An Ultimate Analysis on the The Crown Prosecution Service Dataset

Seyed Vahid Sharifi Fazel

S4217641

Lecturer : Dr. Bhupesh Mishra

June 2023

Install and importing the libraries

In [338…
```r
# Creating a vector of libraries we need
# Install them if you haven't installed them yet
lib = c('readr', 'hash', 'tidyverse', 'data.table', 'zoo', 'dplyr', 'ggmap', 'skimr',
        'devtools', 'visdat', 'DataExplorer', 'inspectdf', 'ggplot2', 'ggcorrplot', 'corrplot', 'reshape2',
       'factoextra', 'e1071', 'forecast', 'mice', 'randomForest', 'reshape2', 'lubridate', 'ggmap', 'GGally',
        'viridis', 'hrbrthemes', 'gganimate', 'caTools', 'caret', 'Rtsne', 'cluster', 'nnet')
```

In [2]:
```r
#Install All the libraries and dependencies we need using the for loop
for (item in lib){
    install.packages(item)
}
```

Note:

Please run the library cells twice in order to prevent potencial errors

In [409…
```r
library(tidyverse)
library(cluster)
library(readr)
library(ggmap)
library(GGally)
library(dplyr)
library(visdat)
library(lubridate)
library(data.table)
library(zoo)
library(tidyverse)
library(ggplot2)
library(ggcorrplot)
library(corrplot)
library(reshape2)
library(factoextra)
library(e1071)
library(randomForest)
library(forecast)
library(mice)
library(skimr)
library(DataExplorer)
library(inspectdf)
library(devtools)
library(glue)
library(repr)
library(viridis)
library(hrbrthemes)
library(gganimate)
library(caTools)
library(caret)
library(Rtsne)
library(nnet)
```

In [410…
```r
# Set the output size of the plots
options(repr.plot.width = 20, repr.plot.height = 15)
```

Import and Merging the datasets

Please Note:

This import method is just working for MacOS. Therefore to run this code on windows, you need to change all forward slashes to back slashes in this code.

In [411…
```r
# Getting current path
current_path <- getwd()
dataset_path <-  paste(current_path, '/Dataset - Assignment',sep = '')
```

In [412…
```r
# Get a list of all the subfolders in the main folder
subfolders <- list.dirs(dataset_path, recursive = FALSE)
```

In [413…
```r
# Initialize an empty list to store dataframes
df_list <- list()
```

In [414…
```r
# Loop through each subfolder
for (subfolder in subfolders) {
  # Get a list of all the csv files in the subfolder
  csv_files <- list.files(subfolder, pattern = "^principal_offence_category_.*\\.csv$", full.names = TRUE)

  # Loop through each csv file
  for (csv_file in csv_files) {
    # Read the csv file
```

```
    df <- read.csv(csv_file, stringsAsFactors = FALSE)

    # Extract the month from the filename
    splitted_path <- strsplit(csv_file,split = '/')
    file_name <- splitted_path[[1]][length(splitted_path[[1]])]
    month <- tolower(substr(file_name, 28, 30))

    # Add the "month" and "folder" columns to the dataframe
    df$month <- month
    df$year <- substr(subfolder, nchar(subfolder) - 3,nchar(subfolder))

    # Append the dataframe to the list
    df_list[[length(df_list) + 1]] <- df
  }
}
```

In [415…

```
# Concatenate all the dataframes into a single dataframe
all_data <- do.call(rbind, df_list)
head(all_data)
```

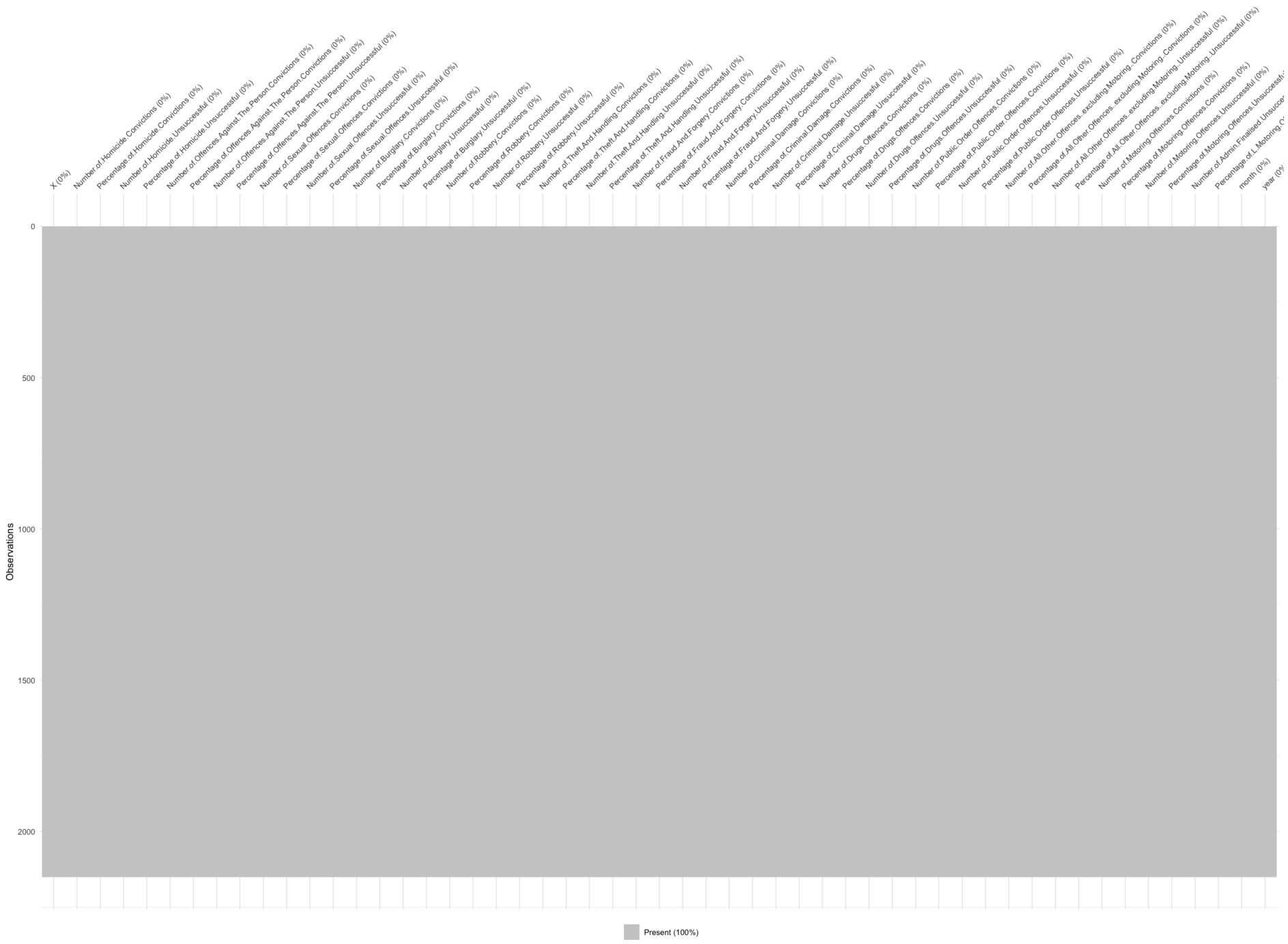
	X	Number.of.Homicide.Convictions	Percentage.of.Homicide.Convictions	Number.of.Homicide.Unsuccessful	Percentage.of.Homicide.Unsuccessful	N
	<chr>	<int>	<chr>	<int>	<chr>	
1	National	81	85.3%	14	14.7%	
2	Avon and Somerset	1	100.0%	0	0.0%	
3	Bedfordshire	0	-	0	-	
4	Cambridgeshire	0	-	0	-	
5	Cheshire	1	50.0%	1	50.0%	
6	Cleveland	0	-	0	-	

Initial Descriptive anlysis for conprehending data structure

Checking for Null values

In [416…

```
vis_miss(all_data)
```



In [417…

```
vis_dat(all_data)
```

Analysis:

Considering these two graphs, our dataset contains no null values.

Data Cleansing

Get a glimpse of our data

In [418…

```
glimpse(all_data)
```

```
Rows: 2,150
Columns: 53
$ X                                                          <chr> "Na…
$ Number.of.Homicide.Convictions                             <int> 81,…
$ Percentage.of.Homicide.Convictions                         <chr> "85…
$ Number.of.Homicide.Unsuccessful                            <int> 14,…
$ Percentage.of.Homicide.Unsuccessful                        <chr> "14…
$ Number.of.Offences.Against.The.Person.Convictions          <chr> "7,…
$ Percentage.of.Offences.Against.The.Person.Convictions      <chr> "74…
$ Number.of.Offences.Against.The.Person.Unsuccessful         <chr> "2,…
$ Percentage.of.Offences.Against.The.Person.Unsuccessful     <chr> "25…
$ Number.of.Sexual.Offences.Convictions                      <chr> "69…
$ Percentage.of.Sexual.Offences.Convictions                  <chr> "72…
$ Number.of.Sexual.Offences.Unsuccessful                     <int> 269…
$ Percentage.of.Sexual.Offences.Unsuccessful                 <chr> "27…
$ Number.of.Burglary.Convictions                             <chr> "1,…
$ Percentage.of.Burglary.Convictions                         <chr> "86…
$ Number.of.Burglary.Unsuccessful                            <int> 226…
$ Percentage.of.Burglary.Unsuccessful                        <chr> "13…
$ Number.of.Robbery.Convictions                              <int> 517…
$ Percentage.of.Robbery.Convictions                          <chr> "81…
$ Number.of.Robbery.Unsuccessful                             <int> 116…
$ Percentage.of.Robbery.Unsuccessful                         <chr> "18…
$ Number.of.Theft.And.Handling.Convictions                   <chr> "10…
$ Percentage.of.Theft.And.Handling.Convictions               <chr> "92…
$ Number.of.Theft.And.Handling.Unsuccessful                  <chr> "84…
$ Percentage.of.Theft.And.Handling.Unsuccessful              <chr> "7.…
$ Number.of.Fraud.And.Forgery.Convictions                    <chr> "66…
$ Percentage.of.Fraud.And.Forgery.Convictions                <chr> "86…
$ Number.of.Fraud.And.Forgery.Unsuccessful                   <int> 108…
$ Percentage.of.Fraud.And.Forgery.Unsuccessful               <chr> "14…
$ Number.of.Criminal.Damage.Convictions                      <chr> "2,…
$ Percentage.of.Criminal.Damage.Convictions                  <chr> "85…
$ Number.of.Criminal.Damage.Unsuccessful                     <int> 391…
$ Percentage.of.Criminal.Damage.Unsuccessful                 <chr> "14…
$ Number.of.Drugs.Offences.Convictions                       <chr> "4,…
$ Percentage.of.Drugs.Offences.Convictions                   <chr> "94…
$ Number.of.Drugs.Offences.Unsuccessful                      <int> 279…
$ Percentage.of.Drugs.Offences.Unsuccessful                  <chr> "5.…
$ Number.of.Public.Order.Offences.Convictions                <chr> "3,…
$ Percentage.of.Public.Order.Offences.Convictions            <chr> "84…
$ Number.of.Public.Order.Offences.Unsuccessful               <int> 654…
$ Percentage.of.Public.Order.Offences.Unsuccessful           <chr> "15…
$ Number.of.All.Other.Offences..excluding.Motoring..Convictions      <chr> "2,…
$ Percentage.of.All.Other.Offences..excluding.Motoring..Convictions  <chr> "83…
$ Number.of.All.Other.Offences..excluding.Motoring..Unsuccessful     <int> 513…
$ Percentage.of.All.Other.Offences..excluding.Motoring..Unsuccessful <chr> "16…
$ Number.of.Motoring.Offences.Convictions                    <chr> "8,…
$ Percentage.of.Motoring.Offences.Convictions                <chr> "86…
$ Number.of.Motoring.Offences.Unsuccessful                   <chr> "1,…
$ Percentage.of.Motoring.Offences.Unsuccessful               <chr> "13…
$ Number.of.Admin.Finalised.Unsuccessful                     <chr> "71…
$ Percentage.of.L.Motoring.Offences.Unsuccessful             <chr> "10…
$ month                                                      <chr> "ap…
$ year                                                       <chr> "20…
```

Dropping the percentage columns as they are useless

Analysis:

As can be seen, the columns which indicate the portion of unsuccessful crimes or conviction ones contain useless data. In fact, we can easily calculate these percentages by a simple division. These proportions are nothing than the number of convictions or unsuccessful crimes, divided by sum of those two categories. Here is the formula to calculate percentage.

Formulas:

$$Total\ number\ of\ X = Number\ of\ X\ Convictions + Number\ of\ X\ Unsuccessful$$

$$Percentage\ of\ Homicide\ Convictions = \frac{Number\ of\ X\ Convictions}{Total\ number\ of\ X}$$

$$Percentage\ of\ Homicide\ Unsuccessful = \frac{Number\ of\ X\ Unsuccessful}{Total\ number\ of\ X}$$

So basically, by dropping the percentage columns we free up almost %50 of ram memory as almost half of the columns belong to these percentages.

In [419…

```r
drop_percentage_columns <- function(dataframe) {
  dataframe <- dataframe %>% select(-starts_with('Per'))
  return(dataframe)
}

all_data <- drop_percentage_columns(all_data)
head(all_data)
```

	X	Number.of.Homicide.Convictions	Number.of.Homicide.Unsuccessful	Number.of.Offences.Against.The.Person.Convictions	Number.of.Offences.Agai
	<chr>	<int>	<int>	<chr>	
1	National	81	14	7,805	
2	Avon and Somerset	1	0	167	
3	Bedfordshire	0	0	69	
4	Cambridgeshire	0	0	99	
5	Cheshire	1	1	140	
6	Cleveland	0	0	85	

Dropping extera characters of beginning of the column names

Analysis:

We know that our columns represent the number of convictions or unsuccessful crimes in each category. Therefore it is not necessary to mention it in every column names that this column refer to the number of a category. As can be seen in our dataframe, the column names start with the "Number.of.", therefore by dropping this part from columns containing these numerics we make the column more handy.

In [420…
```r
# Create a function for dropping extera characters
drop_extera_chars <- function(df, except_cols) {
  colnames(df) <- ifelse(colnames(df) %in% except_cols,
                         colnames(df),
                         substring(colnames(df), 11))
  return(df)
}
```

In [421…
```r
# Deploy it and update the dataset
all_data <- drop_extera_chars(all_data, c('year', 'month'))
head(all_data)
```

		Homicide.Convictions	Homicide.Unsuccessful	Offences.Against.The.Person.Convictions	Offences.Against.The.Person.Unsuccessful	Sexual.Offen
	<chr>	<int>	<int>	<chr>	<chr>	
1	National	81	14	7,805	2,722	
2	Avon and Somerset	1	0	167	45	
3	Bedfordshire	0	0	69	23	
4	Cambridgeshire	0	0	99	23	
5	Cheshire	1	1	140	47	
6	Cleveland	0	0	85	41	

Lower case all the column names

Analysis:

Regarding our dataset, column names contain both capital and small letters. For further analysis, we might need to access columns by typing their names. So we have to always consider the capitalization of some specific words. However, if we just convert all the letters to lowercase, working with data would be way easier, also we will prevent potential mistakes.

In [422…
```r
# Create a function to lower case column names
lower_columns <- function(dataframe) {
  colnames(dataframe) <- tolower(colnames(dataframe))
    return (dataframe)
}
```

In [423…
```r
# Deploy it
all_data <- lower_columns(all_data)
```

In [424…
```r
head(all_data)
```

		homicide.convictions	homicide.unsuccessful	offences.against.the.person.convictions	offences.against.the.person.unsuccessful	sexual.offences.
	<chr>	<int>	<int>	<chr>	<chr>	
1	National	81	14	7,805	2,722	
2	Avon and Somerset	1	0	167	45	
3	Bedfordshire	0	0	69	23	
4	Cambridgeshire	0	0	99	23	
5	Cheshire	1	1	140	47	
6	Cleveland	0	0	85	41	

Convert dots to underscore in column names

Analysis:

Using "." is not common for datasets. Also "." usually have many different functionalities in programming languages, therefore to prevent potential mistakes, also to follow the standard template for namiing the column names, we need to change the letter "." to underscore "_".

In [425…
```r
# Create a function for it
convert_dots_to_underscores <- function(dataframe) {
  col_names <- colnames(dataframe)
  modified_col_names <- gsub("\\.", "_", col_names)
  colnames(dataframe) <- modified_col_names
  return(dataframe)
}
```

In [426…
```r
# Deploy it
all_data = convert_dots_to_underscores(all_data)
```

In [427…
```r
head(all_data)
```

		homicide_convictions	homicide_unsuccessful	offences_against_the_person_convictions	offences_against_the_person_unsuccessful	sexual_offe
	<chr>	<int>	<int>	<chr>	<chr>	
1	National	81	14	7,805	2,722	
2	Avon and Somerset	1	0	167	45	
3	Bedfordshire	0	0	69	23	
4	Cambridgeshire	0	0	99	23	
5	Cheshire	1	1	140	47	
6	Cleveland	0	0	85	41	

Dropping extera chars of ending of the column names

Analysis:

looking at the dataframe, the column names belong to conviction group have the letter "_conviction" at the end, while the other group has the "_unsuccessful". it makes the column names too long, also when we want to visualize our data, there might be some problem when we want to use these column names as the axis ticks. Therefore, it is better to drop these extra characters. Instead, we add "_un" for the unsuccessful category. As a result, columns with just the category names refer to the convictions, while the other refer to the unsuccessful group.

In [428…
```r
# Create a function for it
drop_extera_chars2 <- function(all_data) {
  col_names <- colnames(all_data)
  updated_col_names <- col_names

  for (i in 1:length(col_names)) {
    if (grepl("_convictions", col_names[i])) {
      updated_col_names[i] <- gsub("_convictions", "", col_names[i])
    } else {
      updated_col_names[i] <- gsub("_unsuccessful", "_un", col_names[i])
    }
  }

  colnames(all_data) <- updated_col_names
  return(all_data)
}
```

In [429…
```r
# Deploy it
all_data <- drop_extera_chars2(all_data)
```

In [430…
```r
head(all_data)
```

		homicide	homicide_un	offences_against_the_person	offences_against_the_person_un	sexual_offences	sexual_offences_un	burglary	burglar
	<chr>	<int>	<int>	<chr>	<chr>	<chr>	<int>	<chr>	
1	National	81	14	7,805	2,722	698	269	1,470	
2	Avon and Somerset	1	0	167	45	36	8	37	
3	Bedfordshire	0	0	69	23	5	1	16	
4	Cambridgeshire	0	0	99	23	6	3	8	
5	Cheshire	1	1	140	47	17	3	26	
6	Cleveland	0	0	85	41	11	4	25	

renaming the column contain areas from '' to 'area'

Analysis:

Our dataset contains a column that refers to the areas of the crimes. However, the name of that column is irrelevant to its purpose. So, we rename it to the area that is more meaningful.

In [431…
```r
# Create a function to rename column
rename_first_column_to_area <- function(all_data){
  names(all_data)[1] <- 'area'
  return(all_data)
}

# Deploy it
```

```
all_data <- rename_first_column_to_area(all_data)
head(all_data)
```

	area	homicide	homicide_un	offences_against_the_person	offences_against_the_person_un	sexual_offences	sexual_offences_un	burglary	burglar
	<chr>	<int>	<int>	<chr>	<chr>	<chr>	<int>	<chr>	
1	National	81	14	7,805	2,722	698	269	1,470	
2	Avon and Somerset	1	0	167	45	36	8	37	
3	Bedfordshire	0	0	69	23	5	1	16	
4	Cambridgeshire	0	0	99	23	6	3	8	
5	Cheshire	1	1	140	47	17	3	26	
6	Cleveland	0	0	85	41	11	4	25	

Checking for string inconsistencies in month

Analysis:

Although we obtained the month names from the three initial charagters of each month name, it could be possible to have some string inconsistencies due to the capitalization of the months, therefore, here we check if we have any string inconsistencies fo the month name ot not in our dataset

In [432…

```
# Create a function to check string inconsistencies
string_inconsistency_checker <- function(all_data){
  if(length(unique(all_data$month)) == 12){
  print('There is no string inconsistencies in the month column')
  print('The unique values for months are :')
  print(unique(all_data$month))
  }
}

# Deploy it
string_inconsistency_checker(all_data)
```

```
[1] "There is no string inconsistencies in the month column"
[1] "The unique values for months are :"
 [1] "apr" "aug" "dec" "feb" "jan" "jul" "jun" "mar" "may" "nov" "oct" "sep"
```

Creating time stamp using year and month column

Analysis:

For further analysis, we need to combine both month and years together to generate a new column which is named date. Date column contains the timestamp for each row. It helps categories our rows better for further analysis.

In [433…

```
# Defining our function for it
create_time_stamp <- function(all_data){
  time_stamp <- as.Date(paste(all_data$month, all_data$year, "01", sep = "-"), format = "%B-%Y-%d")
  all_data$date <- as.Date(time_stamp)
  return(all_data)
}
```

In [434…

```
# Running our function
all_data = create_time_stamp(all_data)
head(all_data)
```

	area	homicide	homicide_un	offences_against_the_person	offences_against_the_person_un	sexual_offences	sexual_offences_un	burglary	burglar
	<chr>	<int>	<int>	<chr>	<chr>	<chr>	<int>	<chr>	
1	National	81	14	7,805	2,722	698	269	1,470	
2	Avon and Somerset	1	0	167	45	36	8	37	
3	Bedfordshire	0	0	69	23	5	1	16	
4	Cambridgeshire	0	0	99	23	6	3	8	
5	Cheshire	1	1	140	47	17	3	26	
6	Cleveland	0	0	85	41	11	4	25	

Find out which instance contains null value

In [435…

```
# Defining our function for it
null_finder <- function(all_data){
  null_location <- which(is.na(all_data), arr.ind=TRUE)
  return(null_location)
}
```

In [436…

```
# Running our function
null_coordinates <- null_finder(all_data)
null_coordinates
```

A matrix: 0
× 2 of type
int

row	col

Check for missing data

In [437…
```
# Defining our function for it
check_for_null <- function(all_data){
  if (sum(is.na(all_data)) != 0){
    null_coordinates <- null_finder(all_data)
    glue('We have -{sum(is.na(all_data))}- null value in our dataset. The row number of this value is -{null_coordinates[1]}-')
  }
  else{
    print('There is no null value')
  }
}
```

In [438…
```
# Running our function
check_for_null(all_data)
```

```
[1] "There is no null value"
```

Analysis:

As is shown, ourdataset contains no null value which means we need no further process to exclude the null values.

In [439…
```
head(all_data)
```

	area	homicide	homicide_un	offences_against_the_person	offences_against_the_person_un	sexual_offences	sexual_offences_un	burglary	burglar
	<chr>	<int>	<int>	<chr>	<chr>	<chr>	<int>	<chr>	
1	National	81	14	7,805	2,722	698	269	1,470	
2	Avon and Somerset	1	0	167	45	36	8	37	
3	Bedfordshire	0	0	69	23	5	1	16	
4	Cambridgeshire	0	0	99	23	6	3	8	
5	Cheshire	1	1	140	47	17	3	26	
6	Cleveland	0	0	85	41	11	4	25	

Another glimpse to the dataset and datatype

In [440…
```
glimpse(all_data)
```

```
Rows: 2,150
Columns: 29
$ area                                  <chr> "National", "Avon and Somer…
$ homicide                              <int> 81, 1, 0, 0, 1, 0, 0, 0, 1,…
$ homicide_un                           <int> 14, 0, 0, 0, 1, 0, 0, 0, 0,…
$ offences_against_the_person           <chr> "7,805", "167", "69", "99",…
$ offences_against_the_person_un        <chr> "2,722", "45", "23", "23", …
$ sexual_offences                       <chr> "698", "36", "5", "6", "17"…
$ sexual_offences_un                    <int> 269, 8, 1, 3, 3, 4, 1, 6, 4…
$ burglary                              <chr> "1,470", "37", "16", "8", "…
$ burglary_un                           <int> 226, 2, 1, 0, 3, 10, 1, 3, …
$ robbery                               <int> 517, 9, 4, 6, 1, 5, 1, 8, 6…
$ robbery_un                            <int> 116, 3, 0, 1, 0, 2, 0, 3, 0…
$ theft_and_handling                    <chr> "10,045", "266", "98", "107…
$ theft_and_handling_un                 <chr> "840", "21", "9", "10", "4"…
$ fraud_and_forgery                     <chr> "666", "11", "8", "7", "16"…
$ fraud_and_forgery_un                  <int> 108, 0, 2, 0, 2, 2, 0, 9, 0…
$ criminal_damage                       <chr> "2,259", "54", "20", "21", …
$ criminal_damage_un                    <int> 391, 6, 6, 1, 9, 8, 2, 7, 6…
$ drugs_offences                        <chr> "4,536", "135", "45", "40",…
$ drugs_offences_un                     <int> 279, 2, 2, 2, 10, 7, 2, 9, …
$ public_order_offences                 <chr> "3,549", "68", "29", "45", …
$ public_order_offences_un              <int> 654, 11, 6, 9, 7, 27, 2, 4,…
$ all_other_offences__excluding_motoring_    <chr> "2,640", "66", "11", "6", "…
$ all_other_offences__excluding_motoring__un <int> 513, 16, 6, 2, 6, 5, 1, 15,…
$ motoring_offences                     <chr> "8,283", "188", "40", "79",…
$ motoring_offences_un                  <chr> "1,314", "37", "5", "6", "1…
$ admin_finalised_un                    <chr> "718", "24", "16", "4", "1"…
$ month                                 <chr> "apr", "apr", "apr", "apr",…
$ year                                  <chr> "2014", "2014", "2014", "20…
$ date                                  <date> 2014-04-01, 2014-04-01, 20…
```

Sort the dataframe by the 'date' column in ascending order

In [441…
```
# Creating the function
order_by_date <- function(all_data){
all_data <- all_data[order(all_data$date), ]
}
```

In [442…
```
# Running the function
order_by_date(all_data)
```

```
head(all_data)
```

	area <chr>	homicide <int>	homicide_un <int>	offences_against_the_person <chr>	offences_against_the_person_un <chr>	sexual_offences <chr>	sexual_offences_un <int>	burglary <chr>	burglar
1	National	81	14	7,805	2,722	698	269	1,470	
2	Avon and Somerset	1	0	167	45	36	8	37	
3	Bedfordshire	0	0	69	23	5	1	16	
4	Cambridgeshire	0	0	99	23	6	3	8	
5	Cheshire	1	1	140	47	17	3	26	
6	Cleveland	0	0	85	41	11	4	25	

Visualizing the missing month

Analysis:

Considering our database, we don't have to access all months during the priod from 2014 to 2018. In fact, some month are missing which might affect our future analysis. Consequently, we need to detect these missing months to prevent bias in future. To do so, we will visualize the missing month and note the ones that are missed i our ddataset, It helps us to take better approaches to deal with these missing months.

In [443…

```
#Creating the vectore of month in dataset
available_month <- sort(unique(all_data$date))

# Convert the dates to the year-month format
year_month <- format(available_month, "%Y-%m")

# Create a table to count the frequency of each year-month
month_counts <- table(year_month)

# Generate a sequence of all months in the desired period
all_months <- seq(as.Date("2014-01-01"), as.Date("2018-12-01"), by = "month")
all_year_month <- format(all_months, "%Y-%m")

# Create a data frame with the full sequence of months and their counts
full_month_counts <- data.frame(year_month = all_year_month, count = 0)
full_month_counts$count <- month_counts[full_month_counts$year_month]

# Identify missing months and set their count to 0
full_month_counts$count[is.na(full_month_counts$count)] <- 0

# Reshape the data for heatmap plot
heatmap_data <- dcast(full_month_counts, year_month ~., value.var = "count")

# Set the output size
options(repr.plot.width = 20, repr.plot.height = 15)

# Create a heatmap plot
ggplot(melt(heatmap_data, id.vars = "year_month"), aes(x = variable, y = year_month)) +
  geom_tile(aes(fill = value != 0), color = "white") +
  scale_fill_manual(values = c("white", "lightblue"), guide = 'none') +
  xlab(" ") +
  ylab("Year-Month") +
  ggtitle("Missing Months Visualization") +
  theme_minimal() +
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1),
    plot.title = element_text(size = 25, face = "bold"),
    axis.title.y = element_text(size = 20),
    axis.text.y = element_text(size = 13)
  )
```

Analysis:

Considering the above visualisation, It can be understood that some month are missing during this 5 years period. Therefore, we should note that if we want to compare the crime rate between different years, we should remove these month in order to have a meaningful comparisson. Here is the list of missing months:

November 2015

February 2016

March 2016

April 2017

May 2017

June 2017

April 2018

May 2018

June 2018

November 2018

Shifting date related column to the left

In [444…
```r
# Create our function
column_sorter <- function(all_data) {
  cols <- colnames(all_data)
  cols <- c(cols[1], cols[(length(cols)-2):length(cols)], cols[2:(length(cols)-3)])
  all_data[, cols]
}
```

In [445…
```r
# Deploy it
all_data <- column_sorter(all_data)
```

In [446…
```r
head(all_data)
```

	area	month	year	date	homicide	homicide_un	offences_against_the_person	offences_against_the_person_un	sexual_offences	sexual_offenc
	<chr>	<chr>	<chr>	<date>	<int>	<int>	<chr>	<chr>	<chr>	
1	National	apr	2014	2014-04-01	81	14	7,805	2,722	698	
2	Avon and Somerset	apr	2014	2014-04-01	1	0	167	45	36	
3	Bedfordshire	apr	2014	2014-04-01	0	0	69	23	5	
4	Cambridgeshire	apr	2014	2014-04-01	0	0	99	23	6	
5	Cheshire	apr	2014	2014-04-01	1	1	140	47	17	
6	Cleveland	apr	2014	2014-04-01	0	0	85	41	11	

In [447…
```
raw_data <- all_data
```

Converting values to integer

Analysis:

Some values in the cells contain characters like "," or ".". So basically, nstead of having integer data type, we are dealing with character datatype, while they are numeric representation of the features. Therefore, we must remove these extra characters and convert these characters to numerics for further machine learning processes.

In [448…
```
# Create our function
char_to_int  <- function(dataframe) {
  for (col in names(dataframe[,5:ncol(dataframe)])) {
    if (is.numeric(dataframe[[col]])) {
      dataframe[[col]] <- gsub(",", "", dataframe[[col]])
      dataframe[[col]] <- as.integer(dataframe[[col]])
    } else if (is.character(dataframe[[col]])) {
      dataframe[[col]] <- as.integer(gsub(",", "", dataframe[[col]]))
    }
  }
  return(dataframe)
}
```

In [449…
```
# Deploy it
all_data <- char_to_int(all_data)
```

In [450…
```
head(all_data)
```

	area	month	year	date	homicide	homicide_un	offences_against_the_person	offences_against_the_person_un	sexual_offences	sexual_offenc
	<chr>	<chr>	<chr>	<date>	<int>	<int>	<int>	<int>	<int>	
1	National	apr	2014	2014-04-01	81	14	7805	2722	698	
2	Avon and Somerset	apr	2014	2014-04-01	1	0	167	45	36	
3	Bedfordshire	apr	2014	2014-04-01	0	0	69	23	5	
4	Cambridgeshire	apr	2014	2014-04-01	0	0	99	23	6	
5	Cheshire	apr	2014	2014-04-01	1	1	140	47	17	
6	Cleveland	apr	2014	2014-04-01	0	0	85	41	11	

Check if the Area are unique

Analysis:

There might be some string inconsistencies in the area column which might result in some mistakes in the future. to prevent these kinds of mistakes, we need to check it and find out if all the areas are consistent or not.

In [451…
```
# Check for string consistency in area
glue("The number of unique area is {length(unique(all_data$area))}")
unique(all_data$area)
```

'The number of unique area is 43'
'National' · 'Avon and Somerset' · 'Bedfordshire' · 'Cambridgeshire' · 'Cheshire' · 'Cleveland' · 'Cumbria' · 'Derbyshire' · 'Devon and Cornwall' · 'Dorset' · 'Durham' · 'Dyfed Powys' · 'Essex' · 'Gloucestershire' · 'GreaterManchester' · 'Gwent' · 'Hampshire' · 'Hertfordshire' · 'Humberside' · 'Kent' · 'Lancashire' · 'Leicestershire' · 'Lincolnshire' · 'Merseyside' · 'Metropolitan and City' · 'Norfolk' · 'Northamptonshire' · 'Northumbria' · 'North Wales' · 'North Yorkshire' · 'Nottinghamshire' · 'South Wales' · 'South Yorkshire' · 'Staffordshire' · 'Suffolk' · 'Surrey' · 'Sussex' · 'Thames Valley' · 'Warwickshire' · 'West Mercia' · 'West Midlands' · 'West Yorkshire' · 'Wiltshire'

Region Segmentation

Analysis:

Although we have a column which is named area and reffers to the location of the data, we need more general representation of datapoints' locations. So, add a new column which is called "regioin" to our dataframe. Region is a geographical segmentation of the areas in our dataset. We devide these areas to three main regions that are "North", "Center", and "South". We label the instances with these three groups. The area which are geographically located in the North will be labeled north, one in the south of coutry will be labeled as the south, national area will be labeled with the all, and other areas will be segmented as the center. This segmentation would be helpful for both visualisation and predictive analysis.

In [452…
```
# devide all areas to three main region

region_list <- list(
  North = c("Cleveland", "Cumbria", "Durham", "Humberside", "Lancashire", "Merseyside", "Northumbria", "North Wales", "North Yorkshir
  Center = c("Bedfordshire", "Cambridgeshire", "Derbyshire", "Leicestershire", "Lincolnshire", "Norfolk", "Northamptonshire", "Nottin
  South = c("Avon and Somerset", "Cheshire", "Devon and Cornwall", "Dorset", "Dyfed Powys", "Essex", "Gloucestershire", "GreaterManch
    All = c("National")
)
```

In [453…
```
# Defining a function for mapping the areas to our region map
region_mapper <- function(all_data, region_list) {
    all_data$region <- ifelse(all_data$area %in% region_list$North, "North",
                        ifelse(all_data$area %in% region_list$Center, "Center",
                               ifelse(all_data$area %in% region_list$South, "South", "All")))
    return (all_data)
    }
```

In [454…
```
all_data <- region_mapper(all_data, region_list)
```

In [455…
```
head(all_data)
```

	area	month	year	date	homicide	homicide_un	offences_against_the_person	offences_against_the_person_un	sexual_offences	sexual_offenc
	<chr>	<chr>	<chr>	<date>	<int>	<int>	<int>	<int>	<int>	
1	National	apr	2014	2014-04-01	81	14	7805	2722	698	
2	Avon and Somerset	apr	2014	2014-04-01	1	0	167	45	36	
3	Bedfordshire	apr	2014	2014-04-01	0	0	69	23	5	
4	Cambridgeshire	apr	2014	2014-04-01	0	0	99	23	6	
5	Cheshire	apr	2014	2014-04-01	1	1	140	47	17	
6	Cleveland	apr	2014	2014-04-01	0	0	85	41	11	

Moving the region column next to the area

In [456…
```
# Create our function
region_shifter <- function(all_data) {
  num_cols <- ncol(all_data)
  new_data <- all_data[, c(num_cols, 1:(num_cols - 1))]
  return(new_data)
}
```

In [457…
```
# Deploy it
all_data <- region_shifter(all_data)
```

In [458…
```
head(all_data)
```

	region	area	month	year	date	homicide	homicide_un	offences_against_the_person	offences_against_the_person_un	sexual_offences	…	cr
	<chr>	<chr>	<chr>	<chr>	<date>	<int>	<int>	<int>	<int>	<int>	…	
1	All	National	apr	2014	2014-04-01	81	14	7805	2722	698	…	
2	South	Avon and Somerset	apr	2014	2014-04-01	1	0	167	45	36	…	
3	Center	Bedfordshire	apr	2014	2014-04-01	0	0	69	23	5	…	
4	Center	Cambridgeshire	apr	2014	2014-04-01	0	0	99	23	6	…	
5	South	Cheshire	apr	2014	2014-04-01	1	1	140	47	17	…	
6	North	Cleveland	apr	2014	2014-04-01	0	0	85	41	11	…	

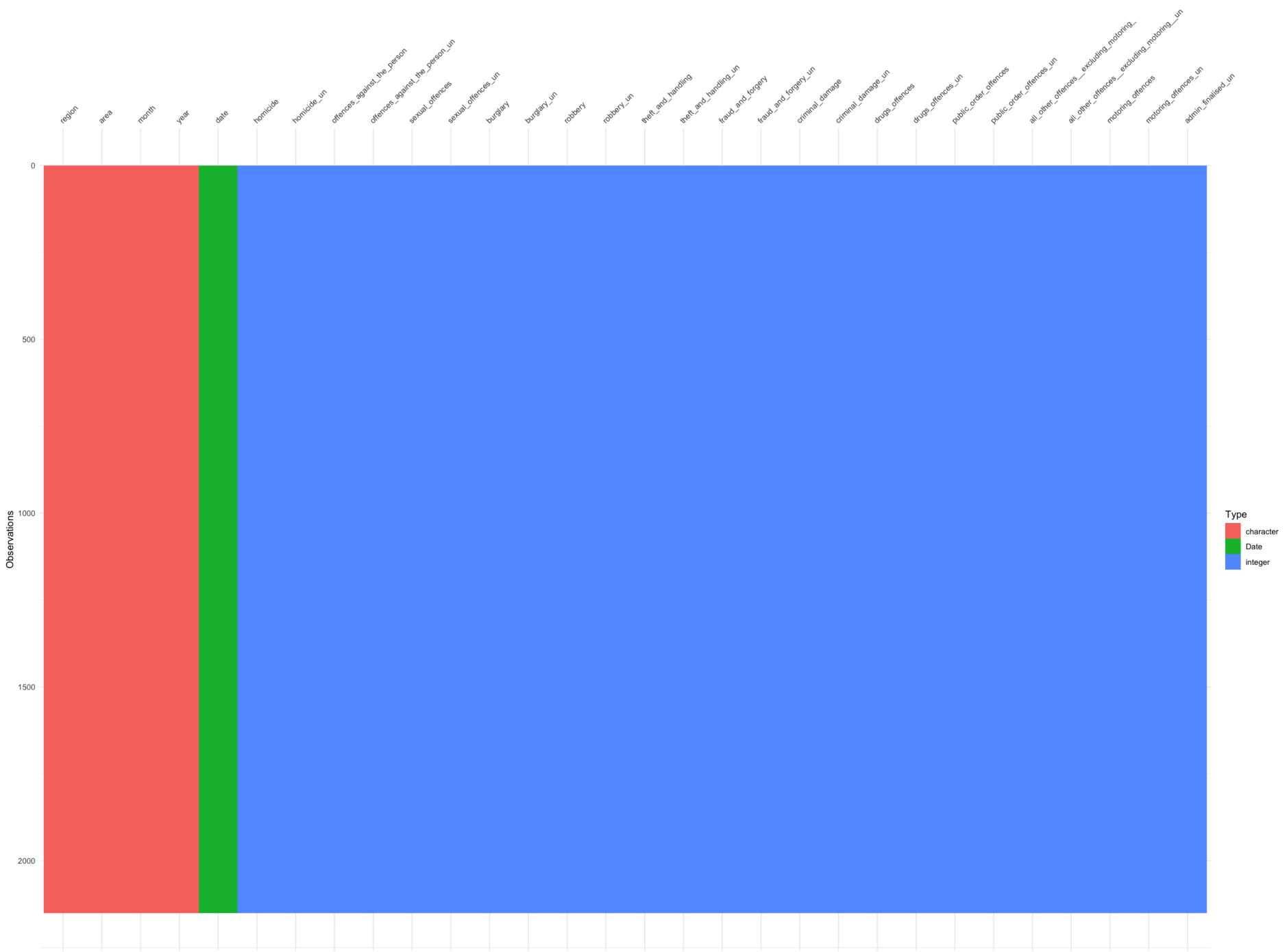
Another glimpse to the dataset and datatypes

In [459…
```
glimpse(all_data)
```

```
Rows: 2,150
Columns: 30
$ region                                    <chr> "All", "South", "Center", "…
$ area                                      <chr> "National", "Avon and Somer…
$ month                                     <chr> "apr", "apr", "apr", "apr",…
$ year                                      <chr> "2014", "2014", "2014", "20…
$ date                                      <date> 2014-04-01, 2014-04-01, 20…
$ homicide                                  <int> 81, 1, 0, 0, 1, 0, 0, 0, 1,…
$ homicide_un                               <int> 14, 0, 0, 0, 1, 0, 0, 0, 0,…
$ offences_against_the_person               <int> 7805, 167, 69, 99, 140, 85,…
$ offences_against_the_person_un            <int> 2722, 45, 23, 23, 47, 41, 1…
$ sexual_offences                           <int> 698, 36, 5, 6, 17, 11, 8, 8…
$ sexual_offences_un                        <int> 269, 8, 1, 3, 3, 4, 1, 6, 4…
$ burglary                                  <int> 1470, 37, 16, 8, 26, 25, 12…
$ burglary_un                               <int> 226, 2, 1, 0, 3, 10, 1, 3, …
$ robbery                                   <int> 517, 9, 4, 6, 1, 5, 1, 8, 6…
$ robbery_un                                <int> 116, 3, 0, 1, 0, 2, 0, 3, 0…
$ theft_and_handling                        <int> 10045, 266, 98, 107, 206, 2…
$ theft_and_handling_un                     <int> 840, 21, 9, 10, 4, 32, 6, 1…
$ fraud_and_forgery                         <int> 666, 11, 8, 7, 16, 6, 5, 11…
$ fraud_and_forgery_un                      <int> 108, 0, 2, 0, 2, 2, 0, 9, 0…
$ criminal_damage                           <int> 2259, 54, 20, 21, 35, 32, 3…
$ criminal_damage_un                        <int> 391, 6, 6, 1, 9, 8, 2, 7, 6…
$ drugs_offences                            <int> 4536, 135, 45, 40, 75, 63, …
$ drugs_offences_un                         <int> 279, 2, 2, 2, 10, 7, 2, 9, …
$ public_order_offences                     <int> 3549, 68, 29, 45, 86, 74, 4…
$ public_order_offences_un                  <int> 654, 11, 6, 9, 7, 27, 2, 4,…
$ all_other_offences__excluding_motoring_   <int> 2640, 66, 11, 6, 50, 28, 64…
$ all_other_offences__excluding_motoring__un <int> 513, 16, 6, 2, 6, 5, 1, 15,…
$ motoring_offences                         <int> 8283, 188, 40, 79, 209, 124…
$ motoring_offences_un                      <int> 1314, 37, 5, 6, 12, 17, 10,…
$ admin_finalised_un                        <int> 718, 24, 16, 4, 1, 10, 12, …
```

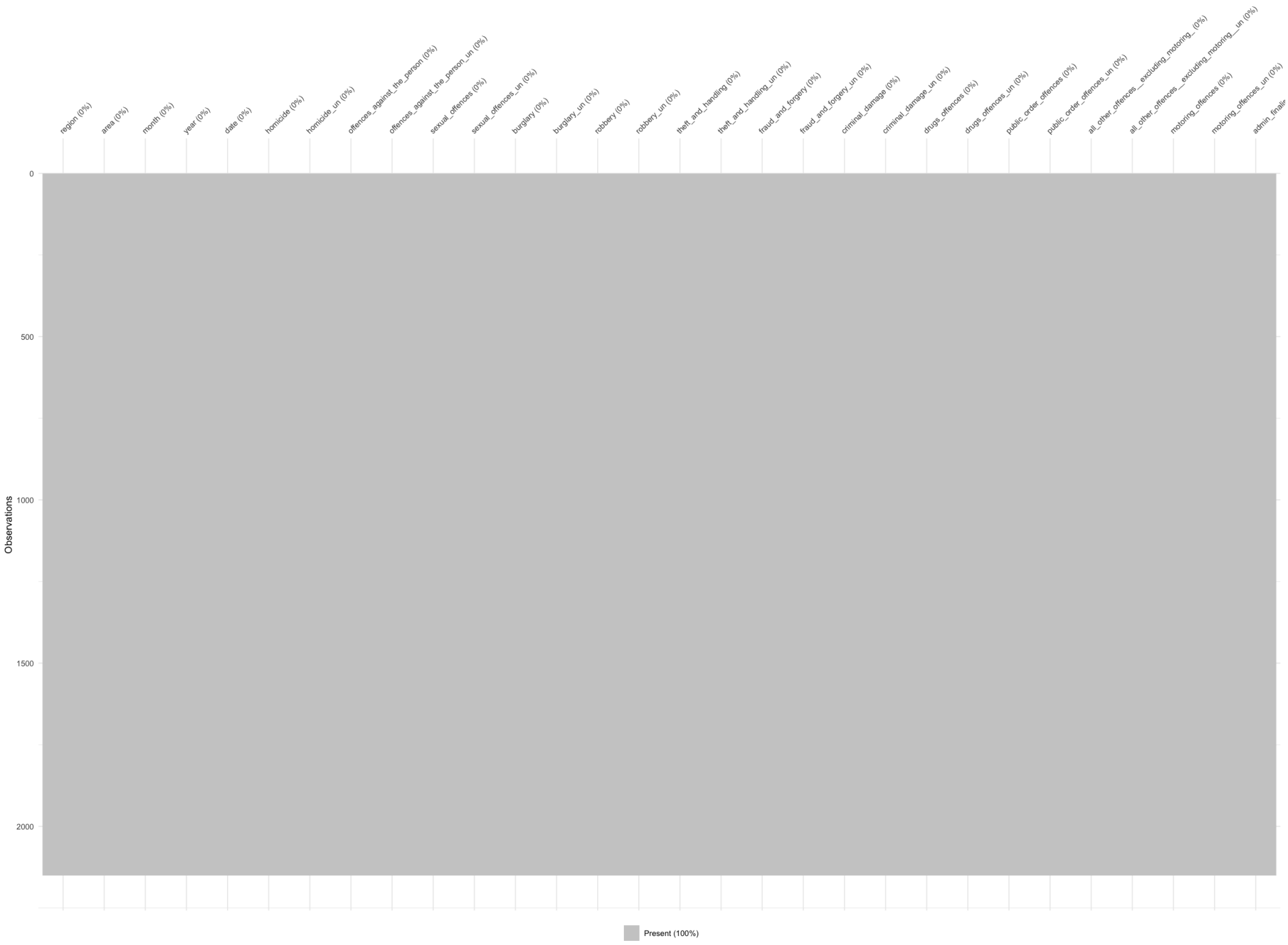
In [460…

```
vis_dat(all_data)
```



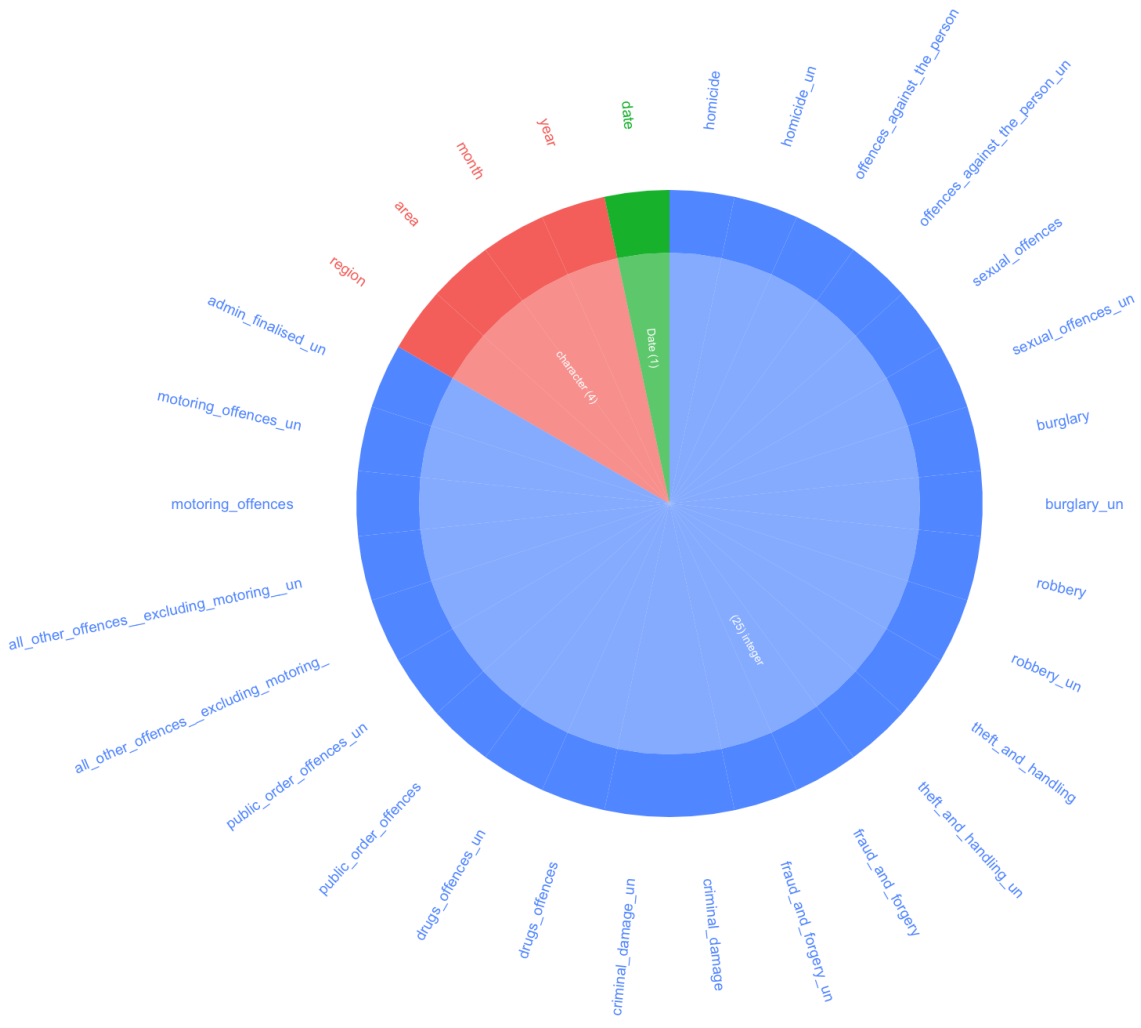
In [461…

```
vis_miss(all_data)
```



In [462…

```
inspect_types(all_data) %>% show_plot()
```

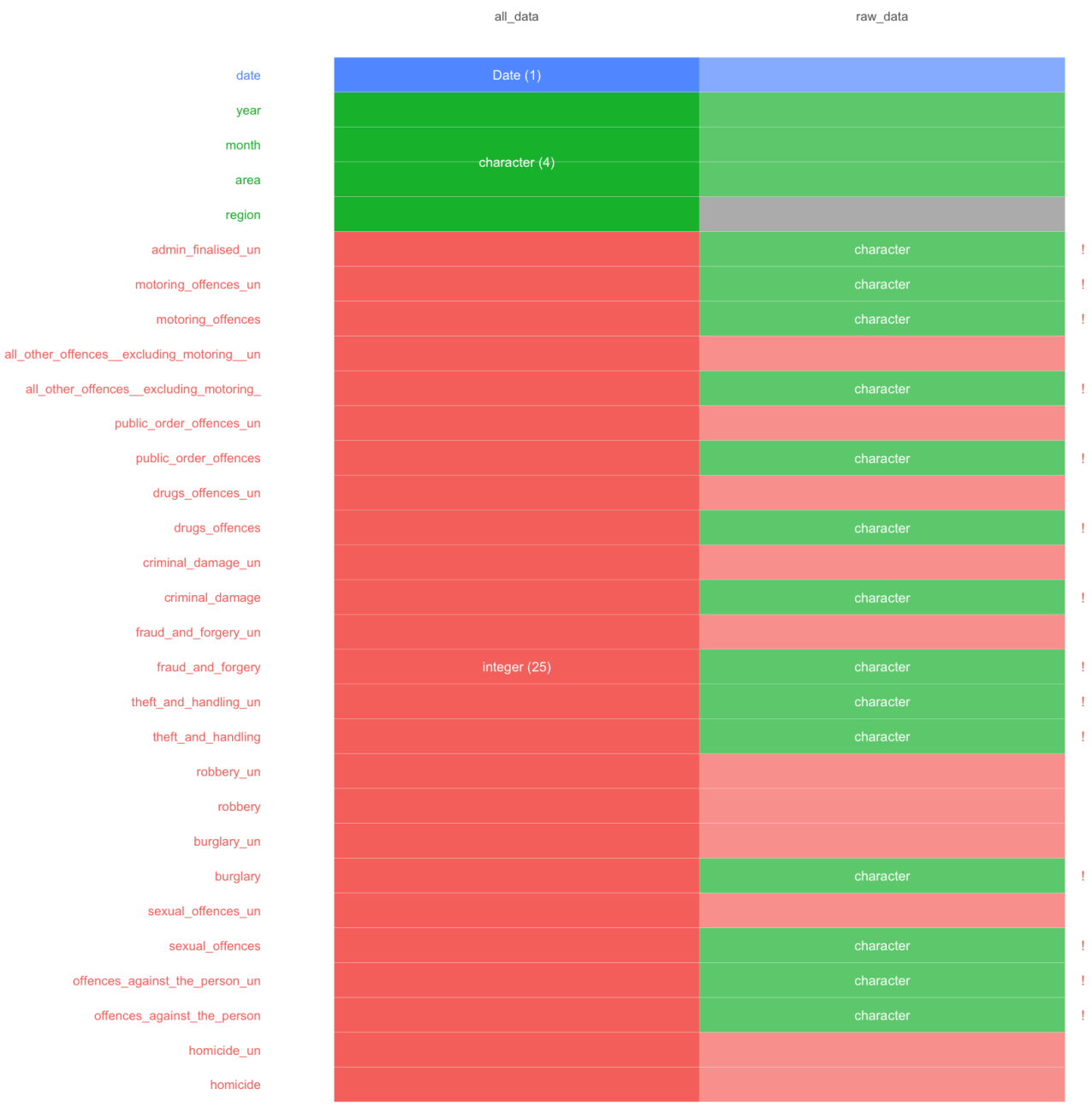


Analysis:

Considering the newvisualization of the data types, we dont have any missing value again which indicates we had no bugs or issues during the data cleaning process. Also, we can see that All the columns which are the numeric representation of our data points are now integers rather than character.

Conparing data type before and after data transformation

In [463…
```
inspect_types(all_data, raw_data) %>% show_plot()
```



Analysis:

This graph is a better illustration of what we have done to our dataset so far. This graph has two columns. All_data column refers to the current cleaned dataset, while raw_data refers to the dataset before the cleansing process. The green cells refer to the data column with the character data type; reds contain numerics inside. Also, we have another category that is the timestamp for our date column. Furthermore, we can see a grey cell for the region in the raw_data dataset that indicates this column didn't exist in our dataset previously.

Splitting the main dataframe

Analysis:

There are two main clusters in our dataset by default. One is for the crimes which were convicted. Another one Is for the unsuccessful. In visualisation and analysis, we should compare the ratio between these categories, so by splitting the main dataset into these two subsets, we can do our analysis for those different categories separately and more accurately.

In [464…
```
# Create a function to split data
splitter <- function(df) {
crime_columns <- !grepl("_un$", colnames(df))
  unsuccesful_columns <- grepl("_un$", colnames(df))
  unsuccesful_columns[1:5] <- TRUE

  return(list(df[, crime_columns], df[, unsuccesful_columns]))
}
```

In [465…
```
# Deploy it, and order new dataframes by date
splitted_dataframes <- splitter(all_data)
crime <- data.frame(splitted_dataframes[1])
ucrime <- data.frame(splitted_dataframes[2])
crime <- crime[order(crime$date), ]
ucrime <- ucrime[order(ucrime$date), ]
all_data <- all_data[order(all_data$date), ]
```

In [466…
```
head(crime)
```

A data.frame: 6 × 17

	region	area	month	year	date	homicide	offences_against_the_person	sexual_offences	burglary	robbery	theft_and_handling	fraud_and_f
	<chr>	<chr>	<chr>	<chr>	<date>	<int>	<int>	<int>	<int>	<int>	<int>	
173	All	National	jan	2014	2014-01-01	51	9087	736	1715	522	11057	
174	South	Avon and Somerset	jan	2014	2014-01-01	0	228	35	49	8	338	
175	Center	Bedfordshire	jan	2014	2014-01-01	0	68	2	7	16	75	
176	Center	Cambridgeshire	jan	2014	2014-01-01	0	101	10	18	6	148	
177	South	Cheshire	jan	2014	2014-01-01	0	170	15	38	10	205	
178	North	Cleveland	jan	2014	2014-01-01	2	119	11	36	3	334	

In [467…
```
head(ucrime)
```

	region	area	month	year	date	homicide_un	offences_against_the_person_un	sexual_offences_un	burglary_un	robbery_un	theft_and_hanc
	<chr>	<chr>	<chr>	<chr>	<date>	<int>	<int>	<int>	<int>	<int>	
173	All	National	jan	2014	2014-01-01	11	2930	286	284	139	
174	South	Avon and Somerset	jan	2014	2014-01-01	0	62	17	1	0	
175	Center	Bedfordshire	jan	2014	2014-01-01	1	29	1	4	7	
176	Center	Cambridgeshire	jan	2014	2014-01-01	0	21	3	4	4	
177	South	Cheshire	jan	2014	2014-01-01	0	40	1	5	0	
178	North	Cleveland	jan	2014	2014-01-01	3	44	6	2	2	

Creating a new column which is the sum of total crimes of different categories

Analysis:

Our dataset contains the number of convictions and unsuccessful crimes separately, but it can't demonstrate the total number of all crimes in a specific month and area. To do so, we need feature engineering. What we need to do is, to sum up all the columns for each individual row. We put the result in a new column which is called "ALL". We create this column for both the crime and ucrime datasets. In the crime dataset, all means the total number of convictions regardless of their category in a specific month for a specific area ea. At the same time, it refers to the total number of unsuccessful crim.

In [468…
```
# Create a function to add a new column
sum_and_add_column <- function(df) {
  # Get the column names from column 5 to the last column
  columns_to_sum <- names(df)[6:ncol(df)]

  # Sum the values in the selected columns
  df$ALL <- rowSums(df[columns_to_sum])

  # Return the modified dataframe
  return(df)
}
```

In [469…
```
# Deploy it on all datasets
crime <- sum_and_add_column(crime)
ucrime <- sum_and_add_column(ucrime)
all_data <- sum_and_add_column(all_data)
```

In [470…
```
head(crime)
```

A data.frame: 6 × 18

	region	area	month	year	date	homicide	offences_against_the_person	sexual_offences	burglary	robbery	theft_and_handling	fraud_and_f
	<chr>	<chr>	<chr>	<chr>	<date>	<int>	<int>	<int>	<int>	<int>	<int>	
173	All	National	jan	2014	2014-01-01	51	9087	736	1715	522	11057	
174	South	Avon and Somerset	jan	2014	2014-01-01	0	228	35	49	8	338	
175	Center	Bedfordshire	jan	2014	2014-01-01	0	68	2	7	16	75	
176	Center	Cambridgeshire	jan	2014	2014-01-01	0	101	10	18	6	148	
177	South	Cheshire	jan	2014	2014-01-01	0	170	15	38	10	205	
178	North	Cleveland	jan	2014	2014-01-01	2	119	11	36	3	334	

In [471…
```
head(ucrime)
```

	region	area	month	year	date	homicide_un	offences_against_the_person_un	sexual_offences_un	burglary_un	robbery_un	theft_and_hanc
	<chr>	<chr>	<chr>	<chr>	<date>	<int>	<int>	<int>	<int>	<int>	
173	All	National	jan	2014	2014-01-01	11	2930	286	284	139	
174	South	Avon and Somerset	jan	2014	2014-01-01	0	62	17	1	0	
175	Center	Bedfordshire	jan	2014	2014-01-01	1	29	1	4	7	
176	Center	Cambridgeshire	jan	2014	2014-01-01	0	21	3	4	4	
177	South	Cheshire	jan	2014	2014-01-01	0	40	1	5	0	
178	North	Cleveland	jan	2014	2014-01-01	3	44	6	2	2	

Yearly Visualisation and Comparison of Crime Rates

Pipeline:

We are going to compare the crime rate in these different years to each other. To do so, we need to sum up all the rows together with respect to their year, then put the result in a row in a new dataframe. However, for some years, we have no data for some months in our dataset, so we should drop the missing months from all years, even those that contain these months, to have the same months for all the years. In this case, we make a bias on our sample as the dropped months are not following a standard randomized sampling method. Therefore, we should note that we can't confidently agree that the trends and patterns in the population follow the patterns in our sample.

Comparing the crime rate in different years

Stacked bar plots:

Successful crimes

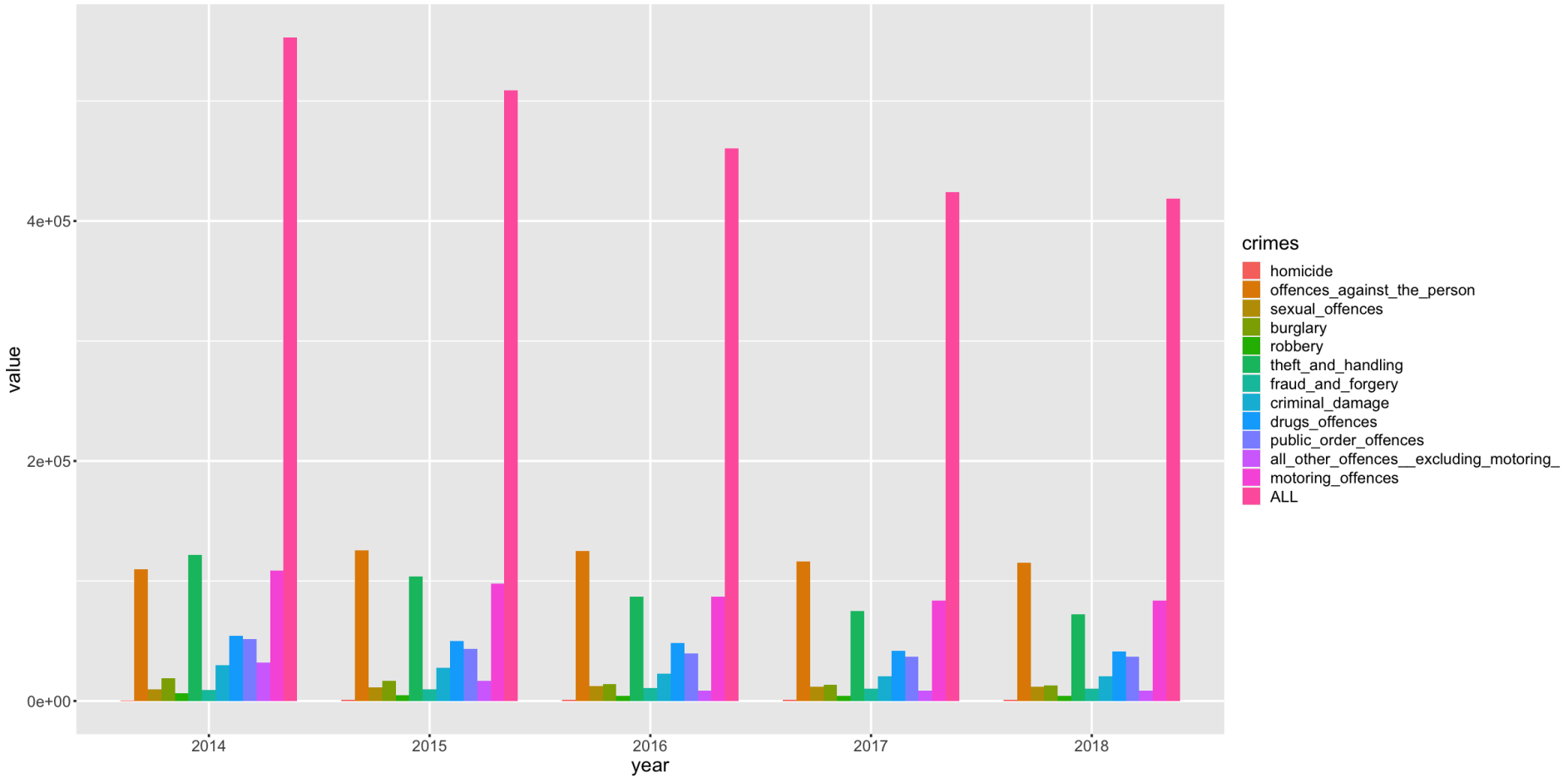
In [472…
```
# Define a function to group our data by the years
group_by_year <- function(dataframe){
  dataframe <- dataframe[,-c(1,2,3,5)]
  dataframe <- group_by(dataframe, year)
  summarise_all(dataframe, list(sum))
}
```

In [473…
```
# Select the months with data available for all the years
splitted_df <- crime[!crime$month %in% c('feb', 'mar', 'apr', 'may','jun', 'nov'), ]
splitted_df <- group_by_year(splitted_df)
head(splitted_df)
```

A tibble: 5 × 14

year	homicide	offences_against_the_person	sexual_offences	burglary	robbery	theft_and_handling	fraud_and_forgery	criminal_damage	drugs_offences	pu
<chr>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	
2014	720	109694	9798	18936	6512	121858	9050	29878	54542	
2015	872	125612	11342	16568	5042	103858	9872	27614	49994	
2016	1084	125054	12584	14296	4314	86844	10782	22858	48068	
2017	1072	116352	12000	13298	4014	75120	10104	20674	41728	
2018	1088	115208	11798	13110	4112	72112	10080	20398	41372	

In [474…
```
# Visualize the stacked bar graph of crime dataset
df <- melt(splitted_df ,  id.vars = 'year', variable.name = 'crimes')
options(repr.plot.width = 20, repr.plot.height =10)
ggplot(df,aes(x = year, y = value)) +
geom_bar(aes(fill = crimes),stat = "identity",position = "dodge", width = 0.8) +
theme(text = element_text(size = 18), element_line(linewidth =1))
```



Analysis:

This chart compares the crime rate of different indexes for years between 2014 and 2018. As is shown, the overall crime rate has decreased moderately from 2014 to 2017. The number of crimes is not significant in this comparison as we dropped February, March, April, May, June, and November month from our dataset because, in some years, we had no data for that month. Nonetheless, considering the ratio of crimes, the overall number of crimes decreased by more than %20 from 2014 to 2017, which is a good thing. However, from 2017 to 2018, this number almost remained steady. It is good to mention that this comparison is just valid for all crimes in our sample, while some categories might follow different patterns. For instance, despite the reduction in an overall number of crimes, the number of sexual offences increased significantly. Also, the number of offences against persons initially increased from 2014 to 2015, and then it gradually declined.

All in all, regarding this graph, apparently, we have an improvement in the overall security of the society in our sample, which might indicated either security organizations outperform in comparison with previous years or we have an improvement in society's overall mindset. However, considering our non-standard way of sampling (dropping five months due to limitations in our dataset), we can't confidently conclude that our population follows the same pattern as our sample.

Unsuccessful crimes

In [475…

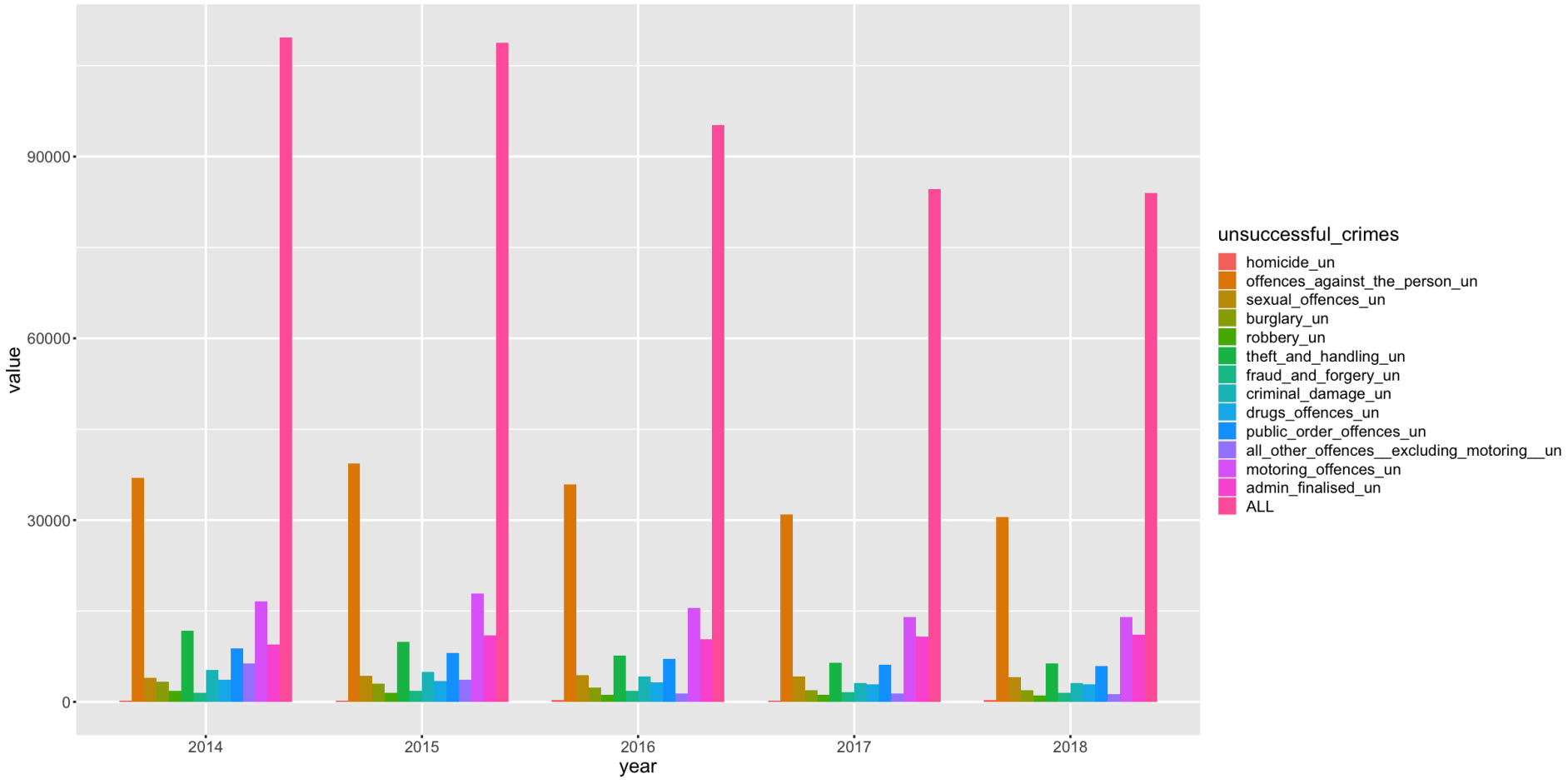
```r
# Select the months with data available for all the years
u_splitted_df <- ucrime[!ucrime$month %in% c('feb', 'mar', 'apr', 'may','jun', 'nov'), ]
u_splitted_df <- group_by_year(u_splitted_df)
head(u_splitted_df)
```

A tibble: 5 × 1

year	homicide_un	offences_against_the_person_un	sexual_offences_un	burglary_un	robbery_un	theft_and_handling_un	fraud_and_forgery_un	criminal_dama
<chr>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	
2014	160	36928	3976	3364	1844	11734	1514	
2015	232	39340	4340	3004	1444	9878	1794	
2016	262	35844	4374	2410	1188	7624	1792	
2017	234	30950	4158	1966	1138	6442	1550	
2018	256	30522	4052	1970	1074	6292	1524	

In [476…

```r
# Visualize the stacked bar graph of ucrime dataset
df <- melt(u_splitted_df ,  id.vars = 'year', variable.name = 'unsuccessful_crimes')
options(repr.plot.width = 20, repr.plot.height =10)
ggplot(df,aes(x = year, y = value)) +
geom_bar(aes(fill = unsuccessful_crimes),stat = "identity",position = "dodge", width = 0.8) +
theme(text = element_text(size = 18), element_line(linewidth =1))
```



Analysis:

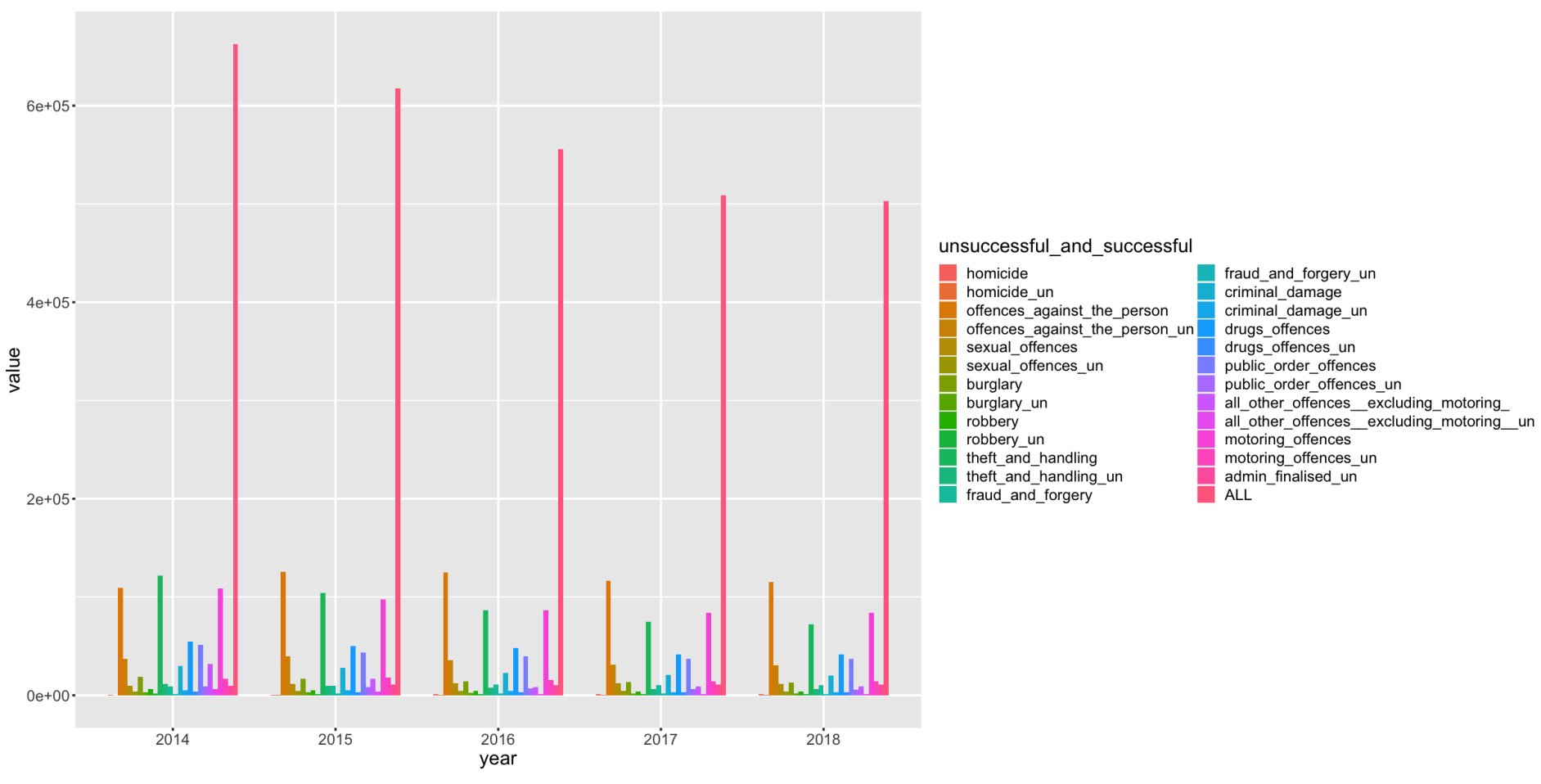
This chart represents the trend of unsuccessful crimes from 2014 to 2018. As is shown, decreasing the number of successful crimes reduces the number of unsuccessful crimes. It means the overall ratio of successful and unsuccessful crimes should be almost the same. To ensure that, we initially need to plot the overall trend of both successful and unsuccessful data. Then we need to evaluate a number for this ratio and track it for different years.

both Successful crimes and Unsuccessful crimes

In [477…
```r
# Select the months with data available for all the years
all_splitted_df <- all_data[!all_data$month %in% c('feb', 'mar', 'apr', 'may','jun', 'nov'), ]
head(group_by_year(all_splitted_df))
```

year	homicide	homicide_un	offences_against_the_person	offences_against_the_person_un	sexual_offences	sexual_offences_un	burglary	burglary_un	robbe
<chr>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<ir
2014	720	160	109694	36928	9798	3976	18936	3364	65
2015	872	232	125612	39340	11342	4340	16568	3004	50
2016	1084	262	125054	35844	12584	4374	14296	2410	43
2017	1072	234	116352	30950	12000	4158	13298	1966	40
2018	1088	256	115208	30522	11798	4052	13110	1970	4'

In [478…
```r
# Visualize the stacked bar graph of all dataset
df <- melt(group_by_year(all_splitted_df) ,  id.vars = 'year', variable.name = 'unsuccessful_and_successful')
options(repr.plot.width = 20, repr.plot.height =10)
ggplot(df,aes(x = year, y = value)) +
geom_bar(aes(fill = unsuccessful_and_successful),stat = "identity",position = "dodge", width = 0.8) +
theme(text = element_text(size = 18), element_line(linewidth =1))
```



Analysis:

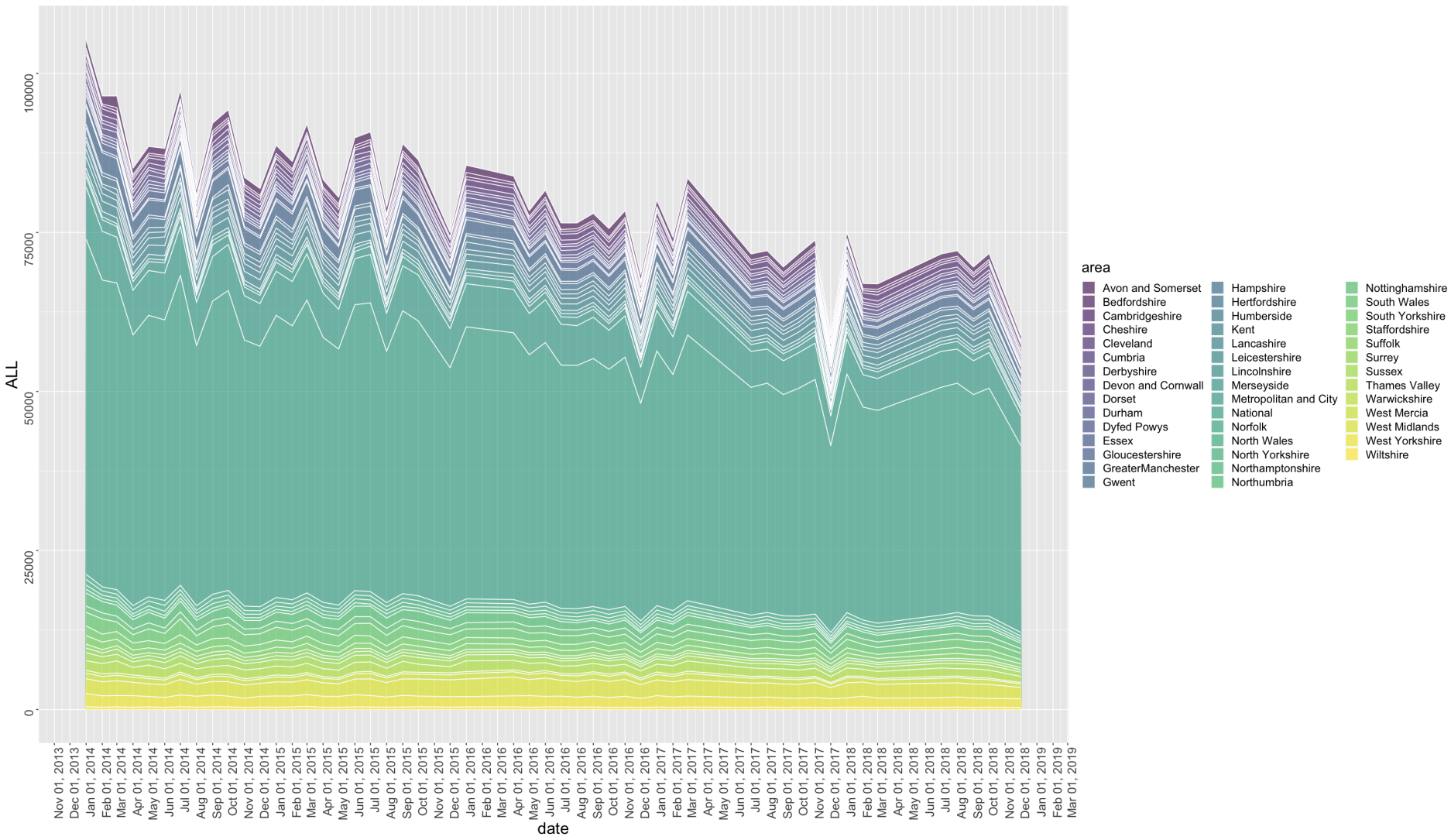
Apparently, the sum of successful and unsuccessful crimes followed the same trends of individual clusters. Therefore, it is expected that the ratio of successful to unsuccessful crimes wouldn't change that much. As a result, for further visualisations, we can use successful crime rates and crime rate words interchangeably.

Stacked Area Charts for different areas

Successful crimes

In [480…

```
# Visualized stacked area chart
options(repr.plot.width = 25.2, repr.plot.height = 15)
ggplot(crime, aes(x = date, y = ALL, fill = area)) +
  geom_area(alpha = 0.6, linewidth = 0.5, colour = "white") +
  scale_x_date(date_labels = "%b %d, %Y", date_breaks = "1 month") +
  scale_fill_viridis(discrete = TRUE) +
  labs(title = "Stacked Area Chart for Successful Crimes with Respect to the Time and Area") +
  theme(plot.title = element_text(size = 30, hjust = 0.5, vjust = 1.5),
    legend.text = element_text(size = 14),
    legend.title = element_text(size = 18),
    axis.text = element_text(angle = 90, hjust = 1, size = 15),
    axis.title.x = element_text(size = 20),
    axis.title.y = element_text(size = 20),
    )
```



Analysis:

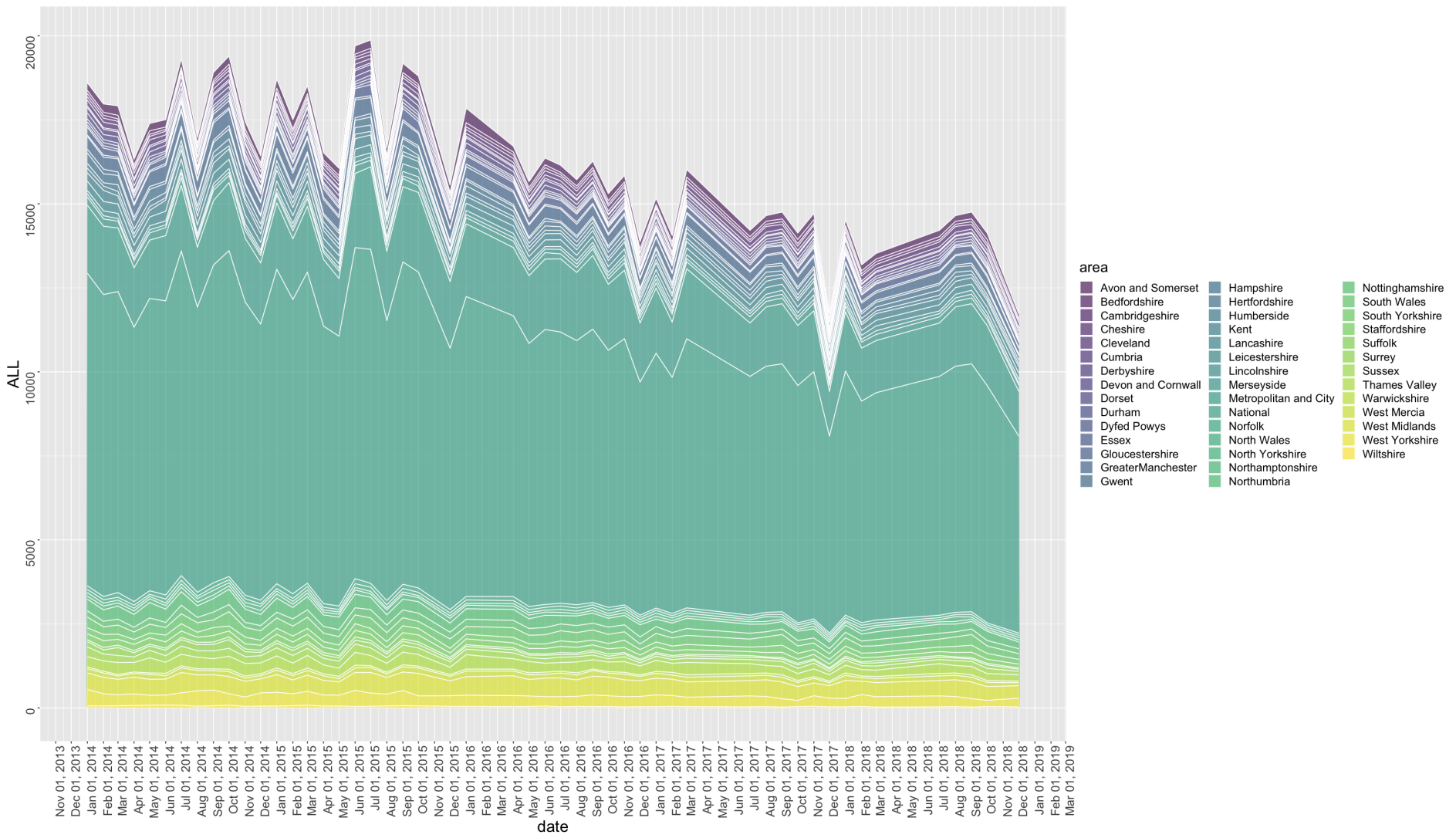
This stacked area chart shows the crime rate pattern of England And wales, also the visual proportion of each area from 2014 to 2018. The Y-axis represents the sum of all crimes in a specific time, while the X-aixs stans for the timeline. This chart offers a better visualisation of the amount and the pattern of total crimes at different times. Previous bar charts only represented a discrete general trend of crimes on a yearly basis. Also, they were biased on six months as we dropped months with no data from our dataset. However, this chart illustrates a continuous visualisation of our data, enabling us to interpret it more accurately.

Therefore, regardless of them, we can understand that the crime rate was at its highest point throughout this period, while it decreased gradually by more than %25. This decline in crime rate can have many different reasons. Nonetheless, the overall circumstance in terms of social security improved.

Unsuccessful crimes

In [481…

```
options(repr.plot.width = 25.2, repr.plot.height = 15)
ggplot(ucrime, aes(x = date, y = ALL, fill = area)) +
  geom_area(alpha = 0.6, linewidth = 0.5, colour = "white") +
 scale_x_date(date_labels = "%b %d, %Y", date_breaks = "1 month")+
  scale_fill_viridis(discrete = TRUE) +
  labs(title = "Stacked Area Chart for Unsuccessful Crimes with Respect to the Time and Area") +
  theme(plot.title = element_text(size = 30, hjust = 0.5, vjust = 1.5),
    legend.text = element_text(size = 14),
    legend.title = element_text(size = 18),
    axis.text = element_text(angle = 90, hjust = 1, size = 15),
    axis.title.x = element_text(size = 20),
    axis.title.y = element_text(size = 20),
       )
```



Analysis:

As we interpreted before, successful and unsuccessful crimes follow the same pattern as total crimes. As a result, we can find the same trend as the successful crimes for the unsuccessful ones. However, there are some minor differences in this trend. Although successful crimes always experienced a downward trend, unsuccessful ones initially had an ascending movement from the beginning of 2014 to the middle of 2015, and then it decreased moderately. In fact, its maximum value belongs to the period between 2015 and 2016. To investigate more, we need to visualize the monthly and yearly ratio trends between successful and unsuccessful crimes.

Comparing successful and unsuccessful crime ratio for different years

To evaluate the ratio of the different successful crimes and unsuccessful crimes, we must divide two different grouped dataframe by the year for successful crimes to the grouped dataframe by the year for unsuccessful crimes. Then, we will visualize it and interpret the relations between these two parameters.

In [484…
```
# Create a dataframe to keep ratio of crime rates in different categories
ratio_df <- splitted_df[,-c(1)] / u_splitted_df[,-c(1,13)]
```

In [485…
```
ratio_df
```

A data.frame: 5 × 13

homicide <dbl>	offences_against_the_person <dbl>	sexual_offences <dbl>	burglary <dbl>	robbery <dbl>	theft_and_handling <dbl>	fraud_and_forgery <dbl>	criminal_damage <dbl>	drugs_offences <dbl>	public_o
4.500000	2.970483	2.464286	5.629013	3.531453	10.38503	5.977543	5.635232	14.93483	
3.758621	3.192984	2.613364	5.515313	3.491690	10.51407	5.502787	5.637811	14.66100	
4.137405	3.488841	2.877000	5.931950	3.631313	11.39087	6.016741	5.465806	14.72672	
4.581197	3.759354	2.886003	6.763988	3.527241	11.66097	6.518710	6.669032	14.56983	
4.250000	3.774589	2.911649	6.654822	3.828678	11.46090	6.614173	6.652968	14.42538	

In [486…
```
# Create a function to add the years column to the ratio_df
insert_column <- function(column, dataset) {
  result <- cbind(column, dataset)
    colnames(result)[1] <- 'year'
  return(result)
}
```

In [488…
```
ratio_df <- insert_column(splitted_df$year, ratio_df)
head(ratio_df)
```

A data.frame: 5 × 15

	year <chr>	year <chr>	homicide <dbl>	offences_against_the_person <dbl>	sexual_offences <dbl>	burglary <dbl>	robbery <dbl>	theft_and_handling <dbl>	fraud_and_forgery <dbl>	criminal_damage <dbl>	drugs_
1	2014	2014	4.500000	2.970483	2.464286	5.629013	3.531453	10.38503	5.977543	5.635232	
2	2015	2015	3.758621	3.192984	2.613364	5.515313	3.491690	10.51407	5.502787	5.637811	
3	2016	2016	4.137405	3.488841	2.877000	5.931950	3.631313	11.39087	6.016741	5.465806	
4	2017	2017	4.581197	3.759354	2.886003	6.763988	3.527241	11.66097	6.518710	6.669032	
5	2018	2018	4.250000	3.774589	2.911649	6.654822	3.828678	11.46090	6.614173	6.652968	

In [489…
```
# Plotting the yearly ratio of successful and unsuccessful crime
```

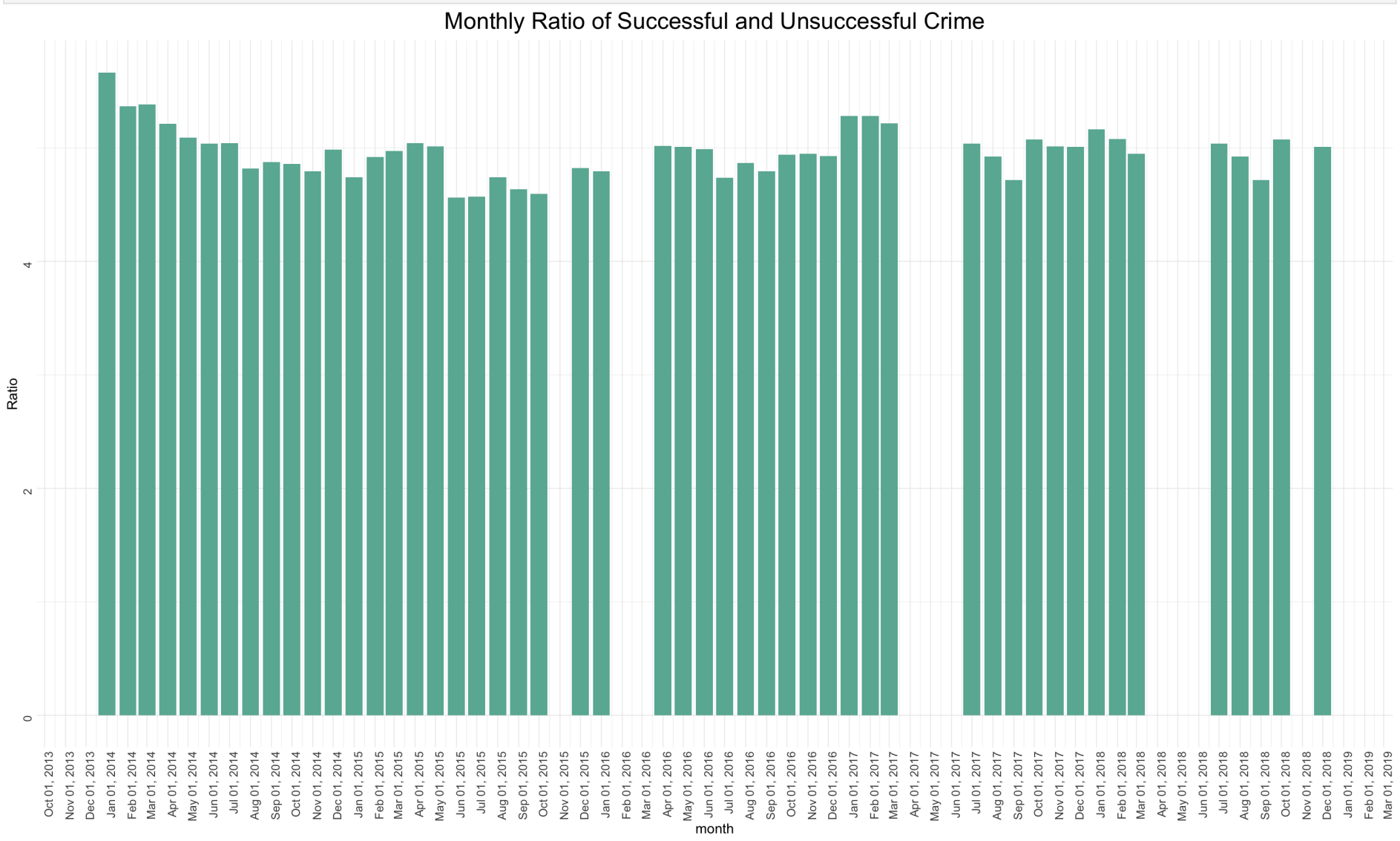
```r
plot_df <- data.frame(
  year = ratio_df$year,
  value = ratio_df$ALL
)

options(repr.plot.width = 25.2, repr.plot.height = 15)

# Create the bar plot with adjusted width
yearly_ratio <- ggplot(plot_df, aes(x = year, y = value)) +
  geom_bar(stat = "identity", width = 0.2, fill = "cyan") +
  labs(title = "Yearly Ratio of Successful and Unsuccessful Crime") +
  labs(x = "Year", y = "Ratio") +
  theme_minimal() +
  theme(axis.text = element_text(size = 15), axis.title = element_text(size = 18),
        plot.title = element_text(size = 30, hjust = 0.5, vjust = 1.5))  # Adjust the size as desired
```

In [490…

```r
# Plotting the monthly ratio of successful and unsuccessful crime

# Slicing dataframe and creating a dataframe of the ratio
u_grouped_by_date <- filter(ucrime, area == "National")[,-c(1,2,3,4)]
grouped_by_date <- filter(crime, area == "National")[,-c(1,2,3,4)]
monthly_ratio_df <- u_grouped_by_date[,c(1,2)]
monthly_ratio_df$ALL <- grouped_by_date[,14]/u_grouped_by_date[,15]

plot_df <- data.frame(
  month = monthly_ratio_df$date,
  value = monthly_ratio_df$ALL
)

options(repr.plot.width = 25.2, repr.plot.height = 15)

# Create the bar plot with adjusted width
monthly_ratio <- ggplot(plot_df, aes(x = month, y = value)) +
  geom_bar(stat = "identity", width = 25, fill = "#69b3a2") +
  labs(title = "Monthly Ratio of Successful and Unsuccessful Crime") +
  scale_x_date(date_labels = "%b %d, %Y", date_breaks = "1 month") +
  labs(x = "month", y = "Ratio") +
  theme_minimal() +
  theme(axis.text = element_text(angle = 90, hjust = 1, size = 15),
        axis.title = element_text(size = 18),
        plot.title = element_text(size = 30, hjust = 0.5, vjust = 1.5))
```

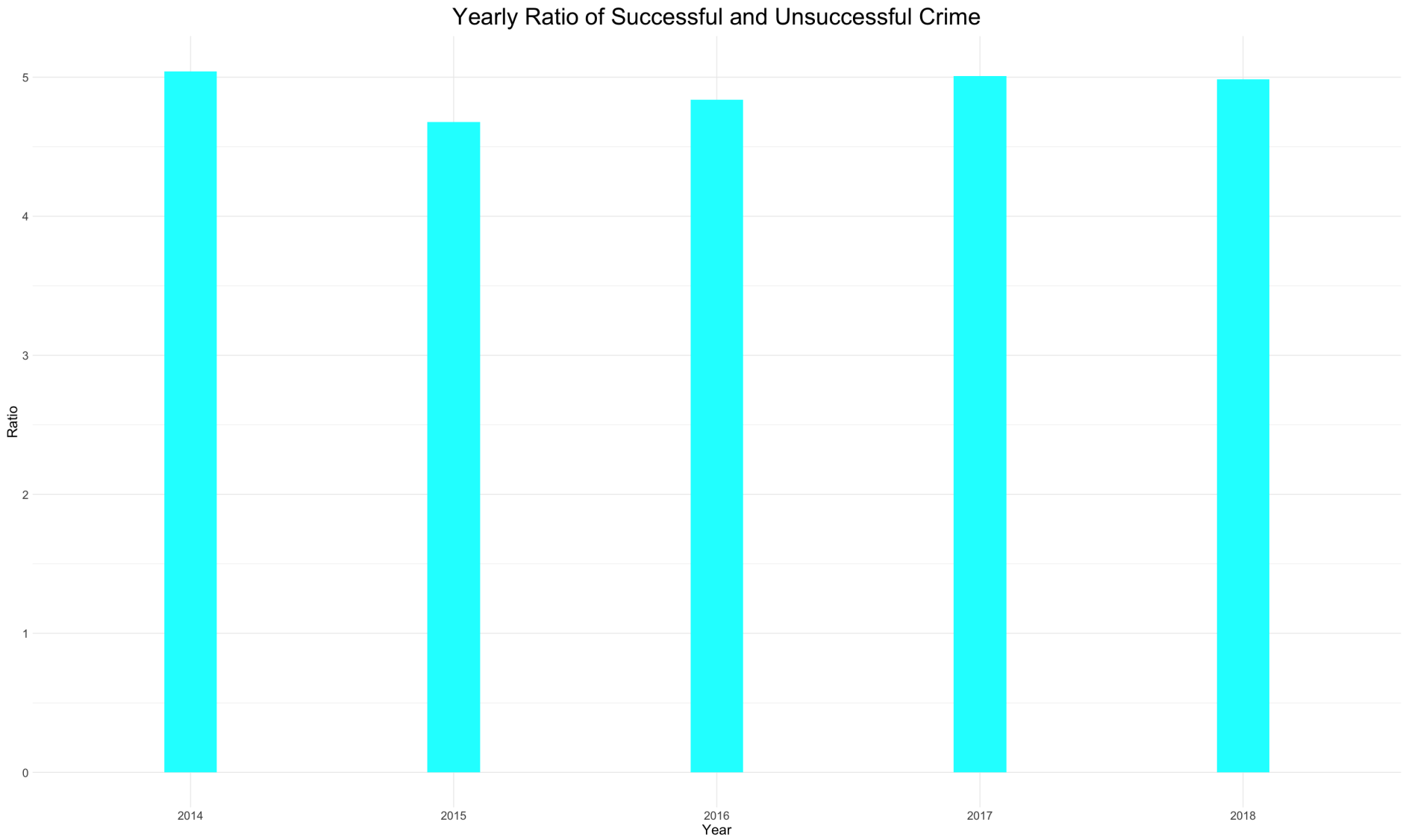
In [491…

```r
monthly_ratio
```



In [492…

```r
yearly_ratio
```



Analysis:

These charts illustrate the ratio between the number of successful and unsuccessful crimes. From 2014 to 2018, this ratio was at its highest point at above 5 in 2014. However, it reduced moderately to its lowest point in the next year, which was good progress. And then, it started to increase gradually from 2015 to 2017, and it almost maintained its initial during the following year. The lower this number, the better performance of the police is. Consequently, we can tell the security organizations worked at their performance in 2015, while their performances declined gently for the rest of this period.

Monthly Visualisation and Comparison of Crime Rates

Pipeline:

We are going to compare the crime rate for different months to each other. To do so, we need to sum up all the rows concerning their month and then put the result in a row in a new dataframe. However, there are some missing months in our dataset. For example, our dataset contains data for January for all these five years, while it just provides data for May for three years. Consequently, By summing up the rows corresponding to the same month, the result for other months are different as the different month has different weights on our dataset.

We have to take an average of each month to solve this issue. To do so, we need to divide all the numbers corresponding to different crime categories in a specific month by the number of years contained in this month. For instance, after summing up all the rows corresponding to May, we should count how many years contain May (in our case, this is 3), and then divide the result but 3. By this action, we remove the bias in our data to make an accurate analysis.

Monthly Successful Crime visualisation

In [497…
```r
# Define a function to group our data by the months
group_by_month <- function(dataframe){
  dataframe <- dataframe[,-c(1,2,4,5)]
  dataframe <- group_by(dataframe, month)
  summarise_all(dataframe, list(sum))
}
```

In [498…
```r
# Deploy it
crime_grouped_by_months <- group_by_month(crime)
crime_grouped_by_months <- crime_grouped_by_months[order(crime_grouped_by_months$month), ]
crime_grouped_by_months
```

A tibble: 12 × 14

month	homicide	offences_against_the_person	sexual_offences	burglary	robbery	theft_and_handling	fraud_and_forgery	criminal_damage	drugs_offences	pu
<chr>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	
apr	580	57628	5406	7974	2678	54782	5020	13010	25994	
aug	536	96610	9442	12080	3814	73440	7934	20068	37674	
dec	934	88562	9054	11120	3558	67284	7446	16784	34630	
feb	562	73744	6946	10462	3490	65996	6482	16318	32366	
jan	724	102642	9170	13562	4238	86544	8900	22360	42994	
jul	1014	102588	10034	13268	4400	79060	8576	21460	41122	
jun	490	60096	5688	8060	2774	54878	4790	13916	26124	
mar	624	78370	7366	11018	3648	70042	6990	16932	33696	
may	526	55428	5232	8010	2674	53802	4694	13160	24608	
nov	486	59946	5922	7658	2406	45916	4928	11846	23536	
oct	902	101150	9840	13304	3708	77678	8832	20140	40344	
sep	726	100368	9982	12874	4276	75786	8200	20610	38940	

In [499…

```
month_sorter <- function(dataframe){
    real_order_of_months <- c("jan", "feb", "mar", "apr", "may", "jun", "jul", "aug", "sep", "oct", "nov", "dec")

    # Convert the "Month" column to a factor with the custom ordering
    dataframe$month <- factor(dataframe$month, levels = real_order_of_months)

    # Sort the dataframe based on the custom ordering of the "Month" column
    dataframe <- dataframe[order(dataframe$month), ]

    return (dataframe)
    }
```

In [500…

```
crime_grouped_by_months <- month_sorter(crime_grouped_by_months)
crime_grouped_by_months
```

A tibble: 12 × 14

month	homicide	offences_against_the_person	sexual_offences	burglary	robbery	theft_and_handling	fraud_and_forgery	criminal_damage	drugs_offences	pu
<fct>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	
jan	724	102642	9170	13562	4238	86544	8900	22360	42994	
feb	562	73744	6946	10462	3490	65996	6482	16318	32366	
mar	624	78370	7366	11018	3648	70042	6990	16932	33696	
apr	580	57628	5406	7974	2678	54782	5020	13010	25994	
may	526	55428	5232	8010	2674	53802	4694	13160	24608	
jun	490	60096	5688	8060	2774	54878	4790	13916	26124	
jul	1014	102588	10034	13268	4400	79060	8576	21460	41122	
aug	536	96610	9442	12080	3814	73440	7934	20068	37674	
sep	726	100368	9982	12874	4276	75786	8200	20610	38940	
oct	902	101150	9840	13304	3708	77678	8832	20140	40344	
nov	486	59946	5922	7658	2406	45916	4928	11846	23536	
dec	934	88562	9054	11120	3558	67284	7446	16784	34630	

In [501…

```
# This function gets the grouped_by dataframe, and devide all each month by the number of month
# available in the dataset. It helps to get an average crime number for each month in each segment
# two November, one February, one March, two April, two May, and June are missed.


divide_rows_by_weights <- function(dataframe) {

  # this is the number of month for each month.
    num_months_in_5year <- c(5, 4, 4, 3, 3, 3, 5, 5, 5, 5, 3, 5)

    dataframe[, -1] <- sweep(dataframe[, -1], 1, num_months_in_5year, "/")

    return (dataframe)
}
```

In [502…

```
average_crime_monthly <- divide_rows_by_weights(crime_grouped_by_months)
average_crime_monthly
```

A tibble: 12 × 14

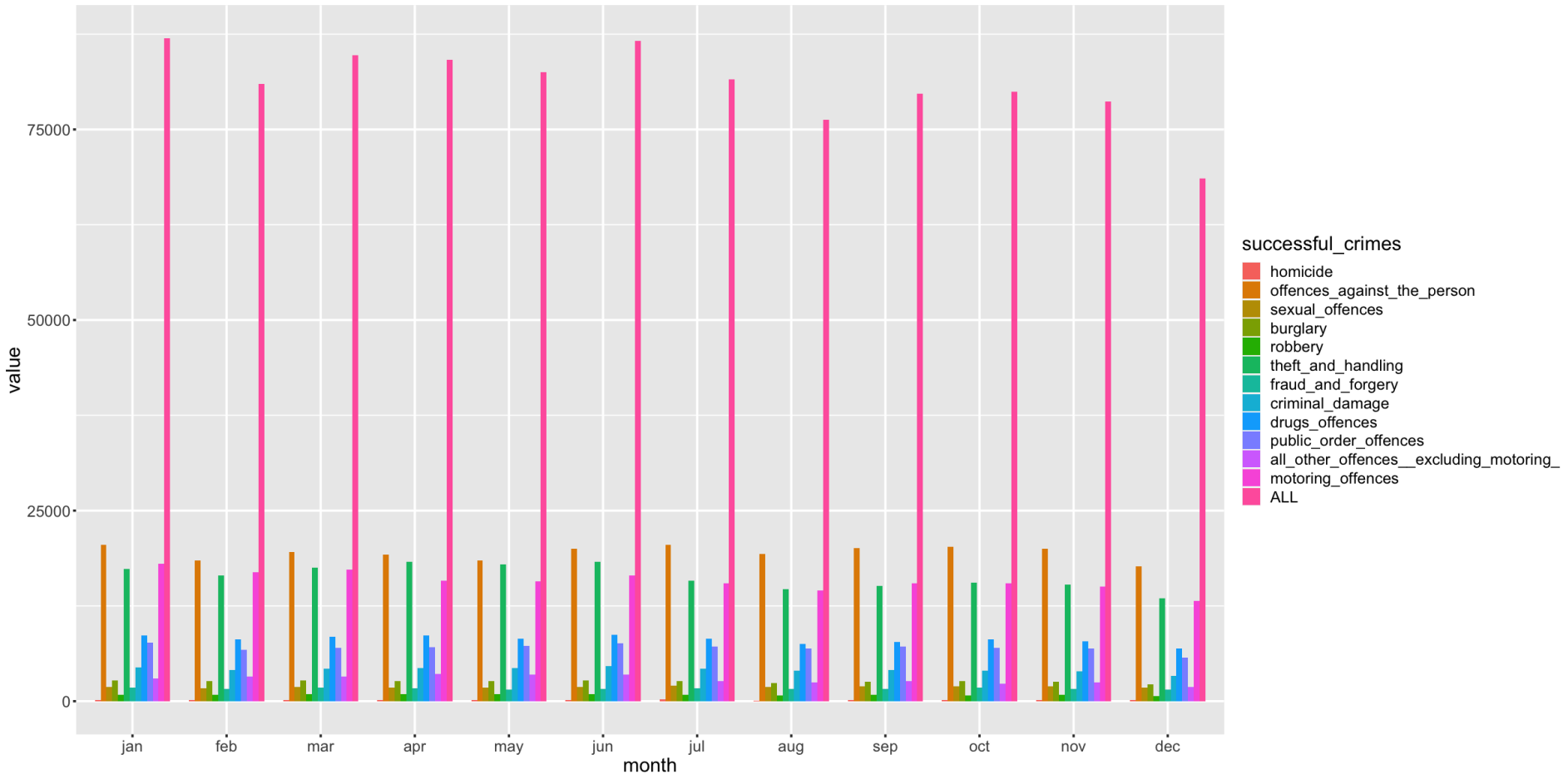
month <fct>	homicide <dbl>	offences_against_the_person <dbl>	sexual_offences <dbl>	burglary <dbl>	robbery <dbl>	theft_and_handling <dbl>	fraud_and_forgery <dbl>	criminal_damage <dbl>	drugs_offences <dbl>
jan	144.8000	20528.40	1834.0	2712.400	847.6000	17308.80	1780.000	4472.000	8598.800
feb	140.5000	18436.00	1736.5	2615.500	872.5000	16499.00	1620.500	4079.500	8091.500
mar	156.0000	19592.50	1841.5	2754.500	912.0000	17510.50	1747.500	4233.000	8424.000
apr	193.3333	19209.33	1802.0	2658.000	892.6667	18260.67	1673.333	4336.667	8664.667
may	175.3333	18476.00	1744.0	2670.000	891.3333	17934.00	1564.667	4386.667	8202.667
jun	163.3333	20032.00	1896.0	2686.667	924.6667	18292.67	1596.667	4638.667	8708.000
jul	202.8000	20517.60	2006.8	2653.600	880.0000	15812.00	1715.200	4292.000	8224.400
aug	107.2000	19322.00	1888.4	2416.000	762.8000	14688.00	1586.800	4013.600	7534.800
sep	145.2000	20073.60	1996.4	2574.800	855.2000	15157.20	1640.000	4122.000	7788.000
oct	180.4000	20230.00	1968.0	2660.800	741.6000	15535.60	1766.400	4028.000	8068.800
nov	162.0000	19982.00	1974.0	2552.667	802.0000	15305.33	1642.667	3948.667	7845.333
dec	186.8000	17712.40	1810.8	2224.000	711.6000	13456.80	1489.200	3356.800	6926.000

In [503…

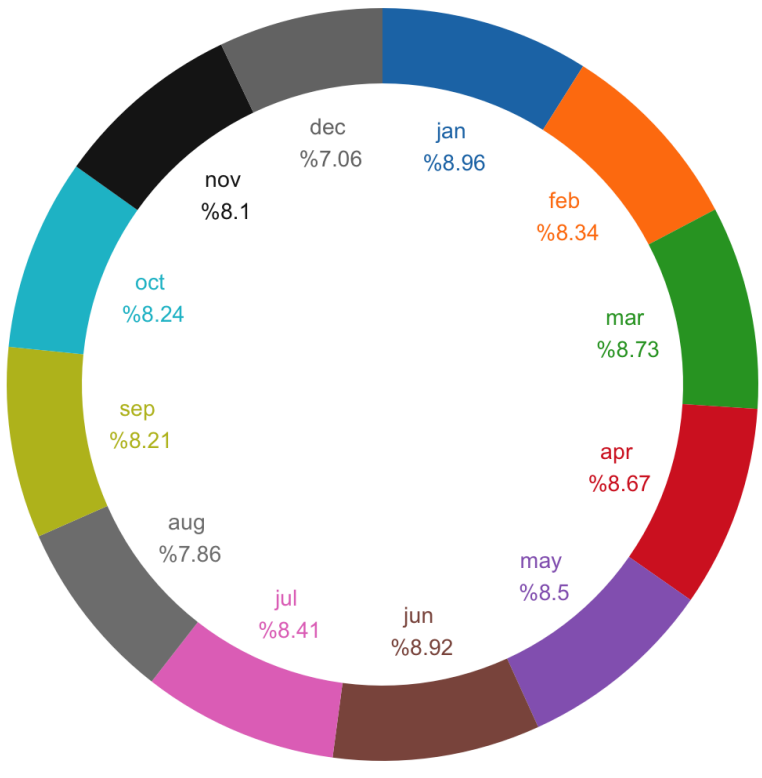
```r
doughnut_plotter <- function(s) {

    # Compute percentages
    s$fraction <- s$ALL / sum(s$ALL)

    # Compute the cumulative percentages (top of each rectangle)
    s$ymax <- cumsum(s$fraction)

    # Compute the bottom of each rectangle
    s$ymin <- c(0, head(s$ymax, n=-1))

    # Compute label position
    s$labelPosition <- (s$ymax + s$ymin) / 2

    # Creating the percentage
    s$percentage <- round(s$ALL/sum(s$ALL), digits = 4)

    # Compute a good label
    s$label <- paste0(s$month, "\n %", 100*s$percentage)


   # Define the custom palette with 12 dark colors
    custom_palette <- c("#1f77b4", "#ff7f0e", "#2ca02c", "#d62728", "#9467bd",
                        "#8c564b", "#e377c2", "#7f7f7f", "#bcbd22", "#17becf",
                        "#1a1a1a", "#757575")

    options(repr.plot.width = 20, repr.plot.height = 10)

    # Make the plot
    ggplot(s, aes(ymax = ymax, ymin = ymin, xmax = 4, xmin = 3, fill = month)) +
      geom_rect() +
      geom_text(x = 2.3, aes(y = labelPosition, label = label, color = month), size = 6) +
      scale_fill_manual(values = custom_palette) +  # Use the custom fill palette
      scale_color_manual(values = custom_palette) +  # Use the custom color palette
      coord_polar(theta = "y") +
      xlim(c(-1, 4)) +
      theme_void() +
      theme(legend.position = "none")

    }
```

In [505…

```r
# Deploy it
df <- melt(average_crime_monthly ,  id.vars = 'month', variable.name = 'successful_crimes')
options(repr.plot.width = 20, repr.plot.height =10)
ggplot(df,aes(x = month, y = value)) +
geom_bar(aes(fill = successful_crimes),stat = "identity",position = "dodge", width = 0.8) +
theme(text = element_text(size = 18), element_line(linewidth =1))
```



In [507…
```
# Plot doughnut plot
doughnut_plotter(average_crime_monthly)
```



Analysis:

The first graph represents the average number of crimes in different categories for different months from 2014 to 2018. Also, the second graph illustrates the average portion of successful crimes for each month. As is shown, the maximum crime rate occurred in January and June. The former includes %8.96 of all crimes, and this number is %8.92 for the latter. Throughout the year, the crime rate fluctuated between 75000 and 82500 crimes per month in most of the months. However, in December, this rate plunged to almost 62500 crimes per month, which just form %7.06 of total successful crimes, which is its lowest amount during the year. Although with respect to the information in this chart, it is not clear why the trend follows this pattern, we can associate this fall in December to different factors. Here is my perspective on it:

Hypothesis:

In most cases, crimes happen in absolute chaos or in extreme quietness. For example, most of the sexual offences_against_person or coercion happens in the streets with either few or no passengers. This provides an opportunity for criminals to approach their victims without interference from different people. Also, in these cases, the victim's chance of survival is lower as he/she can't easily ask for help due to mentioned situation. On the other hand, most of the robberies or steals happen in chaotic places such as transportation, hubs, busy areas, etc.

Due to the Christmas holidays, people mostly spend their time with their families and friends, businesses might be closed or their working time reduced compared to the other months. Overall, the cities' traffic declined this month. Consequently, the streets' general condition is unsuitable for criminals. Their chance of success decreases, so they might prevent risk in this period. In my perspective, it might be one of the possible reasons why the crime rate in December is at its lowest point.

On the other hand, considering these holidays, people return to their cities and towns. In my perspective, the criminals of a neighbourhood are not living in that neighbourhood. They prefer to commit crimes outside of their own living area. Therefore, we can say by travelling people to their own cities during these holidays, the number of real native inhabitant of an area grow compared to temporary citizens. As a result, the number of crimes might fall due to this hypothesis. However, to prove it, we need further analysis to agree or disagree.

Monthly Unuccessful Crime visualisation

In [508…
```
# Group ucrime dataset by month
ucrime_grouped_by_months <- group_by_month(ucrime)
ucrime_grouped_by_months <- ucrime_grouped_by_months[order(ucrime_grouped_by_months$month), ]
ucrime_grouped_by_months
```

A tibble: 12 × 1

month	homicide_un	offences_against_the_person_un	sexual_offences_un	burglary_un	robbery_un	theft_and_handling_un	fraud_and_forgery_un	criminal_dam
<chr>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	
apr	100	17860	2064	1422	650	4676	764	
aug	182	28474	3364	2052	1048	7054	1340	
dec	196	24932	3104	1688	910	5822	1204	
feb	146	22310	2570	1754	902	5590	1006	
jan	196	29802	3362	2338	1236	7756	1414	
jul	254	30316	3744	2274	1230	7088	1440	
jun	124	19150	2152	1564	660	5150	796	
mar	152	23910	2724	1772	960	5974	1094	
may	112	17084	2088	1368	704	4880	774	
nov	162	17508	2330	1330	602	4046	782	
oct	200	29614	3678	2110	1122	7056	1402	
sep	116	30446	3648	2252	1142	7194	1374	

In [509…
```
ucrime_grouped_by_months <- month_sorter(ucrime_grouped_by_months)
ucrime_grouped_by_months
```

A tibble: 12 × 1

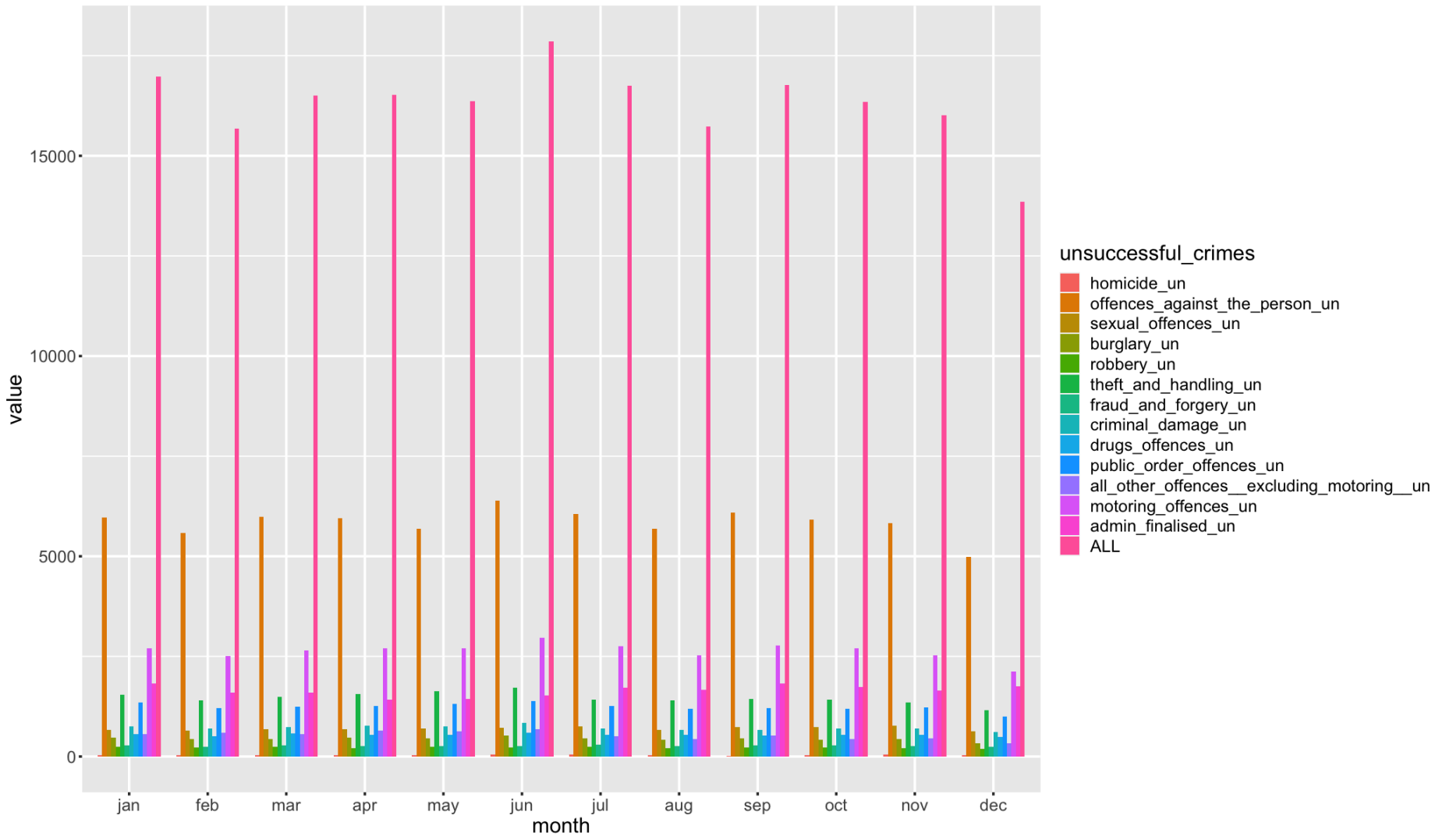
month	homicide_un	offences_against_the_person_un	sexual_offences_un	burglary_un	robbery_un	theft_and_handling_un	fraud_and_forgery_un	criminal_dam
<fct>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	
jan	196	29802	3362	2338	1236	7756	1414	
feb	146	22310	2570	1754	902	5590	1006	
mar	152	23910	2724	1772	960	5974	1094	
apr	100	17860	2064	1422	650	4676	764	
may	112	17084	2088	1368	704	4880	774	
jun	124	19150	2152	1564	660	5150	796	
jul	254	30316	3744	2274	1230	7088	1440	
aug	182	28474	3364	2052	1048	7054	1340	
sep	116	30446	3648	2252	1142	7194	1374	
oct	200	29614	3678	2110	1122	7056	1402	
nov	162	17508	2330	1330	602	4046	782	
dec	196	24932	3104	1688	910	5822	1204	

In [511…
```
# get average number of crimes
average_ucrime_monthly <- divide_rows_by_weights(ucrime_grouped_by_months)
average_ucrime_monthly
```

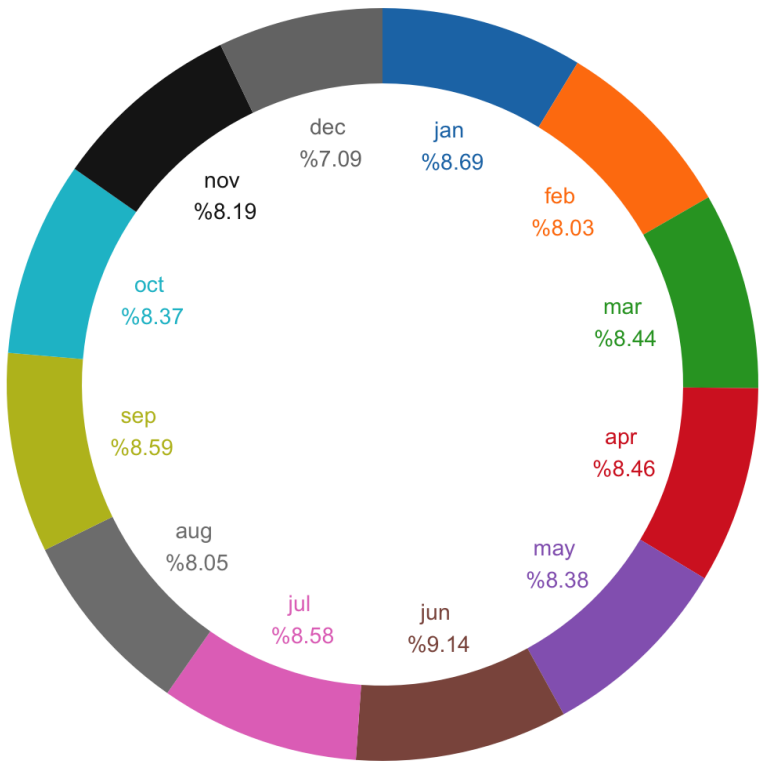
A tibble: 12 ×

month	homicide_un	offences_against_the_person_un	sexual_offences_un	burglary_un	robbery_un	theft_and_handling_un	fraud_and_forgery_un	criminal_dam
<fct>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	
jan	39.20000	5960.400	672.4000	467.6000	247.2000	1551.200	282.8000	75
feb	36.50000	5577.500	642.5000	438.5000	225.5000	1397.500	251.5000	70
mar	38.00000	5977.500	681.0000	443.0000	240.0000	1493.500	273.5000	72
apr	33.33333	5953.333	688.0000	474.0000	216.6667	1558.667	254.6667	76
may	37.33333	5694.667	696.0000	456.0000	234.6667	1626.667	258.0000	74
jun	41.33333	6383.333	717.3333	521.3333	220.0000	1716.667	265.3333	83
jul	50.80000	6063.200	748.8000	454.8000	246.0000	1417.600	288.0000	7(
aug	36.40000	5694.800	672.8000	410.4000	209.6000	1410.800	268.0000	66
sep	23.20000	6089.200	729.6000	450.4000	228.4000	1438.800	274.8000	67
oct	40.00000	5922.800	735.6000	422.0000	224.4000	1411.200	280.4000	70
nov	54.00000	5836.000	776.6667	443.3333	200.6667	1348.667	260.6667	70
dec	39.20000	4986.400	620.8000	337.6000	182.0000	1164.400	240.8000	6(

In [512…
```
# Visualize it
df <- melt(average_ucrime_monthly ,  id.vars = 'month', variable.name = 'unsuccessful_crimes')
options(repr.plot.width = 17, repr.plot.height =10)
ggplot(df,aes(x = month, y = value)) +
geom_bar(aes(fill = unsuccessful_crimes),stat = "identity",position = "dodge", width = 0.8) +
theme(text = element_text(size = 18), element_line(linewidth =1))
```



In [513…
```
doughnut_plotter(average_ucrime_monthly)
```



Analysis:

The analysis we used for the successul crimes works for unsuccessful ones also. The reason is that by increasing the total number of crimes regardless of being successful and unsuccessful, the number of each categories goes up. Consequently, we could anticipate our hypothesis can cover these categories as well.

Seasonal Visualisation and Comparison of Crime Rates

Pipeline:

In this part, we use the average crime rates of different months to form the seasonal dataset. As we remove the effect of weighted months in the previous section, our data here is not biased, so we can do our analysis without any further preprocessing on our dataset.

Seasonal Comparison of Successfull Crimes

In [514…
```
# Create a function to associate month by seasons
calculate_season_sums <- function(df, season_col) {

    season_months <- list(
  Winter = c("jan", "feb", "dec"),
  Spring = c("mar", "apr", "may"),
  Summer = c("jun", "jul", "aug"),
  Autumn = c("sep", "oct", "nov")
```

```
)

  seasons <- names(season_months)
  season_df <- data.frame(matrix(0, nrow = length(seasons), ncol = ncol(df)))

  for (i in 1:length(seasons)) {
    season <- seasons[i]
    season_cols <- which(df[[season_col]] %in% season_months[[season]])

    season_rows <- df[season_cols, ]
    season_sums <- colSums(season_rows[, -1], na.rm = TRUE)

    season_df[i, -1] <- season_sums
  }

  row.names(season_df) <- seasons
    colnames(season_df)[-1] <- colnames(df)[-1]
    colnames(season_df)[1] <- "Season"
    season_df$Season <- rownames(season_df)

    # Step 4: Remove the row names
    rownames(season_df) <- NULL


  return(season_df)
}
```

In [516…

```
# Check the result
calculate_season_sums(average_crime_monthly, "month")
```

A data.frame: 4 × 14

Season	homicide	offences_against_the_person	sexual_offences	burglary	robbery	theft_and_handling	fraud_and_forgery	criminal_damage	drugs_offences
<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
Winter	472.1000	56676.80	5381.3	7551.900	2431.700	47264.60	4889.700	11908.30	23616.30
Spring	524.6667	57277.83	5387.5	8082.500	2696.000	53705.17	4985.500	12956.33	25291.33
Summer	473.3333	59871.60	5791.2	7756.267	2567.467	48792.67	4898.667	12944.27	24467.20
Autumn	487.6000	60285.60	5938.4	7788.267	2398.800	45998.13	5049.067	12098.67	23702.13

In [517…

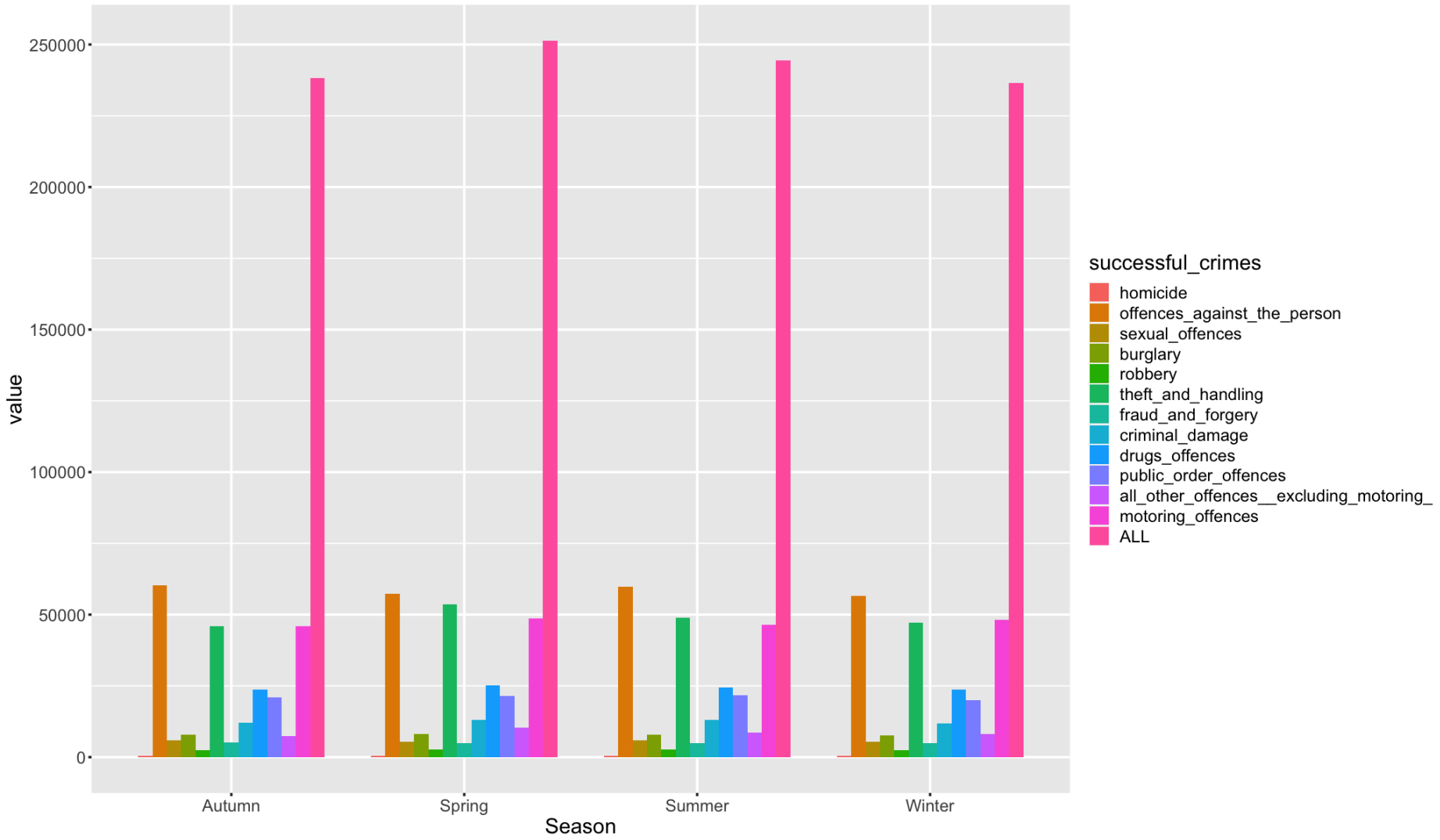
```
# Deploy it
s <- calculate_season_sums(average_crime_monthly, "month")
```

In [518…

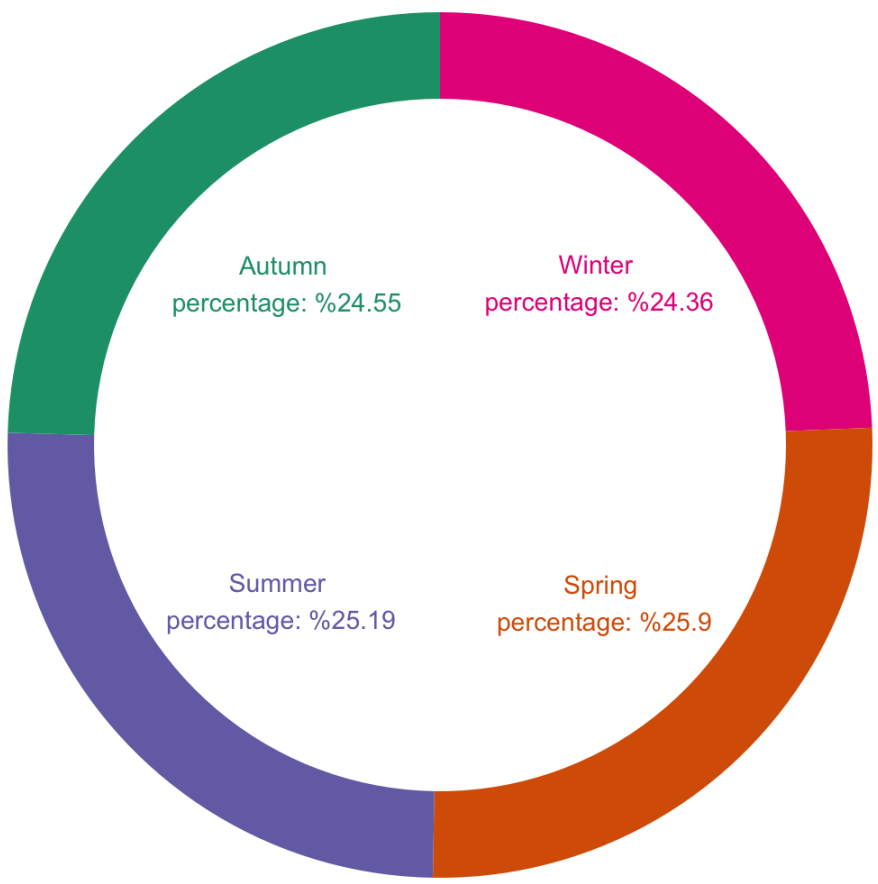
```
# Define the visualization function
doughnut_plotter2 <- function(s){
# Compute percentages
s$fraction <- s$ALL / sum(s$ALL)

# Compute the cumulative percentages (top of each rectangle)
s$ymax <- cumsum(s$fraction)

# Compute the bottom of each rectangle
s$ymin <- c(0, head(s$ymax, n=-1))

# Compute label position
s$labelPosition <- (s$ymax + s$ymin) / 2

# Creating the percentage
s$percentage <- round(s$ALL/sum(s$ALL), digits = 4)

# Compute a good label
s$label <- paste0(s$Season, "\n percentage: %", 100*s$percentage)


# Make the plot
ggplot(s, aes(ymax=ymax, ymin=ymin, xmax=4, xmin=3, fill=Season)) +
  geom_rect() +
  geom_text( x=1.6, aes(y=labelPosition, label=label, color=Season), size=6) + # x here controls label position (inner / outer)
  scale_fill_brewer(palette="Dark2") +
  scale_color_brewer(palette="Dark2") +
  coord_polar(theta="y") +
  xlim(c(-1, 4)) +
  theme_void() +
  theme(legend.position = "none")
    }
```

In [519…

```
df <- melt(s ,  id.vars = 'Season', variable.name = 'successful_crimes')
options(repr.plot.width = 17, repr.plot.height =10)
ggplot(df,aes(x = Season, y = value)) +
geom_bar(aes(fill = successful_crimes),stat = "identity",position = "dodge", width = 0.8) +
theme(text = element_text(size = 18), element_line(linewidth =1))
```



In [520…
```
doughnut_plotter2(s)
```



Analysis:

These charts suggest that. There is not a big difference between the crime rate in different seasons. The proportion of crime fluctuated around %25 for all seasons. However, Spring has the highest rate at %25.9 per cent, which is slightly more than the other month. Overall we can extract meaningful information from these visualisations by itself.

Seasonal Comparison of Unsuccessfull Crimes

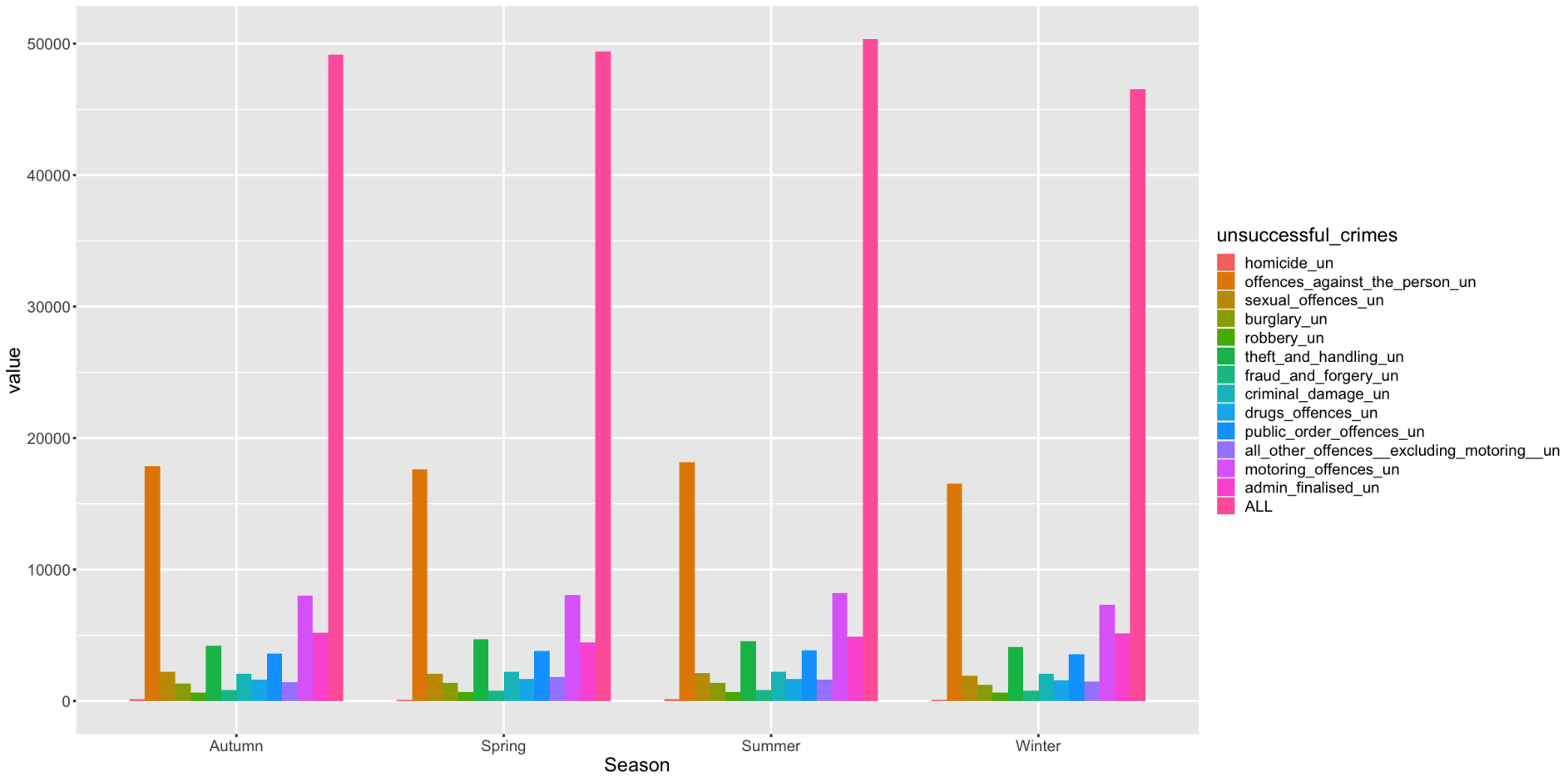
In [521…
```
calculate_season_sums(average_ucrime_monthly, "month")
```

A data.frame:

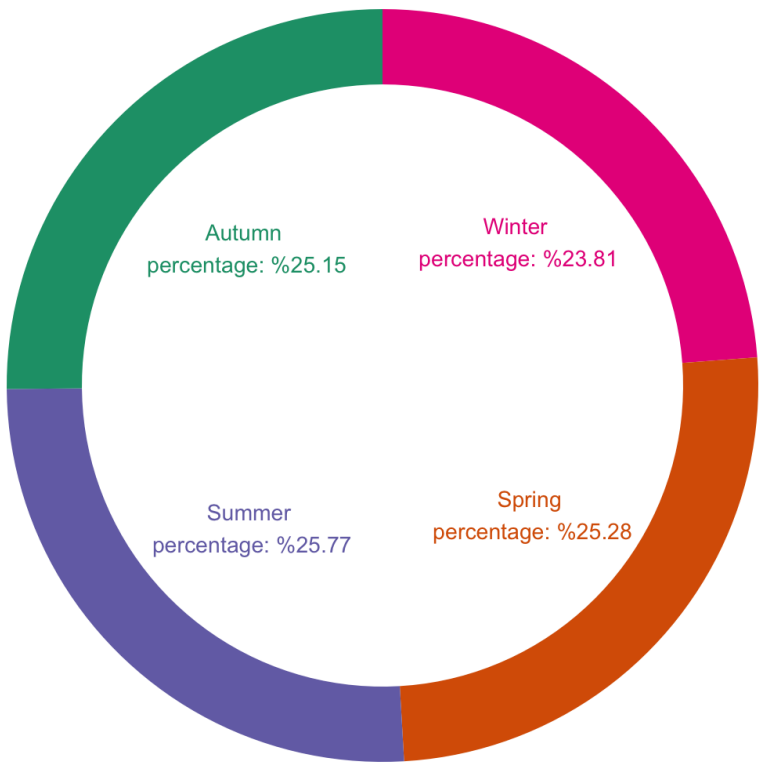
Season	homicide_un	offences_against_the_person_un	sexual_offences_un	burglary_un	robbery_un	theft_and_handling_un	fraud_and_forgery_un	criminal_dar
<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	
Winter	114.9000	16524.30	1935.700	1243.700	654.7000	4113.100	775.1000	
Spring	108.6667	17625.50	2065.000	1373.000	691.3333	4678.833	786.1667	
Summer	128.5333	18141.33	2138.933	1386.533	675.6000	4545.067	821.3333	
Autumn	117.2000	17848.00	2241.867	1315.733	653.4667	4198.667	815.8667	

In [522…
```
us <- calculate_season_sums(average_ucrime_monthly, "month")
```

In [523…
```
df <- melt(us ,  id.vars = 'Season', variable.name = 'unsuccessful_crimes')
options(repr.plot.width = 20, repr.plot.height =10)
ggplot(df,aes(x = Season, y = value)) +
geom_bar(aes(fill = unsuccessful_crimes),stat = "identity",position = "dodge", width = 0.8) +
theme(text = element_text(size = 18), element_line(linewidth =1))
```



In [524…
```
doughnut_plotter2(us)
```



Timeline Crime Rate Visualisation:

Stacked Line Graph for Conviction Groups

In [525…
```
# Create a dataframe grouped by date
grouped_by_date <- filter(crime, area == "National")[,-c(1,2,3,4,18)]
head(grouped_by_date)
```

A data.frame: 6 × 13

	date	homicide	offences_against_the_person	sexual_offences	burglary	robbery	theft_and_handling	fraud_and_forgery	criminal_damage	drugs_offences
	<date>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>
1	2014-01-01	51	9087	736	1715	522	11057	846	2693	4988
2	2014-02-01	61	8366	712	1549	541	10150	758	2455	4565
3	2014-03-01	57	8595	731	1618	569	10716	814	2413	4649
4	2014-04-01	81	7805	698	1470	517	10045	666	2259	4536
5	2014-05-01	69	8178	721	1562	532	10146	754	2403	4510
6	2014-06-01	61	8444	778	1467	512	9938	708	2466	4563

In [526…

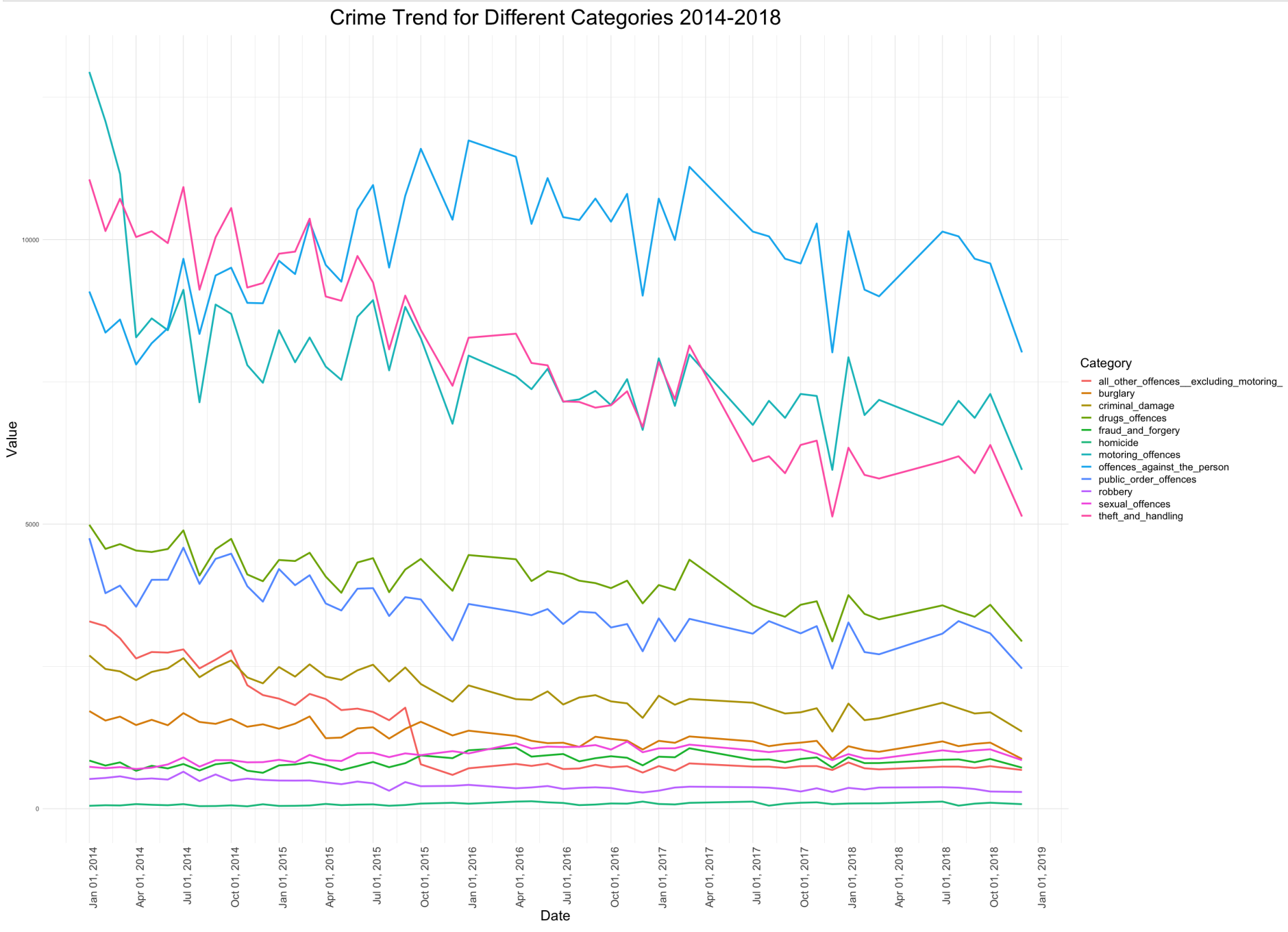
```
# Create a function for visualizing the stacked linear graph
stacked_line_grapher <- function (data){
    df_long <- data %>% pivot_longer(cols = -date, names_to = "Category", values_to = "Value")

    options(repr.plot.width = 25.2, repr.plot.height = 18)

    # Plot the stacked line graph
    p <- ggplot(df_long, aes(x = date, y = Value, group = Category, color = Category)) +
      geom_line(linewidth=1.2) +
      scale_x_date(date_labels = "%b %d, %Y", date_breaks = "3 month") +
      labs(x = "Date", y = "Value", title = "Crime Trend for Different Categories 2014-2018") +
      theme_minimal()+
    theme(axis.text.x = element_text(angle = 90, hjust = 1, size=15),
        plot.title = element_text(size = 30, hjust = 0.5, vjust = 1.5),
        axis.title.x = element_text(size = 20),
        axis.title.y = element_text(size = 20),
        legend.text = element_text(size = 14),
        legend.title = element_text(size = 18))
    return (p)
    }
```

Please Note:

if the visualization isn't shown, please rerun the following code

In [527…

```
stacked_line_grapher(grouped_by_date)
```



Analysis:

This chart shows the trend of convictions between 2014 and 2018. Each line represents the number of convicted cases in CPS in a specific area. The overall trend for categories containing the biggest portion of crimes was descending. While the number of crimes in most of the groups decreased gradually within

this period, there are some that remained steady. Also, we can observe a few categories with a slight upward trend. The general outlook of this graph indicates that the crime rate reduced during this 4-year period. However, to analyze each category more precisely, we need to investigate each one individually.

all_other_offences_excluding_motoring:

In the documentation of our dataset, there is no obvious information about this category. So we don't have enough details to know which subcategory of offences belongs to this group. Nevertheless, we can interpret its trend during our period. This group of offenses starts at its highest amount at almost 3500 convictions per month in January 2014. Then the conviction rate declined moderately until August 2015. From August to October 2015, its amount plunged to almost 75% of its initial value. And finally, it maintained for the rest of the period.

As there is no information about the crimes in this group, we can't find any association between this index and other categories. Nonetheless, we can confidently say some significant changes happened from August to October 2015, which resulted in this steep reduction.

drugs_offences & public_order_offences:

These two categories can be considered as categories with a moderate number of convictions. Compared to the three most common crime groups, these two groups' proportion had the smallest share, almost half of them. Both followed almost the same trend with even the same fluctuation. Therefore, we can understand that there was an association between the convictions of these two groups. Both segments started at their highest points at around 5000 units monthly, then they decreased gradually with a moderate fluctuation. The portion of public_order_offences was always below the drugs_offences portion throughout this period.

The reason why both followed the same trend is not interpretable regarding this chart. However, there is a direct underlying connection between these offenses which should be investigated by relevant organizations. Overall, considering the descending trend in the number of convictions of these two groups, we can say society had a moderate but sustainable improvement in these areas.

criminal_damage:

This category followed almost the same trend as drugs_offences and public_order_offences. The only substantial difference between this and the previous groups is that its portion is almost half of them. Therefore, all the analysis about those segments also works for this one.

homicide, robbery, fraud_and_forgery, sexual_offences, burglary:

These four groups had the minimum proportion of total convictions. They all almost remained steady during this period. However, they had small fluctuations. It seems we had no change in the performance of the security organizations and also criminal activity in these segments as they maintained their trend. Nonetheless, we can see sexual_offences increased very gently throughout this period, which is not significant compared to other categories at all.

motoring_offences:

The motoring offences were at their highest rate at above 12,500 convictions per month in the first month of 2014. Then, it declined deeply by almost 30% after two months in April 2014. It was the biggest decline in this category throughout these 4 years. After this big plunge, the overall trend of this category was downward; however, there were a huge number of fluctuations. At the end of this period in December 2018, it was at its lowest point at around 6000 convictions monthly. Therefore, motoring offences almost halved their initial amount during this period

, which demonstrates a supreme improvement in the performance of security organizations such as the police.

Compared to the other categories, this one has a significant reduction in terms of the number of convictions, which is a positive development. From my perspective, it is worth investigating what infrastructural changes happened in organizations such as the police and also in people's lifestyles that led to this upgrade. Furthermore, the majority of this improvement happened in the two initial months of 2014. So it could be beneficial to understand why and how this development was so fast in those months. Understanding the impactful parameters of this reduction enables governments to take proactive actions to increase the security of society in the future.

offences_against_the_person:

Despite most of the categories, this one followed a completely different pattern. In January 2014, the number of convictions for this group was around 9000, which had the third biggest share among other crimes. While after a short period, it exceeded motoring_offences in June 2014, and then in March 2015, it exceeded theft_and_handling offences and maintained the biggest share among all different categories for the rest of the period. It fluctuated and increased moderately from 2014 to 2016, and it peaked at its highest rate at almost 11500 units per month in January 2016. However, the upward trend turned into a downward one for the rest of this period.

The different pattern of this group compels us to investigate the reasons the convictions had significantly increased from 2014 to 2016, also which actions not only stopped this upward trend but also turned it into a descending one in half a year. Finding solutions to these questions might help governments learn from their history to prevent mistakes in the future and maximize their outcomes. Moreover, further investigations could explain if it was a systematic issue in their performance or if the result was somehow associated with the overall well-being of the people.

As the overall well-being of a population has a direct impact on the quality of their social interactions, the increase in offences_against_the_person might be an indicator of this issue. Hypothetically, there could be some underlying problems in society that contributed to population anger and anxiety, which led to this increase in this segment. All in all, this difference might provide valuable insights, so it is worth investigating them.

theft_and_handling:

After motoring_offences, this category has the biggest proportion of all crimes in January 2014 at around 11000 crimes monthly. As shown, this category has a moderate reduction rate on a yearly basis; however, we can see many fluctuations throughout this period. The trend it followed was not a straight downward linear one, but if we consider it in a period of 4 years, we can see the overall crime rate in this segment fell more than 50% from 11000 to around 5000 per month, which was an outstanding improvement.

As the improvement for this segment was considerable and continuous compared to the other groups, it is worth looking deeply into the relevant organizations that were responsible for this segment's security. This massive improvement could come from fundamental changes in tactics and strategies of these service providers, or it could even come from the social awareness of society. All in all, studies in this area could bring a keen insight into the steps that led to this enormous improvement.

Stacked Line Graph for Usuccessful Crime Groups

In [528…
```
u_grouped_by_date <- filter(ucrime, area == "National")[,-c(1,2,3,4,19)]
head(u_grouped_by_date)
```

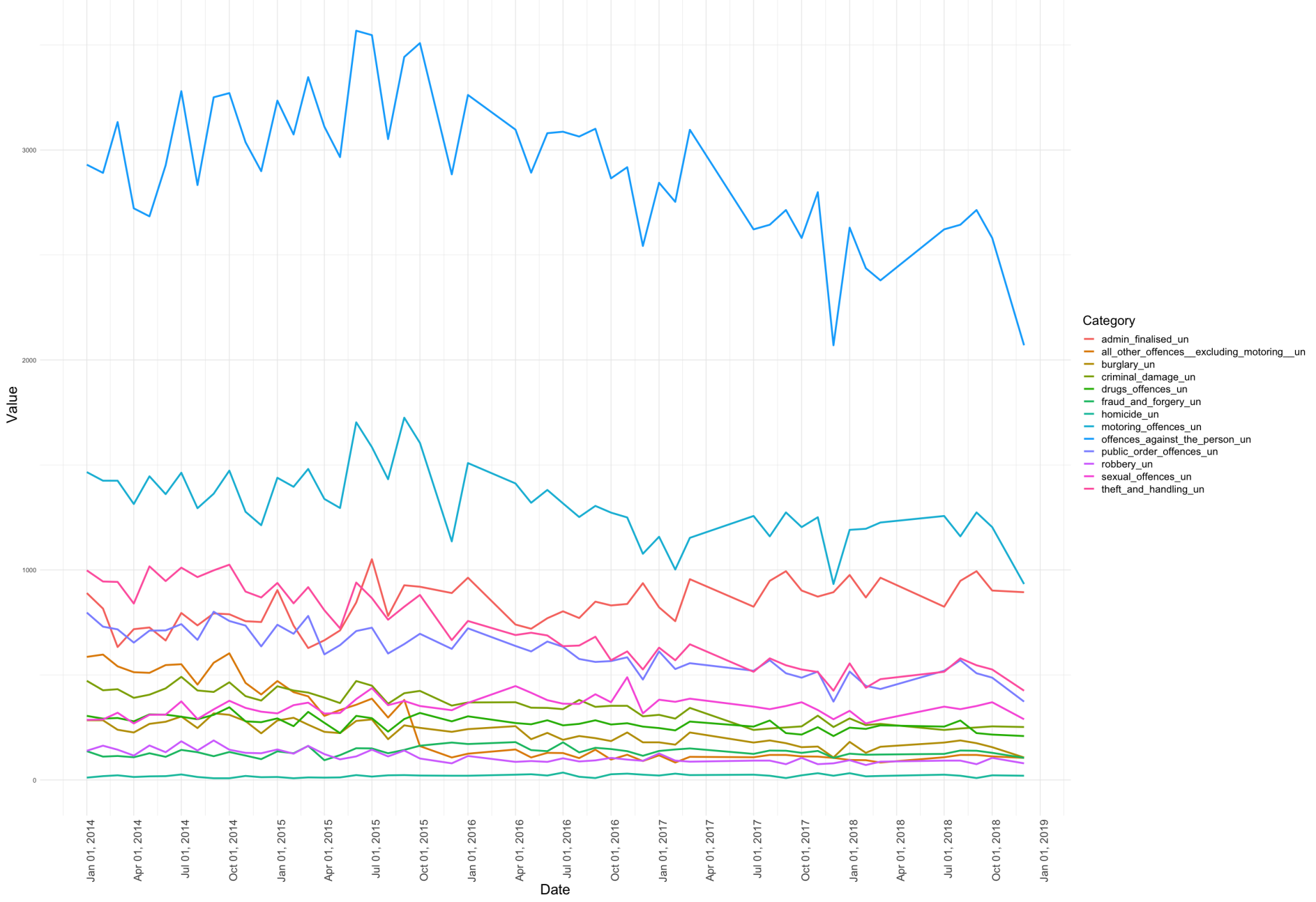
A data.frame: 6 × 1

	date	homicide_un	offences_against_the_person_un	sexual_offences_un	burglary_un	robbery_un	theft_and_handling_un	fraud_and_forgery_un	criminal_
	<date>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	
1	2014-01-01	11	2930	286	284	139	998	137	
2	2014-02-01	18	2891	288	284	163	945	111	
3	2014-03-01	22	3133	320	239	144	943	114	
4	2014-04-01	14	2722	269	226	116	840	108	
5	2014-05-01	17	2684	310	267	164	1017	127	
6	2014-06-01	18	2927	311	277	132	947	110	

Please Note:

if the visualization isn't shown, please rerun the following code

In [529…
```
stacked_line_grapher(u_grouped_by_date)
```



Animated Graph of number of Cnvictions on a Monthly basis

Please Note:

This is an animation graph. The content of this graph is dynamic, while the PDF does not support .gif files. Therefore, to view the result of this plot, please refer to the .ipynb file or open the animated_graph file, which is separately uploaded.

In [115…
```
df_long <- grouped_by_date %>% pivot_longer(cols = -date, names_to = "Category", values_to = "Value")
df_long$Convictions <- as.character(df_long$Value)


options(repr.plot.width = 30, repr.plot.height = 21.4)

# Plot and Animate
a <- ggplot(df_long, aes(x = Category, y = Value, fill = Category)) +
  geom_bar(stat = "identity") +
  labs(title = "Date: {closest_state}") +
  geom_text(aes(label = Convictions, y = Value),
            position = position_dodge(0.9), vjust = -1 ) +
  theme_classic() +
  transition_states(states = date, transition_length = 1, state_length = 1) +
```

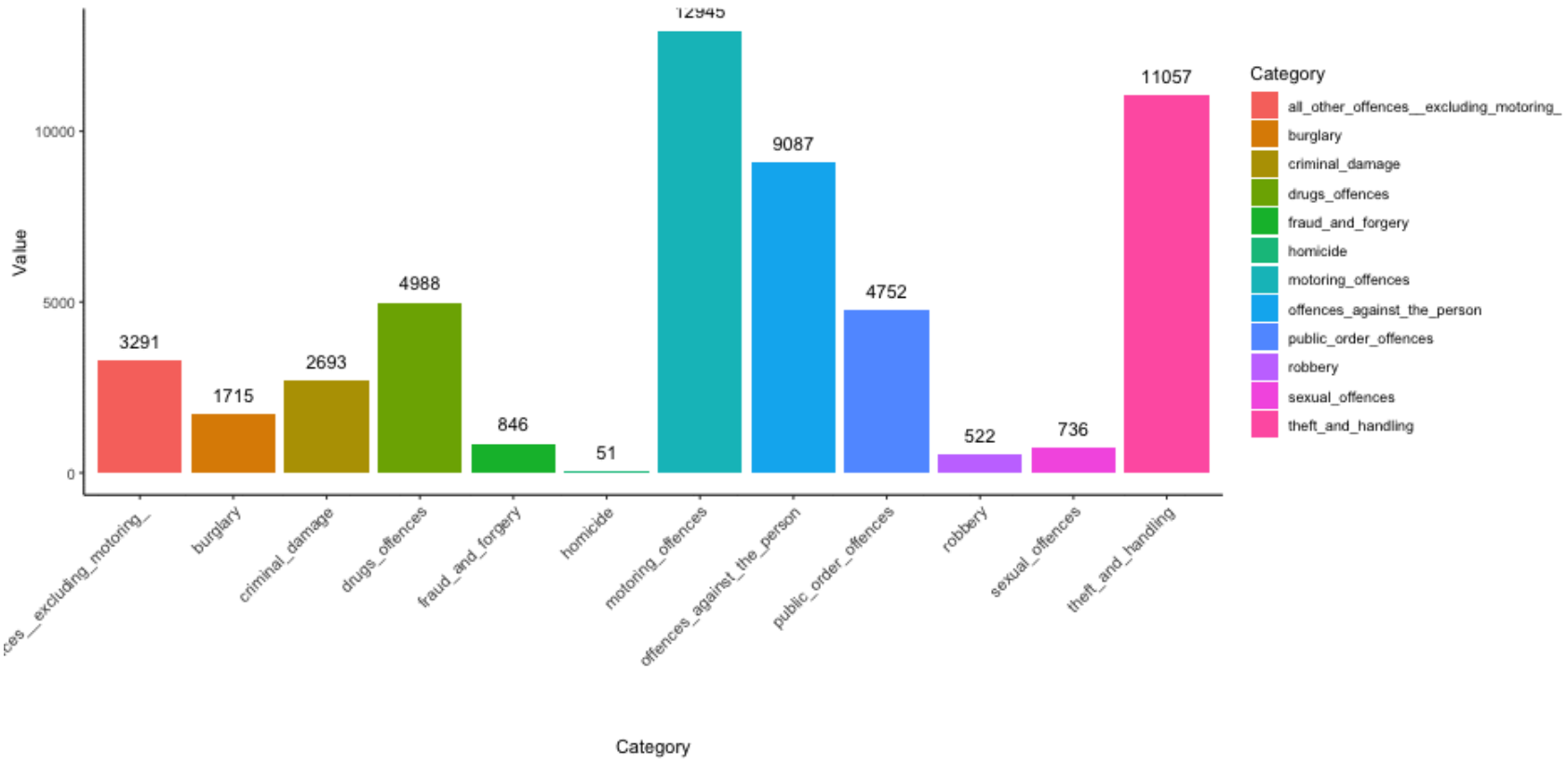
```
  enter_fade() +
  exit_shrink() +
  ease_aes('sine-in-out') +
  theme(plot.title = element_text(hjust = 0.5, vjust = 1, size = 20),
        axis.text.x = element_text(angle = 45, hjust = 1, size = 10))

# Add pauses between frames
a_slow <- animate(a, width = 940, height = 480, nframes = 2*length(unique(df_long$date)) * 2, fps = 5)
```

```
Rendering [------------------------------------------] at 5.1 fps ~ eta: 39s
Rendering [>-----------------------------------------] at 5.1 fps ~ eta: 39s
Rendering [>-----------------------------------------] at 5 fps ~ eta: 39s
Rendering [>-----------------------------------------] at 4.3 fps ~ eta: 45s
Rendering [>-----------------------------------------] at 4.4 fps ~ eta: 44s
Rendering [=>----------------------------------------] at 4.4 fps ~ eta: 43s
Rendering [=>----------------------------------------] at 4.5 fps ~ eta: 43s
Rendering [=>----------------------------------------] at 4.6 fps ~ eta: 42s
Rendering [=>----------------------------------------] at 4.6 fps ~ eta: 41s
Rendering [==>---------------------------------------] at 4.7 fps ~ eta: 40s
Rendering [==>---------------------------------------] at 4.7 fps ~ eta: 39s
Rendering [==>---------------------------------------] at 4.8 fps ~ eta: 39s
Rendering [===>--------------------------------------] at 4.8 fps ~ eta: 39s
Rendering [===>--------------------------------------] at 4.8 fps ~ eta: 38s
Rendering [===>--------------------------------------] at 4.8 fps ~ eta: 37s
Rendering [====>-------------------------------------] at 4.8 fps ~ eta: 37s
Rendering [====>-------------------------------------] at 4.9 fps ~ eta: 37s
Rendering [====>-------------------------------------] at 4.9 fps ~ eta: 36s
Rendering [=====>------------------------------------] at 4.9 fps ~ eta: 36s
Rendering [=====>------------------------------------] at 4.9 fps ~ eta: 35s
Rendering [======>-----------------------------------] at 4.9 fps ~ eta: 35s
Rendering [======>-----------------------------------] at 4.9 fps ~ eta: 34s
Rendering [=======>----------------------------------] at 4.9 fps ~ eta: 34s
Rendering [=======>----------------------------------] at 4.9 fps ~ eta: 33s
Rendering [========>---------------------------------] at 4.9 fps ~ eta: 33s
Rendering [========>---------------------------------] at 4.9 fps ~ eta: 32s
Rendering [=========>--------------------------------] at 4.9 fps ~ eta: 32s
Rendering [=========>--------------------------------] at 4.9 fps ~ eta: 31s
Rendering [==========>-------------------------------] at 4.9 fps ~ eta: 31s
Rendering [==========>-------------------------------] at 4.9 fps ~ eta: 30s
Rendering [===========>------------------------------] at 4.9 fps ~ eta: 30s
Rendering [===========>------------------------------] at 4.9 fps ~ eta: 29s
Rendering [============>-----------------------------] at 4.9 fps ~ eta: 29s
Rendering [============>-----------------------------] at 4.9 fps ~ eta: 28s
Rendering [=============>----------------------------] at 4.9 fps ~ eta: 28s
Rendering [=============>----------------------------] at 4.9 fps ~ eta: 27s
Rendering [==============>---------------------------] at 4.9 fps ~ eta: 26s
Rendering [==============>---------------------------] at 4.9 fps ~ eta: 26s
Rendering [===============>--------------------------] at 4.9 fps ~ eta: 25s
Rendering [===============>--------------------------] at 4.9 fps ~ eta: 25s
Rendering [================>-------------------------] at 4.9 fps ~ eta: 24s
Rendering [================>-------------------------] at 4.9 fps ~ eta: 23s
Rendering [=================>------------------------] at 4.9 fps ~ eta: 23s
Rendering [=================>------------------------] at 4.9 fps ~ eta: 22s
Rendering [==================>-----------------------] at 4.9 fps ~ eta: 22s
Rendering [==================>-----------------------] at 4.9 fps ~ eta: 21s
Rendering [===================>----------------------] at 4.9 fps ~ eta: 21s
Rendering [===================>----------------------] at 4.9 fps ~ eta: 20s
Rendering [====================>---------------------] at 4.9 fps ~ eta: 20s
Rendering [====================>---------------------] at 4.9 fps ~ eta: 19s
Rendering [=====================>--------------------] at 4.9 fps ~ eta: 19s
Rendering [=====================>--------------------] at 4.9 fps ~ eta: 18s
```

```
Rendering [=======================>------------------] at 4.9 fps ~ eta: 18s
Rendering [=======================>------------------] at 4.9 fps ~ eta: 17s
Rendering [=======================>------------------] at 4.9 fps ~ eta: 17s
Rendering [=======================>------------------] at 4.9 fps ~ eta: 16s
Rendering [========================>-----------------] at 4.8 fps ~ eta: 16s
Rendering [========================>-----------------] at 4.8 fps ~ eta: 16s
Rendering [========================>-----------------] at 4.8 fps ~ eta: 15s
Rendering [=========================>----------------] at 4.8 fps ~ eta: 14s
Rendering [=========================>----------------] at 4.8 fps ~ eta: 13s
Rendering [=========================>----------------] at 4.8 fps ~ eta: 13s
Rendering [=========================>----------------] at 4.8 fps ~ eta: 12s
Rendering [=========================>----------------] at 4.8 fps ~ eta: 12s
Rendering [=========================>----------------] at 4.8 fps ~ eta: 11s
Rendering [==========================>---------------] at 4.8 fps ~ eta: 11s
Rendering [==========================>---------------] at 4.8 fps ~ eta: 10s
Rendering [==========================>---------------] at 4.8 fps ~ eta: 10s
Rendering [==========================>---------------] at 4.8 fps ~ eta:  9s
Rendering [==========================>---------------] at 4.8 fps ~ eta:  9s
Rendering [==========================>---------------] at 4.8 fps ~ eta:  8s
Rendering [==========================>---------------] at 4.8 fps ~ eta:  8s
Rendering [==========================>---------------] at 4.8 fps ~ eta:  7s
Rendering [==========================>---------------] at 4.8 fps ~ eta:  7s
Rendering [==========================>---------------] at 4.8 fps ~ eta:  6s
Rendering [==========================>---------------] at 4.8 fps ~ eta:  6s
Rendering [==========================>---------------] at 4.8 fps ~ eta:  5s
Rendering [==========================>---------------] at 4.8 fps ~ eta:  5s
Rendering [==========================>---------------] at 4.7 fps ~ eta:  5s
Rendering [==========================>---------------] at 4.7 fps ~ eta:  4s
Rendering [==========================>---------------] at 4.7 fps ~ eta:  4s
Rendering [==========================>---------------] at 4.7 fps ~ eta:  3s
Rendering [==========================>---------------] at 4.7 fps ~ eta:  3s
Rendering [==========================>---------------] at 4.8 fps ~ eta:  2s
Rendering [==========================>---------------] at 4.8 fps ~ eta:  1s
Rendering [==========================>---------------] at 4.8 fps ~ eta:  1s
Rendering [==========================>---------------] at 4.8 fps ~ eta:  0s
```

In [255…

```
a_slow
```




```
# A tibble: 200 × 7
   format width height colorspace matte filesize density
   <chr>  <int>  <int> <chr>      <lgl>    <int> <chr>
 1 gif      940    480 sRGB       TRUE         0 72x72
 2 gif      940    480 sRGB       TRUE         0 72x72
 3 gif      940    480 sRGB       TRUE         0 72x72
 4 gif      940    480 sRGB       TRUE         0 72x72
 5 gif      940    480 sRGB       TRUE         0 72x72
 6 gif      940    480 sRGB       TRUE         0 72x72
 7 gif      940    480 sRGB       TRUE         0 72x72
 8 gif      940    480 sRGB       TRUE         0 72x72
 9 gif      940    480 sRGB       TRUE         0 72x72
10 gif      940    480 sRGB       TRUE         0 72x72
# ℹ 190 more rows
```

Region-Based Comparison of the Crime Rate

Geographical Heat Map of Crime Rate for England & Wales

In [154…

```r
# Define a function to group our data by the area
group_by_region <- function(dataframe){
  dataframe <- dataframe[,-c(1,3,4,5)]
  dataframe <- group_by(dataframe, area)
  summarise_all(dataframe, list(sum))
}
```

In [155…

```r
# Deploy it
grouped_by_region <- group_by_region(crime)
head(grouped_by_region)
```

A tibble: 6 × 14

area	homicide	offences_against_the_person	sexual_offences	burglary	robbery	theft_and_handling	fraud_and_forgery	criminal_damage	drugs_offe
<chr>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<
Avon and Somerset	109	13081	2184	1704	392	11107	936	3060	
Bedfordshire	45	4241	291	612	348	3661	389	859	
Cambridgeshire	53	5726	445	730	222	4440	485	1175	
Cheshire	55	11260	983	1138	230	8513	722	1985	
Cleveland	45	5856	495	1593	300	11704	536	1864	
Cumbria	20	5012	281	450	72	4242	286	1307	

In [156…

```r
# Removing the National row from the dataset
grouped_by_region = filter(grouped_by_region, area != "National")
head(grouped_by_region)
```

A tibble: 6 × 14

area	homicide	offences_against_the_person	sexual_offences	burglary	robbery	theft_and_handling	fraud_and_forgery	criminal_damage	drugs_offe
<chr>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<
Avon and Somerset	109	13081	2184	1704	392	11107	936	3060	
Bedfordshire	45	4241	291	612	348	3661	389	859	
Cambridgeshire	53	5726	445	730	222	4440	485	1175	
Cheshire	55	11260	983	1138	230	8513	722	1985	
Cleveland	45	5856	495	1593	300	11704	536	1864	
Cumbria	20	5012	281	450	72	4242	286	1307	

In [157…

```r
#Creating a dataframe contains lattitude and longitude of the different areas
area_map <- data.frame(
  area = c("Avon and Somerset", "Bedfordshire", "Cambridgeshire", "Cheshire", "Cleveland", "Cumbria", "Derbyshire", "Devon and Cornwa
```

```
  Longitude = c(-2.4724, -0.4713, 0.08831, -2.77264, -1.26479, -3.09273, -1.57126, -3.84036, -2.28556, -1.6176, -4.05117, 0.56873, -2
  Latitude = c(51.38897, 52.06864, 52.38235, 53.20986, 54.5715, 54.65139, 53.12793, 50.6697, 50.75391, 54.77868, 52.06963, 51.77291,
)
area_map <- area_map[order(area_map$area), ]
head(area_map)
```

A data.frame: 6 × 3

	area	Longitude	Latitude
	<chr>	<dbl>	<dbl>
1	Avon and Somerset	-2.47240	51.38897
2	Bedfordshire	-0.47130	52.06864
3	Cambridgeshire	0.08831	52.38235
4	Cheshire	-2.77264	53.20986
5	Cleveland	-1.26479	54.57150
6	Cumbria	-3.09273	54.65139

In [158…

```
# Adding the total crime number to the area_map dataframe
area_map$crime <- grouped_by_region$ALL
head(area_map)
```

A data.frame: 6 × 4

	area	Longitude	Latitude	crime
	<chr>	<dbl>	<dbl>	<dbl>
1	Avon and Somerset	-2.47240	51.38897	53703
2	Bedfordshire	-0.47130	52.06864	17708
3	Cambridgeshire	0.08831	52.38235	21409
4	Cheshire	-2.77264	53.20986	44326
5	Cleveland	-1.26479	54.57150	33975
6	Cumbria	-3.09273	54.65139	21074

In [175…

```
# Define the bounding box coordinates for England and Wales
ew_bbox <- c(left = -5.797, bottom = 50.064, right = 1.799, top = 55.811)

# Get the map of England and Wales using ggmap
ew_map <- get_stamenmap(ew_bbox, zoom = 7, maptype = "watercolor")

options(repr.plot.width = 15, repr.plot.height =15)

# Create a plot using ggmap and ggplot2
plot <- ggmap(ew_map, extent = "device") +
  geom_point(data = area_map, aes(x = Longitude, y = Latitude, size = crime, color = area), alpha = 0.85) +
  scale_size_continuous(range = c(3, 20)) +
  scale_color_manual(values = rainbow(length(unique(area_map$area)))) +
  geom_text(data = area_map, aes(x = Longitude, y = Latitude, label = area), vjust = -1.5, size = 4) +
  theme_void()
```

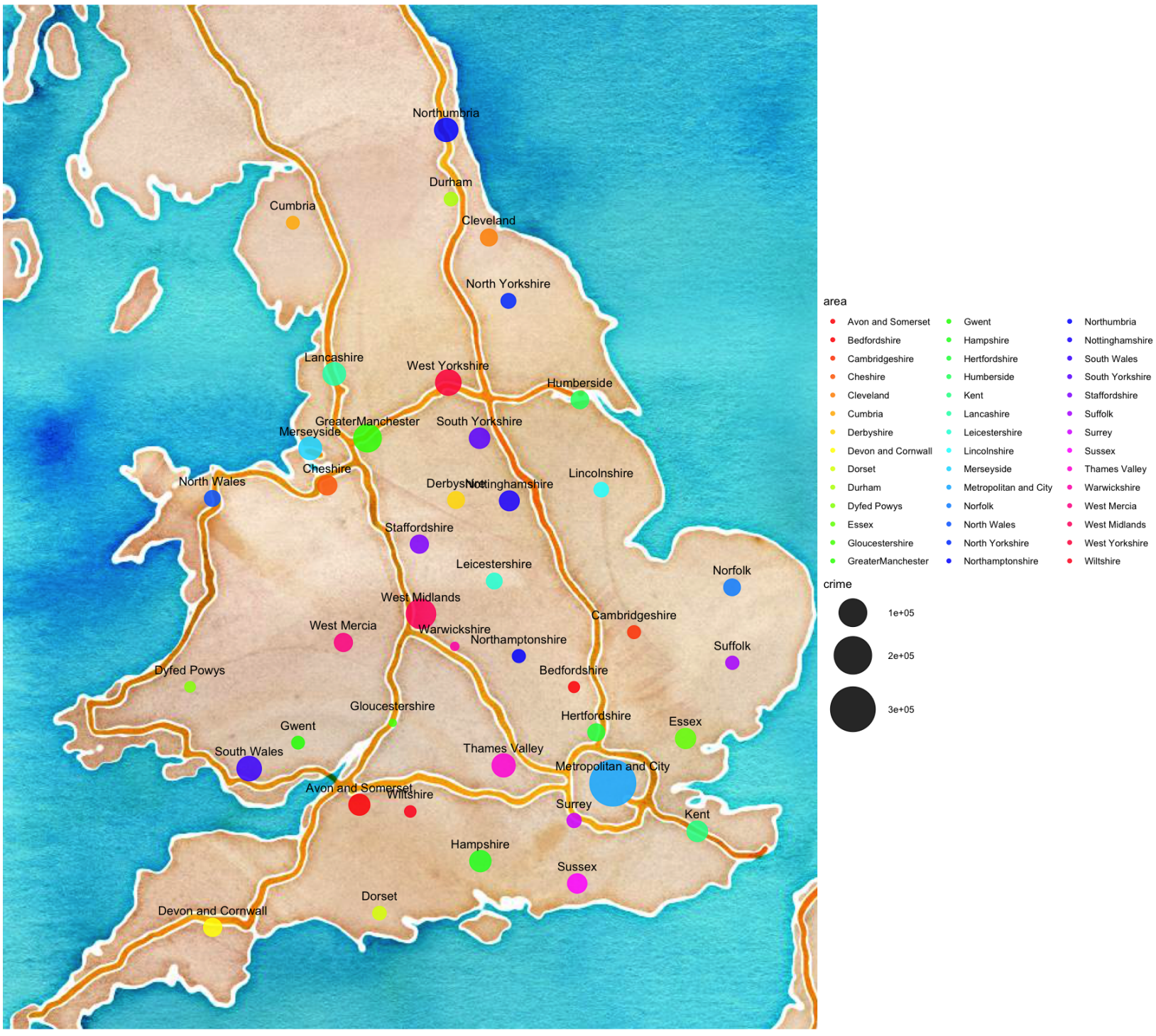
```
i Map tiles by Stamen Design, under CC BY 3.0. Data by OpenStreetMap, under CC BY SA.
```

Please Note:

Sometimes this plot might not run properly based on the network issues from your local machine and google map servers, in that case, please run the above code first, then wait for some seconds, then run the following code.

In [179…

```
# Display the plot
print(plot)
```



Analysis:

This heatmap is a geographical illustration of the crime rate in Wales and England. Bubbles represent areas on our map. The size of bubbles directly relates to the crime rate in that area, which means the bigger the bubble, the higher the crime rate. The Metropolitan and City area located in London, has the most significant size amongst others, demonstrating the massive crime rate in this region. Also, we can see areas such as West Midlands, Greater Manchester, and West Yorkshire with big crime rates compared to other areas. On the other hand, locations such as Gloucestershire and Warwickshire have the lowest crime rate as their bubble is so small.

Stacked Circular Bargraph of Crime Rates Based on Areas

In [180…
```
# Adding regions to the area_map
area_map <- region_mapper(area_map, region_list)
head(area_map)
```

A data.frame: 6 × 5

	area	Longitude	Latitude	crime	region
	<chr>	<dbl>	<dbl>	<dbl>	<chr>
1	Avon and Somerset	-2.47240	51.38897	53703	South
2	Bedfordshire	-0.47130	52.06864	17708	Center
3	Cambridgeshire	0.08831	52.38235	21409	Center
4	Cheshire	-2.77264	53.20986	44326	South
5	Cleveland	-1.26479	54.57150	33975	North
6	Cumbria	-3.09273	54.65139	21074	North

In [181…
```
# Removing Longitue and Latitude from area_map
area_map <- area_map[,-c(2,3)]
head(area_map)
```

A data.frame: 6 × 3

	area	crime	region
	<chr>	<dbl>	<chr>
1	Avon and Somerset	53703	South
2	Bedfordshire	17708	Center
3	Cambridgeshire	21409	Center
4	Cheshire	44326	South
5	Cleveland	33975	North
6	Cumbria	21074	North

In [182…

```
# Rename columns and and
colnames(area_map) <- c('individual', 'value', 'group')
data <- area_map
data <- data[order(data$value), ]
head(data)
```

A data.frame: 6 × 3

	individual	value	group
	<chr>	<dbl>	<chr>
13	Gloucestershire	14852	South
38	Warwickshire	15115	Center
11	Dyfed Powys	17193	South
2	Bedfordshire	17708	Center
42	Wiltshire	18457	South
15	Gwent	20958	South

In [183…

```
# Add a new column for labeling the bars
data$labeler <- paste(as.character(data$individual), '%', as.character(format(round(100 * data$value / sum(data$value), 2), nsmall =
head(data)
```

A data.frame: 6 × 4

	individual	value	group	labeler
	<chr>	<dbl>	<chr>	<chr>
13	Gloucestershire	14852	South	Gloucestershire% 0.74
38	Warwickshire	15115	Center	Warwickshire% 0.75
11	Dyfed Powys	17193	South	Dyfed Powys% 0.85
2	Bedfordshire	17708	Center	Bedfordshire% 0.88
42	Wiltshire	18457	South	Wiltshire% 0.92
15	Gwent	20958	South	Gwent% 1.04

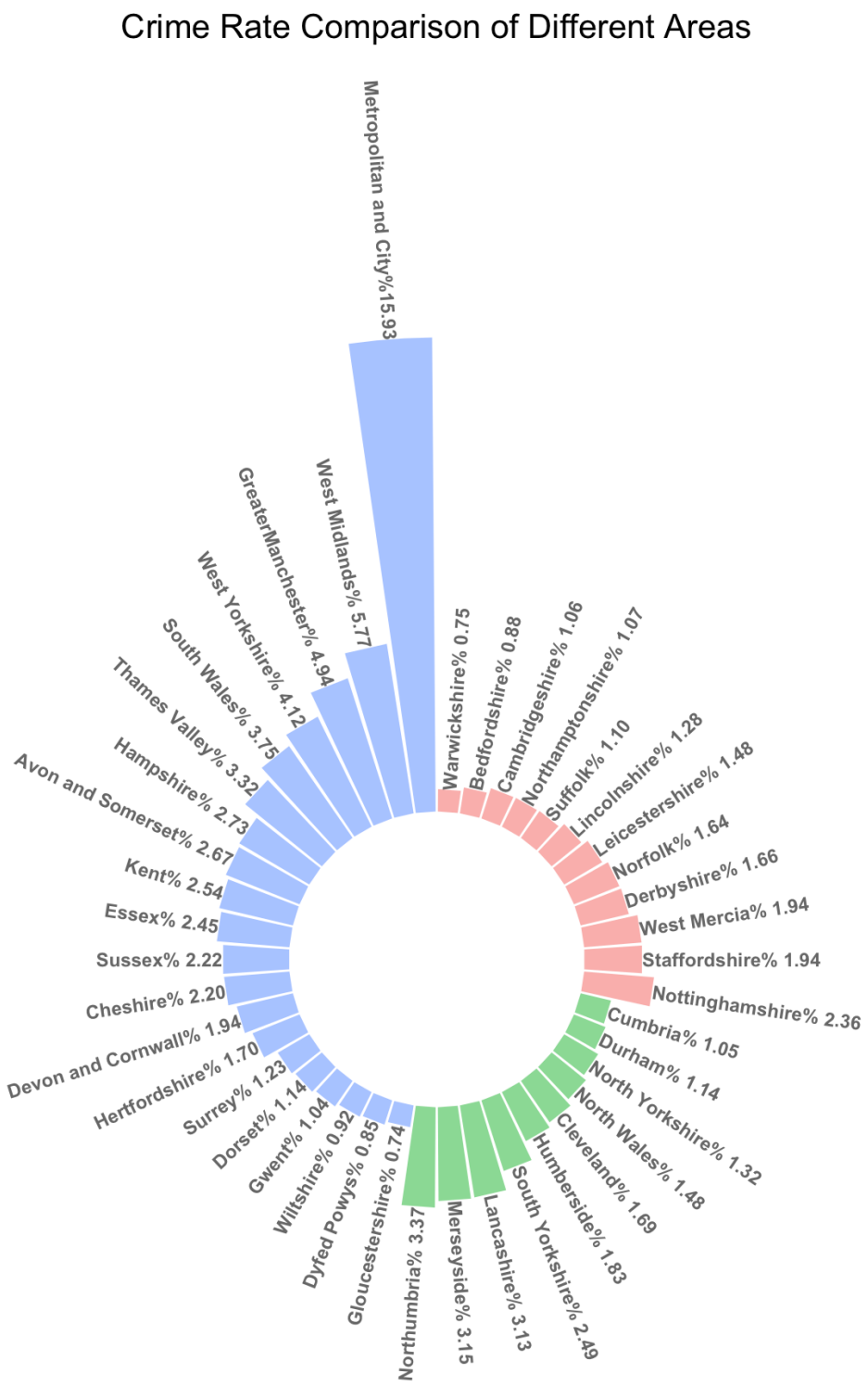
In [184…

```
# Create a function for plotting circular stacked bar graph
circular_stacked_bar_plotter <- function(data){
empty_bar <- 4
to_add <- data.frame(matrix(NA, empty_bar * nlevels(data$group), ncol(data)))
colnames(to_add) <- colnames(data)
to_add$group <- rep(levels(data$group), each = empty_bar)
data <- rbind(data, to_add)
data <- data %>% arrange(group)
data$id <- seq(1, nrow(data))

# Get the name and the y position of each label
label_data <- data
number_of_bar <- nrow(label_data)
angle <- 90 - 360 * (label_data$id - 0.5) / number_of_bar
label_data$hjust <- ifelse(angle < -90, 1, 0)
label_data$angle <- ifelse(angle < -90, angle + 180, angle)

options(repr.plot.width = 18, repr.plot.height = 20)

# Make the plot
p <- ggplot(data, aes(x = as.factor(id), y = value, fill = group)) +
  geom_bar(stat = "identity", alpha = 0.5) +
  ylim(-100000, 390444) +
  theme_minimal() +
  theme(
    legend.position = "bottom",
    axis.text = element_blank(),
    axis.title = element_blank(),
    panel.grid = element_blank(),
    plot.margin = unit(rep(-1, 4), "cm"),
    plot.title = element_text(hjust = 0.5, vjust = 1, size = 25)
  ) +
  coord_polar() +
  geom_text(
    data = label_data,
    aes(x = id, y = value + 10, label = labeler, hjust = hjust),
    color = "black",
    fontface = "bold",
    alpha = 0.6,
    size = 5,
    angle = label_data$angle,
    inherit.aes = FALSE
  ) +
  labs(title = "Crime Rate Comparison of Different Areas")
```

```
  return (p)
}
```

In [185…

```
circular_stacked_bar_plotter(data)
```



Analysis:

This stacked circular bar graph demonstrates the portion of crime rates in each area during the years from 2014 to 2018. What draws our attention first is that most of the crimes happened in southern England and Wales. The colour blue corresponds to the southern region of these two countries. As can be seen, the Metropolitan and City has the highest portion at %15.93 of total crimes in the UK, significantly higher than other areas. This number seems to be rational as London belongs to this area. As a result, we can anticipate that a massive city such as London, with its enormous population, can hold this many crimes within.

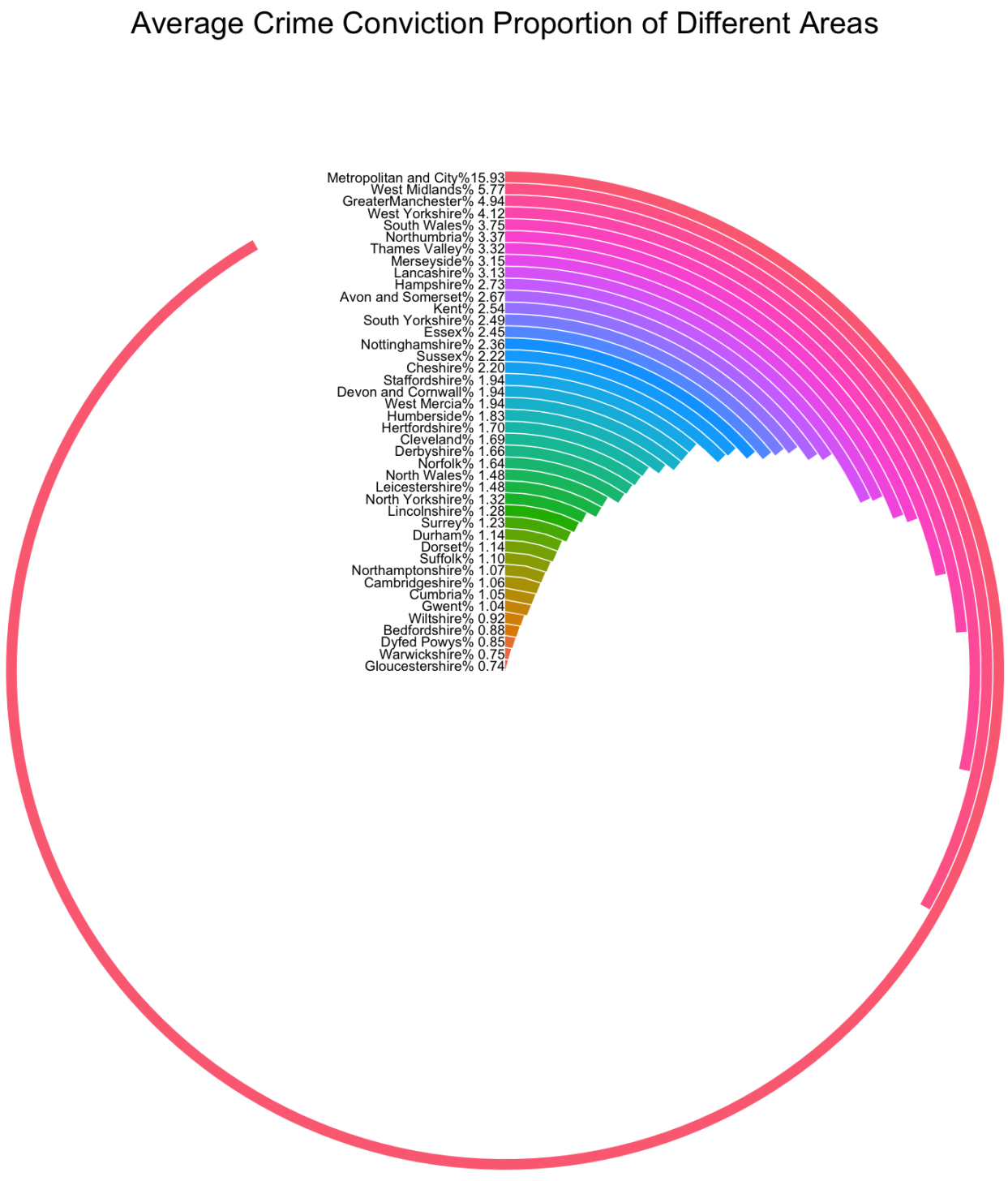
Although the crime rate in the southeast is too high, Gloucestershire, the area with the lowest crime, also belongs to this region. This area just accounts for %0.74 of total crime in the UK. Other areas, such as Warwickshire, with the minimum crime rate, are good places to investigate what social differences resulted in low crime numbers. Although they are less populated than the other areas, which is the most crucial factor, there might be some other underlying reasons contributing to this.

On the other hand, the Central and Northern regions had low crime rates compared to the South. The highest rate in the north and centre belongs to Northumbria and Nottinghamshire, respectively. The former accounting for %3.37, and the latter contains %2.36 of total crime.

Circular Barplot

In [186…

```
# make sample dataframe

Category <- data[order(-data$value),]$labeler
Percent <- data[order(-data$value),]$value

internetImportance<-data.frame(Category,Percent)

# append number to category name
internetImportance$Category <-
     paste0(internetImportance$Category)

# set factor so it will plot in descending order
internetImportance$Category <-
    factor(internetImportance$Category,
    levels=rev(internetImportance$Category))


options(repr.plot.width = 15, repr.plot.height = 15)


# plot

ggplot(internetImportance, aes(x = Category, y = Percent,
    fill = Category)) +
    geom_bar(width = 0.9, stat="identity") +
    coord_polar(theta = "y") +
    xlab("") + ylab("") +
    ylim(c(0,350000)) +
    ggtitle("Average Crime Conviction Proportion of Different Areas") +
    geom_text(data = internetImportance, hjust = 1, size = 4,
              aes(x = Category, y = 0, label = Category)) +
    theme_minimal() +
    theme(legend.position = "none",
     panel.grid.major = element_blank(),
     panel.grid.minor = element_blank(),
     axis.line = element_blank(),
     axis.text.y = element_blank(),
     axis.text.x = element_blank(),
     axis.ticks = element_blank(),
    plot.title = element_text(hjust = 0.5, vjust = 1, size = 25))
```



Analysis:

This is another visualization which demonstrates the convictions portion of different areas for comparison. It offers almost the same insights as the previous comparison. However, users can choose between these two depending on their preferences.

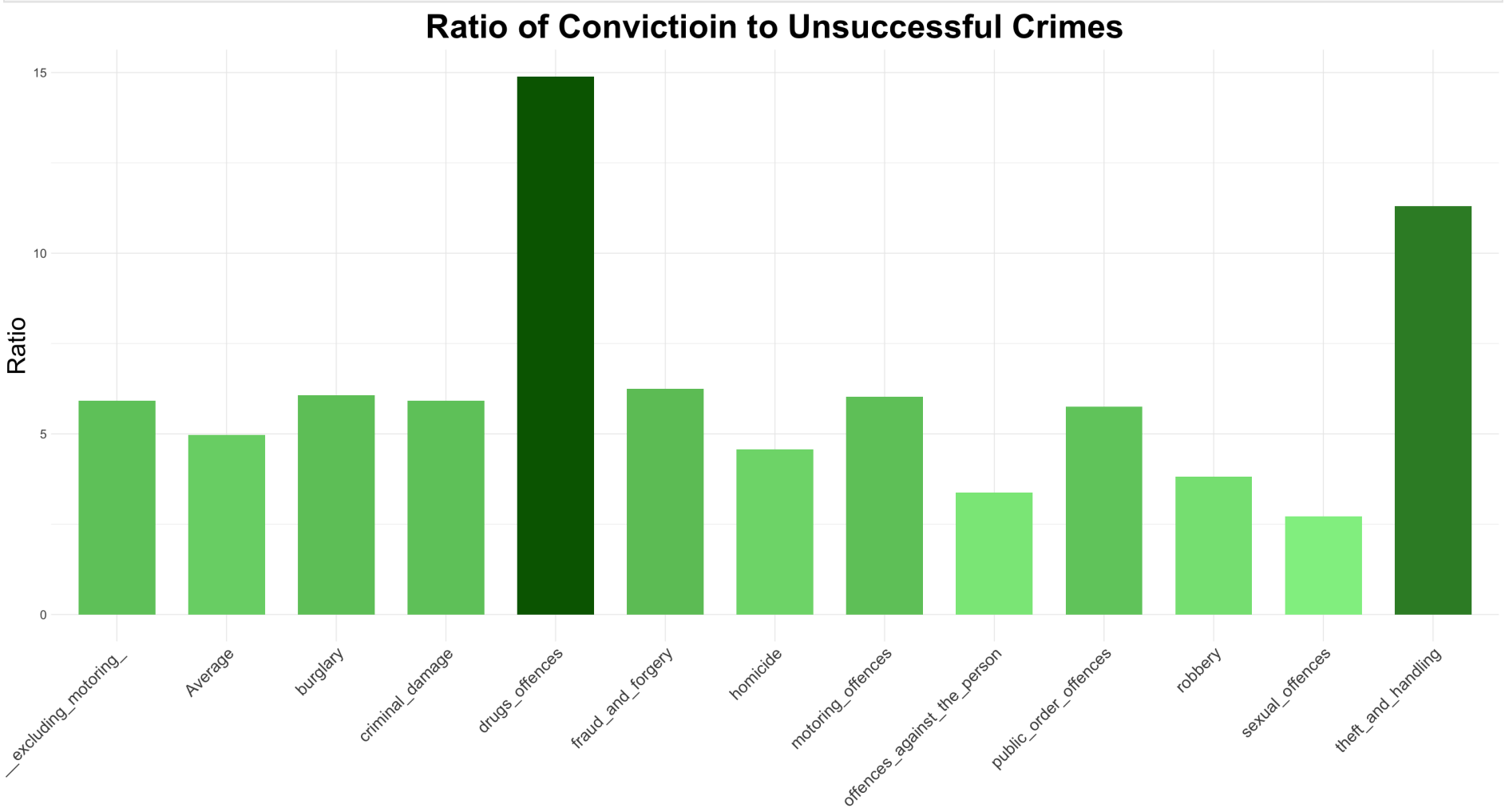
Comparing The Successful & Unsuccessful Crime Ratio in Different Categories

In [187…

```
ratio_plotter <- function(data1, data2){
    u_grouped_by_date <- filter(data2, area == "National")[,-c(1,2,3,4,5)]
    grouped_by_date <- filter(data1, area == "National")[,-c(1,2,3,4,5)]
    ratio_df <- grouped_by_date/u_grouped_by_date[,-c(13)]
    colnames(ratio_df)[13] <- 'Average'


    # Calculate column averages
    column_averages <- colMeans(ratio_df)

    # Convert the column_averages into a data frame
    df <- data.frame(Column = colnames(ratio_df), Ratio = column_averages)

    options(repr.plot.width = 25.2, repr.plot.height = 15)

    # Define the color palette
    num_shades <- nrow(df)
    green_palette <- colorRampPalette(c("lightgreen", "darkgreen"))
    green_colors <- green_palette(num_shades)

    # Create the barplot using ggplot2
    plot <- ggplot(df, aes(x = Column, y = Ratio, fill = Ratio)) +
      geom_bar(stat = "identity", width = 0.7) +
      labs(title = "Ratio of Convictioin to Unsuccessful Crimes",
           x = "Category",
           y = "Ratio") +
      scale_fill_gradient(low = "lightgreen", high = "darkgreen") +
      theme_minimal() +
      theme(plot.title = element_text(hjust = 0.5, vjust = 1, size = 40, face = "bold"),
            axis.text.x = element_text(angle = 45, hjust = 1, size = 20),
            axis.text.y = element_text( size = 15),
            axis.title = element_text(size = 30),
            legend.position = "none")

    return(plot)
}
```

In [188…

```
ratio_plotter(crime, ucrime)
```



Analysis & Hypothesis:

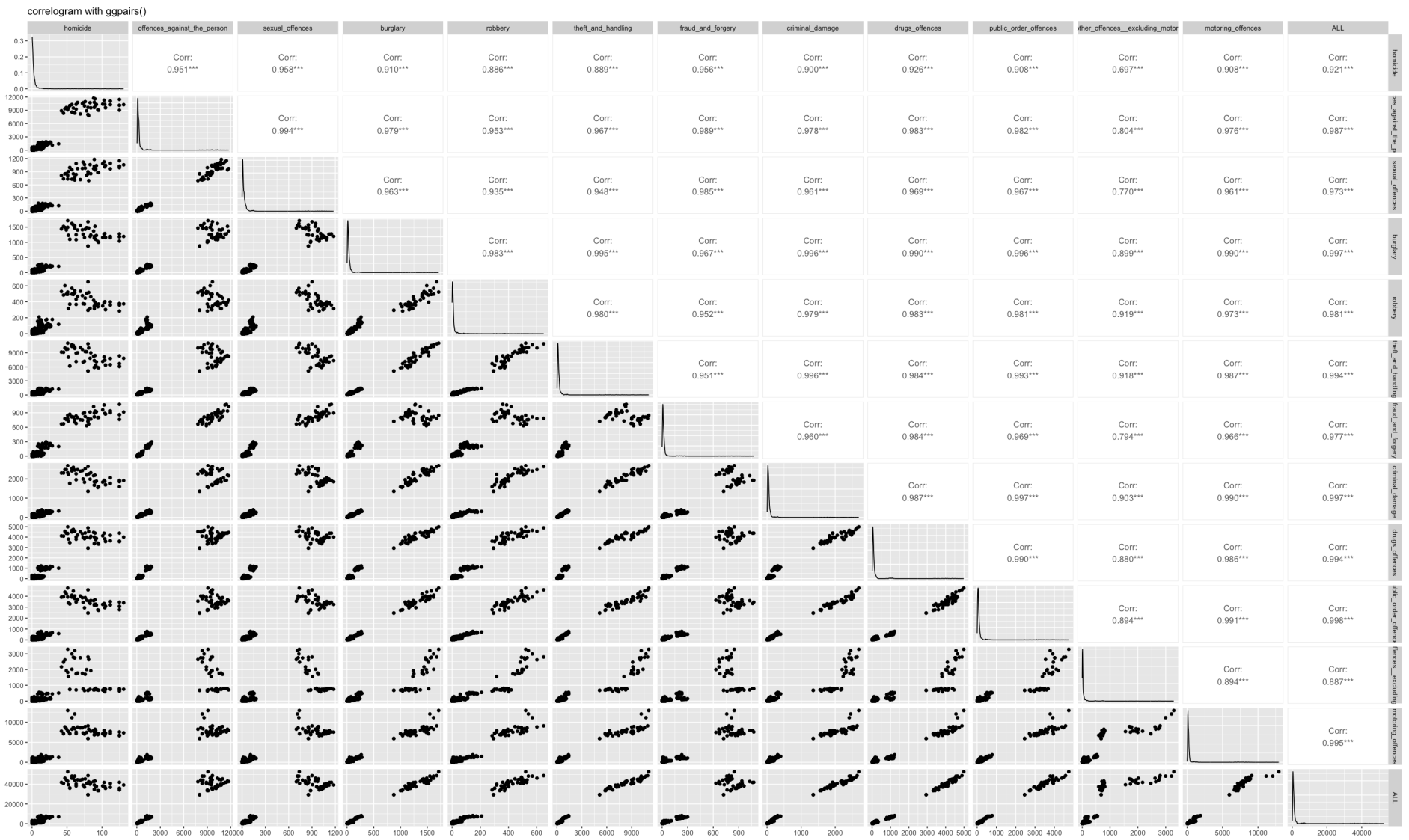
In some crime categories such as drug offences or theft and handling, we can see that the number of convictioins to unsuccessful is too high which means when someone is accused for these crimes, the court final decision is almost always against them. Furthermore, we can assume that even lawyers can't save them as the court and law is restrictive and firm about these kind of offences. Moreover, We can Understand that proving these crimes might be easier than other categories. On the other hand, sexual offences ot offences against the person have the smallest ratio among other categories. The reason might be that usually in these types of crime, proving someone is really guilty is so har. There need to be strong evidence to charge the criminal. However, as there might be few witnesses for these kinds of crime, the number of unsuccessuls are high comparing to convictions.

Correlation Analysis

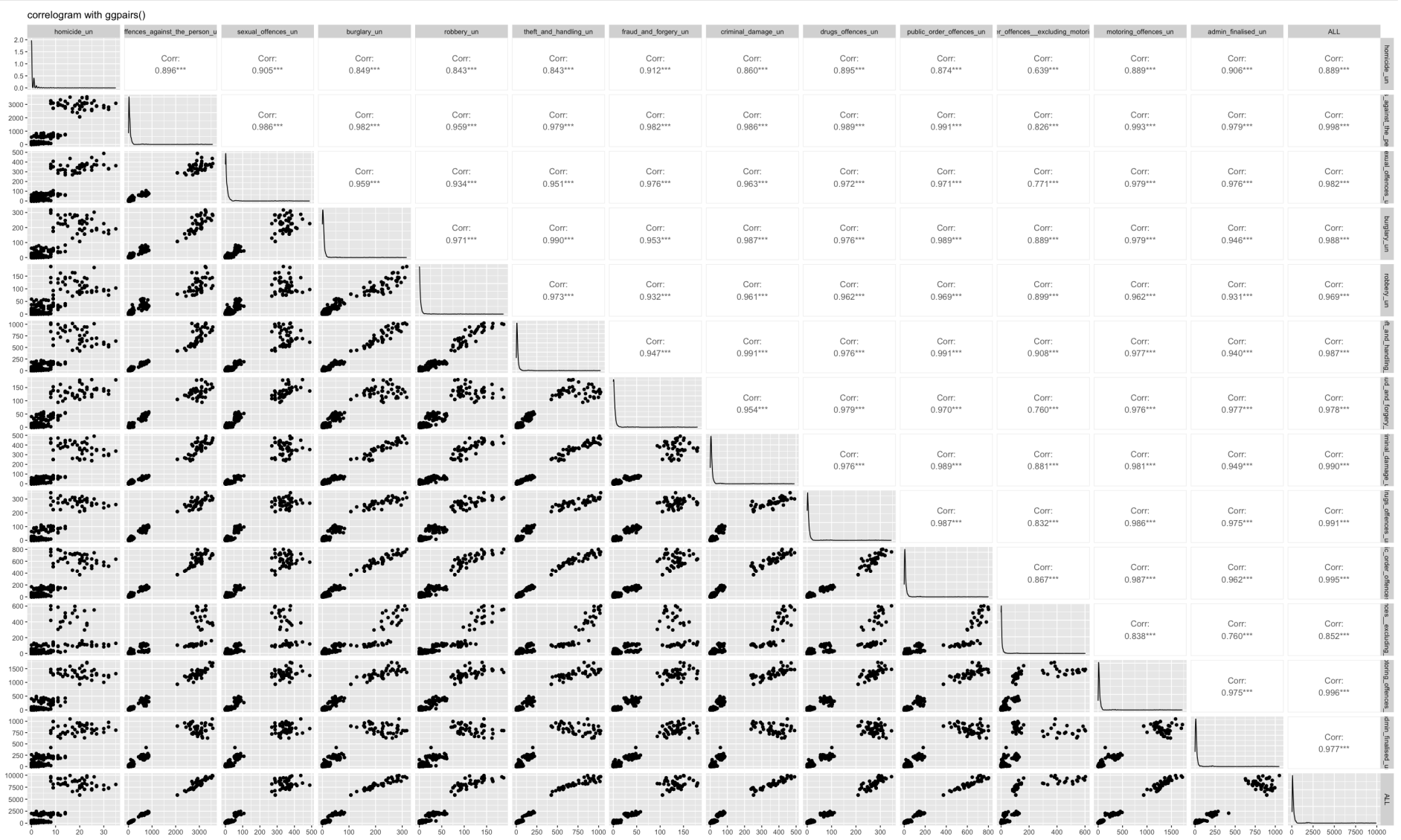
Scatter Plot and Correlation Between Crime Categories:

In [530…
```
# Define a correlation visualiser function
correlation_maker <- function(data){
options(repr.plot.width = 25.2, repr.plot.height = 15)
plot <- ggpairs(data[,-c(1,2,3,4,5)] , title="correlogram with ggpairs()")
return (plot)
}
```

In [190…
```
# Correlation of crime categories for convictions
correlation_maker(crime)
```



In [191…
```
# Correlation of crime categories for Unsuccessful
correlation_maker(ucrime)
```



Analysis:

The first correlation chart represents the correlation of convictions in different categories, while the second one is for unsuccessful crimes. When we compare the trends of convictions and unsuccessful crimes, we observe that both of them follow the same pattern for almost all kinds of visualizations. Consequently, we expect both correlation graphs to show almost the same correlations for similar crime categories. By considering these graphs and comparing the correlation numbers, we can see that they support our assumption. According to the stacked line graph visualization, we can see that different categories in both convictions and unsuccessful crimes follow the same pattern. For example, when we see a decrease in offences against persons, we also observe a decrease in sexual offences. Although they do not exactly mirror each other's ups and downs throughout that period, they have close similarities in

fluctuations, though with different intensities. Consequently, we can say that increasing the number of crimes will be distributed in different categories following their previous distribution. For instance, if 100 crimes include ten robberies, by adding another hundred crimes, we expect that around 10 of the new crimes will also be robberies. Our correlation analysis supports this hypothesis, as we can see that the features are highly correlated. Furthermore, all of the correlations are positive, indicating that by increasing one feature, the other feature will also increase. While most categories are highly correlated, a few, such as offences excluding motoring, are less correlated. Although the correlation coefficient for most feature pairs is above 0.85, this particular category has smaller values, such as 0.69.

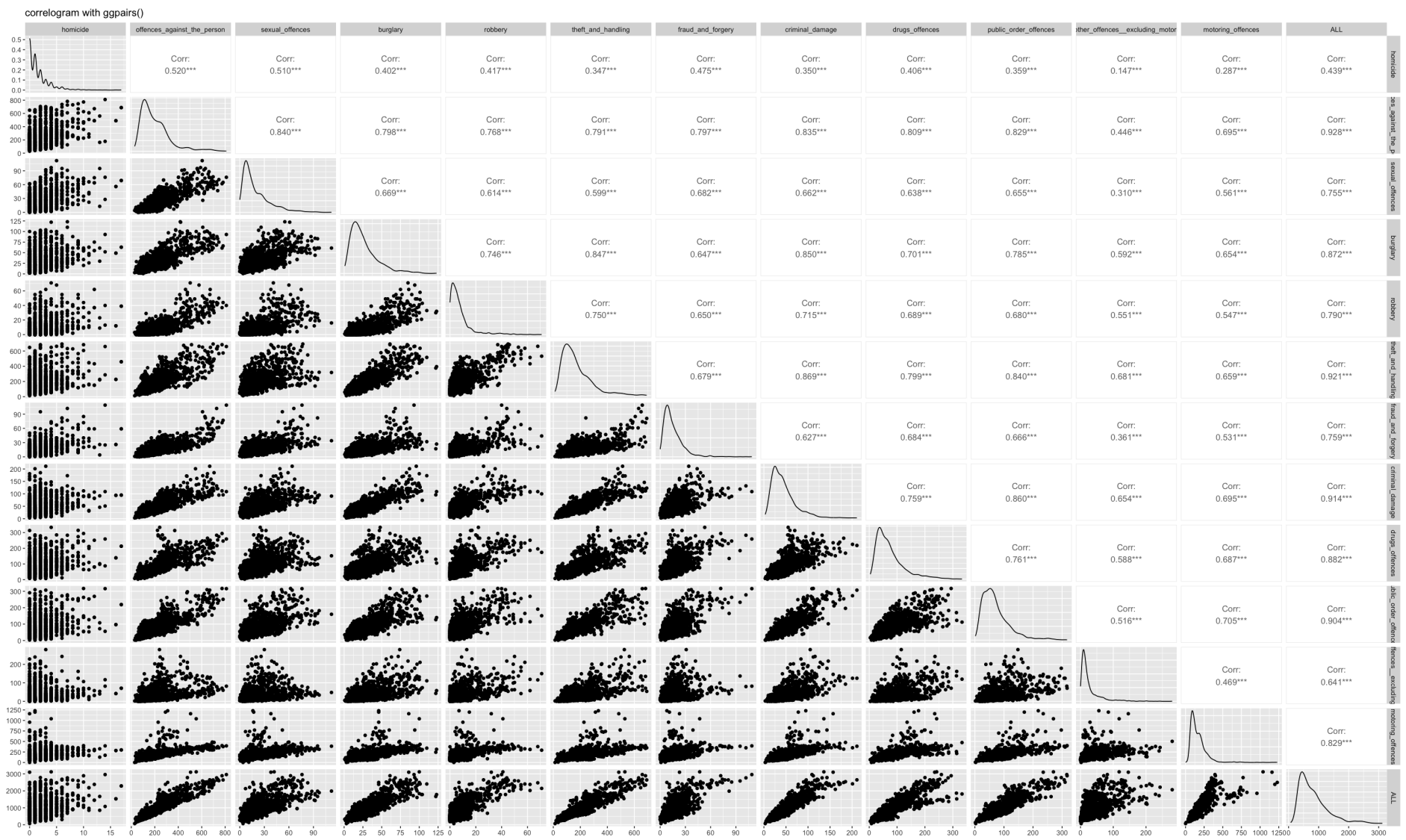
Note:

This correlation analysis may be unreliable as we conducted it for different areas. Therefore, the correlations of different areas could be entirely different based on the crime rate in those areas. Additionally, we can observe three distinct areas in the scatter plots of the correlation analysis for all pairs. The first one is densely located in the bottom left of the graph, representing crimes in areas with a small proportion of the total crimes. The second one is close to the previous cluster and refers to crimes in metropolitan areas and cities. The last cluster belongs to the top corner of the graph, representing national crimes. Therefore, it is understandable that the correlations are high, as the correlation pattern appears more like a straight line between these three clusters.

Biased Approach (excluding National & Metropolitan and City):

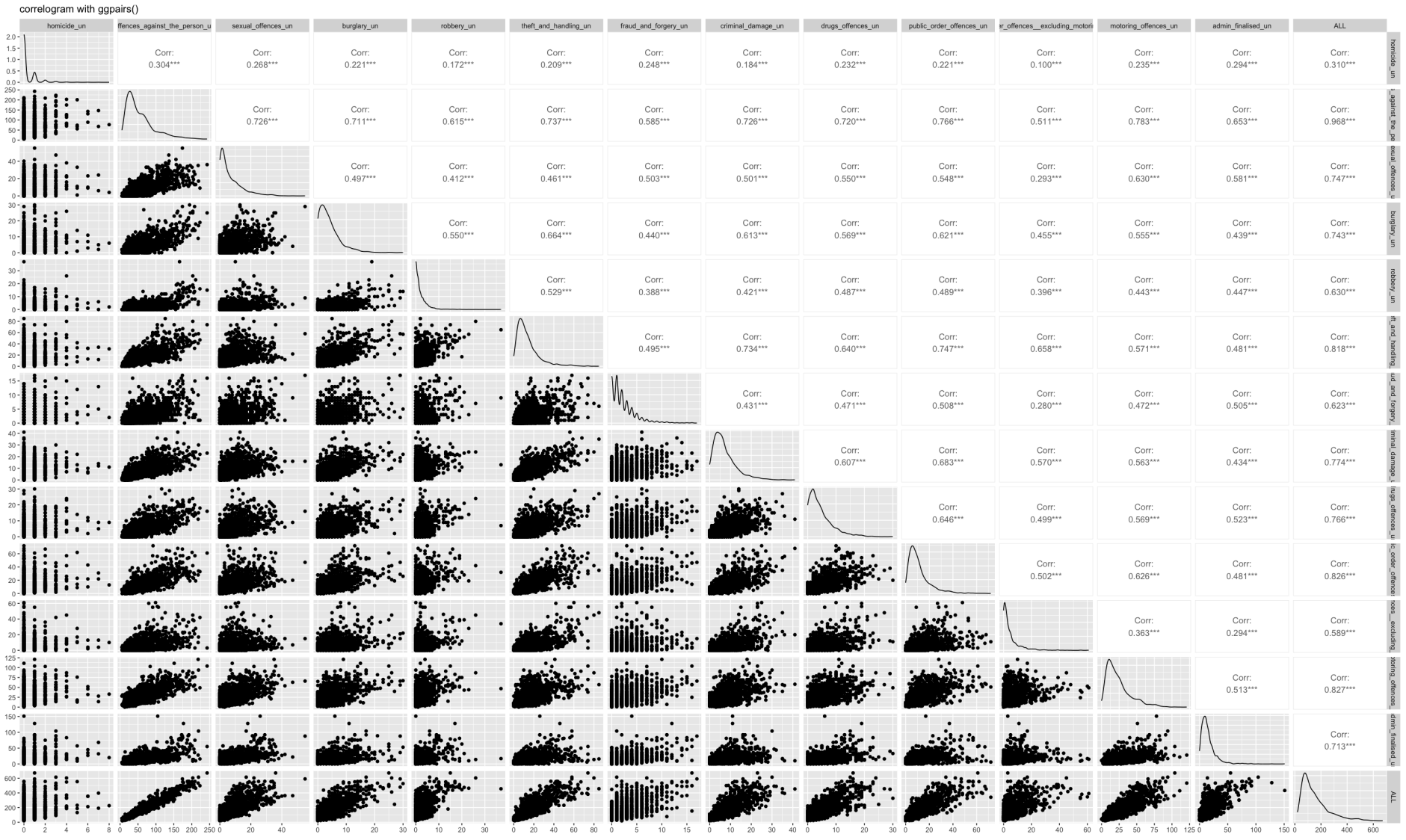
In [192…

```
# Correlation of crime categories for convictions
correlation_maker(filter(crime, area != "National", area !="Metropolitan and City"))
```



In [193…

```
# Correlation of crime categories for Unsuccessful
correlation_maker(filter(ucrime, area != "National", area !="Metropolitan and City"))
```



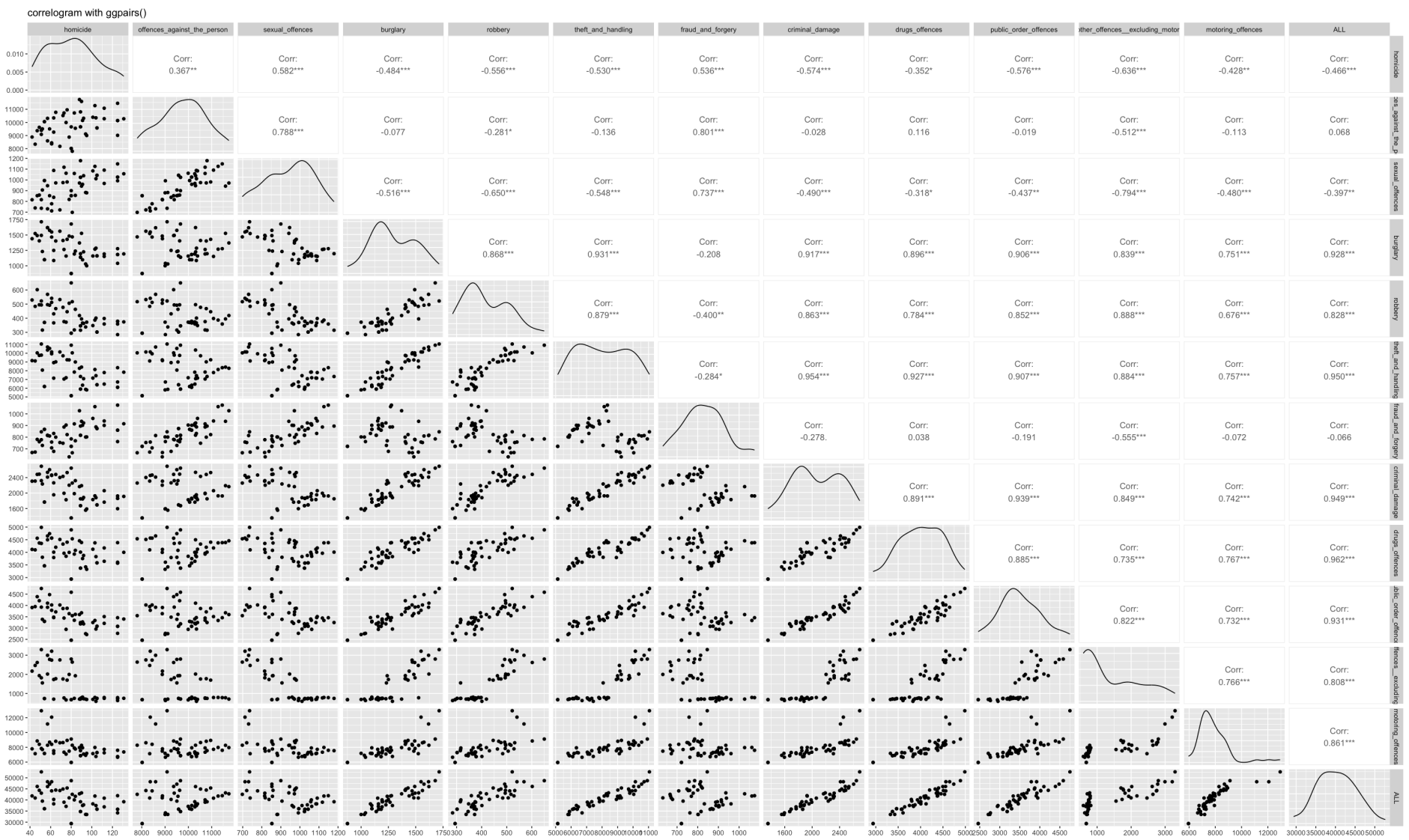
Analysis:

this correlation analysis demonstrate the correlation between different crime categories in the whole country excluding "National" and "Metropolitan and City". As is shown, after excluding these cities, the correlations fell considerabely. It proves our hypothesis about the high correlations were right. So basically, the reason of high correlation between different crime categories must be that the crime weight of different areas are very unequally distributed.

Naive Approach:

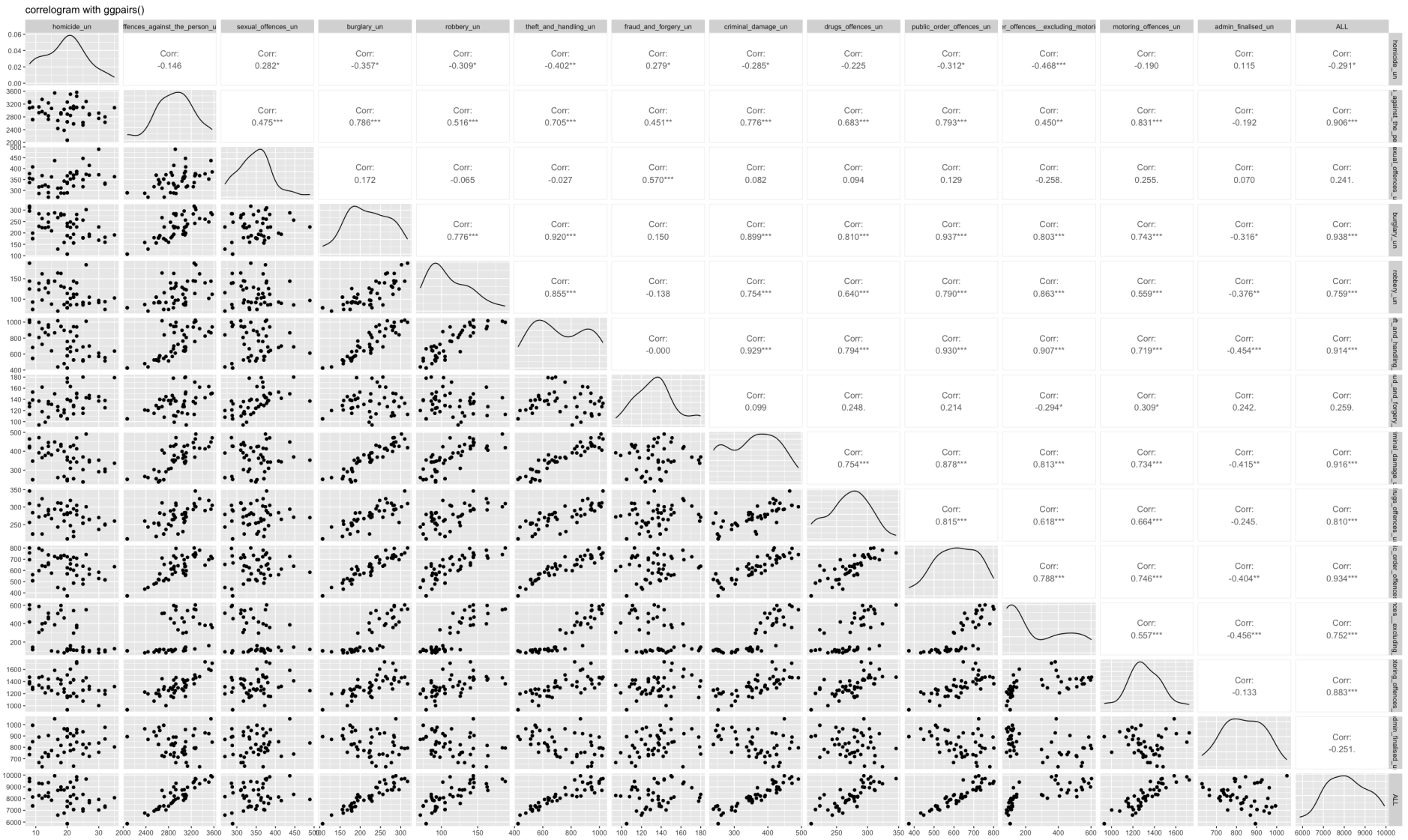
In [194…

```
# Correlation of crime categories for convictions
correlation_maker(filter(crime, area == "National"))
```



In [195…

```
# Correlation of crime categories for Unsuccessful
correlation_maker(filter(ucrime, area == "National"))
```



Analysis:

This correlation analysis Illustrates that if we limit the area to a specific location, we will have a different result. In this case, we limited the correlation to just the national area. With this approach, we naively drop most of the columns and have a very limited number of rows. The result of this analysis can be less valid as we need more data points to interpret the result confidently. However, we can see in this approach, we minimised the skewness of datapoint. Also, by increasing the number of instances for this category, our distributions in different crime groups will become more like a normal distribution, and the skewness will be turned into zero.

Regression

Note:

We will run our models on both conviction and unsuccessful crimes datasets, but we just make analysis on the conviction dataset. Due to the similarities of both models, we assume that the same analysis works also for the unsuccessful dataset.

Simple Linear Regression:

$$Independent\ Variable: X = Date$$

$$Dependent\ Variable: Y = Sum\ of\ Crimes\ in\ Different\ Categories$$

Pipeline:

In this part, we will find the best-fitted line between the date and the number of crimes. To do so, we assume that the date is our independent variable, while the number of crimes is our dependent variable. Therefore, by doing the simple linear regression on this dataset, we will have a model which enables us to predict future crime rates giving it our expected date.

In the visualisation section, we split the datapoints into two sub-dataframe, which are convictions and unsuccessful. That enabled us to interpret visualisations for both categories and compare their ratio and trend simultaneously. In this section, if we just want to use predictive analysis to predict the total number of crimes, we can just create one model which predicts the sum of convictions and unsuccessful based on the original dataset. However, if we want to access the ratio between these two clusters in addition to their total number of crimes, we must deploy two models independently. One is trained on the convictions responsible for predicting the number of convictions. The other one is responsible for predicting the number of unsuccessful crimes. The sum of predictions of these models for a specific day will give us the number of total crimes. Moreover, by dividing the output of these models for a specific date, we can also get the ratio between these two clusters for the given date.

$$Model\_1(date): Predicted\ Number\ of\ Convitions\ for\ date$$

$$Model\_2(date): Predicted\ Number\ of\ Unsuccessful\ for\ date$$

$$Total\ number\ of\ Crimes\ (date) = Model\_1(date) + Model\_2(date)$$

$$Conviction\_Unsuccessful\_Ratio(date) = \frac{Model\_1(date)}{Model\_2(date)}$$

Metrics for Evaluation of the Model:

The metrics we use here for evaluating our model are:

R-squared:

R-squared is a statistical measure indicating how well a regression model fits the observed data. It represents the proportion of the variance in the dependent variable that can be explained by the independent variable(s). Ranging from 0 to 1, a higher value signifies a better fit and stronger relationship between the variables.(Sarkar, 2019)

Root Mean Squared Error (RMSE):

RMSE (Root Mean Square Error) is a statistical measure that quantifies the average difference between predicted and observed values in a regression or prediction model. It is calculated by taking the square root of the mean of the squared differences between predicted and observed values. RMSE provides a single numerical value to evaluate the accuracy and precision of a model, with lower values indicating better performance. (vitalflux.com)

Normalised RMSE:

Normalised RMSE (Root Mean Square Error) is a metric that assesses the accuracy of a model's predictions while considering the variability of the target variable. It is obtained by dividing the RMSE by the range or standard deviation of the target variable, allowing for comparison across different scales. A lower normalised RMSE indicates better prediction performance.

Graph for Visualisation:

Predicted Vs. True Value Graph:

This is a visual representation that compares the predicted values from a model to the actual or true values of the target variable. It shows the relationship between the predicted and true values, allowing for a visual assessment of the model's accuracy. The closer the data points align to a diagonal line, the better the model's predictions align with the true values.

note:

To predict the total number of crimes, we should specify an area for our model. Based on our code, these models predict the number of crimes for the National. Therefore, for using this model to predict other areas, we need to train our model based on the data points for the new location.

In [196…

```r
# Ordering the datasets by date and area
crime <- crime[order(crime$date, crime$area), ]
ucrime <- ucrime[order(ucrime$date, ucrime$area), ]
all_data <- all_data[order(all_data$date, all_data$area), ]
```

In [199…

```r
# Create a function to deploy a simple linear model and interpret and visualize the result
crime_prediction_slm <- function(dataset){
    set.seed(42)
    # Split the dataset into training and test sets
    reg_df <- filter(dataset, area == "National")
    split <- sample.split(reg_df$ALL, SplitRatio = 0.7)
    train_data <- subset(reg_df, split == TRUE)
    test_data <- subset(reg_df, split == FALSE)

    # Perform simple linear regression
    model <- lm(ALL ~ date, data = train_data)

    # Predict on the test set
    predictions <- predict(model, newdata = test_data)

    # Evaluate metrics and score
    RMSE <- caret::RMSE(predictions, test_data$ALL)
    normalized_RMSE <- RMSE / (max(test_data$ALL) - min(test_data$ALL))
    R2 <- summary(model)$r.squared




    # Summary of the regression model
    model_summary <- summary(model)

    # Plot comparison of predictions and true values
    plot_data <- data.frame(Predicted = predictions, True = test_data$ALL)
    plot <- ggplot(plot_data, aes(x = True, y = Predicted)) +
    geom_point(size = 2) +
    geom_abline(intercept = 0, slope = 1, color = "red", linetype = "dashed", linewidth = 1) +
    labs(x = "True Values", y = "Predicted Values", title = "Comparison of Predictions and True Values") +
    theme_minimal() +
    theme(
    axis.text = element_text(size = 21),
    axis.title = element_text(size = 21),
    plot.title = element_text(size = 30),
    axis.text.x = element_text(size = 18),
    axis.text.y = element_text(size = 18)
    )

    # Return results as a list
    results <- list(
    RMSE = RMSE,
    normalized_RMSE = normalized_RMSE,
    R2 = R2,
    ModelSummary = model_summary,
    Plot = plot
    )

    return (results)
}
```

Model 1: Convictions

Simple linear regression for Convictions:

Predictiong the total number of convictions given a specific date.

Deploying the model and save the result

In [200…
```
slm_model_1 = crime_prediction_slm(crime)
RMSE <- slm_model_1$RMSE
normalized_RMSE <- slm_model_1$normalized_RMSE
R2 <- slm_model_1$R2
model_summary <- slm_model_1$ModelSummary
model_1_plot <- slm_model_1$Plot
```

Basic metrics and performance of models

In [201…
```
# Print the metrics and score
cat("Root Mean Squared Error (RMSE):", RMSE, "\n")
cat("Normalized Root Mean Squared Error (RMSE):", normalized_RMSE, "\n")
cat("R-squared (R2):", R2, "\n")
```

```
Root Mean Squared Error (RMSE): 2565.716
Normalized Root Mean Squared Error (RMSE): 0.1336659
R-squared (R2): 0.7428961
```

Model Summary

In [202…
```
print(model_summary)
```

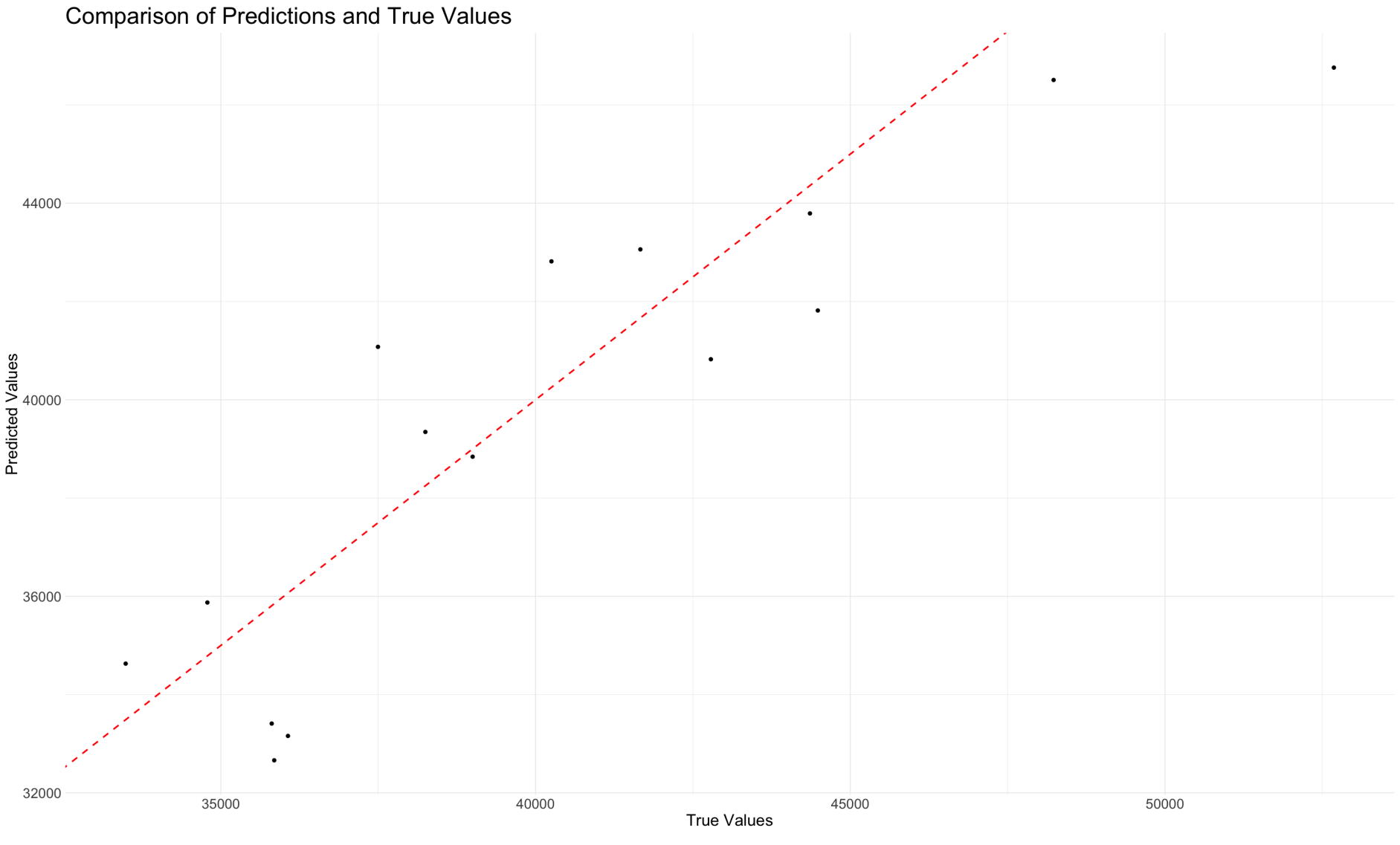
```
Call:
lm(formula = ALL ~ date, data = train_data)

Residuals:
    Min      1Q  Median      3Q     Max
-5766.5 -1471.1   -59.4  1915.4  4410.6

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.774e+05  1.406e+04  12.618 3.55e-14 ***
date        -8.131e+00  8.326e-01  -9.765 2.93e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2510 on 33 degrees of freedom
Multiple R-squared:  0.7429,    Adjusted R-squared:  0.7351
F-statistic: 95.35 on 1 and 33 DF,  p-value: 2.928e-11
```

Analysis:

As can be seen, our simple linear regression model performed well on our convictions dataset. The accuracy of the model is around %74, which is significant regarding the simplicity of our model. Additionally, the normalised RMSE of the model is at around 0.13, which tells us our model predicts the values relatively accurately to the original data points. The problem with simple linear models is that we initially assume that the relation between the variables is linear. Also, this relation is just between one feature and the target. Therefore, even if our dataset follows different patterns, our model can't predict it. Consequently, although this model is very simple and resource-effective, the result is not valuable for interpreting the dataset. To investigate more, we need to deploy more complex models to get a better insight into the dataset.

Plotting the Comparson of Predicted vs True values + regression line

In [203…
```
model_1_plot
```



Analysis:

This graph illustrates the position of the actual values compared to the predicted values. The points in the chart represent the True values, while the dashed red line represents the predicted ones. As we can see, the original position of the data points is almost random without any good bias. Also, the population of the line above and below is equally distributed, indicating our model is not biased.

Model 2: Unsuccessful

Simple linear regression for Unsuccessful:

Predictiong the total number of unsuccessful given a specific date.

Deploying the model and save the result

In [204…
```
slm_model_2 = crime_prediction_slm(ucrime)
RMSE <- slm_model_2$RMSE
normalized_RMSE <- slm_model_2$normalized_RMSE
R2 <- slm_model_2$R2
model_summary <- slm_model_2$ModelSummary
model_2_plot <- slm_model_2$Plot
```

Basic metrics and performance of models

In [205…
```
# Print the metrics and score
cat("Root Mean Squared Error (RMSE):", RMSE, "\n")
cat("Normalized Root Mean Squared Error (RMSE):", normalized_RMSE, "\n")
cat("R-squared (R2):", R2, "\n")
```

```
Root Mean Squared Error (RMSE): 533.4043
Normalized Root Mean Squared Error (RMSE): 0.1778014
R-squared (R2): 0.6734235
```

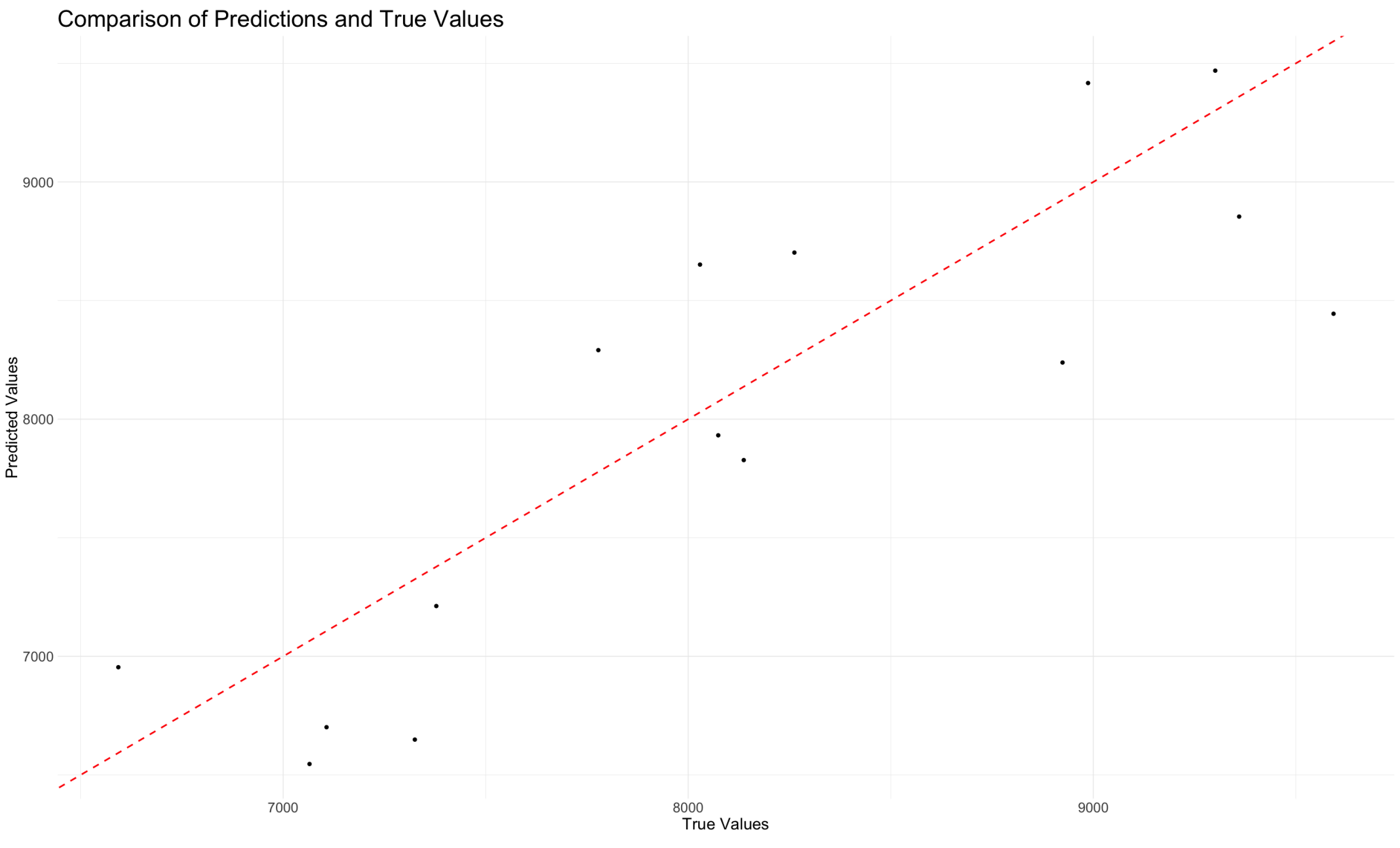
Model Summary

In [206…
```
print(model_summary)
```

```
Call:
lm(formula = ALL ~ date, data = train_data)

Residuals:
     Min       1Q   Median       3Q      Max
-1197.66  -436.81   -47.75   333.51  1390.11

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) 36561.3904  3451.0793  10.594 3.74e-12 ***
date           -1.6858     0.2044  -8.249 1.59e-09 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 616 on 33 degrees of freedom
Multiple R-squared:  0.6734,    Adjusted R-squared:  0.6635
F-statistic: 68.05 on 1 and 33 DF,  p-value: 1.585e-09
```

Plotting the Comparson of Predicted vs True values + regression line

In [207…
```
model_2_plot
```



Multivariate Regression

$$Independent\ Variable: X1 = month\ ,\ X2 = year\ ,\ X3 = area$$

$$Dependent\ Variable: Y = Sum\ of\ Crimes\ in\ Different\ Categories$$

Pipeline:

In this part, we will find the best-fitted line between the "month, area, year" and the number of crimes. To do so, we assume that "month, area, the year" is our independent variable, while the number of crimes is our dependent variable. Therefore, by doing the multivariate linear regression on this dataset, we will have a model which enables us to predict the future crime rates for a specific month of a year in a determined location.

In the visualisation section, we split the data points into two sub-dataframe, which are convictions and unsuccessful. That enabled us to interpret visualisations for both categories and compare their ratio and trend simultaneously. In this section, if we just want to use predictive analysis to predict the total number of crimes, we can just create one model which predicts the sum of convictions and unsuccessful based on the original dataset. However, if we want to access the ratio between these two clusters in addition to their total number of crimes, we must deploy two models independently. One is trained on the convictions responsible for predicting the number of convictions. The other one is responsible for predicting the number of unsuccessful crimes. The sum of predictions of these models for a specific day will give us the number of total crimes. Moreover, by dividing the output of these models for a specific date, we can also get the ratio between these two clusters for the given date.

$$Model\_1(date): Predicted\ Number\ of\ Convitions\ for\ date$$

$$Model\_2(date): Predicted\ Number\ of\ Unsuccessful\ for\ date$$

$$Total\ number\ of\ Crimes\ (date) = Model\_1(date) + Model\_2(date)$$

$$Conviction\_Unsuccessful\_Ratio(date) = \frac{Model\_1(date)}{Model\_2(date)}$$

Metrics for Evaluation of the Model:

The metrics we use here for evaluating our model are:

R-squared:

R-squared is a statistical measure indicating how well a regression model fits the observed data. It represents the proportion of the variance in the dependent variable that can be explained by the independent variable(s). Ranging from 0 to 1, a higher value signifies a better fit and stronger relationship between the variables.(Sarkar, 2019)

Root Mean Squared Error (RMSE):

RMSE (Root Mean Square Error) is a statistical measure that quantifies the average difference between predicted and observed values in a regression or prediction model. It is calculated by taking the square root of the mean of the squared differences between predicted and observed values. RMSE provides a single numerical value to evaluate the accuracy and precision of a model, with lower values indicating better performance. (vitalflux.com)

Normalised RMSE:

Normalised RMSE (Root Mean Square Error) is a metric that assesses the accuracy of a model's predictions while considering the variability of the target variable. It is obtained by dividing the RMSE by the range or standard deviation of the target variable, allowing for comparison across different scales. A lower normalised RMSE indicates better prediction performance.

Graph for Visualisation:

Predicted Vs. True Value Graph:

This is a visual representation that compares the predicted values from a model to the actual or true values of the target variable. It shows the relationship between the predicted and true values, allowing for a visual assessment of the model's accuracy. The closer the data points align to a diagonal line, the better the model's predictions align with the true values.

Residual Plot:

It is a scatter plot showing the residuals (the differences between the observed values and the predicted values) against the corresponding observation numbers. It also includes a dashed red line at y=0 to indicate the ideal or expected residual value of zero. The plot helps assess the distribution and patterns of the residuals in relation to the observations.

Residuals Histogram:

A residuals histogram is a graphical representation that displays the distribution of the residuals from a regression model. It shows the frequency or count of residuals on the y-axis and the range or intervals of residuals on the x-axis. It helps assess the normality assumption and detect any patterns or deviations in the distribution of residuals.

Normality Q-Q plot:

This plot is used to determine the normal distribution of errors. For normally distributed data, observations should lie approximately in a straight line. If the data is non-normal, the points form a curve deviating from a straight line which is a problematic situation.(medium.com)

note:

To predict the total number of crimes, we should specify an area for our model. Based on our code, these models predict the number of crimes for the National. Therefore, for using this model to predict other areas, we need to train our model based on the data points for the new location.

In [208…

```r
# Create a model for deploy, interpret, and visualise the multivariate regression model
perform_regression <- function(dataset, target_variable, independent_variables, split_ratio = 0.7) {
    set.seed(42)
    # Step 1: Split the dataset
    split_data <- caTools::sample.split(dataset[[target_variable]], SplitRatio = split_ratio)
    train_data <- subset(dataset, split_data == TRUE)
```

```r
    test_data <- subset(dataset, split_data == FALSE)

    # Step 2: Perform multivariate regression
    formula <- as.formula(paste(target_variable, paste(independent_variables, collapse = " + "), sep = " ~ "))
    model <- lm(formula, data = train_data)

    # Step 3: Predict on the test set
    predictions <- predict(model, newdata = test_data)

    # Step 4: Evaluate metrics and score
    RMSE <- caret::RMSE(predictions, test_data[[target_variable]])
    normalized_RMSE <- RMSE / (max(test_data[[target_variable]]) - min(test_data[[target_variable]]))
    R2 <- summary(model)$r.squared



    # Plot comparison of predicted and actual values
    plot_data <- data.frame(True = test_data[[target_variable]], Predicted = predictions)
    plot_comparison <- ggplot(plot_data, aes(x = True, y = Predicted)) +
    geom_point(size = 2) +
    geom_abline(intercept = 0, slope = 1, color = "red", linetype = "dashed", size = 1) +
    labs(x = "True Values", y = "Predicted Values", title = "Comparison of Predictions and True Values") +
    theme_minimal() +
    theme(
        axis.text = element_text(size = 21),
        axis.title = element_text(size = 21),
        plot.title = element_text(size = 30),
        axis.text.x = element_text(size = 18),
        axis.text.y = element_text(size = 18)
    )


    # Plot residuals
    residuals <- test_data[[target_variable]] - predictions
    plot_residuals <- ggplot(data.frame(Residuals = residuals), aes(x = seq_along(Residuals), y = Residuals)) +
    geom_point(size = 2) +
    geom_hline(yintercept = 0, color = "red", linetype = "dashed", size = 1) +
    labs(x = "Observation", y = "Residuals", title = "Residual Plot") +
    theme_minimal() +
    theme(
        axis.text = element_text(size = 21),
        axis.title = element_text(size = 21),
        plot.title = element_text(size = 30),
        axis.text.x = element_text(size = 18),
        axis.text.y = element_text(size = 18)
    )


    # Plot smooth histogram of residuals
    resuduals_histogram <- ggplot(data.frame(Residuals = residuals), aes(x = Residuals)) +
    geom_density(fill = "lightblue", color = "black") +
    labs(x = "Residuals", y = "Density", title = "Smooth Histogram of Residuals") +
    theme_minimal() +
    theme(
        axis.text = element_text(size = 21),
        axis.title = element_text(size = 21),
        plot.title = element_text(size = 30),
        axis.text.x = element_text(size = 18),
        axis.text.y = element_text(size = 18)
    )



    # Calculate standardized residuals
    standardized_residuals <- residuals / sd(residuals)

    # Plot Normality Q-Q plot
    plot_normality <- ggplot(data.frame(Standardized_Residuals = standardized_residuals), aes(sample = Standardized_Residuals)) +
    stat_qq() +
    stat_qq_line() +
    labs(x = "Theoretical Quantiles", y = "Standardized Residuals", title = "Normality Q-Q Plot") +
    theme_minimal() +
    theme(
        axis.text = element_text(size = 21),
        axis.title = element_text(size = 21),
        plot.title = element_text(size = 30),
        axis.text.x = element_text(size = 18),
        axis.text.y = element_text(size = 18)
    )


    results <- list(RMSE=RMSE,
                    normalized_RMSE=normalized_RMSE,
                    R2=R2,
                    model_summary = summary(model),
                    comparison_plot = plot_comparison,
                    residuals_plot = plot_residuals,
                    resuduals_histogram=resuduals_histogram,
                    plot_normality=plot_normality)


    # Return the regression model, predictions, and plots
    return(results)
}
```

Model 1: Convictions

Multiariate Linear Regression for Convictions:

Predictiong the total number of convictions given month, area, and year.

Deploying the model and save the result

In [209…
```r
mlm_model_1 = perform_regression(crime, "ALL", c("month", "area", "year"))
RMSE <- mlm_model_1$RMSE
normalized_RMSE <- mlm_model_1$normalized_RMSE
R2 <- mlm_model_1$R2
model_summary <- mlm_model_1$model_summary
comparison_plot <- mlm_model_1$comparison_plot
residuals_plot <- mlm_model_1$residuals_plot
resuduals_histogram <- mlm_model_1$resuduals_histogram
plot_normality <- mlm_model_1$plot_normality
```

Basic metrics and performance of models

In [210…
```r
print(paste("RMSE:", RMSE))
print(paste("Normalized RMSE Score:", normalized_RMSE))
print(paste("R2 Score:", R2))
```

```
[1] "RMSE: 639.610149615189"
[1] "Normalized RMSE Score: 0.0132809416448337"
[1] "R2 Score: 0.983389490723482"
```

Model Summary

In [211…
```r
model_summary
```

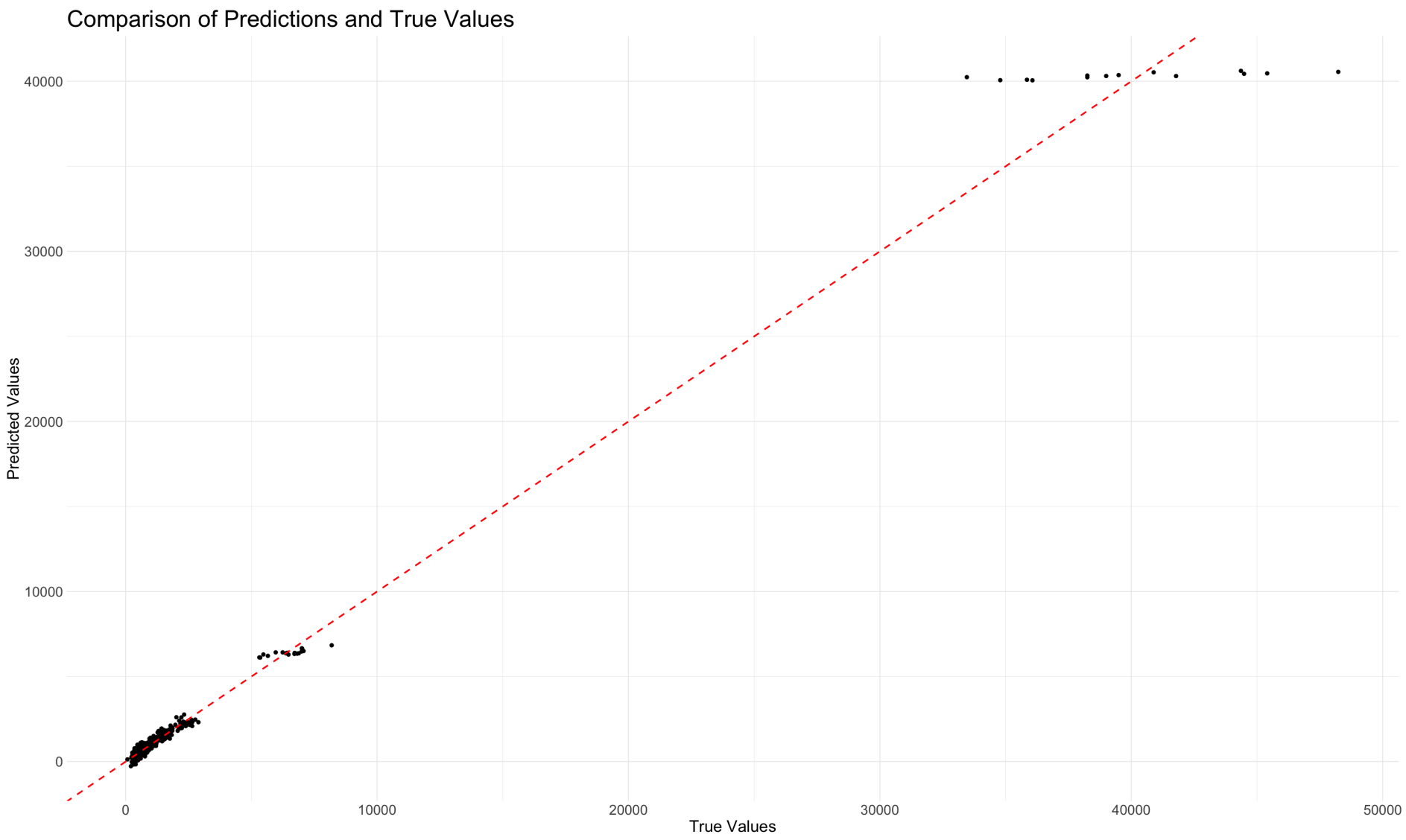
```
Call:
lm(formula = formula, data = train_data)

Residuals:
     Min       1Q   Median       3Q      Max 
-10431.2   -139.6      2.6    147.4  11905.7 

Coefficients:
                          Estimate Std. Error t value Pr(>|t|)    
(Intercept)               1352.5552   179.6082   7.531 8.85e-14 ***
monthaug                   -52.7966   110.5320  -0.478 0.632965    
monthdec                  -307.8131   108.4838  -2.837 0.004612 ** 
monthfeb                   -28.0766   116.6359  -0.241 0.809806    
monthjan                   197.4879   110.8118   1.782 0.074928 .  
monthjul                    49.0334   108.9410   0.450 0.652712    
monthjun                    44.0132   120.8046   0.364 0.715662    
monthmar                   199.6953   116.5042   1.714 0.086732 .  
monthmay                   -54.6992   123.3777  -0.443 0.657580    
monthnov                   -54.1310   119.2171  -0.454 0.649858    
monthoct                    48.0132   110.6168   0.434 0.664317    
monthsep                    23.1510   109.8991   0.211 0.833185    
areaBedfordshire          -750.4147   204.4703  -3.670 0.000251 ***
areaCambridgeshire        -662.3999   204.4658  -3.240 0.001224 ** 
areaCheshire              -200.3300   209.7684  -0.955 0.339734    
areaCleveland             -366.6196   218.0794  -1.681 0.092954 .  
areaCumbria               -651.5537   202.2881  -3.221 0.001306 ** 
areaDerbyshire            -432.6128   203.4216  -2.127 0.033615 *  
areaDevon and Cornwall    -324.5207   209.8153  -1.547 0.122155    
areaDorset                -666.5933   208.4960  -3.197 0.001418 ** 
areaDurham                -626.1598   211.4118  -2.962 0.003108 ** 
areaDyfed Powys           -722.6494   206.9773  -3.491 0.000495 ***
areaEssex                 -128.5206   209.7428  -0.613 0.540136    
areaGloucestershire       -777.1676   209.7471  -3.705 0.000219 ***
areaGreaterManchester      925.8592   216.1597   4.283 1.96e-05 ***
areaGwent                 -675.0932   203.4081  -3.319 0.000926 ***
areaHampshire                0.6593   211.4993   0.003 0.997513    
areaHertfordshire         -416.5635   211.3264  -1.971 0.048893 *  
areaHumberside            -349.5412   207.1517  -1.687 0.091748 .  
areaKent                   -49.6185   212.9603  -0.233 0.815799    
areaLancashire             148.4718   209.9010   0.707 0.479468    
areaLeicestershire        -485.9482   202.2424  -2.403 0.016395 *  
areaLincolnshire          -561.5060   204.6123  -2.744 0.006140 ** 
areaMerseyside             161.4801   204.5681   0.789 0.430025    
areaMetropolitan and City 5286.7238   211.2784  25.023  < 2e-16 ***
areaNational              39227.2365   205.7595 190.646  < 2e-16 ***
areaNorfolk                -450.9583   207.0662  -2.178 0.029579 *  
areaNorth Wales            -467.5957   208.6673  -2.241 0.025186 *  
areaNorth Yorkshire        -553.0612   200.2067  -2.762 0.005810 ** 
areaNorthamptonshire       -672.5033   211.1290  -3.185 0.001477 ** 
areaNorthumbria             261.7907   207.0039   1.265 0.206195    
areaNottinghamshire        -150.5993   208.3021  -0.723 0.469806    
areaSouth Wales             442.7765   203.4538   2.176 0.029694 *  
areaSouth Yorkshire        -101.1162   204.5448  -0.494 0.621136    
areaStaffordshire          -325.6153   205.9087  -1.581 0.114015    
areaSuffolk                -652.5408   204.5701  -3.190 0.001454 ** 
areaSurrey                 -601.7723   209.7642  -2.869 0.004180 ** 
areaSussex                 -195.2862   203.6108  -0.959 0.337661    
areaThames Valley           261.2362   204.6262   1.277 0.201931    
areaWarwickshire           -776.3362   202.3861  -3.836 0.000130 ***
areaWest Mercia            -325.1035   198.3922  -1.639 0.101495    
areaWest Midlands          1210.8444   208.4445   5.809 7.72e-09 ***
areaWest Yorkshire          563.3145   208.3948   2.703 0.006950 ** 
areaWiltshire              -731.1687   203.4867  -3.593 0.000338 ***
year2015                   -164.7273    61.8060  -2.665 0.007779 ** 
year2016                   -293.0123    63.6223  -4.605 4.48e-06 ***
year2017                   -473.7693    66.5582  -7.118 1.72e-12 ***
year2018                   -539.2391    71.0076  -7.594 5.53e-14 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 807.4 on 1447 degrees of freedom
Multiple R-squared:  0.9834,    Adjusted R-squared:  0.9827 
F-statistic:  1503 on 57 and 1447 DF,  p-value: < 2.2e-16
```

Analysis:

As can be seen, our Multivariate Linear Regression model performed perfectly on our convictions dataset. The accuracy of the model is above %98, which is extremely accurate. Moreover, the normalised RMSE of the model is at around 0.013, which is too close to zero. It illustrates our model outperforms in predicting the crime rates given the area, month, and year.

Plotting the Comparson of Predicted vs True values + regression line

In [212…
```
comparison_plot
```



Analysis:

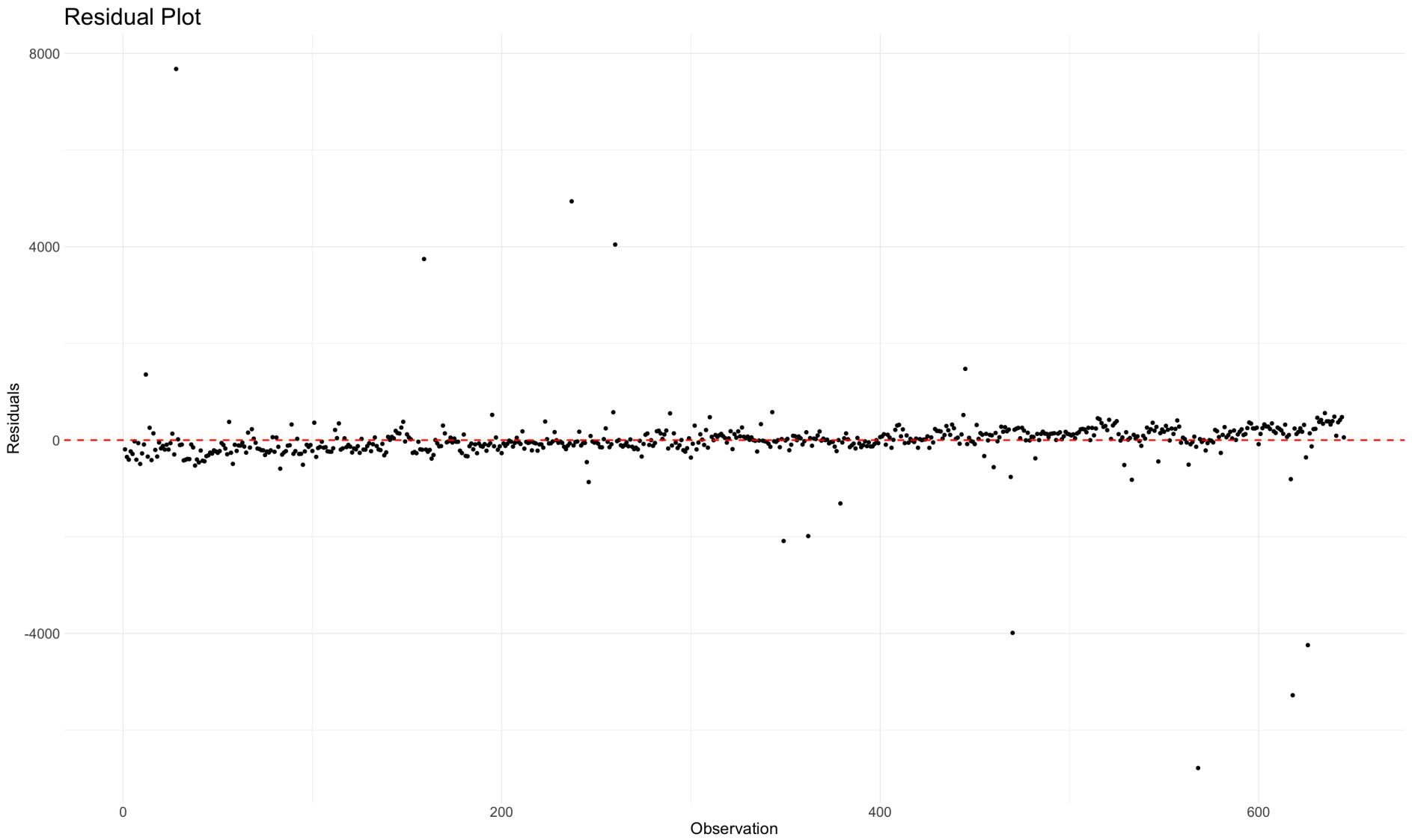
This graph illustrates the position of the true values compared to the predicted values. The points in the chart represent the True values, while the dashed red line represents the predicted ones. As we can see, the original position of the data points is almost random without any bias, which is good. Also, the population of the line above and below is equally distributed, indicating our model is not biased.

As can be seen, we have three different clusters in our dataset. The first and the most dense one belongs to the areas with small, moderate, and above-average crime rates. Because most of our observations belong to these cities, our model predicts them highly accurately. On the other hand, there is another cluster with very low density next to this former cluster. Its data points represent the crime rate in Metropolitan and City. Although the MSE for these data points is less than the previous clusters, our model can predict them accurately.

The final cluster, which is located in the top right corner of the graph, belongs to the observations with the National label for the area. At first glance, we might think they are outliers of the dataset. We can say they are not outliers regarding their symmetrical distribution above and below the diagonal line. Due to the huge difference in the number of crimes and a few numbers of observations in this cluster, our model can not predict them accurately.

Residuals Scatter Plot

In [213…
```
residuals_plot
```

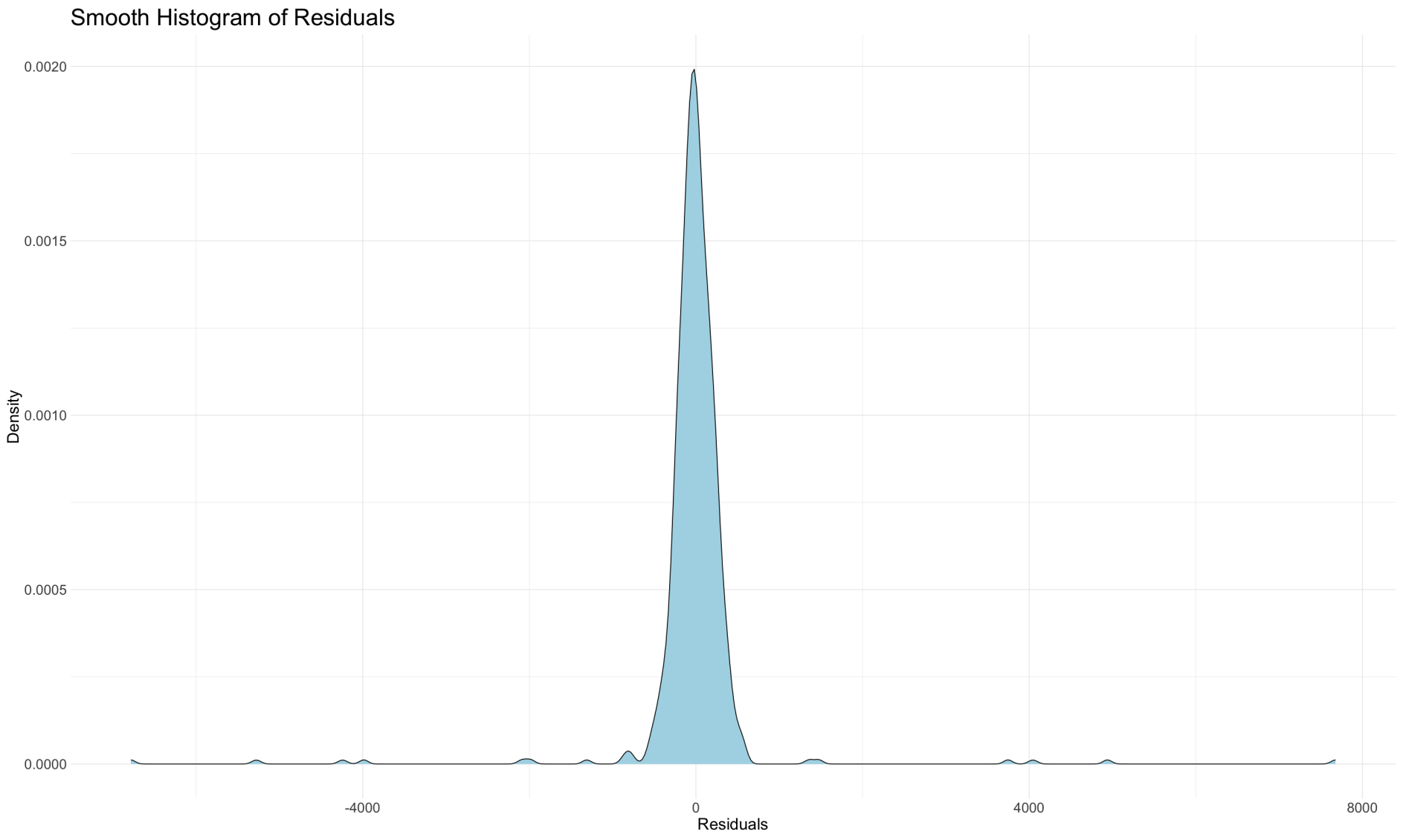


Analysis:

Looking into the residual plot of this analysis, we can say that the resudals are randomly distributed around the baseline (y=0). In fact, if we see any kind of patter or bias in this visualization, we can say that this model is not appropriate for our dataset. However, considering the chart, the residuals are perfectly spreaded around the baseline which means our model is totally appropriate for our study case.

Plotting the Residuals Histogram

In [214…
```
resuduals_histogram
```

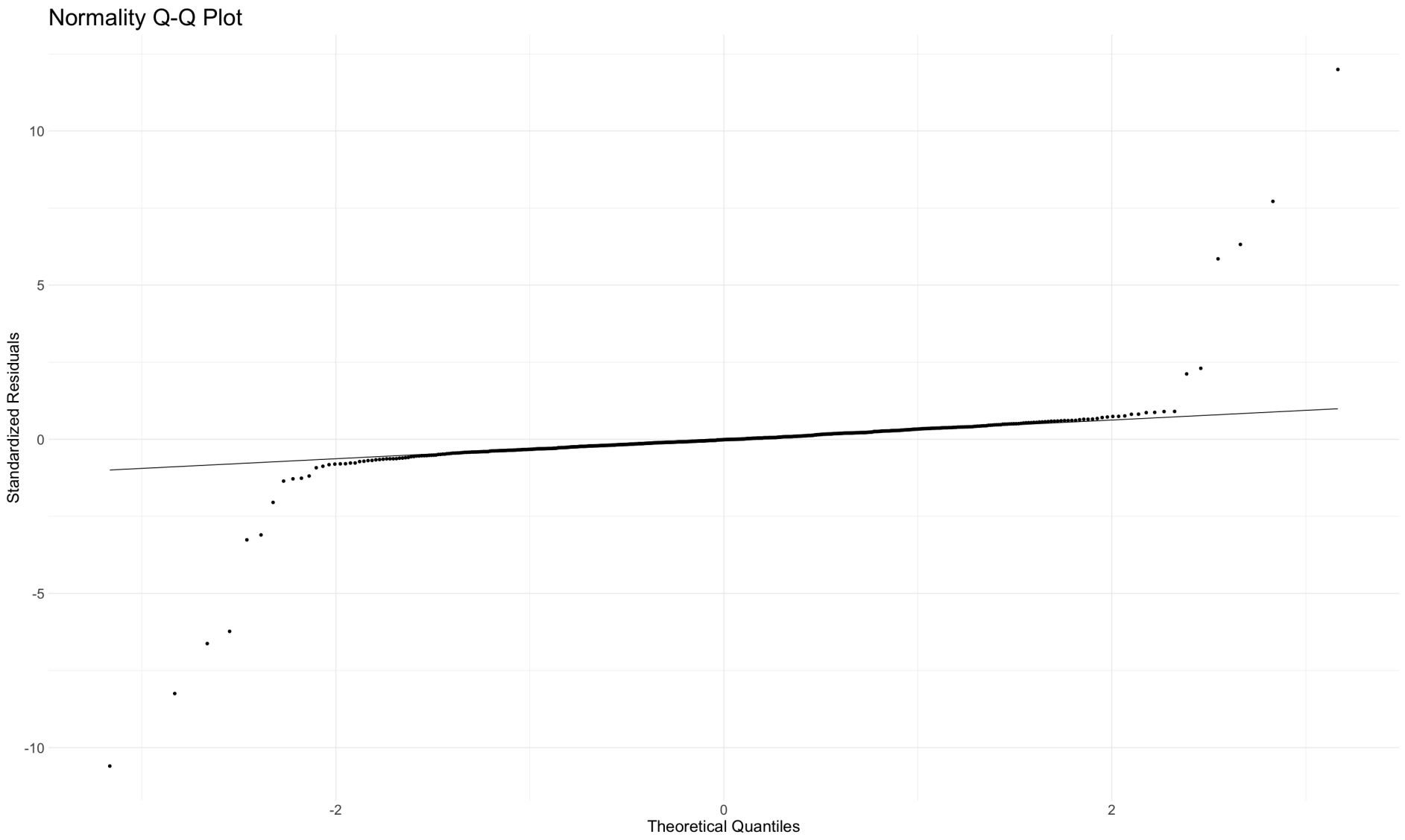


Analysis:

One assumption in linear regression is that the residuals should follow a normal distribution. If the residuals do not exhibit a bell-shaped curve, the model may be biased and linear regression might not be suitable for our dataset. As we can see, the histogram of the residuals has a perfect belly shape with almost zero skewness. The belly is too high because our residual population's standard deviation is almost near zero. Considering this distribution, we can confidently say our model is not biased, and Multivariate Regression is perfectly appropriate for this dataset.

Plotting the Normality Q-Q Plot

In [215…
```
plot_normality
```



Analysis:

In a Normality Q-Q plot, the relationship between the observed data and the theoretical quantiles of a normal distribution is examined. When the data follows a normal distribution, the plot should display the points aligning closely along a straight line. Deviations from this line suggest departures from normality, which can impact the validity of the analysis.

Upon reviewing our Normality Q-Q plot, we observe that the data points are precisely fitted on a straight line. This finding strongly supports the notion that our model is notably suitable for the dataset under investigation. The alignment of the points along the line indicates that the residuals conform to a normal distribution, enhancing the reliability and appropriateness of our analysis.

Model 2: Unsuccessful

Multiariate Linear Regression for Unsuccessful:

Predictiong the total number of Unsuccessful given month, area, and year.

Deploying the model and save the result

In [216…
```
mlm_model_2 = perform_regression(ucrime, "ALL", c("month", "area", "year"))
RMSE <- mlm_model_2$RMSE
normalized_RMSE <- mlm_model_2$normalized_RMSE
R2 <- mlm_model_2$R2
model_summary <- mlm_model_2$model_summary
comparison_plot <- mlm_model_2$comparison_plot
residuals_plot <- mlm_model_2$residuals_plot
resuduals_histogram <- mlm_model_2$resuduals_histogram
plot_normality <- mlm_model_2$plot_normality
```

Basic metrics and performance of models

In [217…
```
print(paste("RMSE:", RMSE))
print(paste("Normalized RMSE Score:", normalized_RMSE))
print(paste("R2 Score:", R2))
```

```
[1] "RMSE: 131.359586270124"
[1] "Normalized RMSE Score: 0.0178526211293998"
[1] "R2 Score: 0.986006724767376"
```

Model Summary

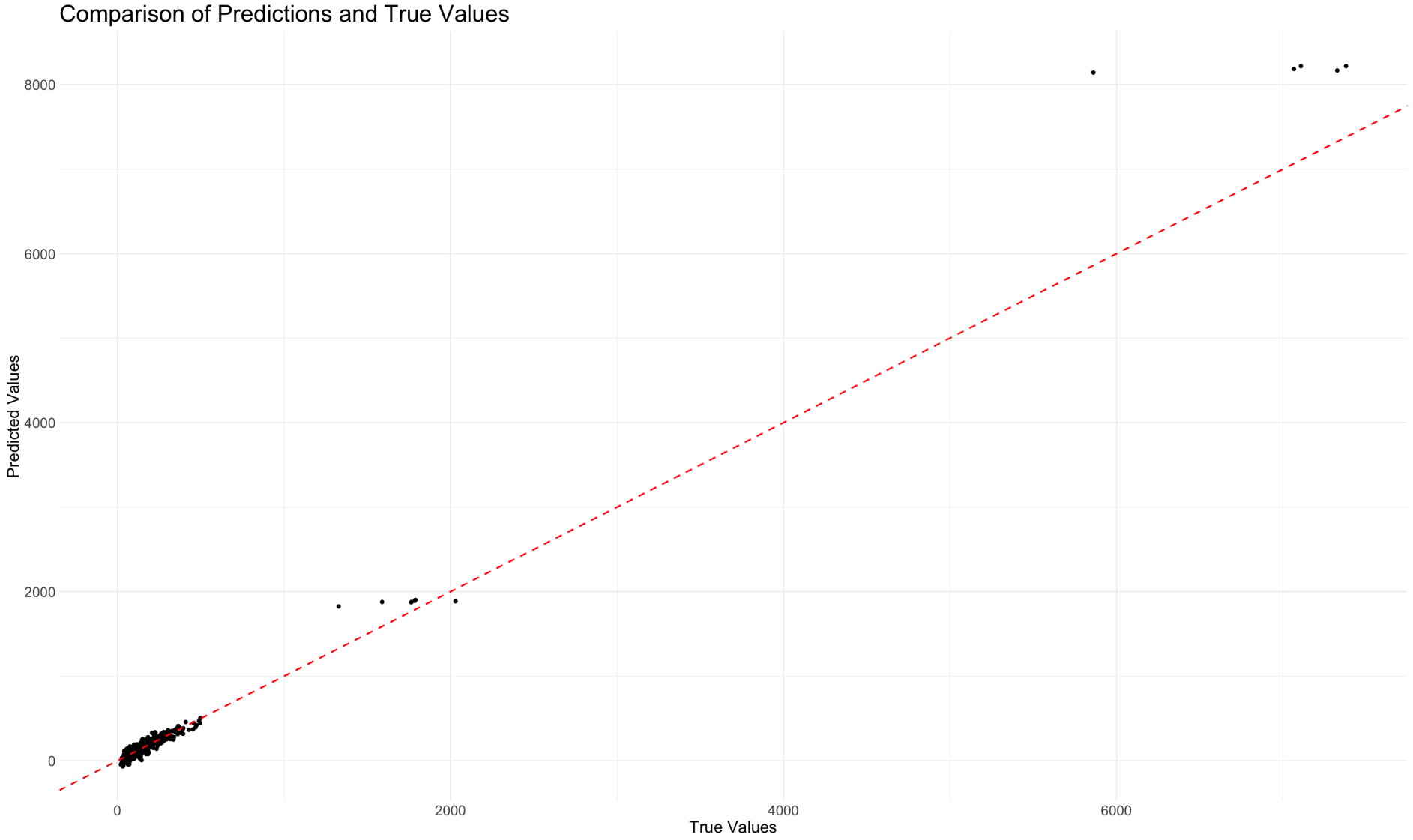
In [218…
```
model_summary
```

```
Call:
lm(formula = formula, data = train_data)

Residuals:
     Min       1Q   Median       3Q      Max
-2257.42   -32.35    -2.92    31.23  1634.23

Coefficients:
                           Estimate Std. Error t value Pr(>|t|)
(Intercept)                 231.806     34.405   6.738 2.30e-11 ***
monthaug                     18.113     22.665   0.799 0.424328
monthdec                    -29.918     22.915  -1.306 0.191901
monthfeb                      9.602     23.950   0.401 0.688523
monthjan                     48.038     22.605   2.125 0.033739 *
monthjul                     46.395     22.523   2.060 0.039579 *
monthjun                     41.211     24.757   1.665 0.096208 .
monthmar                     31.212     24.073   1.297 0.194977
monthmay                     -5.757     25.967  -0.222 0.824572
monthnov                     16.282     24.911   0.654 0.513475
monthoct                     35.522     22.481   1.580 0.114300
monthsep                     46.806     22.831   2.050 0.040530 *
areaBedfordshire           -130.800     40.427  -3.235 0.001241 **
areaCambridgeshire         -139.796     42.465  -3.292 0.001018 **
areaCheshire                -88.470     40.726  -2.172 0.029991 *
areaCleveland              -102.585     41.362  -2.480 0.013243 *
areaCumbria                -166.601     38.912  -4.281 1.98e-05 ***
areaDerbyshire              -86.231     40.121  -2.149 0.031775 *
areaDevon and Cornwall      -86.032     39.620  -2.171 0.030058 *
areaDorset                 -140.590     40.167  -3.500 0.000479 ***
areaDurham                 -140.943     40.707  -3.462 0.000551 ***
areaDyfed Powys            -160.212     40.435  -3.962 7.78e-05 ***
areaEssex                    -1.298     40.418  -0.032 0.974380
areaGloucestershire        -163.811     40.714  -4.023 6.02e-05 ***
areaGreaterManchester       140.454     40.734   3.448 0.000580 ***
areaGwent                  -147.020     41.683  -3.527 0.000433 ***
areaHampshire               -27.587     38.913  -0.709 0.478470
areaHertfordshire           -76.649     40.180  -1.908 0.056634 .
areaHumberside             -103.795     39.157  -2.651 0.008117 **
areaKent                      3.463     40.739   0.085 0.932276
areaLancashire                4.717     40.414   0.117 0.907098
areaLeicestershire          -90.002     42.109  -2.137 0.032731 *
areaLincolnshire           -115.123     40.144  -2.868 0.004193 **
areaMerseyside              -16.690     40.749  -0.410 0.682167
areaMetropolitan and City  1705.973     38.494  44.317  < 2e-16 ***
areaNational               8023.122     38.104 210.558  < 2e-16 ***
areaNorfolk                -119.613     40.734  -2.936 0.003371 **
areaNorth Wales            -128.914     39.883  -3.232 0.001255 **
areaNorth Yorkshire        -136.797     40.139  -3.408 0.000672 ***
areaNorthamptonshire       -124.305     41.067  -3.027 0.002513 **
areaNorthumbria             103.604     39.625   2.615 0.009024 **
areaNottinghamshire         -16.586     40.704  -0.407 0.683721
areaSouth Wales              59.179     40.147   1.474 0.140674
areaSouth Yorkshire         -50.920     40.118  -1.269 0.204543
areaStaffordshire           -57.967     40.748  -1.423 0.155076
areaSuffolk                -155.479     41.379  -3.757 0.000178 ***
areaSurrey                 -106.223     39.880  -2.664 0.007815 **
areaSussex                  -41.536     39.623  -1.048 0.294681
areaThames Valley            82.875     40.179   2.063 0.039323 *
areaWarwickshire           -147.817     40.748  -3.628 0.000296 ***
areaWest Mercia             -76.179     39.890  -1.910 0.056359 .
areaWest Midlands           273.242     38.105   7.171 1.17e-12 ***
areaWest Yorkshire          117.485     41.369   2.840 0.004575 **
areaWiltshire              -162.658     40.735  -3.993 6.84e-05 ***
year2015                     3.448     12.775   0.270 0.787301
year2016                   -51.689     13.181  -3.921 9.20e-05 ***
year2017                   -82.597     13.875  -5.953 3.28e-09 ***
year2018                  -106.586     14.393  -7.405 2.19e-13 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 168.9 on 1478 degrees of freedom
Multiple R-squared:  0.986,     Adjusted R-squared:  0.9855
F-statistic:  1827 on 57 and 1478 DF,  p-value: < 2.2e-16
```

Plotting the Comparson of Predicted vs True values + regression line

In [219…

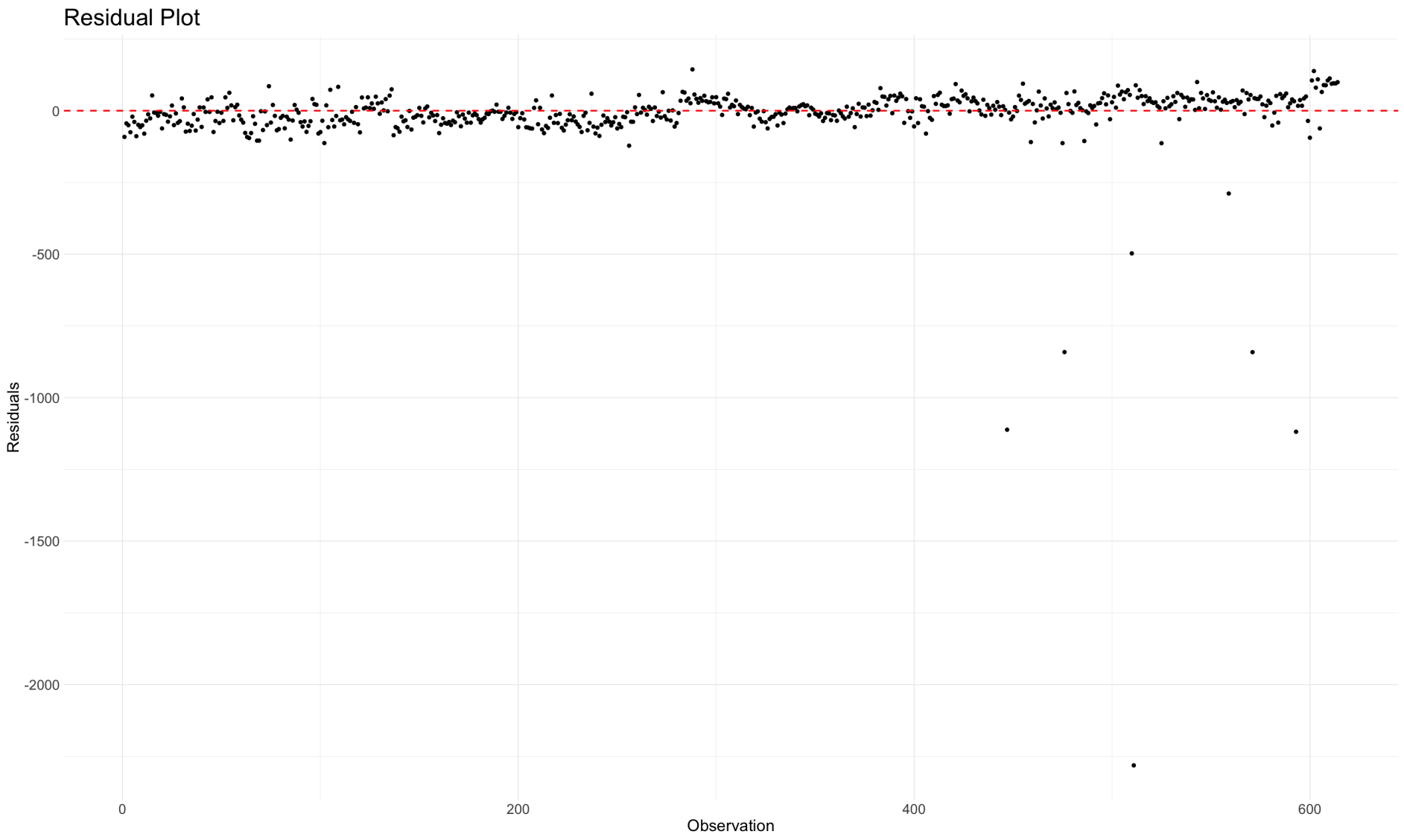
```
comparison_plot
```



Residuals Scatter Plot

In [220…

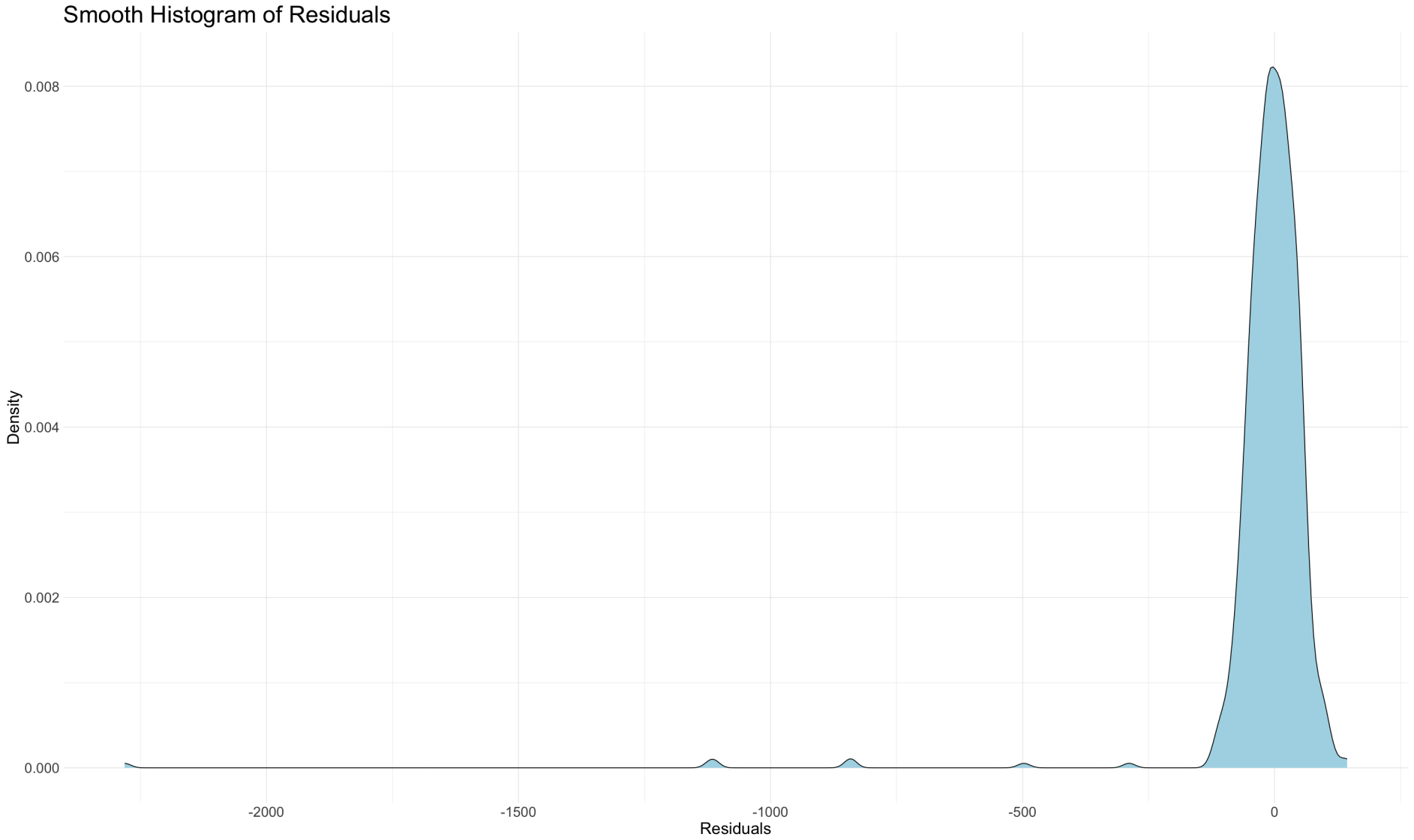
```
residuals_plot
```



Plotting the Residuals Histogram

In [221…

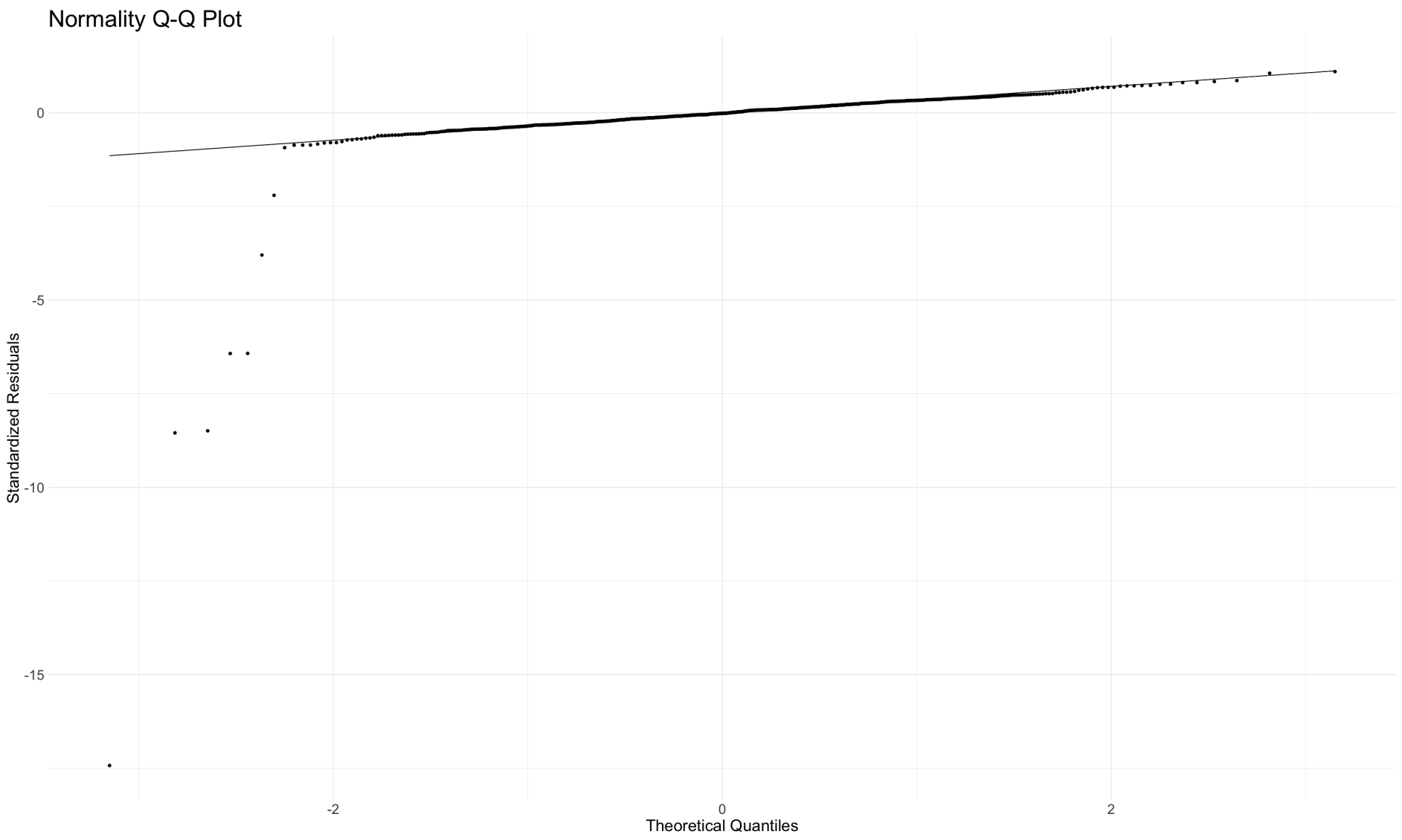
```
resuduals_histogram
```



Plotting the Normality Q-Q Plot

In [222…

```
plot_normality
```



Clustering

K-means:

A well-liked unsupervised machine learning approach called K-means clustering divides a dataset into K different clusters based on similarity. In order to reduce the within-cluster sum of squares, data points are assigned to the cluster with the closest mean (centroid). The approach optimises the clustering objective by updating the cluster centroids and reassigning data points until convergence. K-means implies clusters of similar size and density and are sensitive to the original centroid placement.

pipeline:

As the performance of K-means clustering depends on the initialised K, we should first find the optimum K (K_opt) and then try to implement our models with k = K_opt, K_opt+1, K_opt+2 and compare them visually. To do so, there is two common way:

1. Plotting Silhouette score: Silhouette score measures how well each data point fits into its assigned cluster compared to other clusters. A higher silhouette score (ranging from -1 to 1) indicates better-defined clusters. Therefore, plotting these scores (Y-axis) with respect to different Ks (X-axis) can be a good measure to pick the optimum K. The best K must have the highest Y in the plot.

1. The elbow method: is a technique used to determine the optimal number of clusters in k-means clustering. It involves plotting the within-cluster sum of squares (WCSS) against the number of clusters (k). The idea is to identify the "elbow" point on the plot, which represents a significant drop in the WCSS. This point indicates the number of clusters that provides the most substantial gain in cluster cohesion. Interpreting the plot involves observing where the WCSS starts to decrease at a slower rate, as larger values of k may not significantly improve cluster quality. The elbow point indicates a good balance between reducing WCSS and minimising the number of clusters.

Although the Silhouette score method is more accurate, in this study, we prefer the elbow method as this study aims to show the r-programming skills in addition to knowing different methods.

Visualisation Methods for Clusters:

We use two major visualisation methods for visualising our clusters. Our alternatives are Principal Component Analysis (PCA) and t-Distributed Stochastic Neighbor Embedding (t-SNE). Both of these techniques aim to reduce the dimensionality of multidimensional spaces. PCA looks for orthogonal principle components that explain the most variance to preserve the data's overall structure. However, t-SNE places emphasis on maintaining local links by mapping similar data points in high-dimensional space to nearby points in low-dimensional space.

Visualisation using PCA:

It is a method for visualising high-dimensional data in a lower-dimensional space by combining PCA (Principal Component Analysis) and clustering methods. It entails utilising PCA to decrease the data's dimensionality while keeping key features, labelling data points with clusters using clustering techniques, and showing the data points in a scatter plot using the PCA's reduced coordinates. The visualisation aids in illuminating the underlying cluster structure and patterns in the data by colouring or employing various symbols for each cluster.

Visualisation using t-SNE:

In this method, we use t-SNE to reduce high-dimensional data into a lower-dimensional space while preserving local relationships. The data points are then plotted in a scatter plot using their t-SNE coordinates, with different colours or symbols representing the assigned cluster labels. This visualisation helps uncover the clustering structure and patterns in the data.

Please Note:

In this section, we just deploy clusterings for the convictions dataset.

In [223…
```r
# Create a function for evaluating the WCSS
kmean_withinss <- function(k) {
    cluster <- kmeans(crime[,-c(1,2,3,4,5,18)], k)
    return (cluster$tot.withinss)
}
```

In [224…
```r
# Create a function to perform k-means clustering

perform_kmeans_clustering <- function(data, k) {
  numeric_data <- data[,-c(1,2,3,4,5,18)]
  kmeans_result <- kmeans(numeric_data, centers = k)
  return(kmeans_result)
}
```

In [225…
```r
# Create a function to visualize clusters using PCA

plot_pca_clusters <- function(data, k) {
  pca_result <- prcomp(data, scale. = TRUE)
  reduced_data <- as.data.frame(pca_result$x[, 1:2])
  kmeans_result <- kmeans(reduced_data, centers = k)
  cluster_labels <- kmeans_result$cluster
  data_with_clusters <- cbind(reduced_data, Cluster = as.factor(cluster_labels))
  ggplot(data_with_clusters, aes(x = PC1, y = PC2, color = Cluster)) +
    geom_point(size=2) +
    labs(title = "K-means Clustering (PCA)", x = "Principal Component 1", y = "Principal Component 2") +
    theme_minimal()+
    theme(
        axis.text = element_text(size = 21),
        axis.title = element_text(size = 21),
        plot.title = element_text(size = 30),
        axis.text.x = element_text(size = 18),
        axis.text.y = element_text(size = 18),
        legend.text = element_text(size = 17)
    )

}
```

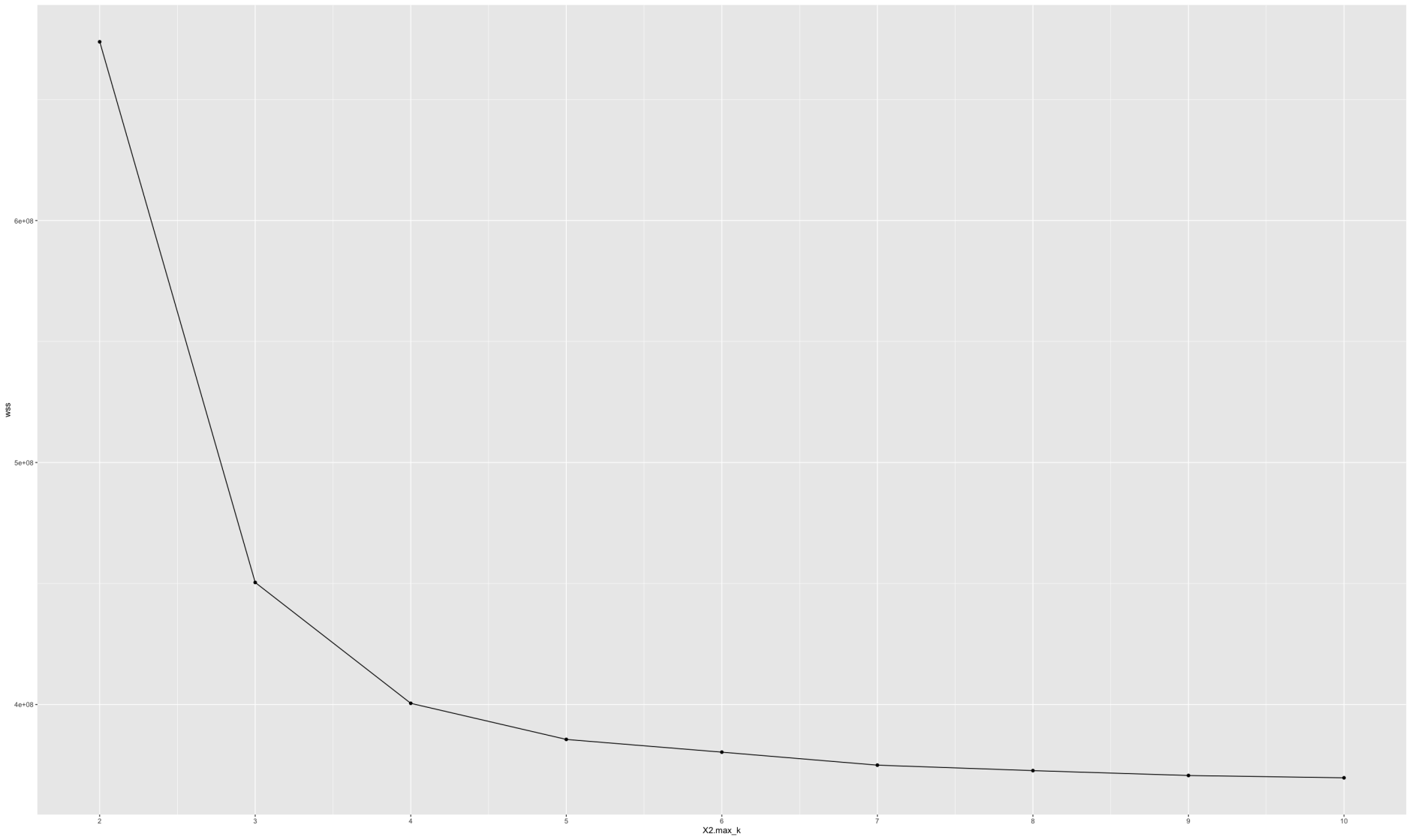
In [226…
```r
# Create a function to visualize clusters using t-SNE

plot_tsne_clusters <- function(data, k) {
  data <- data[!duplicated(data), ]
  kmeans_result <- kmeans(data, centers = k)
  cluster_labels <- kmeans_result$cluster
  data_with_clusters <- cbind(data, Cluster = as.factor(cluster_labels))
  tsne_result <- Rtsne(data, dims = 2, perplexity = 30, verbose = TRUE)
  tsne_data <- as.data.frame(tsne_result$Y)
  tsne_data$Cluster <- as.factor(cluster_labels)
  ggplot(tsne_data, aes(x = V1, y = V2, color = Cluster)) +
    geom_point(size=1.8) +
    labs(title = "K-means Clustering (t-SNE)", x = "Dimension 1", y = "Dimension 2") +
    theme_minimal()+
    theme(
        axis.text = element_text(size = 21),
        axis.title = element_text(size = 21),
        plot.title = element_text(size = 30),
        axis.text.x = element_text(size = 18),
```

```
        axis.text.y = element_text(size = 18),
        legend.text = element_text(size = 17)
    )

}
```

Elbow Method:

We perform elbow method to find the suitable K for our dataset

In [227…
```
# Set maximum cluster
max_k <-10
wss <- sapply(2:max_k, kmean_withinss)
elbow <-data.frame(2:max_k, wss)
ggplot(elbow, aes(x = X2.max_k, y = wss)) +
    geom_point() +
    geom_line() +
    scale_x_continuous(breaks = seq(1, 20, by = 1))
```



Analysis:

This Graph illustrates the effect of changing K on the WSS. As can be seen, we have the biggest change and the shape on the elbow at the K = 3. It means the optimum K for our clustering model is at K = 3. Therefore, our first model will be the K-means(K=3). However, to investigate more, we can also try different models with different K to detect other patterns in the graphs. To do so, we form these models:

1. model 1 : K=3
2. model 2 : K=4
3. model 3 : K=5

For each model, we will provide a summary that interprets unique attributes of our model, such as several instances in each cluster, the centres of different features belonging to each cluster, etc.

Model 1: K=3

K-means Clustering for Convictions:

Performing the model:

In [228…
```
model_k_3 <- perform_kmeans_clustering(crime, 3)
```

Summerize the model attributes:

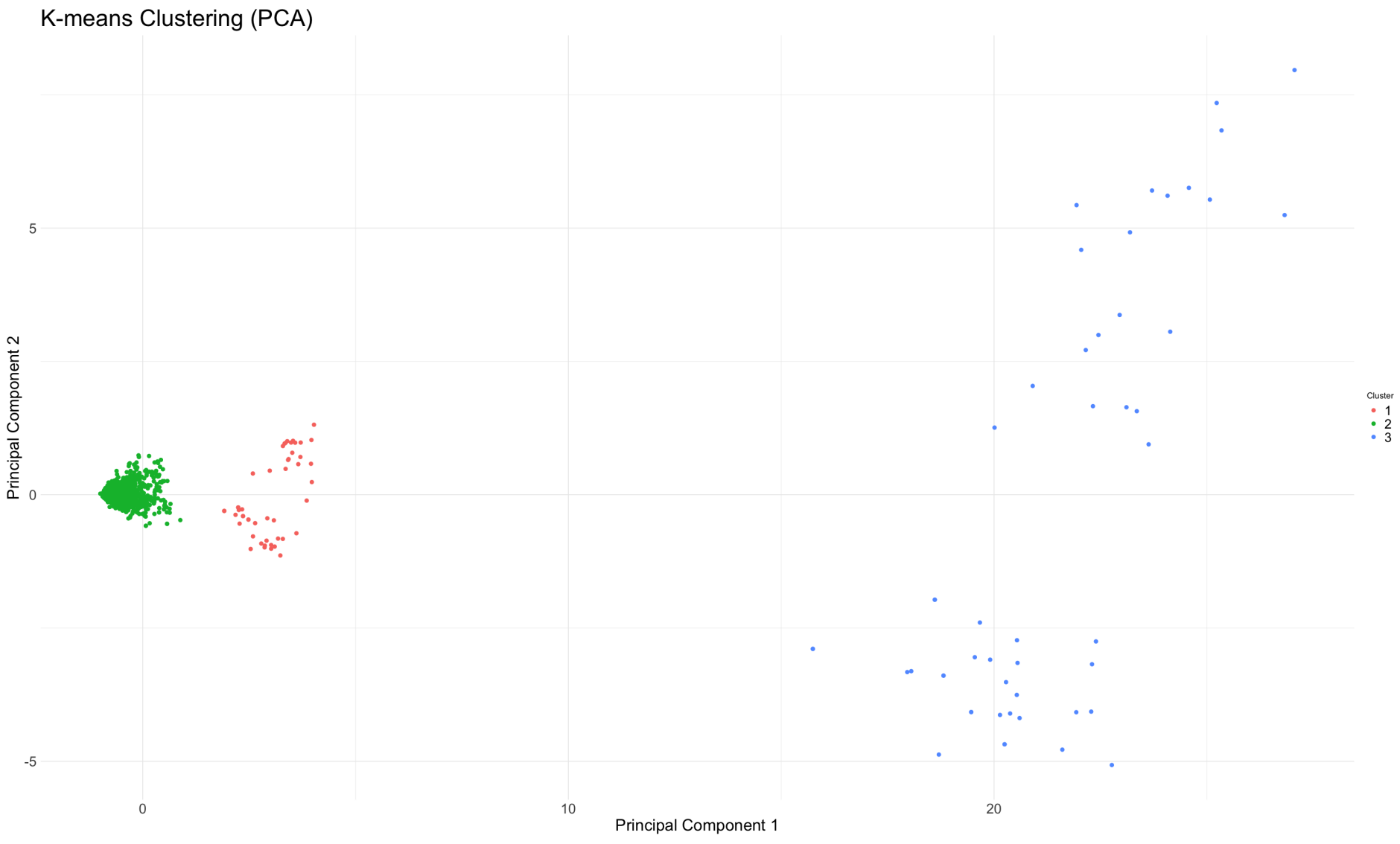
In [229…
```
print("The Number of Instances belong to Each Cluster:")
print(table(model_k_3$cluster))
cat("\n\n\n\n")
print(model_k_3[-c(1)])
```

```
[1] "The Number of Instances belong to Each Cluster:"

   1    2    3
  50   50 2050
```

```
$centers
   homicide offences_against_the_person sexual_offences   burglary    robbery
1 16.280000                   1461.9400       137.08000  203.04000 108.120000
2 81.040000                   9771.3200       940.82000 1293.90000 416.640000
3  1.579512                    202.6678        19.60341   26.60634   7.524878
  theft_and_handling fraud_and_forgery criminal_damage drugs_offences
1          1018.5400         217.62000       265.02000      1007.2800
2          8052.0800         827.92000      2066.04000      4020.2800
3           171.5498          14.88537        43.92732        73.4878
  public_order_offences all_other_offences__excluding_motoring_
1             524.40000                                295.36000
2            3497.78000                               1395.98000
3              72.52146                                 26.84439
  motoring_offences
1         1154.2000
2         7869.3000
3          163.7829

$totss
[1] 12787106772

$withinss
[1]  10661487 343151868  96615561

$tot.withinss
[1] 450428915

$betweenss
[1] 12336677857

$size
[1]   50   50 2050

$iter
[1] 3

$ifault
[1] 0
```

Visualizing clusters using PCA:

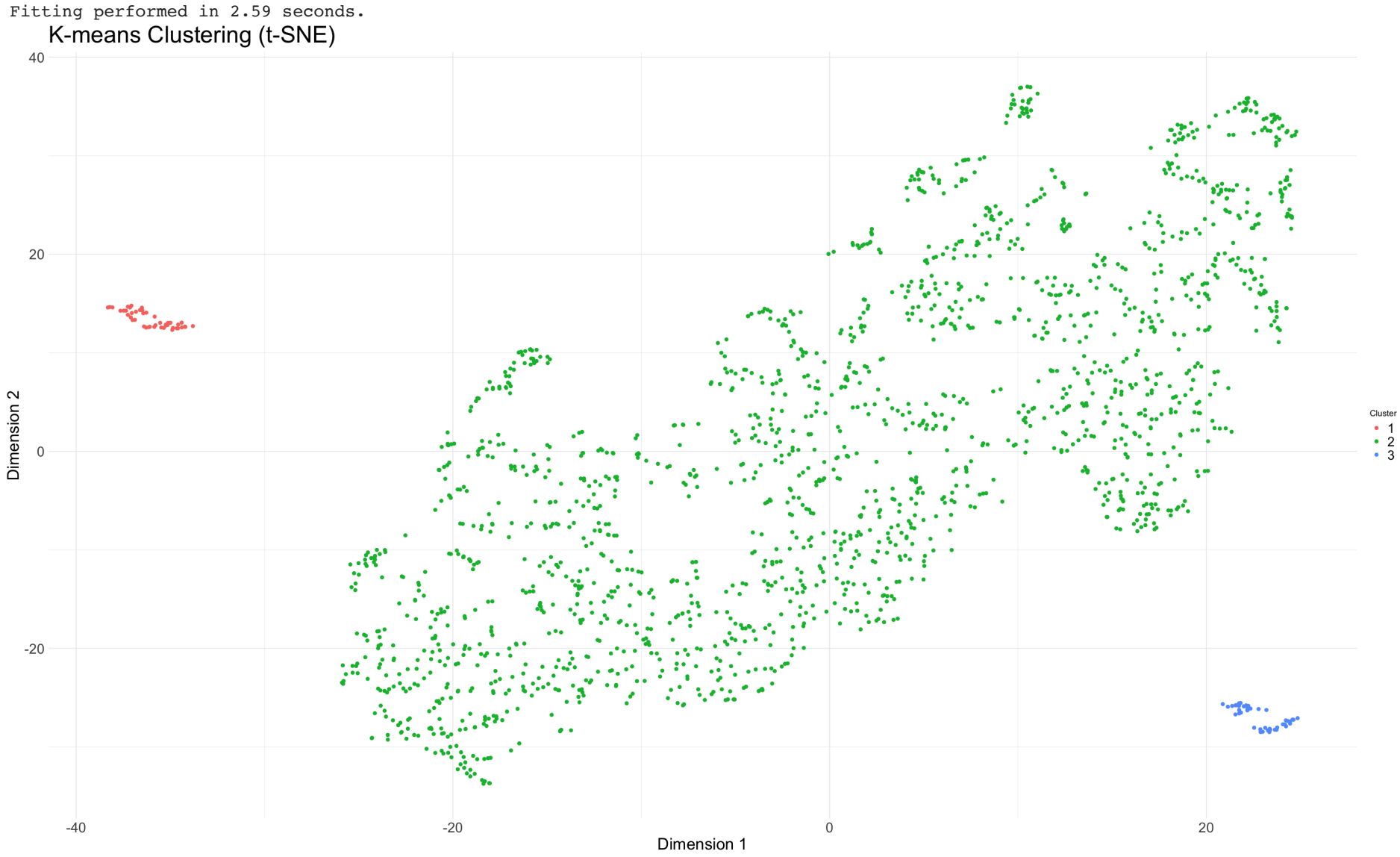
In [230…
```
plot_pca_clusters(crime[,-c(1,2,3,4,5,18)], k = 3)
```



Visualizing clusters using PCA:

In [233…
```
plot_tsne_clusters(crime[,-c(1,2,3,4,5,18)], k = 3)
```

```
Performing PCA
Read the 1935 x 12 data matrix successfully!
Using no_dims = 2, perplexity = 30.000000, and theta = 0.500000
Computing input similarities...
Building tree...
Done in 0.10 seconds (sparsity = 0.063824)!
Learning embedding...
Iteration 50: error is 76.834553 (50 iterations in 0.17 seconds)
Iteration 100: error is 66.164470 (50 iterations in 0.13 seconds)
Iteration 150: error is 65.440766 (50 iterations in 0.13 seconds)
Iteration 200: error is 65.324938 (50 iterations in 0.13 seconds)
Iteration 250: error is 65.315566 (50 iterations in 0.14 seconds)
Iteration 300: error is 1.458525 (50 iterations in 0.13 seconds)
Iteration 350: error is 1.187708 (50 iterations in 0.12 seconds)
Iteration 400: error is 1.084157 (50 iterations in 0.12 seconds)
Iteration 450: error is 1.036579 (50 iterations in 0.13 seconds)
Iteration 500: error is 1.016429 (50 iterations in 0.13 seconds)
Iteration 550: error is 1.003312 (50 iterations in 0.13 seconds)
Iteration 600: error is 0.997867 (50 iterations in 0.12 seconds)
Iteration 650: error is 0.992808 (50 iterations in 0.13 seconds)
Iteration 700: error is 0.985401 (50 iterations in 0.12 seconds)
Iteration 750: error is 0.980684 (50 iterations in 0.12 seconds)
Iteration 800: error is 0.976515 (50 iterations in 0.12 seconds)
Iteration 850: error is 0.972439 (50 iterations in 0.13 seconds)
Iteration 900: error is 0.970188 (50 iterations in 0.12 seconds)
Iteration 950: error is 0.966878 (50 iterations in 0.12 seconds)
Iteration 1000: error is 0.963565 (50 iterations in 0.12 seconds)
Fitting performed in 2.59 seconds.
```



Model 2 : K=4

K-means Clustering for Convictions:

Performing the model:

In [234…
```r
model_k_4 <- perform_kmeans_clustering(crime, 4)
```

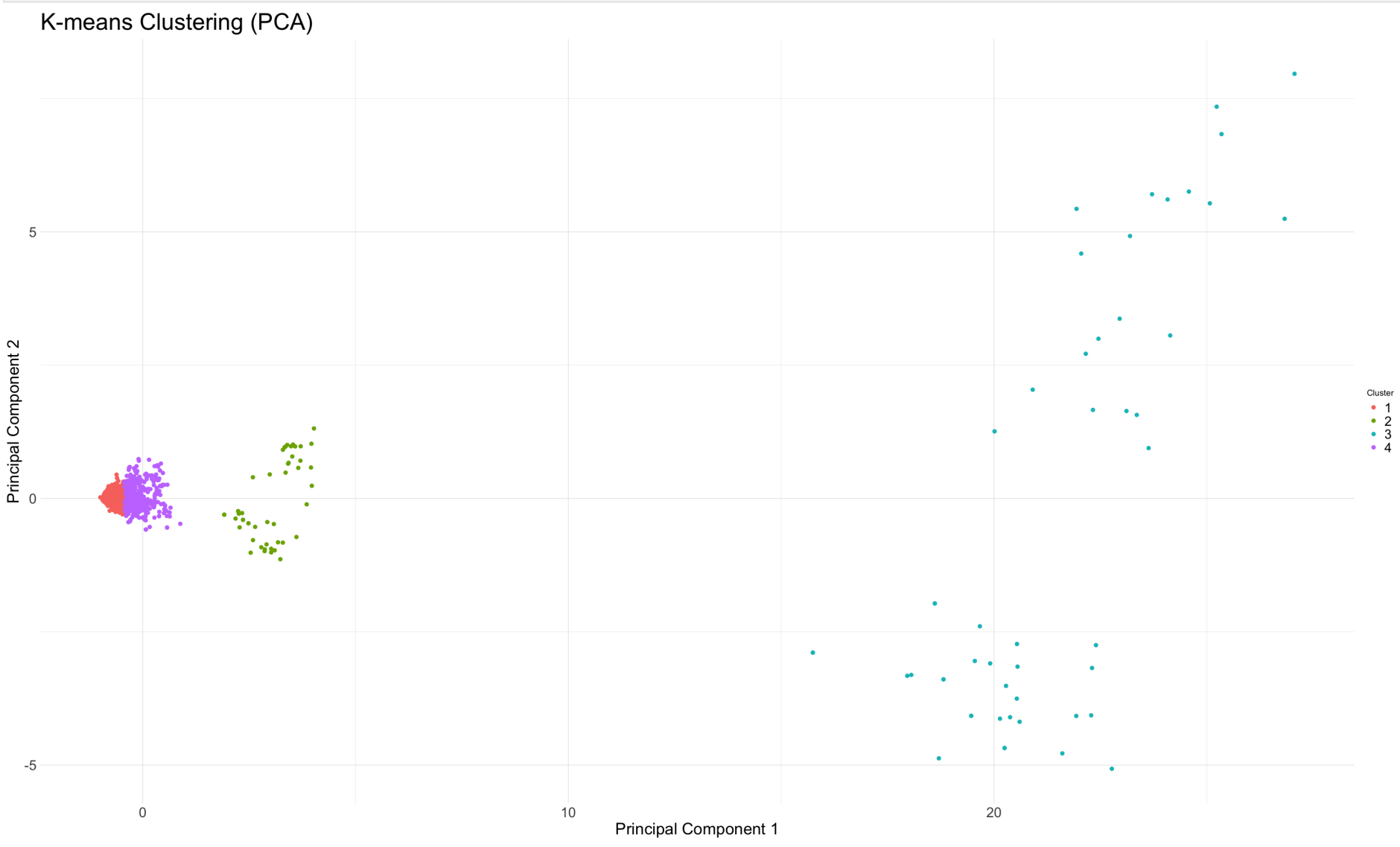
SUmmerize the model attributes:

In [235…
```r
print("The Number of Instances belong to Each Cluster:")
print(table(model_k_4$cluster))
cat("\n\n\n\n")
print(model_k_4[-c(1)])
```

```
[1] "The Number of Instances belong to Each Cluster:"

   1    2    3    4
1562   50   50  488



$centers
   homicide offences_against_the_person sexual_offences   burglary    robbery
1  1.158131                   150.6972        13.58579   19.18182   4.611396
2 16.280000                  1461.9400       137.08000  203.04000 108.120000
3 81.040000                  9771.3200       940.82000 1293.90000 416.640000
4  2.928279                   369.0164        38.86475   50.37090  16.850410
  theft_and_handling fraud_and_forgery criminal_damage drugs_offences
1           125.2292          11.27337        32.71639        52.9219
2          1018.5400         217.62000       265.02000      1007.2800
3          8052.0800         827.92000      2066.04000      4020.2800
4           319.8135          26.44672        79.81148       139.3156
  public_order_offences all_other_offences__excluding_motoring_
1              52.74008                                 17.46671
2             524.40000                                295.36000
3            3497.78000                               1395.98000
4             135.83811                                 56.86066
  motoring_offences
1           123.491
2          1154.200
3          7869.300
4           292.750

$totss
[1] 12787106772

$withinss
[1]  18092402  10661487 343151868  28582800

$tot.withinss
[1] 400488556

$betweenss
[1] 12386618216

$size
[1] 1562   50   50  488

$iter
[1] 4

$ifault
[1] 0
```

Visualizing clusters using PCA:

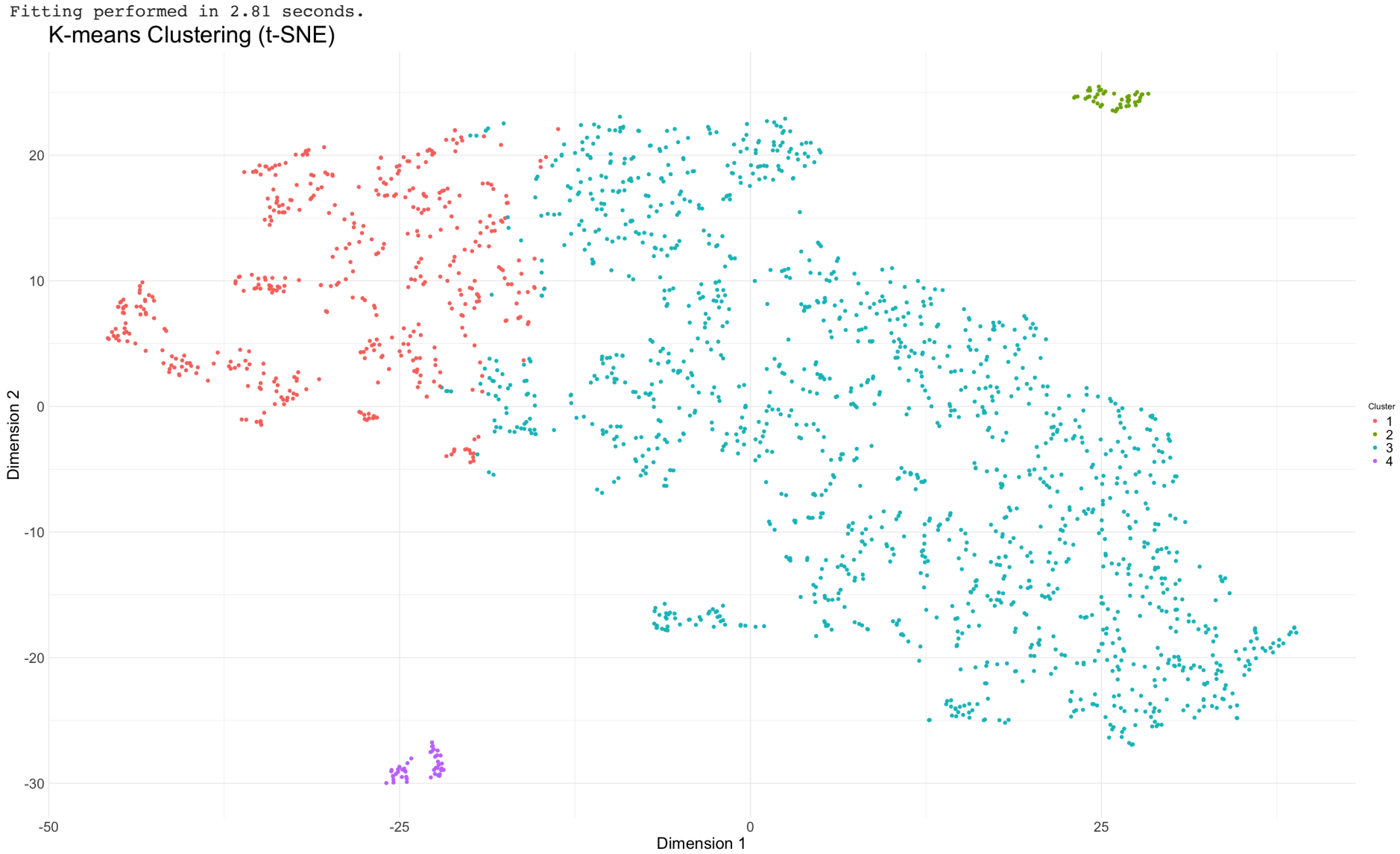
In [236…
```
plot_pca_clusters(crime[,-c(1,2,3,4,5,18)], k = 4)
```



Visualizing clusters using PCA:

In [237…
```
plot_tsne_clusters(crime[,-c(1,2,3,4,5,18)], k = 4)
```

```
Performing PCA
Read the 1935 x 12 data matrix successfully!
Using no_dims = 2, perplexity = 30.000000, and theta = 0.500000
Computing input similarities...
Building tree...
Done in 0.10 seconds (sparsity = 0.063824)!
Learning embedding...
Iteration 50: error is 75.784368 (50 iterations in 0.21 seconds)
Iteration 100: error is 65.989434 (50 iterations in 0.14 seconds)
Iteration 150: error is 65.394075 (50 iterations in 0.14 seconds)
Iteration 200: error is 65.281184 (50 iterations in 0.14 seconds)
Iteration 250: error is 65.295278 (50 iterations in 0.14 seconds)
Iteration 300: error is 1.452039 (50 iterations in 0.13 seconds)
Iteration 350: error is 1.182178 (50 iterations in 0.13 seconds)
Iteration 400: error is 1.075588 (50 iterations in 0.14 seconds)
Iteration 450: error is 1.026929 (50 iterations in 0.14 seconds)
Iteration 500: error is 1.003693 (50 iterations in 0.14 seconds)
Iteration 550: error is 0.988184 (50 iterations in 0.14 seconds)
Iteration 600: error is 0.978605 (50 iterations in 0.14 seconds)
Iteration 650: error is 0.972470 (50 iterations in 0.13 seconds)
Iteration 700: error is 0.967691 (50 iterations in 0.14 seconds)
Iteration 750: error is 0.963287 (50 iterations in 0.14 seconds)
Iteration 800: error is 0.958815 (50 iterations in 0.14 seconds)
Iteration 850: error is 0.955171 (50 iterations in 0.14 seconds)
Iteration 900: error is 0.952789 (50 iterations in 0.14 seconds)
Iteration 950: error is 0.949809 (50 iterations in 0.14 seconds)
Iteration 1000: error is 0.947423 (50 iterations in 0.14 seconds)
Fitting performed in 2.81 seconds.
```



Model 3 : K=5

K-means Clustering for Convictions:

Performing the model:

In [238…
```r
model_k_5 <- perform_kmeans_clustering(crime, 5)
```

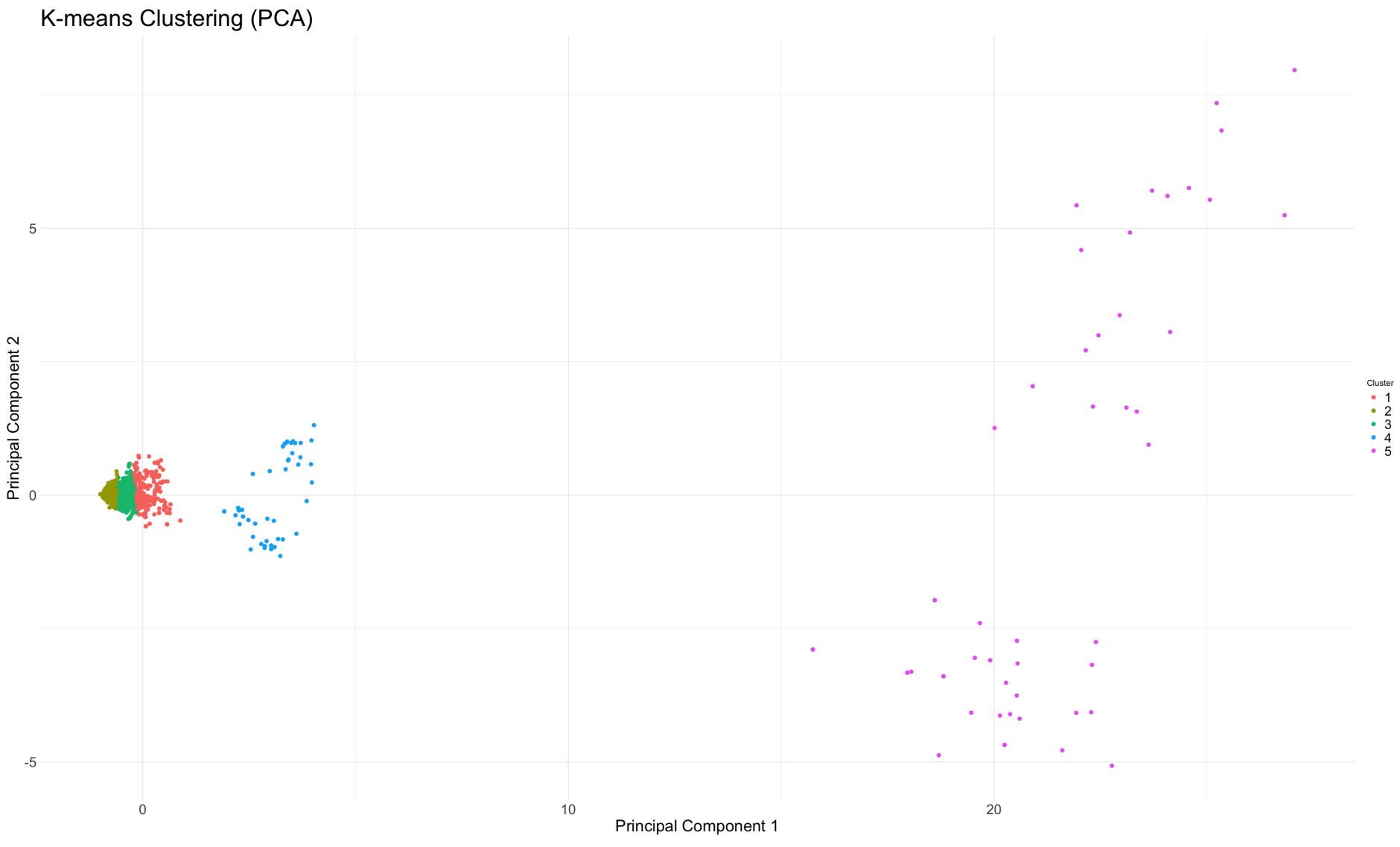
SUmmerize the model attributes:

In [239…
```r
print("The Number of Instances belong to Each Cluster:")
print(table(model_k_5$cluster))
cat("\n\n\n\n")
print(model_k_5[-c(1)])
```

```
[1] "The Number of Instances belong to Each Cluster:"

   1    2    3    4    5
1141   50  178   50  731



$centers
   homicide offences_against_the_person sexual_offences   burglary    robbery
1  0.861525                   122.4969         9.74759   15.77038   3.632778
2 81.040000                  9771.3200       940.82000 1293.90000 416.640000
3  3.685393                   480.6404        48.08989   67.07865  27.308989
4 16.280000                  1461.9400       137.08000  203.04000 108.120000
5  2.187415                   260.1176        28.05062   33.66484   8.782490
  theft_and_handling fraud_and_forgery criminal_damage drugs_offences
1           103.8370          9.038563        27.51797       42.02366
2          8052.0800        827.920000      2066.04000     4020.28000
3           417.8258         34.500000       101.86517      171.96067
4          1018.5400        217.620000       265.02000     1007.28000
5           217.2722         19.235294        55.43228       98.62107
  public_order_offences all_other_offences__excluding_motoring_
1              44.20596                                 13.95706
2            3497.78000                               1395.98000
3             178.28652                                 74.50562
4             524.40000                                295.36000
5              90.96443                                 35.35431
  motoring_offences
1          101.8177
2         7869.3000
3          389.1461
4         1154.2000
5          205.6265

$totss
[1] 12787106772

$withinss
[1]   6741894 343151868  13705928  10661487  11309769

$tot.withinss
[1] 385570945

$betweenss
[1] 12401535827

$size
[1] 1141   50  178   50  731

$iter
[1] 3

$ifault
[1] 0
```

Visualizing clusters using PCA:

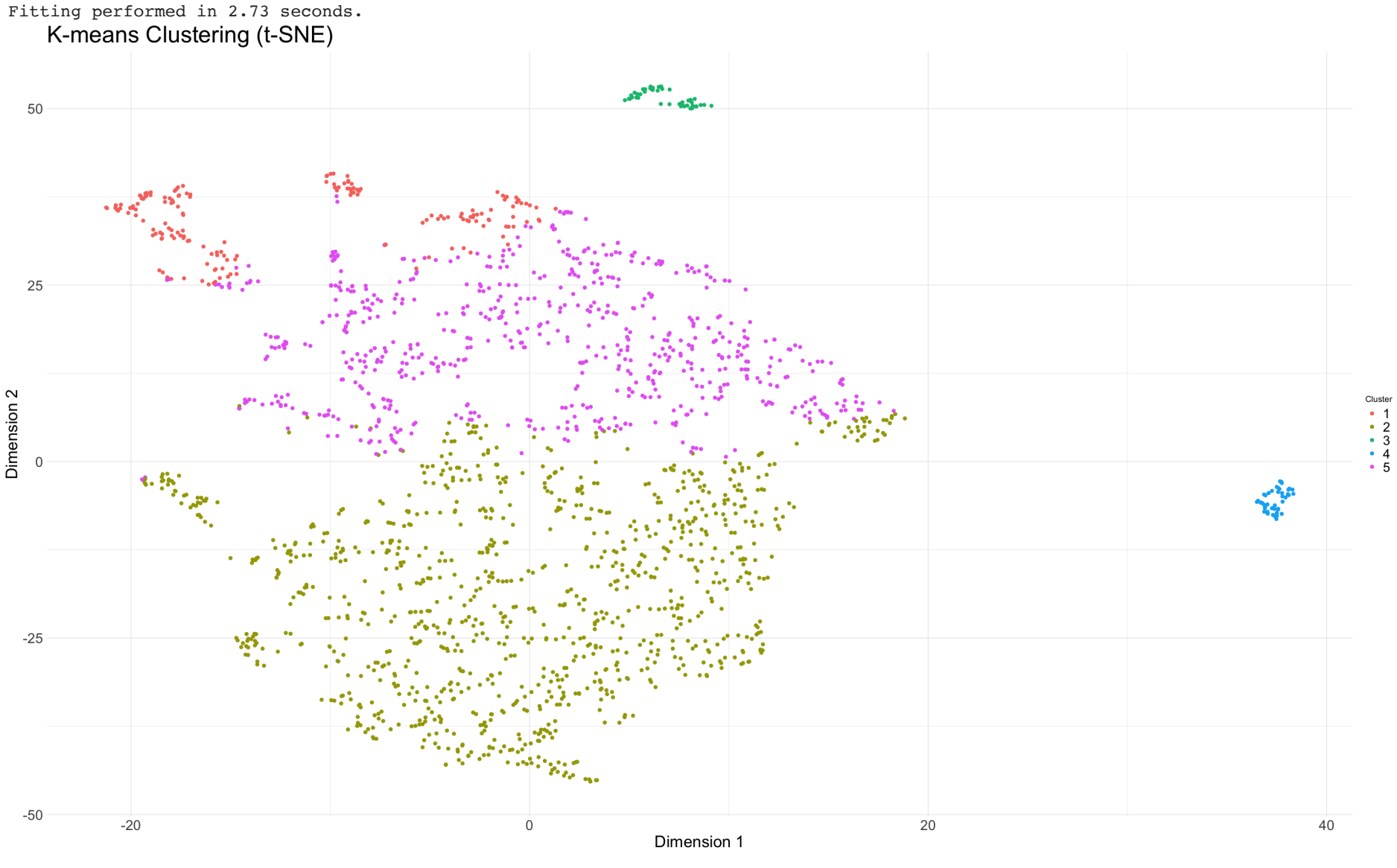
In [240…
```
plot_pca_clusters(crime[,-c(1,2,3,4,5,18)], k = 5)
```



Visualizing clusters using PCA:

In [241…
```
plot_tsne_clusters(crime[,-c(1,2,3,4,5,18)], k = 5)
```

```
Performing PCA
Read the 1935 x 12 data matrix successfully!
Using no_dims = 2, perplexity = 30.000000, and theta = 0.500000
Computing input similarities...
Building tree...
Done in 0.09 seconds (sparsity = 0.063824)!
Learning embedding...
Iteration 50: error is 73.261450 (50 iterations in 0.17 seconds)
Iteration 100: error is 66.012847 (50 iterations in 0.13 seconds)
Iteration 150: error is 65.376362 (50 iterations in 0.14 seconds)
Iteration 200: error is 65.257277 (50 iterations in 0.14 seconds)
Iteration 250: error is 65.232431 (50 iterations in 0.14 seconds)
Iteration 300: error is 1.463469 (50 iterations in 0.13 seconds)
Iteration 350: error is 1.187847 (50 iterations in 0.13 seconds)
Iteration 400: error is 1.078618 (50 iterations in 0.13 seconds)
Iteration 450: error is 1.025390 (50 iterations in 0.14 seconds)
Iteration 500: error is 1.001979 (50 iterations in 0.14 seconds)
Iteration 550: error is 0.990408 (50 iterations in 0.13 seconds)
Iteration 600: error is 0.983142 (50 iterations in 0.13 seconds)
Iteration 650: error is 0.974942 (50 iterations in 0.13 seconds)
Iteration 700: error is 0.966844 (50 iterations in 0.13 seconds)
Iteration 750: error is 0.962789 (50 iterations in 0.14 seconds)
Iteration 800: error is 0.959531 (50 iterations in 0.13 seconds)
Iteration 850: error is 0.956856 (50 iterations in 0.14 seconds)
Iteration 900: error is 0.953592 (50 iterations in 0.14 seconds)
Iteration 950: error is 0.950666 (50 iterations in 0.14 seconds)
Iteration 1000: error is 0.947677 (50 iterations in 0.14 seconds)
Fitting performed in 2.73 seconds.
```



Initial Hypothesis:

The Elbow-Graph represented that the best model should have K=3. We can guess why this graph offered K=3 without interpreting the result of our model deployment. One possible reason could be the initial bias in the dataset. Regarding our dataset, the rows can be divided into three main groups in terms of number of crimes. The first group is the one with instances which refer to the crime rate in National. These instances have an enormous number of crimes for each category as they are the sum of other areas.

The second group could be the one which contains the Metropolitan and City instances. Considering London's vast population, this area has a massive crime rate compared to the other areas. Therefore, possibly, our model clustered them in a separate category. Moreover, the third group includes instances of the other areas. Compared to the two mentioned areas, the crime rates in these areas are low so that they can be clustered in a group.

Analysis:

We deployed these three models to evaluate the elbow graph analysis. The output of these models proves our initial guess for why K=3 is the optimum K for our model. According to the proportion of each cluster in model 3, we can see that the number of instances in two clusters is 50, while the other cluster has 2050 instances. On the other hand, our dataset contains the data for five years except for ten months, so it contains 50 months in total. As a result, we have a total number of 50 instances which refers to the National area, and also 50 instances for the Metropolitan and City. Consequently, the reason why these clusters have 50 instances is highly likely that they are made up of the mentioned instances.

It can be seen that this pattern also repeats for the second model (K=4). This model has two small clusters with 50 instances for each. Apparently, this cluster contains the same instances as the first model (K=3). Furthermore, regarding the t-SNE graph for this model, it can be seen that the bigger cluster split into two smaller clusters. This new segmentation is possibly based on the number of crimes for their instances.

Concerning the t-SNE graph for the third model (K=5), again, two clusters with 50 objects for each can be detected among the clusters, which are the same cluster as previous models. However, despite model 2, the big central cluster split into three subgroups rather than two. Additionally, in models with more than 3 clusters, we can see that some data points share local and global characteristics considering PCA and t-SNE visualisation. Consequently, it again supports that our Elbow-method hypothesis is rational.

Classification

Pipeline:

Initially, we deploy our classification model. Then we calculate the metrics for evaluating our models, such as the accuracy and confusion matrix. Based on the confusion matrix of each model, we calculate Sensitivity, Specificity, Pos Pred Value, Neg Pred Value, Prevalence, Detection Rate, Detection Prevalence, and Balanced Accuracy metrics. These metrics help us to decide which model outperforms. We will use three different classifiers, which are Multinomial Logistic Regression, RandomForest Classifier, and Support Vector Machine (SVM). All metrics will be calculated for those models. Also, a confusion matrix will be plotted to illustrate a visually comprehensive demonstration of the confusion matrix.

Metrics for Evaluation:

Accuracy:

The primary metric that was initially calculated for comparing models is accuracy. It is a performance metric that measures the proportion of correct predictions made by a model, calculated as the ratio of correct predictions to the total number of predictions. (Applied predictive modelling, 2018)

Sensitivity:

It, also known as actual positive rate or recall, is a performance metric in binary classification that measures the proportion of actual positive cases correctly identified by the model. It is calculated as the ratio of true positives to the sum of true positives and false negatives.(Applied predictive modelling, 2018)

Specificity:

It is a performance metric in binary classification that measures the proportion of actual negative cases correctly identified by the model. It is calculated as the ratio of true negatives to the sum of true negatives and false positives.(Applied predictive modelling, 2018)

Positive Predictive Value (Pos Pred Value):

It, also known as precision, quantifies the proportion of predicted positive cases that are actually true positives. It is calculated as the ratio of true positives to the sum of true positives and false positives.(Applied predictive modelling, 2018)

Negative Predictive Value (Neg Pred Value):

It represents the proportion of predicted negative cases that are actually true negatives. It is calculated as the ratio of true negatives to the sum of true negatives and false negatives.(Applied predictive modelling, 2018)

Prevalence:

It is the proportion of the population or dataset that has the condition or outcome of interest. It serves as a baseline measure for interpreting the performance of a classification model.(Applied predictive modelling, 2018)

Detection Rate:

It, also known as hit rate or true positive rate, measures the proportion of actual positive cases correctly identified by the model. It is calculated as the ratio of true positives to the sum of true positives and false negatives.(Applied predictive modelling, 2018)

Detection Prevalence:

It illustrates the proportion of the population or dataset that the model identifies as positive. It is calculated as the ratio of the sum of true positives and false positives to the sum of true positives, false positives, and false negatives.(Applied predictive modelling, 2018)

Balanced Accuracy:

It is an overall evaluation metric that takes into account both sensitivity and specificity. It calculates the average of sensitivity and specificity, providing a balanced measure of the model's performance across both positive and negative cases.(Applied predictive modelling, 2018)

Visualisation:

Confusion Matrix:

It is a visual representation of a classification model's performance, displaying the counts of true positives, true negatives, false positives, and false negatives for each class.

Variable Importance plot: (RandomForest Only)

It is a graphical representation that shows the relative importance or contribution of different input variables in a predictive model. It helps in identifying the key variables that have the most significant impact on the model's predictions and can assist in feature selection, model interpretation, and identifying important factors driving the outcome of interest.

Out-Of-Bag Error plot: (RandomForest Only)

An OOB error vs tree is a graph that shows the relationship between the number of trees in a random forest model and the corresponding OOB error. It helps determine the optimal number of trees for the model. The plot starts with a higher error, which decreases as more trees are added. It helps find the point where additional trees provide minimal improvement in the model's performance.

In [242…

```r
# Creating a function to plot confusion matrix
confusion_matrix_plotter <- function(predicted_classes, actual_classes, model){
predicted_classes <- predict(model, newdata = test_data)
confusion_matrix <- table(predicted_classes, actual_classes)
confusion_df <- as.data.frame(confusion_matrix)
options(repr.plot.width = 25.2, repr.plot.height = 15)
ggplot(data = confusion_df, aes(x = predicted_classes, y = actual_classes, fill = Freq)) +
  geom_tile(colour = "white") +
  geom_text(aes(label = Freq), size = 10, color = "brown") +
  scale_fill_gradient(low = "lightblue", high = "darkblue") +
  labs(x = "Predicted Class", y = "Actual Class") +
  theme_minimal()+
    theme(
        axis.text = element_text(size = 21),
        axis.title = element_text(size = 21),
        plot.title = element_text(size = 30),
```

```
        axis.text.x = element_text(size = 18),
        axis.text.y = element_text(size = 18),
        legend.text = element_text(size = 17)
    )
}
```

In [243…
```
# Creating a function to generate metrics of the model
metrics_summary <- function(predicted_classes, actual_classes){
    predicted_classes <- as.factor(predicted_classes)
    actual_classes <- as.factor(actual_classes)
    levels(predicted_classes) <- levels(actual_classes)
    confusionMatrix(predicted_classes, actual_classes)
}
```

Multinomial Logistic Regression

Deploying the model

In [244…
```
data <- crime[,-c(2,3,4,5)]
set.seed(123)
train_indices <- sample(1:nrow(data), 0.7 * nrow(data))
train_data <- data[train_indices, ]
test_data <- data[-train_indices, ]
model <- multinom(region ~ ., data = train_data)
predicted_classes <- predict(model, newdata = test_data, type = "class")
actual_classes <- test_data$region
accuracy <- sum(predicted_classes == actual_classes) / length(actual_classes)
cat("Accuracy:", accuracy, "\n")
```

```
# weights:  60 (42 variable)
initial  value 2086.373013
iter  10 value 1927.821801
iter  20 value 1885.242910
iter  30 value 1705.317081
iter  40 value 1490.867011
iter  50 value 1237.358987
iter  60 value 1195.949802
iter  70 value 1193.554604
iter  80 value 1193.552303
final  value 1193.548767
converged
Accuracy: 0.6418605
```

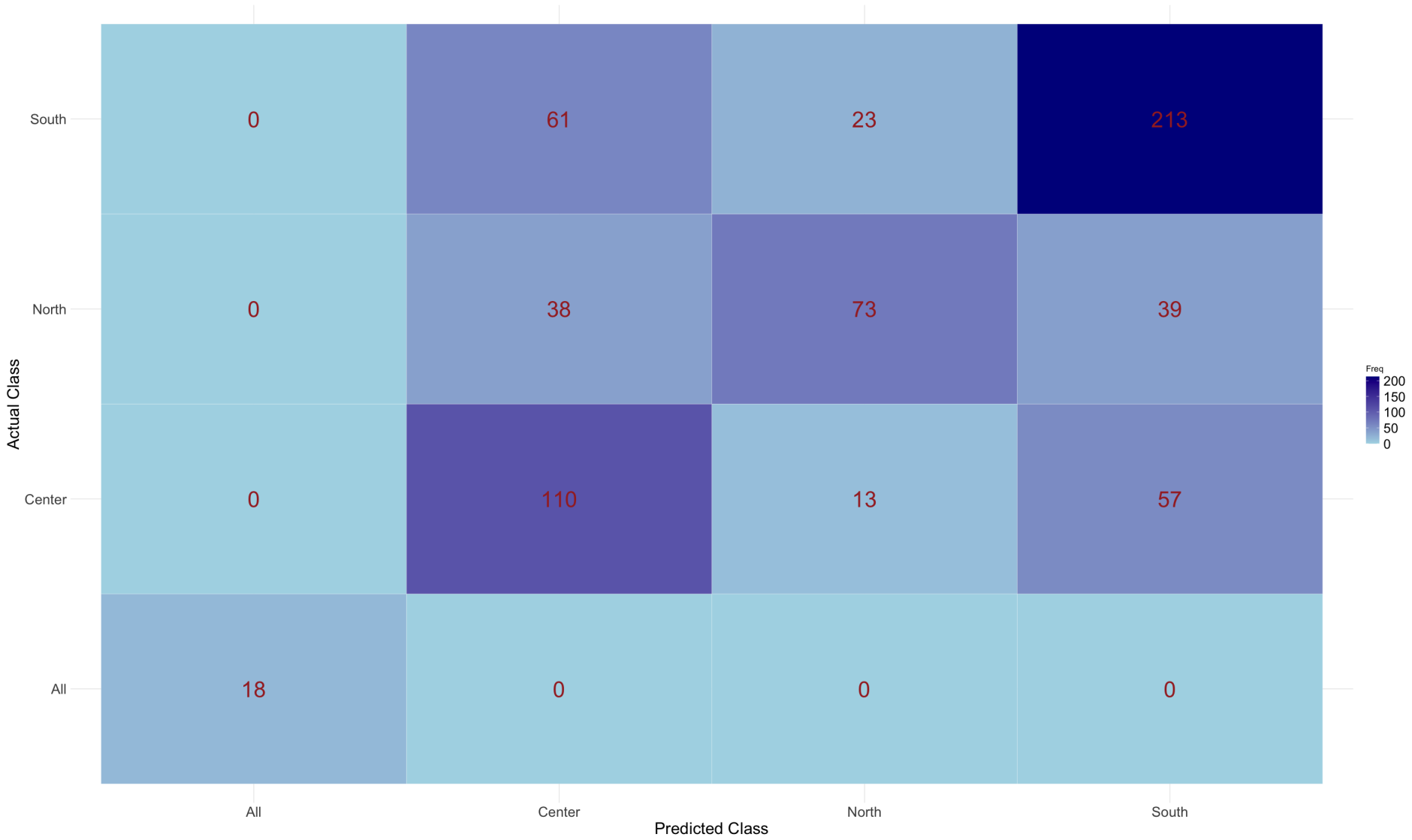
Model Metrics based on Confusion Matrix

In [245…
```
metrics_summary(predicted_classes, actual_classes)
```

```
Confusion Matrix and Statistics

          Reference
Prediction All Center North South
    All     18      0     0     0
    Center   0    110    38    61
    North    0     13    73    23
    South    0     57    39   213

Overall Statistics

               Accuracy : 0.6419
                 95% CI : (0.6035, 0.6789)
    No Information Rate : 0.4605
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4481

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: All Class: Center Class: North Class: South
Sensitivity             1.00000        0.6111       0.4867       0.7172
Specificity             1.00000        0.7871       0.9273       0.7241
Pos Pred Value          1.00000        0.5263       0.6697       0.6893
Neg Pred Value          1.00000        0.8394       0.8563       0.7500
Prevalence              0.02791        0.2791       0.2326       0.4605
Detection Rate          0.02791        0.1705       0.1132       0.3302
Detection Prevalence    0.02791        0.3240       0.1690       0.4791
Balanced Accuracy       1.00000        0.6991       0.7070       0.7207
```

Plotting the Confusion Matrix

In [246…
```
confusion_matrix_plotter(predicted_classes, actual_classes, model)
```



Analysis:

The multinomial Logistic Regression model has an accuracy of around %64, which is not that much significant. Also, considering the confusion metrics, we can see our model has its weakest performance at predicting the regions which are labelled with Center. It has 110 True predictions for this category, while it has a total number of 99 false predictions for this class. Thirty-eight false predictions are predicted as North, while 61 is predicted as South. Considering the metrics and accuracy of the model, this model is not appropriate for our dataset.

RandomForest Classifier

Deploy the model

In [247…

```
data <- crime[,-c(2,3,4,5)]
set.seed(123)
train_indices <- sample(1:nrow(data), 0.7 * nrow(data))
train_data <- data[train_indices, ]
train_data$region <- as.factor(train_data$region)
test_data <- data[-train_indices, ]
model <- randomForest(region ~ ., data = train_data, ntree = 100)
predicted_classes <- predict(model, newdata = test_data)
actual_classes <- test_data$region
accuracy <- sum(predicted_classes == actual_classes) / length(actual_classes)
cat("Accuracy:", accuracy, "\n")
```

```
Accuracy: 0.8465116
```

Model Metrics based on Confusion Matrix

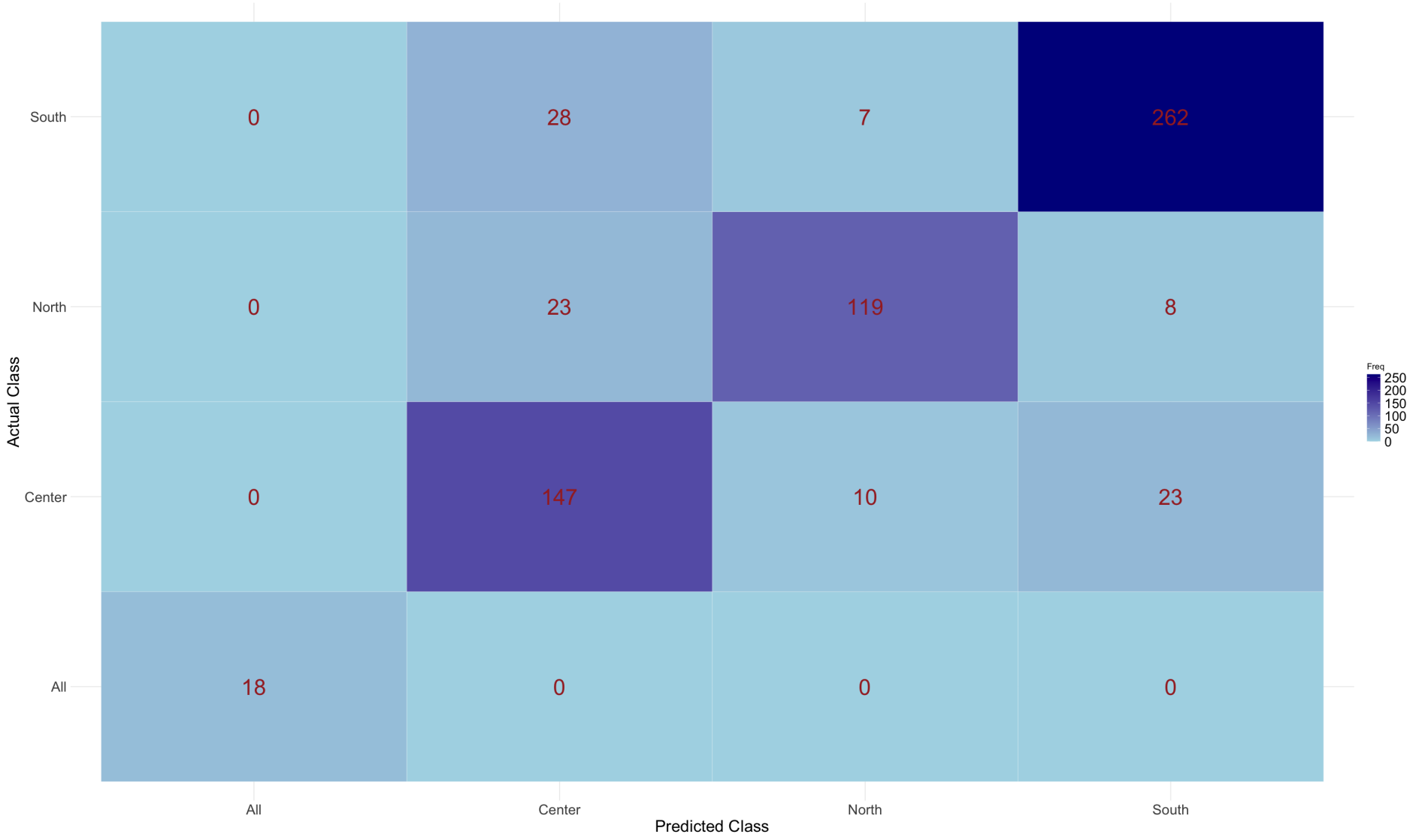
In [248…

```
metrics_summary(predicted_classes, actual_classes)
```

```
Confusion Matrix and Statistics

          Reference
Prediction All Center North South
    All     18      0     0     0
    Center   0    147    23    28
    North    0     11   119     7
    South    0     22     8   262

Overall Statistics

               Accuracy : 0.8465
                 95% CI : (0.8163, 0.8735)
    No Information Rate : 0.4605
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7659

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: All Class: Center Class: North Class: South
Sensitivity             1.00000        0.8167       0.7933       0.8822
Specificity             1.00000        0.8903       0.9636       0.9138
Pos Pred Value          1.00000        0.7424       0.8686       0.8973
Neg Pred Value          1.00000        0.9262       0.9390       0.9008
Prevalence              0.02791        0.2791       0.2326       0.4605
Detection Rate          0.02791        0.2279       0.1845       0.4062
Detection Prevalence    0.02791        0.3070       0.2124       0.4527
Balanced Accuracy       1.00000        0.8535       0.8785       0.8980
```

Plotting Confusion Matrix

In [249…
```
confusion_matrix_plotter(predicted_classes, actual_classes, model)
```

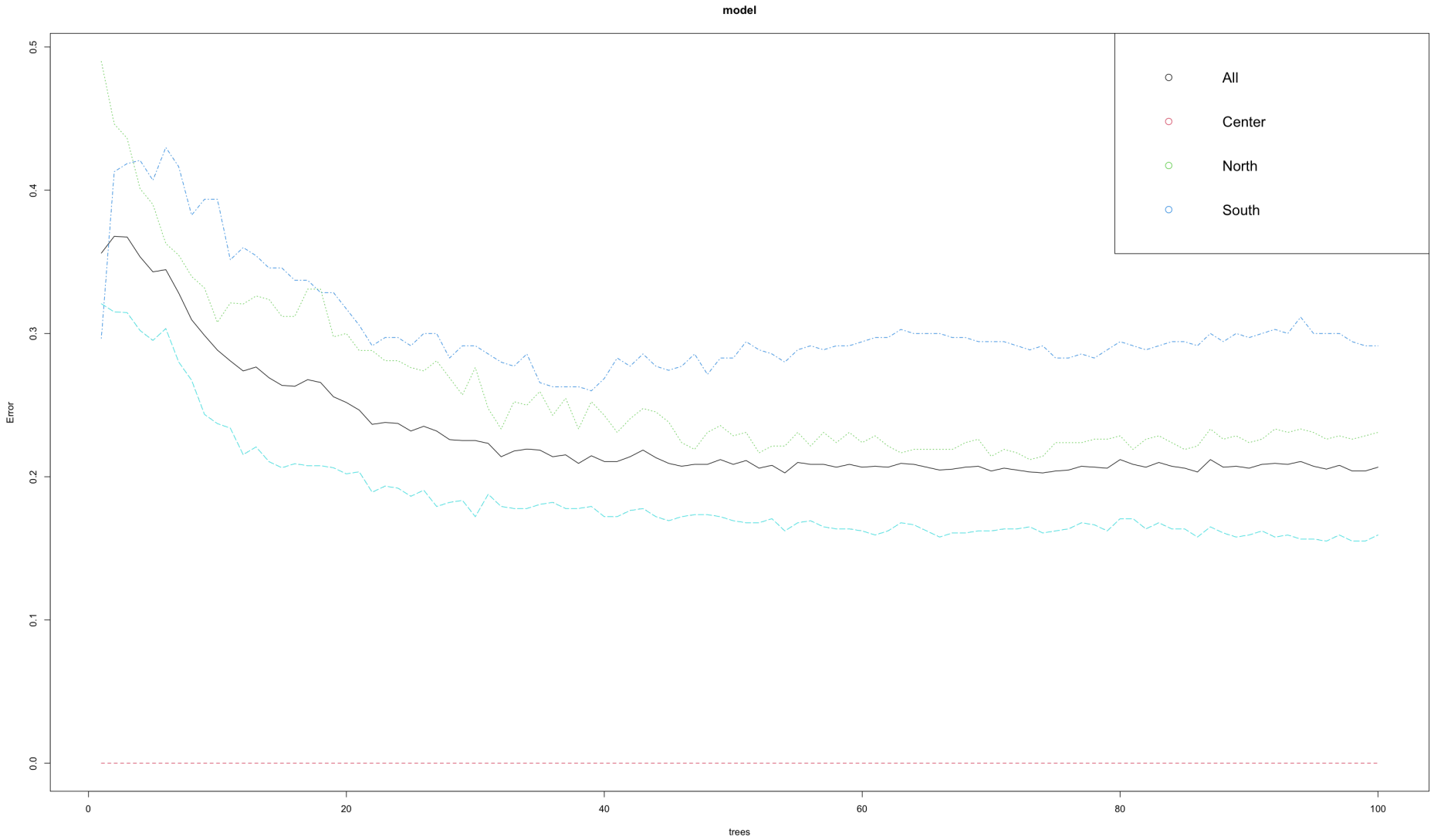


Analysis:

RandomForest model has an accuracy of almost %85, which is significantly better than the Multinomial Logistic Regression model. Furthermore, considering the confusion matrix, our model predicts excellent for categories with "All" label and accurately for South and North. However, the same pattern of false predictions for the Center label can be seen. Although the rate of true positives to false positives for this category reduced significantly compared to the previous model, it needs more improvement.

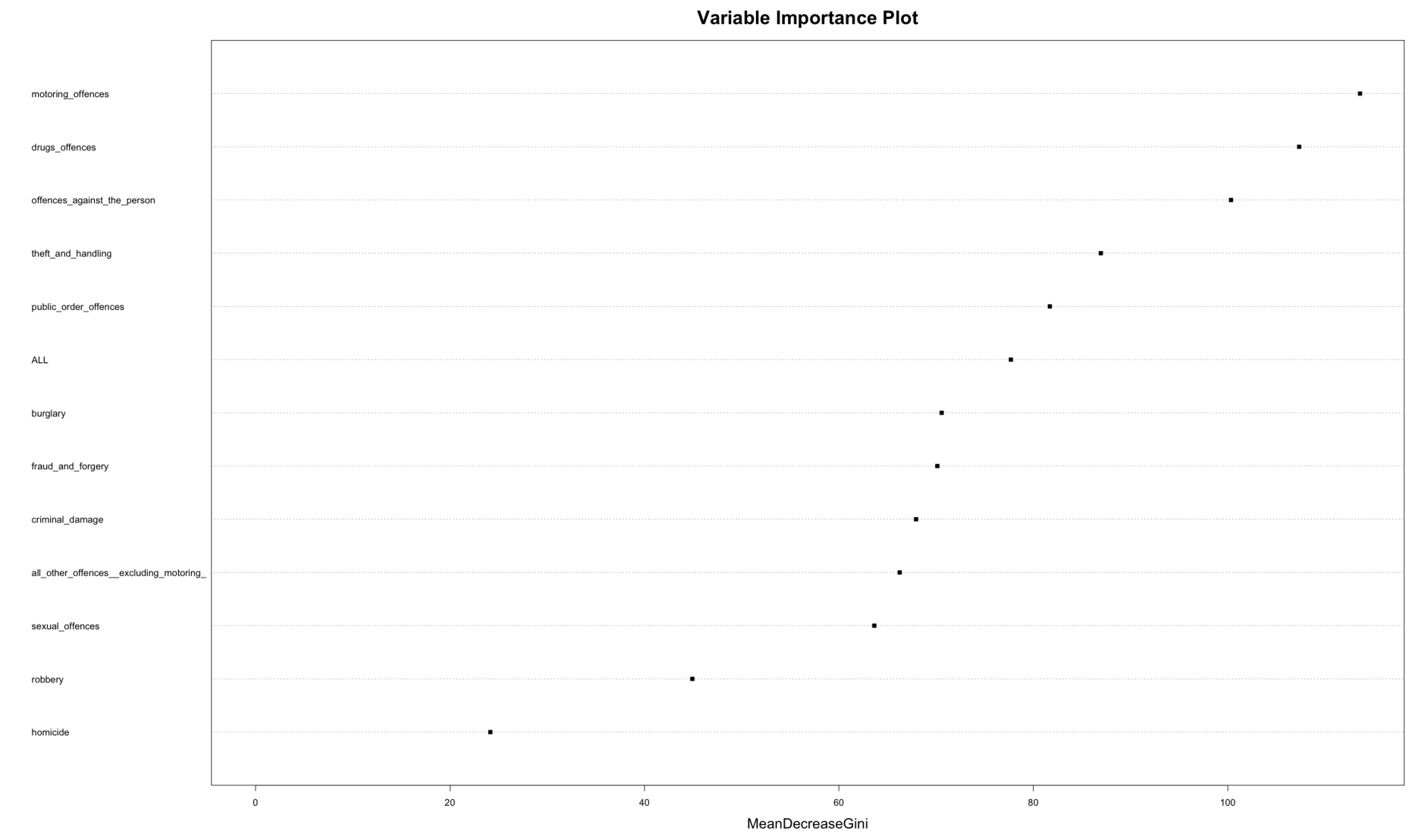
Plotting the OOB Error vs Trees

In [250…
```
plot(model)
legend("topright", legend = levels(train_data$region), col = 1:length(levels(train_data$region)), pch = 1, cex = 1.5)
```



Plotting the Variable Importance plot

In [251…
```
varImpPlot(model, main = "Variable Importance Plot", cex.axis = 1.5, cex.lab = 1.5, cex.main = 2, pch = 15)
```



Analysis:

As can be understood from the name of this graph, it represents the Importance of each variable in our model. Regarding the graph, by removing the motoring_offences column from our dataset, we will have the most significant loss in the model's accuracy. Also, this reduction would be high for removing drug_offences and offences_against_the_person categories. On the other hand, homicide and robbery have less effect on our model's significance. Considering this demonstration, it can be comprehended that the categories with the most crimes have the most considerable effect on the model.

SVM Classifier

Deploying the Model

In [252…
```
set.seed(123)
train_indices <- sample(1:nrow(crime[,-c(2,3,4,5)]), 0.7 * nrow(crime[,-c(2,3,4,5)]))  # 70% for training
train_data <- crime[train_indices, ]
test_data <- crime[-train_indices, ]
train_data$region <- as.factor(train_data$region)
```

```r
svm_model <- svm(region ~ ., data = train_data)
predicted_classes <- predict(svm_model, newdata = test_data)
actual_classes <- test_data$region
accuracy <- sum(predicted_classes == actual_classes) / length(actual_classes)
cat("Accuracy:", accuracy, "\n")
```

```
Accuracy: 1
```

Model Metrics based on Confusion Matrix

In [253…

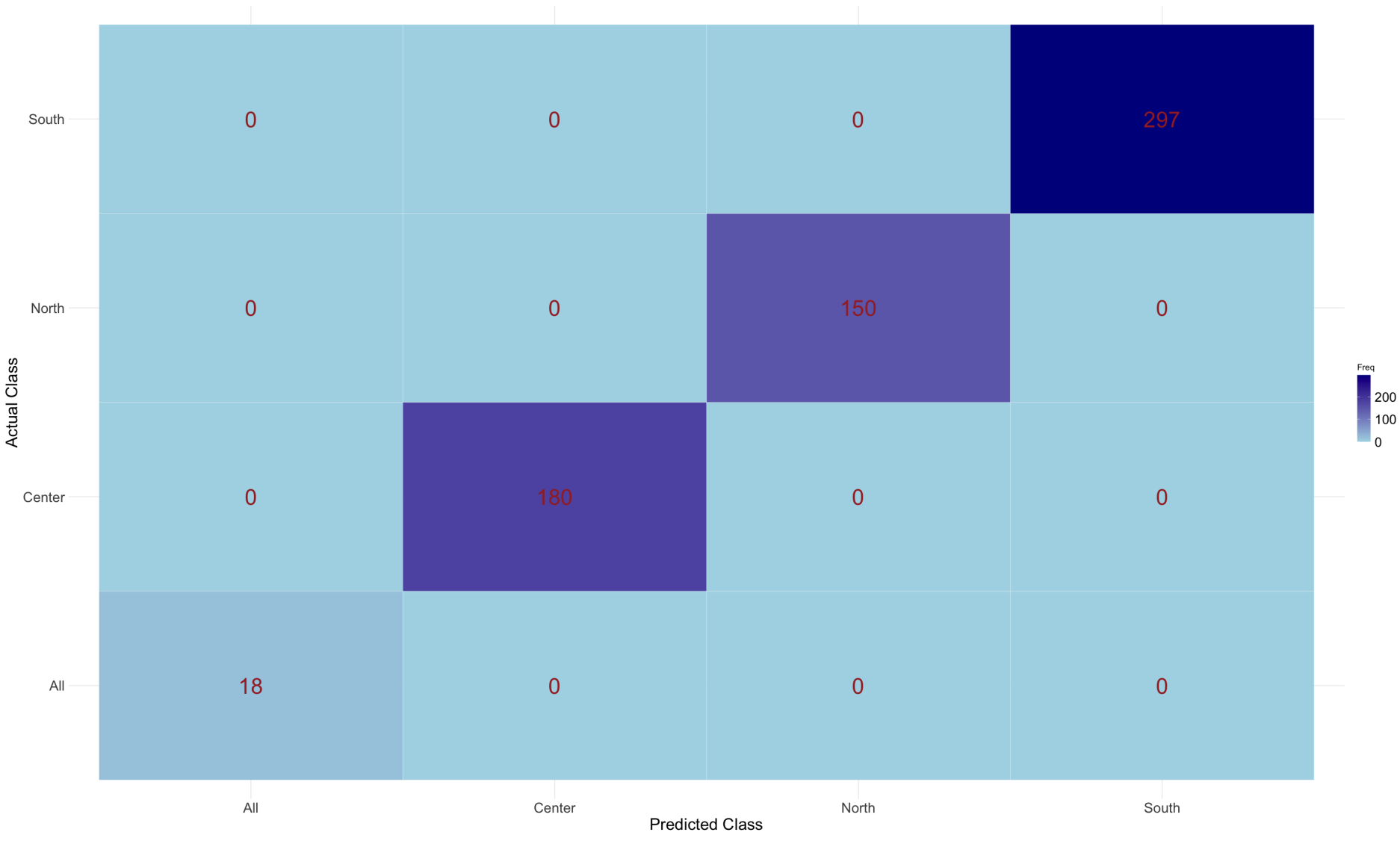
```r
metrics_summary(predicted_classes, actual_classes)
```

```
Confusion Matrix and Statistics

          Reference
Prediction All Center North South
    All     18      0     0     0
    Center   0    180     0     0
    North    0      0   150     0
    South    0      0     0   297

Overall Statistics

               Accuracy : 1
                 95% CI : (0.9943, 1)
    No Information Rate : 0.4605
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: All Class: Center Class: North Class: South
Sensitivity             1.00000        1.0000       1.0000       1.0000
Specificity             1.00000        1.0000       1.0000       1.0000
Pos Pred Value          1.00000        1.0000       1.0000       1.0000
Neg Pred Value          1.00000        1.0000       1.0000       1.0000
Prevalence              0.02791        0.2791       0.2326       0.4605
Detection Rate          0.02791        0.2791       0.2326       0.4605
Detection Prevalence    0.02791        0.2791       0.2326       0.4605
Balanced Accuracy       1.00000        1.0000       1.0000       1.0000
```

Plotting Confusion Matrix

In [254…

```r
confusion_matrix_plotter(predicted_classes, actual_classes, svm_model)
```



Analysis:

Out of the three models evaluated, the SVM model exhibited unparalleled performance, boasting a remarkable accuracy of 100%. Its flawless classification of the dataset based on regional characteristics highlights its exceptional suitability for our analysis. By accurately capturing the intricate patterns and variations in crime rates across different regions, the SVM model enables precise categorization without any errors. This high level of accuracy is particularly valuable when it comes to critical decision-making processes and optimizing resource allocation strategies. Consequently, the SVM model emerges as the indisputable top choice and the preferred classifier for our dataset. Its ability to provide reliable insights into regional crime patterns equips us with the necessary knowledge to develop targeted interventions and implement effective policies tailored to address specific challenges in each region. The outstanding performance of the SVM model instills confidence and solidifies its role as an invaluable tool in our analysis of regional crime data.

References:

1- Sarkar, A., & Goyal, C. (2019). Talent Management and Employee Performance. Sankalpa, 9/10(2/1), 72.

2- Performance metrics for Time-series Forecasting models - Data Analytics. https://vitalflux.com/performance-metrics-for-time-series-forecasting-models/

3- Metrics and Plots for Analyzing Linear regression models https://medium.com/ml-course-microsoft-udacity/metrics-and-plots-for-analyzing-linear-regression-models-43b533547574

4- Kuhn, M., & Johnson, K. (2018). Applied predictive modeling. Springer.