevation) & !i
s.na(slope)
         & !is.na(aspect)& !is.na(spei_wet_mean)& !is.na(spei_wet_median)&
 !is.na(spei_wet_max)
         & !is.na(spei_wet_min)& !is.na(spei_dry_mean)& !is.na(spei_dry_med
ian)& !is.na(spei_dry_max)
         & !is.na(spei_dry_min)& !is.na(spei12)& !is.na(ndvi)& !is.na(spei1
2)& land_cover != 0& burned_area !=0 &!land_cover %in% c(13, 15, 17))

nocrop_dataNA_no0fire <- dataset%>%
  filter(!is.na(burned_area) & !is.na(population) & !is.na(elevation) & !i
s.na(slope)
         & !is.na(aspect)& !is.na(spei_wet_mean)& !is.na(spei_wet_median)&
 !is.na(spei_wet_max)
         & !is.na(spei_wet_min)& !is.na(spei_dry_mean)& !is.na(spei_dry_med
ian)& !is.na(spei_dry_max)
         & !is.na(spei_dry_min)& !is.na(spei12)& !is.na(ndvi)& !is.na(spei1
2)& land_cover != 0& burned_area !=0 &!land_cover %in% c(12, 13,14,  15, 1
7))
```

Part 2 Correlation for data including cropland fire

```
variables <- c("elevation", "aspect", "slope", "population","spei_wet_mean
","spei_wet_median","spei_wet_max","spei_wet_min",
               "spei_dry_mean","spei_dry_median","spei_dry_max","spei_dry_m
in", "spei12", "ndvi")


correlation_results_withcrop <- data.frame(
  variable = character(),
  correlation = numeric(),
  p_value = numeric(),
  stringsAsFactors = FALSE
)


for(var in variables) {
  test_result <- cor.test(
    clean_dataNA_no0fire$burned_area,
    clean_dataNA_no0fire[[var]],
    method = "spearman",
    exact = FALSE,
    use = "complete.obs"
  )

  correlation_results_withcrop <- rbind(correlation_results_withcrop,
                                        data.frame(
                                          variable = var,
```

```
                                  correlation = test_result$estimate,
                                  p_value = test_result$p.value
                                ))
}

print(correlation_results_withcrop)


#plot the correlation
library(corrplot)
library(dplyr)
# Select your numeric variables
cor_data <- clean_dataNA_no0fire %>%
  dplyr::select(burned_area, population, elevation, slope, aspect,
                spei_wet_mean, spei_wet_median, spei_wet_max, spei_wet_min,
                spei_dry_mean, spei_dry_median, spei_dry_max, spei_dry_min,
                spei12, ndvi)
# Rename variables to shorter names
cor_variables_clean <- cor_data %>%
  rename(
    BA = burned_area,
    Population = population,
    Elevation = elevation,
    Slope = slope,
    Aspect = aspect,
    SPEI_Wet_Mean = spei_wet_mean,
    SPEI_Wet_Med = spei_wet_median,
    SPEI_Wet_Max = spei_wet_max,
    SPEI_Wet_Min = spei_wet_min,
    SPEI_Dry_Mean = spei_dry_mean,
    SPEI_Dry_Med = spei_dry_median,
    SPEI_Dry_Max = spei_dry_max,
    SPEI_Dry_Min = spei_dry_min,
    SPEI12 = spei12,
    NDVI = ndvi
  )

# Calculate the correlation matrix
cor_matrix <- cor(cor_data, method = "spearman", use = "complete.obs")

# Create the plot with a higher resolution and adjusted text sizes
png("correlation_plot_withcrop_no0fire.png", width = 1800, height = 2500, r
es = 300)

# The `par()` function sets graphical parameters.
# `mar` is adjusted to provide space for the top labels and the title.
par(mar = c(5, 4, 18, 2) + 0.1)

# The `corrplot()` function arguments are now configured for the desired lo
wer triangle plot.
corrplot(cor_matrix,
         method = "color",
         type = "lower", # To get the lower triangular shape
         addCoef.col = "black",
```

```
         tl.cex = 0.9,
         tl.col = "black",
         number.cex = 0.6,
         tl.srt = 45, # Rotates the top labels to prevent overlap
         col = colorRampPalette(c("#2166AC", "#92C5DE", "#FFFFBF", "#FDAE61
", "#D73027"))(200))

# The `title()` function is adjusted to fit and be positioned correctly.
title("Spearman burned area correlation coefficients \n(croplands included)
", line = 10, cex.main = 1.4)

# Save the plot
dev.off()

# Sort by absolute correlation value
correlation_results_withcrop <- correlation_results_withcrop[order(abs(corr
elation_results_withcrop$correlation), decreasing = TRUE), ]
print("Results sorted by correlation strength:")
print(correlation_results_withcrop)
```

Part 3 Correlation for data excluding cropland fire

```
variables <- c("elevation", "aspect", "slope", "population","spei_wet_mean
","spei_wet_median","spei_wet_max","spei_wet_min",
               "spei_dry_mean","spei_dry_median","spei_dry_max","spei_dry_m
in", "spei12", "ndvi")
correlation_results_withoutcrop <- data.frame(
  variable = character(),
  correlation = numeric(),
  p_value = numeric(),
  stringsAsFactors = FALSE
)
for(var in variables) {
  # Perform the Spearman's correlation test between 'burned_area' and the c
urrent variable.
  test_result <- cor.test(
    nocrop_dataNA_no0fire$burned_area,
    nocrop_dataNA_no0fire[[var]],
    method = "spearman",
    exact = FALSE,
    use = "complete.obs"
  )

  # Store the results in the 'correlation_results_withoutcrop data frame.
  # A new row is created for each variable and appended using `rbind()`.
  correlation_results_withoutcrop <- rbind(correlation_results_withoutcrop,
                                data.frame(
                                  variable = var,
                                  correlation = test_result$estimate,
                                  p_value = test_result$p.value
                                ))
}

for(var in variables) {
  # Perform the correlation test for the second data frame.
```

```r
  test_result <- cor.test(
    nocrop_dataNA_no0fire$burned_area,
    nocrop_dataNA_no0fire[[var]],
    method = "spearman",
    exact = FALSE,
    use = "complete.obs"
  )

  # Append these new results to the same 'correlation_results' data frame.
  correlation_results_withoutcrop <- rbind(correlation_results_withoutcrop,
                                data.frame(
                                  variable = var,
                                  correlation = test_result$estimate,
                                  p_value = test_result$p.value
                                ))
}

# Print the final data frame to the console to see all the calculated corre
lations.
print(correlation_results_withoutcrop)


#plot the correlation
library(corrplot)
library(dplyr)
# Select your numeric variables
cor_data <- nocrop_dataNA_no0fire %>%
  dplyr::select(burned_area, population, elevation, slope, aspect,
                spei_wet_mean, spei_wet_median, spei_wet_max, spei_wet_min,
                spei_dry_mean, spei_dry_median, spei_dry_max, spei_dry_min,
                spei12, ndvi)
# Rename variables to shorter names
cor_variables_clean <- cor_data %>%
  rename(
    BA = burned_area,
    Population = population,
    Elevation = elevation,
    Slope = slope,
    Aspect = aspect,
    SPEI_Wet_Mean = spei_wet_mean,
    SPEI_Wet_Med = spei_wet_median,
    SPEI_Wet_Max = spei_wet_max,
    SPEI_Wet_Min = spei_wet_min,
    SPEI_Dry_Mean = spei_dry_mean,
    SPEI_Dry_Med = spei_dry_median,
    SPEI_Dry_Max = spei_dry_max,
    SPEI_Dry_Min = spei_dry_min,
    SPEI12 = spei12,
    NDVI = ndvi
  )
# Calculate the correlation matrix
cor_matrix <- cor(cor_data, method = "spearman", use = "complete.obs")

# Create the plot with a higher resolution and adjusted text sizes
```

```r
png("correlation_plot_withoutcrop_no0fire.png", width = 1800, height = 250
0, res = 300)

# The `par()` function sets graphical parameters.
# `mar` is adjusted to provide space for the top labels and the title.
par(mar = c(5, 4, 18, 2) + 0.1)

# The `corrplot()` function arguments are now configured for the desired lo
wer triangle plot.
corrplot(cor_matrix,
         method = "color",
         type = "lower", # To get the lower triangular shape
         addCoef.col = "black",
         tl.cex = 0.9,
         tl.col = "black",
         number.cex = 0.6,
         tl.srt = 45, # Rotates the top labels to prevent overlap
         col = colorRampPalette(c("#2166AC", "#92C5DE", "#FFFFBF", "#FDAE61
", "#D73027"))(200))

# The `title()` function is adjusted to fit and be positioned correctly.
title("Spearman burned area correlation coefficients\n(croplands excluded)
", line = 10, cex.main = 1.4)

# Save the plot
dev.off()

# Sort by absolute correlation value
correlation_results_withoutcrop <- correlation_results_withoutcrop[order(ab
s(correlation_results_withoutcrop$correlation), decreasing = TRUE), ]
print("Results sorted by correlation strength:")
print(correlation_results_withoutcrop)
```

Reference

ESRI. *How Emerging Hot Spot Analysis Works—ArcGIS Pro*. Link: https://pro.arcgis.com/en/pro-app/latest/tool-reference/space-time-pattern-mining/learnmoreemerging.htm (access date: 15 October 2025)

Beck, H.E., T.R. McVicar, N. Vergopolan, A. Berg, N.J. Lutsko, A. Dufour, Z. Zeng, X. Jiang, A.I.J.M. van Dijk, D.G. Miralles. High-resolution (1 km) Köppen-Geiger maps for 1901–2099 based on constrained CMIP6 projections, Scientific Data 10, 724, doi:10.1038/s41597-023–02549-6 (2023).