2024 Crime Pattern Analysis in Los Angeles City and USC DPS Area

Chenyi Weng

SSCI586 - GIS Programming and Customization

Professor Jennifer N. Swift

Nov 3, 2024

1. Introduction

The objective of Project 4 is to analyze crime patterns within Los Angeles City for the year 2024, with a specific focus on the USC Department of Public Safety (DPS) patrol area. This project uses Python and ArcGIS Pro to prepare data, develop geospatial tools, and visualize crime trends. Due to technical limitations, the tools could not be uploaded or shared via the online toolbox system; however, they were fully implemented within the ArcGIS Pro environment. This report includes a detailed description of the methodology, data preparation, and the two custom tools developed: Cluster Data Tool and Format DateTime Tool.

2. Study Area

The project focuses on two study areas to provide a comprehensive view of crime patterns in Los Angeles. The primary area of interest is Los Angeles City, offering insights across the broader metropolitan region. Figure 1 illustrates the geographical extent of this area, highlighting its boundaries and location within Southern California. Additionally, a more focused study area, the USC Department of Public Safety (DPS) Area, centers on USC's immediate surroundings to allow for detailed analysis within this specific region.

The USC DPS patrol area became a focal point following a significant event on April 24, 2024, when USC implemented enhanced security measures due to a protest on campus that led to nearly 100 arrests. This incident prompted USC to restrict campus access and increase security within the DPS patrol area. This analysis aims to explore whether these new measures affected crime rates by comparing data before and after the incident.

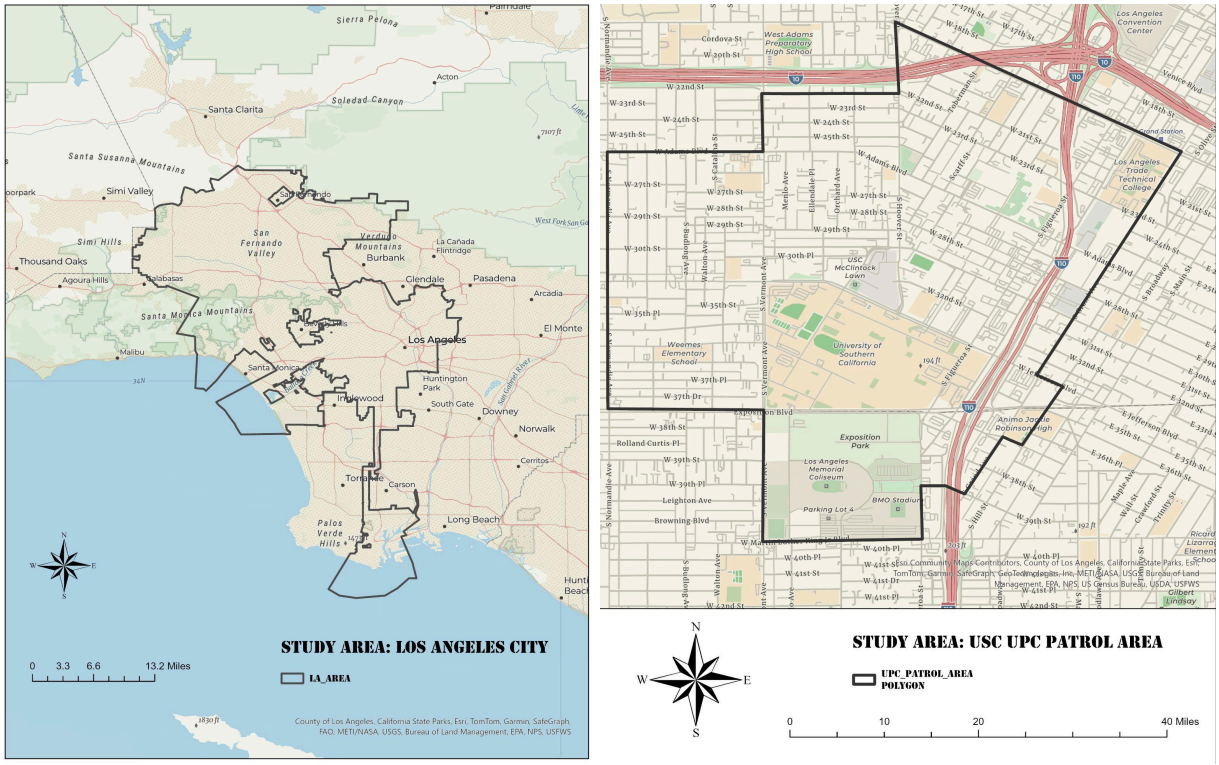


Figure 1. Two Study Area - Los Angeles City and USC Department of Public Safety (DPS) Area

3. Data Collection and Preparation

Three primary datasets were used in this analysis:

1. **UPC_Patrol_Area_1-polygon.shp**: This shapefile defines the USC DPS patrol area boundary, which is central to analyzing crime data specific to the patrol zone. This area is displayed in **Figure 1**.
    - **Creating the UPC Patrol Area Shapefile**:
        - To obtain the USC DPS patrol area shapefile, the patrol area map was initially accessed on Google My Maps. (https://dps.usc.edu/patrol/)
        - **Export to KML**: In Google My Maps, the "⋮" menu was selected to access "Download KML," exporting the patrol area as a KML file.

- **Convert to Shapefile**: The KML file was then converted to a shapefile using MyGeodata Converter. (https://mygeodata.cloud/converter/kml-to-shp).

2. **la_area.shp**: This shapefile outlines the boundary of Los Angeles City, providing a broader geographic context for citywide crime analysis. Figure 1 shows this boundary alongside the USC DPS area.

3. **final_crime_data.csv**: The main crime dataset, sourced from the Los Angeles Open Data Portal, includes records from 2020 to October 14, 2024, with a focus on 2024 data for this project. Initial data processing involved filtering for 2024 records, formatting date and time fields, and clustering crime descriptions. Figure 2 shows this CSV file.

The table below summarizes the key attributes of the three main data sources:

Data Source	**File Type**	**Key Fields**	**Description**
UPC_Patrol_Area_1-polygon.shp	Shapefile	Boundary Polygon	Defines the USC DPS patrol area, used for spatial filtering of crime data within this area.
la_area.shp	Shapefile	Boundary Polygon	Represents the Los Angeles City boundary for contextualizing and filtering city-wide data.
final_crime_data.csv	CSV	DATE OCC, TIME OCC, Crm Cd Desc	Contains cleaned 2024 crime data with fields for datetime, crime description, and locations.

Table 1. The three main data sources

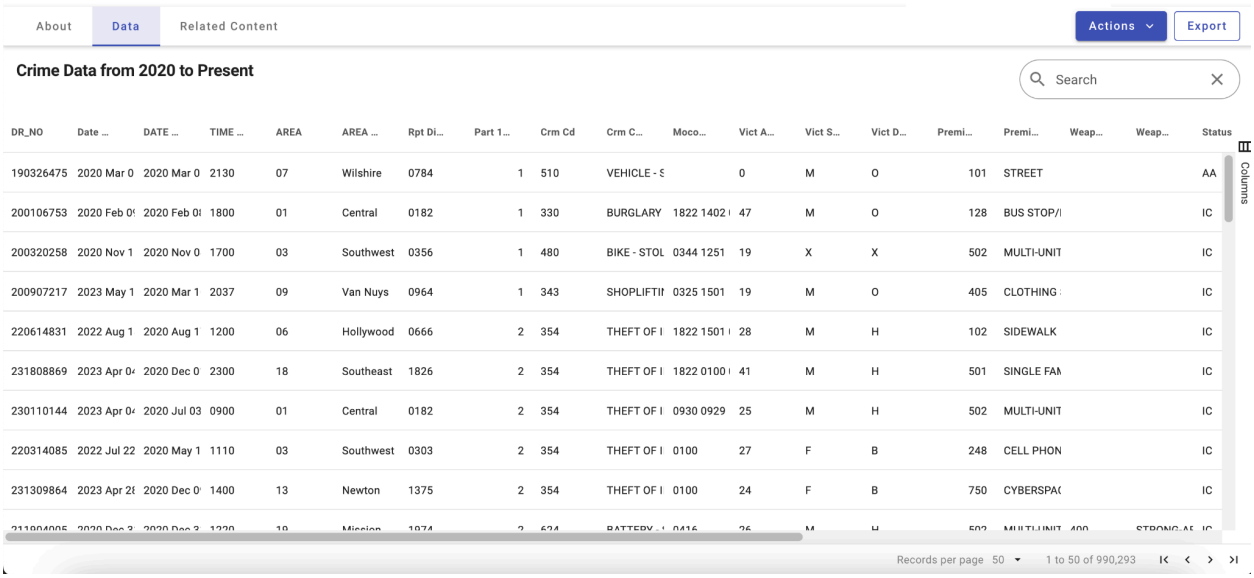


Figure 2. Initial Crime Data from Los Angeles Open Data Portal (2020–Present)

Using Python in Google Colab, the initial data processing (Figure 3, 4, and 5) involved:

- Removing unnecessary columns.
- Filtering to show only the top 15 crime categories.
- Aggregating these categories into 12 primary groups using KMeans clustering.

```
import pandas as pd
df = pd.read_csv("/content/Crime_Data_from_2020_to_Present_20241028.csv")
df = df.drop(df.index[109548:986501])
df.to_csv("cleaned_crime_data.csv", index=False)
```

Figure 3. Data Cleaning in Python - Filtering for 2024 Data Only (Google Colab)

```python
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans
import numpy as np

# Load the data
df = pd.read_csv("/content/final_crime_data.csv")

# Extract the "Crm Cd Desc" column
descriptions = df['Crm Cd Desc']

# Vectorize the text data using TF-IDF
vectorizer = TfidfVectorizer(stop_words='english')
X = vectorizer.fit_transform(descriptions)

# Find the top 15 clusters by varying the number of clusters
inertia_values = []
cluster_range = range(1, 20)
for n_clusters in cluster_range:
    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
    kmeans.fit(X)
    inertia_values.append(kmeans.inertia_)

# Determine the optimal number of clusters using the elbow method
optimal_clusters = 15  # Manually set based on the elbow method

# Perform clustering using KMeans
kmeans = KMeans(n_clusters=optimal_clusters, random_state=42)
kmeans.fit(X)

# Add cluster labels to the original data
df['Cluster'] = kmeans.labels_

# Identify the top samples for each cluster
cluster_counts = df['Cluster'].value_counts().sort_values(ascending=False)
top_15_clusters = cluster_counts.head(15)

# Display the category name for the top 15 clusters
top_15_category = {}
for cluster in top_15_clusters.index:
    cluster_data = df[df['Cluster'] == cluster]['Crm Cd Desc']
    top_category = cluster_data.mode().iloc[0]  # Select the most common category as representative
    top_15_category[cluster] = top_category

# Print the top 15 clusters and their category names
for cluster, category in top_15_category.items():
    print(f"Cluster {cluster}: {category}")

# Save the data with cluster labels to a new file
df.to_csv("/content/clustered_crime_data_top15.csv", index=False)
```

```
Cluster 4: VEHICLE
Cluster 9: ASSAULT WITH DEADLY WEAPON, AGGRAVATED ASSAULT
Cluster 5: BURGLARY FROM VEHICLE
Cluster 0: THEFT FROM MOTOR VEHICLE
Cluster 1: VANDALISM
Cluster 6: SHOPLIFTING
Cluster 7: THEFT PLAIN
Cluster 8: TRESPASSING
Cluster 3: BATTERY
Cluster 13: THEFT OF IDENTITY
Cluster 2: THEFT
Cluster 10: INTIMATE PARTNER
Cluster 12: ROBBERY
Cluster 14: CRIMINAL THREATS
Cluster 11: VIOLATION OF RESTRAINING ORDER
```

Figure 4. Filtering to show only the top 15 crime categories *(Google Colab)*

```python
import pandas as pd

# Load the CSV file
file_path = '/content/final_crime_data.csv'
df = pd.read_csv(file_path)

# Define a function to categorize 'Crm Cd Desc' to reduce classes based on clustering results
def categorize_crime(description):
    # Create categories based on the clustering results and common descriptions
    if 'VEHICLE' in description:
        return 'VEHICLE'
    elif 'BURGLARY' in description:
        return 'BURGLARY'
    elif 'ROBBERY' in description:
        return 'ROBBERY'
    elif 'ASSAULT' in description or 'BATTERY' in description:
        return 'ASSAULT/BATTERY'
    elif 'VANDALISM' in description:
        return 'VANDALISM'
    elif 'TRESPASS' in description:
        return 'TRESPASSING'
    elif 'THEFT' in description:
        return 'THEFT'
    elif 'SHOPLIFTING' in description:
        return 'SHOPLIFTING'
    elif 'THREAT' in description:
        return 'CRIMINAL THREATS'
    elif 'IDENTITY' in description:
        return 'THEFT OF IDENTITY'
    elif 'PARTNER' in description:
        return 'INTIMATE PARTNER VIOLENCE'
    elif 'RESTRAINING' in description:
        return 'VIOLATION OF RESTRAINING ORDER'
    else:
        return 'OTHER'

# Apply the categorization function to 'Crm Cd Desc' column
df['Crm Cd Desc'] = df['Crm Cd Desc'].apply(categorize_crime)

# Check the unique values to verify the categorization
unique_classes = df['Crm Cd Desc'].unique()

# Save the updated dataframe to a new CSV
output_path = '/content/final_crime_data_categorized.csv'
df.to_csv(output_path, index=False)

# Output the unique classes to verify
unique_classes
```

```
array(['VEHICLE', 'TRESPASSING', 'ASSAULT/BATTERY', 'OTHER',
       'SHOPLIFTING', 'VANDALISM', 'THEFT', 'ROBBERY', 'BURGLARY',
       'CRIMINAL THREATS', 'INTIMATE PARTNER VIOLENCE',
       'VIOLATION OF RESTRAINING ORDER'], dtype=object)
```

Figure 5. Categorizing Crime Data into 12 Groups Using KMeans Clustering (Google Colab)

4. Method and Tool Development

4.1. Data Cleaning and Formatting

Using Python and Pandas, the final_crime_data.csv was imported, and only records from 2024 were retained. The "DATE OCC" and "TIME OCC" columns were merged into a new datetime column using the “Format DateTime Tool”.

4.2. Tool Development

To facilitate analysis, two custom tools were developed in ArcGIS Pro as shown in Figure 6 to Figure10:

- **Format DateTime Tool**: This tool formats and merges separate date and time fields into a single datetime column, streamlining the data for temporal analysis.
- **Cluster Data Tool**: This tool enables clustering of similar crime descriptions into top categories for simplified data interpretation.

```python
import arcpy
import pandas as pd

class Toolbox(object):
    def __init__(self):
        """Define the toolbox name and alias"""
        self.label = "ClusterData Toolbox"
        self.alias = "ClusterDataToolbox"
        self.tools = [ClusterDataTool]

class ClusterDataTool(object):
    def __init__(self):
        """Define the tool properties"""
        self.label = "Cluster Data Tool"
        self.description = "Cluster analysis on CSV data to determine the top categories."
        self.canRunInBackground = False

    def getParameterInfo(self):
        """Define parameter definitions"""
        params = []

        # Input CSV file parameter
        input_file = arcpy.Parameter(
            displayName="Input CSV File",
            name="input_file",
            datatype="DEFile",
            parameterType="Required",
            direction="Input"
        )
        params.append(input_file)

        # Output CSV file parameter
        output_file = arcpy.Parameter(
            displayName="Output CSV File",
            name="output_file",
            datatype="DEFile",
            parameterType="Required",
            direction="Output"
        )
```

```
        params.append(output_file)

        # Field to analyze parameter
        field_to_analyze = arcpy.Parameter(
            displayName="Field to Analyze",
            name="field_to_analyze",
            datatype="Field",
            parameterType="Required",
            direction="Input"
        )
        field_to_analyze.parameterDependencies = [input_file.name]
        field_to_analyze.datatype = "GPString"  # Use GPString to allow text fields
        params.append(field_to_analyze)

        # Number of top categories parameter
        top_categories = arcpy.Parameter(
            displayName="Number of Top Categories",
            name="top_categories",
            datatype="GPLong",
            parameterType="Required",
            direction="Input"
        )
        params.append(top_categories)

        return params

    def isLicensed(self):
        """Set whether tool is licensed to execute"""
        return True

    def updateParameters(self, parameters):
        """Modify parameters before internal validation is performed."""
        if parameters[0].valueAsText and not parameters[2].filter.list:
            # Read the CSV file and populate available fields
            input_file_path = parameters[0].valueAsText
            try:
                df = pd.read_csv(input_file_path)
                field_names = df.columns.tolist()
                parameters[2].filter.list = field_names
            except:
                pass
        return

    def updateMessages(self, parameters):
        """Modify messages created by internal validation."""
        # Verify that the input file exists
        if parameters[0].altered and not parameters[0].valueAsText:
            parameters[0].setErrorMessage("Input CSV file is required.")

        # Check if output file path is writable
        if parameters[1].altered:
            output_path = parameters[1].valueAsText
            if output_path and not arcpy.env.workspace:
                parameters[1].setErrorMessage("Please specify a valid output location.")
        return

    def execute(self, parameters, messages):
        """Run the tool"""
        try:
            input_file = parameters[0].valueAsText
            output_file = parameters[1].valueAsText
            field_to_analyze = parameters[2].valueAsText
            top_categories = parameters[3].value

            # Load CSV file into pandas dataframe
            df = pd.read_csv(input_file)

            # Check if the specified field exists in the dataframe
            if field_to_analyze not in df.columns:
                messages.addErrorMessage(f"Field '{field_to_analyze}' not found in CSV.")
                return

            # Perform analysis and reduce classes
            category_counts = df[field_to_analyze].value_counts()
            top_categories_list = category_counts.head(top_categories).index.tolist()

            def categorize(row):
                for category in top_categories_list:
                    if category in row:
                        return category
                return "OTHER"

            df[field_to_analyze] = df[field_to_analyze].apply(categorize)

            # Save the updated dataframe to the output file
            df.to_csv(output_file, index=False)
```

```
            # Display top categories in messages
            messages.addMessage(f"Top {top_categories} categories in '{field_to_analyze}' are: {', '.join(top_categories_list)}")

        except Exception as e:
            messages.addErrorMessage(f"An error occurred: {str(e)}")
```

Figure 6. The python script code for the "Format DateTime Tool"

```
import arcpy
import pandas as pd

class Toolbox(object):
    def __init__(self):
        """Define the toolbox (the name of the toolbox is the name of the .pyt file)."""
        self.label = "Project4 Toolbox"
        self.alias = "Project4_Toolbox"
        # Ensure that the referenced tool class name is correct
        self.tools = [FormatDatetimeTool]  # Defines the tools included in the toolbox

class FormatDatetimeTool(object):
    def __init__(self):
        """Define the tool (tool name is the name of the class)."""
        self.label = "Format DateTime Tool"
        self.description = "Tool to format selected 'Date' and 'Time' columns into a single datetime column"
        self.canRunInBackground = False

    def getParameterInfo(self):
        """Define parameter definitions"""
        params = []

        # Define input CSV file parameter
        input_file = arcpy.Parameter(
            displayName="Input CSV File",
            name="input_file",
            datatype="DEFile",  # File type
            parameterType="Required",  # Required parameter
            direction="Input"  # Input parameter
        )
        params.append(input_file)

        # Define date column parameter
        date_column = arcpy.Parameter(
            displayName="Date Column",
            name="date_column",
            datatype="Field",
            parameterType="Required",
            direction="Input"
        )
        date_column.parameterDependencies = [input_file.name]
        date_column.datatype = "GPString"  # Use GPString to allow text fields
        params.append(date_column)

        # Define time column parameter
        time_column = arcpy.Parameter(
```

```
            displayName="Time Column",
            name="time_column",
            datatype="Field",
            parameterType="Required",
            direction="Input"
        )
        time_column.parameterDependencies = [input_file.name]
        time_column.datatype = "GPString"  # Use GPString to allow text fields
        params.append(time_column)

        # Define output CSV file parameter
        output_file = arcpy.Parameter(
            displayName="Output CSV File",
            name="output_file",
            datatype="DEFile",  # File type
            parameterType="Required",  # Required parameter
            direction="Output"  # Output parameter
        )
        params.append(output_file)

        return params

    def updateParameters(self, parameters):
        """Modify parameters before internal validation is performed."""
        # Populate field options for date and time columns after selecting input file
        if parameters[0].valueAsText and not parameters[1].filter.list:
            input_file_path = parameters[0].valueAsText
            try:
                df = pd.read_csv(input_file_path)
                field_names = df.columns.tolist()
                parameters[1].filter.list = field_names
                parameters[2].filter.list = field_names
            except:
                pass
        return

    def execute(self, parameters, messages):
        """The source code of the tool"""
        input_file = parameters[0].valueAsText  # Get the input CSV file path
        date_column = parameters[1].valueAsText  # Get the date column name
        time_column = parameters[2].valueAsText  # Get the time column name
        output_file = parameters[3].valueAsText  # Get the output CSV file path

        # Read the CSV file using pandas
        try:
            df = pd.read_csv(input_file)
        except FileNotFoundError:
            raise FileNotFoundError("The input CSV file was not found. Please check the path.")

        # Define a helper function for time formatting
        def format_time(time_val):
            try:
                time_str = str(int(time_val)).zfill(4)
                return f"{time_str[:2]}:{time_str[2:]}"
            except ValueError:
                raise ValueError("Invalid value found in the time column. Ensure all values are numeric.")

        # Format the date and time columns and combine them into the new 'DATETIME' column
        try:
            df[time_column] = df[time_column].apply(format_time)
            df['DATETIME'] = pd.to_datetime(df[date_column] + ' ' + df[time_column], format="%Y-%m-%d %H:%M")
        except KeyError as e:
            raise KeyError(f"Missing column in CSV file: {e}")
        except ValueError as e:
            raise ValueError(f"Date or Time formatting error: {e}")

        # Drop original date and time columns
        df.drop([date_column, time_column], axis=1, inplace=True)

        # Save the formatted data to the output file
        try:
            df.to_csv(output_file, index=False)
        except Exception as e:
            raise RuntimeError(f"Failed to save formatted CSV file: {e}")

        messages.addMessage(f"Formatted data saved to {output_file}")
```

Figure 7. The python script code for the "Cluster Data Tool"

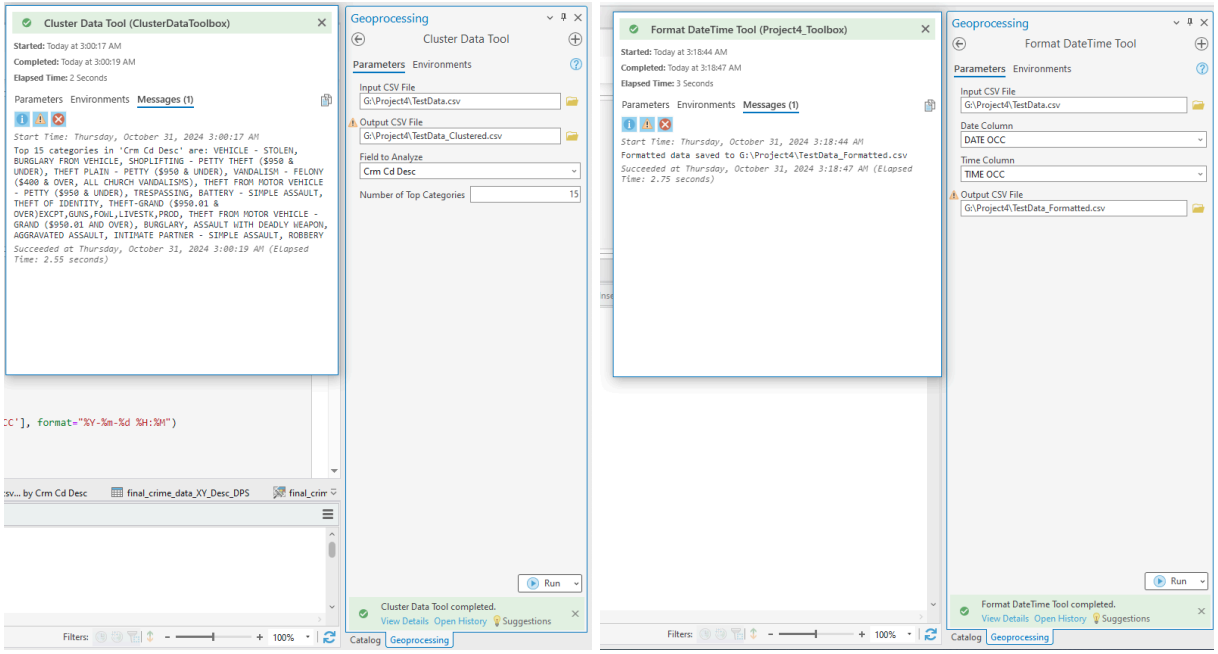


Figure 8. "Cluster Data Tool" and "Format DateTime Tool" Interfaces

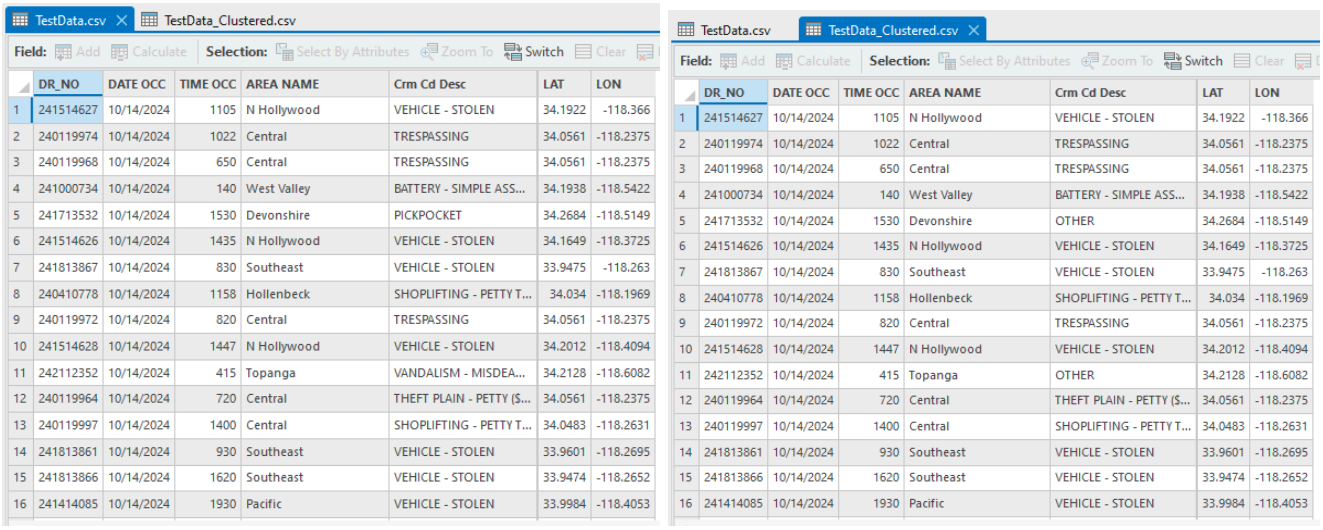
TestData_Clustered.csv | TestData.csv | TestData_Formatted.csv

	DR_NO	DATE OCC	TIME OCC	AREA NAME	Crm Cd Desc	LAT	LON
1	241514627	10/14/2024	1105	N Hollywood	VEHICLE - STOLEN	34.1922	-118.366
2	240119974	10/14/2024	1022	Central	TRESPASSING	34.0561	-118.2375
3	240119968	10/14/2024	650	Central	TRESPASSING	34.0561	-118.2375
4	241000734	10/14/2024	140	West Valley	BATTERY - SIMPLE ASS...	34.1938	-118.5422
5	241713532	10/14/2024	1530	Devonshire	PICKPOCKET	34.2684	-118.5149
6	241514626	10/14/2024	1435	N Hollywood	VEHICLE - STOLEN	34.1649	-118.3725
7	241813867	10/14/2024	830	Southeast	VEHICLE - STOLEN	33.9475	-118.263
8	240410778	10/14/2024	1158	Hollenbeck	SHOPLIFTING - PETTY T...	34.034	-118.1969
9	240119972	10/14/2024	820	Central	TRESPASSING	34.0561	-118.2375
10	241514628	10/14/2024	1447	N Hollywood	VEHICLE - STOLEN	34.2012	-118.4094
11	242112352	10/14/2024	415	Topanga	VANDALISM - MISDEA...	34.2128	-118.6082
12	240119964	10/14/2024	720	Central	THEFT PLAIN - PETTY ($...	34.0561	-118.2375
13	240119997	10/14/2024	1400	Central	SHOPLIFTING - PETTY T...	34.0483	-118.2631
14	241813861	10/14/2024	930	Southeast	VEHICLE - STOLEN	33.9601	-118.2695
15	241813866	10/14/2024	1620	Southeast	VEHICLE - STOLEN	33.9474	-118.2652
16	241414085	10/14/2024	1930	Pacific	VEHICLE - STOLEN	33.9984	-118.4053

TestData_Clustered.csv | TestData.csv | TestData_Formatted.csv

	DR_NO	AREA NAME	Crm Cd Desc	LAT	LON	DATETIME
1	241514627	N Hollywood	VEHICLE - STOLEN	34.1922	-118.366	10/14/2024 11:05:00 AM
2	240119974	Central	TRESPASSING	34.0561	-118.2375	10/14/2024 10:22:00 AM
3	240119968	Central	TRESPASSING	34.0561	-118.2375	10/14/2024 6:50:00 AM
4	241000734	West Valley	BATTERY - SIMPLE ASS...	34.1938	-118.5422	10/14/2024 1:40:00 AM
5	241713532	Devonshire	PICKPOCKET	34.2684	-118.5149	10/14/2024 3:30:00 PM
6	241514626	N Hollywood	VEHICLE - STOLEN	34.1649	-118.3725	10/14/2024 2:35:00 PM
7	241813867	Southeast	VEHICLE - STOLEN	33.9475	-118.263	10/14/2024 8:30:00 AM
8	240410778	Hollenbeck	SHOPLIFTING - PETTY T...	34.034	-118.1969	10/14/2024 11:58:00 AM
9	240119972	Central	TRESPASSING	34.0561	-118.2375	10/14/2024 8:20:00 AM
10	241514628	N Hollywood	VEHICLE - STOLEN	34.2012	-118.4094	10/14/2024 2:47:00 PM
11	242112352	Topanga	VANDALISM - MISDEA...	34.2128	-118.6082	10/14/2024 4:15:00 AM
12	240119964	Central	THEFT PLAIN - PETTY ($...	34.0561	-118.2375	10/14/2024 7:20:00 AM
13	240119997	Central	SHOPLIFTING - PETTY T...	34.0483	-118.2631	10/14/2024 2:00:00 PM
14	241813861	Southeast	VEHICLE - STOLEN	33.9601	-118.2695	10/14/2024 9:30:00 AM
15	241813866	Southeast	VEHICLE - STOLEN	33.9474	-118.2652	10/14/2024 4:20:00 PM
16	241414085	Pacific	VEHICLE - STOLEN	33.9984	-118.4053	10/14/2024 7:30:00 PM

Figure 9. Data Before and After Applying "Format DateTime Tool"



TestData.csv | TestData_Clustered.csv

Field: Add Calculate | Selection: Select By Attributes Zoom To Switch Clear

	DR_NO	DATE OCC	TIME OCC	AREA NAME	Crm Cd Desc	LAT	LON
1	241514627	10/14/2024	1105	N Hollywood	VEHICLE - STOLEN	34.1922	-118.366
2	240119974	10/14/2024	1022	Central	TRESPASSING	34.0561	-118.2375
3	240119968	10/14/2024	650	Central	TRESPASSING	34.0561	-118.2375
4	241000734	10/14/2024	140	West Valley	BATTERY - SIMPLE ASS...	34.1938	-118.5422
5	241713532	10/14/2024	1530	Devonshire	PICKPOCKET	34.2684	-118.5149
6	241514626	10/14/2024	1435	N Hollywood	VEHICLE - STOLEN	34.1649	-118.3725
7	241813867	10/14/2024	830	Southeast	VEHICLE - STOLEN	33.9475	-118.263
8	240410778	10/14/2024	1158	Hollenbeck	SHOPLIFTING - PETTY T...	34.034	-118.1969
9	240119972	10/14/2024	820	Central	TRESPASSING	34.0561	-118.2375
10	241514628	10/14/2024	1447	N Hollywood	VEHICLE - STOLEN	34.2012	-118.4094
11	242112352	10/14/2024	415	Topanga	VANDALISM - MISDEA...	34.2128	-118.6082
12	240119964	10/14/2024	720	Central	THEFT PLAIN - PETTY ($...	34.0561	-118.2375
13	240119997	10/14/2024	1400	Central	SHOPLIFTING - PETTY T...	34.0483	-118.2631
14	241813861	10/14/2024	930	Southeast	VEHICLE - STOLEN	33.9601	-118.2695
15	241813866	10/14/2024	1620	Southeast	VEHICLE - STOLEN	33.9474	-118.2652
16	241414085	10/14/2024	1930	Pacific	VEHICLE - STOLEN	33.9984	-118.4053

TestData.csv | TestData_Clustered.csv

Field: Add Calculate | Selection: Select By Attributes Zoom To Switch Clear

	DR_NO	DATE OCC	TIME OCC	AREA NAME	Crm Cd Desc	LAT	LON
1	241514627	10/14/2024	1105	N Hollywood	VEHICLE - STOLEN	34.1922	-118.366
2	240119974	10/14/2024	1022	Central	TRESPASSING	34.0561	-118.2375
3	240119968	10/14/2024	650	Central	TRESPASSING	34.0561	-118.2375
4	241000734	10/14/2024	140	West Valley	BATTERY - SIMPLE ASS...	34.1938	-118.5422
5	241713532	10/14/2024	1530	Devonshire	OTHER	34.2684	-118.5149
6	241514626	10/14/2024	1435	N Hollywood	VEHICLE - STOLEN	34.1649	-118.3725
7	241813867	10/14/2024	830	Southeast	VEHICLE - STOLEN	33.9475	-118.263
8	240410778	10/14/2024	1158	Hollenbeck	SHOPLIFTING - PETTY T...	34.034	-118.1969
9	240119972	10/14/2024	820	Central	TRESPASSING	34.0561	-118.2375
10	241514628	10/14/2024	1447	N Hollywood	VEHICLE - STOLEN	34.2012	-118.4094
11	242112352	10/14/2024	415	Topanga	OTHER	34.2128	-118.6082
12	240119964	10/14/2024	720	Central	THEFT PLAIN - PETTY ($...	34.0561	-118.2375
13	240119997	10/14/2024	1400	Central	SHOPLIFTING - PETTY T...	34.0483	-118.2631
14	241813861	10/14/2024	930	Southeast	VEHICLE - STOLEN	33.9601	-118.2695
15	241813866	10/14/2024	1620	Southeast	VEHICLE - STOLEN	33.9474	-118.2652
16	241414085	10/14/2024	1930	Pacific	VEHICLE - STOLEN	33.9984	-118.4053

Figure 10. Data Before and After Applying "Cluster Data Tool"

4.3. Spatial Selection of Crime Data in USC DPS Area

Using the ***Select By Location*** tool in ArcGIS Pro, crime points within the USC DPS patrol area were isolated for focused analysis. The detailed steps are as follows:

- **Load Layers**: Ensure that the ***final_crime_data.csv*** file is added as a point layer (***final_crime_data_XY***) and displayed on the map. Load the ***UPC_Patrol_Area_1-polygon.shp*** shapefile to define the patrol area.
- **Select By Location Tool**:
  - In the ***Analysis*** tab in ArcGIS Pro, open the ***Tools*** menu and search for ***Select By Location***.
  - **Input Features**: Choose ***final_crime_data_XY***, the layer containing crime data points.
  - **Relationship**: Set to ***Intersects*** to select all points that intersect with the patrol area.
  - **Selecting Features**: Choose ***UPC_Patrol_Area***, the patrol area boundary.
  - **Output Selection Type**: Select ***New selection***.

        - Click ***Run*** to execute the tool, selecting all crime incidents within the patrol area boundary.
    - **Export Selected Data**:
        - Right-click on ***final_crime_data_XY***, go to ***Data > Export Features,*** and save the selected points to a new layer or CSV file.
        - This process creates a dataset limited to the USC DPS patrol area, supporting targeted analysis within the designated zone.

4.4. Temporal Comparison in USC DPS Area

With the crime data filtered to the USC DPS area, a temporal analysis was conducted by dividing the data into two timeframes:

- **Before Incident**: January 1 to April 23, 2024.
- **After Incident**: April 25 to August 15, 2024.

This comparison aimed to identify potential changes in crime rates following the increased security measures.

5. Results

The analysis of crime patterns in Los Angeles and the USC DPS patrol area reveals notable insights:

5.1. Spatial Distribution of Crime Types in Los Angeles City

Figure 11 shows the spatial distribution of crime types across different areas in Los Angeles City. Each region, such as Central, West Valley, and Hollywood, is marked with distinct colors, helping to visualize which crime types are more common in specific neighborhoods. This map provides a citywide view of crime occurrences, enabling an understanding of how crime patterns differ across Los Angeles.

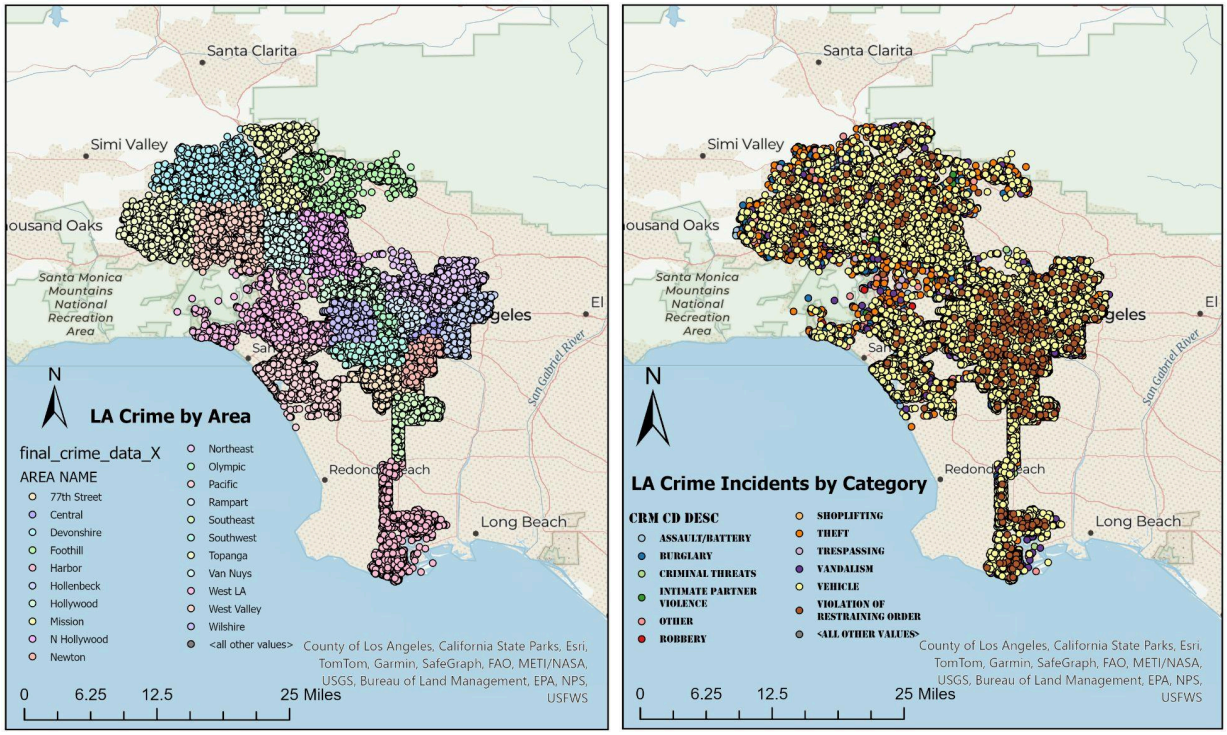


Figure 11. Los Angeles City Crime Distribution by Area and Category

5.2. Crime Density Heatmap for Los Angeles City

Figure 12 illustrates a crime density heatmap for Los Angeles City, with darker colors representing areas of higher crime concentration. This visualization allows easy identification of crime hotspots, highlighting neighborhoods with the highest density of incidents. The heatmap provides valuable insights into where crime reduction efforts might be focused and serves as a basis for comparing crime density before and after the security measures were enforced on April 24.

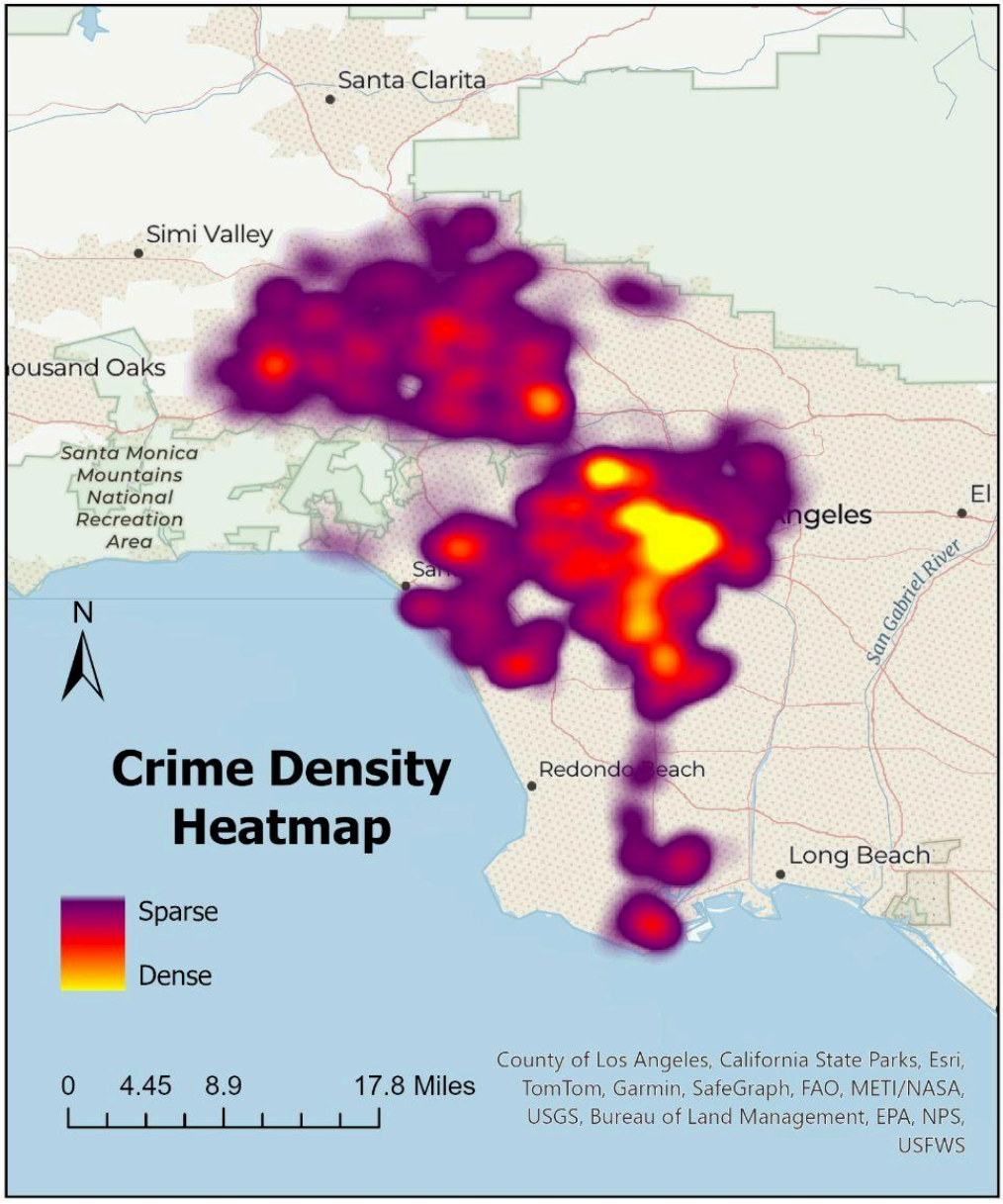


Figure 12. Crime Density Heatmap for Los Angeles City

5.3. Crime Incidents by Category within the USC DPS Patrol Area

Focusing on the USC DPS patrol area, Figure 13 categorizes crime incidents by type, using symbol-based color coding to indicate crime types such as assault, theft, vandalism, and trespassing. This figure helps visualize the specific types of crimes prevalent within the campus environment, allowing the DPS to prioritize resources based on the most common incident categories.

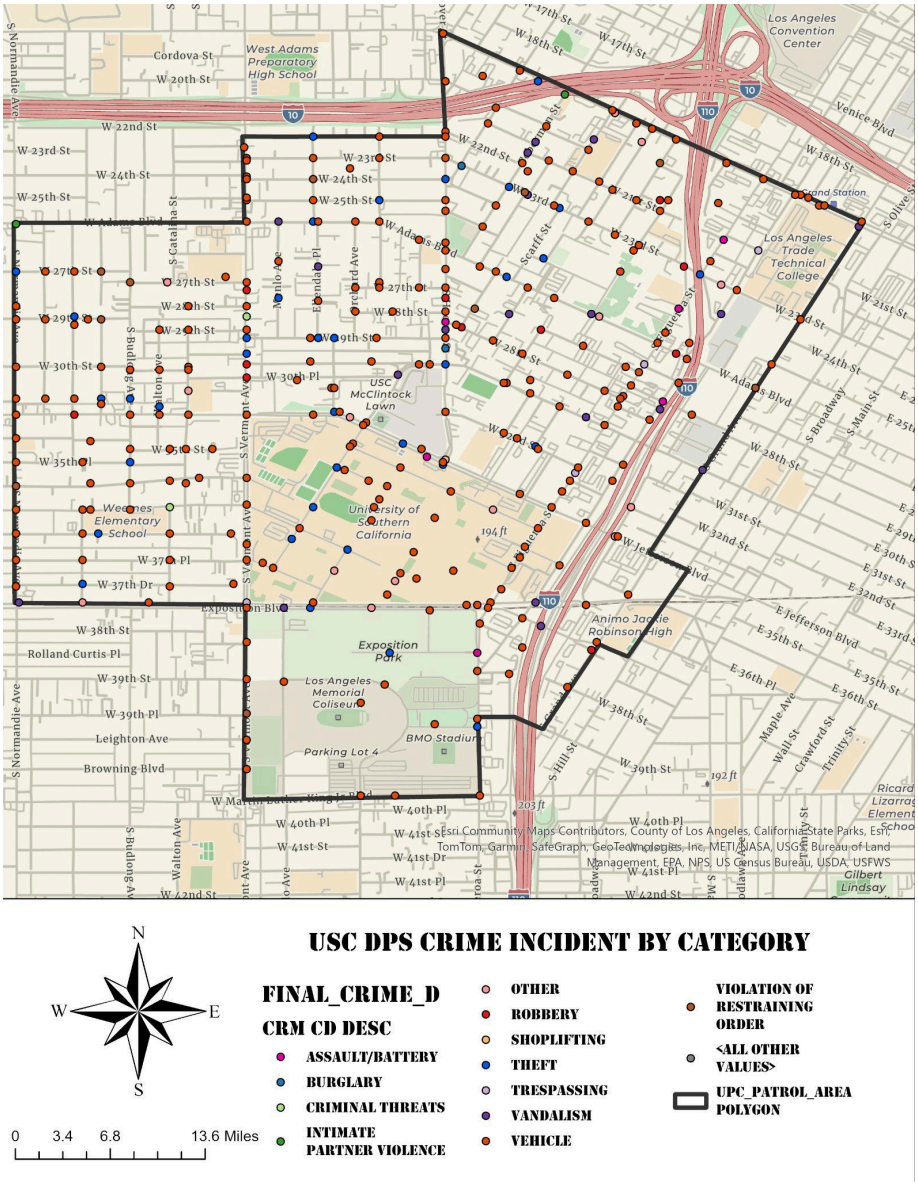


Figure 13. USC DPS Area Crime Incidents by Category

5.4. USC DPS Patrol Area Crime Density Heatmap

Figure 14 presents a heatmap of crime density within the USC DPS patrol area, specifically targeting zones frequented by students and faculty. The heatmap reveals denser clusters of incidents near high-traffic campus locations, which may suggest areas needing additional security patrols or preventive measures.

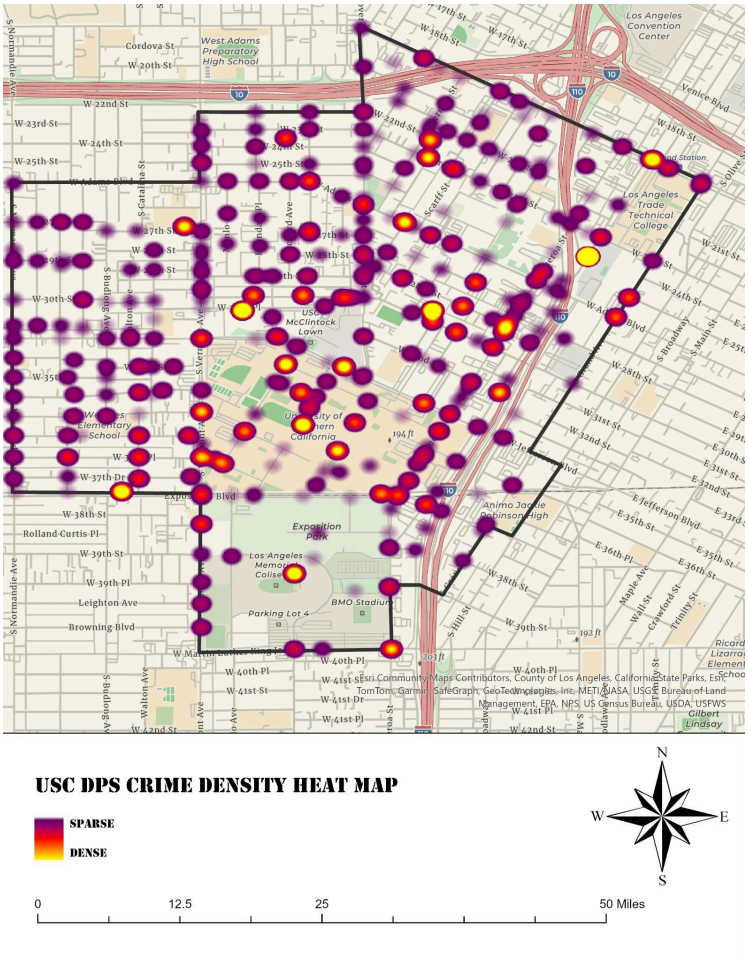


Figure 14. 2024 USC DPS Crime Density Heat Map

5.5. Temporal Comparison of Crime Rates Before and After the April 24 Protest

To evaluate the effectiveness of heightened security measures following the April 24 protest, crime data was divided into two periods: before the protest (January to April) and after (May to August). Figure 15 compares crime rates across Los Angeles City, revealing trends over these two timeframes. This comparison suggests that certain crime types decreased in frequency post-protest, possibly due to increased citywide awareness and enforcement efforts.

In parallel, Figure 16 compares crime rates within the USC DPS patrol area before and after the protest. The figure indicates a noticeable reduction in specific crimes, such as trespassing and vandalism, after the April 24 incident. This reduction aligns with the DPS's goal

of enhancing campus safety, demonstrating that increased security measures may have deterred potential offenders within this zone.

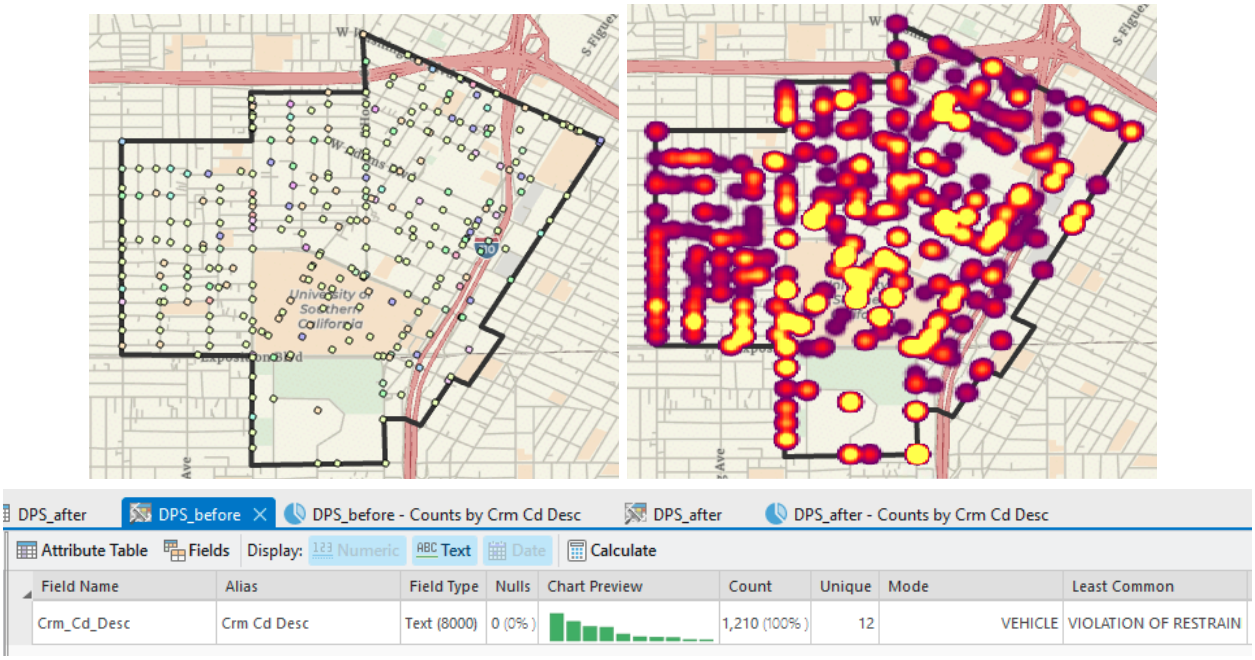


*Figure 15. USC DPS Area Crime Rate **Before** April 24 Protest*

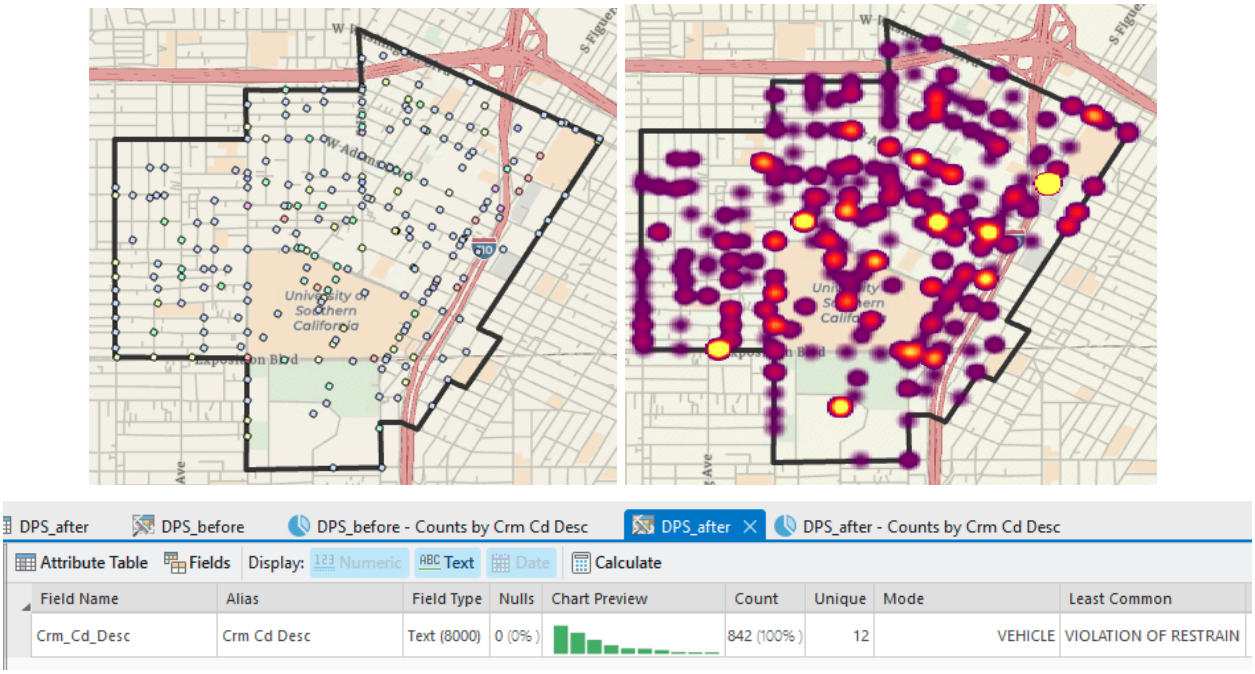


*Figure 16. USC DPS Area Crime Rate **After** April 24 Protest*

5.6. 2024 Top 15 Causes of Crime in Los Angeles City vs. USC DPS Patrol Area

Figure 17 shows pie charts comparing the top 15 causes of crime in both Los Angeles City and the USC DPS patrol area in 2024. In Los Angeles City, vehicle-related crimes and theft are among the most common, while in the USC DPS area, trespassing and vandalism show a higher proportion. This contrast reflects how crime patterns vary based on local context, with campus security facing unique challenges in addressing crimes more relevant to a university environment.

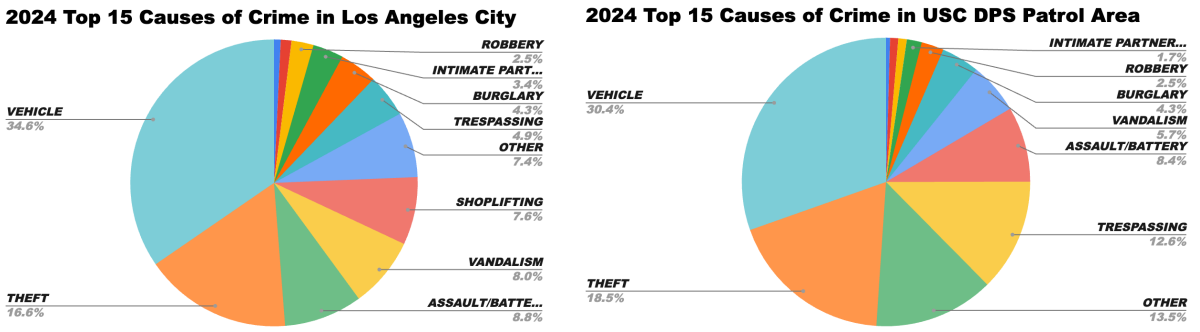


Figure 17. Top 15 Causes of Crime in Los Angeles City and USC DPS Patrol Area in 2024

6. Discussion

The results indicate that the implementation of heightened security measures following the April 24 protest incident had a significant impact on crime patterns, particularly within the USC DPS patrol area. The temporal analysis before and after the incident shows a decline in crimes such as trespassing and vandalism post-protest, suggesting that the increased security presence served as a deterrent.

The visualizations, including the spatial distribution and heatmaps, underscore the importance of targeted security efforts. In Los Angeles City, high-density crime areas—such as those illustrated in Figure 12—highlight neighborhoods that may benefit from enhanced crime

prevention strategies. In the USC DPS patrol area, the focused reduction in specific crimes reflects the effectiveness of targeted security measures.

The ***Cluster Data Tool*** and ***Format DateTime Tool*** facilitated the analysis by consolidating crime categories and enabling temporal analysis. These tools proved essential for managing large datasets and provided a foundation for comparing crime trends across different periods.

7. Conclusion

This project demonstrated the value of custom geospatial tools and spatial analysis techniques in examining crime patterns. By focusing on the impact of security measures following a significant campus event, the analysis provided insights into the effectiveness of these measures in reducing certain types of crime in the USC DPS patrol area. The visualizations and temporal comparisons revealed distinct crime patterns and provided valuable information for future campus security planning.

Future research could expand on this study by analyzing longer timeframes, incorporating additional datasets, and evaluating seasonal trends. Moreover, integrating predictive modeling techniques could help identify potential high-risk periods, allowing security teams to implement proactive crime prevention strategies.

References

ArcGIS Pro Python Reference. "ArcGIS Pro Python Reference Documentation." Esri. Accessed November 4, 2024 https://pro.arcgis.com/en/pro-app/arcpy/main/arcgis-pro-arcpy-reference.htm

Esri Blog. "How to Publish a Web Tool from ArcGIS Pro." Esri. Accessed November 4, 2024 https://www.esri.com/arcgis-blog/products/arcgis-enterprise/sharing-collaboration/how-to-publish-a-web-tool-from-arcgis-pro/

Esri Training Resource. "Updating Real-Time Data Using ArcGIS Python Libraries." Esri. Accessed November 4, 2024 https://www.esri.com/training/catalog/6410bdd94d750615175b1de1/updating-realtime-data-using-arcgis-python-libraries/

Hillsborough County GIS. "Hillsborough County GIS Datasets." Accessed November 4, 2024. https://hcfl.gov/

Los Angeles Open Data Portal. "Crime Data from 2020 to Present." Accessed November 4, 2024 https://data.lacity.org/Public-Safety/Crime-Data-from-2020-to-Present/2nrs-mtv8/about_data

MyGeodata Converter. "Convert KML to Shapefile." Accessed November 4, 2024. https://mygeodata.cloud/converter/kml-to-shp

Python Software Foundation. *Supporting Python 3.* Accessed November 4, 2024. http://python3porting.com/

USC Department of Public Safety. "USC DPS Patrol Area." Accessed November 4, 2024. https://dps.usc.edu/patrol/