# 

**Table of Contents**

[**1. Executive Summary**](#_qgl4f891pcdh) **4**

[**2. Data Overview**](#_y063g79h7i82) **5**

[**3. Data Integration**](#_o2sdp7h96myo) **6**

[3.1. Utilized Code](#_huchcnagvxfh) 6

[3.1.1. get\_all\_files\_from\_directories](#_imgpqnekb6c1) 6

[3.1.2. merge\_files](#_swk659oq8srm) 7

[3.2. Integrated Dataset Shape](#_4zoq5jph46hl) 8

[**4. Data Pre-process**](#_ng2nigqjyj3z) **9**

[4.1. Data Cleaning](#_3d9u0cafa7fz) 9

[4.1.1. Dropping Percentage Columns](#_939pydd9u2fe) 9

[4.1.1.1. Code](#_oxsqvuhmdtb6) 9

[4.1.2. Add & Sort By Date](#_1mqvjcc8uure) 9

[4.1.2.1. Code](#_xzh0p23l72cy) 10

[4.1.3. Shifting Columns](#_pnhn7ted1nst) 10

[4.1.3.1. Code](#_dc5ci630ag7j) 10

[4.1.4. Rename Columns](#_n9cso3bg0y5k) 10

[4.1.4.1. Code](#_g4suc53f02u4) 11

[4.1.5. Remove Special Characters and Convert to Integer](#_k42auig2f6y5) 11

[4.1.5.1. Code](#_quocwybwhjca) 11

[4.1.6. Rename Month Values](#_7y6fkykj9yw5) 12

[4.1.6.1. Code](#_arek3x41v0sr) 12

[4.1.7. Missing Data](#_n7e5fgby3boo) 12

[4.1.7.1. Code](#_eqwjpjd4lhsx) 13

[4.1.8. Label Region Based on County](#_k4bnl1lx5lk6) 13

[4.1.8.1. Code](#_ielsu42qkbq0) 14

[4.1.9. Shift Last Column to 5th Index](#_ipmhklpd0ta8) 15

[4.1.9.1. Code](#_12ohcgd3huem) 15

[4.1.10. Rename Conviction Columns](#_gr79e0m0gt8j) 15

[4.1.10.1. Code](#_gr6fbl3x2sca) 16

[4.2. Dataset Post Cleaning](#_qo96s0kwfo8t) 16

[4.2.1. Glimpse](#_f2p7i28sp5xf) 16

[4.2.2. Visualizing Missing](#_lxh5k2l3g3cl) 17

[4.2.3. Visualizing Data Types](#_n278l4b2b39y) 18

[4.3. Dataset Split](#_9cnwi07zxilw) 18

[4.3.1. Crime Dataset Glimpse](#_wb9szax3x456) 19

[4.3.2. Unsuccessful Crime Dataset Glimpse](#_akwj4qmlilth) 19

[4.4. Code](#_5b010zcbslr5) 19

[**5. Descriptive Analytics**](#_z2sprpez8oun) **21**

[5.1. Attributes Analysis](#_g2wqjevvf90) 21

[5.2. Analysis dependent on Regions](#_9uaguuimla3x) 21

[5.3. Analysis dependent on County](#_tg9j37pxg26q) 21

[5.4. Analysis dependent on Years & Months](#_el294odw6aox) 21

[5.5. Analysis between Crime Types](#_18zb4op7rs1m) 21

[**6. Predictive Analytics**](#_s0bwct5sdzh) **22**

[6.1. Linear Regression](#_31c64sh9tx9v) 22

[6.2. Classification](#_d6q3r7g7mxkr) 22

[6.3. Clustering](#_dv9fuf2di3zq) 22

[**7. Prescriptive Analytics**](#_la6803uw0zws) **23**

[7.1. Time Series Analytics](#_tiot4p9v7oes) 23

[**8. References**](#_xpj9glrszg7s) **24**

# 

# 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.

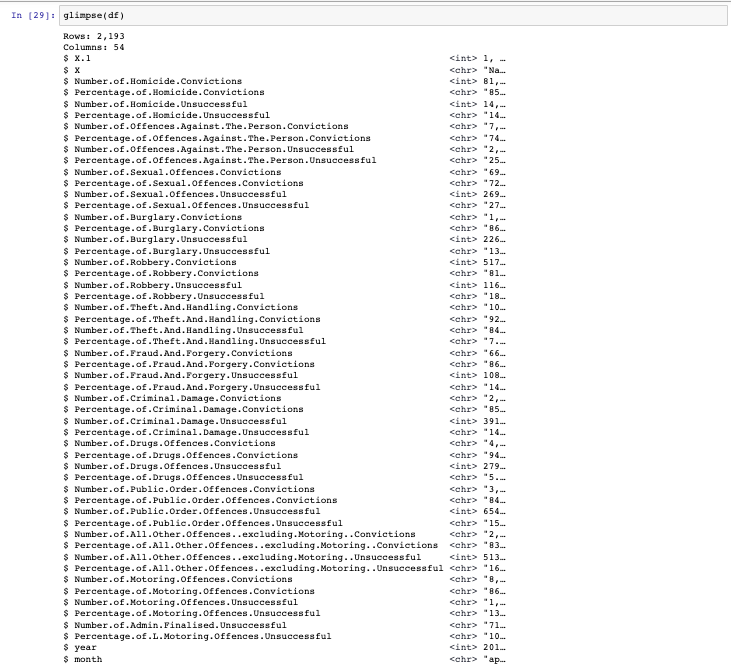
### 3.1.2. merge\_files

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

## 4.1. Data Cleaning

The sections below contain each operation performed with reasoning and impact.

### 4.1.1. Dropping Percentage Columns

#### **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

#### **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

#### **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

#### **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

#### **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.

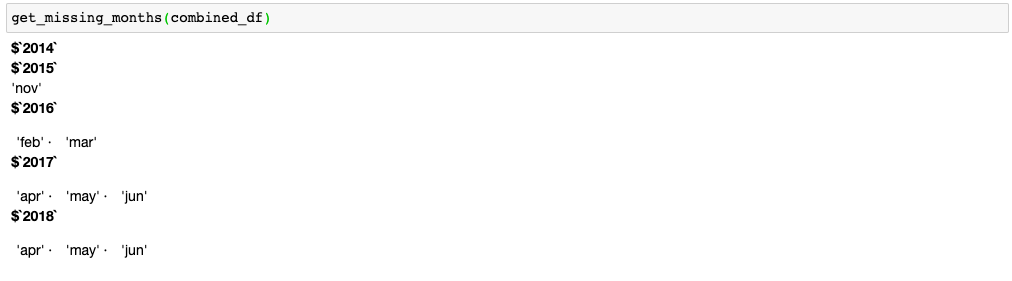
### 4.1.6. Rename Month Values



#### **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) } |
| --- |

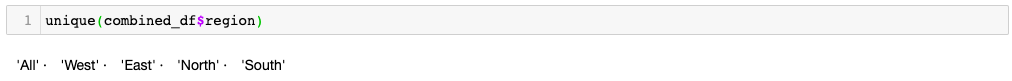
### 4.1.7. Missing Data



#### **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) } |
| --- |

### 4.1.8. Label Region Based on County



| 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) } |
| --- |

### 4.1.9. Shift Last Column to 5th Index

#### **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) } |
| --- |

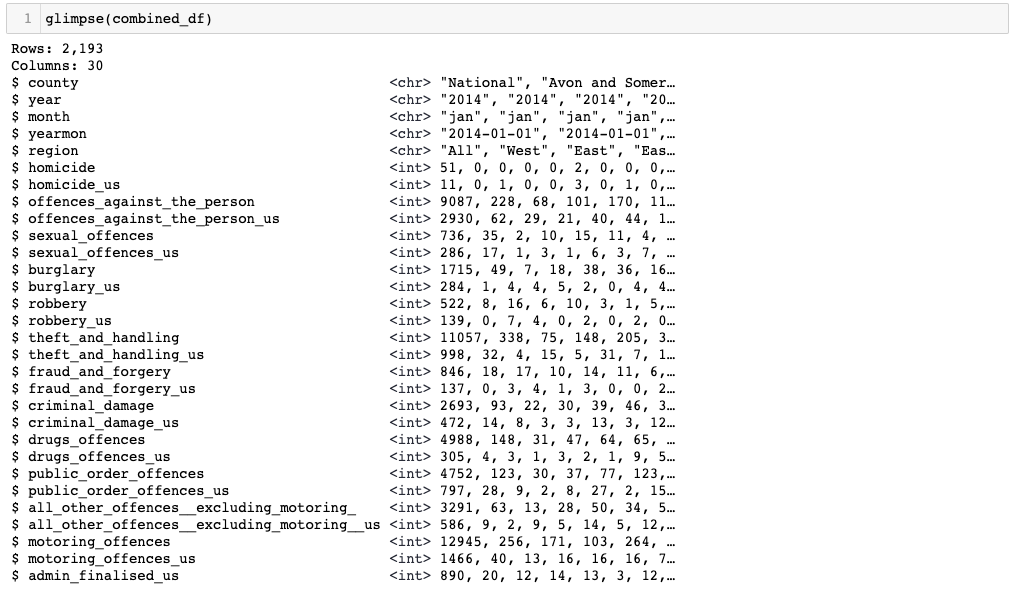
### 4.1.10. Rename Conviction Columns

| 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) } |
| --- |

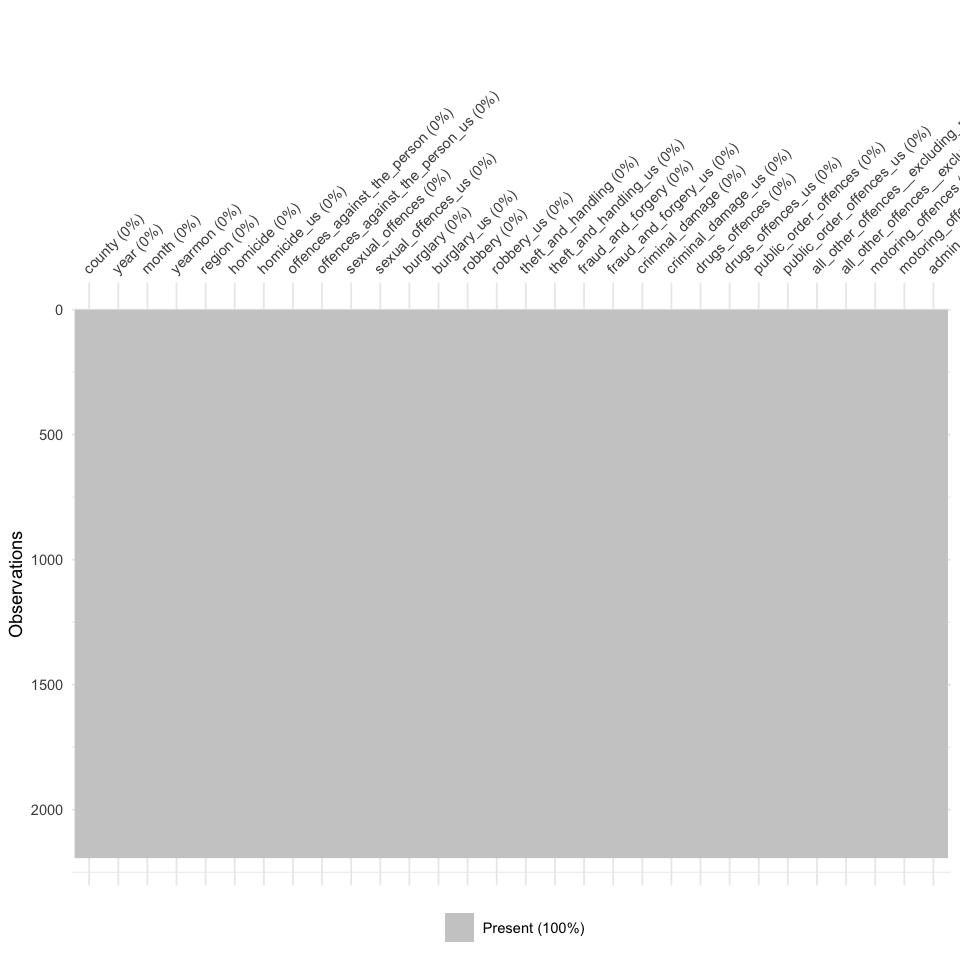
#### **4.1.10.1. Code**

## **4.2.** Dataset Post Cleaning

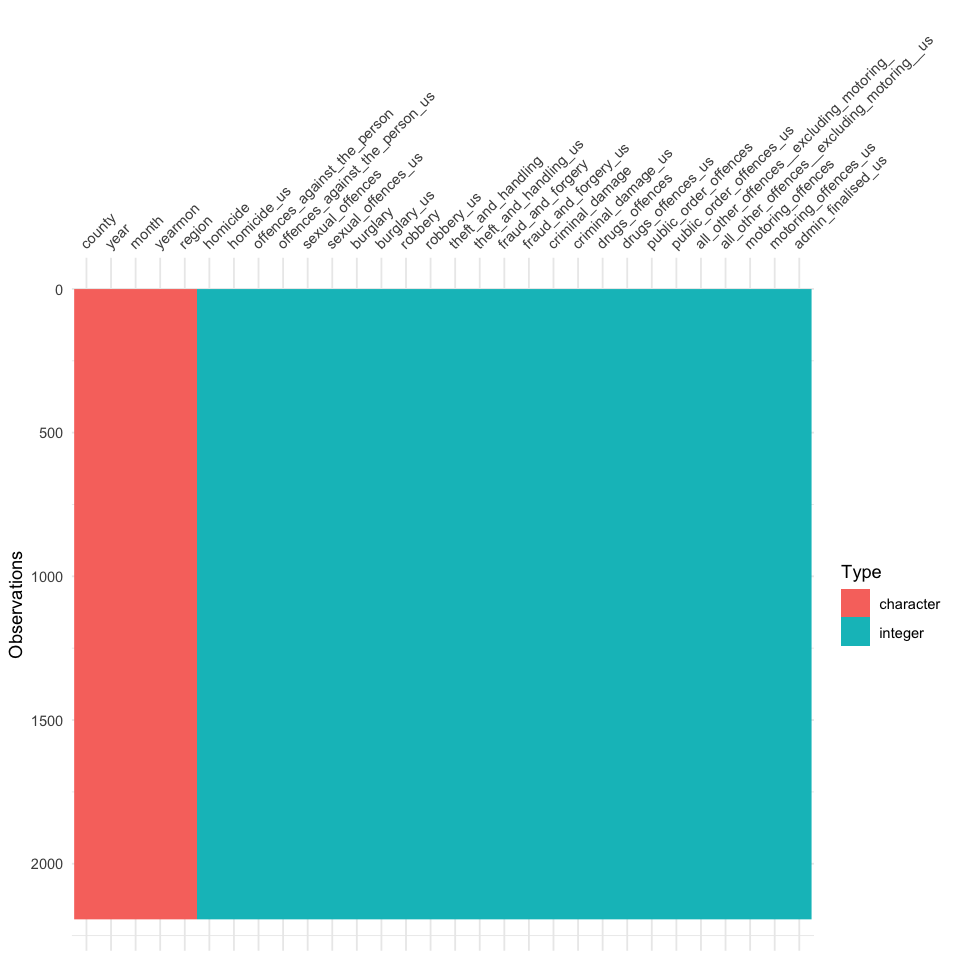
### 4.2.1. Glimpse



### 4.2.2. Visualizing Missing

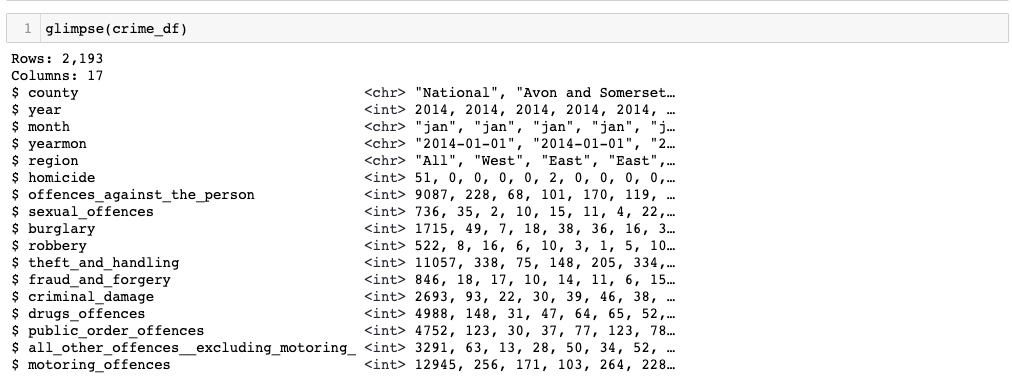


### 4.2.3. Visualizing Data Types

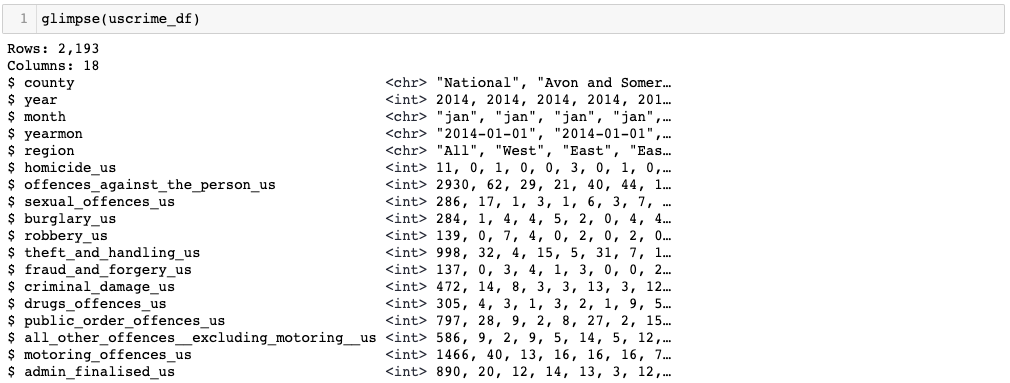


## 4.3. Dataset Split

### 4.3.1. Crime Dataset Glimpse



### 4.3.2. Unsuccessful Crime Dataset Glimpse



## 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)) } |
| --- |

### 

# 5. Descriptive Analytics

Introduction here for descriptive analytics

## 5.1. Attributes Analysis

### 5.1.1. Crimes Dataset

#### 5.1.1.1. County

| Length Class Mode   2193 character character |
| --- |

#### 5.1.1.2. Year

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

#### 5.1.1.3. Month

| Length Class Mode   2193 character character |
| --- |

#### 5.1.1.4. Yearmon

| Length Class Mode   2193 character character |
| --- |

#### 5.1.1.5. Region

| Length Class Mode   2193 character character |
| --- |

#### 5.1.1.6. Homicide

| 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

| 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

| 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

| 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

| 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

| 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

| 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

| 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

| 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

| 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\_

| 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

| 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

| Length Class Mode   2193 character character |
| --- |

#### 5.1.2.2. Year

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

#### 5.1.2.3. Month

| Length Class Mode   2193 character character |
| --- |

#### 5.1.2.4. Yearmon

| Length Class Mode   2193 character character |
| --- |

#### 5.1.2.5. Region

| Length Class Mode   2193 character character |
| --- |

## 5.2. Analysis dependent on Regions

## 5.3. Analysis dependent on Years & Months

## 5.4. Analysis between Crime Types

Correlation

Covariance

## 5.5. Trend Analysis for Successful & Unsuccessful Crimes

# 6. Predictive Analytics

## 6.1. Linear Regression

## 6.2. Classification

## 6.3. Clustering

# 7. Prescriptive Analytics

## 7.1. Time Series Analytics

# 8. References