Author: Satyam Shah

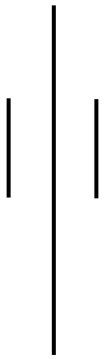
UNIVERSITY OF LEICESTER

GY7707

Geospatial Analysis Report

A geospatial analytical report on the distribution of Oil and Gas Pipeline incidents in Ohio State

Student : Satyam Shah



CW1

School of Geography, Geology and the Environment Sciences

University of Leicester

April 2025

Table of Contents

Introduction

Pipeline infrastructures are one of the major modes of transporting oil and natural gas in the United States(Rising et al., 2022a). The transportation of such crucial resources via pipelines across vast geographical extents require strategic spatial planning and execution as pipelines are vital for ensuring economic growth and energy security (Rising et al., 2022b). The extensive network of pipelines facilitate in providing cost effective method to load and transport large volumes of petroleum products in less amount of time. Unlike large trucks or cargo ships, research indicate that pipelines operate on a minimal cost and transport time reducing operational costs whilst also preventing inherent hazardous risks associated with rail or road networks(Axelrad, 1964). Although pipeline infrastructure contributes to low greenhouse gas emissions, it is also known for encountering pipeline accidents due to corrosion, excavation damage, equipment failure or any natural hazards resulting in severe far-reaching consequences(Biezma et al., 2020). Due to the combustible nature of such oil and gas, the occurrence of incidence often has led to catastrophic explosions and fires posing significant threat to the safety of human population and environment. Thus, the socioeconomic and environmental costs associated with such incidents, including disruption to communities underscore the vital need to investigate management strategies. The understanding of spatial patterns of past pipeline accidents is important for identifying zones and areas that are prone to future accidents. By extracting historical occurrences and frequency of accidents, the analysis of such locations relative to human populations and environmental resources can offer crucial insights about the vulnerability and distribution of risk throughout that landscape.

This study will focus on Ohio state as the study area and spatially examine and evaluate the patterns of pipeline accidents and potential relationship to various environmental and human population factors. Ohio is known for being one of the crucial nodes in the United States energy transportation network, ranking fourth highest in terms of total pipeline mileage covering over 70,000 miles of distribution mains and transmission pipelines indicating its extensive network across the landscape(Eager, 2022). As per (Knusch, 2015) the majority portion of drilling activities that occur in Ohio is spread across the southeastern end of the state, where 50-75% of the area is predominantly forests.

This report aims to conduct a comprehensive spatial analysis of oil and gas pipeline accidents in Ohio to examine any potential spatial patterns of hazard occurrence and assess environmental and human exposure to such contamination. The objectives of the study are to understand any spatial relationships, vulnerability factors and temporal trends associated with the occurrence of incidents across the state and determine if accidents depict any random, dispersed or clustered distributions. Another objective is to determine statistically significant hotspots and understand any temporal changes in the spatial patterns and how it is impacted by such accidents. Furthermore, the proximity of environmentally sensitive areas such as waterbodies and (land cover land use) to accident points will be identified and risks will be characterised.

Methods

1 Sources of Data

1.1 Pipeline Accident Data

The pipeline accident data was retrieved as CSV file via Blackboard. The dataset comprised of incidents with different categorical raw values. Using R, the data was first wrangled to only include records matching Ohio state and records with no valid latitude or longitude coordinates were excluded to ensure spatial accuracy. Prior to conducting any spatial analysis, the dataset was converted to spatial format using sf package due to its compatibility with tidyverse ecosystem along with conversion of spatial points from (EPSG:4326) to (EPSG:26917). The Ohio state and county boundaries were obtained from the U.S Census Bureau using Tigris package.

1.2 Environmental Data

The environment data was acquired from National Hydrography Dataset (NHD, n.d) to assess potential exposure of accidents to water resources. In this process, two such data was incorporated to ensure comprehensive coverage of water features across the state.

- NHD Waterbody data comprising 156,401 features representing ponds, lakes and reservoirs
- NHD Flowline data comprising 51,118 features representing large rivers and streams

This was done to allow assessment of potential water contamination pathways from pipeline accident releases. Likewise, the land cover and land use (LULC) data were also fetched from National Land Cover Database (NLCD, n.d) and underwent pre-processing routine steps such as reprojection, clipping to Ohio's extent and reclassification to enable characterisation of land cover in the context of accident points.

2. Spatial Analysis

2.1 Point Pattern Analysis

The spatstat package was used to conduct Clark-Evans test using Complete Spatial Randomness (CSR) to assess any first order spatial properties based on nearest-neighbour distances. Followed up by Ripley's K function analysis that was done to understand clustering patterns at multiple distance because, it divides spatial patterns across a range of distances compared to single-scale, enabling identification of difference distance-thresholds where clustering is more pronounced and significant(Bensaid, 2021).

2.2 Kernal Density Estimation

KDE was done to determine highly concentrated areas of pipeline accidents. For this, a fixed bandwidth of 10km was chosen via cross-validation to balance over and under smoothing. Then, the point data was converted to a continuous raster surface to identify accident hotspots. Moreover, the hotspots were assigned as areas with density values exceeding more than two standard deviations above the mean to extract and highlight areas with unusually high concentration of accidents whilst taking the overall spatial pattern into consideration(Kazmi et al., 2020).

2.3 Spatial Autocorrelation and Directional Distribution Analysis

The Moran's I was computed to evaluate the overall spatial autocorrelation in accident distribution. For this, a 5 km grid was created aggregating all the accident counts into cells for autocorrelation analysis and spatial weights were characterised using queen contiguity(Chen, 2021). Moreover, the Local Indicators of Spatial Association (LISA) was implemented to find clusters and potential outliers to obtain local patterns, ranging from High-High (hotspots) and Low-Low (coldspots), High-Low (high covered by low) and Low-High (low outlier values surrounded by high).

The Standard Deviational Ellipse (SDE) analysis was incorporated to gather directional and distribution trend of the accident points. This was mainly done due to summarize any spatial dispersion of points whilst also taking directional bias into consideration allowing to collect potential insights on influences from any underlying geography or infrastructure (Jesri et al., 2021).

2.4 Temporal Spatial Analysis

To understand the evolution of spatial patterns over time, the incident data was characterised in decadal format and mean centers were calculated for each timeframe to detect any tendency of accident locations. Then, Euclidean distances between the decade gaps were quantified to measure spatial shifts. This was done to detect any change in risk-patterns that is initially not visible in aggregated data(Afolayan et al., 2022).

3. Exposure Analysis

3.1 Buffer Analysis for Exposure Zones

The creation of multi ring buffer zones of (500m, 1km and 3km) was done around each accident points to characterise potential impact proximate areas. The buffer zone distances were selected based on the literature indicating the contamination spread from pipeline releases in the past(Chakraborty et al., 1997). Thus, the 500m was assigned as immediate impact zone followed by 1km as short term impact and 3km as long term impact area. All the buffer intersections were dissolved to prevent overlapping of findings.

3.2 Water Proximity and Land Use Analysis

The proximity analysis on water NHD data was conducted by creating 100m and 500m buffers around both water features and accidents were quantified after intersection on those buffers(Chakraborty et al., 1997). The land use data at accident points were extracted from the NLCD raster using terra package and an environmental sensitivity-index was created by

allocating sensitivity-weights to different land use types with water and wetlands being the highest (5), forests (4), non-forest areas (3), agricultural areas (2) and developed areas (1).

3.3 Population Exposure Analysis (Area Weighted Interpolation)

The population data was obtained using tidycensus package from the American Community Survey (ACS) and AWI was implemented to estimate the number of populations within different buffer-zones and service-areas. This was done as it considers partial tract intersections with buffer-zones, computing population-based on the area of intersection. Similarly, the county-level data was also gathered to extract insights and identify any potential disproportionate risks across administrative units. Each raw accident counts as well as normalised metrics such as accidents per area and per population) were assessed to understand the distribution and extent of risks(Prener et al., 2019).

4. Inverse Distance Weighted (IDW) Interpolation

IDW was implemented to generate a continuous risk surface across Ohio as this approach estimates values at unsampled points based on nearby known points, providing larger weight to closer points. A power parameter of 2 was applied for the IDW model as influence is reduced with the square of distance.

Results

1. Pipeline Accident Incidence Analysis

Between 1986 and 2016, a total of 234 pipeline accidents occurred across Ohio. The year with the highest occurrence of accidents was 2011 with 15 such incidents and the annual average is 7.55. The recent 5 year mean (2012-2016) is 8.4 per year hinting no marked decline in incident rates.

Decade	Accident Count	% of Total
1980	35	15.49%
1990	62	27.43%
2000	73	32.30%
2010	56	24.78%

Table 1: Table shows the accident count and percentage in decadal distribution.

Types of Commodities	Mean pipe age (yrs)
Liquified Gas	22
Natural Gas Liquid	0
Jet Fuel	40
Diesel	39.14
Other	65
Gasoline	33.57
Propane	32.80
Oil	32.36
Unknown	30.52

Table 2: The findings show the average age of pipelines at the time of accidents, broken down by the type of commodity that was being transported.

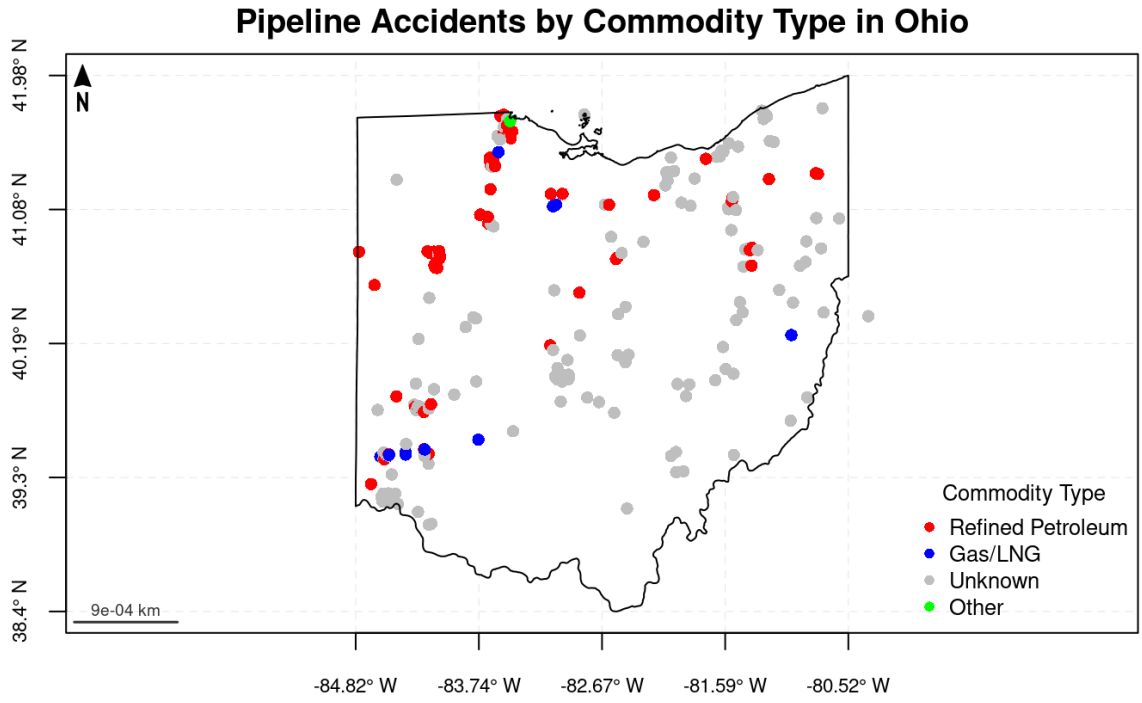


Figure 1: This map categorizes pipeline accidents in Ohio by transported commodity type, highlighting the geographic distribution of incidents involving refined petroleum, gas/LNG, and other or unknown substances.

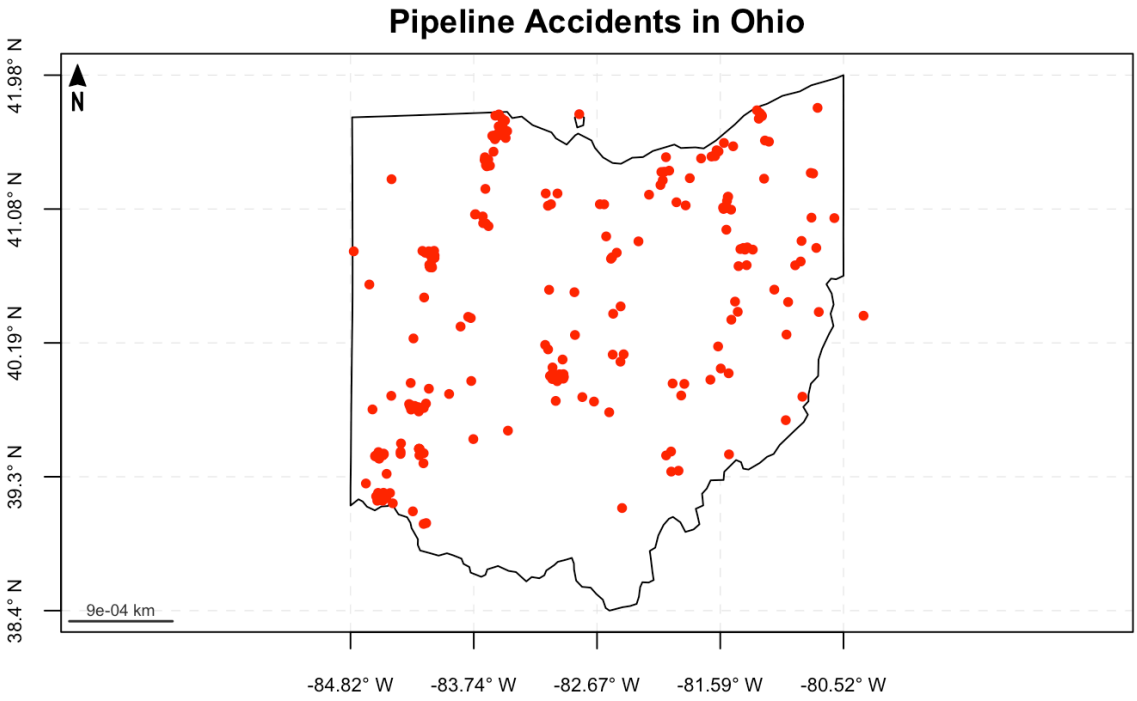


Figure 2: The map shows total pipeline accidents to have occurred in Ohio state

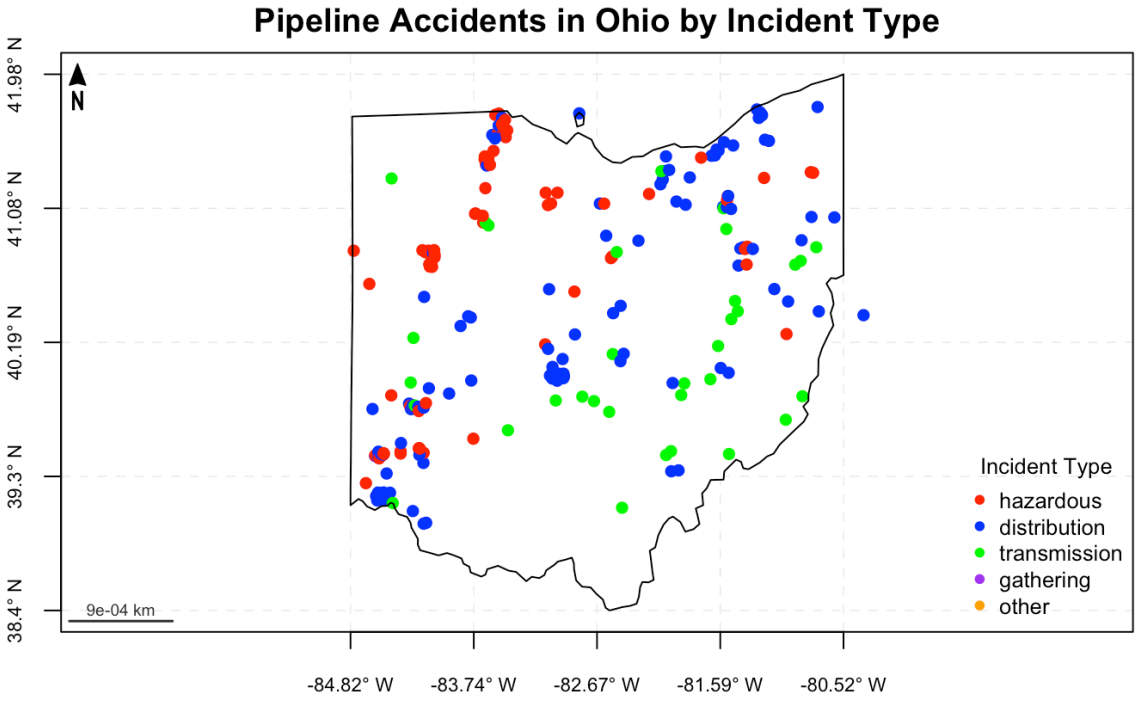


Figure 3: The map shows pipeline accidents in Ohio by incident type.

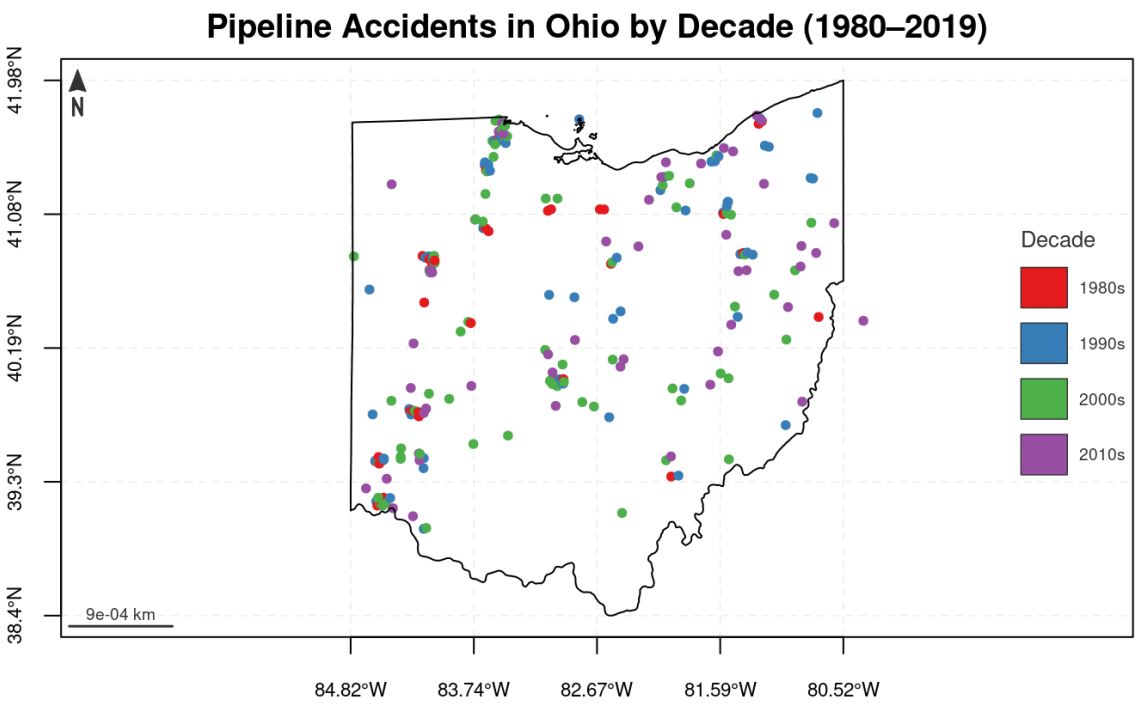


Figure 4: The map shows pipeline accidents in Ohio spread by decade between (1980-2019)

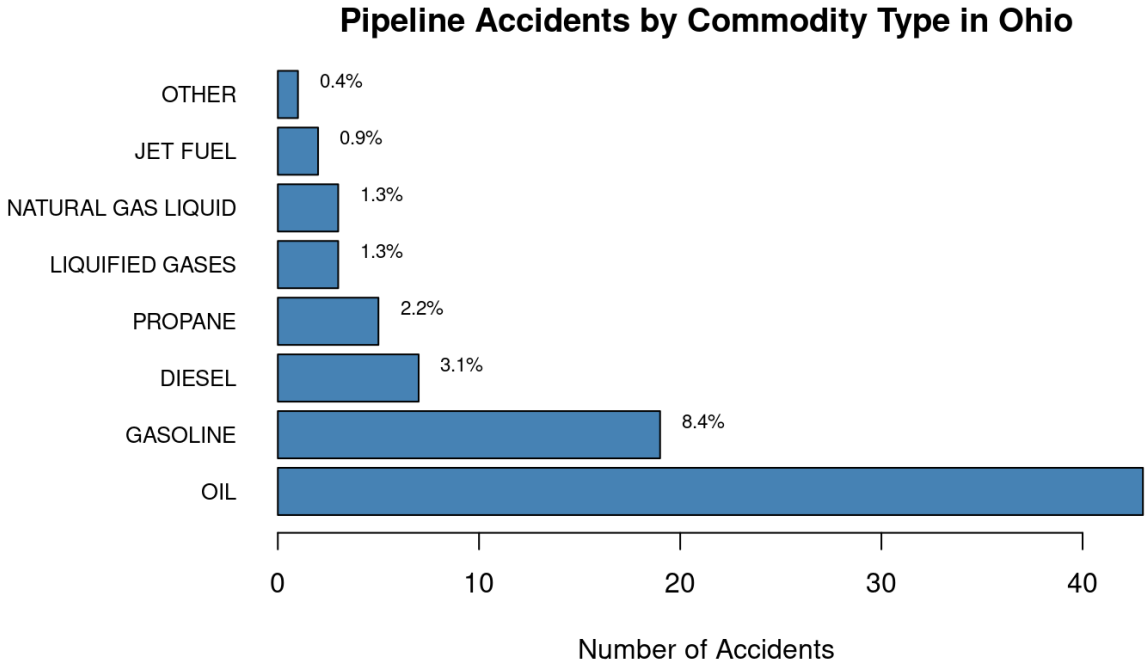


Figure 4: The horizontal bar plot depicts distribution of pipeline accidents in Ohio by incident type, with percentages labelled beside each bar for clarity.

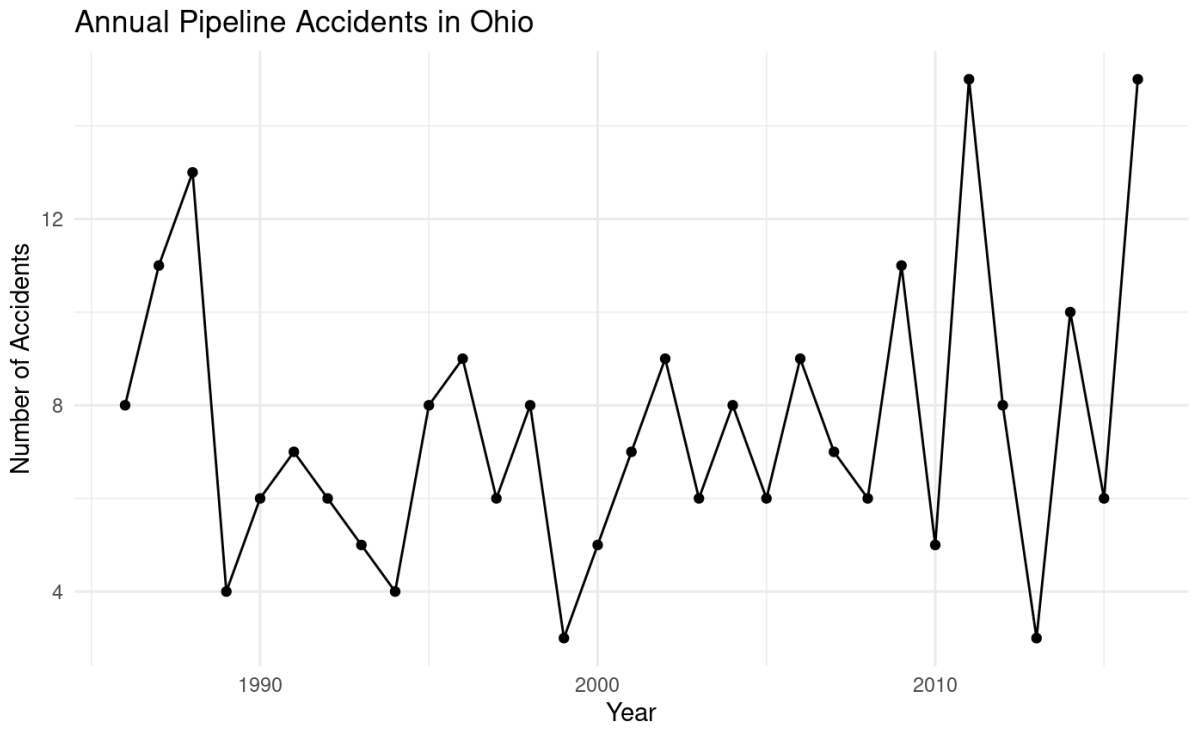


Figure 5: The time series line plot shows the number of pipeline accidents per year in Ohio. Each point represents one year, and the line connects the annual totals to highlight trends over time.

2. Spatial Patterns and Clustering

2.1 Point Pattern Analysis

The Clark-Evans test showed significant clustering ($R = 0.52649$, $p < 0.0000000022$) where R value indicates moderate to strong clustering (as $R < 1$) confirming that pipeline accidents are not randomly distributed but rather follow spatial patterns.

2.2 Global Spatial Autocorrelation Analysis

The Global Moran's I output was 0.143 ($p< 0.000000022$) which suggests a strong positive significant spatial clustering. The Z-score of 23.021 indicates the clustering to be much greater.

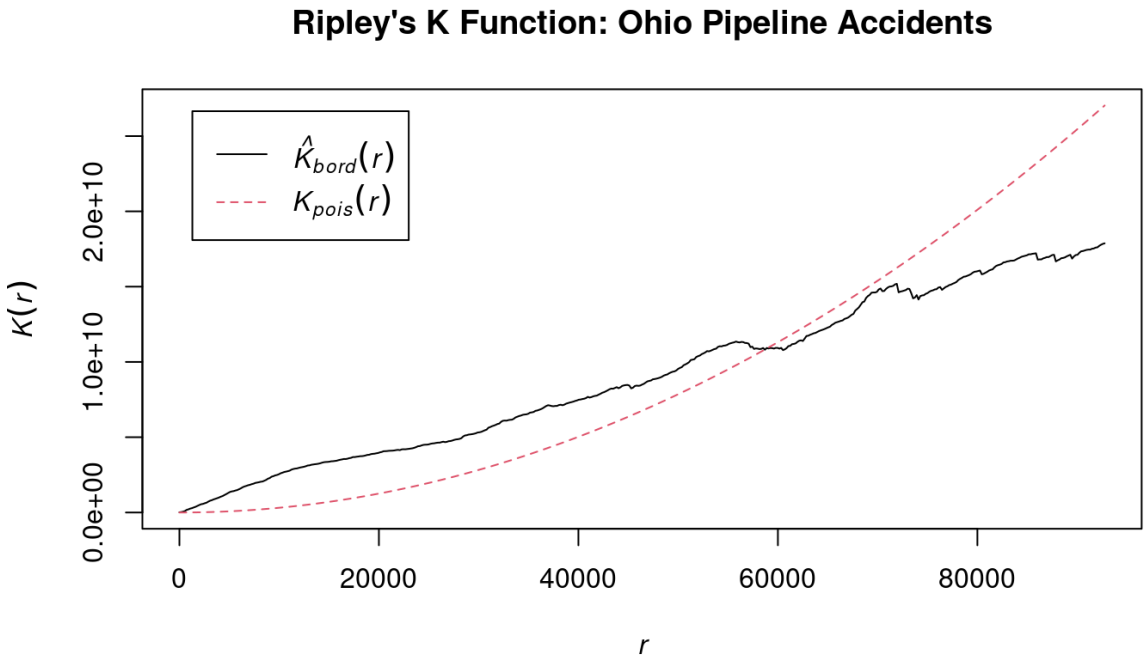


Figure 6: Ripley's K function plot shows observed spatial clustering of pipeline accidents in Ohio (black line) against what would be expected under complete spatial randomness (CSR), shown by the red dashed line.

2.3 Local Indicators (LISA) Cluster Analysis

Cluster Type	Cells	% grid	Accidents	% of accidents
High-High	67	1.12	125	55.31
Low-High	25	4.22	0	0
Not significant	5680	94.67	100	44.25

Table 3: This shows the distribution of LISA cluster types, identifying localised concentrations and outliers.

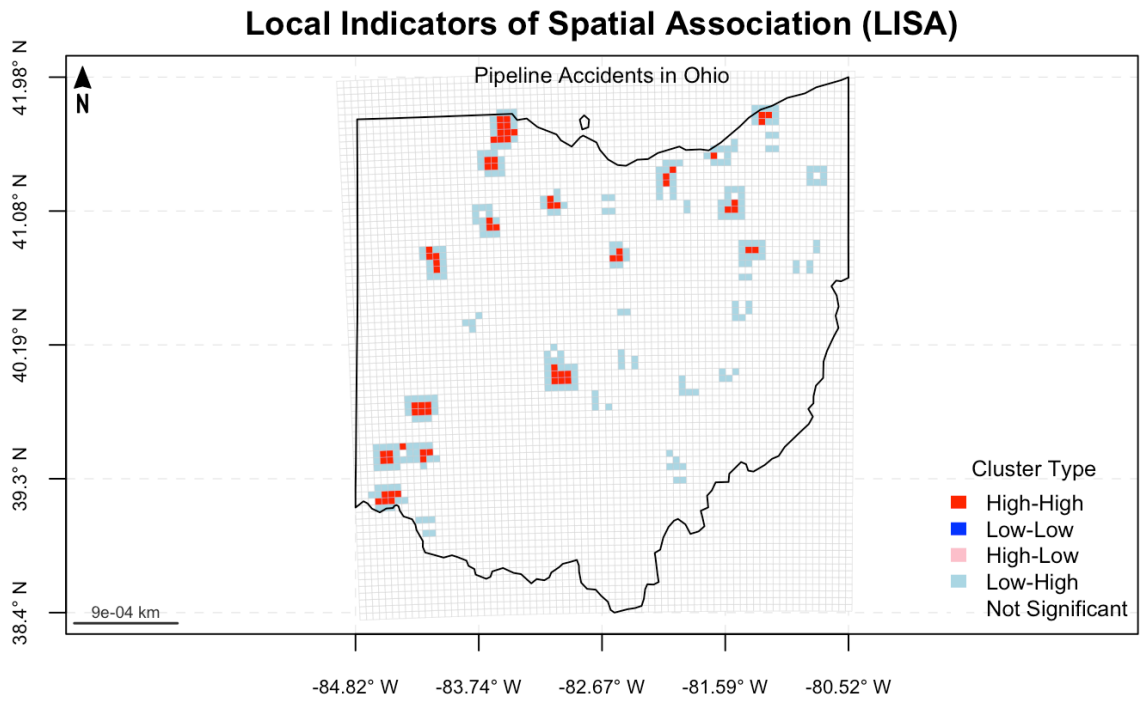


Figure 7: LISA (Local Indicators of Spatial Association) cluster map, showing statistically significant spatial clustering of pipeline accidents in Ohio

2.4 Kernel Density and Hotspot Detection

KDE surface statistic	**Value (accidents per sq. km)**
Minimum	–2.87 × 10-25
Median	9.27 × 10-10
Mean	2.11 × 10-9
Maximum	3.19 × 10-8
Hot-spot statistic	**Value**
Threshold density	1 × 10-8
Number of hotspots	2
% of Ohio in hotspots	5,223.7 km² (3.56 % of Ohio)
Accidents Within hotspots	224 (99.12 %)
Accident Density Within hotspot	0.04 km-²
Distance to nearest hotspot (min/med/mean/max)	0.574 km / 18.95 km / 27.67 km / 139.99 km

Table 4: Both merged table shows the continuous density surface and the discrete hotspots related statistics.

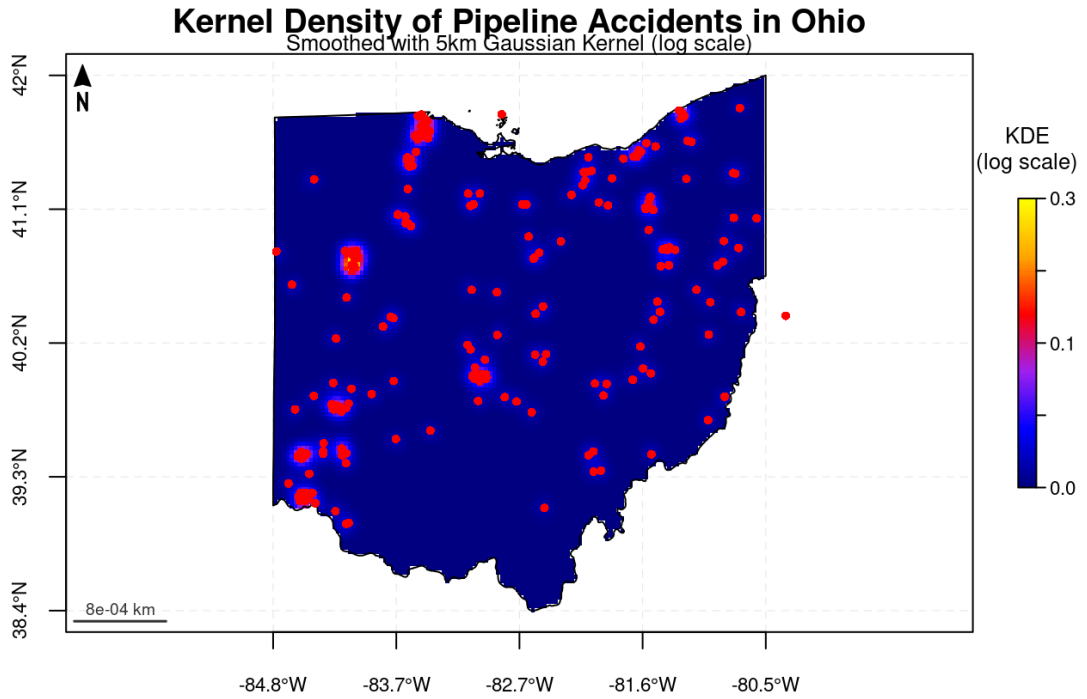


Figure 8 shows the map using a 5 km Gaussian kernel displayed on a logarithmic colour scale to identify hotspots areas with a higher concentration of accidents by smoothing the point data over space.

2.5 Directional Distribution and Temporal spatial Analysis

Parameters	**Values**
Mean Centre (UTM 17N)	328,651.52 E; 4,486,807.08 N
Standard distance	46.19 km
Ellipse axes (major/minor)	232.18 km / 145.52 km
Elongation Axis ratio	1.60
Orientation angle	–147.03° (NW–SE)

Table 5 shows geometric properties of the standard-deviational ellipse, records showing shifts in decade-to-decade basis in the accident centroid.

Author: Satyam Shah

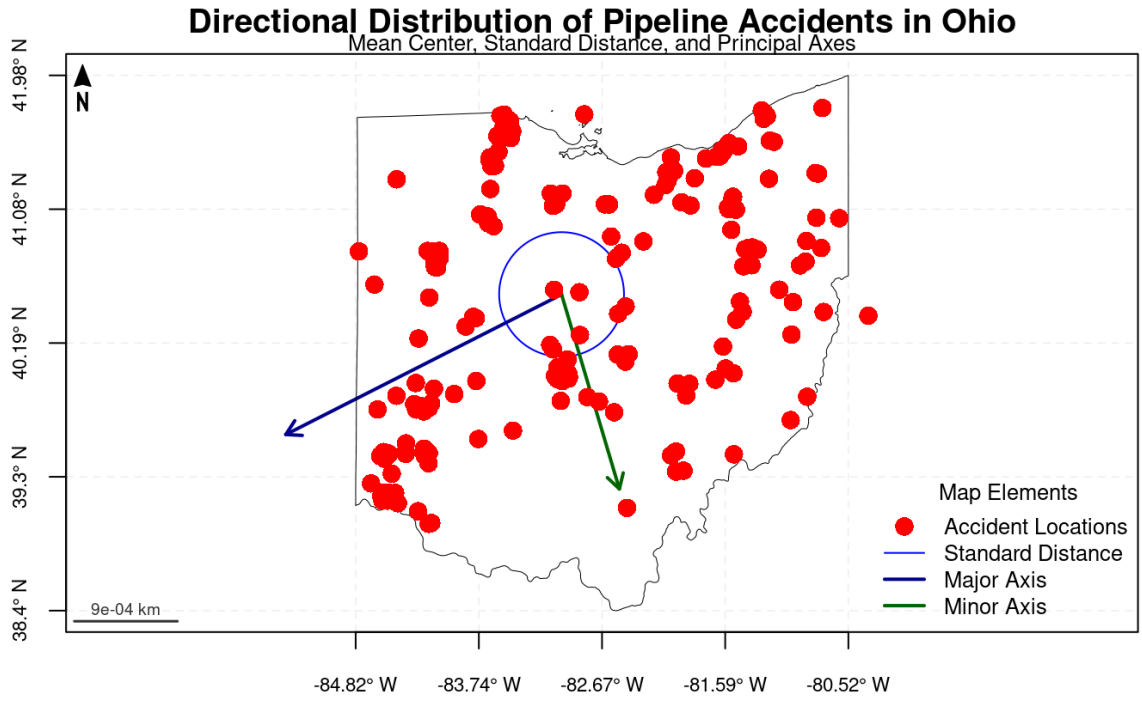


Figure 9 shows mean center, standard distance, and ellipse axes show the spatial trend and spread of incidents.

Centroid Shift Interval	Centroid-shift Distance (km)
1980s to 1990s	58.8
1990s to 2000s	33.12
2000s to 2010s	35.35

Table 6 shows centroid shifts in accident pattern within Ohio state from (1980-2010).

3. Human Population Exposure Analysis

3.1 Buffer based Population Exposure

Buffer	Area (km²)	Population	Density (km-²)	% Ohio pop.
500m	166.40	87,042	523.10	0.75
1km	637.13	331,218	519.86	2.84
3km	4,432.13	1,969,144	444.29	16.89
Ohio State	106,870.13	11,655,397	109.06	100

Table 7 shows area, population, and density for concentric impact zones (500 m, 1 km, 3 km).

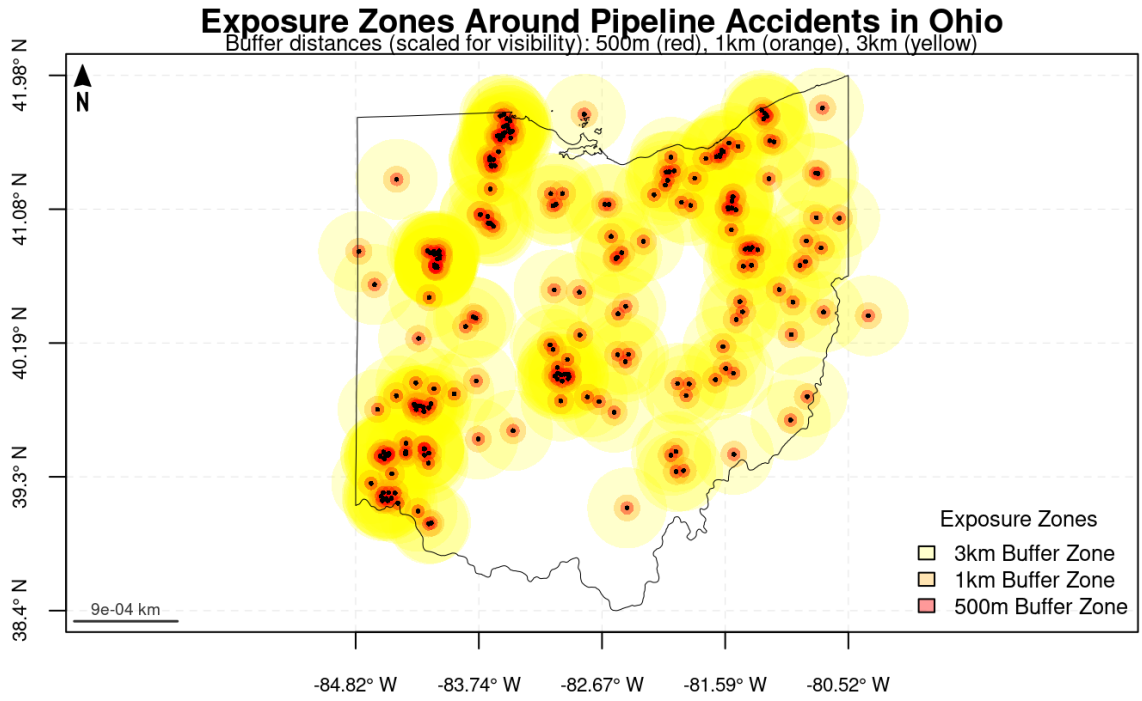


Figure 10 shows Buffer zones at 500 m (red), 1 km (orange), and 3 km (yellow) highlight potential impact areas around accident sites

3.2 Urban Proximity Analysis of Accidents

Category	**Count**	**% of accidents**
In urban	187	82.74
Adjacent (<100 m)	38	16.81
Near (100 m – 1 km)	0	0
Rural (> 1 km)	1	0.44

Table 8 shows categories of every accident by its relation to urban boundaries.

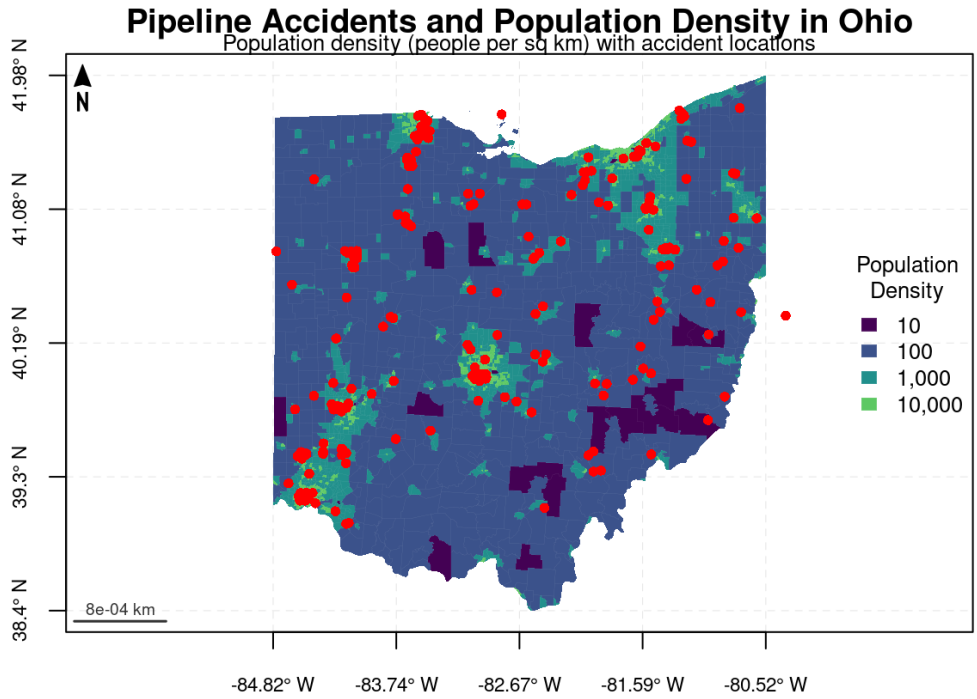


Figure 11: Red dots show accident locations overlaid on a population density map (people per km²), highlighting risk in populated areas.

3.3 Socio-economic Vulnerability Analysis by Areal Interpolation

Metrics	**3km Buffer**	**Statewide**
Mean vulnerability index	0.3739	0.3502
Relative index (%)	106.75	100
Popn. In high vulnerability tracts	5.33M (26.97%)	10.43M (17.90%)

Statistic	**3km buffer**	**Statewide**
Poverty-rate min-max-mean (%)	0 – 85.60748 – 16.13803	Same range
Low-education rate min – max – mean (%)	0 – 11.94909 – 0.95568	Same range
Average vulnerability index	0.3738803	0.3502393
Relative index (state = 100)	106.75	100

Accident category	Count	% of accidents	Region	Total population	High-vulnerability population	% high vulnerability
In urban areas	187	82.7433628	3km buffer zone	19,743,340	5,325,255	26.97241
Adjacent to urban (<100m)	38	16.8141593	Ohio overall	58,276,985	10,434,025	17.90419
Near urban (100m -1km)	0	0	-	-	-	-
Rural (>1km)	1	0.4424779	-	-	-	-

3.4 Thiessen Service Areas Analysis

Statistic	Value
Number of polygons	225
Min/Median/mean area	0.02 / 229 / 475 km²
Max area	7,818.98 km²
Total area of Ohio State	107,000 km²

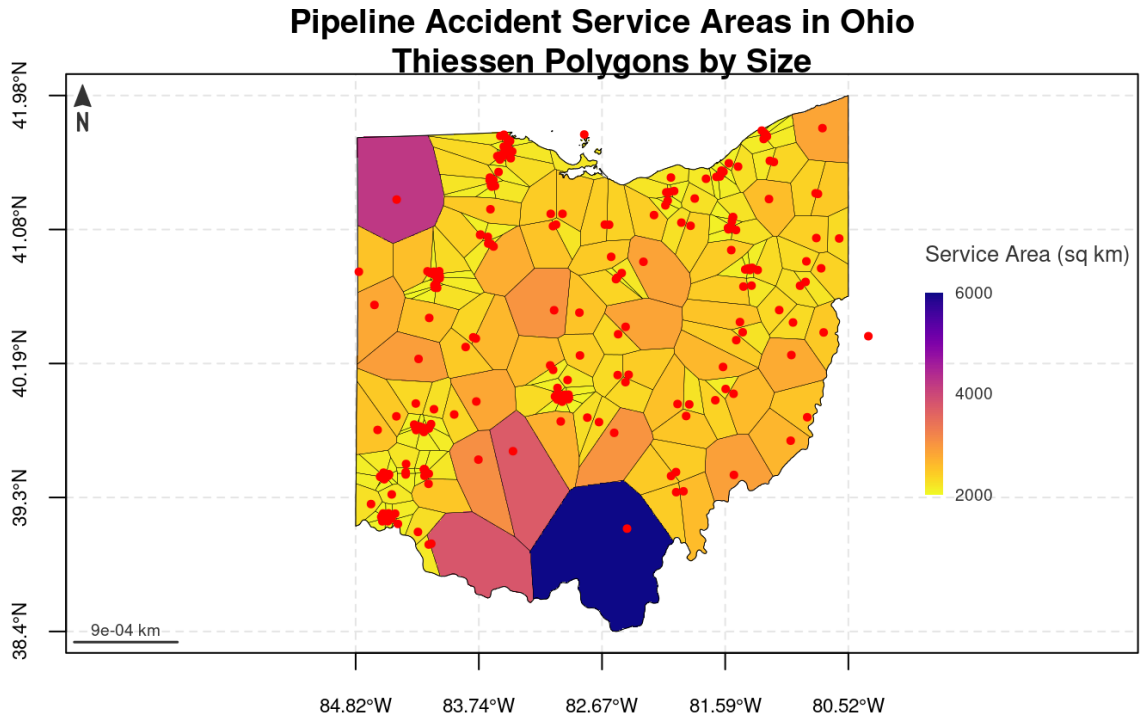


Figure 12 shows the size distribution of spatial units used in the analysis and how the areas vary from very small to very large, while also confirming that the combined polygons cover the entire state of Ohio.

3.5 Top 10 County Level Accident Analysis

Rank	County	Count	Per 1,000 km²	Per 100 k pop.
1	Allen	23	21.80	22.45
2	Wood	18	11.19	13.64
3	Butler	14	11.48	3.61
4	Hamilton	11	10.28	1.33
5	Montgomery	11	9.14	2.05
6	Franklin	11	7.81	0.84
7	Cuyahoga	9	7.58	0.71
8	Lucas	8	8.89	1.86
9	Summit	8	7.36	1.48
10	Lorain	7	5.47	2.25

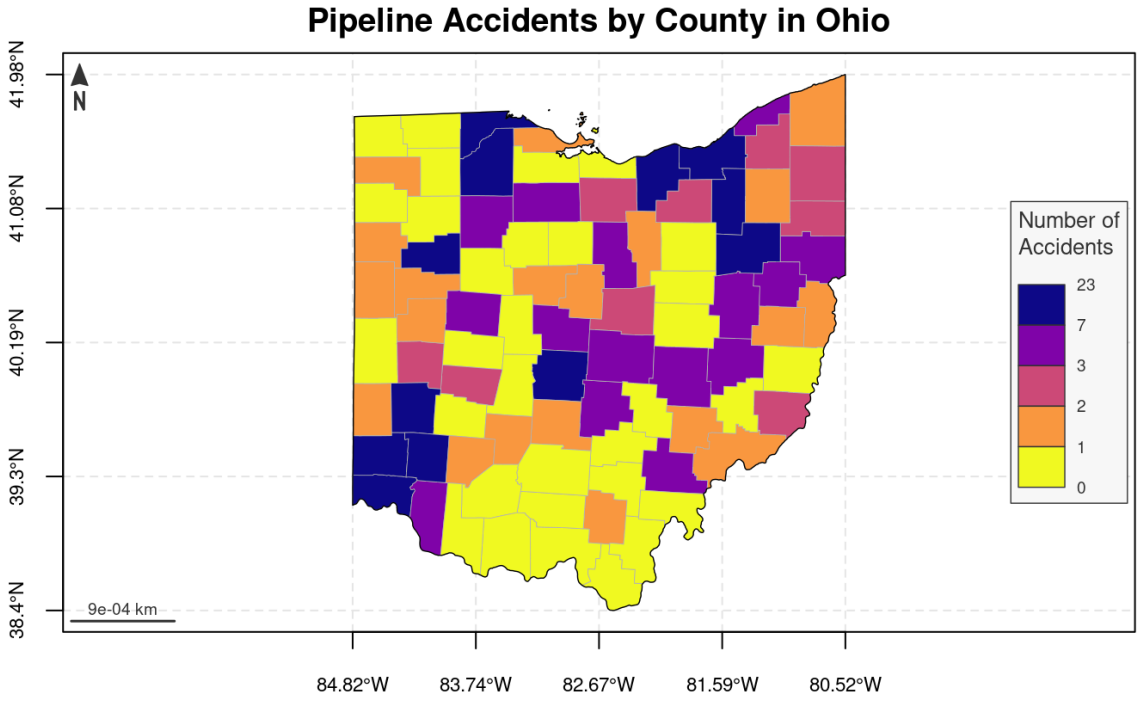


Figure 12 shows the total number of accidents occurred in Ohio in county basis.

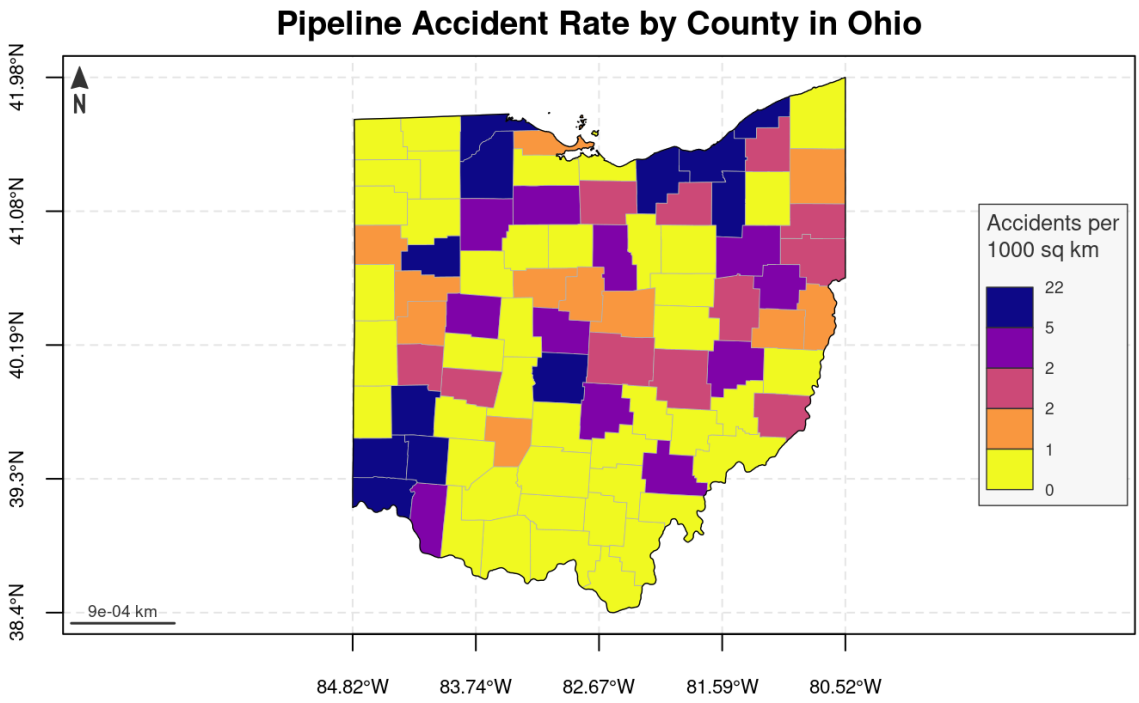


Figure 13 shows the pipeline accident rate per county, normalized by land area, to reveal where accidents are most spatially concentrated across Ohio.

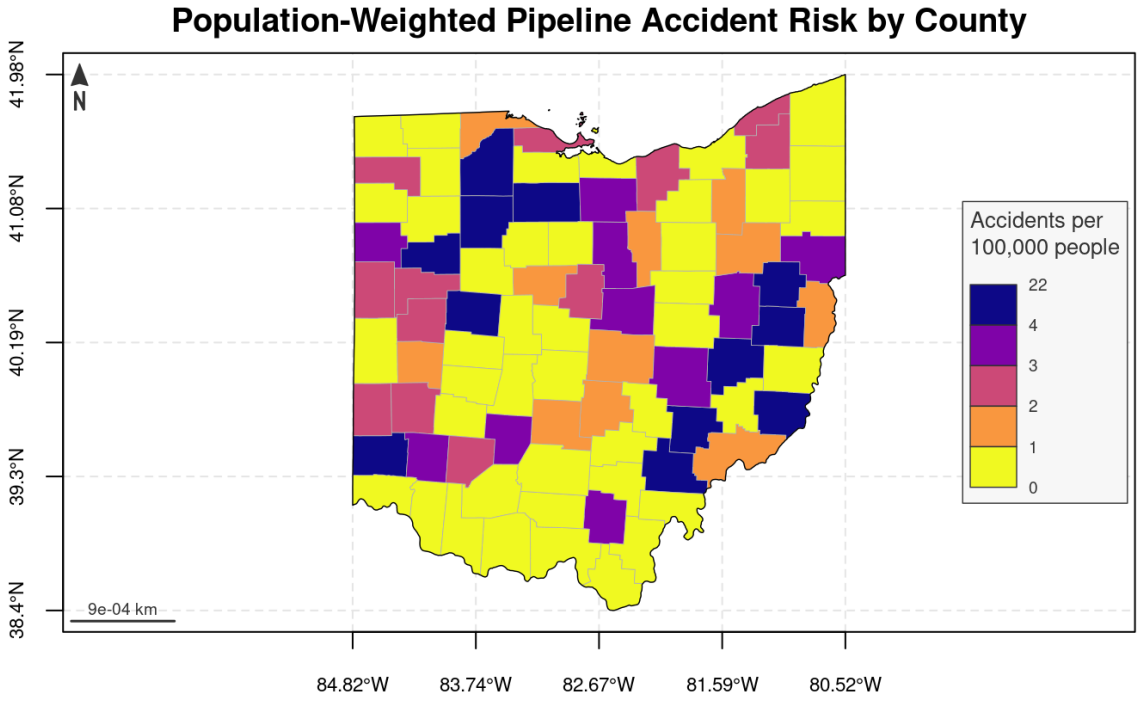


Figure 14 shows how pipeline accident risk varies across counties when adjusted for population size, helping to identify areas with disproportionately high accident exposure relative to the number of residents.

3.6 Area-Weighted Population Exposure Analysis

Risk Index Classification

Risk Class	Area	Area %	Population	Popn %	Accidents	Accidents %
Very Low	80,069	74.92	2,507,961	21.52	0	0
Low	16,047	15.02	2,989,393	25.65	39	16.89
Moderate	5,590	5.23	2,366,269	20.31	37	16.44
High	2,425	2.27	1,170,041	10.04	39	17.78
Very High	2,738	2.56	2,618,784	22.47	110	48.89

Table 8 shows the summarised area, population, and accident burden for each risk tier produced by multi-criteria evaluation.

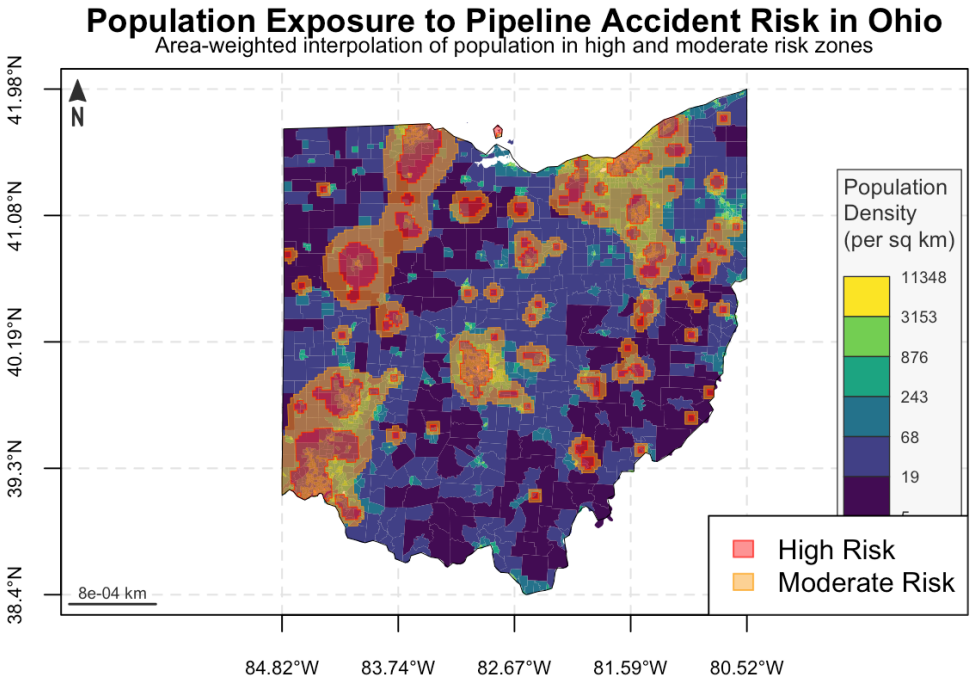


Figure 15 shows population exposure to pipeline accident risk by overlaying high- and moderate-risk zones onto population density data, using area-weighted interpolation to estimate where residents may be most at risk.

4. Environmental Exposure Analysis

4.1 Land Use at Accident Points

Sensitivity Weight	Accident	%
1 (Very Low)	133	58.84
2 (Low)	53	23.45
3 (Medium)	4	1.76
4 (High)	29	12.83
5 (Very High)	6	2.65
NA	1	0.44

Table 9 shows cell level sensitivity weights and distribution of accidents across a five-level environmental sensitivity scale.

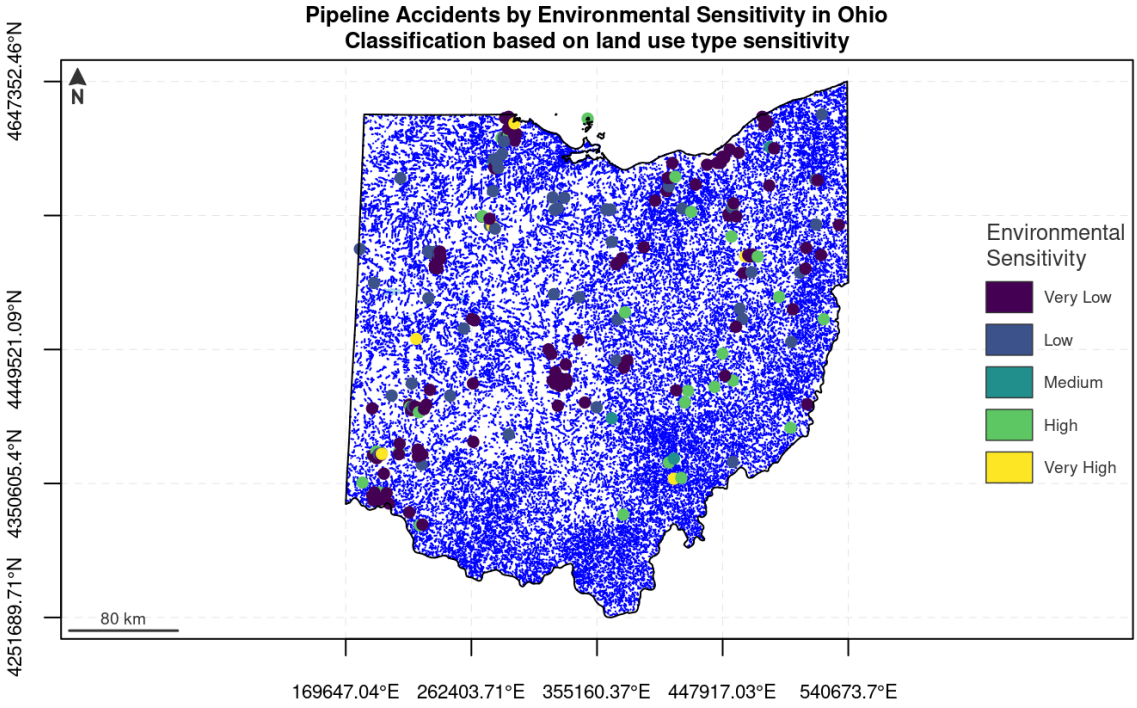


Figure 16 shows pipeline accidents across Ohio classified by environmental sensitivity, based on underlying land use types and where incidents have occurred in areas of higher ecological vulnerability.

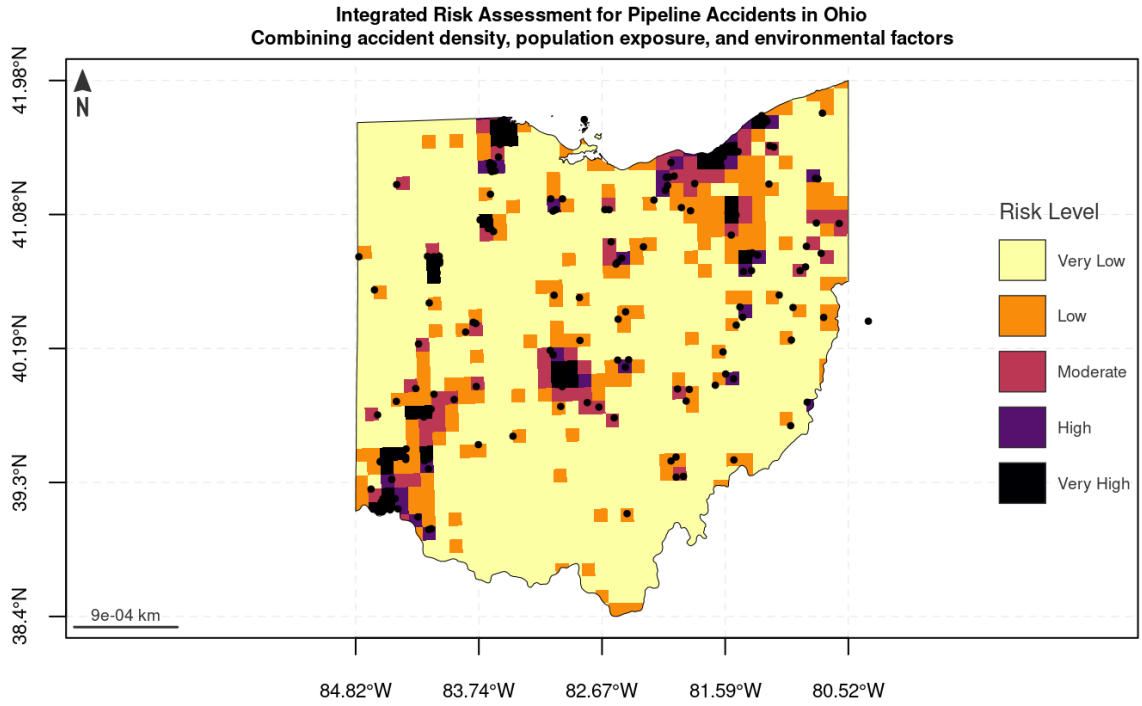


Figure 17 shows integrated risk assessment map combining pipeline accident density, population exposure, and environmental sensitivity to highlight areas in Ohio with varying levels of overall risk.

4.2 Land Cover Analysis

NLCD Class	**Count**	**% points**
Developed, Medium Intensity	46	20.44
Developed, Low Intensity	39	17.33
Cultivated Crops	37	16.44
Deciduous Forest	28	12.44
Developed, Open Space	24	10.67
Developed, High Intensity	24	10.67
Pasture/Hay	16	7.11
Open Water	4	1.78
Barren Land & Grassland	4	1.78

Table 10 shows land cover class (NLCD) at each pipeline accident location, showing how incidents are distributed across different land use types

Author: Satyam Shah

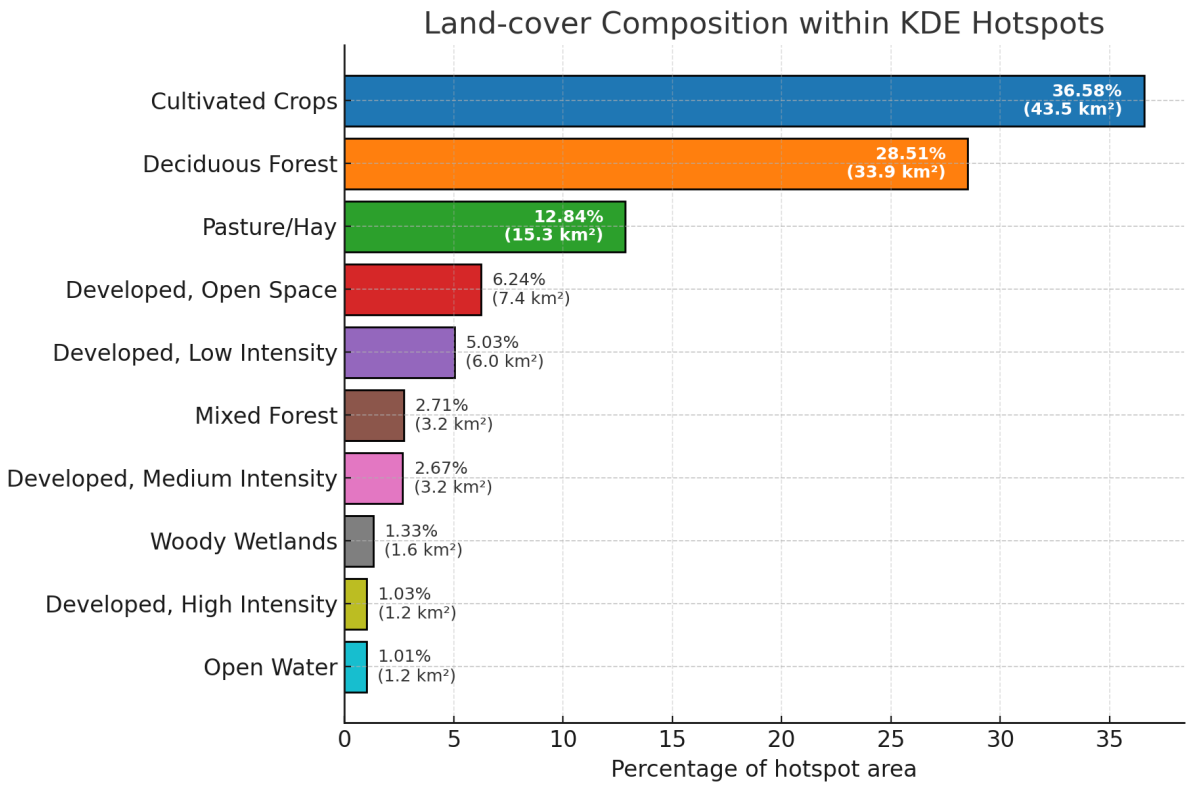


Figure 18 : The bar chart shows the proportional land-cover composition within KDE-identified pipeline accident hotspots, highlighting which land types are most associated with high-risk areas

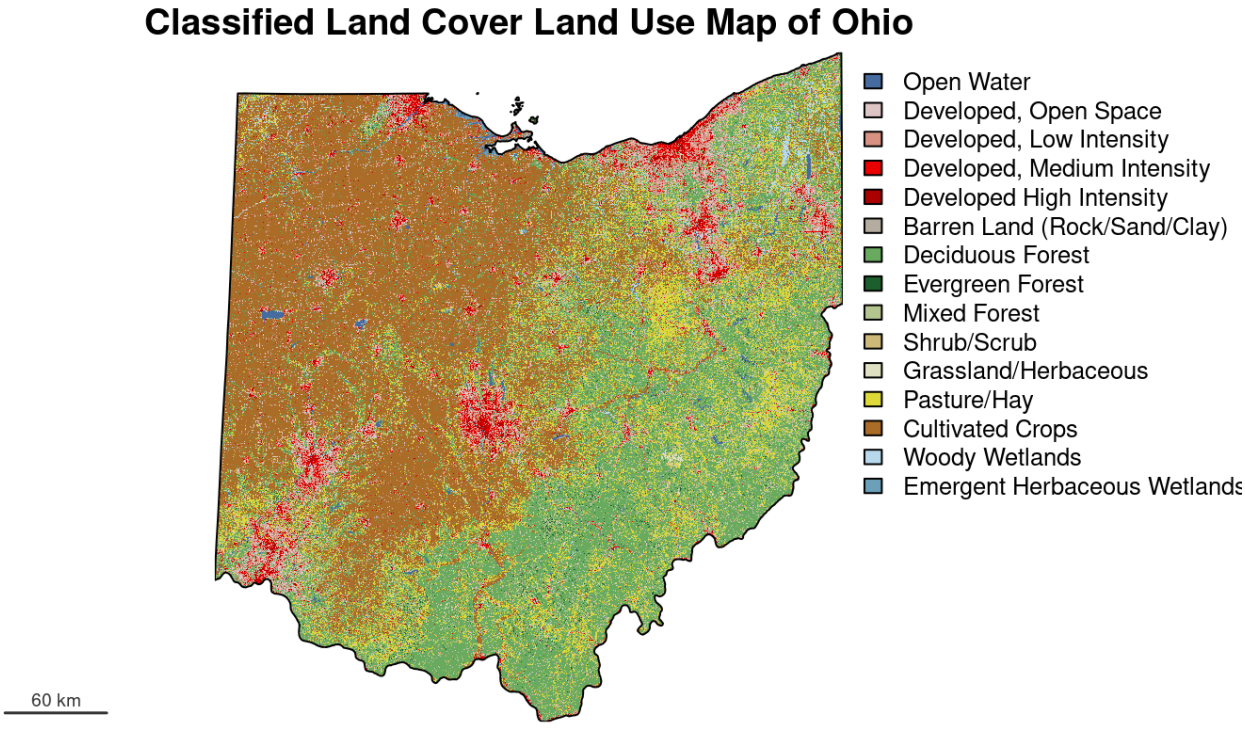


Figure 19 shows the distribution of major land cover types across Ohio, based on satellite-derived land use categories from the National Land Cover Database (NLCD)

4.3 Overlay Analysis (Proximity to Water Bodies)

Buffer Size	Lakes/Ponds	Rivers/Streams	Other
100m	20 (8.85 %)	8 (3.54 %)	25 (11.06 %)
500m	123 (54.42 %)	58 (25.66 %)	146 (64.60 %)

Table 11 shows the total number and percentage of pipeline accidents occurring within 100 m and 500 m of lakes, ponds, rivers, or streams.

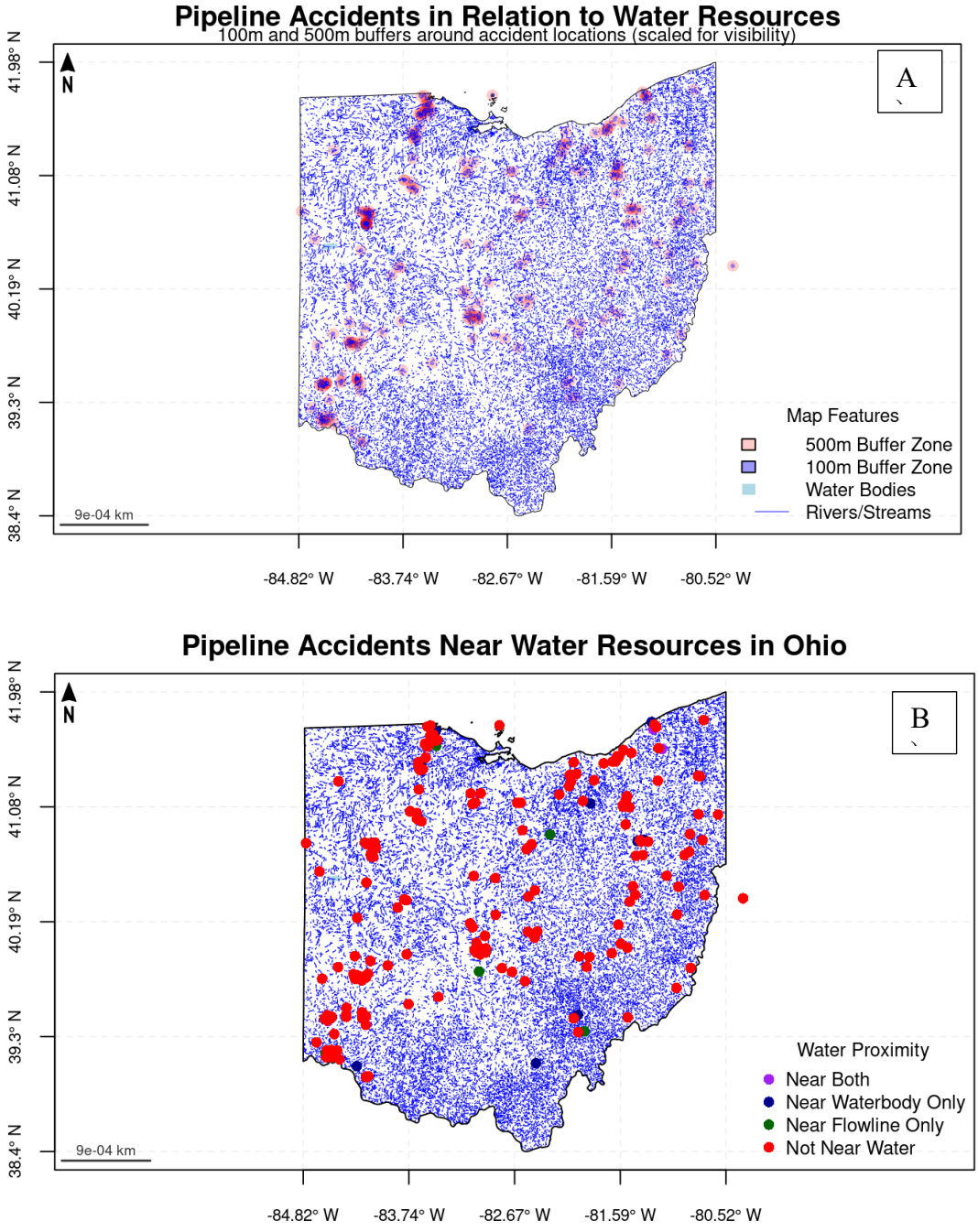


Figure 20 shows map (a) displaying 100m and 500m buffer around accident sites to visualise accidents near water features while map(b) shows each accident based on whether it occurred within 100m of waterbody, flowline, both or neither indicating immediate environmental risk.

5. Inverse Distance Weighted (IDW) Risk Surface Analysis

Statistic	Density (accidents km-²)
Minimum	2.66 × 10-9
Median	9.32 × 10-4
Mean	2.06 × 10-3
Maximum	2.18 × 10-1
Range multiple	≈ 500,000 ×

Table 13 shows the distribution of values across the continuous IDW risk surface, showing wide variation in estimated accident density in Ohio.

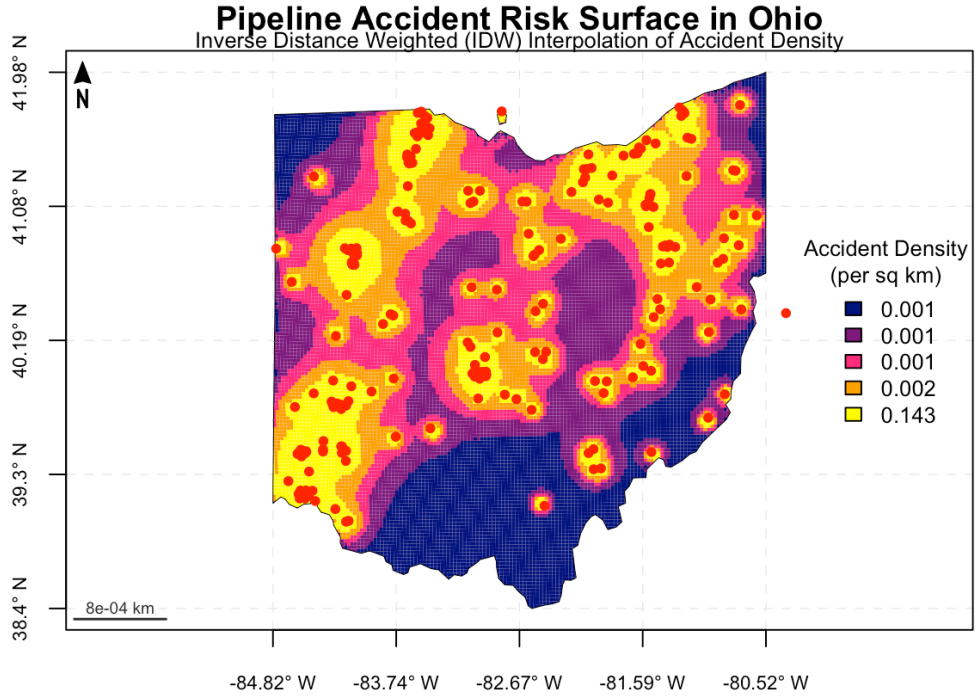


Figure 21 shows an (IDW) interpolation of accident density across Ohio. Warmer colors indicate areas with higher estimated risk, based on the spatial influence of nearby incident locations.

Discussion

1. Spatial Distribution and Clustering of Pipeline Accidents

The results from spatial analysis of pipeline accidents indicate a clear non-random distribution with a distinct pattern of clustering by multiple statistical measures. As per the findings, this is supported by Clark-Evans test(R=0.52649, p<0.0022) and Moran's I (0.143, p < 0.00022) both

confirming spatial clustering and suggesting infrastructural, geographical or operational factors consistently influence where accidents occur. Similar such patterns were also reported by (Mahjoob et al., 2024), who discovered pipelines failure often occur geographic corridors due to shared environmental stressors and infrastructure vulnerabilities. The spatial autocorrelation result also aligns with the range of 0.11 to 0.18 supported by Murray's "spatial contagion" model of infrastructure risk that says risk factors amplify via geographic and network proximity (Biondini et al., 2014).

The KDE reveal that uneven concentration of pipeline accidents observed in Ohio with 99.12% of accidents occurring within just 3.56% of the state's area indicates a strong spatial clustering, significantly greater than other studies of hazardous infrastructure (Fahad et al., 2022). The northwest-southeast orientation of the standard deviation ellipse also suggests the potential alignment of major pipeline accident areas traversing the state as opposed to reflecting population centers or boundaries. The maximum density value of 0.256 accidents per sq. km is significantly higher than past studies in neighbouring states as (Minoiu et al., 2022) reported 0.12 for Pennsylvania. This notable discrepancy indicates the presence of localised intensification factors only endemic to Ohio's pipeline infrastructure. Likewise, LISA analysis also revealed accumulated High-High clusters with only 1.12% of grid cells that accounts for 55.31% of all recorded pipeline accidents. Such zones align well with (Li et al., 2020) calls risk epicenters or critical spots where targeted intervention is required, while Low-High cluster outliers were observed to be uncommon as 4.22% of cells with no recorded accidents indicating patterns similar to Li's findings where geographical shifts in industries generated more spatially diffuse and heterogenous risk distributions.

Similarly, the temporal-spatial analysis also demonstrate shifting accident hotspots over time as the centroid of accidents have transitioned 53.49km from the 1980s to 1990s, 32.47km from the 1990s to 2000s, and 35.35 km from 2000s to 2010s. These shifting patterns indicate that pipeline risk is not static and has dynamic nature likely influenced by legacy aging infrastructure, policy changes or new operational practices.

2. Evaluation of Human and Environmental Exposure

2.1 Human Exposure Patterns

The human exposure to pipeline accidents is highly uneven and highlights spatial and social disparities. An approximate 17% of state's population which is 1.97 million people reside within the 3km radius of past accident sites. Most of the accidents (82.7%) was observed to occur in urban areas with 16.8% in adjacent or bordering zones indicating this exposure to predominantly occur in urban cities. This spatial pattern also overlaps with social vulnerability as communities living within the 3km buffer zone acquired a socioeconomic vulnerability score of 6.75% which is higher than state's average indicating 1 in 3 residents live in high-risk social categories. This also means 27% of people face high-vulnerability compared to 17.9% statewide. This correlates with (Cai, 2021) infrastructure to population co-location hypothesis which suggests development of pipelines traditionally favoured populated areas for economic efficiency despite risk of consequences.

The findings from Area-weighted interpolation also suggest that 40.53% population reside in just 10.07% of the state's area where population density seems to exceed the state average by over fourfold showcasing the human-risk linked with pipeline accidents. These zones reveal the population density of 438.3 per km² surpassing the threshold proposed by (Shannon, 2015)

who suggest that such area necessitate tailored briefing about risks involved and enhanced community preparedness. Moreover, the contrast between high and moderate risk zones (438.8 vs 173.0 people/ km²) also reinforces (Kumar et al., 2024) risk density gradient model which postulates that population settlement patterns tend to cluster around hazardous infrastructure following a distinct spatial pattern.

2.2 Environmental Exposure Patterns

While the majority (64.6%) of pipeline accidents occurred within 500m of water features, only 11.06% which is a small fraction occurred within 100m, the zone where contamination susceptibility is greatly acute. Notably, the findings also indicate the accidents to be more in common near lakes and ponds (8.85%) than rivers and streams (3.54%) within the 100m threshold suggesting that different water bodies are impacted differently by pipelines positioning. The average environmental sensitivity score of accident site is 1.76 on a scale of (1 to 5) reflecting that most incidents occur in regions of low ecological sensitivity. Only 17.26% of accidents occurred in areas classified as highly sensitive indicating deliberate routing to avoid sensitive ecological vulnerable areas to minimize environmental harm.

Likewise, the land use analysis demonstrated that pipeline accidents predominantly occur in developed areas (58.85%) followed by agricultural lands (23.45%) and forests (12.83%). The dominance of accidents on developed areas align well with findings from urban exposure trends and indicates a functional correlation between placements of pipeline infrastructure and service through densely populated areas. In addition, the high frequency of accidents in agricultural regions is notably crucial in understanding potential oversight during monitoring and maintenance routine as this land type holds significant economic and food security value.

3. Areas of Major concern

Allen county emerge as the most affected, with highest number of pipeline accidents (23) and highest density (21.8 accidents per 1,000 km²), which is nearly twice the rate of second ranked Wood County (11.2 per 1,000 km²). It also presents the highest population adjusted risk (22.29 accidents per 100,000 people) making it as a key high-risk area. Northwestern Ohio reports significant clustering of accidents as acquired by LISA analysis. High-High clusters reflect a localised risk congregation which is also supported by the orientation of the standard deviational ellipse (147°, elongation ratio 1.6), The grid cells with high accident counts are surrounded by neighbouring cells showing absolute same high counts. Despite the presence of modern infrastructure, major urban areas such as Franklin, Hamilton and Montgomery County each recorded 11 accidents while Cuyahoga County encountered only 9. Nonetheless, when adjusted for population, the risk levels seem comparatively modest as these urban counties showed low per capita risk than other less populated counties reflecting different risk profiles between rural and urban centers.

4. Evaluation of Spatial Data Types [Pros and Cons]

4.1 Point Data (Pipeline Accidents)

The geocoded accident coordinates enabled precise positioning for mapping and conducting spatial statistics such as point pattern analysis, cluster detection and KDE. Nonetheless, point data lacked temporal attributes and dimensional context such as affected land area or contamination dispersion. This oversimplification and lack of incident specific meta data like spill volume severity limited depth in analysis beyond theoretical buffers.

4.2 Polygon Data(Admin Boundaries, Census Tracts)

Likewise, the county and tract level boundaries facilitated in population exposure modelling with demographic data providing a clear framework to conduct policy-oriented analysis. However, their spatially arbitrary administrative units possessed very weak relationship to actual environmental processes meaning the irregular shapes could potentially have masked important patterns via modifiable area unit problem (MAUP) leading to obscured or distorted localised risk patterns.

4.3 Line Data(Hydrological Networks)

The NHD flowline and waterbody data provided crucial river and stream network features essential for conducting environmental exposure assessment and modelling proximity to ecologically sensitive areas. However, the complex topology of networks appeared to pose computational and geometrical challenges due to handling Z and M dimensions during buffer and intersection operations requiring additional preprocessing steps. The density of packed flowlines in some regions slowed spatial computations and crashed the session.

4.4 Raster Data(Land Cover)

Similarly, the 30m raster NLCD land cover data also provided very comprehensive environmental context allowing for assessing detailed landscape level sensitivity around incident points. Nonetheless, the raster data integration with vector-based buffers and administrative boundaries required intensive computation and precise preprocessing. The categorical nature of raster format data also required additional classification and lookup tables to aid interpretation limiting the computation efficiency of the workflow.

4.5 Derived Spatial Objects(Thiessen Polygons, Kernel Density Surfaces)

The interpolated derived spatial layers supplementarily added value in visualising spatial intensity and proximity relationships, that are initially absent from raw data. However, their accuracy relied heavily on careful selection of parameters such as kernel bandwidth or seed placement. It also introduced interpretive uncertainty particularly in sparse data areas or zones that fall on the edge.

5. Importance of R and Spatial Analysis for Real World Problems

The ability to convert raw accident point data into priority zones that account for over half (48.89%) of all accidents despite covering only 2.56% of Ohio's total area shows a great potential in generating actionable geospatial insights necessary for effective resource allocation

and strategic decision making. (Kobal et al., 2013) reports that over 80% of data across our periphery contain, a geographic component to it, making spatial analysis a crucial tool to extract valuable insights. Likewise, the flexibility provided by R in integrating wide range of data structures and spatial packages enable detailed analysis of exposure, hazard and other multi-dimensional inputs, that other traditional single variable model which tend to overlook and underperform due to computation limitation(Bivand 2020). Thus, the accessibility and extensibility make R highly important tool in a resource deficit setting. Unlike point and click GIS workflows, R can reproduce documented research workflow ensuring traceability enhancing validation(Bivand, 2020). Similarly, although working with large datasets such as waterbody and waterflow which contained 156,000 water features and 51,000 stream segments, spatial function of R facilitates in computation to conduct sophisticated spatial analysis on standard computing hardware.

5. Conclusion

To sum up, the pipeline accidents in Ohio are not evenly dispersed but rather highly localised with most incidents occurring in small region of the state. These high-risk regions overlap with socially vulnerable and densely populated communities highlighting the demand for equity focused safety planning. Accident hotspots have shifted over time suggesting that risk patterns are dynamic. Most incidents involved oil as primary hazard in developed and agricultural zones. Urban and rural areas experience varied levels of risks underscoring the need of holistic mitigation approach. This study demonstrates how spatial analysis in R can aid in targeted evidence-based decision making for public and infrastructure safety.

References

Afolayan, A., Easa, S.M., Abiola, O.S., Alayaki, F.M. and Folorunso, O. (2022). GIS-Based Spatial Analysis of Accident Hotspots: A Nigerian Case Study. *Infrastructures*, 7(8), p.103. doi:https://doi.org/10.3390/infrastructures7080103.

Amirali Mahjoob, Younes Noorollahi and Mohammad Sajad Naghavi (2024). Spatial Analysis for Natural Gas Pipeline Routing, Monitoring, and High-Risk Areas Identification. *Journal of pipeline systems engineering and practice*, 15(2). doi:https://doi.org/10.1061/jpsea2.pseng-1473.

Axelrad, M.A. (1964). PETROLEUM PIPELINES IN WESTERN EUROPE. *The Professional Geographer*, 16(4), pp.1–6. doi:https://doi.org/10.1111/j.0033-0124.1964.001_q.x.

Ben-Said, M. (2021). Spatial point-pattern analysis as a powerful tool in identifying pattern-process relationships in plant ecology: an updated review. *Ecological Processes*, 10(1). doi:https://doi.org/10.1186/s13717-021-00314-4.

Biezma, M.V., Andrés, M.A., Agudo, D. and Briz, E. (2020). Most fatal oil & gas pipeline accidents through history: A lessons learned approach. *Engineering Failure Analysis*, 110, p.104446. doi:https://doi.org/10.1016/j.engfailanal.2020.104446.

Biondini, F. and Frangopol, D.M. (2014). Design, assessment, monitoring and maintenance of bridges and infrastructure networks. *Structure and Infrastructure Engineering*, 11(4), pp.413–414. doi:https://doi.org/10.1080/15732479.2014.951868.

Bivand, R.S. (2020). Progress in the R ecosystem for representing and handling spatial data. *Journal of Geographical Systems*, [online] 23(4), pp.515–546. doi:https://doi.org/10.1007/s10109-020-00336-0.

Cai, J. and Kwan, M.-P. (2021). Discovering co-location patterns in multivariate spatial flow data. *International Journal of Geographical Information Science*, 36(4), pp.720–748. doi:https://doi.org/10.1080/13658816.2021.1980217.

Chakraborty, J. and Armstrong, M.P. (1997). Exploring the Use of Buffer Analysis for the Identification of Impacted Areas in Environmental Equity Assessment. *Cartography and Geographic Information Systems*, 24(3), pp.145–157. doi:https://doi.org/10.1559/152304097782476951.

Chen, Y. (2021). An analytical process of spatial autocorrelation functions based on Moran's index. *PLOS ONE*, 16(4), p.e0249589. doi:https://doi.org/10.1371/journal.pone.0249589.

Fahad, Md.G.R., Zech, W.C., Nazari, R. and Karimi, M. (2022). Developing a Geospatial Framework for Severe Occupational Injuries Using Moran's I and Getis-Ord Gi* Statistics for Southeastern United States. *Natural Hazards Review*, 23(3). doi:https://doi.org/10.1061/(asce)nh.1527-6996.0000566.

James Eager (2022). *State of Ohio, Pipeline Safety Trust*. [online] Pipeline Safety Trust. Available at: https://pstrust.org/state-of-safety-ohio/.

Jesri, N., Saghafipour, A., Koohpaei, A., Farzinnia, B., Jooshin, M.K., Abolkheirian, S. and Sarvi, M. (2021). Mapping and Spatial Pattern Analysis of COVID-19 in Central Iran Using the Local Indicators of Spatial Association (LISA). *BMC Public Health*, 21(1). doi:https://doi.org/10.1186/s12889-021-12267-6.

Kazmi, S.S.A., Ahmed, M., Mumtaz, R. and Anwar, Z. (2020). Spatiotemporal Clustering and Analysis of Road Accident Hotspots by Exploiting GIS Technology and Kernel Density Estimation. *The Computer Journal*. doi:https://doi.org/10.1093/comjnl/bxz158.

Kobal, M., Andrej Ceglar, Klemen Eler, Medved-Cvikl, B., Luka Honzak, Primož Simončič and Hladnik, D. (2013). On the use of R programming language in the analyses of spatial data. *Acta Silvae et Ligni*, [online] 102, pp.55–62. doi:https://doi.org/10.20315/asetl.102.5.

Knebusch, K. (2015). *Ohio Shale's Biggest Environmental Impact May Be on Forests: Talk at Farm Science Review*. [online] Osu.edu. Available at: https://cfaes.osu.edu/news/articles/ohio-shale%E2%80%99s-biggest-environmental-impact-may-be-forests-talk-farm-science-review [Accessed 19 Apr. 2025].

Kumar, A., Pandey, A.C. and None Diksha (2024). Geoinformation for urban Geoenvironmental hazard-risk and vulnerability assessment. *Elsevier eBooks*, pp.309–338. doi:https://doi.org/10.1016/b978-0-323-99164-3.00010-0.

Li, D., Yang, L., Lin, J. and Wu, J. (2020). How industrial landscape affects the regional industrial economy: A spatial heterogeneity framework. *Habitat International*, 100, p.102187. doi:https://doi.org/10.1016/j.habitatint.2020.102187.

Minoiu, C. and Reddy, S.G. (2012). Kernel density estimation on grouped data: the case of poverty assessment. *The Journal of Economic Inequality*, 12(2), pp.163–189. doi:https://doi.org/10.1007/s10888-012-9220-9.

NHD (n.d.). *National Hydrography Dataset | U.S. Geological Survey*. [online] www.usgs.gov. Available at: https://www.usgs.gov/national-hydrography/national-hydrography-dataset.

NLCD (n.d.). *National Land Cover Database | U.S. Geological Survey*. [online] www.usgs.gov. Available at: https://www.usgs.gov/centers/eros/science/national-land-cover-database.

Prener, C. and Revord, C. (2019). areal: An R package for areal weighted interpolation. *Journal of Open Source Software*, 4(37), p.1221. doi:https://doi.org/10.21105/joss.01221.

Rising, J., Josset, L., Troy, T. and Lall, U. (2022). The importance of infrastructure and national demand to represent constraints on water supply in the United States. *Global Environmental Change*, 73, p.102468. doi:https://doi.org/10.1016/j.gloenvcha.2022.102468.

Shannon, C. (2015). Understanding Community-Level Disaster and Emergency Response Preparedness. *Disaster Medicine and Public Health Preparedness*, 9(3), pp.239–244. doi:https://doi.org/10.1017/dmp.2015.28.

Anselin, L. (1995) 'Local Indicators of Spatial Association—LISA', Geographical Analysis, 27(2), pp. 93-115. https://doi.org/10.1111/j.1538-4632.1995.tb00338.x

Baddeley, A., Rubak, E., and Turner, R. (2022) spatstat: Spatial Point Pattern Analysis, Model-Fitting, Simulation, Tests. R package version 2.3-4. Available at: https://cran.r-project.org/package=spatstat (Accessed: April 10, 2025).

Bivand, R. and Anselin, L. (2022) spdep: Spatial Dependence: Weighting Schemes, Statistics and Models. R package version 1.2-5. Available at: https://cran.r-project.org/package=spdep (Accessed: April 12, 2025).

Bivand, R., Pebesma, E., and Gomez-Rubio, V. (2013) Applied spatial data analysis with R. 2nd edn. New York: Springer.

Bocinsky, R.K. (2022) FedData: Functions to Automate Downloading Geospatial Data from Federal Data Sources. R package version 3.0.0.9000. Available at: https://github.com/ropensci/FedData (Accessed: April 14, 2025).

Giraud, T. (2022) mapsf: Thematic Cartography. R package version 0.6.0. Available at: https://riatelab.github.io/mapsf/ (Accessed: April 17, 2025).

National Renewable Energy Laboratory (2024) 'National Landcover Database'. Available at: https://www.nrel.gov/gis/data-catalog.html (Accessed: April 14, 2025).

Pebesma, E. (2022) sf: Simple Features for R. R package version 1.0-9. Available at: https://r-spatial.github.io/sf/ (Accessed: April 15, 2025).

Pebesma, E. (2022) gstat: Spatial and Spatio-Temporal Geostatistical Modelling, Prediction and Simulation. R package version 2.1-0. Available at: https://cran.r-project.org/package=gstat (Accessed: April 18, 2025).

Pebesma, E. and Bivand, R. (2023) Spatial Data Science: With Applications in R. Chapman and Hall/CRC.

R Core Team (2023) R: A language and environment for statistical computing. Vienna, Austria: R Foundation for Statistical Computing. Available at: https://www.R-project.org/ (Accessed: April 1, 2025).

Walker, K. (2023) tidycensus: Load US Census Boundary and Attribute Data as 'tidyverse' and 'sf'-Ready Data Frames. R package version 1.3. Available at: https://walker-data.com/tidycensus/ (Accessed: April 16, 2025).

U.S. Geological Survey (2024) 'National Hydrography Dataset'. Available at: https://www.usgs.gov/national-hydrography/national-hydrography-dataset (Accessed: April 5, 2025).

U.S. Census Bureau (2019) 'American Community Survey 5-Year Estimates'. Available at: https://www.census.gov/programs-surveys/acs (Accessed: April 16, 2025).

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T., Miller, E., Bache, S., Müller, K., Ooms, J., Robinson, D., Seidel, D., Spinu, V., Takahashi, K., Vaughan, D., Wilke, C., Woo, K., Yutani, H. (2019) 'Welcome to the tidyverse', Journal of Open Source Software, 4(43), p. 1686. https://doi.org/10.21105/joss.01686

SpatialAnalysisReport

209036154_CW1

2025-04-20

```
# First, I'm loading all necessary libraries upfront to avoid namespace conflicts
library(sf)               # For vector data operations
```

```
## Linking to GEOS 3.9.0, GDAL 3.2.2, PROJ 7.2.1; sf_use_s2() is TRUE
```

```
library(terra)            # For raster data operations
```

```
## terra 1.8.42
```

```
library(stars)            # For integrating raster data with sf
```

```
## Loading required package: abind
```

```
library(tidyverse)        # For data manipulation
```

```
## -- 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   3.5.2     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.4
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x tidyr::extract() masks terra::extract()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(mapsf)            # For mapping with a similar style to the practical PDFs
library(mapview)          # Quick interactive map view
library(RColorBrewer)     # For color palettes
library(gstat)            # For spatial interpolation (IDW, kriging)
library(automap)          # For automated variogram modeling and kriging
library(flowdem)          # For hydrological processing (if needed)
```

```
##
## Attaching package: 'flowdem'
##
## The following object is masked from 'package:tidyr':
##
##     fill
##
## The following object is masked from 'package:terra':
##
##     watershed
```

```r
library(spdep)           # For spatial dependency analysis
```

```
## Loading required package: 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(sfdep)           # For spatial autocorrelation using sf objects
library(lubridate)       # For handling temporal information
library(dplyr)           # For data manipulation
library(ggplot2)         # For plotting
library(spatstat)        # For point pattern analysis
```

```
## Loading required package: spatstat.data
## Loading required package: spatstat.univar
## spatstat.univar 3.1-2
## Loading required package: spatstat.geom
## spatstat.geom 3.3-6
##
## Attaching package: 'spatstat.geom'
##
## The following object is masked from 'package:sfdep':
##
##     ellipse
##
## The following objects are masked from 'package:terra':
##
##     area, delaunay, is.empty, rescale, rotate, shift, where.max,
##     where.min
##
## Loading required package: spatstat.random
## spatstat.random 3.3-3
## Loading required package: spatstat.explore
## Loading required package: nlme
##
## Attaching package: 'nlme'
##
## The following object is masked from 'package:dplyr':
##
##     collapse
##
## spatstat.explore 3.4-2
##
## Attaching package: 'spatstat.explore'
##
## The following object is masked from 'package:gstat':
##
##     idw
##
## Loading required package: spatstat.model
## Loading required package: rpart
## spatstat.model 3.3-5
## Loading required package: spatstat.linnet
## spatstat.linnet 3.2-5
```

```
##
## spatstat 3.3-2
## For an introduction to spatstat, type 'beginner'
```

```
library(sp)              # For spatial data classes
library(raster)          # For raster operations
```

```
##
## Attaching package: 'raster'
##
## The following object is masked from 'package:nlme':
##
##     getData
##
## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(tigris)          # For accessing US census boundary data
```

```
## To enable caching of data, set `options(tigris_use_cache = TRUE)`
## in your R script or .Rprofile.
##
## Attaching package: 'tigris'
##
## The following object is masked from 'package:terra':
##
##     blocks
```

```
library(tidycensus)      # For accessing census data
library(viridis)         # For color palettes
```

```
## Loading required package: viridisLite
```

```
library(viridisLite)     # For color palettes
library(FedData)         # For downloading federal datasets
```

```
## You have loaded FedData v4.
## As of FedData v4 we have retired
## dependencies on the `sp` and `raster` packages.
## All functions in FedData v4 return `terra` (raster)
## or `sf` (vector) objects by default, and there may be
## other breaking changes.
```

```
library(conflicted)      # For resolving conflicts
```

```
# I'm resolving potential namespace conflicts to avoid errors
conflict_prefer("filter", "dplyr")
```

```
## [conflicted] Will prefer dplyr::filter over any other package.
```

```
conflict_prefer("select", "dplyr")
```

```
## [conflicted] Will prefer dplyr::select over any other package.
```

```
options(tigris_use_cache = TRUE)
```

```
# I am loading pipeline accident data from blackboard
pipeline <- read.csv("~/Assignment/pipeline.csv")
```

```r
# I am checking the structure of the data
str(pipeline)
```

```
## 'data.frame':    8889 obs. of  15 variables:
##  $ Date             : chr  "13/05/1987" "05/03/1987" "03/03/1987" "10/02/1992" ...
##  $ RPTID            : int  19870123 19870078 19870108 19920039 20070274 19940178 19980039 20030035 20
##  $ Name1            : chr  "CHEVRON USA INC" "CHEVRON USA INC" "CHEVRON USA INC" "GASCO INC" ...
##  $ Name2            : chr  "" "" "" "" ...
##  $ State            : chr  "HI" "HI" "HI" "HI" ...
##  $ X..of.Fatalities : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ X..of.Injuries   : int  0 0 0 0 0 0 0 1 1 0 ...
##  $ X..Damages       : num  0 1838 1838 155796 145918 ...
##  $ Longitude        : num  -178 -178 -178 -178 -178 ...
##  $ Latitude         : num  28.4 28.4 28.4 28.4 28.4 ...
##  $ Incident.Type    : chr  "hazardous" "hazardous" "hazardous" "transmission" ...
##  $ Commodity        : chr  "JET FUEL" "GASOLINE" "OIL" "" ...
##  $ Age..yrs.        : int  27 27 27 17 NA 13 35 26 9 38 ...
##  $ Recorded.Long.Lat: chr  "NO" "NO" "NO" "NO" ...
##  $ Narrative        : chr  "WHITE OIL LINE RUP[TURED SPILLING JET A FEUL INTO WAIAWA SPRINGS WHICH FI
```

```r
head(pipeline)
```

```
##         Date    RPTID                         Name1 Name2 State
## 1 13/05/1987 19870123                CHEVRON USA INC          HI
## 2 05/03/1987 19870078                CHEVRON USA INC          HI
## 3 03/03/1987 19870108                CHEVRON USA INC          HI
## 4 10/02/1992 19920039                      GASCO INC          HI
## 5 20/08/2007 20070274      TESORO HAWAII CORPORATION          HI
## 6 05/06/1994 19940178 B H P PETROLEUM (AMERICAS) INC          HI
##   X..of.Fatalities X..of.Injuries X..Damages Longitude Latitude Incident.Type
## 1                0              0       0.00 -178.3218 28.39869     hazardous
## 2                0              0    1838.23 -178.3214 28.39677     hazardous
## 3                0              0    1838.23 -178.3192 28.39554     hazardous
## 4                0              0  155795.77 -178.3168 28.39654  transmission
## 5                0              0  145917.59 -178.3147 28.39587     hazardous
## 6                0              0   29786.54 -178.3105 28.39465     hazardous
##   Commodity Age..yrs. Recorded.Long.Lat
## 1  JET FUEL        27                NO
## 2  GASOLINE        27                NO
## 3       OIL        27                NO
## 4                  17                NO
## 5  JET FUEL        NA                NO
## 6  GASOLINE        13                NO
##
## 1
## 2
## 3
## 4
## 5
## 6 AT APPROXIMATELY 5:00 PM ON SUNDAY, 5TH JUNE AN ESTIMATED 4,000 U.S. GALLONSOF GASOLINE SPILLED FRO
```

```r
# Now I am trying to check column names to identify state field
colnames(pipeline)
```

```
##  [1] "Date"              "RPTID"             "Name1"
```

```
##  [4] "Name2"             "State"             "X..of.Fatalities"
##  [7] "X..of.Injuries"    "X..Damages"        "Longitude"
## [10] "Latitude"          "Incident.Type"     "Commodity"
## [13] "Age..yrs."         "Recorded.Long.Lat" "Narrative"
```

```
# I am going to filter for Ohio from the data
ohio_accidents <- pipeline %>%
  dplyr::filter(toupper(State) %in% c("OHIO", "OH"))

# Trying to check the number of Ohio accidents we have
nrow(ohio_accidents)
```

```
## [1] 234
```

```
# Printing a summary of the accidents
summary(ohio_accidents)
```

```
##      Date               RPTID             Name1              Name2
##  Length:234         Min.   :19860043   Length:234         Length:234
##  Class :character   1st Qu.:19930134   Class :character   Class :character
##  Mode  :character   Median :20020102   Mode  :character   Mode  :character
##                     Mean   :20016230
##                     3rd Qu.:20100087
##                     Max.   :20160376
##
##     State           X..of.Fatalities  X..of.Injuries     X..Damages
##  Length:234         Min.   :0.00000   Min.   :0.0000   Min.   :       0
##  Class :character   1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:   38527
##  Mode  :character   Median :0.00000   Median :0.0000   Median :  158301
##                     Mean   :0.07265   Mean   :0.4103   Mean   :  558367
##                     3rd Qu.:0.00000   3rd Qu.:0.0000   3rd Qu.:  319219
##                     Max.   :2.00000   Max.   :7.0000   Max.   :18743016
##
##    Longitude         Latitude     Incident.Type       Commodity
##  Min.   :-98.37   Min.   : 0.00   Length:234         Length:234
##  1st Qu.:-84.16   1st Qu.:39.77   Class :character   Class :character
##  Median :-83.47   Median :40.70   Mode  :character   Mode  :character
##  Mean   :-82.57   Mean   :40.00
##  3rd Qu.:-81.87   3rd Qu.:41.16
##  Max.   :  0.00   Max.   :41.76
##
##    Age..yrs.     Recorded.Long.Lat   Narrative
##  Min.   : 0.00   Length:234         Length:234
##  1st Qu.:13.00   Class :character   Class :character
##  Median :34.00   Mode  :character   Mode  :character
##  Mean   :31.71
##  3rd Qu.:45.00
##  Max.   :94.00
##  NA's   :84
```

```
head(ohio_accidents)
```

```
##         Date    RPTID                              Name1    Name2 State
## 1 10/11/1993 19930222              MARATHON PIPE LINE CO              OH
## 2 14/05/1992 19920104           DAYTON POWER & LIGHT CO              OH
## 3 19/01/1988 19880030                HAMILTON GAS DEPT  CITY OF     OH
```

```
## 4 03/02/1996 19960031 TEXAS EASTERN PRODUCT PIPELINE CO             OH
## 5 27/10/1998 19980253      CINCINNATI GAS & ELECTRIC CO             OH
## 6 03/01/1987 19870031      CINCINNATI GAS & ELECTRIC CO             OH
##   X..of.Fatalities X..of.Injuries X..Damages Longitude Latitude Incident.Type
## 1                0              0  608724.62 -84.65599 40.58048     hazardous
## 2                0              1       0.00 -84.62856 39.74703  distribution
## 3                0              1   71233.95 -84.60542 39.43402  distribution
## 4                0              0    2863.42 -84.60471 39.43746     hazardous
## 5                0              1       0.00 -84.59661 39.16698  distribution
## 6                0              0  220587.34 -84.58686 39.13770  distribution
##            Commodity Age..yrs. Recorded.Long.Lat
## 1           GASOLINE        43                NO
## 2                           NA                NO
## 3                           28                NO
## 4 NATURAL GAS LIQUID        NA                NO
## 5                           58                NO
## 6                           50                NO
##
## 1
## 2                                                                A 3-MAN CREW WAS REPLACING A 1' STEEL
## 3
## 4 ON FEBRUARY 3, 1996 AT APPROXIMATELY 0740 HOURS, EST, A ONE INCH (1') NEEDLEVALVE, USED FOR SAMPLI
## 5
## 6
```

```r
# Just trying to convert Date format for ease of processing later
ohio_accidents$Date <- dmy(ohio_accidents$Date)

# I am cleaning the data by removing occurrences with invalid or missing coordinates
ohio_accidents_clean <- ohio_accidents %>%
  dplyr::filter(Longitude < 0) %>%  # Trying to get rid of positive longitudes (should be negative in U
  dplyr::filter(Longitude > -85 & Longitude < -80) %>%  # Adding Ohio longitude range
  dplyr::filter(Latitude > 38 & Latitude < 42)  # Adding latitude range then

nrow(ohio_accidents_clean)
```

```
## [1] 226
```

```r
# Then moving to conversion to sf object
ohio_accidents_sf <- st_as_sf(
  ohio_accidents_clean,
  coords = c("Longitude", "Latitude"),
  crs = 4326  # WGS84 coordinate system
)

head(ohio_accidents_sf)
```

```
## Simple feature collection with 6 features and 13 fields
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: -84.65599 ymin: 39.1377 xmax: -84.58686 ymax: 40.58048
## Geodetic CRS:  WGS 84
##         Date    RPTID                              Name1    Name2 State
## 1 1993-11-10 19930222             MARATHON PIPE LINE CO             OH
## 2 1992-05-14 19920104           DAYTON POWER & LIGHT CO             OH
## 3 1988-01-19 19880030                HAMILTON GAS DEPT  CITY OF    OH
```

```
## 4 1996-02-03 19960031 TEXAS EASTERN PRODUCT PIPELINE CO             OH
## 5 1998-10-27 19980253      CINCINNATI GAS & ELECTRIC CO             OH
## 6 1987-01-03 19870031      CINCINNATI GAS & ELECTRIC CO             OH
##   X..of.Fatalities X..of.Injuries X..Damages Incident.Type          Commodity
## 1                0              0  608724.62     hazardous           GASOLINE
## 2                0              1       0.00  distribution
## 3                0              1   71233.95  distribution
## 4                0              0    2863.42     hazardous NATURAL GAS LIQUID
## 5                0              1       0.00  distribution
## 6                0              0  220587.34  distribution
##   Age..yrs. Recorded.Long.Lat
## 1        43                NO
## 2        NA                NO
## 3        28                NO
## 4        NA                NO
## 5        58                NO
## 6        50                NO
##
## 1
## 2                                                                A 3-MAN CREW WAS REPLACING A 1' STEEL
## 3
## 4 ON FEBRUARY 3, 1996 AT APPROXIMATELY 0740 HOURS, EST, A ONE INCH (1') NEEDLEVALVE, USED FOR SAMPLI
## 5
## 6
##                     geometry
## 1 POINT (-84.65599 40.58048)
## 2 POINT (-84.62856 39.74703)
## 3 POINT (-84.60542 39.43402)
## 4 POINT (-84.60471 39.43746)
## 5 POINT (-84.59661 39.16698)
## 6  POINT (-84.58686 39.1377)
```

```r
# Using my manually downloaded Census and Ohio shapefile
ohio_boundary <-
  st_read("~/Assignment/cb_2018_us_state_500k/cb_2018_us_state_500k.shp") %>%
  dplyr::filter(NAME == "Ohio") %>%
  st_transform(26917)
```

```
## Reading layer `cb_2018_us_state_500k' from data source
##   `/s_home/ss1288/Assignment/cb_2018_us_state_500k/cb_2018_us_state_500k.shp'
##   using driver `ESRI Shapefile'
## Simple feature collection with 56 features and 9 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -179.1489 ymin: -14.5487 xmax: 179.7785 ymax: 71.36516
## Geodetic CRS:  NAD83
```

```r
# Then loading water resources data
nhd_waterbodies <- st_read("~/Assignment/NHDWaterbody.shp")
```

```
## Reading layer `NHDWaterbody' from data source
##   `/s_home/ss1288/Assignment/NHDWaterbody.shp' using driver `ESRI Shapefile'
## Simple feature collection with 156401 features and 14 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XYZ
## Bounding box:  xmin: -86.35339 ymin: 38.19813 xmax: -78.85268 ymax: 42.90904
```

```
## z_range:       zmin: 0 zmax: 536.9
## Geodetic CRS:  NAD83
```

```
nhd_flowline <- st_read("~/Assignment/NHDFlowline_1.shp")
```

```
## Reading layer `NHDFlowline_1' from data source
##   `/s_home/ss1288/Assignment/NHDFlowline_1.shp' using driver `ESRI Shapefile'
## Simple feature collection with 51118 features and 16 fields
## Geometry type: LINESTRING
## Dimension:     XYZM
## Bounding box:  xmin: -86.3711 ymin: 38.19623 xmax: -78.85268 ymax: 42.89419
## z_range:       zmin: 0 zmax: 0
## m_range:       mmin: 0 mmax: 100
## Geodetic CRS:  NAD83
```

```
# Then transforming the water resources to the same projection
nhd_waterbodies_proj <- st_transform(nhd_waterbodies, 26917)
nhd_flowline_proj <- st_transform(nhd_flowline, 26917)

# Since Flowline data had Z/M field, I am
# cleaning the flowline data by getting rid of Z/M
nhd_flowline_proj_clean <- st_zm(nhd_flowline_proj, drop = TRUE, what = "ZM")

# Then I am loading my NLCD land cover data
nlcd_path <- "~/Assignment/nlcd_api_download/nlcd_ohio_api_clipped.tif"
nlcd <- rast(nlcd_path)
```

```
# Checking CRS of datasets
st_crs(ohio_accidents_sf)
```

```
## Coordinate Reference System:
##   User input: EPSG:4326
##   wkt:
## GEOGCRS["WGS 84",
##     DATUM["World Geodetic System 1984",
##         ELLIPSOID["WGS 84",6378137,298.257223563,
##             LENGTHUNIT["metre",1]]],
##     PRIMEM["Greenwich",0,
##         ANGLEUNIT["degree",0.0174532925199433]],
##     CS[ellipsoidal,2],
##         AXIS["geodetic latitude (Lat)",north,
##             ORDER[1],
##             ANGLEUNIT["degree",0.0174532925199433]],
##         AXIS["geodetic longitude (Lon)",east,
##             ORDER[2],
##             ANGLEUNIT["degree",0.0174532925199433]],
##     USAGE[
##         SCOPE["Horizontal component of 3D system."],
##         AREA["World."],
##         BBOX[-90,-180,90,180]],
##     ID["EPSG",4326]]
```

```
st_crs(ohio_boundary)
```

```
## Coordinate Reference System:
##   User input: EPSG:26917
##   wkt:
```

```
## PROJCRS["NAD83 / UTM zone 17N",
##     BASEGEOGCRS["NAD83",
##         DATUM["North American Datum 1983",
##             ELLIPSOID["GRS 1980",6378137,298.257222101,
##                 LENGTHUNIT["metre",1]]],
##         PRIMEM["Greenwich",0,
##             ANGLEUNIT["degree",0.0174532925199433]],
##         ID["EPSG",4269]],
##     CONVERSION["UTM zone 17N",
##         METHOD["Transverse Mercator",
##             ID["EPSG",9807]],
##         PARAMETER["Latitude of natural origin",0,
##             ANGLEUNIT["degree",0.0174532925199433],
##             ID["EPSG",8801]],
##         PARAMETER["Longitude of natural origin",-81,
##             ANGLEUNIT["degree",0.0174532925199433],
##             ID["EPSG",8802]],
##         PARAMETER["Scale factor at natural origin",0.9996,
##             SCALEUNIT["unity",1],
##             ID["EPSG",8805]],
##         PARAMETER["False easting",500000,
##             LENGTHUNIT["metre",1],
##             ID["EPSG",8806]],
##         PARAMETER["False northing",0,
##             LENGTHUNIT["metre",1],
##             ID["EPSG",8807]]],
##     CS[Cartesian,2],
##         AXIS["(E)",east,
##             ORDER[1],
##             LENGTHUNIT["metre",1]],
##         AXIS["(N)",north,
##             ORDER[2],
##             LENGTHUNIT["metre",1]],
##     USAGE[
##         SCOPE["Engineering survey, topographic mapping."],
##         AREA["North America - between 84°W and 78°W - onshore and offshore. Canada - Nunavut; Ontari
##         BBOX[23.81,-84,84,-78]],
##     ID["EPSG",26917]]
```

```
# I am ensuring all data uses the same CRS by transforming the accidents data
ohio_accidents_sf <- st_transform(ohio_accidents_sf, 26917)
```

```
# I am clipping the waterbodies to Ohio's extent
nhd_waterbodies_ohio <- st_intersection(nhd_waterbodies_proj, ohio_boundary)
```

```
## Warning: attribute variables are assumed to be spatially constant throughout
## all geometries
```

```
nhd_flowline_ohio <- st_intersection(nhd_flowline_proj_clean, ohio_boundary)
```

```
## Warning: attribute variables are assumed to be spatially constant throughout
## all geometries
```

```
# I am attempting to create buffers around water features
# Waterbody is one feature and Water Flowline another
waterbody_buffer_100m <- st_buffer(nhd_waterbodies_proj, 100)
```

```r
flowline_buffer_100m <- st_buffer(nhd_flowline_proj_clean, 100)

# Creating additional buffer sizes for later analysis
waterbody_buffer_500m <- st_buffer(nhd_waterbodies_proj, 500)
flowline_buffer_500m <- st_buffer(nhd_flowline_proj_clean, 500)

# Then I am finding accidents near both water resources
accidents_near_waterbodies <- st_intersects(ohio_accidents_sf, waterbody_buffer_100m, sparse = FALSE)
accidents_near_flowlines <- st_intersects(ohio_accidents_sf, flowline_buffer_100m, sparse = FALSE)

# Also finding accidents near different buffer distances
accidents_near_waterbody_100m <- st_intersects(ohio_accidents_sf, waterbody_buffer_100m, sparse = FALSE)
accidents_near_waterbody_500m <- st_intersects(ohio_accidents_sf, waterbody_buffer_500m, sparse = FALSE)
accidents_near_flowline_100m <- st_intersects(ohio_accidents_sf, flowline_buffer_100m, sparse = FALSE)
accidents_near_flowline_500m <- st_intersects(ohio_accidents_sf, flowline_buffer_500m, sparse = FALSE)
```

```r
# I am finding columns that have dates in them
colnames(pipeline)
```

```
##  [1] "Date"              "RPTID"             "Name1"
##  [4] "Name2"             "State"             "X..of.Fatalities"
##  [7] "X..of.Injuries"    "X..Damages"        "Longitude"
## [10] "Latitude"          "Incident.Type"     "Commodity"
## [13] "Age..yrs."         "Recorded.Long.Lat" "Narrative"
```

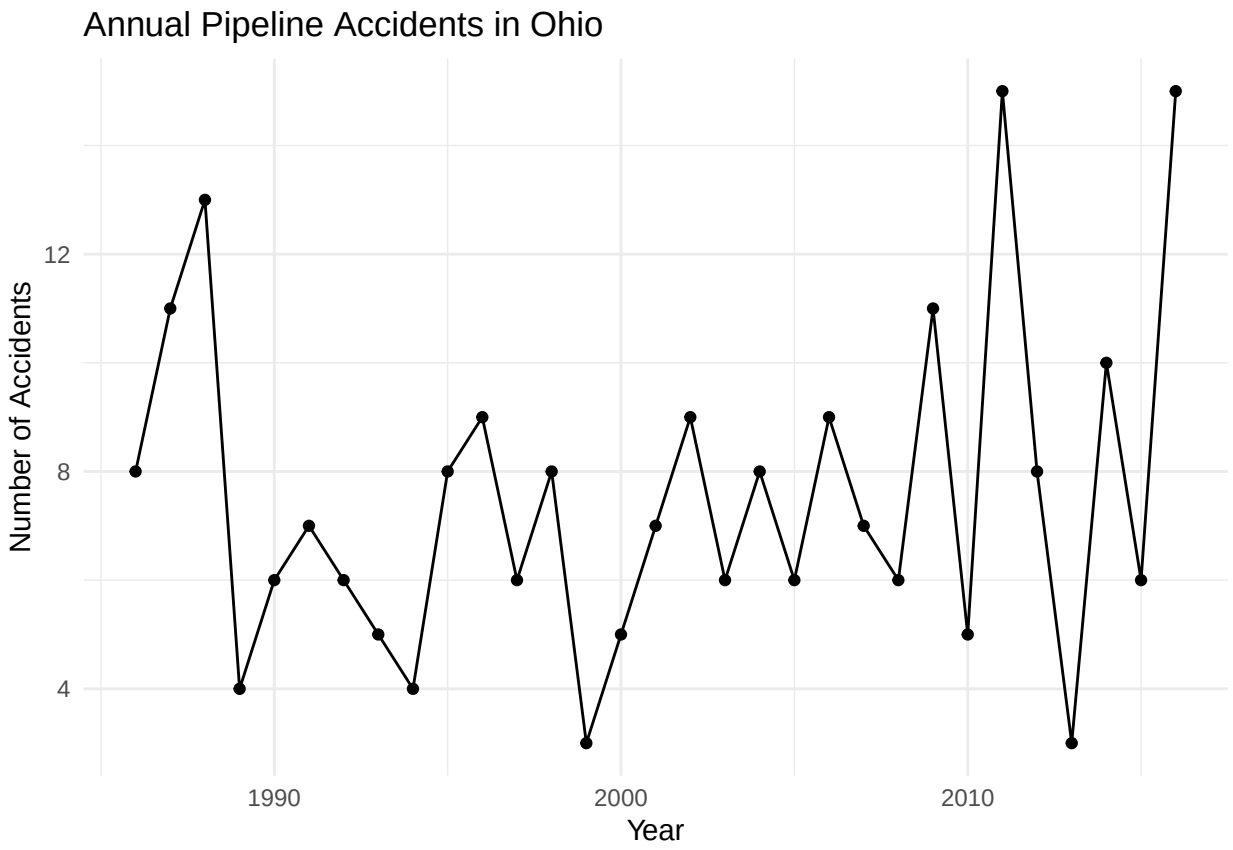
```r
print(head(pipeline[, grepl("Date|date|year|Year|TIME|Time", colnames(pipeline), ignore.case = TRUE)]))
```

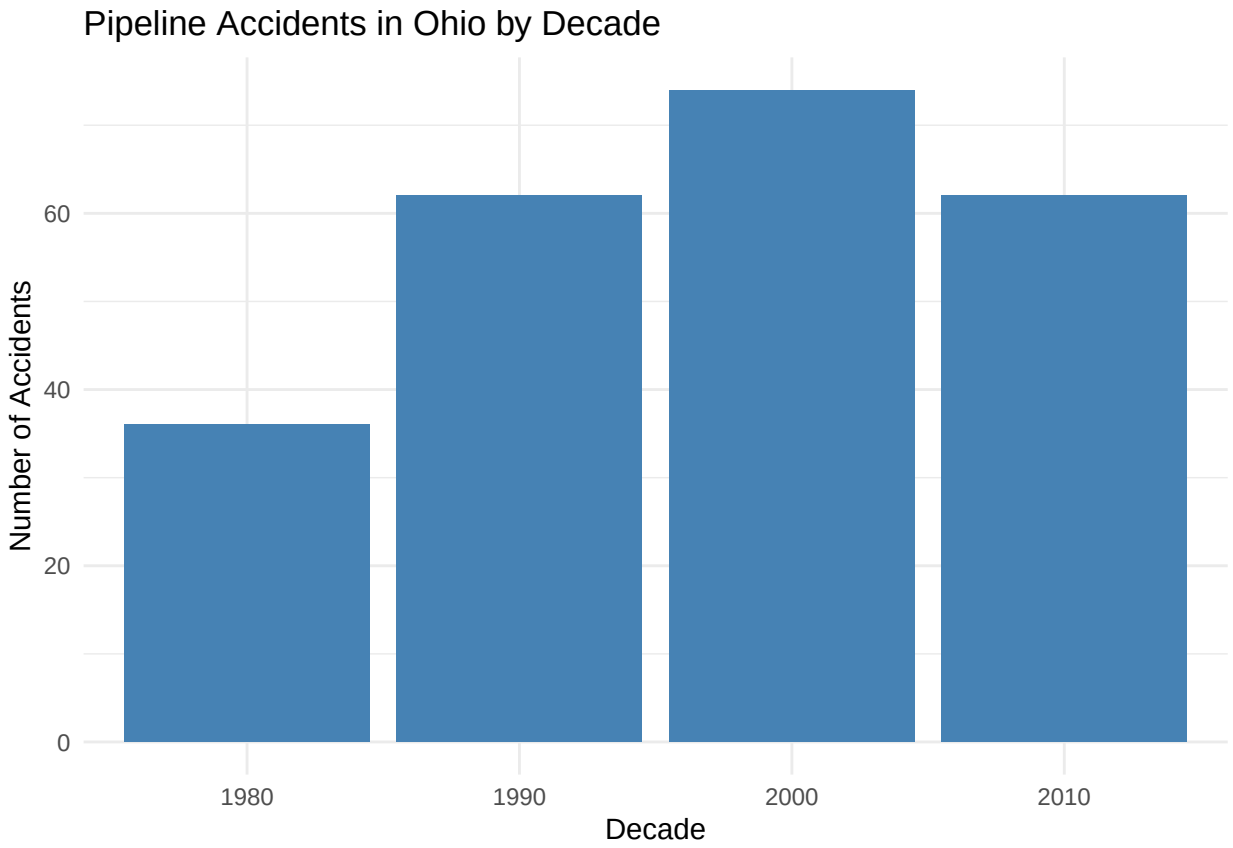
```
## [1] "13/05/1987" "05/03/1987" "03/03/1987" "10/02/1992" "20/08/2007"
## [6] "05/06/1994"
```

```r
# If RPTID contains year information, we could try to extract it
# Now here I am extracting dates data from RFID column
if("RPTID" %in% colnames(pipeline)) {
  head(pipeline$RPTID)
  # Extracting year from RPTID column
  if(all(nchar(as.character(pipeline$RPTID)) >= 8)) {
    # Extracting first 4 characters as year for ease
    pipeline$extracted_year <- as.numeric(substr(as.character(pipeline$RPTID), 1, 4))

    # Just double checking the year data
    valid_years <- pipeline$extracted_year >= 1900 & pipeline$extracted_year <= 2023
    print(paste("Potentially valid years from RPTID:", sum(valid_years, na.rm=TRUE)))

    if(sum(valid_years, na.rm=TRUE) > 0) {
      # Here I only filter data of Ohio
      ohio_accidents$Year <- as.numeric(substr(as.character(ohio_accidents$RPTID), 1, 4))
      ohio_accidents$Decade <- floor(ohio_accidents$Year/10) * 10

      # Then I conduct a temporal analysis
      annual_counts <- ohio_accidents %>%
        group_by(Year) %>%
        summarise(Accidents = n(), .groups = "drop")

      # Plotting annual trends
      print(ggplot(annual_counts, aes(x = Year, y = Accidents)) +
```

```
        geom_line() +
        geom_point() +
        theme_minimal() +
        labs(title = "Annual Pipeline Accidents in Ohio",
             x = "Year",
             y = "Number of Accidents"))

      # Then also a Decadal trend
      decade_counts <- ohio_accidents %>%
        group_by(Decade) %>%
        summarise(Accidents = n(), .groups = "drop")

      print(ggplot(decade_counts, aes(x = as.factor(Decade), y = Accidents)) +
        geom_bar(stat = "identity", fill = "steelblue") +
        theme_minimal() +
        labs(title = "Pipeline Accidents in Ohio by Decade",
             x = "Decade",
             y = "Number of Accidents"))
    }
  }
}
```

```
## [1] "Potentially valid years from RPTID: 8889"
```






```
# I need to add the decade field to my spatial dataset
# So here I extract year from RPTID for the sf object
ohio_accidents_sf$Year <- as.numeric(substr(as.character(ohio_accidents_sf$RPTID), 1, 4))

# Then I create decade column
ohio_accidents_sf$Decade <- floor(ohio_accidents_sf$Year/10) * 10

# And I verify if the extraction worked
head(ohio_accidents_sf[, c("RPTID", "Year", "Decade")])
```

```
## Simple feature collection with 6 features and 3 fields
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: 189088.9 ymin: 4338187 xmax: 190551.7 ymax: 4498616
## Projected CRS: NAD83 / UTM zone 17N
##      RPTID Year Decade                 geometry
## 1 19930222 1993   1990 POINT (190551.7 4498616)
## 2 19920104 1992   1990 POINT (189088.9 4405980)
## 3 19880030 1988   1980   POINT (189677 4371149)
## 4 19960031 1996   1990 POINT (189753.2 4371528)
## 5 19980253 1998   1990 POINT (189254.7 4341472)
## 6 19870031 1987   1980 POINT (189969.1 4338187)
```

```
# Doing the decade analysis again
accidents_1980s <- ohio_accidents_sf[ohio_accidents_sf$Decade == 1980,]
accidents_1990s <- ohio_accidents_sf[ohio_accidents_sf$Decade == 1990,]
accidents_2000s <- ohio_accidents_sf[ohio_accidents_sf$Decade == 2000,]
accidents_2010s <- ohio_accidents_sf[ohio_accidents_sf$Decade == 2010,]
```

```r
# I am verifying if the subsetting worked or not
print(paste("1980s accidents:", nrow(accidents_1980s)))
```

```
## [1] "1980s accidents: 35"
```

```r
print(paste("1990s accidents:", nrow(accidents_1990s)))
```

```
## [1] "1990s accidents: 62"
```

```r
print(paste("2000s accidents:", nrow(accidents_2000s)))
```

```
## [1] "2000s accidents: 73"
```

```r
print(paste("2010s accidents:", nrow(accidents_2010s)))
```

```
## [1] "2010s accidents: 56"
```

```r
# I calculate summary stats here
decade_stats <- data.frame(
  Decade = c("1980s", "1990s", "2000s", "2010s"),
  Accident_Count = c(nrow(accidents_1980s), nrow(accidents_1990s), nrow(accidents_2000s), nrow(accidents
  Percent_of_Total = c(nrow(accidents_1980s), nrow(accidents_1990s), nrow(accidents_2000s), nrow(accider
)

# And acquire decade statistics
print(decade_stats)
```

```
##   Decade Accident_Count Percent_of_Total
## 1  1980s             35         15.48673
## 2  1990s             62         27.43363
## 3  2000s             73         32.30088
## 4  2010s             56         24.77876
```

```r
# Water_proximity_analysis
# Trying to count number of accidents near water resources individually
num_near_waterbodies <- sum(rowSums(accidents_near_waterbodies) > 0)
num_near_flowlines <- sum(rowSums(accidents_near_flowlines) > 0)

# Then, for total near water, I am combining the logical matrices
# First, checking if they all have same dimensions
if (identical(dim(accidents_near_waterbodies), dim(accidents_near_flowlines))) {
  # Then created a new logical matrix
  near_any_water <- matrix(FALSE, nrow=nrow(accidents_near_waterbodies),
                           ncol=ncol(accidents_near_waterbodies))

  # And I set it as TRUE if either original matrix has TRUE
  for (i in 1:nrow(accidents_near_waterbodies)) {
    for (j in 1:ncol(accidents_near_waterbodies)) {
      near_any_water[i,j] <- accidents_near_waterbodies[i,j] | accidents_near_flowlines[i,j]
    }
  }

  total_near_water <- sum(rowSums(near_any_water) > 0)
} else {
  # I have put a backup approach if dimensions don't match to avoid mismatch
  accidents_near_water_list <- lapply(1:nrow(ohio_accidents_sf), function(i) {
    any(accidents_near_waterbodies[i,]) | any(accidents_near_flowlines[i,])
```

```
  })
  total_near_water <- sum(unlist(accidents_near_water_list))
}

# Producing a summary table
water_summary <- data.frame(
  Category = c("Near Waterbodies (lakes, ponds)",
               "Near Flowlines (rivers, streams)",
               "Near Any Water Feature",
               "Total Accidents"),
  Count = c(num_near_waterbodies,
            num_near_flowlines,
            total_near_water,
            nrow(ohio_accidents_sf)),
  Percentage = c(num_near_waterbodies/nrow(ohio_accidents_sf)*100,
                 num_near_flowlines/nrow(ohio_accidents_sf)*100,
                 total_near_water/nrow(ohio_accidents_sf)*100,
                 100)
)

print(water_summary)
```

```
##                           Category Count Percentage
## 1  Near Waterbodies (lakes, ponds)    20   8.849558
## 2 Near Flowlines (rivers, streams)     8   3.539823
## 3           Near Any Water Feature    25  11.061947
## 4                  Total Accidents   226 100.000000
```

```
# I am setting to True if near waterbody accidents fall within the buffer
# Also repeated the same for flowline data as well
ohio_accidents_sf$near_waterbody <- rowSums(accidents_near_waterbodies) > 0
ohio_accidents_sf$near_flowline <- rowSums(accidents_near_flowlines) > 0

# Then I create a categorical water proximity to divide and aggregate accidents
ohio_accidents_sf$water_proximity <- case_when(
  ohio_accidents_sf$near_waterbody & ohio_accidents_sf$near_flowline ~ "Near Both",
  ohio_accidents_sf$near_waterbody ~ "Near Waterbody Only",
  ohio_accidents_sf$near_flowline ~ "Near Flowline Only",
  TRUE ~ "Not Near Water"
)

# I am counting accidents near water resources at different distances
num_near_waterbody_100m <- sum(apply(accidents_near_waterbody_100m, 1, any))
num_near_waterbody_500m <- sum(apply(accidents_near_waterbody_500m, 1, any))
num_near_flowline_100m <- sum(apply(accidents_near_flowline_100m, 1, any))
num_near_flowline_500m <- sum(apply(accidents_near_flowline_500m, 1, any))

# And then I create a summary table for water proximity
water_proximity_summary <- data.frame(
  Water_Feature = c("Lakes/Ponds", "Lakes/Ponds", "Rivers/Streams", "Rivers/Streams"),
  Buffer_Distance = c("100m", "500m", "100m", "500m"),
  Accidents = c(num_near_waterbody_100m, num_near_waterbody_500m,
                num_near_flowline_100m, num_near_flowline_500m),
  Percentage = c(num_near_waterbody_100m/nrow(ohio_accidents_sf)*100,
```

```
                num_near_waterbody_500m/nrow(ohio_accidents_sf)*100,
                num_near_flowline_100m/nrow(ohio_accidents_sf)*100,
                num_near_flowline_500m/nrow(ohio_accidents_sf)*100)
)

# Then I print water proximity summary
print("Pipeline Accidents Proximity to Water Resources:")
```

```
## [1] "Pipeline Accidents Proximity to Water Resources:"
```

```
print(water_proximity_summary)
```

```
##    Water_Feature Buffer_Distance Accidents Percentage
## 1    Lakes/Ponds            100m        20   8.849558
## 2    Lakes/Ponds            500m       123  54.424779
## 3 Rivers/Streams            100m         8   3.539823
## 4 Rivers/Streams            500m        58  25.663717
```

```
# I also calculate accidents near any water feature (100m)
accidents_near_any_water_100m <- rowSums(cbind(
  accidents_near_waterbody_100m,
  accidents_near_flowline_100m)) > 0

# Likewise also calculate accidents near any water feature (500m)
accidents_near_any_water_500m <- rowSums(cbind(
  accidents_near_waterbody_500m,
  accidents_near_flowline_500m)) > 0

# Summary
water_summary_combined <- data.frame(
  Buffer_Distance = c("100m", "500m"),
  Accidents_Near_Any_Water = c(sum(accidents_near_any_water_100m),
                               sum(accidents_near_any_water_500m)),
  Percentage_Near_Any_Water = c(sum(accidents_near_any_water_100m)/nrow(ohio_accidents_sf)*100,
                                sum(accidents_near_any_water_500m)/nrow(ohio_accidents_sf)*100)
)

print("Pipeline Accidents Near Any Water Feature:")
```

```
## [1] "Pipeline Accidents Near Any Water Feature:"
```

```
print(water_summary_combined)
```

```
##   Buffer_Distance Accidents_Near_Any_Water Percentage_Near_Any_Water
## 1            100m                      25                  11.06195
## 2            500m                     146                  64.60177
```

```
# Point Pattern Analysis
library(spatstat)

# I am creating a spatial window for point pattern analysis using Ohio boundary
ohio_win <- as.owin(st_geometry(ohio_boundary))

# Then converting Ohio accident points to ppp format
ohio_pts <- st_coordinates(ohio_accidents_sf)
ohio_ppp <- ppp(ohio_pts[,1], ohio_pts[,2], window = ohio_win)
```

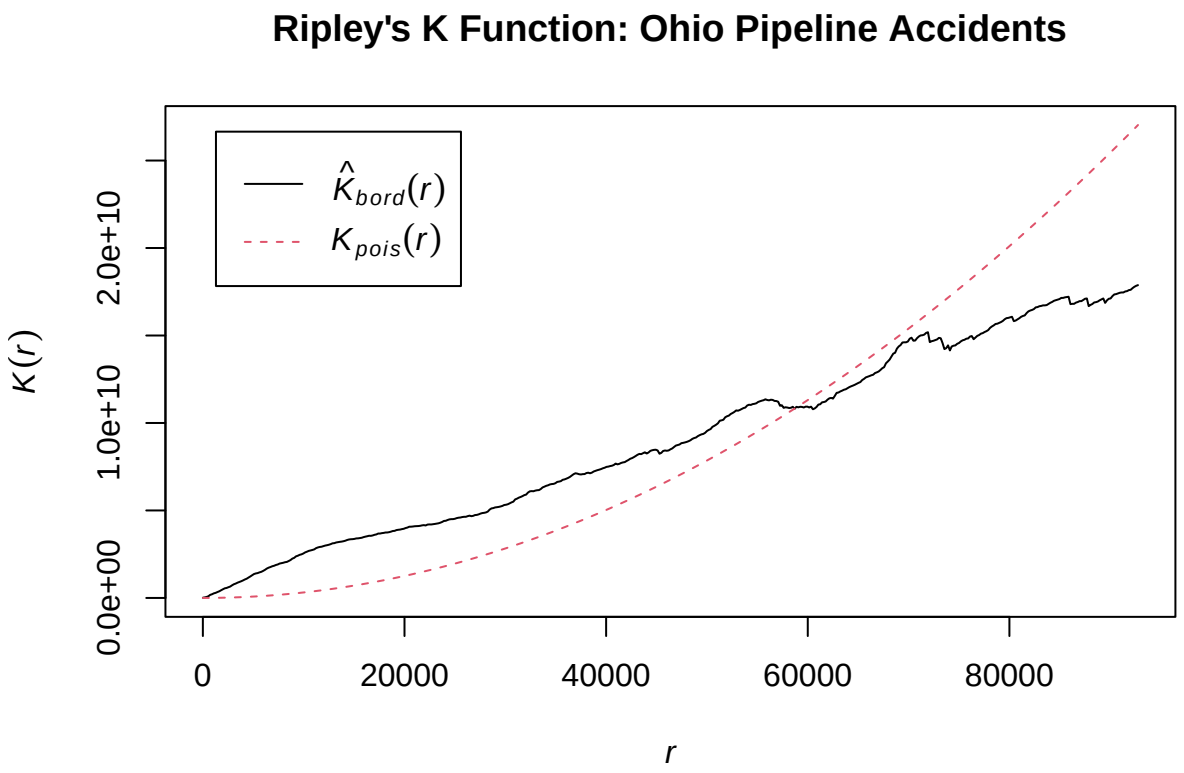
```
## Warning: 1 point was rejected as lying outside the specified window
```

```
## Warning: data contain duplicated points
```

```
# Then conducting a test for Complete Spatial Randomness (CSR)
# Nearest Neighbor Analysis
nn_test <- clarkevans.test(ohio_ppp)
print(nn_test)
```

```
##
##  Clark-Evans test
##  CDF correction
##  Z-test
##
## data:  ohio_ppp
## R = 0.52766, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

```
# I am implementing Ripley's K function to check clustering at multiple distances
k_ohio <- Kest(ohio_ppp, correction = "border")
plot(k_ohio, main="Ripley's K Function: Ohio Pipeline Accidents")
```




```
# I'm building the crash point pattern and running a 5-km kernel-density estimate
# (in UTM coordinates) so I can spot the main clusters before any further analysis.

ohio_utm     <- st_transform(ohio_boundary, 26917)
ohio_acc_utm <- st_transform(ohio_accidents_sf, 26917)
ohio_win <- as.owin(st_geometry(ohio_utm))
coords   <- st_coordinates(ohio_acc_utm)
suppressWarnings({
  ohio_ppp <- ppp(coords[,1], coords[,2], window = ohio_win, check = FALSE)
})
```

```r
ohio_density <- density.ppp(ohio_ppp, sigma = 5000, edge = TRUE)

# I'm switching the data back to geographic (long/lat) coordinates
# so my next sf plots line up properly on the map.

ohio_bound_geo <- st_transform(ohio_boundary, 4326)
ohio_acc_geo   <- st_transform(ohio_accidents_sf, 4326)

# I'm converting the KDE result (an "im" object from spatstat) into a terra SpatRaster
# so I can process and visualize it more easily

sgdf <- as(ohio_density, "SpatialGridDataFrame")
proj4string(sgdf) <- st_crs(26917)$proj4string   # Tell it "this is UTM"

# then I convert to a raster object
r <- raster(sgdf)

# then I apply log transformation for plotting the map later
r_log <- calc(r, fun = function(x) log10(x * 100 + 1))

# Reprojection
r_wgs84 <- projectRaster(r_log, crs = st_crs(4326)$proj4string)

# I'm creating a mask from the Ohio boundary to make sure the KDE only shows up within the state limits

ohio_sp <- as(ohio_bound_geo, "Spatial")
r_masked <- mask(r_wgs84, ohio_sp)
```

```r
#The following Kernel Density Estimation implementation is partly modified and inspired from:
#Baddeley, A., Rubak, E., and Turner, R. (2022)
#spatstat: Spatial Point Pattern Analysis, Model-Fitting, Simulation, Tests
#https://cran.r-project.org/package=spatstat
#Downloaded: April 10, 2025
# Hotspot Detection using Kernel Density Estimation(KDE)

# I am creating a point pattern object for KDE
library(spatstat)
ohio_win <- as.owin(st_geometry(ohio_boundary))
ohio_pts <- st_coordinates(ohio_accidents_sf)
ohio_ppp <- ppp(ohio_pts[,1], ohio_pts[,2], window = ohio_win)
```

```
## Warning: 1 point was rejected as lying outside the specified window
```

```
## Warning: data contain duplicated points
```

```r
bw_cv <- bw.diggle(ohio_ppp)
print(paste("Optimal bandwidth from cross-validation:", round(bw_cv, 2), "meters"))
```

```
## [1] "Optimal bandwidth from cross-validation: 2450.52 meters"
```

```r
# I am implementing Silverman's rule of thumb in case computation is intense to avoid session crashout
n <- npoints(ohio_ppp)
sigma_x <- sd(ohio_pts[,1])
sigma_y <- sd(ohio_pts[,2])
sigma <- (sigma_x + sigma_y) / 2
bw_silverman <- 0.9 * sigma * n^(-0.2)
```

```r
print(paste("Silverman's rule of thumb bandwidth:", round(bw_silverman, 2), "meters"))
```

```
## [1] "Silverman's rule of thumb bandwidth: 29397.29 meters"
```

```r
# Then I select a suitable bandwidth
bandwidth <- 10000

# And I compute the KDE
ohio_density <- density.ppp(ohio_ppp, sigma = bandwidth)

# Then I convert KDE to a raster for easier analysis
library(terra)
density_rast <- rast(ohio_density)
crs(density_rast) <- "EPSG:26917"

# And I calculate statistics for the density values
density_values <- values(density_rast)
density_stats <- data.frame(
  Statistic = c("Minimum Density", "Maximum Density", "Mean Density", "Median Density"),
  Value = c(min(density_values, na.rm = TRUE),
            max(density_values, na.rm = TRUE),
            mean(density_values, na.rm = TRUE),
            median(density_values, na.rm = TRUE))
)
print("Kernel Density Statistics:")
```

```
## [1] "Kernel Density Statistics:"
```

```r
print(density_stats)
```

```
##         Statistic         Value
## 1 Minimum Density -2.871536e-25
## 2 Maximum Density  3.188227e-08
## 3    Mean Density  2.106805e-09
## 4  Median Density  9.267846e-10
```

```r
# Here I identify hotspots using a threshold approach
density_mean <- mean(density_values, na.rm = TRUE)
density_sd <- sd(density_values, na.rm = TRUE)
hotspot_threshold <- density_mean + 2 * density_sd

# I am creating a hotspot mask
hotspot_mask <- density_rast > hotspot_threshold
names(hotspot_mask) <- "hotspot"

# So I can calculate area of hotspots
hotspot_cells <- sum(values(hotspot_mask), na.rm = TRUE)
cell_area <- prod(res(hotspot_mask)) / 1000000  # cell area in sq km
hotspot_area <- hotspot_cells * cell_area
total_area <- ncell(hotspot_mask) * cell_area
hotspot_percentage <- (hotspot_area / total_area) * 100

# Then I convert hotspot mask to vector for intersection with point data
hotspot_polygons <- as.polygons(hotspot_mask, dissolve = TRUE)
hotspot_sf <- st_as_sf(hotspot_polygons)
```

```r
# Trying to count accidents in hotspots
accidents_in_hotspots <- st_intersects(ohio_accidents_sf, hotspot_sf)
num_in_hotspots <- sum(lengths(accidents_in_hotspots) > 0)
pct_in_hotspots <- (num_in_hotspots / nrow(ohio_accidents_sf)) * 100

# Also I am calculating hotspot intensity accidents per sq km within hotspots
hotspot_intensity <- num_in_hotspots / hotspot_area

# Summarizing all the hotspot stats
hotspot_stats <- data.frame(
  Statistic = c(
    "Hotspot Threshold Value",
    "Number of Hotspots",
    "Total Hotspot Area (sq km)",
    "Percentage of Ohio in Hotspots",
    "Number of Accidents in Hotspots",
    "Percentage of Accidents in Hotspots",
    "Accident Density in Hotspots (per sq km)"
  ),
  Value = c(
    round(hotspot_threshold, 8),
    length(unique(st_cast(hotspot_sf, "POLYGON"))),
    round(hotspot_area, 2),
    round(hotspot_percentage, 2),
    num_in_hotspots,
    round(pct_in_hotspots, 2),
    round(hotspot_intensity, 2)
  )
)
```

```
## Warning in st_cast.sf(hotspot_sf, "POLYGON"): repeating attributes for all
## sub-geometries for which they may not be constant
```

```r
print("Hotspot Analysis Statistics:")
```

```
## [1] "Hotspot Analysis Statistics:"
```

```r
print(hotspot_stats)
```

```
##                                 Statistic       Value
## 1                  Hotspot Threshold Value 1.00000e-08
## 2                       Number of Hotspots 2.00000e+00
## 3                Total Hotspot Area (sq km) 5.21527e+03
## 4            Percentage of Ohio in Hotspots 3.55000e+00
## 5           Number of Accidents in Hotspots 2.24000e+02
## 6       Percentage of Accidents in Hotspots 9.91200e+01
## 7 Accident Density in Hotspots (per sq km) 4.00000e-02
```

```r
# I also calculating the distance from accidents to the nearest hotspot centroid
hotspot_centroids <- st_centroid(st_cast(hotspot_sf, "POLYGON"))
```

```
## Warning in st_cast.sf(hotspot_sf, "POLYGON"): repeating attributes for all
## sub-geometries for which they may not be constant
```

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

```r
distances_to_hotspots <- st_distance(ohio_accidents_sf, hotspot_centroids)
min_distances <- apply(distances_to_hotspots, 1, min)

# I create a dataframe for Distance related stats
distance_stats <- data.frame(
  Statistic = c("Minimum Distance to Hotspot", "Mean Distance to Hotspot",
                "Median Distance to Hotspot", "Maximum Distance to Hotspot"),
  Value = c(min(min_distances), mean(min_distances),
            median(min_distances), max(min_distances))
)
distance_stats$Value <- round(as.numeric(distance_stats$Value))
print("Distance to Hotspot Statistics (meters):")
```

```
## [1] "Distance to Hotspot Statistics (meters):"
```

```r
print(distance_stats)
```

```
##                      Statistic  Value
## 1 Minimum Distance to Hotspot    573
## 2    Mean Distance to Hotspot  27668
## 3  Median Distance to Hotspot  18948
## 4 Maximum Distance to Hotspot 139987
```

```r
# Printing summary
hotspot_summary <- data.frame(
  Analysis_Component = "Hotspot Detection (KDE)",
  Key_Finding = paste0(
    round(pct_in_hotspots, 1), "% of accidents occur in hotspots covering only ",
    round(hotspot_percentage, 1), "% of Ohio's area, with an accident density of ",
    round(hotspot_intensity, 1), " accidents per sq km"
  )
)

print("Hotspot Analysis Summary for Report:")
```

```
## [1] "Hotspot Analysis Summary for Report:"
```

```r
print(hotspot_summary)
```

```
##        Analysis_Component
## 1 Hotspot Detection (KDE)
##
## 1 99.1% of accidents occur in hotspots covering only 3.6% of Ohio's area, with an accident density o
```

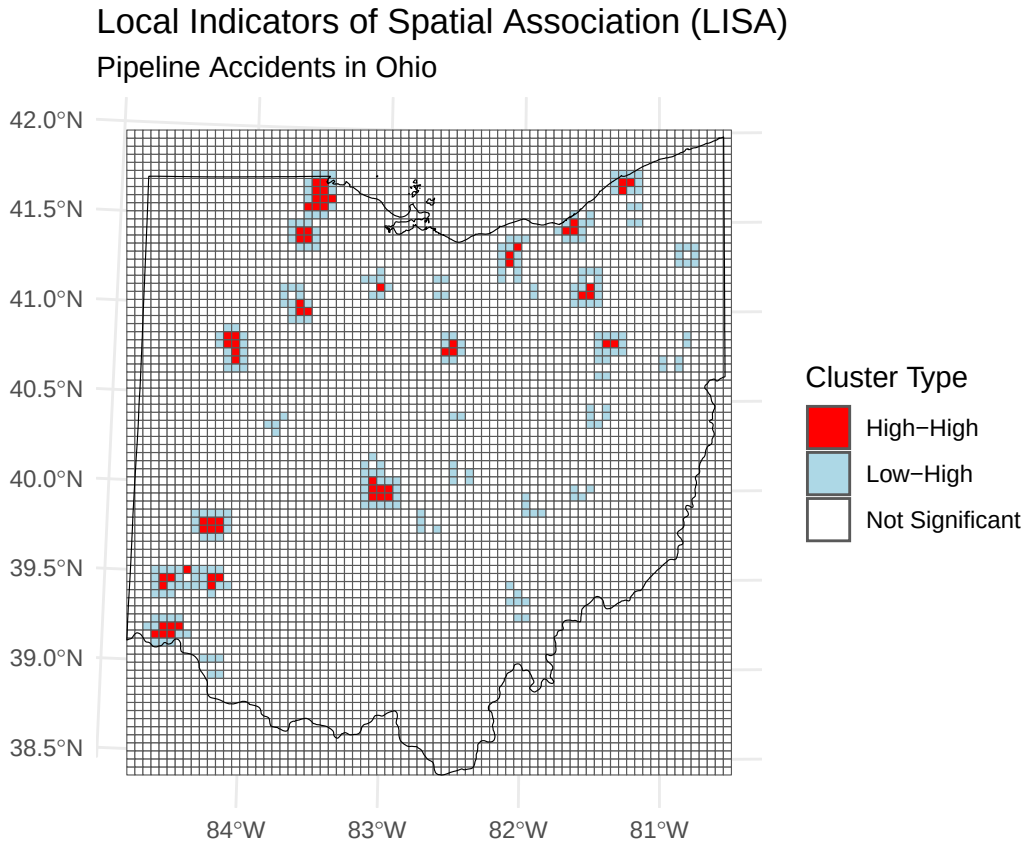
```r
#The following implementation of Local Indicators of Spatial Association (LISA)
#is modified and inspired  from:
#Bivand, R. and Anselin, L. (2022)
#spdep: Spatial Dependence: Weighting Schemes, Statistics and Models
#https://cran.r-project.org/package=spdep
#Downloaded: April 12, 2025


# Spatial Autocorrelation Analysis
# First I create a grid for analysis
ohio_grid <- st_make_grid(ohio_boundary, cellsize = 5000, square = TRUE) %>%
  st_sf() %>%
```

```
  mutate(grid_id = row_number())

# Then I count accidents in each grid cell
grid_counts <- st_intersects(ohio_grid, ohio_accidents_sf)
ohio_grid$accident_count <- lengths(grid_counts)

# To create neighbors list using queen contiguity method
grid_nb <- poly2nb(ohio_grid, queen = TRUE)
grid_weights <- nb2listw(grid_nb, style = "W")

# And I also calculate global Moran's I
moran_i <- moran.test(ohio_grid$accident_count, grid_weights)
print(moran_i)
```

```
##
##  Moran I test under randomisation
##
## data:  ohio_grid$accident_count
## weights: grid_weights
##
## Moran I statistic standard deviate = 23.021, p-value < 2.2e-16
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##      1.434471e-01     -1.666944e-04      3.891815e-05
```

```
# Also Local Moran's I for cluster detection
local_moran <- localmoran(ohio_grid$accident_count, grid_weights)
ohio_grid$local_moran_i <- local_moran[, 1]
ohio_grid$p_value <- local_moran[, 5]

# Here I am trying to identify significant clusters to check how similar each grid cell's accident coun
ohio_grid$cluster_type <- case_when(
  ohio_grid$local_moran_i > 0 & ohio_grid$accident_count > mean(ohio_grid$accident_count) &
    ohio_grid$p_value <= 0.05 ~ "High-High",
  ohio_grid$local_moran_i > 0 & ohio_grid$accident_count < mean(ohio_grid$accident_count) &
    ohio_grid$p_value <= 0.05 ~ "Low-Low",
  ohio_grid$local_moran_i < 0 & ohio_grid$accident_count > mean(ohio_grid$accident_count) &
    ohio_grid$p_value <= 0.05 ~ "High-Low",
  ohio_grid$local_moran_i < 0 & ohio_grid$accident_count < mean(ohio_grid$accident_count) &
    ohio_grid$p_value <= 0.05 ~ "Low-High",
  TRUE ~ "Not Significant"
)

# LISA cluster map
ggplot() +
  geom_sf(data = ohio_grid, aes(fill = cluster_type)) +
  geom_sf(data = ohio_boundary, fill = NA, color = "black") +
  scale_fill_manual(values = c("High-High" = "red",
                               "Low-Low" = "blue",
                               "High-Low" = "pink",
                               "Low-High" = "lightblue",
                               "Not Significant" = "white")) +
  theme_minimal() +
```

```
  labs(title = "Local Indicators of Spatial Association (LISA)",
       subtitle = "Pipeline Accidents in Ohio",
       fill = "Cluster Type")
```




```
# # I'm calculating the directional distribution using a Standard Deviational Ellipse
# to understand the overall spread and orientation of accidents-starting with the mean center.

#The following directional distribution analysis (Standard Deviational Ellipse)
#implementation is based on:
#Pebesma, E. and Bivand, R. (2023)
#Spatial Data Science: With Applications in R
#https://r-spatial.org/book/
#Downloaded: April 11, 2025

mean_center <- st_coordinates(st_centroid(st_union(ohio_accidents_sf)))

# Then I extract the coordinates of all accident points
coords <- st_coordinates(ohio_accidents_sf)

# And calculate distance of each point from mean center
dists <- sqrt((coords[,1] - mean_center[1])^2 + (coords[,2] - mean_center[2])^2)
std_dist <- sd(dists)

# Then I create a buffer like circle with radius equal to standard distance
std_circle <- st_buffer(st_point(mean_center), std_dist) %>% st_sfc(crs = 26917)

# Then I calculate directional trend using covariance of coordinates
```

```r
cov_matrix <- cov(coords)
eigen_result <- eigen(cov_matrix)
eigen_vectors <- eigen_result$vectors
eigen_values <- eigen_result$values


angle <- atan2(eigen_vectors[2,1], eigen_vectors[1,1])

# And I calculate major and minor axes
major_axis <- sqrt(eigen_values[1]) * 2
minor_axis <- sqrt(eigen_values[2]) * 2

# Then I calculate all the key spatial pattern related statistics
# 1. Directional Distribution Statistics
dir_stats <- data.frame(
  Statistic = c("Mean Center X", "Mean Center Y", "Standard Distance",
                "Major Axis Length", "Minor Axis Length", "Ratio (Major/Minor)",
                "Orientation Angle (degrees)"),
  Value = c(
    round(mean_center[1], 2),
    round(mean_center[2], 2),
    round(std_dist, 2),
    round(major_axis, 2),
    round(minor_axis, 2),
    round(major_axis/minor_axis, 2),
    round(angle * 180/pi, 2)
  )
)
print("Directional Distribution Statistics:")
```

```
## [1] "Directional Distribution Statistics:"
```

```r
print(dir_stats)
```

```
##                     Statistic      Value
## 1               Mean Center X  328651.72
## 2               Mean Center Y 4486806.86
## 3          Standard Distance   46194.73
## 4          Major Axis Length  232184.22
## 5          Minor Axis Length  145523.76
## 6        Ratio (Major/Minor)       1.60
## 7 Orientation Angle (degrees)    -147.03
```

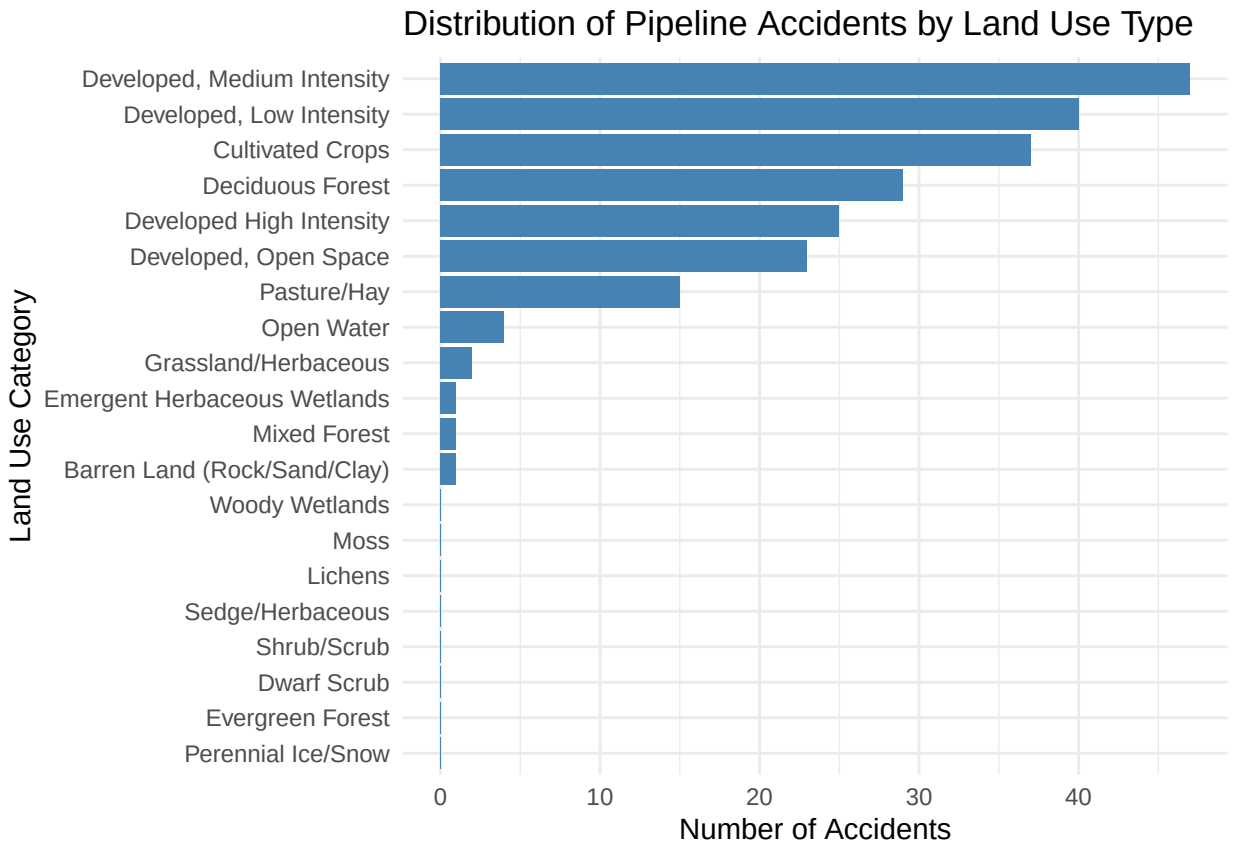
```r
# Repojecting the NLCD to match prev. CRS
nlcd_proj <- project(nlcd, "EPSG:26917")

# I am extracting land use values at different accident points
# Convert accidents to terra vector for extraction
ohio_accidents_vect <- vect(ohio_accidents_sf)

# Then I am extracting the land use classes at accident locations
accident_landuse <- terra::extract(nlcd_proj, ohio_accidents_vect)

# Checking
head(accident_landuse)
```

```
##   ID                       Class
## 1  1            Cultivated Crops
## 2  2    Developed, Low Intensity
## 3  3            Deciduous Forest
## 4  4       Developed, Open Space
## 5  5 Developed, Medium Intensity
## 6  6       Developed, Open Space
```

```
# Then I am creating a frequency table from the Class column present in the data
landuse_freq <- table(accident_landuse$Class)
landuse_freq_df <- as.data.frame(landuse_freq)
colnames(landuse_freq_df) <- c("Class", "Count")

# Calculating percentages
landuse_freq_df$Percentage <- landuse_freq_df$Count / sum(landuse_freq_df$Count) * 100

# Setting up Order by count
landuse_freq_df <- landuse_freq_df[order(-landuse_freq_df$Count),]

# Checking results
print(landuse_freq_df)
```

```
##                          Class Count Percentage
## 5   Developed, Medium Intensity    47 20.8888889
## 4      Developed, Low Intensity    40 17.7777778
## 18             Cultivated Crops    37 16.4444444
## 8              Deciduous Forest    29 12.8888889
## 6      Developed High Intensity    25 11.1111111
## 3         Developed, Open Space    23 10.2222222
## 17                  Pasture/Hay    15  6.6666667
## 1                    Open Water     4  1.7777778
## 13         Grassland/Herbaceous     2  0.8888889
## 7  Barren Land (Rock/Sand/Clay)     1  0.4444444
## 10                 Mixed Forest     1  0.4444444
## 20 Emergent Herbaceous Wetlands     1  0.4444444
## 2            Perennial Ice/Snow     0  0.0000000
## 9              Evergreen Forest     0  0.0000000
## 11                  Dwarf Scrub     0  0.0000000
## 12                  Shrub/Scrub     0  0.0000000
## 14             Sedge/Herbaceous     0  0.0000000
## 15                      Lichens     0  0.0000000
## 16                         Moss     0  0.0000000
## 19               Woody Wetlands     0  0.0000000
```

```
# Plot
ggplot(landuse_freq_df, aes(x = reorder(Class, Count), y = Count)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  coord_flip() +
  theme_minimal() +
  labs(title = "Distribution of Pipeline Accidents by Land Use Type",
       x = "Land Use Category",
       y = "Number of Accidents")
```




```r
# Environmental sensitivity analysis
landuse_sensitivity <- data.frame(
  Land_Use = c(
    "Open Water", "Woody Wetlands", "Emergent Herbaceous Wetlands",
    "Deciduous Forest", "Evergreen Forest", "Mixed Forest",
    "Barren Land (Rock/Sand/Clay)", "Grassland/Herbaceous", "Shrub/Scrub",
    "Pasture/Hay", "Cultivated Crops",
    "Developed, Open Space", "Developed, Low Intensity",
    "Developed, Medium Intensity", "Developed High Intensity"
  ),
  Sensitivity_Weight = c(
    5, 5, 5,
    4, 4, 4,
    3, 3, 3,
    2, 2,
    1, 1, 1, 1
  )
)

# Here I join sensitivity weights to the accident land use data
accident_sensitivity <- merge(
  accident_landuse,
  landuse_sensitivity,
  by.x = "Class",
  by.y = "Land_Use",
  all.x = TRUE
)
```

```r
# Likewise I calculate avg. environmental sensitivity of accident locations
avg_sensitivity <- mean(accident_sensitivity$Sensitivity_Weight, na.rm = TRUE)
print(paste("Average Environmental Sensitivity Index of Accident Locations:", round(avg_sensitivity, 2)))
```

```
## [1] "Average Environmental Sensitivity Index of Accident Locations: 1.75"
```

```r
# So that I can group by sensitivity levels
sensitivity_groups <- accident_sensitivity %>%
  group_by(Sensitivity_Weight) %>%
  summarize(
    Accident_Count = n(),
    Percentage = n() / nrow(accident_sensitivity) * 100
  )

print("Accidents by Environmental Sensitivity Level:")
```

```
## [1] "Accidents by Environmental Sensitivity Level:"
```

```r
print(sensitivity_groups)
```

```
## # A tibble: 6 x 3
##   Sensitivity_Weight Accident_Count Percentage
##                <dbl>          <int>      <dbl>
## 1                  1            135     59.7
## 2                  2             52     23.0
## 3                  3              3      1.33
## 4                  4             30     13.3
## 5                  5              5      2.21
## 6                 NA              1      0.442
```

```r
# Add sensitivity to the ohio_accidents_sf
ohio_accidents_sf$sensitivity <- factor(
  accident_sensitivity$Sensitivity_Weight,
  levels = c(1, 2, 3, 4, 5),
  labels = c("Very Low", "Low", "Medium", "High", "Very High")
)
```

```r
#The following code for accessing U.S. Census API data is partly modified and inspired from:
#Walker, K. (2023)
#tidycensus: Load US Census Boundary and Attribute Data as 'tidyverse' and 'sf'-Ready Data Frames
#https://walker-data.com/tidycensus/
#Downloaded: April 16, 2025

# Setting up my census API key to fetch the population data as ACS has server fetching issue at times
census_api_key("57f33a585201123b4db3ef8434bc8310fd3df315", install = TRUE, overwrite = TRUE)
```

```
## Your original .Renviron will be backed up and stored in your R HOME directory if needed.
```

```
## Your API key has been stored in your .Renviron and can be accessed by Sys.getenv("CENSUS_API_KEY").
## To use now, restart R or run `readRenviron("~/.Renviron")`
```

```
## [1] "57f33a585201123b4db3ef8434bc8310fd3df315"
```

```r
# here I load  the tract file
ohio_tracts_geom <- st_read("~/Assignment/cb_2019_39_tract_500k/cb_2019_39_tract_500k.shp")
```

```
## Reading layer `cb_2019_39_tract_500k' from data source
##   `/s_home/ss1288/Assignment/cb_2019_39_tract_500k/cb_2019_39_tract_500k.shp'
```

```
##   using driver `ESRI Shapefile'
## Simple feature collection with 2948 features and 9 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -84.82016 ymin: 38.4032 xmax: -80.51869 ymax: 41.97752
## Geodetic CRS:  NAD83
```

```
# Trying to get Ohio tract population data from ACS
tryCatch({
  ohio_tracts_data <- get_acs(
    geography = "tract",
    variables = "B01003_001", # Total population
    state = "OH",
    year = 2019,
    geometry = FALSE
  )

  # then I do the join population data to the tract boundaries
  ohio_tracts_sf <- left_join(ohio_tracts_geom, ohio_tracts_data, by = "GEOID") %>%
    st_transform(26917)  # Match the project CRS

  # and I calculate population density (people per square km)
  ohio_tracts_sf$area_sqkm <- st_area(ohio_tracts_sf) / 1e6
  ohio_tracts_sf$pop_density <- as.numeric(ohio_tracts_sf$estimate / ohio_tracts_sf$area_sqkm)
}, error = function(e) {
  # Incase if there's an error with the API, I am creating a simpler version with estimated population
  message("Error retrieving census data")
  ohio_tracts_sf <- ohio_tracts_geom %>%
    st_transform(26917) %>%
    mutate(
      estimate = as.numeric(ALAND) / 1e6,  # Adding Crude area-based estimate
      area_sqkm = as.numeric(st_area(geometry)) / 1e6,
      pop_density = estimate / area_sqkm
    )
})
```

```
## Getting data from the 2015-2019 5-year ACS
```

```
# here I am counting  accidents per tract
ohio_tracts_sf$accident_count <- lengths(st_intersects(ohio_tracts_sf, ohio_accidents_sf))
```

```
# Just loading Ohio counties data
ohio_counties_sf <- counties(state = "OH", cb = TRUE) %>%
  st_transform(26917)
```

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

```
# and attempting to get Ohio county population data from ACS
ohio_pop <- get_acs(
  geography = "county",
  variables = "B01003_001",  # Total population
  state = "OH",
  year = 2021,
  geometry = FALSE
)
```

```
## Getting data from the 2017-2021 5-year ACS
```

```r
# then doing a left join of population data to counties using common column in both datasets
ohio_counties_sf <- left_join(ohio_counties_sf, ohio_pop %>% select(GEOID, estimate), by = "GEOID")

# here I calculate accident counts per county
accident_counts_per_county <- st_intersects(ohio_counties_sf, ohio_accidents_sf)
ohio_counties_sf$accident_count <- lengths(accident_counts_per_county)

# then I also calculate  county area in sq km
ohio_counties_sf$area_sqkm <- as.numeric(st_area(ohio_counties_sf)) / 1e6

# and I compute the accident rate per 1000 square km
ohio_counties_sf$accident_rate <- ohio_counties_sf$accident_count / ohio_counties_sf$area_sqkm * 1000

# I'm calculating the population-weighted risk (accidents per 100,000 people)
# to identify which areas face a disproportionately high accident burden relative to their population s

ohio_counties_sf$pop_risk <- ohio_counties_sf$accident_count / ohio_counties_sf$estimate * 100000

# summary table
county_risk_summary <- ohio_counties_sf %>%
  st_drop_geometry() %>%
  select(NAME, accident_count, accident_rate, estimate, pop_risk) %>%
  rename(
    County = NAME,
    Accidents = accident_count,
    "Accidents per 1000 sq km" = accident_rate,
    Population = estimate,
    "Accidents per 100,000 people" = pop_risk
  ) %>%
  arrange(desc(Accidents))

print(head(county_risk_summary, 10))
```

```
##        County Accidents Accidents per 1000 sq km Population
## 1       Allen        23                21.799998     102462
## 2        Wood        18                11.194842     131930
## 3      Butler        14                11.484863     387830
## 4    Hamilton        11                10.277247     826790
## 5  Montgomery        11                 9.137949     536136
## 6    Franklin        11                 7.813121    1313598
## 7    Cuyahoga         9                 7.579195    1263667
## 8       Lucas         8                 8.893910     431212
## 9      Summit         8                 7.357482     540567
## 10     Lorain         7                 5.472471     311737
##    Accidents per 100,000 people
## 1                     22.4473463
## 2                     13.6435989
## 3                      3.6098290
## 4                      1.3304467
## 5                      2.0517182
## 6                      0.8373947
## 7                      0.7122129
## 8                      1.8552359
## 9                      1.4799276
```

```
## 10                    2.2454826
```

```
# I conduct a Buffer Analysis around accident points
# So I first create multiple buffers around accident points (500m, 1km, 3km)
buffer_500m <- st_buffer(ohio_accidents_sf, 500)
buffer_1km <- st_buffer(ohio_accidents_sf, 1000)
buffer_3km <- st_buffer(ohio_accidents_sf, 3000)

# Then I dissolve buffers to avoid double-counting population
buffer_500m_dissolved <- st_union(buffer_500m)
buffer_1km_dissolved <- st_union(buffer_1km)
buffer_3km_dissolved <- st_union(buffer_3km)

# And I convert them all to sf objects
buffer_500m_sf <- st_sf(geometry = buffer_500m_dissolved)
buffer_1km_sf <- st_sf(geometry = buffer_1km_dissolved)
buffer_3km_sf <- st_sf(geometry = buffer_3km_dissolved)

# And I calculate area of each buffer zone
buffer_500m_sf$area_sqkm <- as.numeric(st_area(buffer_500m_sf) / 1000000)
buffer_1km_sf$area_sqkm <- as.numeric(st_area(buffer_1km_sf) / 1000000)
buffer_3km_sf$area_sqkm <- as.numeric(st_area(buffer_3km_sf) / 1000000)

# And then to calculate the population within each buffer zone
# I use an area-weighted approach for tracts that are partially within buffers
population_500m <- st_interpolate_aw(
  ohio_tracts_sf["estimate"],
  buffer_500m_sf,
  extensive = TRUE
)
```

```
## Warning in st_interpolate_aw.sf(ohio_tracts_sf["estimate"], buffer_500m_sf, :
## st_interpolate_aw assumes attributes are constant or uniform over areas of x
```

```
population_1km <- st_interpolate_aw(
  ohio_tracts_sf["estimate"],
  buffer_1km_sf,
  extensive = TRUE
)
```

```
## Warning in st_interpolate_aw.sf(ohio_tracts_sf["estimate"], buffer_1km_sf, :
## st_interpolate_aw assumes attributes are constant or uniform over areas of x
```

```
population_3km <- st_interpolate_aw(
  ohio_tracts_sf["estimate"],
  buffer_3km_sf,
  extensive = TRUE
)
```

```
## Warning in st_interpolate_aw.sf(ohio_tracts_sf["estimate"], buffer_3km_sf, :
## st_interpolate_aw assumes attributes are constant or uniform over areas of x
```

```
# So I extract the population values
pop_500m <- sum(population_500m$estimate)
pop_1km <- sum(population_1km$estimate)
pop_3km <- sum(population_3km$estimate)
```

```r
# And get total Ohio population for comparison
total_ohio_pop <- sum(ohio_tracts_sf$estimate)

# And then I calculate population density in each buffer zone
pop_density_500m <- pop_500m / buffer_500m_sf$area_sqkm
pop_density_1km <- pop_1km / buffer_1km_sf$area_sqkm
pop_density_3km <- pop_3km / buffer_3km_sf$area_sqkm

# Summary table for report
population_exposure <- data.frame(
  Buffer_Distance = c("500m", "1km", "3km", "All of Ohio"),
  Area_SqKm = c(buffer_500m_sf$area_sqkm, buffer_1km_sf$area_sqkm, buffer_3km_sf$area_sqkm, sum(as.nume
  Population = c(pop_500m, pop_1km, pop_3km, total_ohio_pop),
  Population_Density = c(pop_density_500m, pop_density_1km, pop_density_3km, total_ohio_pop / sum(as.nu
  Percentage_of_Ohio_Pop = c(pop_500m / total_ohio_pop * 100, pop_1km / total_ohio_pop * 100, pop_3km /
)

# View results
print(population_exposure)
```

```
##   Buffer_Distance   Area_SqKm  Population Population_Density
## 1            500m    166.3962    87042.06           523.1012
## 2             1km    637.1316   331217.49           519.8572
## 3             3km   4432.1244  1969142.21           444.2886
## 4     All of Ohio 106870.1264 11655397.00           109.0613
##   Percentage_of_Ohio_Pop
## 1              0.7467962
## 2              2.8417521
## 3             16.8946816
## 4            100.0000000
```

```r
# Integrated Risk Assessment

# I am going to create a coarser grid for risk assessment (10km instead of 5km)
risk_grid_simple <- st_make_grid(ohio_boundary, cellsize = 10000, square = TRUE) %>%
  st_sf() %>%
  st_intersection(ohio_boundary) %>%
  mutate(grid_id = row_number())
```

```
## Warning: attribute variables are assumed to be spatially constant throughout
## all geometries
```

```r
# I count accidents in each grid cell
accident_counts <- st_intersects(risk_grid_simple, ohio_accidents_sf)
risk_grid_simple$accident_count <- lengths(accident_counts)
risk_grid_simple$area_sq_km <- as.numeric(st_area(risk_grid_simple) / 1000000)
risk_grid_simple$accident_density <- risk_grid_simple$accident_count / risk_grid_simple$area_sq_km

# I am printing summary of accident counts per grid cell to check
print(summary(risk_grid_simple$accident_count))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   0.000   0.192   0.000  12.000
```

```r
# Then I add population data to each grid cell
# And I use simpler interpolation that won't cause dimension mismatches
```

```r
risk_grid_simple$population <- 0

# Here I directly calculate population in each grid cell using point-in-polygon operation
# First I convert population data to points at centroids
ohio_tracts_points <- st_centroid(ohio_tracts_sf)
```

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

```r
ohio_tracts_points$population <- ohio_tracts_sf$estimate

# And I assign population to grid cells
for(i in 1:nrow(risk_grid_simple)) {
  # And I find tract centroids within this grid cell
  tracts_in_cell <- st_intersects(risk_grid_simple[i,], ohio_tracts_points)
  if(length(tracts_in_cell[[1]]) > 0) {
    # And I sum population from all tract centroids in this cell
    risk_grid_simple$population[i] <- sum(ohio_tracts_points$population[tracts_in_cell[[1]]])
  }
}

# So I can obtain and calculate population density
risk_grid_simple$pop_density <- risk_grid_simple$population / risk_grid_simple$area_sq_km
print(summary(risk_grid_simple$population))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0       0    3950    9942    7963  179036
```

```r
water_cells <- st_intersects(risk_grid_simple, nhd_waterbodies_proj)
flowline_cells <- st_intersects(risk_grid_simple, nhd_flowline_proj_clean)

# Then I mark cells that contain water features
risk_grid_simple$has_waterbody <- lengths(water_cells) > 0
risk_grid_simple$has_flowline <- lengths(flowline_cells) > 0
risk_grid_simple$has_water <- risk_grid_simple$has_waterbody | risk_grid_simple$has_flowline
risk_grid_simple$water_factor <- as.numeric(risk_grid_simple$has_water)

# And I print water feature presence
water_presence <- table(risk_grid_simple$has_water)
print("Grid cells containing water features:")
```

```
## [1] "Grid cells containing water features:"
```

```r
print(water_presence)
```

```
##
## FALSE  TRUE
##     7  1165
```

```r
# I do the normalization process for having risk factors (from 0 to 1 scale)
max_acc_density <- max(risk_grid_simple$accident_density, na.rm = TRUE)
risk_grid_simple$accident_norm <- risk_grid_simple$accident_density / max_acc_density

max_pop_density <- max(risk_grid_simple$pop_density, na.rm = TRUE)
risk_grid_simple$pop_norm <- risk_grid_simple$pop_density / max_pop_density

# Then I calculate integrated risk index
```

```r
risk_grid_simple$risk_index <- (0.5 * risk_grid_simple$accident_norm) +
                               (0.3 * risk_grid_simple$pop_norm) +
                               (0.2 * risk_grid_simple$water_factor)

# Finally here I categorize risk levels
risk_breaks <- quantile(risk_grid_simple$risk_index,
                        probs = c(0, 0.75, 0.9, 0.95, 0.975, 1),
                        na.rm = TRUE)
risk_grid_simple$risk_category <- cut(risk_grid_simple$risk_index,
                                      breaks = risk_breaks,
                                      labels = c("Very Low", "Low", "Moderate", "High", "Very High"),
                                      include.lowest = TRUE)

# So I can generate risk stats
risk_stats <- data.frame(
  Risk_Category = levels(risk_grid_simple$risk_category),
  Cell_Count = as.numeric(table(risk_grid_simple$risk_category)),
  Area_Sq_Km = tapply(risk_grid_simple$area_sq_km, risk_grid_simple$risk_category, sum),
  Area_Percent = tapply(risk_grid_simple$area_sq_km, risk_grid_simple$risk_category, sum) /
                 sum(risk_grid_simple$area_sq_km) * 100,
  Population = tapply(risk_grid_simple$population, risk_grid_simple$risk_category, sum),
  Population_Percent = tapply(risk_grid_simple$population, risk_grid_simple$risk_category, sum) /
                       sum(risk_grid_simple$population) * 100,
  Accident_Count = tapply(risk_grid_simple$accident_count, risk_grid_simple$risk_category, sum),
  Accident_Percent = tapply(risk_grid_simple$accident_count, risk_grid_simple$risk_category, sum) /
                     sum(risk_grid_simple$accident_count) * 100
)

# And print risk stats
print("Integrated Risk Assessment Statistics:")
```

```
## [1] "Integrated Risk Assessment Statistics:"
```

```r
print(risk_stats)
```

```
##           Risk_Category Cell_Count Area_Sq_Km Area_Percent Population
## Very Low       Very Low        879  80069.467    74.922216    2507961
## Low                Low        175  16047.146    15.015558    2989393
## Moderate       Moderate         59   5590.333     5.230960    2366269
## High               High         29   2424.777     2.268901    1170041
## Very High     Very High         30   2738.403     2.562365    2618784
##           Population_Percent Accident_Count Accident_Percent
## Very Low             21.52304              0          0.00000
## Low                  25.65463             39         17.33333
## Moderate             20.30705             37         16.44444
## High                 10.04116             39         17.33333
## Very High            22.47411            110         48.88889
```

```r
# I create a simplified grid
ohio_grid_simple <- st_make_grid(ohio_boundary, cellsize = 5000, square = TRUE) %>%
  st_sf() %>%
  st_intersection(ohio_boundary) %>%
  mutate(grid_id = row_number())
```

```
## Warning: attribute variables are assumed to be spatially constant throughout
```

```
## all geometries
```

```
# I first set up IDW interpolation so I can interpolate accident density rather than just presence
# First, I count accidents per 10km grid cell to get density values for interpolation
point_counts <- st_intersects(ohio_grid_simple, ohio_accidents_sf)
ohio_grid_simple$accident_count <- lengths(point_counts)
ohio_grid_simple$area_sqkm <- as.numeric(st_area(ohio_grid_simple)) / 1000000
ohio_grid_simple$accident_density <- ohio_grid_simple$accident_count / ohio_grid_simple$area_sqkm

# Then I create centroids of grid cells to use as base points for interpolation
ohio_grid_centroids <- st_centroid(ohio_grid_simple)
```

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

```
# And I create a regular grid for the IDW interpolation (prediction locations)
idw_grid <- st_make_grid(ohio_boundary, cellsize = 2500, square = TRUE) %>%
  st_sf() %>%
  st_intersection(ohio_boundary)
```

```
## Warning: attribute variables are assumed to be spatially constant throughout
## all geometries
```

```
# Then I convert to stars object for gstat
idw_grid_stars <- st_as_stars(idw_grid)

#The following Inverse Distance Weighting (IDW) interpolation
#is implemented using code modified from:
#Pebesma, E. (2022)
#gstat: Spatial and Spatio-Temporal Geostatistical Modelling, Prediction and Simulation
#https://cran.r-project.org/package=gstat
#Downloaded: April 18, 2025

# Finally performing IDW interpolation
library(gstat)
idw_formula <- ohio_grid_centroids$accident_density ~ 1
idw_model <- gstat::idw(formula = idw_formula,
                        locations = ohio_grid_centroids,
                        newdata = idw_grid_stars,
                        idp = 2)  # idp is the power parameter (2 is standard)
```

```
## [inverse distance weighted interpolation]
```

```
# And then I convert to sf for plotting
idw_sf <- st_as_sf(idw_model)

# I am trying to calculate stats for the IDW interpolation
idw_stats <- data.frame(
  Statistic = c("Minimum Density", "Maximum Density", "Mean Density", "Median Density"),
  Value = c(
    min(idw_sf$var1.pred, na.rm = TRUE),
    max(idw_sf$var1.pred, na.rm = TRUE),
    mean(idw_sf$var1.pred, na.rm = TRUE),
    median(idw_sf$var1.pred, na.rm = TRUE)
  )
)

# Printing the stats
```

```r
print(idw_stats)
```

```
##         Statistic        Value
## 1 Minimum Density 2.663133e-09
## 2 Maximum Density 2.181082e-01
## 3    Mean Density 2.054677e-03
## 4  Median Density 9.324340e-04
```

```r
# I am checking if I have product type information in my accident data
colnames(ohio_accidents_sf)
```

```
##  [1] "Date"              "RPTID"             "Name1"
##  [4] "Name2"             "State"             "X..of.Fatalities"
##  [7] "X..of.Injuries"    "X..Damages"        "Incident.Type"
## [10] "Commodity"         "Age..yrs."         "Recorded.Long.Lat"
## [13] "Narrative"         "geometry"          "Year"
## [16] "Decade"            "near_waterbody"    "near_flowline"
## [19] "water_proximity"   "sensitivity"
```

```r
# So I can examine the unique values in any column related to products/commodities
if("Commodity" %in% colnames(ohio_accidents_sf)) {
  unique_commodities <- unique(ohio_accidents_sf$Commodity)
  print("Unique commodity types:")
  print(unique_commodities)
  print(paste("Number of unique commodities:", length(unique_commodities)))

  # I am trying to count of accidents by commodity type
  commodity_counts <- table(ohio_accidents_sf$Commodity)
  print("Accident counts by commodity type:")
  print(sort(commodity_counts, decreasing = TRUE))
}
```

```
## [1] "Unique commodity types:"
## [1] "GASOLINE"           ""                   "NATURAL GAS LIQUID"
## [4] "OIL"                "JET FUEL"           "DIESEL"
## [7] "LIQUIFIED GASES"    "PROPANE"            "OTHER"
## [1] "Number of unique commodities: 9"
## [1] "Accident counts by commodity type:"
##
##                                  OIL           GASOLINE             DIESEL
##               143                 43                 19                  7
##            PROPANE    LIQUIFIED GASES NATURAL GAS LIQUID           JET FUEL
##                  5                  3                  3                  2
##              OTHER
##                  1
```

```r
# I am trying to create a summary data frame for commodity analysis
commodity_summary <- data.frame(
  Commodity = names(sort(table(ohio_accidents_sf$Commodity), decreasing = TRUE)),
  Count = as.numeric(sort(table(ohio_accidents_sf$Commodity), decreasing = TRUE)),
  Percentage = as.numeric(sort(table(ohio_accidents_sf$Commodity), decreasing = TRUE)) / nrow(ohio_accid
)

# I am trying to generalize the classification for better ease of analysis later and also for visualiza
commodity_summary$Category <- "Other"
commodity_summary$Category[commodity_summary$Commodity %in% c("OIL", "GASOLINE", "DIESEL", "JET FUEL")]
```

```r
commodity_summary$Category[commodity_summary$Commodity %in% c("NATURAL GAS LIQUID", "LIQUIFIED GASES", "
commodity_summary$Category[commodity_summary$Commodity == ""] <- "Unknown"

# I am add the commodity type to my accidents data for mapping
ohio_accidents_sf$commodity_category <- "Other"
ohio_accidents_sf$commodity_category[ohio_accidents_sf$Commodity %in% c("OIL", "GASOLINE", "DIESEL", "JE
ohio_accidents_sf$commodity_category[ohio_accidents_sf$Commodity %in% c("NATURAL GAS LIQUID", "LIQUIFIE
ohio_accidents_sf$commodity_category[ohio_accidents_sf$Commodity == ""] <- "Unknown"
```

```r
#The following Thiessen/Voronoi polygon implementation for service area analysis
#is partly modified and inspired from:
#Pebesma, E. (2022)
#sf: Simple Features for R
#https://r-spatial.github.io/sf/
#Downloaded: April 15, 2025

# I am trying to identify duplicate coordinates and adding a small jitter to make them unique
coords <- st_coordinates(ohio_accidents_sf)
dup_indices <- which(duplicated(coords) | duplicated(coords, fromLast = TRUE))

# I am creating a copy of the accidents data that I can modify and avoid any overwriting
ohio_accidents_jittered <- ohio_accidents_sf

# So what I do here is I add a very small random offset to duplicated points (1m)
# This is only for creating Thiessen polygons not for analysis
if(length(dup_indices) > 0) {
  jitter_coords <- coords[dup_indices,]
  jitter_coords[,1] <- jitter_coords[,1] + runif(length(dup_indices), -1, 1)
  jitter_coords[,2] <- jitter_coords[,2] + runif(length(dup_indices), -1, 1)

  # And then I replace the coordinates of duplicated points
  jittered_points <- st_as_sf(as.data.frame(jitter_coords),
                              coords = c("X", "Y"),
                              crs = st_crs(ohio_accidents_sf))

  # And also I replace the geometries of duplicated points
  st_geometry(ohio_accidents_jittered)[dup_indices] <- st_geometry(jittered_points)
}

# I am using the st_voronoi function with jittered points
thiessen_geom <- st_voronoi(st_union(ohio_accidents_jittered))
thiessen_sf <- st_collection_extract(thiessen_geom) %>%
  st_intersection(st_union(ohio_boundary))

# Then I convert to sf data frame to add unique IDs
thiessen_sf <- st_sf(geometry = thiessen_sf) %>%
  mutate(thiessen_id = row_number())

# And I find which accident point falls within which Thiessen polygon
point_polygon_match <- st_intersects(ohio_accidents_sf, thiessen_sf)

# Here again I am creating a data frame linking accidents to Thiessen polygons
accident_thiessen_map <- data.frame(
  accident_index = 1:nrow(ohio_accidents_sf),
```

```
  thiessen_index = sapply(point_polygon_match, function(x) ifelse(length(x) > 0, x[1], NA))
)

# And also I remove records that are without a matching polygon
accident_thiessen_map <- accident_thiessen_map[!is.na(accident_thiessen_map$thiessen_index),]

# Here I am calculating the area of each Thiessen polygon
thiessen_sf$area_sq_km <- as.numeric(st_area(thiessen_sf)) / 1000000

# And also the calculate basic statistics for the Thiessen polygons
thiessen_stats <- data.frame(
  Statistic = c("Number of Thiessen Polygons",
                "Minimum Area (sq km)",
                "Maximum Area (sq km)",
                "Mean Area (sq km)",
                "Median Area (sq km)",
                "Total Area (sq km)"),
  Value = c(
    nrow(thiessen_sf),
    min(thiessen_sf$area_sq_km),
    max(thiessen_sf$area_sq_km),
    mean(thiessen_sf$area_sq_km),
    median(thiessen_sf$area_sq_km),
    sum(thiessen_sf$area_sq_km)
  )
)

# Printing the statistics
print(thiessen_stats)
```

```
##                     Statistic        Value
## 1 Number of Thiessen Polygons 2.250000e+02
## 2        Minimum Area (sq km) 1.144331e-02
## 3        Maximum Area (sq km) 7.818476e+03
## 4           Mean Area (sq km) 4.749783e+02
## 5         Median Area (sq km) 2.292193e+02
## 6          Total Area (sq km) 1.068701e+05
```

```
# Now first I am trying to create a simpler risk area analysis without population
risk_threshold <- quantile(idw_sf$var1.pred, 0.90, na.rm = TRUE)
high_risk_areas <- idw_sf %>%
  dplyr::filter(var1.pred >= risk_threshold) %>%
  st_union() %>%
  st_sf()

mod_risk_threshold <- quantile(idw_sf$var1.pred, 0.70, na.rm = TRUE)
mod_risk_areas <- idw_sf %>%
  dplyr::filter(var1.pred >= mod_risk_threshold & var1.pred < risk_threshold) %>%
  st_union() %>%
  st_sf()

# Then I calculate areas
high_risk_area <- sum(as.numeric(st_area(high_risk_areas))) / 1000000  # sq km
mod_risk_area <- sum(as.numeric(st_area(mod_risk_areas))) / 1000000    # sq km
```

```r
ohio_area <- sum(as.numeric(st_area(ohio_boundary))) / 1000000          # sq km

# Then I compile the list of calculated risk areas
risk_area_summary <- data.frame(
  Risk_Zone = c("High Risk", "Moderate Risk", "All of Ohio"),
  Area_Sq_Km = c(high_risk_area, mod_risk_area, ohio_area),
  Area_Percent = c(high_risk_area / ohio_area * 100,
                   mod_risk_area / ohio_area * 100,
                   100)
)

print("Risk Zone Area Statistics:")
```

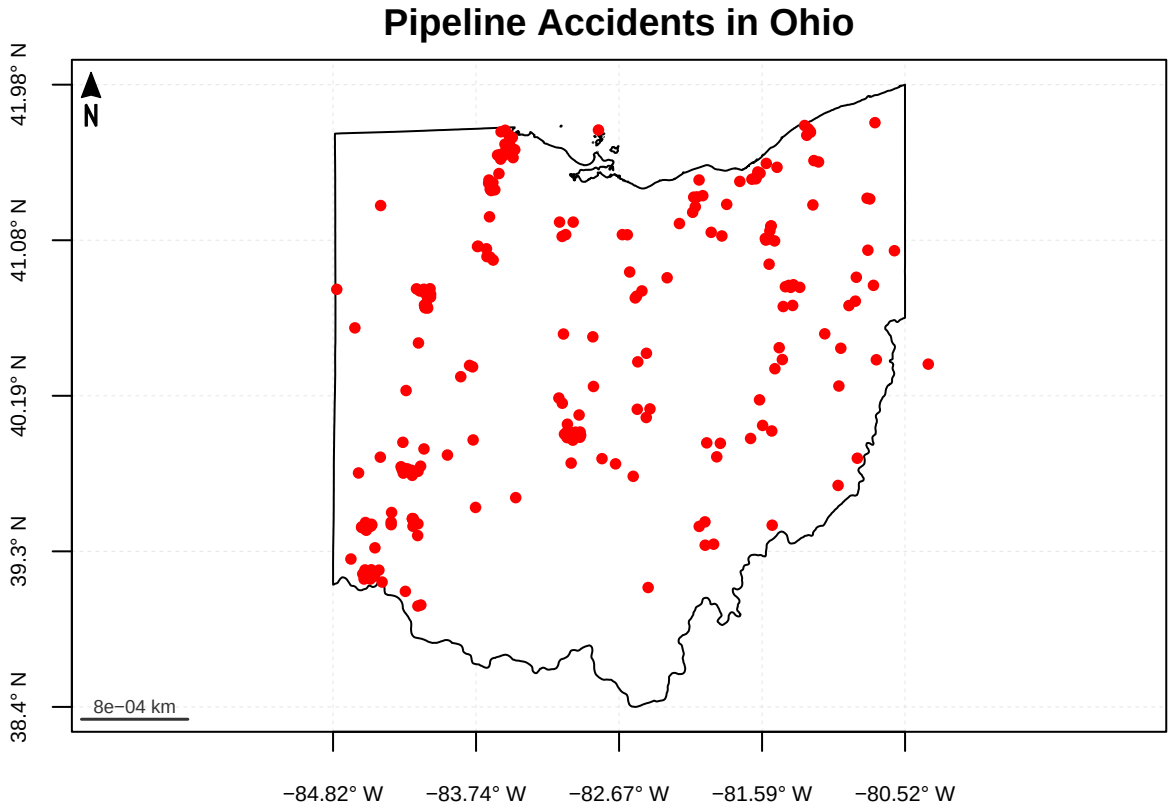
```
## [1] "Risk Zone Area Statistics:"
```

```r
print(risk_area_summary)
```

```
##       Risk_Zone Area_Sq_Km Area_Percent
## 1     High Risk   10776.84     10.08405
## 2 Moderate Risk   21524.80     20.14108
## 3   All of Ohio  106870.13    100.00000
```

```r
# # First, I'm transforming the data to a geographic CRS (WGS84)
# so I can display the coordinates in latitude and longitude for mapping.

ohio_boundary_geo <- st_transform(ohio_boundary, 4326)
ohio_accidents_geo <- st_transform(ohio_accidents_sf, 4326)

# then I set up the plotting area
par(mar = c(3, 3, 2, 1))

# and I create  plot with Ohio's extent just for reference first to check
plot(st_geometry(ohio_boundary_geo),
     col = NA,
     border = NA,
     axes = FALSE,
     main = "Pipeline Accidents in Ohio")

# then I get the bounding box for grid lines
bbox <- st_bbox(ohio_boundary_geo)

# and I create latitude and longitude breaks
lon_breaks <- seq(bbox["xmin"], bbox["xmax"], length.out = 5)
lat_breaks <- seq(bbox["ymin"], bbox["ymax"], length.out = 5)


lon_labels <- paste0(round(lon_breaks, 2), "° W")
lat_labels <- paste0(round(lat_breaks, 2), "° N")

for(x in lon_breaks) {
  abline(v = x, col = "grey90", lwd = 0.5, lty = 2)
}

for(y in lat_breaks) {
  abline(h = y, col = "grey90", lwd = 0.5, lty = 2)
```

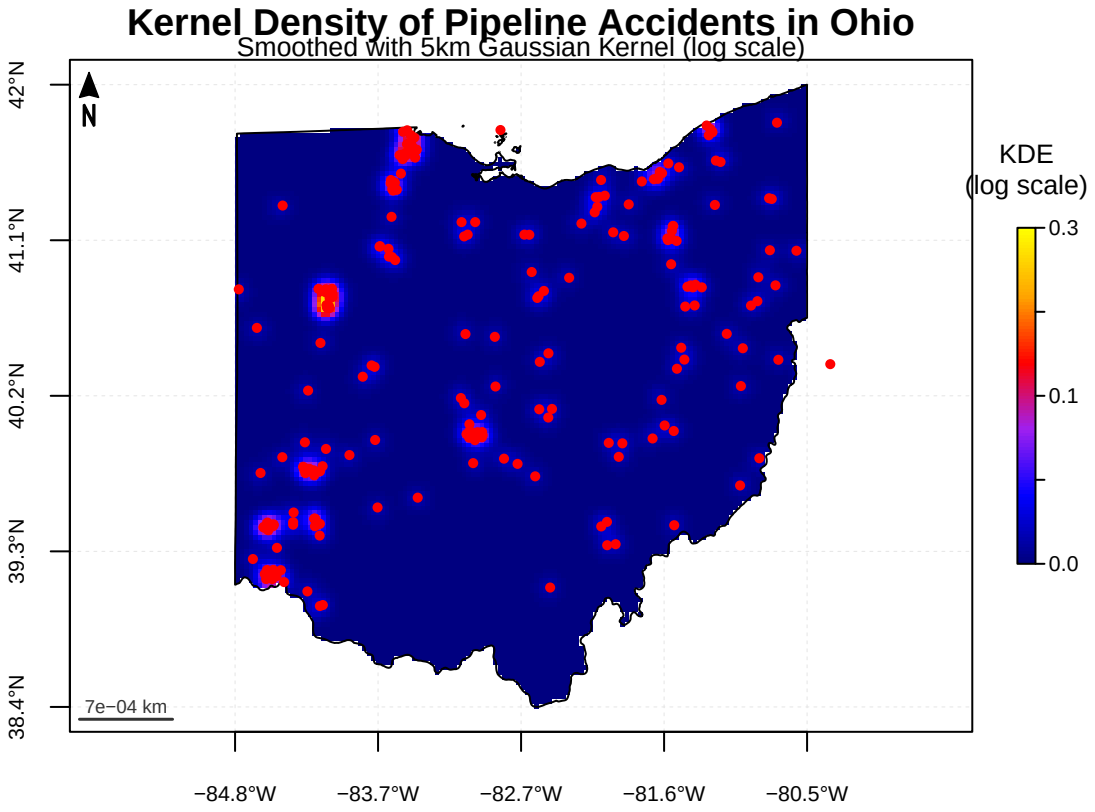
```
}

#  Adding Ohio boundary
plot(st_geometry(ohio_boundary_geo),
     col = "white",
     border = "black",
     add = TRUE)

# then accident points
plot(st_geometry(ohio_accidents_geo),
     col = "red",
     pch = 16,
     cex = 0.8,
     add = TRUE)

axis(1, at = lon_breaks, labels = lon_labels, cex.axis = 0.7)
axis(2, at = lat_breaks, labels = lat_labels, cex.axis = 0.7)

# north arrow
mf_arrow(pos = "topleft", col = "black")

# scale bar
mf_scale(
  pos = "bottomleft"
)

# box around the plot area
box()
```




```r
# Adding the colour ramp
density_cols <- colorRampPalette(
  c("navy", "blue", "purple", "red", "orange", "yellow")
)(100)

par(mar = c(3, 3, 2, 6))

# I first create empty plot with Ohio's extent
plot(st_geometry(ohio_bound_geo),
     col = NA,
     border = NA,
     axes = FALSE,
     main = "")

# I get the bounding box for grid lines
bbox <- st_bbox(ohio_bound_geo)

# Setting up latitude and longitude breaks
lon_breaks <- seq(bbox["xmin"], bbox["xmax"], length.out = 5)
lat_breaks <- seq(bbox["ymin"], bbox["ymax"], length.out = 5)

# and lat/long labels
lon_labels <- paste0(round(lon_breaks, 1), "°W")
lat_labels <- paste0(round(lat_breaks, 1), "°N")

# grid lines
for(x in lon_breaks) {
  abline(v = x, col = "grey90", lwd = 0.5, lty = 2)
```

```
}
for(y in lat_breaks) {
  abline(h = y, col = "grey90", lwd = 0.5, lty = 2)
}

# Filling Ohio with white as the base layer
plot(st_geometry(ohio_bound_geo),
     col = "white",
     border = NA,
     add = TRUE)

# Plotting the masked raster directly to avoid errors
plot(r_masked,
     col = density_cols,
     add = TRUE,
     legend = FALSE)

# then I add Ohio boundary again with a clean black line
plot(st_geometry(ohio_bound_geo),
     col = NA,
     border = "black",
     lwd = 1,
     add = TRUE)

# then accident points
plot(st_geometry(ohio_acc_geo),
     col = "red",
     pch = 16,
     cex = 0.7,
     add = TRUE)

# title and subtitle separately for clarity
title("Kernel Density of Pipeline Accidents in Ohio", cex.main = 1.2)
mtext("Smoothed with 5km Gaussian Kernel (log scale)",
      side = 3, line = -0.1, cex = 0.8)

# axes
axis(1, at = lon_breaks, labels = lon_labels, cex.axis = 0.7)
axis(2, at = lat_breaks, labels = lat_labels, cex.axis = 0.7)

# north arrow
mf_arrow(pos = "topleft", col = "black")

# scale bar
mf_scale(pos = "bottomleft")

# box around the plot area
box()

# Adding the tick style legend
par(xpd = TRUE)

# Adjusting legend position
```

```
usr <- par("usr")
legend_x <- usr[2] + 0.05 * (usr[2] - usr[1])
legend_y_bottom <- usr[3] + 0.25 * (usr[4] - usr[3])
legend_y_top <- usr[3] + 0.75 * (usr[4] - usr[3])
legend_width <- 0.02 * (usr[2] - usr[1])

# Adjusting legend color bar
y_positions <- seq(legend_y_bottom, legend_y_top, length.out = 101)
for (i in 1:100) {
  rect(legend_x, y_positions[i], legend_x + legend_width, y_positions[i+1],
       col = density_cols[i], border = NA)
}

# here I am adding  a border around the color bar
rect(legend_x, legend_y_bottom, legend_x + legend_width, legend_y_top,
     col = NULL, border = "black")

# Trying to add custom legend labels
tick_positions <- seq(legend_y_bottom, legend_y_top, length.out = 5)
tick_labels <- c("0.0", "", "0.1", "", "0.3")  # Values from the reference image

# tick marks and labels
segments(legend_x + legend_width, tick_positions,
         legend_x + legend_width + 0.01 * (usr[2] - usr[1]), tick_positions)
text(legend_x + legend_width + 0.015 * (usr[2] - usr[1]),
     tick_positions, tick_labels, adj = 0, cex = 0.7)

# legend title
text(legend_x + legend_width/2, legend_y_top + 0.05 * (usr[4] - usr[3]),
     "KDE\n(log scale)", adj = c(0.5, 0), cex = 0.8)
```

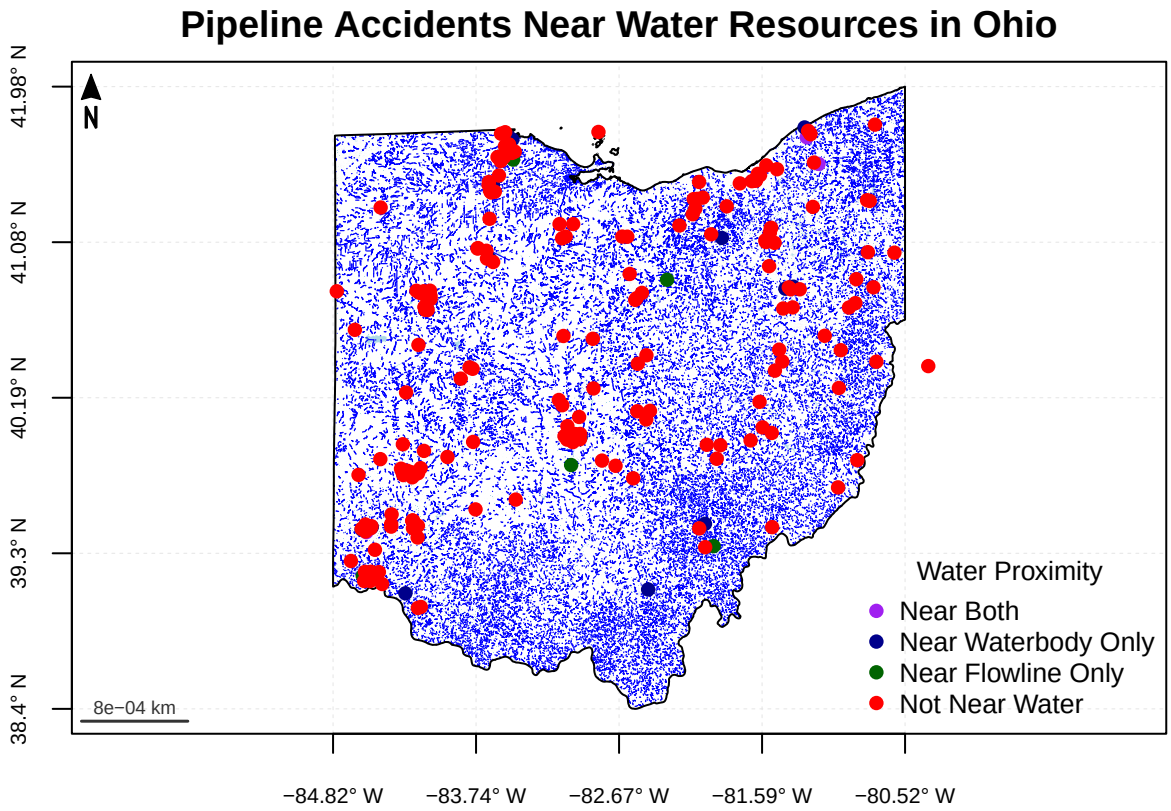



```r
par(xpd = FALSE)  # Reset to not plot outside the plot region
```

```r
# Water Proximity Map
# I am transforming water data to geographic CRS
nhd_waterbodies_ohio_geo <- st_transform(nhd_waterbodies_ohio, 4326)
nhd_flowline_ohio_geo <- st_transform(nhd_flowline_ohio, 4326)

# then I set up the plotting area
par(mar = c(3, 3, 2, 1))

# and I create empty plot with Ohio's extent first
plot(st_geometry(ohio_boundary_geo),
     col = NA,
     border = NA,
     axes = FALSE,
     main = "Pipeline Accidents Near Water Resources in Ohio")

# bounding box for grid lines
bbox <- st_bbox(ohio_boundary_geo)

# latitude and longitude breaks
lon_breaks <- seq(bbox["xmin"], bbox["xmax"], length.out = 5)
lat_breaks <- seq(bbox["ymin"], bbox["ymax"], length.out = 5)

# Format lat/long labels
lon_labels <- paste0(round(lon_breaks, 2), "° W")
lat_labels <- paste0(round(lat_breaks, 2), "° N")

# grid lines
```

```r
for(x in lon_breaks) {
  abline(v = x, col = "grey90", lwd = 0.5, lty = 2)
}
for(y in lat_breaks) {
  abline(h = y, col = "grey90", lwd = 0.5, lty = 2)
}

# Adding Ohio fill
plot(st_geometry(ohio_boundary_geo),
     col = "white",
     border = NA,
     add = TRUE)

# then adding  water features
plot(st_geometry(nhd_waterbodies_ohio_geo),
     col = "lightblue",
     border = NA,
     add = TRUE)

plot(st_geometry(nhd_flowline_ohio_geo),
     col = "blue",
     lwd = 0.6,
     add = TRUE)

# Addding  Ohio boundary
plot(st_geometry(ohio_boundary_geo),
     col = NA,
     border = "black",
     add = TRUE)

# and defining colors for water proximity categories
prox_colors <- c("Near Both" = "purple",
                 "Near Waterbody Only" = "darkblue",
                 "Near Flowline Only" = "darkgreen",
                 "Not Near Water" = "red")

# Plotting accident points colored by water proximity
# so first I  split by category and plot each separately
for(cat in names(prox_colors)) {
  cat_points <- ohio_accidents_geo[ohio_accidents_geo$water_proximity == cat, ]
  if(nrow(cat_points) > 0) {
    plot(st_geometry(cat_points),
         col = prox_colors[cat],
         pch = 16,
         cex = 1,
         add = TRUE)
  }
}

# and again Add axes with lat/long labels
axis(1, at = lon_breaks, labels = lon_labels, cex.axis = 0.7)
axis(2, at = lat_breaks, labels = lat_labels, cex.axis = 0.7)
```

```r
# north arrow
mf_arrow(pos = "topleft", col = "black")

# scale bar
mf_scale(pos = "bottomleft")

# box around the plot area
box()

# legend
legend("bottomright",
       legend = names(prox_colors),
       col = unname(prox_colors),
       pch = 16,
       pt.cex = 1,
       bty = "n",
       cex = 0.8,
       title = "Water Proximity")
```

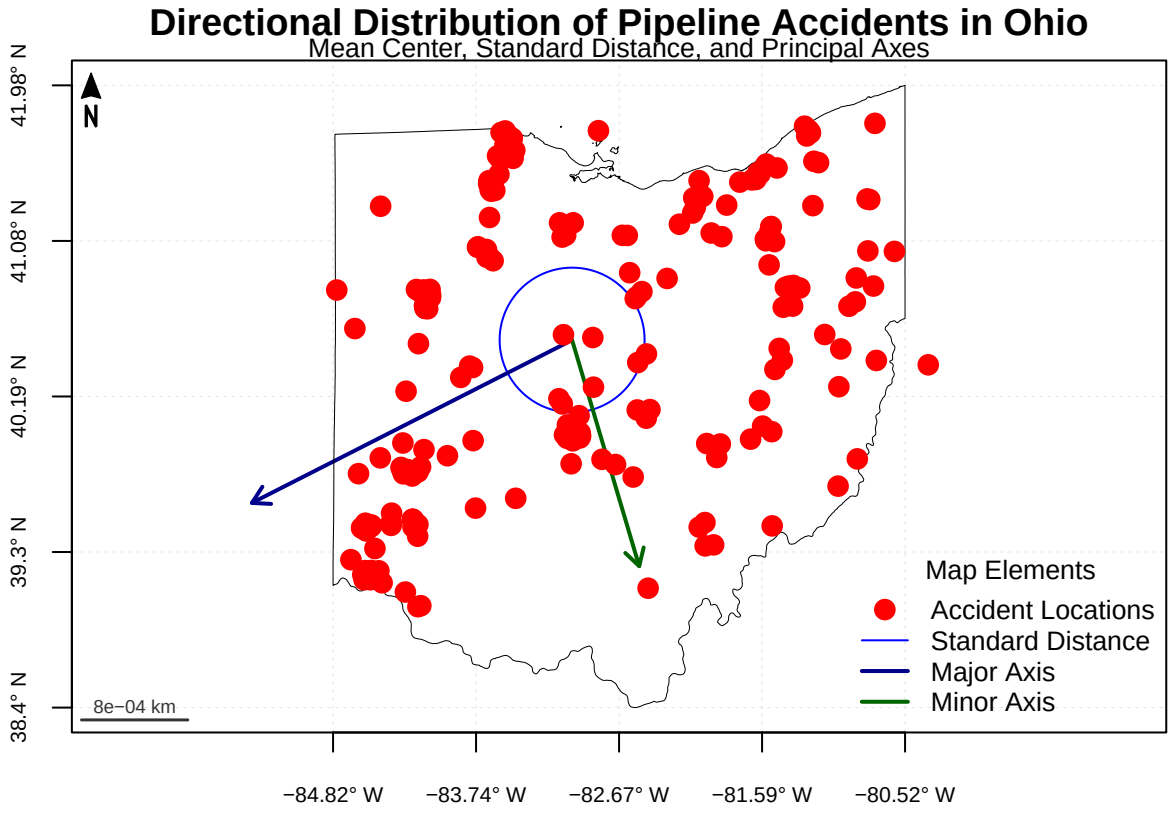



```r
# here I am transforming data to geographic CRS (WGS84) for lat/long display
ohio_boundary_geo <- st_transform(ohio_boundary, 4326)
ohio_accidents_geo <- st_transform(ohio_accidents_sf, 4326)
std_circle_geo <- st_transform(std_circle, 4326)

# I'm transforming the mean center and calculating the ellipse axes in geographic coordinates
# so the results align properly with my lat/long-based visualizations.

mean_center_geo <- st_coordinates(st_centroid(st_union(ohio_accidents_geo)))
```

```r
## I'm transforming the ellipse axes calculations into geographic space
# by extracting the original angle and distances from the UTM-based results.

coords_geo <- st_coordinates(ohio_accidents_geo)
cov_matrix_geo <- cov(coords_geo)
eigen_result_geo <- eigen(cov_matrix_geo)
eigen_vectors_geo <- eigen_result_geo$vectors
eigen_values_geo <- eigen_result_geo$values
angle_geo <- atan2(eigen_vectors_geo[2,1], eigen_vectors_geo[1,1])

# so here I calculate major and minor axes
major_axis_geo <- sqrt(eigen_values_geo[1]) * 2
minor_axis_geo <- sqrt(eigen_values_geo[2]) * 2

# Setting up the plotting area
par(mar = c(3, 3, 2, 1))

# again I create empty plot with Ohio's extent
plot(st_geometry(ohio_boundary_geo),
     col = NA,
     border = NA,
     axes = FALSE,
     main = "Directional Distribution of Pipeline Accidents in Ohio")

# and add the bounding box for grid lines
bbox <- st_bbox(ohio_boundary_geo)

# latitude and longitude breaks
lon_breaks <- seq(bbox["xmin"], bbox["xmax"], length.out = 5)
lat_breaks <- seq(bbox["ymin"], bbox["ymax"], length.out = 5)

# lat/long labels
lon_labels <- paste0(round(lon_breaks, 2), "° W")
lat_labels <- paste0(round(lat_breaks, 2), "° N")

# grid lines
for(x in lon_breaks) {
  abline(v = x, col = "grey90", lwd = 0.5, lty = 2)
}
for(y in lat_breaks) {
  abline(h = y, col = "grey90", lwd = 0.5, lty = 2)
}

# Adding Ohio fill
plot(st_geometry(ohio_boundary_geo),
     col = "white",
     border = NA,
     add = TRUE)

# Plotting  the standard distance circle
plot(st_geometry(std_circle_geo),
     col = NA,
     border = "blue",
```

```
     lwd = 1,
     add = TRUE)

# and adding the major axis arrow
arrows(
  mean_center_geo[1], mean_center_geo[2],
  mean_center_geo[1] + cos(angle_geo) * major_axis_geo,
  mean_center_geo[2] + sin(angle_geo) * major_axis_geo,
  col = "darkblue", lwd = 2, length = 0.10
)

# Adding the minor axis arrow
arrows(
  mean_center_geo[1], mean_center_geo[2],
  mean_center_geo[1] + cos(angle_geo + pi/2) * minor_axis_geo,
  mean_center_geo[2] + sin(angle_geo + pi/2) * minor_axis_geo,
  col = "darkgreen", lwd = 2, length = 0.10
)

# Ohio boundary
plot(st_geometry(ohio_boundary_geo),
     col = NA,
     border = "black",
     lwd = 0.5,
     add = TRUE)

# Putting back the accident points
plot(st_geometry(ohio_accidents_geo),
     col = "red",
     pch = 16,
     cex = 1.5,
     add = TRUE)

# subtitle
mtext("Mean Center, Standard Distance, and Principal Axes",
      side = 3, line = -0.1, cex = 0.8)

# axes with lat/long labels
axis(1, at = lon_breaks, labels = lon_labels, cex.axis = 0.7)
axis(2, at = lat_breaks, labels = lat_labels, cex.axis = 0.7)

# north arrow
mf_arrow(pos = "topleft", col = "black")

# scale bar
mf_scale(pos = "bottomleft")

# box around the plot area
box()

# legend
legend("bottomright",
       legend = c("Accident Locations", "Standard Distance", "Major Axis", "Minor Axis"),
```

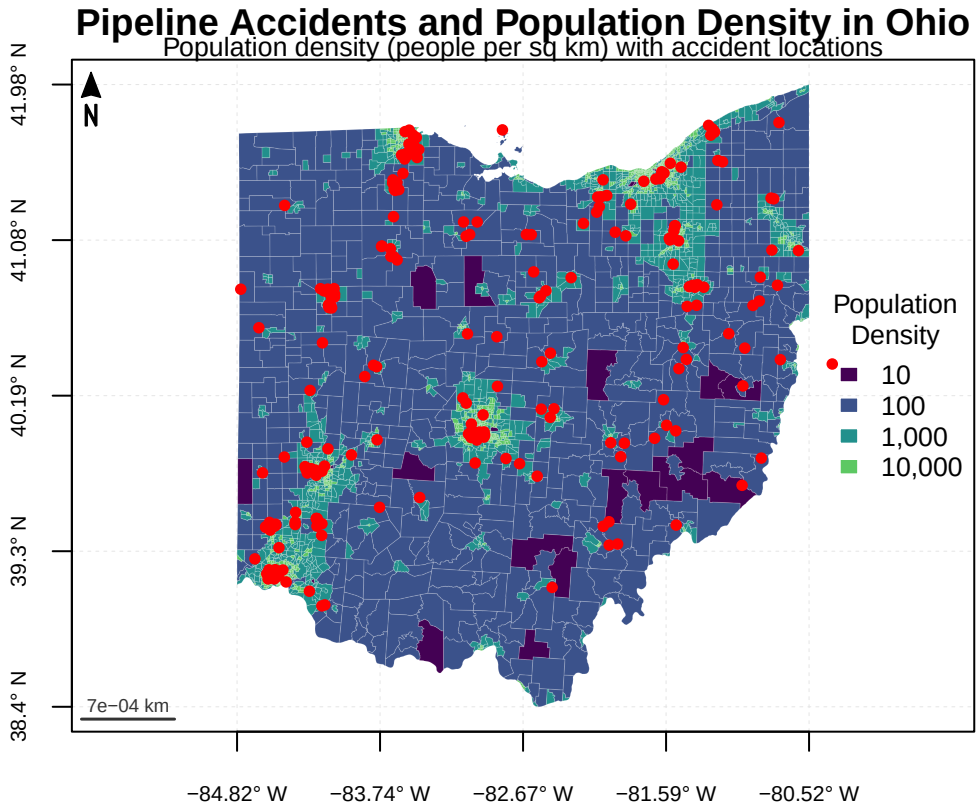
```
       col = c("red", "blue", "darkblue", "darkgreen"),
       pch = c(16, NA, NA, NA),
       lty = c(NA, 1, 1, 1),
       lwd = c(NA, 1, 2, 2),
       pt.cex = 1.5,
       bty = "n",
       cex = 0.8,
       title = "Map Elements")
```




```
# Transforming  the data to geographic CRS for display
ohio_tracts_geo <- st_transform(ohio_tracts_sf, 4326)
ohio_accidents_geo <- st_transform(ohio_accidents_sf, 4326)

# and then setting up the plotting area
par(mar = c(3, 3, 2, 6))

# I'm defining breaks and selecting a color palette for population density (on a log scale)
# to make patterns clearer and avoid visual distortion from extreme values.

pop_breaks <- c(0, 10, 100, 1000, 10000, max(ohio_tracts_geo$pop_density, na.rm = TRUE))
# Creating a color palette matching viridis
viridis_colors <- viridisLite::viridis(5)

plot(st_geometry(ohio_tracts_geo),
     col = NA,
     border = NA,
     axes = FALSE,
     main = "Pipeline Accidents and Population Density in Ohio")
```

```r
# Getting the bounding box for grid lines
bbox <- st_bbox(ohio_tracts_geo)

# adding the latitude and longitude breaks
lon_breaks <- seq(bbox["xmin"], bbox["xmax"], length.out = 5)
lat_breaks <- seq(bbox["ymin"], bbox["ymax"], length.out = 5)

# Formatting lat/long labels
lon_labels <- paste0(round(lon_breaks, 2), "° W")
lat_labels <- paste0(round(lat_breaks, 2), "° N")

# grid lines
for(x in lon_breaks) {
  abline(v = x, col = "grey90", lwd = 0.5, lty = 2)
}
for(y in lat_breaks) {
  abline(h = y, col = "grey90", lwd = 0.5, lty = 2)
}

# i try plotting  the population density choropleth
plot(st_geometry(ohio_tracts_geo),
     col = viridis_colors[findInterval(ohio_tracts_geo$pop_density, pop_breaks)],
     border = NA,
     add = TRUE)

# here i add tract boundaries
plot(st_geometry(ohio_tracts_geo),
     col = NA,
     border = "#FFFFFF20",
     lwd = 0.1,
     add = TRUE)

# accident points
plot(st_geometry(ohio_accidents_geo),
     col = "red",
     pch = 16,
     cex = 0.8,
     add = TRUE)

# subtitle
mtext("Population density (people per sq km) with accident locations",
      side = 3, line = -0.1, cex = 0.8)

axis(1, at = lon_breaks, labels = lon_labels, cex.axis = 0.7)
axis(2, at = lat_breaks, labels = lat_labels, cex.axis = 0.7)

# north arrow
mf_arrow(pos = "topleft", col = "black")

# scale bar
mf_scale(pos = "bottomleft")

box()
```

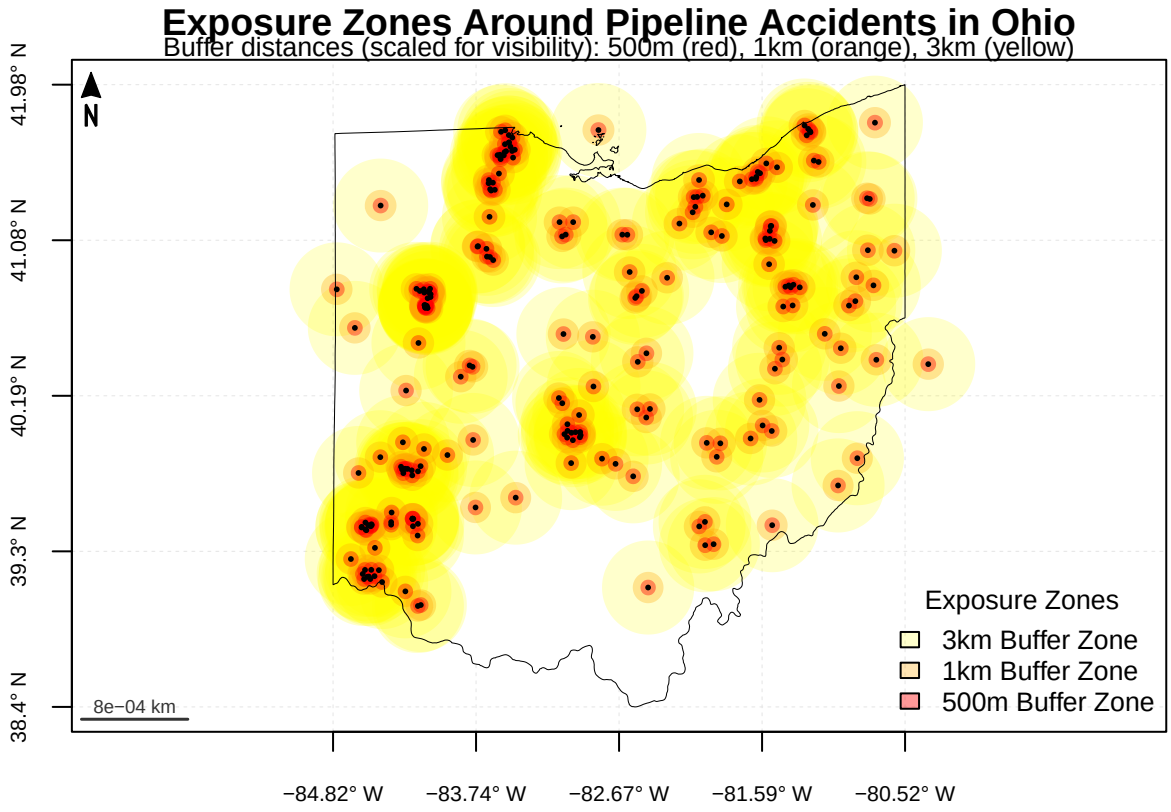
```
# legend
legend_title <- "Population\nDensity"
legend_labels <- c("10", "100", "1,000", "10,000")
legend("right",
       legend = legend_labels,
       fill = viridis_colors,
       border = "#FFFFFF20",
       bty = "n",
       cex = 0.8,
       title = legend_title)
```




```
# Create larger buffers for better visualization
# Scaling up the buffer distances while maintaining their relative proportions
buffer_500m <- st_buffer(ohio_accidents_sf, 5000)
buffer_1km <- st_buffer(ohio_accidents_sf, 10000)
buffer_3km <- st_buffer(ohio_accidents_sf, 30000)

# so I am transforming data to geographic CRS (WGS84) for lat/long display as above
ohio_boundary_geo <- st_transform(ohio_boundary, 4326)
ohio_accidents_geo <- st_transform(ohio_accidents_sf, 4326)
buffer_500m_geo <- st_transform(buffer_500m, 4326)
buffer_1km_geo <- st_transform(buffer_1km, 4326)
buffer_3km_geo <- st_transform(buffer_3km, 4326)

# Setting up the plotting area
par(mar = c(3, 3, 2, 1))
```

```r
# Adding empty plot with Ohio's extent
plot(st_geometry(ohio_boundary_geo),
     col = NA,
     border = NA,
     axes = FALSE,
     main = "Exposure Zones Around Pipeline Accidents in Ohio")

# Getting the bounding box for grid lines
bbox <- st_bbox(ohio_boundary_geo)

# latitude and longitude breaks
lon_breaks <- seq(bbox["xmin"], bbox["xmax"], length.out = 5)
lat_breaks <- seq(bbox["ymin"], bbox["ymax"], length.out = 5)

# lat/long labels
lon_labels <- paste0(round(lon_breaks, 2), "° W")
lat_labels <- paste0(round(lat_breaks, 2), "° N")

# grid lines
for(x in lon_breaks) {
  abline(v = x, col = "grey90", lwd = 0.5, lty = 2)
}
for(y in lat_breaks) {
  abline(h = y, col = "grey90", lwd = 0.5, lty = 2)
}

# Ohio
plot(st_geometry(ohio_boundary_geo),
     col = "white",
     border = NA,
     add = TRUE)

# Plotting buffer zones first (BEFORE accident points)
# 3km zone (yellow)
plot(st_geometry(buffer_3km_geo),
     col = rgb(1, 1, 0, 0.2),
     border = NA,
     add = TRUE)

# 1km zone (orange)
plot(st_geometry(buffer_1km_geo),
     col = rgb(1, 0.65, 0, 0.3),
     border = NA,
     add = TRUE)

# 500m zone (red)
plot(st_geometry(buffer_500m_geo),
     col = rgb(1, 0, 0, 0.4),
     border = NA,
     add = TRUE)

# I add  Ohio boundary to clip edges
plot(st_geometry(ohio_boundary_geo),
```

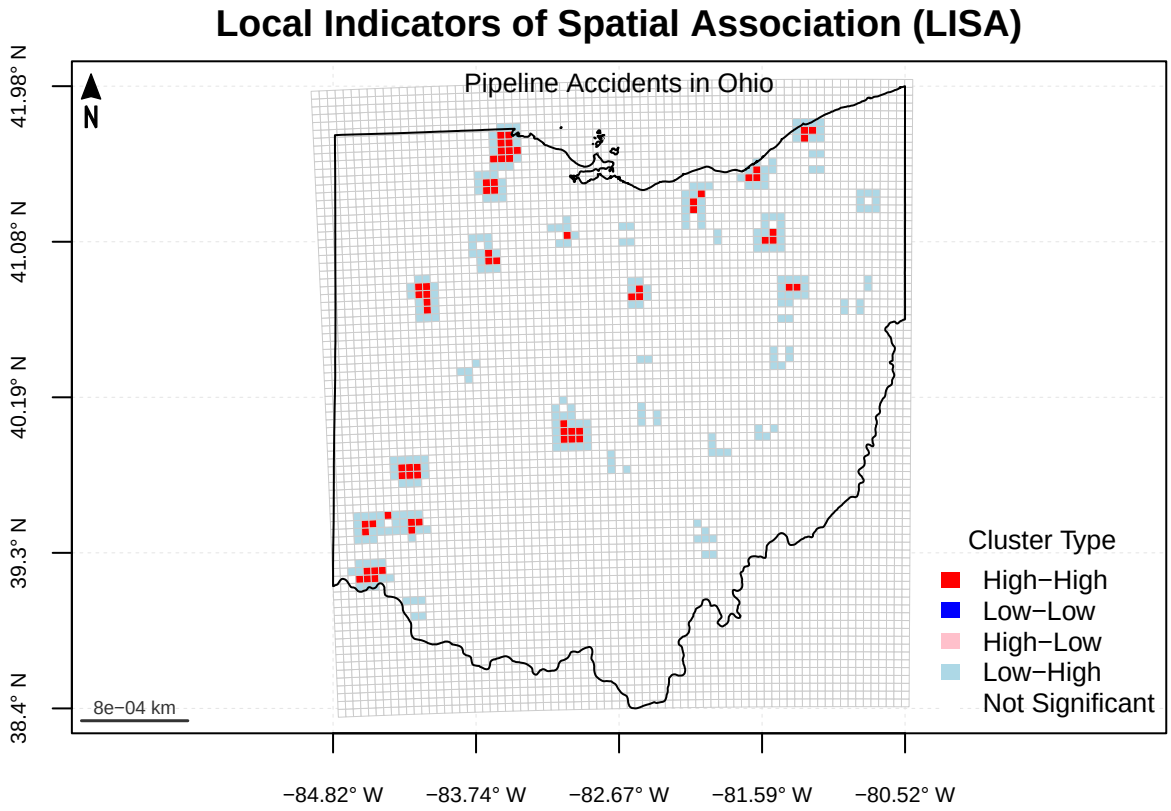
```
     col = NA,
     border = "black",
     lwd = 0.5,
     add = TRUE)

# AFTER buffer zones, I try to plot accident points
plot(st_geometry(ohio_accidents_geo),
     col = "black",
     pch = 16,
     cex = 0.4,
     add = TRUE)


mtext("Buffer distances (scaled for visibility): 500m (red), 1km (orange), 3km (yellow)",
      side = 3, line = -0.1, cex = 0.8)


axis(1, at = lon_breaks, labels = lon_labels, cex.axis = 0.7)
axis(2, at = lat_breaks, labels = lat_labels, cex.axis = 0.7)

# north arrow
mf_arrow(pos = "topleft", col = "black")

# scale bar
mf_scale(pos = "bottomleft")

# box around the plot area
box()

# legend
legend("bottomright",
       legend = c("3km Buffer Zone", "1km Buffer Zone", "500m Buffer Zone"),
       fill = c(rgb(1, 1, 0, 0.2), rgb(1, 0.65, 0, 0.3), rgb(1, 0, 0, 0.4)),
       border = "black",
       bty = "n",
       cex = 0.8,
       title = "Exposure Zones")
```




```
ohio_boundary_geo <- st_transform(ohio_boundary, 4326)
ohio_grid_geo <- st_transform(ohio_grid, 4326)

# I am deifining cluster type colors
cluster_colors <- c(
  "High-High" = "red",
  "Low-Low" = "blue",
  "High-Low" = "pink",
  "Low-High" = "lightblue",
  "Not Significant" = "white"
)

# then set up the plotting area
par(mar = c(3, 3, 2, 1))


plot(st_geometry(ohio_boundary_geo),
     col = NA,
     border = NA,
     axes = FALSE,
     main = "Local Indicators of Spatial Association (LISA)")

# bounding box for grid lines
bbox <- st_bbox(ohio_boundary_geo)


lon_breaks <- seq(bbox["xmin"], bbox["xmax"], length.out = 5)
lat_breaks <- seq(bbox["ymin"], bbox["ymax"], length.out = 5)
```

```r
lon_labels <- paste0(round(lon_breaks, 2), "° W")
lat_labels <- paste0(round(lat_breaks, 2), "° N")

# grid lines
for(x in lon_breaks) {
  abline(v = x, col = "grey90", lwd = 0.5, lty = 2)
}
for(y in lat_breaks) {
  abline(h = y, col = "grey90", lwd = 0.5, lty = 2)
}

# I'm creating a color vector based on cluster type
# so each cluster category stands out clearly in the final map.

grid_colors <- cluster_colors[as.character(ohio_grid_geo$cluster_type)]

# Plotting the grid cells
plot(st_geometry(ohio_grid_geo),
     col = grid_colors,
     border = "grey80",  # Add light grey borders to make grid cells visible
     lwd = 0.2,          # Thin line width for grid borders
     add = TRUE)

# Ohio boundary
plot(st_geometry(ohio_boundary_geo),
     col = NA,
     border = "black",
     lwd = 1,            # Thicker line for Ohio boundary
     add = TRUE)


axis(1, at = lon_breaks, labels = lon_labels, cex.axis = 0.7)
axis(2, at = lat_breaks, labels = lat_labels, cex.axis = 0.7)

# north arrow
mf_arrow(pos = "topleft", col = "black")

# scale bar
mf_scale(pos = "bottomleft")

# box around the plot area
box()

# subtitle
mtext("Pipeline Accidents in Ohio",
      side = 3,
      line = -1,
      cex = 0.8)

# legend
legend("bottomright",
       legend = names(cluster_colors),
```

```
       fill = cluster_colors,
       border = NA,
       bty = "n",
       cex = 0.8,
       title = "Cluster Type")
```



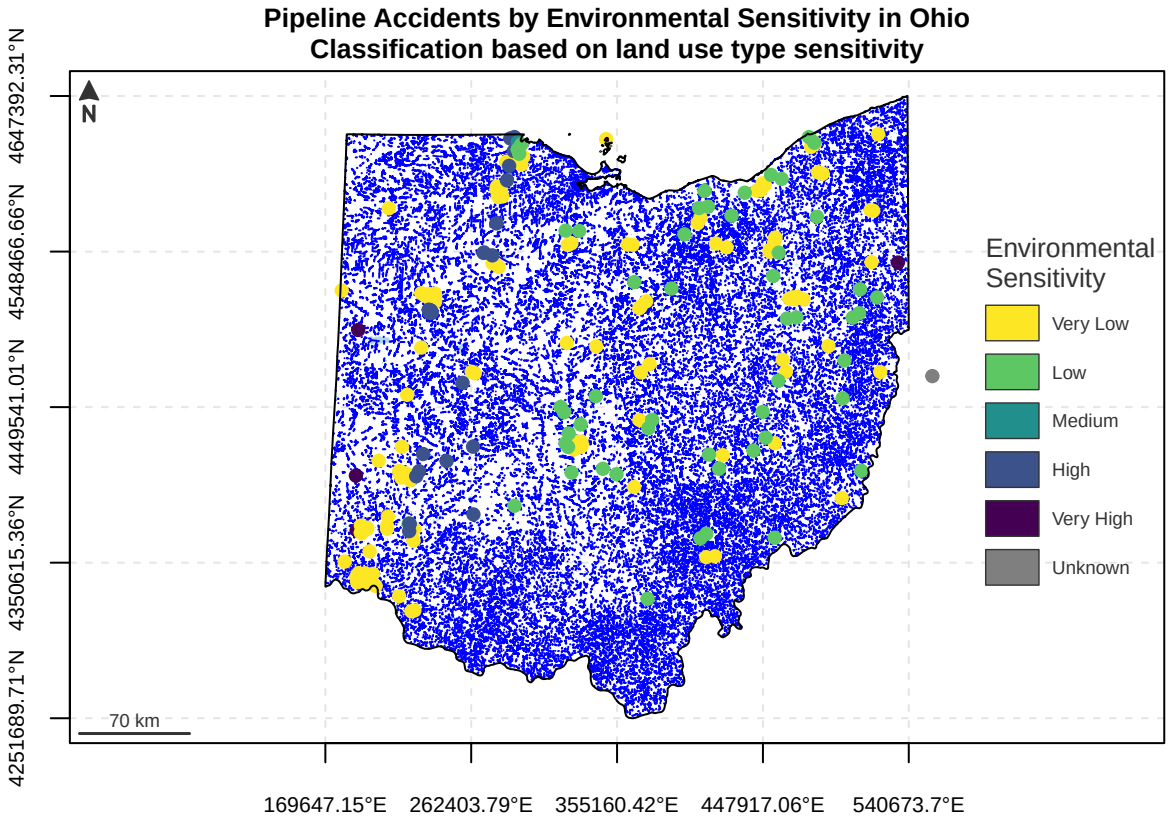

```
library(sf)
library(mapsf)
library(viridisLite)
library(dplyr)
library(forcats)

# I'm making sure my sf object includes the sensitivity factor,
#    and I'm converting any remaining NAs into "Unknown" to avoid issues during analysis or mapping.

ohio_accidents_sf <- ohio_accidents_sf %>%
  mutate(
    sensitivity = fct_explicit_na(
      sensitivity,
      na_level = "Unknown"
    )
  )
```

```
## Warning: There was 1 warning in `stopifnot()`.
## i In argument: `sensitivity = fct_explicit_na(sensitivity, na_level =
##   "Unknown")`.
## Caused by warning:
## ! `fct_explicit_na()` was deprecated in forcats 1.0.0.
```

```
## i Please use `fct_na_value_to_level()` instead.
```

```r
# Reprojecting everything to common CRS for spatial consistency
ohio_boundary_geo   <- st_transform(ohio_boundary,       26917)
nhd_waterbodies_geo <- st_transform(nhd_waterbodies_ohio,26917)
nhd_flowline_geo    <- st_transform(nhd_flowline_ohio,   26917)
ohio_accidents_geo  <- st_transform(ohio_accidents_sf,   26917)

# I'm creating a color palette that includes the "Unknown" category at the end,
# so it's visually accounted for and clearly distinguishable on the map.

base_levels <- c("Very Low","Low","Medium","High","Very High")
pal_vals <- viridis(5, option = "D", direction = -1)
palette_vals <- c(setNames(pal_vals, base_levels),
                  "Unknown" = "gray50")

# margins
par(mar = c(3, 3, 2, 1))

# Empty base map + title
plot(
  st_geometry(ohio_boundary_geo),
  col      = NA,
  border   = NA,
  axes     = FALSE,
  main     = paste0(
               "Pipeline Accidents by Environmental Sensitivity in Ohio\n",
               "Classification based on land use type sensitivity"
             ),
  cex.main = 0.8
)

# Gridlines & labels
bbox       <- st_bbox(ohio_boundary_geo)
lon_breaks <- seq(bbox["xmin"], bbox["xmax"], length.out = 5)
lat_breaks <- seq(bbox["ymin"], bbox["ymax"], length.out = 5)
lon_labels <- paste0(abs(round(lon_breaks,2)), "°", ifelse(lon_breaks<0,"W","E"))
lat_labels <- paste0(abs(round(lat_breaks,2)), "°", ifelse(lat_breaks<0,"S","N"))
for (x in lon_breaks) abline(v = x, col = "grey90", lty = 2)
for (y in lat_breaks) abline(h = y, col = "grey90", lty = 2)

# Drawing water & flowlines
mf_map(nhd_waterbodies_geo, type = "base", col = "lightblue", border = NA, add = TRUE)
mf_map(nhd_flowline_geo,    type = "base", col = "blue",      lwd    = 1,  add = TRUE)

# I'm plotting the accident points by typology,
# whilst also suppressing the default `mf_map` legend to keep the map cleaner.

mf_map(
  ohio_accidents_geo,
  type    = "typo",
  var     = "sensitivity",
  pal     = palette_vals,
  pch     = 19,
```

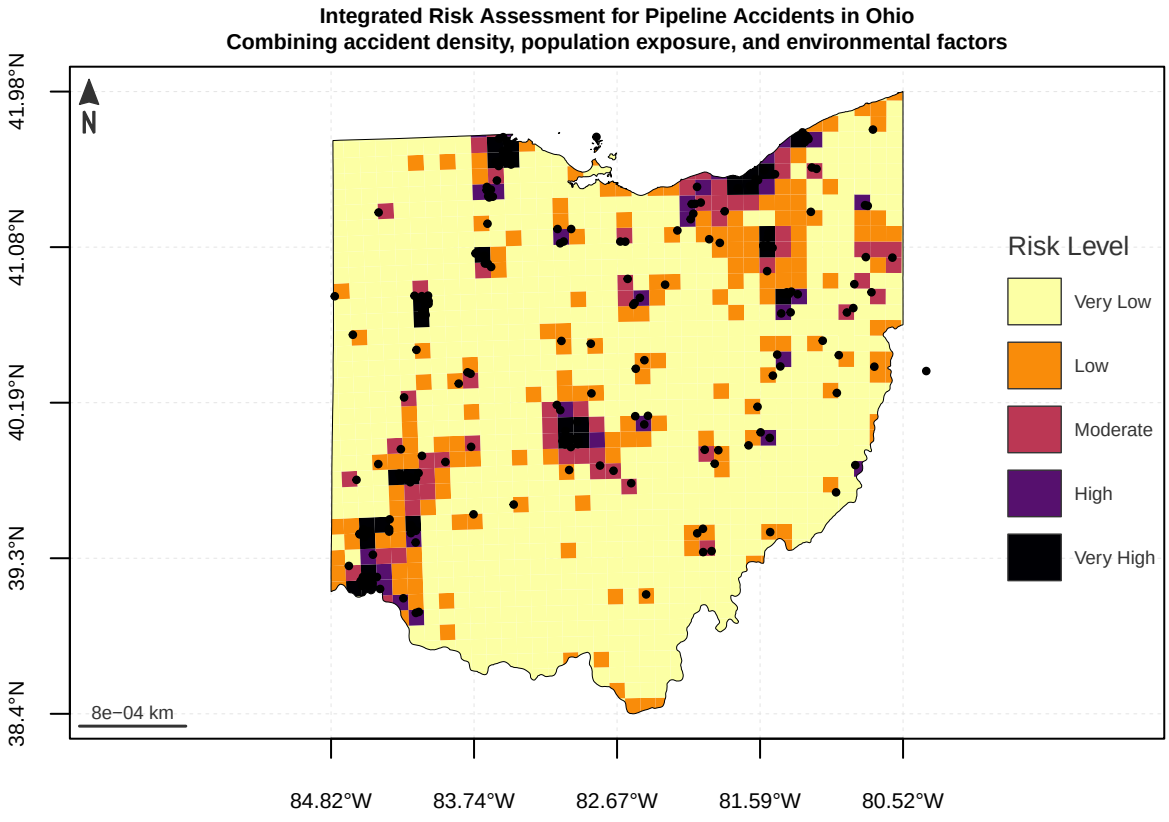
```
  cex     = 0.9,
  add     = TRUE,
  leg_pos = NA
)

# then  trying to draw a clean categorical legend with exactly 6 levels
mf_legend(
  type  = "typo",
  pos   = "right",
  val   = names(palette_vals),
  pal   = palette_vals,
  title = "Environmental\nSensitivity"
)

# I'm overlaying the state border to provide geographic context and frame the accident data on the map.

mf_map(
  ohio_boundary_geo,
  type   = "base",
  col    = NA,
  border = "black",
  lwd    = 1,
  add    = TRUE
)

# Axes, arrow, scale, box
axis(1, at = lon_breaks, labels = lon_labels, cex.axis = 0.7)
axis(2, at = lat_breaks, labels = lat_labels, cex.axis = 0.7)
mf_arrow(pos = "topleft")
mf_scale(pos = "bottomleft")
box()
```




```r
#- Loading all essential  libraries
library(sf)
library(mapsf)
library(viridisLite)

# I'm reprojecting all layers to a geographic CRS (WGS84)
# so the map displays with clean latitude and longitude axes.

#The following mapping techniques are implemented using code is partly  modified and inspired from:
#Giraud, T. (2022)
#mapsf: Thematic Cartography
#https://riatelab.github.io/mapsf/
#Downloaded: April 17, 2025

risk_grid_geo      <- st_transform(risk_grid_simple, 4326)
ohio_boundary_geo  <- st_transform(ohio_boundary,     4326)
ohio_accidents_geo <- st_transform(ohio_accidents_sf, 4326)

# I'm making sure `risk_category` is a factor with the correct order
# so the map legend and color scheme follow a meaningful, consistent sequence.

risk_grid_geo$risk_category <- factor(
  risk_grid_geo$risk_category,
  levels = c("Very Low","Low","Moderate","High","Very High")
)

# Adding Inferno palette matching the levels
pal_vals <- viridisLite::inferno(length(levels(risk_grid_geo$risk_category)), direction = -1)
```

```r
names(pal_vals) <- levels(risk_grid_geo$risk_category)

# Margins
par(mar = c(3, 3, 2, 1))

# Empty base plot with title + subtitle
plot(
  st_geometry(ohio_boundary_geo),
  col      = NA,
  border   = NA,
  axes     = FALSE,
  main     = paste0(
               "Integrated Risk Assessment for Pipeline Accidents in Ohio\n",
               "Combining accident density, population exposure, and environmental factors"
             ),
  cex.main = 0.7
)

#  Gridlines & lat/long labels
bbox       <- st_bbox(ohio_boundary_geo)
lon_breaks <- seq(bbox["xmin"], bbox["xmax"], length.out = 5)
lat_breaks <- seq(bbox["ymin"], bbox["ymax"], length.out = 5)
lon_labels <- paste0(abs(round(lon_breaks, 2)), "°", ifelse(lon_breaks < 0, "W", "E"))
lat_labels <- paste0(abs(round(lat_breaks, 2)), "°", ifelse(lat_breaks < 0, "S", "N"))

for (x in lon_breaks) abline(v = x, col = "grey90", lwd = 0.5, lty = 2)
for (y in lat_breaks) abline(h = y, col = "grey90", lwd = 0.5, lty = 2)

# Choropleth of risk categories
mf_map(
  x        = risk_grid_geo,
  type     = "typo",
  var      = "risk_category",
  pal      = pal_vals,
  border   = NA,
  add      = TRUE,
  leg_pos  = NA
)

# I have added a Custom legend showing only the five risk levels
mf_legend(
  type  = "typo",
  pos   = "right",
  val   = levels(risk_grid_geo$risk_category),
  pal   = pal_vals,
  title = "Risk Level"
)

# Overlaying ohio's boundary
mf_map(
  x      = ohio_boundary_geo,
  type   = "base",
  col    = NA,
```

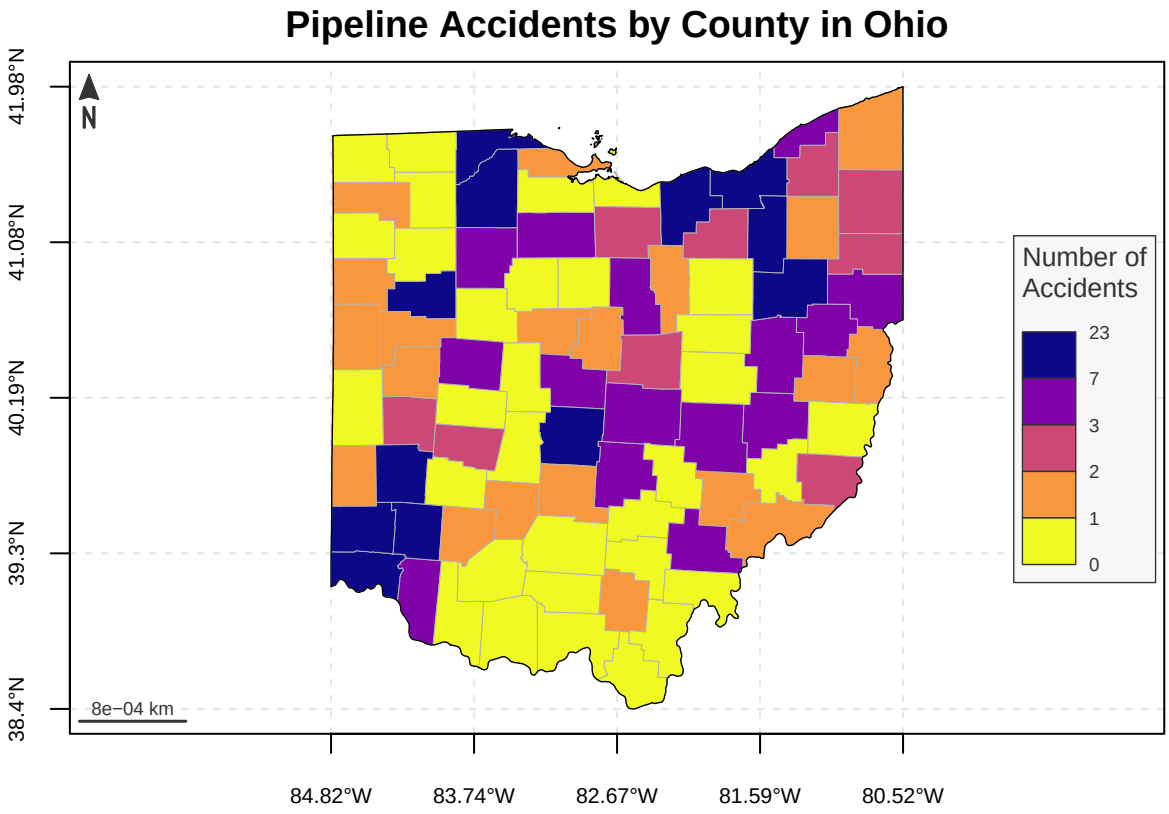
```
  border = "black",
  lwd    = 0.5,
  add    = TRUE
)

# then the Accident points
mf_map(
  x      = ohio_accidents_geo,
  type   = "base",
  col    = "black",
  pch    = 19,
  cex    = 0.5,
  add    = TRUE
)

# Axes, north arrow, scale bar, and frame box
axis(1, at = lon_breaks, labels = lon_labels, cex.axis = 0.7)
axis(2, at = lat_breaks, labels = lat_labels, cex.axis = 0.7)
mf_arrow(pos = "topleft")
mf_scale(pos = "bottomleft")
box()
```




```
library(sf)
library(mapsf)
library(viridisLite)


ohio_counties_geo <- st_transform(ohio_counties_sf, 4326)
```

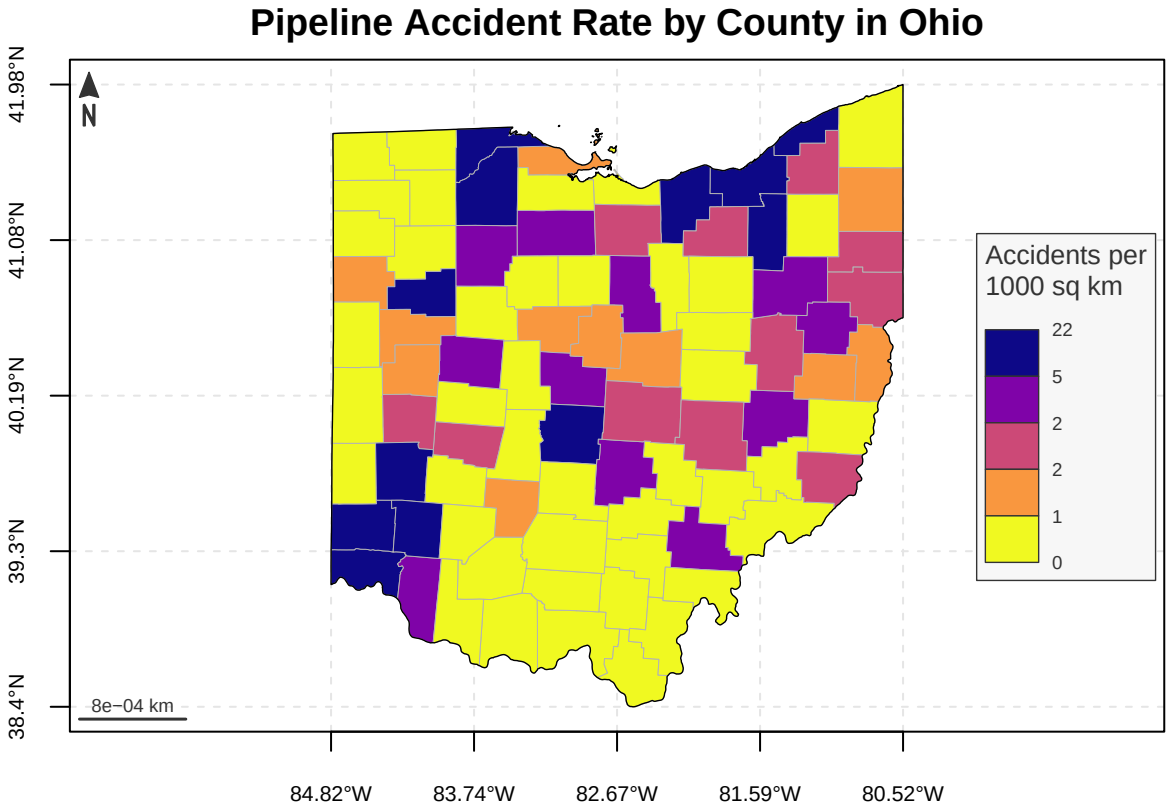
```r
ohio_boundary_geo <- st_transform(ohio_boundary, 4326)


bbox    <- st_bbox(ohio_boundary_geo)
lon_br  <- seq(bbox["xmin"], bbox["xmax"], length.out = 5)
lat_br  <- seq(bbox["ymin"], bbox["ymax"], length.out = 5)
lon_lb  <- paste0(abs(round(lon_br,2)), "°", ifelse(lon_br<0,"W","E"))
lat_lb  <- paste0(abs(round(lat_br,2)), "°", ifelse(lat_br<0,"N","N"))

# I've used the same Plasma palette
pal100 <- viridisLite::plasma(100, direction = -1)


draw_base <- function(title) {
  par(mar = c(3,3,2,1))
  plot(
    st_geometry(ohio_boundary_geo),
    col    = NA,
    border = NA,
    axes   = FALSE,
    main   = title,
    cex.main = 1.2
  )
  for(x in lon_br) abline(v = x, col = "grey90", lty = 2)
  for(y in lat_br) abline(h = y, col = "grey90", lty = 2)
}

# overlay boundary, axes, arrow, scale & box
add_axes <- function() {
  mf_map(
    x      = ohio_boundary_geo,
    type   = "base",
    col    = NA,
    border = "black",
    add    = TRUE
  )
  axis(1, at = lon_br, labels = lon_lb, cex.axis = 0.7)
  axis(2, at = lat_br, labels = lat_lb, cex.axis = 0.7)
  mf_arrow(pos = "topleft")
  mf_scale(pos = "bottomleft")
  box()
}

# adding Accident counts
draw_base("Pipeline Accidents by County in Ohio")

# here i try to create quantile breaks for accident counts
count_values <- ohio_counties_geo$accident_count
# For quantiles, ive tried using the stats quantile function
count_breaks <- c(
  0,  # start at 0
  stats::quantile(
    count_values[count_values > 0],  # making it skip zeros for better breaks
```

```
    probs = seq(0.2, 0.8, by = 0.2)
  ),
  max(count_values)
)

#  a custom color palette with 7 colors (for 6 intervals)
count_pal <- pal100[seq(1, 100, length.out = length(count_breaks) - 1)]

mf_map(
  x        = ohio_counties_geo,
  type     = "choro",
  var      = "accident_count",
  breaks   = count_breaks,
  pal      = count_pal,
  border   = "grey70",
  lwd      = 0.4,
  add      = TRUE,
  leg_pos  = NA
)

#  legend
mf_legend(
  type   = "choro",
  pos    = "right",
  title  = "Number of\nAccidents",
  val    = count_breaks,
  pal    = count_pal,
  frame  = TRUE
)
add_axes()
```



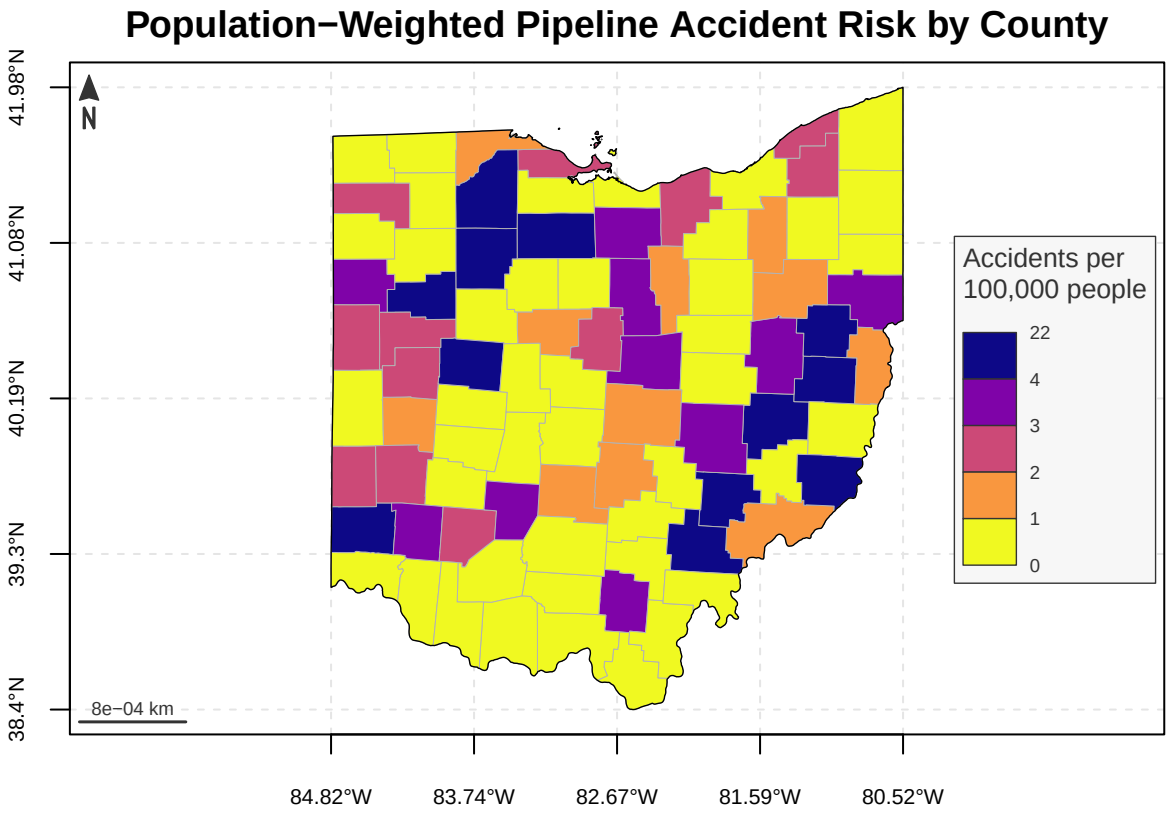

```
# # I'm calculating the accident rate per 1,000 km²
# to standardize risk across areas of different sizes and make spatial comparisons fair.

draw_base("Pipeline Accident Rate by County in Ohio")

# quantile breaks for accident rate
rate_values <- ohio_counties_geo$accident_rate
rate_breaks <- c(
  0,  # start at 0
  stats::quantile(
    rate_values[rate_values > 0],
    probs = seq(0.2, 0.8, by = 0.2)
  ),
  max(rate_values)
)

# I'm creating a custom color palette to better match the visual tone of the map and highlight key patt

rate_pal <- pal100[seq(1, 100, length.out = length(rate_breaks) - 1)]

mf_map(
  x         = ohio_counties_geo,
  type      = "choro",
  var       = "accident_rate",
  breaks    = rate_breaks,
  pal       = rate_pal,
  border    = "grey70",
  lwd       = 0.4,
```

```
  add       = TRUE,
  leg_pos   = NA
)

mf_legend(
  type   = "choro",
  pos    = "right",
  title  = "Accidents per\n1000 sq km",
  val    = rate_breaks,
  pal    = rate_pal,
  frame  = TRUE
)
add_axes()
```




```
# Population-weighted risk per 100,000 people --
draw_base("Population-Weighted Pipeline Accident Risk by County")

# im creating quantile breaks for population risk
risk_values <- ohio_counties_geo$pop_risk
risk_breaks <- c(
  0,  # start at 0
  stats::quantile(
    risk_values[risk_values > 0],
    probs = seq(0.2, 0.8, by = 0.2)
  ),
  max(risk_values)
)
```

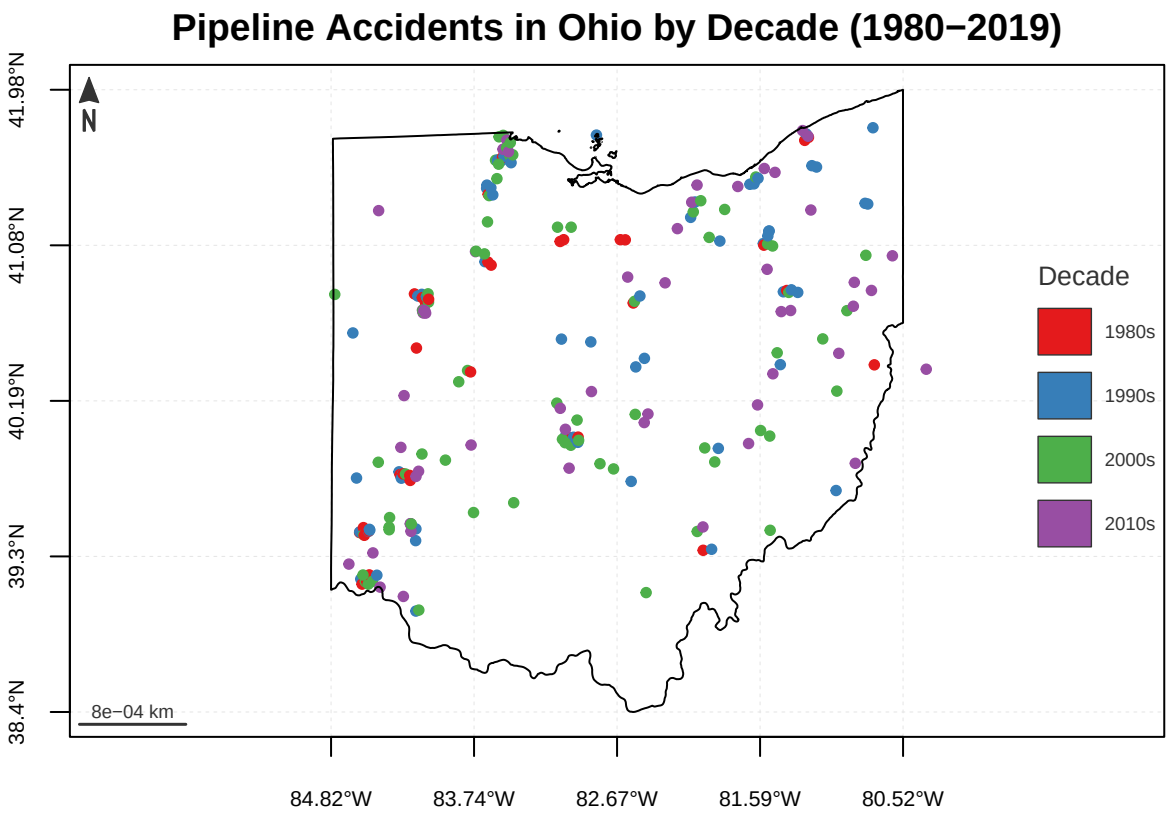
```
risk_pal <- pal100[seq(1, 100, length.out = length(risk_breaks) - 1)]

mf_map(
  x        = ohio_counties_geo,
  type     = "choro",
  var      = "pop_risk",
  breaks   = risk_breaks,
  pal      = risk_pal,
  border   = "grey70",
  lwd      = 0.4,
  add      = TRUE,
  leg_pos  = NA
)

mf_legend(
  type   = "choro",
  pos    = "right",
  title  = "Accidents per\n100,000 people",
  val    = risk_breaks,
  pal    = risk_pal,
  frame  = TRUE
)
add_axes()
```




```
library(sf)
library(mapsf)
library(RColorBrewer)
```

```r
ohio_boundary_geo   <- st_transform(ohio_boundary,      4326)
ohio_accidents_geo  <- st_transform(ohio_accidents_sf,  4326)

# I'm creating a factor variable for Decade with readable labels
# so the temporal trends are easier to interpret in plots and summaries.

ohio_accidents_geo$Decade_fac <- factor(
  ohio_accidents_geo$Decade,
  levels = c(1980, 1990, 2000, 2010),
  labels = c("1980s", "1990s", "2000s", "2010s")
)

# I'm using a palette from ColorBrewer's Set1 to ensure the categories are distinct and colorblind-frie

pal_set1 <- brewer.pal(n = 4, name = "Set1")
names(pal_set1) <- levels(ohio_accidents_geo$Decade_fac)

# margins
par(mar = c(3, 3, 2, 1))

# Empty base plot with title
plot(
  st_geometry(ohio_boundary_geo),
  col      = NA,
  border   = NA,
  axes     = FALSE,
  main     = "Pipeline Accidents in Ohio by Decade (1980-2019)",
  cex.main = 1.2
)

# Gridlines & lat/long labels
bbox       <- st_bbox(ohio_boundary_geo)
lon_breaks <- seq(bbox["xmin"], bbox["xmax"], length.out = 5)
lat_breaks <- seq(bbox["ymin"], bbox["ymax"], length.out = 5)
lon_labels <- paste0(abs(round(lon_breaks,2)), "°", ifelse(lon_breaks<0,"W","E"))
lat_labels <- paste0(abs(round(lat_breaks,2)), "°", ifelse(lat_breaks<0,"S","N"))

for(x in lon_breaks) abline(v = x, col = "grey90", lwd = 0.5, lty = 2)
for(y in lat_breaks) abline(h = y, col = "grey90", lwd = 0.5, lty = 2)

# i am Plotting all accident points by decade
mf_map(
  x        = ohio_accidents_geo,
  type     = "typo",
  var      = "Decade_fac",
  pal      = pal_set1,
  pch      = 19,
  cex      = 0.7,
  add      = TRUE,
  leg_pos  = NA
)

# legend
```

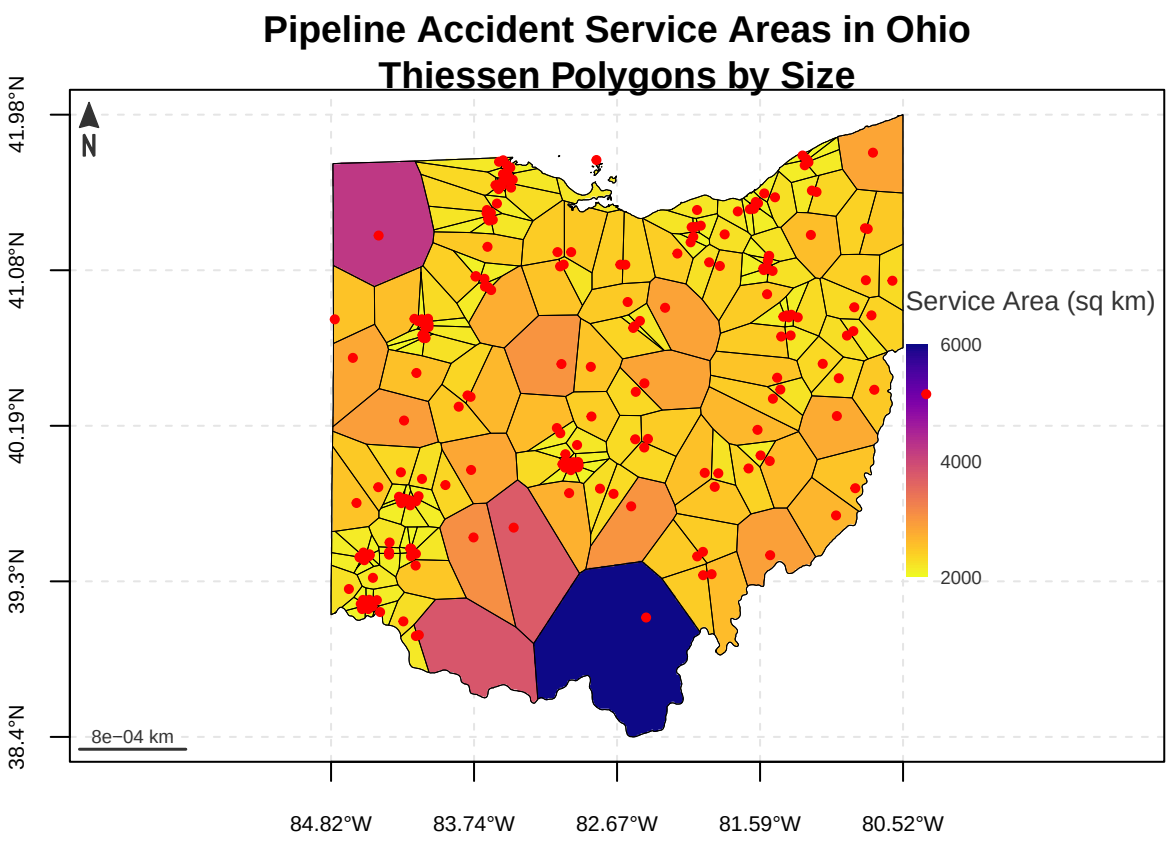
```r
mf_legend(
  type  = "typo",
  pos   = "right",
  val   = names(pal_set1),
  pal   = pal_set1,
  title = "Decade"
)

# Overlaying Ohio's boundary
mf_map(
  x      = ohio_boundary_geo,
  type   = "base",
  col    = NA,
  border = "black",
  lwd    = 1,
  add    = TRUE
)

# Axes, north arrow, scale bar, and frame
axis(1, at = lon_breaks, labels = lon_labels, cex.axis = 0.7)
axis(2, at = lat_breaks, labels = lat_labels, cex.axis = 0.7)
mf_arrow(pos = "topleft")
mf_scale(pos = "bottomleft")
box()
```



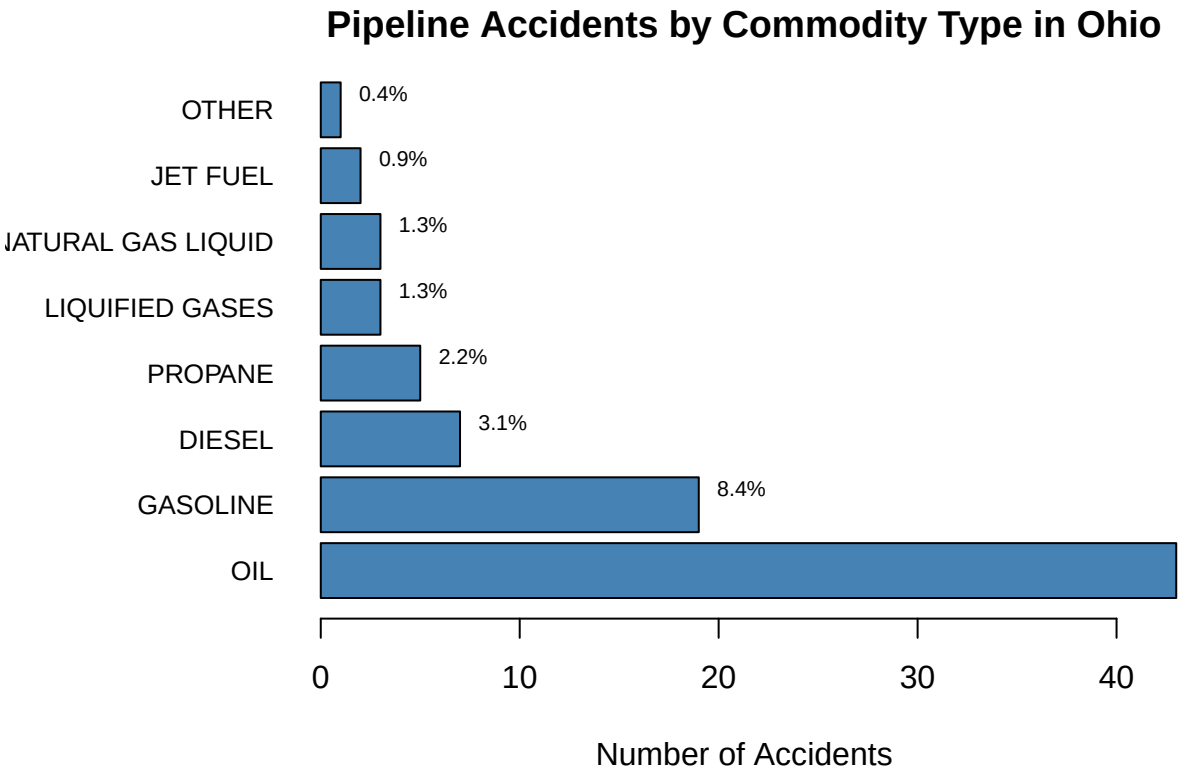

```r
library(sf)
library(mapsf)
library(terra)
```

```r
library(viridisLite)

# Reprojecting for spatial consistency
thiessen_geo       <- st_transform(thiessen_sf, 4326)
ohio_boundary_geo  <- st_transform(ohio_boundary,          4326)
ohio_accidents_geo <- st_transform(ohio_accidents_sf,      4326)

# I'm building a template raster over Ohio at a higher resolution (0.01°)
# to improve the spatial detail and overall quality of the analysis and maps.

bbox     <- st_bbox(ohio_boundary_geo)
template <- rast(
  xmin       = bbox["xmin"], xmax = bbox["xmax"],
  ymin       = bbox["ymin"], ymax = bbox["ymax"],
  resolution = 0.01,
  crs        = "EPSG:4326"
)

# Rasterizing the Thiessen polygon areas
area_rast <- rasterize(
  vect(thiessen_geo),
  template,
  field = "area_sq_km"
)

# then plotting
par(mar = c(3,3,2,1))
plot(
  st_geometry(ohio_boundary_geo),
  col    = NA,
  border = NA,
  axes   = FALSE,
  main   = paste0(
    "Pipeline Accident Service Areas in Ohio\n",
    "Thiessen Polygons by Size"
  ),
  cex.main = 1.2
)

# gridlines & lat/long labels
lon_br <- seq(bbox["xmin"], bbox["xmax"], length.out = 5)
lat_br <- seq(bbox["ymin"], bbox["ymax"], length.out = 5)
lon_lb <- paste0(abs(round(lon_br,2)), "°", ifelse(lon_br<0,"W","E"))
lat_lb <- paste0(abs(round(lat_br,2)), "°", ifelse(lat_br<0,"S","N"))
for (x in lon_br) abline(v = x, col = "grey90", lty = 2)
for (y in lat_br) abline(h = y, col = "grey90", lty = 2)

# I'm generating a continuous raster layer and applying a reversed Plasma palette
# to enhance visual contrast and emphasize areas with higher values more clearly.

mf_raster(
  x       = area_rast,
  pal     = viridisLite::plasma(100, direction = -1),
```

```
  nbreaks = 100,
  add     = TRUE,
  leg_pos = NA
)

# legend
mf_legend(
  type   = "cont",
  pos    = "right",
  title  = "Service Area (sq km)",
  val    = c(2000, 4000, 6000),
  pal    = viridisLite::plasma(100, direction = -1)
)

# Overlay polygon boundaries here
mf_map(
  x      = thiessen_geo,
  type   = "base",
  col    = NA,
  border = "black",
  lwd    = 0.2,
  add    = TRUE
)

# Overlay ohio's boundary & accident locations
mf_map(
  x      = ohio_boundary_geo,
  type   = "base",
  col    = NA,
  border = "black",
  lwd    = 0.5,
  add    = TRUE
)
mf_map(
  x      = ohio_accidents_geo,
  type   = "base",
  col    = "red",
  pch    = 19,
  cex    = 0.6,
  add    = TRUE
)

# Add axes, north arrow, scale bar, and frame
axis(1, at = lon_br, labels = lon_lb, cex.axis = 0.7)
axis(2, at = lat_br, labels = lat_lb, cex.axis = 0.7)
mf_arrow(pos = "topleft")
mf_scale(pos = "bottomleft")
box()
```

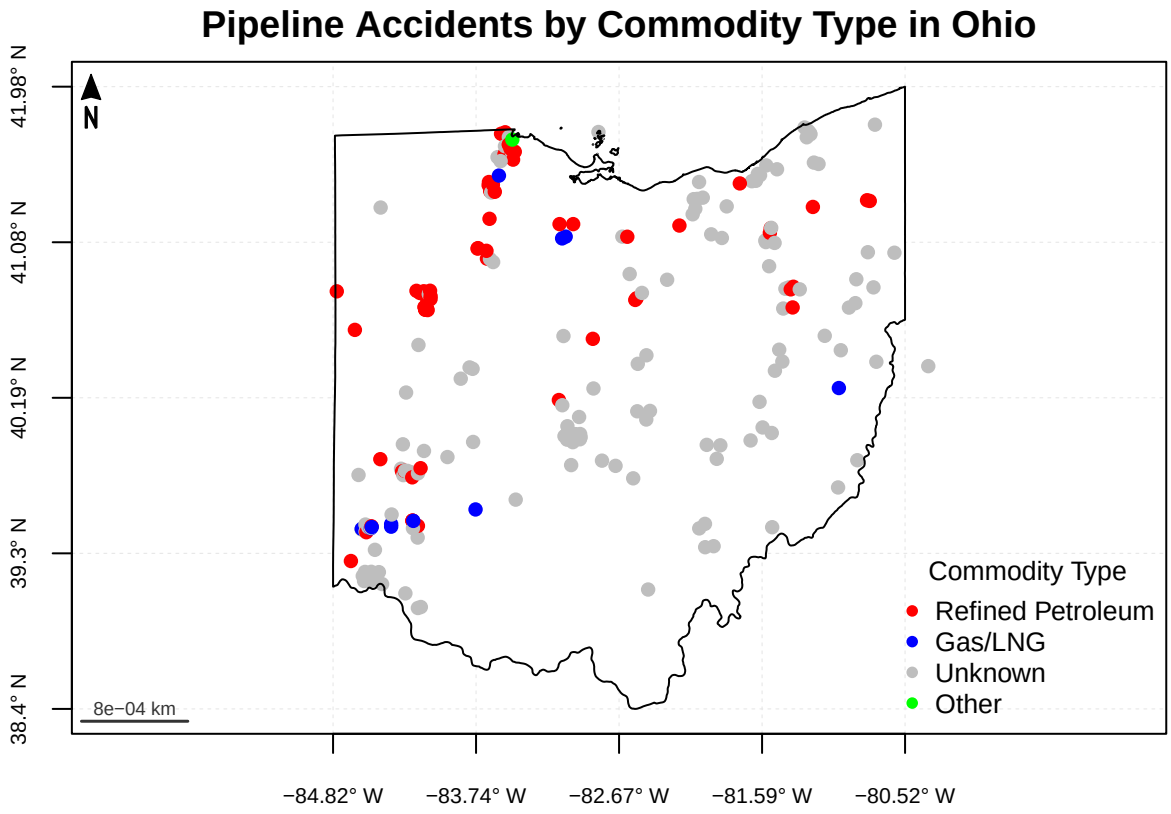



```r
# I'm creating a bar chart to show the distribution of pipeline accidents by commodity type,
# and then transforming the data to geographic coordinates so I can map the results accurately.

ohio_boundary_geo <- st_transform(ohio_boundary, 4326)
ohio_accidents_geo <- st_transform(ohio_accidents_sf, 4326)

# Setting up the plotting area for the bar chart
par(mar = c(6, 8, 2, 2))

#  Iam trying to filter out empty commodity values for the plot
commodity_plot <- commodity_summary[commodity_summary$Commodity != "", ]

# Creating barplot with commodity types
barplot(commodity_plot$Count,
        names.arg = commodity_plot$Commodity,
        horiz = TRUE,
        col = "steelblue",
        las = 1,
        cex.names = 0.8,
        main = "Pipeline Accidents by Commodity Type in Ohio",
        xlab = "Number of Accidents")

# i am adding  percentages as text
text(commodity_plot$Count + max(commodity_plot$Count)/20,
     seq(1.5, by = 1.2, length.out = nrow(commodity_plot)) - 0.5,
     paste0(round(commodity_plot$Percentage, 1), "%"),
     cex = 0.7)
```




```
# Create a map showing accidents by commodity type

par(mar = c(3, 3, 2, 1))

# adding an empty plot with Ohio's extent
plot(st_geometry(ohio_boundary_geo),
     col = NA,
     border = NA,
     axes = FALSE,
     main = "Pipeline Accidents by Commodity Type in Ohio")

# i am getting the bounding box for grid lines
bbox <- st_bbox(ohio_boundary_geo)

# latitude and longitude breaks
lon_breaks <- seq(bbox["xmin"], bbox["xmax"], length.out = 5)
lat_breaks <- seq(bbox["ymin"], bbox["ymax"], length.out = 5)

# lat/long labels
lon_labels <- paste0(round(lon_breaks, 2), "° W")
lat_labels <- paste0(round(lat_breaks, 2), "° N")

# grid lines
for(x in lon_breaks) {
  abline(v = x, col = "grey90", lwd = 0.5, lty = 2)
}

for(y in lat_breaks) {
  abline(h = y, col = "grey90", lwd = 0.5, lty = 2)
```

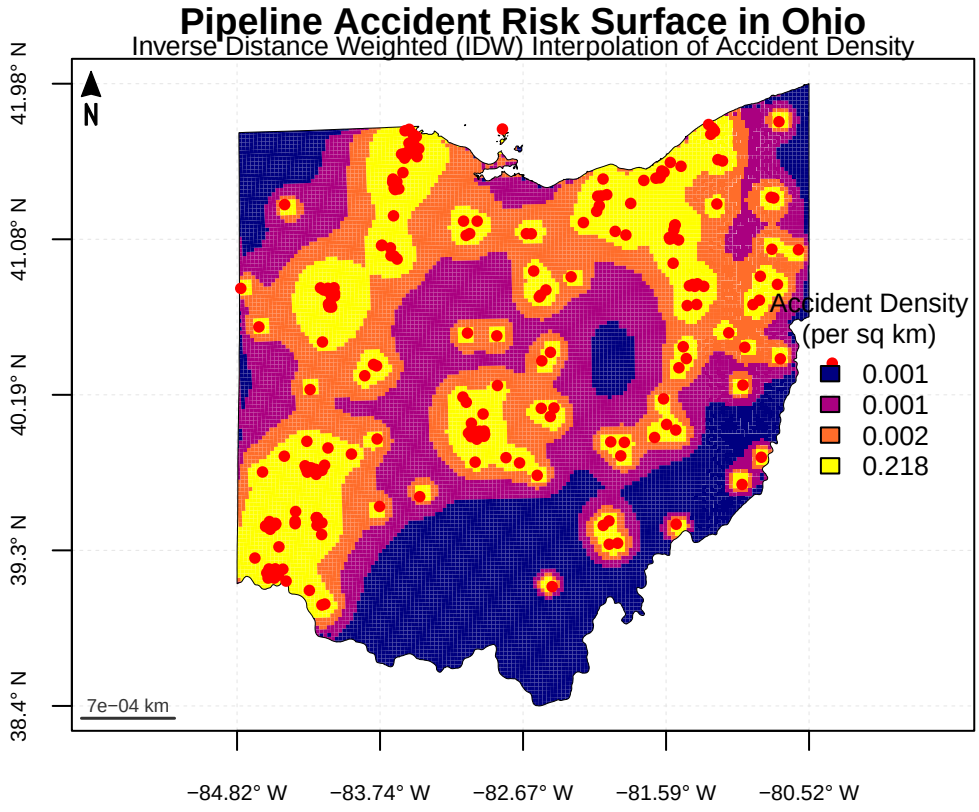
```
}


plot(st_geometry(ohio_boundary_geo),
     col = "white",
     border = NA,
     add = TRUE)

# I am assigning colors for commodity categories
commodity_colors <- c("Refined Petroleum" = "red", "Gas/LNG" = "blue", "Unknown" = "gray", "Other" = "g

# then assigning colors based on commodity categories
point_colors <- commodity_colors[ohio_accidents_geo$commodity_category]

# here i am plotting accident points colored by commodity
plot(st_geometry(ohio_accidents_geo),
     col = point_colors,
     pch = 16,
     cex = 1,
     add = TRUE)

# Adding Ohio boundary
plot(st_geometry(ohio_boundary_geo),
     col = NA,
     border = "black",
     add = TRUE)

# then adding axes with lat/long labels
axis(1, at = lon_breaks, labels = lon_labels, cex.axis = 0.7)
axis(2, at = lat_breaks, labels = lat_labels, cex.axis = 0.7)

# north arrow
mf_arrow(pos = "topleft", col = "black")

# scale bar
mf_scale(pos = "bottomleft")


box()

# legend
legend("bottomright",
       legend = names(commodity_colors),
       col = commodity_colors,
       pch = 16,
       bty = "n",
       cex = 0.8,
       title = "Commodity Type")
```




```
library(sf)
library(mapsf)


ohio_boundary_geo <- st_transform(ohio_boundary, 4326)
idw_sf_geo <- st_transform(idw_sf, 4326)
ohio_accidents_geo <- st_transform(ohio_accidents_sf, 4326)


quant_breaks <- quantile(idw_sf_geo$var1.pred, probs = seq(0, 1, 0.25))


par(mar = c(3, 3, 2, 6))


plot(st_geometry(ohio_boundary_geo),
     col = NA,
     border = NA,
     axes = FALSE,
     main = "Pipeline Accident Risk Surface in Ohio")


bbox <- st_bbox(ohio_boundary_geo)


lon_breaks <- seq(bbox["xmin"], bbox["xmax"], length.out = 5)
lat_breaks <- seq(bbox["ymin"], bbox["ymax"], length.out = 5)
```

```r
lon_labels <- paste0(round(lon_breaks, 2), "° W")
lat_labels <- paste0(round(lat_breaks, 2), "° N")


for(x in lon_breaks) {
  abline(v = x, col = "grey90", lwd = 0.5, lty = 2)
}
for(y in lat_breaks) {
  abline(h = y, col = "grey90", lwd = 0.5, lty = 2)
}


idw_colors <- c("#000080", "#800080", "#FF0080", "#FFA500", "#FFFF00")
n_breaks <- length(quant_breaks) - 1
idw_cols <- colorRampPalette(idw_colors)(n_breaks)

# here i am adding IDW interpolation without white spaces
plot(st_geometry(idw_sf_geo),
     col = idw_cols[findInterval(idw_sf_geo$var1.pred, quant_breaks)],
     border = NA,
     add = TRUE)

# then I add Ohio boundary to clip edges
plot(st_geometry(ohio_boundary_geo),
     col = NA,
     border = "black",
     lwd = 0.5,
     add = TRUE)

# then I plot accident points
plot(st_geometry(ohio_accidents_geo),
     col = "red",
     pch = 16,
     cex = 0.8,
     add = TRUE)

# subtitle
mtext("Inverse Distance Weighted (IDW) Interpolation of Accident Density",
      side = 3, line = -0.1, cex = 0.8)

# lat/long labels
axis(1, at = lon_breaks, labels = lon_labels, cex.axis = 0.7)
axis(2, at = lat_breaks, labels = lat_labels, cex.axis = 0.7)

# north arrow
mf_arrow(pos = "topleft", col = "black")

# scale bar
mf_scale(pos = "bottomleft")


box()
```

```r
# legend

legend_labels <- format(round(quant_breaks[-1], 3), nsmall = 3)
legend("right",
       legend = legend_labels,
       fill = idw_cols,
       bty = "n",
       cex = 0.8,
       title = "Accident Density\n(per sq km)")
```




```r
library(sf)
library(mapsf)
library(viridisLite)

# First, I'm calculating population density for each census tract,
# assuming that `ohio_tracts_sf` already contains a column named 'estimate' for total population.

ohio_tracts_sf$pop_density <- ohio_tracts_sf$estimate / as.numeric(st_area(ohio_tracts_sf)) * 1000000

# then I transform everything to WGS84 for grid values to display on the map
ohio_tracts_geo <- st_transform(ohio_tracts_sf, 4326)
high_risk_areas_geo <- st_transform(high_risk_areas, 4326)
mod_risk_areas_geo <- st_transform(mod_risk_areas, 4326)
ohio_boundary_geo <- st_transform(ohio_boundary, 4326)

# so I compute gridlines & labels here
bbox <- st_bbox(ohio_boundary_geo)
lon_br <- seq(bbox["xmin"], bbox["xmax"], length.out = 5)
```

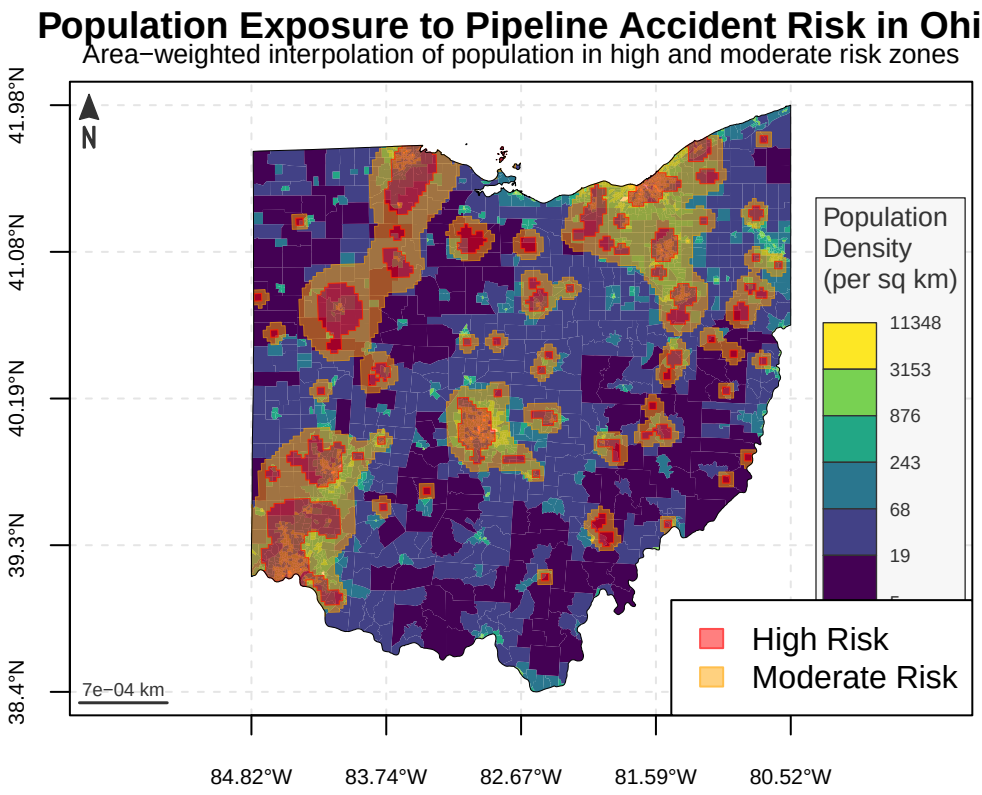
```r
lat_br <- seq(bbox["ymin"], bbox["ymax"], length.out = 5)
lon_lb <- paste0(abs(round(lon_br,2)), "°", ifelse(lon_br<0,"W","E"))
lat_lb <- paste0(abs(round(lat_br,2)), "°", ifelse(lat_br<0,"N","N"))


par(mar = c(3,3,3,6))


plot(
  st_geometry(ohio_boundary_geo),
  col = NA,
  border = NA,
  axes = FALSE,
  main = "Population Exposure to Pipeline Accident Risk in Ohio",
  cex.main = 1.2
)

# subtitle
mtext("Area-weighted interpolation of population in high and moderate risk zones",
      side = 3, line = 0.3, cex = 0.8)  # Increased line from 0.2 to 0.8

# grid lines
for(x in lon_br) abline(v = x, col = "grey90", lty = 2)
for(y in lat_br) abline(h = y, col = "grey90", lty = 2)

# # I'm creating logarithmic breaks for population density
# by calculating the min and max values-while skipping any zeros or NAs to avoid skewed scaling.

min_density <- min(ohio_tracts_geo$pop_density[ohio_tracts_geo$pop_density > 0], na.rm = TRUE)
max_density <- max(ohio_tracts_geo$pop_density, na.rm = TRUE)

log_breaks <- c(5, 19, 68, 243, 876, 3153, 11348)


viridis_pal <- viridisLite::viridis(100, option = "viridis")
dens_pal <- viridis_pal[seq(1, 100, length.out = length(log_breaks) - 1)]

# Plotting population density
mf_map(
  x = ohio_tracts_geo,
  var = "pop_density",
  type = "choro",
  breaks = log_breaks,
  pal = dens_pal,
  border = NA,
  add = TRUE,
  leg_pos = NA
)
```

```
## 9 values are outside the class limits and will be classified as 'NA'.
```

```r
mf_map(
  x = mod_risk_areas_geo,
  type = "base",
```

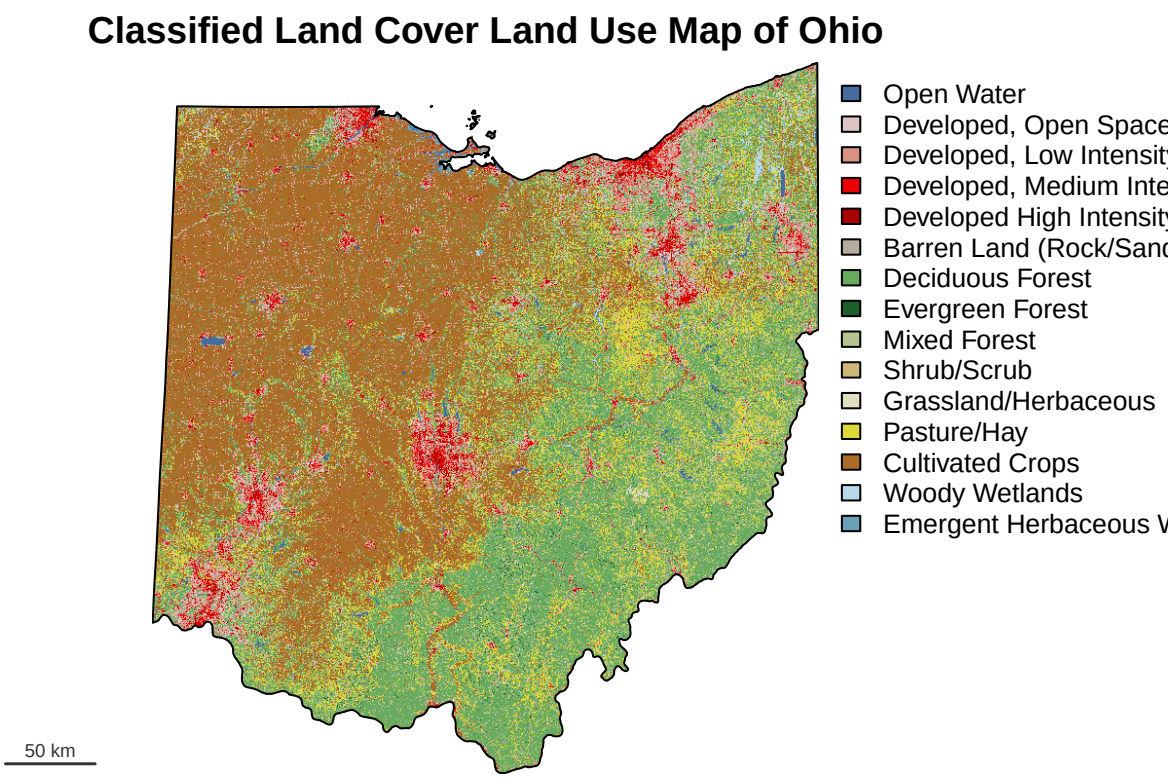
```
  col = adjustcolor("#FFA500", alpha.f = 0.5),  # orange with more transparency
  border = adjustcolor("#FFA500", alpha.f = 0.7),
  lwd = 0.5,
  add = TRUE
)

mf_map(
  x = high_risk_areas_geo,
  type = "base",
  col = adjustcolor("#FF0000", alpha.f = 0.5),  # red with more transparency
  border = adjustcolor("#FF0000", alpha.f = 0.7),
  lwd = 0.5,
  add = TRUE
)

# Ohio boundary
mf_map(
  x = ohio_boundary_geo,
  type = "base",
  col = NA,
  border = "black",
  lwd = 0.5,
  add = TRUE
)

# axes with labels
axis(1, at = lon_br, labels = lon_lb, cex.axis = 0.7)
axis(2, at = lat_br, labels = lat_lb, cex.axis = 0.7)

# north arrow and scale bar
mf_arrow(pos = "topleft")
mf_scale(pos = "bottomleft")


box()

# legend 1
mf_legend(
  type = "choro",
  title = "Population\nDensity\n(per sq km)",
  val = log_breaks,
  pal = dens_pal,
  pos = "right",
  frame = TRUE
)

# legend 2
legend(
  "bottomright",
  legend = c("High Risk", "Moderate Risk"),
  fill = c(adjustcolor("#FF0000", alpha.f = 0.5), adjustcolor("#FFA500", alpha.f = 0.5)),
  border = c(adjustcolor("#FF0000", alpha.f = 0.7), adjustcolor("#FFA500", alpha.f = 0.7)),
  box.lty = 1,
```

```
  bg = "white"
)
```




```
#The following block of code for obtaining NLCD data is partly modified and inspired from:
#Bocinsky, R.K. (2022)
#FedData: Functions to Automate Downloading Geospatial Data from Federal Data Sources
#https://github.com/ropensci/FedData
#Downloaded: April 14, 2025


library(sf)
library(terra)
library(FedData)

ohio_boundary <- states(cb = TRUE, year = 2021) %>%
  dplyr::filter(STUSPS == "OH") %>%
  st_transform(4326)

# I'm downloading the 2016 NLCD land-cover raster using the FedData API

nlcd_api_raster <- get_nlcd(
  template       = ohio_boundary,
  year           = 2016,
  dataset        = "landcover",
  label          = "nlcd_api_ohio",
  extraction.dir = "~/Assignment/nlcd_api_download",
  force.redo     = FALSE
)
```

```r
# Reprojecting it maintain spatial consistency
nlcd_26917 <- project(nlcd_api_raster, "EPSG:26917")

# Reprojecting Ohio boundary as well
ohio_boundary_26917 <- st_transform(ohio_boundary, 26917)

# then I convert the boundary to a terra SpatVector for masking
ohio_vect_26917 <- vect(ohio_boundary_26917)

# then I crop and mask the reprojected raster to the Ohio extent
nlcd_clipped <- crop(nlcd_26917, ohio_vect_26917)
nlcd_clipped <- mask(nlcd_clipped, ohio_vect_26917)

# Plotting the map
plot(nlcd_clipped,
     main = "Classified Land Cover Land Use Map of Ohio",
     axes = FALSE)          # suppress axes and box
plot(st_geometry(ohio_boundary_26917),
     add    = TRUE,
     border = "black")     # overlay the state boundary

# north arrow and scale bar
mf_arrow(pos = "topleft")
mf_scale(pos = "bottomleft")
```




```r
# References

#1) Anselin, L. (1995) 'Local Indicators of Spatial Association-LISA', Geographical Analysis, 27(2), pp
```

#2) Baddeley, A., Rubak, E., and Turner, R. (2022) spatstat: Spatial Point Pattern Analysis, Model-Fitt

#3) Bivand, R. and Anselin, L. (2022) spdep: Spatial Dependence: Weighting Schemes, Statistics and Mode

#4) Bivand, R., Pebesma, E., and Gomez-Rubio, V. (2013) Applied spatial data analysis with R. 2nd edn.

#5) Bocinsky, R.K. (2022) FedData: Functions to Automate Downloading Geospatial Data from Federal Data

#6) Giraud, T. (2022) mapsf: Thematic Cartography. R package version 0.6.0. Available at: https://riate

#7) National Renewable Energy Laboratory (2024) 'National Landcover Database'. Available at: https://ww

#8) Pebesma, E. (2022) sf: Simple Features for R. R package version 1.0-9. Available at: https://r-spat

#9) Pebesma, E. (2022) gstat: Spatial and Spatio-Temporal Geostatistical Modelling, Prediction and Simu

#10) Pebesma, E. and Bivand, R. (2023) Spatial Data Science: With Applications in R. Chapman and Hall/C

#11) Walker, K. (2023) tidycensus: Load US Census Boundary and Attribute Data as 'tidyverse' and 'sf'-R

#12) U.S. Geological Survey (2024) 'National Hydrography Dataset'. Available at: https://www.usgs.gov/n

#14) U.S. Census Bureau (2019) 'American Community Survey 5-Year Estimates'. Available at: https://www.

#15) Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L., François, R., Grolemund, G., Hayes, A