Data Science: Capstone module
Choose Your Own!
Road Accident Detection and Response

Rob Meekings

02/01/2021

Contents

1 Executive Summary

1.1 This report summarizes the findings of exploratory data analysis performed on traffic accident data in the UK. The data is published by the UK Department for Transport and is publicly available via the data.gov.uk website. The exploration of the data leads to the development of statistical models to predict the number of casualties, given that an accident has occurred.

1.2 The final model, an ensemble based on a random forest, presented in this paper scores a RMSE of 0.704.

1.3 The exploratory analysis raises a number of questions for further study and opportunities to further extend and develop the model. These are presented at the end of the paper along with some research questions that could go a long way to improving our understanding in this field, potentially improving road safety by reducing the number of accidents.

2 Introduction

2.1 This report summarizes the development and results of the Road Accident Detection and Response (RADAR) system, developed in R by Rob Meekings for the *"Choose Your Own!"* assignment of the Data Science: Capstone module (HarvardX PH125.9x).

2.2 The goal of this project is to train a machine learning algorithm on a subset of the UK road accident data for the years 2016-2019 (the training set) to correctly predict number and severity of accidents in a subset of this same accident data that is not used in making predictions (known as the test or validation set).

2.3 The structure of this report reflects the sequence of steps taken in developing this model. Section 3 describes how the source data was imported into the R environment, cleaned and prepared for analysis and modelling. In section 4 we then discuss the exploratory data analysis that was performed to get a feel for the data and how to model it. We then look at the steps required to develop a model in section 5, summarize our results in section 6, and then consider any further work or next steps that would be likely to improve the model in section 7.

2.4 This report is intended to be read by someone familiar with the module and course content, as such the reader should be familiar with the R programming language and data science topics.

2.5 This report complies with the (edX Honor Code).

3 Source Data

3.1 The source data for this project comes from the Department for Transport in the UK and relates to road accidents in the UK in the years 2016-2019. The data is assumed to be accurate, complete and reliable. This data contains a row for every recorded accident, associated datasets containing details of vehicles and any casualties are also available.

Download and cleaning of the data

3.2 The data is downloaded from the Department for Transport's space on the data.gov.uk platform using the code below. The data can be accessed in a browser through the web page at https://data.gov.uk/dataset/cb7ae6f0-4be6-4935-9277-47e5ce24a11f/road-safety-data.

3.3 A helper function is used to automate some of the mechanical and repetive code needed to download a zip file, extract a csv from it and read this into R as a data frame.

```
############################################################
# Create training set and validation hold-out test set
############################################################

# Whilst this looks very organised and uniform some of the files are hosted by amazon,
# some by the DfT, some are zips and some are csvs, some of the csvs have short names,
# some have long names, to overcome this we have prepared this helper function
get_src_data <- function(url, file_name, is_zip) {
  dl <- tempfile()        # create a tempfile to receive the downloaded file
  download.file(url, dl, mode="wb")  # download the file at the given url
  if (is_zip == TRUE) {   # unzip "file_name" and process as a csv with a header
    output <- fread(text = gsub("::", "\t", readLines(unzip(dl, file_name))), header=TRUE)
  }
  else {                  # process the file as a csv with a header
    output <- fread(text = gsub("::", "\t", readLines(dl)), header=TRUE)
  }
  rm(dl)                  # clear/release the tempfile
  return(output)          # return the dataframe derived from the source file
}
```

3.4 Using this function the code needed to download and unzip the accident and vehicle data we are interested is greatly simplified, despite the variation in files from year to year: the 2019 files are csv's in a zip file, the 2018 files are csv's that have not been zipped, the 2016 and 2017 files are zipped and the vehicle information files have standardized short names, but the accident filenames differ.

3.5 The next step would be to combine these data sets to create files for all accidents and vehicles. If we try to do this immediately we get errors reported:

> Error: not compatible:
>
> - Incompatible types for column `Longitude`: character vs double
> - Incompatible types for column `Latitude`: character vs double
> - Incompatible types for column `Speed_limit`: integer vs character

3.6 We can address these issues by coaxing the longitude, latitude and speed limit data from character to numeric. Having done this we can concatenate the annual data to create two composite files.

```
# Need to clean up speed limits in acc16
acc16 %>% mutate(nSpeed = as.numeric(Speed_limit)) %>%
  filter(is.na(nSpeed)) %>% head()
```

```
# Check that "NULL" is the only problem
acc16 %>% filter(Speed_limit != "NULL") %>%
  mutate(nSpeed = as.numeric(Speed_limit)) %>%
  filter(is.na(nSpeed)) %>% head()

# Coerce to numeric, accept na for "NULL"
acc16$Speed_limit <- as.numeric(acc16$Speed_limit)
#Warning messages: NAs introduced by coercion
# This is okay, we understand why

# Need to clean up Latitude and Longitude in acc17
acc17 %>% mutate(nLong = as.numeric(Longitude)) %>%
  filter(is.na(nLong)) %>% head()
# Check if empty text and "NULL" is the only problem
acc17 %>% filter(!Longitude %in% c("", "NULL")) %>%
  mutate(nLong = as.numeric(Longitude)) %>%
  filter(is.na(nLong)) %>% head()

# Coerce to numeric
acc17$Latitude <- as.numeric(acc17$Latitude)
acc17$Longitude <- as.numeric(acc17$Longitude)
#Warning messages: NAs introduced by coercion
# This is okay, we understand why

# Concatenate datasets, the others are more straightforward
accidents <- union(union(acc19, acc18), union(acc17, acc16))
vehicles <- union(union(veh19, veh18), union(veh17, veh16))
```

3.7 We should be able to get some predictive power from the vehicle data, however for simplicity we will limit the data we consider to the first vehicle's data (using Vehicle_Reference = 1 to indicate this) and restrict ourselves to those data items that intuitively seem most likely to relate to accidents: vehicle age and type, the driver's gender, age and the IMD decile, a wealth or deprivation measure. We are assuming that there is a logic to the way the data is collected and organised such that there may be more significance to the vehicle with reference 1. We then join this data onto the accident data using the Accident_Index key field.

```
# Now take the first vehicle from each accident
acc_veh_types <- vehicles %>%
  filter(Vehicle_Reference == 1)

# We can drop fields that are unlikely to tell us much about accidents, we keep
acc_veh_types <- acc_veh_types[, c(1,3,15,16,17,20,21)]

# Copy these fields across to
accidents <- left_join(accidents, acc_veh_types, by="Accident_Index")

# Garbage collection, free up some resources.
rm(acc16, acc17, acc18, acc19)
rm(veh16, veh17, veh18, veh19)
rm(acc_veh_types, vehicles)
```

Feature development, date-based information

3.8 Now we have cleaned up the accident data and augmented it with information about the first vehicle and its driver we can derive some further information that may be relevant based upon what we know about where and when the

accident occurred. We use functions from the library *lubridate* to convert the date and time to a *POSIXct* which we can then easily manipulate in R.

```
# Convert date and time variables to a composite date time
accidents$accident_time <- dmy_hm(paste(accidents$Date, " ", accidents$Time))
```

3.9 We should be able to improve on the light conditions data, which is likely to be subjective, by adding some objective data based on the sun's position given the loacation, date and time. We first of all find the times for dawn and dusk for the date and location given, and then find where the time of the accident falls.

```
# We want to augment the data with time of day, we'll add this to the composite data
# this might be expected to have a number of roles to play in accident frequency,
# the reduced visibility from low sun, or the greyness of twilight, increased volumes,
# traffic type with single occupant commuter cars; we can separate this from a rush hour
# effect by adding a period that is fixed in time, say 7:30-9am and 5-6:30pm.
#
# Take the dates and times and location we'll use these to find sun times

geosp <- data.frame(datetime = accidents$accident_time,
  lat = accidents$Latitude,
  lon <- accidents$Longitude)

# Double check the column names are what we need for the inputs of
# suncalc::getSunlightTimes()

colnames(geosp) <- c("datetime", "lat", "lon")

# Extract the date

geosp$date <- date(geosp$datetime)

# Now calculate sun times
# Note: There are other options we could pick for defining the phases of the
# day, we could analyse the impact of the choice we have made however it
# is unlikely to be material

suntimes <- getSunlightTimes(data=geosp,
                 keep = c("sunset", "sunriseEnd",
                          "nightEnd", "night", "nadir"))

#Find time of day using the rules:
# if no night then
#   if we are before sunrise end, or after sunset,
#     check which side of nadir we are and make dawn or dusk
#   otherwise day time
# otherwise if between night and nightEnd then night
# if between night end and sunrise end then dawn,
# if between sunset and night then dusk,
# otherwise day time
# Note simplification as we are only looking at times on given
# day, rather than previous day - assumes sun times change slowly
# so sunset today is a good proxy for sunset yesterday, etc.

geosp$time_of_day <- factor(
  ifelse(is.na(suntimes$night),
```

```
        ifelse(geosp$datetime < suntimes$sunriseEnd | geosp$datetime > suntimes$sunset,
               ifelse(geosp$datetime < suntimes$nadir, "Dawn", "Dusk"), "Day"),
        ifelse(geosp$datetime < suntimes$nightEnd | geosp$datetime > suntimes$night, "Night",
               ifelse(geosp$datetime < suntimes$sunriseEnd, "Dawn",
                      ifelse(geosp$datetime > suntimes$sunset,"Dusk", "Day")))))

# We can then copy this derived data items back to the accidents dataset
accidents$time_of_day <- geosp$time_of_day

#Garbage collection
rm(suntimes, geosp)

#Add rush hour flag
accidents$rush_hour <- ifelse(is.na(accidents$accident_time), "N",
                              ifelse(accidents$Day_of_Week %in% c(1, 7), "N",
                                     ifelse(hour(accidents$accident_time) == 8 |
                                       (hour(accidents$accident_time) == 7 &
                                          minute(accidents$accident_time) >= 30), "AM",
                                       ifelse(hour(accidents$accident_time) == 17 |
                                        (hour(accidents$accident_time) == 18 &
                                           minute(accidents$accident_time) <= 30), "PM","N"))))

accidents$year <- year(accidents$accident_time)
accidents$month <- month(accidents$accident_time)
```

Creating factors

3.10 We can convert numeric keys to factors to flag to R that these are discrete values and their meaning may be different to their numeric value: the information conveyed by a key value of 3 may not have any relationship to the information associated with a key value of 2, despite the numerical relationship. For severity a higher key value indicates a more severe accident, 1 (fatal) is worse that 2 (serious), which is in turn worse than 3 (slight).

```
# Convert numeric variables to factors
accidents$Accident_Severity <- factor(accidents$Accident_Severity, ordered = T)
accidents$Junction_Detail <- factor(accidents$Junction_Detail)
accidents$Weather_Conditions <- factor(accidents$Weather_Conditions)
accidents$Light_Conditions <- factor(accidents$Light_Conditions)
accidents$Road_Surface_Conditions <- factor(accidents$Road_Surface_Conditions )
accidents$Carriageway_Hazards <- factor(accidents$Carriageway_Hazards )
accidents$Urban_or_Rural_Area <- factor(accidents$Urban_or_Rural_Area)
accidents$rush_hour <- factor(ifelse(is.na(accidents$rush_hour), "N", accidents$rush_hour))
accidents$Speed_limit <- factor(accidents$Speed_limit)
accidents$month <- factor(accidents$month)
accidents$year <- factor(accidents$year)
accidents$Day_of_Week <- factor(accidents$Day_of_Week)
accidents$Vehicle_Type <- factor(accidents$Vehicle_Type)
accidents$Sex_of_Driver <- factor(accidents$Sex_of_Driver)
accidents$Age_Band_of_Driver <- factor(accidents$Age_Band_of_Driver)
accidents$Driver_IMD_Decile <- factor(accidents$Driver_IMD_Decile)

# Add a field to count the number of vehicles after the first
accidents$Other_Vehicles <- accidents$Number_of_Vehicles - 1

# Regularize field names
```

```
accidents <- accidents %>%
  rename(
    Local_Authority_District = `Local_Authority_(District)`,
    Local_Authority_Highway = `Local_Authority_(Highway)`,
    First_Road_Class = `1st_Road_Class`,
    First_Road_Number = `1st_Road_Number`,
    Second_Road_Class = `2nd_Road_Class`,
    Second_Road_Number = `2nd_Road_Number`,
    Pedestrian_Crossing_Control = `Pedestrian_Crossing-Human_Control`,
    Pedestrian_Crossing_Facilities = `Pedestrian_Crossing-Physical_Facilities`,
    Police_Attended_Scene = Did_Police_Officer_Attend_Scene_of_Accident
    )
```

Training and Hold-out datasets

3.11 We split the accident data into training and hold-out datasets, holding about 10% of the data back from our modelling and analysis to validate the model. The RMSE reported will be based on predictions made on the hold-out data based on a model trained on the training set.

```
# Set seed to make partition repeatably random
set.seed(1, sample.kind="Rounding")
# Create sampled partition index
sampled_index <- createDataPartition(y = accidents$Number_of_Casualties,
                                     times = 1, p = 0.1, list = FALSE)
# Create the training set as those obs not in our sample
sampled_training <- accidents[-sampled_index,]
# Create the validation set as those obs in our sample
sampled_validation <- accidents[sampled_index,]
# Check proportion of data in training set
train_propn <- nrow(sampled_training) / nrow(accidents)

# Datasets have been stored to enable easier development, does not need to be run each time

# saveRDS(sampled_training, "training.rds")
# saveRDS(sampled_validation, "validation.rds")

# Copies of these datasets can be downloaded from github using this code

#dl <- tempfile()        # create a tempfile to receive the downloaded file
#download.file("https://github.com/robmeekings/accidents/raw/main/training.rds",
#              dl, mode="wb")  # download the file at the given url
#sampled_training <- readRDS(dl)

#dl <- tempfile()        # create a tempfile to receive the downloaded file
#download.file("https://github.com/robmeekings/accidents/raw/main/validation.rds",
#              dl, mode="wb")  # download the file at the given url
#sampled_validation <- readRDS(dl)

# Garbage collection
rm(accidents)
```

3.12 This segmentation gives us 90% of the data to train the models on and the remainder to validate the models.

4 Exploratory Data Analysis

4.1 The table below lists the data from the first few rows of the the training dataset. The column names are descriptive of their contents and contain foreign keys to reference tables.

```
# Get a sample of the data, pick the first record
t(head(sampled_training, n=1))
```

```
##                                [,1]
## Accident_Index                 "2019010128300"
## Location_Easting_OSGR          "528218"
## Location_Northing_OSGR         "180407"
## Longitude                      "-0.154"
## Latitude                       "51.5"
## Police_Force                   "1"
## Accident_Severity              "3"
## Number_of_Vehicles             "2"
## Number_of_Casualties           "3"
## Date                           "18/02/2019"
## Day_of_Week                    "2"
## Time                           "17:50"
## Local_Authority_District       "1"
## Local_Authority_Highway        "E09000033"
## First_Road_Class               "3"
## First_Road_Number              "4202"
## Road_Type                      "1"
## Speed_limit                    "30"
## Junction_Detail                "1"
## Junction_Control               "2"
## Second_Road_Class              "3"
## Second_Road_Number             "4202"
## Pedestrian_Crossing_Control    "0"
## Pedestrian_Crossing_Facilities "5"
## Light_Conditions               "1"
## Weather_Conditions             "1"
## Road_Surface_Conditions        "1"
## Special_Conditions_at_Site     "0"
## Carriageway_Hazards            "0"
## Urban_or_Rural_Area            "1"
## Police_Attended_Scene          "3"
## LSOA_of_Accident_Location      "E01004762"
## Vehicle_Type                   "9"
## Sex_of_Driver                  "1"
## Age_of_Driver                  "58"
## Age_Band_of_Driver             "9"
## Age_of_Vehicle                 "-1"
## Driver_IMD_Decile              "2"
## accident_time                  "2019-02-18 17:50:00"
## time_of_day                    "Dusk"
## rush_hour                      "PM"
## year                           "2019"
## month                          "2"
## Other_Vehicles                 "1"
```

4.2 The structure of the dataset is:

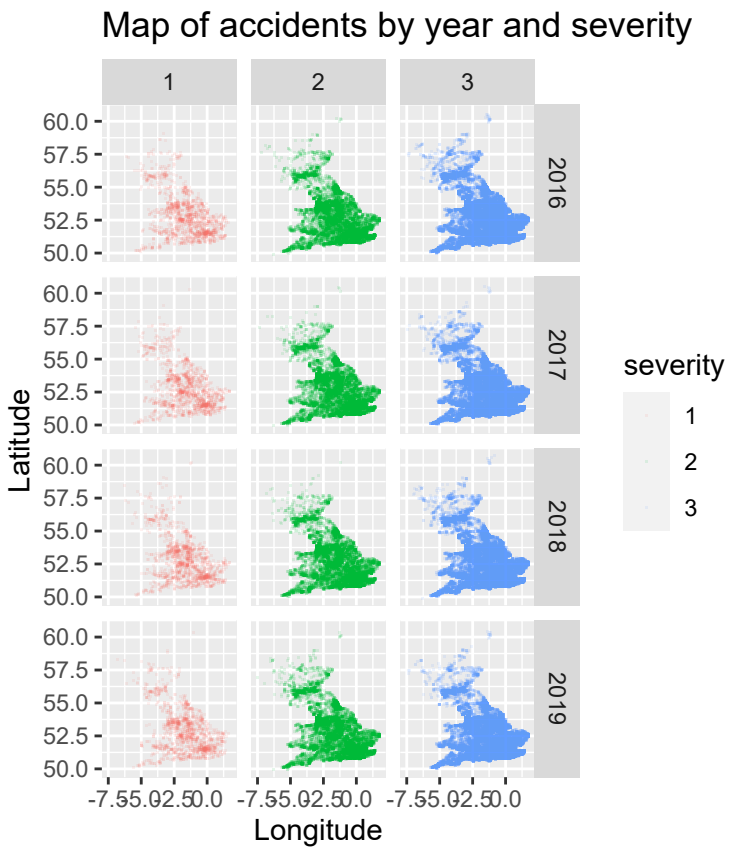
```
# List the variables that make up the input data
str(sampled_training, strict.width="wrap", vec.len=0)
```

```
## Classes 'data.table' and 'data.frame':   456096 obs. of  44 variables:
## $ Accident_Index : chr ...
## $ Location_Easting_OSGR : int NULL ...
## $ Location_Northing_OSGR : int NULL ...
## $ Longitude : num NULL ...
## $ Latitude : num NULL ...
## $ Police_Force : int NULL ...
## $ Accident_Severity : Ord.factor w/ 3 levels "1"<"2"<"3": NULL ...
## $ Number_of_Vehicles : int NULL ...
## $ Number_of_Casualties : int NULL ...
## $ Date : chr ...
## $ Day_of_Week : Factor w/ 7 levels "1","2","3","4",..: NULL ...
## $ Time : chr ...
## $ Local_Authority_District : int NULL ...
## $ Local_Authority_Highway : chr ...
## $ First_Road_Class : int NULL ...
## $ First_Road_Number : int NULL ...
## $ Road_Type : int NULL ...
## $ Speed_limit : Factor w/ 7 levels "-1","20","30",..: NULL ...
## $ Junction_Detail : Factor w/ 10 levels "-1","0","1","2",..: NULL ...
## $ Junction_Control : int NULL ...
## $ Second_Road_Class : int NULL ...
## $ Second_Road_Number : int NULL ...
## $ Pedestrian_Crossing_Control : int NULL ...
## $ Pedestrian_Crossing_Facilities: int NULL ...
## $ Light_Conditions : Factor w/ 6 levels "-1","1","4","5",..: NULL ...
## $ Weather_Conditions : Factor w/ 10 levels "-1","1","2","3",..: NULL ...
## $ Road_Surface_Conditions : Factor w/ 6 levels "-1","1","2","3",..: NULL ...
## $ Special_Conditions_at_Site : int NULL ...
## $ Carriageway_Hazards : Factor w/ 7 levels "-1","0","1","2",..: NULL ...
## $ Urban_or_Rural_Area : Factor w/ 4 levels "-1","1","2","3": NULL ...
## $ Police_Attended_Scene : int NULL ...
## $ LSOA_of_Accident_Location : chr ...
## $ Vehicle_Type : Factor w/ 21 levels "-1","1","2","3",..: NULL ...
## $ Sex_of_Driver : Factor w/ 4 levels "-1","1","2","3": NULL ...
## $ Age_of_Driver : int NULL ...
## $ Age_Band_of_Driver : Factor w/ 12 levels "-1","1","2","3",..: NULL ...
## $ Age_of_Vehicle : int NULL ...
## $ Driver_IMD_Decile : Factor w/ 11 levels "-1","1","2","3",..: NULL ...
## $ accident_time : POSIXct, format: ...
## $ time_of_day : Factor w/ 4 levels "Dawn","Day","Dusk",..: NULL ...
## $ rush_hour : Factor w/ 3 levels "AM","N","PM": NULL ...
## $ year : Factor w/ 4 levels "2016","2017",..: NULL ...
## $ month : Factor w/ 12 levels "1","2","3","4",..: NULL ...
## $ Other_Vehicles : num NULL ...
## - attr(*, ".internal.selfref")=<externalptr>
```

Distribution of accidents

4.3 We can now visualize where these accidents happen using the latitude and longitude, splitting the data by year and severity.

```
# Use ggplot to plot the location of accidents using their longitude and latitude by
# year and severity
sampled_training %>% mutate(severity = paste0(Accident_Severity, "")) %>%
    filter(!is.na(Longitude) & !is.na(year)) %>%
    group_by(Longitude, Latitude, year, severity) %>%
    ggplot(aes(x = Longitude, y = Latitude, group=severity, colour=severity)) +
      geom_point(alpha=0.1, shape=".") +
      coord_fixed() +
      facet_grid(year~severity) +
      labs(title="Map of accidents by year and severity")
```



4.4 We notice that the relative weight of these plots stays fairly steady over time and between severity levels, especially for severity levels 2 and 3. The intensely shaded areas correspond to major population centres - London in the south east, Bristol, Cardiff and Swansea in the south west, then Birmingham in the middle of England, Manchester and Liverpool in the north west of England, and further north, Edinburgh and Glasgow in Scotland. The density of accidents is sufficient that the familiar coastal outline of England is quite well defined, whilst the coasts of Scotland and Wales appear much more sparse.

4.5 The number of casualties is our target in this modelling, so it will help us to look at this variable. The overwhelming majority (79%) of accidents involve only one casualty. This lack of variability in our response variable may make our modelling harder.

```r
# Sumamrize the distribution of numbers of casualties
summary(sampled_training$Number_of_Casualties)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0     1.0     1.0     1.3     1.0    59.0
```

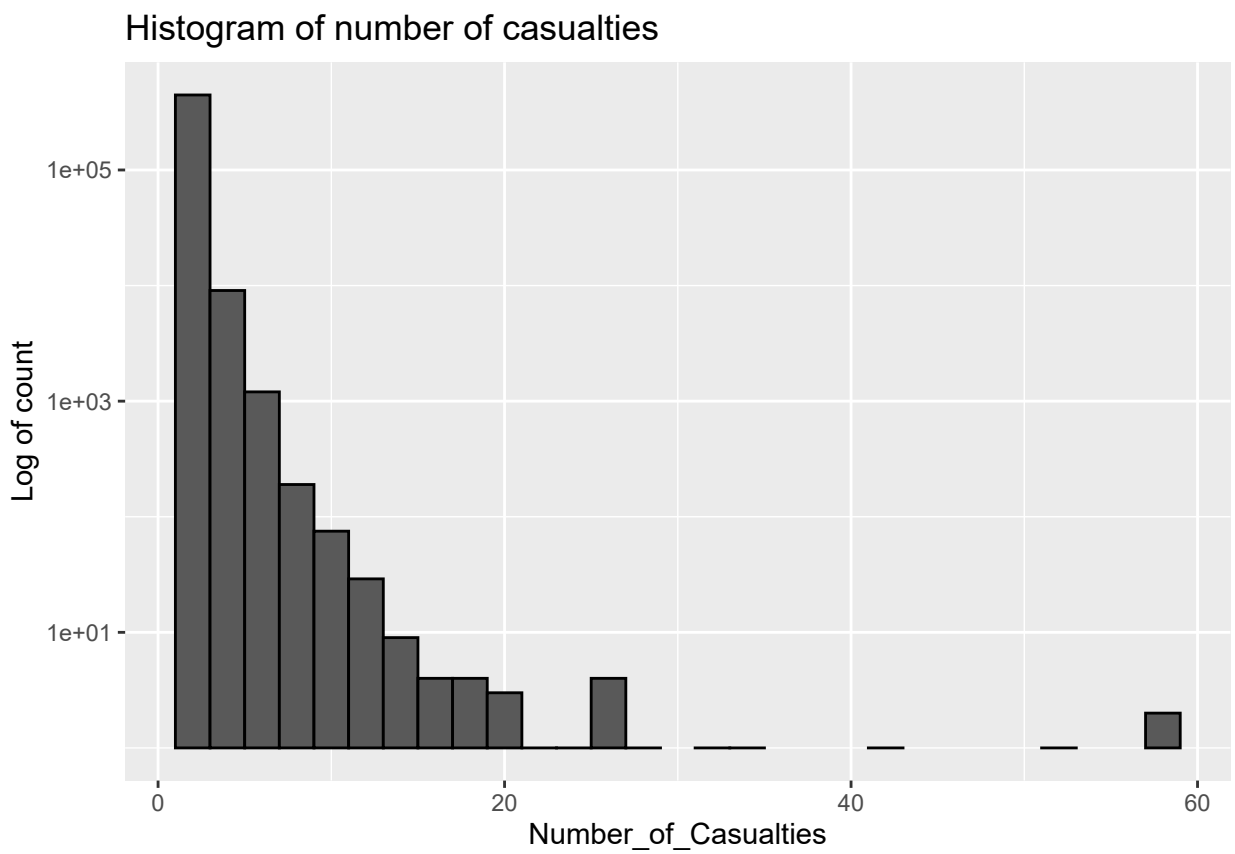
```r
# get the mean and std dev of the number of casualties
stats <- bind_rows(tibble(Statistic ="Mean",
                          Value = mean(sampled_training$Number_of_Casualties)),
                   tibble(Statistic ="Std deviation",
                          Value = sd(sampled_training$Number_of_Casualties)))

stats
```

Statistic	Value
Mean	1.315
Std deviation	0.771

4.6 If we use a logarithmic y axis we can see how the number of casualties is distributed:

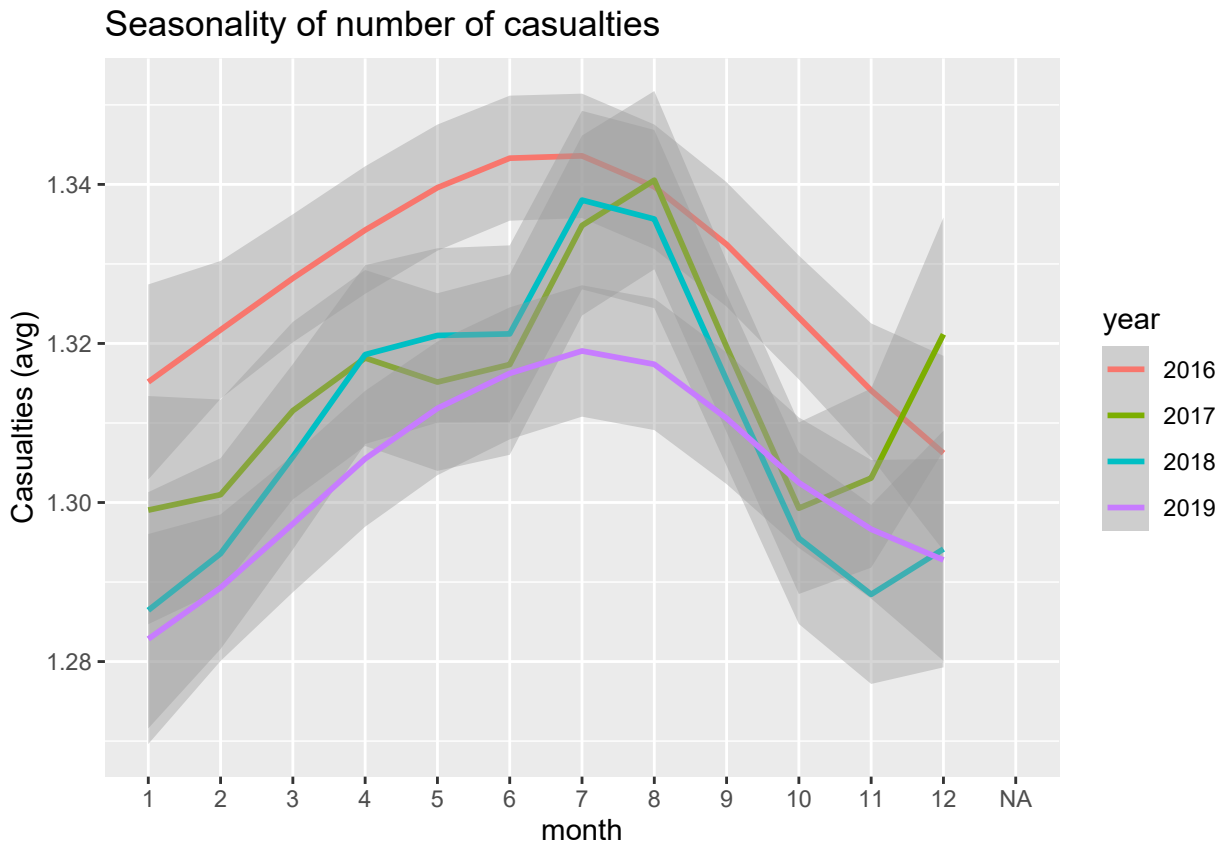
```r
# Histogram of number of casualties, with log y-axis
sampled_training %>%
  ggplot(aes(Number_of_Casualties)) +
  geom_histogram(bins = 30, color = "black") +
  scale_y_log10() +
  labs(title="Histogram of number of casualties", y="Log of count")
```



4.7 We can look at how casualties vary over time, if we plot subsequent years over the same range of months we find

a clear pattern in greater casualties in the summer months.

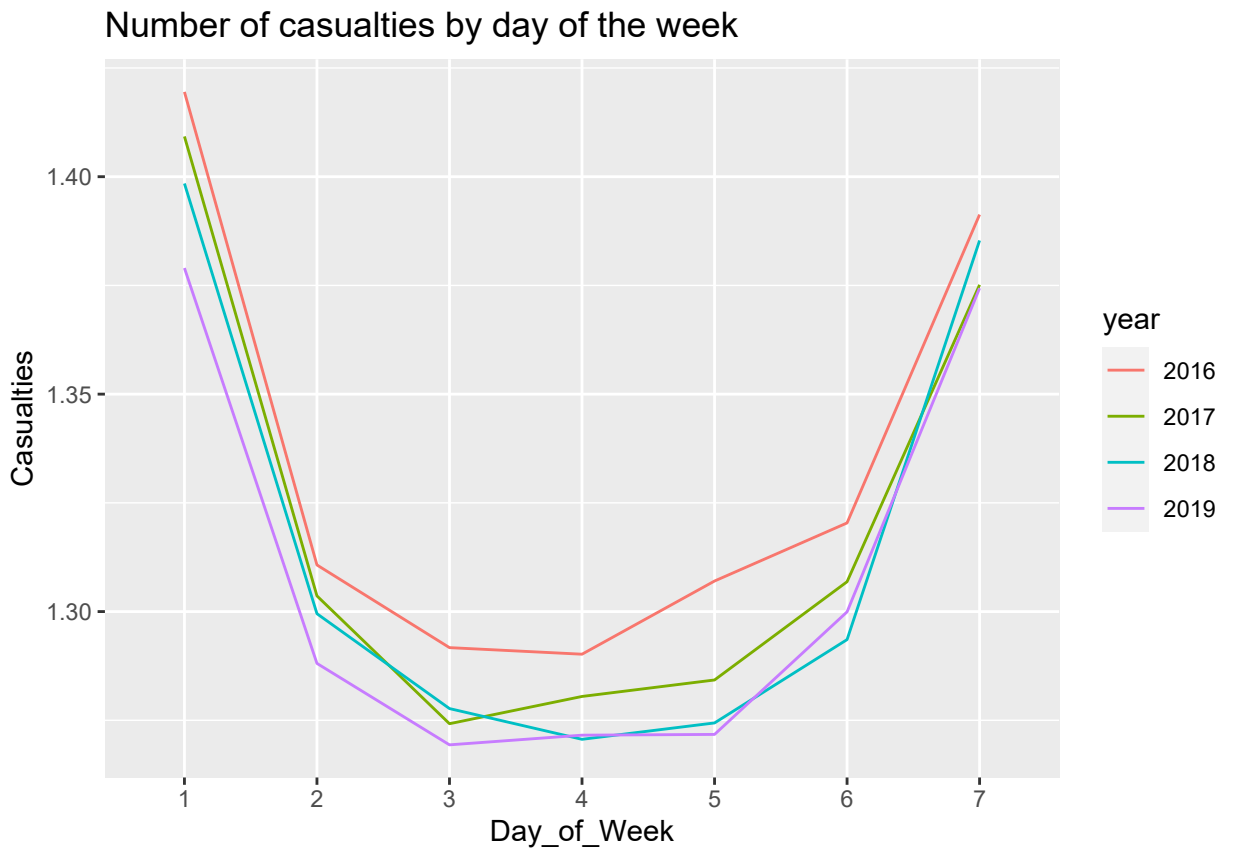
```
sampled_training %>%
  ggplot(aes(x=month, y=Number_of_Casualties, group=year, colour=year)) +
    geom_smooth() +
    labs(title="Seasonality of number of casualties", y="Casualties (avg)")
```



Notice that the average number of casualties per accident, appears to be decreasing year-on-year, if the total number of accidents is also decreasing this should be a good sign, although the effect is very small, 2% over the period. Notice too, that the monthly fluctations due to seasonality are also very small, of the order of 2% , too.

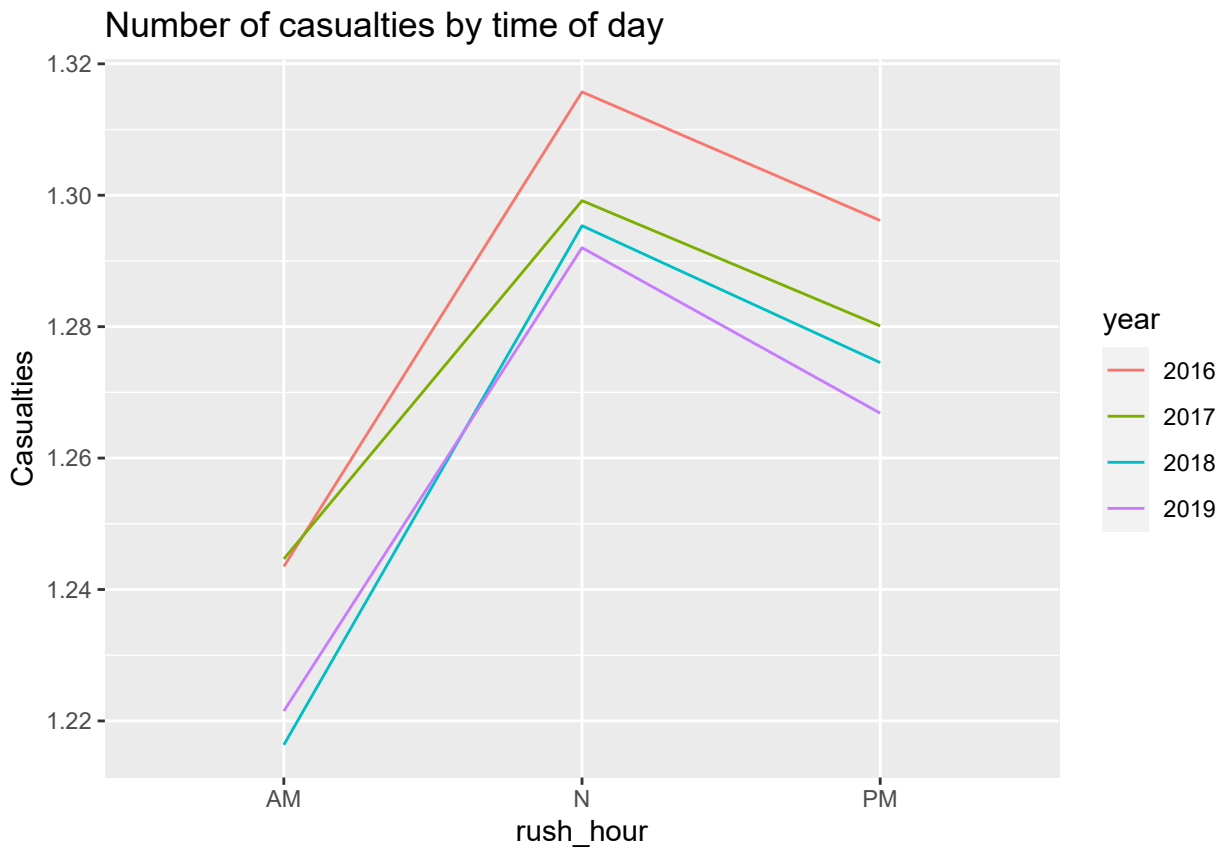
4.8 The distribution through the week appears fairly stable over time, with Sundays (1) being the worst day for the number of casualties, followed by Saturday (7), and with lower levels through the rest of the week, with mid-week appearing to have the least casualties per accident. This suggests that there may be variations in road-use between weekdays and the weekend: single occupant commuter cars may bring down the number of casualties, if not the number of accidents, mid-week.

```
sampled_training %>% filter(!is.na(Day_of_Week) & !is.na(year)) %>%
  group_by(Day_of_Week, year) %>%
  summarize(cbar = mean(Number_of_Casualties)) %>%
  ungroup() %>%
  ggplot(aes(x=Day_of_Week, y=cbar, group=year, colour=year)) +
  geom_line() +
  labs(title="Number of casualties by day of the week", y="Casualties")
```



4.9 Having seen an apparent relationship between the number of casualties and the day of the week, does rush hour help to explain the number of casualties. In 3.9, above, we chose defined rush-hour as the periods 7:30-9am and 5-6:30pm. These should be the periods when there are the most vehicles on the roads.

```
sampled_training %>% filter(!is.na(rush_hour) & !is.na(year) & !Day_of_Week %in% c(1,7)) %>%
  group_by(rush_hour, year) %>%
  summarize(cbar = mean(Number_of_Casualties)) %>%
  ungroup() %>%
  ggplot(aes(x=rush_hour, y=cbar, group=year, colour=year)) +
  geom_line() +
  labs(title="Number of casualties by time of day", y="Casualties")
```



4.10 We see that rush hour driving has lower average casualty rates than at other times of the day. This suggests that there is a single occupancy vehicle effect, car pooling is not particularly widespread in the UK. The trend for the evening rush hour to be more dangerous than the morning, along with the rising trend in numbers of casualties per accident through the course of the week, might suggest that driver fatigue could play a role in the number of casualties.

5 Model development

Model assessment

5.1 We will assess the models we develop by looking at the *square root of the mean of the squared errors* (RMSE), a better model will have a lower RMSE. The code to define a RMSE function, below, is adapted from the course lecture notes, the rm.na parameter in the mean function call omits missing values from the calculation.

```
# define a function to calculate rmse between observed and predicted data
# the function should ignore missing values
RMSE <- function(true_ratings, predicted_ratings){
    sqrt(mean((true_ratings - predicted_ratings)^2, rm.na=TRUE))
}
```

The simplest model

5.2 The simplest model assumes that response values are equal to a constant and a normal noise term. This is a stretch with a left-censored count values, where we don't have negative numbers of casualties, but gives us a base line to compare later models to.

```
# calcualte the mean number of casualties
mean_casualties <- mean(sampled_training$Number_of_Casualties)

# calculate an RMSE score for using the mean as a predictor
rmse_mean <- RMSE(sampled_validation$Number_of_Casualties, mean_casualties)

# store the results to a tibble, we will add rows to this as we go
rmse_results <- tibble(method = "Just the mean", RMSE = rmse_mean)
```

5.3 We find the mean is 1.315 and when we apply this to the data as our predicted value and calculate the RMSE we get an RMSE of 0.746 .

Linear model for variable selection

5.4 We will start by looking at the results of fitting a linear multivariate model to the data to look for correlations, taken with out exploratory analysis, above, this should point to some approaches to modelling and help highlight explanatory variables.

```
# fit a linear model using the lm() function to predict the number of casualties
model.linear <- lm(Number_of_Casualties ~
            Police_Force +
            Other_Vehicles +
            Speed_limit +
            First_Road_Class +
            Second_Road_Class +
            Weather_Conditions +
            Light_Conditions +
            Road_Surface_Conditions +
            Carriageway_Hazards +
            Urban_or_Rural_Area +
            time_of_day +
            Day_of_Week +
            rush_hour +
            Age_Band_of_Driver +
```

```
            Sex_of_Driver +
            Vehicle_Type +
            Age_of_Vehicle +
            Driver_IMD_Decile +
            year +
            month,
          data=sampled_training)

# Apply the model to the validation data to test it
pred.linear <- data.frame(casualties =
                            predict.lm(model.linear, newdata=sampled_validation))

# Flag any records without a prediction and exclude from RMSE
miss_idx <- is.na(pred.linear$casualties)

# Calculate the RMSE for the model
rmse_linear <- RMSE(sampled_validation[!miss_idx,]$Number_of_Casualties, pred.linear[!miss_idx,])

# Store the RMSE value in our results tibble
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Linear model",
                                     RMSE = rmse_linear))

# Output the coefficients of the linear model
summary(model.linear)
```

```
##
## Call:
## lm(formula = Number_of_Casualties ~ Police_Force + Other_Vehicles +
##     Speed_limit + First_Road_Class + Second_Road_Class + Weather_Conditions +
##     Light_Conditions + Road_Surface_Conditions + Carriageway_Hazards +
##     Urban_or_Rural_Area + time_of_day + Day_of_Week + rush_hour +
##     Age_Band_of_Driver + Sex_of_Driver + Vehicle_Type + Age_of_Vehicle +
##     Driver_IMD_Decile + year + month, data = sampled_training)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -3.80  -0.36  -0.20   0.05  57.19
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)               3.04e+00   7.88e-01    3.85  0.00012 ***
## Police_Force             -2.44e-04   4.94e-05   -4.95  7.4e-07 ***
## Other_Vehicles            2.31e-01   1.58e-03  146.62  < 2e-16 ***
## Speed_limit20             2.24e-02   8.43e-02    0.27  0.79030
## Speed_limit30             6.56e-02   8.42e-02    0.78  0.43606
## Speed_limit40             1.68e-01   8.43e-02    1.99  0.04675 *
## Speed_limit50             2.03e-01   8.44e-02    2.41  0.01601 *
## Speed_limit60             2.59e-01   8.43e-02    3.07  0.00212 **
## Speed_limit70             2.40e-01   8.44e-02    2.84  0.00445 **
## First_Road_Class         -7.62e-03   8.52e-04   -8.94  < 2e-16 ***
## Second_Road_Class        -8.10e-05   3.60e-04   -0.23  0.82195
## Weather_Conditions1       1.35e-01   1.71e-01    0.79  0.43024
## Weather_Conditions2       1.20e-01   1.71e-01    0.71  0.48077
```

```
## Weather_Conditions3       8.27e-02   1.72e-01    0.48  0.63019
## Weather_Conditions4       1.43e-01   1.71e-01    0.84  0.40222
## Weather_Conditions5       1.23e-01   1.71e-01    0.72  0.47042
## Weather_Conditions6       1.50e-01   1.74e-01    0.86  0.39003
## Weather_Conditions7       8.57e-02   1.71e-01    0.50  0.61724
## Weather_Conditions8       1.17e-01   1.71e-01    0.68  0.49366
## Weather_Conditions9       8.75e-02   1.71e-01    0.51  0.60818
## Light_Conditions1        -1.09e-01   2.69e-01   -0.41  0.68472
## Light_Conditions4        -9.68e-02   2.69e-01   -0.36  0.71939
## Light_Conditions5        -1.03e-01   2.70e-01   -0.38  0.70212
## Light_Conditions6        -5.52e-02   2.69e-01   -0.21  0.83753
## Light_Conditions7        -1.32e-01   2.69e-01   -0.49  0.62502
## Road_Surface_Conditions1  2.00e-02   1.40e-02    1.43  0.15348
## Road_Surface_Conditions2  4.63e-02   1.43e-02    3.24  0.00120 **
## Road_Surface_Conditions3  2.97e-02   2.68e-02    1.11  0.26907
## Road_Surface_Conditions4  2.51e-02   1.71e-02    1.47  0.14241
## Road_Surface_Conditions5  4.10e-02   3.53e-02    1.16  0.24541
## Carriageway_Hazards0      1.38e-02   1.30e-02    1.06  0.28863
## Carriageway_Hazards1     -2.11e-02   2.80e-02   -0.75  0.45233
## Carriageway_Hazards2      3.19e-02   1.76e-02    1.82  0.06922 .
## Carriageway_Hazards3      7.86e-02   3.03e-02    2.59  0.00959 **
## Carriageway_Hazards6      6.64e-02   2.81e-02    2.36  0.01816 *
## Carriageway_Hazards7     -4.84e-02   2.26e-02   -2.15  0.03188 *
## Urban_or_Rural_Area1     -2.47e+00   7.33e-01   -3.37  0.00075 ***
## Urban_or_Rural_Area2     -2.41e+00   7.33e-01   -3.29  0.00101 **
## Urban_or_Rural_Area3     -2.47e+00   1.04e+00   -2.38  0.01710 *
## time_of_dayDay            5.34e-02   6.83e-03    7.81  5.5e-15 ***
## time_of_dayDusk           8.04e-02   6.82e-03   11.78  < 2e-16 ***
## time_of_dayNight          9.65e-02   6.81e-03   14.18  < 2e-16 ***
## Day_of_Week2             -7.55e-02   4.43e-03  -17.04  < 2e-16 ***
## Day_of_Week3             -9.53e-02   4.40e-03  -21.67  < 2e-16 ***
## Day_of_Week4             -9.41e-02   4.38e-03  -21.49  < 2e-16 ***
## Day_of_Week5             -9.01e-02   4.36e-03  -20.65  < 2e-16 ***
## Day_of_Week6             -7.51e-02   4.28e-03  -17.52  < 2e-16 ***
## Day_of_Week7             -1.25e-02   4.42e-03   -2.83  0.00473 **
## rush_hourN                5.31e-02   4.04e-03   13.16  < 2e-16 ***
## rush_hourPM               1.91e-02   5.15e-03    3.70  0.00021 ***
## Age_Band_of_Driver1       1.94e-02   1.10e-01    0.18  0.85931
## Age_Band_of_Driver2       1.85e-02   3.48e-02    0.53  0.59460
## Age_Band_of_Driver3       8.74e-02   1.71e-02    5.12  3.0e-07 ***
## Age_Band_of_Driver4       1.64e-01   6.54e-03   25.15  < 2e-16 ***
## Age_Band_of_Driver5       1.10e-01   6.12e-03   17.93  < 2e-16 ***
## Age_Band_of_Driver6       7.81e-02   5.59e-03   13.98  < 2e-16 ***
## Age_Band_of_Driver7       6.24e-02   5.79e-03   10.78  < 2e-16 ***
## Age_Band_of_Driver8       3.91e-02   5.91e-03    6.62  3.6e-11 ***
## Age_Band_of_Driver9       3.71e-02   6.35e-03    5.84  5.2e-09 ***
## Age_Band_of_Driver10      6.53e-02   7.18e-03    9.09  < 2e-16 ***
## Age_Band_of_Driver11      9.40e-02   7.75e-03   12.13  < 2e-16 ***
## Sex_of_Driver1            2.16e-01   1.61e-01    1.34  0.17875
## Sex_of_Driver2            2.20e-01   1.61e-01    1.37  0.17148
## Sex_of_Driver3            1.65e-01   1.61e-01    1.03  0.30331
## Vehicle_Type1            -1.59e-01   2.65e-02   -6.01  1.8e-09 ***
## Vehicle_Type2            -1.65e-01   2.89e-02   -5.72  1.1e-08 ***
## Vehicle_Type3            -1.65e-01   2.66e-02   -6.21  5.3e-10 ***
```

```
## Vehicle_Type4           -1.34e-01   2.83e-02   -4.74  2.2e-06 ***
## Vehicle_Type5           -1.52e-01   2.69e-02   -5.67  1.4e-08 ***
## Vehicle_Type8            5.92e-02   2.68e-02    2.21  0.02742 *
## Vehicle_Type9            7.52e-02   2.60e-02    2.90  0.00379 **
## Vehicle_Type10           2.99e-01   3.63e-02    8.23  < 2e-16 ***
## Vehicle_Type11           2.09e-01   2.68e-02    7.79  6.8e-15 ***
## Vehicle_Type16          -2.28e-01   9.75e-02   -2.34  0.01917 *
## Vehicle_Type17          -1.49e-01   3.70e-02   -4.03  5.7e-05 ***
## Vehicle_Type18           3.24e-02   1.19e-01    0.27  0.78517
## Vehicle_Type19           1.88e-02   2.63e-02    0.71  0.47475
## Vehicle_Type20          -4.45e-03   3.04e-02   -0.15  0.88352
## Vehicle_Type21          -3.28e-02   2.74e-02   -1.20  0.23148
## Vehicle_Type22          -2.93e-02   4.50e-02   -0.65  0.51528
## Vehicle_Type23          -9.81e-02   7.49e-02   -1.31  0.19027
## Vehicle_Type90           5.71e-02   2.94e-02    1.94  0.05245 .
## Vehicle_Type97          -7.90e-02   3.58e-02   -2.21  0.02715 *
## Vehicle_Type98          -1.26e-02   2.99e-02   -0.42  0.67267
## Age_of_Vehicle           1.81e-03   1.95e-04    9.30  < 2e-16 ***
## Driver_IMD_Decile1       5.82e-02   4.85e-03   12.00  < 2e-16 ***
## Driver_IMD_Decile2       2.57e-02   4.69e-03    5.47  4.4e-08 ***
## Driver_IMD_Decile3       1.39e-02   4.71e-03    2.95  0.00319 **
## Driver_IMD_Decile4       9.69e-03   4.74e-03    2.04  0.04087 *
## Driver_IMD_Decile5       5.55e-03   4.83e-03    1.15  0.25082
## Driver_IMD_Decile6      -3.60e-03   4.90e-03   -0.74  0.46180
## Driver_IMD_Decile7      -1.39e-02   5.03e-03   -2.77  0.00568 **
## Driver_IMD_Decile8      -6.03e-03   5.12e-03   -1.18  0.23846
## Driver_IMD_Decile9      -9.61e-03   5.24e-03   -1.83  0.06661 .
## Driver_IMD_Decile10     -1.90e-02   5.50e-03   -3.46  0.00055 ***
## year2017                -4.34e-03   3.02e-03   -1.44  0.15027
## year2018                -9.63e-03   3.06e-03   -3.15  0.00163 **
## year2019                -1.33e-02   3.10e-03   -4.29  1.8e-05 ***
## month2                   2.48e-03   5.48e-03    0.45  0.65048
## month3                   1.25e-02   5.50e-03    2.28  0.02262 *
## month4                   3.48e-02   5.63e-03    6.18  6.6e-10 ***
## month5                   3.11e-02   5.51e-03    5.64  1.7e-08 ***
## month6                   3.35e-02   5.52e-03    6.07  1.3e-09 ***
## month7                   4.60e-02   5.49e-03    8.37  < 2e-16 ***
## month8                   5.27e-02   5.56e-03    9.49  < 2e-16 ***
## month9                   1.53e-02   5.44e-03    2.81  0.00493 **
## month10                  1.64e-02   5.37e-03    3.06  0.00219 **
## month11                 -5.33e-03   5.22e-03   -1.02  0.30697
## month12                  3.73e-03   5.36e-03    0.70  0.48643
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.733 on 455655 degrees of freedom
##   (332 observations deleted due to missingness)
## Multiple R-squared:  0.0966, Adjusted R-squared:  0.0964
## F-statistic:  451 on 108 and 455655 DF,  p-value: <2e-16
```

5.5 Although this linear model is not intended as, in any way, a final model, we can still use it to make predictions and find it produces an RMSE of 0.707 , an improvement on the simplest model.

5.6 We are most interested in variables with large effects and high significance (low p-values). Here visualizing the variables on a scatter plot can help.

5.7 We observe that the variables xxx have a relatively small effect and significance and we exclude these from the modelling.

Poisson and negative binomial models

5.8 The distributions most often used for modelling count data are the poisson and the negative binomial. We will start by fitting both of these and comparing the results.

```
# fit a poisson glm to predict the number of casualties
model.poisson <- glm(Number_of_Casualties ~ Police_Force +
                     Other_Vehicles +
                     Speed_limit +
                     First_Road_Class +
                     Second_Road_Class +
                     First_Road_Class:Second_Road_Class +
                     Road_Surface_Conditions +
                     time_of_day +
                     Day_of_Week +
                     rush_hour +
                     Age_Band_of_Driver +
                     Vehicle_Type +
                     Age_of_Vehicle +
                     Driver_IMD_Decile +
                     year +
                     month,
                 data=sampled_training,
                 family=poisson(link=log))

# Apply the model to the validation data to test it
pred.poisson <- data.frame(casualties =
                             predict(model.poisson, newdata=sampled_validation))

# Flag any records without a prediction and exclude from RMSE
miss_idx <- is.na(pred.poisson$casualties)

# Calculate the RMSE for the model
rmse_poisson <- RMSE(sampled_validation[!miss_idx,]$Number_of_Casualties, pred.poisson[!miss_idx,])

# Store the RMSE value in our results tibble
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Poisson model",
                                     RMSE = rmse_poisson))

# Output the coefficients of the poisson model
summary(model.poisson)
```

```
##
## Call:
## glm(formula = Number_of_Casualties ~ Police_Force + Other_Vehicles +
##     Speed_limit + First_Road_Class + Second_Road_Class + First_Road_Class:Second_Road_Class +
##     Road_Surface_Conditions + time_of_day + Day_of_Week + rush_hour +
##     Age_Band_of_Driver + Vehicle_Type + Age_of_Vehicle + Driver_IMD_Decile +
##     year + month, family = poisson(link = log), data = sampled_training)
##
## Deviance Residuals:
```

```
##    Min      1Q  Median      3Q     Max
## -6.871  -0.314  -0.183   0.008  17.485
##
## Coefficients:
##                                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)                    -1.89e-01   1.17e-01   -1.62  0.10464
## Police_Force                   -6.48e-05   5.82e-05   -1.11  0.26560
## Other_Vehicles                  1.49e-01   1.59e-03   94.14  < 2e-16 ***
## Speed_limit20                   3.21e-02   1.11e-01    0.29  0.77153
## Speed_limit30                   7.66e-02   1.10e-01    0.69  0.48789
## Speed_limit40                   1.71e-01   1.11e-01    1.55  0.12183
## Speed_limit50                   2.04e-01   1.11e-01    1.84  0.06591 .
## Speed_limit60                   2.55e-01   1.11e-01    2.31  0.02084 *
## Speed_limit70                   2.17e-01   1.11e-01    1.96  0.04949 *
## First_Road_Class               -1.12e-02   1.26e-03   -8.94  < 2e-16 ***
## Second_Road_Class              -8.16e-03   1.32e-03   -6.18  6.6e-10 ***
## Road_Surface_Conditions1        6.49e-02   1.40e-02    4.65  3.4e-06 ***
## Road_Surface_Conditions2        8.01e-02   1.42e-02    5.66  1.5e-08 ***
## Road_Surface_Conditions3        5.07e-02   2.46e-02    2.06  0.03917 *
## Road_Surface_Conditions4        6.61e-02   1.80e-02    3.67  0.00024 ***
## Road_Surface_Conditions5        7.78e-02   3.93e-02    1.98  0.04768 *
## time_of_dayDay                  3.60e-02   7.52e-03    4.79  1.7e-06 ***
## time_of_dayDusk                 6.32e-02   8.20e-03    7.71  1.3e-14 ***
## time_of_dayNight                7.76e-02   8.00e-03    9.70  < 2e-16 ***
## Day_of_Week2                   -5.58e-02   5.19e-03  -10.75  < 2e-16 ***
## Day_of_Week3                   -7.16e-02   5.17e-03  -13.85  < 2e-16 ***
## Day_of_Week4                   -7.03e-02   5.14e-03  -13.66  < 2e-16 ***
## Day_of_Week5                   -6.71e-02   5.12e-03  -13.11  < 2e-16 ***
## Day_of_Week6                   -5.58e-02   5.00e-03  -11.14  < 2e-16 ***
## Day_of_Week7                   -9.69e-03   5.11e-03   -1.90  0.05779 .
## rush_hourN                      4.37e-02   4.92e-03    8.88  < 2e-16 ***
## rush_hourPM                     1.84e-02   6.24e-03    2.95  0.00316 **
## Age_Band_of_Driver1             4.78e-02   1.45e-01    0.33  0.74109
## Age_Band_of_Driver2             4.97e-02   4.66e-02    1.07  0.28589
## Age_Band_of_Driver3             1.06e-01   2.18e-02    4.86  1.2e-06 ***
## Age_Band_of_Driver4             1.57e-01   6.79e-03   23.07  < 2e-16 ***
## Age_Band_of_Driver5             1.15e-01   6.27e-03   18.40  < 2e-16 ***
## Age_Band_of_Driver6             9.14e-02   5.62e-03   16.27  < 2e-16 ***
## Age_Band_of_Driver7             7.96e-02   5.89e-03   13.51  < 2e-16 ***
## Age_Band_of_Driver8             6.19e-02   6.06e-03   10.23  < 2e-16 ***
## Age_Band_of_Driver9             6.08e-02   6.66e-03    9.13  < 2e-16 ***
## Age_Band_of_Driver10            8.29e-02   7.71e-03   10.75  < 2e-16 ***
## Age_Band_of_Driver11            1.05e-01   8.35e-03   12.59  < 2e-16 ***
## Vehicle_Type1                  -1.30e-01   3.42e-02   -3.79  0.00015 ***
## Vehicle_Type2                  -1.34e-01   3.74e-02   -3.58  0.00034 ***
## Vehicle_Type3                  -1.31e-01   3.43e-02   -3.81  0.00014 ***
## Vehicle_Type4                  -1.05e-01   3.64e-02   -2.88  0.00402 **
## Vehicle_Type5                  -1.14e-01   3.46e-02   -3.29  0.00099 ***
## Vehicle_Type8                   4.93e-02   3.45e-02    1.43  0.15241
## Vehicle_Type9                   6.87e-02   3.35e-02    2.05  0.04021 *
## Vehicle_Type10                  2.26e-01   4.36e-02    5.18  2.2e-07 ***
## Vehicle_Type11                  1.56e-01   3.44e-02    4.53  6.0e-06 ***
## Vehicle_Type16                 -1.58e-01   1.28e-01   -1.24  0.21593
## Vehicle_Type17                 -8.59e-02   4.64e-02   -1.85  0.06428 .
```

```
## Vehicle_Type18                      3.26e-03   1.56e-01    0.02  0.98335
## Vehicle_Type19                      2.70e-02   3.39e-02    0.80  0.42513
## Vehicle_Type20                      1.08e-02   3.84e-02    0.28  0.77928
## Vehicle_Type21                     -8.46e-03   3.50e-02   -0.24  0.80906
## Vehicle_Type22                     -3.95e-02   5.91e-02   -0.67  0.50370
## Vehicle_Type23                     -7.74e-02   9.65e-02   -0.80  0.42248
## Vehicle_Type90                      5.28e-02   3.74e-02    1.41  0.15793
## Vehicle_Type97                     -6.39e-02   4.62e-02   -1.38  0.16651
## Vehicle_Type98                      7.10e-03   3.80e-02    0.19  0.85189
## Age_of_Vehicle                      1.56e-03   2.29e-04    6.80  1.1e-11 ***
## Driver_IMD_Decile1                  4.80e-02   5.67e-03    8.47  < 2e-16 ***
## Driver_IMD_Decile2                  2.51e-02   5.52e-03    4.55  5.4e-06 ***
## Driver_IMD_Decile3                  1.56e-02   5.56e-03    2.81  0.00497 **
## Driver_IMD_Decile4                  1.40e-02   5.57e-03    2.52  0.01172 *
## Driver_IMD_Decile5                  1.20e-02   5.67e-03    2.11  0.03471 *
## Driver_IMD_Decile6                  5.40e-03   5.75e-03    0.94  0.34716
## Driver_IMD_Decile7                 -1.62e-03   5.92e-03   -0.27  0.78404
## Driver_IMD_Decile8                  4.13e-03   6.00e-03    0.69  0.49200
## Driver_IMD_Decile9                  1.59e-03   6.15e-03    0.26  0.79535
## Driver_IMD_Decile10                -5.49e-03   6.47e-03   -0.85  0.39670
## year2017                           -3.70e-03   3.58e-03   -1.03  0.30097
## year2018                           -8.55e-03   3.62e-03   -2.36  0.01829 *
## year2019                           -1.15e-02   3.68e-03   -3.12  0.00180 **
## month2                              2.23e-03   6.55e-03    0.34  0.73289
## month3                              9.53e-03   6.56e-03    1.45  0.14617
## month4                              2.75e-02   6.67e-03    4.12  3.7e-05 ***
## month5                              2.53e-02   6.55e-03    3.86  0.00011 ***
## month6                              2.80e-02   6.56e-03    4.26  2.0e-05 ***
## month7                              3.73e-02   6.51e-03    5.74  9.7e-09 ***
## month8                              4.24e-02   6.56e-03    6.46  1.0e-10 ***
## month9                              1.34e-02   6.47e-03    2.08  0.03770 *
## month10                             1.39e-02   6.39e-03    2.17  0.02973 *
## month11                            -3.51e-03   6.25e-03   -0.56  0.57464
## month12                             2.76e-03   6.39e-03    0.43  0.66557
## First_Road_Class:Second_Road_Class  1.96e-03   2.88e-04    6.82  9.2e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 138519  on 455763  degrees of freedom
## Residual deviance: 119707  on 455680  degrees of freedom
##   (332 observations deleted due to missingness)
## AIC: 1105730
##
## Number of Fisher Scoring iterations: 5
```

5.9 This poisson model gives a RMSE of 1.271 , not as good as our previous linear model.

```
# fit a negative binomial glm to predict the number of casualties
model.negbin <- glm.nb(Number_of_Casualties ~ Police_Force +
                    Other_Vehicles +
                    Speed_limit +
                    First_Road_Class +
                    Second_Road_Class +
```

```
                    First_Road_Class:Second_Road_Class +
                    Road_Surface_Conditions +
                    Carriageway_Hazards +
                    time_of_day * Day_of_Week +
                    rush_hour +
                    Age_Band_of_Driver +
                    Vehicle_Type +
                    Age_of_Vehicle +
                    Driver_IMD_Decile +
                    year +
                    month,
                  data=sampled_training)

# Apply the model to the validation data to test it
pred.negbin <- data.frame(casualties =
                            predict(model.negbin, newdata=sampled_validation))

# Flag any records without a prediction and exclude from RMSE
miss_idx <- is.na(pred.negbin$casualties)

# Calculate the RMSE for the model
rmse_negbin <- RMSE(sampled_validation[!miss_idx,]$Number_of_Casualties, pred.negbin[!miss_idx,])

# Store the RMSE value in our results tibble
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Negative binomial model",
                                     RMSE = rmse_negbin))

# Output the coefficients of the negative binomial model
summary(model.negbin)
```

```
##
## Call:
## glm.nb(formula = Number_of_Casualties ~ Police_Force + Other_Vehicles +
##     Speed_limit + First_Road_Class + Second_Road_Class + First_Road_Class:Second_Road_Class +
##     Road_Surface_Conditions + Carriageway_Hazards + time_of_day *
##     Day_of_Week + rush_hour + Age_Band_of_Driver + Vehicle_Type +
##     Age_of_Vehicle + Driver_IMD_Decile + year + month, data = sampled_training,
##     init.theta = 65461.68457, link = log)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -6.885  -0.314  -0.183   0.008  17.477
##
## Coefficients:
##                                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)                       -1.81e-01   1.19e-01   -1.53  0.12637
## Police_Force                      -6.71e-05   5.83e-05   -1.15  0.24999
## Other_Vehicles                     1.49e-01   1.59e-03   93.82  < 2e-16 ***
## Speed_limit20                      3.08e-02   1.11e-01    0.28  0.78039
## Speed_limit30                      7.54e-02   1.11e-01    0.68  0.49509
## Speed_limit40                      1.70e-01   1.11e-01    1.54  0.12432
## Speed_limit50                      2.03e-01   1.11e-01    1.83  0.06729 .
## Speed_limit60                      2.55e-01   1.11e-01    2.30  0.02122 *
```

```
## Speed_limit70                  2.16e-01   1.11e-01    1.95  0.05079 .
## First_Road_Class              -1.13e-02   1.26e-03   -8.95  < 2e-16 ***
## Second_Road_Class             -8.17e-03   1.32e-03   -6.18  6.4e-10 ***
## Road_Surface_Conditions1       5.10e-02   1.69e-02    3.02  0.00256 **
## Road_Surface_Conditions2       6.62e-02   1.71e-02    3.88  0.00010 ***
## Road_Surface_Conditions3       3.62e-02   2.64e-02    1.37  0.17030
## Road_Surface_Conditions4       5.21e-02   2.04e-02    2.56  0.01060 *
## Road_Surface_Conditions5       6.34e-02   4.04e-02    1.57  0.11674
## Carriageway_Hazards0           2.42e-02   1.66e-02    1.45  0.14647
## Carriageway_Hazards1          -1.00e-02   3.40e-02   -0.29  0.76835
## Carriageway_Hazards2           3.40e-02   2.18e-02    1.56  0.11828
## Carriageway_Hazards3           3.69e-02   3.31e-02    1.11  0.26498
## Carriageway_Hazards6           5.49e-02   3.49e-02    1.57  0.11541
## Carriageway_Hazards7          -1.72e-02   2.76e-02   -0.62  0.53313
## time_of_dayDay                 2.10e-02   2.26e-02    0.93  0.35221
## time_of_dayDusk                5.17e-02   2.44e-02    2.12  0.03369 *
## time_of_dayNight               5.13e-02   2.36e-02    2.17  0.02988 *
## Day_of_Week2                  -7.46e-02   2.85e-02   -2.62  0.00883 **
## Day_of_Week3                  -1.02e-01   2.88e-02   -3.56  0.00038 ***
## Day_of_Week4                  -8.73e-02   2.84e-02   -3.08  0.00209 **
## Day_of_Week5                  -1.01e-01   2.84e-02   -3.56  0.00037 ***
## Day_of_Week6                  -7.24e-02   2.85e-02   -2.54  0.01122 *
## Day_of_Week7                   1.07e-02   3.12e-02    0.34  0.73103
## rush_hourN                     4.40e-02   4.96e-03    8.87  < 2e-16 ***
## rush_hourPM                    1.92e-02   6.32e-03    3.04  0.00236 **
## Age_Band_of_Driver1            4.83e-02   1.45e-01    0.33  0.73813
## Age_Band_of_Driver2            4.90e-02   4.66e-02    1.05  0.29260
## Age_Band_of_Driver3            1.05e-01   2.18e-02    4.84  1.3e-06 ***
## Age_Band_of_Driver4            1.57e-01   6.79e-03   23.06  < 2e-16 ***
## Age_Band_of_Driver5            1.15e-01   6.27e-03   18.39  < 2e-16 ***
## Age_Band_of_Driver6            9.15e-02   5.62e-03   16.27  < 2e-16 ***
## Age_Band_of_Driver7            7.97e-02   5.89e-03   13.52  < 2e-16 ***
## Age_Band_of_Driver8            6.21e-02   6.06e-03   10.24  < 2e-16 ***
## Age_Band_of_Driver9            6.09e-02   6.66e-03    9.15  < 2e-16 ***
## Age_Band_of_Driver10           8.29e-02   7.71e-03   10.75  < 2e-16 ***
## Age_Band_of_Driver11           1.05e-01   8.35e-03   12.58  < 2e-16 ***
## Vehicle_Type1                 -1.29e-01   3.42e-02   -3.78  0.00016 ***
## Vehicle_Type2                 -1.34e-01   3.74e-02   -3.58  0.00034 ***
## Vehicle_Type3                 -1.31e-01   3.43e-02   -3.81  0.00014 ***
## Vehicle_Type4                 -1.05e-01   3.64e-02   -2.88  0.00402 **
## Vehicle_Type5                 -1.14e-01   3.46e-02   -3.28  0.00103 **
## Vehicle_Type8                  4.87e-02   3.45e-02    1.41  0.15794
## Vehicle_Type9                  6.85e-02   3.35e-02    2.05  0.04060 *
## Vehicle_Type10                 2.26e-01   4.36e-02    5.17  2.4e-07 ***
## Vehicle_Type11                 1.55e-01   3.44e-02    4.51  6.4e-06 ***
## Vehicle_Type16                -1.57e-01   1.28e-01   -1.23  0.21894
## Vehicle_Type17                -8.55e-02   4.64e-02   -1.84  0.06542 .
## Vehicle_Type18                 2.18e-03   1.56e-01    0.01  0.98887
## Vehicle_Type19                 2.69e-02   3.39e-02    0.79  0.42756
## Vehicle_Type20                 1.06e-02   3.84e-02    0.28  0.78264
## Vehicle_Type21                -8.70e-03   3.50e-02   -0.25  0.80367
## Vehicle_Type22                -3.97e-02   5.91e-02   -0.67  0.50201
## Vehicle_Type23                -7.74e-02   9.65e-02   -0.80  0.42293
## Vehicle_Type90                 5.25e-02   3.74e-02    1.40  0.16024
```

```
## Vehicle_Type97                     -6.37e-02   4.62e-02   -1.38  0.16776
## Vehicle_Type98                      6.97e-03   3.80e-02    0.18  0.85460
## Age_of_Vehicle                      1.55e-03   2.29e-04    6.77  1.3e-11 ***
## Driver_IMD_Decile1                  4.80e-02   5.67e-03    8.47  < 2e-16 ***
## Driver_IMD_Decile2                  2.50e-02   5.52e-03    4.54  5.7e-06 ***
## Driver_IMD_Decile3                  1.57e-02   5.56e-03    2.82  0.00482 **
## Driver_IMD_Decile4                  1.40e-02   5.57e-03    2.51  0.01196 *
## Driver_IMD_Decile5                  1.21e-02   5.67e-03    2.12  0.03361 *
## Driver_IMD_Decile6                  5.44e-03   5.75e-03    0.95  0.34393
## Driver_IMD_Decile7                 -1.52e-03   5.92e-03   -0.26  0.79761
## Driver_IMD_Decile8                  4.13e-03   6.01e-03    0.69  0.49180
## Driver_IMD_Decile9                  1.63e-03   6.15e-03    0.26  0.79140
## Driver_IMD_Decile10                -5.45e-03   6.47e-03   -0.84  0.39954
## year2017                           -3.55e-03   3.58e-03   -0.99  0.32117
## year2018                           -8.48e-03   3.62e-03   -2.34  0.01925 *
## year2019                           -1.14e-02   3.68e-03   -3.10  0.00196 **
## month2                              1.84e-03   6.55e-03    0.28  0.77887
## month3                              8.86e-03   6.57e-03    1.35  0.17723
## month4                              2.68e-02   6.68e-03    4.00  6.2e-05 ***
## month5                              2.43e-02   6.56e-03    3.71  0.00021 ***
## month6                              2.72e-02   6.56e-03    4.15  3.4e-05 ***
## month7                              3.65e-02   6.52e-03    5.59  2.2e-08 ***
## month8                              4.17e-02   6.57e-03    6.35  2.2e-10 ***
## month9                              1.28e-02   6.48e-03    1.97  0.04899 *
## month10                             1.33e-02   6.40e-03    2.09  0.03704 *
## month11                            -3.80e-03   6.25e-03   -0.61  0.54362
## month12                             2.45e-03   6.39e-03    0.38  0.70103
## First_Road_Class:Second_Road_Class  1.96e-03   2.88e-04    6.82  8.8e-12 ***
## time_of_dayDay:Day_of_Week2         1.90e-02   2.92e-02    0.65  0.51418
## time_of_dayDusk:Day_of_Week2        7.59e-03   3.18e-02    0.24  0.81132
## time_of_dayNight:Day_of_Week2       2.64e-02   3.13e-02    0.85  0.39780
## time_of_dayDay:Day_of_Week3         3.38e-02   2.94e-02    1.15  0.25040
## time_of_dayDusk:Day_of_Week3        1.15e-02   3.19e-02    0.36  0.71990
## time_of_dayNight:Day_of_Week3       3.36e-02   3.14e-02    1.07  0.28486
## time_of_dayDay:Day_of_Week4         1.49e-02   2.90e-02    0.51  0.60708
## time_of_dayDusk:Day_of_Week4        7.94e-03   3.16e-02    0.25  0.80155
## time_of_dayNight:Day_of_Week4       3.46e-02   3.10e-02    1.12  0.26484
## time_of_dayDay:Day_of_Week5         3.45e-02   2.90e-02    1.19  0.23421
## time_of_dayDusk:Day_of_Week5        3.85e-02   3.15e-02    1.22  0.22221
## time_of_dayNight:Day_of_Week5       2.82e-02   3.10e-02    0.91  0.36337
## time_of_dayDay:Day_of_Week6         1.15e-02   2.92e-02    0.39  0.69315
## time_of_dayDusk:Day_of_Week6        1.33e-02   3.16e-02    0.42  0.67368
## time_of_dayNight:Day_of_Week6       4.06e-02   3.08e-02    1.32  0.18769
## time_of_dayDay:Day_of_Week7        -2.72e-02   3.19e-02   -0.85  0.39302
## time_of_dayDusk:Day_of_Week7       -1.80e-02   3.41e-02   -0.53  0.59899
## time_of_dayNight:Day_of_Week7      -3.17e-03   3.32e-02   -0.10  0.92392
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Negative Binomial(65462) family taken to be 1)
##
##     Null deviance: 138515  on 455763  degrees of freedom
## Residual deviance: 119671  on 455656  degrees of freedom
##   (332 observations deleted due to missingness)
```

```
## AIC: 1105753
##
## Number of Fisher Scoring iterations: 1
##
##
##               Theta:  65462
##           Std. Err.:  15342
## Warning while fitting theta: iteration limit reached
##
##  2 x log-likelihood:  -1105535
```

5.10 The negative binomial model gives an RMSE of 1.271 about the same as the poisson model. We can test this degree of similarity by using an anova (analysis of variance) test.

```
# ANOVA poisson vs neg bin
anova(model.poisson, model.negbin)
```

Resid. Df	Resid. Dev	Df	Deviance
455680	119707	NA	NA
455656	119671	24	35.9

5.11 Comparing the model summaries we see that the two models are not dissimilar, and score similar RMSEs, this is confirmed by the ANOVA test which shows us that the differences between these models are not significant.

Response transformation and zero inflation

5.12 Our response data is heavily skewed to the value 1, the number of casualties is always one or more. This is quite inefficient, all of those 1s are uninformative: if there has been an accident there will have been a casualty, so we can look at whether we can improve the model by transforming the response variable to one less than the number of casualties.

```
# Add an adjusted number of casualties to only count any after the first
sampled_training$casualties <- sampled_training$Number_of_Casualties - 1

# fit a negative binomial glm to predict the number of casualties after the first
model.negbin_adj <- glm.nb(casualties ~ Police_Force +
                     Other_Vehicles +
                     Speed_limit +
                     First_Road_Class +
                     Second_Road_Class +
                     First_Road_Class:Second_Road_Class +
                     Road_Surface_Conditions +
                     Carriageway_Hazards +
                     time_of_day * Day_of_Week +
                     rush_hour +
                     Age_Band_of_Driver +
                     Vehicle_Type +
                     Age_of_Vehicle +
                     Driver_IMD_Decile +
                     year +
                     month,

                   data=sampled_training)

# Apply the model to the validation data to test it
pred.negbin_adj <- data.frame(casualties =
```

```
                            predict(model.negbin_adj, newdata=sampled_validation) + 1)

# Flag any records without a prediction and exclude from RMSE
miss_idx <- is.na(pred.negbin_adj$casualties)

# Calculate the RMSE for the model
rmse_negbin_adj <- RMSE(sampled_validation[!miss_idx,]$Number_of_Casualties, pred.negbin_adj[!miss_idx,]

# Store the RMSE value in our results tibble
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Neg bin model, response adj",
                                 RMSE = rmse_negbin_adj))
```

5.13 This is an improvement, with a higher RMSE score, 1.988 , but a lower AIC (6.019×10^5 vs 1.106×10^6 for the negative binomial model).

5.14 Looking at the response values we now have many more zeroes in our data than we would expect; this motivates consideration of a zero-inflated model. We can use the *pscl* library's *zeroinfl()* function to fit a zero-inflated negative binomial model. This should create a two step model, firstly classifying accidents as having either a zero or positive number of casualties after the first, and secondly a distribution-based model ofthe positive values.

```
# fit a zero inflated negative binomial model to predict the number of casualties after the first
model.negbin_zin <- zeroinfl(casualties ~ Police_Force +
                  Other_Vehicles +
                  Speed_limit +
                  First_Road_Class +
                  Second_Road_Class +
                  First_Road_Class:Second_Road_Class +
                  Road_Surface_Conditions +
                  Carriageway_Hazards +
                  time_of_day * Day_of_Week +
                  rush_hour +
                  Age_Band_of_Driver +
                  Vehicle_Type +
                  Age_of_Vehicle +
                  Driver_IMD_Decile +
                  year +
                  month,

                data=sampled_training,
                dist = "negbin", EM=FALSE)

# Apply the model to the validation data to test it
pred.negbin_zin <- data.frame(casualties =
                            predict(model.negbin_zin, newdata=sampled_validation) + 1)

# Flag any records without a prediction and exclude from RMSE
miss_idx <- is.na(pred.negbin_zin$casualties)

# Calculate the RMSE for the model
rmse_negbin_zin <- RMSE(sampled_validation[!miss_idx,]$Number_of_Casualties, pred.negbin_zin[!miss_idx,]

# Store the RMSE value in our results tibble
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Neg bin model, zero-inflated",
```

```
                                        RMSE = rmse_negbin_zin))
```

5.15 This gives us a RMSE score of 1.121 and we can test whether this zero inflated model is statistically better than the adjusted response model using the Vuong closeness test.

```
# Test for Vuong closeness
vuong(model.negbin_adj, model.negbin_zin)
```

```
## Vuong Non-Nested Hypothesis Test-Statistic:
## (test-statistic is asymptotically distributed N(0,1) under the
##  null that the models are indistinguishible)
## -------------------------------------------------------------
##                Vuong z-statistic             H_A p-value
## Raw                    -41.9 model2 > model1  <2e-16
## AIC-corrected          -40.8 model2 > model1  <2e-16
## BIC-corrected          -34.9 model2 > model1  <2e-16
```

The Vuong z-statstic suggests that the zero-inflated model is better.

Review of count models

5.16 The RMSE scores we've achieved with the count models are poor. The zero-inflated negative binomial model is less good than the linear model we started with. This suggests that, contrary to our intuition, a count model may not be appropriate. This is largely driven by the preponderance of single casualty accidents, predicting that there will be one casualty outperforms these models and achieves an RMSE score of 0.83.

Decision trees and random forests

5.17 The performance of the models we have looked at so far motivates a fresh approach. We want to find a way of reflecting the variablility in the number of casualties with the variability of other data items. One way of doing this is with decision trees, however we can go a step further and develop a random forest model.

5.18 Fitting a random forest model can be very resource intensive, so we will try using a subset of our training data, taking a tenth of the data should give us enough to start with.

```
# Set the seed to get reproducible randomness
set.seed(1, sample.kind="Rounding")

# Create sampled partition index to get a subset
subsample_index <- createDataPartition(y = sampled_training$Number_of_Casualties,
                                       times = 1, p = 0.1, list = FALSE)
# Create the subset as those obs in the index
subsample_training <- sampled_training[subsample_index,]
```

5.19 We can now fit a random forest model to this subset.

```
# Set the seed to get reproducible randomness
set.seed(1, sample.kind="Rounding")
# fit a random forest model to predict the number of casualties
fit <- randomForest(Number_of_Casualties~Police_Force +
                      Other_Vehicles +
                      Speed_limit +
                      First_Road_Class +
                      Second_Road_Class +
                      Weather_Conditions +
                      Light_Conditions +
                      Road_Surface_Conditions +
```

```
                      Carriageway_Hazards +
                      Urban_or_Rural_Area +
                      time_of_day +
                      Day_of_Week +
                      rush_hour +
                      Age_Band_of_Driver +
                      Sex_of_Driver +
                      Vehicle_Type +
                      Age_of_Vehicle +
                      Driver_IMD_Decile +
                      year +
                      month,
                    data = subsample_training,
                    na.action = na.omit)

# Apply the model to the validation data to test it
pred.rf <- data.frame(casualties = predict(fit, sampled_validation))

# Flag any records without a prediction and exclude from RMSE
miss_idx <- is.na(pred.rf$casualties)

# Calculate the RMSE for the model
rmse_rf <- RMSE(sampled_validation[!miss_idx,]$Number_of_Casualties, pred.rf[!miss_idx,])

# Store the RMSE value in our results tibble
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Random Forest",
                                 RMSE = rmse_rf))
```

5.20 This random forest model gives an RMSE score of 0.708 , significantly better than the count-based (poisson and negative binomial) models we saw earlier.

Model tuning

5.21 We can get some diagnostic statstics from this model by using the varImp function.

```
# List variable importance scores from the random forest model
varImp(fit)
```

	Overall
Police_Force	2242
Other_Vehicles	1646
Speed_limit	1177
First_Road_Class	1028
Second_Road_Class	772
Weather_Conditions	586
Light_Conditions	515
Road_Surface_Conditions	404
Carriageway_Hazards	150
Urban_or_Rural_Area	324
time_of_day	626
Day_of_Week	2073
rush_hour	341
Age_Band_of_Driver	2456
Sex_of_Driver	571
Vehicle_Type	1806
Age_of_Vehicle	1988
Driver_IMD_Decile	2954
year	1315
month	3232

5.22 This suggests that we can simplify the model without losing a huge amount of predictive power. With a simpler model we can process a larger volume of data and improve the model as we will train it on a more diverse range of examples. We can also tune the model, investigating the numbers of trees to use, the size of nodes and the number of variables available for splitting at each tree node. With these refinements we should be able to improve our RMSE score and do better than the linear model.

Simple ensemble

5.23 We have seen that the variables that were important in our linear model were different to those highlighted in our random forest model. This would allow us to use a simple ensemble, a combination of these two models.

```
# Apply the model to the validation data to test it
pred.rf_lin <- ifelse(is.na(pred.rf$casualties + pred.linear$casualties),
                  mean_casualties,
                  (pred.rf$casualties + pred.linear$casualties) / 2)

# Calculate the RMSE for the model
rmse_rf_lin <- RMSE(sampled_validation$Number_of_Casualties, pred.rf_lin)

# Store the RMSE value in our results tibble
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Random Forest and linear",
                                 RMSE = rmse_rf_lin))
```

5.24 This ensemble model results in an RMSE of 0.704 an improvement on both the random forest and linear models we have seen so far.

6 Results

6.1 We have seen that we can improve the accuracy reported by our modelling by developing our choice of variables and distribution. The rmse scores for the models we have developed are summarized in the table below:

```
as.data.frame(rmse_results)
```

method	RMSE
Just the mean	0.746
Linear model	0.707
Poisson model	1.271
Negative binomial model	1.271
Neg bin model, response adj	1.988
Neg bin model, zero-inflated	1.121
Random Forest	0.708
Random Forest and linear	0.704

6.2 The achieved model performance is a little disappointing, and suggests that, with hindsight, the lack of variation in the response variable means it is less than ideal for modelling. However there is clearly scope to improve the accuracy of the random forest model and to therby improve the ensemble.

6.3 The random forest model is highly resource intensive and limited the amount of data that could be processed using the hardware available. This suggests that a better model could be achieved with more data and with tuning of the model parameters.

7 Conclusions

Summary

7.1 We have looked in some depth at the accident data and this exploratory data analysis has allowed us to see some ways to extend and improve the casualty model. The final model we present here is an ensemble of a linear and a random forest model. The ensemble model outperforms both of its constituent models, suggesting that these very different models are complementary.

7.2 This also work suggests that the variables identified using a linear model may not be the only ones relevant in predicting the number of casualties in a traffic accident, and that a tree-based model may be insightful, for example highlighting the significance of car maintenance through the proxies of IMD decile (wealth) and car age.

Limitations and further work

Conditionality of the data 7.3 The models discussed here are all conditional models, derived from data collected once accidents have happened. These models don't help predict when or where an accident is likely to happen. Once an accident has happened, the first human instinct is to react to help and support the casualties and counting them, identifying them, is an intrinsic first step on arriving and assessing the scene.

7.4 On its own this model is therefore of limited use, although the random forest gives us insight into parameters we might not have looked into from the linear model. The model developed here could be useful as part of a suite of models designed to predict accidents. The first step in this scheme might be a parallel model to predict accident severity, this would be a natural companion and could be developed in the same conditional manner.

An integrated suite of models 7.5 A separate model, or module, in this system would then predict where and when accidents were likely to happen. This would make the predictions of this model more useful because an accident propensity model would take it out of the conditional world of "once an accident has happened". This model could rely on other external data: driver numbers and congestion, weather forecasts, roadworks and other factors could be fed in to it. Such a model could allow emergency services to prioritize between accidents, and plan crew rosters based on the likelihood of an accident in a given place at a given time for some weather condtions.

7.6 In the previous section we alluded to the use of a wider range of input data sources, there is, however, data associated with the accident records that we have not looked at, such as the casualty records and details of other vehicles. For example by only looking at the first vehicle, say a motorbike, we might miss the exposure of a larger number of people, say on a coach, to danger. We could also engineer model features such as the number of vehicles involved as this ought to contribute to the number of accidents.

Optimising the random forest 7.7 The random forest model is not in its final form, with lots of variables and limited data, due to limited processing power. The results of this modelling could, however, be used to restrict the number of variables and increase the number of rows of data within a processing envelope. With more computing power this parsimony might not be necessary.

7.8 The random forest model should be tuned and optimised using a multi-fold design to ensure robust optimisation. As we identified in 5.21 there is ample scope for this, with parameters such as the numbers of trees, and the size and composition of nodes all configurable.

Weighting the ensemble 7.9 The ensemble chosen here is a simple average of the two models, we could optimize the weighting of these models in the ensemble. As a first step we might do this by using a ten-fold design again, and looking at how the accuracy changes if our weighting is 10% linear prediction, 90% random forest, through to 90% linear, 10% random forest, testing on the reserved fold each time.

Literature review 7.10 We have relied on the data contained here to inform our modelling decisions, however we might also review the associated literature and talk to experts in the field. A quick scan of some of the academic output on this subject suggests that some of the drivers of accident severity are dangerous driving, whether under the influence of stimulants, such as alcohol, or breaking speed limits. These are variables that we do not have access to, at the current time, but which could lead to a much more predictive model.
