USAMA SHAHID

ID: 4214832

DATA ANALYSIS AND VISUALIZATION REPORT

Crown Prosecution Service Case Outcomes

Table of Contents

1. Executive Summary

The data utilized to construct this analysis was derived from an uneven array of months; while those appended to the various years involved do not feature in their archive, they have nevertheless been taken into consideration.

It is a detailed outline for a report on data pre-processing, cleaning and analysis of crime dataset. The report covers various steps in the data pre-processing and cleaning such as dropping percentage columns, adding and sorting by date, shifting columns, removing special characters and converting to integer, and handling missing data.

It then covers the descriptive analytics of the cleaned dataset, analyzing various attributes such as county, year, month, region, and different types of crimes such as homicide, sexual offences, burglary, and more. The report also includes code snippets for the various steps in the pre-processing and cleaning process.

The report also includes predictive analytics and time series analysis. It includes linear and multiple regression, clustering and classification. For linear and multiple regression, it covers hypothesis, dataset summary, model summary, accuracy matrix, predicted vs actual plot, and evaluating statistics. It also includes code snippets for splitting the data and implementing linear and multiple regression.

For clustering, it covers K-means and Hierarchical clustering, with hypothesis, dataset summary, cluster dendrogram and summary. It also includes code snippets for removing non-numeric columns and implementing K-means and hierarchical clustering.

For classification, it covers SVM and Random Forest, with hypothesis, dataset summary, model summary, classification report, model ROC and code snippets for implementing SVM and Random Forest.

For Time series analytics, it covers ARIMA, Hypothesis, Dataset Summary, Model Summary, Accuracy Matrix, Predicted vs Actual Plot, and Evaluating Statistics. It also includes code snippets for splitting the data and implementing ARIMA models.

2. Data Overview

The dataset utilized for this report is the Crown Prosecution Service Case Outcomes by Principal Offense Category (POC) obtained from data.gov.uk website. The CPS outcomes are categorized into convictions and unsuccessful verdicts, with data spanning from 2014 to 2018 collected on a monthly basis in forty-two (42) counties throughout England where applicable.

The convictions comprise of guilty pleas, trials resulting in convictions and verdicts rendered against respondents who have not appeared in court. An incomplete outcome encompasses all other categories, equally comprising discontinuances and withdrawals; discharged committals; dismissals or acquittals; as well as administrative finalizations.

The offenses recorded comprise homicide, violations against the person such as sexual assault, burglary, robbery and theft; in addition to handling fraud or forgery along with criminal damage committed to public places and automobiles. All other offenses except motoring-related offenses are included within this category.

3. Data Integration

The dataset was scattered in the shape of directories for each year whereas each directory contained all possible months data for the parent directory representing the year. In response to the dataset shape, the chosen strategy was to first write a generic function that reads all possible files within the directories and create a hashmap. Afterwards, another function that reads the hashmap and merges all the files into a singular dataframe and within the process it creates columns for year and month extracted from the directory name and the file name respectively.

3.1. Utilized Code

The sections below contain each operation's code and description respectively.

3.1.1. Get_all_files_from_directories

This function is used to fetch all the files listed under specified directories and return a hash containing directories and filenames.

```
get_all_files_from_directories <- function () {
    files <- hash()

    files["2014"] <- list.files("dataset/2014", pattern=".csv")
    files["2015"] <- list.files("dataset/2015", pattern=".csv")
    files["2016"] <- list.files("dataset/2016", pattern=".csv")
    files["2017"] <- list.files("dataset/2017", pattern=".csv")
    files["2018"] <- list.files("dataset/2018", pattern=".csv")

    return(files)
}
```

3.1.2. Merge_files

The code takes input of a hashmap, reads and merges all the files into one dataframe by creating appropriate columns to distinguish data.

```
merge_files <- function(hash) {
    year <- names(hash)

    combined_df <- do.call(rbind, lapply(year, function(y) {
    do.call(rbind, lapply(hash[[y]], function(f) {
        f_name <- paste("dataset/", y, "/", f, sep="")
            df <- read.csv(f_name, stringsAsFactors = FALSE)
            df$year <- y
            df$month <- tolower(gsub(".csv", "", as.list(strsplit(f,
"_")[[1]])[4]))
            df
    }))
  }))
  return(combined_df)
}
```

3.2. Integrated Dataset Shape

After the integration was conducted, the dataset took the following structure as shared below.

```
In [29]: glimpse(df)

Rows: 2,193
Columns: 54
$ X.1                                                          <int> 1, …
$ 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…
$ year                                                         <int> 201…
$ month                                                        <chr> "ap…
```

4. Data Pre-process

In order to carry out all data analytics with accurate results as well as to enable all the statistical algorithm's smooth working, a series of data pre-processing steps are

performed to make sure the data is converted into highest quality. We would first look at each step one by one with reasoning and impact. Along with that, we would explore the code utilized to execute each operation.

4.1. Data Cleaning

Data cleaning is the process of identifying and correcting or removing inaccuracies and inconsistencies in data. This process is important for data analytics because it ensures that the data being analyzed is accurate and reliable, which in turn leads to more accurate and reliable insights. Without proper data cleaning, the results of data analysis may be misleading or incorrect. Additionally, data cleaning can also make the data more usable by removing unnecessary information and making it more consistent. This can make it easier and more efficient to work with the data. The sections below contain each operation performed with reasoning and impact.

4.1.1. Dropping Percentage Columns

By removing these columns, the data set is made more manageable and the analysis can be focused on the most important variables. Additionally, percentage columns can be dropped when the data is skewed, which can cause problems in modeling and can lead to inaccurate results. Moreover, if required they can be re-generated using other attributes.

4.1.1.1. Code

The functions drops all the columns that keyword “Percentage” in their column name from the dataframe.

```
drop_percentage_columns <- function(dataframe) {
  col_names <- colnames(dataframe)
  to_drop <- grep("Percentage", col_names, value = TRUE)
  dataframe <- dataframe[, !(col_names %in% to_drop)]
  return(dataframe)
}
```

4.1.2. Add & Sort By Date

This data cleaning operation is adding a new column to the dataset that combines the year and month columns, creating a date field. This can be useful for sorting the dataset by the date field, making it easier to analyze the data over time. The impact of this operation is that it will make the dataset more easily readable and usable for analysis by adding a date field, and it will sort the data by that field, which can make it easier to track trends over time. The reasoning behind this is to make the data more usable and to make it easier to track patterns and trends over time.

4.1.2.1. Code

The function creates a new column based of year & month and sorts the dataframe based of that column.

```
sort_by_yearmon <- function(dataframe){
  dataframe$yearmon <- as.Date(paste(dataframe$year, dataframe$month,
"01", sep = "-"), "%Y-%b-%d")
  dataframe <- dataframe[order(dataframe$yearmon),]
  return(dataframe)
}
```

4.1.3. Shifting Columns

This operation is used to re-organize the data set to make it easier to navigate, read, understand, interpret and utilize.

4.1.3.1. Code

The function shifts the last two columns after the first column.

```
shift_columns <- function(dataframe){
  cols <- colnames(dataframe)
  cols <- c(cols[1], cols[(length(cols)-2):length(cols)],
cols[2:(length(cols)-3)])
  dataframe[, cols]
}
```

4.1.4. Rename Columns

This operation is used to remove repetitive text from the column names to make them more shorter, readable and easier to use as well as visualize during various analyses.

4.1.4.1. Code

The function removes repetitive keywords like “Number.of.” from the column names and renames them.

```
rename_columns <- function(dataframe){
  colnames(dataframe) <- gsub("Number.of.", "", colnames(dataframe))
  colnames(dataframe) <- gsub("\\.", "_", colnames(dataframe))
  colnames(dataframe) <- tolower(colnames(dataframe))
  return(dataframe)
}
```

4.1.5. Remove Special Characters and Convert to Integer

This operation removes any unnecessary text such as “,” within the column values and converts them into integer type so they can be used easily during data analysis. Most of the models require integer input or some perform better within the format. Mainly, the base nature of these columns should be Integer.

4.1.5.1. Code

The function is used to remove special characters and convert most of the data frame columns to integer type.

```
remove_special_characters_and_convert_to_integer <-
function(dataframe){
  dataframe <- dataframe %>%
    mutate_all(funs(gsub(",", "", .)))
  dataframe[,5:ncol(dataframe)] <-
sapply(dataframe[,5:ncol(dataframe)], as.integer)
  return(dataframe)
}
```

4.1.6. Rename Month Values

As shown in the figure below, there were multiple representations of a single month within the dataframe which were considered as different months when they represented the same month. This operation makes sure that all the months are converted into a singular format to remove noise.

```
unique(combined_df$month)
```

'january' · 'february' · 'march' · 'april' · 'may' · 'june' · 'july' · 'august' · 'september' · 'october' · 'november' · 'december' · 'jul' · 'aug' · 'sep' · 'oct' · 'nov' · 'dec' · 'jan' · 'feb' · 'mar'

4.1.6.1. Code

The code renames the month values to have a consistent format.

```r
convert_months <- function(df){
  df$month <- gsub("january", "jan", df$month)
  df$month <- gsub("february", "feb", df$month)
  df$month <- gsub( "march", "mar",df$month)
  df$month <- gsub("april", "apr", df$month)
  df$month <- gsub( "may", "may", df$month)
  df$month <- gsub("june","jun",  df$month)
  df$month <- gsub( "july", "jul",df$month)
  df$month <- gsub("august","aug",  df$month)
  df$month <- gsub("september", "sep", df$month)
  df$month <- gsub( "october","oct", df$month)
  df$month <- gsub("november", "nov", df$month)
  df$month <- gsub("december","dec",  df$month)
  return(df)
}
```

4.1.7. Missing Data

This column visualizes the missing months within respective years that are missing from the dataset. The data wasn't imputed as in time series when the missing data is not missing at random, or when there are no strong assumptions about how the missing data is generated. In this case, imputing data may introduce bias in the data, leading to inaccurate analysis. Also, imputing the data may not be able to provide accurate results. So, at the earlier stage of analysis the missing data is left as it is but we impute it at later stage of analysis as required.

```r
get_missing_months(combined_df)
```

```
$`2014`
$`2015`
'nov'
$`2016`
'feb' · 'mar'
$`2017`
'apr' · 'may' · 'jun'
$`2018`
'apr' · 'may' · 'jun'
```

4.1.7.1. Code

The function returns all the missing data of months under the respective year.

```r
get_missing_months <- function(dataframe){
  years <- unique(dataframe$year)
  missing_months <- list()
  for (year in years){
    months <- unique(dataframe[dataframe$year == year,]$month)
    all_months <- c("jan", "feb", "mar", "apr", "may", "jun", "jul",
"aug", "sep", "oct", "nov", "dec")
    diff <- setdiff(all_months, months)
    missing_months[[year]] <- diff
  }

  return(missing_months)
}
```

4.1.8. Label Region Based on County

This operation is creating a new column in the dataframe called "region" and assigning values to it based on the values of the "county" column. The values are being determined by using the county_region_map list as a lookup table. This can be useful for grouping and analyzing the data by region, rather than individual counties. It can also be useful for creating aggregate statistics for regions, rather than individual counties. Additionally, it will make the data more organized and readable, allowing for more efficient analysis. The impact of this operation is that it will add a new level of analysis to the data, by dividing the county data into regions. The reasoning behind this is to make the data more usable and to make it easier to track patterns and trends for regions, which can be more meaningful for some analysis.

```r
1 unique(combined_df$region)
```

'All' · 'West' · 'East' · 'North' · 'South'

```r
county_region_map <- list(
  "National" = "All",
  "Avon and Somerset" = "West",
  "Bedfordshire" = "East",
  "Cambridgeshire" = "East",
  "Cheshire" = "North",
  "Cleveland" = "North",
  "Cumbria" = "North",
  "Derbyshire" = "East",
  "Devon and Cornwall" = "West",
  "Dorset" = "West",
  "Durham" = "North",
  "Dyfed Powys" = "West",
  "Essex" = "East",
  "Gloucestershire" = "West",
  "GreaterManchester" = "North",
  "Gwent" = "West",
  "Hampshire" = "South",
  "Hertfordshire" = "East",
  "Humberside" = "North",
  "Kent" = "South",
  "Lancashire" = "North",
  "Leicestershire" = "East",
  "Lincolnshire" = "East",
  "Merseyside" = "North",
  "Metropolitan and City" = "South",
  "Norfolk" = "East",
  "Northamptonshire" = "East",
  "Northumbria" = "North",
  "North Wales" = "North",
  "North Yorkshire" = "North",
  "Nottinghamshire" = "East",
  "South Wales" = "West",
  "South Yorkshire" = "North",
  "Staffordshire" = "West",
  "Suffolk" = "East",
  "Surrey" = "South",
  "Sussex" = "South",
  "Thames Valley" = "South",
  "Warwickshire" = "West",
  "West Mercia" = "West",
  "West Midlands" = "West",
  "West Yorkshire" = "North",
  "Wiltshire" = "West"
)
```

4.1.8.1. Code

The function creates a new column named region and labels values based on input map and existing county column values.

```
label_county_region <- function(df, county_region) {
  region <- c()

  for (i in 1:nrow(df)) {
    county <- df$x[i]
    region[i] <- county_region[[county]]
  }

  df$region <- region

  return(df)
}
```

4.1.9. Shift Last Column to 5th Index

This functions changes the order of columns in the dataset to make it easier to manage and utilize.

4.1.9.1. Code

This function takes a dataframe as input and moves the last column to the 5th position and returns the modified dataframe.

```
move_last_column_to_5th <- function(dataframe) {
  ncols <- ncol(dataframe)
  region <- dataframe[, ncols]
  dataframe <- dataframe[, -ncols]
  dataframe <- cbind(dataframe[, 1:4], region, dataframe[,
5:(ncols-1)])
  return(dataframe)
}
```

4.1.10. Rename Conviction Columns

This operation was too used to make column names even shorter by removing “convictions” from the crime columns names and converting “unsuccessful” into a short form of “us” in the column names.

4.1.10.1. Code

This function renames the columns of a dataframe, removing the "_convictions" and "_unsuccessful" substrings from the column names and replacing the latter with "_us" if it exists.

```
rename_conviction_columns <- function(dataframe){
  col_names <- colnames(dataframe)
  for (i in 1:length(col_names)){
    if (grepl("_convictions", col_names[i])){
      names(dataframe)[names(dataframe) == col_names[i]] <-
gsub("_convictions", "" ,col_names[i])
    }else{
      names(dataframe)[names(dataframe) == col_names[i]] <-
gsub("_unsuccessful", "_us" ,col_names[i])
    }
  }
  return(dataframe)
}
```

4.2. Dataset Post Cleaning

After all of the steps have been performed the dataset takes shape as visualized in the below sections.

4.2.1. Glimpse

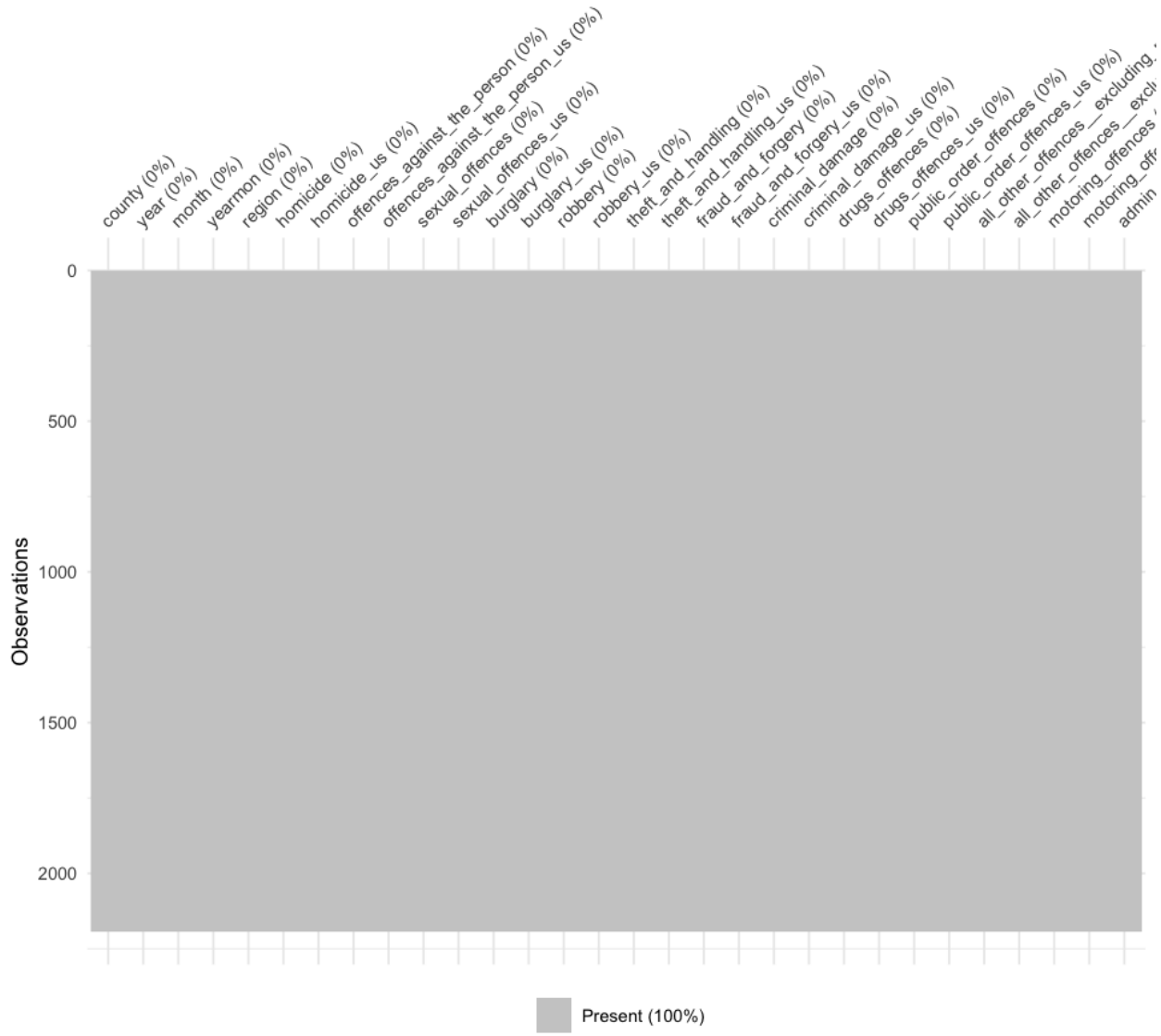
The glimpse() function in R is a part of the dplyr package and it provides a quick and easy way to look at the structure and first few rows of a dataframe or tibble. It is useful for quickly checking the structure and contents of a dataframe without having to print the entire dataset.

```
1 glimpse(combined_df)
```

```
Rows: 2,193
Columns: 30
$ county                                    <chr> "National", "Avon and Somer…
$ year                                      <chr> "2014", "2014", "2014", "20…
$ month                                     <chr> "jan", "jan", "jan", "jan",…
$ yearmon                                   <chr> "2014-01-01", "2014-01-01",…
$ region                                    <chr> "All", "West", "East", "Eas…
$ homicide                                  <int> 51, 0, 0, 0, 0, 2, 0, 0, 0,…
$ homicide_us                               <int> 11, 0, 1, 0, 0, 3, 0, 1, 0,…
$ offences_against_the_person               <int> 9087, 228, 68, 101, 170, 11…
$ offences_against_the_person_us            <int> 2930, 62, 29, 21, 40, 44, 1…
$ sexual_offences                           <int> 736, 35, 2, 10, 15, 11, 4, …
$ sexual_offences_us                        <int> 286, 17, 1, 3, 1, 6, 3, 7, …
$ burglary                                  <int> 1715, 49, 7, 18, 38, 36, 16…
$ burglary_us                               <int> 284, 1, 4, 4, 5, 2, 0, 4, 4…
$ robbery                                   <int> 522, 8, 16, 6, 10, 3, 1, 5,…
$ robbery_us                                <int> 139, 0, 7, 4, 0, 2, 0, 2, 0…
$ theft_and_handling                        <int> 11057, 338, 75, 148, 205, 3…
$ theft_and_handling_us                     <int> 998, 32, 4, 15, 5, 31, 7, 1…
$ fraud_and_forgery                         <int> 846, 18, 17, 10, 14, 11, 6,…
$ fraud_and_forgery_us                      <int> 137, 0, 3, 4, 1, 3, 0, 0, 2…
$ criminal_damage                           <int> 2693, 93, 22, 30, 39, 46, 3…
$ criminal_damage_us                        <int> 472, 14, 8, 3, 3, 13, 3, 12…
$ drugs_offences                            <int> 4988, 148, 31, 47, 64, 65, …
$ drugs_offences_us                         <int> 305, 4, 3, 1, 3, 2, 1, 9, 5…
$ public_order_offences                     <int> 4752, 123, 30, 37, 77, 123,…
$ public_order_offences_us                  <int> 797, 28, 9, 2, 8, 27, 2, 15…
$ all_other_offences__excluding_motoring_   <int> 3291, 63, 13, 28, 50, 34, 5…
$ all_other_offences__excluding_motoring__us <int> 586, 9, 2, 9, 5, 14, 5, 12,…
$ motoring_offences                         <int> 12945, 256, 171, 103, 264, …
$ motoring_offences_us                      <int> 1466, 40, 13, 16, 16, 16, 7…
$ admin_finalised_us                        <int> 890, 20, 12, 14, 13, 3, 12,…
```

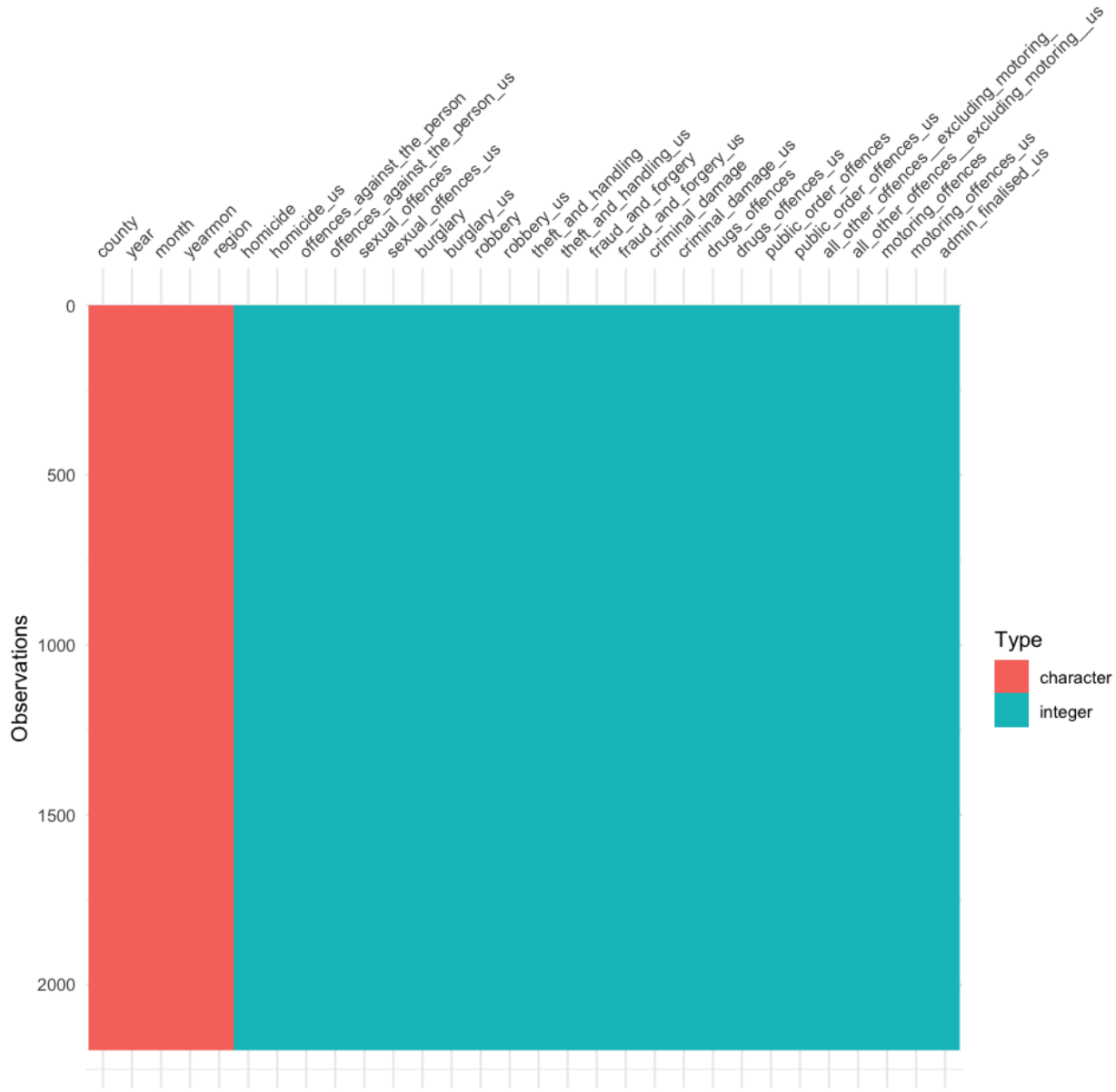
4.2.2. Visualizing Missing

All of the rows have complete data and there are no Nulls or NaNs within the dataset.



4.2.3. Visualizing Data Types

All of the successful and unsuccessful crime columns are successfully converted into Integer types which basically compose 80% of the dataframe and 20% person belongs to other columns.



4.3. Dataset Split

As our dataset had two major types of integer columns, we split the data based on it. The two main categories are Crimes and Un-successful crimes. By splitting the dataset, we can eradicate the repetitive filtering operation during the analysis and also provide a more focused baseline for making hypothesis or conducting any analysis operation.

4.3.1. Crime Dataset Glimpse

The dataset below shows a glimpse of rows, columns and few initial values for the Crime dataset, which contains columns that represent the convictions where the users have either admitted or found guilty of doing the respective crime.

```
1 glimpse(crime_df)
```

```
Rows: 2,193
Columns: 17
$ county                                <chr> "National", "Avon and Somerset…
$ year                                  <int> 2014, 2014, 2014, 2014, 2014, …
$ month                                 <chr> "jan", "jan", "jan", "jan", "j…
$ yearmon                               <chr> "2014-01-01", "2014-01-01", "2…
$ region                                <chr> "All", "West", "East", "East",…
$ homicide                              <int> 51, 0, 0, 0, 0, 2, 0, 0, 0, 0,…
$ offences_against_the_person           <int> 9087, 228, 68, 101, 170, 119, …
$ sexual_offences                       <int> 736, 35, 2, 10, 15, 11, 4, 22,…
$ burglary                              <int> 1715, 49, 7, 18, 38, 36, 16, 3…
$ robbery                               <int> 522, 8, 16, 6, 10, 3, 1, 5, 10…
$ theft_and_handling                    <int> 11057, 338, 75, 148, 205, 334,…
$ fraud_and_forgery                     <int> 846, 18, 17, 10, 14, 11, 6, 15…
$ criminal_damage                       <int> 2693, 93, 22, 30, 39, 46, 38, …
$ drugs_offences                        <int> 4988, 148, 31, 47, 64, 65, 52,…
$ public_order_offences                 <int> 4752, 123, 30, 37, 77, 123, 78…
$ all_other_offences__excluding_motoring_ <int> 3291, 63, 13, 28, 50, 34, 52, …
$ motoring_offences                     <int> 12945, 256, 171, 103, 264, 228…
```

4.3.2. Unsuccessful Crime Dataset Glimpse

The dataset below shows a glimpse of rows, columns and few initial values for the Unsuccessful Crime dataset, which contains columns that represent the unsuccessful cases where the users didn't either admit not found guilty of doing the respective crime.

```
1 glimpse(uscrime_df)
```

```
Rows: 2,193
Columns: 18
$ county                                    <chr> "National", "Avon and Somer…
$ year                                      <int> 2014, 2014, 2014, 2014, 201…
$ month                                     <chr> "jan", "jan", "jan", "jan",…
$ yearmon                                   <chr> "2014-01-01", "2014-01-01",…
$ region                                    <chr> "All", "West", "East", "Eas…
$ homicide_us                               <int> 11, 0, 1, 0, 0, 3, 0, 1, 0,…
$ offences_against_the_person_us            <int> 2930, 62, 29, 21, 40, 44, 1…
$ sexual_offences_us                        <int> 286, 17, 1, 3, 1, 6, 3, 7, …
$ burglary_us                               <int> 284, 1, 4, 4, 5, 2, 0, 4, 4…
$ robbery_us                                <int> 139, 0, 7, 4, 0, 2, 0, 2, 0…
$ theft_and_handling_us                     <int> 998, 32, 4, 15, 5, 31, 7, 1…
$ fraud_and_forgery_us                      <int> 137, 0, 3, 4, 1, 3, 0, 0, 2…
$ criminal_damage_us                        <int> 472, 14, 8, 3, 3, 13, 3, 12…
$ drugs_offences_us                         <int> 305, 4, 3, 1, 3, 2, 1, 9, 5…
$ public_order_offences_us                  <int> 797, 28, 9, 2, 8, 27, 2, 15…
$ all_other_offences__excluding_motoring__us <int> 586, 9, 2, 9, 5, 14, 5, 12,…
$ motoring_offences_us                      <int> 1466, 40, 13, 16, 16, 16, 7…
$ admin_finalised_us                        <int> 890, 20, 12, 14, 13, 3, 12,…
```

4.4. Code

The function splits the data frame into 2 dataframes based on their column names.

```
split_dataframe <- function(df){
  crime_columns = !grepl("_us$", colnames(df))
  unsuccesful_columns = grepl("_us$",  colnames(df))
  unsuccesful_columns[0:5] <- TRUE
  df1 <- df[, crime_columns]
  df2 <- df[, unsuccesful_columns]
  return(list(df1, df2))
}
```

5. Descriptive Analytics

Descriptive analysis is a type of statistical analysis that involves summarizing and describing a set of data. This type of analysis is used to describe the main characteristics of a dataset, such as the central tendency (mean, median, mode), dispersion (range, variance, standard deviation), and distribution of the data. It also includes the use of visualizations such as histograms, box plots, and scatter plots to represent the data and its characteristics (Cote 2021).

The goal of descriptive analysis is to provide a clear and concise summary of the data, highlighting important patterns and trends. It is used to understand the nature of the data and to identify any outliers or unusual observations. Descriptive analysis can also be used to identify any missing data or errors in the dataset.

Some examples of descriptive analysis include:

- Summarizing a dataset using measures of central tendency and dispersion
- Creating a frequency distribution table or a histogram to show the distribution of a variable
- Creating a box plot to show the distribution of a variable and identify outliers
- Creating a scatter plot to show the relationship between two variables.

Descriptive analysis is an essential step in data analysis as it helps to understand the data and identify patterns, which is crucial for any further statistical analysis or modeling. It is also the first step in Exploratory Data Analysis (EDA) which is an approach to analyzing data sets to summarize their main characteristics, often with visual methods.

5.1. Attributes Analysis

Attribute analysis is a process used to understand the characteristics of each variable or feature in a dataset. This can include identifying the data type, missing values, range, distribution, and any patterns or trends in the data.

As our dataset is divided into categories, we investigate each of their columns one by one below.

5.1.1. Crimes Dataset

5.1.1.1. County

This column has all possible county values from england.

```
Length     Class      Mode
     2193 character character
```

5.1.1.2. Year

This column has 5 values because we are using a dataset from 5 years which are 2014, 2015, 2016, 2017 and 2018.

```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   2014    2015    2016    2016    2017    2018
```

5.1.1.3. Month

This column contains months ranging from January to December.

```
Length     Class      Mode
     2193 character character
```

5.1.1.4. Yearmon

This column contains combined values of year and month.

```
Length     Class      Mode
     2193 character character
```

5.1.1.5. Region

This column contains 5 possible values which are All, East, West, North and South.

```
Length     Class      Mode
     2193 character character
```

5.1.1.6. Homicide

It represents that the minimum value of the variable is 0, 25% of the values are below 0, the median value is 1, the mean value is 3.798, 75% of the values are below 3 and the maximum value is 131. It also indicates that the variable has some outliers as the max value is quite high as compared to the mean and the median.

```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000   0.000   1.000   3.798   3.000 131.000
```

5.1.1.7. Offences_against_the_person

It represents that the minimum value of the variable is 29, 25% of the values are below 115, the median value is 179, the mean value is 454.9, 75% of the values are below 272 and the maximum value is 11741. It also indicates that the variable has some outliers as the max value is quite high as compared to the mean and the median.

```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   29.0   115.0   179.0   454.9   272.0 11741.0
```

5.1.1.8. Sexual_offences

It represents that the minimum value of the variable is 0, 25% of the values are below 8, the median value is 15, the mean value is 43.78, 75% of the values are below 29 and the maximum value is 1179. It also indicates that the variable has some outliers as the max value is quite high as compared to the mean and the median.

```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00    8.00   15.00   43.78   29.00 1179.00
```

5.1.1.9. Burglary

It represents that the minimum value of the variable is 1, 25% of the values are below 14, the median value is 23, the mean value is 60.09, 75% of the values are below 38 and the maximum value is 1715. It also indicates that the variable has some outliers as the max value is quite high as compared to the mean and the median.

```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.00   14.00   23.00   60.09   38.00 1715.00
```

5.1.1.10. Robbery

It represents that the minimum value of the variable is 0, 25% of the values are below 2, the median value is 5, the mean value is 19.33, 75% of the values are below 10 and the maximum value is 650. It also indicates that the variable has some outliers as the max value is quite high as compared to the mean and the median.

```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00    2.00    5.00   19.33   10.00  650.00
```

5.1.1.11. Theft_and_handling

It represents that the minimum value of the variable is 13, 25% of the values are below 95, the median value is 147, the mean value is 373.1, 75% of the values are below 237 and the maximum value is 11057. It also indicates that the variable has some outliers as the max value is quite high as compared to the mean and the median.

```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   13.0    95.0   147.0   373.1   237.0 11057.0
```

5.1.1.12. Fraud_and_forgery

It represents that the minimum value of the variable is 0, 25% of the values are below 8, the median value is 13, the mean value is 38.57, 75% of the values are below 21 and the maximum value is 1075. It also indicates that the variable has some outliers as the max value is quite high as compared to the mean and the median.

```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00    8.00   13.00   38.57   21.00 1075.00
```

5.1.1.13. Criminal_damage

It represents that the minimum value of the variable is 3, 25% of the values are below 25, the median value is 40, the mean value is 95.82, 75% of the values are below 59 and the maximum value is 2693. It also indicates that the variable has some outliers as the max value is quite high as compared to the mean and the median.

```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   3.00   25.00   40.00   95.82   59.00 2693.00
```

5.1.1.14. Drugs_offences

It represents that the minimum value of the variable is 4, 25% of the values are below 38, the median value is 63, the mean value is 186.6, 75% of the values are below 100 and the maximum value is 4988. It also indicates that the variable has some outliers as the max value is quite high as compared to the mean and the median.

```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    4.0    38.0    63.0   186.6   100.0  4988.0
```

5.1.1.15. Public_order_offences

It represents that the minimum value of the variable is 2, 25% of the values are below 39, the median value is 63, the mean value is 162.4, 75% of the values are below 100 and the maximum value is 4752. It also indicates that the variable has some outliers as the max value is quite high as compared to the mean and the median.

```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    2.0    39.0    63.0   162.4   100.0  4752.0
```

5.1.1.16. All_other_offences__excluding_motoring_

It represents that the minimum value of the variable is 0, 25% of the values are below 9, the median value is 16, the mean value is 64.34, 75% of the values are below 35 and the maximum value is 3291. It also indicates that the variable has some outliers as the max value is quite high as compared to the mean and the median.

```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00    9.00   16.00   64.34   35.00 3291.00
```

5.1.1.17. Motoring_offences

It represents that the minimum value of the variable is 1, 25% of the values are below 95, the median value is 143, the mean value is 365.5, 75% of the values are below 216 and the maximum value is 12945. It also indicates that the variable has some outliers as the max value is quite high as compared to the mean and the median.

```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    1.0    95.0   143.0   365.5   216.0 12945.0
```

5.1.2. Unsuccessful Crimes Dataset

5.1.2.1. County

This column has all possible county values from england.

```
Length     Class      Mode
     2193 character character
```

5.1.2.2. Year

This column has 5 values because we are using a dataset from 5 years which are 2014, 2015, 2016, 2017 and 2018.

```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   2014    2015    2016    2016    2017    2018
```

5.1.2.3. Month

This column contains months ranging from January to December.

```
Length     Class      Mode
     2193 character character
```

5.1.2.4. Yearmon

This column contains combined values of year and month.

```
Length     Class      Mode
     2193 character character
```

5.1.2.5. Region

This column contains 5 possible values which are All, East, West, North and South.

```
Length     Class      Mode
     2193 character character
```

5.1.2.6. Homicide_us

It represents that the minimum value of the variable is 0, 25% of the values are below 0, the median value is 0, the mean value is 0.9138, 75% of the values are below 1 and the maximum value is 35. It also indicates that the variable has some outliers as the max value is quite high as compared to the mean and the median.

```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0000  0.0000  0.0000  0.9138  1.0000 35.0000
```

5.1.2.7. Offences_against_the_person_us

It represents that the minimum value of the variable is 5, 25% of the values are below 27, the median value is 46, the mean value is 135.4, 75% of the values are below 77 and the maximum value is 3568. It also indicates that the variable has some outliers as the max value is quite high as compared to the mean and the median.

```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    5.0    27.0    46.0   135.4    77.0  3568.0
```

5.1.2.8. Sexual_offences_us

It represents that the minimum value of the variable is 0, 25% of the values are below 1, the median value is 4, the mean value is 16.19, 75% of the values are below 11 and the maximum value is 489. It also indicates that the variable has some outliers as the max value is quite high as compared to the mean and the median.

```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00    1.00    4.00   16.19   11.00  489.00
```

5.1.2.9. Burglary_us

It represents that the minimum value of the variable is 0, 25% of the values are below 1, the median value is 3, the mean value is 10.14, 75% of the values are below 6 and the

maximum value is 317. It also indicates that the variable has some outliers as the max value is quite high as compared to the mean and the median.

```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00    1.00    3.00   10.14    6.00  317.00
```

5.1.2.10. Robbery_us

It represents that the minimum value of the variable is 0, 25% of the values are below 0, the median value is 1, the mean value is 5.16, 75% of the values are below 3 and the maximum value is 188. It also indicates that the variable has some outliers as the max value is quite high as compared to the mean and the median.

```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00    0.00    1.00    5.16    3.00  188.00
```

5.1.2.11. Theft_and_handling_us

It represents that the minimum value of the variable is 0, 25% of the values are below 6, the median value is 11, the mean value is 33.43, 75% of the values are below 19 and the maximum value is 1025. It also indicates that the variable has some outliers as the max value is quite high as compared to the mean and the median.

```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00    6.00   11.00   33.43   19.00 1025.00
```

5.1.2.12. Fraud_and_forgery_us

It represents that the minimum value of the variable is 0, 25% of the values are below 1, the median value is 2, the mean value is 6.232, 75% of the values are below 4 and the maximum value is 180. It also indicates that the variable has some outliers as the max value is quite high as compared to the mean and the median.

```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000   1.000   2.000   6.232   4.000 180.000
```

5.1.2.13. Criminal_damage_us

It represents that the minimum value of the variable is 0, 25% of the values are below 3, the median value is 6, the mean value is 16.43, 75% of the values are below 10 and the maximum value is 491. It also indicates that the variable has some outliers as the max value is quite high as compared to the mean and the median.

```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00    3.00    6.00   16.43   10.00  491.00
```

5.1.2.14. Drugs_offences_us

It represents that the minimum value of the variable is 0, 25% of the values are below 2, the median value is 4, the mean value is 12.57, 75% of the values are below 7 and the maximum value is 346. It also indicates that the variable has some outliers as the max value is quite high as compared to the mean and the median.

```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00    2.00    4.00   12.57    7.00  346.00
```

5.1.2.15. Public_order_offences_us

It represents that the minimum value of the variable is 0, 25% of the values are below 5, the median value is 9, the mean value is 28.45, 75% of the values are below 16 and the maximum value is 801. It also indicates that the variable has some outliers as the max value is quite high as compared to the mean and the median.

```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00    5.00    9.00   28.45   16.00  801.00
```

5.1.2.16. All_other_offences__excluding_motoring__us

It represents that the minimum value of the variable is 0, 25% of the values are below 1, the median value is 3, the mean value is 11.91, 75% of the values are below 7 and the maximum value is 603. It also indicates that the variable has some outliers as the max value is quite high as compared to the mean and the median.

```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00    1.00    3.00   11.91    7.00  603.00
```

5.1.2.17. Motoring_offences_us

It represents that the minimum value of the variable is 0, 25% of the values are below 11, the median value is 20, the mean value is 60.95, 75% of the values are below 34 and the maximum value is 1725. It also indicates that the variable has some outliers as the max value is quite high as compared to the mean and the median.

```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00   11.00   20.00   60.95   34.00 1725.00
```

5.1.2.18. Admin_finalised_us

It represents that the minimum value of the variable is 0, 25% of the values are below 7, the median value is 12, the mean value is 38.82, 75% of the values are below 21 and the maximum value is 1051. It also indicates that the variable has some outliers as the max value is quite high as compared to the mean and the median.

```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00    7.00   12.00   38.82   21.00 1051.00
```

5.2. Analysis dependent on Regions

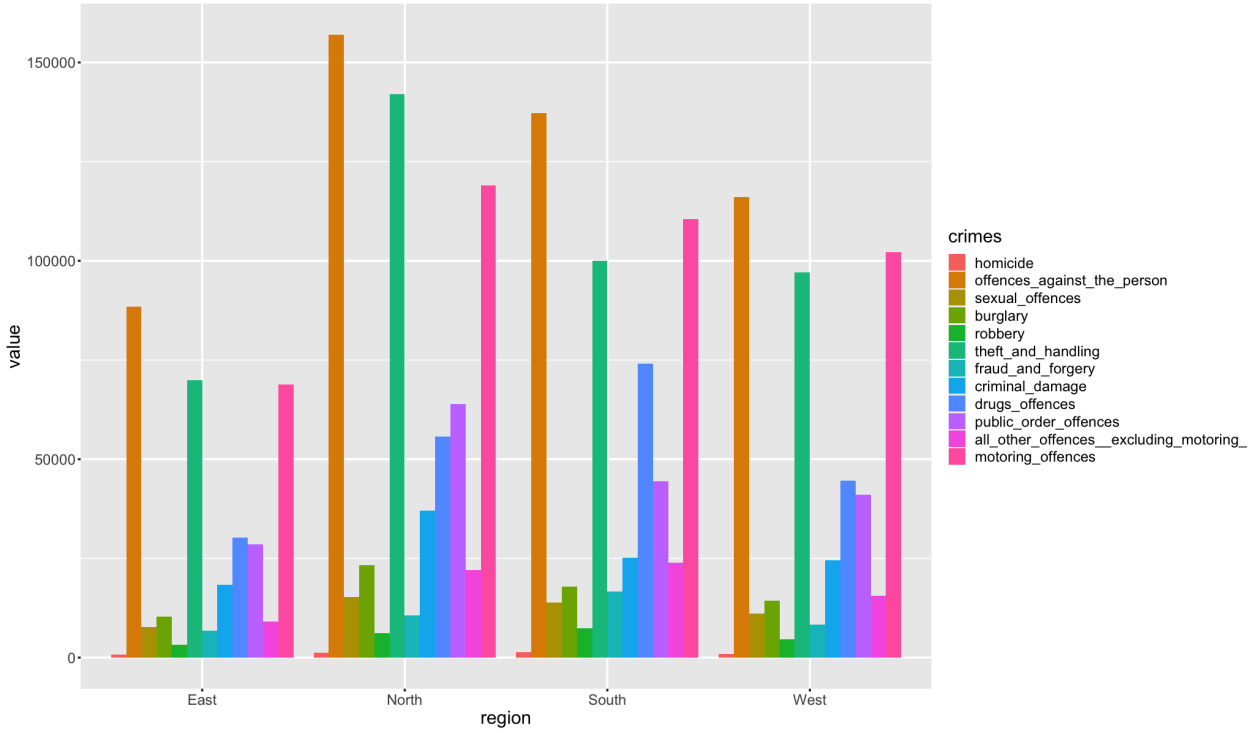
In this section of analysis, we try to understand the relations between crime times and their occurrence in various regions of the englands using graph visualization and statistical analysis.

5.2.1. Region & All Types

We consider both datasets, i.e. crimes and un-successful crimes datasets and visualize their occurrence grouped by crime types in different regions of england.

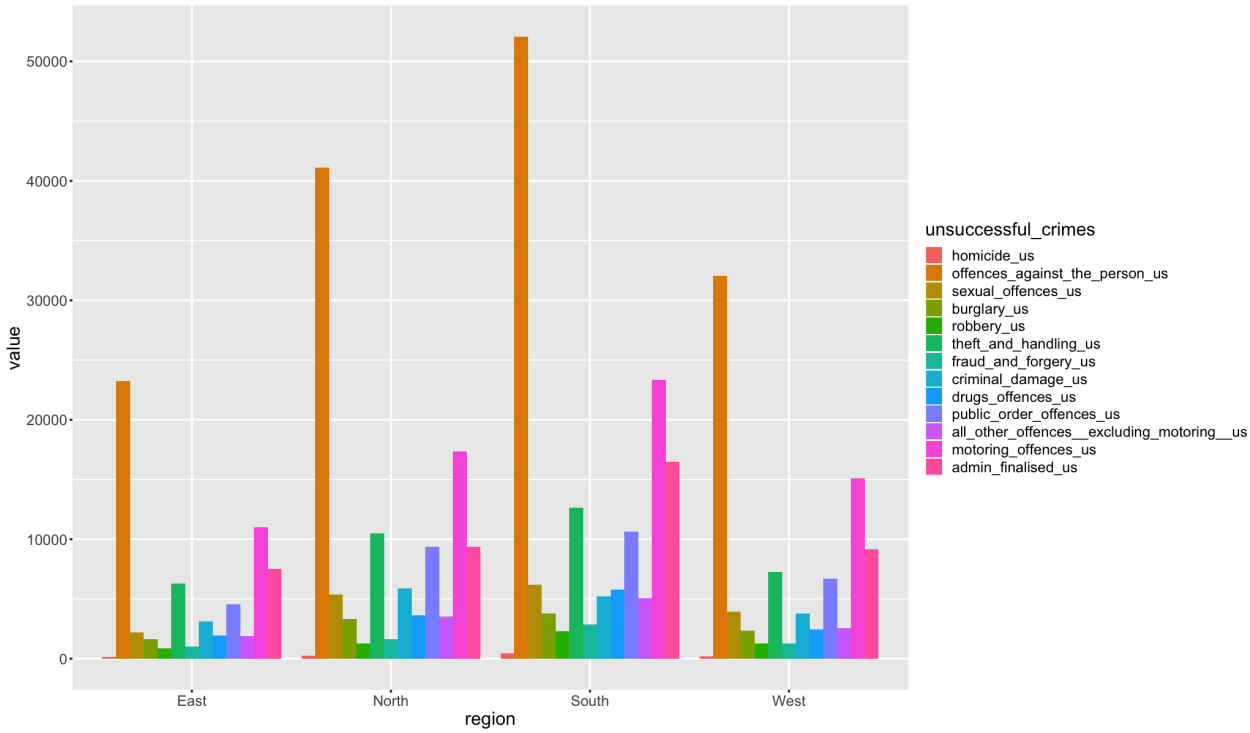
5.2.1.1. Crimes

The graph below considers the crime cases and shows that offenses below the person are the highest in every region of england especially North side. Robbery is the second highest and it maintains the trend across all regions of England and topping it in the North. The third highest is Motoring within all the regions and all other crimes follow in a similar way. Generally, North has the highest peaks of crimes and East can be observed to be the Lowest.



5.2.1.2. Unsuccessful Crimes

The graph below considers the unsuccessful cases and shows that unsuccessful offenses below the person are the highest in every region of England, especially the South side. Unsuccessful Motoring Offences is the second highest and it maintains the trend across all regions of England and topping it in the South. The third highest is Unsuccessful Admin Finalized within all the regions and all other crimes follow in a similar way. Generally, South has the highest peaks of unsuccessful cases and East can be observed to be the Lowest.



5.2.1.3. Code

This function takes a dataframe as input and groups it by the "region" column. It removes the first four columns, removes any rows where the "region" column is "All" and then groups the remaining data by the "region" column. Then it applies the sum function to all columns in the dataframe and return the summarized dataframe.

```
group_by_region <- function(dataframe){
  dataframe <- dataframe[,-c(1:4)]
  dataframe <- dataframe[dataframe$region != "All",]
  dataframe <- group_by(dataframe, region)
  summarise_all(dataframe, funs(sum))
}
```

5.2.2. Region & Sum of All

In this section, we consider and compare the total count of crimes and unsuccessful cases in the various of regions of England.

5.2.2.1. Crimes

There have been a total 2048554 crimes across all regions of England where North has the highest number, i.e. 653326 while East has the lowest, i.e. 342048. Whereas West and South are 2nd and 3rd with 572792 and 480388 crimes respectively.

```
1 group_by_region_all_crimes(crime_df)
```

A tibble: 5 × 2

region	sum_of_all
<chr>	<dbl>
All	2048554
East	342048
North	653326
South	572792
West	480388

5.2.2.2. Unsuccessful Crimes

There have been a total 412978 unsuccessful cases across all regions of England where South has the highest number, i.e. 146850 while East has the lowest, i.e. 65468. Whereas North and West are 2nd and 3rd with 112594 and 88066 unsuccessful cases respectively.

```
]: 1 group_by_region_all_crimes(uscrime_df)
```

A tibble: 5 × 2

region	sum_of_all
<chr>	<dbl>
All	412978
East	65468
North	112594
South	146850
West	88066

5.2.2.3. Code

This function takes a dataframe as input, removes the first four columns, creates a new column called "sum_of_all" that contains the sum of all numeric columns, keeps only the "region" and "sum_of_all" columns and then groups the data by the "region" column. It then applies the sum function to all columns in the dataframe and return the summarized dataframe.

```
group_by_region_all_crimes <- function(dataframe){
  dataframe <- dataframe[,-c(1:4)]
  dataframe$sum_of_all <- rowSums(dataframe[, sapply(dataframe,
is.numeric)])
  dataframe <- dataframe[, c("region", "sum_of_all")]
  dataframe <- group_by(dataframe, region)
  summarise_all(dataframe, funs(sum))
}
```

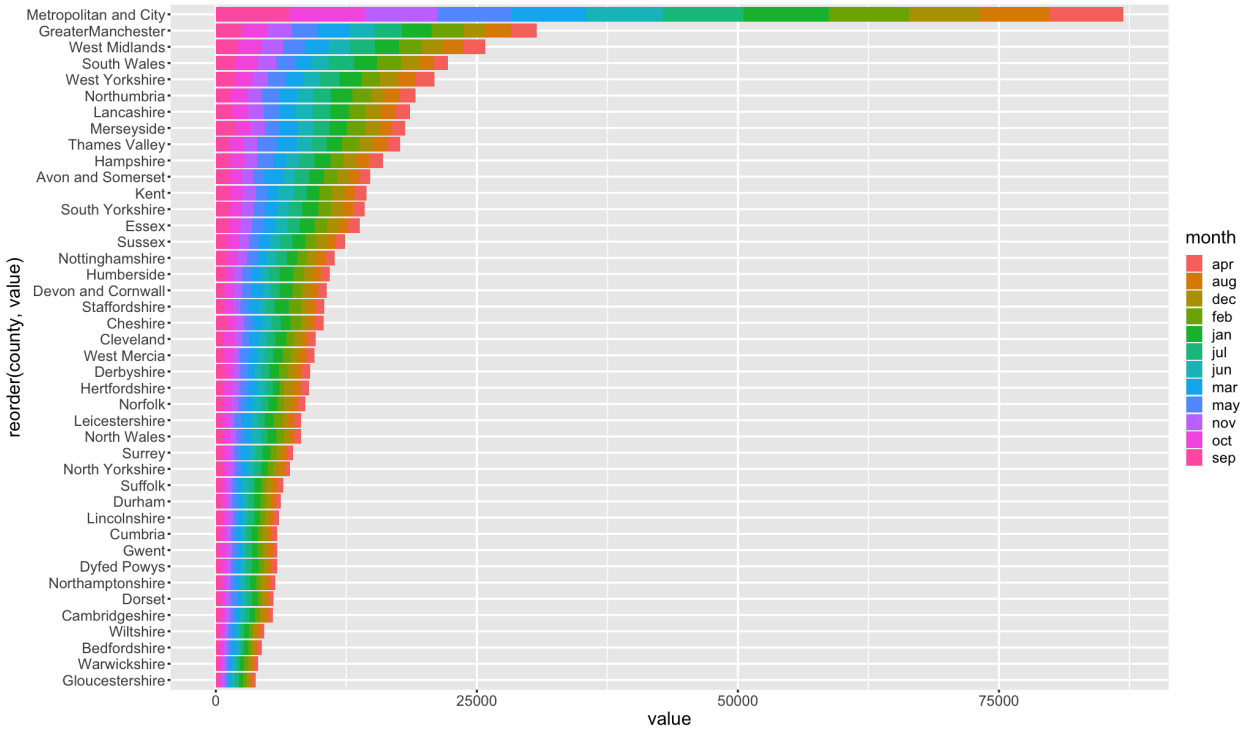
5.3. Analysis dependent on Years & Months

In this section of analysis, we try to understand the relations between successful and unsuccessful crime occurrences in various counties of England with respect to years and months using graph visualization.

5.3.1. 2014

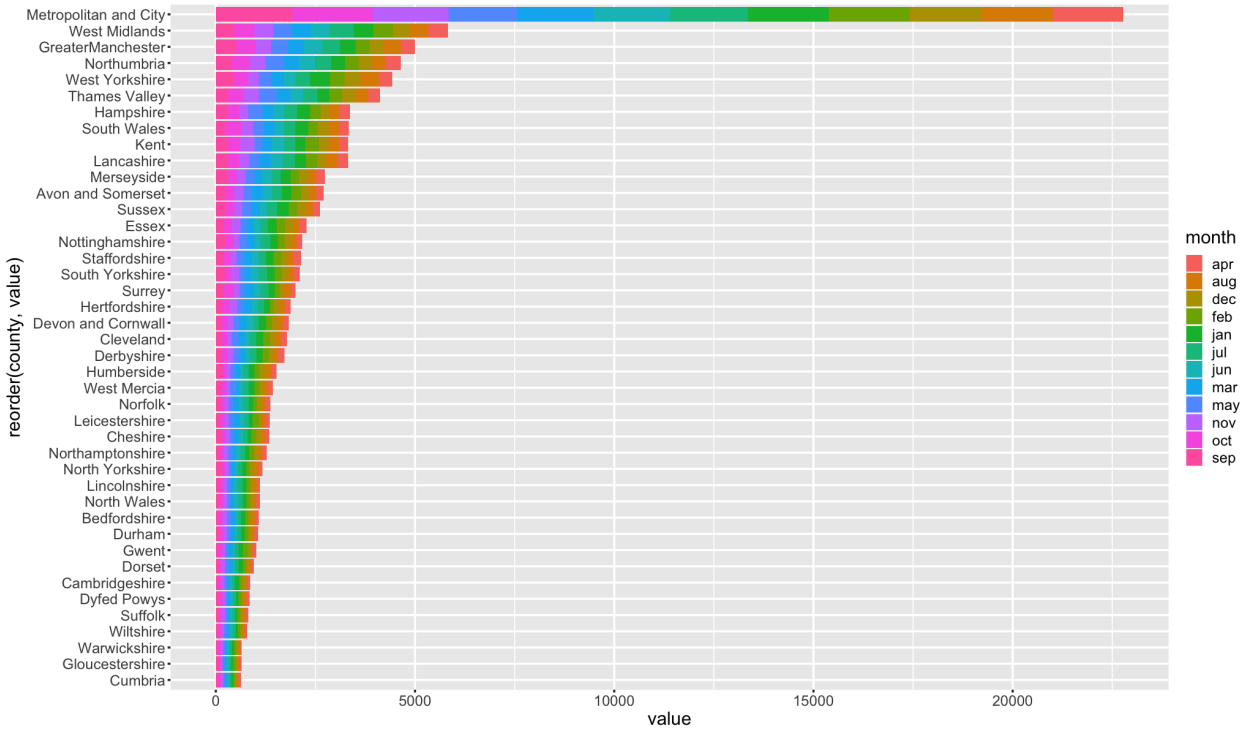
5.3.1.1. Crimes

The graph below shows that the Metropolitan and City had the 3X number of crimes as compared to all counties. Meanwhile, Gloucestershire, Warwickshire, Bedfordshire and Suffolk had the lowest crime rates. The crimes tend to increase in more dense population places.



5.3.1.2. Unsuccessful Crimes

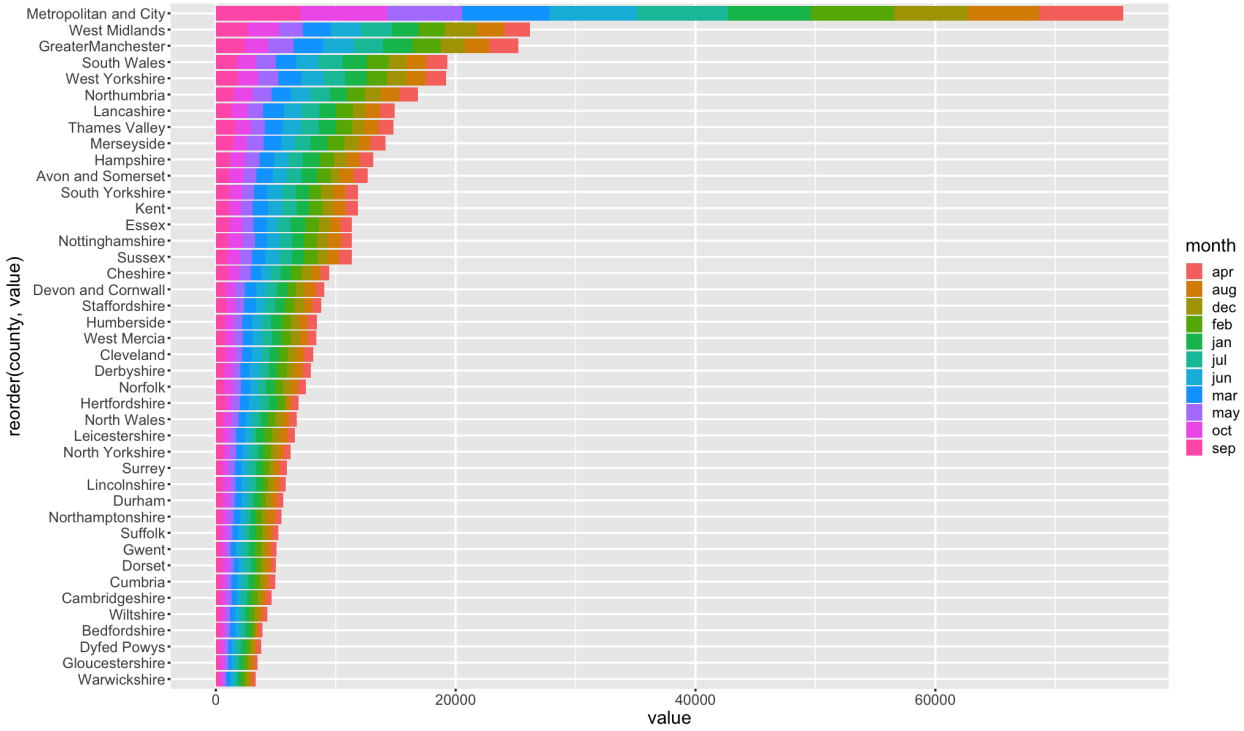
The graph below shows that the Metropolitan and City had the 3X number of unsuccessful crimes as compared to all counties. Meanwhile, Cumbrie, Gloucestershire, Warwickshire, Bedfordshire and Suffolk had the lowest unsuccessful crime rates. The unsuccessful crimes tend to increase in more dense population places.



5.3.2. 2015

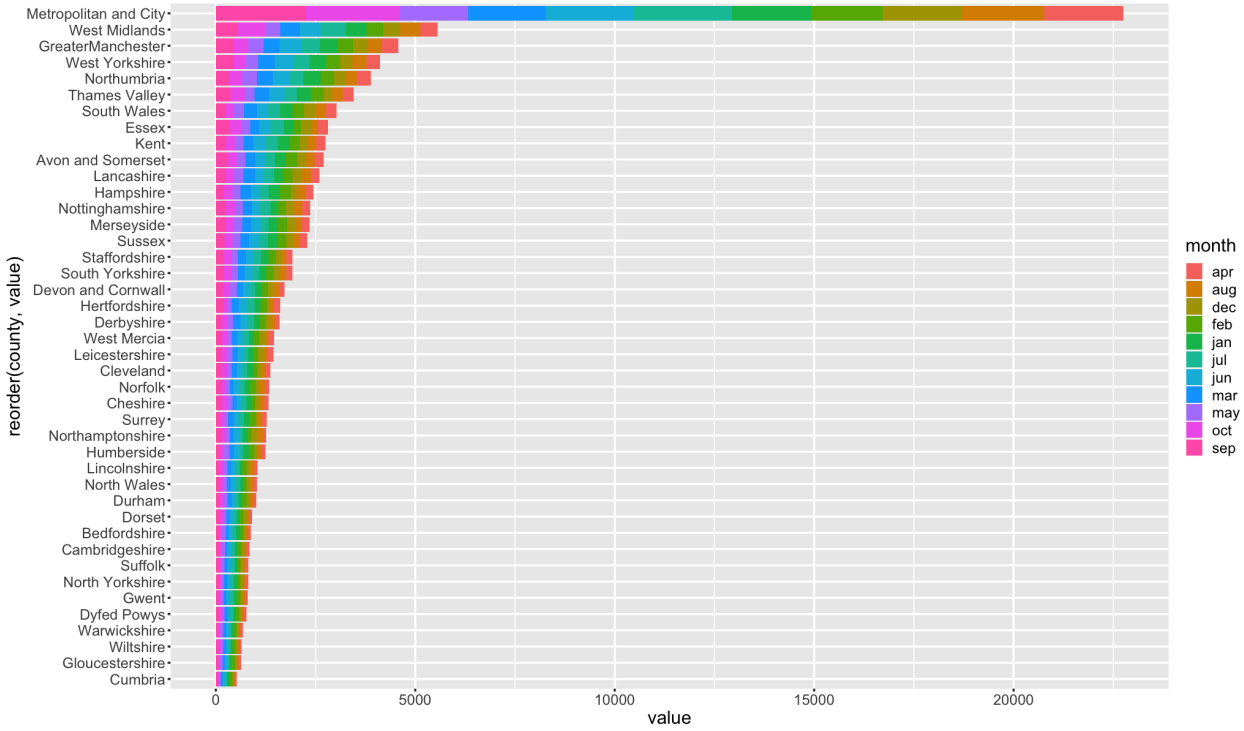
5.3.2.1. Crimes

It follows the trend from 2014 and shows that the Metropolitan and City had the 3X number of crimes as compared to all counties. Meanwhile, Warwickshire, Gloucestershire, Dylad Powys , Bedfordshire had the lowest crime rates. The crimes tend to increase in more dense population places.



5.3.2.2. Unsuccessful Crimes

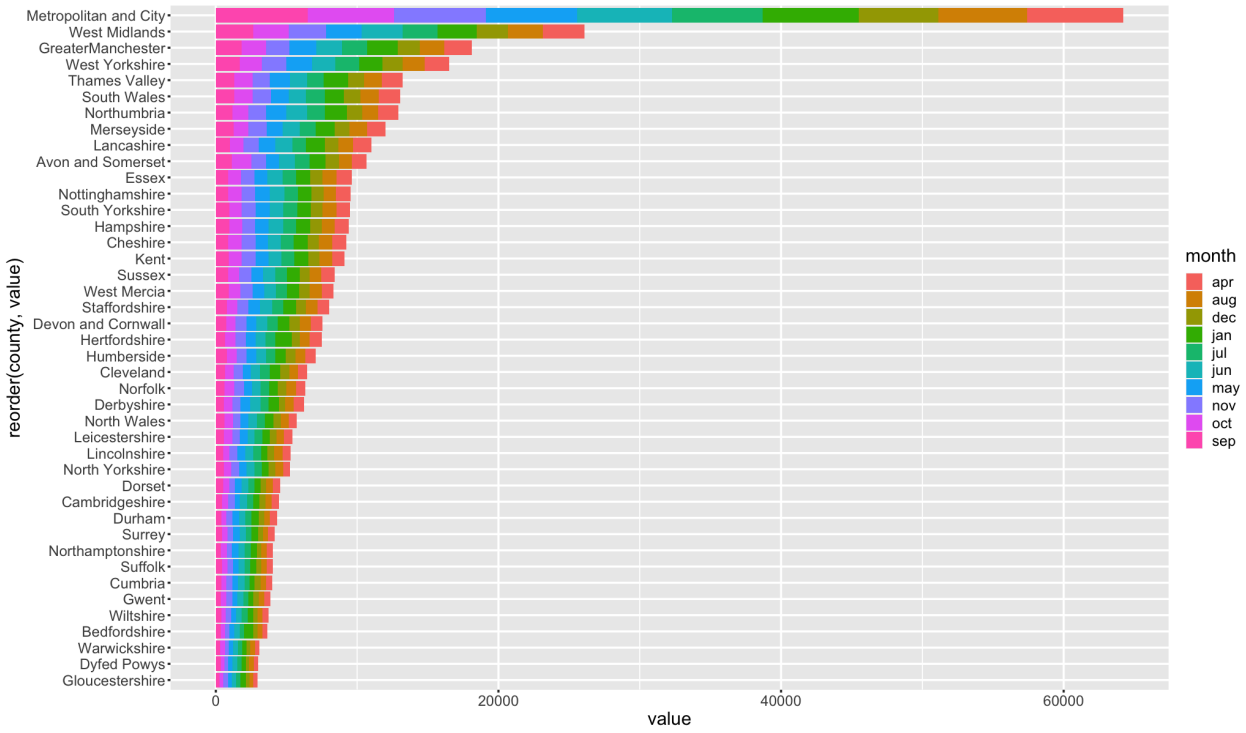
It follows the trend from 2014 and shows that the Metropolitan and City had the 3X number of unsuccessful crimes as compared to all counties. Meanwhile, Cumbrie, Gloucestershire, Warwickshire had the lowest unsuccessful crime rates. The unsuccessful crimes tend to increase in more dense population places.



5.3.3. 2016

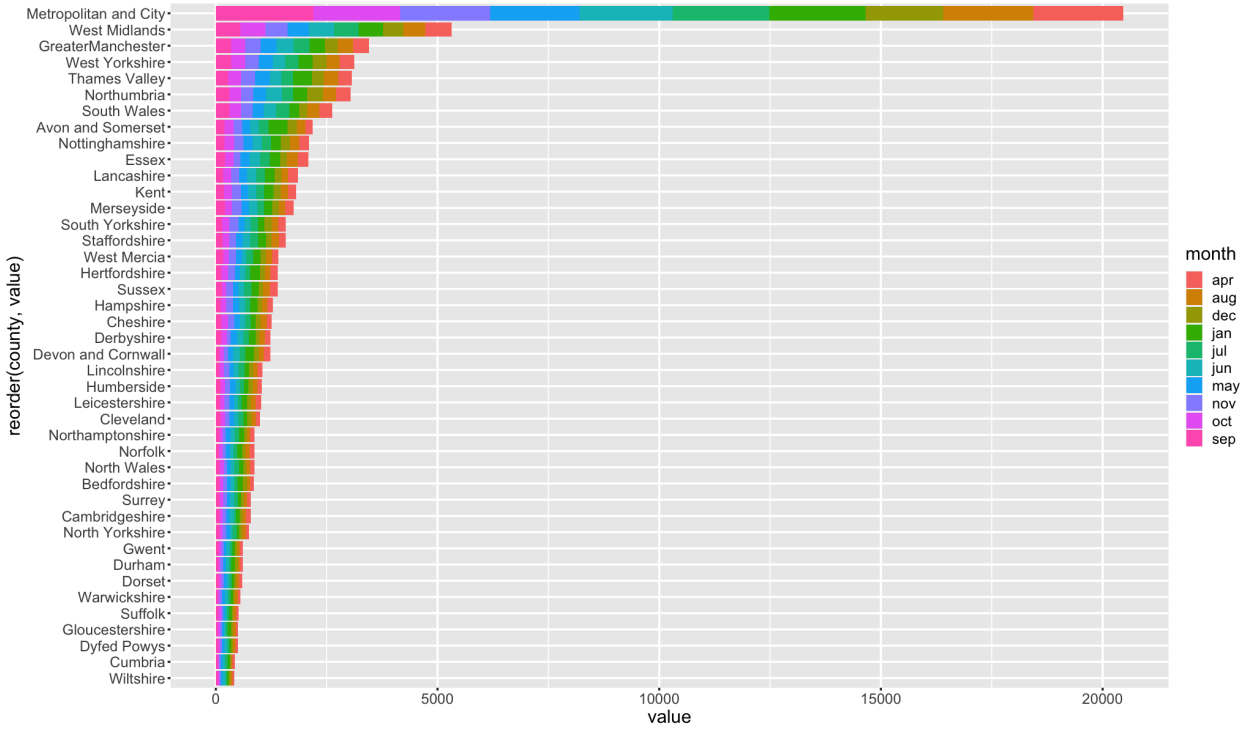
5.3.3.1. Crimes

It follows the trend from 2015 when it comes to the crime rates with the lower ones competing with each other rather than the ones at the top.



5.3.3.2. Unsuccessful Crimes

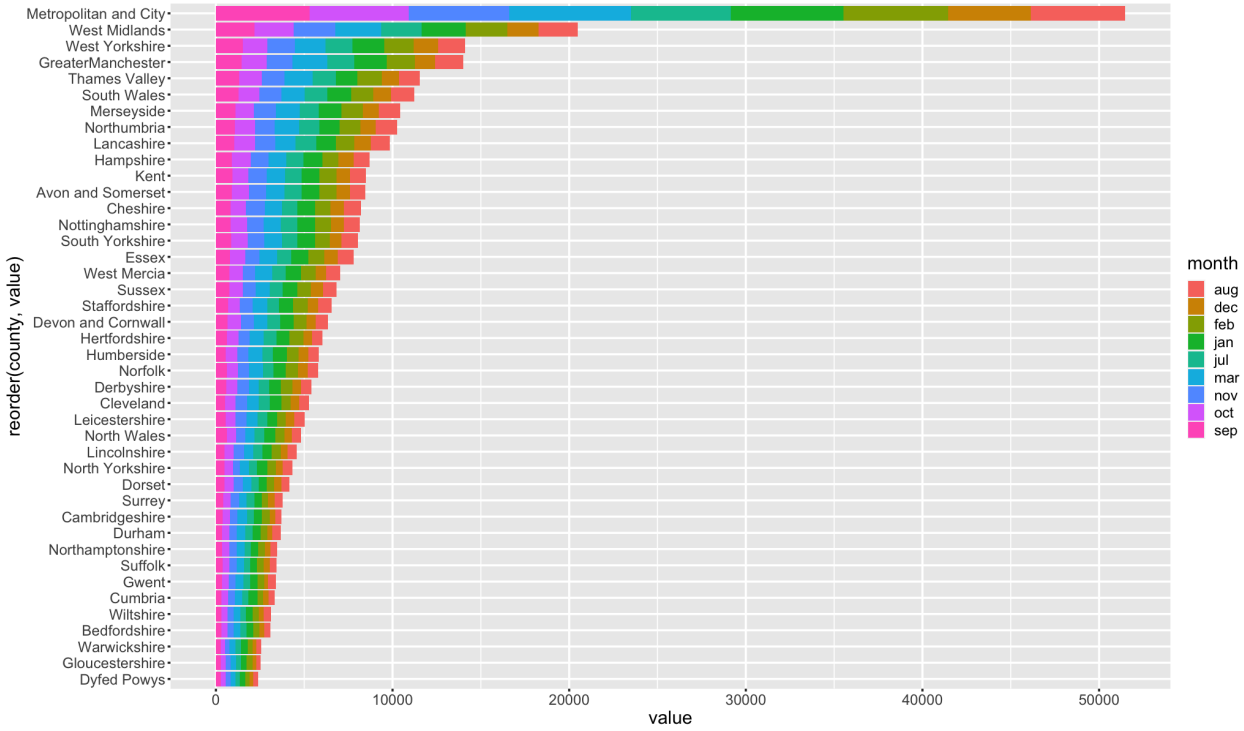
It follows the same trend as 2015 for unsuccessful crime rates with the lower ones competing with each other rather than the ones at the top.


Metropolitan and City
West Midlands
GreaterManchester
West Yorkshire
Thames Valley
Northumbria
South Wales
Avon and Somerset
Nottinghamshire
Essex
Lancashire
Kent
Merseyside
South Yorkshire
Staffordshire
West Mercia
Hertfordshire
Sussex
Hampshire
Cheshire
Derbyshire
Devon and Cornwall
Lincolnshire
Humberside
Leicestershire
Cleveland
Northamptonshire
Norfolk
North Wales
Bedfordshire
Surrey
Cambridgeshire
North Yorkshire
Gwent
Durham
Dorset
Warwickshire
Suffolk
Gloucestershire
Dyfed Powys
Cumbria
Wiltshire
reorder(county, value)
0
5000
10000
15000
20000
value
month
apr
aug
dec
jan
jul
jun
may
nov
oct
sep

5.3.4. 2017

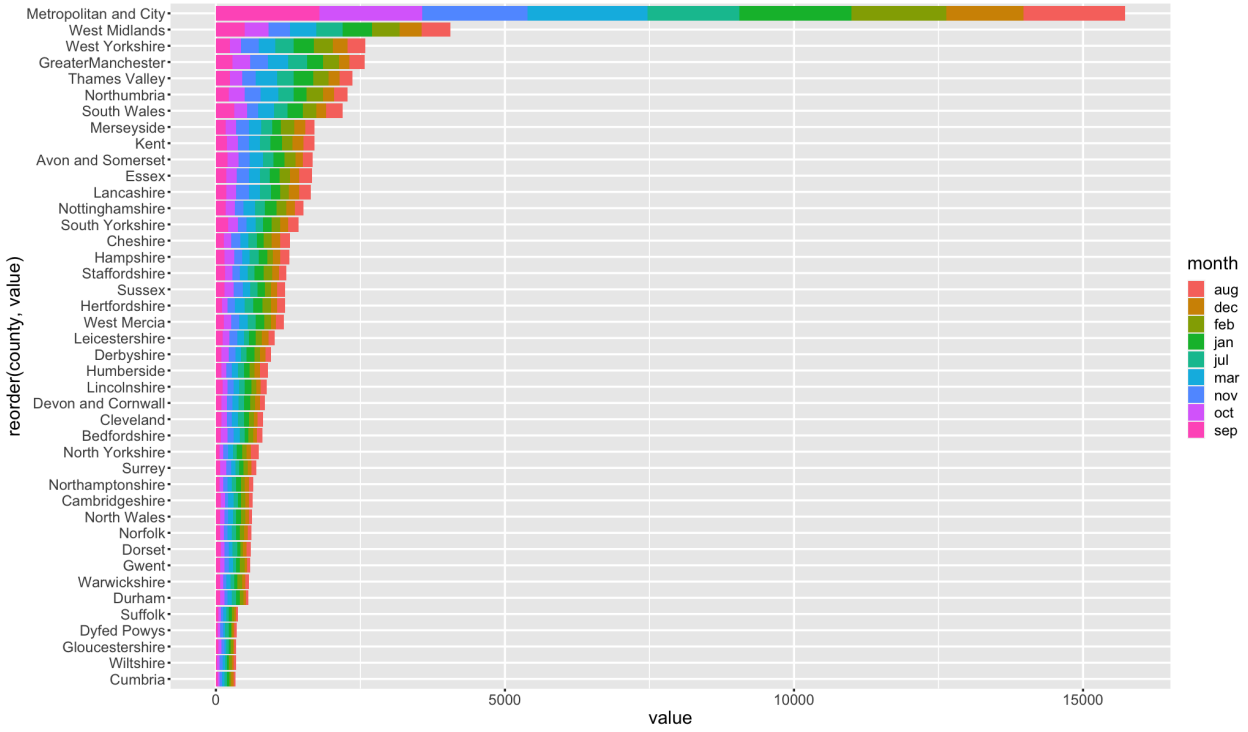
5.3.4.1. Crimes

It follows the trend from 2016 when it comes to the crime rates with the lower ones competing with each other rather than the ones at the top.



5.3.4.2. Unsuccessful Crimes

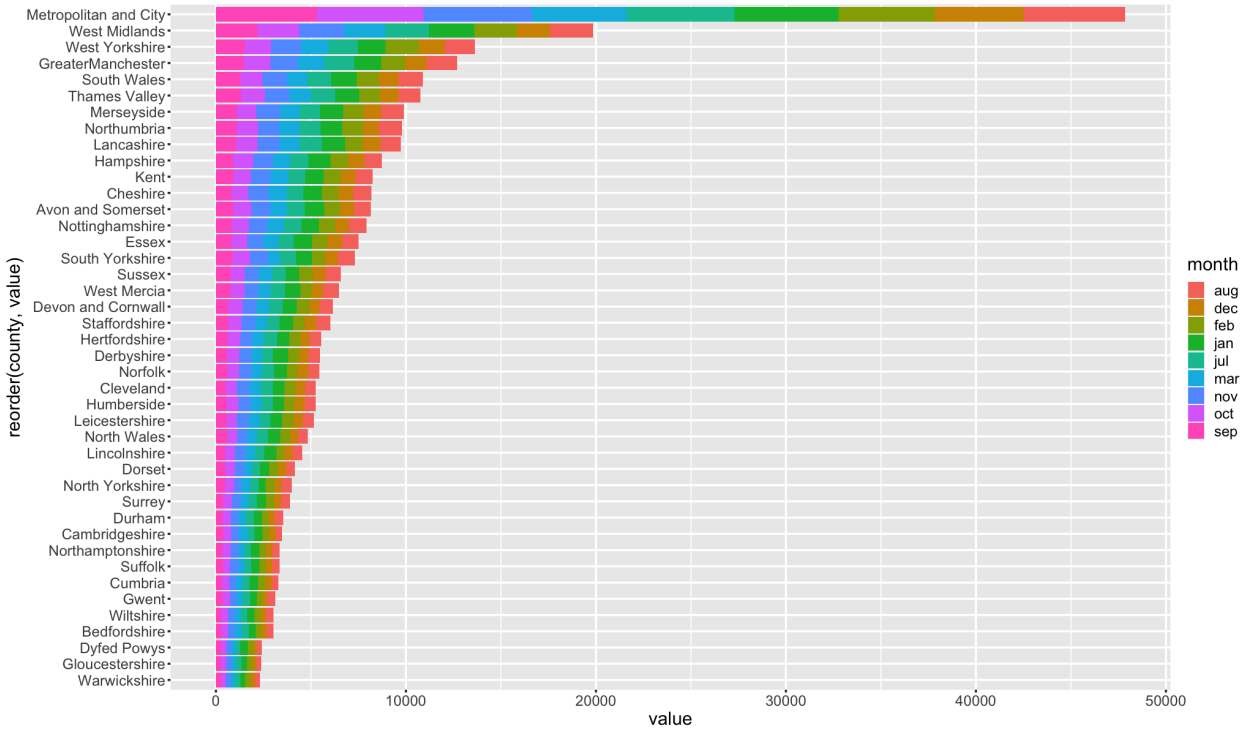
It follows the same trend as 2016 for unsuccessful crime rates with the lower ones competing with each other rather than the ones at the top.



5.3.5. 2018

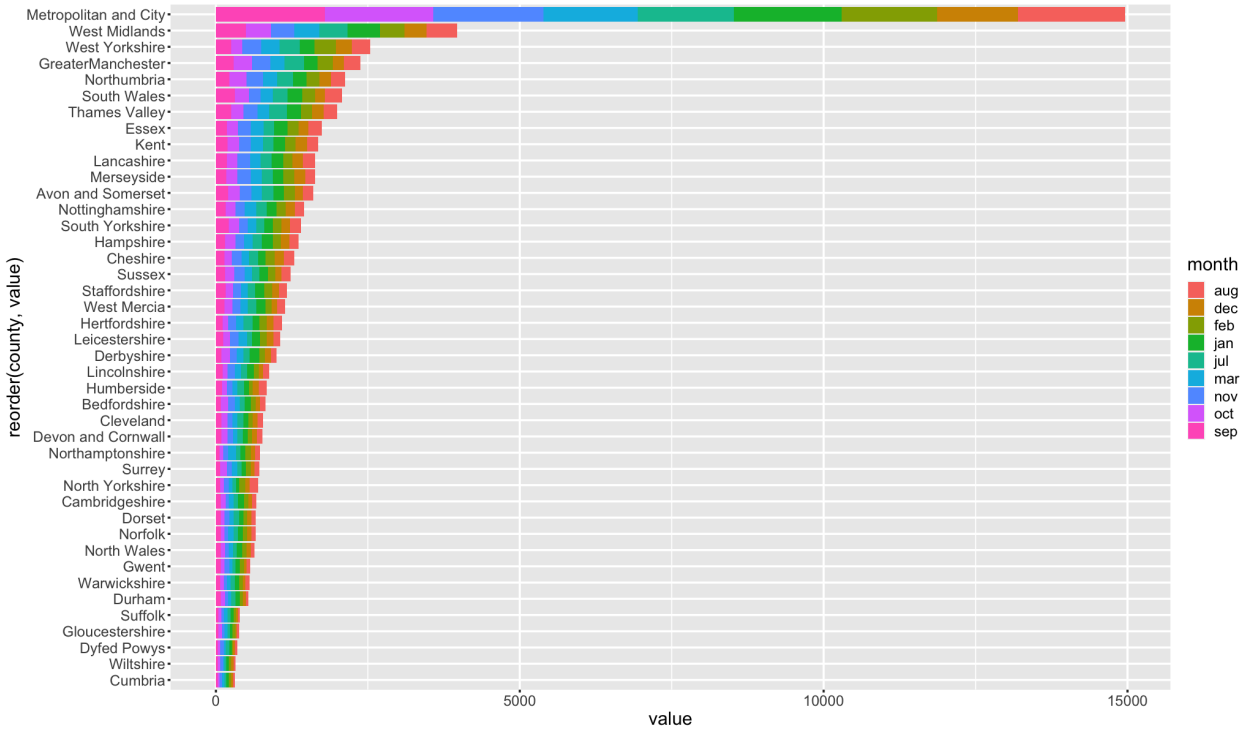
5.3.5.1. Crimes

It follows the trend from 2017 when it comes to the crime rates with the lower ones competing with each other rather than the ones at the top.



5.3.5.2. Unsuccessful Crimes

It follows the same trend as 2017 for unsuccessful crime rates with the lower ones competing with each other rather than the ones at the top.



5.3.6. Code

5.3.6.1. Group_by_year_month

This function takes a dataframe and year as inputs and groups the data by county and month. It filters the dataframe to only include rows where the year column matches the input year, removes any rows where the county column is "National", removes the year, yearmon and region columns, creates a new column called "sum_of_all" that contains the sum of all numeric columns, keeps only the "county", "month" and "sum_of_all"

columns, groups the data by the "county" and "month" columns and then applies the sum function to all columns in the dataframe and return the summarized dataframe.

```
group_by_year_month <- function(dataframe, year){
  dataframe <- dataframe[dataframe$year == year,]
  dataframe <- dataframe[dataframe$county != "National",]

  dataframe <- dplyr::select(dataframe, -c("year", "yearmon",
"region"))
  dataframe$sum_of_all <- rowSums(dataframe[, sapply(dataframe,
is.numeric)])

  dataframe <- dplyr::select(dataframe, c("county", "month",
"sum_of_all"))
  dataframe <- group_by(dataframe, county, month)
  summarise_all(dataframe, funs(sum))
}
```

5.3.6.2. Plot_graph

This function takes a dataframe and year as inputs and plots a stacked bar chart using ggplot2 library. It first applies the function group_by_year_month to the input dataframe and year to obtain a dataframe grouped by county and month. Then it uses the melt function to reshape the dataframe so that each county and month is represented in a single row with columns for the crimes and values. It then plots the data using ggplot2 library with the x-axis being the 'value' column, the y-axis being the 'county' column reordered by 'value' and the fill aesthetic being the 'month' column. Additionally, it changes the default plot size, text size, and line width.

```
plot_graph <- function(in_df, year){
  df <- melt(group_by_year_month(in_df, year) ,  id.vars =
c('county', 'month'), variable.name = 'crimes')
  options(repr.plot.width = 17, repr.plot.height =10)
  ggplot(df, aes(x = value, y = reorder(county, value))) +
  geom_bar(aes(fill = month), stat = "identity", position = "stack",
width = 0.9) +
  theme(text = element_text(size = 18), element_line(linewidth =1))
}
```

5.4. Analysis between Crime Types

In this section of analysis, we try to understand the correlation and covariance between successful and unsuccessful crime.

5.4.1. Correlation

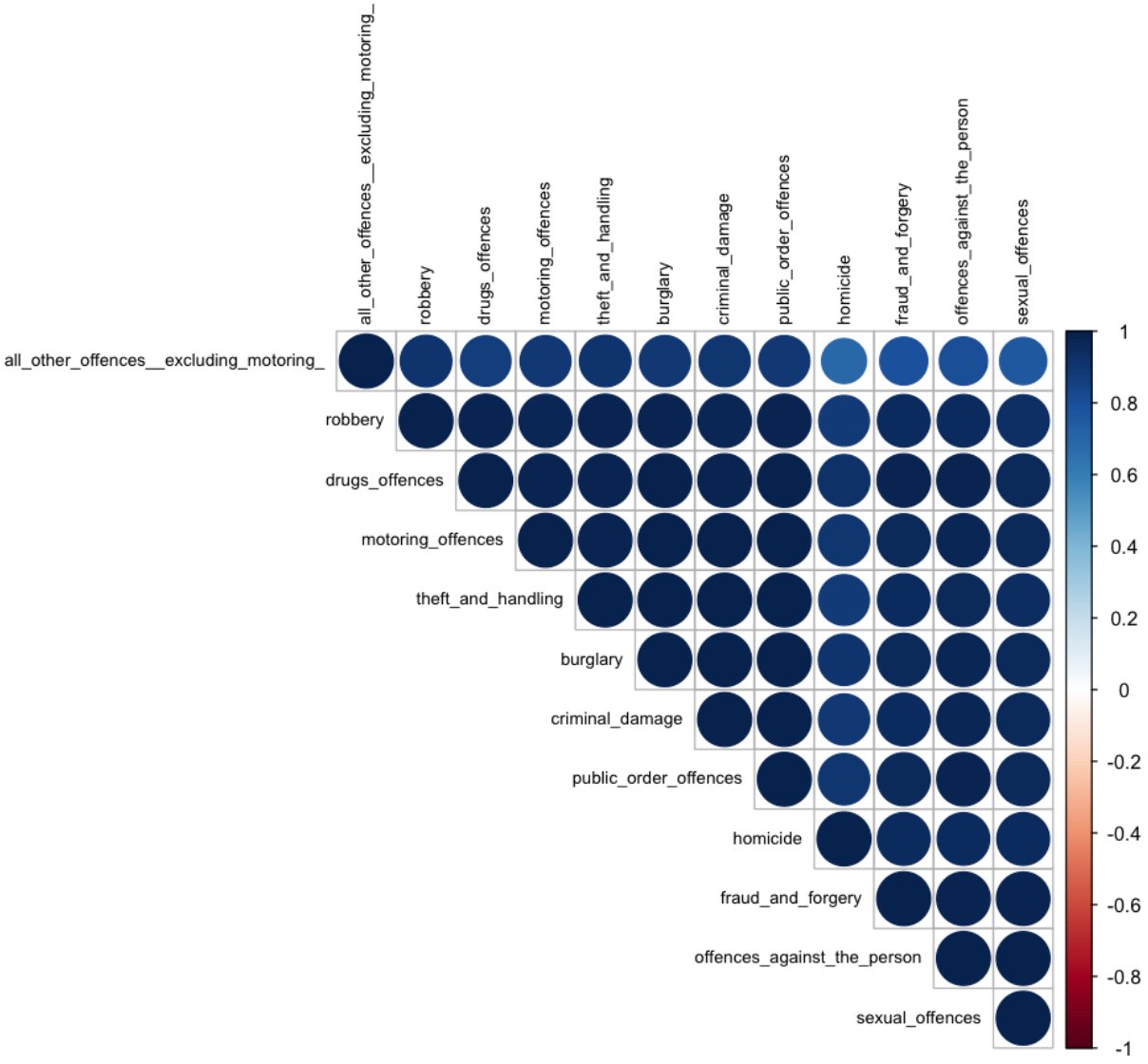
Correlation is a statistical measure that describes the relationship between two or more variables. It is a normalized version of covariance, which means that it scales the relationship between variables in terms of a value between -1 and 1. A value of 1 indicates a perfect positive correlation, meaning that as one variable increases, the other also increases, and vice versa. A value of -1 indicates a perfect negative correlation, meaning that as one variable increases, the other decreases, and vice versa. A value of 0 indicates no correlation, meaning that there is no relationship between the variables.

For example, in the context of crime, a positive correlation between the number of burglaries and the number of thefts would indicate that an increase in burglaries is associated with an increase in thefts, and vice versa. A negative correlation between the number of burglaries and the number of drug offenses, on the other hand, would indicate that an increase in burglaries is associated with a decrease in drug offenses, and vice versa.

Correlation is typically represented as a numerical value, such as a Pearson correlation coefficient, and can be used in conjunction with other statistical measures, such as regression analysis, to better understand the relationship between variables.

It's worth mentioning that correlation does not indicate causality, it just indicates that two variables are related and in which direction the relationship is.

5.4.1.1. Crimes



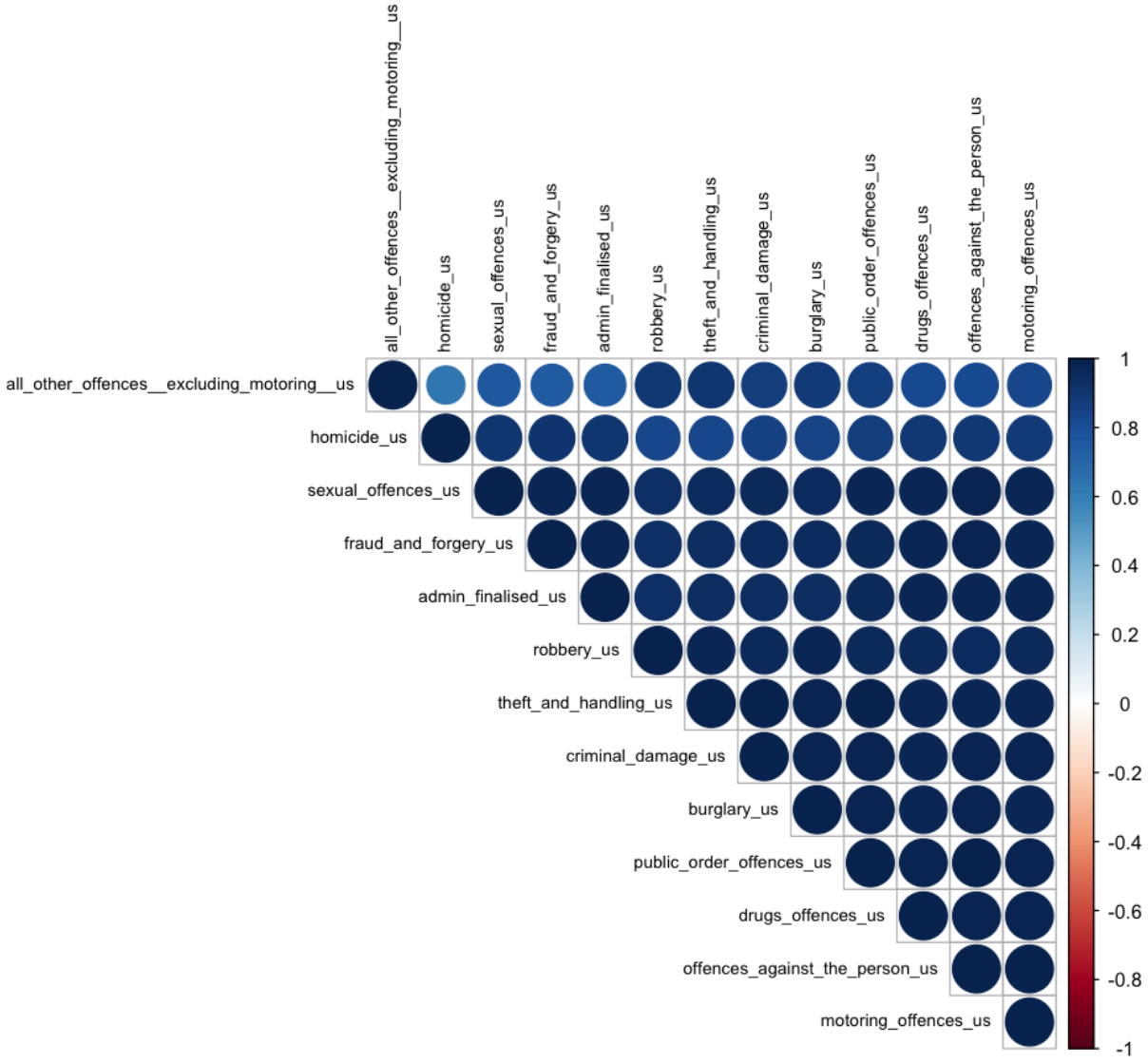
This is a correlation matrix, which shows the correlation coefficients between different types of crimes. The matrix shows the correlation between each type of crime with every other type of crime. Each cell in the matrix represents the correlation coefficient between the two types of crime corresponding to the row and column of that cell. The correlation coefficient ranges between -1 and 1, where a value of 1 indicates a perfect positive correlation, a value of -1 indicates a perfect negative correlation, and a value of 0 indicates no correlation.

The correlation matrix can be used to understand the relationships between different types of crime. For example, it can be seen that homicide has a high correlation with offences against the person (0.95), sexual offences (0.96), burglary (0.91) and robbery (0.89) which suggest that those crimes tend to happen together.

On the other hand, it can be seen that the lowest correlation is between motoring offences and other crimes. For example, the correlation between motoring offences and homicide is 0.91, between motoring offences and offences against the person is 0.98, and between motoring offences and sexual offences is 0.96. These values are lower than the correlation between other types of crimes, such as homicide, offences against the person and sexual offences, which have a correlation of 0.95, 0.99 and 0.99 respectively.

This suggests that motoring offences are not as closely related to other types of crime as other types of crime are related to each other. This might indicate that people who commit motoring offences are different from people who commit other types of crimes, and that different intervention and prevention strategies may be needed to address these types of crime.

5.4.1.2. Unsuccessful Crimes



The correlation matrix above represents the correlation between different types of unsuccessful crimes. The values in the matrix range from -1 to 1, where 1 represents a perfect positive correlation, -1 represents a perfect negative correlation, and 0 represents no correlation. The correlation coefficient of a pair of variables is a measure of the strength and direction of the linear relationship between the two variables.

From the correlation matrix, we can see that some unsuccessful crimes have a higher correlation than others. For example, unsuccessful homicide has a high positive correlation with unsuccessful offences against the person (0.89), unsuccessful sexual offences (0.90), and unsuccessful burglary (0.85). These crimes are likely to occur together or be related in some way. On the other hand, crimes such as unsuccessful homicide and unsuccessful motoring offences have a lower correlation (0.91) indicating that they are less likely to occur together.

Additionally, from the matrix we can observe that unsuccessful crimes that have low correlation with other types of unsuccessful crimes, for example homicide_us and motoring_offences_us have a correlation of 0.91 which is lower than other crimes and this indicates that these unsuccessful crimes might be less related or less likely to happen together.

5.4.1.3. Code

This function takes a dataframe as input and plots a correlation matrix using the corrplot library. It first selects only the numeric columns of the dataframe, then it calculates the correlation matrix using the cor() function. It then changes the default plot size, and it plots the correlation matrix using the corrplot() function. This function plots the correlation matrix and returns the correlation matrix as output. The function corrplot() uses the "upper" type, "hclust" order, and other parameters like cex and color to customize the appearance of the matrix.

```
corr_matrix_graph <- function(dataframe){
  dataframe <- dplyr::select(dataframe, -c("year", "yearmon",
"region"))
  num_cols <- sapply(dataframe, is.numeric)
  corr_matrix <- cor(dataframe[,num_cols])
  options(repr.plot.width = 8, repr.plot.height =8)
  corrplot(corr_matrix, type = "upper", order = "hclust", tl.cex =
0.7, tl.col = "black", is.corr = TRUE, mar = c(0, 0, 0, 0))
  return(corr_matrix)
}
```

5.4.2. Covariance

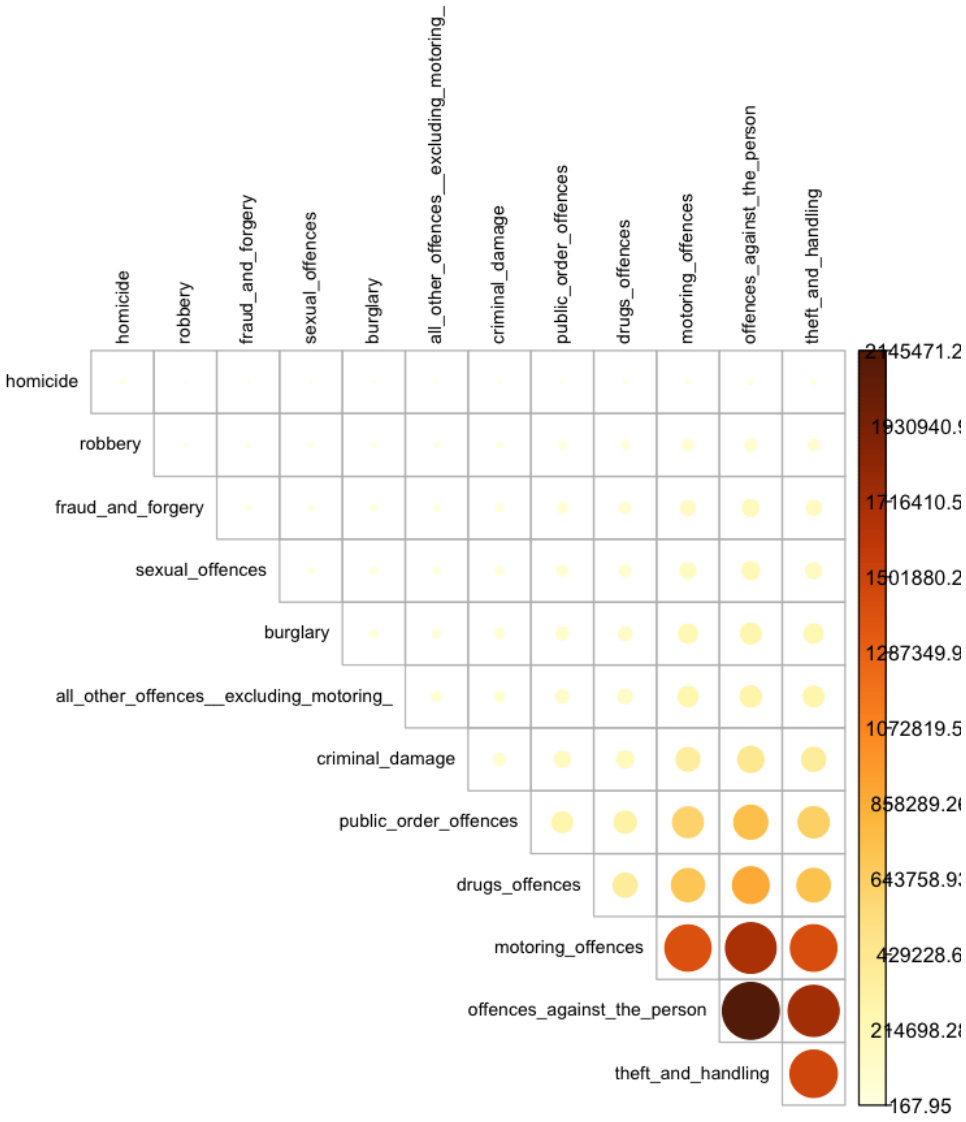
Covariance is a statistical measure that describes the relationship between two or more variables. It measures the degree to which two variables change together, and it can indicate whether the variables have a positive or negative relationship. A positive covariance means that the variables tend to increase or decrease together, while a negative covariance means that the variables tend to move in opposite directions.

For example, in the context of crime, a positive covariance between the number of burglaries and the number of thefts would indicate that an increase in burglaries is associated with an increase in thefts, and vice versa. A negative covariance between the number of burglaries and the number of drug offenses, on the other hand, would indicate that an increase in burglaries is associated with a decrease in drug offenses, and vice versa.

Covariance is typically represented as a numerical value, and it can be used in conjunction with correlation to better understand the relationship between variables. Correlation is a normalized version of covariance, allowing to compare the strength of the relationship independent of the scale of the variables.

It's worth mentioning that covariance alone does not indicate causality, it just indicates that two variables are related, but it doesn't tell us in which direction the relationship is.

5.4.2.1. Crimes

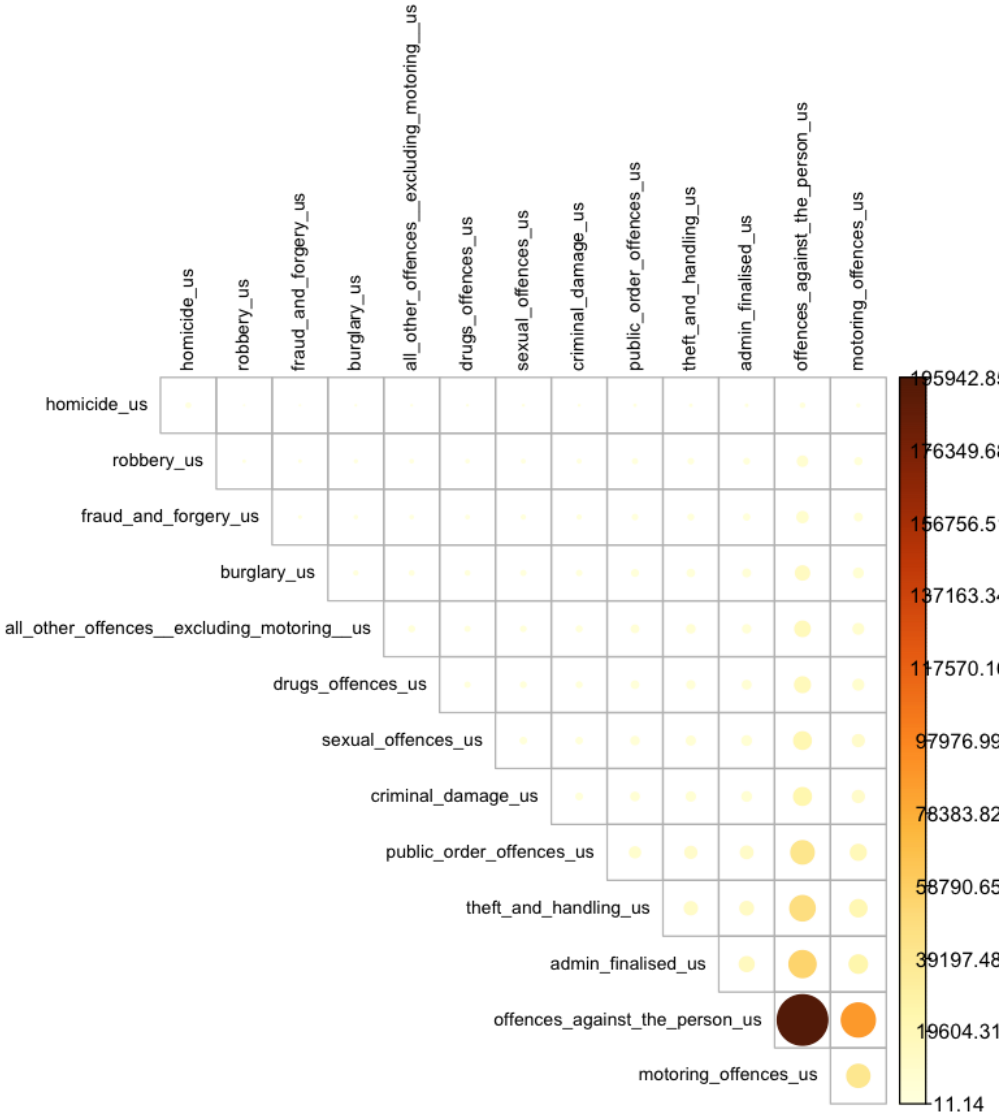


The covariance matrix for crimes is a matrix that shows the relationship between different types of crimes. Each cell in the matrix shows the covariance between two types of crimes. The covariance between two types of crimes is a measure of how much the two types of crimes vary together. A positive covariance indicates that the two types of crimes tend to increase or decrease together, while a negative covariance indicates that the two types of crimes tend to move in opposite directions. The larger the covariance, the stronger the relationship between the two types of crimes.

The top two covariance values in the matrix above are the values for "offences_against_the_person" and "homicide" (2145471.23) and "sexual_offences" and "offences_against_the_person" (206691.68). A high covariance value indicates that there is a strong linear relationship between the two variables. In this case, it suggests that there is a strong linear relationship between "offences_against_the_person" and "homicide" and between "sexual_offences" and "offences_against_the_person".

The lowest two covariance values in the matrix above are the values for "homicide" and "motoring_offences" (13975.66) and "all_other_offences__excluding_motoring_" and "motoring_offences" (263138.87). A low covariance value indicates that there is a weak linear relationship between the two variables. In this case, it suggests that there is a weak linear relationship between "homicide" and "motoring_offences" and between "all_other_offences__excluding_motoring_" and "motoring_offences".

5.4.2.2. Unsuccessful Crimes



The covariance matrix for unsuccessful crimes is a matrix that shows the relationship between different types of unsuccessful crimes. Each cell in the matrix shows the covariance between two types of crimes. The covariance between two types of crimes is a measure of how much the two types of crimes vary together. A positive covariance indicates that the two types of crimes tend to increase or decrease together, while a negative covariance indicates that the two types of crimes tend to move in opposite

directions. The larger the covariance, the stronger the relationship between the two types of crimes.

The highest covariance in the matrix for unsuccessful crimes is between offences against the person and sexual offences, with a covariance value of 195942.85. This suggests that there is a strong positive relationship between these two types of crime, such that when the rate of offences against the person increases, the rate of sexual offences also increases, and vice versa.

The second highest covariance is between offences against the person and burglary, with a covariance value of 14800.29. This indicates that there is also a positive relationship between these two types of crime, but not as strong as the relationship between offences against the person and sexual offences.

On the other hand, the lowest covariance in the matrix is between homicide and motoring offences, with a covariance value of 102.27. This suggests that there is a very weak relationship or no relationship between these two types of crime. Similarly, the second lowest covariance is between homicide and all other offences (excluding motoring) with a covariance value of 273.88, which also suggests a weak relationship between these two types of crime.

5.4.2.3. Code

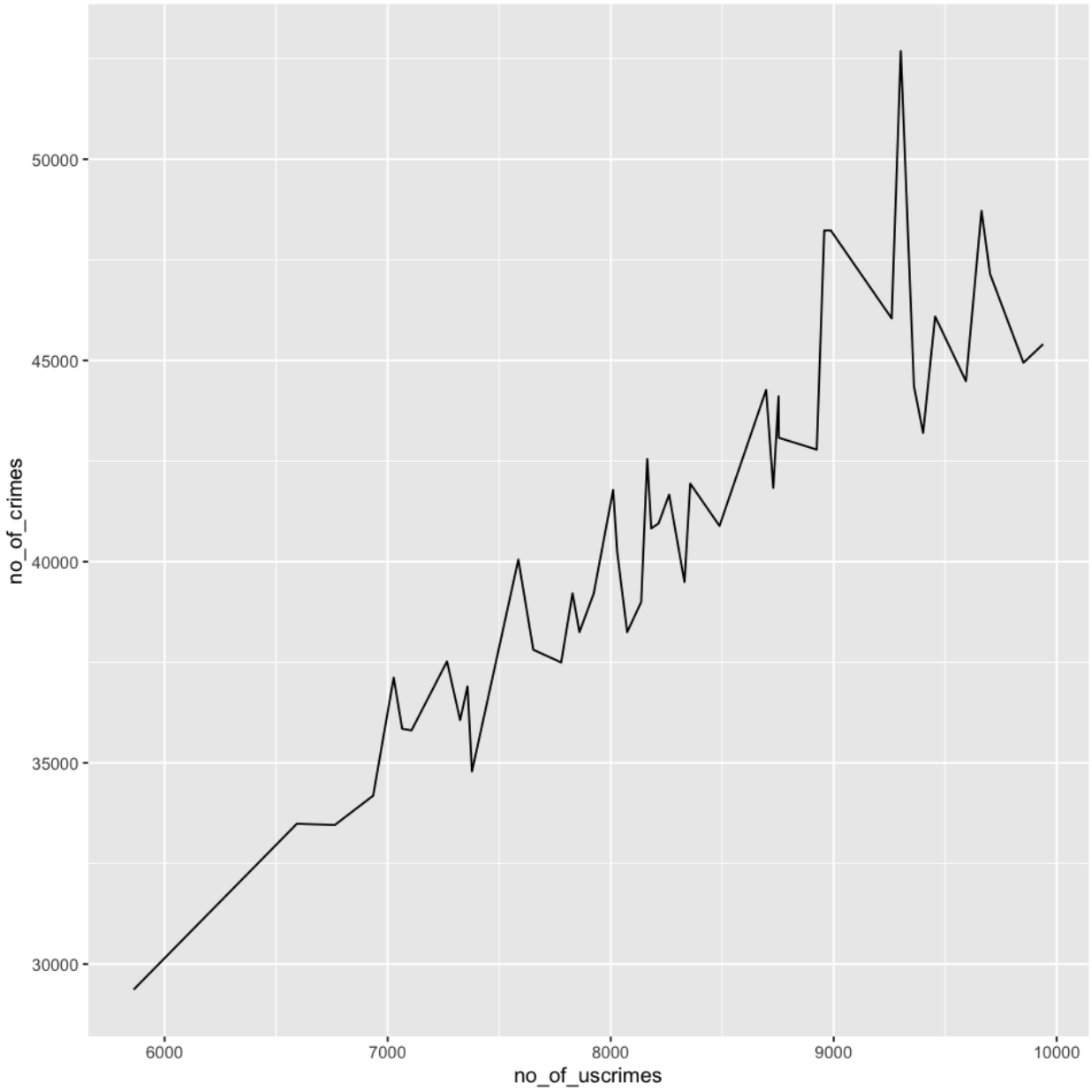
This function takes a dataframe as input and plots a covariance matrix using the corrplot library. It first selects only the numeric columns of the dataframe, then it calculates the covariance matrix using the cov() function. It then changes the default plot size, and it plots the covariance matrix using the corrplot() function with the is.corr parameter set to false. This function plots the covariance matrix and returns the covariance matrix as output. The function corrplot() uses the "upper" type, "hclust" order, and other parameters like cex and color to customize the appearance of the matrix. The difference with previous function is that it uses the cov() function to calculate the covariance matrix instead of correlation matrix which is the case in 'corr_matrix_graph' function.

```
cov_matrix_graph <- function(dataframe){
  dataframe <- dplyr::select(dataframe, -c("year", "yearmon",
"region"))
  num_cols <- sapply(dataframe, is.numeric)
  num_cols <- names(num_cols[num_cols])
  cov_matrix <- cov(dataframe[,num_cols])
  options(repr.plot.width = 8, repr.plot.height =8)
  corrplot(cov_matrix, type = "upper", order = "hclust", tl.cex =
0.7, tl.col = "black", is.corr = FALSE, mar = c(0, 0, 0, 0))
  return(cov_matrix)
}
```

5.5. Trend Analysis for Successful & Unsuccessful Crimes

In this section of analysis, we try to understand the trend between successful and unsuccessful crime.

5.5.1. Visualization



The visualization above of successful and unsuccessful crimes show that the number of cases for un-successful crimes increase with the increase in number of crimes.

The pros of crime cases and unsuccessful crime cases increasing together could include the following:

- It may indicate that law enforcement efforts are effective in detecting and addressing criminal activity.

- It may also suggest that the criminal justice system is working well in identifying and pursuing cases that are unlikely to result in successful convictions.

However, there are also several cons to consider:

- An increase in unsuccessful crime cases could indicate a lack of resources or a backlog in the criminal justice system, potentially leading to delays and inefficiencies in the process.
- It may also suggest that the police is not able to effectively investigate and prosecute crimes.
- An increase in crime and unsuccessful crime cases together may also lead to a strain on the criminal justice system, and could lead to a decrease in public trust and confidence in the system.
- It could also lead to the overburdening of the criminal justice system and lead to less serious crimes being overlooked.
- It may also suggest that the justice system is not effectively identifying and addressing the root causes of crime, which could lead to continued high levels of crime in the long-term.

5.5.2. Code

This function takes two dataframes (crime and uscrime) as input and combines them into a single dataframe. It first filters the crime and uscrime dataframes to only include rows where the county column is "National". Then it creates new columns in each dataframe that contain the sum of all numeric columns. It then removes unnecessary columns and merge both dataframes on the yearmon column. Finally, it returns the merged dataframe.

```r
group_and_combine_dfs <- function(crime, uscrime) {
  crime <- crime[crime$county == "National", ]
  uscrime <- uscrime[uscrime$county == "National", ]

  numeric_columns_crime <- dplyr::select(crime, -c("county", "year",
"month" ,"yearmon", "region"))
  crime$no_of_crimes <- rowSums(numeric_columns_crime)

  numeric_columns_uscrime <- dplyr::select(uscrime, -c("county",
"year", "month" ,"yearmon", "region"))
  uscrime$no_of_uscrimes <- rowSums(numeric_columns_uscrime)

  crime <- dplyr::select(crime, c("yearmon", "no_of_crimes"))
  uscrime <- dplyr::select(uscrime, c("yearmon", "no_of_uscrimes"))

  merged_df <- merge(crime, uscrime, by.x = "yearmon", by.y =
"yearmon")

  return(merged_df)
}
```

6. Predictive Analytics

Predictive analysis is a method of using data, statistical algorithms and machine learning techniques to identify the likelihood of future outcomes based on historical data. The goal of predictive analysis is to understand the data and to develop models that can make predictions or identify patterns and relationships that can be used to make decisions (“Predictive analytics”, n.d.). We employ the listed predictive techniques:

- Regression
- Clustering
- Classification

6.1. Regression

Regression is a statistical method used to analyze the relationship between a dependent variable (also known as the response variable or outcome variable) and one or more independent variables (also known as predictor variables or explanatory variables). The goal of regression is to find the best-fitting line (or curve) that describes the relationship between the dependent variable and the independent variables. This line (or curve) is known as the regression line or model (Sykes 1993, #).

There are several different types of regression, including linear regression, multiple regression, logistic regression, and polynomial regression. We would be employing Linear & Multiple Regression.

6.1.1. Linear Regression

Linear regression is a statistical method that is used to model the relationship between a dependent variable (also known as the outcome or response variable) and one or more independent variables (also known as explanatory or predictor variables). It assumes that there is a linear relationship between the independent variables and the dependent variable, and tries to find the best fitting line through the data points.

6.1.1.1. Hypothesis

To be able to predict the number of crimes occurring based on only date as an independent variable.

6.1.1.2. Dataset Summary

We would be going through the train and test dataset summaries for training the model and analyzing the results respectively. This is a summary of a datasets that contains two variables: yearmon and crimes. The variable "yearmon" is the date in the format of year-month, and the variable "crimes" is the number of crimes recorded for each date.

6.1.1.2.1. Train

We can see that the minimum date is January 2014, and the minimum number of crimes recorded is 29,367. The median date is September 2015, and the median number of crimes recorded is 41,308. The maximum date is December 2017, and the maximum number of crimes recorded is 52,683.

```
# Train Dataset
summary(dplyr::select(crime_test_train[[2]], c("yearmon", "crimes")))
```

```
   yearmon               crimes
Min.   :2014-01-01   Min.   :29367
1st Qu.:2014-11-08   1st Qu.:38249
Median :2015-09-16   Median :41308
Mean   :2015-11-08   Mean   :41317
3rd Qu.:2016-10-24   3rd Qu.:44336
Max.   :2017-12-01   Max.   :52683
```

6.1.1.2.2. Test

We can see that the minimum date is January 2018, and the minimum number of crimes recorded is 29,367. The median date is August 2018, and the median number of crimes recorded is 35,808. The maximum date is December 2018, and the maximum number of crimes recorded is 37,521.

```r
# Test Dataset
summary(dplyr::select(crime_test_train[[1]], c("yearmon", "crimes")))
```

```
   yearmon              crimes
Min.   :2018-01-01   Min.   :29367
1st Qu.:2018-03-01   1st Qu.:33488
Median :2018-08-01   Median :35808
Mean   :2018-07-01   Mean   :34804
3rd Qu.:2018-10-01   3rd Qu.:36066
Max.   :2018-12-01   Max.   :37521
```

6.1.1.3. Model Summary

```r
summary(lr_model)
```

```
Call:
lm(formula = crimes ~ yearmon, data = train)

Residuals:
    Min      1Q  Median      3Q     Max
-5273.2 -1805.3   297.9  1929.6  5366.7

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.898e+05  1.530e+04  12.401 2.76e-15 ***
yearmon     -8.864e+00  9.135e-01  -9.704 4.56e-12 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2537 on 40 degrees of freedom
Multiple R-squared:  0.7019,    Adjusted R-squared:  0.6944
F-statistic: 94.17 on 1 and 40 DF,  p-value: 4.557e-12
```

This is the summary of a linear regression model that is used to predict the number of crimes (dependent variable) based on the year and month (independent variable). The summary includes information about the coefficients of the model, the significance of the model, and the goodness of fit.

The coefficients table shows the estimates of the model's parameters. The first coefficient is the y-intercept (i.e. the predicted value of the dependent variable when the

independent variable is zero) and is 189800. The second coefficient is the slope of the line, which is -8.864.

The t-value and p-value in the coefficients table indicate the significance of the coefficients. A small p-value (in this case p-value less than .05) indicates that the coefficient is statistically significant, and the t-value is used to test the null hypothesis of the coefficient being zero. A t-value greater than 2 in absolute value usually indicates that the coefficient is statistically significant.

The Residuals section shows the distribution of the residuals, which are the differences between the observed values of the dependent variable and the fitted (predicted) values. The Min, 1Q, Median, 3Q, and Max values give an idea of the range of residuals and whether they are symmetrically distributed.

The Residual standard error, R-squared, and the F-statistic are all measures of the goodness of fit. The residual standard error is a measure of the average deviation of residuals from zero, R-squared is a measure of the proportion of variation in the dependent variable that is explained by the independent variable, and the F-statistic is a measure of the overall significance of the model.

In this case, the R-squared of 0.7019 and the low p-value of the F-statistic (4.557e-12) indicates that the model has a good fit and it is able to explain 70.19% of the variance in the number of crimes.

6.1.1.4. Accuracy Matrix

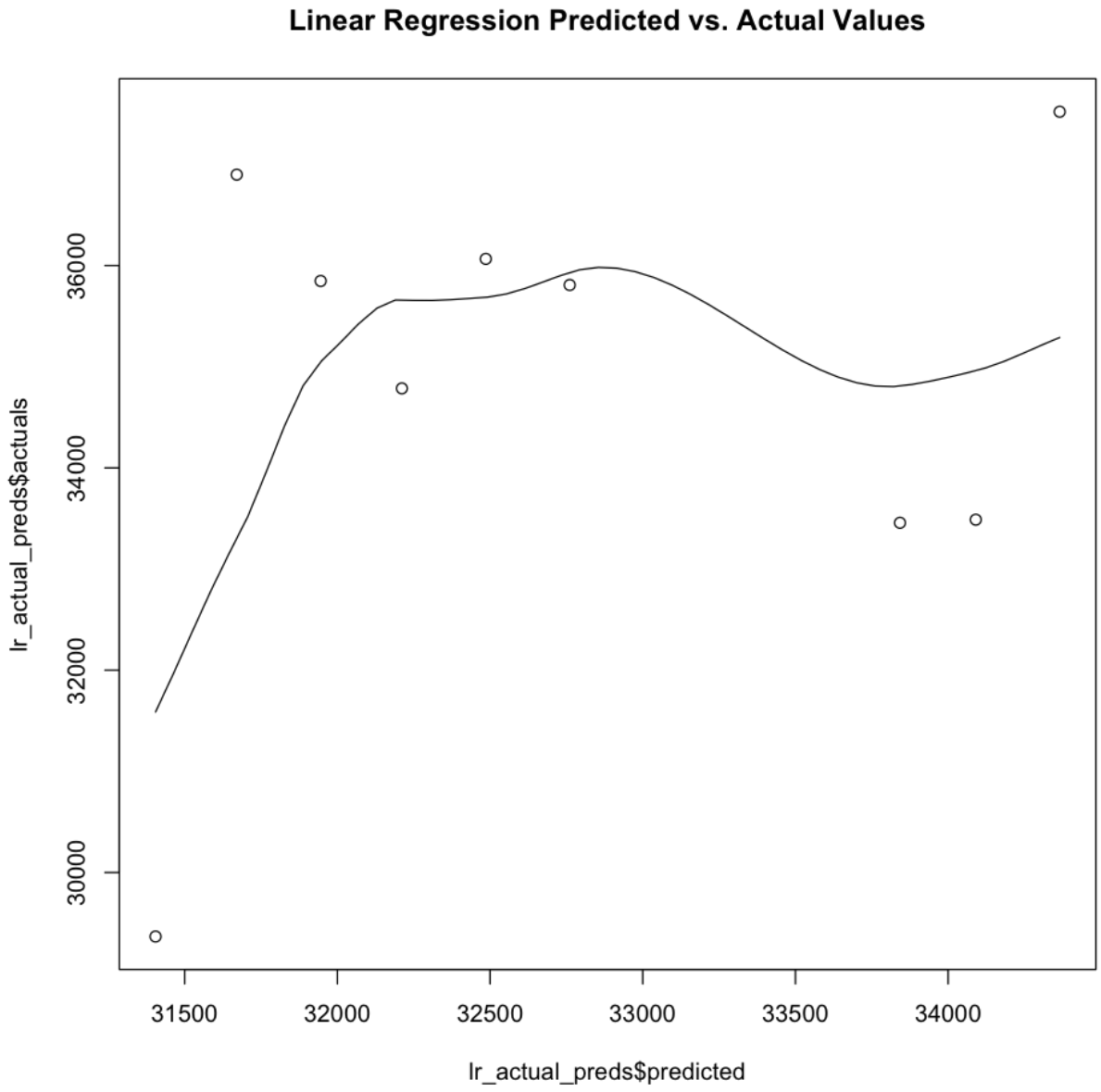
```
1 lr_accuracy
```

A matrix: 2 × 2 of type dbl

	actuals	predicted
actuals	1.000000	0.234283
predicted	0.234283	1.000000

The accuracy matrix is showing the correlation between the actual values and the predicted values. The diagonal values (1.000000 and 1.000000) indicate that there is a perfect correlation between the actual and predicted values for the same class. The other value (0.234283) shows the correlation between the actual and predicted values for different classes. This correlation is relatively low, indicating that there is not a strong association between the actual and predicted values when they are different. This suggests that the model may not have a high level of accuracy in predicting the outcomes.

6.1.1.5. Predicted vs Actual Plot



6.1.1.6. Evaluating Stats

These values represent the difference between the predicted value and the actual value, with the MAE being the average of the absolute errors, the MSE being the average of the squared errors, and the RMSE being the square root of the MSE.

MAE	2723.76659024003

MSE	9546383.52268908
RMSE	3089.72224037843

6.1.1.7. Code

6.1.1.7.1. Split Data

This function takes a dataframe as input and separates it into two dataframes, one for testing and one for training. It first filters the dataframe to include only rows where the county column is "National", it converts yearmon column to date format and creates a new column called 'crimes' which is the sum of all numeric columns. Then it separates the dataframe into two dataframes based on the year column, where one dataframe includes only rows with year equal to 2018 which is used for testing and the other one includes all other years which is used for training. Finally, it removes unnecessary columns from both dataframes and returns a list containing test and train dataframe.

```
lr_split_data <- function(dataframe){
  dataframe <- dataframe[dataframe$county == "National", ]
  dataframe$yearmon <- as.Date(dataframe$yearmon)

  numeric_columns_df <- dplyr::select(dataframe, -c("county", "year",
"month" ,"yearmon", "region"))
  dataframe$crimes <- rowSums(numeric_columns_df)

  test <- dataframe[dataframe$year == 2018,]
  train <- dataframe[dataframe$year != 2018,]

  test <- dplyr::select(test, -c("county", "year", "month",
"region"))
  train <- dplyr::select(train, -c("county", "year", "month",
"region"))

  return(list(test, train))
}
```

6.1.1.7.2. Linear Regression

This function takes two dataframes as inputs (test and train) and uses them to perform a linear regression. It first fits a linear model using the lm() function with 'yearmon' as predictor and 'crimes' as response variable. Then, it uses the model to make predictions on the test data using the predict() function. It creates a new dataframe containing the actual values from the test data and the predicted values. Finally, it calculates the correlation between actual and predicted values using the cor() function and returns a list containing the model, the dataframe of actual and predicted values, and the correlation value.

```
linear_regression <- function(test, train){
  # training the model
  model <- lm(crimes ~ yearmon, data = train)

  # using the model for predictions on test data
  preds <- predict(model, test, type = "response")
  actuals_preds <- data.frame(cbind(actuals=test$crimes,
predicted=preds))

  # calcultaing correaltion
  correlation_accuracy <- cor(actuals_preds)
  return(list(model, actuals_preds, correlation_accuracy))
}
```

6.1.2. Multiple Regression

Multiple regression is a statistical technique that uses multiple independent variables to predict a single dependent variable. It is an extension of simple linear regression, which uses only one independent variable to predict a dependent variable. In multiple regression, the relationship between the independent variables and the dependent variable is represented by an equation with one dependent variable and multiple independent variables. The equation is used to predict the value of the dependent variable based on the values of the independent variables.

Multiple regression is useful for analyzing the relationship between multiple independent variables and a single dependent variable, and for understanding how changes in the independent variables affect the dependent variable. It is widely used in fields such as economics, psychology, and marketing to understand complex relationships between variables.

6.1.2.1. Hypothesis

To be able to predict the number of crimes occurring based on county, year, month as an independent variable.

6.1.2.2. Dataset Summary

This is a summary of the data in a dataframe, specifically the dataframe has 5 columns: 'county', 'year', 'month', 'crimes' and it's showing the summary statistics for each column.

6.1.2.2.1. Train

The train dataset consists of dara from 2014, 2015, 2016, and 2017. The summary statistics for crimes are showing the minimum value of 70, first quartile of 476, median of 734, mean of 983.7, third quartile of 1107.2 and maximum of 8194.

```r
# Train Dataset
summary(dplyr::select(crime_test_train[[2]], c("county", "year", "month", "crimes")))
```

```
   county              year         month              crimes      
Length:1764        Min.   :2014   Length:1764        Min.   :  70.0  
Class :character   1st Qu.:2014   Class :character   1st Qu.: 476.0  
Mode  :character   Median :2015   Mode  :character   Median : 734.0  
                   Mean   :2015                      Mean   : 983.7  
                   3rd Qu.:2016                      3rd Qu.:1107.2  
                   Max.   :2017                      Max.   :8194.0
```

6.1.2.2.2. Test

The train dataset consists of dara from 2018. The summary statistics for crimes are showing the minimum value of 190, first quartile of 408, median of 626, mean of 828.7, third quartile of 920.0 and maximum of 5400.

```r
# Test Dataset
summary(dplyr::select(crime_test_train[[1]], c("county", "year", "month", "crimes")))
```

```
   county              year         month              crimes      
Length:378         Min.   :2018   Length:378         Min.   : 190.0  
Class :character   1st Qu.:2018   Class :character   1st Qu.: 408.2  
Mode  :character   Median :2018   Mode  :character   Median : 626.0  
                   Mean   :2018                      Mean   : 828.7  
                   3rd Qu.:2018                      3rd Qu.: 942.0  
                   Max.   :2018                      Max.   :5700.0  
```

6.1.2.3. Model Summary

```r
summary(mr_model)
```

```
Call:
lm(formula = crimes ~ county + year + month, data = train)

Residuals:
     Min       1Q   Median       3Q      Max 
-1711.71   -69.02     1.28    63.95  1372.47 

Coefficients:
                              Estimate Std. Error t value Pr(>|t|)    
(Intercept)                 147114.215   7180.806  20.487  < 2e-16 ***
countyBedfordshire            -750.762     35.636 -21.067  < 2e-16 ***
countyCambridgeshire          -671.738     35.636 -18.850  < 2e-16 ***
countyCheshire                -222.429     35.636  -6.242 5.45e-10 ***
countyCleveland               -408.405     35.636 -11.460  < 2e-16 ***
countyCumbria                 -675.929     35.636 -18.967  < 2e-16 ***
countyDerbyshire              -426.643     35.636 -11.972  < 2e-16 ***
countyDevon and Cornwall      -309.738     35.636  -8.692  < 2e-16 ***
countyDorset                  -649.190     35.636 -18.217  < 2e-16 ***
countyDurham                  -636.238     35.636 -17.854  < 2e-16 ***
countyDyfed Powys             -750.238     35.636 -21.053  < 2e-16 ***
countyEssex                    -95.643     35.636  -2.684  0.00735 ** 
countyGloucestershire         -804.429     35.636 -22.573  < 2e-16 ***
countyGreaterManchester        989.405     35.636  27.764  < 2e-16 ***
countyGwent                   -674.976     35.636 -18.941  < 2e-16 ***
countyHampshire                 15.190     35.636   0.426  0.66997    
countyHertfordshire           -410.143     35.636 -11.509  < 2e-16 ***
countyHumberside              -340.167     35.636  -9.545  < 2e-16 ***
countyKent                     -63.619     35.636  -1.785  0.07440 .  
countyLancashire               185.262     35.636   5.199 2.25e-07 ***
countyLeicestershire          -507.952     35.636 -14.254  < 2e-16 ***
countyLincolnshire            -590.167     35.636 -16.561  < 2e-16 ***
countyMerseyside               195.214     35.636   5.478 4.94e-08 ***
countyMetropolitan and City   5517.643     35.636 154.831  < 2e-16 ***
countyNorfolk                 -435.810     35.636 -12.229  < 2e-16 ***
countyNorth Wales             -502.500     35.636 -14.101  < 2e-16 ***
countyNorth Yorkshire         -562.238     35.636 -15.777  < 2e-16 ***
countyNorthamptonshire        -663.667     35.636 -18.623  < 2e-16 ***
countyNorthumbria              299.167     35.636   8.395  < 2e-16 ***
countyNottinghamshire         -145.500     35.636  -4.083 4.65e-05 ***
countySouth Wales              458.738     35.636  12.873  < 2e-16 ***
countySouth Yorkshire          -69.571     35.636  -1.952  0.05107 .  
countyStaffordshire           -305.381     35.636  -8.569  < 2e-16 ***
countySuffolk                 -653.024     35.636 -18.325  < 2e-16 ***
countySurrey                  -601.952     35.636 -16.891  < 2e-16 ***
countySussex                  -181.690     35.636  -5.098 3.80e-07 ***
countyThames Valley            254.071     35.636   7.130 1.48e-12 ***
countyWarwickshire            -797.548     35.636 -22.380  < 2e-16 ***
countyWest Mercia             -318.119     35.636  -8.927  < 2e-16 ***
countyWest Midlands           1238.429     35.636  34.752  < 2e-16 ***
countyWest Yorkshire           576.524     35.636  16.178  < 2e-16 ***
countyWiltshire               -733.238     35.636 -20.575  < 2e-16 ***
year                           -72.451      3.564 -20.331  < 2e-16 ***
monthaug                       -44.159     19.328  -2.285  0.02245 *  
monthdec                      -119.737     19.328  -6.195 7.29e-10 ***
monthfeb                       42.190     20.609   2.047  0.04079 *  
monthjan                      105.727     19.328   5.470 5.16e-08 ***
monthjul                       36.067     19.328   1.866  0.06221 .  
monthjun                       29.579     20.575   1.438  0.15071    
monthmar                      102.746     20.609   4.985 6.81e-07 ***
monthmay                      -19.222     20.575  -0.934  0.35030    
monthnov                      -16.842     20.711  -0.813  0.41622    
monthoct                       11.287     19.328   0.584  0.55932    
monthsep                       13.364     19.328   0.691  0.48938    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 163.3 on 1710 degrees of freedom
Multiple R-squared:  0.975,     Adjusted R-squared:  0.9742 
F-statistic:  1256 on 53 and 1710 DF,  p-value: < 2.2e-16
```

The model summary above describes a model that is used to predict the number of crimes based on the county, year, and month. The formula used to fit this model is "crimes ~ county + year + month" which means that the number of crimes is being predicted based on the county, year and month. The residuals section shows the minimum, first quartile, median, third quartile, and maximum values of the residuals which are the differences between the predicted values and the actual values. The coefficients section shows the estimate of the coefficients for each predictor variable, the standard error of the estimate, the t-value, and the p-value. The coefficients can be used to interpret the effect of each predictor variable on the response variable. The residual standard error is the standard deviation of the residuals and is used to measure the amount of error in the model. The multiple R-squared value is a statistical measure of how well the model fits the data. The F-statistic and p-value are used to test whether the overall model is significant. In this case, the model has a high R-squared value of 0.975, and a low residual standard error of 163.3, which indicates that the model has a good fit to the data, and the predictor variables are highly significant with p-value < 2.2e-16.

6.1.2.4. Model Accuracy

```
1 mr_accuracy
```

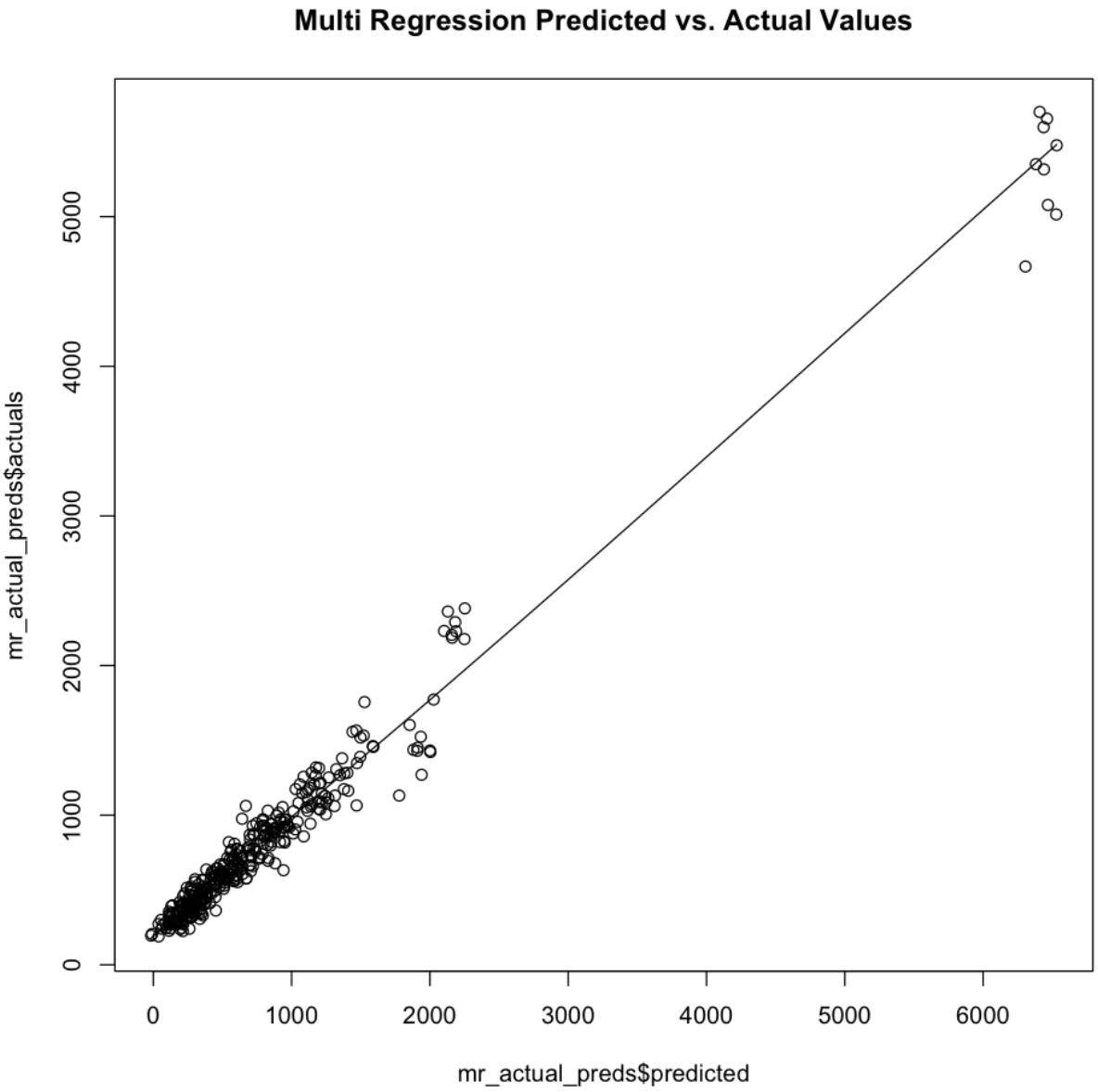
A matrix: 2 × 2 of type dbl

	actuals	predicted
actuals	1.0000000	0.9890405
predicted	0.9890405	1.0000000

The matrix above represents the accuracy of the linear regression model. The first value in the top left corner represents the correlation between the actual values of the dependent variable (crimes) and the predicted values from the model. This value is 1, indicating a perfect correlation between the actual and predicted values. The second value on the top row represents the correlation between the actual values and the predicted values. This value is 0.9890405, which is very high and indicates that the

model has a high degree of accuracy in predicting the dependent variable. The bottom left value represents the correlation between the predicted values and the actual values, which is also 0.9890405, which confirms the high accuracy of the model. The bottom right value represents the correlation between the predicted values and the predicted values, which is 1, indicating that the predicted values are perfectly correlated with themselves.

6.1.2.5. Actual vs Predicted Plot



6.1.2.6. Evaluating Stats

These values represent the difference between the predicted value and the actual value, with the MAE being the average of the absolute errors, the MSE being the average of the squared errors, and the RMSE being the square root of the MSE.

MAE	146.094951776508
MSE	55675.067652928
RMSE	235.955647639398

6.1.2.7. Code

6.1.2.7.1. Split Data

This function is used for splitting a given dataframe into a test and train dataset. It first filters out all rows that have "National" in the "county" column, then it creates a new column called "crimes" which is the sum of all numeric columns in the dataframe. It then filters the dataframe again to create a "test" dataset with only rows that have 2018 in the "year" column and a "train" dataset with all remaining rows. The function returns a list containing the test and train datasets.

```
split_data <- function(dataframe){
  dataframe <- dataframe[dataframe$county != "National", ]

  numeric_columns_df <- dplyr::select(dataframe, -c("county", "year",
"month" ,"yearmon", "region"))
  dataframe$crimes <- rowSums(numeric_columns_df)

  test <- dataframe[dataframe$year == 2018,]
  train <- dataframe[dataframe$year != 2018,]

  return(list(test, train))
}
```

6.1.2.7.2. Multiple Regression

This function creates a multiple linear regression model using the 'county', 'year', and 'month' variables as predictors and 'crimes' variable as the response variable. The model is trained using the 'train' dataframe, and predictions are made on the 'test' dataframe. The function also calculates the correlation between the actual values and the predicted values of 'crimes' variable and returns the model, the actuals vs predicted values dataframe, and correlation accuracy.

```
multiple_regression <- function(test, train){
  # training the model
  model <- lm(crimes ~ county + year + month, data = train)

  # using the model for predictions on test data
  preds <- predict(model, test, type = "response")
  actuals_preds <- data.frame(cbind(actuals=test$crimes,
predicted=preds))

  # calcultaing correaltion
  correlation_accuracy <- cor(actuals_preds)

  return(list(model, actuals_preds, correlation_accuracy))
}
```

6.2. Clustering

Clustering is a technique used in unsupervised machine learning to group similar data points together. The goal of clustering is to find patterns or structure in the data by grouping similar observations into clusters. Clustering algorithms work by dividing a dataset into a specified number of clusters based on the similarity of the data points within each cluster. Clustering is often used in applications such as market segmentation, image compression, and anomaly detection. There are various clustering algorithms such as K-means, Hierarchical clustering, and DBSCAN etc. We would be working with K-Means and Hierarchical clustering (Steinbach 2000, #).

6.2.1. KMeans

K-Means is a type of unsupervised machine learning algorithm used for clustering. The goal of K-Means is to partition a set of points (also called "samples" or "observations") into K clusters, where each point belongs to the cluster with the nearest mean. The algorithm works by first randomly initializing K cluster centroids, then repeatedly reassigning each point to the cluster corresponding to the closest centroid, and finally adjusting the position of the centroids to the mean of the points in the new clusters. This process is repeated until the centroids no longer move or a maximum number of iterations is reached. The number of clusters, K, is a user-specified parameter. The result of the K-Means algorithm is a partition of the data into K clusters.

6.2.1.1. Hypothesis

To be able to cluster all of the crime types into clusters.

6.2.1.2. Dataset Summary

We use all of the numeric attributes consisting of crime values for different crime types as our input to the model.

```
1 summary(km_data)
```

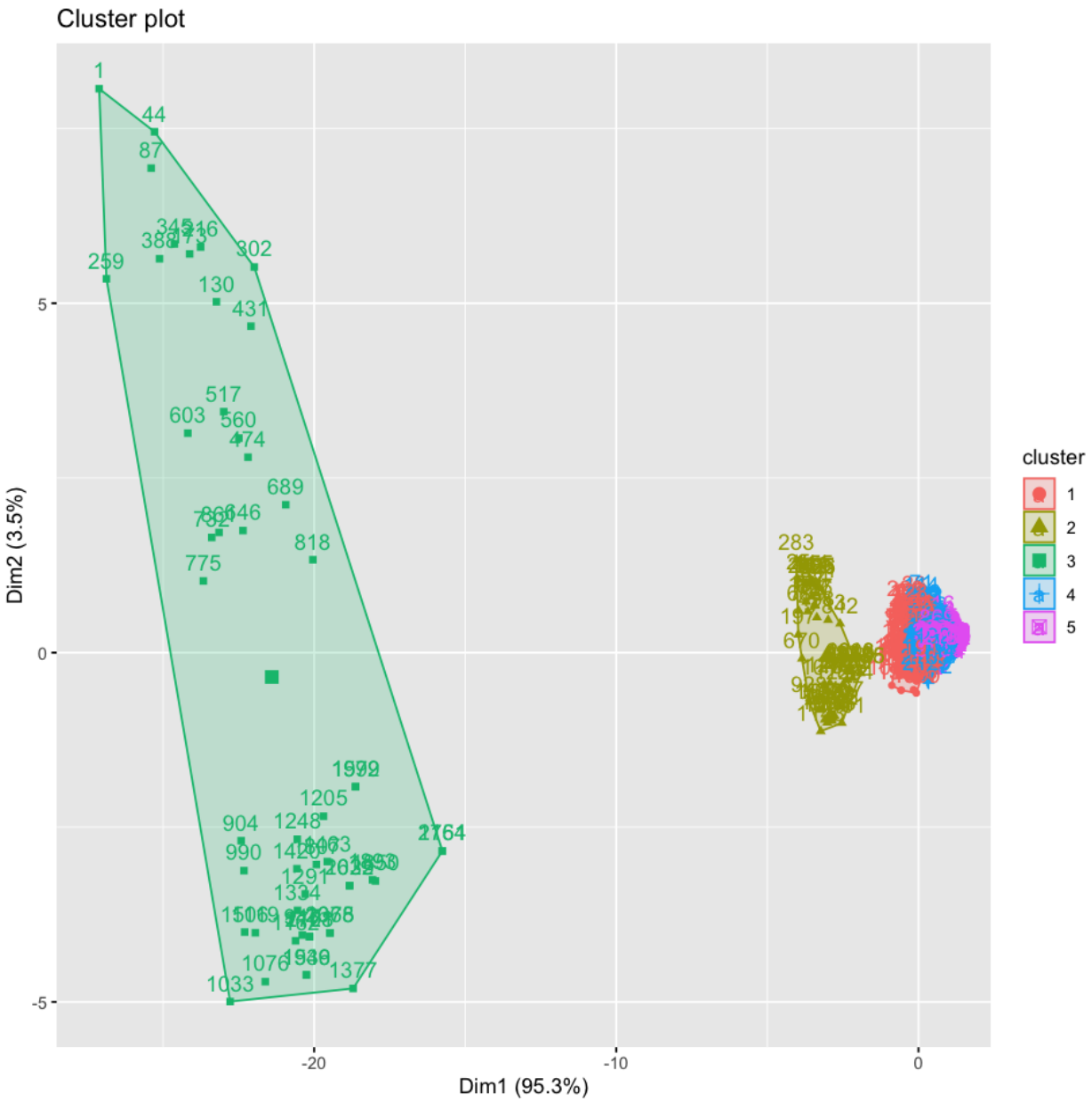
```
   homicide       offences_against_the_person sexual_offences
Min.   :  0.000   Min.   :   29.0             Min.   :   0.00
1st Qu.:  0.000   1st Qu.:  115.0             1st Qu.:   8.00
Median :  1.000   Median :  179.0             Median :  15.00
Mean   :  3.798   Mean   :  454.9             Mean   :  43.78
3rd Qu.:  3.000   3rd Qu.:  272.0             3rd Qu.:  29.00
Max.   :131.000   Max.   :11741.0             Max.   :1179.00
   burglary          robbery       theft_and_handling fraud_and_forgery
Min.   :   1.00   Min.   :  0.00   Min.   :   13.0    Min.   :   0.00
1st Qu.:  14.00   1st Qu.:  2.00   1st Qu.:   95.0    1st Qu.:   8.00
Median :  23.00   Median :  5.00   Median :  147.0    Median :  13.00
Mean   :  60.09   Mean   : 19.33   Mean   :  373.1    Mean   :  38.57
3rd Qu.:  38.00   3rd Qu.: 10.00   3rd Qu.:  237.0    3rd Qu.:  21.00
Max.   :1715.00   Max.   :650.00   Max.   :11057.0    Max.   :1075.00
criminal_damage   drugs_offences   public_order_offences
Min.   :   3.00   Min.   :   4.0   Min.   :   2.0
1st Qu.:  25.00   1st Qu.:  38.0   1st Qu.:  39.0
Median :  40.00   Median :  63.0   Median :  63.0
Mean   :  95.82   Mean   : 186.6   Mean   : 162.4
3rd Qu.:  59.00   3rd Qu.: 100.0   3rd Qu.: 100.0
Max.   :2693.00   Max.   :4988.0   Max.   :4752.0
all_other_offences__excluding_motoring_ motoring_offences
Min.   :   0.00                         Min.   :    1.0
1st Qu.:   9.00                         1st Qu.:   95.0
Median :  16.00                         Median :  143.0
Mean   :  64.34                         Mean   :  365.5
3rd Qu.:  35.00                         3rd Qu.:  216.0
Max.   :3291.00                         Max.   :12945.0
```

6.2.1.3. With 4 Clusters

The graph below shows results of the K-Means algorithm with 4 clusters as input. We can see that the algorithm has been able to find patterns and cluster the similar types of crime together.

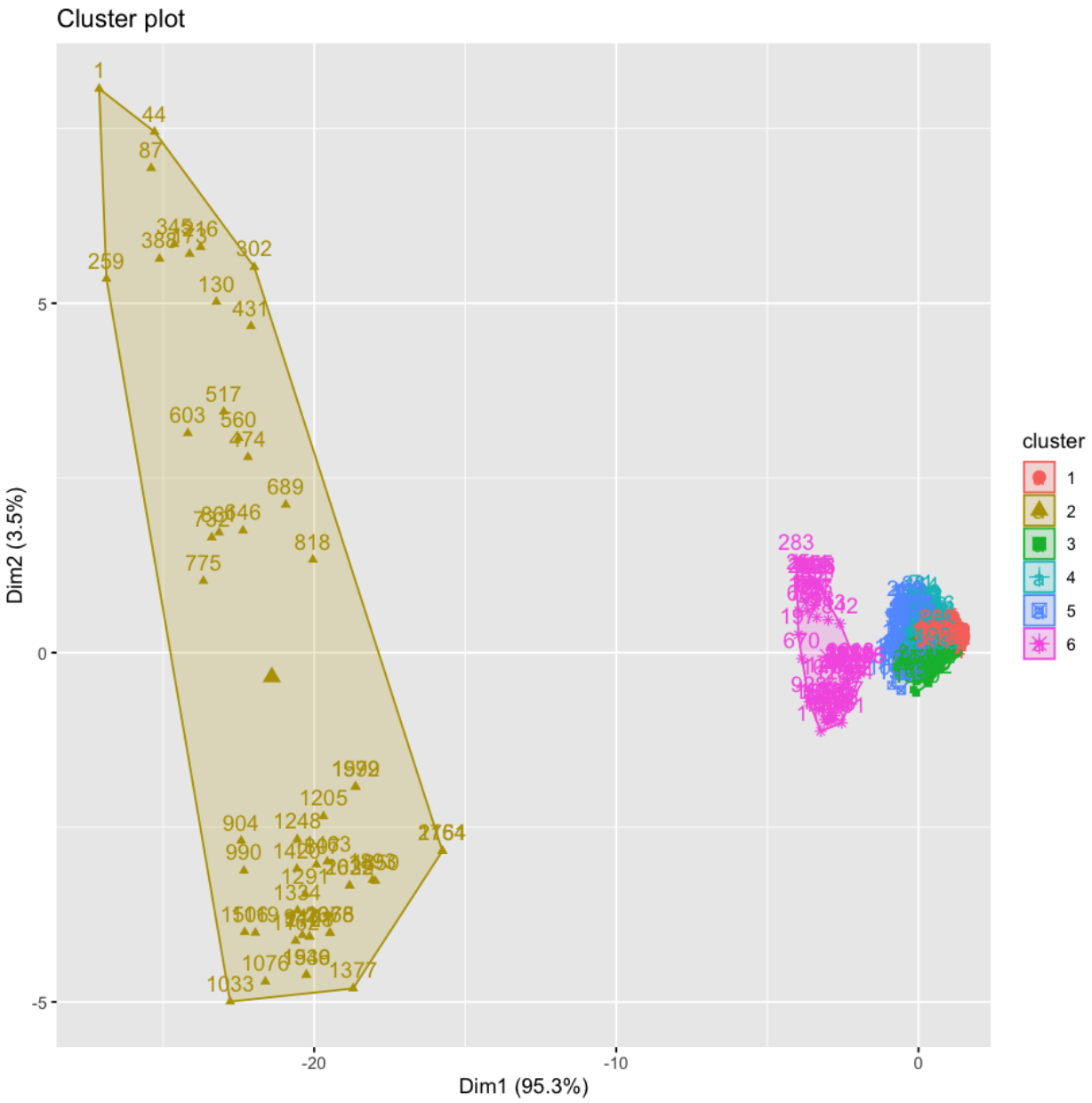
6.2.1.4. With 5 Clusters

The graph below shows results of the K-Means algorithm with 5 clusters as input. There's a further cluster added to the left side of the cluster as they have been broken down from a large cluster into smaller ones.



6.2.1.5. With 6 Clusters

The graph below shows results of the K-Means algorithm with 6 clusters as input. There's a further cluster added to the left side of the clusters as they have been broken down into further small clusters.



6.2.1.6. Summary

The best number of Clusters for our algorithms to successfully work is 5 because as we increase the size anymore it starts breaking down the clusters into smaller ones which are already really close and if we reduce them the clustering becomes really general. Hence, our model is successful in clustering all of the crime types.

6.2.1.7. Code

6.2.1.7.1. Remove Non Numeric Columns

It is used to remove all non-important attributes from the dataframe to shape for model training.

```
remove_non_numeric_cols <- function(dataframe) {
  dataframe <- dplyr::select(dataframe, -c("county", "year", "month"
,"yearmon", "region"))
  return(dataframe)
}
```

6.2.1.7.2. K Means Clustering

The functions scales the data to have similar Integer values using scale and then applies the K-Means algorithm with specified number of clusters.

```
kmeans_clustering <- function(dataframe, clusters){
  dataframe <- scale(dataframe)
  model <- kmeans(dataframe, centers = clusters, nstart = 25)
  return(model)
}
```

6.2.2. Hierarchical Clustering

Hierarchical Clustering is a method of cluster analysis which seeks to build a hierarchy of clusters. It is a type of Unsupervised Machine Learning algorithm which is used to classify a set of data points into different groups(clusters) based on their similarity. There are two main types of Hierarchical Clustering: Agglomerative and Divisive.

Agglomerative Hierarchical Clustering starts with each data point as a single-point cluster and then successively merges (agglomerates) the closest clusters, until all points are included in a single cluster or a stopping criterion is met.

Divisive Hierarchical Clustering starts with the whole data set as a single cluster and successively splits (divides) the cluster into smaller clusters, until each data point forms its own cluster or a stopping criterion is met.

The result of hierarchical clustering is a tree-based representation of the objects, called dendrogram. It shows how the objects are related to one another and the level of similarity between them. The user can decide on the number of clusters by cutting the dendrogram at the desired level.

6.2.2.1. Hypothesis

To be able to cluster the Counties using all of the crime type values.

6.2.2.2. Data Summary

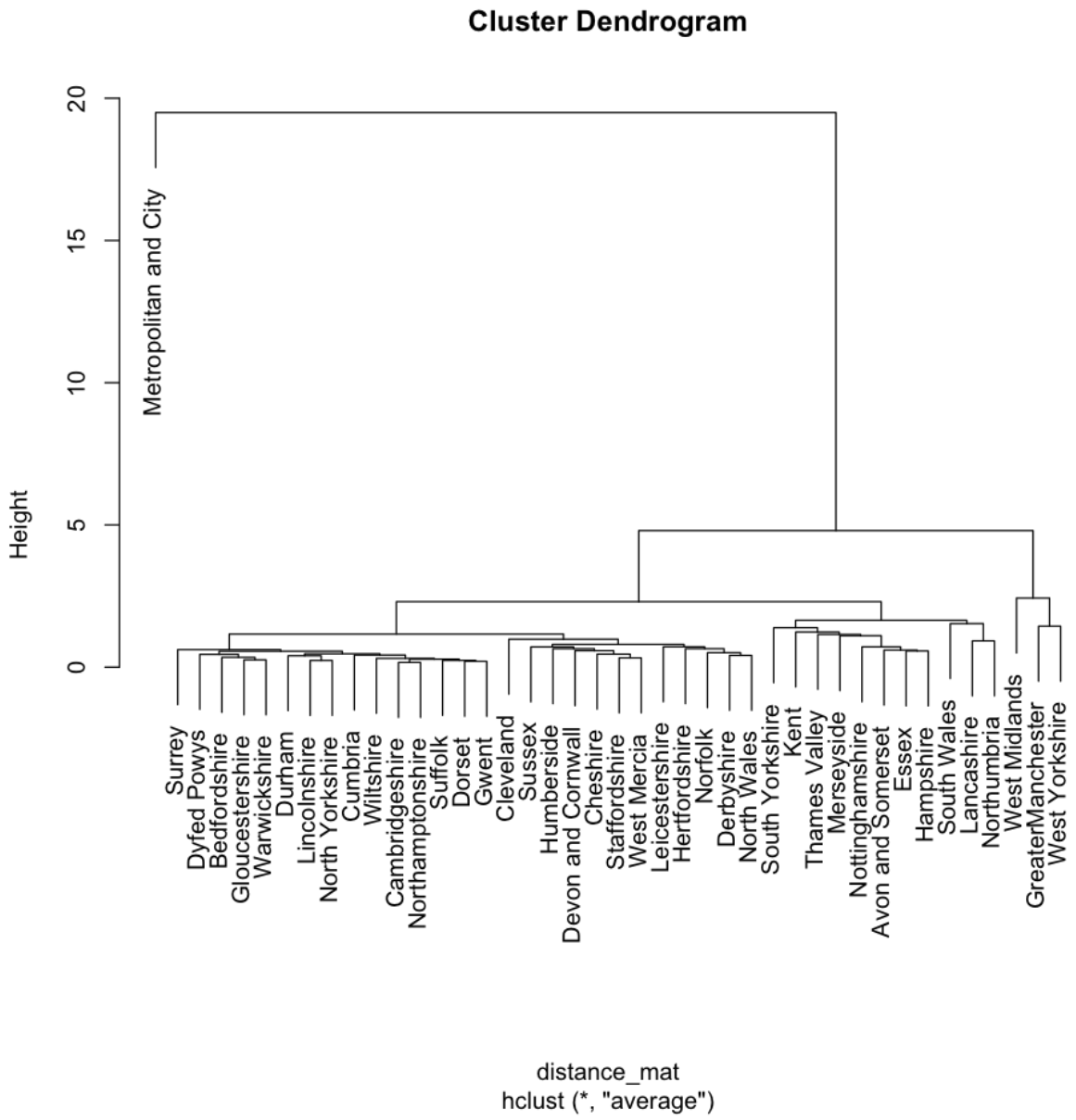
We use all of the crime types along with Counties as our row-names as input to the model.

```
summary(hc_data)
```

```
   homicide          offences_against_the_person sexual_offences
Min.   :-0.675156   Min.   :-0.7309             Min.   :-0.7916
1st Qu.:-0.441098   1st Qu.:-0.5070             1st Qu.:-0.6200
Median :-0.323133   Median :-0.2450             Median :-0.3656
Mean   : 0.000000   Mean   : 0.0000             Mean   : 0.0000
3rd Qu.:-0.002942   3rd Qu.: 0.1658             3rd Qu.: 0.2194
Max.   : 5.541414   Max.   : 5.3936             Max.   : 4.8600
   burglary            robbery          theft_and_handling fraud_and_forgery
Min.   :-0.74225   Min.   :-0.52551   Min.   :-0.9002    Min.   :-0.48988
1st Qu.:-0.51463   1st Qu.:-0.41951   1st Qu.:-0.6168    1st Qu.:-0.36277
Median :-0.28732   Median :-0.22339   Median :-0.2260    Median :-0.22695
Mean   : 0.00000   Mean   : 0.00000   Mean   : 0.0000    Mean   : 0.00000
3rd Qu.: 0.08542   3rd Qu.:-0.06666   3rd Qu.: 0.2385    3rd Qu.: 0.01061
Max.   : 5.40077   Max.   : 5.64960   Max.   : 5.0322    Max.   : 6.06289
criminal_damage   drugs_offences     public_order_offences
Min.   :-0.8810   Min.   :-0.51378   Min.   :-0.80852
1st Qu.:-0.5549   1st Qu.:-0.37550   1st Qu.:-0.50512
Median :-0.2316   Median :-0.23833   Median :-0.25195
Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.00000
3rd Qu.: 0.1832   3rd Qu.:-0.02663   3rd Qu.: 0.08909
Max.   : 5.1917   Max.   : 6.02116   Max.   : 5.34749
all_other_offences__excluding_motoring_ motoring_offences
Min.   :-0.56304                        Min.   :-0.7146
1st Qu.:-0.44967                        1st Qu.:-0.5229
Median :-0.21516                        Median :-0.3095
Mean   : 0.00000                        Mean   : 0.0000
3rd Qu.: 0.03726                        3rd Qu.: 0.1013
Max.   : 5.72932                        Max.   : 5.5173
```

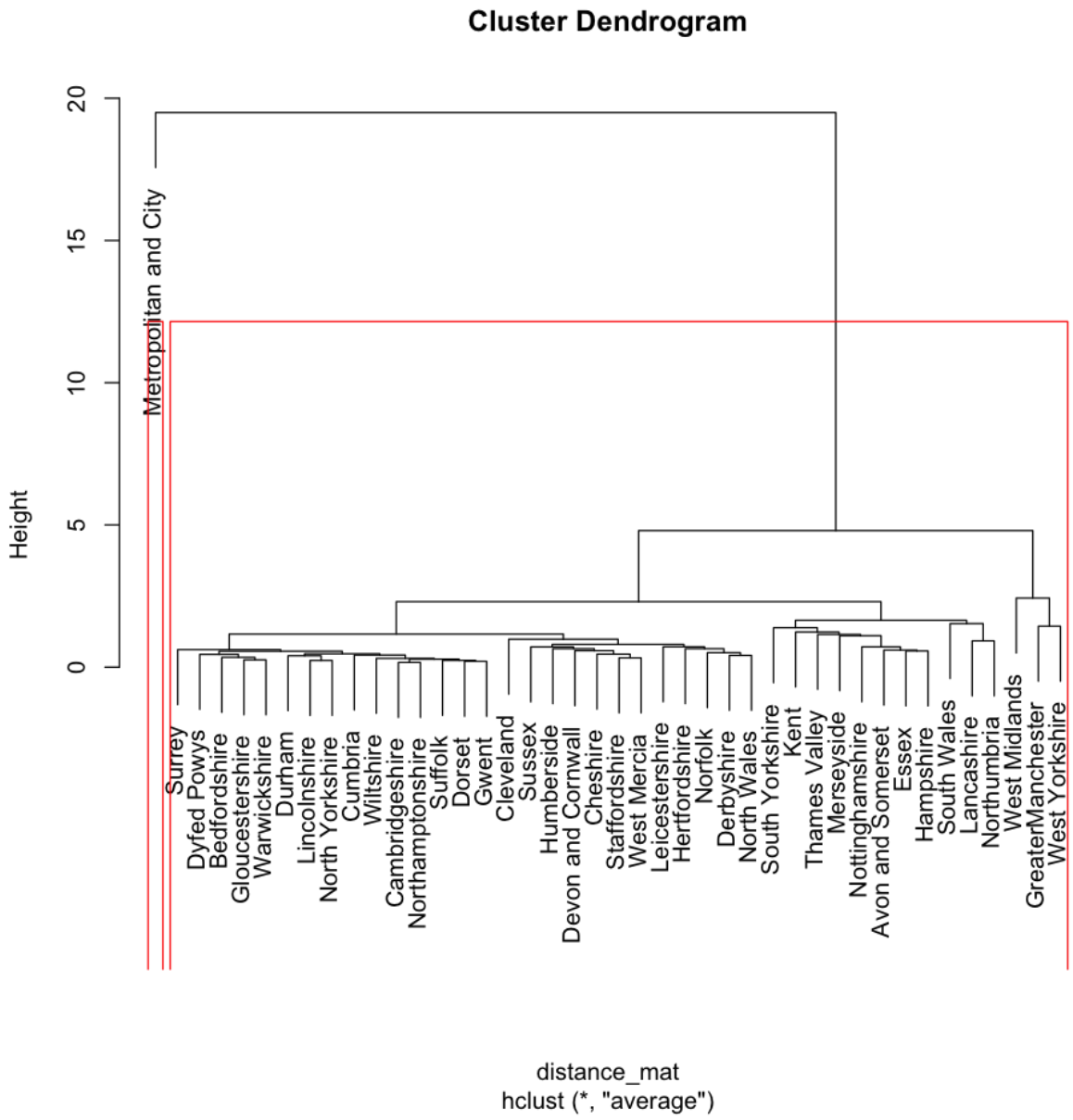
6.2.2.3. Cluster Dendrogram

The graph below shows that our algorithm has successfully constructed a dendrogram for the counties that are similar to each other. For example, Metropolitan and City county is different because it always had the most number of crimes.



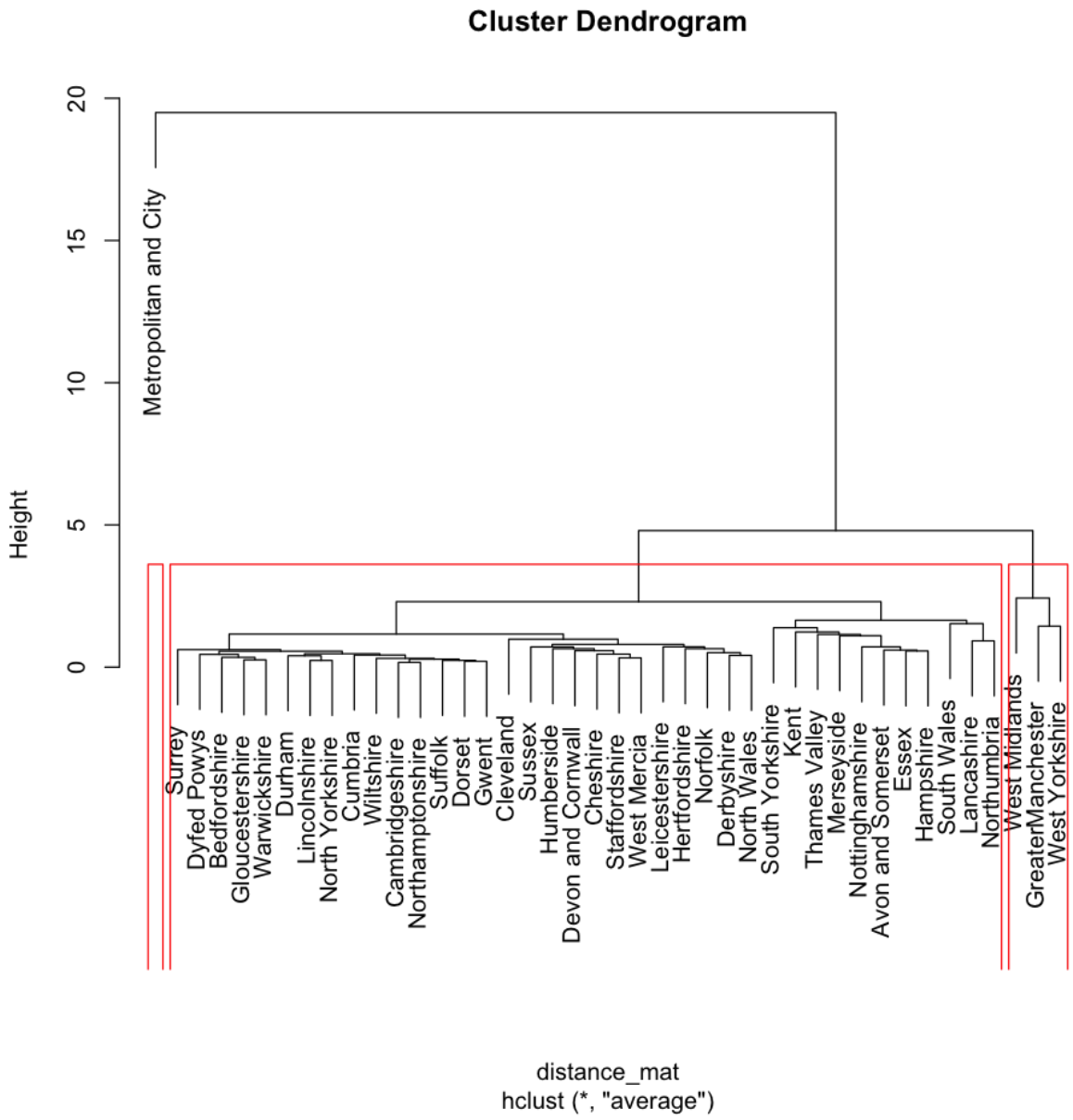
6.2.2.4. Dendrogram with 2 Clusters

We cluster the constructed dendrogram elements with 2 size and it splits it into two, i.e. Metropolitan & City county and all the other counties.



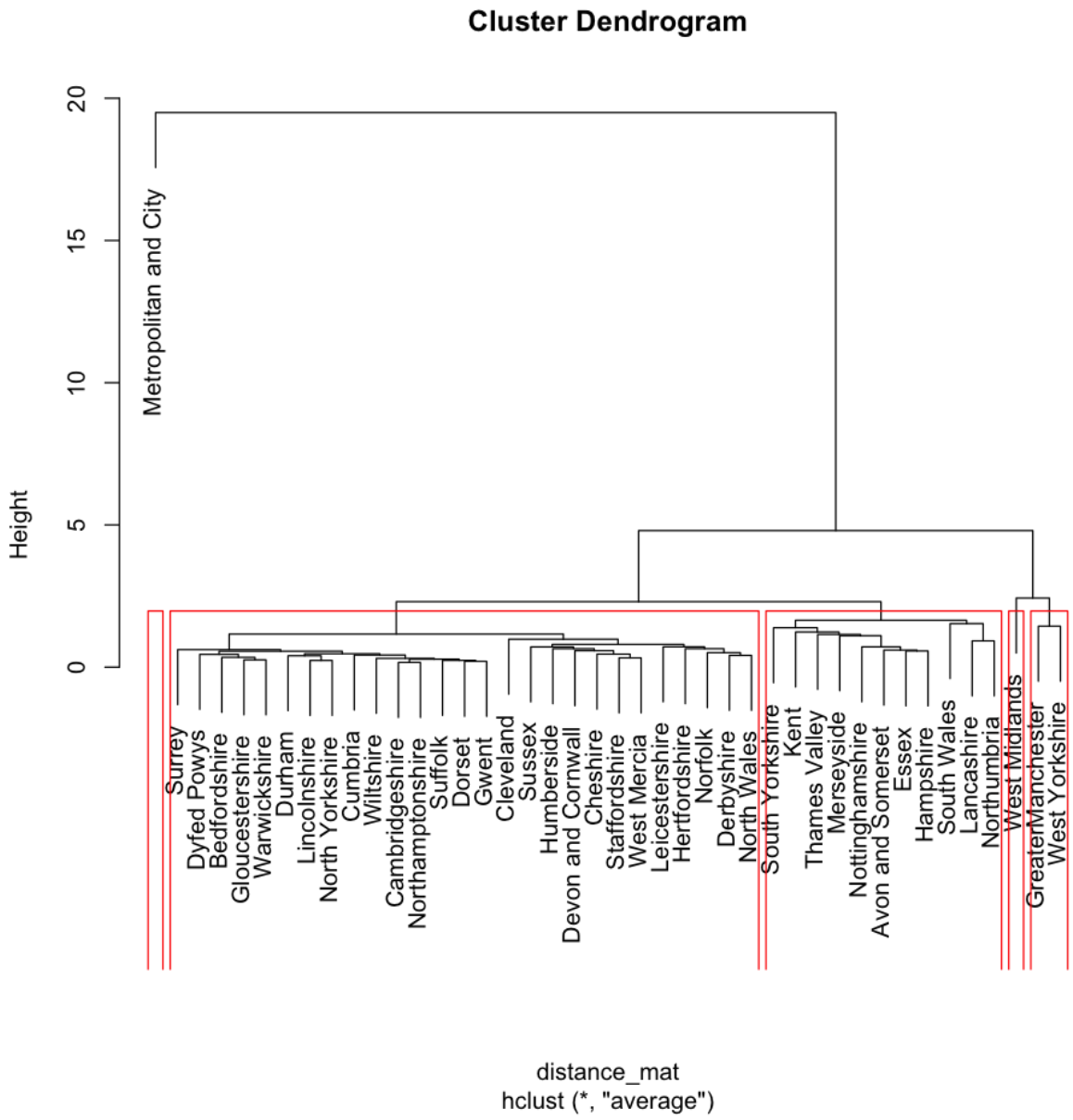
6.2.2.5. Dendrogram with 3 Clusters

We cluster the constructed dendrogram elements with 3 size and it introduces another cluster to the very left branch of the dendrogram with 3 counties under it.



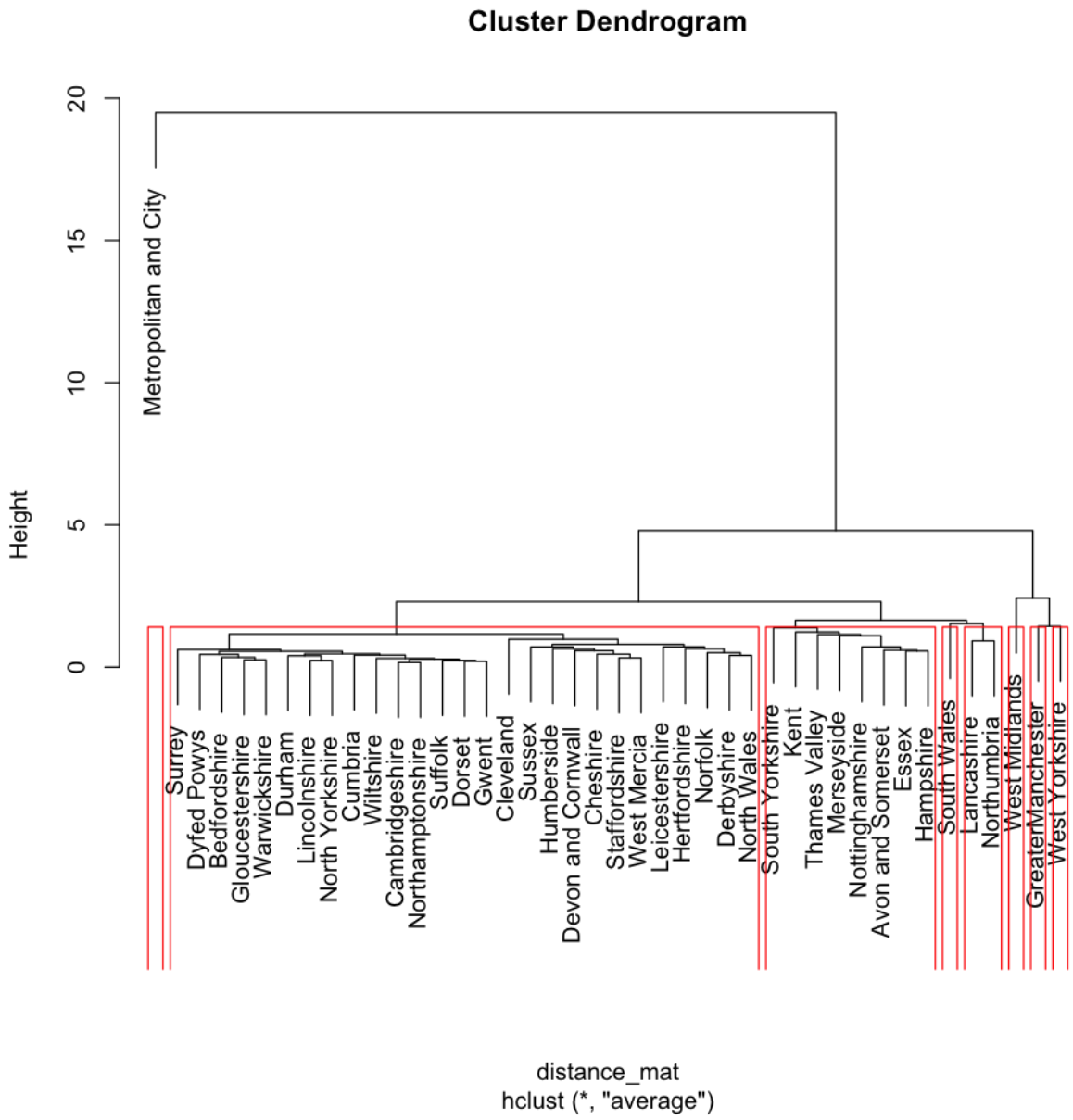
6.2.2.6. Dendrogram with 5 Clusters

We cluster the constructed dendrogram elements with 5 sizes and it helps to further cluster the counties in the dendrogram with similar attributes with better results.



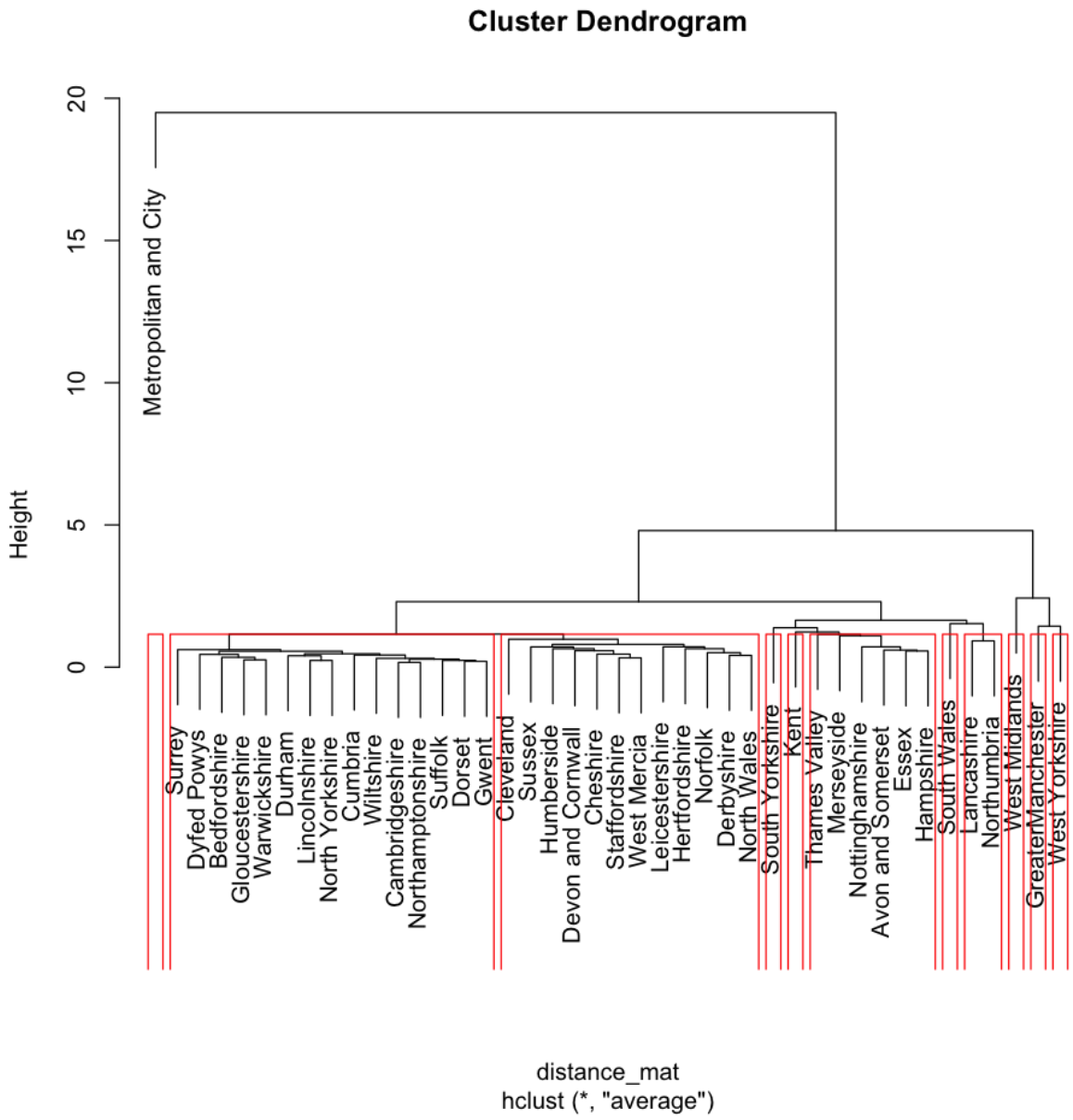
6.2.2.7. Dendrogram with 8 Clusters

We cluster the constructed dendrogram elements with 8 sizes and it helps to further cluster the counties in the dendrogram with similar attributes onto the left side of the dendrogram with better results.



6.2.2.8. Dendrogram with 11 Clusters

We cluster the constructed dendrogram elements with 11 sizes and it helps break down all of the dendrogram branches into a separate cluster containing all of the similar counties within a cluster. We consider this our final results and can finally come to the conclusion that our model has successfully clustered all the Counties.



6.2.2.8.1. Code

6.2.2.8.2. Group By Country

This functions filters and groups the data based on county and then returns scaled dataframe.

```
group_by_county <- function(dataframe){
  dataframe <- dplyr::select(dataframe, -c("year", "month"
```

```
,"yearmon", "region"))
  dataframe <- dataframe[dataframe$county != "National",]
  dataframe <- group_by(dataframe, county)
  dataframe <- summarise_all(dataframe, funs(sum))
  dataframe2 <- dataframe[,-1]
  rownames(dataframe2) <- dataframe$county
  dataframe2 <- scale(dataframe2)
  return(dataframe2)
}
```

6.2.2.8.3. HC Clustering

This function is used to create a distance matrix and apply HC clustering on the provided dataframe.

```
hc_clustering <- function(dataframe) {
  distance_mat <- dist(dataframe, method = 'euclidean')
  set.seed(240)
  model <- hclust(distance_mat, method = "average")
  return(list(distance_mat, model))
}
```

6.3. Classification

Classification is a supervised machine learning method that is used to predict the class or category of an input sample based on a set of features. It is used to categorize a set of data into predefined classes. The process of classification begins with training a model on a labeled dataset, where the target variable is already known. The model learns the relationship between the input features and the target variable, and is then able to predict the class of new, unseen data.

There are two main types of classification: binary classification and multi-class classification.

Binary classification is used to predict one of two possible outcomes. For example, whether an email is spam or not.

Multi-class classification is used to predict one of multiple possible outcomes. For example, classifying images of handwritten digits as 0-9.

There are various classification algorithms, such as Logistic Regression, k-Nearest Neighbors (k-NN), Decision Trees, Random Forest, Naive Bayes, and Neural Networks. Each algorithm has its own strengths and weaknesses and the choice of algorithm depends on the characteristics of the data and the specific problem at hand. We would be using SVM and random forest (Amrani 2018, #).

6.3.1. SVM

Support Vector Machines (SVMs) is a type of supervised learning algorithm that can be used for classification or regression tasks. The goal of an SVM is to find the best boundary (or hyperplane) that separates the data into different classes. The boundary is chosen so that it maximizes the margin, which is the distance between the boundary and the closest data points from each class, also known as support vectors.

SVMs can handle both linear and non-linear classification problems by using a technique called the kernel trick. The kernel trick allows SVMs to transform the input data into a higher dimensional space where a linear boundary can separate the classes. Common kernels used in SVMs include linear, polynomial, and radial basis function (RBF).

SVMs have been successful in many real-world applications, particularly in image and document classification, bioinformatics, and text classification. One of the main advantages of SVMs is their ability to handle high-dimensional data and their robustness to overfitting. However, they can be sensitive to the choice of kernel, and the results may be affected by the scale of the input features.

6.3.1.1. Hypothesis

To be able to successfully classify the region based on county, yearmon and sum of all crime values.

6.3.1.2. Dataset Summary

We split the dataset into test and train portions where the test set only contains 2018 data while the other years data would be used as training. The region has four classes that we have to focus on classifying.

6.3.1.2.1. Train

```
# Train
summary(svm_test_train[[2]])
```

```
               county        yearmon            no_of_crimes      region
Avon and Somerset:  42   Min.   :2014-01-01   Min.   :  70.0   East :462
Bedfordshire     :  42   1st Qu.:2014-11-01   1st Qu.: 476.0   North:546
Cambridgeshire   :  42   Median :2015-09-16   Median : 734.0   South:252
Cheshire         :  42   Mean   :2015-11-08   Mean   : 983.7   West :504
Cleveland        :  42   3rd Qu.:2016-11-01   3rd Qu.:1107.2
Cumbria          :  42   Max.   :2017-12-01   Max.   :8194.0
(Other)          :1512
```

6.3.1.2.2. Test

```
# Test
summary(svm_test_train[[1]])
```

```
               county       yearmon            no_of_crimes      region
Avon and Somerset:  9   Min.   :2018-01-01   Min.   : 190.0   East : 99
Bedfordshire     :  9   1st Qu.:2018-03-01   1st Qu.: 408.2   North:117
Cambridgeshire   :  9   Median :2018-08-01   Median : 626.0   South: 54
Cheshire         :  9   Mean   :2018-07-01   Mean   : 828.7   West :108
Cleveland        :  9   3rd Qu.:2018-10-01   3rd Qu.: 942.0
Cumbria          :  9   Max.   :2018-12-01   Max.   :5700.0
(Other)          :324
```

6.3.1.3. Model Summary

The model summary shows that we use SVM for classification using radial kernel with 1 as cost and there are 4 classes within our dataset.

```
1 summary(svm_model)
```

```
Call:
svm(formula = region ~ county + yearmon + no_of_crimes, data = train_df,
    type = "C-classification", kernal = "radial", gamma = 0.1, cost = 1)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  478

 ( 129 116 156 77 )


Number of Classes:  4

Levels:
 East North South West
```

6.3.1.4. Classification Report

```
Confusion Matrix and Statistics

          Reference
Prediction East North South West
     East    99     0     0    0
     North    0   117     0    0
     South    0     0    54    0
     West     0     0     0  108

Overall Statistics

               Accuracy : 1
                 95% CI : (0.9903, 1)
    No Information Rate : 0.3095
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1

 Mcnemar's Test P-Value : NA
```

```
Statistics by Class:

                     Class: East Class: North Class: South Class: West
Sensitivity               1.0000       1.0000       1.0000      1.0000
Specificity               1.0000       1.0000       1.0000      1.0000
Pos Pred Value            1.0000       1.0000       1.0000      1.0000
Neg Pred Value            1.0000       1.0000       1.0000      1.0000
Prevalence                0.2619       0.3095       0.1429      0.2857
Detection Rate            0.2619       0.3095       0.1429      0.2857
Detection Prevalence      0.2619       0.3095       0.1429      0.2857
Balanced Accuracy         1.0000       1.0000       1.0000      1.0000
```

This matrix and statistics report for the SVM model, in this case, the model is trained to classify the direction as East, North, South, West. The report shows the performance of the model in terms of accuracy, kappa, sensitivity, specificity, Positive Predictive Value, Negative Predictive Value, and Balanced Accuracy.

The confusion matrix shows the number of true positives, true negatives, false positives, and false negatives for each class. The overall accuracy of the model is 1, which means the model has 100% accuracy. The Kappa statistic is also 1, which indicates perfect agreement between the predicted and actual labels.

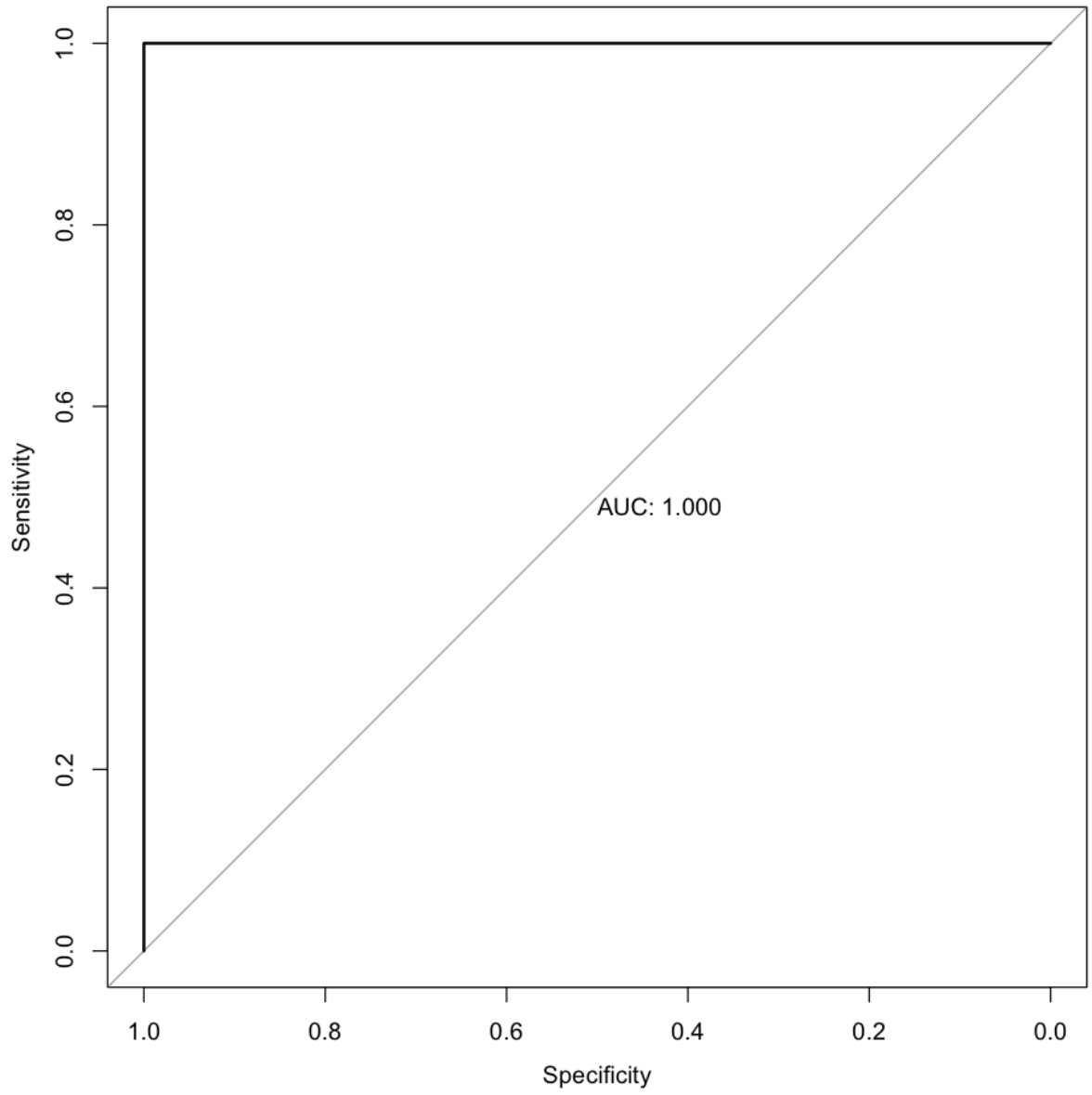
The Sensitivity, Specificity, Positive Predictive Value, and Negative Predictive Value are all 1, which means the model is able to correctly predict all the observations in the test data set. The Detection Rate, Detection Prevalence, and Prevalence are also 1, which means the model is able to detect all the observations in the test data set.

The Balanced Accuracy is also 1, which means the model is able to classify all the observations in the test data set with 100% accuracy. Additionally, the McNemar's Test P-Value is NA, which means the test is not applicable in this case.

Overall, the report indicates that the model is performing exceptionally well, with 100% accuracy in classifying the directions East, North, South, West.

6.3.1.5. Model Roc

A perfect ROC for model training which means the model is well fitted.



6.3.1.6. Code

6.3.1.6.1. SVM Model Generate

This function is used to create an SVM model, make test predictions, create confusion matrix and plot roc curves for the analysis.

```
svm_model_generate <- function(train_df, test_df) {
  set.seed(123)

  svm_model <- svm(region ~ county + yearmon + no_of_crimes, data =
train_df, type = 'C-classification', kernal = "radial", gamma = 0.1,
cost = 1)

  test_predictions <- predict(svm_model, test_df)
  confusion_matrix <- confusionMatrix(as.factor(test_predictions),
as.factor(test_df$region))
  model_roc = multiclass.roc(test_df$region ~
as.numeric(as.factor(test_predictions)), plot=TRUE, print.auc = TRUE)

  return(list(svm_model, confusion_matrix))
}
```

6.3.2. Random Forest

Random Forest is a supervised machine learning algorithm that can be used for both classification and regression tasks. It is an ensemble method, which means it combines the predictions of multiple models to improve the overall performance.

The idea behind Random Forest is to create many decision trees (hence the name "forest"), and to combine their predictions. Each decision tree is built on a different subset of the data, and each tree is grown to its maximum depth, making it highly complex. This is done to reduce the correlation between the trees, which makes the final predictions more robust and accurate.

The process of building a Random Forest model involves randomly selecting a subset of features and a subset of training data, to grow each tree. At the end of the process, the final output is the class or value that is predicted by the majority of the decision trees.

Random Forest is a powerful algorithm that can handle large datasets and high-dimensional data. It also has the ability to identify the most important features in the data, which makes it useful for feature selection. However, it can be computationally expensive to train and may require a lot of memory.

6.3.2.1. Hypothesis

To be able to successfully classify the region based on county, yearmon and sum of all unsuccessful crime values.

6.3.2.2. Dataset Summary

We split the dataset into test and train portions where the test set only contains 2018 data while the other years data would be used as training. The region has four classes that we have to focus on classifying.

6.3.2.2.1. Train

```
In [2807]:
# Train
summary(rf_test_train[[2]])
```

```
                county        yearmon            no_of_crimes      region
Avon and Somerset:  42   Min.   :2014-01-01   Min.   :  20.0   East :462
Bedfordshire     :  42   1st Qu.:2014-11-01   1st Qu.:  79.0   North:546
Cambridgeshire   :  42   Median :2015-09-16   Median : 128.0   South:252
Cheshire         :  42   Mean   :2015-11-08   Mean   : 198.6   West :504
Cleveland        :  42   3rd Qu.:2016-11-01   3rd Qu.: 212.0
Cumbria          :  42   Max.   :2017-12-01   Max.   :2458.0
(Other)          :1512
```

6.3.2.2.2. Test

```
# Test
summary(rf_test_train[[1]])
```

```
                county        yearmon            no_of_crimes      region
Avon and Somerset:   9   Min.   :2018-01-01   Min.   :  20.0   East : 99
Bedfordshire     :   9   1st Qu.:2018-03-01   1st Qu.:  70.0   North:117
Cambridgeshire   :   9   Median :2018-08-01   Median : 110.5   South: 54
Cheshire         :   9   Mean   :2018-07-01   Mean   : 165.9   West :108
Cleveland        :   9   3rd Qu.:2018-10-01   3rd Qu.: 177.8
Cumbria          :   9   Max.   :2018-12-01   Max.   :1812.0
(Other)          :324
```

6.3.2.3. Model Train Summary

The model train is attached below.

```
1 summary(rf_model)
```

```
                Length  Class  Mode
call                 4 -none- call
type                 1 -none- character
predicted         1764 factor numeric
err.rate          2500 -none- numeric
confusion           20 -none- numeric
votes             7056 matrix numeric
oob.times         1764 -none- numeric
classes              4 -none- character
importance           3 -none- numeric
importanceSD         0 -none- NULL
localImportance      0 -none- NULL
proximity      3111696 -none- numeric
ntree                1 -none- numeric
mtry                 1 -none- numeric
forest              14 -none- list
y                 1764 factor numeric
test                 0 -none- NULL
inbag                0 -none- NULL
terms                3 terms  call
```

6.3.2.4. Classification Report

```
Confusion Matrix and Statistics

          Reference
Prediction East North South West
     East    99     0     0    0
     North    0   117     0    0
     South    0     0    54    0
     West     0     0     0  108

Overall Statistics

               Accuracy : 1
                 95% CI : (0.9903, 1)
    No Information Rate : 0.3095
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1
```

```
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: East Class: North Class: South Class: West
Sensitivity               1.0000       1.0000       1.0000      1.0000
Specificity               1.0000       1.0000       1.0000      1.0000
Pos Pred Value            1.0000       1.0000       1.0000      1.0000
Neg Pred Value            1.0000       1.0000       1.0000      1.0000
Prevalence                0.2619       0.3095       0.1429      0.2857
Detection Rate            0.2619       0.3095       0.1429      0.2857
Detection Prevalence      0.2619       0.3095       0.1429      0.2857
Balanced Accuracy         1.0000       1.0000       1.0000      1.0000
```

This matrix and statistics report for the Random Forest model, in this case, the model is trained to classify the direction as East, North, South, West. The report shows the performance of the model in terms of accuracy, kappa, sensitivity, specificity, Positive Predictive Value, Negative Predictive Value, and Balanced Accuracy.

The confusion matrix shows the number of true positives, true negatives, false positives, and false negatives for each class. The overall accuracy of the model is 1, which means the model has 100% accuracy. The Kappa statistic is also 1, which indicates perfect agreement between the predicted and actual labels.

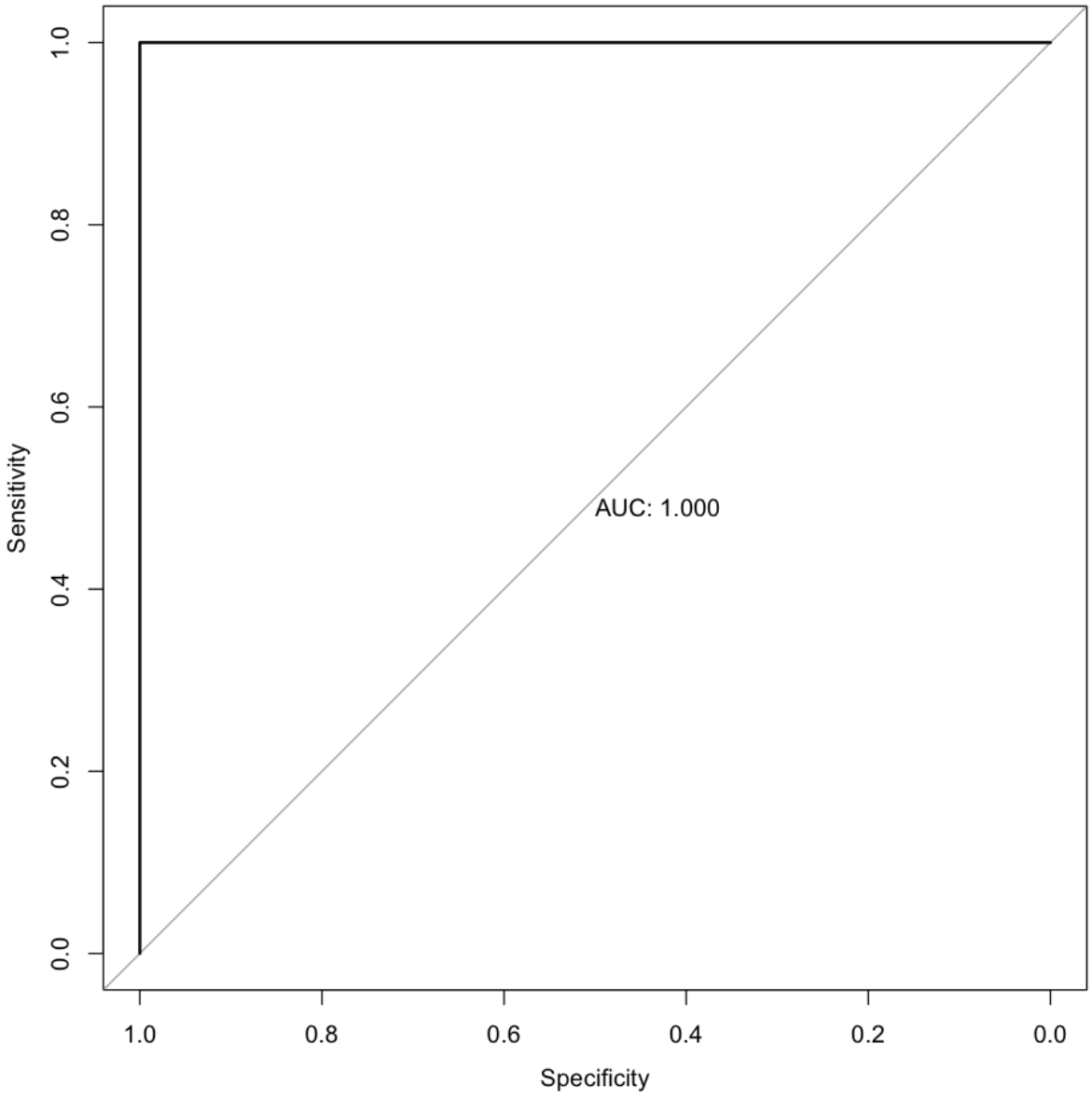
The Sensitivity, Specificity, Positive Predictive Value, and Negative Predictive Value are all 1, which means the model is able to correctly predict all the observations in the test data set. The Detection Rate, Detection Prevalence, and Prevalence are also 1, which means the model is able to detect all the observations in the test data set.

The Balanced Accuracy is also 1, which means the model is able to classify all the observations in the test data set with 100% accuracy. Additionally, the McNemar's Test P-Value is NA, which means the test is not applicable in this case.

Overall, the report indicates that the model is performing exceptionally well, with 100% accuracy in classifying the directions East, North, South, West.

6.3.2.5. Model Roc

A perfect ROC for model training which means the model is well fitted.



6.3.2.6. Code

6.3.2.6.1. Rf Model Generate

This function is used to create a Random Forest model, make test predictions, create confusion matrix and plot roc curves for the analysis.

```
rf_model_generate <- function(train_df, test_df) {
  set.seed(123)

  rf_model <- randomForest(region ~ county + yearmon + no_of_crimes,
data = train_df, proximity=TRUE)

  test_predictions <- predict(rf_model, test_df)
  confusion_matrix <- confusionMatrix(as.factor(test_predictions),
as.factor(test_df$region))
  model_roc = multiclass.roc(test_df$region ~
as.numeric(as.factor(test_predictions)), plot=TRUE, print.auc = TRUE)

  return(list(rf_model, confusion_matrix))
}
```

6.3.3. Joint Code

This function is used to split our dataframe into test and train using filtering.

```
classify_split_data <- function(dataframe){
  dataframe <- dataframe[dataframe$county != "National", ]
  dataframe$yearmon <- as.Date(dataframe$yearmon)
  dataframe$region <-  as.factor(dataframe$region)
  dataframe$county <-  as.factor(dataframe$county)

  numeric_columns_df <- dplyr::select(dataframe, -c("county", "year",
"month" ,"yearmon", "region"))
  dataframe$no_of_crimes <- rowSums(numeric_columns_df)

  test <- dataframe[dataframe$year == 2018,]
  train <- dataframe[dataframe$year != 2018,]
```

```
  test <- dplyr::select(test, c("county", "yearmon" ,"no_of_crimes"
,"region"))
  train <- dplyr::select(train, c("county", "yearmon", "no_of_crimes"
,"region"))


  return(list(test, train))
}
```

7. Time Series Analytics

Time Series Analytics is a field of study that focuses on the analysis and modeling of time-based data. It is used to understand patterns, trends, and dependencies in data that changes over time. Time series data is found in many fields, such as finance, economics, weather forecasting, and sensor data.

One of the most important aspects of time series analytics is the ability to identify patterns and trends in the data. This can be done using techniques such as smoothing, decomposition, and forecasting. Smoothing techniques, such as moving averages, can be used to remove noise from the data, while decomposition techniques, such as seasonal decomposition of time series (STL), can be used to separate the data into its various components, such as trend, seasonality, and residuals. Forecasting techniques, such as ARIMA, can be used to predict future values based on past data.

7.0.1. ARIMA

ARIMA (AutoRegressive Integrated Moving Average) is a statistical model used for time series forecasting. It is a linear model that combines three components: autoregression (AR), differencing (I), and moving average (MA) (“Time Series Analysis using Arima Model” 2021).

- The autoregression (AR) component models the dependence between an observation and a number of lagged observations.
- The differencing (I) component models the dependence between an observation and the differences between consecutive observations. This is used to make the time series data stationary, meaning that the statistical properties of the data do not change over time.
- The moving average (MA) component models the dependence between an observation and a residual error from a moving average model applied to lagged observations.

The ARIMA model is defined by three parameters: (p,d,q) where p is the order of the autoregression component, d is the order of differencing, and q is the order of the moving average component.

The process of fitting an ARIMA model to a time series data is often referred as ARIMA modeling, which consists of:

- Identifying the order of differencing required to make the series stationary
- Identifying the values of p and q that best model the autocorrelation and partial autocorrelation in the stationary series
- Fitting the final model and making predictions

ARIMA is a powerful tool for time series forecasting, but it has some limitations. It assumes that the time series is stationary, which may not be the case for all datasets. It also assumes that the errors are normally distributed, which may not be true in some cases. It also assumes that time series data is linear, which may not be the case for some nonlinear time series data. In such cases, other techniques such as Exponential Smoothing or Deep learning techniques like LSTM may be more appropriate.

7.0.2. Hypothesis

To be able to predict the future crime occurrences.

7.0.2.1. Dataset Summary

We group and filter the dataframe to get year, month, date and sum of all the crimes.

7.0.2.1.1. Data Frame

```
1 crime_df_mr <- check_and_add_missing_rows(crime_df)
```

```
1 head(crime_df_mr)
```

A data.frame: 6 × 4

	year	month	no_of_crimes	date
	<dbl>	<chr>	<dbl>	<date>
1	2014	jan	52683	2014-01-01
44	2014	feb	48230	2014-02-01
87	2014	mar	48231	2014-03-01
130	2014	apr	42549	2014-04-01
173	2014	may	44265	2014-05-01
216	2014	jun	44110	2014-06-01

7.0.2.1.2. Time Series

We convert our data frame into Time Series format so we can feed it to the model. The model only works with time Series Data.

```
1 crime_ts <- convert_to_ts(crime_df_mr)
```

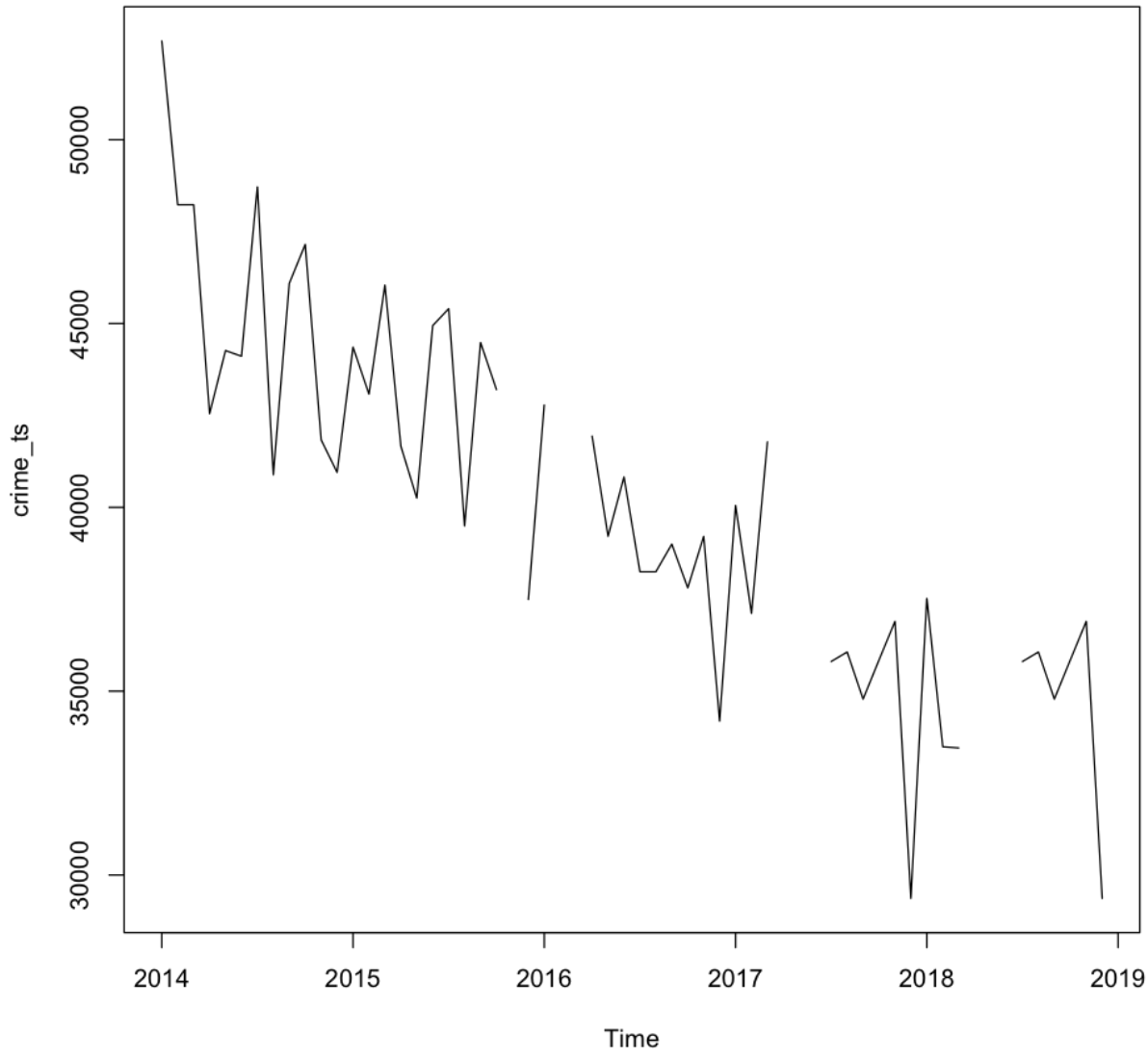
```
1 crime_ts
```

A Time Series: 5 × 12

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2014	52683	48230	48231	42549	44265	44110	48715	40888	46092	47152	41836	40952
2015	44359	43079	46044	41665	40252	44943	45404	39495	44484	43202	NA	37496
2016	42785	NA	NA	41938	39213	40826	38249	38249	39000	37811	39209	34186
2017	40052	37116	41780	NA	NA	NA	35808	36066	34786	35848	36899	29367
2018	37521	33488	33457	NA	NA	NA	35808	36066	34786	35848	36899	29367

7.0.2.2. Time Series Data Plot

We plot the Time Series data plot below. There is some missing data within the plot that have a bad impact on our analysis or cause our diagnostic tests to fail. The good thing to notice is that the sum of crimes is going down every year.



7.0.2.3. Filling Missing Data with Mice

In order to fill all the missing data, we use the Mice packing for importing all the missing data.

7.0.2.3.1. Mice Summary

The summary highlights the Mice model summary with the operations required to conduct for imputing missing values.

```
1 summary(crime_df_mice)
```

```
Class: mids
Number of multiple imputations:  5
Imputation methods:
        year        month no_of_crimes         date
          ""           ""        "pmm"           ""
PredictorMatrix:
             year month no_of_crimes date
year            0     0            1    1
month           1     0            1    1
no_of_crimes    1     0            0    1
date            1     0            1    0
Number of logged events:  26
  it im          dep     meth
1  0  0              constant
2  1  1 no_of_crimes      pmm
3  1  2 no_of_crimes      pmm
4  1  3 no_of_crimes      pmm
5  1  4 no_of_crimes      pmm
6  1  5 no_of_crimes      pmm

out
1
month
2 mice detected that your data are (nearly) multi-collinear.\nIt applied a ridge penalty to continue calculations, bu
t the results can be unstable.\nDoes your dataset contain duplicates, linear transformation, or factors with unique r
espondent names?
3 mice detected that your data are (nearly) multi-collinear.\nIt applied a ridge penalty to continue calculations, bu
t the results can be unstable.\nDoes your dataset contain duplicates, linear transformation, or factors with unique r
espondent names?
4 mice detected that your data are (nearly) multi-collinear.\nIt applied a ridge penalty to continue calculations, bu
t the results can be unstable.\nDoes your dataset contain duplicates, linear transformation, or factors with unique r
espondent names?
5 mice detected that your data are (nearly) multi-collinear.\nIt applied a ridge penalty to continue calculations, bu
t the results can be unstable.\nDoes your dataset contain duplicates, linear transformation, or factors with unique r
espondent names?
6 mice detected that your data are (nearly) multi-collinear.\nIt applied a ridge penalty to continue calculations, bu
t the results can be unstable.\nDoes your dataset contain duplicates, linear transformation, or factors with unique r
```

7.0.2.3.2. Mice Filled Data

The final dataset when all the missing values have been filled.

```
crime_df_filled <- complete(crime_df_mice)
```

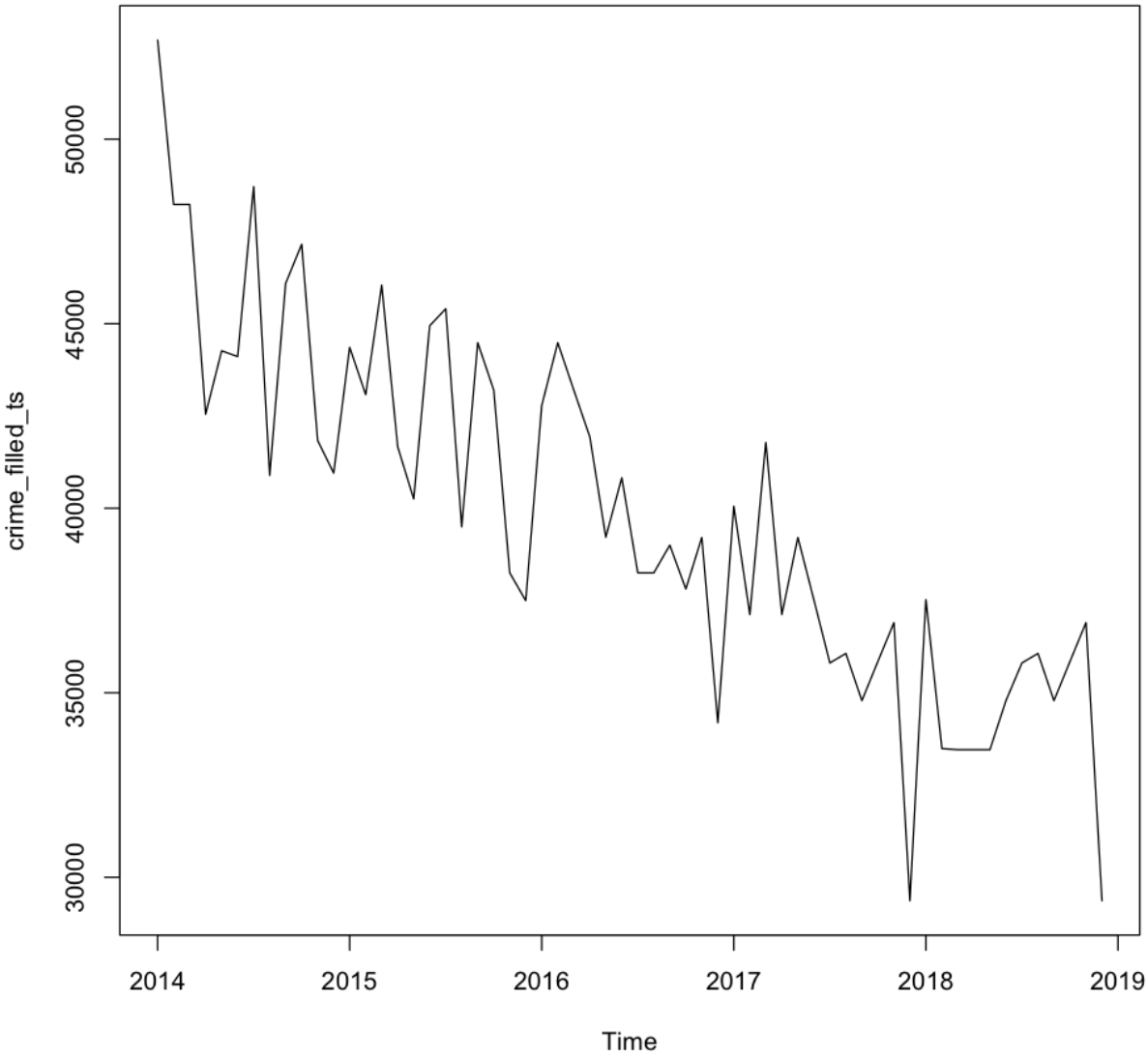
```
crime_filled_ts <- convert_to_ts(crime_df_filled)
crime_filled_ts
```

A Time Series: 5 × 12

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2014	52683	48230	48231	42549	44265	44110	48715	40888	46092	47152	41836	40952
2015	44359	43079	46044	41665	40252	44943	45404	39495	44484	43202	38249	37496
2016	42785	44484	43202	41938	39213	40826	38249	38249	39000	37811	39209	34186
2017	40052	37116	41780	37116	39209	37521	35808	36066	34786	35848	36899	29367
2018	37521	33488	33457	33457	33457	34786	35808	36066	34786	35848	36899	29367

7.0.2.4. Filled Time Series Data Plot

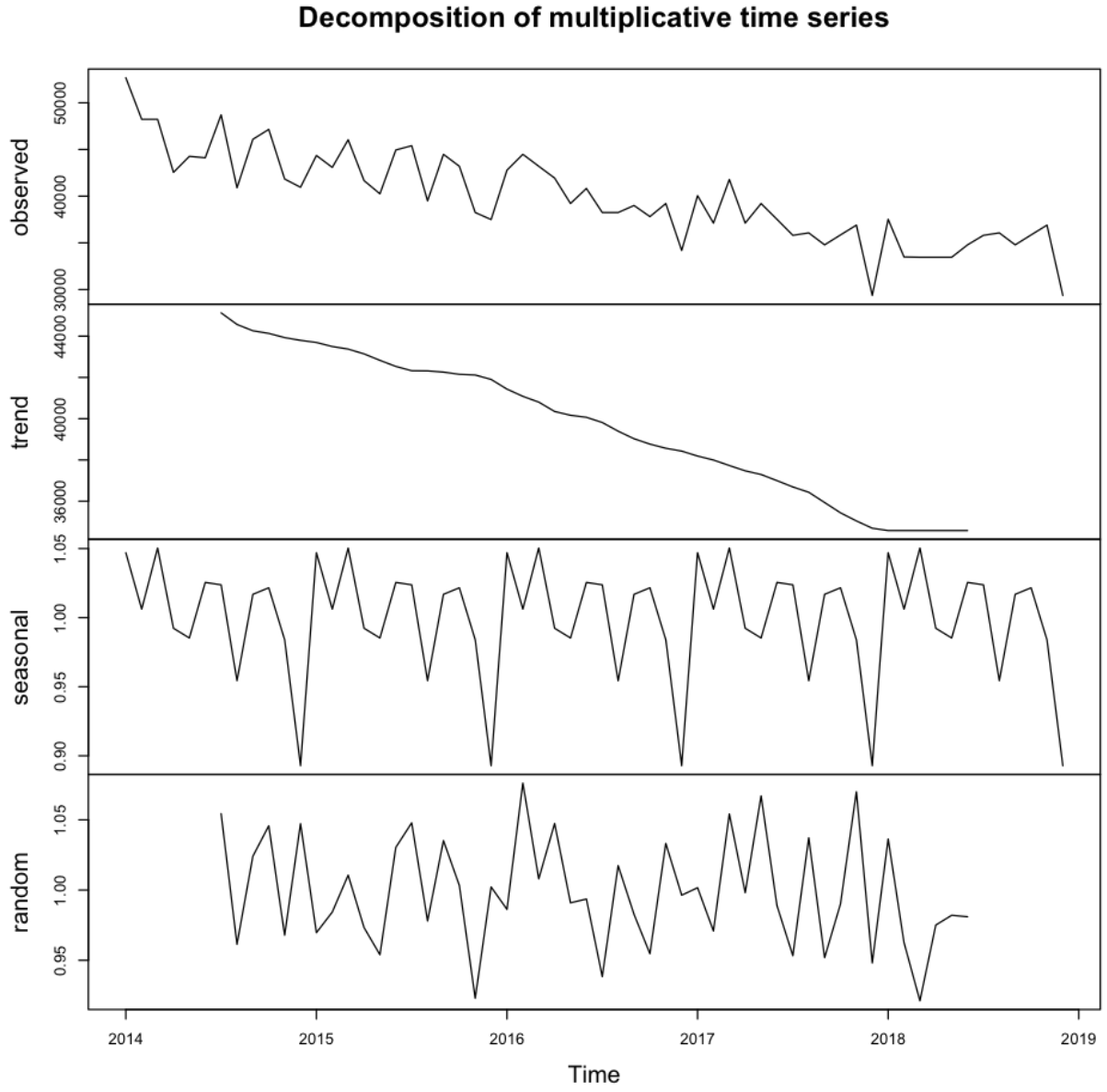
Now, the missing values have been filled and we can see that our data has gotten smoother without any inconsistencies.



7.0.2.5. Data Patterns Check

Decomposition of multiplicative time series is a method of breaking down a time series into its components, such as trend, seasonality, and residuals. The method is used for time series data that exhibit a multiplicative relationship between the components, meaning that the magnitude of the seasonal and trend components are related to the level of the series.

The decomposition process involves separating the time series into its trend, seasonal and irregular components. The trend component captures the long-term changes in the data, the seasonal component captures the repetitive patterns that occur at specific time intervals, and the irregular component captures the random and unpredictable fluctuations in the data.



7.0.2.6. Diagnostic Check

In order to apply forecasts, we first need to perform and pass some diagnostic tests which would decide whether we can forecast or not.

7.0.2.6.1. Residuals

A residual matrix, also known as the residuals array, is a matrix that contains the difference between the observed values and the predicted values of a model. It is used to evaluate the performance of a model and to identify patterns and trends in the residuals.

```
model_fit_resd = residuals(model_fit)
model_fit_resd
```

A Time Series: 5 × 12

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2014	52.68293	-2817.12064	-1525.25067	-5259.26881	-2299.04701	-1748.73550	2330.85849	-4425.85186	852.47380	1450.86870	-3104.46658	-2958.16364
2015	-2117.01619	110.76176	2961.14054	1241.22625	-1574.01033	3632.24274	241.30446	-867.63870	1121.02581	-1122.18826	-2467.99738	-2005.61603
2016	1706.77087	3740.91728	-392.67459	1170.79825	-991.72158	-2026.79483	-4343.76944	500.05021	-1984.50182	-1841.59632	3131.33533	-2268.04590
2017	922.37775	-3318.29025	3037.27627	-1660.15001	2574.13037	-810.48996	-703.30746	-255.65761	-1933.47221	388.87182	466.14035	-4069.71330
2018	1535.92651	-1086.49023	-3723.23461	179.47258	-1169.74271	1523.79172	3199.54086	2434.42153	1293.50177	1346.66023	1381.32089	-1841.94733

7.0.2.6.2. Box-Ljung test

As the p-value here is non-zero for the Box-Ljung test so our model has successfully passed the test.

```
Box.test(model_fit_resd, lag=10,type="Ljung-Box")
```

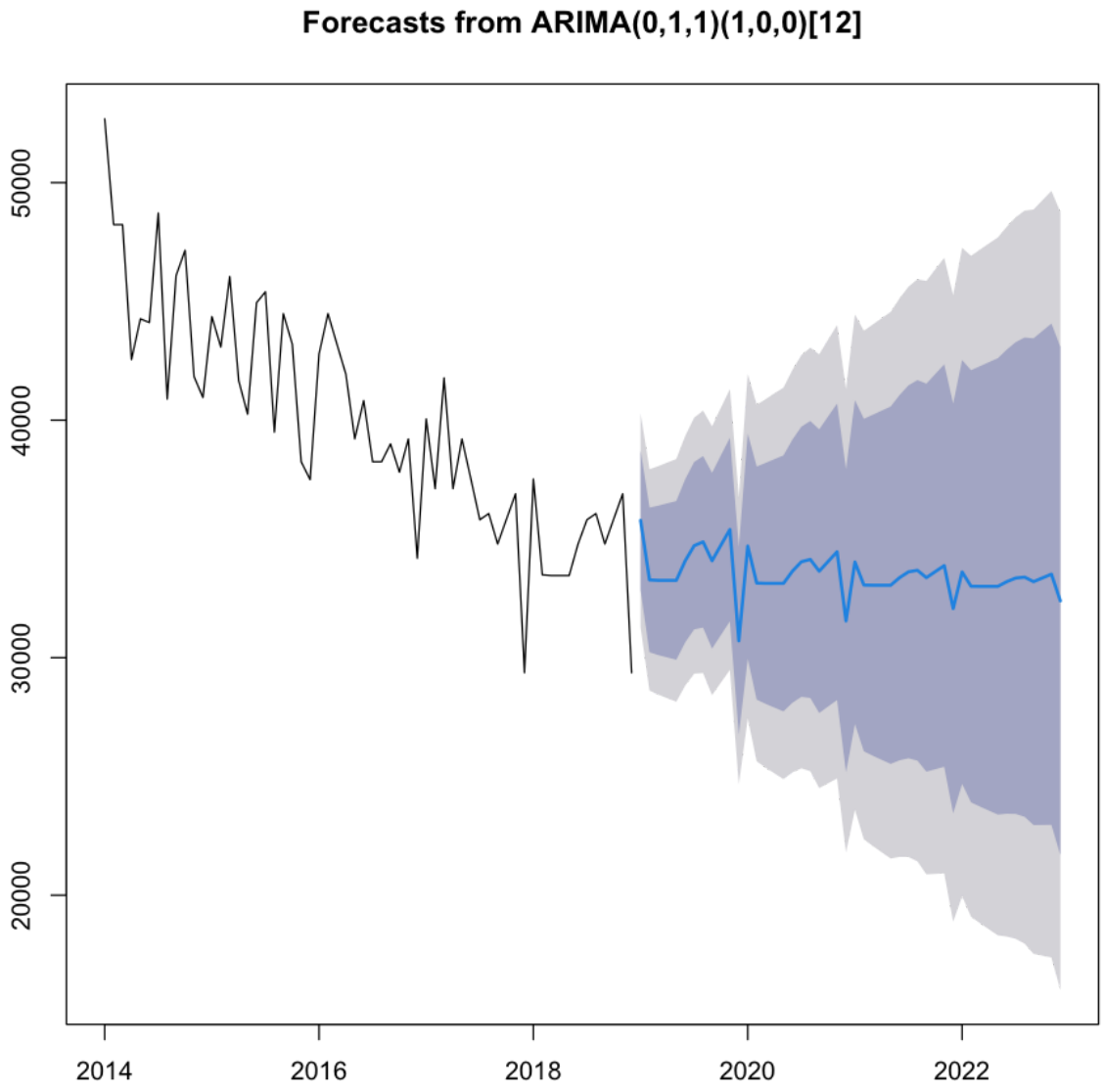
```
        Box-Ljung test

data:  model_fit_resd
X-squared = 4.6749, df = 10, p-value = 0.9118
```

7.0.2.7. Forecasting

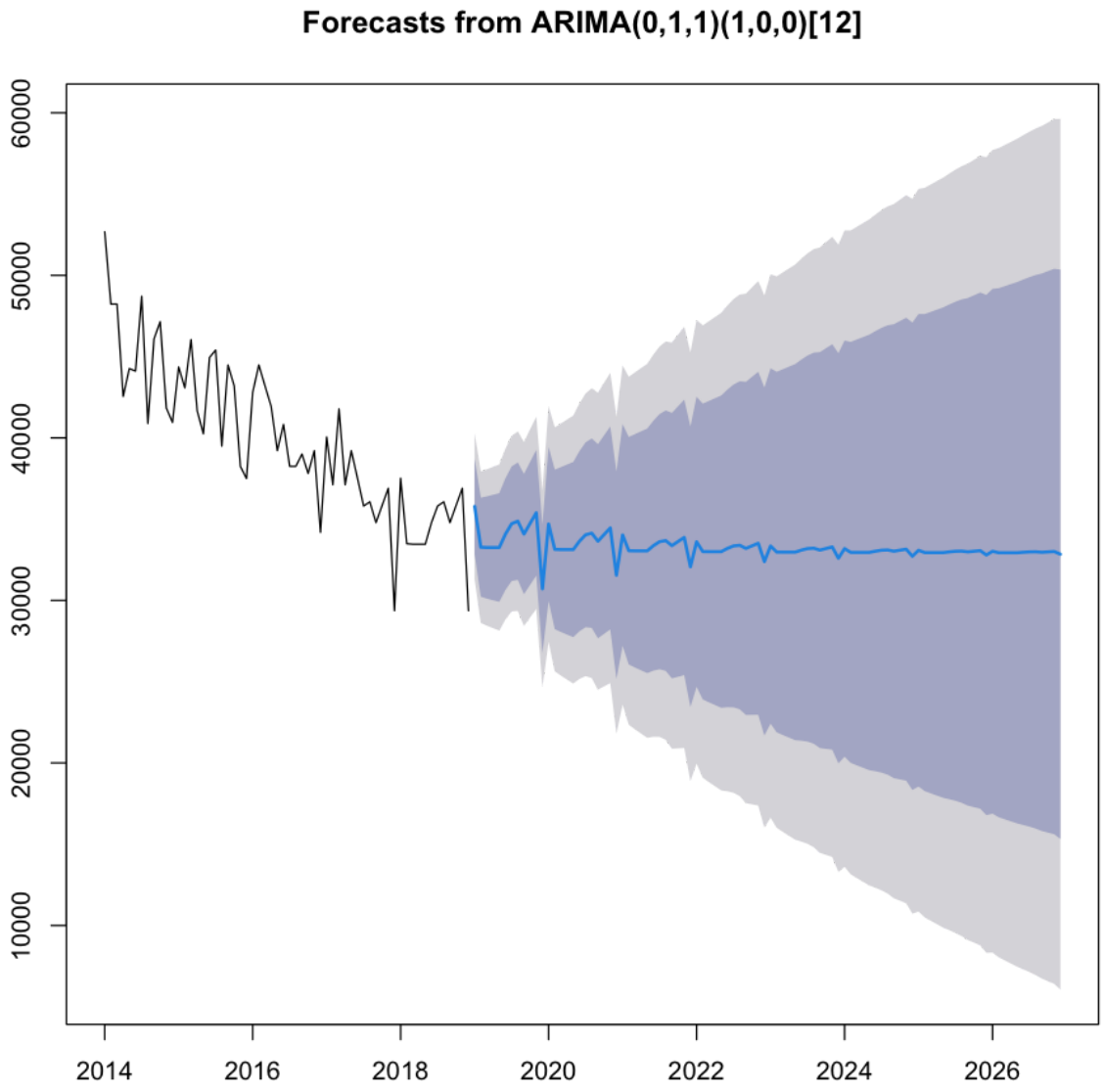
7.0.2.7.1. Years

The four year forecast shows that the crimes wouldn't decrease but they would stay within the range of 30,000 which means the current crime rate would be maintained.



7.0.2.7.2. Years

The 8 year forecast shows that the crimes would eventually stay within the range of 30,000 which means the current crime rate would be maintained.



7.0.3. Hypothesis

To be able to predict the future unsuccessful crime occurrences.

7.0.3.1. Dataset Summary

We group and filter the dataframe to get year, month, date and sum of all the crimes.

7.0.3.1.1. Data Frame

```
1 uscrime_df_mr <- check_and_add_missing_rows(uscrime_df)
```

```
1 head(uscrime_df_mr)
```

A data.frame: 6 × 4

	year	month	no_of_crimes	date
	<dbl>	<chr>	<dbl>	<date>
1	2014	jan	9301	2014-01-01
44	2014	feb	8987	2014-02-01
87	2014	mar	8958	2014-03-01
130	2014	apr	8164	2014-04-01
173	2014	may	8697	2014-05-01
216	2014	jun	8753	2014-06-01

7.0.3.1.2. Time Series

We convert our data frame into Time Series format so we can feed it to the model. The model only works with time Series Data.

```
1 uscrime_ts <- convert_to_ts(uscrime_df_mr)
```

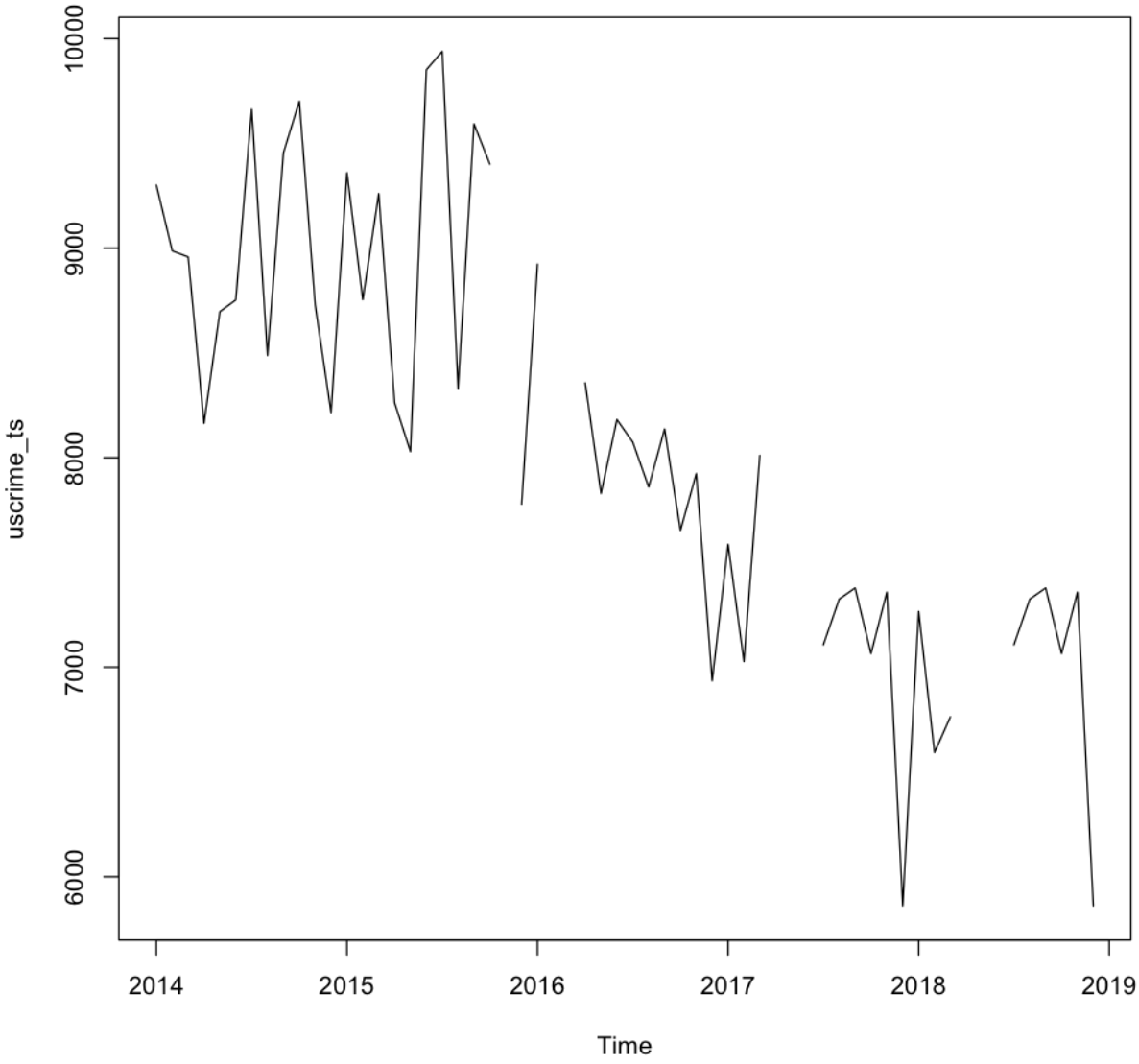
```
1 uscrime_ts
```

A Time Series: 5 × 12

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2014	9301	8987	8958	8164	8697	8753	9663	8488	9455	9701	8729	8215
2015	9360	8754	9260	8262	8029	9851	9939	8331	9593	9401	NA	7778
2016	8924	NA	NA	8357	7829	8182	8074	7860	8137	7653	7924	6935
2017	7586	7027	8011	NA	NA	NA	7107	7325	7378	7065	7358	5861
2018	7266	6593	6763	NA	NA	NA	7107	7325	7378	7065	7358	5861

7.0.3.2. Time Series Data Plot

We plot the Time Series data plot below. There is some missing data within the plot that have a bad impact on our analysis or cause our diagnostic tests to fail. The good thing to notice is that the sum of unsuccessful crimes is going down every year.



7.0.3.3. Filling Missing Data with Mice

In order to fill all the missing data, we use the Mice packing for importing all the missing data.

7.0.3.3.1. Mice Summary

The summary highlights the Mice model summary with the operations required to conduct for imputing missing values.

```
1 summary(crime_df_mice)
```

```
Class: mids
Number of multiple imputations:  5
Imputation methods:
        year        month no_of_crimes         date
          ""           ""        "pmm"           ""
PredictorMatrix:
             year month no_of_crimes date
year            0     0            1    1
month           1     0            1    1
no_of_crimes    1     0            0    1
date            1     0            1    0
Number of logged events:  26
  it im          dep     meth
1  0  0             constant
2  1  1 no_of_crimes      pmm
3  1  2 no_of_crimes      pmm
4  1  3 no_of_crimes      pmm
5  1  4 no_of_crimes      pmm
6  1  5 no_of_crimes      pmm

out
1
month
2 mice detected that your data are (nearly) multi-collinear.\nIt applied a ridge penalty to continue calculations, bu
t the results can be unstable.\nDoes your dataset contain duplicates, linear transformation, or factors with unique r
espondent names?
3 mice detected that your data are (nearly) multi-collinear.\nIt applied a ridge penalty to continue calculations, bu
t the results can be unstable.\nDoes your dataset contain duplicates, linear transformation, or factors with unique r
espondent names?
4 mice detected that your data are (nearly) multi-collinear.\nIt applied a ridge penalty to continue calculations, bu
t the results can be unstable.\nDoes your dataset contain duplicates, linear transformation, or factors with unique r
espondent names?
5 mice detected that your data are (nearly) multi-collinear.\nIt applied a ridge penalty to continue calculations, bu
t the results can be unstable.\nDoes your dataset contain duplicates, linear transformation, or factors with unique r
espondent names?
6 mice detected that your data are (nearly) multi-collinear.\nIt applied a ridge penalty to continue calculations, bu
t the results can be unstable.\nDoes your dataset contain duplicates, linear transformation, or factors with unique r
espondent names?
```

7.0.3.3.2. Mice Filled Data

The final dataset when all the missing values have been filled.

```
1 uscrime_df_filled <- complete(uscrime_df_mice)
```

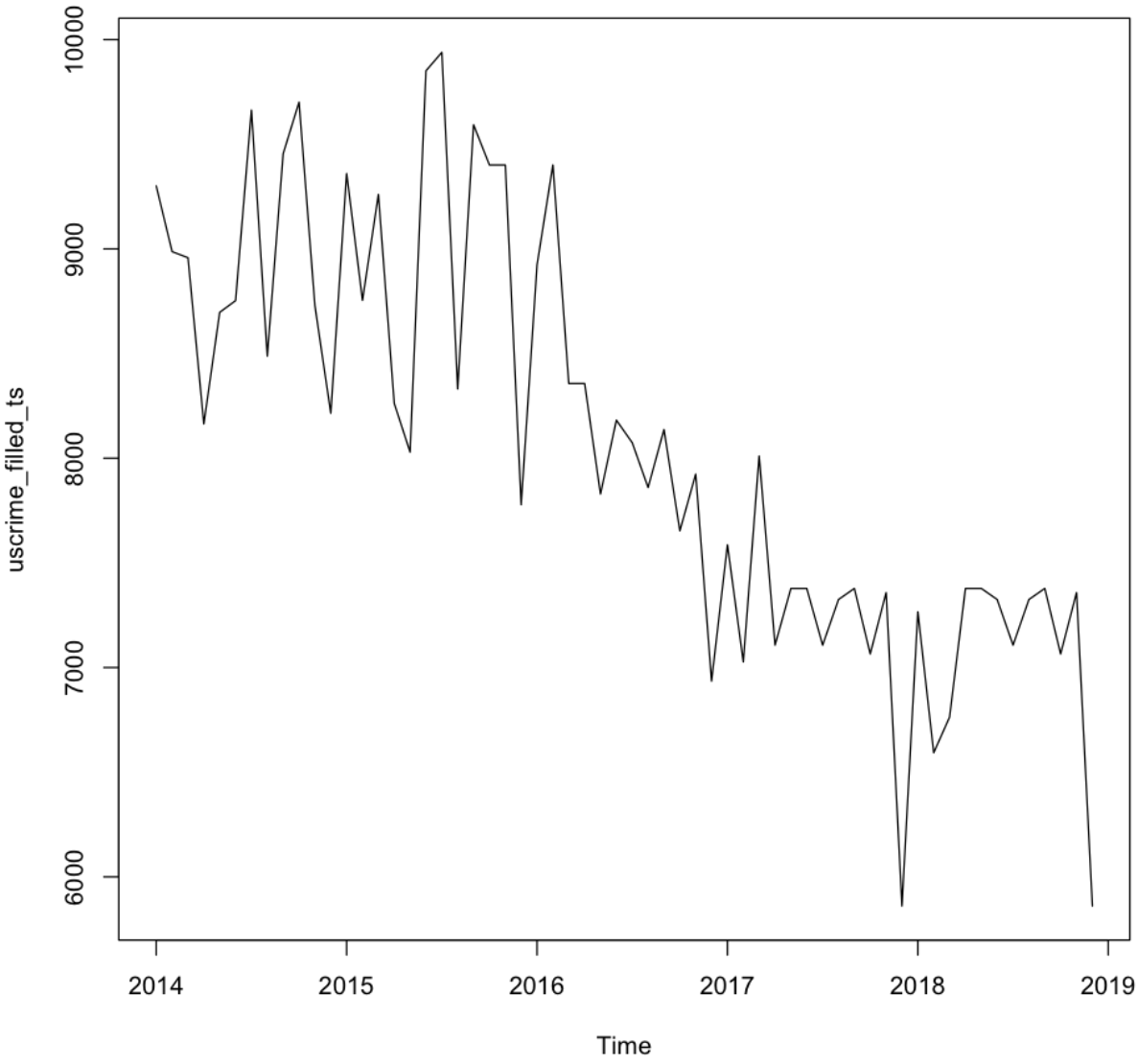
```
1 uscrime_filled_ts <- convert_to_ts(uscrime_df_filled)
2 uscrime_filled_ts
```

A Time Series: 5 × 12

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2014	9301	8987	8958	8164	8697	8753	9663	8488	9455	9701	8729	8215
2015	9360	8754	9260	8262	8029	9851	9939	8331	9593	9401	9401	7778
2016	8924	9401	8357	8357	7829	8182	8074	7860	8137	7653	7924	6935
2017	7586	7027	8011	7107	7378	7378	7107	7325	7378	7065	7358	5861
2018	7266	6593	6763	7378	7378	7325	7107	7325	7378	7065	7358	5861

7.0.3.4. Filled Time Series Data Plot

Now, the missing values have been filled and we can see that our data has gotten smoother without any inconsistencies.

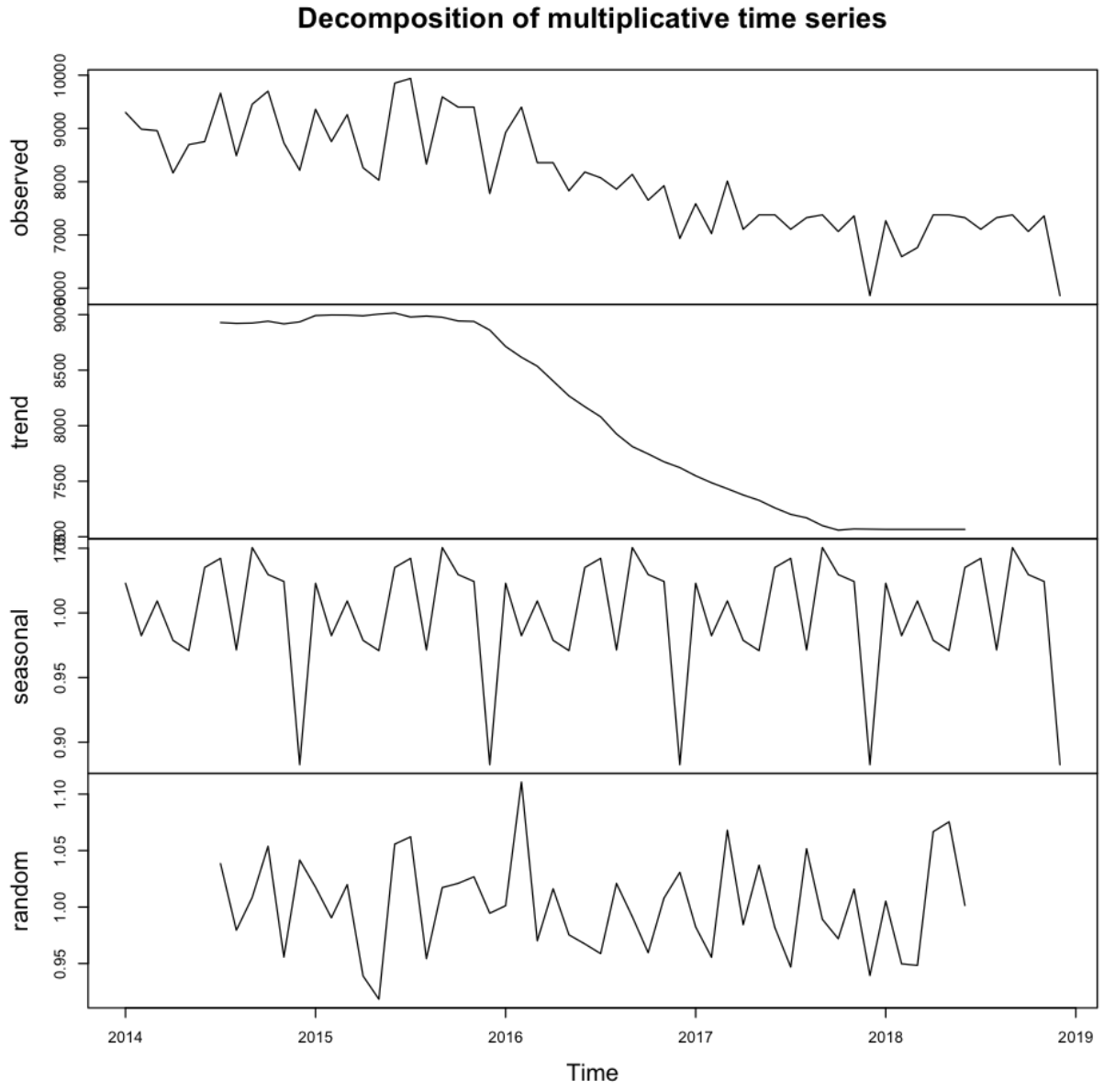


7.0.3.5. Data Patterns Check

Decomposition of multiplicative time series is a method of breaking down a time series into its components, such as trend, seasonality, and residuals. The method is used for time series data that exhibit a multiplicative relationship between the components, meaning that the magnitude of the seasonal and trend components are related to the level of the series.

The decomposition process involves separating the time series into its trend, seasonal and irregular components. The trend component captures the long-term changes in the data, the seasonal component captures the repetitive patterns that occur at specific time

intervals, and the irregular component captures the random and unpredictable fluctuations in the data.



7.0.3.6. Diagnostic Check

In order to apply forecasts, we first need to perform and pass some diagnostic tests which would decide whether we can forecast or not.

7.0.3.6.1. Residuals

A residual matrix, also known as the residuals array, is a matrix that contains the difference between the observed values and the predicted values of a model. It is used to evaluate the performance of a model and to identify patterns and trends in the residuals.

Diagnostic Test

```
usmodel_fit_resd = residuals(usmodel_fit)
usmodel_fit_resd
```

A Time Series: 5 × 12

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov
2014	9.3009866	-184.6590836	-171.7426935	-768.3397151	-236.9017128	76.1207517	910.2567203	-143.3026945	364.0966858	598.6718211	-366.2352251
2015	85.1603686	-97.0125442	395.0224909	-153.6738397	-802.1243901	1081.7432464	881.5679471	-489.3174699	-32.2586050	-193.1839976	363.6906093
2016	-489.2274222	660.9885490	-531.4383321	-176.4284433	-387.5102290	-959.3864052	-935.1917926	109.7481526	0.6474521	-414.5426023	-122.1532814
2017	-6.1042621	-804.1424117	799.4552977	125.0218371	410.9685260	152.7459052	-205.2884291	89.8826012	54.7137227	-24.5957064	101.8118706
2018	208.9648324	118.9446772	-330.2807739	760.2473357	682.6043658	337.5924582	21.1725225	4.0016333	20.4292888	-122.4085619	8.5315887

7.0.3.6.2. Box-Ljung test

As the p-value here is non-zero for the Box-Ljung test so our model has successfully passed the test.

```
Box.test(usmodel_fit_resd, lag=10,type="Ljung-Box")
```

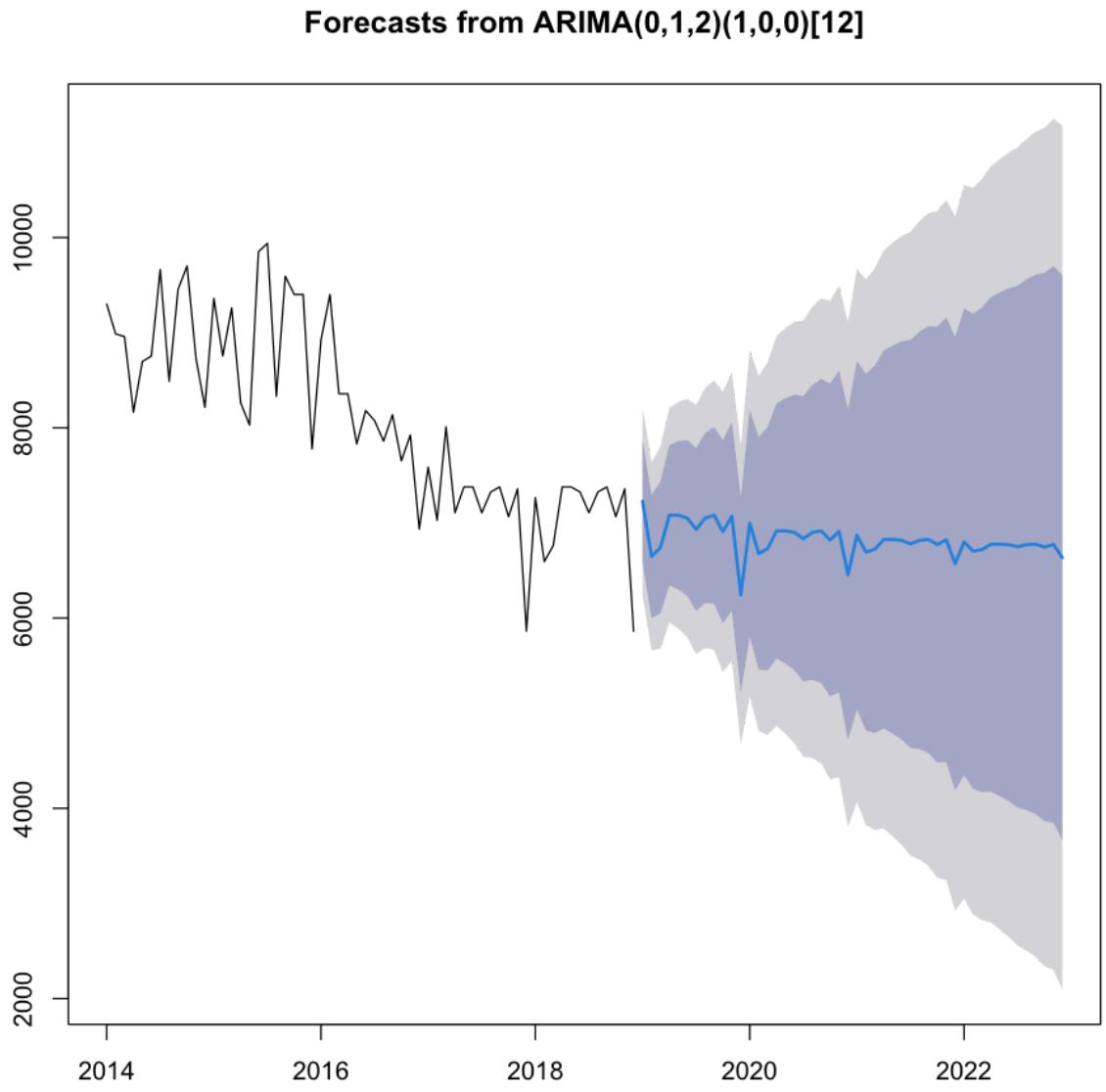
```
        Box-Ljung test

data:  usmodel_fit_resd
X-squared = 5.4063, df = 10, p-value = 0.8624
```

7.0.3.7. Forecasting

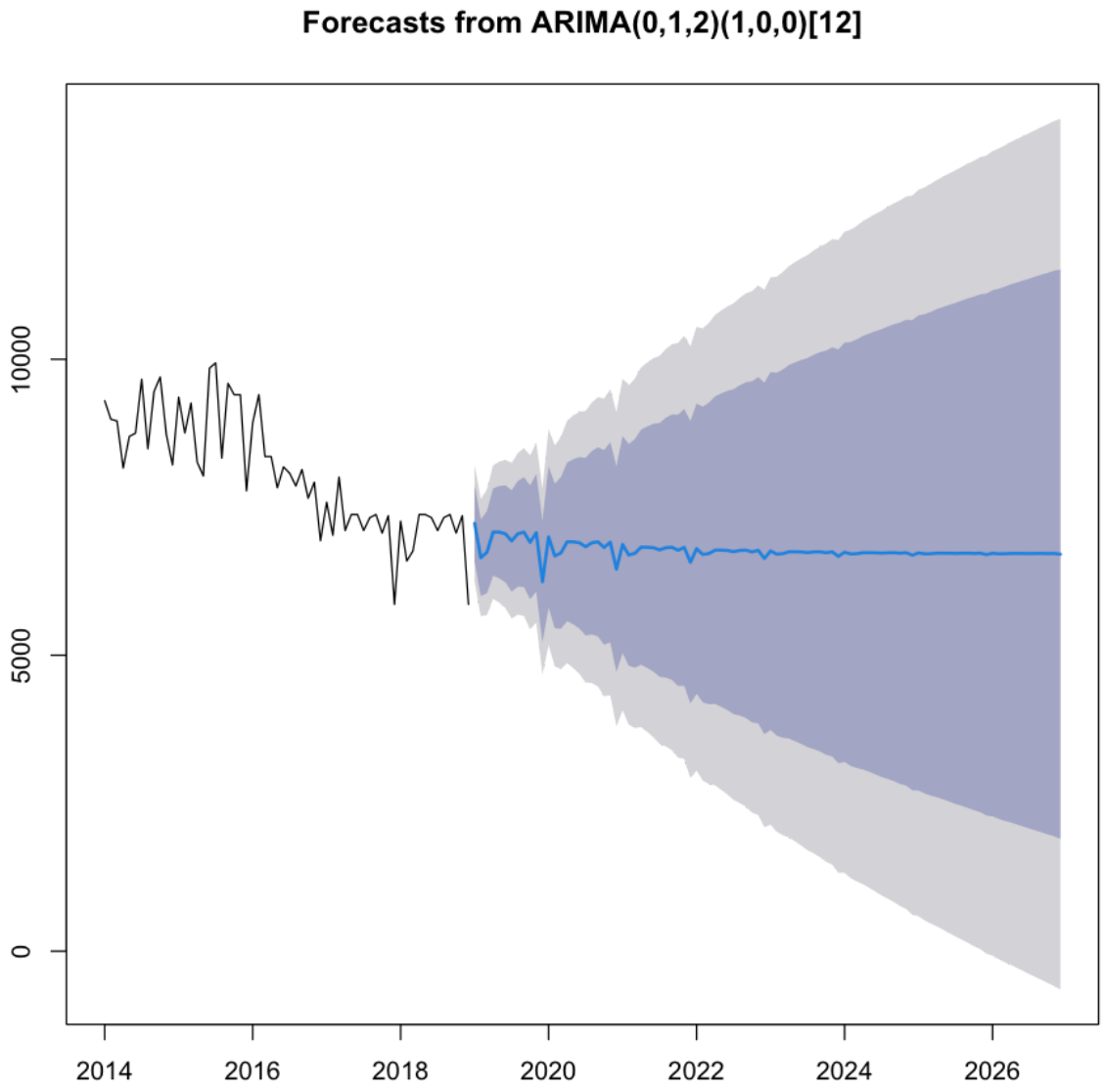
7.0.3.7.1. Years

The four year forecast shows that the crimes wouldn't decrease but they would stay within the range of 7000 which means the current unsuccessful crime rate would be maintained.



7.0.3.7.2. Years

The 8 year forecast shows that the crimes would eventually stay within the range of 7000 which means the current unsuccessful crime rate would be maintained.



7.0.4. Joint Code

7.0.4.0.1. Check and Add Missing Rows with NA

The function finds whether there are any missing months in the dataset and if so then it adds a new row by assigning NA to it.

```
check_and_add_missing_rows <- function(dataframe){
  dataframe <- dataframe[dataframe$county == "National",]
```

```
  numeric_columns_df <- dplyr::select(dataframe, -c("county", "year",
"month" ,"yearmon", "region"))
  dataframe$no_of_crimes <- rowSums(numeric_columns_df)
  df <- dplyr::select(dataframe, c("year", "month","no_of_crimes"))

  unique_years <- c(2014, 2015, 2016, 2017, 2018)
  months <- c("jan", "feb", "mar", "apr", "may", "jun", "jul", "aug",
"sep", "oct", "nov", "dec")
  for(year in unique_years){
    missing_months <- setdiff(months, df$month[df$year == year])
    for(missing_month in missing_months){
      new_row <- data.frame(year = year, month = missing_month,
no_of_crimes = NA)
      df <- rbind(df, new_row)
    }
  }

  df$date <- as.Date(paste(df$year, df$month, "01", sep = "-"),
"%Y-%b-%d")
  df <- df[order(df$date),]

  return(df)
}
```

7.0.4.0.2. Convert to Time Series

The function converts a dataframe into Time Series format.

```
convert_to_ts <- function(df){
  months <- c("jan", "feb", "mar", "apr", "may", "jun", "jul", "aug",
"sep", "oct", "nov", "dec")
  ts_obj <- ts(df$no_of_crimes, start = c(df$year[1],
match(df$month[1], months)), frequency = 12)

  return(ts_obj)
}
```

8. References

Amrani, Yassine A. 2018. “Random Forest and Support Vector Machine based Hybrid Approach to Sentiment Analysis.” 10.

Cote, Catherine. 2021. “What Is Descriptive Analytics? 5 Examples | HBS Online.” HBS Online. https://online.hbs.edu/blog/post/descriptive-analytics.

“Predictive analytics.” n.d. Wikipedia. Accessed January 16, 2023. https://en.wikipedia.org/wiki/Predictive_analytics.

Steinbach, Michael. 2000. *A Comparison of Document Clustering Techniques*, no. 2000 (May), 20.

Sykes, Alan 0. 1993. *An Introduction to Regression Analysis*, 34.

“Time Series Analysis using Arima Model.” 2021. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/11/performing-time-series-analysis-using-arima-model-in-r/.