Geospatial Data Science - Final project

Analysis of US road accidents and construction sites

Moritz Peist Noemi Lucchi Simon Vellin

Contents

```r
library(data.table)   # For faster data manipulation
library(tidyverse)    # For data manipulation and visualization
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.4
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::between()     masks data.table::between()
## x dplyr::filter()      masks stats::filter()
## x dplyr::first()       masks data.table::first()
## x lubridate::hour()    masks data.table::hour()
## x lubridate::isoweek() masks data.table::isoweek()
## x dplyr::lag()         masks stats::lag()
## x dplyr::last()        masks data.table::last()
## x lubridate::mday()    masks data.table::mday()
## x lubridate::minute()  masks data.table::minute()
## x lubridate::month()   masks data.table::month()
## x lubridate::quarter() masks data.table::quarter()
## x lubridate::second()  masks data.table::second()
## x purrr::transpose()   masks data.table::transpose()
## x lubridate::wday()    masks data.table::wday()
## x lubridate::week()    masks data.table::week()
## x lubridate::yday()    masks data.table::yday()
## x lubridate::year()    masks data.table::year()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(sf)           # For spatial data handling
```

```
## Linking to GEOS 3.12.2, GDAL 3.9.3, PROJ 9.4.1; sf_use_s2() is TRUE
```

```r
library(leaflet)      # For interactive maps
library(leaflet.extras) # For additional leaflet features
library(mapview)      # For easier map visualization
library(tmap)         # For thematic maps
library(tigris)       # For US road networks
```

```
## To enable caching of data, set `options(tigris_use_cache = TRUE)`
## in your R script or .Rprofile.
```

```r
library(future)       # For parallel processing
library(future.apply) # For parallel processing with apply functions
library(sf)           # For spatial data handling

# Create directories if they don't exist
if (!dir.exists("./data/tigris")) {
  dir.create("./data/tigris", recursive = TRUE)
}

# Set custom cache directory (optional)
options(tigris_cache_dir = "./data/tigris")
# Configure tigris to use caching
options(tigris_use_cache = TRUE)

# Load Caltrans State Highway Network
aadt <- st_read("data/Traffic_Volumes_AADT/Traffic_Volumes_AADT.shp")
```

```
## Reading layer `Traffic_Volumes_AADT' from data source
##   `C:\Users\mmpei\OneDrive\Uni\BSE\Trimester 2\Geospatial\bse_geospatial\final\data\Traffic_Volumes_A
##   using driver `ESRI Shapefile'
## Simple feature collection with 13874 features and 15 fields
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: -13833100 ymin: 3834975 xmax: -12723710 ymax: 5161800
## Projected CRS: WGS 84 / Pseudo-Mercator
```

```r
# Load accidents

# Efficient approach
df.acc <- fread("data/us_accidents/US_accidents_March23.csv")[
  # Filter date range of 2021
  lubridate::year(as.Date(Start_Time)) == 2021 &
  # And California
  State == "CA"
][, `:=`(
  # Add year, quarter, month columns
  year = data.table::year(Start_Time),
  quarter = data.table::quarter(Start_Time),
  month = data.table::month(Start_Time),
  # Calculate duration (assuming End_Time exists in the dataset)
  duration = as.numeric(difftime(End_Time, Start_Time, units = "days"))
)] %>%
  as_tibble()  # Convert to tibble only at the end for performance
```

```r
df.const <- fread("data/us_constructions/US_constructions_Dec21.csv")[
  # Filter date range of 2021
  lubridate::year(as.Date(Start_Time)) == 2021 &
  # And California
  State == "CA"
][, `:=`(
  # Add year, quarter, month columns
  year = year(Start_Time),
  quarter = quarter(Start_Time),
  month = month(Start_Time),
```

```r
  # Calculate duration (assuming End_Time exists in the dataset)
  duration = as.numeric(difftime(End_Time, Start_Time, units = "days"))
)][
  duration > 1  # Filter out constructions lasting less than a day - majority
] %>%
  as_tibble()  # Convert to tibble only at the end for performance
```

```r
# California Road Construction Safety Analysis
# Examining the causal impact of road construction on traffic accidents

# Load libraries
library(data.table)   # For faster data manipulation
library(tidyverse)    # For data manipulation and visualization
library(sf)           # For spatial data handling
library(leaflet)      # For interactive maps
library(leaflet.extras) # For additional leaflet features
library(tigris)       # For US geographic data
library(fixest)       # For fixed effects models
library(lubridate)    # For date handling
library(patchwork)    # For combining plots
library(RColorBrewer) # For color palettes

# 1. Data Exploration -------------------------------------------------------

# Quick summaries
cat("Accidents dataset:", nrow(df.acc), "records\n")
```

```
## Accidents dataset: 341876 records
```

```r
cat("Construction dataset:", nrow(df.const), "records\n")
```

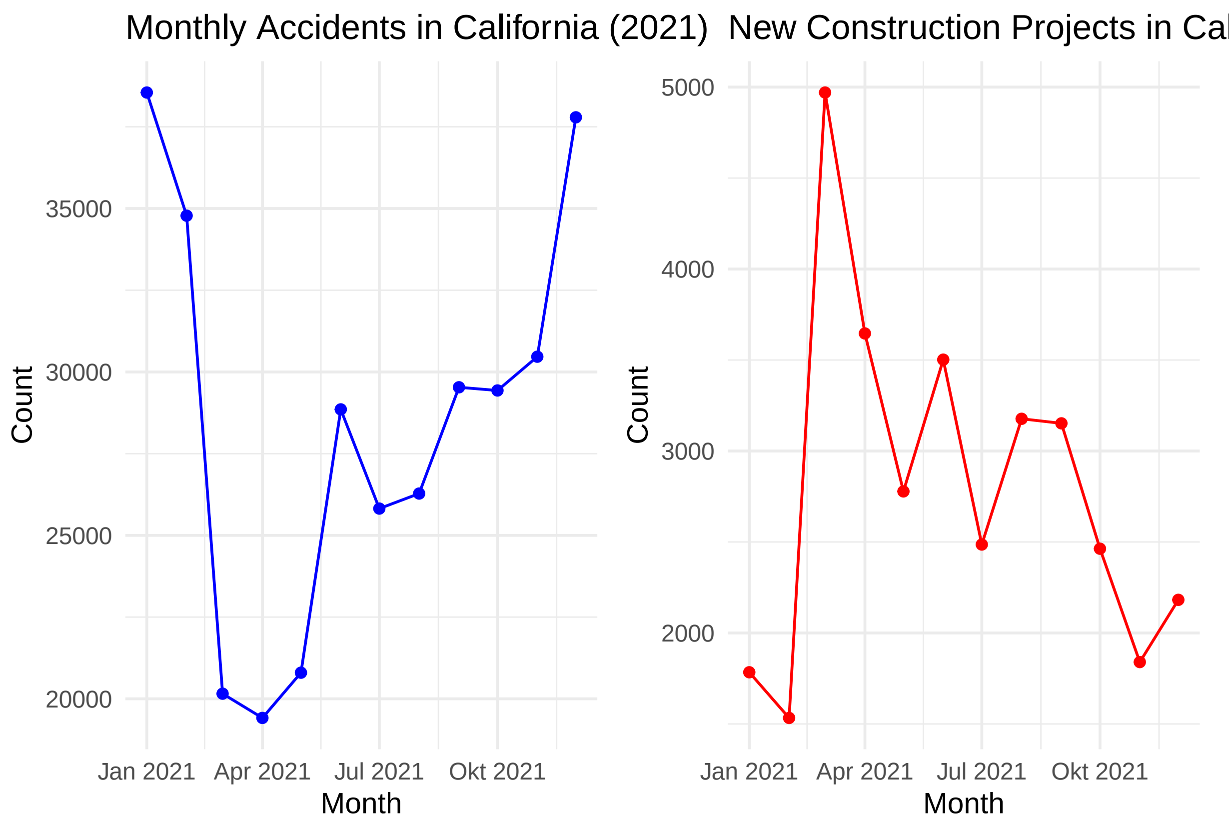
```
## Construction dataset: 33513 records
```

```r
cat("AADT dataset:", nrow(aadt), "records\n")
```

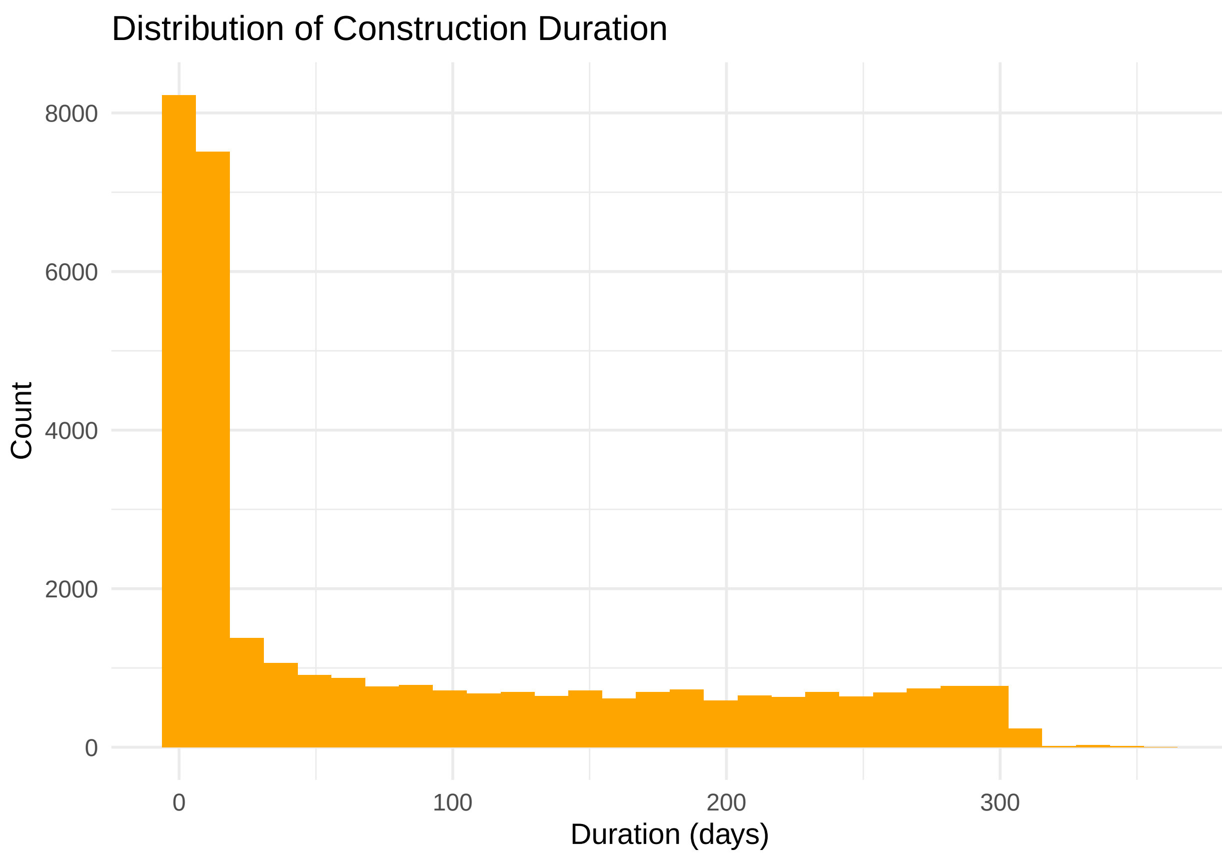
```
## AADT dataset: 13874 records
```

```r
# Check temporal distribution of data
acc_monthly <- df.acc %>%
  mutate(month = floor_date(as.Date(Start_Time), "month")) %>%
  count(month)

const_monthly <- df.const %>%
  mutate(month = floor_date(as.Date(Start_Time), "month")) %>%
  count(month)

# Plot monthly patterns
p1 <- ggplot(acc_monthly, aes(x = month, y = n)) +
  geom_line(color = "blue") +
  geom_point(color = "blue") +
  labs(title = "Monthly Accidents in California (2021)",
       x = "Month", y = "Count") +
  theme_minimal()

p2 <- ggplot(const_monthly, aes(x = month, y = n)) +
  geom_line(color = "red") +
```

```r
  geom_point(color = "red") +
  labs(title = "New Construction Projects in California (2021)",
       x = "Month", y = "Count") +
  theme_minimal()

p1 + p2
```




```r
# Check construction durations
df.const %>%
  ggplot(aes(x = duration)) +
  geom_histogram(bins = 30, fill = "orange") +
  labs(title = "Distribution of Construction Duration",
       x = "Duration (days)", y = "Count") +
  theme_minimal()
```




```r
# 2. Spatial Processing ----------------------------------------------------

# Check if we need to convert to sf objects
if (!inherits(df.acc, "sf")) {
  df.acc.sf <- df.acc %>%
    filter(!is.na(Start_Lat) & !is.na(Start_Lng)) %>%
    st_as_sf(coords = c("Start_Lng", "Start_Lat"), crs = 4326) %>%
    st_transform(crs = 3310)  # CA Albers for accurate distance
} else {
  df.acc.sf <- df.acc
}

if (!inherits(df.const, "sf")) {
  df.const.sf <- df.const %>%
    filter(!is.na(Start_Lat) & !is.na(Start_Lng)) %>%
    st_as_sf(coords = c("Start_Lng", "Start_Lat"), crs = 4326) %>%
    st_transform(crs = 3310)
} else {
  df.const.sf <- df.const
}

# Create construction buffers (500m)
const_buffers <- df.const.sf %>%
  st_buffer(dist = 500)  # 500m buffer

# Add buffer metadata
```

```r
const_buffers <- const_buffers %>%
  mutate(
    buffer_id = paste0("c", row_number()),
    start_date = as.Date(Start_Time),
    end_date = as.Date(End_Time),
    construction_duration = as.numeric(difftime(end_date, start_date, units = "days"))
  )

# 3. Urban-Rural Classification --------------------------------------------

# Download urban areas for California
ca_urban_areas <- tigris::urban_areas(cb = TRUE, year = 2019) %>%
  st_transform(st_crs(const_buffers)) %>%
  # Filter to just California urban areas using the NAME10 column
  filter(grepl(", CA$", NAME10))

# Classify construction sites as urban or rural
const_urban <- st_join(
  const_buffers,
  ca_urban_areas %>% select(urban_name = NAME10),
  join = st_intersects
)

const_urban <- const_urban %>%
  mutate(is_urban = !is.na(urban_name))

# Visualize urban/rural distribution
urban_rural_counts <- const_urban %>%
  st_drop_geometry() %>%
  count(is_urban) %>%
  mutate(percentage = n / sum(n) * 100)

print(urban_rural_counts)
```

```
## # A tibble: 2 x 3
##   is_urban     n percentage
##   <lgl>    <int>      <dbl>
## 1 FALSE     9803       29.2
## 2 TRUE     23795       70.8
```

```r
# 4. Identify Accidents in Construction Zones -------------------------------

# Join accidents to construction buffers
accidents_in_buffers <- st_join(
  df.acc.sf %>%
    select(accident_id = ID, accident_date = Start_Time, Severity),
  const_urban %>%
    select(buffer_id, construction_id = ID, start_date, end_date, is_urban),
  join = st_intersects
)

# Classify accidents by timing relative to construction
accidents_in_buffers <- accidents_in_buffers %>%
  mutate(
```

```r
    accident_date = as.Date(accident_date),
    time_period = case_when(
      is.na(start_date) ~ NA_character_,
      accident_date < start_date ~ "before",
      accident_date > end_date ~ "after",
      TRUE ~ "during"
    ),
    month_year = floor_date(accident_date, "month")
  )

# Count accidents by construction period
period_counts <- accidents_in_buffers %>%
  st_drop_geometry() %>%
  filter(!is.na(time_period)) %>%
  count(time_period)

print(period_counts)
```

```
## # A tibble: 3 x 2
##   time_period      n
##   <chr>        <int>
## 1 after       381112
## 2 before      590600
## 3 during      308601
```

```r
# 5. Create Panel Dataset for DiD Analysis ---------------------------------

# Generate all buffer-month combinations
all_months <- seq(as.Date("2021-01-01"), as.Date("2021-12-01"), by = "month")
panel_grid <- expand_grid(
  buffer_id = unique(const_urban$buffer_id),
  month = all_months
)

# Join buffer characteristics
panel_grid <- panel_grid %>%
  left_join(
    const_urban %>%
      st_drop_geometry() %>%
      select(buffer_id, construction_id = ID, start_date, end_date, is_urban),
    by = "buffer_id"
  )
```

```
## Warning in left_join(., const_urban %>% st_drop_geometry() %>% select(buffer_id, : Detected an unexpe
## i Row 5473 of `x` matches multiple rows in `y`.
## i Row 1 of `y` matches multiple rows in `x`.
## i If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.
```

```r
# Create treatment indicator
panel_grid <- panel_grid %>%
  mutate(
    treatment = month >= start_date & month <= end_date,
    post_treatment = month > end_date
  )
```

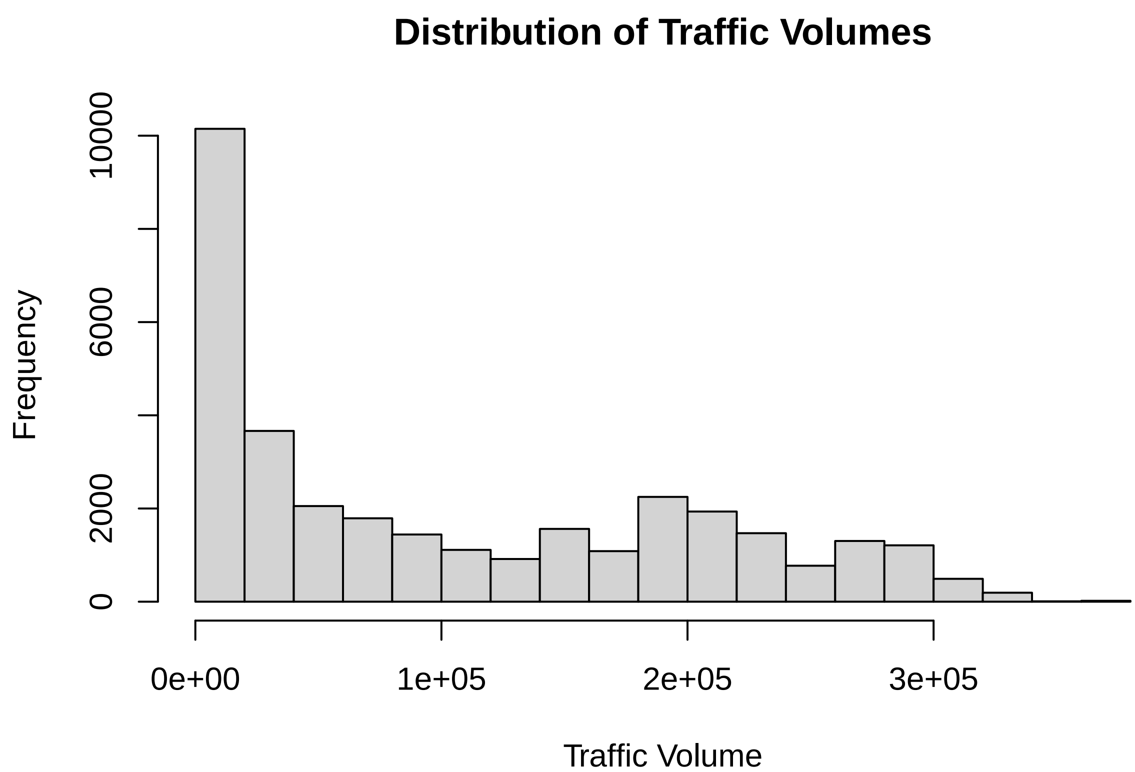
```r
# Count accidents by buffer and month
monthly_accidents <- accidents_in_buffers %>%
  st_drop_geometry() %>%
  mutate(month = floor_date(accident_date, "month")) %>%
  count(buffer_id, month) %>%
  rename(accident_count = n)

# Join accident counts to panel
panel_data <- panel_grid %>%
  left_join(monthly_accidents, by = c("buffer_id", "month")) %>%
  replace_na(list(accident_count = 0))

# Transform data to ensure matching CRS
aadt_sf <- st_transform(aadt, st_crs(const_buffers))

# Convert AADT columns to numeric
aadt_sf <- aadt_sf %>%
  mutate(
    BACK_AADT_num = as.numeric(as.character(BACK_AADT)),
    AHEAD_AADT_num = as.numeric(as.character(AHEAD_AADT))
  )

# Directly find the nearest AADT point for each construction buffer
nearest_indices <- st_nearest_feature(const_buffers, aadt_sf)
nearest_distances <- st_distance(const_buffers, aadt_sf[nearest_indices,], by_element = TRUE)

# Create a dataframe with buffer IDs and nearest AADT values
buffer_aadt <- const_buffers %>%
  select(buffer_id) %>%
  bind_cols(
    aadt_sf[nearest_indices, ] %>%
    select(BACK_AADT_num, AHEAD_AADT_num)
  ) %>%
  mutate(
    distance_to_nearest = as.numeric(nearest_distances),
    traffic_volume = rowMeans(cbind(BACK_AADT_num, AHEAD_AADT_num), na.rm = TRUE),
    n_points = 1
  ) %>%
  st_drop_geometry() %>%
  select(buffer_id, traffic_volume, n_points, distance_to_nearest)
```

```
## New names:
## * `geometry` -> `geometry...2`
## * `geometry` -> `geometry...5`
```

```r
# If you still want to use the buffer approach for points within a certain distance
buffer_radius <- 1000  # meters
buffer_aadt <- buffer_aadt %>%
  mutate(
    method = if_else(distance_to_nearest <= buffer_radius, "within_buffer", "nearest_feature")
  )

# Check the distribution of traffic volume values
summary(buffer_aadt$traffic_volume)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.      NA's
##     107.5   12400.0   72000.0  104450.8  192500.0  379000.0        91
```

```r
hist(buffer_aadt$traffic_volume, main="Distribution of Traffic Volumes",
     xlab="Traffic Volume", breaks=20)
```




```r
# Join to panel data
panel_data <- panel_data %>%
  left_join(buffer_aadt, by = "buffer_id")

# Create normalized accident rate
panel_data <- panel_data %>%
  mutate(
    accident_rate = ifelse(traffic_volume > 0,
                          accident_count / traffic_volume * 10000,  # Per 10,000 vehicles
                          NA)
  )

# 6. DiD Analysis -----------------------------------------------------------

# Basic DiD model
did_model <- feols(
  accident_count ~ treatment | buffer_id + month,
  data = panel_data
)

# DiD with urban/rural interaction
```

```r
did_urban_model <- feols(
  accident_count ~ treatment * is_urban | buffer_id + month,
  data = panel_data %>% filter(!is.na(is_urban))
)
```

```
## The variable 'is_urbanTRUE' has been removed because of collinearity (see $collin.var).
```

```r
# DiD with normalized accident rate
did_rate_model <- feols(
  accident_rate ~ treatment | buffer_id + month,
  data = panel_data %>% filter(!is.na(accident_rate))
)

# DiD with urban/rural heterogeneity and normalized rate
did_rate_urban_model <- feols(
  accident_rate ~ treatment * is_urban | buffer_id + month,
  data = panel_data %>% filter(!is.na(accident_rate), !is.na(is_urban))
)
```

```
## The variable 'is_urbanTRUE' has been removed because of collinearity (see $collin.var).
```

```r
# Display results
summary(did_model)
```

```
## OLS estimation, Dep. Var.: accident_count
## Observations: 403,176
## Fixed-effects: buffer_id: 33,513,  month: 12
## Standard-errors: Clustered (buffer_id)
##               Estimate Std. Error t value   Pr(>|t|)
## treatmentTRUE 0.095565   0.021102 4.52864 5.9569e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## RMSE: 3.15753     Adj. R2: 0.749162
##                 Within R2: 7.927e-5
```

```r
summary(did_urban_model)
```

```
## OLS estimation, Dep. Var.: accident_count
## Observations: 403,176
## Fixed-effects: buffer_id: 33,513,  month: 12
## Standard-errors: Clustered (buffer_id)
##                              Estimate Std. Error  t value   Pr(>|t|)
## treatmentTRUE               -0.056000   0.013275 -4.21857 2.4650e-05 ***
## treatmentTRUE:is_urbanTRUE   0.216203   0.029065  7.43855 1.0423e-13 ***
## ... 1 variable was removed because of collinearity (is_urbanTRUE)
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## RMSE: 3.15737     Adj. R2: 0.749187
##                 Within R2: 1.827e-4
```

```r
summary(did_rate_model)
```

```
## OLS estimation, Dep. Var.: accident_rate
## Observations: 402,084
## Fixed-effects: buffer_id: 33,422,  month: 12
## Standard-errors: Clustered (buffer_id)
##               Estimate Std. Error t value Pr(>|t|)
```

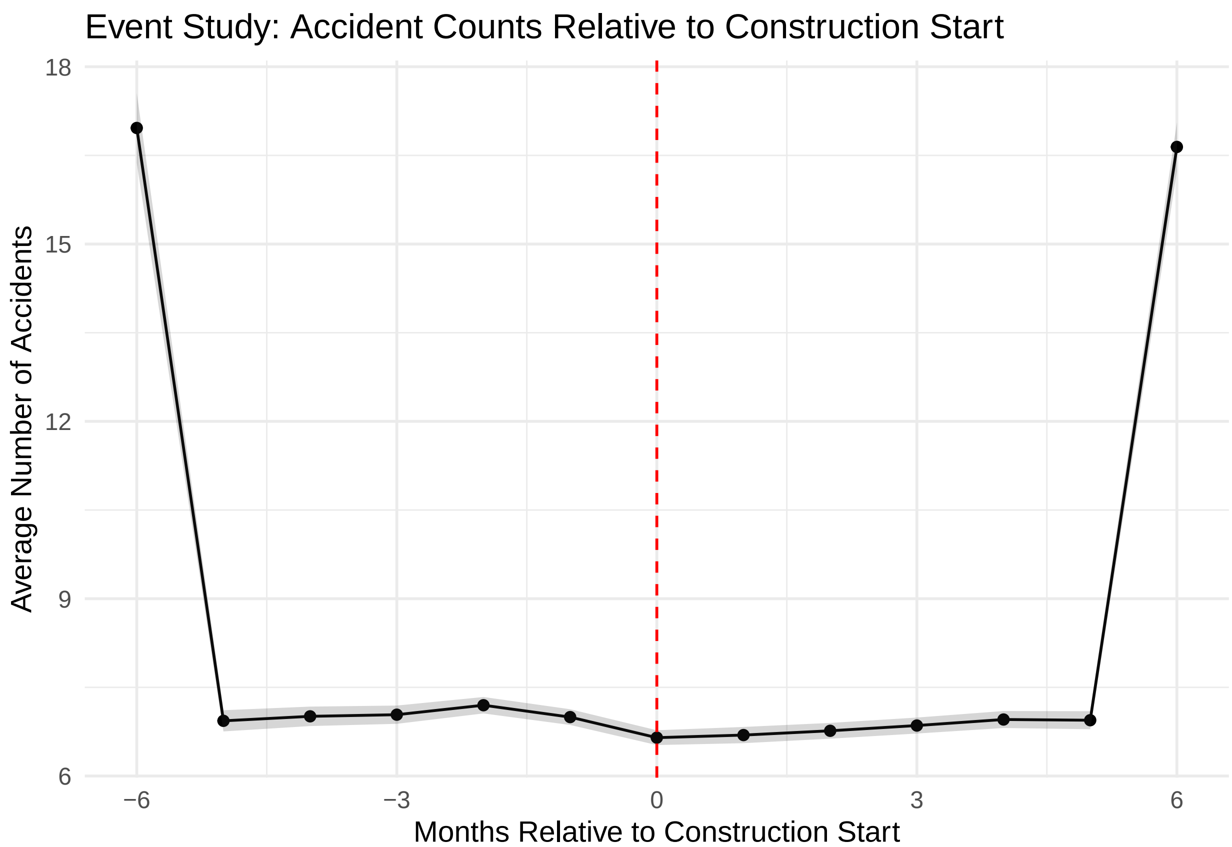
```
## treatmentTRUE  0.02592   0.016666 1.55526  0.11989
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## RMSE: 3.12652     Adj. R2: 0.178132
##                 Within R2: 5.956e-6
```

```
summary(did_rate_urban_model)
```

```
## OLS estimation, Dep. Var.: accident_rate
## Observations: 402,084
## Fixed-effects: buffer_id: 33,422,  month: 12
## Standard-errors: Clustered (buffer_id)
##                              Estimate Std. Error   t value Pr(>|t|)
## treatmentTRUE               -0.031739   0.047719 -0.665124 0.505976
## treatmentTRUE:is_urbanTRUE  0.082284   0.048022  1.713439 0.086641 .
## ... 1 variable was removed because of collinearity (is_urbanTRUE)
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## RMSE: 3.12649     Adj. R2: 0.178143
##                 Within R2: 2.127e-5
```

```
# 7. Event Study Analysis ---------------------------------------------------

# Create relative time variable (months to/from construction start)
event_data <- accidents_in_buffers %>%
  st_drop_geometry() %>%
  filter(!is.na(time_period)) %>%
  mutate(
    accident_month = floor_date(accident_date, "month"),
    construction_start_month = floor_date(start_date, "month"),
    relative_month = interval(construction_start_month, accident_month) %/% months(1)
  ) %>%
  # Group relative months beyond a certain range
  mutate(
    relative_month_grouped = case_when(
      relative_month < -6 ~ -6,
      relative_month > 6 ~ 6,
      TRUE ~ relative_month
    )
  ) %>%
  # Count accidents by relative month
  count(buffer_id, relative_month_grouped)

# Calculate average accidents by relative month
event_plot_data <- event_data %>%
  group_by(relative_month_grouped) %>%
  summarize(
    mean_accidents = mean(n),
    se = sd(n) / sqrt(n()),
    lower_ci = mean_accidents - 1.96 * se,
    upper_ci = mean_accidents + 1.96 * se
  )

# Plot event study
ggplot(event_plot_data, aes(x = relative_month_grouped, y = mean_accidents)) +
```

```r
  geom_point() +
  geom_line() +
  geom_ribbon(aes(ymin = lower_ci, ymax = upper_ci), alpha = 0.2) +
  geom_vline(xintercept = 0, linetype = "dashed", color = "red") +
  labs(
    title = "Event Study: Accident Counts Relative to Construction Start",
    x = "Months Relative to Construction Start",
    y = "Average Number of Accidents"
  ) +
  theme_minimal()
```

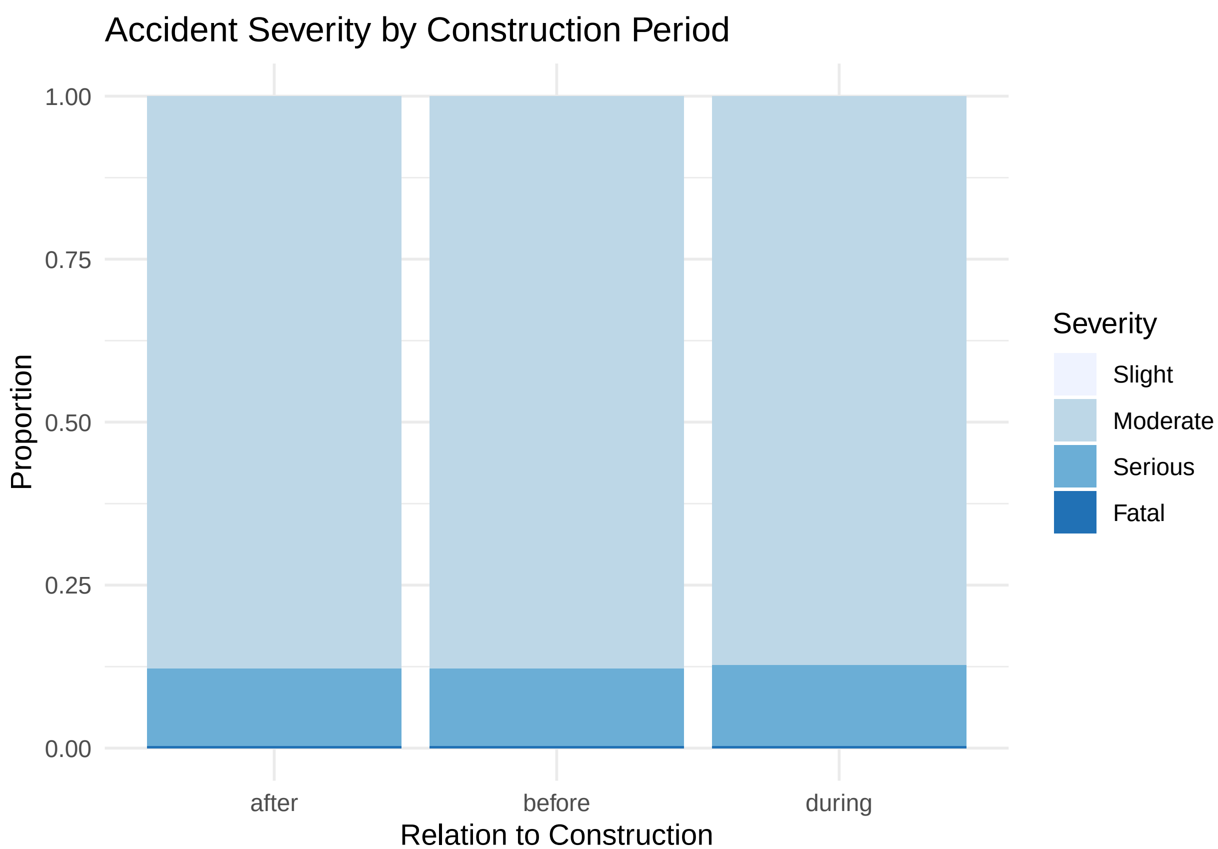



```r
# 8. Heterogeneity Analysis: Urban vs. Rural -------------------------------

# Analyze accident patterns by urban/rural status
urban_rural_patterns <- accidents_in_buffers %>%
  st_drop_geometry() %>%
  filter(!is.na(time_period), !is.na(is_urban)) %>%
  count(is_urban, time_period) %>%
  group_by(is_urban) %>%
  mutate(proportion = n / sum(n))

# Plot urban/rural patterns
ggplot(urban_rural_patterns, aes(x = time_period, y = proportion, fill = factor(is_urban))) +
  geom_col(position = "dodge") +
  scale_fill_brewer(palette = "Set1", labels = c("Rural", "Urban"), name = "Area Type") +
  labs(
```

```r
    title = "Accident Patterns by Construction Period and Area Type",
    x = "Construction Period",
    y = "Proportion of Accidents"
  ) +
  theme_minimal()
```


```r
# 9. Severity Analysis ------------------------------------------------------

# Analyze accident severity by construction period
severity_analysis <- accidents_in_buffers %>%
  st_drop_geometry() %>%
  filter(!is.na(time_period)) %>%
  count(time_period, Severity) %>%
  group_by(time_period) %>%
  mutate(proportion = n / sum(n)) %>%
  ungroup()

# Convert Severity to a factor before plotting
severity_analysis$Severity <- factor(severity_analysis$Severity,
                                      levels = c(1, 2, 3, 4),
                                      labels = c("Slight", "Moderate", "Serious", "Fatal"))
# Plot severity distribution
ggplot(severity_analysis, aes(x = time_period, y = proportion, fill = Severity)) +
  geom_col() +
  labs(
    title = "Accident Severity by Construction Period",
```

```r
    x = "Relation to Construction",
    y = "Proportion"
  ) +
  scale_fill_brewer(palette = "Blues") +
  theme_minimal()
```




```r
# 10. Summary Statistics ----------------------------------------------------

summary_stats <- panel_data %>%
  group_by(treatment, is_urban) %>%
  summarize(
    num_observations = n(),
    mean_accidents = mean(accident_count),
    median_accidents = median(accident_count),
    max_accidents = max(accident_count),
    mean_accident_rate = mean(accident_rate, na.rm = TRUE)
  ) %>%
  ungroup()
```

```
## `summarise()` has grouped output by 'treatment'. You can override using the
## `.groups` argument.
```

```r
print(summary_stats)
```

```
## # A tibble: 4 x 7
##   treatment is_urban num_observations mean_accidents median_accidents
##   <lgl>     <lgl>               <int>          <dbl>            <int>
```

```
## 1 FALSE     FALSE                91383         0.526                0
## 2 FALSE     TRUE                220203         4.23                 1
## 3 TRUE      FALSE                26253         0.505                0
## 4 TRUE      TRUE                 65337         4.55                 1
## # i 2 more variables: max_accidents <int>, mean_accident_rate <dbl>
```

```
# 11. Map Visualization ----------------------------------------------------

# Create a simplified map for visualization
#ca_state <- states(state = "CA", cb = TRUE) %>%
#  st_transform(st_crs(const_urban))

# Sample a subset of data for mapping (for performance)
#set.seed(123)
#sample_buffers <- const_urban %>%
#  sample_n(min(100, nrow(const_urban)))
#
#sample_accidents <- df.acc.sf %>%
#  sample_n(min(1000, nrow(df.acc.sf)))
#
# Create the map
#map <- leaflet() %>%
#  addProviderTiles(providers$CartoDB.Positron) %>%
#  addPolygons(data = ca_state, weight = 1, color = "#333333",
#              fillOpacity = 0.1) %>%
#  addPolygons(data = sample_buffers, color = ~ifelse(is_urban, "blue", "red"),
#              weight = 1, fillOpacity = 0.3,
#              popup = ~paste("Buffer ID:", buffer_id,
#                             "<br>Urban:", ifelse(is_urban, "Yes", "No"),
#                             "<br>Start:", start_date,
#                             "<br>End:", end_date)) %>%
#  addCircles(data = sample_accidents, radius = 50, color = "purple",
#             fillOpacity = 0.8, weight = 1)

# Display map
#map

# 12. Conclusions and Summary ----------------------------------------------

# Print main findings
cat("\nMain Findings:\n")
```

```
##
## Main Findings:
```

```
cat("- Effect of construction on accident count:", round(coef(did_model)[1], 4), "\n")
```

```
## - Effect of construction on accident count: 0.0956
```

```
cat("- Urban areas effect:", round(coef(did_urban_model)[2], 4), "\n")
```

```
## - Urban areas effect: 0.2162
```

```
cat("- Effect on accident rate:", round(coef(did_rate_model)[1], 4), "\n")
```

```
## - Effect on accident rate: 0.0259
```

```r
# Policy implications would be discussed based on results
```

```r
# Enhanced Fixed-Effects Models for Construction Safety Analysis

library(fixest)      # For advanced fixed effects models
library(lfe)         # For high-dimensional fixed effects
```

```
## Lade nötiges Paket: Matrix

##
## Attache Paket: 'Matrix'

## Die folgenden Objekte sind maskiert von 'package:tidyr':
##
##     expand, pack, unpack

##
## Attache Paket: 'lfe'

## Das folgende Objekt ist maskiert 'package:fixest':
##
##     fepois
```

```r
library(spdep)       # For spatial weights and tests
```

```
## Lade nötiges Paket: spData

## To access larger datasets in this package, install the spDataLarge
## package with: `install.packages('spDataLarge',
## repos='https://nowosad.github.io/drat/', type='source')`
```

```r
library(splm)        # For spatial panel models
library(ggplot2)     # For visualization
library(dplyr)       # For data manipulation
library(sf)          # For spatial data handling

# 1. Create spatial weights for modeling spatial dependence -----------------

# First, create a simplified representation for spatial weights calculation
# We'll use buffer centroids to create a neighbor structure
buffer_centroids <- const_buffers %>%
  st_centroid()
```

```
## Warning: st_centroid assumes attributes are constant over geometries
```

```r
# Create a data frame with coordinates for easier joining
buffer_coords <- buffer_centroids %>%
  mutate(
    X = st_coordinates(.)[,1],
    Y = st_coordinates(.)[,2]
  ) %>%
  st_drop_geometry() %>%
  select(buffer_id, X, Y)

# Create a neighbor list based on distance threshold
# (Assuming 5km as a reasonable threshold for spatial influence)
coords <- st_coordinates(buffer_centroids)
buffer_nb <- spdep::dnearneigh(coords, d1 = 0, d2 = 5000)
```

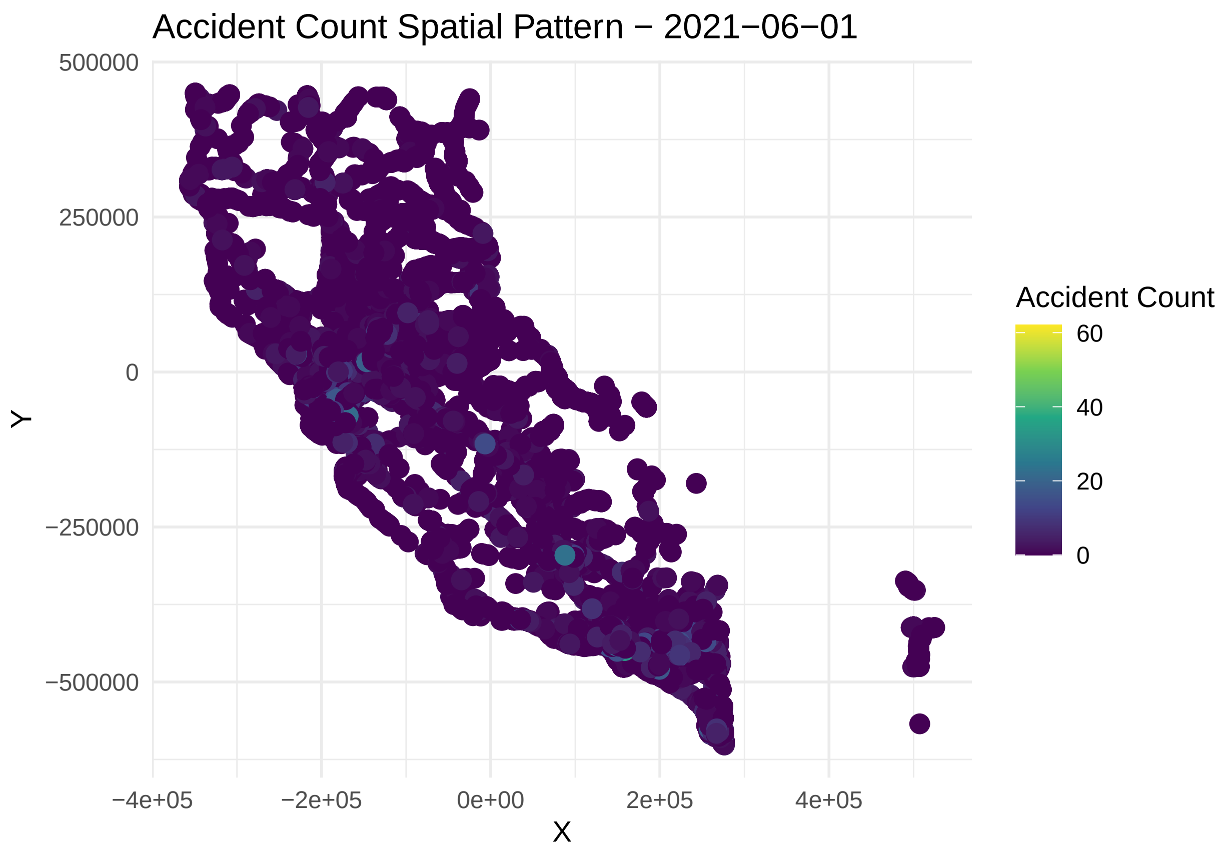
```r
# Check for isolated points
card_nb <- spdep::card(buffer_nb)
isolated_points <- which(card_nb == 0)
print(paste("Number of isolated points:", length(isolated_points)))
```

```
## [1] "Number of isolated points: 181"
```

```r
# Then convert to spatial weights matrix format with zero.policy
buffer_listw <- spdep::nb2listw(buffer_nb, style = "W", zero.policy = TRUE)

# Test for spatial autocorrelation in our outcome
# (using a sample month)
test_month <- as.Date("2021-06-01")

# Now join with panel_data
monthly_pattern <- panel_data %>%
  filter(month == test_month) %>%
  left_join(buffer_coords, by = "buffer_id")

# Alternative approach: Create weights list directly from monthly_pattern
if(nrow(monthly_pattern) > 10) {
  # Extract coordinates as a matrix
  coords_monthly <- monthly_pattern %>%
    select(X, Y) %>%
    as.matrix()

  # Create neighborhood list specific to this month's data
  monthly_nb <- spdep::dnearneigh(coords_monthly, d1 = 0, d2 = 5000)

  # Create weights list with zero policy
  monthly_listw <- spdep::nb2listw(monthly_nb, style = "W", zero.policy = TRUE)

  # Run Moran's I test with the monthly-specific weights
  moran_test <- moran.test(
    monthly_pattern$accident_count,
    monthly_listw,
    na.action = na.omit
  )

  print(moran_test)

  # Visualize the spatial pattern
  moran_map <- ggplot(monthly_pattern, aes(x = X, y = Y, color = accident_count)) +
    geom_point(size = 3) +
    scale_color_viridis_c() +
    theme_minimal() +
    labs(title = paste("Accident Count Spatial Pattern -", test_month),
         color = "Accident Count")

  print(moran_map)
}
```

```
##
##  Moran I test under randomisation
##
```

```
## data:  monthly_pattern$accident_count
## weights: monthly_listw
## n reduced by no-neighbour observations
##
## Moran I statistic standard deviate = 243.14, p-value < 2.2e-16
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##      3.785006e-01     -2.992578e-05      2.423843e-06
```




```
# 2. Enhanced Fixed Effects Models ------------------------------------------

# We'll implement several model specifications for robustness

# Model 1: Basic DiD with two-way fixed effects (buffer and time)
model_base <- feols(
  accident_count ~ treatment | buffer_id + month,
  data = panel_data
)

# Model 2: Add county fixed effects interaction with time
# First, get county for each buffer
county_info <- const_buffers %>%
  st_join(
    tigris::counties("CA", cb = TRUE) %>%
      st_transform(st_crs(const_buffers)) %>%
      select(county_name = NAME)
```

```r
  ) %>%
  st_drop_geometry() %>%
  select(buffer_id, county_name)
```

```
## Retrieving data for the year 2022
```

```r
# Join county info to panel data
panel_data <- panel_data %>%
  left_join(county_info, by = "buffer_id")
```

```
## Warning in left_join(., county_info, by = "buffer_id"): Detected an unexpected many-to-many relations
## i Row 217 of `x` matches multiple rows in `y`.
## i Row 1 of `y` matches multiple rows in `x`.
## i If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.
```

```r
# Join coordinate info to panel data for spatial models
panel_data <- panel_data %>%
  left_join(buffer_coords, by = "buffer_id")

# County x Time fixed effects model
model_county_time <- feols(
  accident_count ~ treatment | buffer_id + county_name^month,
  data = panel_data %>% filter(!is.na(county_name))
)
```

```r
# Model 3: Urban/Rural heterogeneity with county-time FE
model_urban_county <- feols(
  accident_count ~ treatment * is_urban | buffer_id + county_name^month,
  data = panel_data %>% filter(!is.na(county_name), !is.na(is_urban))
)

# Model 4: Perhaps using Severity which doesn't exist yet
model_road_chars <- feols(
  accident_count ~ treatment * is_urban | buffer_id + county_name^month,
  data = panel_data %>%
    filter(!is.na(county_name), !is.na(is_urban))
)

# Model 5: AADT-controlled model (traffic volume as control)
model_aadt <- feols(
  accident_count ~ treatment * is_urban + log(traffic_volume + 1) |
    buffer_id + county_name^month,
  data = panel_data %>%
    filter(!is.na(county_name), !is.na(traffic_volume), !is.na(is_urban))
)
```

```r
# 3. Spatial Econometric Models ---------------------------------------------

# Prepare panel data structure for spatial econometrics
spdata <- panel_data %>%
  filter(!is.na(county_name), month == test_month)

# Create spatial weights directly from the filtered data
spw_coords <- spdata %>%
  select(X, Y) %>%
```

```r
  as.matrix()

# Create neighborhood list and weights list
sp_nb <- spdep::dnearneigh(spw_coords, d1 = 0, d2 = 5000)
spw <- spdep::nb2listw(sp_nb, style = "W", zero.policy = TRUE)

# If you want to run spatial lag model (make sure spdata has all required columns)
#if(requireNamespace("spatialreg", quietly = TRUE)) {
#  spatial_model <- spatialreg::lagsarlm(
#    accident_count ~ treatment + is_urban + log(traffic_volume + 1),
#    data = spdata,
#    listw = spw,
#    zero.policy = TRUE
#  )
#  print(summary(spatial_model))
#}
library(spdep)
library(Matrix)
spatial_model <- spatialreg::lagsarlm(
  accident_count ~ treatment + is_urban + log(traffic_volume + 1),
  data = spdata,
  listw = spw,
  method = "Matrix", # This is key
  zero.policy = TRUE
)
print(summary(spatial_model))
```

```
##
## Call:spatialreg::lagsarlm(formula = accident_count ~ treatment + is_urban +
##     log(traffic_volume + 1), data = spdata, listw = spw, method = "Matrix",
##     zero.policy = TRUE)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -20.43148  -2.08865  -0.48100   0.71637  52.78005
##
## Type: lag
## Regions with no neighbours included:
##  113 313 318 368 966 1057 1111 1824 2663 2726 2799 2922 3163 3290 3920 4255 4274 4366 5080 5319 5395
## Coefficients: (numerical Hessian approximate standard errors)
##                          Estimate Std. Error  z value  Pr(>|z|)
## (Intercept)             -4.259990   0.246886 -17.2549 < 2.2e-16
## treatmentTRUE            0.615766   0.061124  10.0741 < 2.2e-16
## is_urbanTRUE            -0.522981   0.111338  -4.6972 2.637e-06
## log(traffic_volume + 1)  0.489116   0.029161  16.7731 < 2.2e-16
##
## Rho: 0.76971, LR test value: 9840.5, p-value: < 2.22e-16
## Approximate (numerical Hessian) standard error: 0.0059203
##     z-value: 130.01, p-value: < 2.22e-16
## Wald statistic: 16903, p-value: < 2.22e-16
##
## Log likelihood: -105341.4 for lag model
## ML residual variance (sigma squared): 25.345, (sigma: 5.0344)
## Number of observations: 34476
```

```
## Number of parameters estimated: 6
## AIC: 210690, (AIC for lm: 220530)
```

```r
# 4. Robust Inference with Spatial HAC Standard Errors ---------------------

# Create geographic coordinates for Conley standard errors
panel_data_with_coords <- panel_data %>%
  filter(!is.na(is_urban)) %>%
  st_as_sf(coords = c("X", "Y"), crs = 3310) %>%  # Convert to SF with CA Albers
  st_transform(crs = 4326) %>%  # Transform to WGS84 (geographic coordinates)
  mutate(
    longitude = st_coordinates(.)[,1],  # Extract longitude
    latitude = st_coordinates(.)[,2]    # Extract latitude
  ) %>%
  st_drop_geometry()  # Remove geometry column

# Now run the model with the geographic coordinates
model_spatial_se <- feols(
  accident_count ~ treatment * is_urban,
  data = panel_data_with_coords,
  vcov = vcov_conley(lat = "latitude", lon = "longitude", cutoff = 50),
  panel.id = c("buffer_id", "month")
)
```

```r
# 5. Evaluate Model Robustness ---------------------------------------------

# Compare models with different specifications
model_comparison <- etable(
  model_base, model_county_time, model_urban_county,
  model_road_chars, model_aadt, model_spatial_se,
  headers = c("Base", "County×Time", "Urban Het.",
              "Road Chars.", "AADT", "Spatial SE")#,
  #stat = c("adj.r2", "aic", "bic", "n")
)

print(model_comparison)
```

```
##                                        model_base  model_county_time
##                                              Base        County×Time
## Dependent Var.:                    accident_count     accident_count
##
## treatmentTRUE                 0.0956*** (0.0211) 0.0931*** (0.0226)
## is_urbanTRUE
## treatmentTRUE x is_urbanTRUE
## log(traffic_volume+1)
## Constant
## Fixed-Effects:                ------------------ ------------------
## buffer_id                                    Yes                Yes
## month                                        Yes                 No
## county_name-month                             No                Yes
## _____________________________ ------------------ ------------------
## S.E. type                         by: buffer_id      by: buffer_id
## Observations                             403,176            414,804
## R2                                       0.77002            0.77964
## Within R2                                7.93e-5            7.66e-5
```

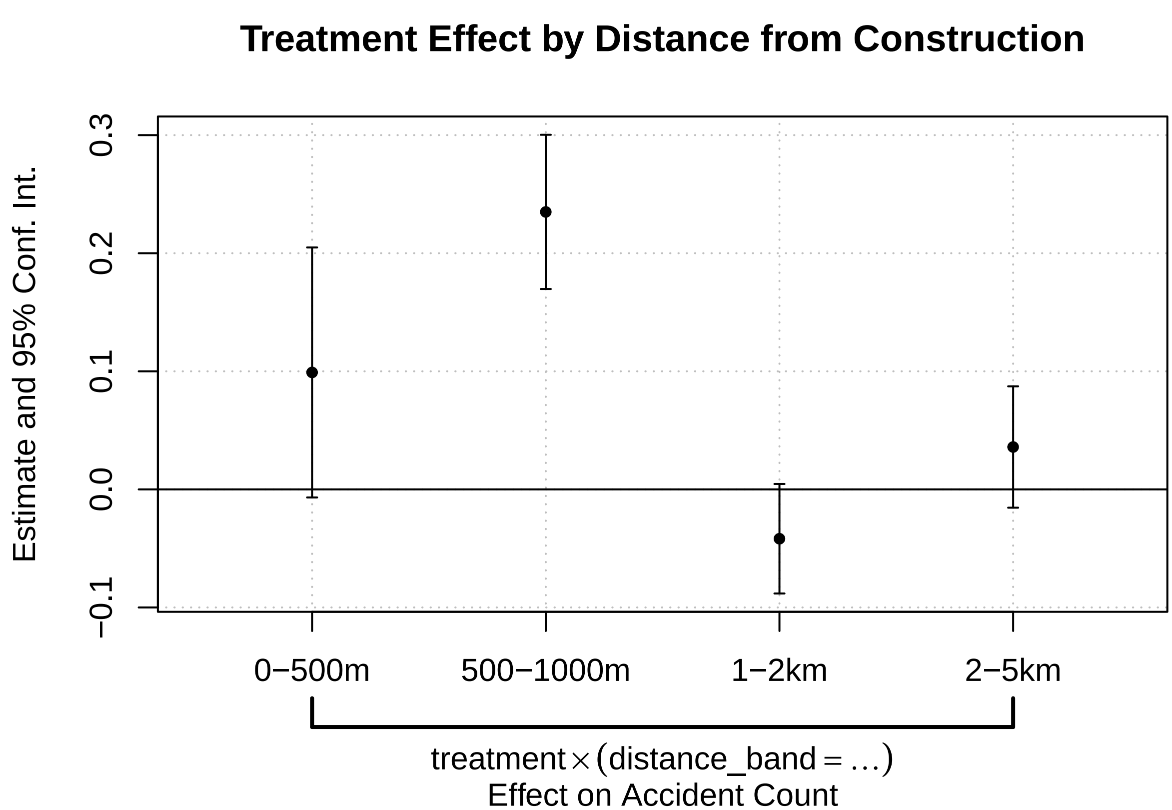
```
##
##                              model_urban_county   model_road_chars
##                                     Urban Het.         Road Chars.
## Dependent Var.:                 accident_count     accident_count
##
## treatmentTRUE                  -0.0230. (0.0134)  -0.0230. (0.0134)
## is_urbanTRUE                   35.20*** (0.6410)  35.20*** (0.6410)
## treatmentTRUE x is_urbanTRUE 0.1647*** (0.0314) 0.1647*** (0.0314)
## log(traffic_volume+1)
## Constant
## Fixed-Effects:               ------------------ ------------------
## buffer_id                                   Yes                Yes
## month                                        No                 No
## county_name-month                           Yes                Yes
## ____________________________ __________________ __________________
## S.E. type                         by: buffer_id      by: buffer_id
## Observations                            414,804            414,804
## R2                                      0.77966            0.77966
## Within R2                               0.00013            0.00013
##
##                                      model_aadt   model_spatial_se
##                                            AADT         Spatial SE
## Dependent Var.:                  accident_count     accident_count
##
## treatmentTRUE                   -0.0229. (0.0134)   -0.0240 (0.0358)
## is_urbanTRUE                 -2,370.3*** (47.37)  3.737*** (0.4433)
## treatmentTRUE x is_urbanTRUE  0.1660*** (0.0315)    0.3301 (0.3234)
## log(traffic_volume+1)         1,869.2*** (36.60)
## Constant                                         0.5260*** (0.0850)
## Fixed-Effects:               ------------------- ------------------
## buffer_id                                    Yes                 No
## month                                         No                 No
## county_name-month                            Yes                 No
## ____________________________ ___________________ __________________
## S.E. type                          by: buffer_id      Conley (50km)
## Observations                             413,712            414,852
## R2                                       0.77940            0.07056
## Within R2                                0.00014                 --
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# 6. Spatial Spillover Testing ---------------------------------------------

# Create distance bands from construction sites
panel_data <- panel_data %>%
  mutate(
    distance_band = cut(
      distance_to_nearest,
      breaks = c(0, 500, 1000, 2000, 5000, Inf),
      labels = c("0-500m", "500-1000m", "1-2km", "2-5km", ">5km")
    )
  )

# Test for spillover effects across distance bands
```

```r
spillover_model <- feols(
  accident_count ~ i(distance_band, treatment, ref = ">5km") | buffer_id + month,
  data = panel_data %>% filter(!is.na(distance_band))
)

print(summary(spillover_model))
```

```
## OLS estimation, Dep. Var.: accident_count
## Observations: 248,460
## Fixed-effects: buffer_id: 20,287,  month: 12
## Standard-errors: Clustered (buffer_id)
##                                      Estimate Std. Error  t value   Pr(>|t|)
## distance_band::0-500m:treatment      0.099022   0.054013  1.83329 6.6774e-02 .
## distance_band::500-1000m:treatment  0.234965   0.033316  7.05265 1.8119e-12 ***
## distance_band::1-2km:treatment     -0.041802   0.023652 -1.76740 7.7177e-02 .
## distance_band::2-5km:treatment      0.035887   0.026220  1.36867 1.7112e-01
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## RMSE: 1.97119     Adj. R2: 0.599397
##                 Within R2: 3.3e-4
```

```r
# Plot spillover effects
plot_spillover <- coefplot(spillover_model,
          drop = "Intercept",
          xlab = "Effect on Accident Count",
          main = "Treatment Effect by Distance from Construction")
```



```r
print(plot_spillover)
```

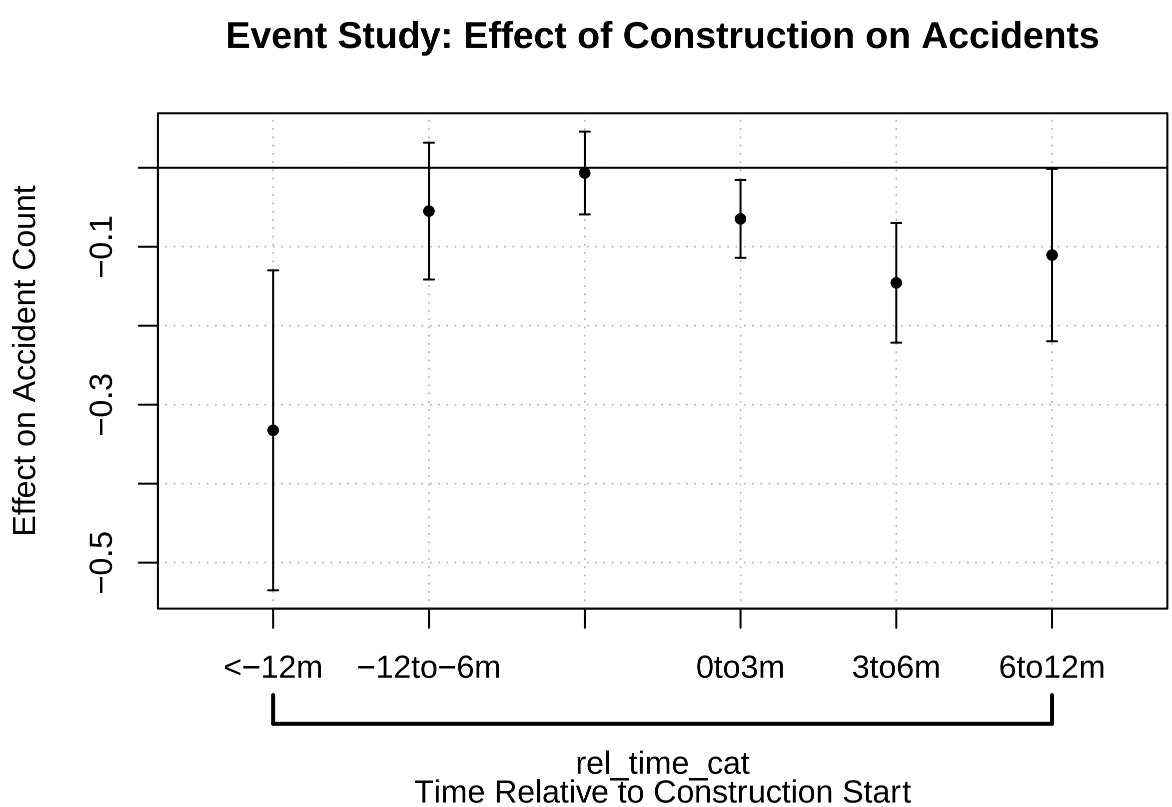
```
## $prms
##                                        estimate       ci_low     ci_high
## distance_band::0-500m:treatment     0.09902169 -0.006848226 0.204891607
## distance_band::500-1000m:treatment  0.23496513  0.169663304 0.300266959
## distance_band::1-2km:treatment     -0.04180195 -0.088161257 0.004557353
## distance_band::2-5km:treatment      0.03588654 -0.015506780 0.087279857
##                                                         estimate_names
## distance_band::0-500m:treatment       distance_band::0-500m:treatment
## distance_band::500-1000m:treatment distance_band::500-1000m:treatment
## distance_band::1-2km:treatment         distance_band::1-2km:treatment
## distance_band::2-5km:treatment         distance_band::2-5km:treatment
##                                                     estimate_names_raw id x
## distance_band::0-500m:treatment       distance_band::0-500m:treatment  1 1
## distance_band::500-1000m:treatment distance_band::500-1000m:treatment  1 2
## distance_band::1-2km:treatment         distance_band::1-2km:treatment  1 3
## distance_band::2-5km:treatment         distance_band::2-5km:treatment  1 4
##                                             y
## distance_band::0-500m:treatment     0.09902169
## distance_band::500-1000m:treatment  0.23496513
## distance_band::1-2km:treatment     -0.04180195
## distance_band::2-5km:treatment      0.03588654
##
## $is_iplot
## [1] FALSE
##
## $at
## [1] 1 2 3 4
##
## $labels
## [1] "0-500m"    "500-1000m" "1-2km"     "2-5km"
```

```r
# 7. Pre-trends and Placebo Tests ------------------------------------------

# Create relative time periods for event study
panel_data <- panel_data %>%
  mutate(
    rel_time = as.numeric(difftime(month, start_date, units = "days")) / 30,
    rel_time_cat = cut(rel_time,
                       breaks = c(-Inf, -12, -6, -3, 0, 3, 6, 12, Inf),
                       labels = c("<-12m", "-12to-6m", "-6to-3m", "-3to0m",
                                  "0to3m", "3to6m", "6to12m", ">12m"))
  )

# Event study model
event_study_model <- feols(
  accident_count ~ i(rel_time_cat, ref = "-3to0m") | buffer_id + county_name^month,
  data = panel_data %>% filter(!is.na(county_name), !is.na(rel_time_cat))
)

print(summary(event_study_model))
```

```
## OLS estimation, Dep. Var.: accident_count
```

```
## Observations: 414,804
## Fixed-effects: buffer_id: 33,509,  county_name^month: 696
## Standard-errors: Clustered (buffer_id)
##                           Estimate Std. Error   t value   Pr(>|t|)
## rel_time_cat::<-12m     -0.332420   0.103346 -3.216566 0.00129858 **
## rel_time_cat::-12to-6m -0.054854   0.044175 -1.241753 0.21433650
## rel_time_cat::-6to-3m  -0.006634   0.026747 -0.248012 0.80412670
## rel_time_cat::0to3m    -0.064698   0.025162 -2.571250 0.01013748 *
## rel_time_cat::3to6m    -0.145692   0.038645 -3.770065 0.00016349 ***
## rel_time_cat::6to12m   -0.110536   0.055667 -1.985668 0.04707828 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## RMSE: 3.08579     Adj. R2: 0.759854
##                 Within R2: 1.504e-4
```

```r
# Plot event study
plot_event_study <- coefplot(event_study_model,
                           xlab = "Time Relative to Construction Start",
                           ylab = "Effect on Accident Count",
                           main = "Event Study: Effect of Construction on Accidents")
```




```r
print(plot_event_study)
```

```
## $prms
##                           estimate      ci_low      ci_high
## rel_time_cat::<-12m     -0.332420257 -0.53498265 -0.129857866
## rel_time_cat::-12to-6m -0.054854282 -0.14143859  0.031730027
```

```
## rel_time_cat::-6to-3m  -0.006633676 -0.05905952  0.045792173
## rel_time_cat::0to3m    -0.064698487 -0.11401741 -0.015379569
## rel_time_cat::3to6m    -0.145692413 -0.22143706 -0.069947769
## rel_time_cat::6to12m   -0.110535781 -0.21964462 -0.001426945
##                                estimate_names     estimate_names_raw id x
## rel_time_cat::<-12m       rel_time_cat::<-12m    rel_time_cat::<-12m  1 1
## rel_time_cat::-12to-6m rel_time_cat::-12to-6m rel_time_cat::-12to-6m  1 2
## rel_time_cat::-6to-3m   rel_time_cat::-6to-3m  rel_time_cat::-6to-3m  1 3
## rel_time_cat::0to3m       rel_time_cat::0to3m    rel_time_cat::0to3m  1 4
## rel_time_cat::3to6m       rel_time_cat::3to6m    rel_time_cat::3to6m  1 5
## rel_time_cat::6to12m     rel_time_cat::6to12m   rel_time_cat::6to12m  1 6
##                                   y
## rel_time_cat::<-12m    -0.332420257
## rel_time_cat::-12to-6m -0.054854282
## rel_time_cat::-6to-3m  -0.006633676
## rel_time_cat::0to3m    -0.064698487
## rel_time_cat::3to6m    -0.145692413
## rel_time_cat::6to12m   -0.110535781
##
## $is_iplot
## [1] FALSE
##
## $at
## [1] 1 2 3 4 5 6
##
## $labels
## [1] "<-12m"    "-12to-6m" "-6to-3m"  "0to3m"    "3to6m"    "6to12m"
```

Fixed effects model for rigor

```r
# 8. Heterogeneity Analysis ------------------------------------------------

# Add interaction with construction type (if available)
if("construction_type" %in% names(panel_data)) {
  het_model <- feols(
    accident_count ~ i(construction_type, treatment) | buffer_id + county_name^month,
    data = panel_data %>% filter(!is.na(county_name), !is.na(construction_type))
  )

  print(summary(het_model))

  # Plot heterogeneity
  plot_het <- coefplot(het_model,
                       xlab = "Effect on Accident Count",
                       main = "Treatment Effect by Construction Type")

  print(plot_het)
}

# 9. Robustness Checks -----------------------------------------------------

# Alternative fixed effects specification
alt_fe_model <- feols(
  accident_count ~ treatment | buffer_id + county_name^month,
  data = panel_data %>% filter(!is.na(county_name))
```

```r
)

# Poisson model for count data
#poisson_model <- fepois(
#  accident_count ~ treatment | buffer_id + county_name^month,
#  data = panel_data %>% filter(!is.na(county_name))
#)

# Compare additional models
robustness_check <- etable(
  model_county_time, alt_fe_model, #poisson_model,
  headers = c("Main", "Alt FE")#, "Poisson")
  #stat = c("adj.r2", "aic", "bic", "n")
)

print(robustness_check)
```

```
##                     model_county_time       alt_fe_model
##                                  Main             Alt FE
## Dependent Var.:        accident_count     accident_count
##
## treatmentTRUE     0.0931*** (0.0226) 0.0931*** (0.0226)
## Fixed-Effects:    ------------------ ------------------
## buffer_id                        Yes                Yes
## county_name-month                Yes                Yes
## _________________ __________________ __________________
## S.E.: Clustered        by: buffer_id      by: buffer_id
## Observations                 414,804            414,804
## R2                           0.77964            0.77964
## Within R2                    7.66e-5            7.66e-5
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# 10. Export Results -------------------------------------------------------

# Save model comparison to CSV
write.csv(model_comparison, "model_comparison_results.csv", row.names = FALSE)
```