```
---
title: "courseworkQ2a"
author: "Chris_Pang"
date: "2024-04-08"
output: html_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

For Coursework Q, Student ID: 220457882

Creating databse. Cleaning and preparing datas
```{r database}

library(DBI)
library(dplyr)
library(stringr)
library(ggplot2)
library(ggthemes)
# set the working directory that contains the files
setwd("/Users/dataverse_files")

# ======== create the database ========
if (file.exists("airline2.db"))
  file.remove("airline2.db")
conn <- dbConnect(RSQLite::SQLite(), "airline2.db")



# ======== write to the database ========
# load in the data from the csv files
airports <- read.csv("airports.csv", header = TRUE)
carriers <- read.csv("carriers.csv", header = TRUE)
planes <- read.csv("plane-data.csv", header = TRUE)
dbWriteTable(conn, "airports", airports)
dbWriteTable(conn, "carriers", carriers)
dbWriteTable(conn, "planes", planes)

for(i in c(1991:2000)) {
  ontime <- read.csv(paste0(i, ".csv"), header = TRUE)
  if(i == 2000) {
    dbWriteTable(conn, "ontime", ontime)
  } else {
    dbWriteTable(conn, "ontime", ontime, append = TRUE)
  }
}

```
```
Warning: The working directory was changed to /Users/dataverse_files inside a notebook chunk. The working directory will be reset when the chunk is finished running. Use the knitr root.dir option in t
working directory for notebook chunks.[1] TRUE
Error: Table ontime exists in database, and both overwrite and append are FALSE
```

Answering Q2a, plotting the graph individuallly from 1991 to 1999.
```{r Q2a}
#=================Q2a Best time of the day to fly throught 1991 to 1999==========================
library(DBI)
library(dplyr)
library(ggplot2)
library(RColorBrewer)
time_segment <- function(deptime) {
  hour <- as.numeric(substr(deptime, 1, 2))

  if (hour >= 2 && hour < 6) {
    return('Early Morning')
  } else if (hour >= 6 && hour < 10) {
    return('Morning')
  } else if (hour >= 10 && hour < 14) {
    return('Midday')
  } else if (hour >= 14 && hour < 17) {
    return('Afternoon')
  } else if (hour >= 17 && hour < 21) {
    return('Evening')
  } else {
    return('Night')
  }
}

# Initialize a list to store each year's lowest delay count and corresponding segments
lowest_delay_counts_by_year <- list()

for (year in 1991:1999) {
  query <- sprintf("SELECT CAST(DepTime AS INTEGER) AS DepTime FROM ontime WHERE ArrDelay > 15 AND DepDelay > 15 AND Year = %d", year)
  timeOfDayDf <- dbGetQuery(conn, query)
  timeOfDayDf$DepTime <- sprintf("%04d", timeOfDayDf$DepTime)
  timeOfDayDf$Segment <- sapply(timeOfDayDf$DepTime, time_segment)

  segment_order <- factor(c('Early Morning', 'Morning', 'Midday', 'Afternoon', 'Evening', 'Night'),
                          levels = c('Early Morning', 'Morning', 'Midday', 'Afternoon', 'Evening', 'Night'))
  timeOfDayDf$Segment <- factor(timeOfDayDf$Segment, levels = levels(segment_order))

```
```

```
  segment_counts_ordered <- as.data.frame(table(timeOfDayDf$Segment))
  colnames(segment_counts_ordered) <- c("Segment", "Count")

  # Find the lowest delay count for the year and store it
  min_delay_count <- min(segment_counts_ordered$Count)
  lowest_delay_segments <- segment_counts_ordered$Segment[which.min(segment_counts_ordered$Count)]
  lowest_delay_counts_by_year[[as.character(year)]] <- list(count = min_delay_count, segment = lowest_delay_segments)

  # Plotting
  ggplot(segment_counts_ordered, aes(x = Segment, y = Count, fill = Segment)) +
    geom_bar(stat = "identity", show.legend = FALSE) +
    geom_text(aes(label = Count), vjust = -0.3, size = 3.5) +
    scale_fill_brewer(palette = "Pastel1") +
    labs(title = sprintf("Count of Flights Delayed by Time of Day in %d", year),
         x = "Time of Day Segment",
         y = "Number of Flights") +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
    guides(fill = guide_legend(title = "Time Segments", override.aes = list(fill = brewer.pal(6, "Pastel1")))) +
    scale_x_discrete(labels = c("Early Morning: 0200-0600", "Morning: 0600-1000", "Midday: 1000-1400",
                                "Afternoon: 1400-1700", "Evening: 1700-2100", "Night: 2100-0200"))

  print(ggplot2::last_plot()) # Print the plot for each year
}



#==================Q2a plotting the 9 years collated graph ============================

years <- names(lowest_delay_counts_by_year)
lowest_counts <- sapply(lowest_delay_counts_by_year, function(info) info$count)
segments <- sapply(lowest_delay_counts_by_year, function(info) info$segment)

# Convert counts to numeric
lowest_counts <- as.numeric(lowest_counts)

# Determine the range of y-values to dynamically position text within the graph
y_min <- min(lowest_counts)
y_max <- max(lowest_counts)
y_range <- y_max - y_min
offset <- y_range * 0.02  # Offset for text positioning, 1% of the y-range
plot(years, lowest_counts, type = "o", col = "#008080", xlab = "Year", ylab = "Number of Delayed Flights", main = "Lowest Number of Delayed
grid()
lines(years, lowest_counts, type = "o", col = "#008080")

#adjusting text size
text_size <- 0.501


 # Adjust the offset to position text below the points
 offset <- -abs(offset)  # Ensure offset is negative to move text below

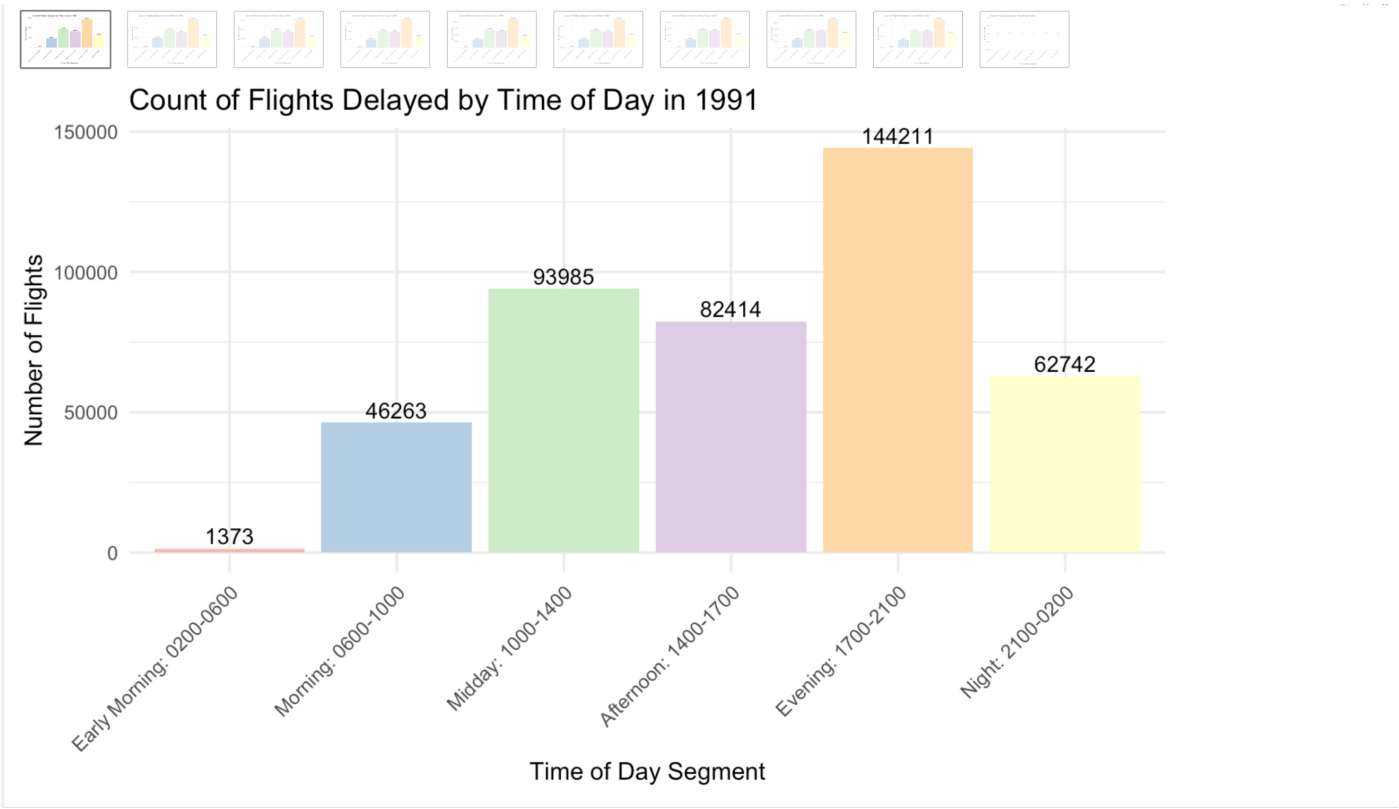
for (i in 1:length(years)) {
  year <- years[i]
  count <- lowest_counts[i]
  segment <- segments[i]

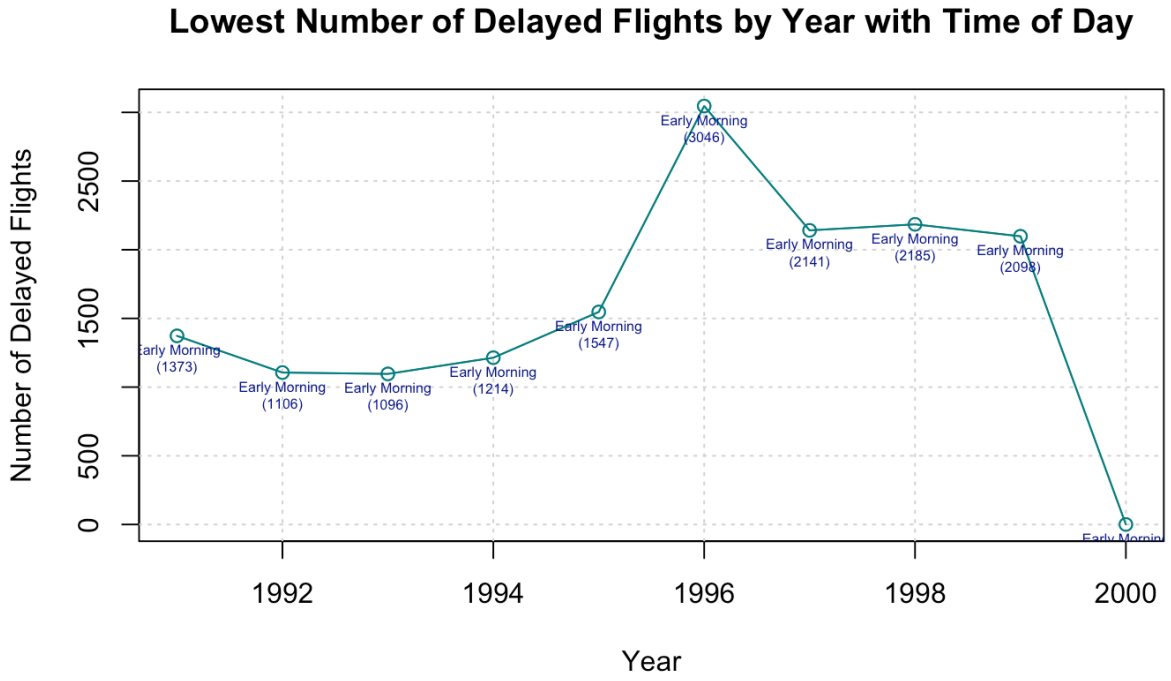


  vertical_position <- count + offset  # Adjust vertical position to be below the point

  # Place text below each point. Adjust 'adj' for fine-tuning horizontal alignment if necessary.
  text(year, vertical_position, labels = paste(segment, "\n(", count, ")", sep=""), col = "darkblue", cex = text_size, adj = c(0.5, 1)
}

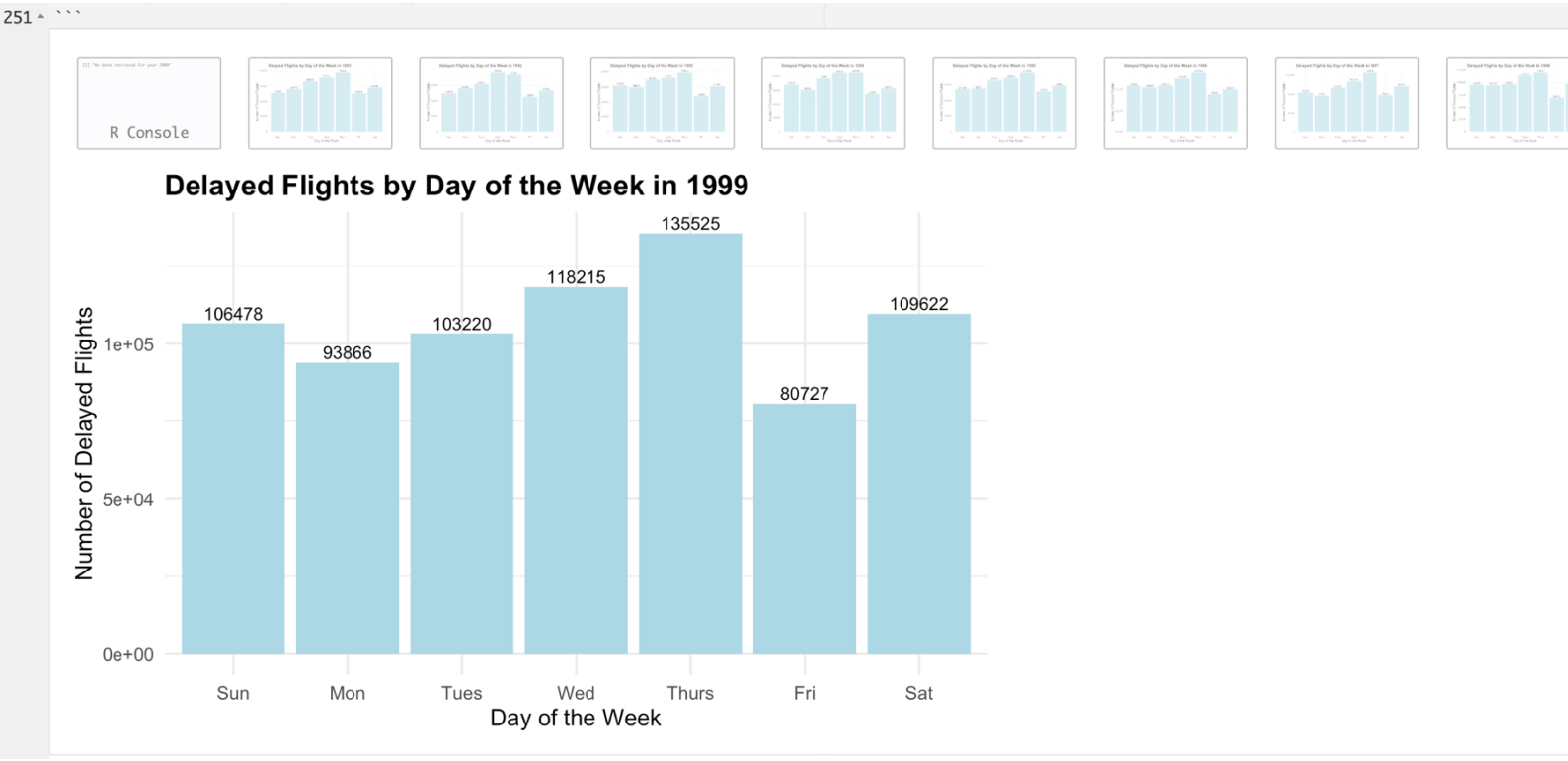

```

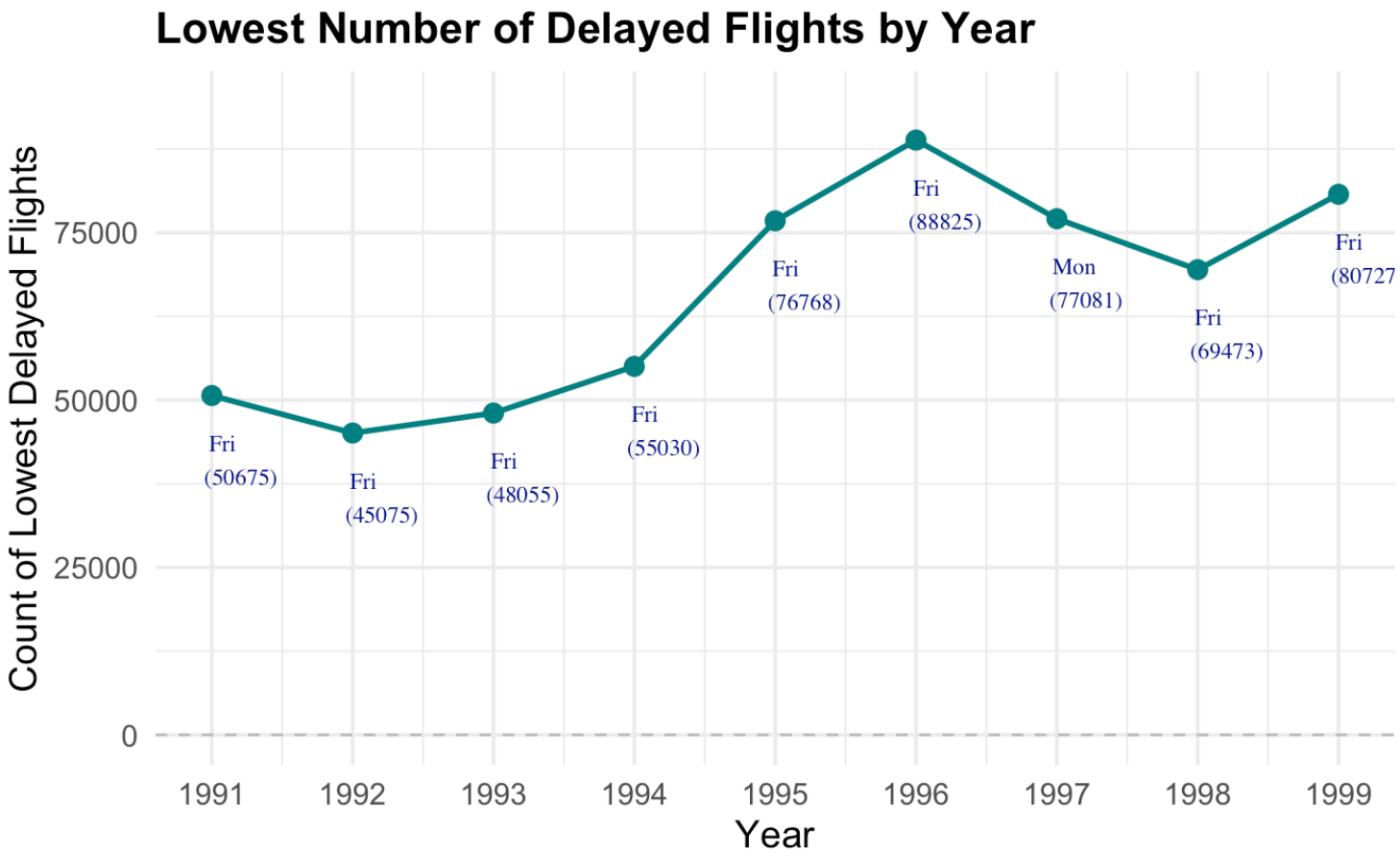

Count of Flights Delayed by Time of Day in 1991
Number of Flights
150000
100000
50000
0
1373
46263
93985
82414
144211
62742
Early Morning: 0200-0600
Morning: 0600-1000
Midday: 1000-1400
Afternoon: 1400-1700
Evening: 1700-2100
Night: 2100-0200
Time of Day Segment


Lowest Number of Delayed Flights by Year with Time of Day
Number of Delayed Flights
0
500
1500
2500
Early Morning (1373)
Early Morning (1106)
Early Morning (1096)
Early Morning (1214)
Early Morning (1547)
Early Morning (3046)
Early Morning (2141)
Early Morning (2185)
Early Morning (2098)
1992
1994
1996
1998
2000
Year

Answering Q2a,plotting best day of the week to fly individually from 1991-1999

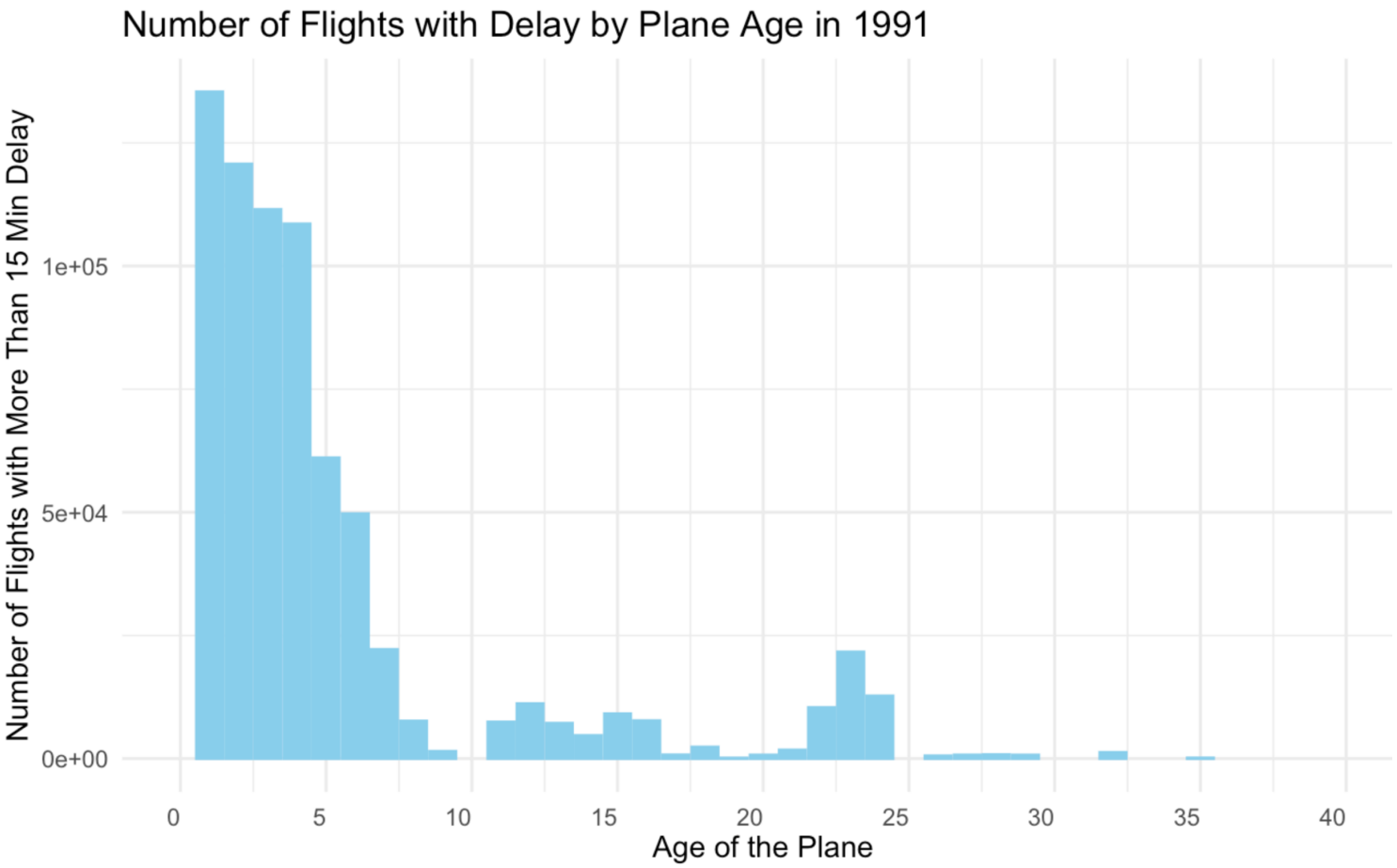
```{r Q2a Day of the week}

#==================Q2a best day of the week to fly to minimize delay ============================
lowest_delay_day_by_year <- list()

for (year in 1991:2000) {
  query <- sprintf("
    SELECT DayOfWeek, COUNT(*) AS Count
    FROM ontime
    WHERE Year = %d AND ArrDelay > 15 AND DepDelay > 15
    GROUP BY DayOfWeek
    ORDER BY DayOfWeek", year)

  # Execute the query
  dayOfWeekCounts <- dbGetQuery(conn, query)

  # Initialize a dataframe for all days with counts set to 0
  counts <- data.frame(DayOfWeek = 1:7, Count = rep(0, 7))

  # Check if any data is retrieved
  if (nrow(dayOfWeekCounts) == 0) {
    print(paste("No data retrieved for year", year))
    next  # Skip this year if no data is returned
  }

  # Update counts based on the query result
  for (row in 1:nrow(dayOfWeekCounts)) {
    # Ensure that 'DayOfWeek' column exists and has valid data
    if ("DayOfWeek" %in% names(dayOfWeekCounts) && !is.na(dayOfWeekCounts[row, "DayOfWeek"])) {
      day <- dayOfWeekCounts[row, "DayOfWeek"]
      if (day >= 1 && day <= 7) {
        counts[day, "Count"] <- dayOfWeekCounts[row, "Count"]
      } else {
        print(paste("Invalid day:", day, "in year", year))
      }
    } else {
      print(paste("Missing or invalid 'DayOfWeek' data in year", year))
    }
  }

  # Plotting a bar chart for each year
  tick_label <- c('Sun', 'Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat')
  ggplot(counts, aes(x = factor(DayOfWeek, labels = tick_label), y = Count)) +
    geom_bar(stat = "identity", fill = "lightblue") +
    geom_text(aes(label = Count), vjust = -0.3, size = 3.5) +
    labs(x = 'Day of the Week', y = 'Number of Delayed Flights',
         title = sprintf('Delayed Flights by Day of the Week in %d', year)) +
    theme_minimal(base_size = 13) +
    theme(plot.title = element_text(size = 16, face = "bold"))

  # Force the plot to display
  print(ggplot2::last_plot())

  # Store the day with the least delays for the year
  min_delay_count <- min(counts$Count)
  min_delay_day <- tick_label[which.min(counts$Count)]
  lowest_delay_day_by_year[[as.character(year)]] <- list(Day = min_delay_day, Count = min_delay_count)
}



#==================Q2a  plotting the 9 years collated graph ============================

# Create a data frame from the list
data <- data.frame(
  Year = as.numeric(names(lowest_delay_day_by_year)),
  Lowest_Count = sapply(lowest_delay_day_by_year, function(info) info$Count),
  Lowest_Days = sapply(lowest_delay_day_by_year, function(info) info$Day)
)

# Plotting
ggplot(data, aes(x = Year, y = Lowest_Count)) +
  geom_line(color = "#008080", size = 1) +  # Using hex code for teal
  geom_point(color = "#008080", size = 3) +  # Using hex code for teal
  geom_text(aes(label = paste(Lowest_Days, "\n(", Lowest_Count, ")", sep="")),
            nudge_y = (max(data$Lowest_Count) - min(data$Lowest_Count)) * 0.03,
            size = 3, color = "darkblue",
            hjust = 0.1, vjust = 2,
            check_overlap = TRUE,
            family = "serif") +
  labs(title = 'Lowest Number of Delayed Flights by Year',
       x = 'Year', y = 'Count of Lowest Delayed Flights') +
  theme_minimal(base_size = 14) +
  theme(plot.title = element_text(size = 16, face = "bold")) +
  scale_x_continuous(breaks = data$Year) +  # Ensure X axis has all years as breaks
  scale_y_continuous(expand = expansion(mult = c(0.05, 0.1))) +  # Add some padding around Y axis
  geom_hline(yintercept = 0, linetype = "dashed", color = "gray")

# Adjust plot theme for better readability
theme_set(theme_minimal(base_size = ))
```

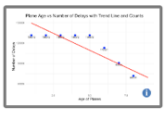

R Console
Delayed Flights by Day of the Week in 1999
Number of Delayed Flights
0e+00
5e+04
1e+05
106478
93866
103220
118215
135525
80727
109622
Sun
Mon
Tues
Wed
Thurs
Fri
Sat
Day of the Week


Lowest Number of Delayed Flights by Year
Count of Lowest Delayed Flights
0
25000
50000
75000
Fri
(50675)
Fri
(45075)
Fri
(48055)
Fri
(55030)
Fri
(76768)
Fri
(88825)
Mon
(77081)
Fri
(69473)
Fri
(80727
1991
1992
1993
1994
1995
1996
1997
1998
1999
Year

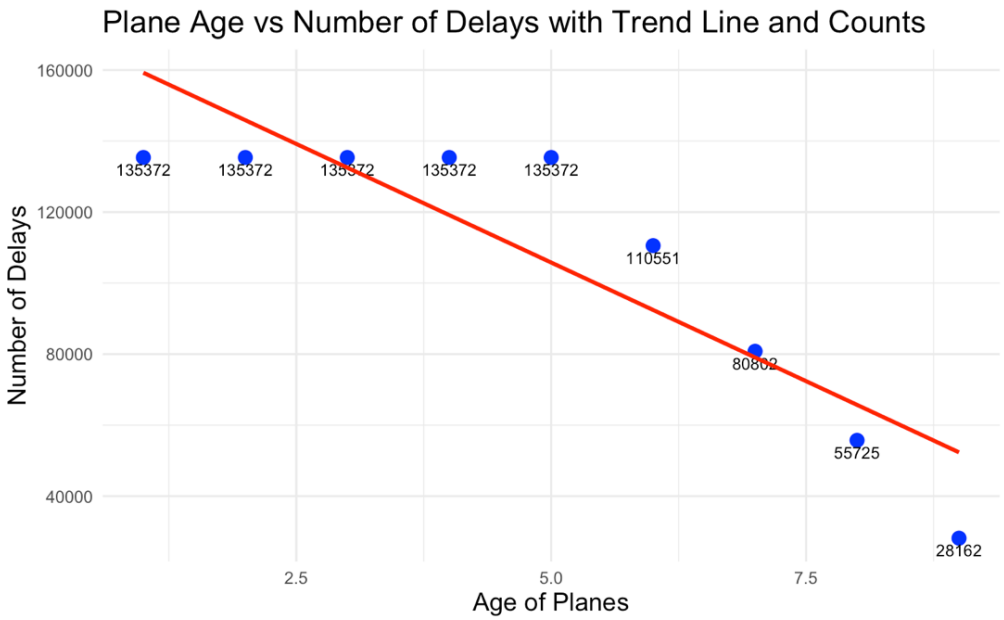
For Q2b, Plotting do older planes suffer more delay graph individually from 1991-1999

```{r Q2b do older planes suffer more delay}

#=================Q2b plotting do older planes suffer more delay on a year to year basis==========================

# Initialize an empty list to store the data
max_delay_age_by_year <- list()

for (year in 1991:1999) {
  # Construct the SQL query
  query <- sprintf("
    SELECT CAST(planes.year AS INTEGER) AS year
    FROM planes
    JOIN ontime ON planes.TailNum = ontime.TailNum
    WHERE CAST(ontime.ArrDelay AS INTEGER) > 15 AND CAST(ontime.DepDelay AS INTEGER) > 15
    AND planes.year IS NOT NULL
    AND CAST(ontime.Year AS INTEGER) >= %s
  ", year)

  # Execute the query
  dateOfPlane <- dbGetQuery(conn, query)

  # Check if data was fetched
  if (nrow(dateOfPlane) == 0) {
    cat(sprintf("No data fetched for %d.\n", year))
    next
  }

  # Calculate the age of the plane in that year
  dateOfPlane$`Age in year` <- year - dateOfPlane$year

  # Countby age
  age_counts <- count(dateOfPlane, `Age in year`)

  # Find the age with the maximum count
  max_age <- age_counts[which.max(age_counts$n),]
  max_delay_age_by_year[[as.character(year)]] <- list(Age = max_age$`Age in year`, Delays = max_age$n)

  # Plot the graph for the specific year
  p <- ggplot(age_counts, aes(x = `Age in year`, y = n, fill = `Age in year`)) +
    geom_bar(stat = "identity", color = "skyblue", fill = "skyblue") +
    scale_x_continuous(limits = c(0, 40), breaks = seq(0, 40, by = 5)) +
    labs(title = sprintf("Number of Flights with Delay by Plane Age in %d", year),
         x = "Age of the Plane",
         y = "Number of Flights with More Than 15 Min Delay") +
    theme_minimal() +
    theme(axis.text.x = element_text( vjust = 0.5, hjust=1))

  print(p)
}

# Output the results
for (year in names(max_delay_age_by_year)) {
  cat(sprintf("In %s, the age with the most delays was %s years with %s delays.\n",
              year, max_delay_age_by_year[[year]]$Age, max_delay_age_by_year[[year]]$Delays))
}


#==================plotting the 9 years collated regression graph============================
 library(ggplot2)
 library(dplyr)

 # Convert the list to a data frame
 data <- bind_rows(max_delay_age_by_year, .id = "Year") %>%
   mutate(Year = as.numeric(Year), Age = as.numeric(Age), Delays = as.numeric(Delays)) %>%
   arrange(Year)

 # Plotting with annotations
 p <- ggplot(data, aes(x = Age, y = Delays)) +
   geom_point(color = "blue", size = 3, show.legend = TRUE, aes(label = Delays)) +
   geom_text(aes(label = Delays), vjust = 1.5, color = "black", size = 3) +
   labs(title = "Plane Age vs Number of Delays with Counts",
        x = "Age of Planes", y = "Number of Delays") +
   theme_minimal() +
   theme(plot.title = element_text(size = 16),
         axis.title = element_text(size = 13)) +
   geom_smooth(method = "lm", color = "red", se = FALSE, show.legend = TRUE) +
   ggtitle("Plane Age vs Number of Delays with Trend Line and Counts") +
   xlab("Age of Planes") + ylab("Number of Delays") +
   theme(legend.position = "bottom") +
   scale_color_manual(values = c("blue", "red"), labels = c("Delays vs. Age", "Line of Best Fit"))

 print(p)

 # Optionally, calculate the correlation coefficient
 correlation <- cor(data$Age, data$Delays, use = "complete.obs")
 cat(sprintf("Correlation coefficient between plane age and number of delays: %.2f\n", correlation))


```





[38;5;232m`geom_smooth()` using formula = 'y ~ x' [39m



Plotting for q2c)

```{r Q2c}

library(tidyverse)

set.seed(42)

for (year in 1991:2000) {
  # Subset the data for the current year
  year_data <- filter(ontime, Year == year)

  # Impute missing values in features
  imputed_data <- year_data %>%
    select(DepTime, ArrTime, Distance) %>%
    bind_cols(Year = year_data$Year, Diverted = year_data$Diverted) %>%
    replace_na(list(DepTime = mean(year_data$DepTime, na.rm = TRUE),
                    ArrTime = mean(year_data$ArrTime, na.rm = TRUE),
                    Distance = mean(year_data$Distance, na.rm = TRUE))).

  # Splitting the data into features (X) and target (y)
  X <- imputed_data %>% select(DepTime, ArrTime, Distance)
  y <- imputed_data$Diverted

  # Splitting dataset into training and testing sets
  trainIndex <- createDataPartition(y, p = .8, list = FALSE, times = 1)
  X_train <- X[trainIndex, ]
  X_test <- X[-trainIndex, ]
  y_train <- y[trainIndex]
  y_test <- y[-trainIndex]

  # Initializing and fitting the logistic regression model
  logistic_model <- train(X_train, y_train, method = 'glm', family = 'binomial', trControl = trainControl(method = 'none'))

  # Getting coefficients and intercept
  coefficients <- c(intercept = coef(logistic_model$finalModel)['(Intercept)'], coef(logistic_model$finalModel)[-1])

  # Creating a DataFrame to hold coefficients for the current year
  temp_df <- data.frame(t(coefficients), Year = year)

  # Adding temporary DataFrame to the main coefficients DataFrame
  coefficients_df <- bind_rows(coefficients_df, temp_df)
}

# Setting 'Year' as the DataFrame index for plotting
coefficients_df$Year <- factor(coefficients_df$Year)

# Plotting the coefficients for each feature across years
coefficients_df %>%
  pivot_longer(cols = -Year, names_to = "Feature", values_to = "Coefficient") %>%
  ggplot(aes(x = Year, y = Coefficient, color = Feature, group = Feature)) +
  geom_line() +
  geom_point() +
  labs(title = 'Logistic Regression Coefficients for Diverted Flights Across Years',
       x = 'Year', y = 'Coefficient Value') +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  scale_color_brewer(palette = "Dark2")


```