CI6320: Data Analysis 2
K1651915

Domin Thomas

2/7/2020

Contents

1 Introduction

The Road Accidents Data set contains over 250,000 observations from UK road accidents. The data in its raw form isn't clean and to practice proper data hygiene, to avoid corrupted results, the data needs to be cleaned before creating visualisations. For this report, R will be used instead of Tableau as it provides more control over the data with its powerful libraries.

R is a free and open source programming language for statistical computing and graphics, supported by the R Foundation. It is widely used among statisticians and data miners for developing statistical software and for data analysis. It has a vast array of libraries with powerful functions. R also has its own markdown language, R Markdown (.rmd), which allows for creating documents explaining the process of data mining, cleaning, and creating visualisations with the results from running the R scripts. It supports the LaTeX format, which is used in the scientific community to create theses and publishable scientific papers.

This entire document is composed using the R Markdown language. Hence, all code used to clean, prepare and filter the data, along with the code for creating the visualisations are included, where necesarry, in this report. These code blocks can be recognised, as they are syntax highlighted. The R Markdown for this document can be accessed on GitHub[1]. [1]:https://github.com/dominthomas/private

Any text output from running an R command will have a double hash prefix, for e.g. in R, the output from a print command to print 'Hello World!' will be shown as follows:

```
print("Hello World!")
```

```
## [1] "Hello World!"
```

The syntax highlighted statement is the R code, followed by its output on the next line with the double hash (##) prefix. Please note, a single hash prefix is an inline comment.

For this report, the **dplyr** package will be used for data manipulation and the **ggplot2** package will be used to create the data visualisations.

The **dplyr** package is a very powerful package that is often used for data manipulation, it can be used to connect to an external database which then allows the user to use the **dplyr** syntax for data manipulation instead of SQL. This is one of the most used tools by Data Scientists for data exploration as it allows for platform independent data manipulation. In the case of this coursework, **dplyr** will be used on a data frame, which is created from the RoadData.csv file.

Let's begin by reading the data in from the csv file as a data frame:

```
data <- read.csv(
  "/home/dthomas/Documents/KU/Advanced_Data_Modelling/private/RoadData.csv")
```

To get the accurate size of the data, print the number of observations (rows) and variables (columns):

```
nrow(data)
```

```
## [1] 285331
```

```
ncol(data)
```

```
## [1] 70
```

So there are 285,331 observations and 70 variables. This is an impressive amount of data and because of the large amount of variables, it's more efficient to clean the data on an as needed basis, i.e. just before feeding the data to each visualisation.

Please note that all visualisations are embedded into this PDF Document as Scalable Vector Graphics (SVG), so if a visualisation looks too small, one can zoom in without loosing image resolution.

2 Data Visualisations

2.1 Number of Accidents vs Sex of Driver vs Age of Driver

2.1.1 Aim

Motor-vehicle insurance companies often categorise men at a higher risk and age is also a factor that is taken into account.

Are younger males more likely to be involved in a motor-vehicle accident compared to younger females?

To answer this question, we need all male and female driver ages, which can be obtained by filtering the data through **dplyr**.

2.1.2 Data Preparation & Visualisation

- Remove -1 values from ‘age_of_driver’ column
- Retrieve data from ‘age_of_driver’ column where values equal 1 and assign it to a data frame 1
- Retrieve data from ‘age_of_driver’ column where values equal 2 and assign it to a data frame 2
- Create a data frame with two columns, age and sex, the latter being populated with ‘Male’ or ‘Female’ depending on the sex of the variable
- Combine the two data frames together to form a single data frame with male and female ages
- Create two histograms with repsect to age, one for male and and another for female, and overlay the plots

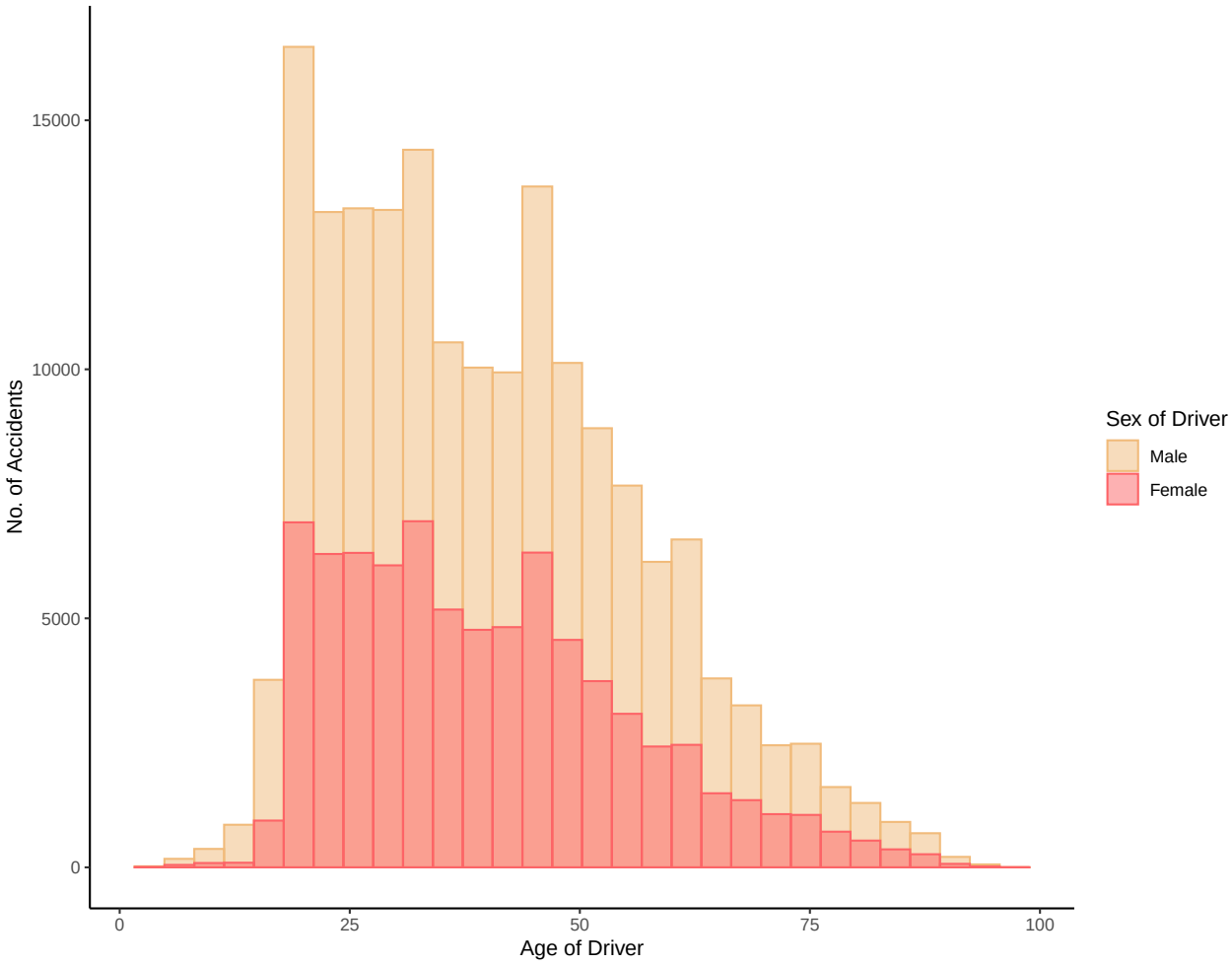


Figure 1: Driver Sex vs Driver Age vs No. of Accidents

As you can see from Figure 2, there were more male driver accidents than female driver accidents, where younger drivers are more likely to be involved in an accident irrespective of sex. But, it is important to know that the conclusion from the visualisation doesn't take into account how many male and female drivers there are in the UK. So if there are more male drivers than female drivers, statistically speaking, a higher proportion of accidents can be linked to male drivers.

2.2 Geo-spatial plot of accidents

2.2.1 Aim

Are accidents more likely to occur in densely populated areas such as city centres? To answer this question, a Geo-spatial plot is required, with a density heat-map using the longitude and latitude data.

2.2.2 Data Preparation & Visualisation

For this visualisation, we only need the longitude and latitude data. Now, longitude and latitude data is likely to have large amounts of variation, so, it is futile to try and obtain unique values and look for outliers. However, **ggplot2** has built in pattern recognition for this case, which will automatically omit evident outliers. This means we can proceed straight to creating the visualisation.

- Obtain the UK Map layout, and fill it with colour
- Add a layer plotting all accidents on the map as points using ‘longitude’ & ‘latitude’ data
- Adjust the density ‘alpha’ attribute so areas with a higher amount of accidents are brighter
- Add a heatmap density layer to better visualise hotspot density
- Obtain the top 9 populous cities in the UK and add them as labels to the Geo-plot, to see if there is a correlation.

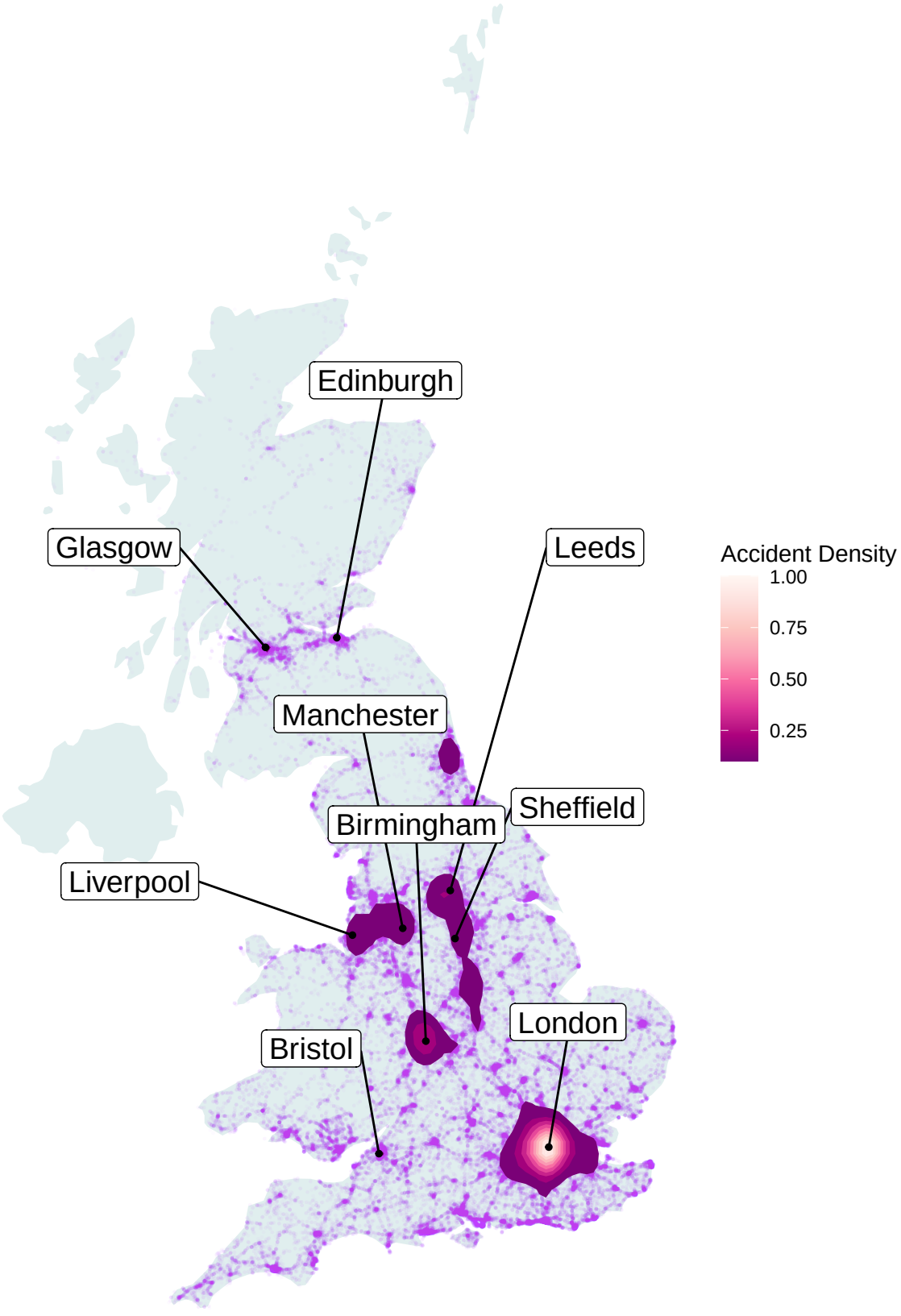


Figure 2: Geospatial plot of accidents

Looking at the Geo-spatial plot, we can easily see that major UK cities have more accidents than the UK countryside, this can be due to the areas being more densely populated, which means more vehicles and people, which in-turn increases the risk of an accident.

2.3 Districts with the highest number of accidents

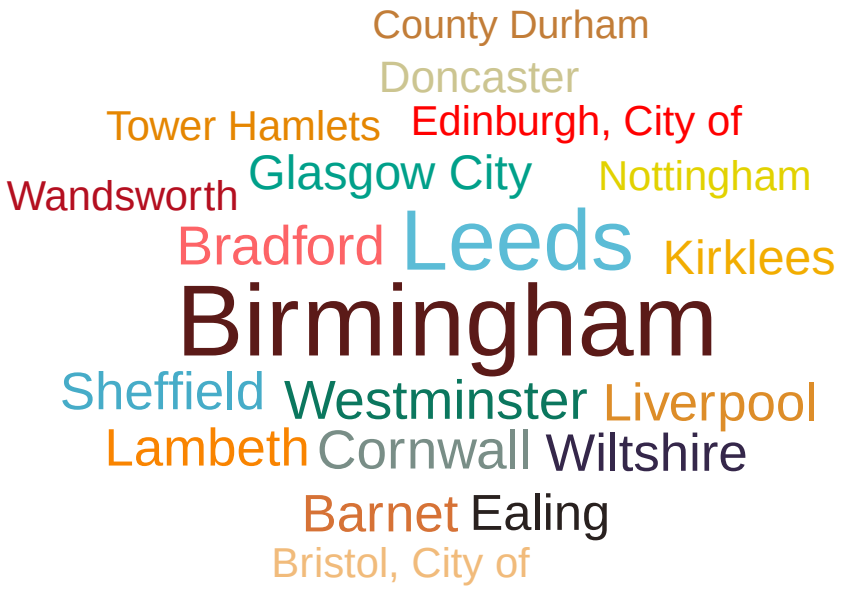
2.3.1 Aim

What are the top 20 accident prone districts?

Are there particular districts where accidents are more likely to occur? To answer this question, the local authority district IDs needs to be obtained. But let's also make this plot a bit more useful, where the proportion of accident severity is also shown. For this visualisation, it should be a combination of a Word Cloud and a bar plot.

2.3.2 Data Preparation & Visualisation

- Obtain district IDs from the 'local_authority_.district.' column
- Create a data frame with the frequency of each unique 'local_authority_.district.' variable
- Sort the data frame by descending order, so the most frequent local districts are at the top
- Retrieve the top 20 districts from the data frame
- Substitute the values for each district ID with it's corresponding value from the data dictionary
- Create the word cloud
- Create the stacked bar plot



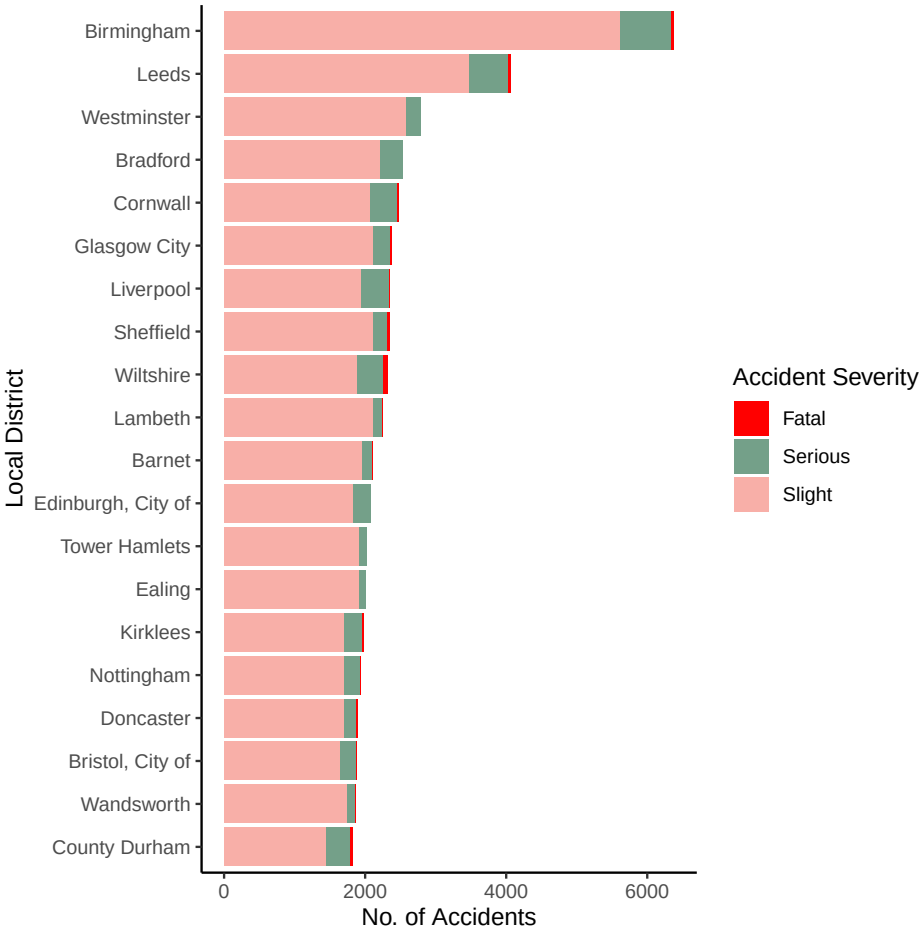


Figure 3: Most accident prone districts with accident severity

As you can see Birmingham had the highest amount of accidents compared to any other local district. But why isn't London at the top? Well London is a Metropolis, made up of several local districts.

2.4 Weather and Light conditions

2.4.1 Aim

In what weather and light conditions are accidents more likely to occur? Are accidents more likely to occur during night than day? How does weather impact the number of accidents?

2.4.2 Data Preparation & Visualisation

For this visualisation, two density plots are best suited, one for weather vs No. of Accidents, and another for light conditions vs No. of Accidents.

A density plot is quite simple, where peaks correlate to the number of values, or how 'dense' the data is for that particular value.

- Retrieve 'weather_conditions'
  - Filter out any values outside 1 - 7
  - Substitute values from the Data Dictionary
- Retrieve 'light_conditions'
  - Filter out any values outside 1 - 6
  - Substitute values from the Data Dictionary
- Create the weather conditions vs accident density plot
- Create the light conditions vs accident density plot

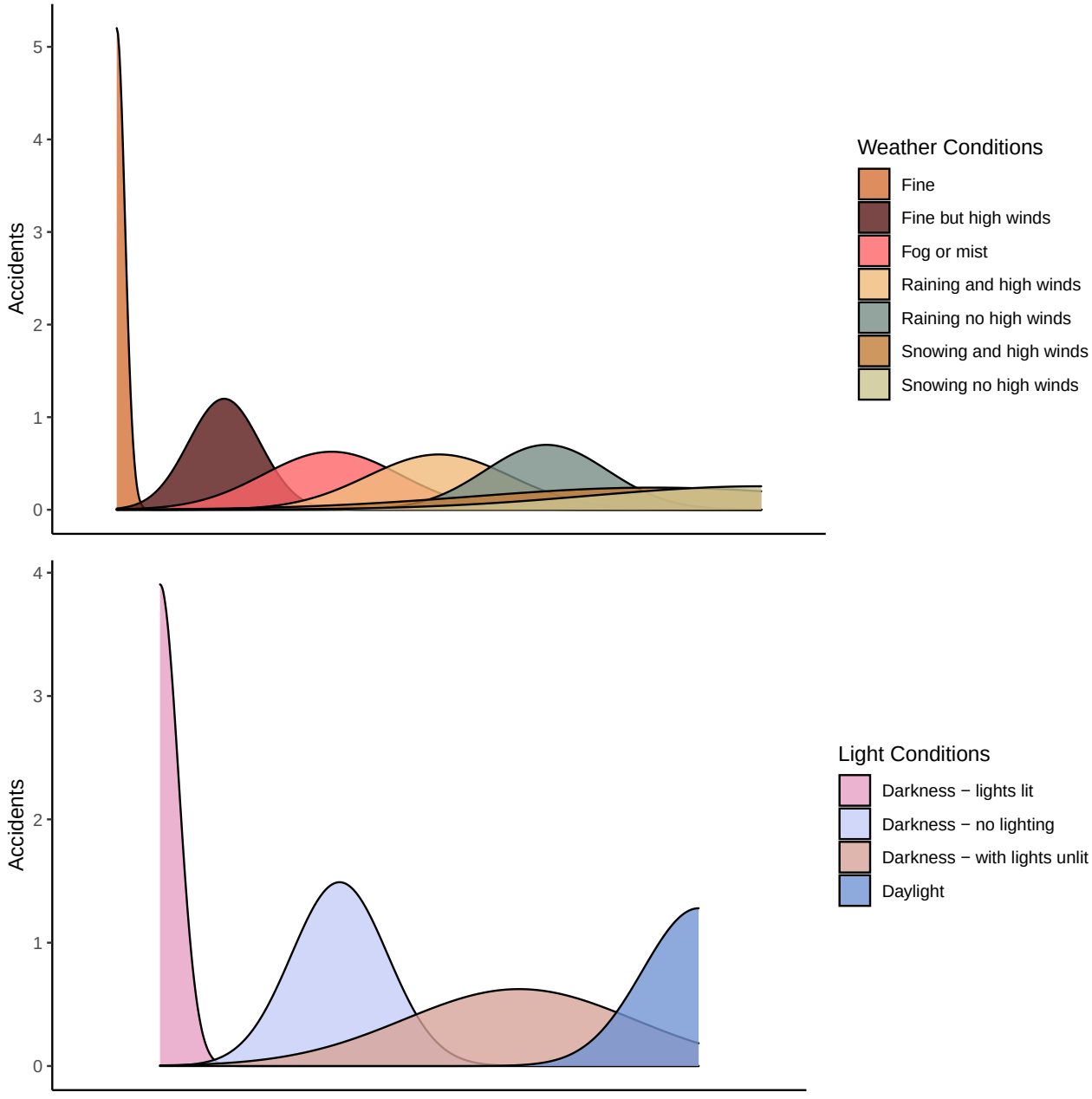


Figure 4: Effect of Weather and Light Conditions

From the above visualisation, we can see that most accidents occur at night with street lights on, in fine weather conditions.

2.5 Weekday and Time vs No. of Accidents

2.5.1 Aim

Is there a correlation between the number of accidents with weekday and time? This will tell us which weekday and time pose a higher risk to road vehicles. It will also tell us if there is a particular weekday with more accidents than any other.

2.5.2 Data Preparation & Visualisation

For the weekday plot:

- Substitute weekday ID in the ‘day_of_week’ column with corresponding values from the Data Dictionary
- Create the weekday bar plot by taking the frequency of each frequecy into account.
- Arrange the bars in descending order
- Retrieve the time data from the ‘time’ column
- Convert time data to POSIX time
- Create the time density plot

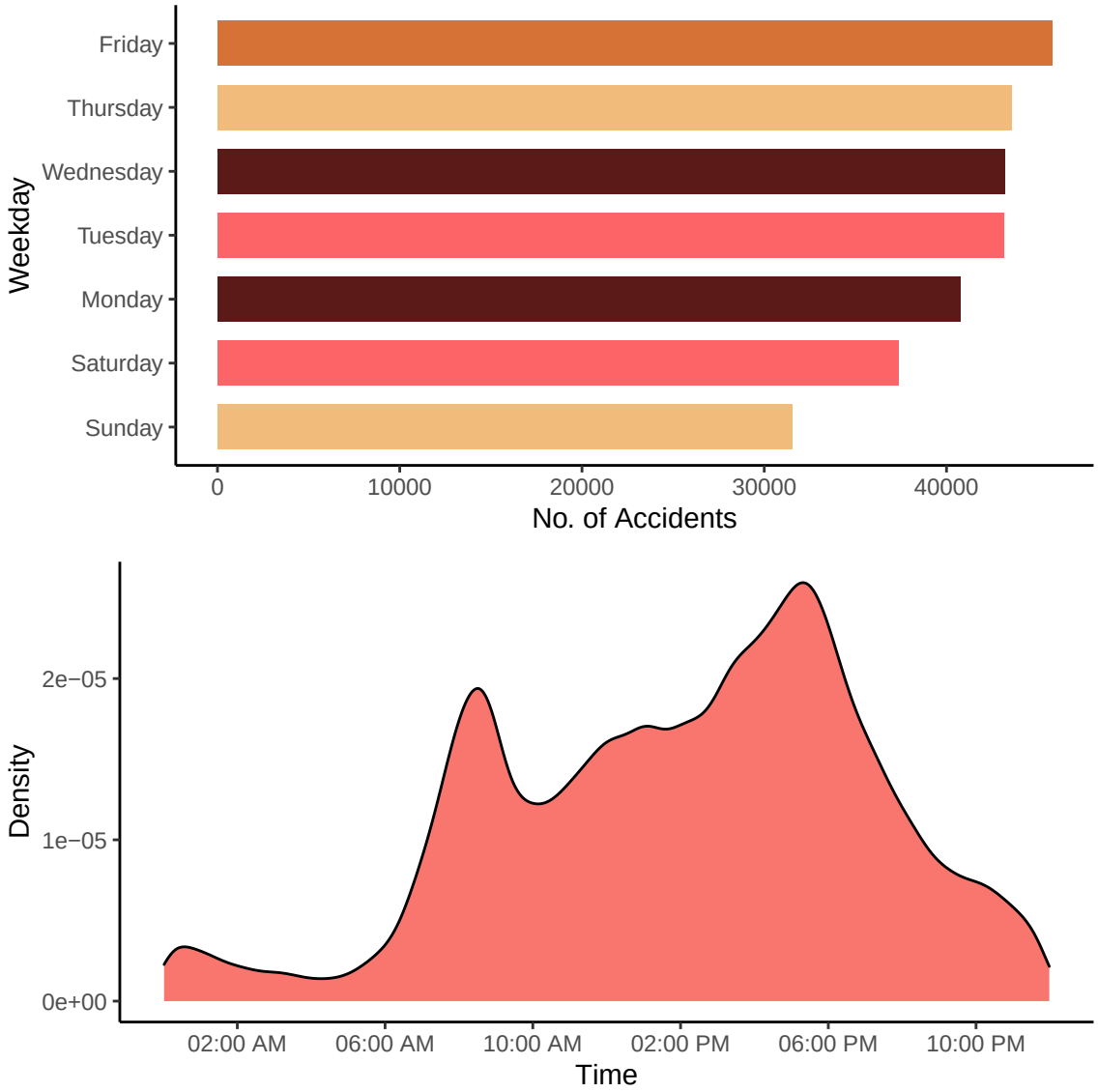


Figure 5: Weekdays vs No. of Accidents and Time vs No. of Accidents

From the above visualisation, we can conclude that Friday had the most number of accidents followed by Thursday. Time also had an effect, where the highest peak was around the evening rush, 5 - 6pm, followed by the second highest peak during the morning rush 8 - 10am.

2.6 Seasonality with Journey Purpose and the deadliest days of the year

2.6.1 Aim

Is there a seasonality to the frequency of accidents and what journey purpose accounted for the majority of accidents? Also what was the deadliest day of the year, i.e. which day had the most accidents?

2.6.2 Data Preparation & Visualisation

For this plot a line graph would be best along with a bar plot for the deadliest days.

- Retrieve data from the ‘journey_purpose_of_driver’ & ‘date’ columns where journey purpose values are in the range 1 - 5.
- Convert the date values to Month name abbreviations.
- Create a data frame with the frequency of the month
- Substitute journey purpose ID with its corresponding value in the Data Dictionary
- Create the seasonality plot with months on the X axis and No. of Accidents on the Y axis.
- For the deadliest day plot, first retrieve the dates from the ‘date’ column
- Convert the date values to date objects
- Create a data frame with the frequency of each date.
- Sort the data frame in descending order, so the dates with the highest frequency is at the top.
- Retrieve the top 6 dates.
- Create the bar plot for the top 6 deadliest dates.

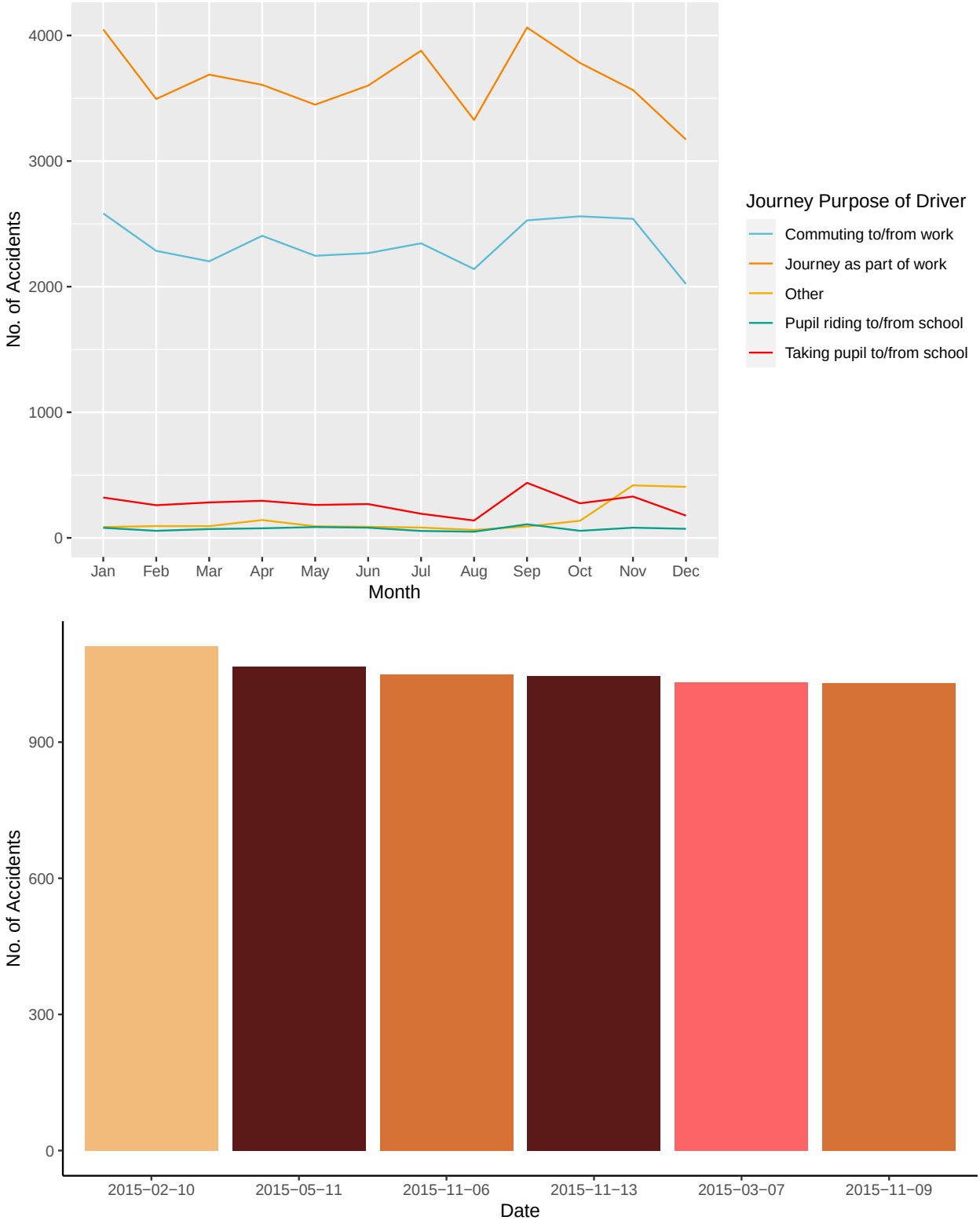


Figure 6: Seasonality and Journey Purpose of drivers with the top 6 deadliest days

Figure 6 shows ‘Journey as part of work’ had the highest number of accidents with clear seasonal peaks around September and January, when schools open and right after the New Year holiday. The second bar plot shows that the deadliest day of that year was the 10th of February, followed by the 11th of May.

2.7 Road Types, Road Class, Speed Limits and Roads with the highest number of fatalities

2.7.1 Aim

What type of roads are accidents more likely to occur? Does speed limit have an effect on the number of accidents? And on which road in the UK did the most amount of fatal accidents occur?

To answer these questions, a unique dashboard is in order. Where two doughnut charts will present the data for speed limits and its proportion of accidents and road classes with its proportion of accidents.

This can be followed by two vertical bar graphs, one displaying the top 5 deadliest roads in the UK, and another displaying the top 5 road types with the highest amount of accidents in the UK.

2.7.2 Data Preparation & Visualisation

For the first doughnut chart, speed limits and its proportion of accident:

- Retireve speed limits
- Filter out values 0 and 10
- Create a data frame with the frequency of each speed limit
- Calculate the fractions for each speed limit with respect to the sum of frequencies
- Calculate the cumulative sum for each frequency and assign it to each speed limit
- Calculate each cumulative sum’s neighbouring value, this is so ggplot knows where a value starts and ends.
- Create the doughnut plot
- Assign the percentage value for each speed limit to the reference legend labels

For the second doughnut chart, road class and its proportion of accidents:

- Retrieve road class values
- Filter out all values that equals 6
- Substitute each road class ID with its corresponding value from the Data Dictionary
- Create a data frame with each road class and its frequency
- Calculate the fractions for each road class with respect to the sum of frequencies
- Calculate the cumulative sum for each frequency and assign it to each road class

- Calculate each cumulative sum’s neighbouring value, this is so ggplot knows where a value starts and ends.
- Create the doughnut plot

For the third plot, a bar graph, representing the top 5 roads in the UK with the most fatalities:

- Paste road class and road numbers together to obtain road names
- Create a data frame with the frequencies of each road name
- Arrange the data frame in descending order, so the road names with the most fatalities is at the top
- Retrieve the top 5 road names with the most fatalities
- Create the bar plot

For the fourth plot, a bar graph, road types with the highest number of accidents:

- Retrieve the road type data
- Filter out values -1 and 9
- Substitute road type IDs with their corresponing values in the Data Dictionary
- Create the road bar plot

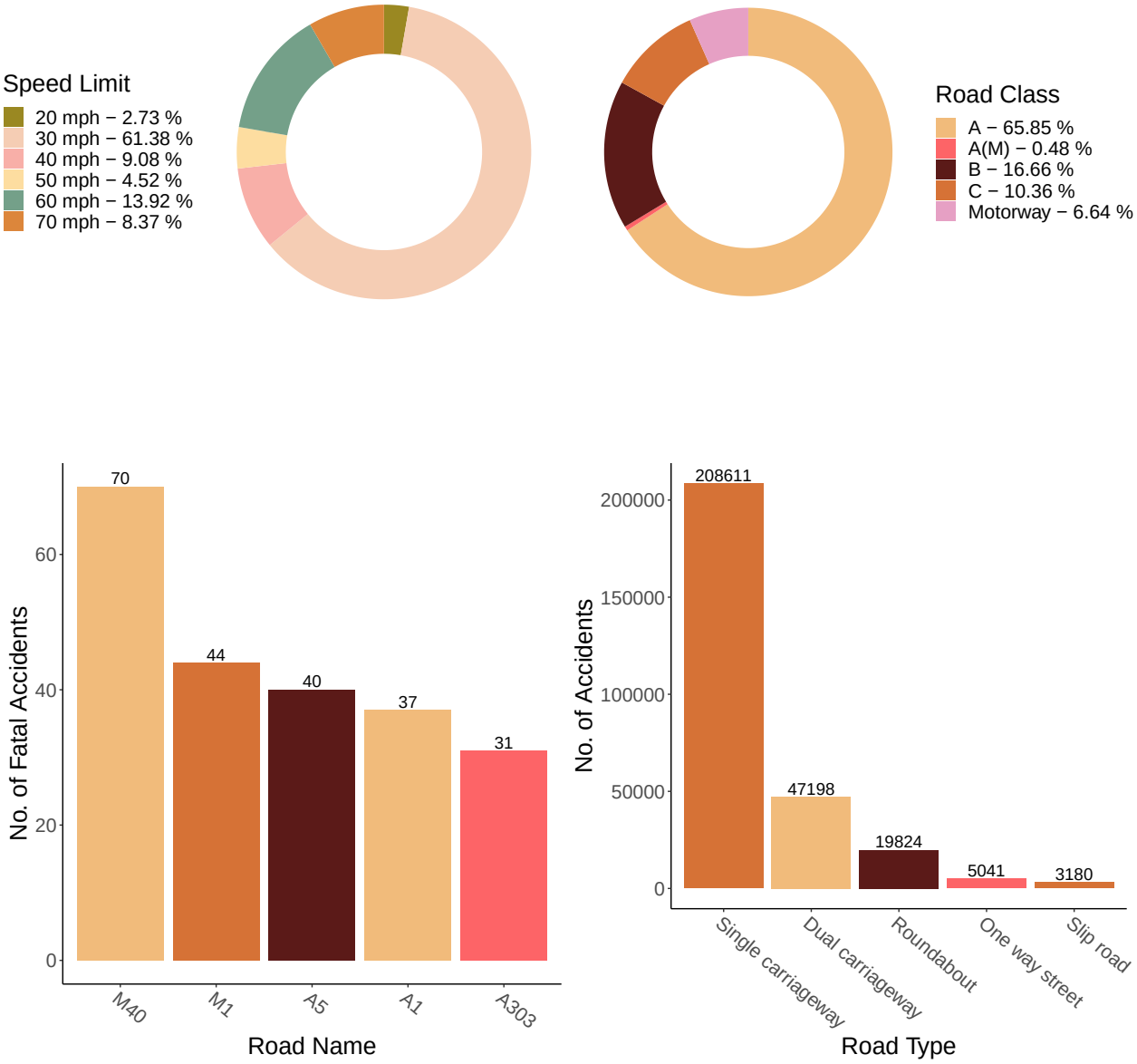


Figure 7: Proportion of accidents to Speed Limit, Road Class and Road Type with the top 5 fatal roads

From Figure 7, we can conclude that accidents are more likely to occur in 30mph zones. A roads in the UK account for the highest number of accidents and single carriageways have the highest number of accidents.

Finally, the road with the most fatalities is the M40.

2.8 Vehicle Manouvre

2.8.1 Aim

What vehicle manoeuvre caused the most amount of accidents?

A good way to represent this information is through a tree map.

2.8.2 Data Preparation & Visualisation

- Retrieve the vehicle manoeuvre data
- Filter out -1 values
- Create a data frame with the frequency of each vehicle manoeuvre
- Arrange the data in descending order
- Retrieve the top 10 manoeuvres
- Substitute each vehicle manoeuvre ID with its corresponding value in the Data Dictionary
- Create the tree map

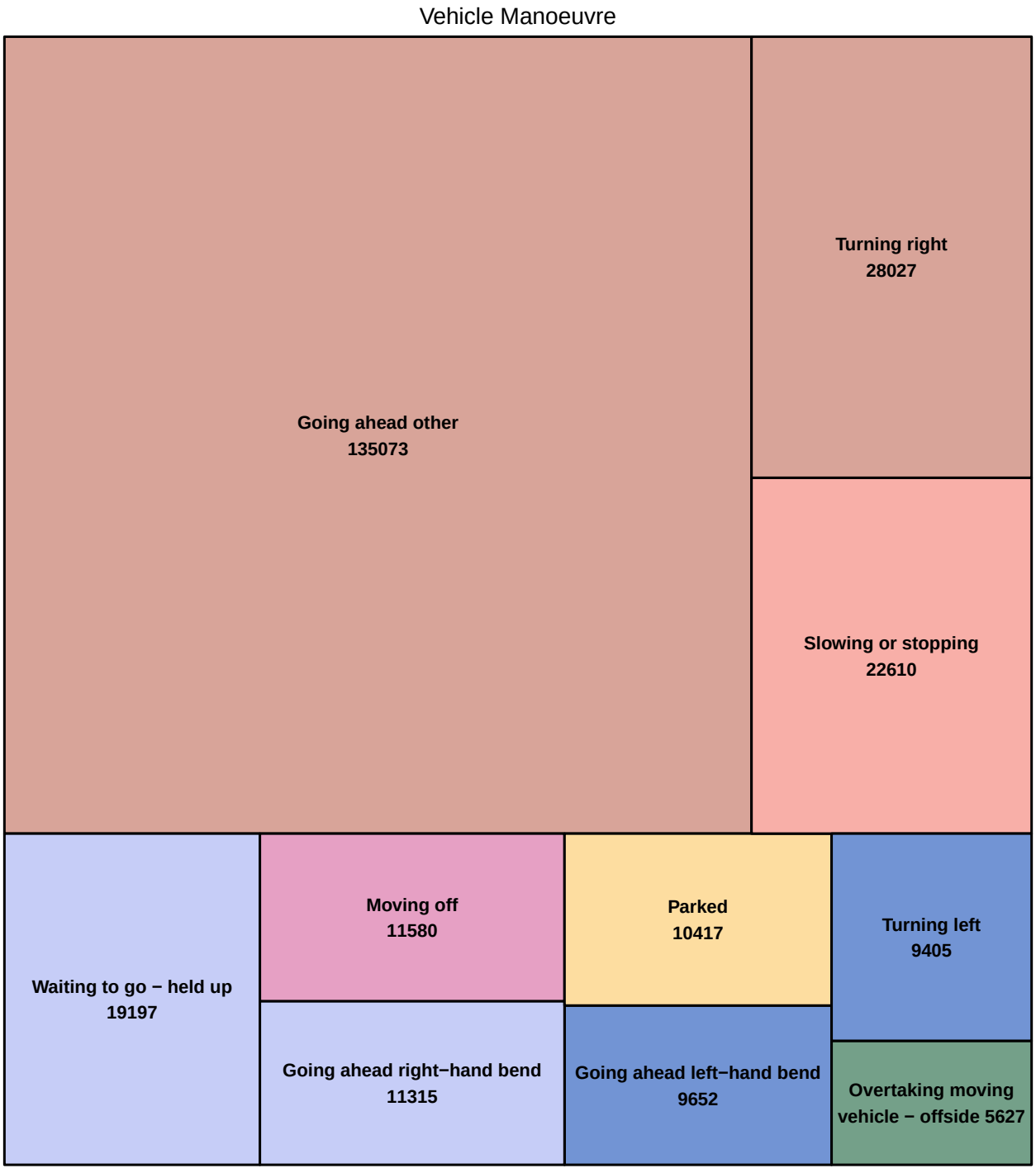


Figure 8: Vehicle Manoeuvres vs No. of Accidents

From Figure 8, we can clearly see that ‘Going ahead other’ caused the highest number of accidents. Interestingly, turning right seems to be a hazardous manoeuvre.

2.9 Hierarchical clustering of Vehicle Types with Propulsion Codes and Accident Severity

2.9.1 Aim

Can we see unique clusters of vehicle types? How similar are these vehicle types with each other? This information is very useful to build a hierarchy of clusters.

2.9.2 Data Preparation & Visualisation

For this visualisation, a dendrogram is best suited.

A dendrogram is a tree-structured graph used in heat maps to visualise the result of a hierarchical clustering calculation.

The dendrogram lists the objects which are clustered along the x-axis, and the distance at which the cluster was formed along the y-axis.

Distances along the x-axis are not particularly meaningful in a dendrogram; the observations are often equally spaced to make the dendrogram easier to read.

The result of a clustering is presented either as the distance or the similarity between the clustered rows or columns depending on the selected distance measure.

The steps for this visualisation is better explained with the R code.

- First, let’s retrieve the vehicle type propulsion code & accident severity data and filter out -1 values from both vehicle type and propulsion code.

```
# Retrieve vehicle type and propulsion code,
# filter out -1 values from both columns.
d <- data %>%
  select(vehicle_type, propulsion_code, accident_severity) %>%
  filter(vehicle_type != -1 & propulsion_code != -1)

# Let's see the top 6 rows of the data:
head(d)
```

```
##   vehicle_type propulsion_code accident_severity
## 1            9               1                 3
## 2            9               2                 3
## 3           20               2                 3
## 4            9               2                 3
## 5            3               1                 3
## 6            9               2                 3
```

- Now, let’s extract all unique variables from ‘vehicle_type’.

```r
# Retrieve all unique vehicle types
d1 <- data.frame(unique(d$vehicle_type))

# Let's see the top 6 rows:
head(d1)
```

```
##   unique.d.vehicle_type.
## 1                      9
## 2                     20
## 3                      3
## 4                      8
## 5                     11
## 6                      5
```

- Now create a data frame with vehicle type as the row name, with propulsion codes and accident severity codes as columns.
- Then assign the frequency of each propulsion code for each vehicle type.
- And finally assign the frequency of each accident severity for each vehicle type

```r
# Convert the vehicle type column values to row names instead
d1 <- d1[,-1]
rownames(d1) <- unique(d$vehicle_type)

# Retrieve all unique vehicle types for the upcoming for-loop
vehicle_types <- c(unique(d$vehicle_type))

# All unique propulsion codes for the upcoming for-loop
p_codes <- c(unique(d$propulsion_code))

# All unique accident severity values for the upcoming for-loop
a_severity <- c(unique(d$accident_severity))

# Populate with frequency of propulsion code for each vehicle_type
# and accident severity frequency
for(vehicle_type in vehicle_types)
{
  for(p_code in p_codes)
  {
    p <-
      as.data.frame(
        table(d$vehicle_type[d$vehicle_type == vehicle_type
                             & d$propulsion_code == p_code]))

    p <- p$Freq

    if(length(p) < 1)
    {
      p <- 0
    }

    p_code <- paste("p", p_code, sep = "")
```

```
    d1[[p_code]][rownames(d1) == vehicle_type] <- p
  }

  for(severity in a_severity)
  {
    s <- as.data.frame(
      table(d$vehicle_type[d$vehicle_type == vehicle_type
                           & d$accident_severity == severity]))

    s <- s$Freq

    if(length(s) < 1)
    {
      s <- 0
    }

    s_code <- paste("s", severity, sep="")

    d1[[s_code]][rownames(d1) == vehicle_type] <- s
  }
}#

# Let's take a peek at the data:
head(d1)
```

```
##        p1    p2   p8 p3 p6 p12  p7 p5 p4 p10 p9     s3    s2   s1
## 9  100065 61229 1563 95 54  50 148  5  0   1  0 141777 19457 1976
## 20      1   824    0  0  0   0   0  0  0   0  0    671   132   22
## 3    7972     3    0  2  0   0   0  0  0   0  0   6251  1653   73
## 8     481  3999  577  2  3   5  16  4  0   0  0   4556   491   40
## 11     10  5667    0  6  0   0   0 29  0   0  0   4827   818   67
## 5    6216     8    0  0  0   0   0  0  0   0  0   3834  2141  249
```

- And finally substitute each vehicly type ID with its corresponding value from the Data Dictionary.

```
# Substitute vehicle type IDs with their corresponding values
# from the Data Dictionary
rownames(d1)[rownames(d1) == 1] <- "Pedal cycle"
rownames(d1)[rownames(d1) == 2] <- "Motorcycle <= 50cc"
rownames(d1)[rownames(d1) == 3] <- "Motorcycle <= 125cc"
rownames(d1)[rownames(d1) == 4] <- "Motorcycle 125cc - 500cc"
rownames(d1)[rownames(d1) == 5] <- "Motorcycle > 500cc"
rownames(d1)[rownames(d1) == 8] <- "Taxi/Private hire"
rownames(d1)[rownames(d1) == 9] <- "Car"
rownames(d1)[rownames(d1) == 10] <- "Minibus"
rownames(d1)[rownames(d1) == 11] <- "Bus or coach"
rownames(d1)[rownames(d1) == 16] <- "Ridden horse"
rownames(d1)[rownames(d1) == 17] <- "Agricultural Vehicle"
rownames(d1)[rownames(d1) == 18] <- "Tram"
rownames(d1)[rownames(d1) == 19] <- "Van / Goods <=3.5t"
rownames(d1)[rownames(d1) == 20] <- "Goods > 3.5t & < 7.5t"
rownames(d1)[rownames(d1) == 21] <- "Goods >= 7.5t"
```

```r
rownames(d1)[rownames(d1) == 22] <- "Mobility scooter"
rownames(d1)[rownames(d1) == 23] <- "Electric motorcycle"
rownames(d1)[rownames(d1) == 90] <- "Other vehicle"
rownames(d1)[rownames(d1) == 97] <- "Motorcycle - unknown cc"
rownames(d1)[rownames(d1) == 98] <- "Goods - unknown mgw"

# Let's take a peek at the data:
head(d1)
```

```
##                          p1    p2   p8 p3 p6 p12  p7 p5 p4 p10 p9     s3    s2
## Car                  100065 61229 1563 95 54  50 148  5  0   1  0 141777 19457
## Goods > 3.5t & < 7.5t     1   824    0  0  0   0   0  0  0   0  0    671   132
## Motorcycle <= 125cc    7972     3    0  2  0   0   0  0  0   0  0   6251  1653
## Taxi/Private hire       481  3999  577  2  3   5  16  4  0   0  0   4556   491
## Bus or coach             10  5667    0  6  0   0   0 29  0   0  0   4827   818
## Motorcycle > 500cc     6216     8    0  0  0   0   0  0  0   0  0   3834  2141
##                          s1
## Car                    1976
## Goods > 3.5t & < 7.5t    22
## Motorcycle <= 125cc      73
## Taxi/Private hire        40
## Bus or coach             67
## Motorcycle > 500cc      249
```

- Create the dendrogram

```r
# Load the dendextend package for customising the dendrogram
library(dendextend)

# Scale the values, this is a data normalisation step.
d1 <- scale(d1)

# Create the dendrogram by calculating the Euclidean distance
# between each row's propulsion codes and accident severity codes.
d1 %>%
  dist() %>%
  hclust %>%
  as.dendrogram() -> dend

# Customise the dendrogram by giving it nodes and
# adding colours from the Wes Anderson colour palette.v
dend <- dend %>%
  set("labels_cex",1.2) %>%
  set("leaves_pch", 19) %>%
  set("leaves_cex", 0.8) %>%
  set("leaves_col",
      value = c(wes_palette("GrandBudapest1"),
                                wes_palette("Darjeeling1")[1],
                                rep(wes_palette("Darjeeling1")[2], 7),
                                wes_palette("Darjeeling1")[3:5],
                                wes_palette("GrandBudapest2"))) %>%
  set("labels_col",
      value = c(wes_palette("GrandBudapest1"),
```

```
                wes_palette("Darjeeling1"),
                wes_palette("GrandBudapest2")), k=11) %>%
  set("branches_k_color", value = c(wes_palette("GrandBudapest1"),
                                    wes_palette("Darjeeling1"),
                                    wes_palette("GrandBudapest2")), k=11)
```

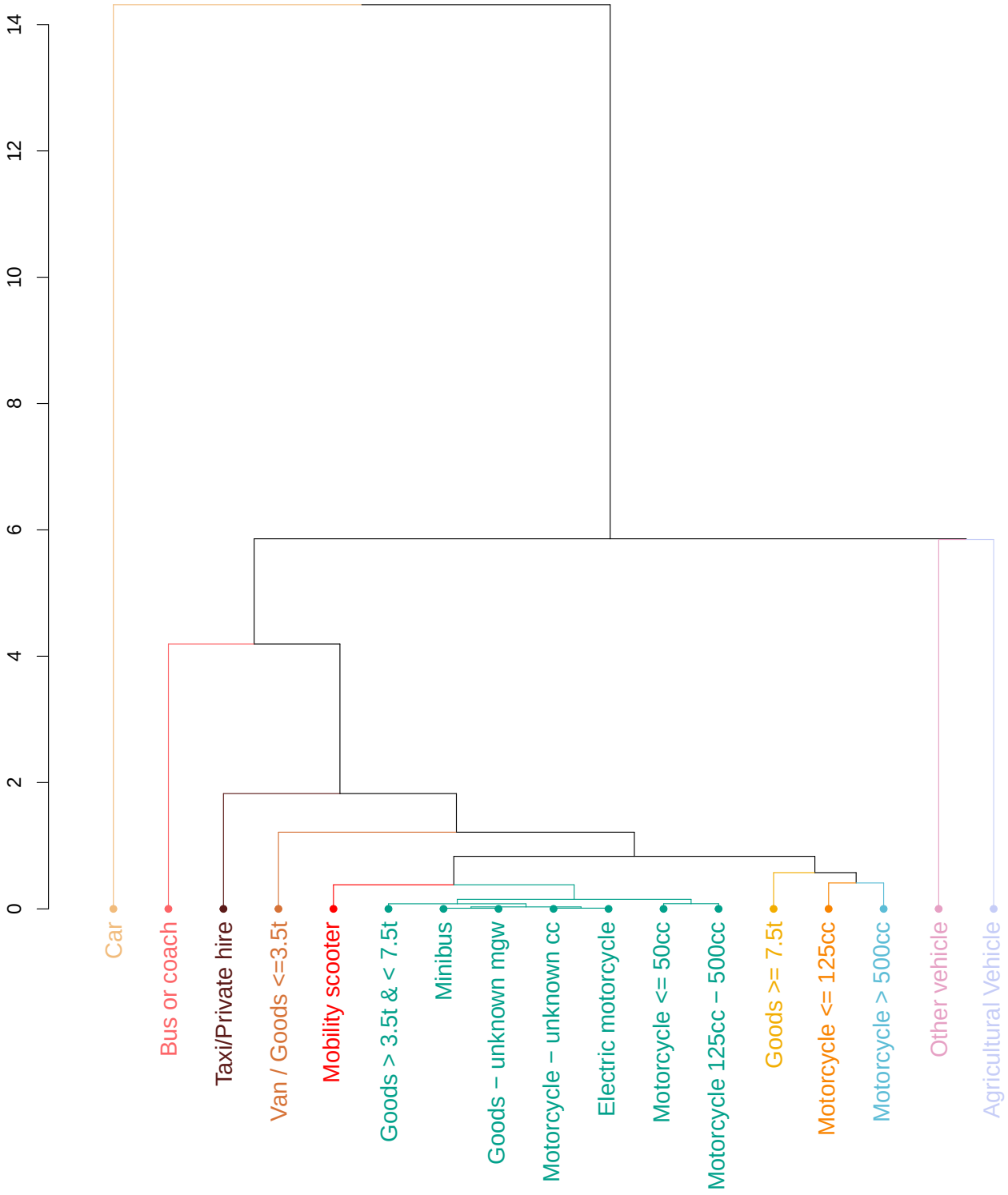


Figure 9: Hierarchical clustering of Vehicle Types

We can observe 11 major clusters with large variances with each other. How different a cluster is shown with each node's height and not by their widths.

So in this instance, Motorcycles over 500cc and Motorcycles less than or eqaul to 125cc are in the same cluster group. From there, there is a parent cluster, where Motorcycles over 500cc and Motorcycles less than or equal to 125cc is the first group, and Goods Vehicles over or equal to 7.5 tonnes is the other group in this parent cluster.

This process is repeated for each cluster until you reach the top, where every node is part of the entire cluster.

There are some very interesting observations here, cars and taxis are in totally different clusters. Another interesting observation is how similar Goods Vehicles over 7.5 tonnes are to motorcyles over 500cc, there is a common saying motorcylists and large goods vehicles are a dangerous combination, and with relation to accident severity, the dendrogram does seem to add evidence to this saying.

2.10 Using Machine Learning to compute variable importance in accidents

2.10.1 Aim

How important are the following variables in determining an accident and its severity?

- Age of Driver
- Vehicle Type
- Age of Vehicle
- Day of Week
- Weather Conditions
- Road Surface Conditions
- Light Conditions
- Sex of Driver
- Speed Limit
- Engine Capacity

2.10.2 Data Preparation & Visualisation

The best way to calculate a variable's importance in this dataset is to use Machine Learning.

We can create a model using the Random Forest algorithm that predicts an accident's severity, fatal, serious or slight, from the variables mentioned above.

So we'd need to train the model with the data, and test the model's accuracy. We'd need to achieve above 80% accuracy in classifying the accident severity from the aforementioned variables. As a general rule of thumb, 70% accuracy is the minimum threshold, but let's aim a bit higher with 80% to achieve reliable results.

Once we've trained and created the model, we can then see what variables are the most important for making our predictions, and create a horizontal bar plot from it.

For this visualisation, data preparation steps with the R code will be easier to understand:

- First extract the aforementioned variables from the data and filter them:

```
# Read data
data <- read.csv(
  "/home/dthomas/Documents/KU/Advanced_Data_Modelling/private/RoadData.csv")

# Load dplyr
library(dplyr)

# Extract and filter data
accident_ml <- data %>%
```

```
  select(accident_severity,
         age_of_driver, vehicle_type, age_of_vehicle,
         day_of_week, weather_conditions, road_surface_conditions,
         light_conditions, sex_of_driver, speed_limit,
         engine_capacity_.cc.) %>%
# Filter values for accident severity
  filter(accident_severity %in% c(1,2,3)) %>%

# Filter out null values
  filter(vehicle_type != -1
         & age_of_driver != -1
         & age_of_vehicle != -1 & road_surface_conditions != -1
         & sex_of_driver != -1 & engine_capacity_.cc. != -1)

# Now let's count the number of rows after filtering:
nrow(accident_ml)
```

```
## [1] 183197
```

- From here we need to normalise age of driver and age of vehicle using log transformation, this is because there is a possibility for ages to have large variances.

```
# Normalise data using log transformation.
accident_ml$age_of_driver = log(accident_ml$age_of_driver)
accident_ml$age_of_vehicle = log(accident_ml$age_of_vehicle)
```

- Now we can move onto machine learning, create a data partition of 60:40, 60% of the data will be used for training the model, and 40% of the data will be used for testing and evaluating the accuracy of the model.

```
# Load library for Machine Learning
library(caret)

# Create data partition for training and testing 60:40 split
train_row_numbers <- createDataPartition(accident_ml$accident_severity,
                                         p = 0.6, list=FALSE)

# Partition data
train <- accident_ml[train_row_numbers,]
test <- accident_ml[-train_row_numbers,]
```

- Let’s start training the model using ‘ranger’, a Random Forest model that utilises all available CPU cores for training.

```
# Train the model,
# this will take approximately 36.6 minutes on a Ryzen 8 core CPU.

# Create a Random Forest model, as it best suits this scenario.
# Start training.
model_rf = train(as.factor(accident_severity) ~ .,
```

```
                data = train,
                importance = "permutation",
                method = "ranger")

# Save the model to disk for later usage.
saveRDS(model_rf, "./model_rf.rds")
```

- Let's evaluate the model's accuracy in predicting accident severity.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     1     2     3
##          1   682     8    18
##          2    38  6141   325
##          3   326  3731 62009
##
## Overall Statistics
##
##                Accuracy : 0.9393
##                  95% CI : (0.9376, 0.941)
##     No Information Rate : 0.8509
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7251
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity          0.652008  0.62156   0.9945
## Specificity          0.999640  0.99427   0.6287
## Pos Pred Value       0.963277  0.94419   0.9386
## Neg Pred Value       0.994984  0.94401   0.9524
## Prevalence           0.014274  0.13483   0.8509
## Detection Rate       0.009307  0.08380   0.8462
## Detection Prevalence 0.009662  0.08876   0.9016
## Balanced Accuracy    0.825824  0.80792   0.8116
```

- We are seeing over 93% accuracy in classifying accident severity
- Now let's obtain and store the variable importance from our model

```
# Let's obtain and store variable_importance as a variable
variable_importance <- varImp(model_rf)
```

- The variable importance is in percentages, so all we need to do now is plot the data.

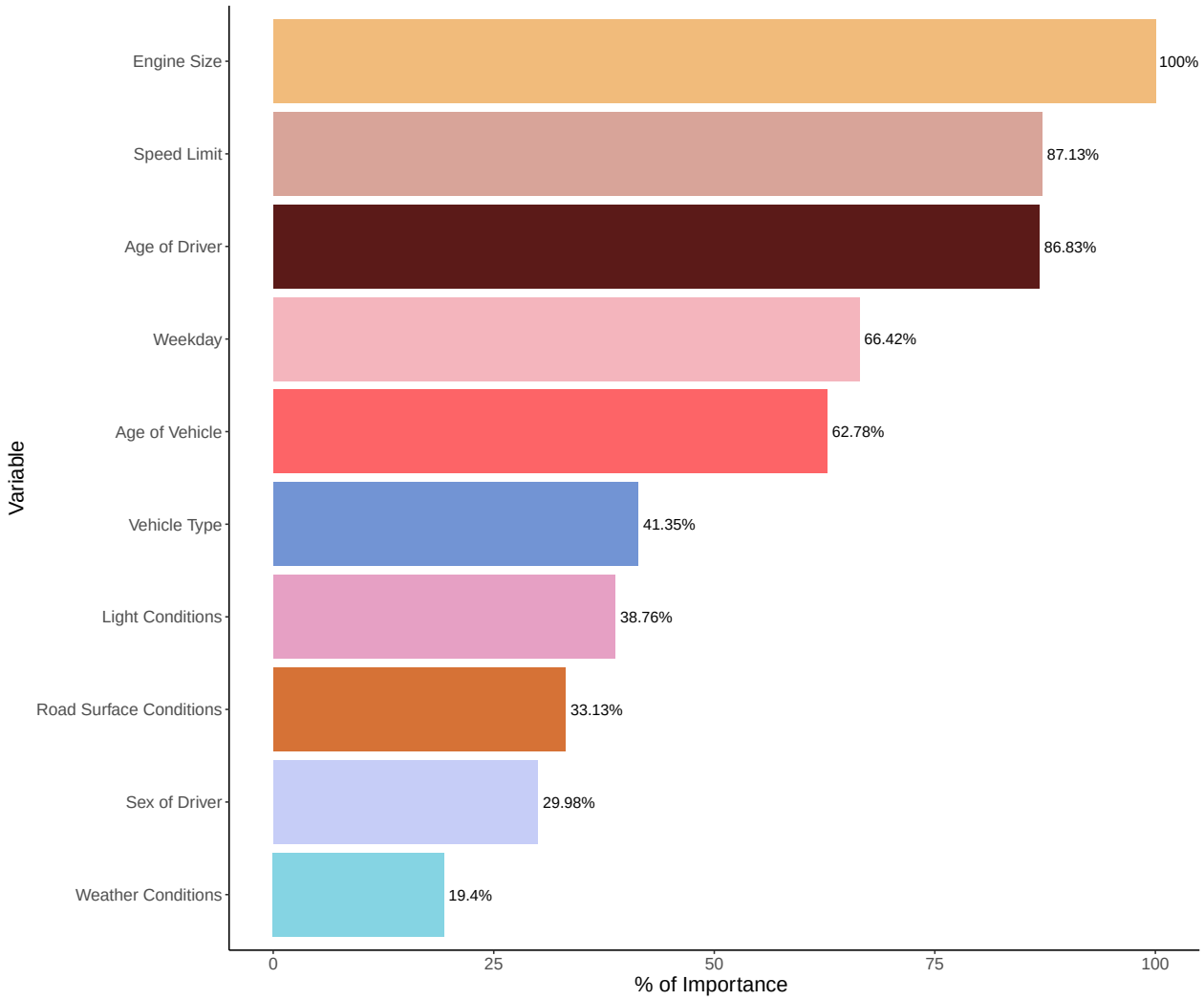


Figure 10: Importance of a variable for accidents

3 Conclusion

4 References