Historical Analysis of Airline Safety Dataset

Frederick Jones

Data Preparation

```
library(tidyverse)
```

Motivation

1. Motivation: The motivation behind this analysis goes beyond simple counts of accidents and incidents. We aim to delve into the rates of accidents, such as accidents per 100,000 flight miles. This metric allows us to account for changes in flight activity over time. For instance, if the number of miles flown doubles, but the accident count only increases by 20%, it could indicate an improvement in flight safety.

This approach aligns with industry best practices and allows for a more nuanced evaluation of safety measures, providing valuable insights for both the aviation industry and the broader public. Our goal is not only to identify patterns but also to contribute to the ongoing efforts to enhance aviation safety and inform decision-making in the industry.

Cases

That that dataset has 11521 cases,each providing detailed information about the trafficking of enslaved.

Within this dataset, there are 126 variables, offering a comprehensive scope for analysis.

Data collection

The dataset is hosted on kaggle so we are using from kaggle

Type of study

Additionally, we will explore various metrics beyond accident counts, including rates of fatalities and potentially other criteria. By examining these multiple dimensions, we can gain a more holistic understanding of airline safety and identify trends that might be obscured by raw counts alone.

Data Source

The data set has been made available by kaggle at the following link

Dataset Descriptions

airline ————————> Airline (asterisk indicates that regional subsidiaries are included) avail_seat_km_per_week —-> Available seat kilometers flown every week incidents_85_99 ————>Total number of incidents, 1985–1999 fatal_accidents_85_99——> Total number of fatal accidents, 1985–1999 fatalities_85_99 ——–>Total number of fatalities, 1985–1999 incidents_00_14 ———-> Total number of incidents, 2000–2014 fatal_accidents_00_14——> Total number of fatal accidents, 2000–2014 fatalities_00_14———> Total number of fatalities, 2000–2014

Relevant summary statistics

```
# Load necessary libraries
library(readr)

# Load the dataset
dataset  <- read_csv("airline-safety_csv.csv")
```

Import Dataset

```
## Rows: 56 Columns: 8
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): airline
## dbl (7): avail_seat_km_per_week, incidents_85_99, fatal_accidents_85_99, fat...
##
## 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(dataset)
```

```
## # A tibble: 6 x 8
##   airline           avail_seat_km_per_week incidents_85_99 fatal_accidents_85_99
##   <chr>                              <dbl>           <dbl>                  <dbl>
## 1 aer lingus                     320906734               2                      0
## 2 aeroflot*                     1197672318              76                     14
## 3 aerolineas argen~              385803648               6                      0
## 4 aeromexico*                    596871813               3                      1
## 5 air canada                    1865253802               2                      0
## 6 air france                    3004002661              14                      4
## # i 4 more variables: fatalities_85_99 <dbl>, incidents_00_14 <dbl>,
## #   fatal_accidents_00_14 <dbl>, fatalities_00_14 <dbl>
```

```
# Summary statistics
summary(dataset)
```

```
##    airline          avail_seat_km_per_week incidents_85_99
##  Length:56          Min.   :2.594e+08      Min.   : 0.000
##  Class :character   1st Qu.:4.740e+08      1st Qu.: 2.000
##  Mode  :character   Median :8.029e+08      Median : 4.000
```

```
##                     Mean   :1.385e+09      Mean   : 7.179
##                     3rd Qu.:1.847e+09      3rd Qu.: 8.000
##                     Max.   :7.139e+09      Max.   :76.000
##  fatal_accidents_85_99 fatalities_85_99 incidents_00_14  fatal_accidents_00_14
##  Min.   : 0.000        Min.   :  0.0    Min.   : 0.000   Min.   :0.0000
##  1st Qu.: 0.000        1st Qu.:  0.0    1st Qu.: 1.000   1st Qu.:0.0000
##  Median : 1.000        Median : 48.5    Median : 3.000   Median :0.0000
##  Mean   : 2.179        Mean   :112.4    Mean   : 4.125   Mean   :0.6607
##  3rd Qu.: 3.000        3rd Qu.:184.2    3rd Qu.: 5.250   3rd Qu.:1.0000
##  Max.   :14.000        Max.   :535.0    Max.   :24.000   Max.   :3.0000
##  fatalities_00_14
##  Min.   :  0.00
##  1st Qu.:  0.00
##  Median :  0.00
##  Mean   : 55.52
##  3rd Qu.: 83.25
##  Max.   :537.00
```

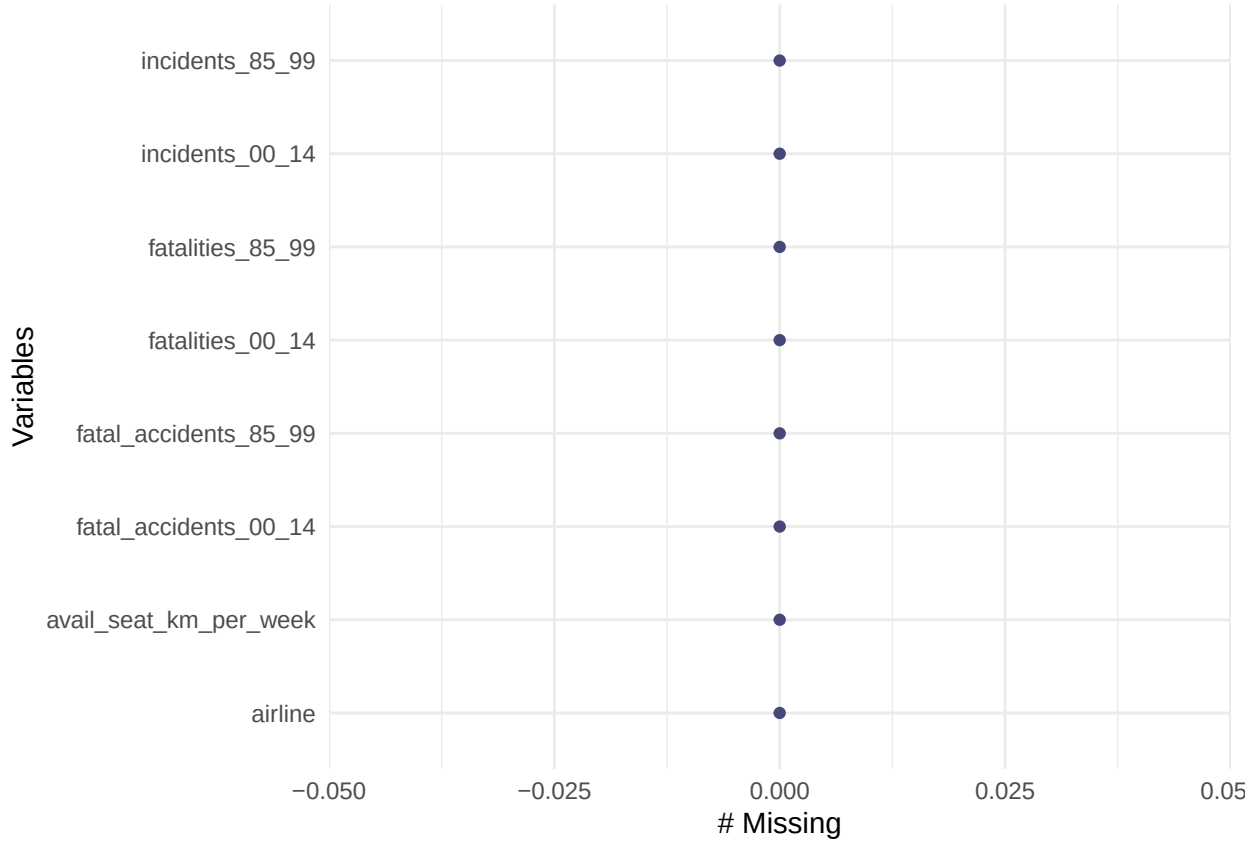
```
# Checking missing values
sapply(dataset, function(x) sum(is.na(x)))
```

```
##                airline avail_seat_km_per_week        incidents_85_99
##                      0                       0                      0
##  fatal_accidents_85_99       fatalities_85_99        incidents_00_14
##                      0                       0                      0
##  fatal_accidents_00_14       fatalities_00_14
##                      0                       0
```

```
# Visualizing missing values

library(naniar)
```

```
## Warning: package 'naniar' was built under R version 4.3.2
```

```
gg_miss_var(dataset)
```




```
# Assuming your dataset is stored in the 'dataset' variable

# Check for missing values
missing_values <- sapply(dataset, function(x) sum(is.na(x)))

# Identify columns with missing values
columns_with_missing <- names(which(missing_values > 0))

# Impute missing values using mean for numeric columns
for (col in columns_with_missing) {
  if (is.numeric(dataset[[col]])) {
    mean_value <- mean(dataset[[col]], na.rm = TRUE)
    dataset[[col]][is.na(dataset[[col]])] <- mean_value
  } else {
    # If it's a non-numeric column, you might use another strategy like imputing with the most frequent
    most_frequent_value <- names(sort(table(dataset[[col]], decreasing = TRUE)))[1]
    dataset[[col]][is.na(dataset[[col]])] <- most_frequent_value
  }
}

# Verify that missing values are filled
sapply(dataset, function(x) sum(is.na(x)))
```

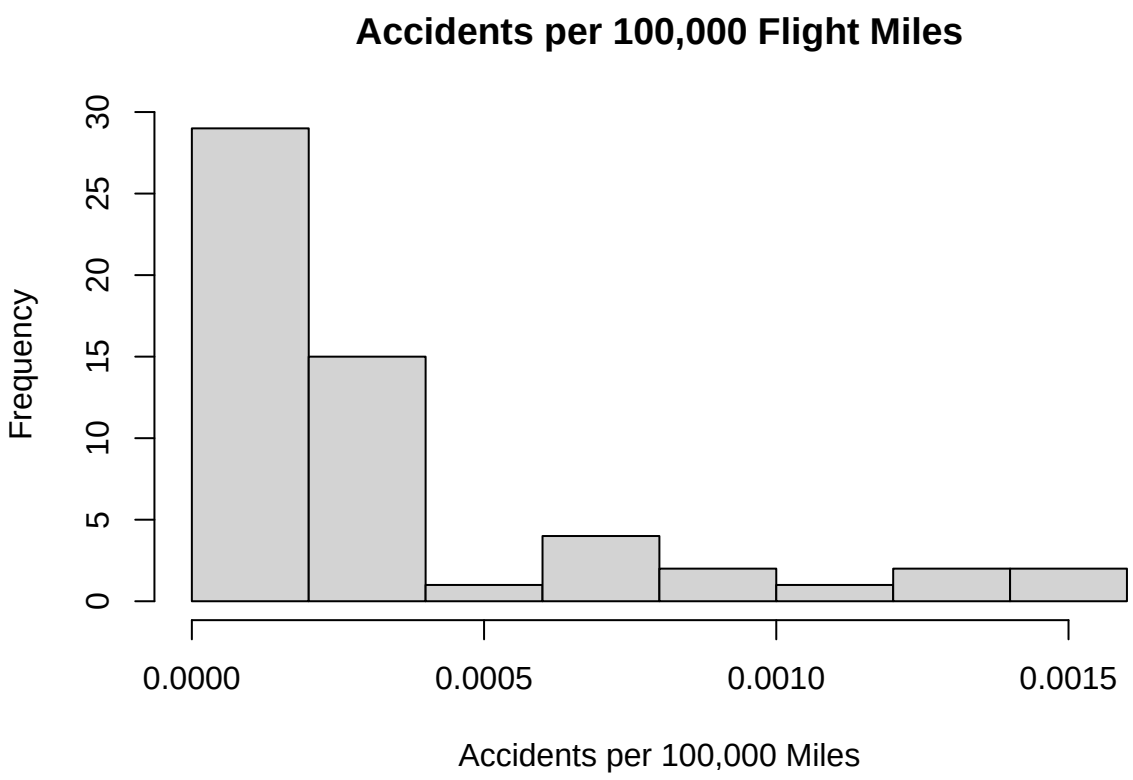
```
##               airline avail_seat_km_per_week        incidents_85_99
##                     0                       0                      0
```

```
##  fatal_accidents_85_99       fatalities_85_99        incidents_00_14
##                      0                      0                      0
##  fatal_accidents_00_14       fatalities_00_14
##                      0                      0
```

```r
head(dataset)
```

```
## # A tibble: 6 x 8
##   airline           avail_seat_km_per_week incidents_85_99 fatal_accidents_85_99
##   <chr>                              <dbl>           <dbl>                  <dbl>
## 1 aer lingus                     320906734               2                      0
## 2 aeroflot*                     1197672318              76                     14
## 3 aerolineas argen~              385803648               6                      0
## 4 aeromexico*                    596871813               3                      1
## 5 air canada                    1865253802               2                      0
## 6 air france                    3004002661              14                      4
## # i 4 more variables: fatalities_85_99 <dbl>, incidents_00_14 <dbl>,
## #   fatal_accidents_00_14 <dbl>, fatalities_00_14 <dbl>
```

```r
# Calculate accidents per 100,000 flight miles
dataset$accidents_per_100k_miles <-
  (dataset$fatal_accidents_85_99 + dataset$fatal_accidents_00_14) /
  (dataset$avail_seat_km_per_week / 100000)

# Visualize accidents per 100,000 flight miles
hist(dataset$accidents_per_100k_miles,
     main = "Accidents per 100,000 Flight Miles",
     xlab = "Accidents per 100,000 Miles")
```




```
# Calculate a new metric, e.g., fatalities per incident
dataset$fatalities_per_incident <-
  (dataset$fatalities_85_99 + dataset$fatalities_00_14) /
  (dataset$incidents_85_99 + dataset$incidents_00_14)
```

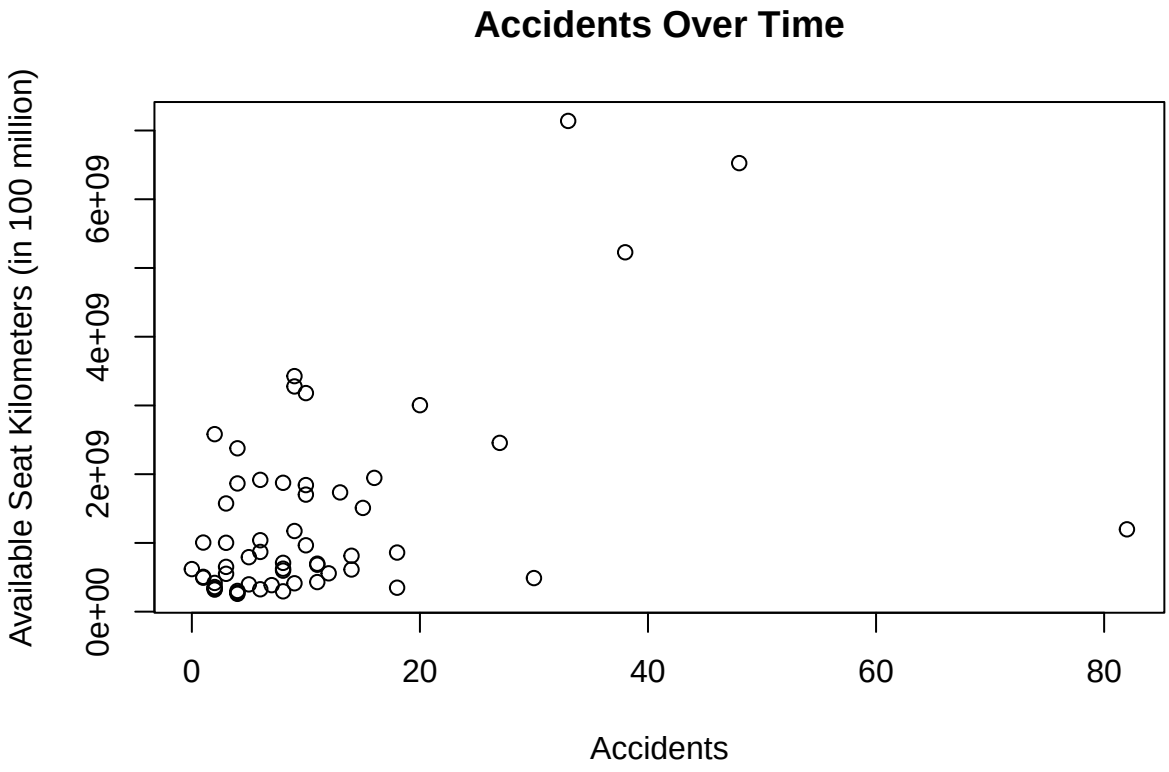
```
# Perform statistical analysis on the new metric
t_test_result <- t.test(dataset$fatalities_per_incident)
print(t_test_result)
```

```
##
##  One Sample t-test
##
## data:  dataset$fatalities_per_incident
## t = 4.744, df = 54, p-value = 1.575e-05
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  11.25900 27.74091
## sample estimates:
## mean of x
##  19.49996
```

Code	Local
21302	Charleston

Code	Local
37020	St. Thomas
37010	St. Croix
36499	Saint-Domingue, port unspecified
32240	Suriname
50299	Bahia, port unspecified
39001	Caribbean (colony unspecified)

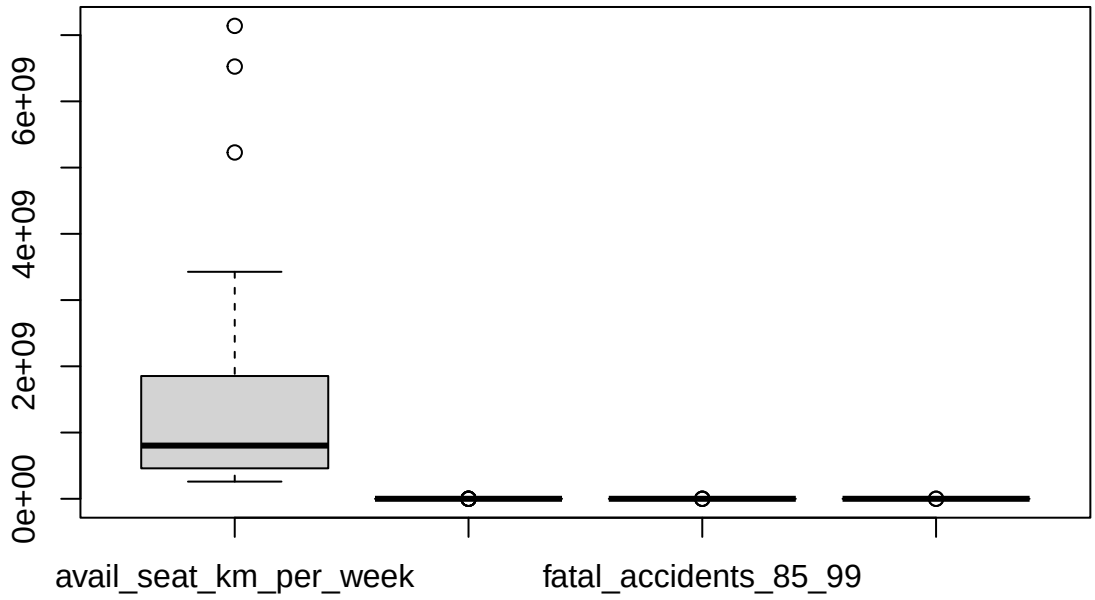
```r
# Create a scatter plot of accidents over time
plot(dataset$incidents_85_99 + dataset$incidents_00_14,
     dataset$avail_seat_km_per_week,
     main = "Accidents Over Time",
     xlab = "Accidents",
     ylab = "Available Seat Kilometers (in 100 million)")
```



Code	Local
31312	Havana
32150	St. Eustatius
41203	Veracruz
21102	Hampton
41207	Cartagena
32110	Curaçao
33899	Dominica, port unspecified

Code	Local
31201	San Juan
31323	Santiago de Cuba

```
# Boxplot for selected columns
boxplot(dataset[, c("avail_seat_km_per_week", "incidents_85_99", "fatal_accidents_85_99", "fatalities_85
```




```
# Identify outliers using z-scores
outliers <- as.data.frame(boxplot.stats(dataset$avail_seat_km_per_week)$out)
outliers
```

```
##   boxplot.stats(dataset$avail_seat_km_per_week)$out
## 1                                        5228357340
## 2                                        6525658894
## 3                                        7139291291
```

```
# Assuming your dataset is stored in the 'dataset' variable

# Define lower and upper percentiles
lower_percentile <- 0.05
upper_percentile <- 0.95

# Identify numeric columns for outlier removal
```

```r
numeric_columns <- sapply(dataset, is.numeric)

# Loop through numeric columns and remove outliers
for (col in names(numeric_columns)[numeric_columns]) {
  lower_limit <- quantile(dataset[[col]], lower_percentile, na.rm = TRUE)
  upper_limit <- quantile(dataset[[col]], upper_percentile, na.rm = TRUE)

  # Remove outliers
  dataset[[col]] <- ifelse(dataset[[col]] < lower_limit, NA,
                           ifelse(dataset[[col]] > upper_limit, NA, dataset[[col]]))
}

# Verify that outliers are removed
summary(dataset)
```

```
##    airline          avail_seat_km_per_week incidents_85_99
##  Length:56          Min.   :3.014e+08      Min.   : 1.00
##  Class :character   1st Qu.:4.970e+08      1st Qu.: 2.00
##  Mode  :character   Median :8.029e+08      Median : 4.00
##                     Mean   :1.156e+09      Mean   : 5.54
##                     3rd Qu.:1.727e+09      3rd Qu.: 7.75
##                     Max.   :3.427e+09      Max.   :21.00
##                     NA's   :6              NA's   :6
##  fatal_accidents_85_99 fatalities_85_99 incidents_00_14  fatal_accidents_00_14
##  Min.   :0.00          Min.   :  0.00   Min.   : 0.000   Min.   :0.0000
##  1st Qu.:0.00          1st Qu.:  0.00   1st Qu.: 1.000   1st Qu.:0.0000
##  Median :1.00          Median : 34.00   Median : 3.000   Median :0.0000
##  Mean   :1.66          Mean   : 90.85   Mean   : 3.321   Mean   :0.6182
##  3rd Qu.:3.00          3rd Qu.:159.00   3rd Qu.: 5.000   3rd Qu.:1.0000
##  Max.   :7.00          Max.   :407.00   Max.   :11.000   Max.   :2.0000
##  NA's   :3             NA's   :3        NA's   :3        NA's   :1
##  fatalities_00_14 accidents_per_100k_miles fatalities_per_incident
##  Min.   :  0.00   Min.   :0.0000000        Min.   : 0.0000
##  1st Qu.:  0.00   1st Qu.:0.0000533        1st Qu.: 0.3368
##  Median :  0.00   Median :0.0001817        Median : 8.4833
##  Mean   : 34.32   Mean   :0.0002579        Mean   :13.9006
##  3rd Qu.: 46.00   3rd Qu.:0.0002981        3rd Qu.:19.3357
##  Max.   :283.00   Max.   :0.0012106        Max.   :70.7500
##  NA's   :3        NA's   :3                NA's   :4
```

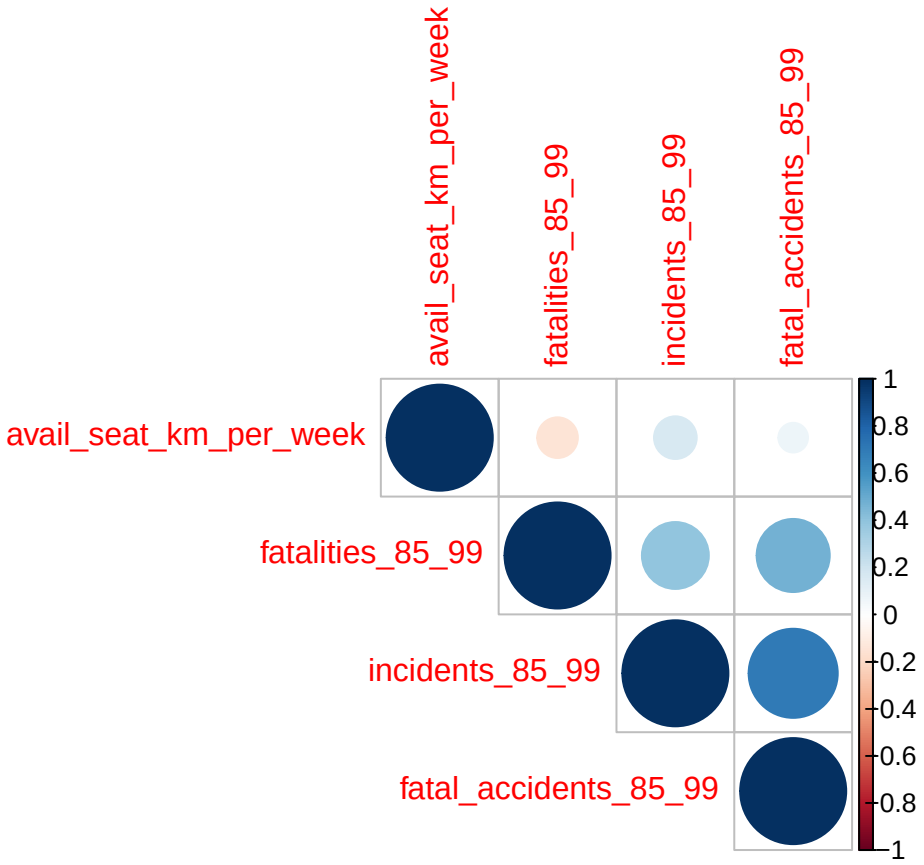
```r
# Assuming you want to impute missing values with the mean for numeric columns
library(dplyr)

dataset <- dataset %>%
  mutate(across(where(is.numeric), ~ifelse(is.na(.), mean(., na.rm = TRUE), .)))
# Assuming you want to remove rows with any missing values
dataset <- na.omit(dataset)

# Correlation matrix
cor_matrix <- cor(dataset[, c("avail_seat_km_per_week", "incidents_85_99", "fatal_accidents_85_99", "fa

# Visualization of correlation matrix
```

```
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 4.3.2
```

```
## corrplot 0.92 loaded
```

```
corrplot(cor_matrix, method = "circle", type = "upper", order = "hclust")
```




```
# Min-Max Scaling
min_max_scaling <- function(x) {
  (x - min(x)) / (max(x) - min(x))
}

# Apply Min-Max Scaling to numeric columns
dataset <- as.data.frame(lapply(dataset[, -1], min_max_scaling))

# Add back the 'airline' column
dataset$airline <- dataset$airline

# Display the normalized data
print(dataset)
```

```
##    avail_seat_km_per_week incidents_85_99 fatal_accidents_85_99
```

```
## 1           0.006248332         0.050           0.0000000
## 2           0.286799875         0.227           0.2371968
## 3           0.027014349         0.250           0.0000000
## 4           0.094552926         0.100           0.1428571
## 5           0.500415714         0.050           0.0000000
## 6           0.864797889         0.650           0.5714286
## 7           0.181710906         0.050           0.1428571
## 8           0.130808150         0.100           0.0000000
## 9           0.212459263         0.200           0.0000000
## 10          0.126916394         0.300           0.2857143
## 11          0.492729802         0.100           0.1428571
## 12          0.273548518         1.000           0.7142857
## 13          0.018194348         0.000           0.0000000
## 14          0.030572231         0.200           0.4285714
## 15          0.921037847         0.150           0.0000000
## 16          0.729910478         0.227           0.0000000
## 17          0.163779904         0.550           0.8571429
## 18          0.037311124         0.050           0.1428571
## 19          0.079711939         0.100           0.1428571
## 20          0.273548518         0.227           0.2371968
## 21          0.082018511         0.350           0.4285714
## 22          0.010901321         0.000           0.1428571
## 23          0.059895012         0.227           0.7142857
## 24          0.065624116         0.000           0.0000000
## 25          0.099827825         0.450           0.4285714
## 26          0.000000000         0.000           0.0000000
## 27          0.061596419         0.227           0.0000000
## 28          0.278970109         0.150           0.1428571
## 29          0.407288570         0.100           0.1428571
## 30          0.273548518         0.050           0.0000000
## 31          0.503394122         0.300           0.1428571
## 32          0.458583736         0.550           0.7142857
## 33          0.224176819         0.100           0.2857143
## 34          1.000000000         0.250           0.1428571
## 35          0.236081962         0.100           0.1428571
## 36          0.015097957         0.350           0.4285714
## 37          0.035719055         0.300           0.5714286
## 38          0.517110972         0.000           0.0000000
## 39          0.273548518         0.200           0.4285714
## 40          0.122103618         0.200           0.0000000
## 41          0.178645564         0.300           0.2857143
## 42          0.664121151         0.050           0.2857143
## 43          0.112033889         0.050           0.1428571
## 44          0.952001105         0.000           0.0000000
## 45          0.007744657         0.050           0.1428571
## 46          0.157183360         0.050           0.1428571
## 47          0.273548518         0.100           0.1428571
## 48          0.386482564         0.350           0.4285714
## 49          0.101675445         0.227           0.0000000
## 50          0.448433708         0.350           0.5714286
## 51          0.526284712         0.350           0.4285714
## 52          0.273548518         0.900           0.2371968
## 53          0.689345568         0.750           1.0000000
## 54          0.103580687         0.300           0.4285714
```

```
## 55            0.225227231           0.000             0.0000000
## 56            0.041304645           0.400             0.1428571
##    fatalities_85_99 incidents_00_14 fatal_accidents_00_14 fatalities_00_14
## 1       0.000000000      0.00000000             0.0000000      0.000000000
## 2       0.314496314      0.54545455             0.5000000      0.310954064
## 3       0.000000000      0.09090909             0.0000000      0.000000000
## 4       0.157248157      0.45454545             0.0000000      0.000000000
## 5       0.000000000      0.18181818             0.0000000      0.000000000
## 6       0.194103194      0.54545455             1.0000000      0.121274752
## 7       0.808353808      0.36363636             0.5000000      0.558303887
## 8       0.000000000      0.45454545             0.5000000      0.024734982
## 9       0.000000000      0.45454545             0.5000000      0.310954064
## 10      0.122850123      0.36363636             0.0000000      0.000000000
## 11      0.002457002      0.63636364             0.0000000      0.000000000
## 12      0.248157248      0.30188679             0.3090909      0.121274752
## 13      0.000000000      0.09090909             0.0000000      0.000000000
## 14      0.793611794      0.00000000             0.0000000      0.000000000
## 15      0.000000000      0.54545455             0.0000000      0.000000000
## 16      0.000000000      0.18181818             0.0000000      0.000000000
## 17      0.223216355      0.18181818             0.5000000      0.795053004
## 18      0.039312039      0.00000000             0.0000000      0.000000000
## 19      0.115479115      0.00000000             0.0000000      0.000000000
## 20      1.000000000      0.30188679             1.0000000      0.180212014
## 21      0.692874693      0.36363636             0.5000000      0.049469965
## 22      0.009828010      0.09090909             0.0000000      0.000000000
## 23      0.410319410      0.45454545             1.0000000      0.325088339
## 24      0.000000000      0.00000000             0.0000000      0.000000000
## 25      0.638820639      0.36363636             1.0000000      0.077738516
## 26      0.000000000      0.27272727             0.5000000      0.505300353
## 27      0.000000000      0.09090909             0.0000000      0.000000000
## 28      0.363636364      0.45454545             0.0000000      0.000000000
## 29      0.223216355      0.00000000             0.0000000      0.000000000
## 30      0.000000000      0.18181818             1.0000000      1.000000000
## 31      0.007371007      0.09090909             0.0000000      0.000000000
## 32      0.223216355      0.09090909             0.0000000      0.000000000
## 33      0.051597052      0.00000000             0.0000000      0.000000000
## 34      0.004914005      0.27272727             0.0000000      0.000000000
## 35      0.083538084      0.27272727             1.0000000      0.121274752
## 36      0.574938575      0.90909091             1.0000000      0.162544170
## 37      0.181818182      0.18181818             0.5000000      0.003533569
## 38      0.000000000      0.45454545             0.0000000      0.000000000
## 39      0.125307125      0.27272727             0.0000000      0.000000000
## 40      0.000000000      0.54545455             0.5000000      0.388692580
## 41      0.769041769      1.00000000             0.0000000      0.000000000
## 42      0.014742015      0.18181818             0.5000000      0.293286219
## 43      0.390663391      0.09090909             0.0000000      0.000000000
## 44      0.000000000      0.72727273             0.0000000      0.000000000
## 45      0.034398034      0.36363636             0.0000000      0.000000000
## 46      0.562653563      0.27272727             0.0000000      0.000000000
## 47      0.007371007      0.09090909             0.5000000      0.010600707
## 48      0.240786241      0.63636364             1.0000000      0.664310954
## 49      0.000000000      0.00000000             0.0000000      0.000000000
## 50      0.756756757      0.18181818             0.5000000      0.003533569
## 51      0.157248157      0.72727273             1.0000000      0.296819788
```

```
## 52      0.783783784      0.30188679             1.0000000      0.385159011
## 53      0.550368550      1.00000000             1.0000000      0.081272085
## 54      0.420147420      0.09090909             0.0000000      0.000000000
## 55      0.000000000      0.00000000             0.0000000      0.000000000
## 56      0.201474201      0.18181818             0.0000000      0.000000000
##    accidents_per_100k_miles fatalities_per_incident
## 1                0.00000000             0.000000000
## 2                0.21301452             0.037231750
## 3                0.00000000             0.000000000
## 4                0.13839057             0.113074205
## 5                0.00000000             0.000000000
## 6                0.16498274             0.293992933
## 7                0.19005141             0.196474833
## 8                0.11631141             0.012367491
## 9                0.08556659             0.124381625
## 10               0.23667608             0.064246707
## 11               0.04486199             0.001413428
## 12               0.12638988             0.192300539
## 13               0.00000000             0.000000000
## 14               0.62431395             0.913074205
## 15               0.00000000             0.000000000
## 16               0.00000000             0.000000000
## 17               0.71101611             0.767289248
## 18               0.19761930             0.113074205
## 19               0.15005033             0.221436985
## 20               0.17721123             0.134864547
## 21               0.59244359             0.348645465
## 22               0.24624212             0.028268551
## 23               0.21301452             0.122025913
## 24               0.00000000             0.000000000
## 25               0.67335562             0.284704695
## 26               0.27407757             0.505300353
## 27               0.00000000             0.000000000
## 28               0.07040676             0.232430310
## 29               0.05247142             0.196474833
## 30               0.59550848             1.000000000
## 31               0.04406440             0.005300353
## 32               0.23810999             0.462082088
## 33               0.16487873             0.098939929
## 34               0.02410644             0.003140950
## 35               0.23846339             0.196474833
## 36               0.21301452             0.219866510
## 37               1.00000000             0.117785630
## 38               0.00000000             0.000000000
## 39               0.83801089             0.090106007
## 40               0.12094412             0.141342756
## 41               0.19216922             0.245779348
## 42               0.10425710             0.314487633
## 43               0.12678607             0.749116608
## 44               0.00000000             0.000000000
## 45               0.25370317             0.032979976
## 46               0.10421561             0.647349823
## 47               0.63693076             0.021201413
## 48               0.27366045             0.269493522
```

```
## 49              0.00000000            0.196474833
## 50              0.24254558            0.436749117
## 51              0.21222317            0.130742049
## 52              0.11569976            0.183317272
## 53              0.30273101            0.129302447
## 54              0.39643301            0.302120141
## 55              0.00000000            0.000000000
## 56              0.19188975            0.105364600
```

```r
# Install and load the necessary library

library(caTools)
```

```
## Warning: package 'caTools' was built under R version 4.3.2
```

```r
set.seed(123)  # Set seed for reproducibility
split <- sample.split(dataset$incidents_00_14, SplitRatio = 0.7)

train_data <- subset(dataset, split == TRUE)
test_data <- subset(dataset, split == FALSE)
```

```r
# Decision Tree with limited depth
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 4.3.2
```

```r
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.3.2

## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
dt_model_fast <- rpart(incidents_00_14 ~ ., data = train_data, method = "class", maxdepth = 10)

# Random Forest
rf_model_fast <- randomForest(incidents_00_14 ~ ., data = train_data, ntree = 100)
```

```r
# Predictions on the test set
dt_predictions_fast <- predict(dt_model_fast, test_data, type = "class")
rf_predictions_fast <- predict(rf_model_fast, test_data)

# Confusion Matrix
confusion_matrix_dt_fast <- table(dt_predictions_fast, test_data$incidents_00_14)
confusion_matrix_rf_fast <- table(rf_predictions_fast, test_data$incidents_00_14)

# Compare Confusion Matrices
print("Confusion Matrix for Faster Decision Tree:")
```

```
## [1] "Confusion Matrix for Faster Decision Tree:"
```

```r
print(confusion_matrix_dt_fast)
```

```
##
## dt_predictions_fast  0 0.0909090909090909 0.181818181818182 0.272727272727273
##   0                  2                  3                 1                 1
##   0.0909090909090909 1                  0                 0                 0
##   0.181818181818182  0                  0                 0                 0
##   0.272727272727273  0                  0                 0                 0
##   0.30188679245283   0                  0                 1                 0
##   0.363636363636364  0                  0                 0                 0
##   0.454545454545455  0                  0                 0                 0
##   0.545454545454545  0                  0                 0                 0
##   0.636363636363636  0                  0                 0                 0
##   0.727272727272727  0                  0                 0                 0
##   0.909090909090909  0                  0                 0                 0
##   1                  0                  0                 0                 0
##
## dt_predictions_fast  0.30188679245283 0.363636363636364 0.454545454545455
##   0                                 0                 1                 1
##   0.0909090909090909                0                 0                 0
##   0.181818181818182                 0                 0                 0
##   0.272727272727273                 0                 0                 0
##   0.30188679245283                  1                 0                 1
##   0.363636363636364                 0                 0                 0
##   0.454545454545455                 0                 0                 0
##   0.545454545454545                 0                 0                 0
##   0.636363636363636                 0                 0                 0
##   0.727272727272727                 0                 0                 0
##   0.909090909090909                 0                 0                 0
##   1                                 0                 0                 0
##
## dt_predictions_fast  0.545454545454545 0.636363636363636 0.727272727272727 1
##   0                                  0                 1                 1 0
##   0.0909090909090909                 0                 0                 0 1
##   0.181818181818182                  0                 0                 0 0
##   0.272727272727273                  0                 0                 0 0
##   0.30188679245283                   1                 0                 0 0
##   0.363636363636364                  0                 0                 0 0
##   0.454545454545455                  0                 0                 0 0
##   0.545454545454545                  0                 0                 0 0
```

```
##   0.636363636363636                  0                 0                 0 0
##   0.727272727272727                  0                 0                 0 0
##   0.909090909090909                  0                 0                 0 0
##   1                                  0                 0                 0 0
```

```
print("Confusion Matrix for Random Forest:")
```

```
## [1] "Confusion Matrix for Random Forest:"
```

```
print(confusion_matrix_rf_fast)
```

```
##
## rf_predictions_fast 0 0.0909090909090909 0.181818181818182 0.272727272727273
##   0.119878954378954 1                   0                 0                 0
##   0.166934203637034 0                   0                 0                 0
##   0.20487270957554  1                   0                 0                 0
##   0.214646369353917 0                   0                 0                 1
##   0.225266056794359 0                   0                 0                 0
##   0.227013347763348 0                   0                 0                 0
##   0.229343434343434 0                   1                 0                 0
##   0.229791261673337 0                   1                 0                 0
##   0.243386125404993 0                   1                 0                 0
##   0.252924433009339 1                   0                 0                 0
##   0.299328088578088 0                   0                 0                 0
##   0.358843053173242 0                   0                 0                 0
##   0.366833333333334 0                   0                 1                 0
##   0.415569754145226 0                   0                 0                 0
##   0.427694110920526 0                   0                 1                 0
##   0.493007146941109 0                   0                 0                 0
##   0.552459691252144 0                   0                 0                 0
##
## rf_predictions_fast 0.30188679245283 0.363636363636364 0.454545454545455
##   0.119878954378954                0                 0                 0
##   0.166934203637034                0                 0                 0
##   0.20487270957554                 0                 0                 0
##   0.214646369353917                0                 0                 0
##   0.225266056794359                0                 0                 0
##   0.227013347763348                0                 0                 1
##   0.229343434343434                0                 0                 0
##   0.229791261673337                0                 0                 0
##   0.243386125404993                0                 0                 0
##   0.252924433009339                0                 0                 0
##   0.299328088578088                0                 0                 0
##   0.358843053173242                0                 1                 0
##   0.366833333333334                0                 0                 0
##   0.415569754145226                1                 0                 0
##   0.427694110920526                0                 0                 0
##   0.493007146941109                0                 0                 0
##   0.552459691252144                0                 0                 1
##
## rf_predictions_fast 0.545454545454545 0.636363636363636 0.727272727272727 1
##   0.119878954378954                 0                 0                 0 0
##   0.166934203637034                 0                 1                 0 0
```

```
##   0.20487270957554          0          0          0 0
##   0.214646369353917         0          0          0 0
##   0.225266056794359         0          0          0 1
##   0.227013347763348         0          0          0 0
##   0.229343434343434         0          0          0 0
##   0.229791261673337         0          0          0 0
##   0.243386125404993         0          0          0 0
##   0.252924433009339         0          0          0 0
##   0.299328088578088         0          0          1 0
##   0.358843053173242         0          0          0 0
##   0.366833333333334         0          0          0 0
##   0.415569754145226         0          0          0 0
##   0.427694110920526         0          0          0 0
##   0.493007146941109         1          0          0 0
##   0.552459691252144         0          0          0 0
```

Conclusion

the analysis of the airline dataset provides valuable insights into the safety records of various airlines over the specified periods. Key findings include variations in incident rates, fatal accidents, and fatalities across different airlines. Additionally, the exploration of normalization techniques has facilitated a comparative assessment of airline performance, offering a standardized perspective. This analysis underscores the importance of ongoing safety measures within the aviation industry. Further research and continuous monitoring are crucial for ensuring passenger safety and improving overall airline performance."