Project

2023-03-30

```
library(readr)
library(tidyr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(modelr)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
```

```
## v tibble  3.2.1     v stringr 1.5.0
## v purrr   1.0.1     v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(mlbench)
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(smotefamily)
library(usmap)
```

Load the dataset

```
df_rail <- read_csv("Rail_Equipment_Accident_Incident_Data.csv")
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
```

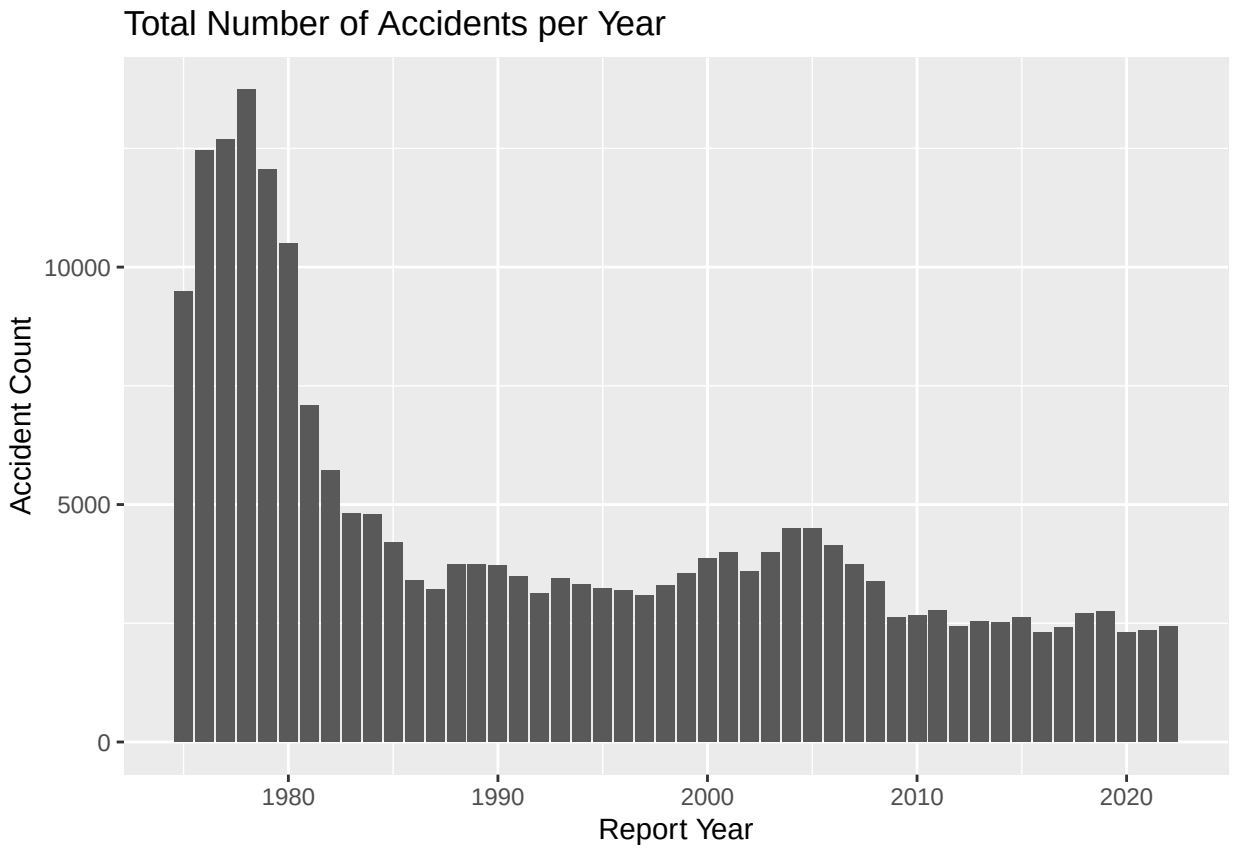
```
##   problems(dat)
```

```
## Rows: 216100 Columns: 160
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (91): Reporting Railroad Code, Reporting Railroad Name, Accident Number,...
## dbl (67): Report Year, Hazmat Cars, Hazmat Cars Damaged, Hazmat Released Car...
## lgl  (2): Adjunct Code 3, Adjunct Code Name 3
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(df_rail)
```

Visualization 1: Total Number of Accidents per Year

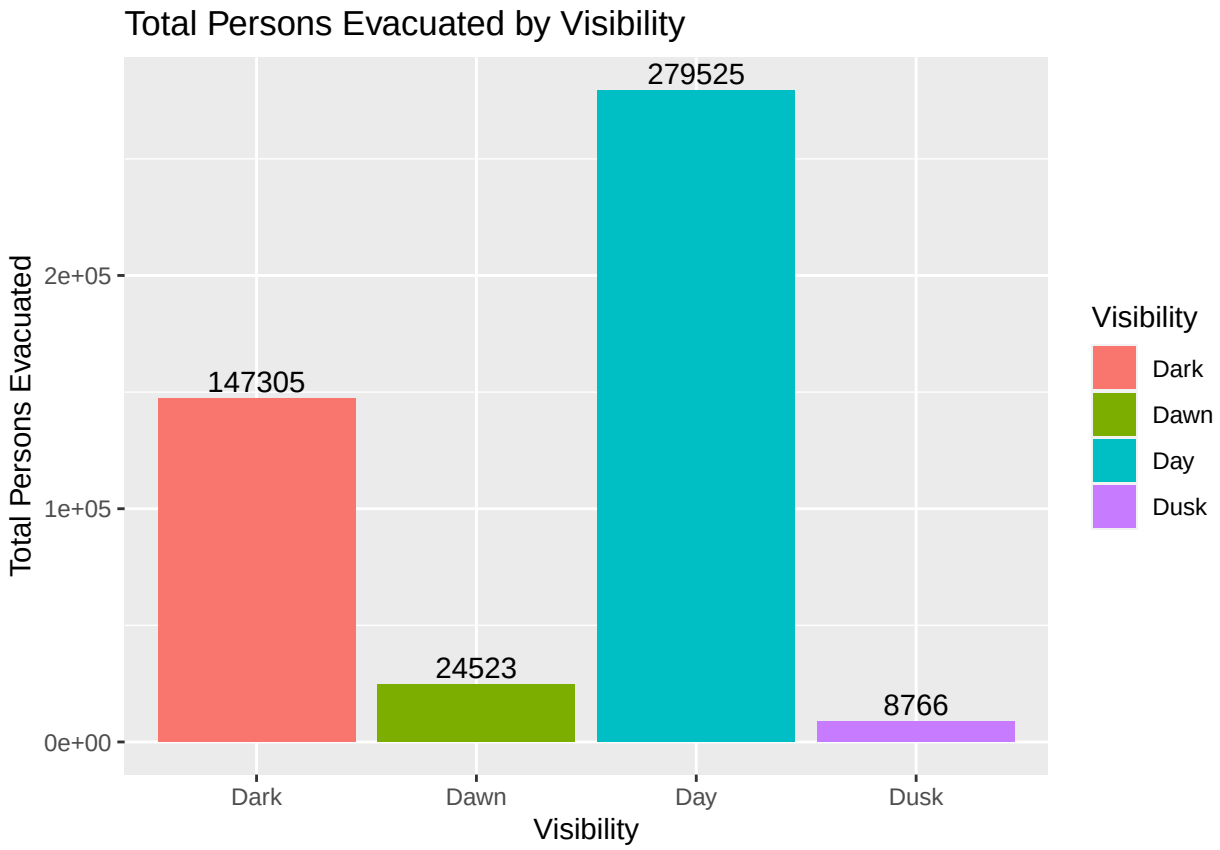
```
#storing how many accidents occurred per year in a data frame
df <- df_rail %>%
  group_by(`Report Year`) %>%
  summarise(counts = n())

#dropping NA values from data frame
df <- df %>% drop_na()
#df

ggplot(df, aes(x = `Report Year`, y = counts)) +
  geom_bar(stat = "identity") +
  labs(x="Report Year",
       y="Accident Count",
       title="Total Number of Accidents per Year")
```



Observation: From the above visualization, we can conclude that railroad accidents have overall decreased in the past 48 years. Additionally, the accidents reported over the past 10 years have all been around 2500.

Visualization 2: Total Persons Evacuated by Visibility

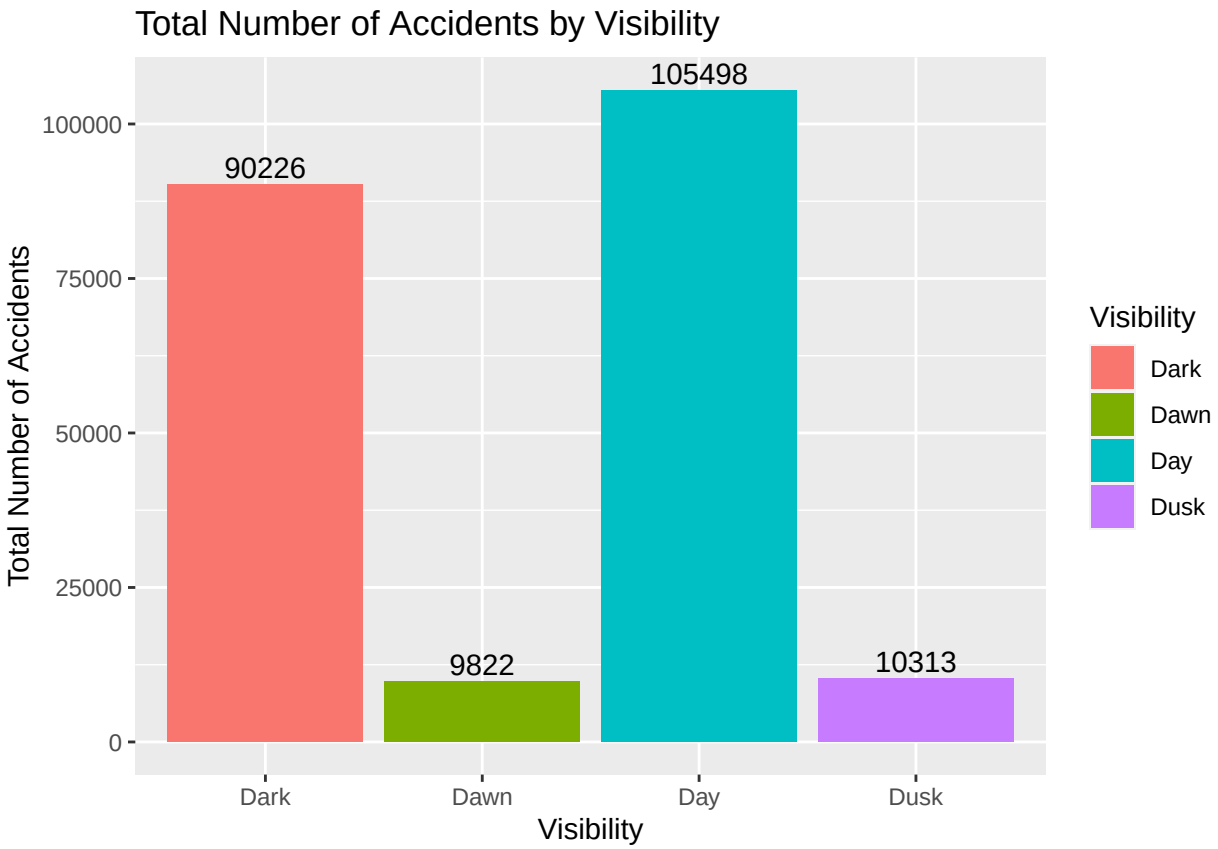
```
#storing total evacuated for each category of visibility in a data frame
df1 <- df_rail %>%
  select(`Persons Evacuated`, `Visibility`) %>%
  group_by(`Visibility`) %>%
  summarise(total_persons_evacuated = sum(`Persons Evacuated`))

#dropping NA values from data frame
df1 <- df1 %>% drop_na()
#df1

ggplot(df1, aes(Visibility, total_persons_evacuated, fill=Visibility))+
  geom_bar(stat = "identity") +
  geom_text(aes(label = total_persons_evacuated), vjust = -0.3) +
  labs(x="Visibility",
       y="Total Persons Evacuated",
       title="Total Persons Evacuated by Visibility")
```



Observation: From the above visualization, we can conclude that more number of people were evacuated during the Day and the least number of people are evacuated during Dusk.

Visualization 3: Total Number of Accidents by Visibility

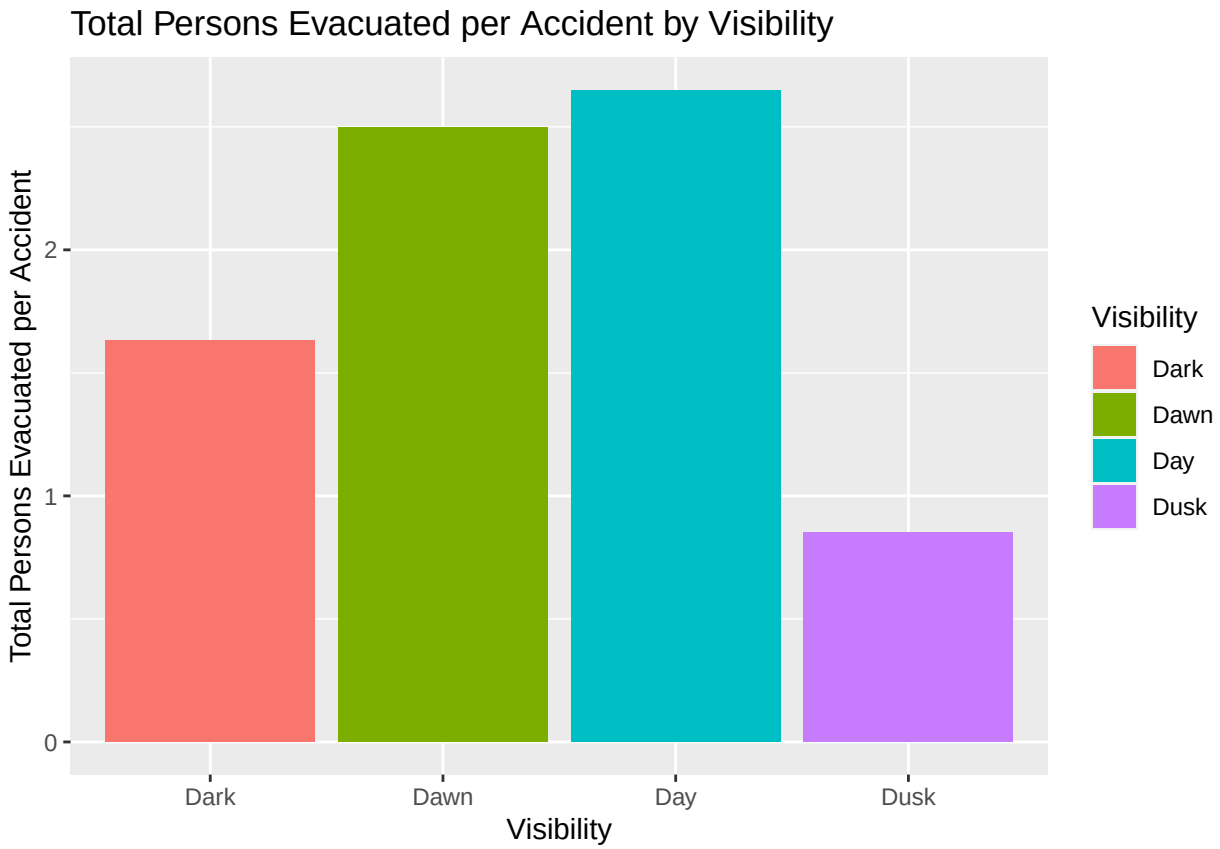
```
#storing total accidents for each category of visibility in a data frame
df2 <- df_rail %>%
  select(`Persons Evacuated`, `Visibility`) %>%
  group_by(`Visibility`) %>%
  summarise(count_of_accidents = n())

#dropping NA values from data frame
df2 <- df2 %>% drop_na()
#df2

ggplot(df2, aes(Visibility, count_of_accidents, fill=Visibility))+
  geom_bar(stat = "identity") +
  geom_text(aes(label = count_of_accidents), vjust = -0.3) +
  labs(x="Visibility",
       y="Total Number of Accidents",
       title="Total Number of Accidents by Visibility")
```



Observation: From the above visualization, we can conclude that the most accidents occurred during the Day (with Dark being a close second for the visibility with the most accidents). The least number of accidents occurred during Dawn and Dusk.

Visualization 4: Total Persons Evacuated per Accident by Visibility

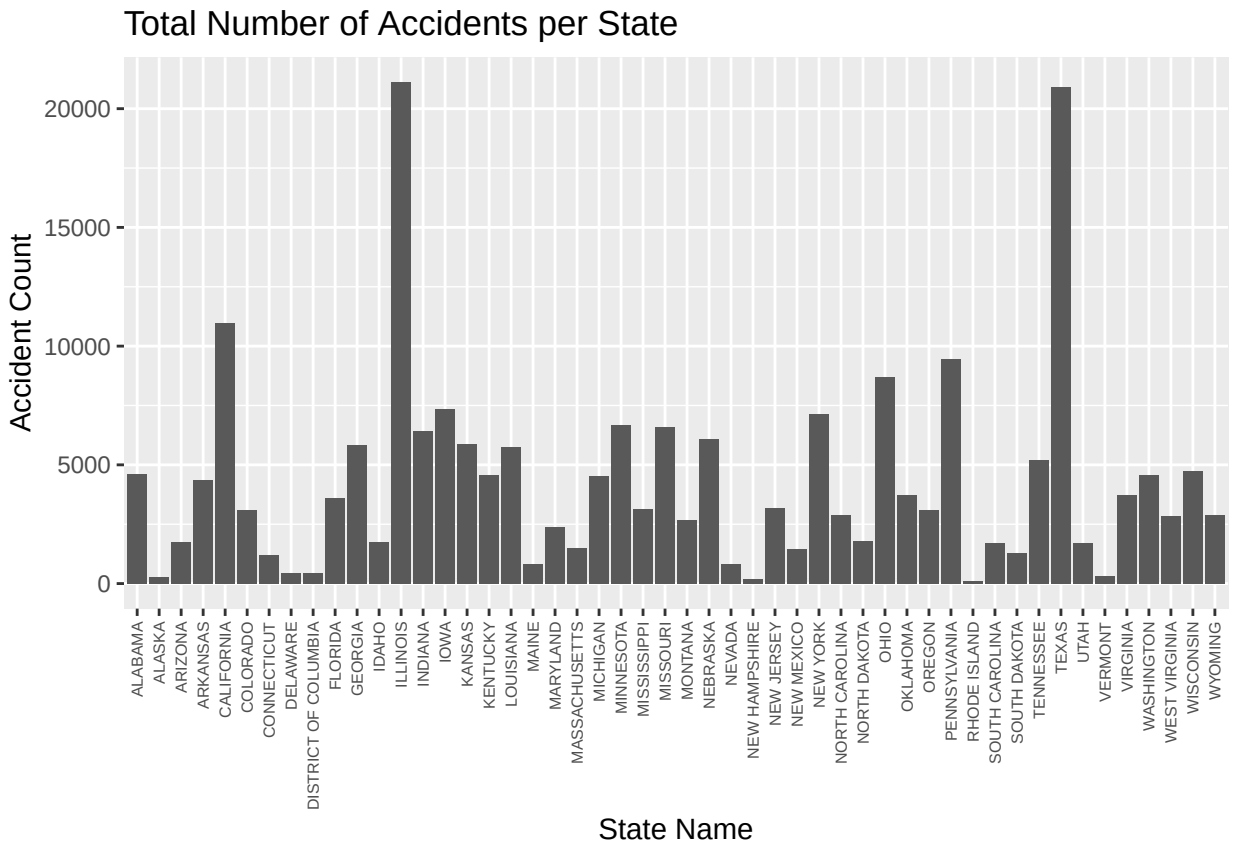
```
#storing the proportion of people evacuated for each category of visibility in a data frame
df3 <- df_rail %>%
  select(`Persons Evacuated`, `Visibility`) %>%
  group_by(`Visibility`) %>%
  summarise(evacuation_accident_ratio = sum(`Persons Evacuated`)/n())

#dropping NA values from data frame
df3 <- df3 %>% drop_na()
#df3

ggplot(df3, aes(Visibility, evacuation_accident_ratio, fill=Visibility))+
  geom_bar(stat = "identity") +
  labs(x="Visibility",
       y="Total Persons Evacuated per Accident",
       title="Total Persons Evacuated per Accident by Visibility")
```



Observation: From the above visualization, we can conclude that on average more people are evacuated during the Day and the least number of people are evacuated during Dusk.

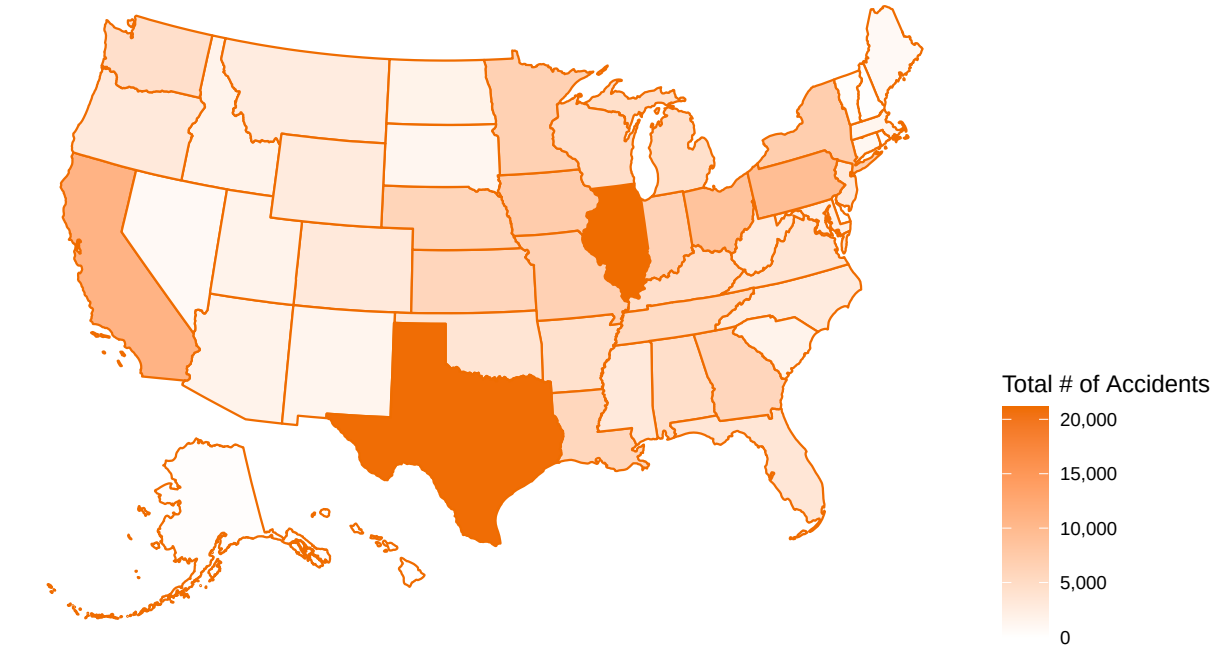
Visualization 5: Total Number of Accidents per State

```
#storing the total accidents for each state in a data frame
df4 <- df_rail %>%
  group_by(`State Name`) %>%
  summarise(counts = n())

#dropping NA values from data frame
df4 <- df4 %>% drop_na()
#df4

ggplot(df4, aes(x = `State Name`, y = counts)) +
  geom_bar(stat = "identity") +
    theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1, size = 6)) +
  labs(x="State Name",
       y="Accident Count",
       title="Total Number of Accidents per State")
```



Visualization 6: Total Number of Accidents per State Heatmap

```
df4Copy <- df4

statepop$full <- toupper(statepop$full)

colnames(df4Copy)[1] <- "full"

total <- merge(statepop,df4Copy,by="full", all.x = TRUE)

total[is.na(total)] <- 0

plot_usmap(data = total, values = "counts", include = c("AL", "AK", "AZ", "AR", "CA", "CO", "CT", "DE",
  scale_fill_continuous(low = "white", high = "#ef6c00", name = "Total # of Accidents", label = scales:
```

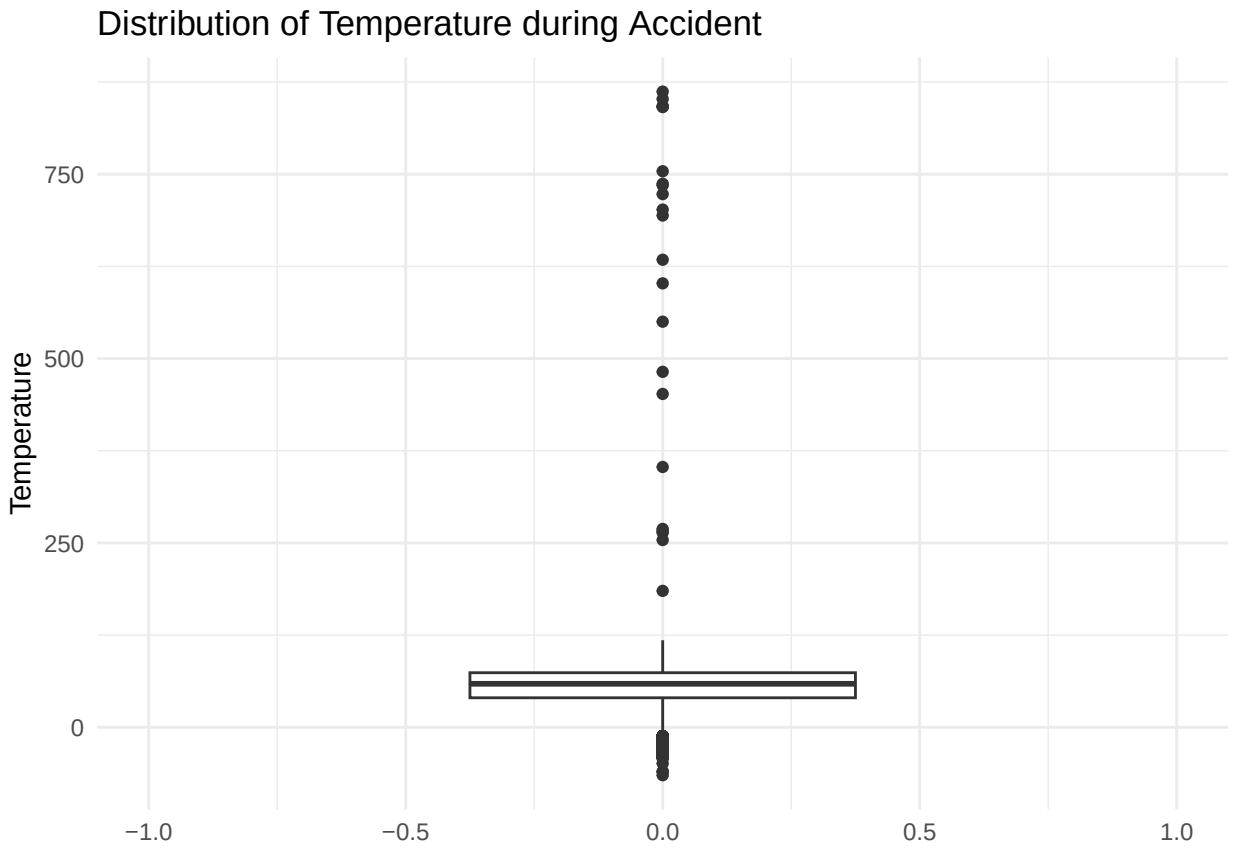


Observation: From the above two visualizations, we can conclude that the most number of accidents are caused in Illinois and Texas followed by California. The least number of accidents occurred in Rhode Island and New Hampshire. The number of accidents may be a reflection on how many trains in general go to that state. Often states with high volumes of tourists or with large import/exports have a greater chance of accidents.

Visualization 6: Distribution of Temperature during Accident

```
df5 <- df_rail

ggplot(df5, aes(y=Temperature)) +
  geom_boxplot() +
  lims(x=c(-1, 1)) +
  labs(title="Distribution of Temperature during Accident",
       y="Temperature") +
  theme_minimal()
```

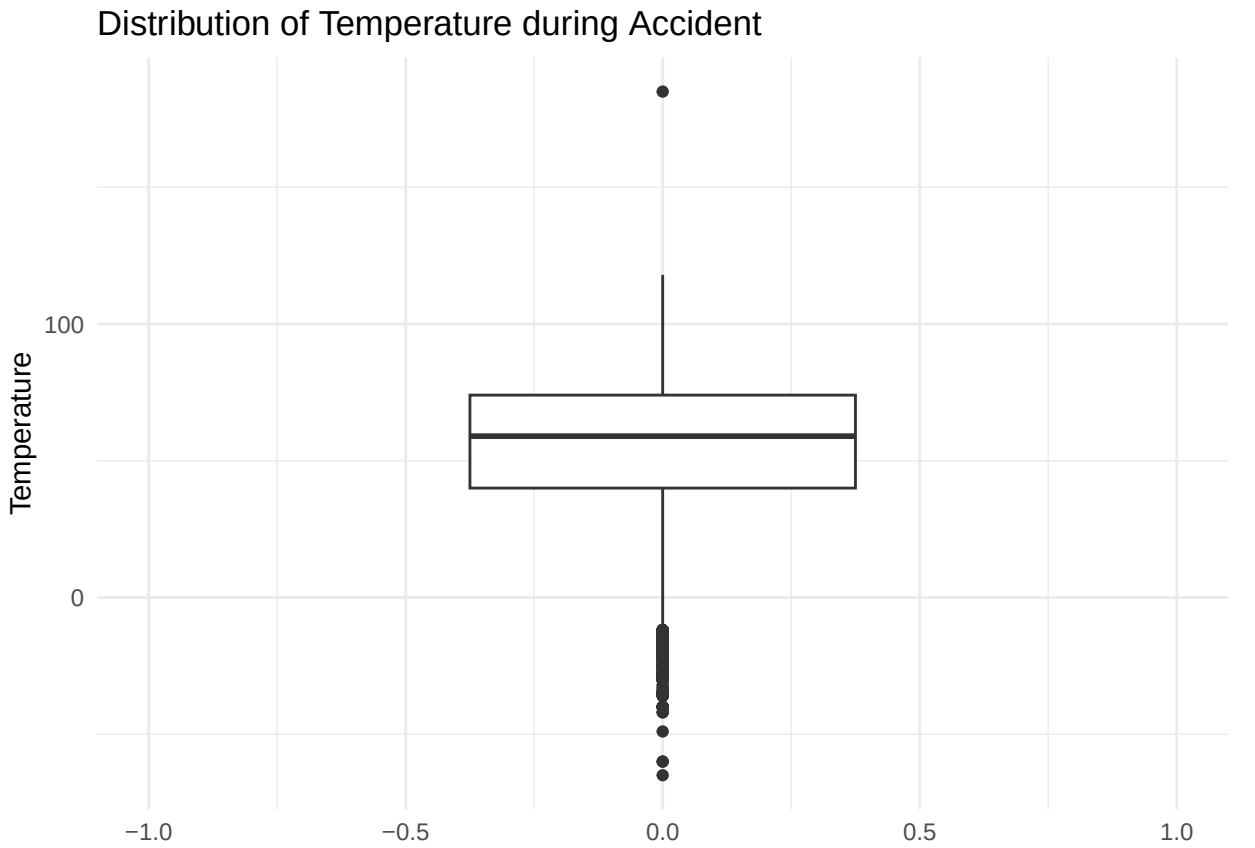
```
## Warning: Removed 1 rows containing non-finite values (`stat_boxplot()`).
```



Observation: From the above visualization, we can conclude that the temperatures range from -75 to 815 degrees Fahrenheit. There are a few outliers where the temperature is extremely high so we can filter out values greater than 200 degrees Fahrenheit.

Visualization 7: Distribution of Temperature during Accident with Outliers Excluded

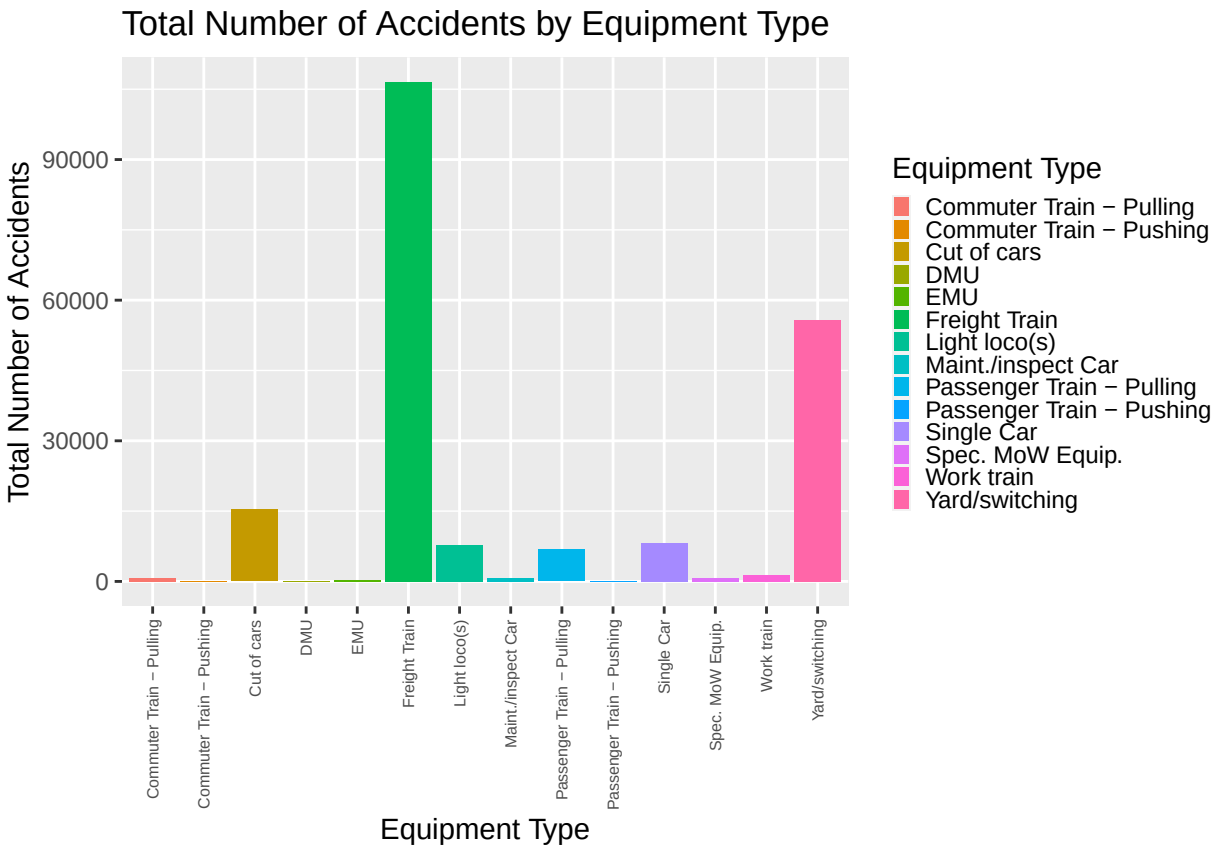
```
#filtering out all temperatures above 200 F and storing data in a data frame
df5_filtered <- filter(df5, Temperature < 200)

ggplot(df5_filtered, aes(y=Temperature)) +
  geom_boxplot() +
  lims(x=c(-1, 1)) +
  labs(title="Distribution of Temperature during Accident",
       y="Temperature") +
  theme_minimal()
```



Observation: From the above visualization, we can conclude that the average temperature during railroad accidents is around 60 degrees Fahrenheit.

Visualization 8: Total Number of Accidents by Equipment Type

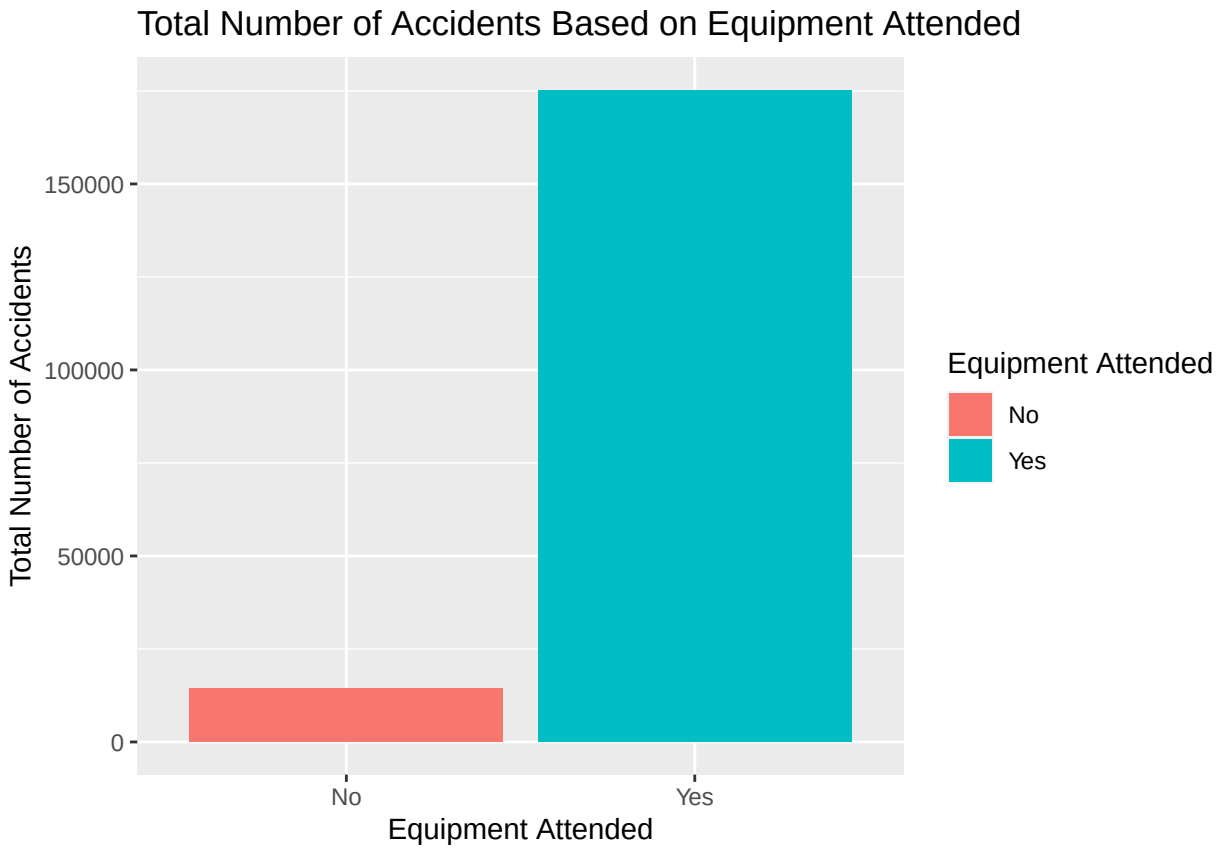
```
#storing the total accidents by equipment type in a data frame
df6 <- df_rail %>%
  group_by(`Equipment Type`) %>%
  summarise(count_of_accidents = n())

#dropping NA values from data frame
df6 <- df6 %>% drop_na()
#df6

ggplot(df6, aes(`Equipment Type`, count_of_accidents, fill=`Equipment Type`))+
  geom_bar(stat = "identity") +
  labs(x="Equipment Type",
       y="Total Number of Accidents",
       title="Total Number of Accidents by Equipment Type") +
    theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1, size = 6), legend.key.size=unit(
```



Observation: From the above visualization, we can conclude that most of the accidents were caused when the equipment type was Freight Train followed by Yard/switching. The least accidents were caused when the equipment type was DMU.

Visualization 9: Total Number of Accidents Based on Equipment Attended

```
#The `Equipment Attended` column contains "yes", "no", 0, and 7 values. Since 0 and 7 appear to be inco
df7 <- filter(df_rail, `Equipment Attended` == 'No' | `Equipment Attended` == 'Yes')

ggplot(df7, aes(`Equipment Attended`, fill=`Equipment Attended`))+
  geom_bar(stat="count",
           position="identity") +
  labs(x="Equipment Attended",
       y="Total Number of Accidents",
       title="Total Number of Accidents Based on Equipment Attended")
```

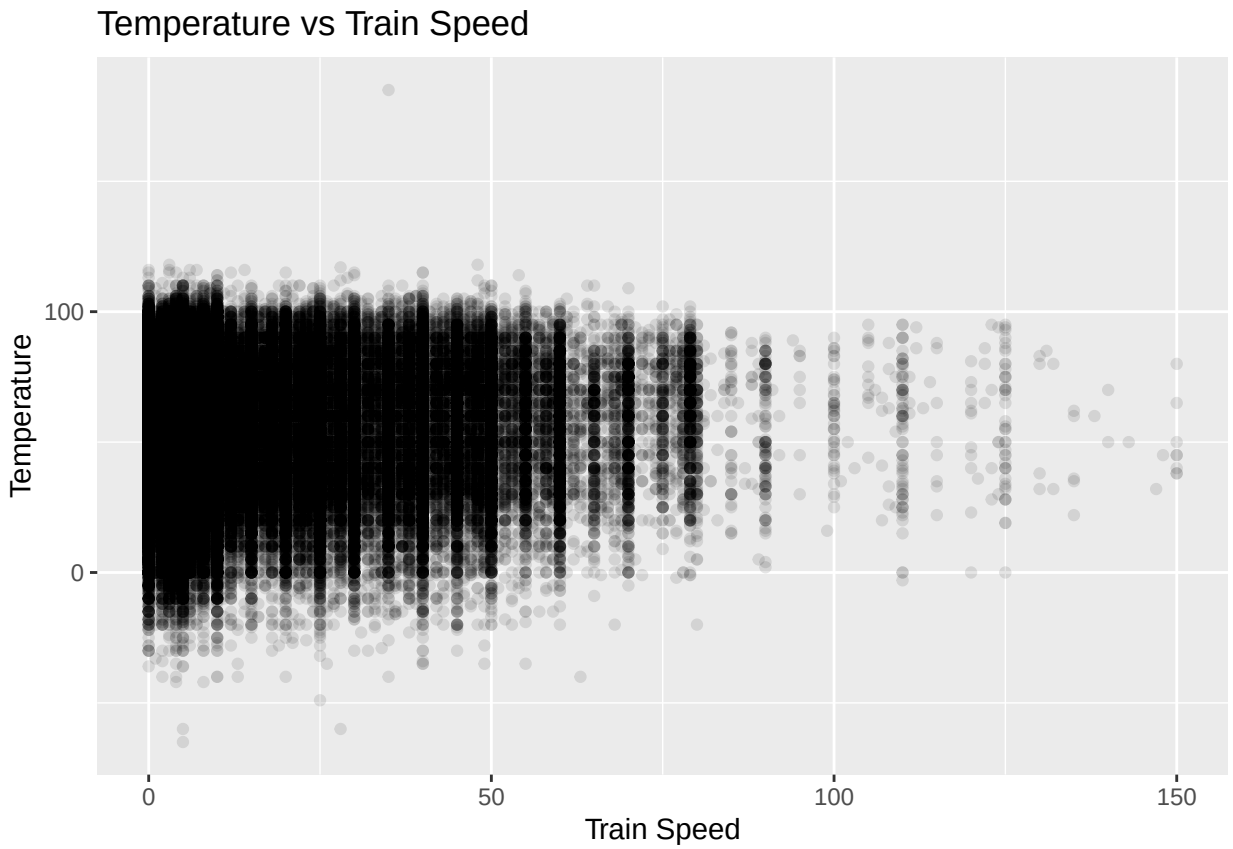


Observation: From the above visualization, we can conclude that for most of the accidents, the equipment was attended when the accident took place.

Visualization 10: Temperature vs Train Speed

```
#filtering out temperture outliers
df8 <- filter(df_rail, Temperature < 200)

ggplot(data = df8,
       mapping = aes(x=`Train Speed`, y=`Temperature`)) +
       geom_point(alpha=0.1) +
       labs(x="Train Speed",
       y="Temperature",
       title="Temperature vs Train Speed")
```

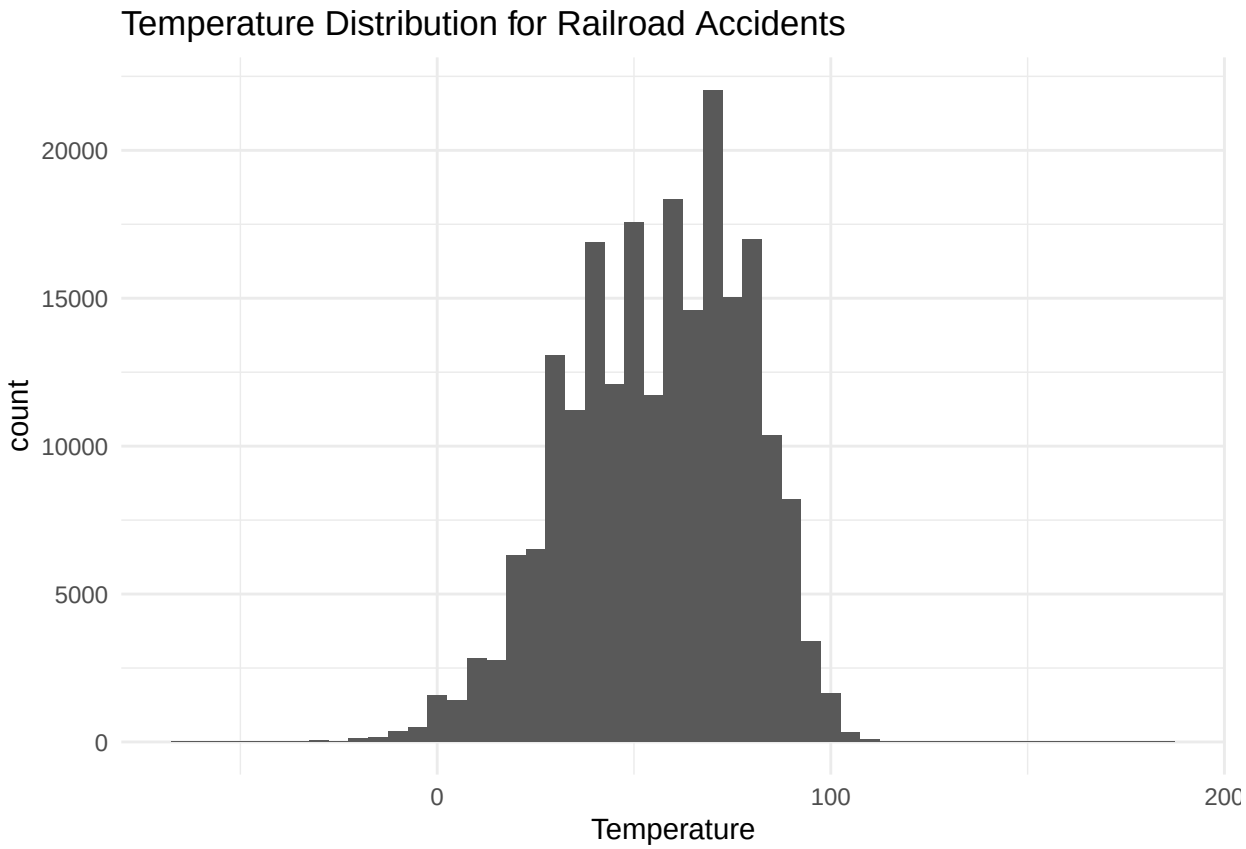
```
## Warning: Removed 2 rows containing missing values (`geom_point()`).
```



Observation: From the above visualization, we can conclude that temperatures usually ranged from -50 to 125 degrees Fahrenheit when the train speed is between 0 and 75. The range to temperatures got smaller as train speeds increased over 75.

Visualization 11: Temperature Distribution for Railroad Accidents

```
ggplot(df8, aes(x=Temperature)) +
  geom_histogram(binwidth=5) +
  labs(x="Temperature",
       title="Temperature Distribution for Railroad Accidents") +
  theme_minimal()
```

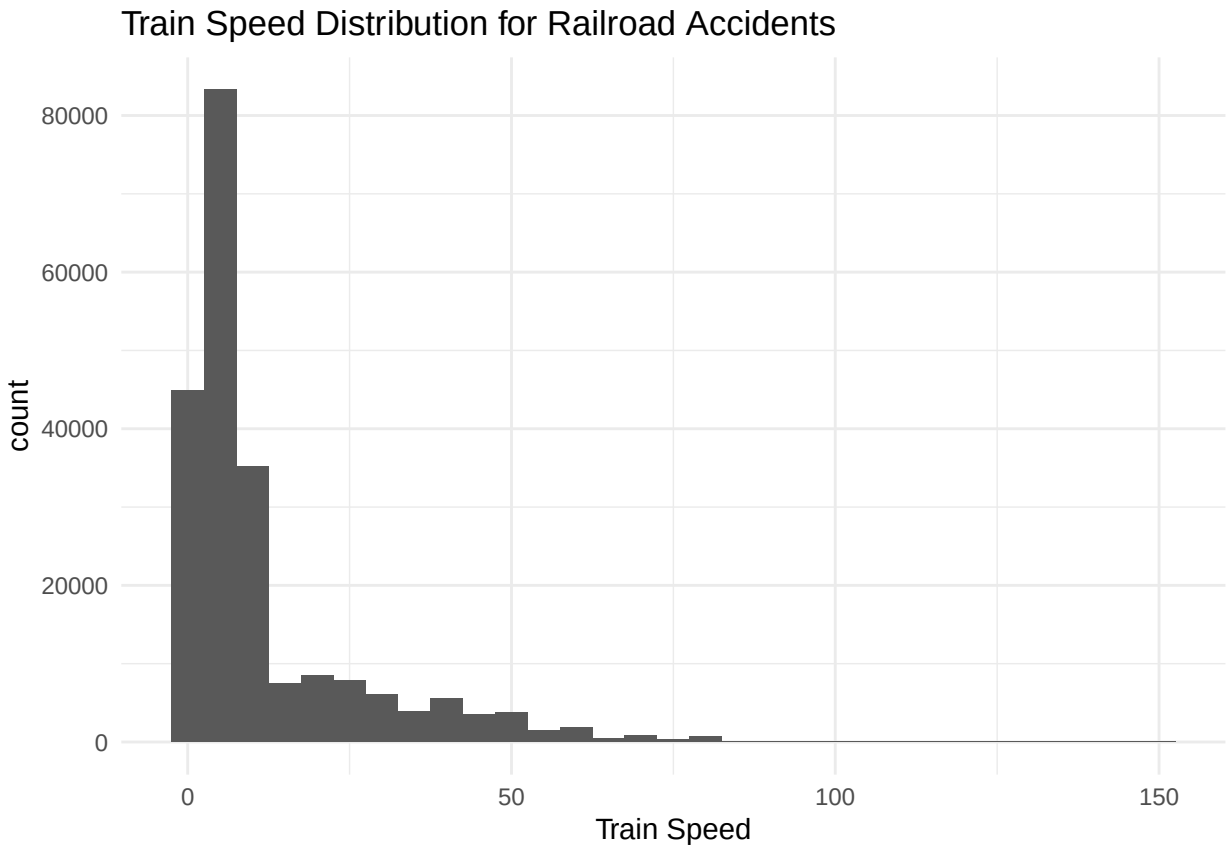


Observation: From the above visualization, we can conclude that temperature produces a normal distribution and most of the accidents are caused when the temperature is around 60 degrees Fahrenheit.

Visualization 12: Train Speed Distribution for Railroad Accidents

```
ggplot(df8, aes(x=`Train Speed`)) +
  geom_histogram(binwidth=5) +
  labs(x="Train Speed",
       title="Train Speed Distribution for Railroad Accidents") +
  theme_minimal()
```

```
## Warning: Removed 2 rows containing non-finite values (`stat_bin()`).
```



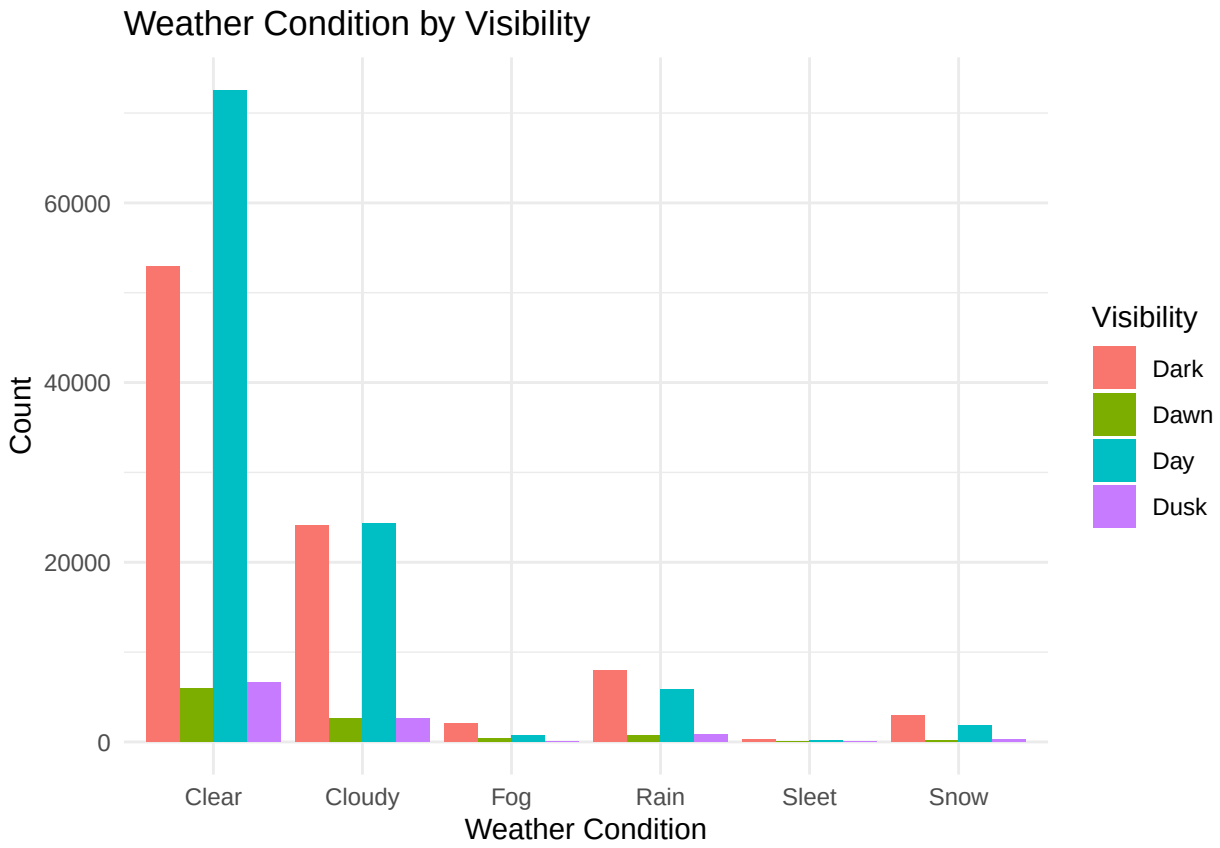
Observation: From the above visualization, we can conclude that most of the accidents are caused when the Train speed is between 0 to 15 or when it is low.

Visualization 12: Weather Condition by Visibility

```
viz_data <- df_rail %>% select(`Weather Condition`, `Weather Condition Code`, `Visibility`, `Visibility

viz_data <- na.omit(viz_data)
#Remove empty string data
rows_with_empty_strings_visibility <- grepl("^$", viz_data$Visibility)
rows_with_empty_strings_weather <- grepl("^$", viz_data$`Weather Condition`)
viz_data <- viz_data[!rows_with_empty_strings_visibility,]
viz_data <- viz_data[!rows_with_empty_strings_weather,]

viz_data <- viz_data %>%
  group_by(`Weather Condition`, Visibility) %>%
  summarize(Count = n())
```

```
## `summarise()` has grouped output by 'Weather Condition'. You can override using
## the `.groups` argument.
```

```
viz_data <- na.omit(viz_data)

ggplot(viz_data, aes(x = `Weather Condition`, y = Count, fill = Visibility)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Weather Condition by Visibility", x = "Weather Condition", y = "Count", fill = "Visibili
  theme_minimal()
```

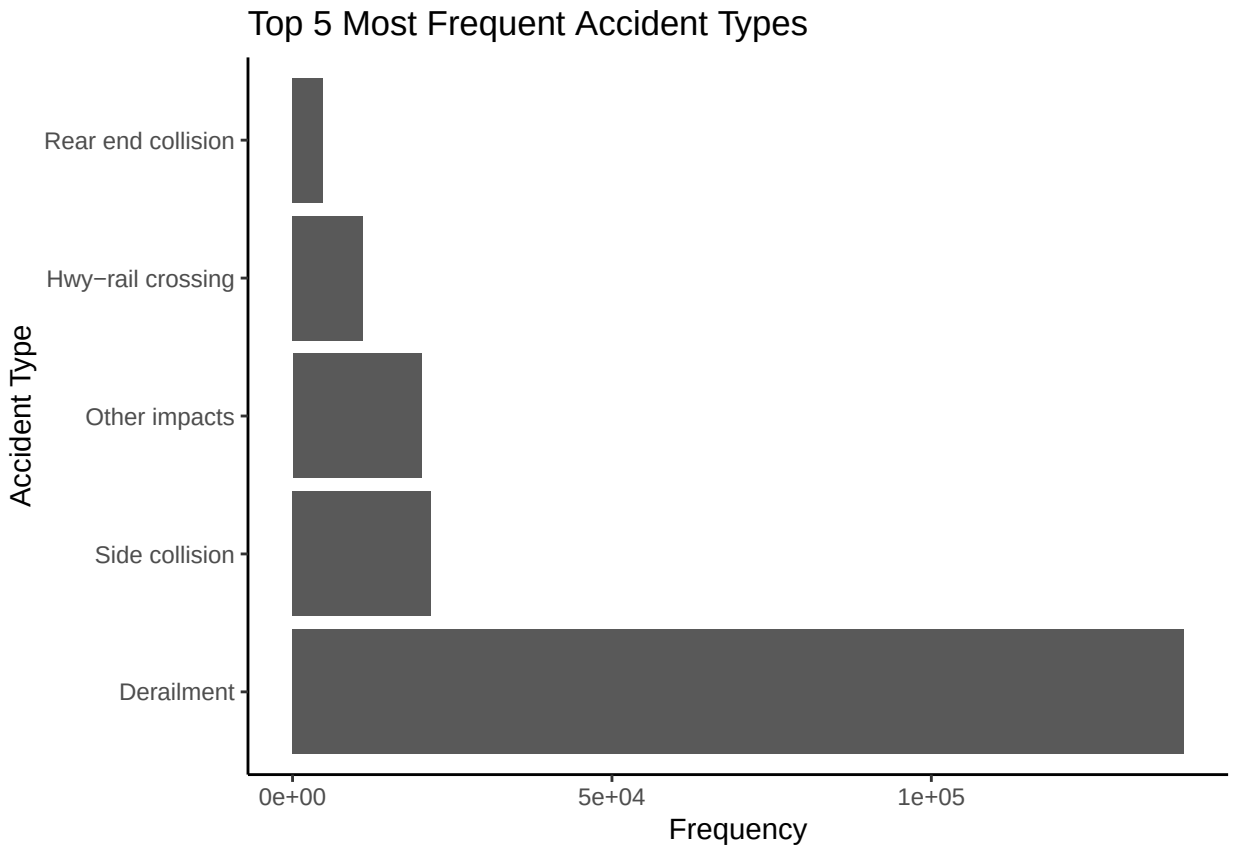


Observation: From the above visualization, we can conclude that the most accidents were caused in the day with a clear weather. Across all weather conditions we can notice that most accidents occurred either in the dark or in the day. The least accidents occurred during sleet.

Visualization 13: Top 5 Most Frequent Accident Types

```
viz2_data <- df_rail %>% select(`Accident Type`)
ggplot(viz2_data, aes(x = `Accident Type`)) +
  geom_bar() +
  labs(title = "Top 5 Most Frequent Accident Types",
       x = "Accident Type",
       y = "Frequency") +
  coord_flip() +
  theme_classic() +
  scale_x_discrete(limits = names(sort(table(viz2_data$`Accident Type`), decreasing = TRUE)[1:5]))
```

```
## Warning: Removed 19017 rows containing non-finite values (`stat_count()`).
```

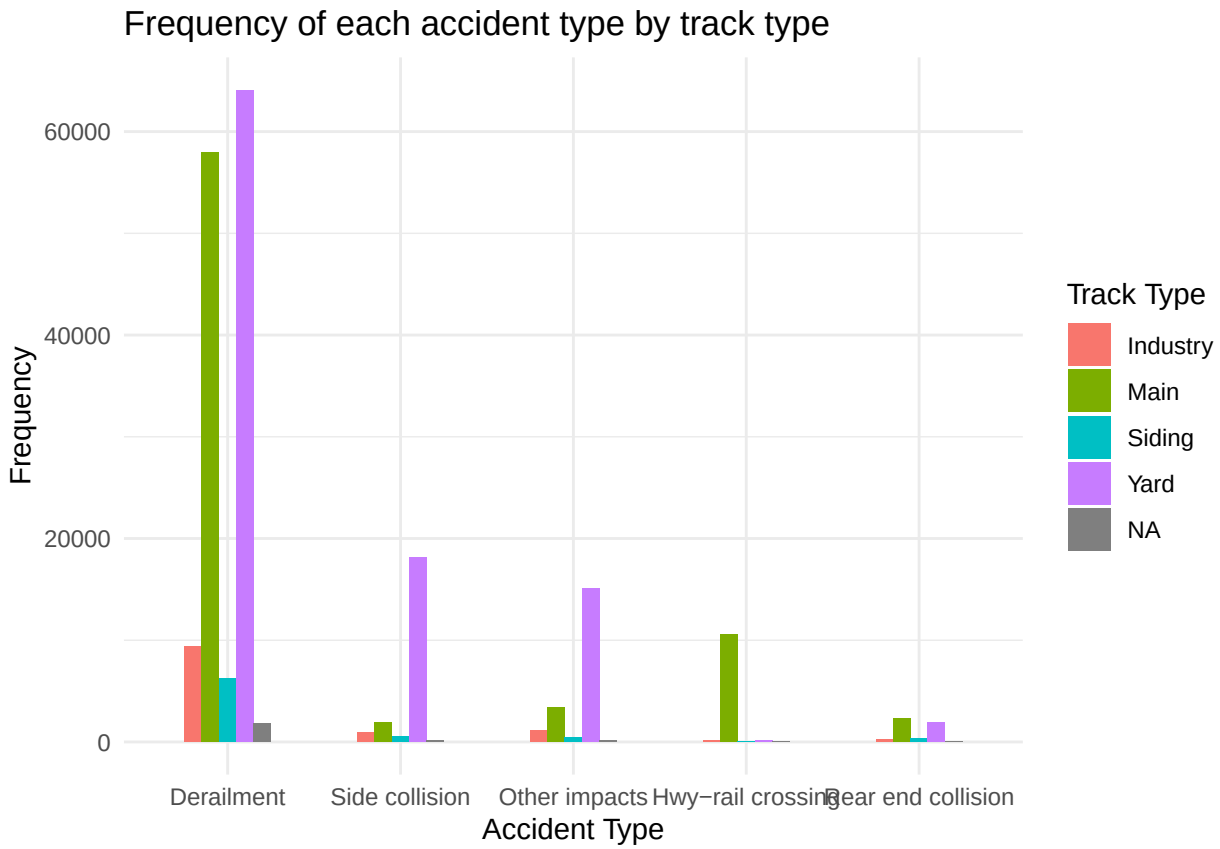


Observation: From the above visualization, we can conclude that Derailment was the most frequent type of accident that occurred. Rear-end collision was the least frequent type of accident.

Visualization 14: Frequency of each accident type by track type

```
viz3_data <- df_rail %>% select(`Track Type`, `Accident Type`)

#Pre-processing to remove empty values
rows_with_empty_strings_track <- grepl("^$", viz3_data$`Track Type`)
viz3_data <- viz3_data[!rows_with_empty_strings_track,]

#Visualization
ggplot(viz3_data, aes(x = `Accident Type`, fill = `Track Type`)) +
  geom_bar(position = "dodge", width = 0.5) +
  labs(title = "Frequency of each accident type by track type",
       x = "Accident Type",
       y = "Frequency",
       fill = "Track Type") +
  theme_minimal() +
  scale_x_discrete(limits = names(sort(table(viz3_data$`Accident Type`), decreasing = TRUE)[1:5]))
```

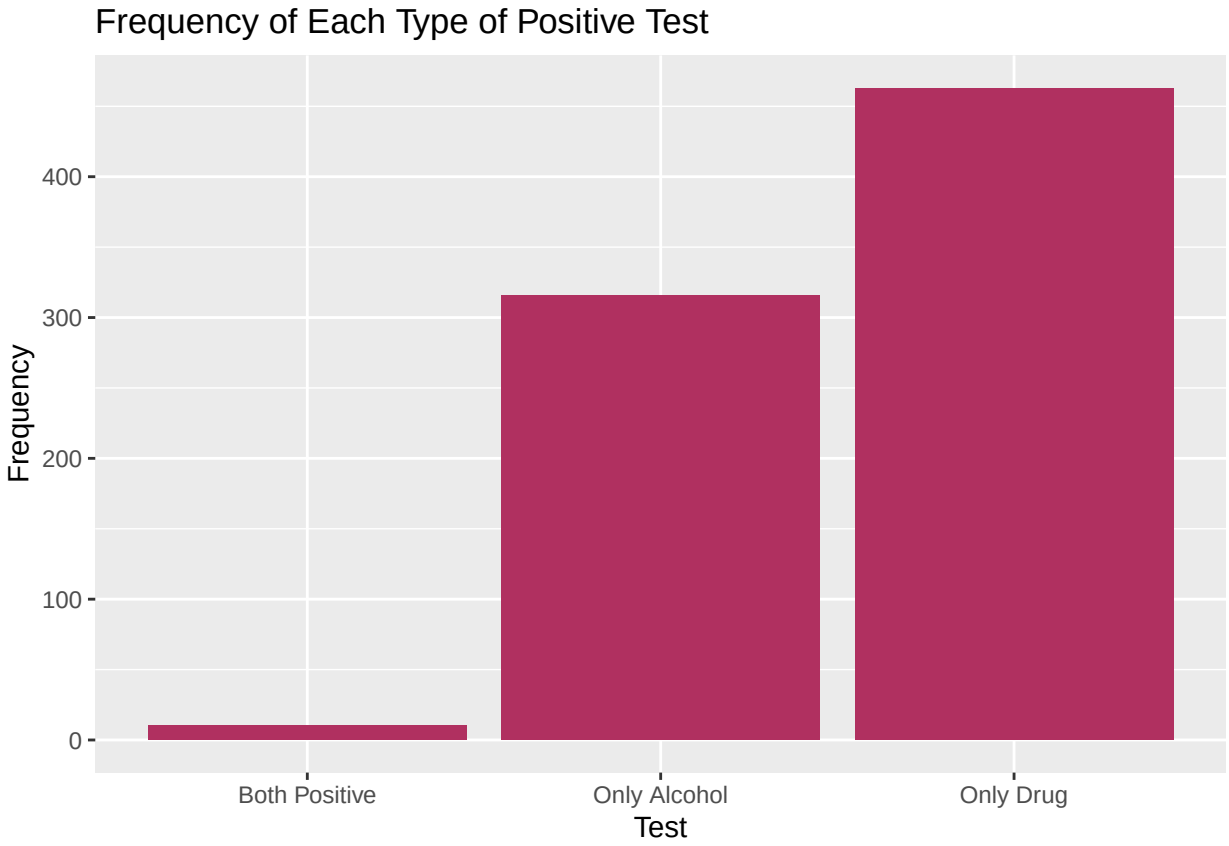
```
## Warning: Removed 19017 rows containing non-finite values (`stat_count()`).
```



Observation: From the above visualization, we can conclude that Derailment was the most frequent type of accident that occurred. Across all accident types, the accidents often involved either main or yard track types.

Visualization 15: Frequency of Each Type of Positive Test

```
drug_impact <- df_rail %>% select(`Positive Alcohol Tests`, `Positive Drug Tests`)
drug_impact <- na.omit(drug_impact)

drug_impact$`Positive Alcohol Tests` <- as.logical(drug_impact$`Positive Alcohol Tests`)
drug_impact$`Positive Drug Tests` <- as.logical(drug_impact$`Positive Drug Tests`)

counts <- table(drug_impact)

viz4_data <- as.data.frame(counts)
viz4_data$labels <- c("Both Negative", "Only Alcohol", "Only Drug", "Both Positive")
viz4_data <- viz4_data[-1,]

# Create stacked bar plot
ggplot(viz4_data, aes(x = labels, y = Freq)) +
  geom_bar(stat = "identity", fill = "maroon") +
  labs(y = "Frequency", x="Test", title="Frequency of Each Type of Positive Test")
```



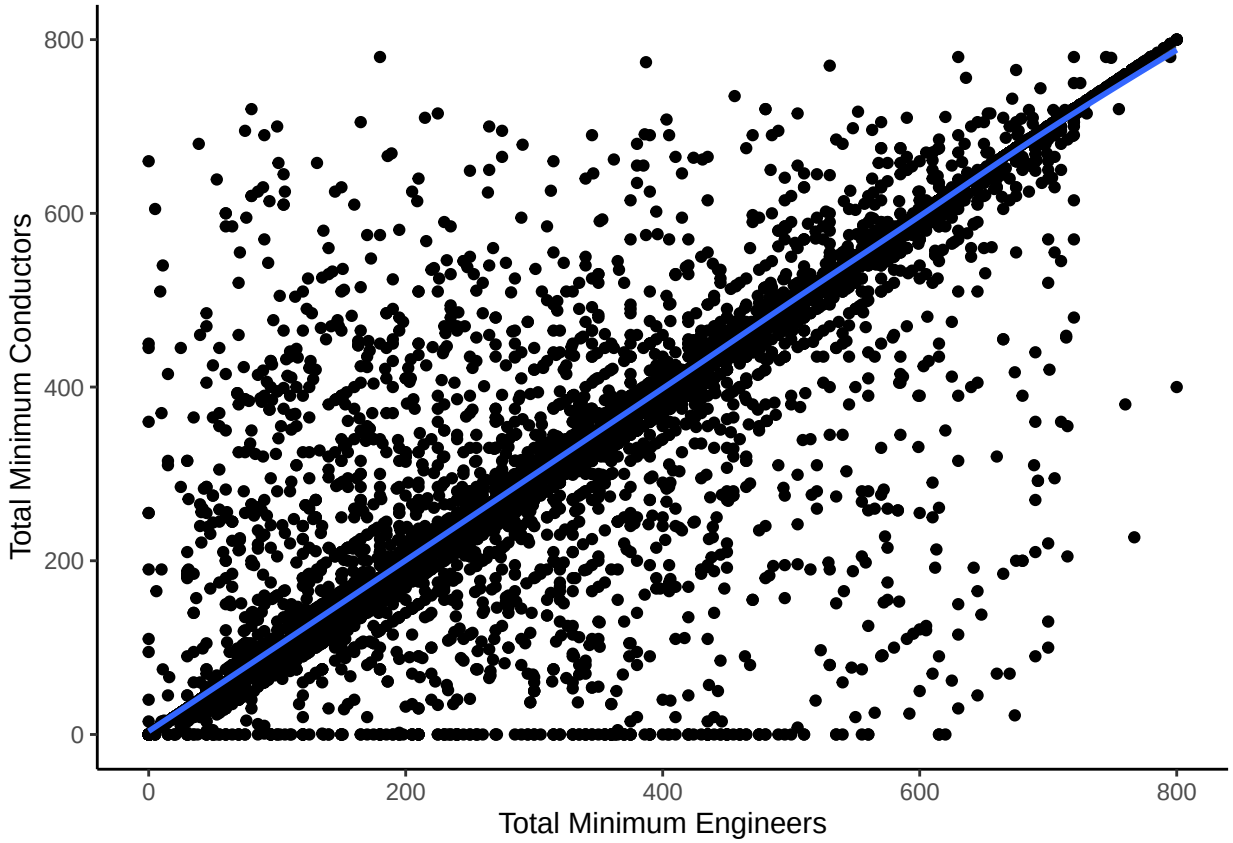
Observation: From the above visualization, we can conclude that majority of the tests done showed the presence of drugs. It was pretty rare for both alcohol and drug tests to be positive simultaneously.

Visualization 16: Relationship between Total Minimum Engineers and Total Minimum Conductors

```
viz5_data <- df_rail %>% select(`Hours Engineers On Duty`, `Minutes Engineers On Duty`, `Hours Conducto
viz5_data$totalMinEngineers = (60 * viz5_data$`Hours Engineers On Duty`) + viz5_data$`Minutes Engineers
viz5_data$totalMinConductors = (60 * viz5_data$`Hours Conductors On Duty`) + viz5_data$`Minutes Conduct

viz5_data <- na.omit(viz5_data)

engineers_iqr <- IQR(viz5_data$totalMinEngineers)
conductors_iqr <- IQR(viz5_data$totalMinConductors)
engineers_upper <- quantile(viz5_data$totalMinEngineers, 0.75) + 1.5 * engineers_iqr
engineers_lower <- quantile(viz5_data$totalMinEngineers, 0.25) - 1.5 * engineers_iqr
conductors_upper <- quantile(viz5_data$totalMinConductors, 0.75) + 1.5 * conductors_iqr
conductors_lower <- quantile(viz5_data$totalMinConductors, 0.25) - 1.5 * conductors_iqr

# filter the data to remove outliers
viz5_data <- viz5_data %>%
  filter(
    totalMinEngineers >= engineers_lower,
    totalMinEngineers <= engineers_upper,
    totalMinConductors >= conductors_lower,
    totalMinConductors <= conductors_upper
  )
```

```
# create the plot with the filtered data
ggplot(viz5_data, aes(x = totalMinEngineers, y = totalMinConductors)) +
  geom_point() +
  geom_smooth() +
  labs(x = "Total Minimum Engineers", y = "Total Minimum Conductors") +
  theme_classic()
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```



Observation: From the above visualization, we can conclude that as the number of minimum engineers increased, the number of minimum conductors also increases. There is a strong, positive, linear relationship between the two factors.

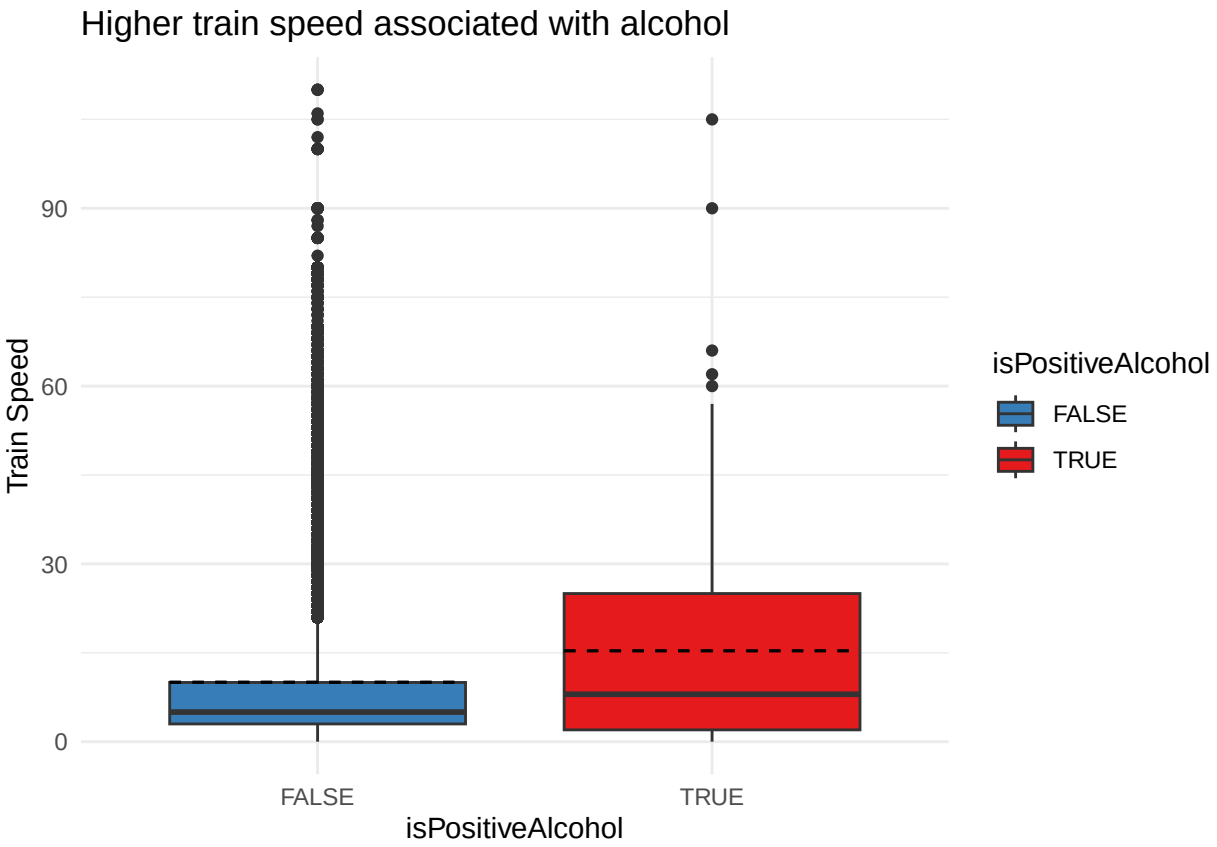
Visualization 17: Higher train speed associated with alcohol

```
viz6_data <- df_rail %>% select(`Train Speed`, `Positive Alcohol Tests`, `Positive Drug Tests`)

viz6_data$isPositiveAlcohol <-as.logical(viz6_data$`Positive Alcohol Tests`)
viz6_data$isPositiveDrug <-as.logical(viz6_data$`Positive Drug Tests`)

viz6_data <- na.omit(viz6_data)

ggplot(viz6_data, aes(x=isPositiveAlcohol, y=`Train Speed`, fill=isPositiveAlcohol)) +
geom_boxplot() +
scale_fill_brewer(palette="Set1", direction=-1) +
labs(y="Train Speed",
title="Higher train speed associated with alcohol") +
```

```
theme_minimal() +
  stat_summary(fun.y = mean, geom = "errorbar", aes(ymax = ..y.., ymin = ..y..),
               width = .75, linetype = "dashed",
               position = position_dodge2(width = 0.75))
```

```
## Warning: The `fun.y` argument of `stat_summary()` is deprecated as of ggplot2 3.3.0.
## i Please use the `fun` argument instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## Warning: The dot-dot notation (`..y..`) was deprecated in ggplot2 3.4.0.
## i Please use `after_stat(y)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

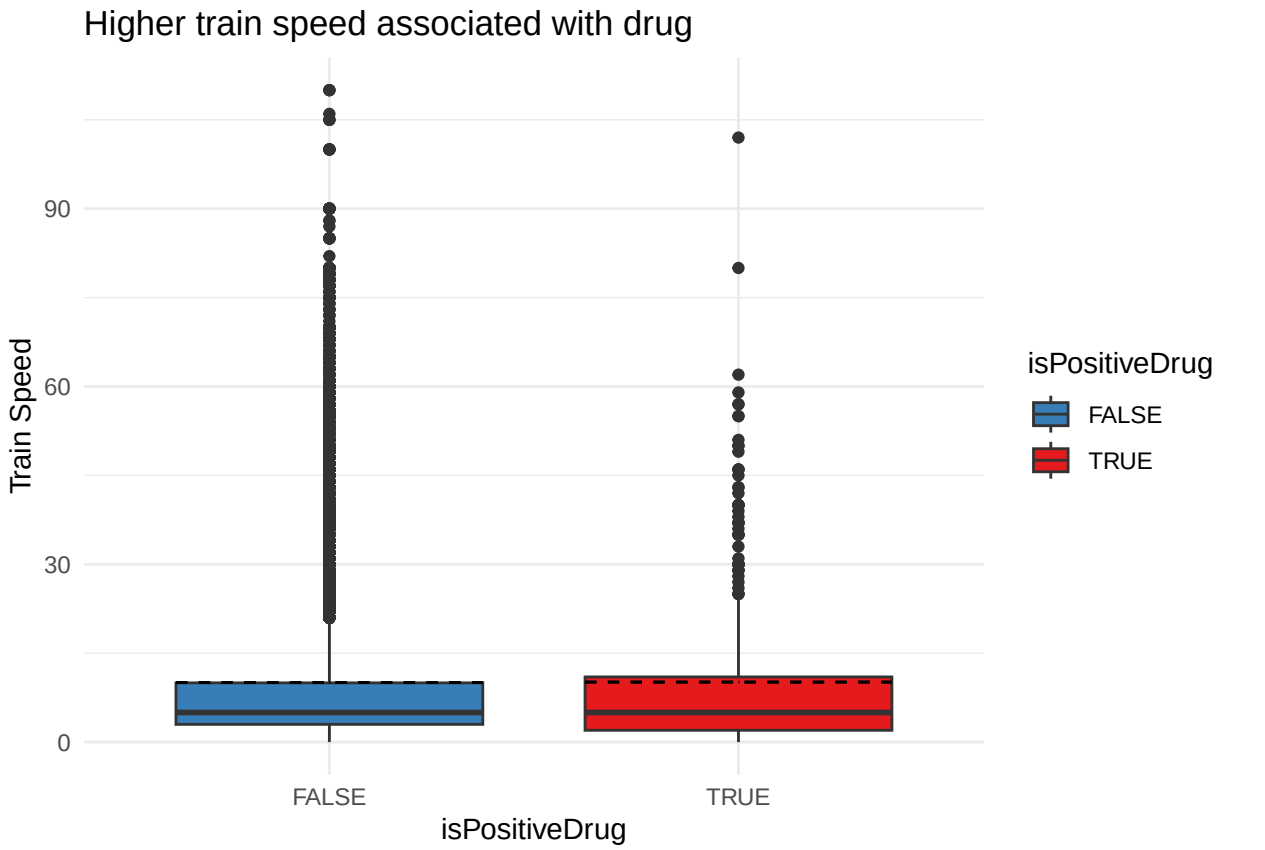


Observation: From the above visualization, we can conclude that the train speed range increases with the presence of a positive alcohol test. The median (solid line) and mean (dashed line) train speed is also higher with the presence of a positive alcohol test.

Visualization 18: Higher train speed associated with drug

```
ggplot(viz6_data, aes(x=isPositiveDrug, y=`Train Speed`, fill=isPositiveDrug)) +
geom_boxplot() +
scale_fill_brewer(palette="Set1", direction=-1) +
labs(y="Train Speed",
title="Higher train speed associated with drug") +
```

```
theme_minimal() +
  stat_summary(fun = mean, geom = "errorbar", aes(ymax = ..y.., ymin = ..y..),
               width = .75, linetype = "dashed",
               position = position_dodge2(width = 0.75))
```



Observation: From the above visualization, we can conclude that the train speed range increases with the presence of a positive drug test. The median (solid line) train speed, however, is around 5 regardless of the outcome of the drug test. The mean (dashed line) train speed also the same regardless of the outcome of the drug test.

##Regression Testing

```
set.seed(2)
df_part <- resample_partition(df_rail,p=c(train=0.5, test=0.5))
head(df_part)
```

```
## $train
## <resample [108,049 x 160]> 1, 2, 3, 4, 5, 6, 7, 8, 13, 16, ...
##
## $test
## <resample [108,051 x 160]> 9, 10, 11, 12, 14, 15, 18, 19, 20, 23, ...
```

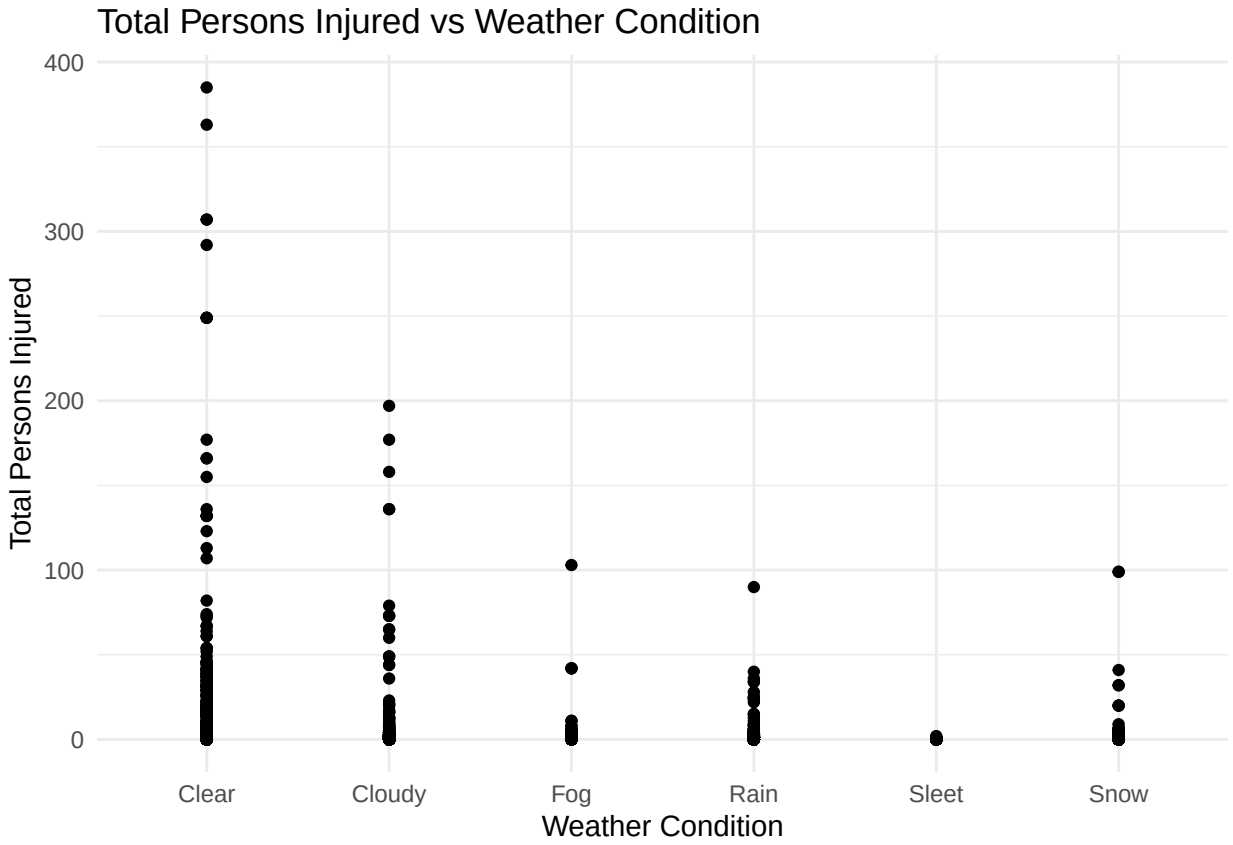
```
# Filtering out the required columns and omitting NA values:

df_rail_filtered <- df_rail %>%
  select(`Temperature`,`Weather Condition Code`, `Weather Condition`,`Visibility Code`,
              `Visibility`, `Train Speed`, `Positive Alcohol Tests`,
              `Positive Drug Tests`,`Engineers On Duty`,
         `Minutes Engineers On Duty`, `First Car Position`, `Accident Type`,
```

```
        `Total Persons Injured`) %>%
  na.omit()
```

Plotting to check the relationship between `Total Persons Injured` and candidate predictors

```
ggplot(df_rail_filtered, aes(x=(`Weather Condition`), y=(`Total Persons Injured`))) +
  geom_point() +
  geom_smooth() +
  geom_smooth(method="lm", color="red") +
  labs(x="Weather Condition", y="Total Persons Injured", title="Total Persons Injured vs Weather Conditi
  theme_minimal()
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using formula = 'y ~ x'
```

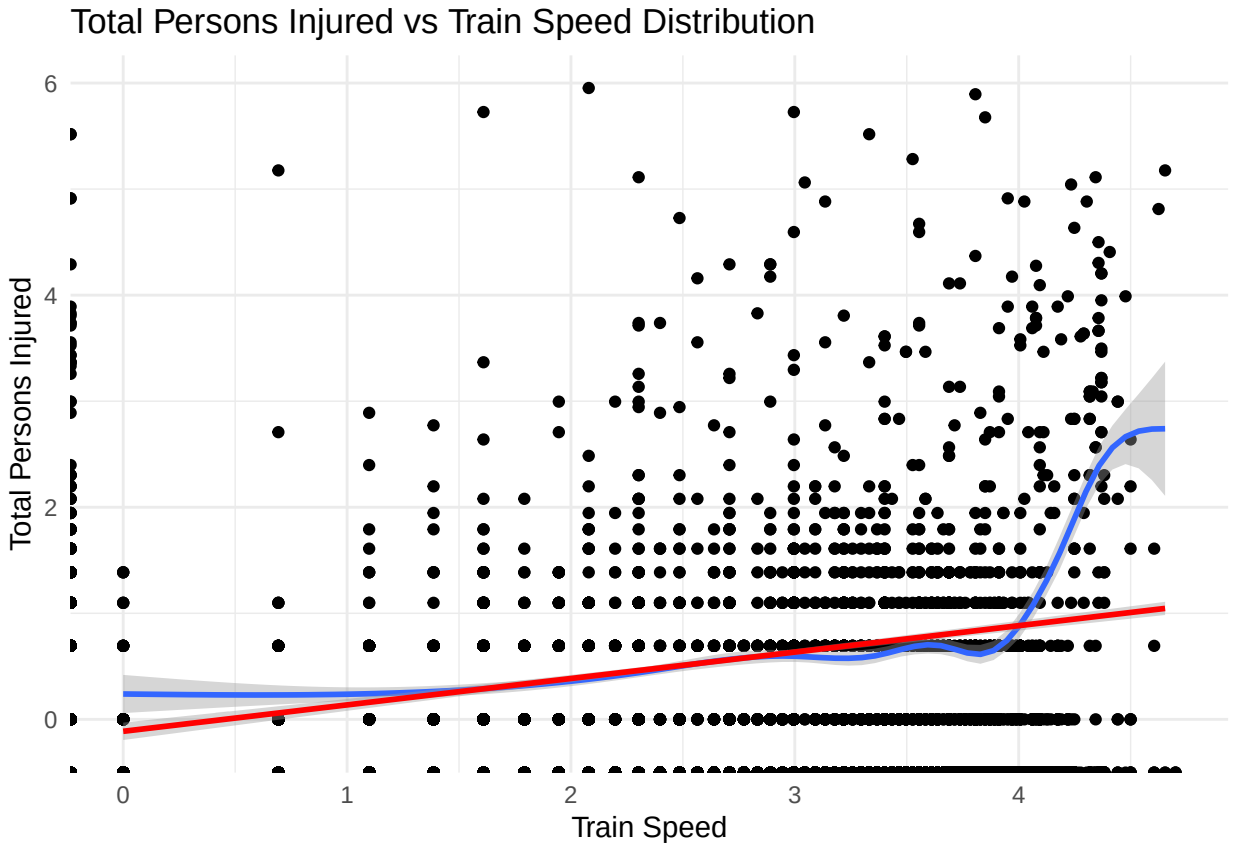



```
ggplot(df_rail_filtered, aes(x=(`Train Speed`), y=(`Total Persons Injured`))) +
  geom_point() +
  geom_smooth() +
  geom_smooth(method="lm", color="red") +
  labs(x="Train Speed", y="Total Persons Injured", title="Total Persons Injured vs Train Speed Distribu
  theme_minimal()
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using formula = 'y ~ x'
```

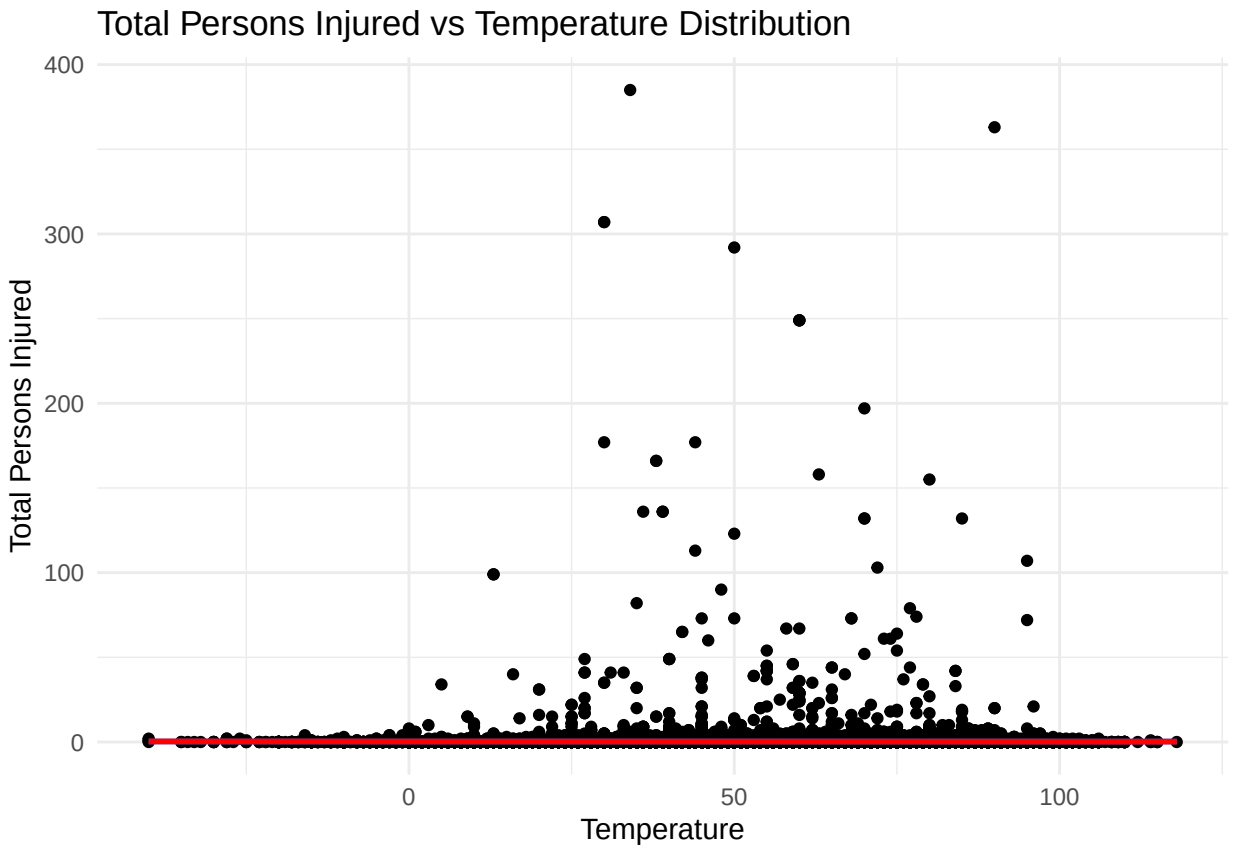

```
ggplot(df_rail_filtered, aes(x=log(`Train Speed`), y=log(`Total Persons Injured`))) +
  geom_point() +
  geom_smooth() +
  geom_smooth(method="lm", color="red") +
  labs(x="Train Speed", y="Total Persons Injured" , title="Total Persons Injured vs Train Speed Distribu
  theme_minimal()
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
## Warning: Removed 56363 rows containing non-finite values (`stat_smooth()`).
## `geom_smooth()` using formula = 'y ~ x'
## Warning: Removed 56363 rows containing non-finite values (`stat_smooth()`).
```



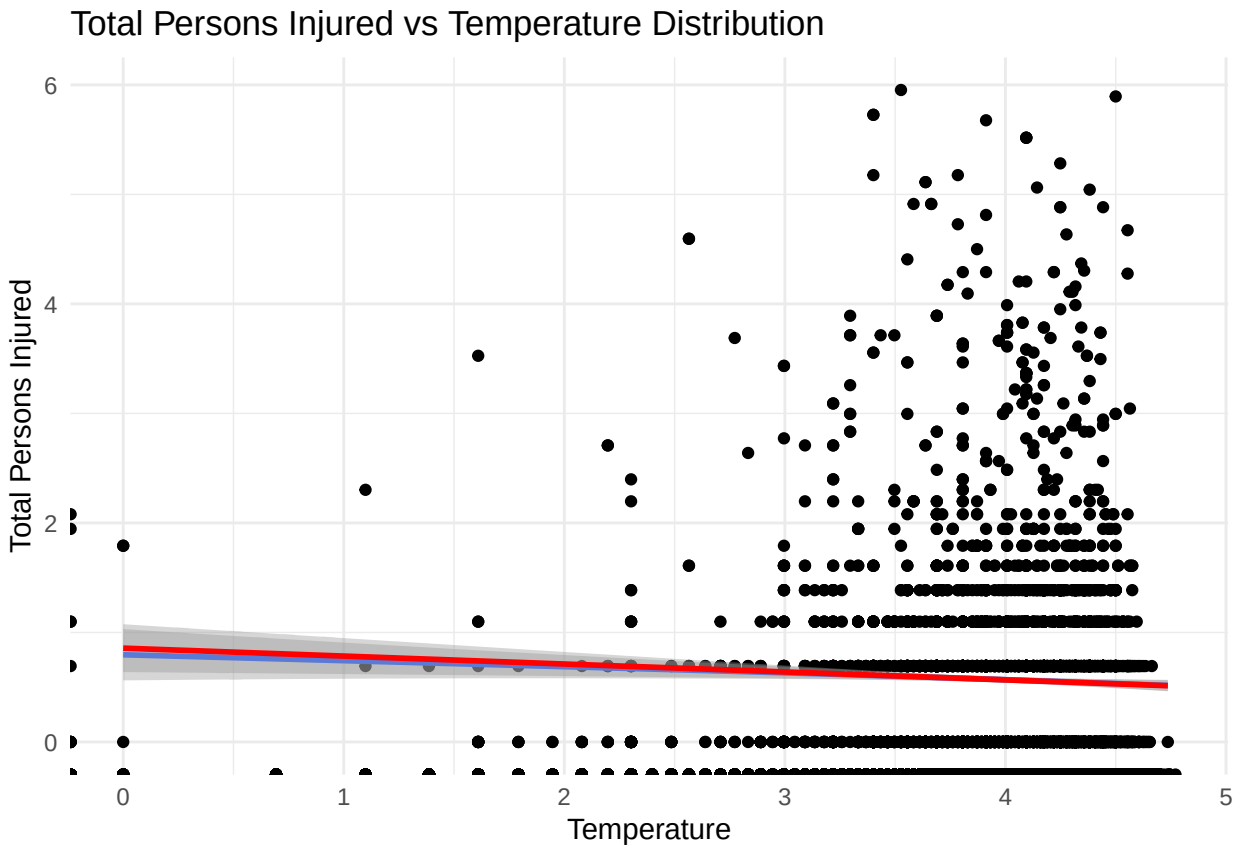

```
ggplot(df_rail_filtered, aes(x=(`Temperature`), y=(`Total Persons Injured`))) +
  geom_point() +
  geom_smooth() +
  geom_smooth(method="lm", color="red") +
  labs(x="Temperature", y="Total Persons Injured", title="Total Persons Injured vs Temperature Distribu
  theme_minimal()
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using formula = 'y ~ x'
```




```
ggplot(df_rail_filtered, aes(x=log(`Temperature`), y=log(`Total Persons Injured`))) +
  geom_point() +
  geom_smooth() +
  geom_smooth(method="lm", color="red") +
  labs(x="Temperature", y="Total Persons Injured", title="Total Persons Injured vs Temperature Distribu
  theme_minimal()
```

```
## Warning in log(Temperature): NaNs produced
## Warning in log(Temperature): NaNs produced
## Warning in log(Temperature): NaNs produced
## Warning in log(Temperature): NaNs produced
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
## Warning: Removed 55793 rows containing non-finite values (`stat_smooth()`).
## `geom_smooth()` using formula = 'y ~ x'
## Warning: Removed 55793 rows containing non-finite values (`stat_smooth()`).
## Warning: Removed 445 rows containing missing values (`geom_point()`).
```

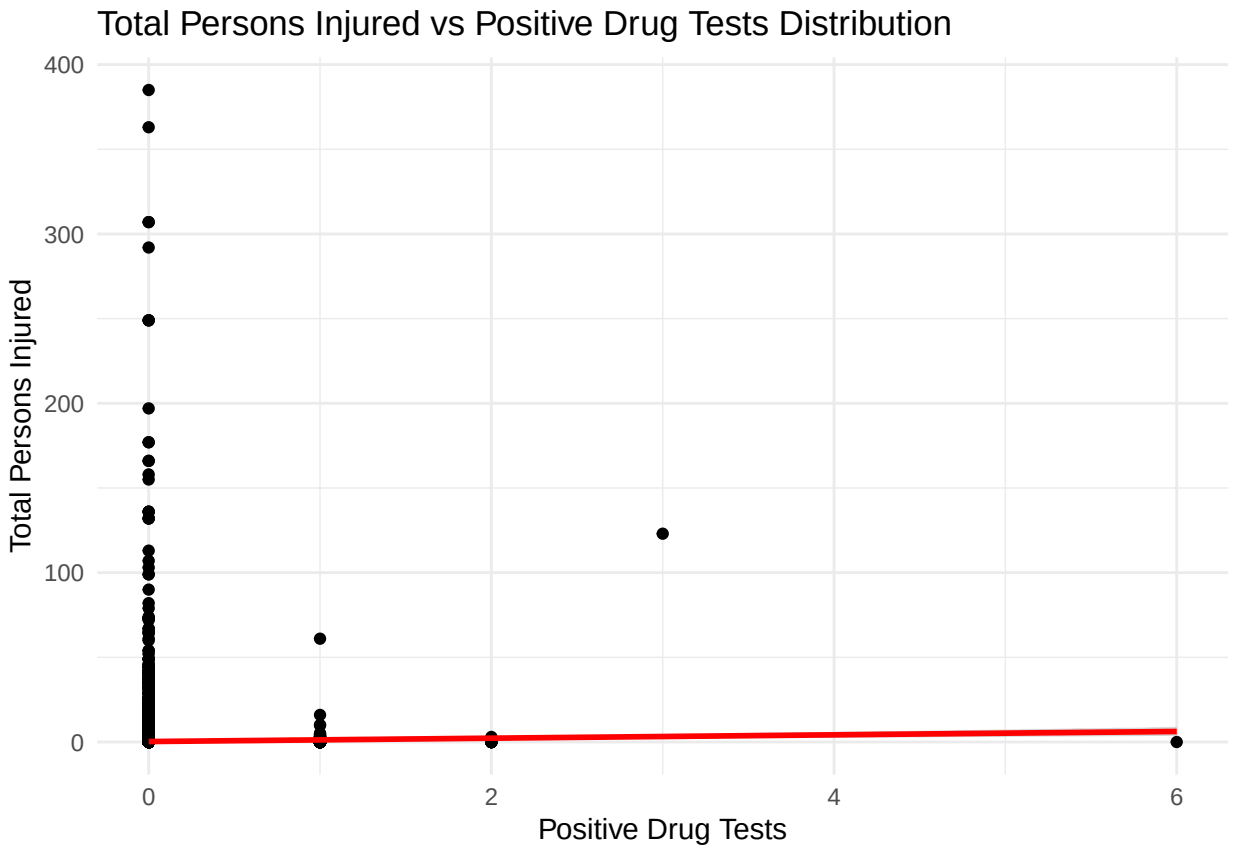



```
ggplot(df_rail_filtered, aes(x=(`Positive Drug Tests`), y=(`Total Persons Injured`))) +
  geom_point() +
  geom_smooth() +
  geom_smooth(method="lm", color="red") +
  labs(x="Positive Drug Tests", y="Total Persons Injured", title="Total Persons Injured vs Positive Drug
  theme_minimal()
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```

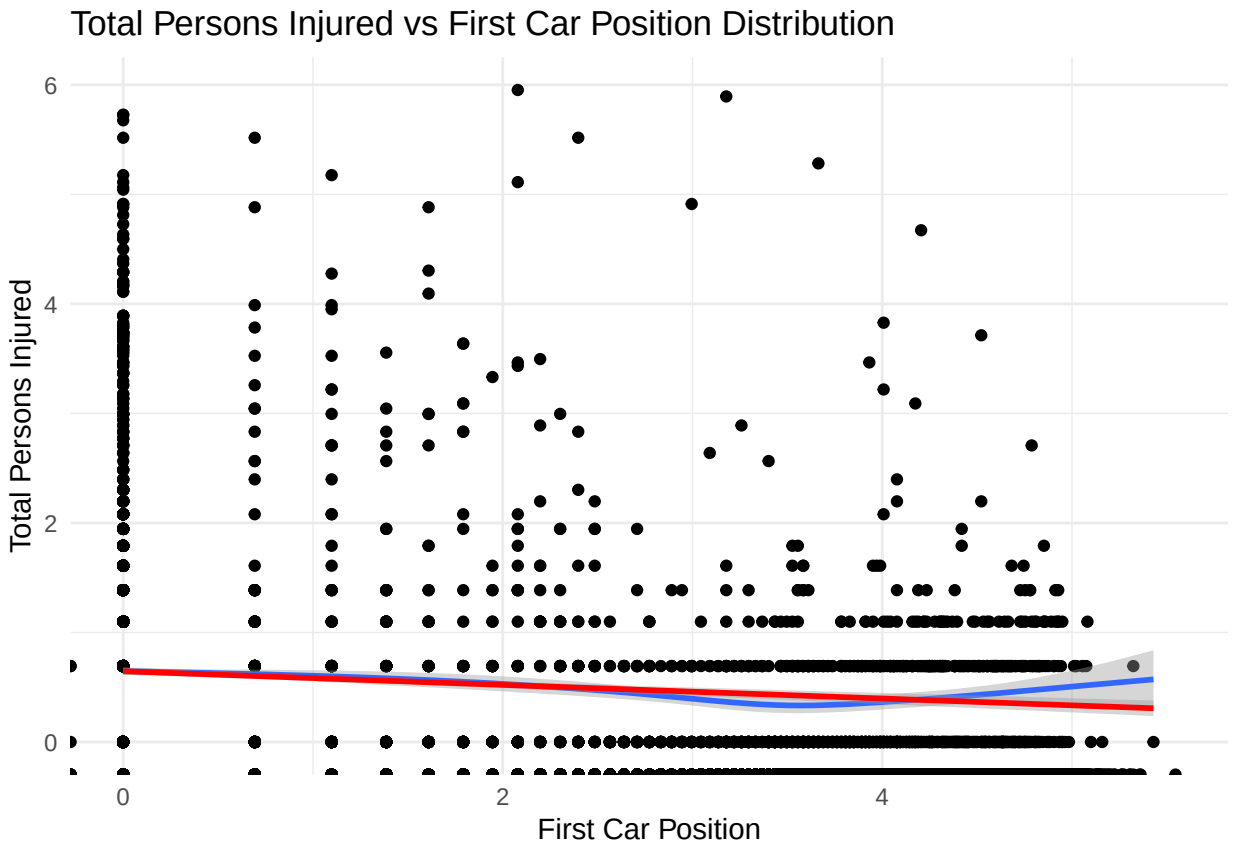
```
## Warning: Computation failed in `stat_smooth()`
## Caused by error in `smooth.construct.cr.smooth.spec()`:
## ! x has insufficient unique values to support 10 knots: reduce k.
```

```
## `geom_smooth()` using formula = 'y ~ x'
```




```
ggplot(df_rail_filtered, aes(x=log(`First Car Position`), y=log(`Total Persons Injured`))) +
  geom_point() +
  geom_smooth() +
  geom_smooth(method="lm", color="red") +
  labs(x="First Car Position", y="Total Persons Injured", title="Total Persons Injured vs First Car Posi
  theme_minimal()
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
## Warning: Removed 55754 rows containing non-finite values (`stat_smooth()`).
## `geom_smooth()` using formula = 'y ~ x'
## Warning: Removed 55754 rows containing non-finite values (`stat_smooth()`).
```



Fitting the linear regression models

```
fit_1 <- lm((`Total Persons Injured`) ~ (`Weather Condition Code`), data=df_part$train)
fit_2 <- lm((`Total Persons Injured`) ~ (`Train Speed`), data=df_part$train)
fit_3 <- lm((`Total Persons Injured`) ~ (`Temperature`), data=df_part$train)
fit_4 <- lm((`Total Persons Injured`) ~ (`Positive Drug Tests`), data=df_part$train)
fit_5 <- lm((`Total Persons Injured`) ~ (`Minutes Engineers On Duty`), data=df_part$train)
fit_6 <- lm((`Total Persons Injured`) ~ (`First Car Position`), data=df_part$train)
```

#rsme values were high

```
rmse(fit_1, df_part$test)
```

```
## [1] 4.163183
```

```
rmse(fit_2, df_part$test)
```

```
## [1] 4.153368
```

```
rmse(fit_3, df_part$test)
```

```
## [1] 4.159247
```

```
rmse(fit_4, df_part$test)
```

```
## [1] 5.052945
```

```
rmse(fit_5, df_part$test)
```

```
## [1] 4.433511
```

```
rmse(fit_6, df_part$test)
```

```
## [1] 4.256498
```

Observation: As observed from the above graphs, there is not a strong linear relationship between the `Total Persons Injured` and candidate predictors so we will try other models.

Logistic Regression

```
# Transforming the `Total Persons Injured` to a binary isInjured column:

df_rail_filtered$isInjured <-ifelse(df_rail_filtered$`Total Persons Injured` > 0, "Injured", "No Injury
head(df_rail_filtered)
```

```
## # A tibble: 6 x 14
##   Temperature Weather ~1 Weath~2 Visib~3 Visib~4 Train~5 Posit~6 Posit~7 Engin~8
##         <dbl>      <dbl> <chr>     <dbl> <chr>     <dbl>   <dbl>   <dbl>   <dbl>
## 1          60          1 Clear         1 Dawn          6       0       0       1
## 2          60          1 Clear         1 Dawn          4       0       0       1
## 3          30          1 Clear         2 Day           3       0       0       1
## 4          26          1 Clear         2 Day           5       0       0       1
## 5          72          3 Rain          4 Dark         10       0       0       1
## 6          42          1 Clear         3 Dusk          8       0       0       1
## # ... with 5 more variables: `Minutes Engineers On Duty` <dbl>,
## #   `First Car Position` <dbl>, `Accident Type` <chr>,
## #   `Total Persons Injured` <dbl>, isInjured <chr>, and abbreviated variable
## #   names 1: `Weather Condition Code`, 2: `Weather Condition`,
## #   3: `Visibility Code`, 4: Visibility, 5: `Train Speed`,
## #   6: `Positive Alcohol Tests`, 7: `Positive Drug Tests`,
## #   8: `Engineers On Duty`
```

```
# Checking the classes and converting isInjured as factor:

df_rail_filtered$isInjured <- as.factor(df_rail_filtered$isInjured)
class(df_rail_filtered$isInjured)
```

```
## [1] "factor"
```

```
class(df_rail_filtered$`Train Speed`)
```

```
## [1] "numeric"
```

```
# Creating test and train partitions:

set.seed(1)
train <- createDataPartition(df_rail_filtered$isInjured, p=0.6, list=FALSE)

table(df_rail_filtered$isInjured[train])
```

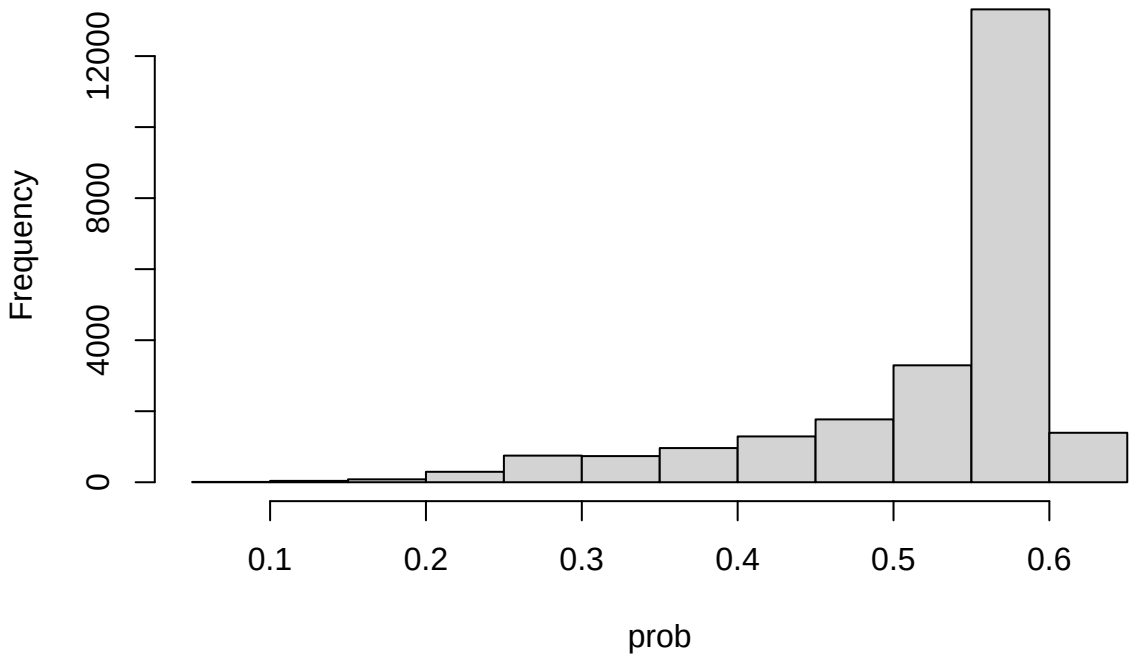
```
##
##   Injured No Injury
##      2453     33451
```

```
df_train <- df_rail_filtered[as.integer(train),]
df_test <- df_rail_filtered[-as.integer(train),]
```

```r
# Down sampling:

undersampled_data <- downSample(df_train, y = df_train$isInjured)

# Fitting the model:

fit <- glm(isInjured ~ `Train Speed`, data=undersampled_data, family=binomial(link="logit"))

summary(fit)
```

```
##
## Call:
## glm(formula = isInjured ~ `Train Speed`, family = binomial(link = "logit"),
##     data = undersampled_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.3730  -1.2144   0.3183   1.0660   2.0480
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)    0.448941   0.040447   11.10   <2e-16 ***
## `Train Speed` -0.030190   0.001977  -15.27   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6801.2  on 4905  degrees of freedom
## Residual deviance: 6536.5  on 4904  degrees of freedom
## AIC: 6540.5
##
## Number of Fisher Scoring iterations: 4
```

```r
# Calculating the predicted probabilities and plotting histogram:

prob <- predict(fit, newdata=df_test, type="response")

pred <- ifelse(prob > 0.5, "pos", "neg")

hist(prob)
```




```
mean(pred == df_test$isInjured, na.rm=TRUE)
```

```
## [1] 0
```

```
table(pred, df_test$isInjured)
```

```
##
## pred  Injured No Injury
##   neg     747      5186
##   pos     888     17114
```

```
# Functions for calculating sesnstivity and specificity:

sens <- function(c, p, ref, positive = levels(ref)[2])
{
  mean((p > c)[ref == positive], na.rm=TRUE)
}

sens(0.5, prob, df_test$isInjured)
```

```
## [1] 0.7674439
```

```
spec <- function(c, p, ref, negative = levels(ref)[1])
{
  mean((p < c)[ref == negative], na.rm=TRUE)
}

spec(0.5, prob, df_test$isInjured)
```

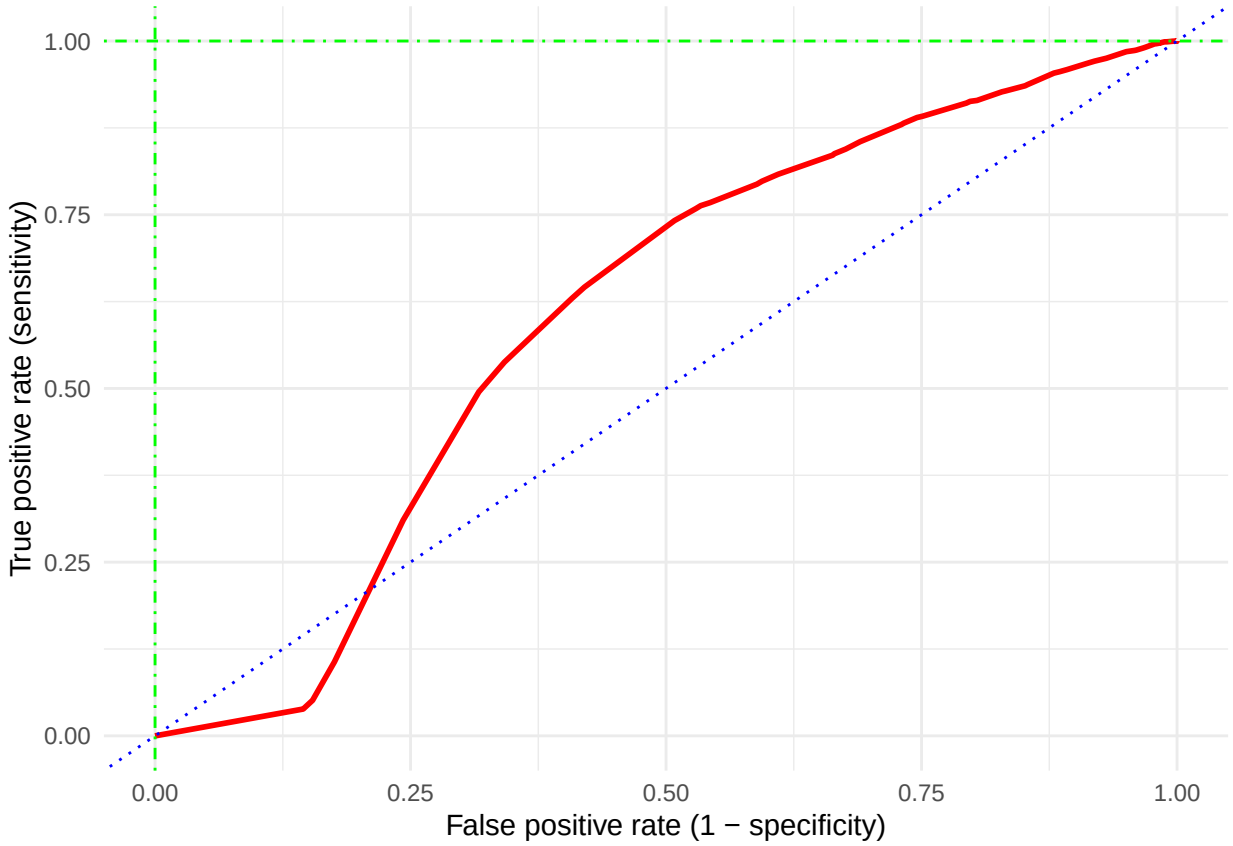
```
## [1] 0.4568807
```

```r
# Plotting the ROC curve:

roc <- tibble(p=seq(from=0, to=1, by=0.01)) %>%
  mutate(sensitivity = map_dbl(p, sens, p=prob, ref=df_test$isInjured),
         specificity = map_dbl(p, spec, p=prob, ref=df_test$isInjured),
         TPR=sensitivity,
         FPR=1 - specificity)
roc
```

```
## # A tibble: 101 x 5
##        p sensitivity specificity   TPR   FPR
##    <dbl>       <dbl>       <dbl> <dbl> <dbl>
##  1  0           1       0         1    1
##  2  0.01        1       0         1    1
##  3  0.02        1       0         1    1
##  4  0.03        1       0         1    1
##  5  0.04        1       0         1    1
##  6  0.05        1       0         1    1
##  7  0.06        1.00    0         1.00 1
##  8  0.07        1.00    0.000612  1.00 0.999
##  9  0.08        1.00    0.000612  1.00 0.999
## 10  0.09        1.00    0.000612  1.00 0.999
## # ... with 91 more rows
```

```r
ggplot(roc, aes(x=FPR, y=TPR)) +
  geom_path(color="red", size=1) +
  geom_vline(xintercept=0, color="green", linetype="dotdash") +
  geom_hline(yintercept=1, color="green", linetype="dotdash") +
  geom_abline(intercept=0, slope=1, color="blue", linetype="dotted") +
  labs(x="False positive rate (1 - specificity)",
       y="True positive rate (sensitivity)") +
  theme_minimal()
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```




```
# Calculating the AUC value:

sum(roc$TPR[-1] * abs(diff(roc$FPR))) # AUC
```

```
## [1] 0.5756423
```

```
# Fitting the model using a different function, including down sampling:

fit2 <- train(isInjured ~ `Train Speed`, data=df_train,
              method="glm", family=binomial(link="logit"),
              preProcess="medianImpute",
              trControl=trainControl(method="none",
                                     sampling="down"),
              na.action=na.pass)

confusionMatrix(predict(fit2, df_test, na.action=na.pass),
                df_test$isInjured)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Injured No Injury
##   Injured       747      5186
##   No Injury     888     17114
##
##                Accuracy : 0.7462
##                  95% CI : (0.7407, 0.7517)
##     No Information Rate : 0.9317
```

```
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1011
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.45688
##             Specificity : 0.76744
##          Pos Pred Value : 0.12591
##          Neg Pred Value : 0.95067
##              Prevalence : 0.06831
##          Detection Rate : 0.03121
##    Detection Prevalence : 0.24788
##       Balanced Accuracy : 0.61216
##
##        'Positive' Class : Injured
##
```

```
# Testing other models for comparison:
fit3 <- train(isInjured ~ `Temperature`, data=df_train,
              method="glm", family=binomial(link="logit"),
              preProcess="medianImpute",
              trControl=trainControl(method="none",
                                     sampling="down"),
              na.action=na.pass)

confusionMatrix(predict(fit3, df_test, na.action=na.pass),
                df_test$isInjured)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Injured No Injury
##   Injured       854     11354
##   No Injury     781     10946
##
##                Accuracy : 0.493
##                  95% CI : (0.4866, 0.4994)
##     No Information Rate : 0.9317
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0033
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.52232
##             Specificity : 0.49085
##          Pos Pred Value : 0.06995
##          Neg Pred Value : 0.93340
##              Prevalence : 0.06831
##          Detection Rate : 0.03568
##    Detection Prevalence : 0.51005
##       Balanced Accuracy : 0.50659
##
##        'Positive' Class : Injured
```

##