# 

**Table of Contents**

[**1. Executive Summary**](#_qgl4f891pcdh) **6**

[**2. Data Overview**](#_y063g79h7i82) **7**

[**3. Data Integration**](#_o2sdp7h96myo) **8**

[3.1. Utilized Code](#_huchcnagvxfh) 8

[3.1.1. get\_all\_files\_from\_directories](#_imgpqnekb6c1) 8

[3.1.2. merge\_files](#_swk659oq8srm) 9

[3.2. Integrated Dataset Shape](#_4zoq5jph46hl) 10

[**4. Data Pre-process**](#_ng2nigqjyj3z) **11**

[4.1. Data Cleaning](#_3d9u0cafa7fz) 11

[4.1.1. Dropping Percentage Columns](#_939pydd9u2fe) 11

[4.1.1.1. Code](#_oxsqvuhmdtb6) 11

[4.1.2. Add & Sort By Date](#_1mqvjcc8uure) 11

[4.1.2.1. Code](#_xzh0p23l72cy) 12

[4.1.3. Shifting Columns](#_pnhn7ted1nst) 12

[4.1.3.1. Code](#_dc5ci630ag7j) 12

[4.1.4. Rename Columns](#_n9cso3bg0y5k) 12

[4.1.4.1. Code](#_g4suc53f02u4) 13

[4.1.5. Remove Special Characters and Convert to Integer](#_k42auig2f6y5) 13

[4.1.5.1. Code](#_quocwybwhjca) 13

[4.1.6. Rename Month Values](#_7y6fkykj9yw5) 14

[4.1.6.1. Code](#_arek3x41v0sr) 14

[4.1.7. Missing Data](#_n7e5fgby3boo) 14

[4.1.7.1. Code](#_eqwjpjd4lhsx) 15

[4.1.8. Label Region Based on County](#_k4bnl1lx5lk6) 15

[4.1.8.1. Code](#_ielsu42qkbq0) 16

[4.1.9. Shift Last Column to 5th Index](#_ipmhklpd0ta8) 17

[4.1.9.1. Code](#_12ohcgd3huem) 17

[4.1.10. Rename Conviction Columns](#_gr79e0m0gt8j) 17

[4.1.10.1. Code](#_gr6fbl3x2sca) 18

[4.2. Dataset Post Cleaning](#_qo96s0kwfo8t) 18

[4.2.1. Glimpse](#_f2p7i28sp5xf) 18

[4.2.2. Visualizing Missing](#_lxh5k2l3g3cl) 19

[4.2.3. Visualizing Data Types](#_n278l4b2b39y) 20

[4.3. Dataset Split](#_9cnwi07zxilw) 20

[4.3.1. Crime Dataset Glimpse](#_wb9szax3x456) 21

[4.3.2. Unsuccessful Crime Dataset Glimpse](#_akwj4qmlilth) 21

[4.4. Code](#_5b010zcbslr5) 21

[**5. Descriptive Analytics**](#_z2sprpez8oun) **23**

[5.1. Attributes Analysis](#_g2wqjevvf90) 23

[5.1.1. Crimes Dataset](#_r2d8bxeajx4) 23

[5.1.1.1. County](#_bk8np1f1w7) 23

[5.1.1.2. Year](#_6fn5mdn7uy2t) 23

[5.1.1.3. Month](#_cmsnht614cy1) 23

[5.1.1.4. Yearmon](#_vyklxtar30bz) 23

[5.1.1.5. Region](#_bj7j9flpmobz) 23

[5.1.1.6. Homicide](#_nrjk3ba8tyet) 24

[5.1.1.7. Offences\_against\_the\_person](#_5bciktqa2sk0) 24

[5.1.1.8. Sexual\_offences](#_syrhgxm3xe0s) 24

[5.1.1.9. Burglary](#_bfmi26wx01m) 24

[5.1.1.10. Robbery](#_d5fyow5lfjqd) 24

[5.1.1.11. Theft\_and\_handling](#_2jfr9lw3nh5t) 24

[5.1.1.12. Fraud\_and\_forgery](#_ayojsujfgmog) 25

[5.1.1.13. Criminal\_damage](#_nh9iji4hyesj) 25

[5.1.1.14. Drugs\_offences](#_n0pfevbw2kh) 25

[5.1.1.15. Public\_order\_offences](#_rpwp8bcmzl4w) 25

[5.1.1.16. All\_other\_offences\_\_excluding\_motoring\_](#_4epd6y3i1sni) 25

[5.1.1.17. Motoring\_offences](#_j6jofdmm3dp7) 25

[5.1.2. Unsuccessful Crimes Dataset](#_wx1zhvalqu4f) 26

[5.1.2.1. County](#_w813zrppde2) 26

[5.1.2.2. Year](#_pojf7k4mxhx5) 26

[5.1.2.3. Month](#_2v8qoom6xmmo) 26

[5.1.2.4. Yearmon](#_8t1kio5exzhf) 26

[5.1.2.5. Region](#_qlk8xfgxfm2w) 26

[5.1.2.6. Homicide\_us](#_qewz0oruid3e) 26

[5.1.2.7. Offences\_against\_the\_person\_us](#_u8c4wux5iu3a) 27

[5.1.2.8. Sexual\_offences\_us](#_kq7fsm5bgd25) 27

[5.1.2.9. Burglary\_us](#_q9td48dx8nty) 27

[5.1.2.10. Robbery\_us](#_yz2ebf7u3mvk) 27

[5.1.2.11. Theft\_and\_handling\_us](#_f0qryi4oitst) 27

[5.1.2.12. Fraud\_and\_forgery\_us](#_pdxqvi9nfwcn) 27

[5.1.2.13. Criminal\_damage\_us](#_suxw2lbdef9i) 28

[5.1.2.14. Drugs\_offences\_us](#_tqfnqzbnni1k) 28

[5.1.2.15. Public\_order\_offences\_us](#_o9f8zbdqufa3) 28

[5.1.2.16. All\_other\_offences\_\_excluding\_motoring\_\_us](#_3sj1qavl37r2) 28

[5.1.2.17. Motoring\_offences\_us](#_lgps1dnrrozy) 28

[5.1.2.18. Admin\_finalised\_us](#_ov2wl92hdy1l) 28

[5.2. Analysis dependent on Regions](#_9uaguuimla3x) 29

[5.2.1. Region & All Types](#_8lmy8qag1t0z) 29

[5.2.1.1. Crimes](#_zdk1q56paqmu) 29

[5.2.1.2. Unsuccessful Crimes](#_ryw1x5hq5rq7) 30

[5.2.1.3. Code](#_1uc4lcmvcu3p) 30

[5.2.2. Region & Sum of All](#_rs9yzesunsn) 31

[5.2.2.1. Crimes](#_409jol547mc1) 31

[5.2.2.2. Unsuccessful Crimes](#_3u9l9xj0lxrk) 31

[5.2.2.3. Code](#_4x5iwdtgdrkz) 31

[5.3. Analysis dependent on Years & Months](#_2poj612l8fuu) 32

[5.3.1. 2014](#_9i5jdseewqua) 32

[5.3.1.1. Crimes](#_l0kvq2r74rie) 32

[5.3.1.2. Unsuccessful Crimes](#_z2wap3myl3r4) 33

[5.3.2. 2015](#_rj5vmiyqkh23) 34

[5.3.2.1. Crimes](#_4hl4db8ucacl) 34

[5.3.2.2. Unsuccessful Crimes](#_xhgqfynjblvh) 35

[5.3.3. 2016](#_4kzlmyy1ra87) 36

[5.3.3.1. Crimes](#_qbsvkl3xh277) 36

[5.3.3.2. Unsuccessful Crimes](#_gra9vw1fjuys) 37

[5.3.4. 2017](#_eqt8jqb02prj) 38

[5.3.4.1. Crimes](#_lb7aw06trecw) 38

[5.3.4.2. Unsuccessful Crimes](#_2uagvsij1t3x) 39

[5.3.5. 2018](#_manuwzbhf6dl) 40

[5.3.5.1. Crimes](#_4g8isa5l6e6g) 40

[5.3.5.2. Unsuccessful Crimes](#_sg39cw72xj5v) 41

[5.3.6. Code](#_cdygc5src5mq) 41

[5.3.6.1. group\_by\_year\_month](#_537vrep2xm9z) 41

[5.3.6.2. plot\_graph](#_eq4meomdjiu5) 42

[5.4. Analysis between Crime Types](#_tg9j37pxg26q) 43

[5.4.1. Correlation](#_cu6ulwsdy9x1) 43

[5.4.1.1. Crimes](#_20kyajk97moe) 43

[5.4.1.2. Unsuccessful Crimes](#_has3eqc8t6pb) 44

[5.4.1.3. Code](#_pjcqtbr9gydp) 44

[5.4.2. Covariance](#_4vdx6k9heddu) 46

[5.4.2.1. Crimes](#_7o2bmlx84py4) 46

[5.4.2.2. Unsuccessful Crimes](#_j6kaf5mij7rc) 47

[5.4.2.3. Code](#_fki3m3m50ccu) 47

[5.5. Trend Analysis for Successful & Unsuccessful Crimes](#_18zb4op7rs1m) 49

[5.5.1. Visualization](#_krrmrly89uwm) 49

[5.5.2. Code](#_ecs87vgu8ri0) 49

[**6. Predictive Analytics**](#_s0bwct5sdzh) **51**

[6.1. Regression](#_31c64sh9tx9v) 51

[6.1.1. Linear Regression](#_1ultevhpnuk) 51

[6.1.2. Multiple Regression](#_hd6r34jpd87d) 51

[6.2. Clustering](#_hnd790d1uikw) 51

[6.2.1. KMeans](#_j1c9hpxremsp) 51

[6.2.2. Hierarchical Clustering](#_cmi8i6mdo3ty) 51

[6.3. Classification](#_d6q3r7g7mxkr) 51

[6.3.1. SVM](#_s8h2f0nusb2l) 51

[6.3.2. Random Forest](#_mq2t9ciywmz0) 51

[**7. Prescriptive Analytics**](#_la6803uw0zws) **52**

[7.1. Time Series Analytics](#_tiot4p9v7oes) 52

[7.1.1. Crime](#_gb1gvoka8tpl) 52

[7.1.2. Un successful Crimes](#_hokay09clcds) 52

[**8. References**](#_xpj9glrszg7s) **53**

# 

# 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.

In this analytics report, data exportation and integration were performed. Subsequently, an intensive cleaning of the desired data structure and methodology for performing a descriptive analysis, predictive analytics and prescriptive with hypothesis testing at each stage of analysis.

# 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**

| 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) } |
| --- |

The above R programming language code creates a function called get\_all\_files\_from\_directories which is used to store a list of files from different directories. It calls the list.files() function, which is used to list all the files in a directory. The pattern argument is used to specify the type of files that are to be listed. In this case, it is set to ".csv" which indicates that only files with the ".csv" extension should be listed. It then stores each of the resulting lists into a hash called 'files' and finally returns this hash.

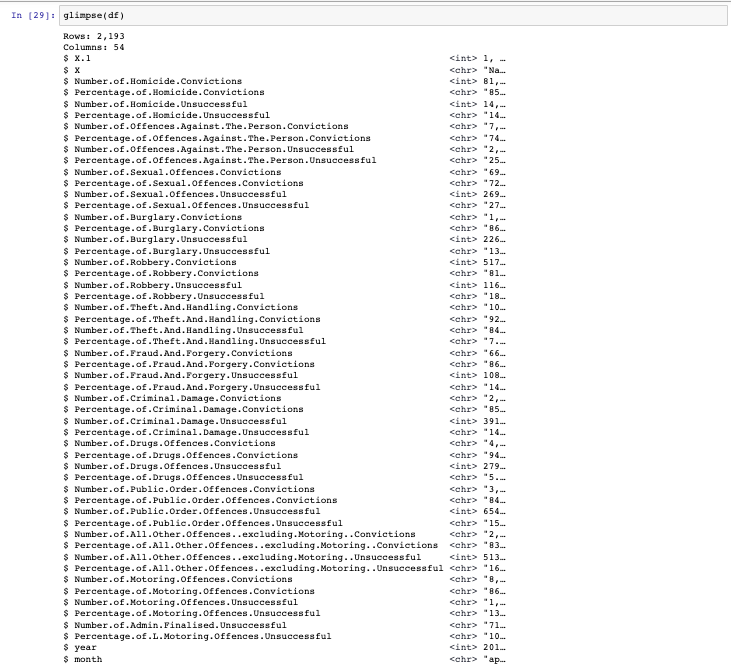
### 3.1.2. merge\_files

| 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) } |
| --- |

The above code is used to merge multiple files into one. The function takes a hash as an argument, which contains the names of the years as keys and the names of the files as values. The function then uses the lapply() function to loop through the years and the files, and the do.call() function to combine the data frames. The read.csv() function is used to read the files, and the gsub() and strsplit() functions are used to extract the month from the file name. Finally, the year and month are added to the data frame and the combined data frame is returned.

## 3.2. Integrated Dataset Shape

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



# 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**

| 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) } |
| --- |

The above code is a function written and it takes a dataframe as an argument. The first line of the function stores the column names of the dataframe in a variable called "col\_names". The second line of the code uses the "grep" function to search for any column names that contain the word "Percentage" and stores the results in a variable called "to\_drop". The third line of the code uses the "%in%" operator to subset the dataframe and remove any columns that are stored in the "to\_drop" variable. Finally, the fourth line of the code returns the modified 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**

| 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) } |
| --- |

The above code is a function that sorts a dataframe by year and month. The function takes a dataframe as an argument and creates a new column called "yearmon" which is a date format of the year and month. The function then orders the dataframe by the newly created "yearmon" column and returns the sorted 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**

| shift\_columns <- function(dataframe){  cols <- colnames(dataframe)  cols <- c(cols[1], cols[(length(cols)-2):length(cols)], cols[2:(length(cols)-3)])  dataframe[, cols] } |
| --- |

The above code is a function that takes a dataframe as an argument. The function first creates a vector of the column names of the dataframe. It then reorders the vector by shifting the last two columns to the beginning of the vector and the second to last three columns to the end of the vector. Finally, the function returns the dataframe with the columns reordered according to the new vector.

### 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**

| rename\_columns <- function(dataframe){  colnames(dataframe) <- gsub("Number.of.", "", colnames(dataframe))  colnames(dataframe) <- gsub("\\.", "\_", colnames(dataframe))  colnames(dataframe) <- tolower(colnames(dataframe))  return(dataframe) } |
| --- |

The above code is a function and it takes a dataframe as an argument. The function uses the gsub() function to remove the string "Number.of." from the column names of the dataframe, and then uses the gsub() function again to replace all periods in the column names with underscores. Finally, the function uses the tolower() function to convert all of the column names to lowercase. The function then returns the modified 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**

| 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) } |
| --- |

The above code is used to remove special characters and convert the dataframe to an integer. The dataframe is modified using the mutate\_all function, which replaces all commas with an empty string. The fifth to last column of the dataframe is then converted to an integer using the sapply function. Finally, the modified dataframe is returned.

### 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.

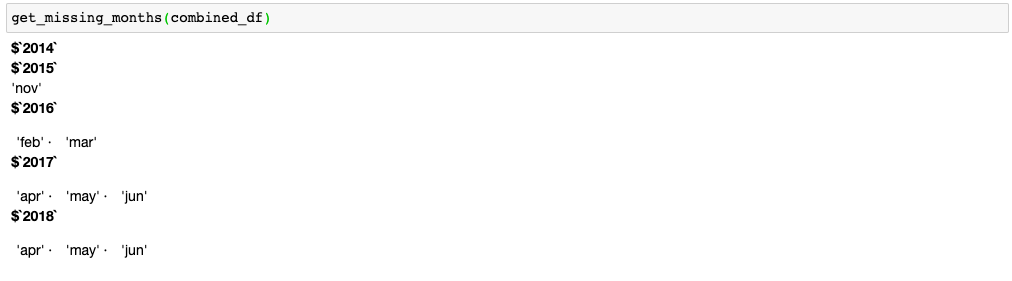


#### **4.1.6.1. Code**

| 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.



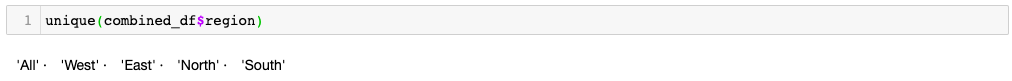
#### **4.1.7.1. Code**

| 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) } |
| --- |

The function above finds all unique years in the dataframe and assigns them to the variable "years". It then initializes an empty list called "missing\_months" which will be used to store the missing months for each year. It then loops through each year, finds all unique months for that year in the dataframe and compares it to a predefined list of all months "all\_months" and finds the difference between the two using the "setdiff" function. Finally, it assigns this difference to an element of the "missing\_months" list corresponding to that year and returns the final list of 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.



| 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**

| 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) } |
| --- |

This function takes in a dataframe "df" and a list "county\_region" as inputs. The function is adding a new column to the dataframe called "region" and populating it based on the "county\_region" list.

It initializes an empty vector called "region" and then loops through each row in the dataframe. For each row, it gets the value of the column "x" and uses it as a key to look up the corresponding value in the "county\_region" list. The value found is then assigned to the corresponding element in the "region" vector. After the loop, it adds this "region" vector as a new column to the input dataframe and returns the modified dataframe as output.

### 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**

| 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) } |
| --- |

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

| 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) } |
| --- |

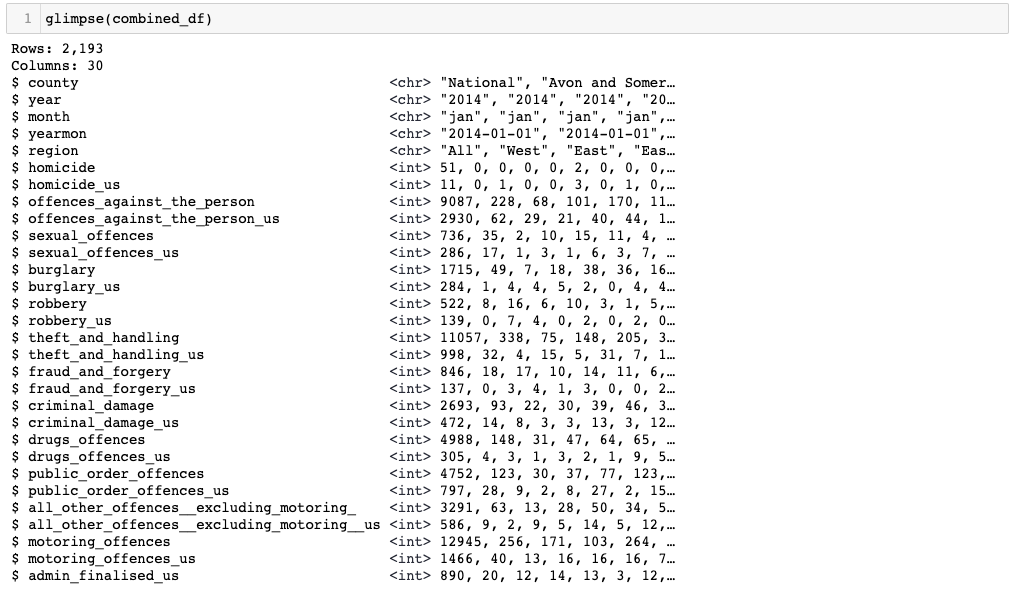
This function renames the columns of a dataframe, removing the "\_convictions" and "\_unsuccessful" substrings from the column names and replacing the latter with "\_us" if it exists.

## **4.2.** Dataset Post Cleaning

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

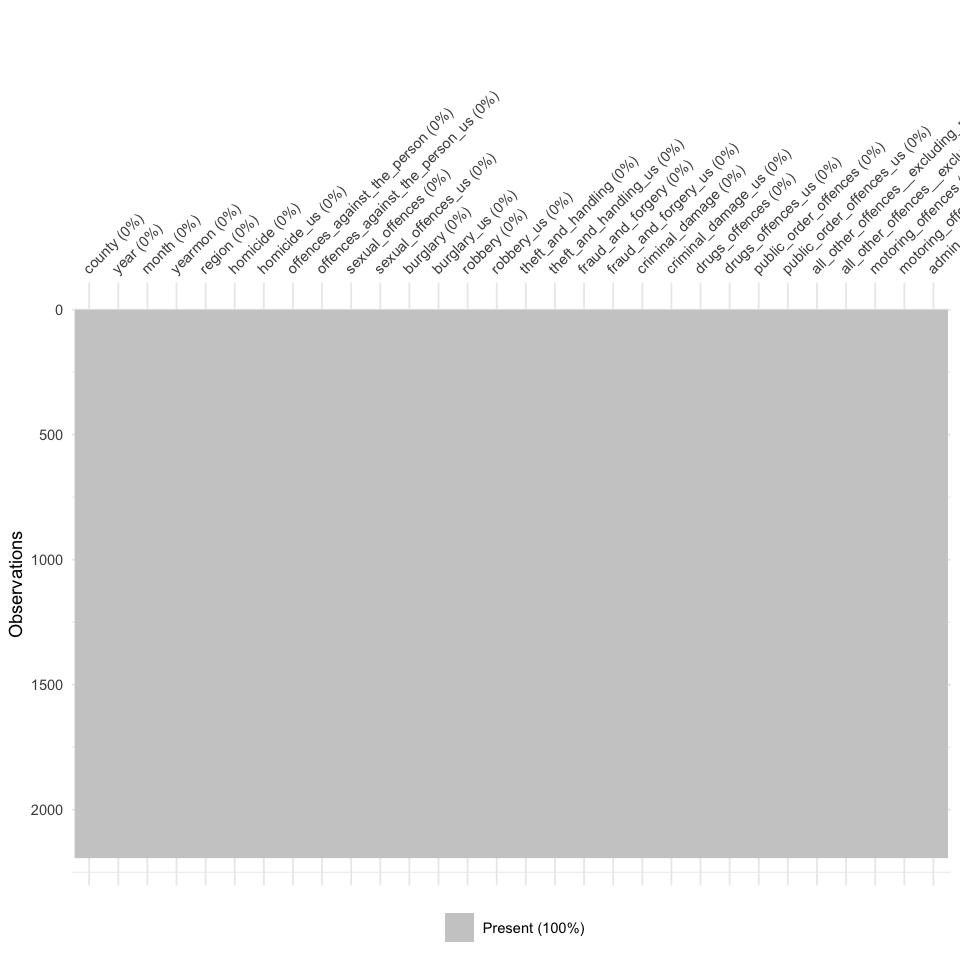
### 4.2.1. Glimpse

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.



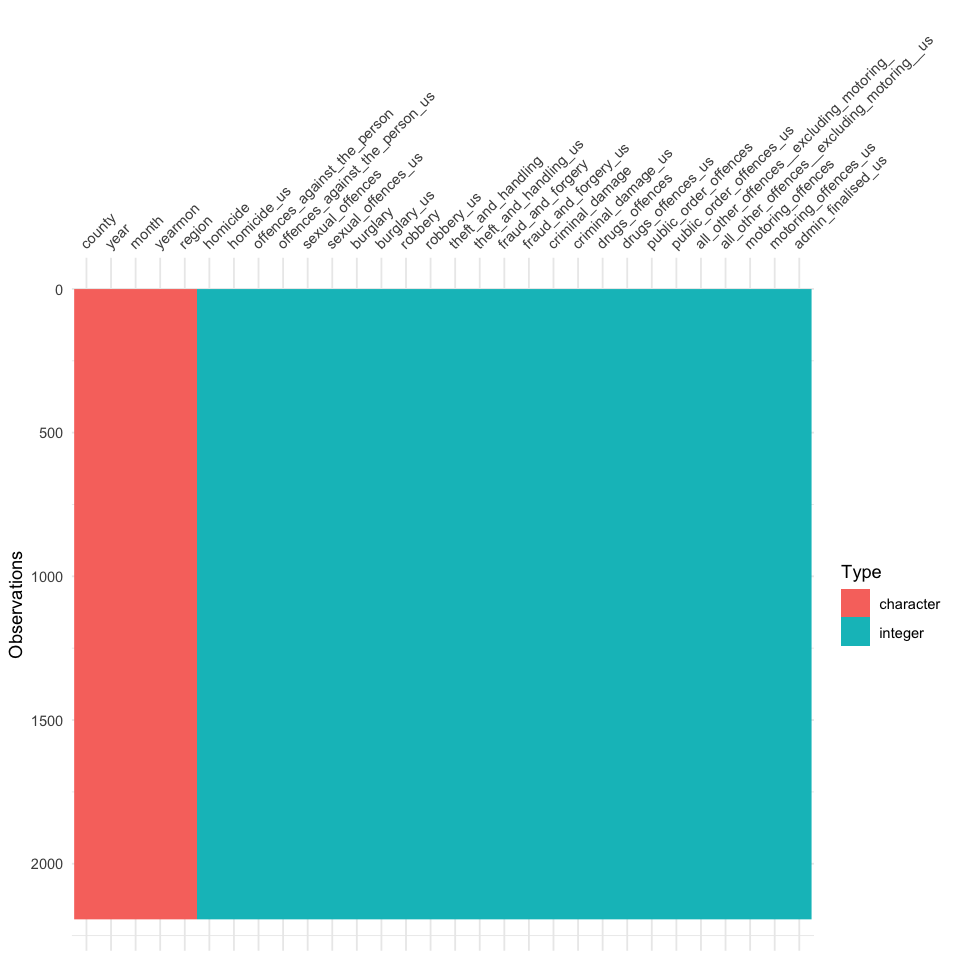
### 4.2.2. Visualizing Missing

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.

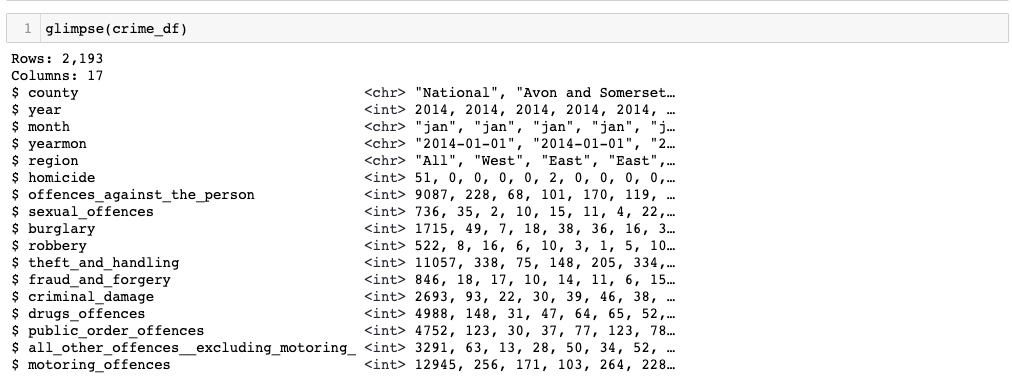


## 4.3. Dataset Split

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.

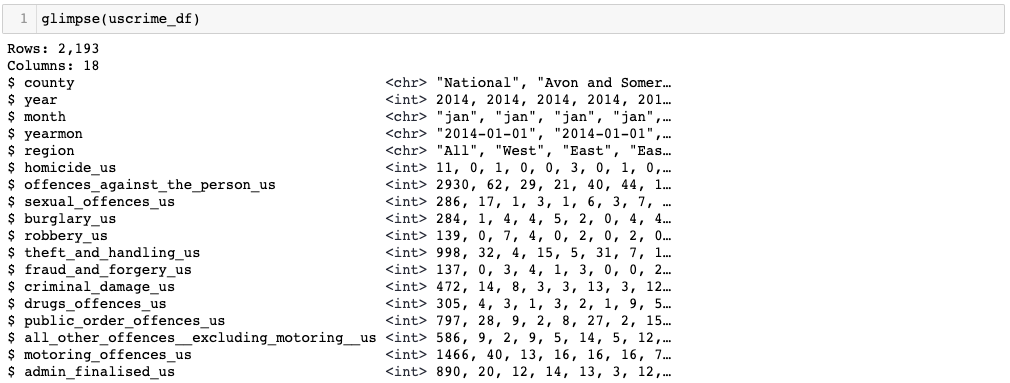
### 4.3.1. Crime Dataset Glimpse

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.



### 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.



## 4.4. Code

| 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)) } |
| --- |

This function takes a dataframe as input and splits it into two dataframes. The first one contain only columns that are not ending with "\_us" and second one contains only columns that are ending with "\_us". The first 5 columns of second dataframe is also included in the second dataframe. Finally, it returns a list containing both dataframes.

### 

# 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.

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

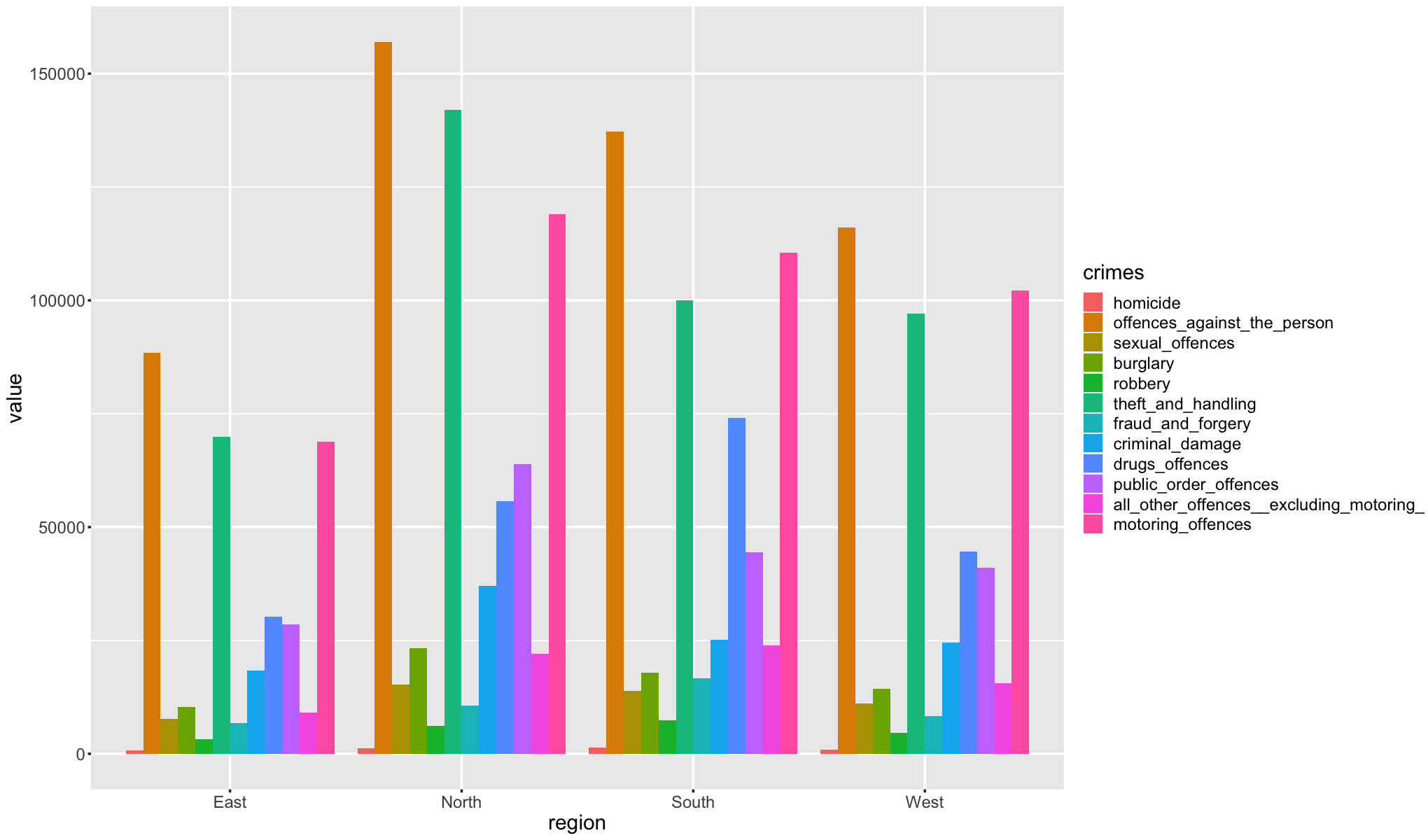
In this section of analysis, we try to understand the relations between crime times and their occurrence in various regions of the englands using graph visualization and statistical analysis.

### 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.

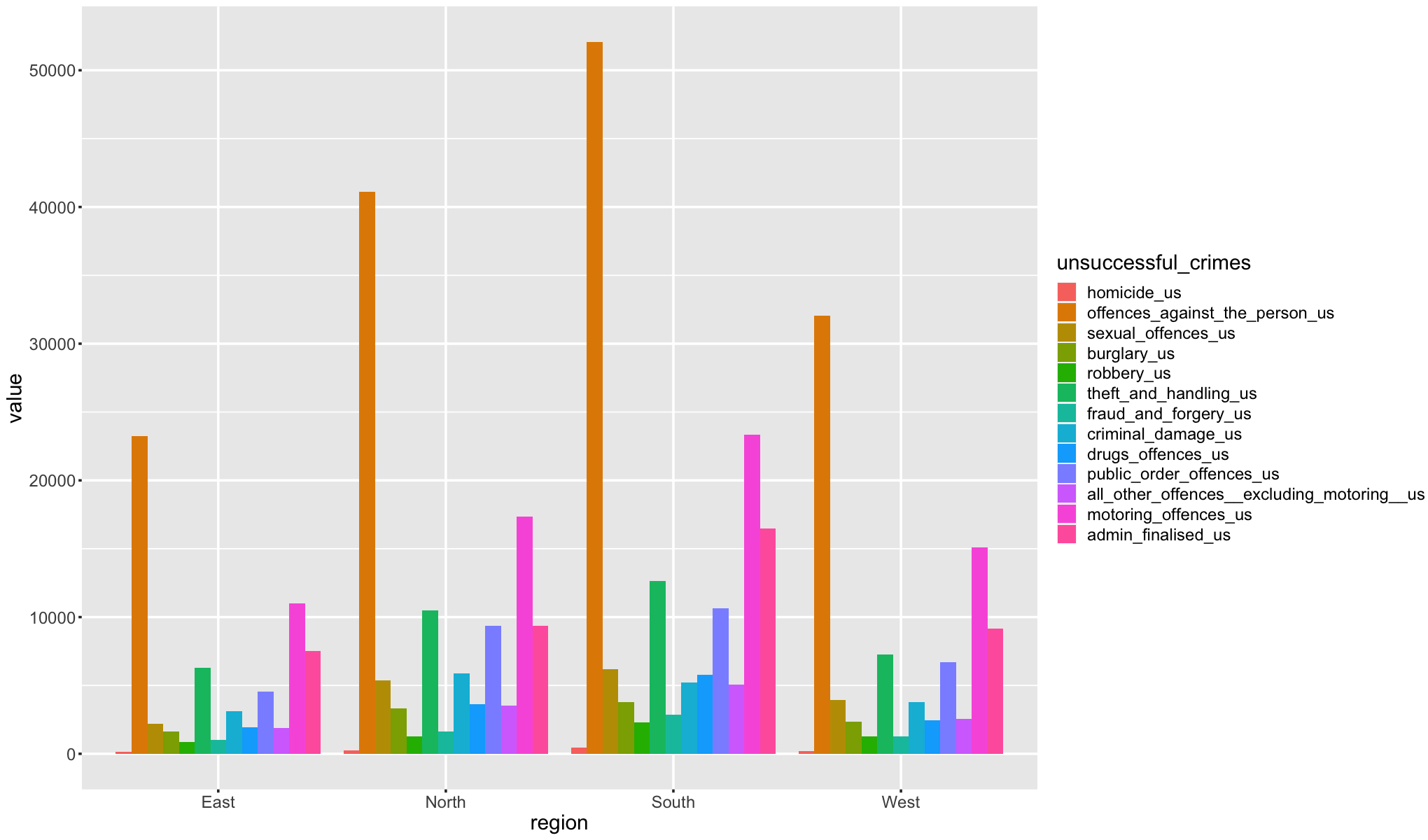
#### 5.2.1.1. Crimes

The graph below considered 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

| 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)) } |
| --- |

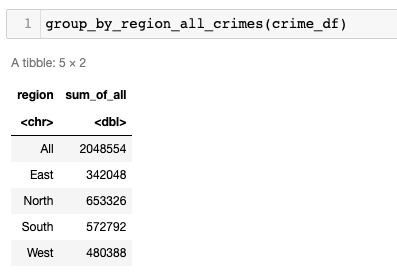
This function takes a dataframe as input and groups it by the "region" column. It removes the first four columns, removes any rows where the "region" column is "All" and then groups the remaining data by the "region" column. Then it applies the sum function to all columns in the dataframe and return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

### 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.

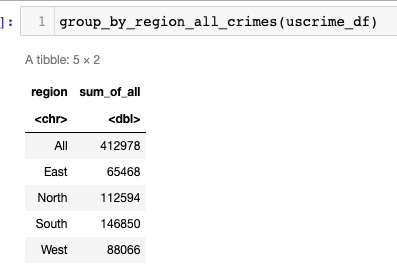
#### 5.2.2.1. Crimes

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.



#### 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.



#### 5.2.2.3. Code

| 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)) } |
| --- |

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.

## 5.3. Analysis dependent on Years & Months

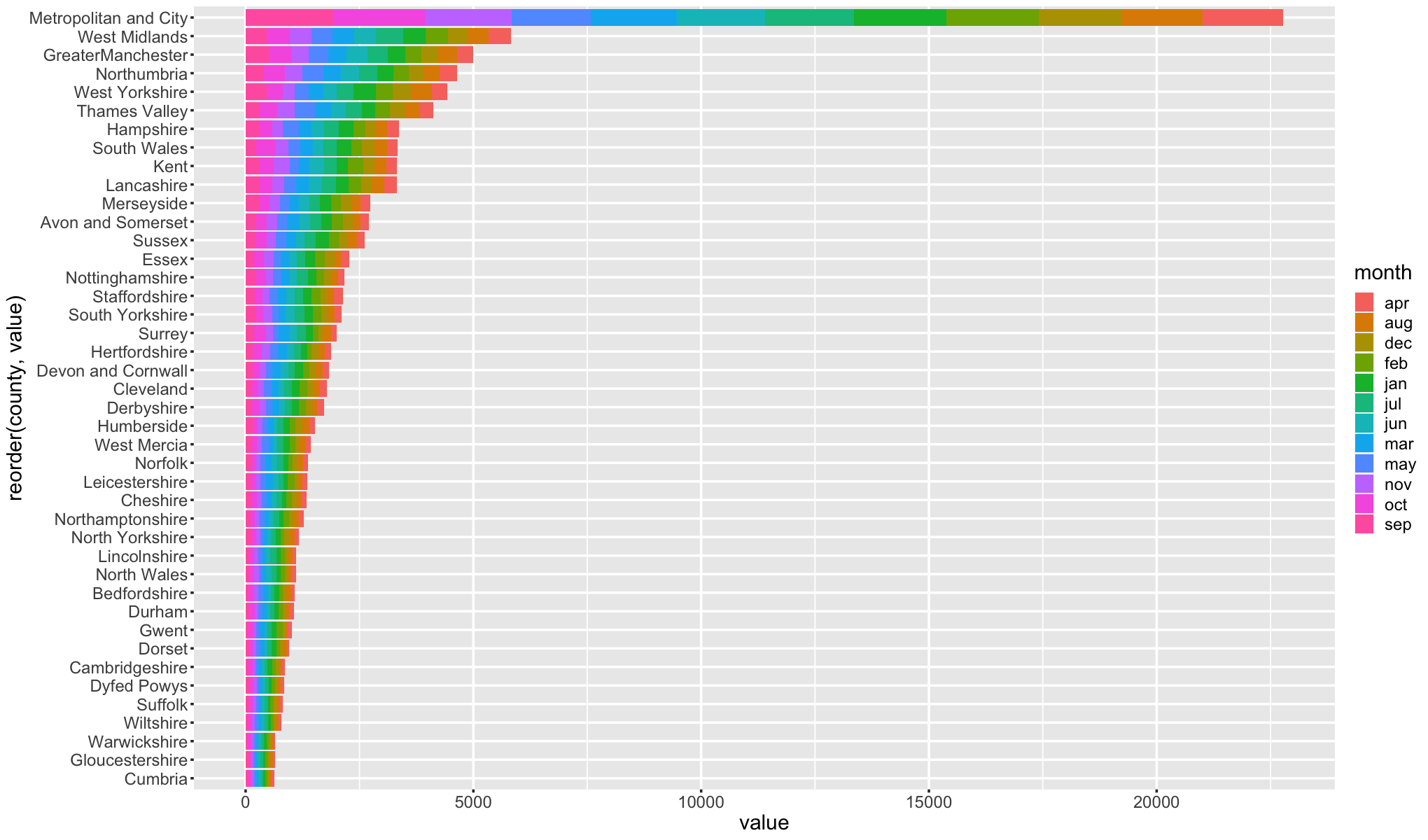
In this section of analysis, we try to understand the relations between successful and unsuccessful crime occurrences in various counties of England with respect to years and months using graph visualization.

### 5.3.1. 2014

#### 5.3.1.1. Crimes

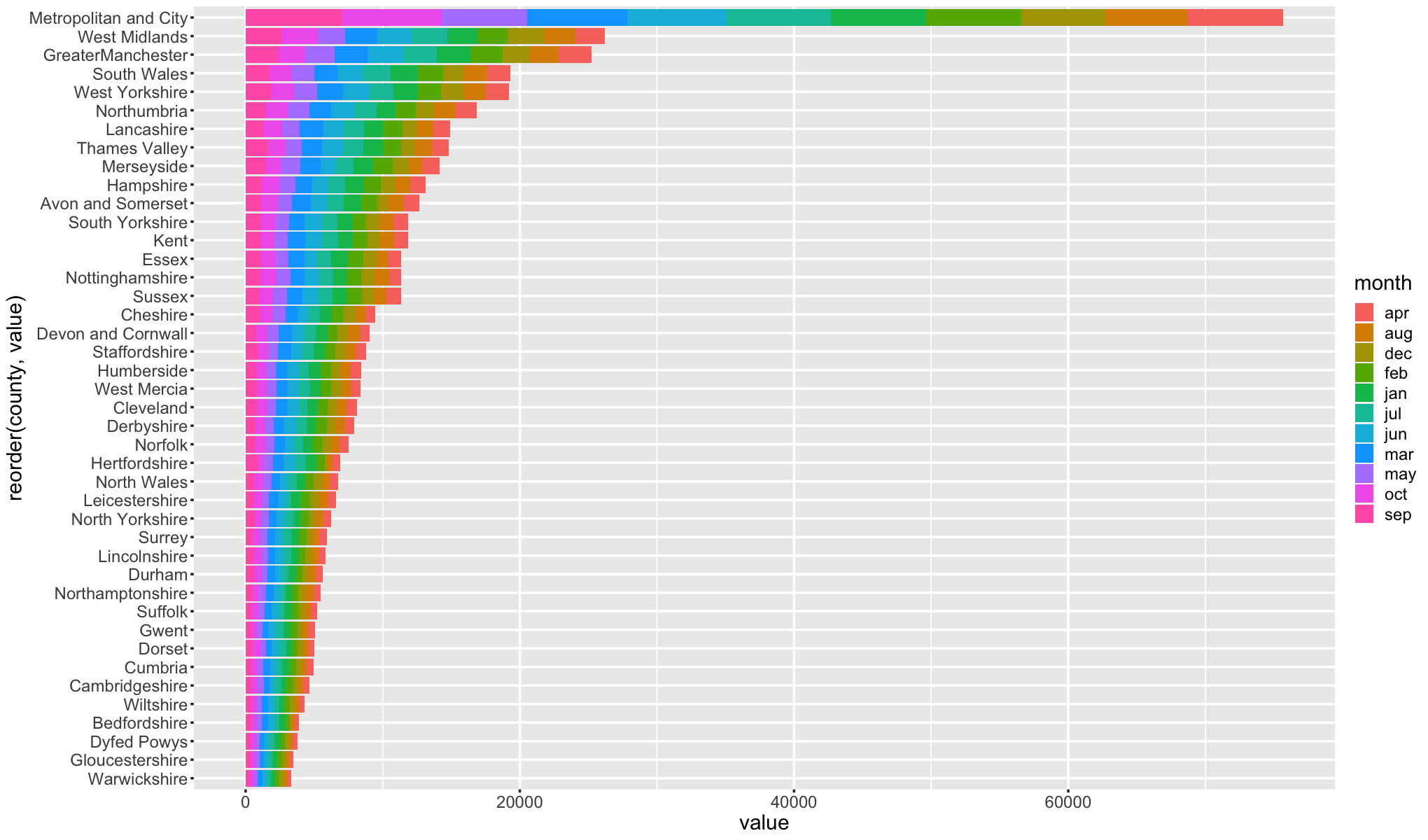


#### 5.3.1.2. Unsuccessful Crimes

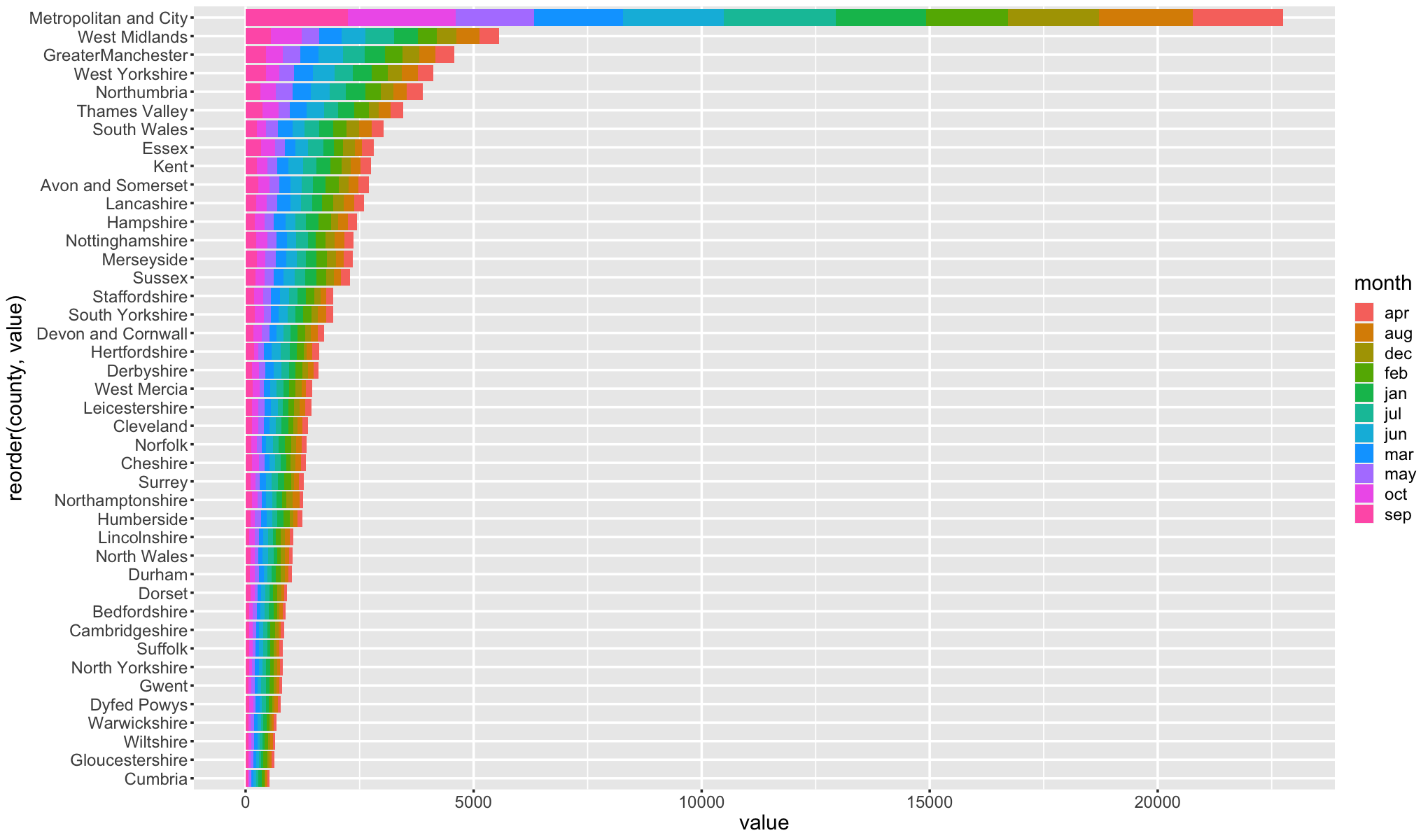


### 5.3.2. 2015

#### 5.3.2.1. Crimes

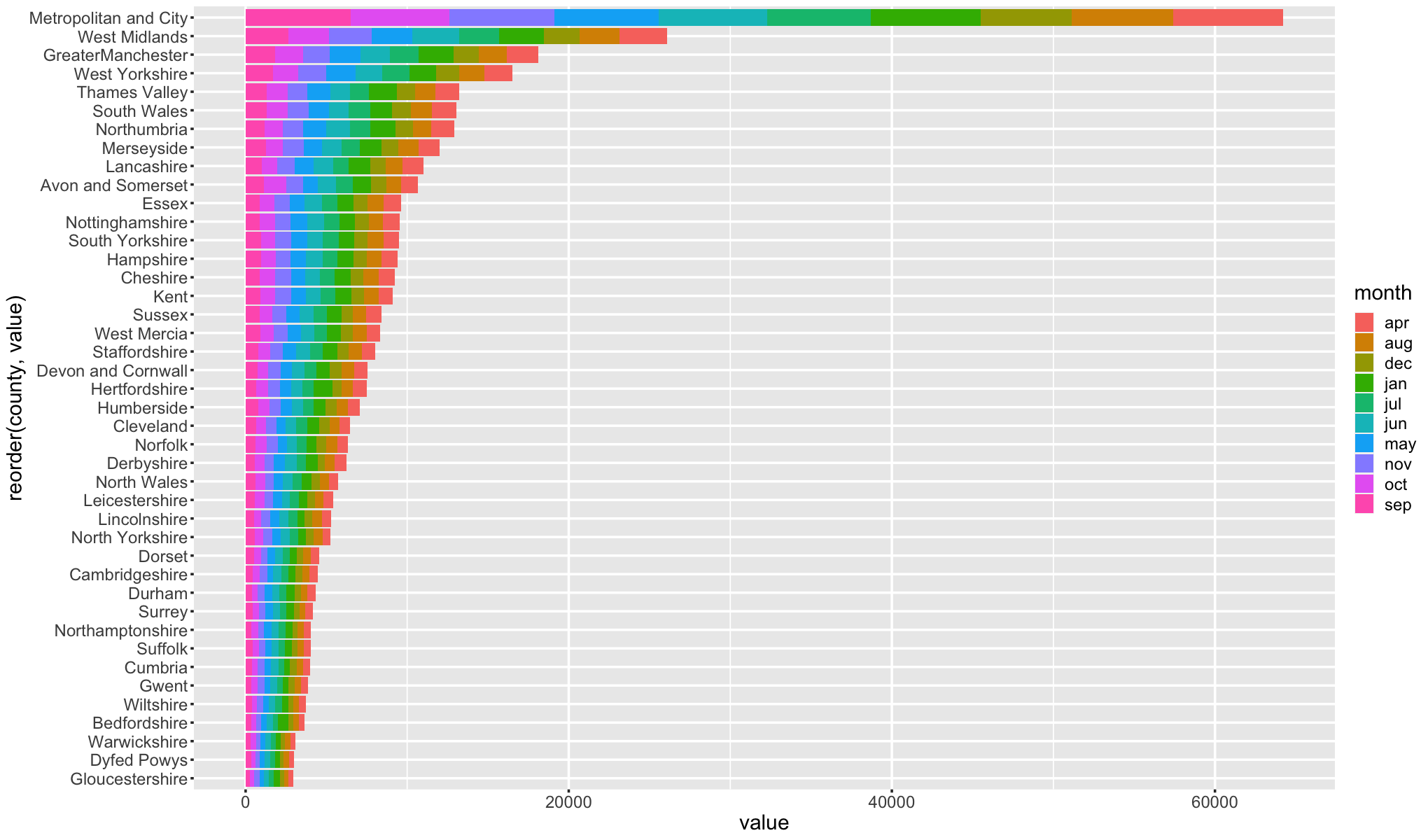


#### 5.3.2.2. Unsuccessful Crimes

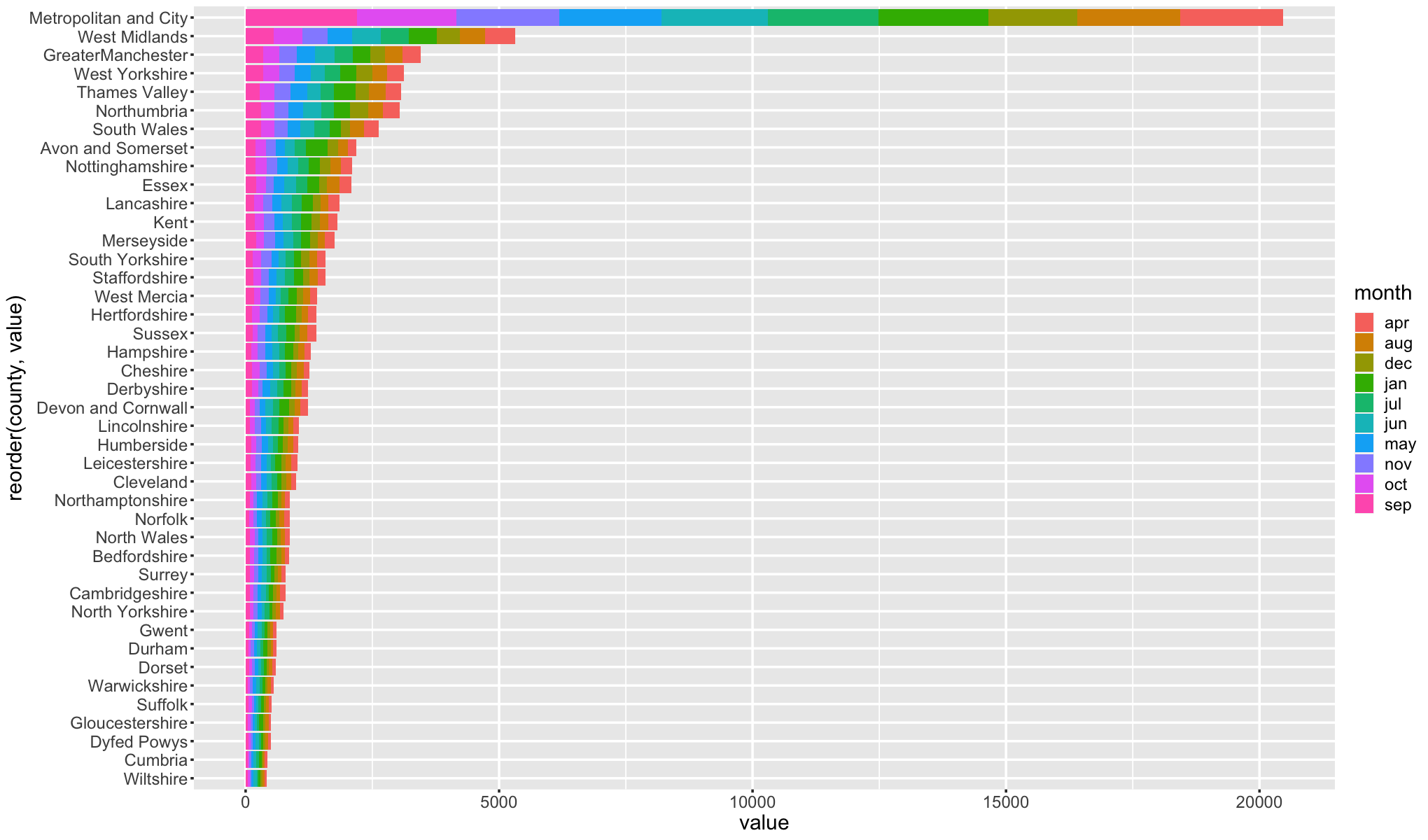


### 5.3.3. 2016

#### 5.3.3.1. Crimes

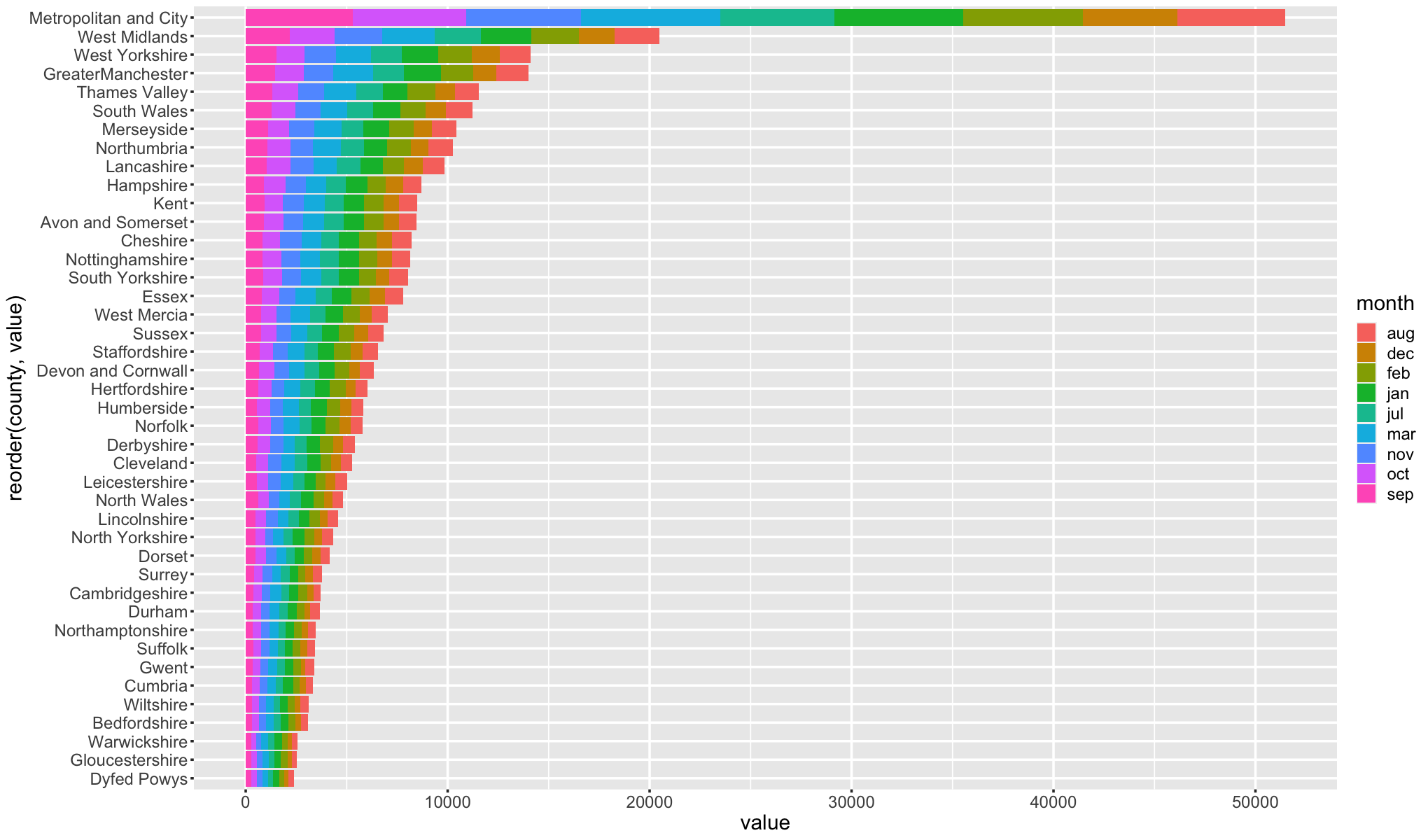


#### 5.3.3.2. Unsuccessful Crimes

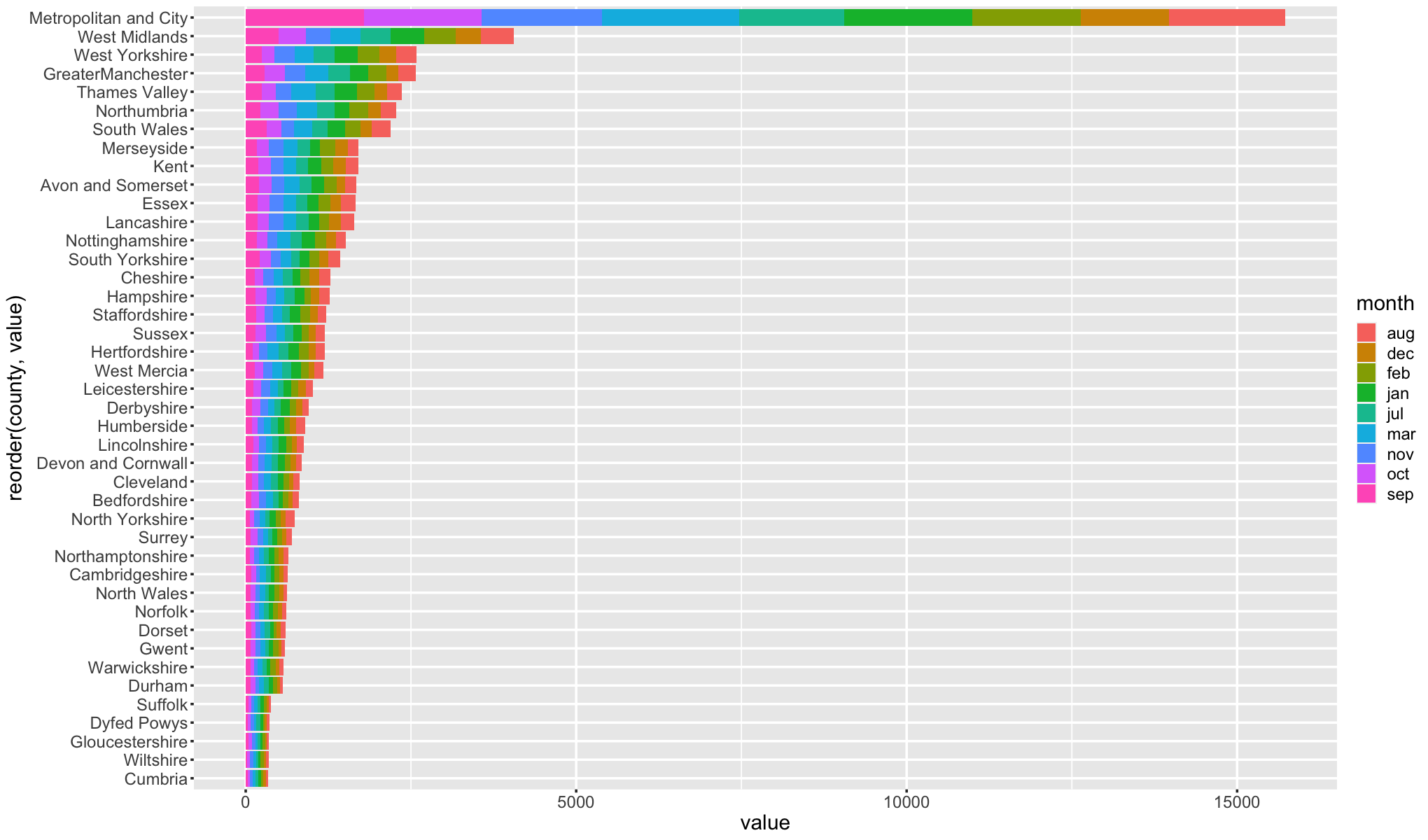


### 5.3.4. 2017

#### 5.3.4.1. Crimes

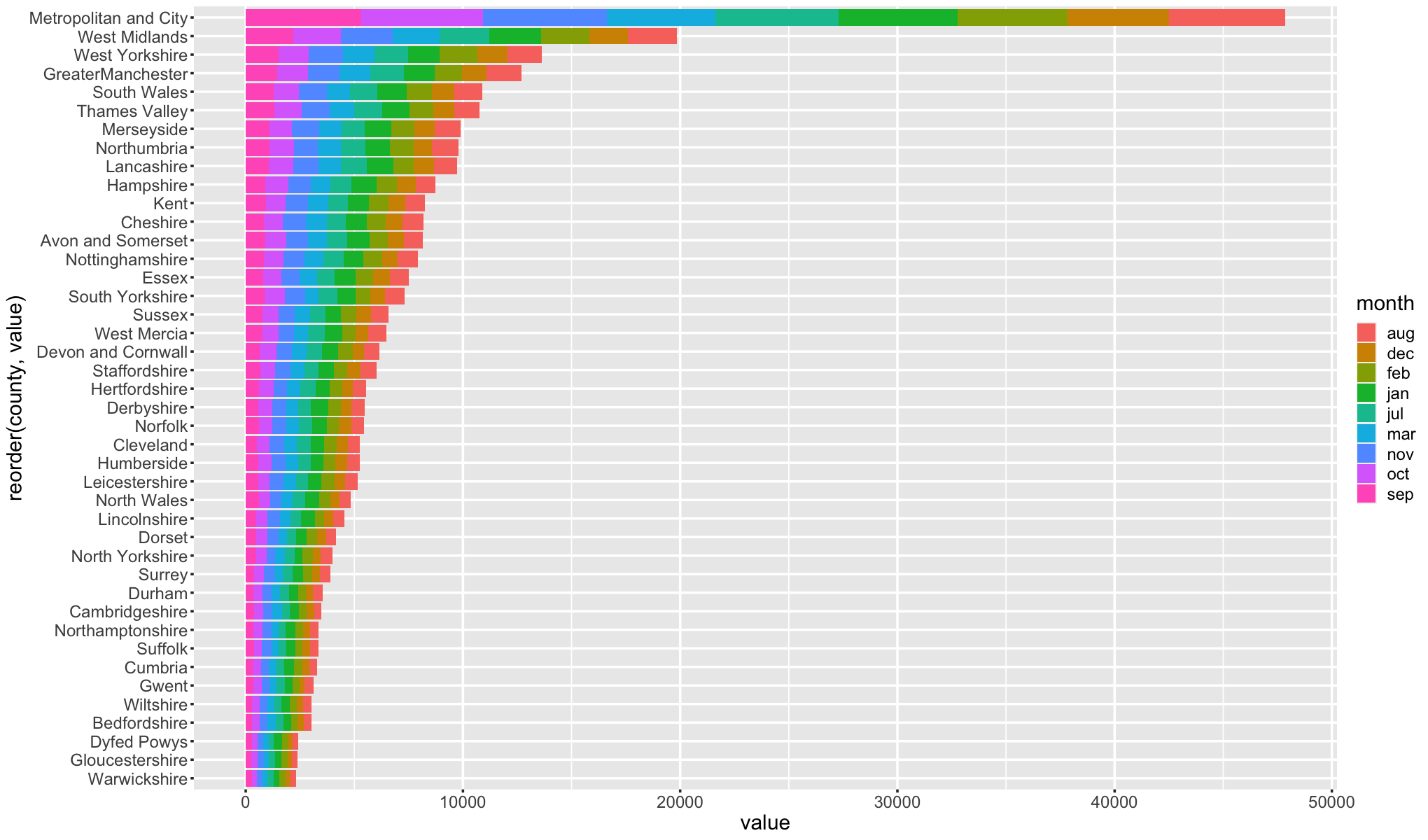


#### 5.3.4.2. Unsuccessful Crimes

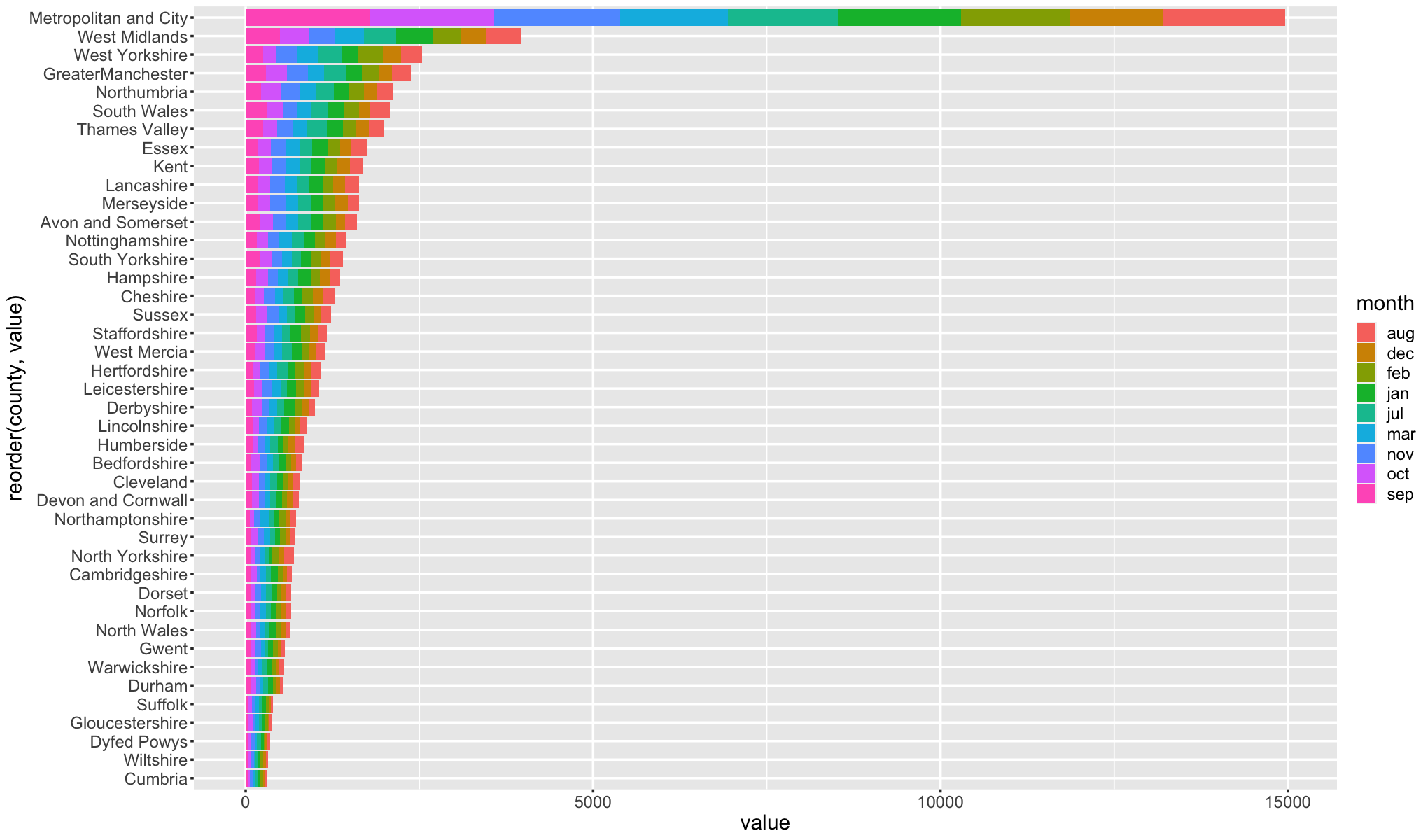


### 5.3.5. 2018

#### 5.3.5.1. Crimes



#### 5.3.5.2. Unsuccessful Crimes



### 5.3.6. Code

#### 5.3.6.1. group\_by\_year\_month

| 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)) } |
| --- |

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.

#### 5.3.6.2. plot\_graph

| 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)) } |
| --- |

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.

## 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

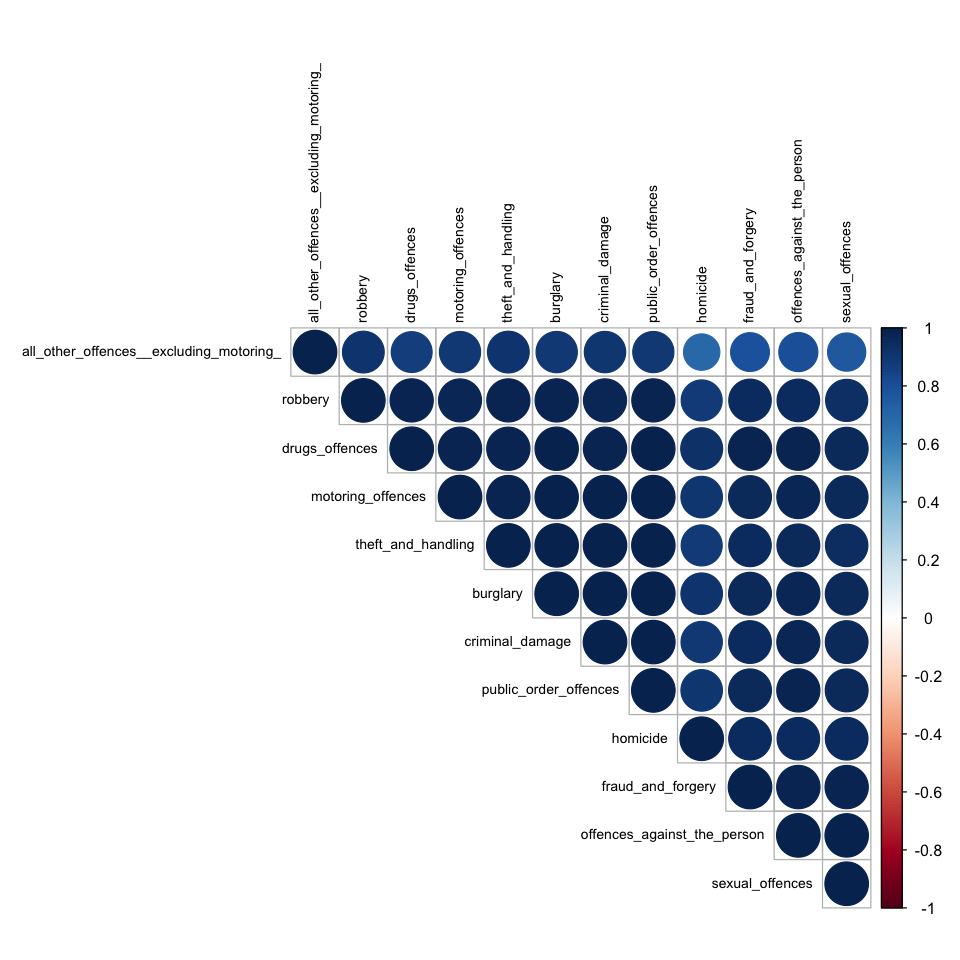
Correlation is a statistical measure that describes the relationship between two or more variables. It is a normalized version of covariance, which means that it scales the relationship between variables in terms of a value between -1 and 1. A value of 1 indicates a perfect positive correlation, meaning 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



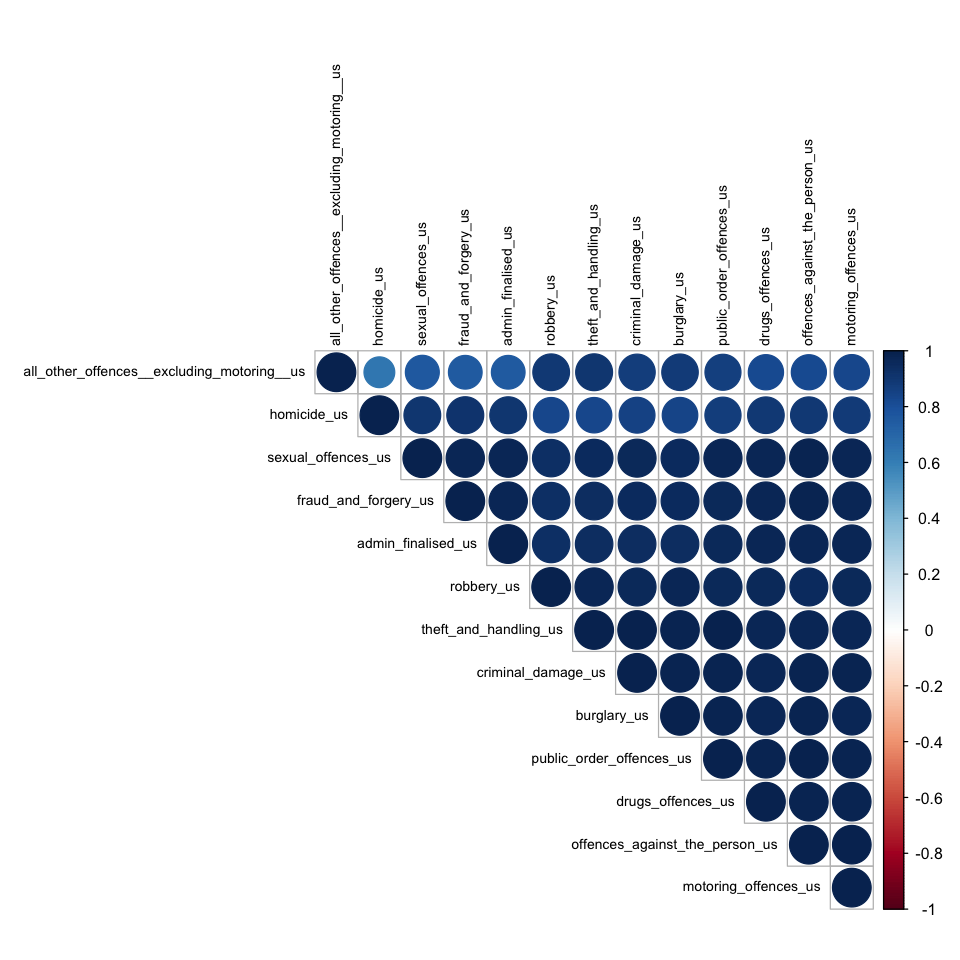
This is a correlation matrix, which shows the correlation coefficients between different types of crimes. The matrix shows the correlation between each type of crime with every other type of crime. Each cell in the matrix represents the correlation coefficient between the two types of crime corresponding 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

| 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) } |
| --- |

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.

### 5.4.2. Covariance

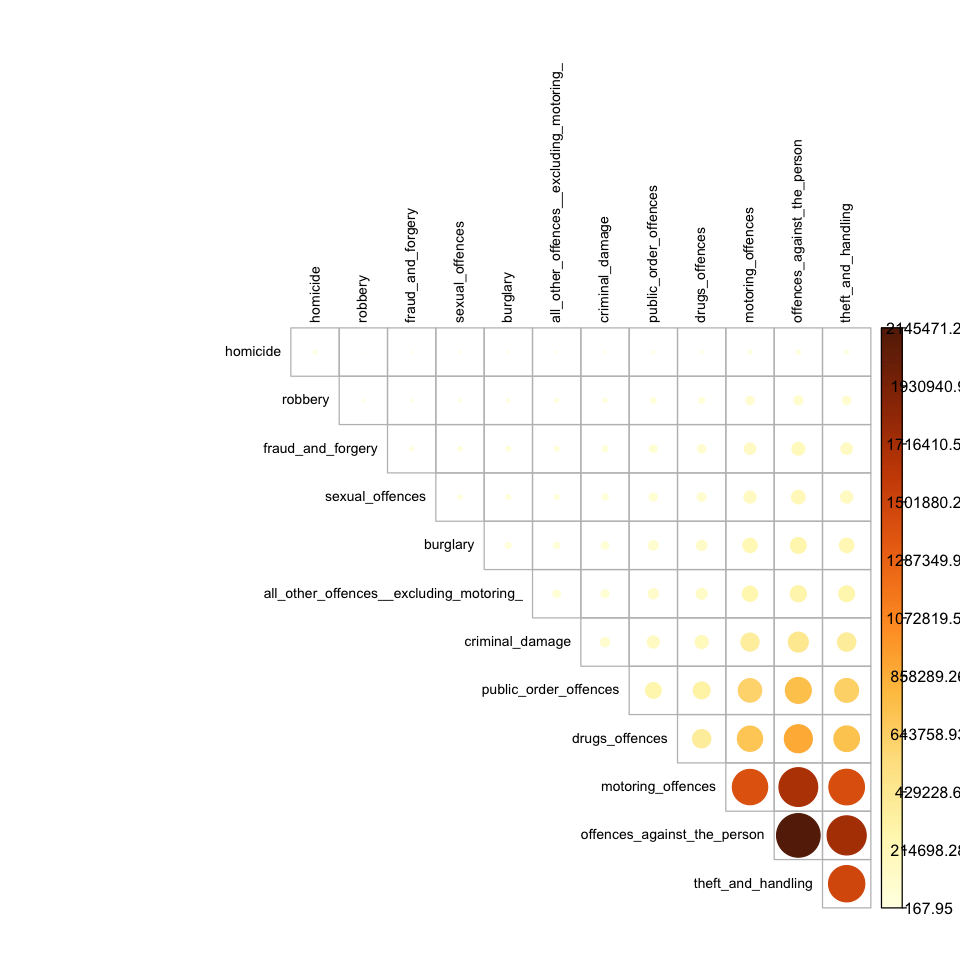
Covariance is a statistical measure that describes the relationship between two or more variables. It measures the degree to which two variables change together, and it can indicate whether the variables have a positive or negative relationship. A positive covariance means that the variables tend to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

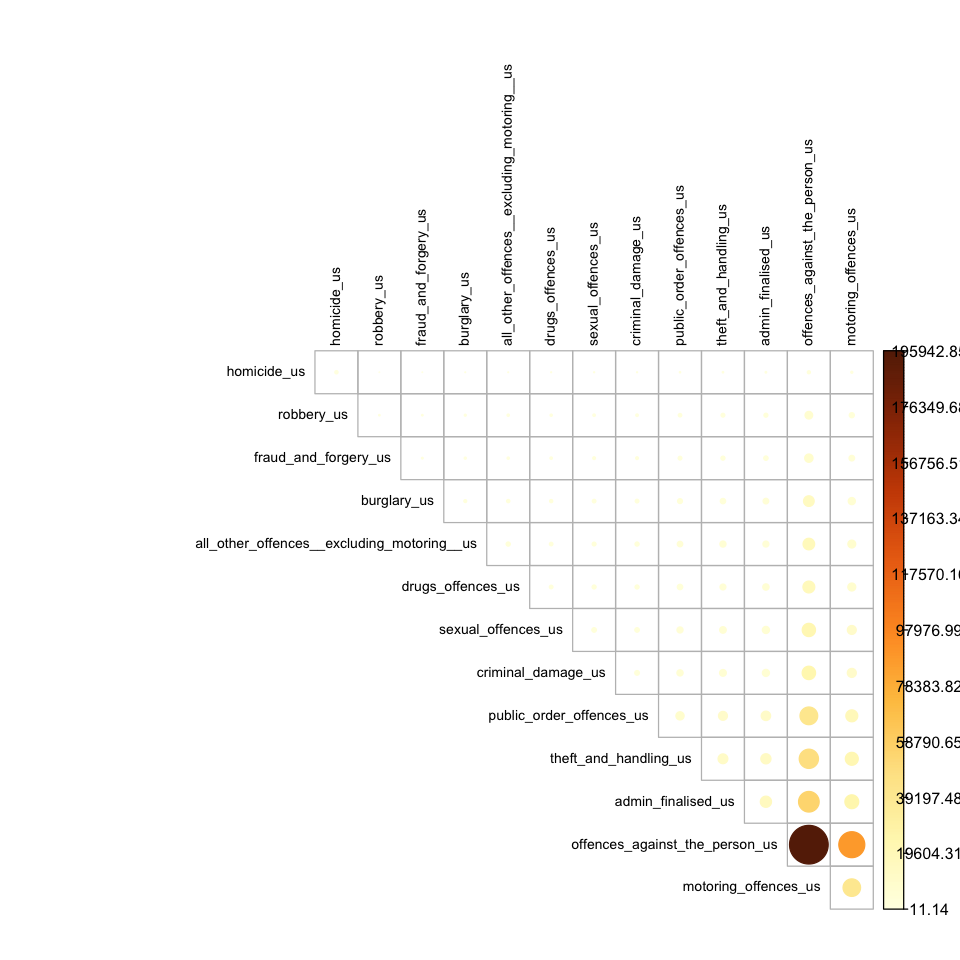


The covariance matrix for crimes is a matrix that shows the relationship between different types of crimes. Each cell in the matrix shows the covariance between two types of crimes. The covariance between two types of crimes is a measure of how much the two types of crimes vary together. A positive covariance 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

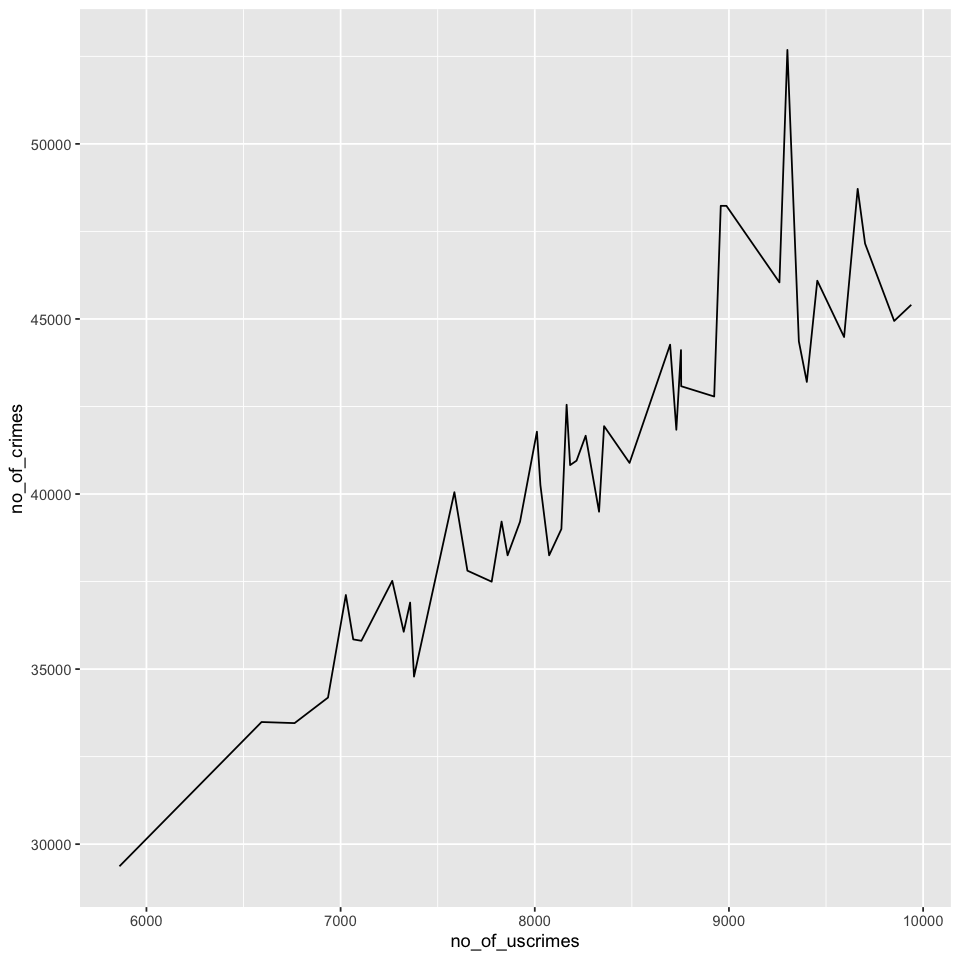
| cov\_matrix\_graph <- function(dataframe){  dataframe <- dplyr::select(dataframe, -c("year", "yearmon", "region"))  num\_cols <- sapply(dataframe, is.numeric)  num\_cols <- names(num\_cols[num\_cols])  cov\_matrix <- cov(dataframe[,num\_cols])  options(repr.plot.width = 8, repr.plot.height =8)  corrplot(cov\_matrix,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

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.

## 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

| 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) } |
| --- |

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.

# 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. 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.

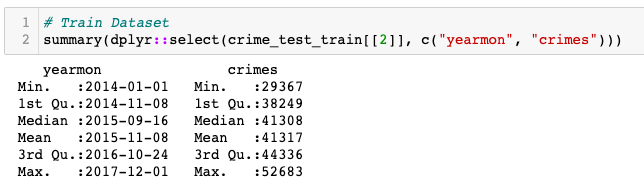
There are several different types of regression, including linear regression, multiple regression, logistic regression, and polynomial regression. We would be employing Linear & Multiple Regression.

### 6.1.1. Linear Regression

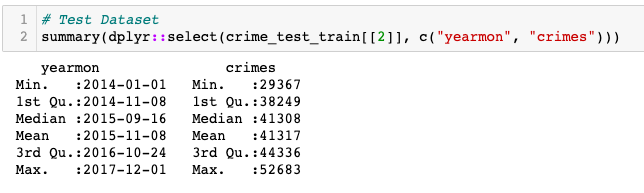
Linear regression is a statistical method that is used to model the relationship between a dependent variable (also known as the outcome or response variable) and one or more independent variables (also known as explanatory or predictor variables). It assumes that there is a linear relationship between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

#### 6.1.1.1. Dataset Summary

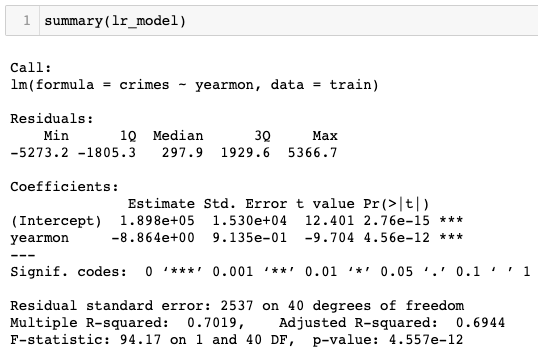
##### 6.1.1.1.1. Train



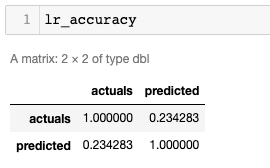
##### 6.1.1.1.2. Test



#### 6.1.1.2. Model Summary



#### 6.1.1.3. Accuracy Matrix



#### 6.1.1.4. Predicted vs Actual Plot



#### 6.1.1.5. Evaluating Stats

| MAE | 2723.76659024003 |
| --- | --- |
| MSE | 9546383.52268908 |
| RMSE | 3089.72224037843 |

#### 6.1.1.6. Code

##### 6.1.1.6.1. Split Data

| 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)) } |
| --- |

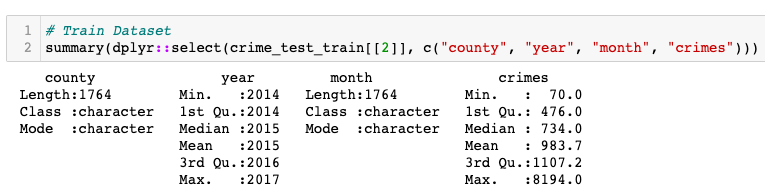
##### 6.1.1.6.2. Linear Regression

| 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)) } |
| --- |

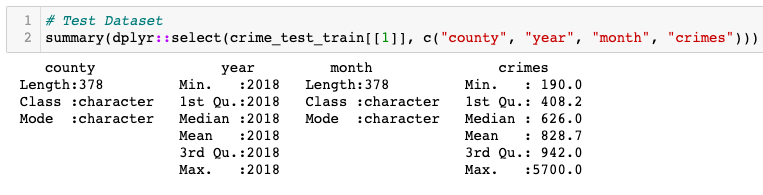
### 6.1.2. Multiple Regression

#### 6.1.2.1. Dataset Summary

##### 6.1.2.1.1. Train



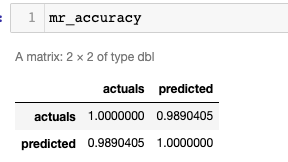
##### 6.1.2.1.2. Test



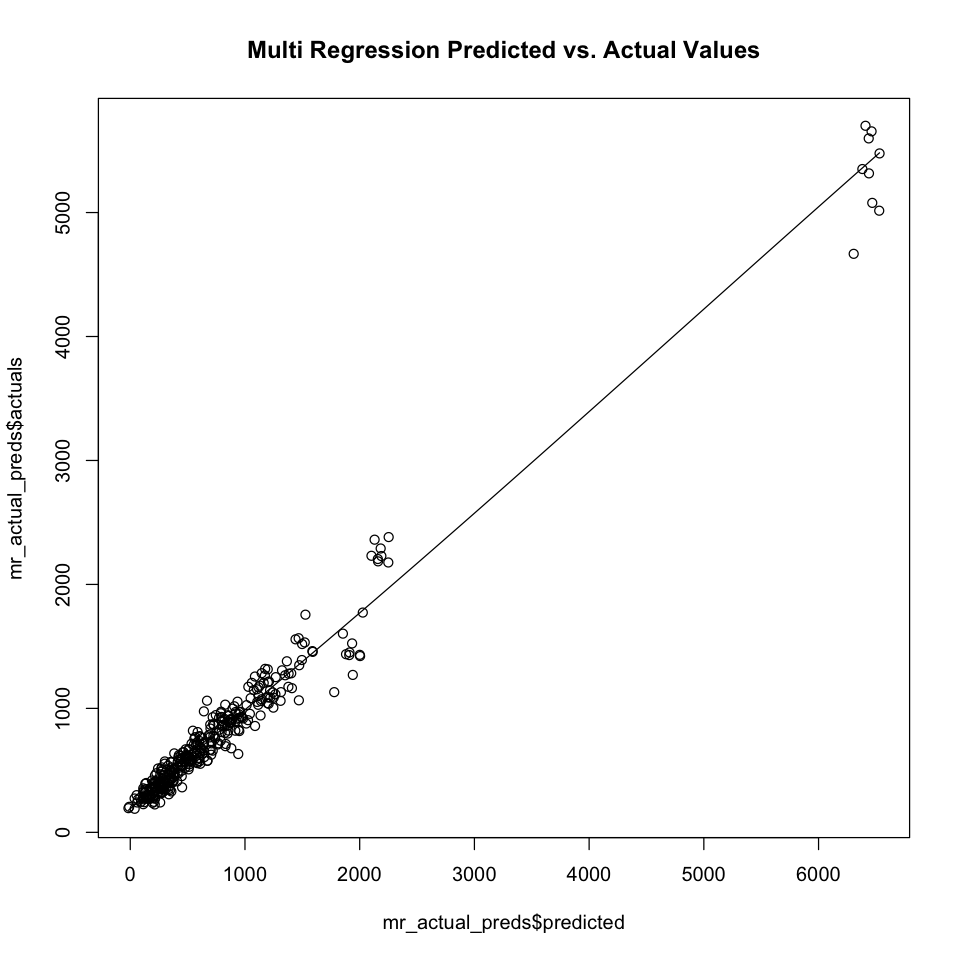
#### 6.1.2.2. Model Summary

| 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 |
| --- |

#### 6.1.2.3. Model Accuracy



#### 6.1.2.4. Actual vs Predicted Plot



#### 6.1.2.5. Evaluating Stats

| MAE | 146.094951776508 |
| --- | --- |
| MSE | 55675.067652928 |
| RMSE | 235.955647639398 |

#### 6.1.2.6. Code

##### 6.1.2.6.1. Split Data

| 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)) } |
| --- |

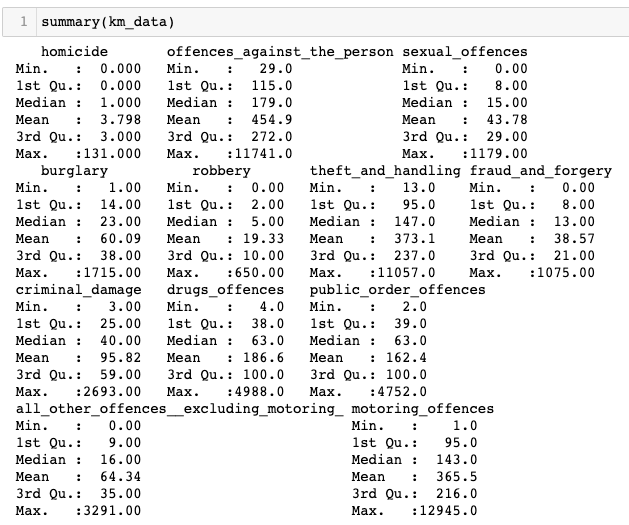
##### 6.1.2.6.2. Multiple Regression

| 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

### 6.2.1. KMeans

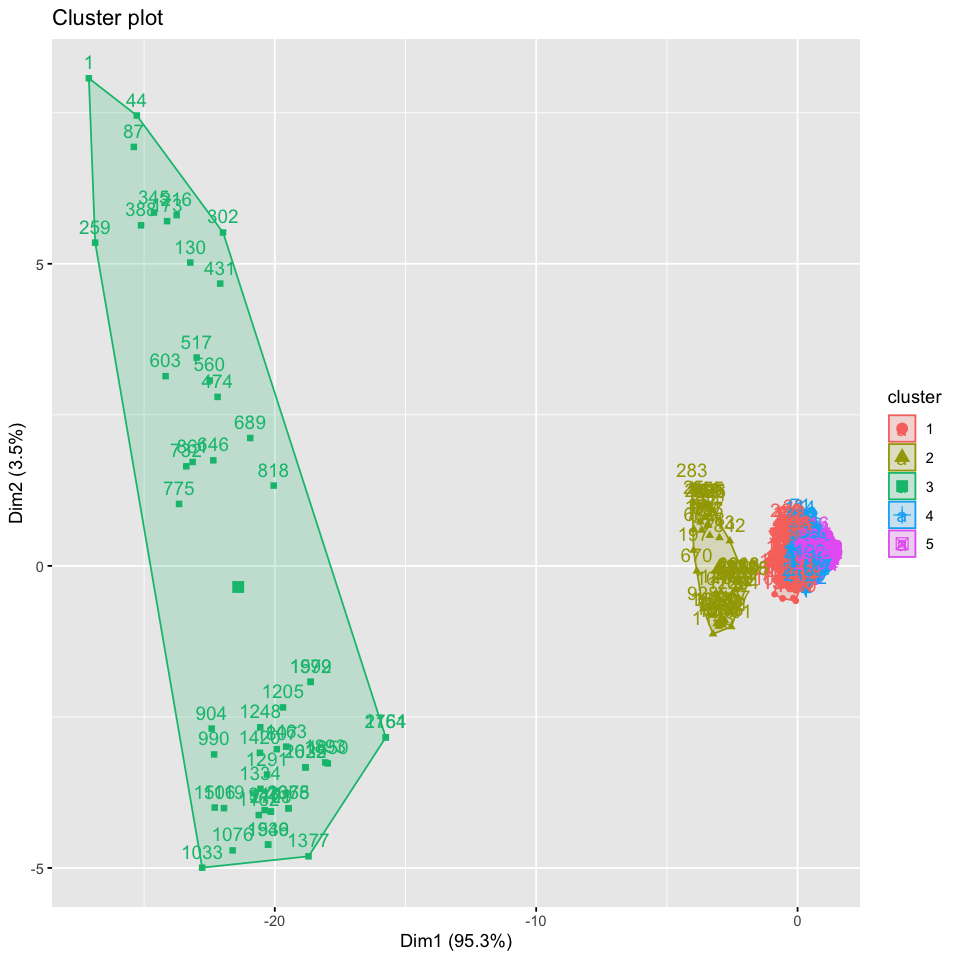
#### 6.2.1.1. Dataset Summary



#### 6.2.1.2. With 4 Clusters



#### 6.2.1.3. With 5 Clusters



#### 6.2.1.4. With 6 Clusters



#### 6.2.1.5. Summary

Text Here

#### 6.2.1.6. Code

##### 6.2.1.6.1. Remove Non Numeric Columns

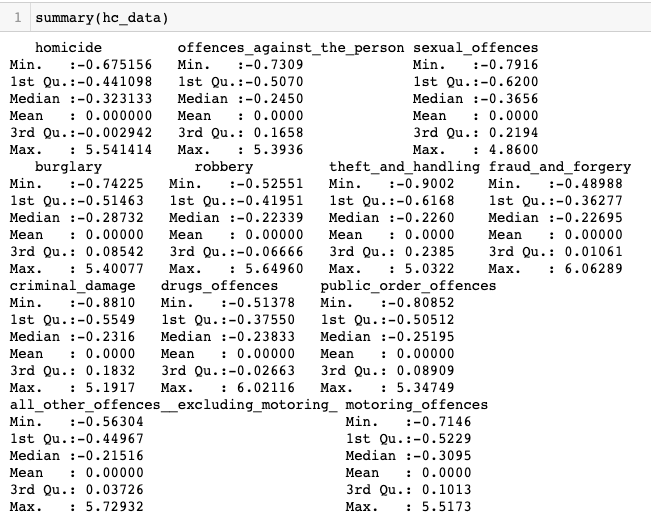
| remove\_non\_numeric\_cols <- function(dataframe) {  dataframe <- dplyr::select(dataframe, -c("county", "year", "month" ,"yearmon", "region"))  return(dataframe) } |
| --- |

##### 6.2.1.6.2. K Means Clustering

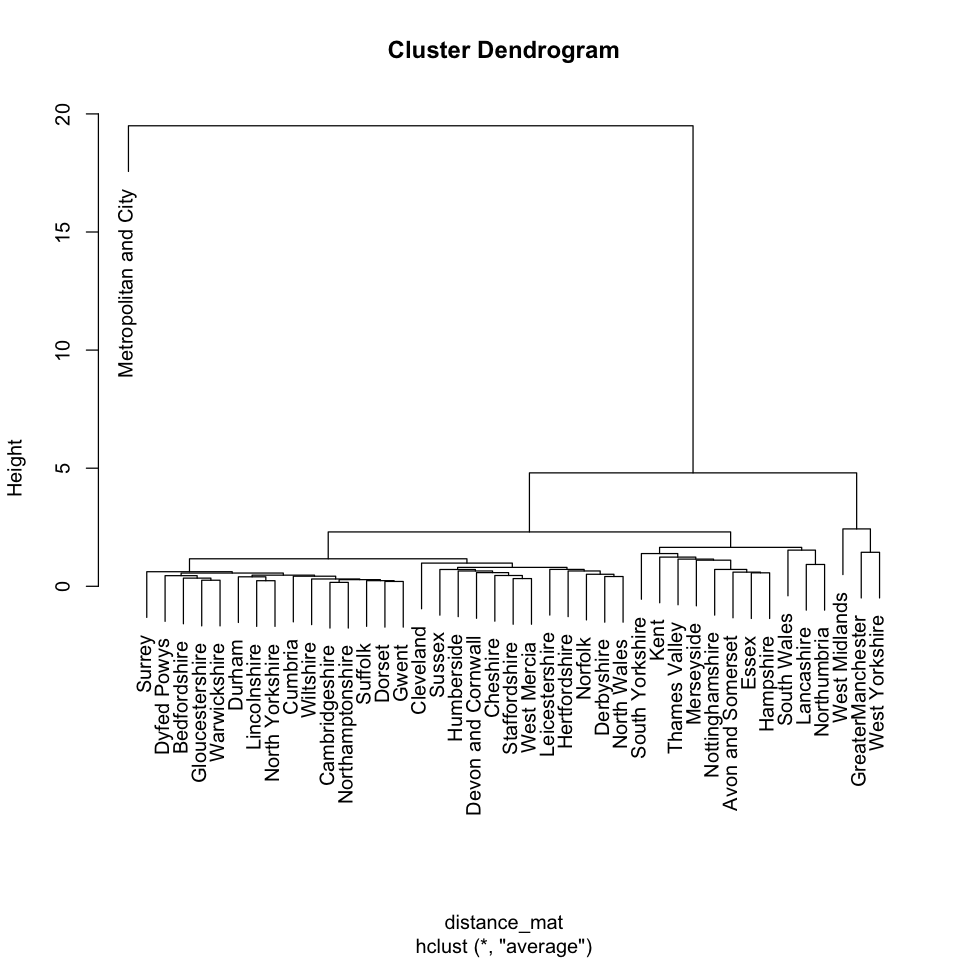
| kmeans\_clustering <- function(dataframe, clusters){  dataframe <- scale(dataframe)  model <- kmeans(dataframe, centers = clusters, nstart = 25)  return(model) } |
| --- |

### 6.2.2. Hierarchical Clustering

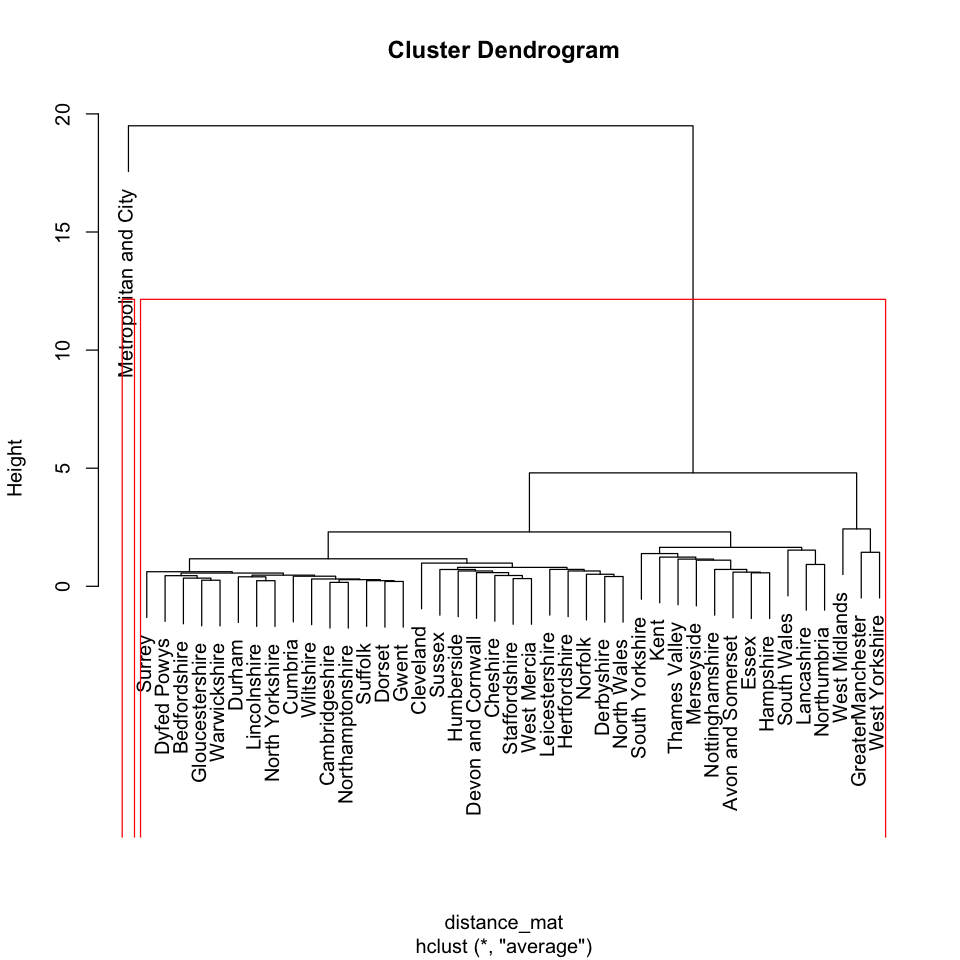
#### 6.2.2.1. Data Summary



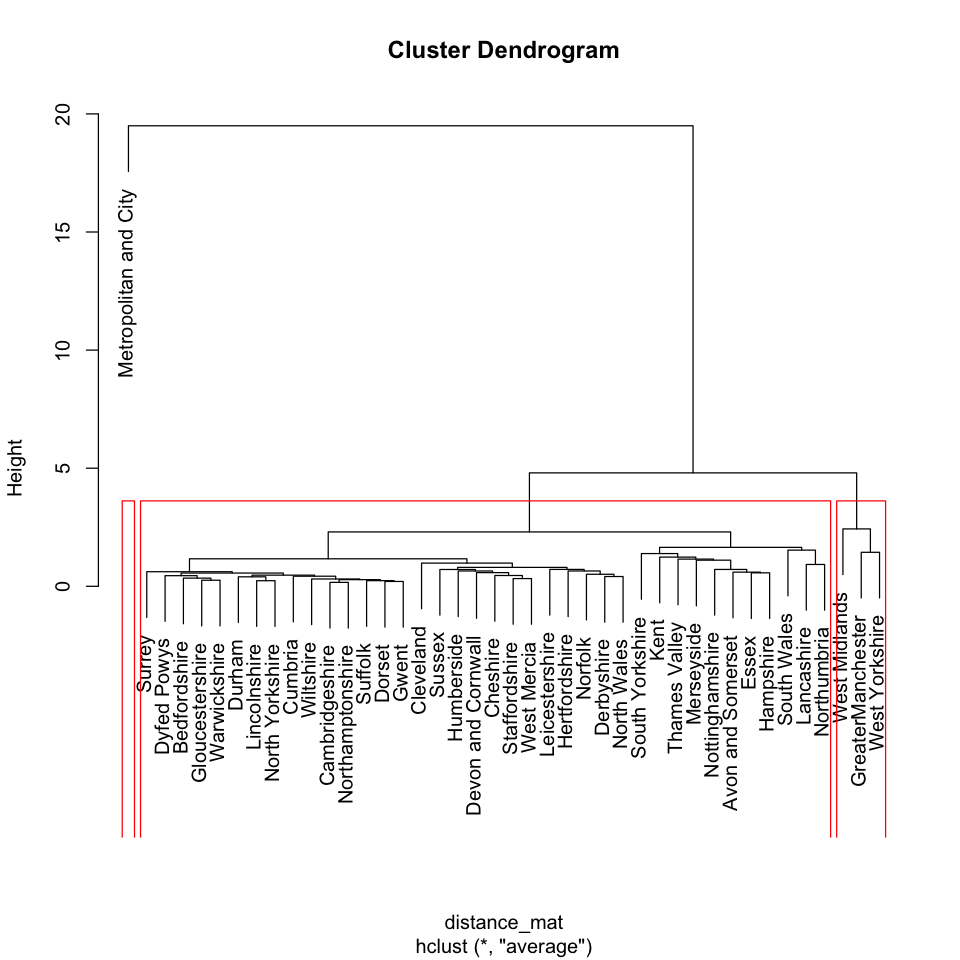
#### 6.2.2.2. Cluster Dendrogram



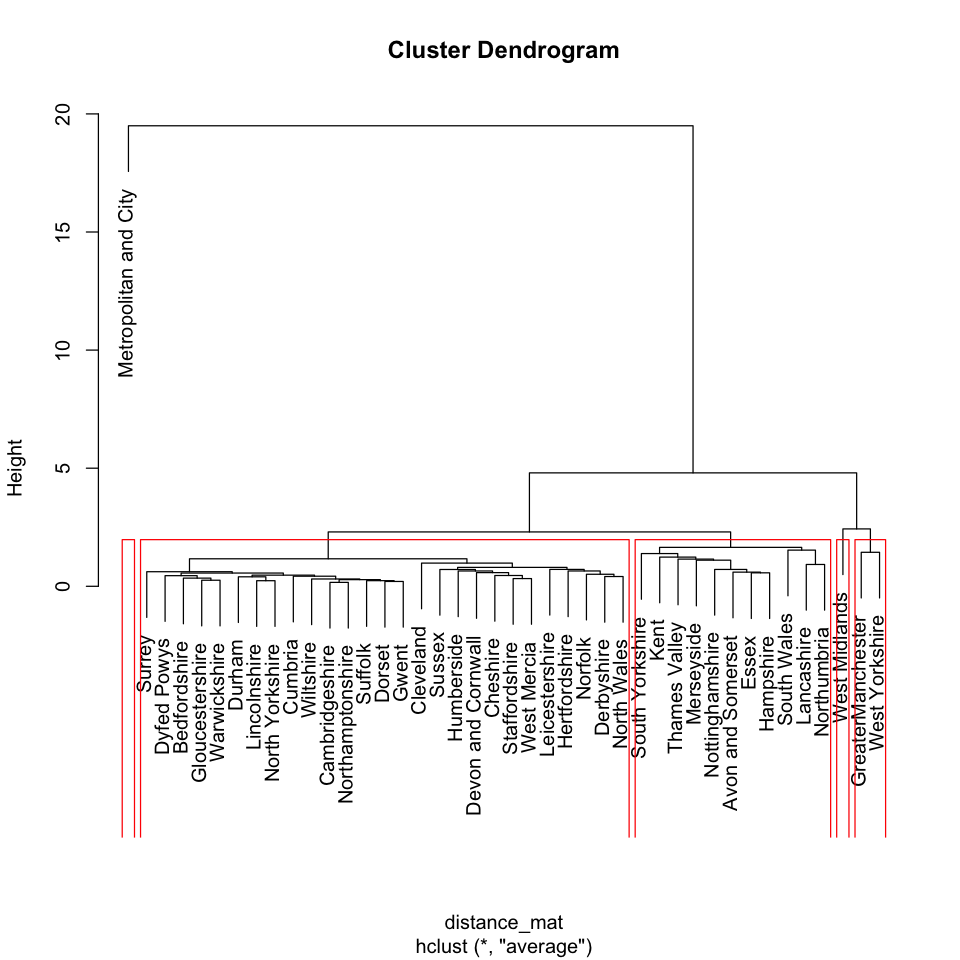
#### 6.2.2.3. Dendrogram with 2 Clusters



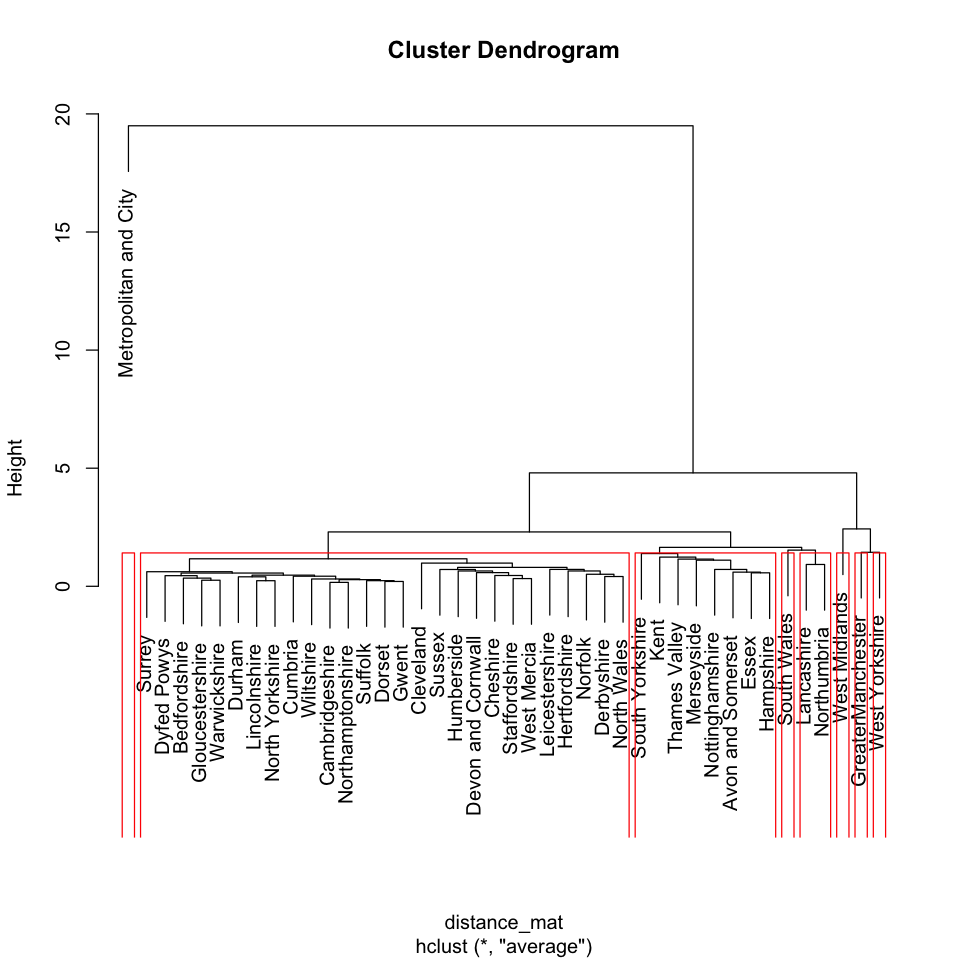
#### 6.2.2.4. Dendrogram with 3 Clusters



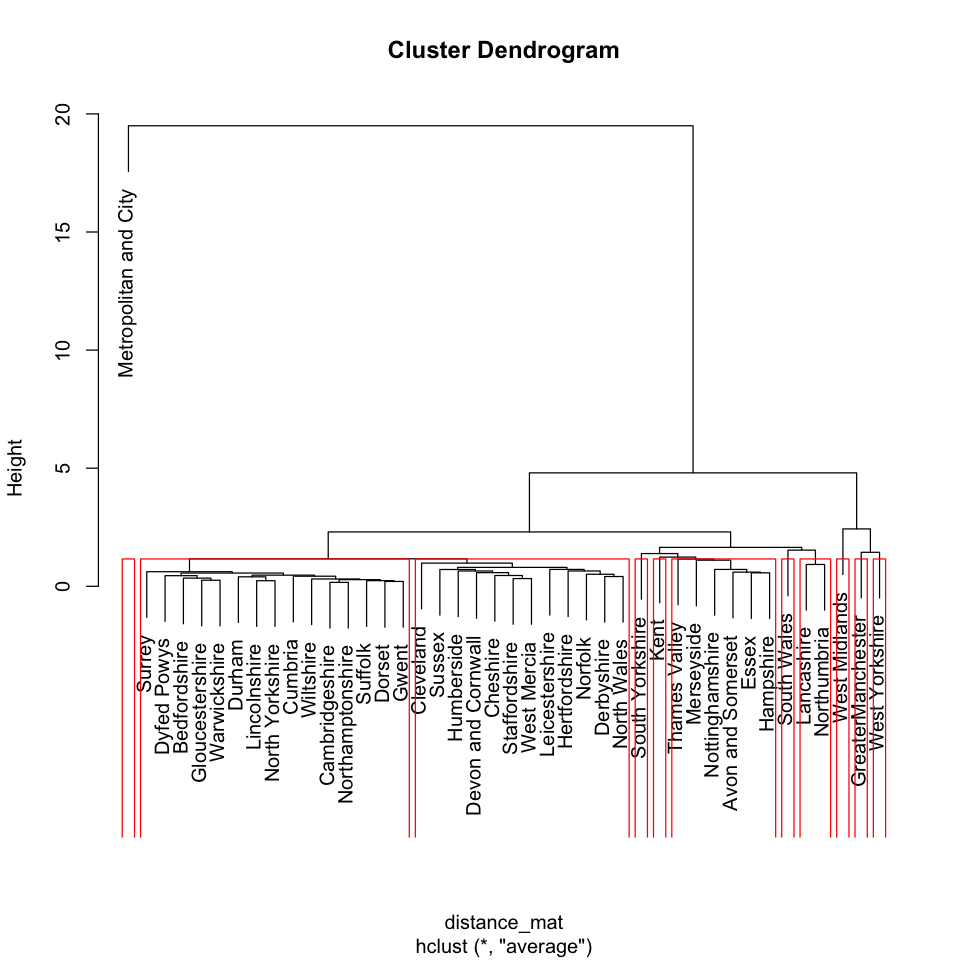
#### 6.2.2.5. Dendrogram with 5 Clusters



#### 6.2.2.6. Dendrogram with 8 Clusters



#### 6.2.2.7. Dendrogram with 11 Clusters



##### 6.2.2.7.1. Code

##### 6.2.2.7.2. Group By Country

| 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) } |
| --- |

##### 6.2.2.7.3. HC Clustering

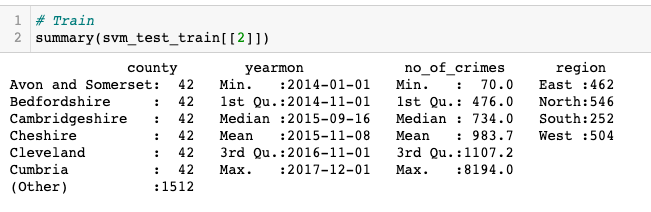
| hc\_clustering <- function(dataframe) {  distance\_mat <- dist(dataframe, method = 'euclidean')  set.seed(240)  model <- hclust(distance\_mat, method = "average")  return(list(distance\_mat, model)) } |
| --- |

## 6.3. Classification

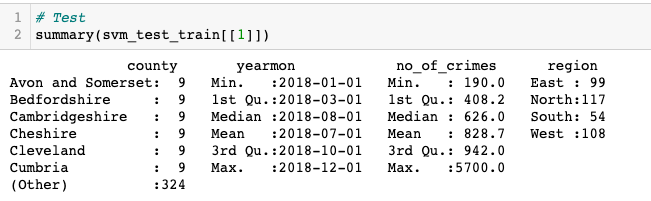
### 6.3.1. SVM

#### 6.3.1.1. Dataset Summary

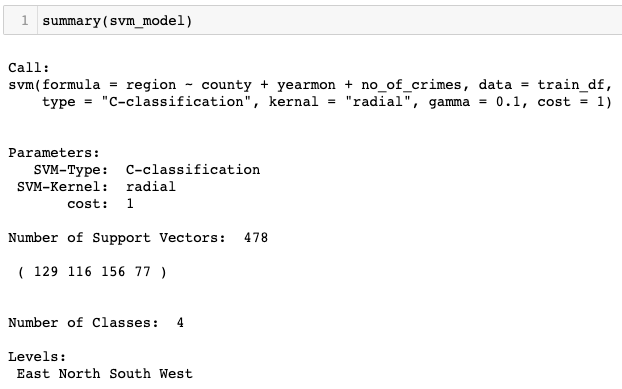
##### 6.3.1.1.1. Train



##### 6.3.1.1.2. Test



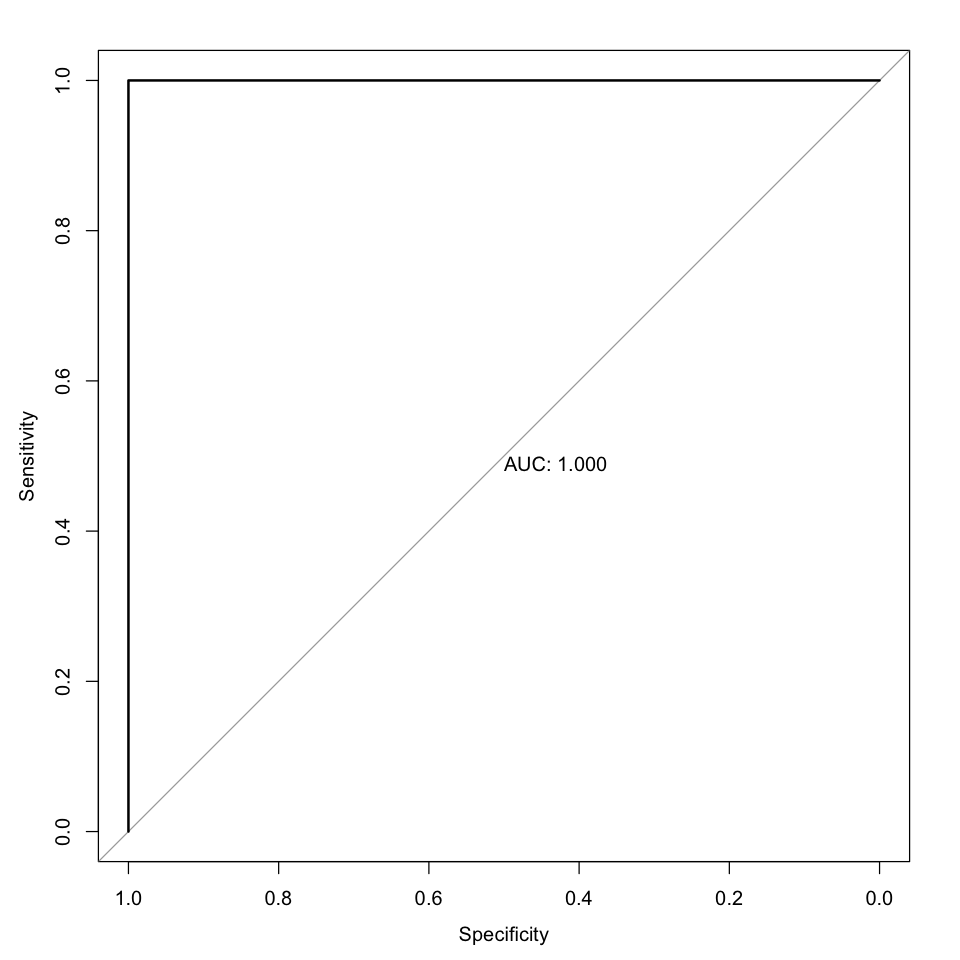
#### 6.3.1.2. Model Summary



#### 6.3.1.3. 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 |
| --- |

#### 6.3.1.4. Model Roc



#### 6.3.1.5. Code

##### 6.3.1.5.1. SVM Model Generate

| 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)) } |
| --- |

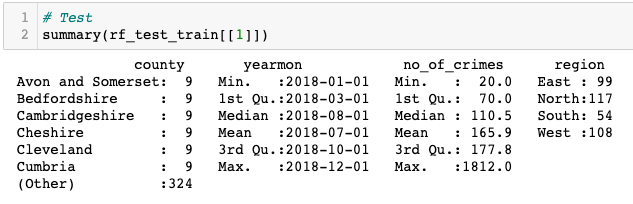
### 6.3.2. Random Forest

#### 6.3.2.1. Dataset Summary

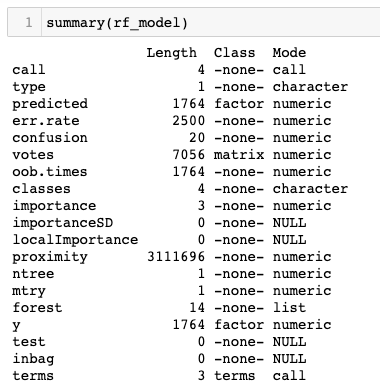
##### 6.3.2.1.1. Train



##### 6.3.2.1.2. Test



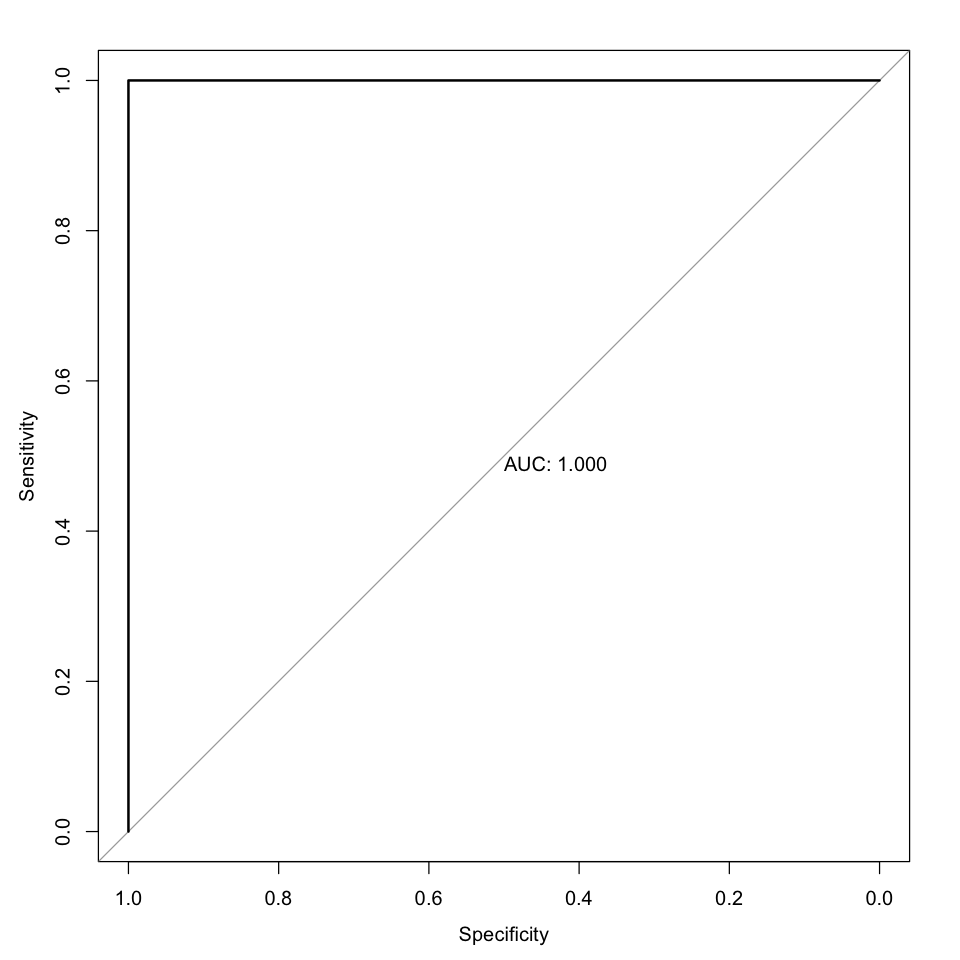
#### 6.3.2.2. Model Summary



#### 6.3.2.3. 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 |
| --- |

#### 6.3.2.4. Model Roc



#### 6.3.2.5. Code

##### 6.3.2.5.1. Rf Model Generate

| 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. Summary

### 6.3.4. Joint Code

| 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)) } |
| --- |

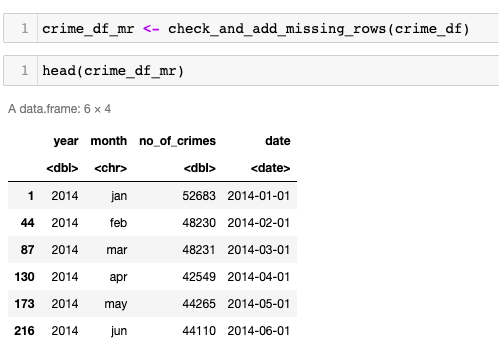
### 

# 7. Time Series Analytics

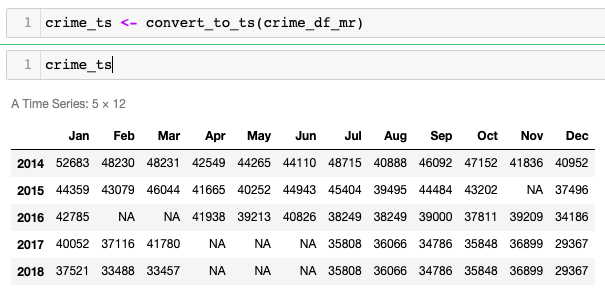
### 7.0.1. Crime

#### 7.0.1.1. Dataset Summary

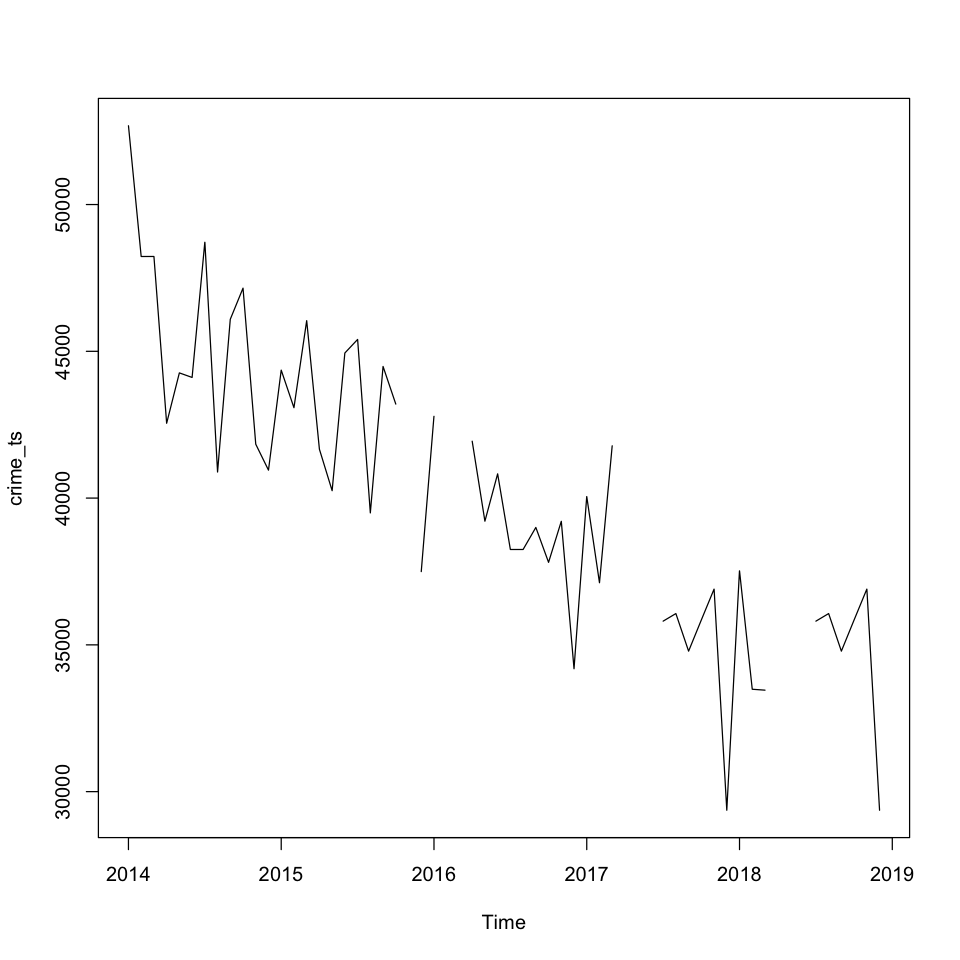
##### 7.0.1.1.1. Data Frame



##### 7.0.1.1.2. Time Series

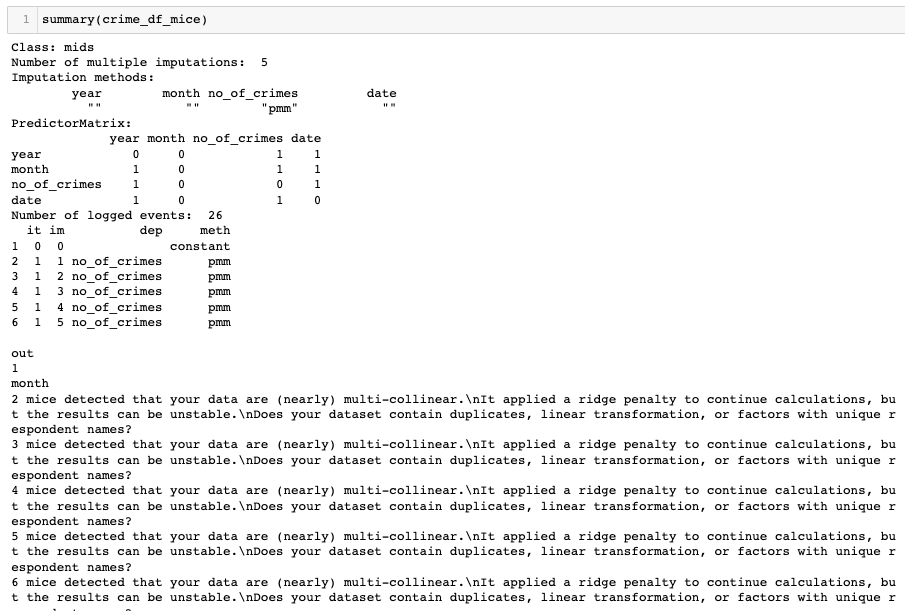


#### 7.0.1.2. Time Series Data Plot

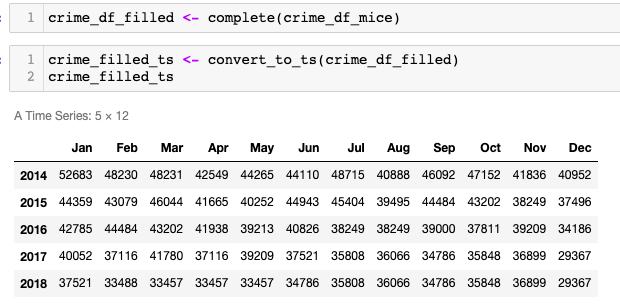


#### 7.0.1.3. Filling Missing Data with Mice

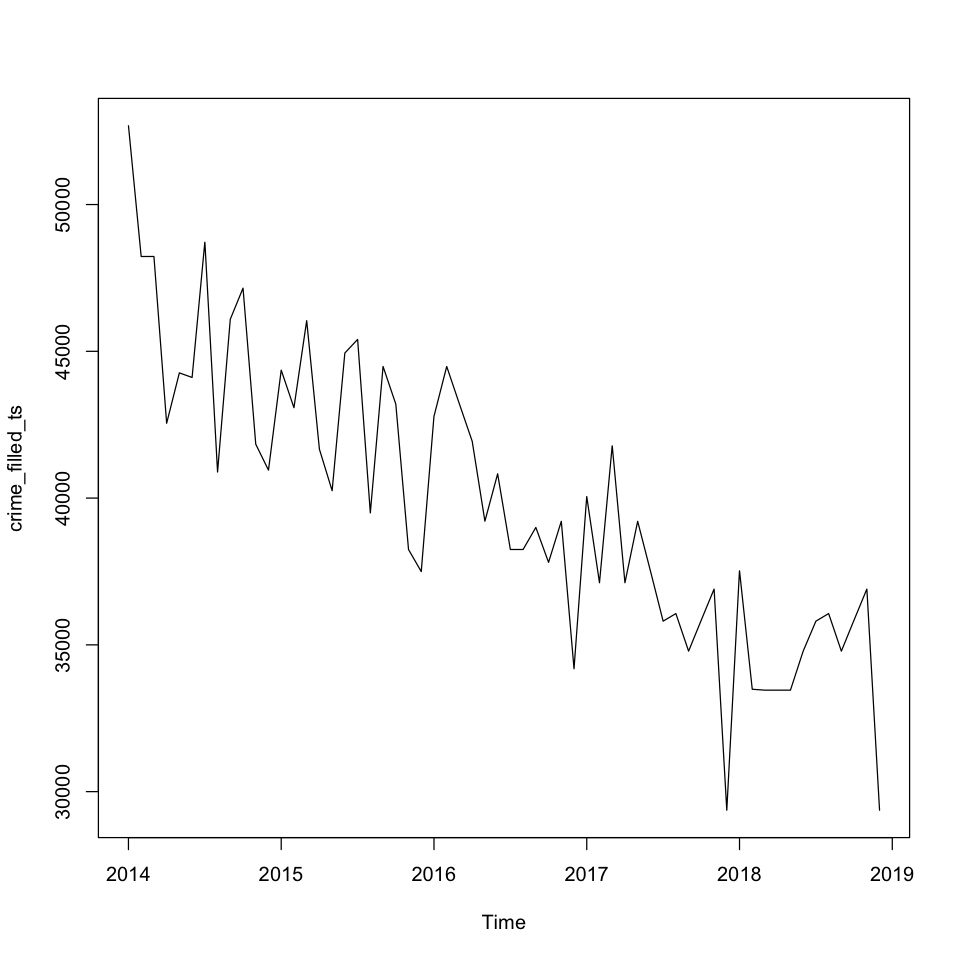
##### 7.0.1.3.1. Mice Summary



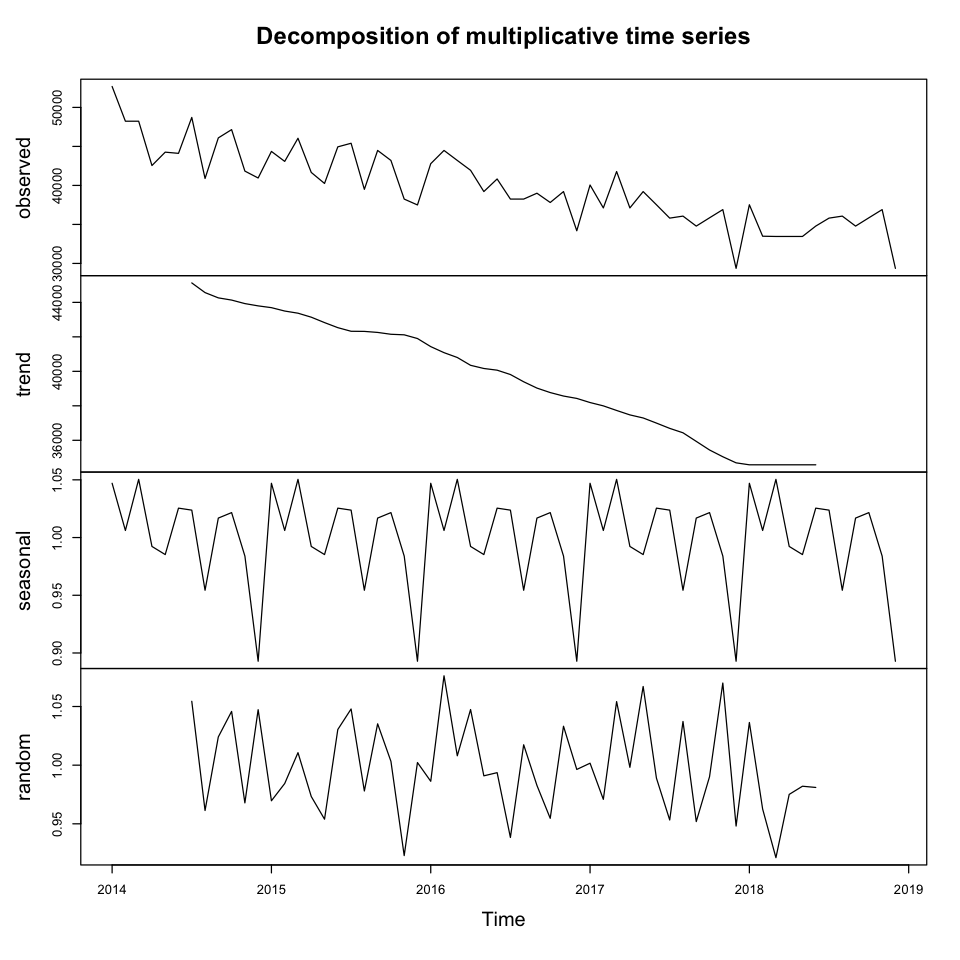
##### 7.0.1.3.2. Mice Filled Data



#### 7.0.1.4. Filled Time Series Data Plot

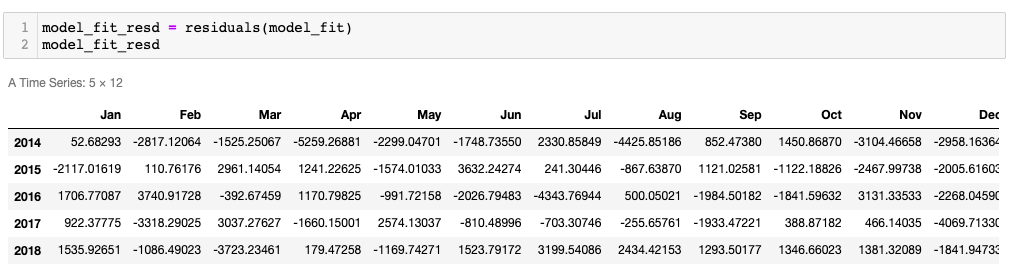


#### 7.0.1.5. Data Patterns Check

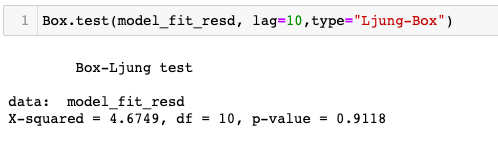


#### 7.0.1.6. Diagnostic Check

##### 7.0.1.6.1. Residuals

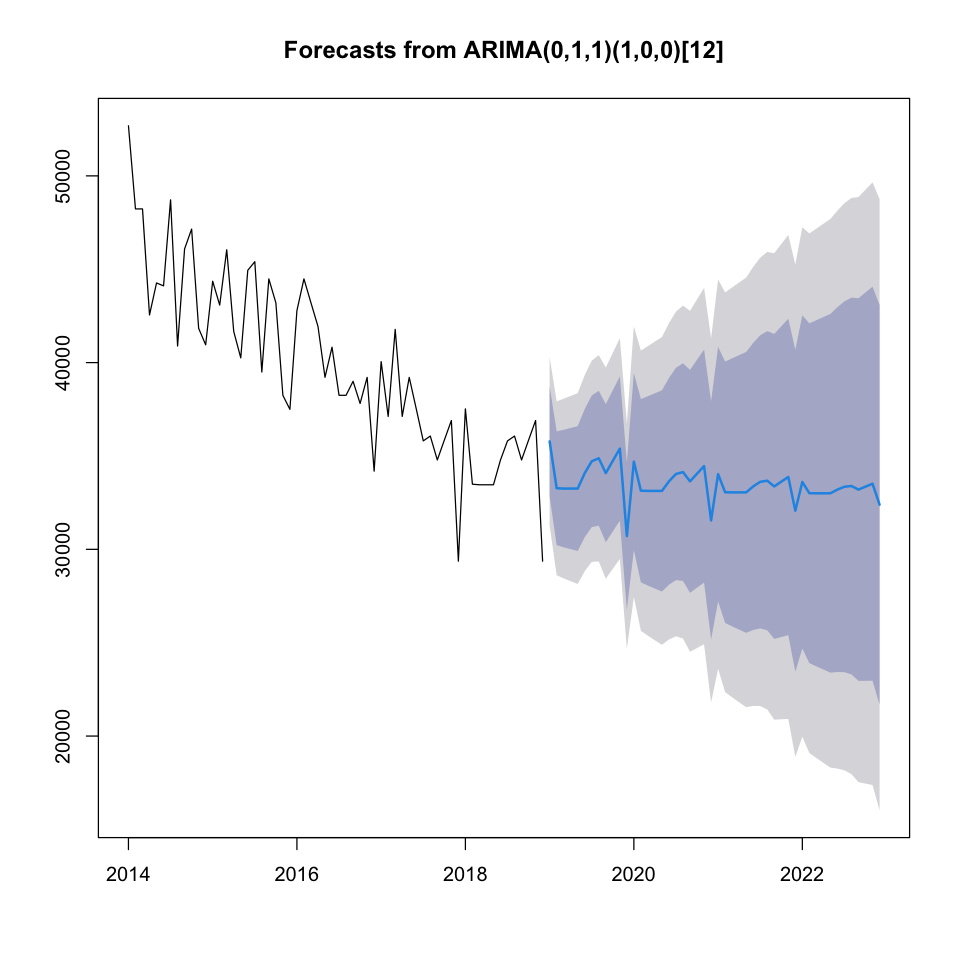


##### 7.0.1.6.2. Box-Ljung test

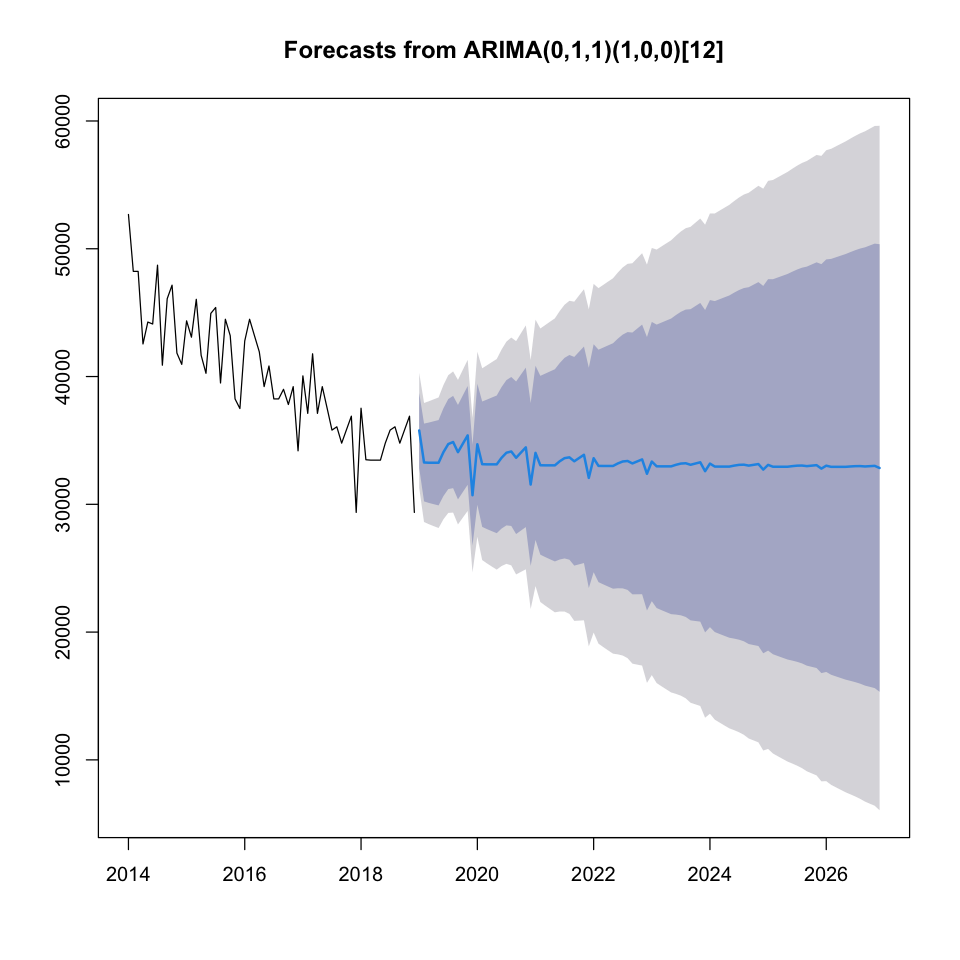


#### 7.0.1.7. Forecasting

##### 7.0.1.7.1. Years



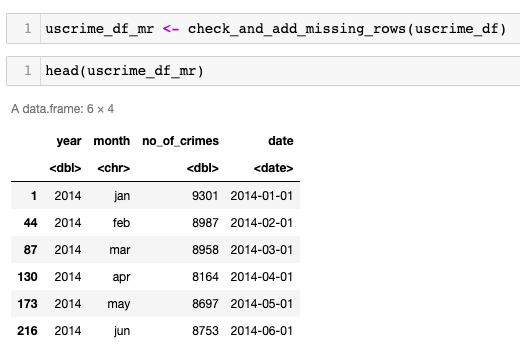
##### 7.0.1.7.2. Years



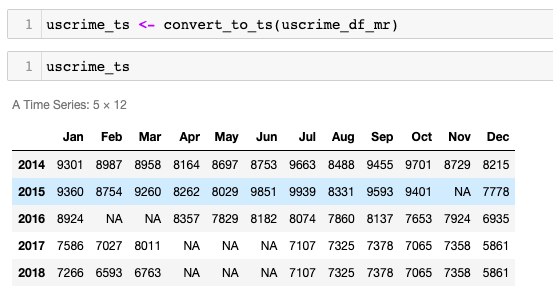
### 7.0.2. Un successful Crimes

#### 7.0.2.1. Dataset Summary

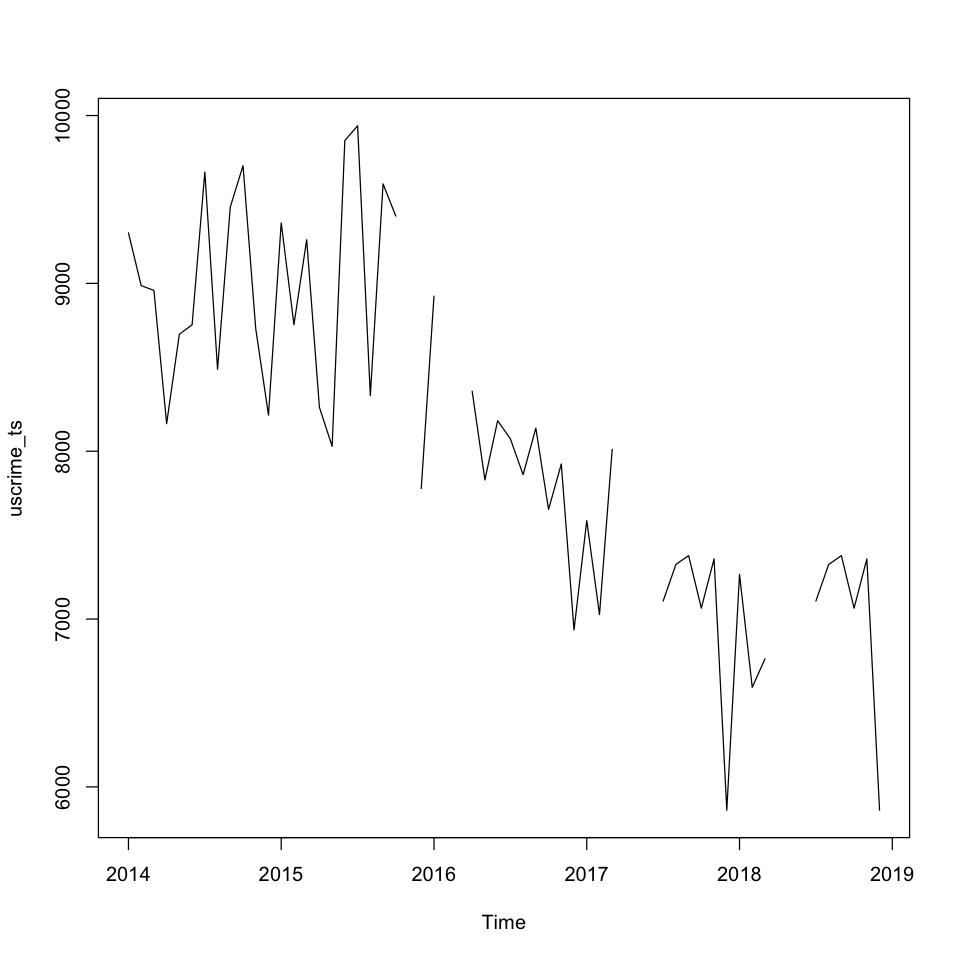
##### 7.0.2.1.1. Data Frame



##### 7.0.2.1.2. Time Series

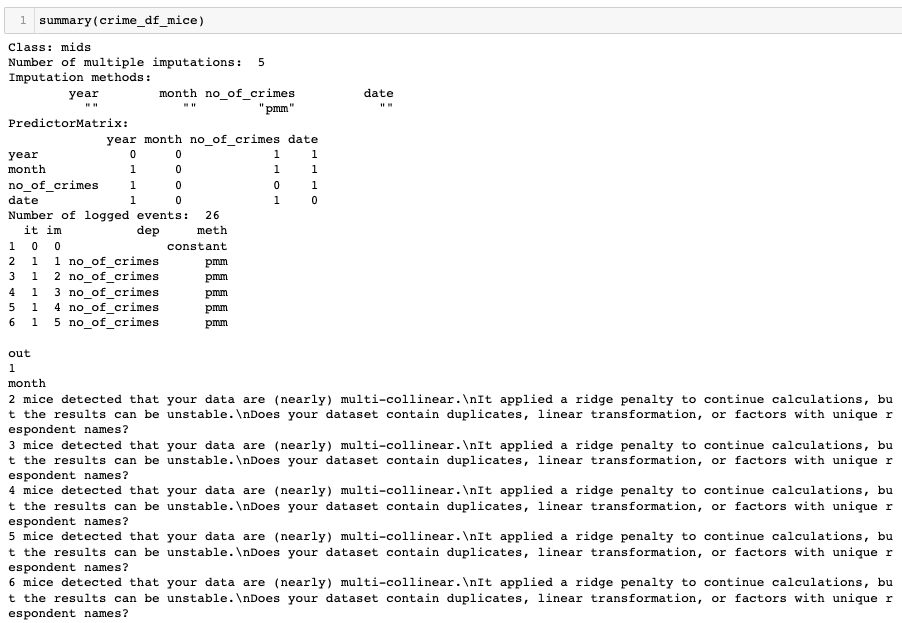


#### 7.0.2.2. Time Series Data Plot

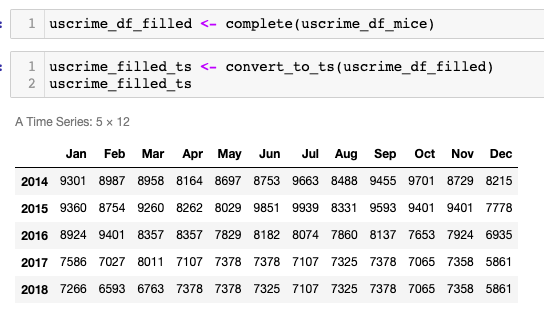


#### 7.0.2.3. Filling Missing Data with Mice

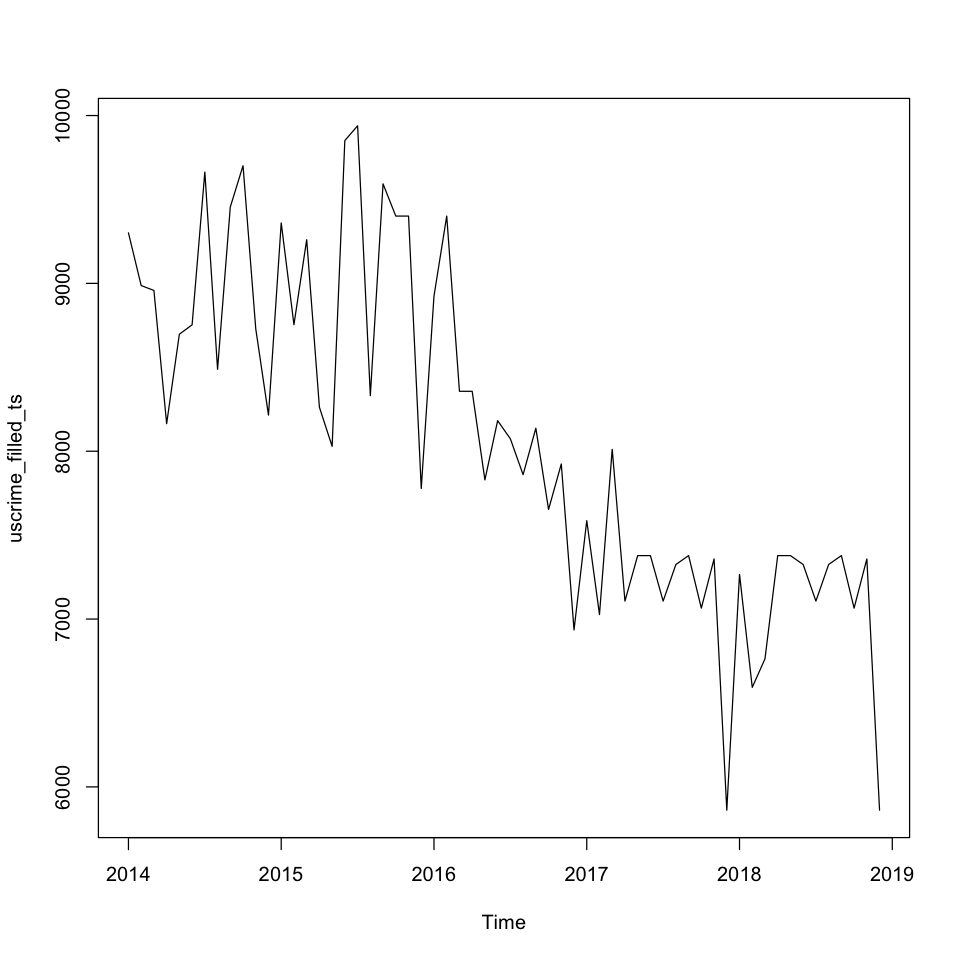
##### 7.0.2.3.1. Mice Summary



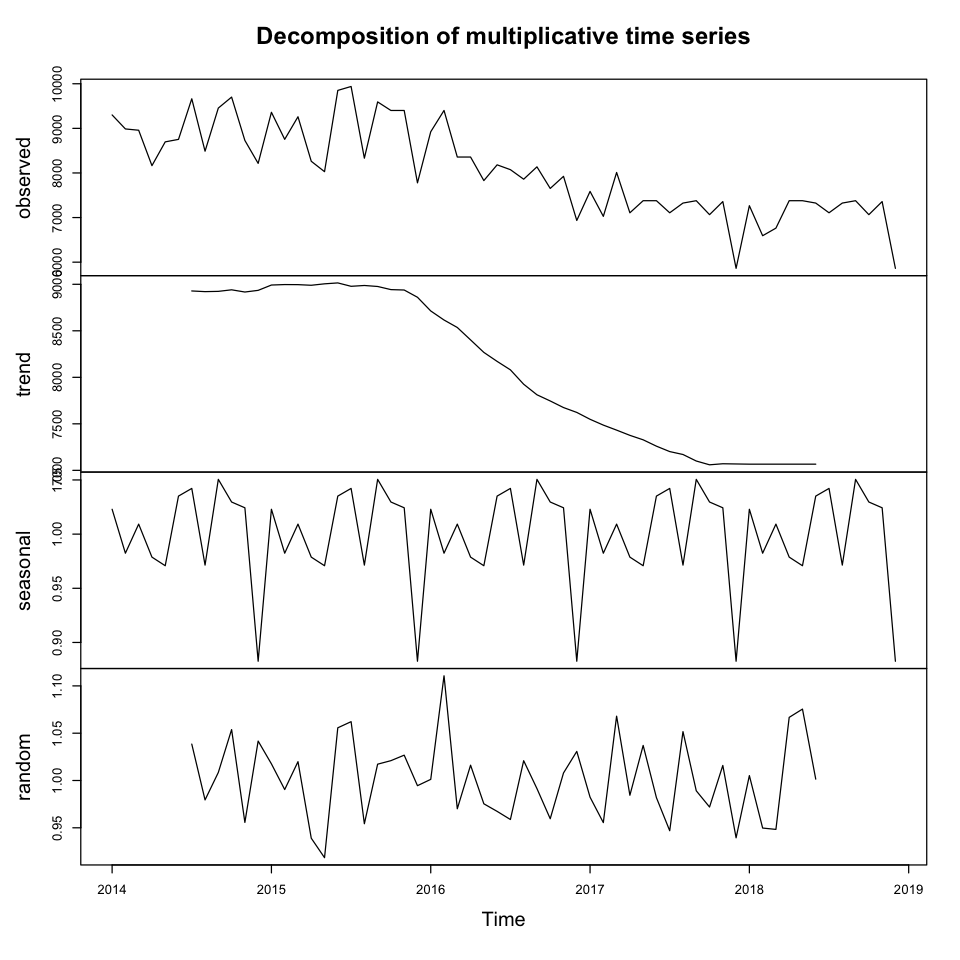
##### 7.0.2.3.2. Mice Filled Data



#### 7.0.2.4. Filled Time Series Data Plot

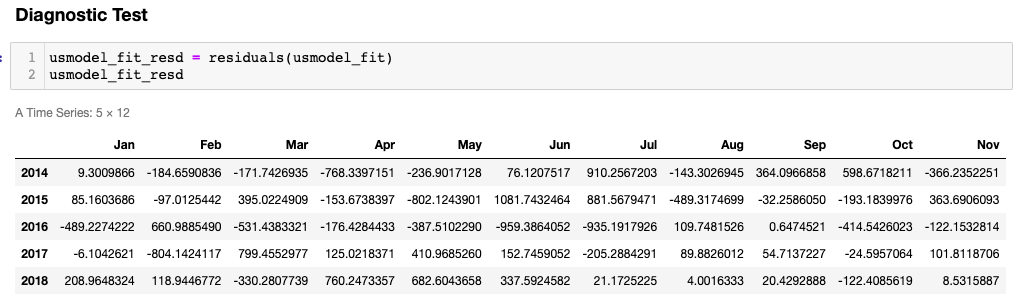


#### 7.0.2.5. Data Patterns Check

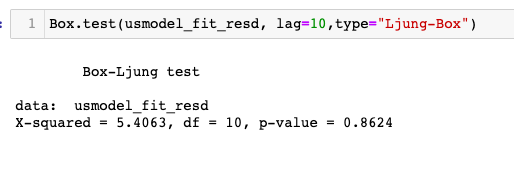


#### 7.0.2.6. Diagnostic Check

##### 7.0.2.6.1. Residuals

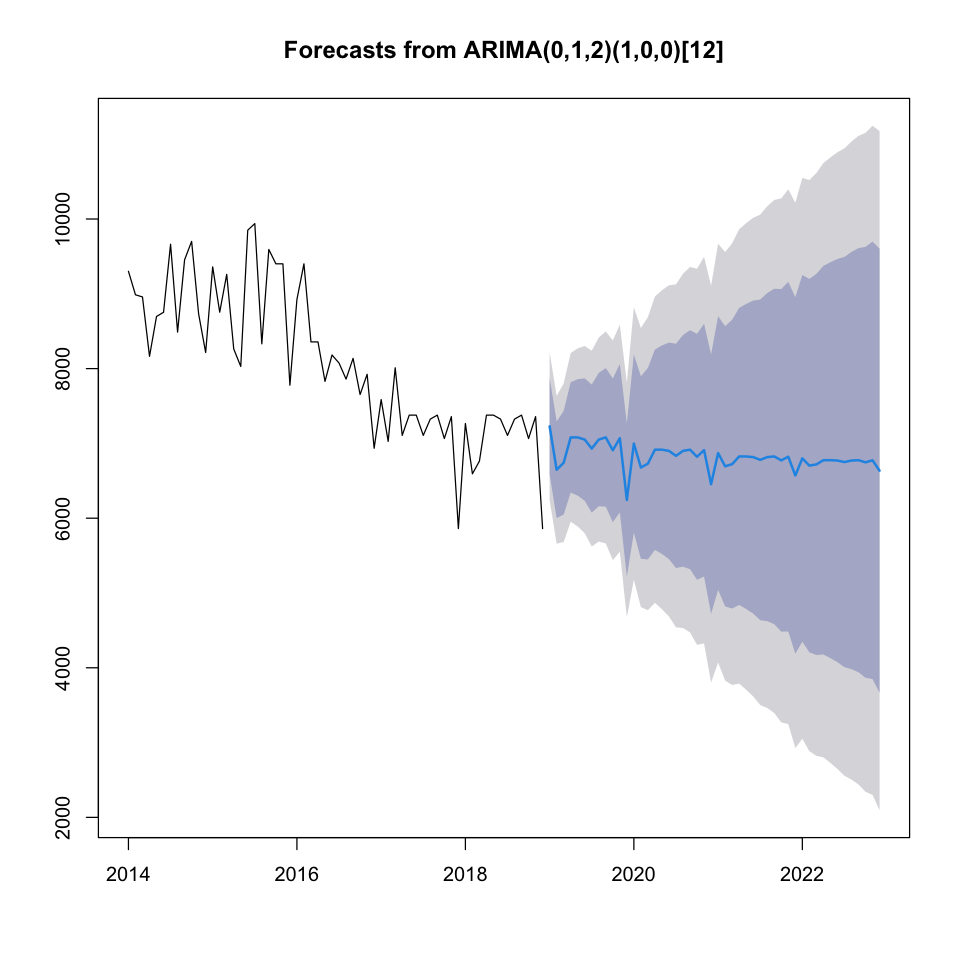


##### 7.0.2.6.2. Box-Ljung test

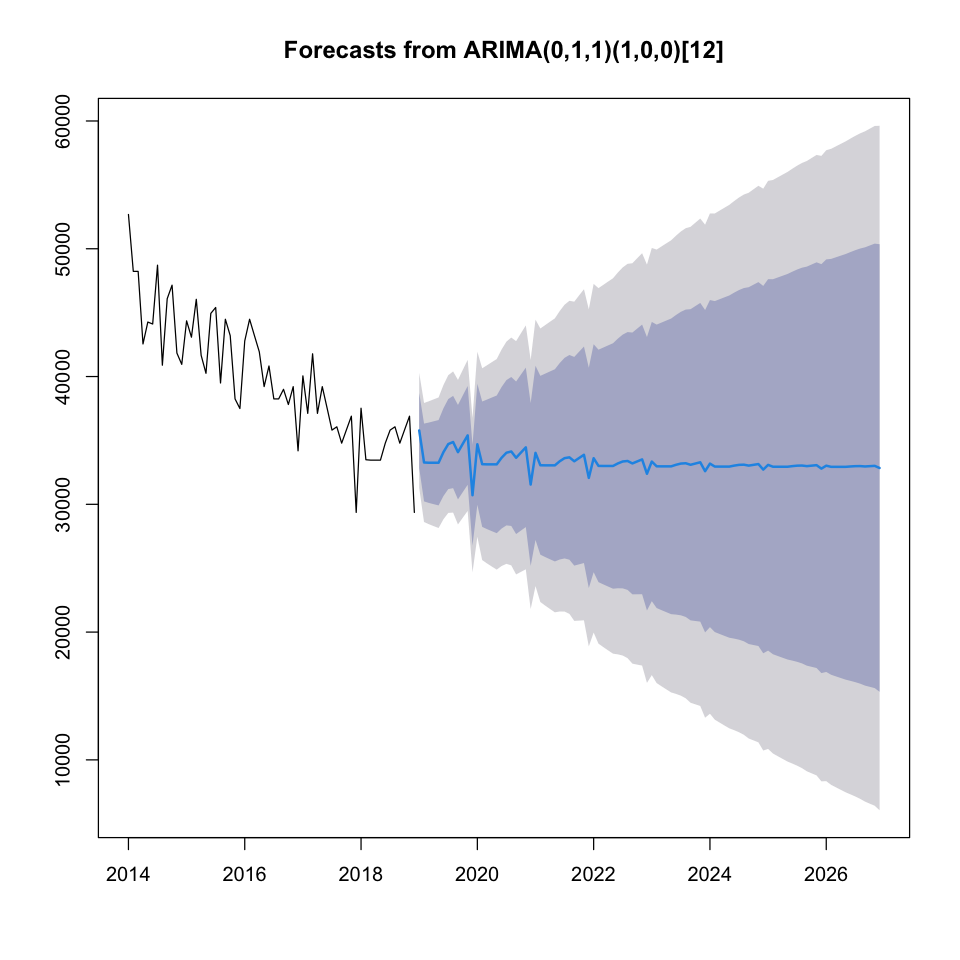


#### 7.0.2.7. Forecasting

##### 7.0.2.7.1. Years



##### 7.0.2.7.2. Years



### 7.0.3. Summary

### 7.0.4. Joint Code

##### 7.0.4.0.1. Check and Add Missing Rows with NA

| 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

| 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