A Predictive Analysis of Time Spent in Airport Arrivals in US

Imogen Meers, Sarah Deussing

2024-10-02

Introduction

Topic We want to understand airport arrival wait patterns by predicting the likelihood of delays and the time spent waiting in customs/passport control. We want to get all flight arrivals and custom wait times from Chicago O'Hare Airport from the last month. To do so, we will use:

1. A Form API of wait times for customs from international flights arriving into Chicago O'Hare, one of the major US airports.
   - https://awt.cbp.gov/
2. A REST API of historical flight arrivals data
   - https://rapidapi.com/oag-oag-default/api/flight-info-api Endpoint: GET /status - Get flight status by specifying arrival date range and arrivals airport

Our motivation for pursuing this problem is our experiences with international travel. We understand the stress of making connections and wasting time walking around new airports during travel. We chose Chicago O'Hare as it is our local international airport and one of the major airports within the US.

Significance This research problem is significant as oftentimes, flights take longer than expected or airports have unexpected delays. We want to be able to minimize wasted time in the airport and understand the likelihood of making a connecting flight based on historical flight data from a number of different airports.

When flying internationally into the United States, this information can be very useful to help plan one's travel. Based on their predicted total wait and delay time, travelers can select connecting flights that minimize wasted time waiting around in the airport and allow for adequate times between flights. This prediction information will also help travelers decide what time to book taxis or hire cars, or decide what times to avoid because of business.

Data Acquisition Process

The Form API data from: https://awt.cbp.gov/. For this API, we are submitting 3 form elements: airportID, start date, and end date. Within each request, we added a short wait time to ensure we are not overloading the API with requests.

The REST API data from: https://rapidapi.com/oag-oag-default/api/flight-info-api For this API, we had to create a free account with a limited number of requests. Each API request provided 100 rows per page, and we extracted the desired information from nested tables in each row of the json file. Because we only have the free version of the API, we have a limited number of requests from the API. Our solution is to clean the JSON into a data frame with each request then append that data frame to a CSV.

Flight Arrivals Data

We obtained free API keys for the OAG info API and identified the GET status endpoint as the most suitable for our project. This endpoint provides data on:

- Core Schedule information e.g. carrier, flight number, departure/arrival times, flight date, departure/destination airports
- Estimate and actual times for gates
- Estimate and actual times for runway
- Flight delays, cancellations, diversions and recovery
- Aircraft type and tail number
- Departure/arrival gates and terminals
- Check-in desk and baggage carousel
- Codeshare

These will all be useful features to predict the future flight arrival information.

Challenges

Originally, we wanted to get at least a year's worth of data but when we calculated the number of flights per day (up to 1000) and the page limit of 100, we realized this world take a huge amount of requests far beyond the usage limit for this API. We are only using a Basic Plan, which is 50 total requests with a maximum of 10 per minute.

We estimated that by using 2 x API Keys, we would roughly be able to obtain a week's worth of data. However, upon implementation we realized that there was a row for every code-share, meaning some flights that were operated by an alliance of airlines had 7 different flight codes so 7 rows describing the same physical flight.

Finally, we decided on also filtering our request to include just American Airlines this way we avoid the code-share issue and could still get enough data to analyze over time. We were able to get a week's worth of flight data into Chicago O'Hare and via American Airlines.

Implementation

Below is our R code, which includes the initial GET request and then a loop that facilitates pagination with a next cursor.

These are the features that differ from our first attempt at getting arrivals data but we soon realized the magnitude of the data and decided to pivot our request to only request American Airlines, rather than all airlines.

We chose American Airlines because it is the airline we fly frequently ourselves and has both domestic and international flights.

```r
#Our first API key
url <- "https://flight-info-api.p.rapidapi.com/status"
api_key = '2a26d82d2fmsh3b29970cdca97d1p138b78jsn60a78272a81e'
api_host = 'flight-info-api.p.rapidapi.com'
```

```r
#Our original query string without the filter
queryString <- list(
  version = "v2",
  ArrivalDateTime = "2024-08-01T00:00/2024-09-01T00:00",
  ArrivalAirport = "ORD",
  FlightType = "Scheduled",
  CodeType = "IATA",
  ServiceType = "Passenger")

  #The original line we used to write our JSOn
  write_json(content, 'flightAPIJson.json')

  #The original line we used to write our flight data to CSV
  write.csv(flight_data, 'requestA.csv', row.names=FALSE)
```

This is the final code that we used for our American Airlines only requests. The query string changes to include the CarrierCode = ‘AA’ and the csv and json file names change so we write to another place.

```r
library(httr)
library(jsonlite)
library(tidyr)

#Query String for only American Airlines to ORD
#We started from 15:52 as we already had data up to 15:51 from our initial attempt with all airlines
queryString <- list(
  version = "v2",
  ArrivalDateTime = "2024-08-01T15:52/2024-09-01T00:00",
  ArrivalAirport = "ORD",
  FlightType = "Scheduled",
  CodeType = "IATA",
  ServiceType = "Passenger",
  CarrierCode = "AA"



#Initial API Request
url <- 'https://flight-info-api.p.rapidapi.com/status'

#GET request
#We used a second API key as we reached our limit on the first one
response <- VERB('GET', url = url, query = queryString, add_headers('x-rapidapi-key' = '572d8c59b1mshca

content <- content(response, "text")

json <- fromJSON(content)

list_data <- jsonlite::flatten(json$data) #flatten data frame, still with some nesting, will need to be

write_json(content, 'flightAPIJson3.json')

paging_next <- parse_response(list_data)
```

```
#Subsequent API requests where we use the cursor to navigate to the next page
while (paging_next != ""){

  response <- VERB('GET', url = paging_next, add_headers('x-rapidapi-key' = '572d8c59b1mshca4d962b7338c

  #This we break the loop if a request fails, including 429 when we reach limit
  if (status_code(response) != 200) {
    print("Error with request")
    break #break out of while

  }

  content <- content(response, "text")
  json <- fromJSON(content)
  list_data <- jsonlite::flatten(json$data) #flatten data frame, still with some nesting, will need to


  #Writing JSON resp to file incase something happens
  existing_data <- fromJSON("flightAPIJson3.json")

  combined_data <- append(existing_data, content)

  write_json(combined_data, 'flightAPIJson3.json')

  #Parse data and write to CSV
  paging_next <- parse_response(list_data)

  Sys.sleep(7) #Add a pause to make sure we stay below the 10 requests per min limit


}


##Function that parses data from the request into a df and then writes to a csv
parse_response(list_data){

  flight_data = data.frame()
  for (i in 1:nrow(list_data)) {

    data = list_data[i,]
    new_row1 <- data.frame(
      airline = data$carrier.iata,
      timeOfFlight = data$elapsedTime,
      deptAirport = data$departure.airport.iata,
      arrAirport = data$arrival.airport.iata,
      arrDay = data$arrival.date.local,
      arrTime = data$arrival.time.local,
      deptCountry = data$departure.country.code,
      flightNumber = data$flightNumber,

      estimatedOutGateVariation = ifelse(is.null(list_data$statusDetails[[i]]$departure$estimatedTime$ou
      estimatedOutGate = ifelse(is.null(list_data$statusDetails[[i]]$departure$estimatedTime$outGate), 
      estimatedOffGround = ifelse(is.null(list_data$statusDetails[[i]]$departure$estimatedTime$offGroun
```

```r
      actualOutGateVariation = ifelse(is.null(list_data$statusDetails[[i]]$departure$actualTime$outGateV
      actualOutGate = ifelse(is.null(list_data$statusDetails[[i]]$departure$actualTime$outGate), NA, lis
      actualOffGround = ifelse(is.null(list_data$statusDetails[[i]]$departure$actualTime$offGround), NA

      estimatedInGateVariation = ifelse(is.null(list_data$statusDetails[[i]]$arrival$estimatedTime$inGat
      estimatedInGate = ifelse(is.null(list_data$statusDetails[[i]]$arrival$estimatedTime$inGate), NA, l
      estimatedOnGround = ifelse(is.null(list_data$statusDetails[[i]]$arrival$estimatedTime$onGround), N
      actualInGateVariation = ifelse(is.null(list_data$statusDetails[[i]]$arrival$actualTime$inGateVaria
      actualInGate = ifelse(is.null(list_data$statusDetails[[i]]$arrival$actualTime$inGate), NA, list_da
      actualOnGround = ifelse(is.null(list_data$statusDetails[[i]]$arrival$actualTime$onGround), NA, lis

      arrTerminal = ifelse(is.null(list_data$statusDetails[[i]]$arrival$actualTerminal), NA, list_data$s
      arrGate = ifelse(is.null(list_data$statusDetails[[i]]$arrival$gate), NA, list_data$statusDetails[

      numStops  = data$segmentInfo.numberOfStops,
      connections = I(list(connections(list_data$segmentInfo.numberOfStops[[i]], list_data$segmentInfo.n
      miles = data$distance.accumulatedGreatCircleMiles
    )



    flight_data <- rbind(flight_data, new_row1)
  }
  # Identify list columns
  list_columns <- sapply(flight_data, is.list)

  # Convert list columns to character vectors
  flight_data[list_columns] <- lapply(flight_data[list_columns], function(x) sapply(x, toString))
  # Append new data to the same CSV file
  write.table(flight_data, "AmericanAirlines2.csv",
              append = TRUE,
              sep = ",",
              col.names = FALSE,
              row.names = FALSE,
              quote = FALSE)
  paging_next <- json$paging$`next`
  return(paging_next)
}


#Function that turns connections into a list to insert into df
connections <- function(stops, stops_list) {
  conn_list <- list()
  if (stops == 0) {
    conn_list <- append(conn_list, "None")
  }
  else {
    for (i in 1:stops) {
      conn_list <- append(conn_list, list(stops_list$station[i]))
    }
  }
  return (conn_list)
}
```

Airport Wait Times

We obtained this data using FORM request on the https://awt.cbp.gov/ website. At first, we searched the page html to find the appropriate ID for Chicago O'Hare Aiport then we specified the date range for wait time collection. This will be performed after the flight API request as this will guide how many days we need to get wait times for.

Implementation Below is our R code, which submits a HTML form and returns a HTML table to clean into a data frame.

```r
library(httr) # httr is organised around the six most common http verbs: GET(), PATCH(), POST(), HEAD()
library(rvest) # Easily Harvest (Scrape) Web Pages
library(tidyverse)
library(stringr)
library(jsonlite)


#Found in the HTML of the website
ord_id <- 'ORD'
t5_id <- 'A392'
#form request
awt_session <- session("https://awt.cbp.gov/")
form <- html_form(awt_session)[[1]]
form_filled <- html_form_set(form, Id = 'ORD', FromDate = '08/01/2024', ToDate = '09/01/2024')
answer <- session_submit(awt_session, form_filled)
answer
tbl <- data.frame(html_table(answer)[[1]])
for (i in (5:20)){
  colnames(tbl)[i] <- paste0(paste0(tbl[1, i], " "), gsub("\\s+", "_", tbl[2, i]))
}

colnames(tbl)[c(5,7)] <- c('US_Average_Wait_Time', 'Non_US_Average_Wait_Times')
tbl <- tbl %>% select(-All.12) %>% slice(-c(1, 2))
write.csv(tbl, 'waitTimes1.csv', row.names=FALSE)
```

```r
wait_data <- read.csv('waitTimes1.csv')
head(wait_data)
```

```
##   Airport   Terminal     Date        Hour US_Average_Wait_Time
## 1     ORD Terminal 5 8/1/2024 0000 - 0100                   20
## 2     ORD Terminal 5 8/1/2024 0100 - 0200                   16
## 3     ORD Terminal 5 8/1/2024 0200 - 0300                   14
## 4     ORD Terminal 5 8/1/2024 0400 - 0500                   13
## 5     ORD Terminal 5 8/1/2024 0600 - 0700                   34
## 6     ORD Terminal 5 8/1/2024 0700 - 0800                   19
##   U.S..Citizen.Max_Wait_Time Non_US_Average_Wait_Times
## 1                         37                        21
## 2                         31                        18
## 3                         28                        13
## 4                         28                        16
## 5                         67                        39
## 6                         45                        25
##   Non.U.S..Citizen.Max_Wait_Time Wait.Times.Average_Wait_Time
## 1                             38                          21
## 2                             35                          17
## 3                             28                          13
## 4                             27                          15
## 5                             67                          37
## 6                             46                          21
##   Wait.Times.Max_Wait_Time Number.Of.Passengers.Time.Interval.0.15
## 1                       38                                     103
## 2                       35                                     234
## 3                       28                                     123
## 4                       28                                      84
## 5                       67                                     131
## 6                       46                                      88
##   Number.Of.Passengers.Time.Interval.16.30
## 1                                      223
## 2                                      291
## 3                                       98
## 4                                       87
## 5                                       88
## 6                                      107
##   Number.Of.Passengers.Time.Interval.31.45
## 1                                       61
## 2                                       24
## 3                                        0
## 4                                        0
## 5                                      166
## 6                                       59
##   Number.Of.Passengers.Time.Interval.46.60
## 1                                        0
## 2                                        0
## 3                                        0
## 4                                        0
## 5                                      206
## 6                                        2
##   Number.Of.Passengers.Time.Interval.61.90
## 1                                        0
## 2                                        0
## 3                                        0
## 4                                        0
## 5                                       75
## 6                                        0
##   Number.Of.Passengers.Time.Interval.91.120
## 1                                         0
## 2                                         0
## 3                                         0
## 4                                         0
## 5                                         0
## 6                                         0
##   Number.Of.Passengers.Time.Interval.120_plus X.Excluded X.Total X.Flights
## 1                                           0          9     396         2
## 2                                           0         15     564         3
## 3                                           0          7     228         1
## 4                                           0          6     177         1
## 5                                           0         20     686         3
## 6                                          0          7     263          1
```

Data Description

The Form API data from: https://awt.cbp.gov/. With this API, we get a list of wait times for the desired airport within the date range we provide. This dataset includes average and max wait times for US, Non-US citizens, and all people.

The REST API data from: https://rapidapi.com/oag-oag-default/api/flight-info-api Each request (and associated json) had 100 rows. Each row contained 17 elements: carrier, serviceSuffix, flightNumber, flightType, departure, arrival, elapsedTime, cargoTonnage, aircraftType, serviceType, segmentInfo, distance, codeshare, scheduleInstanceKey, statusKey, and statusDetails. Within departure, arrival, distance, and statusDetails were further information about the flight. Cleaning the data required finding the elements that we wanted to use in our analysis and making them into a data frame. Most data points could be put directly into the data frame without manipulation. However, the connecting flights column was created with a function that extracted the connecting flights, if they existed. Also some fields did not exist in the request so we needed validation to enter NA in order for the data frame to still be created.

##Exploratory Analysis

```r
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
```

```r
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

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

```r
library(stringr)
library(hms)
```

```
##
## Attaching package: 'hms'

## The following object is masked from 'package:lubridate':
##
##     hms
```

```r
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```r
library(tidyr)
library(car)
```

```
## Loading required package: carData

##
## Attaching package: 'car'

## The following object is masked from 'package:dplyr':
##
##     recode
```

```r
library(ggplot2)
library(countrycode)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'

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

```r
library(lubridate)
```

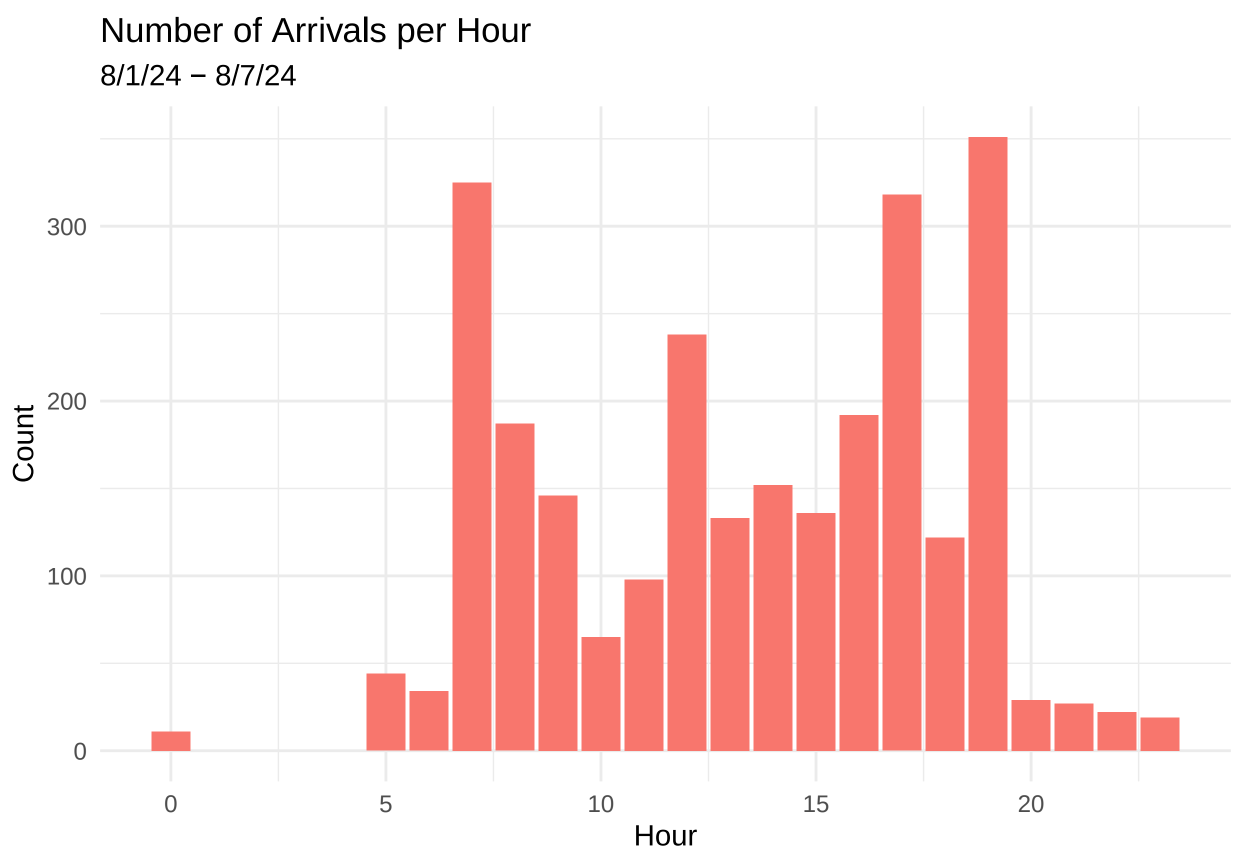
```r
aa <- read.csv('AmericanAirlines_all.csv')
summary(aa)
```

Summary Statistics for Flights

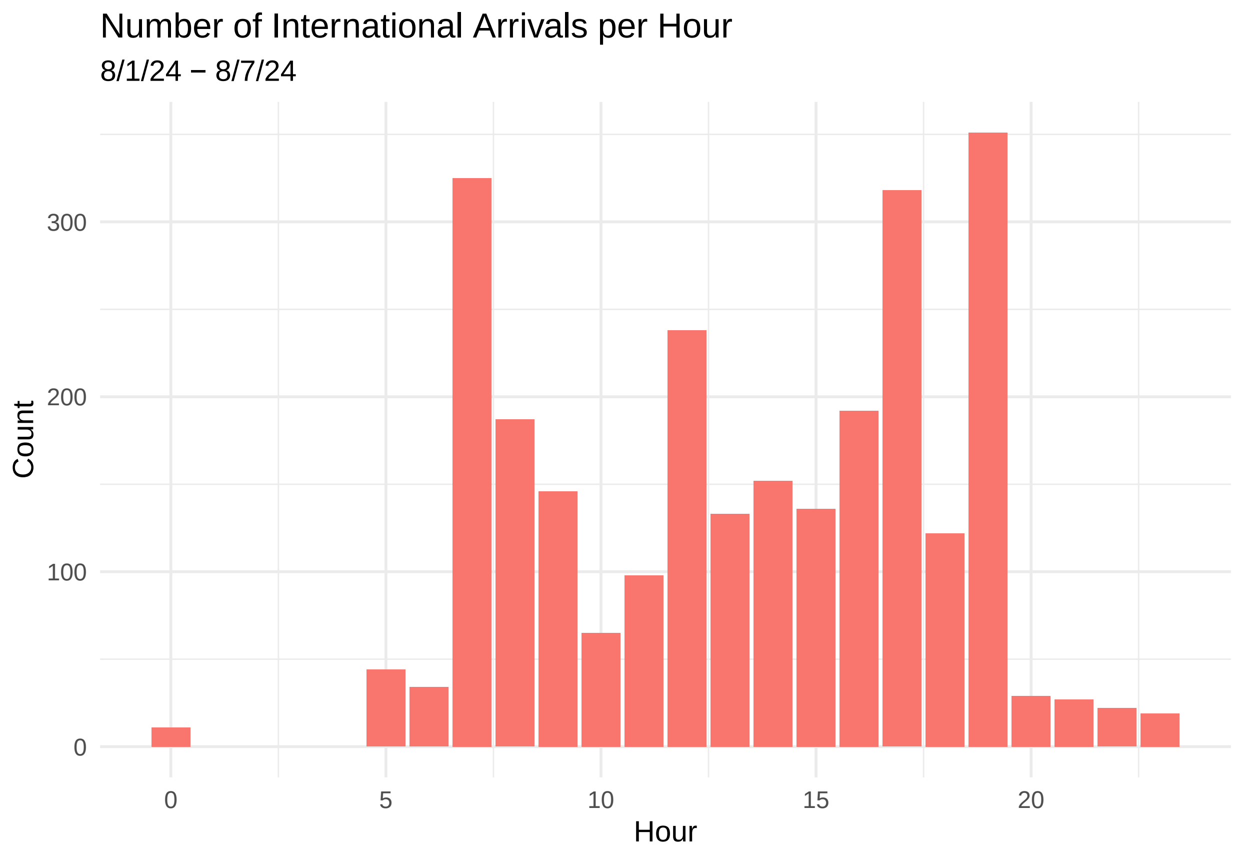
```
##        X           airline           timeOfFlight   deptAirport
##  Min.   :   1   Length:2649        Min.   : 57.0   Length:2649
##  1st Qu.: 663   Class :character   1st Qu.: 92.0   Class :character
##  Median :1325   Mode  :character   Median :132.0   Mode  :character
##  Mean   :1325                      Mean   :158.4
##  3rd Qu.:1987                      3rd Qu.:175.0
##  Max.   :2649                      Max.   :900.0
##   arrAirport           arrDay             arrTime           deptCountry
##  Length:2649        Length:2649        Length:2649        Length:2649
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
```

```
##
##
##
##   flightNumber  estimatedOutGateVariation estimatedOutGate   estimatedOffGround
##  Min.   :  12   Length:2649               Length:2649        Length:2649
##  1st Qu.:1805   Class :character          Class :character   Class :character
##  Median :3478   Mode  :character          Mode  :character   Mode  :character
##  Mean   :3488
##  3rd Qu.:5887
##  Max.   :9240
##  actualOutGateVariation actualOutGate      actualOffGround
##  Length:2649            Length:2649        Length:2649
##  Class :character       Class :character   Class :character
##  Mode  :character       Mode  :character   Mode  :character
##
##
##
##  estimatedInGateVariation estimatedInGate    estimatedOnGround
##  Length:2649              Length:2649        Length:2649
##  Class :character         Class :character   Class :character
##  Mode  :character         Mode  :character   Mode  :character
##
##
##
##  actualInGateVariation actualInGate       actualOnGround     arrTerminal
##  Length:2649           Length:2649        Length:2649        Length:2649
##  Class :character      Class :character   Class :character   Class :character
##  Mode  :character      Mode  :character   Mode  :character   Mode  :character
##
##
##
##    arrGate             numStops connections            miles
##  Length:2649        Min.   :0   Length:2649        Min.   :  66.93
##  Class :character   1st Qu.:0   Class :character   1st Qu.: 298.04
##  Mode  :character   Median :0   Mode  :character   Median : 606.66
##                     Mean   :0                      Mean   : 863.54
##                     3rd Qu.:0                      3rd Qu.: 975.12
##                     Max.   :0                      Max.   :7119.65
```

```r
aa$CountryName <- countrycode(aa$deptCountry, origin = "iso2c", destination = "country.name")


# Plot of most common hours
aa$arrHour <- as.numeric(format(strptime(aa$arrTime, format = "%H:%M"), "%H"))
aa$international <- ifelse(aa$deptAirport !=  'US', 1,0)
arrivals <- aa %>%
  group_by(arrHour) %>%
  summarize(Count = n())


ggplot(aa, aes(x=arrHour)) + geom_bar(fill = '#F8766D') +
  labs(title = 'Number of Arrivals per Hour',subtitle = '8/1/24 - 8/7/24', x = 'Hour', y = 'Count') + t
```

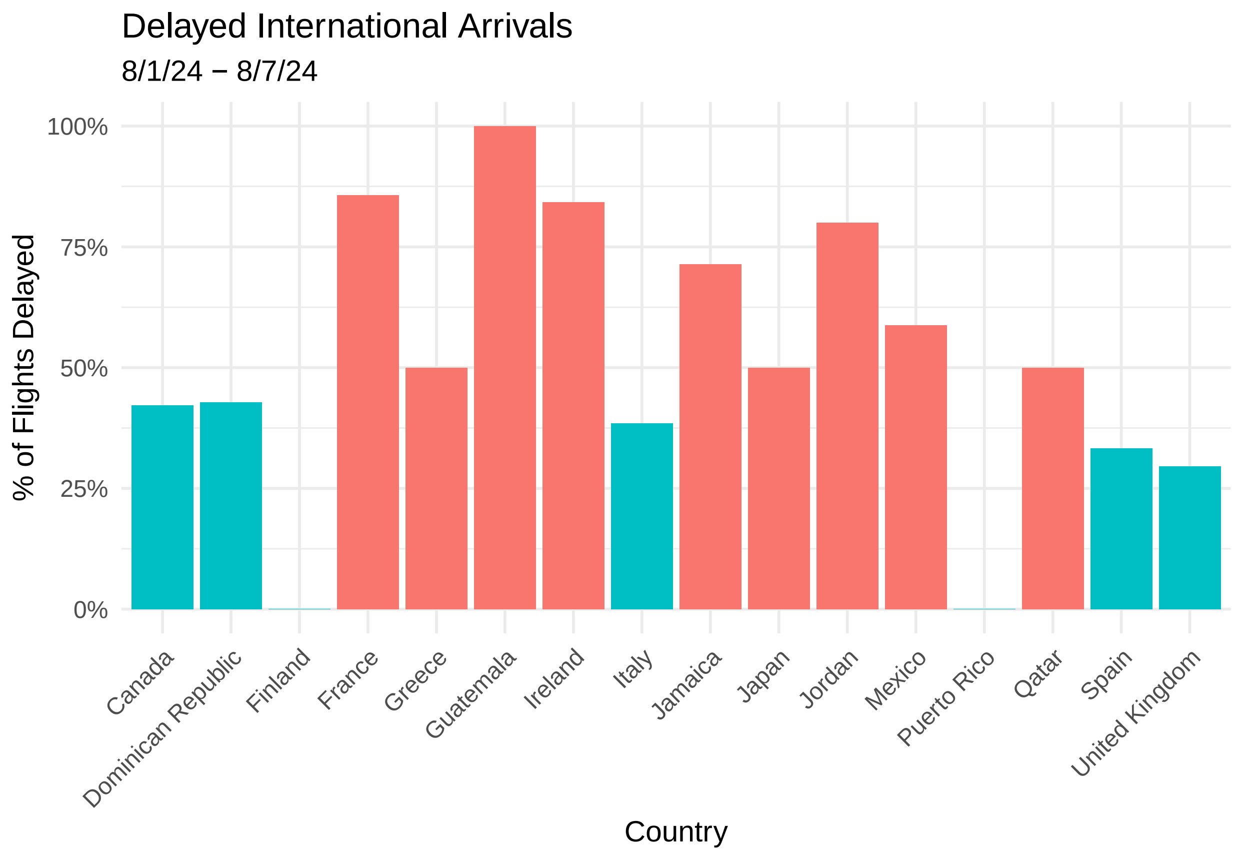


```r
ggplot(aa[aa$international==1,], aes(x=arrHour)) + geom_bar(fill = '#F8766D') +
  labs(title = 'Number of International Arrivals per Hour',subtitle = '8/1/24 - 8/7/24', x = 'Hour', y =
```



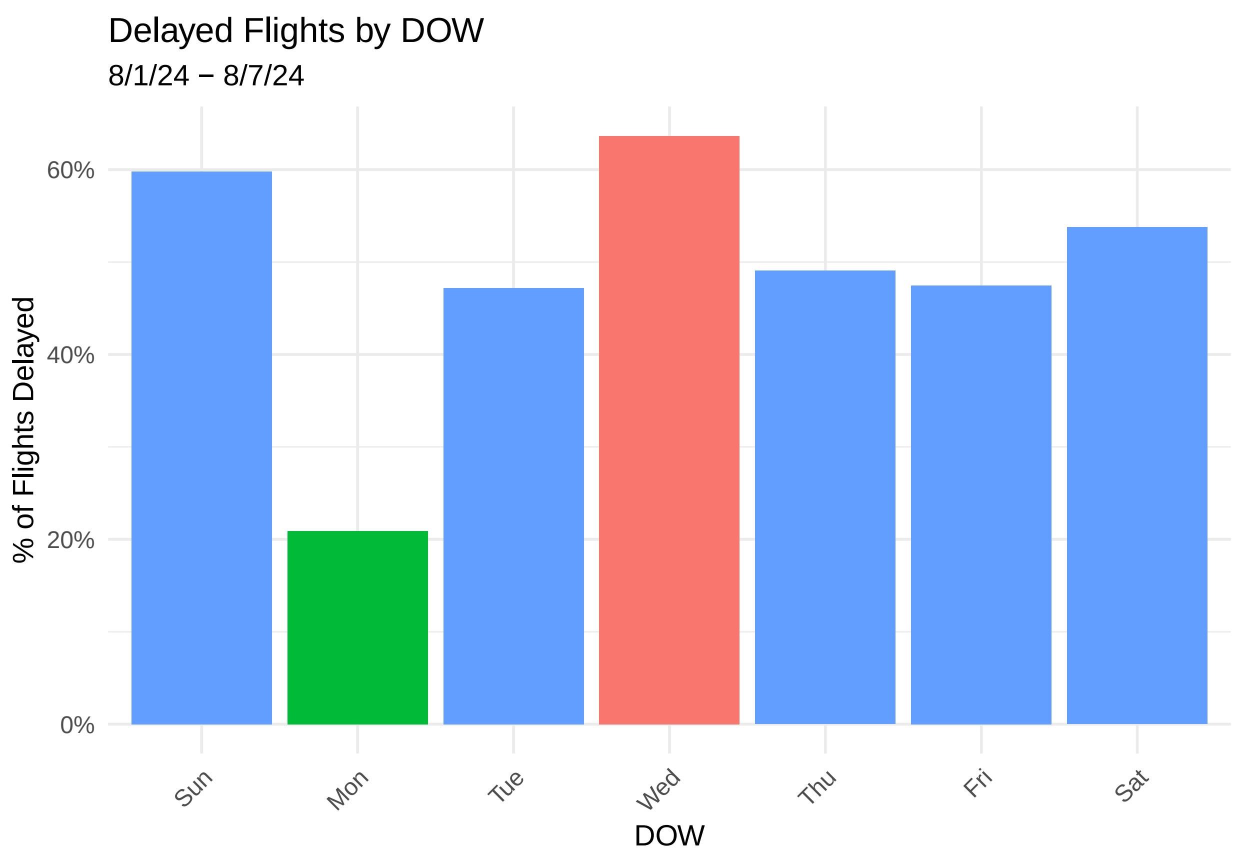
The most common hours for both domestic and flights to arrive are around the middle of the day. However, domestic flights peak earlier than international flights starting to pick up at around 7am. This makes sense as to account for time difference, very few international flights (from American Airlines which are generally coming from Europe) land early in the morning in the US.

```
international_delay <- aa[aa$deptCountry != 'US',] %>% mutate(delayed = ifelse(grepl('^-', actualInGate
international_delay$color <- ifelse(international_delay$over, '#00BA38',"#F8766D")

dow_delay <- aa
dow_delay$weekday <- wday(aa$arrDay, label = TRUE)
dow_delay <- dow_delay%>% mutate(delayed = ifelse(grepl('^-', actualInGateVariation), 1, 0)) %>% select

# Plot of delays in international airports
ggplot(international_delay, aes(x = CountryName, y = del, fill =color)) +
  geom_bar(stat = "identity") +
  guides(x = guide_axis(angle = 45)) +
  labs(title = 'Delayed International Arrivals', subtitle = '8/1/24 - 8/7/24', x = 'Country', y = '% of
  scale_y_continuous(labels = scales::percent) +
    guides(fill = "none") +  # Remove the legend
  theme_minimal()
```



When looking at delays by country, there is no significant outliers for countries that are consistently delayed. However, we can see that in this period there were no delayed flights from PR or Finland. This may be useful going forward to use country as a feature to predict delay. However, it would be more useful if we could compare different airlines per country as then people could make an informed choice on which airline to choose based on delays.

```r
# Plot of delays by dow
dow_delay$color <- ifelse(dow_delay$weekday == 'Mon', "green", ifelse(dow_delay$weekday == 'Wed', "blue
ggplot(dow_delay, aes(x = weekday, y = del, fill = color)) +
  geom_bar(stat = "identity") +
  guides(x = guide_axis(angle = 45)) +
  labs(title = 'Delayed Flights by DOW', subtitle = '8/1/24 - 8/7/24', x = 'DOW', y = '% of Flights Dela
  scale_y_continuous(labels = scales::percent) +
  guides(fill = "none") +  # Remove the legend
  theme_minimal()
```



For all flights, we can see that Wednesday experiences the most delays and Monday seems to have significantly more on-time flights than the others days. This could be because it is the start of the week and there are no delays from previous days that have a knock-on effect at this point.

```
wait <- read.csv('waitTimes1.csv')
summary(wait)
```

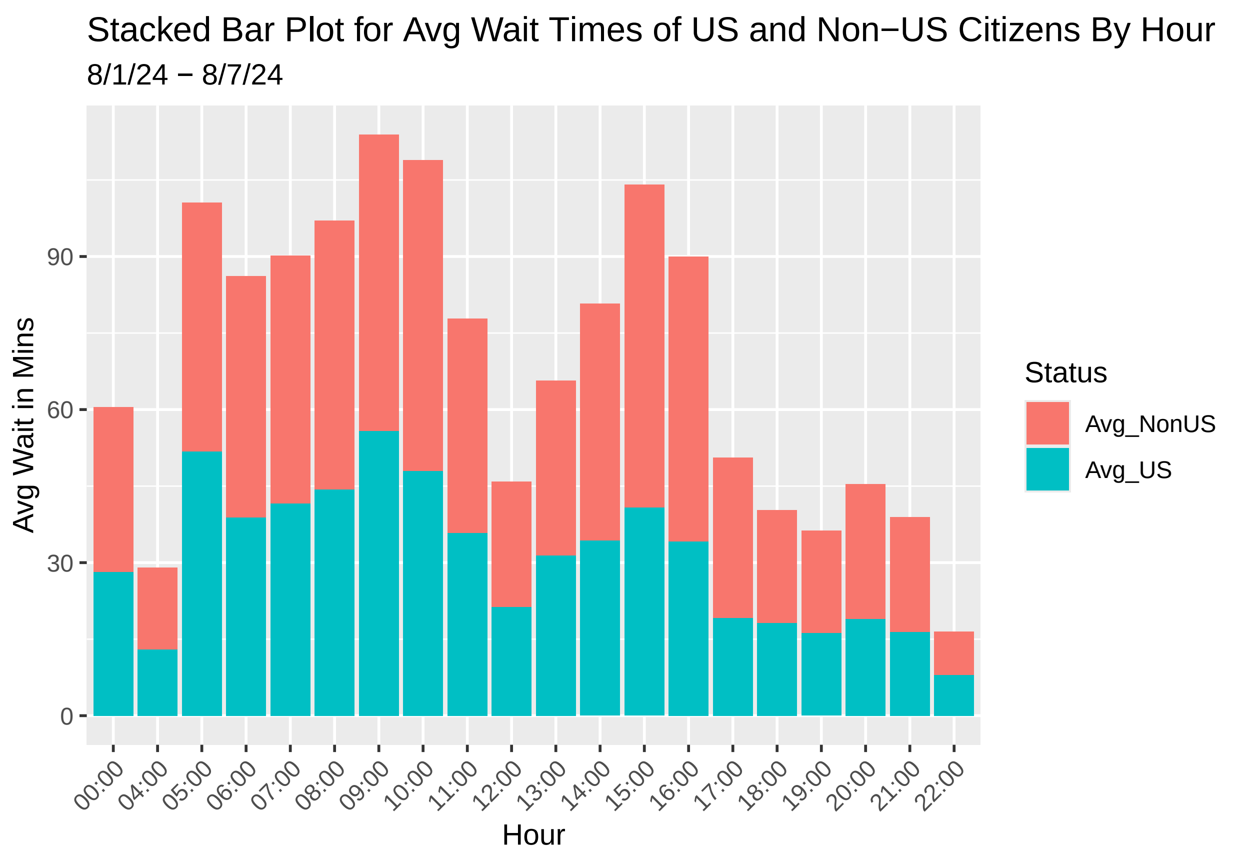
Summary Statistics for Wait Times

```
##    Airport            Terminal             Date               Hour          
##  Length:640         Length:640         Length:640         Length:640        
##  Class :character   Class :character   Class :character   Class :character  
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character  
##                                                                             
##                                                                             
##                                                                             
##  US_Average_Wait_Time U.S..Citizen.Max_Wait_Time Non_US_Average_Wait_Times
##  Min.   :  4.00       Min.   : 11.00             Min.   :  5.00           
##  1st Qu.: 15.00       1st Qu.: 38.75             1st Qu.: 18.00           
##  Median : 23.00       Median : 57.00             Median : 32.00           
##  Mean   : 28.13       Mean   : 65.20             Mean   : 37.85           
##  3rd Qu.: 36.00       3rd Qu.: 86.00             3rd Qu.: 52.00           
##  Max.   :120.00       Max.   :188.00             Max.   :129.00           
##  Non.U.S..Citizen.Max_Wait_Time Wait.Times.Average_Wait_Time
##  Min.   : 11.00                 Min.   :  4.00
##  1st Qu.: 41.00                 1st Qu.: 17.00
##  Median : 62.00                 Median : 28.00
##  Mean   : 69.17                 Mean   : 32.42
##  3rd Qu.: 93.00                 3rd Qu.: 43.00
##  Max.   :193.00                 Max.   :123.00
##  Wait.Times.Max_Wait_Time Number.Of.Passengers.Time.Interval.0.15
##  Min.   : 13.00           Min.   :   0.00
##  1st Qu.: 41.00           1st Qu.:  77.75
##  Median : 64.00           Median : 188.00
##  Mean   : 69.89           Mean   : 296.28
##  3rd Qu.: 93.00           3rd Qu.: 419.00
##  Max.   :193.00           Max.   :1894.00
##  Number.Of.Passengers.Time.Interval.16.30
##  Min.   :   0.0
##  1st Qu.:  71.5
##  Median : 145.0
##  Mean   : 249.6
##  3rd Qu.: 350.0
##  Max.   :1259.0
##  Number.Of.Passengers.Time.Interval.31.45
##  Min.   :   0.0
##  1st Qu.:  31.0
##  Median :  91.0
##  Mean   : 165.1
##  3rd Qu.: 223.5
##  Max.   :1154.0
##  Number.Of.Passengers.Time.Interval.46.60
##  Min.   :  0.0
##  1st Qu.:  0.0
##  Median : 43.0
##  Mean   :106.9
##  3rd Qu.:139.0
##  Max.   :842.0
##  Number.Of.Passengers.Time.Interval.61.90
##  Min.   :   0.0
##  1st Qu.:   0.0
##  Median :   3.5
##  Mean   : 103.4
##  3rd Qu.: 121.0
##  Max.   :1264.0
##  Number.Of.Passengers.Time.Interval.91.120
##  Min.   :  0.00
##  1st Qu.:  0.00
##  Median :  0.00
##  Mean   : 38.33
##  3rd Qu.:  3.00
##  Max.   :925.00
##  Number.Of.Passengers.Time.Interval.120_plus   X.Excluded       X.Total
##  Min.   :  0.00                              Min.   :  2.0   Min.   : 103.0
##  1st Qu.:  0.00                              1st Qu.: 10.0   1st Qu.: 363.8
##  Median :  0.00                              Median : 21.0   Median : 706.0
##  Mean   : 11.48                              Mean   : 29.9   Mean   :1001.0
##  3rd Qu.:  0.00                             3rd Qu.: 45.0   3rd Qu.:1521.5
##  Max.   :568.00                             Max.   :113.0   Max.   :3413.0
##    X.Flights
##  Min.   : 1.000
##  1st Qu.: 2.000
##  Median : 3.000
##  Mean   : 4.427
##  3rd Qu.: 7.000
##  Max.   :14.000
```

```
per_date <- wait %>% group_by(Date) %>%
  summarise(Avg_US =  mean(US_Average_Wait_Time), Avg_NonUS = mean(Non_US_Average_Wait_Times)) %>%
  pivot_longer(cols = c(Avg_US, Avg_NonUS), names_to = 'Status')
per_date$weekday <- wday(as.Date(per_date$Date), label=TRUE)

per_date_max <- wait %>% group_by(Date) %>%
  summarise(Max_US =  max(U.S..Citizen.Max_Wait_Time),
            Max_NonUS = max(Non.U.S..Citizen.Max_Wait_Time)) %>%
  pivot_longer(cols = c(Max_US, Max_NonUS), names_to = 'Status')
per_date_max$weekday <- wday(as.Date(per_date_max$Date), label = TRUE)


#per_hour_wait <- wait[,c(5, 7, 23)]
# Merged dataframe
result <- read.csv('AA_wait_merged.csv')
per_hour_wait <- result %>% group_by(arrStart) %>% summarise(Avg_US =  mean(US_Average_Wait_Time), Avg_

stacked_hourly <- ggplot(per_hour_wait, aes(x = arrStart, y = value, fill = Status)) +
  geom_bar(stat = "identity") +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1)) +
  labs(title = "Stacked Bar Plot for Avg Wait Times of US and Non-US Citizens By Hour",
       subtitle = '8/1/24 - 8/7/24',
       x = "Hour",
       y = "Avg Wait in Mins",)
  theme_minimal()
```

```
## List of 136
##  $ line                            :List of 6
##   ..$ colour       : chr "black"
##   ..$ linewidth    : num 0.5
##   ..$ linetype     : num 1
##   ..$ lineend      : chr "butt"
##   ..$ arrow        : logi FALSE
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_line" "element"
##  $ rect                            :List of 5
##   ..$ fill         : chr "white"
##   ..$ colour       : chr "black"
##   ..$ linewidth    : num 0.5
##   ..$ linetype     : num 1
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_rect" "element"
##  $ text                            :List of 11
##   ..$ family       : chr ""
##   ..$ face         : chr "plain"
##   ..$ colour       : chr "black"
##   ..$ size         : num 11
##   ..$ hjust        : num 0.5
##   ..$ vjust        : num 0.5
##   ..$ angle        : num 0
##   ..$ lineheight   : num 0.9
##   ..$ margin       : 'margin' num [1:4] 0points 0points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : logi FALSE
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ title                           : NULL
##  $ aspect.ratio                    : NULL
##  $ axis.title                      : NULL
##  $ axis.title.x                    :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 1
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 2.75points 0points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.title.x.top                :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 0
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 2.75points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.title.x.bottom             : NULL
##  $ axis.title.y                    :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 1
##   ..$ angle        : num 90
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 2.75points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.title.y.left               : NULL
##  $ axis.title.y.right              :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 1
##   ..$ angle        : num -90
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 0points 2.75points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text                       :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : chr "grey30"
##   ..$ size         : 'rel' num 0.8
##   ..$ hjust        : NULL
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text.x                     :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 1
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 2.2points 0points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text.x.top                 :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 0
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 2.2points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text.x.bottom              : NULL
##  $ axis.text.y                     :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : num 1
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 2.2points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text.y.left                : NULL
##  $ axis.text.y.right               :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : num 0
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 0points 2.2points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text.theta                 : NULL
##  $ axis.text.r                     :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : num 0.5
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 2.2points 0points 2.2points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.ticks                      : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ axis.ticks.x                    : NULL
##  $ axis.ticks.x.top                : NULL
##  $ axis.ticks.x.bottom             : NULL
##  $ axis.ticks.y                    : NULL
##  $ axis.ticks.y.left               : NULL
##  $ axis.ticks.y.right              : NULL
##  $ axis.ticks.theta                : NULL
##  $ axis.ticks.r                    : NULL
##  $ axis.minor.ticks.x.top          : NULL
##  $ axis.minor.ticks.x.bottom       : NULL
##  $ axis.minor.ticks.y.left         : NULL
##  $ axis.minor.ticks.y.right        : NULL
##  $ axis.minor.ticks.theta          : NULL
##  $ axis.minor.ticks.r              : NULL
##  $ axis.ticks.length               : 'simpleUnit' num 2.75points
##   ..- attr(*, "unit")= int 8
##  $ axis.ticks.length.x             : NULL
##  $ axis.ticks.length.x.top         : NULL
##  $ axis.ticks.length.x.bottom      : NULL
##  $ axis.ticks.length.y             : NULL
##  $ axis.ticks.length.y.left        : NULL
##  $ axis.ticks.length.y.right       : NULL
##  $ axis.ticks.length.theta         : NULL
##  $ axis.ticks.length.r             : NULL
##  $ axis.minor.ticks.length         : 'rel' num 0.75
##  $ axis.minor.ticks.length.x       : NULL
##  $ axis.minor.ticks.length.x.top   : NULL
##  $ axis.minor.ticks.length.x.bottom: NULL
##  $ axis.minor.ticks.length.y       : NULL
##  $ axis.minor.ticks.length.y.left  : NULL
##  $ axis.minor.ticks.length.y.right : NULL
##  $ axis.minor.ticks.length.theta   : NULL
##  $ axis.minor.ticks.length.r       : NULL
##  $ axis.line                       : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ axis.line.x                     : NULL
##  $ axis.line.x.top                 : NULL
##  $ axis.line.x.bottom              : NULL
##  $ axis.line.y                     : NULL
##  $ axis.line.y.left                : NULL
##  $ axis.line.y.right               : NULL
##  $ axis.line.theta                 : NULL
##  $ axis.line.r                     : NULL
##  $ legend.background               : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ legend.margin                   : 'margin' num [1:4] 5.5points 5.5points 5.5points 5.5points
##   ..- attr(*, "unit")= int 8
##  $ legend.spacing                  : 'simpleUnit' num 11points
##   ..- attr(*, "unit")= int 8
##  $ legend.spacing.x                : NULL
##  $ legend.spacing.y                : NULL
##  $ legend.key                      : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ legend.key.size                 : 'simpleUnit' num 1.2lines
##   ..- attr(*, "unit")= int 3
##  $ legend.key.height               : NULL
##  $ legend.key.width                : NULL
##  $ legend.key.spacing              : 'simpleUnit' num 5.5points
##   ..- attr(*, "unit")= int 8
##  $ legend.key.spacing.x            : NULL
##  $ legend.key.spacing.y            : NULL
##  $ legend.frame                    : NULL
##  $ legend.ticks                    : NULL
##  $ legend.ticks.length             : 'rel' num 0.2
##  $ legend.axis.line                : NULL
##  $ legend.text                     :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : 'rel' num 0.8
##   ..$ hjust        : NULL
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ legend.text.position            : NULL
##  $ legend.title                    :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : num 0
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ legend.title.position           : NULL
##  $ legend.position                 : chr "right"
##  $ legend.position.inside          : NULL
##  $ legend.direction                : NULL
##  $ legend.byrow                    : NULL
##  $ legend.justification            : chr "center"
##  $ legend.justification.top        : NULL
##  $ legend.justification.bottom     : NULL
##  $ legend.justification.left       : NULL
##  $ legend.justification.right      : NULL
##  $ legend.justification.inside     : NULL
##  $ legend.location                 : NULL
##  $ legend.box                      : NULL
##  $ legend.box.just                 : NULL
##  $ legend.box.margin               : 'margin' num [1:4] 0cm 0cm 0cm 0cm
##   ..- attr(*, "unit")= int 1
##  $ legend.box.background           : list()
##    ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##   $ legend.box.spacing             : 'simpleUnit' num 11points
##    ..- attr(*, "unit")= int 8
##    [list output truncated]
##   - attr(*, "class")= chr [1:2] "theme" "gg"
##   - attr(*, "complete")= logi TRUE
##   - attr(*, "validate")= logi TRUE
```

```
stacked_hourly
```



If we look at the average wait time per hour, there is not much difference between US and Non-US Citizens. However, you can see between 3-4am and 5-11pm it is significantly faster compared to the later hours between 11pm and 2am and the middle of the day between 5am and 4pm. This doesn't really make sense as our flight data indicated that all international flights are landing after 5pm so that would be when international arrivals is busiest.

There are two explanations for why this data is not as expected. 1. We only used American Airlines, which flies to a lot of international destination but not all. Specifically, not many countries in the Pacific. Most international flights from Europe will land in the late afternoon/evening in the US because of time difference. So there could be a large number of flights that are landing in the middle of the day from other airlines that are increasing wait times but are not showing up in our analysis.

2. Number of flights might not be exactly proportional to wait times as certain international citizens take longer to get through customs than others. For example, a large number of flights from Ireland may not take too long to get through border control as they have pre-clearance in Ireland. However, a singular flight from Australia or China might take a lot longer to get through customs as the US has different/more strict immigration policies.

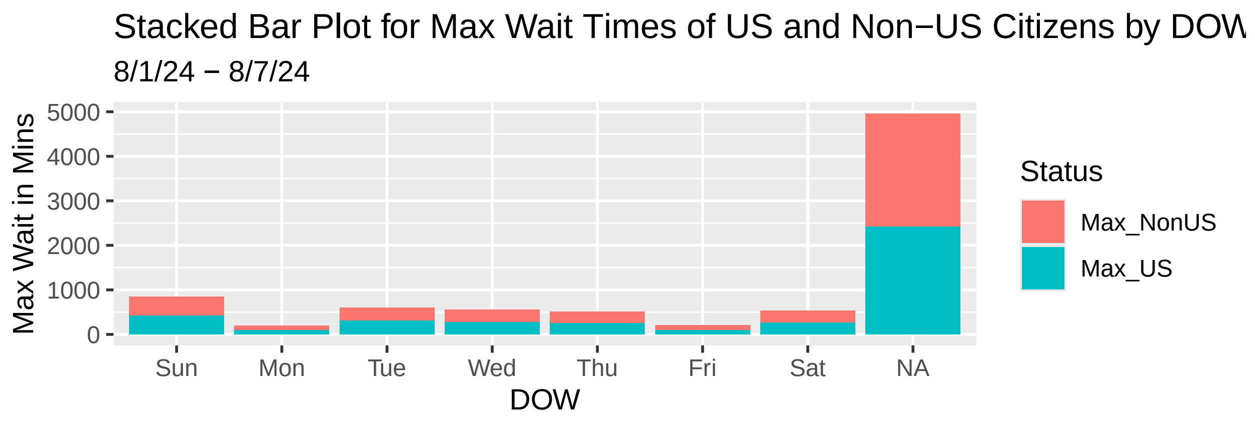
```r
# Create the stacked bar plot
stacked_avg <- ggplot(per_date, aes(x = weekday, y = value, fill = Status)) +
  geom_bar(stat = "identity") +
  labs(title = "Stacked Bar Plot for Avg Wait Times of US and Non-US Citizens by DOW",
       subtitle = '8/1/24 - 8/7/24',
       x = "DOW",
       y = "Average Wait in Mins",)
  theme_minimal()
```

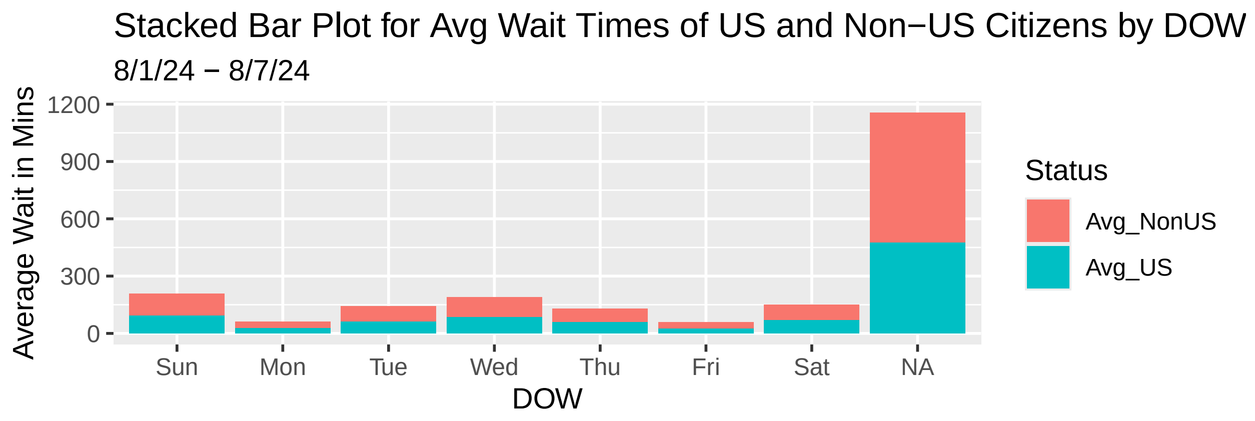
```
## List of 136
##  $ line                            :List of 6
##   ..$ colour       : chr "black"
##   ..$ linewidth    : num 0.5
##   ..$ linetype     : num 1
##   ..$ lineend      : chr "butt"
##   ..$ arrow        : logi FALSE
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_line" "element"
##  $ rect                            :List of 5
##   ..$ fill         : chr "white"
##   ..$ colour       : chr "black"
##   ..$ linewidth    : num 0.5
##   ..$ linetype     : num 1
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_rect" "element"
##  $ text                            :List of 11
##   ..$ family       : chr ""
##   ..$ face         : chr "plain"
##   ..$ colour       : chr "black"
##   ..$ size         : num 11
##   ..$ hjust        : num 0.5
##   ..$ vjust        : num 0.5
##   ..$ angle        : num 0
##   ..$ lineheight   : num 0.9
##   ..$ margin       : 'margin' num [1:4] 0points 0points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : logi FALSE
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ title                           : NULL
##  $ aspect.ratio                    : NULL
##  $ axis.title                      : NULL
##  $ axis.title.x                    :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 1
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 2.75points 0points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.title.x.top               :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 0
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 2.75points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.title.x.bottom            : NULL
##  $ axis.title.y                   :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 1
##   ..$ angle        : num 90
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 2.75points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.title.y.left              : NULL
##  $ axis.title.y.right             :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 1
##   ..$ angle        : num -90
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 0points 2.75points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text                      :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : chr "grey30"
##   ..$ size         : 'rel' num 0.8
##   ..$ hjust        : NULL
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text.x                    :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 1
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 2.2points 0points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text.x.top                :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 0
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 2.2points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text.x.bottom             : NULL
##  $ axis.text.y                    :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : num 1
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 2.2points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text.y.left               : NULL
##  $ axis.text.y.right              :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : num 0
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 0points 2.2points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text.theta                 : NULL
##  $ axis.text.r                     :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : num 0.5
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 2.2points 0points 2.2points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.ticks                      : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ axis.ticks.x                    : NULL
##  $ axis.ticks.x.top                : NULL
##  $ axis.ticks.x.bottom             : NULL
##  $ axis.ticks.y                    : NULL
##  $ axis.ticks.y.left               : NULL
##  $ axis.ticks.y.right              : NULL
##  $ axis.ticks.theta                : NULL
##  $ axis.ticks.r                    : NULL
##  $ axis.minor.ticks.x.top          : NULL
##  $ axis.minor.ticks.x.bottom       : NULL
##  $ axis.minor.ticks.y.left         : NULL
##  $ axis.minor.ticks.y.right        : NULL
##  $ axis.minor.ticks.theta          : NULL
##  $ axis.minor.ticks.r              : NULL
##  $ axis.ticks.length               : 'simpleUnit' num 2.75points
##   ..- attr(*, "unit")= int 8
##  $ axis.ticks.length.x             : NULL
##  $ axis.ticks.length.x.top         : NULL
##  $ axis.ticks.length.x.bottom      : NULL
##  $ axis.ticks.length.y             : NULL
##  $ axis.ticks.length.y.left        : NULL
##  $ axis.ticks.length.y.right       : NULL
##  $ axis.ticks.length.theta         : NULL
##  $ axis.ticks.length.r             : NULL
##  $ axis.minor.ticks.length         : 'rel' num 0.75
##  $ axis.minor.ticks.length.x       : NULL
##  $ axis.minor.ticks.length.x.top   : NULL
##  $ axis.minor.ticks.length.x.bottom: NULL
##  $ axis.minor.ticks.length.y       : NULL
##  $ axis.minor.ticks.length.y.left  : NULL
##  $ axis.minor.ticks.length.y.right : NULL
##  $ axis.minor.ticks.length.theta   : NULL
##  $ axis.minor.ticks.length.r       : NULL
##  $ axis.line                       : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ axis.line.x                     : NULL
##  $ axis.line.x.top                 : NULL
##  $ axis.line.x.bottom              : NULL
##  $ axis.line.y                     : NULL
##  $ axis.line.y.left                : NULL
##  $ axis.line.y.right               : NULL
##  $ axis.line.theta                 : NULL
##  $ axis.line.r                     : NULL
##  $ legend.background               : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ legend.margin                   : 'margin' num [1:4] 5.5points 5.5points 5.5points 5.5points
##   ..- attr(*, "unit")= int 8
##  $ legend.spacing                  : 'simpleUnit' num 11points
##   ..- attr(*, "unit")= int 8
##  $ legend.spacing.x                : NULL
##  $ legend.spacing.y                : NULL
##  $ legend.key                      : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ legend.key.size                 : 'simpleUnit' num 1.2lines
##   ..- attr(*, "unit")= int 3
##  $ legend.key.height               : NULL
##  $ legend.key.width                : NULL
##  $ legend.key.spacing              : 'simpleUnit' num 5.5points
##   ..- attr(*, "unit")= int 8
##  $ legend.key.spacing.x            : NULL
##  $ legend.key.spacing.y            : NULL
##  $ legend.frame                    : NULL
##  $ legend.ticks                    : NULL
##  $ legend.ticks.length             : 'rel' num 0.2
##  $ legend.axis.line                : NULL
##  $ legend.text                     :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : 'rel' num 0.8
##   ..$ hjust        : NULL
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ legend.text.position            : NULL
##  $ legend.title                    :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : num 0
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ legend.title.position           : NULL
##  $ legend.position                 : chr "right"
##  $ legend.position.inside          : NULL
##  $ legend.direction                : NULL
##  $ legend.byrow                    : NULL
##  $ legend.justification            : chr "center"
##  $ legend.justification.top        : NULL
##  $ legend.justification.bottom     : NULL
##  $ legend.justification.left       : NULL
##  $ legend.justification.right      : NULL
##  $ legend.justification.inside     : NULL
##  $ legend.location                 : NULL
##  $ legend.box                      : NULL
##  $ legend.box.just                 : NULL
##  $ legend.box.margin               : 'margin' num [1:4] 0cm 0cm 0cm 0cm
##   ..- attr(*, "unit")= int 1
##  $ legend.box.background           : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ legend.box.spacing              : 'simpleUnit' num 11points
##   ..- attr(*, "unit")= int 8
##   [list output truncated]
##  - attr(*, "class")= chr [1:2] "theme" "gg"
##  - attr(*, "complete")= logi TRUE
##  - attr(*, "validate")= logi TRUE
```

```r
stacked_max <- ggplot(per_date_max, aes(x = weekday, y = value, fill = Status)) +
  geom_bar(stat = "identity") +
  labs(title = "Stacked Bar Plot for Max Wait Times of US and Non-US Citizens by DOW",
       subtitle = '8/1/24 - 8/7/24',
       x = "DOW",
       y = "Max Wait in Mins",)
  theme_minimal()
```

```
## List of 136
##  $ line                            :List of 6
##   ..$ colour       : chr "black"
##   ..$ linewidth    : num 0.5
##   ..$ linetype     : num 1
##   ..$ lineend      : chr "butt"
##   ..$ arrow        : logi FALSE
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_line" "element"
##  $ rect                            :List of 5
##   ..$ fill         : chr "white"
##   ..$ colour       : chr "black"
##   ..$ linewidth    : num 0.5
##   ..$ linetype     : num 1
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_rect" "element"
##  $ text                            :List of 11
##   ..$ family       : chr ""
##   ..$ face         : chr "plain"
##   ..$ colour       : chr "black"
##   ..$ size         : num 11
##   ..$ hjust        : num 0.5
##   ..$ vjust        : num 0.5
##   ..$ angle        : num 0
##   ..$ lineheight   : num 0.9
##   ..$ margin       : 'margin' num [1:4] 0points 0points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : logi FALSE
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ title                           : NULL
##  $ aspect.ratio                    : NULL
##  $ axis.title                      : NULL
##  $ axis.title.x                    :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 1
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 2.75points 0points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.title.x.top                :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 0
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 2.75points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.title.x.bottom             : NULL
##  $ axis.title.y                    :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 1
##   ..$ angle        : num 90
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 2.75points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.title.y.left               : NULL
##  $ axis.title.y.right              :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 1
##   ..$ angle        : num -90
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 0points 2.75points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text                       :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : chr "grey30"
##   ..$ size         : 'rel' num 0.8
##   ..$ hjust        : NULL
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text.x                     :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 1
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 2.2points 0points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text.x.top                :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 0
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 2.2points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text.x.bottom             : NULL
##  $ axis.text.y                    :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : num 1
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 2.2points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text.y.left               : NULL
##  $ axis.text.y.right              :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : num 0
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 0points 2.2points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text.theta                : NULL
##  $ axis.text.r                    :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : num 0.5
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 2.2points 0points 2.2points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.ticks                     : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ axis.ticks.x                   : NULL
##  $ axis.ticks.x.top               : NULL
##  $ axis.ticks.x.bottom            : NULL
##  $ axis.ticks.y                   : NULL
##  $ axis.ticks.y.left              : NULL
##  $ axis.ticks.y.right             : NULL
##  $ axis.ticks.theta               : NULL
##  $ axis.ticks.r                   : NULL
##  $ axis.minor.ticks.x.top         : NULL
##  $ axis.minor.ticks.x.bottom      : NULL
##  $ axis.minor.ticks.y.left        : NULL
##  $ axis.minor.ticks.y.right       : NULL
##  $ axis.minor.ticks.theta         : NULL
##  $ axis.minor.ticks.r             : NULL
##  $ axis.ticks.length              : 'simpleUnit' num 2.75points
##   ..- attr(*, "unit")= int 8
##  $ axis.ticks.length.x            : NULL
##  $ axis.ticks.length.x.top        : NULL
##  $ axis.ticks.length.x.bottom     : NULL
##  $ axis.ticks.length.y            : NULL
##  $ axis.ticks.length.y.left       : NULL
##  $ axis.ticks.length.y.right      : NULL
##  $ axis.ticks.length.theta        : NULL
##  $ axis.ticks.length.r            : NULL
##  $ axis.minor.ticks.length        : 'rel' num 0.75
##  $ axis.minor.ticks.length.x      : NULL
##  $ axis.minor.ticks.length.x.top  : NULL
##  $ axis.minor.ticks.length.x.bottom: NULL
##  $ axis.minor.ticks.length.y      : NULL
##  $ axis.minor.ticks.length.y.left : NULL
##  $ axis.minor.ticks.length.y.right : NULL
##  $ axis.minor.ticks.length.theta  : NULL
##  $ axis.minor.ticks.length.r      : NULL
##  $ axis.line                      : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ axis.line.x                    : NULL
##  $ axis.line.x.top                : NULL
##  $ axis.line.x.bottom             : NULL
##  $ axis.line.y                    : NULL
##  $ axis.line.y.left               : NULL
##  $ axis.line.y.right              : NULL
##  $ axis.line.theta                : NULL
##  $ axis.line.r                    : NULL
##  $ legend.background              : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ legend.margin                  : 'margin' num [1:4] 5.5points 5.5points 5.5points 5.5points
##   ..- attr(*, "unit")= int 8
##  $ legend.spacing                  : 'simpleUnit' num 11points
##   ..- attr(*, "unit")= int 8
##  $ legend.spacing.x                : NULL
##  $ legend.spacing.y                : NULL
##  $ legend.key                      : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ legend.key.size                 : 'simpleUnit' num 1.2lines
##   ..- attr(*, "unit")= int 3
##  $ legend.key.height               : NULL
##  $ legend.key.width                : NULL
##  $ legend.key.spacing              : 'simpleUnit' num 5.5points
##   ..- attr(*, "unit")= int 8
##  $ legend.key.spacing.x            : NULL
##  $ legend.key.spacing.y            : NULL
##  $ legend.frame                    : NULL
##  $ legend.ticks                    : NULL
##  $ legend.ticks.length             : 'rel' num 0.2
##  $ legend.axis.line                : NULL
##  $ legend.text                     :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : 'rel' num 0.8
##   ..$ hjust        : NULL
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ legend.text.position            : NULL
##  $ legend.title                    :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : num 0
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ legend.title.position           : NULL
##  $ legend.position                 : chr "right"
##  $ legend.position.inside          : NULL
##  $ legend.direction                : NULL
##  $ legend.byrow                    : NULL
##  $ legend.justification            : chr "center"
##  $ legend.justification.top        : NULL
##  $ legend.justification.bottom     : NULL
##  $ legend.justification.left       : NULL
##  $ legend.justification.right      : NULL
##  $ legend.justification.inside     : NULL
##  $ legend.location                 : NULL
##  $ legend.box                      : NULL
##  $ legend.box.just                 : NULL
##  $ legend.box.margin               : 'margin' num [1:4] 0cm 0cm 0cm 0cm
##   ..- attr(*, "unit")= int 1
##  $ legend.box.background           : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ legend.box.spacing              : 'simpleUnit' num 11points
##   ..- attr(*, "unit")= int 8
##   [list output truncated]
##  - attr(*, "class")= chr [1:2] "theme" "gg"
##  - attr(*, "complete")= logi TRUE
##  - attr(*, "validate")= logi TRUE
```

```
grid.arrange(stacked_max, stacked_avg, nrow =2)
```





You can see there is not much difference in Max Wait Times but for Average Wait Times, they are slightly longer for Non-US citizens, which is expected. In addition, Wednesday and Monday seem to be the busiest and Sunday is the quietest for arriving in O'Hare.

##Discussion If you are most concerned about flight delay, the best day to fly would be Monday as that had a significantly smaller % of delayed flights and the worst days would be Sunday or Wednesday. This is most useful for domestic flyers as that is the majority of our data and they do not have to take into account wait times in customs.

However, if you are flying internationally, you want to make sure your flight lands after 4pm to ensure the shortest wait times in customs. Furthermore, for flights from Puerto Rico and Finland, American Airlines seems like a reliable/on time service to choose but for some other countries, you may want to consider another airline.

Data Manipulation

Merging the Dataframes From our two APIs, we had a dataframe of wait times (by hour) and a week of American Airlines flights into Chicago O'Hare. To merge the two dataframes, we had to find the row within the wait times dataframe where the arrival time (of the AA flight) fit into the hour-long windows of the wait times dataframe.

We began by cleaning both the AA and wait dataframes. - Converting the date column to a date - Splitting the hour-long windows into start and end hours (ex: 0000-0100 into 00:00, 01:00) - Converting all times into total minutes for easy comparison.

Here is the code for the merge.

```r
result <- data.frame()
for (i in 1:nrow(aa)) {
  df1 <- aa[i,]
  matched_row <- wait %>%
    filter(Date == df1$arrDay,
           arrStartMins <= df1$arrMinutes,
           arrEndMins >= df1$arrMinutes)
  combined_row <- df1
  if (nrow(matched_row) > 0) {
    for (j in 5:ncol(matched_row)) {
      col_name <- colnames(matched_row)[j]
      combined_row[[col_name]] <- matched_row[1, j]
    }
  result <- rbind(result, combined_row)
  }
}
```

Within this dataframe merge, NAs presented some challenges. Whenever an NA was present in the current row of the AA flights dataframe, all the new values to be merged resulted in NA when using column bind (cbind). Our solution was to introduce another if statement that binded these two pieces to 'combined_row' separately.

We then saved this final dataframe as a csv file.

```r
result <- read.csv('AA_wait_merged.csv')
head(result)
```

```
##   airline timeOfFlight deptAirport arrAirport     arrDay arrTime deptCountry
## 1      AA          249         LAX        ORD 2024-08-01   00:19          US
## 2      AA          213         PHX        ORD 2024-08-01   00:23          US
## 3      AA          248         SEA        ORD 2024-08-01   05:00          US
## 4      AA          253         SFO        ORD 2024-08-01   06:04          US
## 5      AA          254         LAX        ORD 2024-08-01   06:13          US
## 6      AA           74         ATW        ORD 2024-08-01   06:59          US
##   flightNumber estimatedOutGateVariation    estimatedOutGate estimatedOffGround
## 1          516                  00:16:00 2024-08-01 01:26:00               <NA>
## 2         630                 02:03:00 2024-08-01 03:53:00               <NA>
## 3        2368                 00:05:00 2024-08-01 05:57:00               <NA>
## 4        2113                 00:35:00 2024-08-01 07:26:00               <NA>
## 5        1597                 00:00:00 2024-08-01 06:59:00               <NA>
## 6        6048                 00:00:00 2024-08-01 10:45:00               <NA>
##   actualOutGateVariation       actualOutGate     actualOffGround
## 1               00:24:00 2024-08-01 01:34:00 2024-08-01 01:45:00
## 2               01:55:00 2024-08-01 03:45:00 2024-08-01 04:01:00
## 3               00:26:00 2024-08-01 06:18:00 2024-08-01 06:34:00
## 4               00:45:00 2024-08-01 07:36:00 2024-08-01 07:54:00
## 5              -00:05:00 2024-08-01 06:54:00 2024-08-01 07:19:00
## 6              -00:09:00 2024-08-01 10:36:00 2024-08-01 10:51:00
##   estimatedInGateVariation     estimatedInGate   estimatedOnGround
## 1                 00:13:00 2024-08-01 05:32:00                <NA>
## 2                 01:47:00 2024-08-01 07:10:00                <NA>
## 3                 00:12:00 2024-08-01 10:12:00                <NA>
## 4                 00:42:00 2024-08-01 11:46:00                <NA>
## 5                -00:11:00 2024-08-01 11:02:00                <NA>
## 6                -00:14:00 2024-08-01 11:45:00 2024-08-01 11:34:00
##   actualInGateVariation        actualInGate      actualOnGround arrTerminal
## 1              00:15:00 2024-08-01 05:34:00 2024-08-01 05:23:00           3
## 2              01:48:00 2024-08-01 07:11:00 2024-08-01 07:01:00           3
## 3              00:03:00 2024-08-01 10:03:00 2024-08-01 09:54:00          T3
## 4              00:45:00 2024-08-01 11:49:00 2024-08-01 11:33:00           3
## 5             -00:08:00 2024-08-01 11:05:00 2024-08-01 10:53:00           3
## 6             -00:10:00 2024-08-01 11:49:00 2024-08-01 11:34:00           3
##   arrGate numStops connections   miles deptTime arrMinutes US_Average_Wait_Time
## 1     H16        0        None 1741.16    01:34         19                   20
## 2     K20        0        None 1437.49    03:45         23                   20
## 3      K4        0        None 1716.10    06:18        300                   13
## 4      H6        0        None 1842.58    07:36        364                   34
## 5      H8        0        None 1741.16    06:54        373                   34
## 6      K2        0        None  161.07    10:36        419                   34
##   US_Max_Wait_Time Non_US_Average_Wait_Times Non_US_Max_Wait_Time
## 1               37                        21                   38
## 2               37                        21                   38
## 3               28                        16                   27
## 4               67                        39                   67
## 5               67                        39                   67
## 6               67                        39                   67
##   Wait.Times.Average_Wait_Time Wait.Times.Max_Wait_Time
## 1                           21                       38
## 2                           21                       38
## 3                           15                       28
## 4                           37                       67
## 5                           37                       67
## 6                           37                       67
##   Number.Of.Passengers.Time.Interval.0.15
## 1                                     103
## 2                                     103
## 3                                      84
## 4                                     131
## 5                                     131
## 6                                     131
##   Number.Of.Passengers.Time.Interval.16.30
## 1                                      223
## 2                                      223
## 3                                       87
## 4                                       88
## 5                                       88
## 6                                       88
##   Number.Of.Passengers.Time.Interval.31.45
## 1                                       61
## 2                                       61
## 3                                        0
## 4                                      166
## 5                                      166
## 6                                      166
##   Number.Of.Passengers.Time.Interval.46.60
## 1                                        0
## 2                                        0
## 3                                        0
## 4                                      206
## 5                                      206
## 6                                      206
##   Number.Of.Passengers.Time.Interval.61.90
## 1                                        0
## 2                                        0
## 3                                        0
## 4                                       75
## 5                                       75
## 6                                       75
##   Number.Of.Passengers.Time.Interval.91.120
## 1                                         0
## 2                                         0
## 3                                         0
## 4                                         0
## 5                                         0
## 6                                         0
##   Number.Of.Passengers.Time.Interval.120_plus X.Excluded X.Total X.Flights
## 1                                           0          9     396         2
## 2                                           0          9     396         2
## 3                                           0          6     177         1
## 4                                           0         20     686         3
## 5                                           0         20     686         3
## 6                                           0         20     686         3
##   arrStart arrEnd arrStartMins arrEndMins DayOfWeek delayed inGateVarMins
## 1    00:00  01:00            0         60       Thu       0            15
## 2    00:00  01:00            0         60       Thu       0           108
## 3    04:00  05:00          240        300       Thu       0             3
## 4    06:00  07:00          360        420       Thu       0            45
## 5    06:00  07:00          360        420       Thu       1            -8
## 6    06:00  07:00          360        420       Thu       1           -10
##   outGateVarMins outGateVarEst deptEstTime
## 1             24            16       01:26
## 2            115           123       03:53
## 3             26             5       05:57
## 4             45            35       07:26
## 5             -5             0       06:59
## 6             -9             0       10:45
```

```r
summary(result)
```

```
##    airline           timeOfFlight   deptAirport         arrAirport
##  Length:2572        Min.   : 57.0   Length:2572        Length:2572
##  Class :character   1st Qu.: 91.0   Class :character   Class :character
##  Mode  :character   Median :131.0   Mode  :character   Mode  :character
##                     Mean   :158.4
##                     3rd Qu.:175.0
##                     Max.   :900.0
##
##     arrDay             arrTime           deptCountry         flightNumber
##  Length:2572        Length:2572        Length:2572        Min.   :  12
##  Class :character   Class :character   Class :character   1st Qu.:1866
##  Mode  :character   Mode  :character   Mode  :character   Median :3500
##                                                           Mean   :3523
##                                                           3rd Qu.:5943
##                                                           Max.   :9240
##
##  estimatedOutGateVariation estimatedOutGate   estimatedOffGround
##  Length:2572               Length:2572        Length:2572
##  Class :character          Class :character   Class :character
##  Mode  :character          Mode  :character   Mode  :character
##
##
##
##
##  actualOutGateVariation actualOutGate      actualOffGround
##  Length:2572            Length:2572        Length:2572
##  Class :character       Class :character   Class :character
##  Mode  :character       Mode  :character   Mode  :character
##
##
##
##
##  estimatedInGateVariation estimatedInGate    estimatedOnGround
##  Length:2572              Length:2572        Length:2572
##  Class :character         Class :character   Class :character
##  Mode  :character         Mode  :character   Mode  :character
##
##
##
##
##  actualInGateVariation actualInGate       actualOnGround     arrTerminal
##  Length:2572           Length:2572        Length:2572        Length:2572
##  Class :character      Class :character   Class :character   Class :character
##  Mode  :character      Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##    arrGate             numStops connections            miles
##  Length:2572        Min.   :0   Length:2572        Min.   :  66.93
##  Class :character   1st Qu.:0   Class :character   1st Qu.: 295.02
##  Mode  :character   Median :0   Mode  :character   Median : 605.15
##                     Mean   :0                      Mean   : 863.80
##                     3rd Qu.:0                      3rd Qu.: 926.25
##                     Max.   :0                      Max.   :7119.65
##
##    deptTime           arrMinutes     US_Average_Wait_Time US_Max_Wait_Time
##  Length:2572        Min.   :  19.0   Min.   : 4.00        Min.   : 13.00
##  Class :character   1st Qu.: 549.0   1st Qu.:17.00        1st Qu.: 46.00
##  Mode  :character   Median : 854.0   Median :24.00        Median : 69.00
##                     Mean   : 828.6   Mean   :31.25        Mean   : 73.66
##                     3rd Qu.:1065.0   3rd Qu.:43.00        3rd Qu.:101.00
##                     Max.   :1371.0   Max.   :85.00        Max.   :175.00
##
##  Non_US_Average_Wait_Times Non_US_Max_Wait_Time Wait.Times.Average_Wait_Time
##  Min.   :  5.0             Min.   : 11.00       Min.   : 5.00
##  1st Qu.: 18.0             1st Qu.: 46.00       1st Qu.:17.00
##  Median : 34.0             Median : 70.00       Median :28.00
##  Mean   : 39.8             Mean   : 76.18       Mean   :34.42
##  3rd Qu.: 57.0             3rd Qu.:101.00       3rd Qu.:47.00
##  Max.   :103.0             Max.   :176.00       Max.   :90.00
##
##  Wait.Times.Max_Wait_Time Number.Of.Passengers.Time.Interval.0.15
##  Min.   : 13.0            Min.   :   0.0
##  1st Qu.: 46.0            1st Qu.: 102.5
##  Median : 70.0            Median : 297.0
##  Mean   : 76.9            Mean   : 432.7
##  3rd Qu.:102.0            3rd Qu.: 637.0
##  Max.   :176.0            Max.   :1749.0
##
##  Number.Of.Passengers.Time.Interval.16.30
##  Min.   :   0.0
##  1st Qu.: 108.0
##  Median : 229.0
##  Mean   : 365.1
##  3rd Qu.: 634.0
##  Max.   :1259.0
##
##  Number.Of.Passengers.Time.Interval.31.45
##  Min.   :  0.0
##  1st Qu.: 55.0
##  Median :136.0
##  Mean   :211.2
##  3rd Qu.:291.0
##  Max.   :840.0
##
##  Number.Of.Passengers.Time.Interval.46.60
##  Min.   :  0.0
##  1st Qu.:  2.0
##  Median : 62.0
##  Mean   :124.4
##  3rd Qu.:188.0
##  Max.   :668.0
```

```
##
##  Number.Of.Passengers.Time.Interval.61.90
##  Min.   :   0.0
##  1st Qu.:   0.0
##  Median :  27.0
##  Mean   : 146.9
##  3rd Qu.: 152.0
##  Max.   :1264.0
##
##  Number.Of.Passengers.Time.Interval.91.120
##  Min.   :  0.00
##  1st Qu.:  0.00
##  Median :  0.00
##  Mean   : 64.15
##  3rd Qu.: 41.00
##  Max.   :875.00
##
##  Number.Of.Passengers.Time.Interval.120_plus   X.Excluded     X.Total
##  Min.   :  0.00                             Min.   :  2   Min.   : 139
##  1st Qu.:  0.00                             1st Qu.: 20   1st Qu.: 647
##  Median :  0.00                             Median : 41   Median :1360
##  Mean   : 16.54                             Mean   : 42   Mean   :1403
##  3rd Qu.:  0.00                             3rd Qu.: 60   3rd Qu.:1971
##  Max.   :345.00                             Max.   :103   Max.   :3264
##
##    X.Flights       arrStart           arrEnd          arrStartMins
##  Min.   : 1.000   Length:2572        Length:2572        Min.   :   0.0
##  1st Qu.: 2.000   Class :character   Class :character   1st Qu.: 540.0
##  Median : 7.000   Mode  :character   Mode  :character   Median : 840.0
##  Mean   : 6.069                                         Mean   : 798.4
##  3rd Qu.: 9.000                                         3rd Qu.:1020.0
##  Max.   :13.000                                         Max.   :1320.0
##
##    arrEndMins      DayOfWeek           delayed       inGateVarMins
##  Min.   :  60.0   Length:2572        Min.   :0.0000   Min.   : -55.0
##  1st Qu.: 600.0   Class :character   1st Qu.:0.0000   1st Qu.: -15.0
##  Median : 900.0   Mode  :character   Median :0.0000   Median :  -2.0
##  Mean   : 858.4                      Mean   :0.4899   Mean   :  20.4
##  3rd Qu.:1080.0                      3rd Qu.:1.0000   3rd Qu.:  26.0
##  Max.   :1380.0                      Max.   :1.0000   Max.   :1244.0
##                                                       NA's   :159
##  outGateVarMins    outGateVarEst     deptEstTime
##  Min.   : -58.00   Min.   : -50.00   Length:2572
##  1st Qu.:  -6.00   1st Qu.:   0.00   Class :character
##  Median :  -1.00   Median :   0.00   Mode  :character
##  Mean   :  21.77   Mean   :  24.68
##  3rd Qu.:  18.00   3rd Qu.:  15.00
##  Max.   :1248.00   Max.   :1140.00
##  NA's   :109       NA's   :12
```

This ‘result’ dataframe will be used for all analysis and modeling in the next sections.

Regression Analysis & Results

Logistic Model Because delay is a binary classifier (1 - delay, 0 - no delay), we used a logistic regression model to predict whether a flight would be delayed. A logistic model is able to handle categorical predictors, and because it is a fairly simple model in design, is much more efficient than some other models. One limitation of using this model is that it assumes a linear relationship between our predictors and the log-odds of our outcome variable (delayed). Below is the code for this model. The predictors are: departure airport, departure time estimated variation, and day of week.

```
no_na <- na.omit(result)
delay_mod <- glm(delayed ~ deptAirport + DayOfWeek + outGateVarEst,
             data = no_na,
             family = binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
predicted_prob <- predict(delay_mod, type = "response")
predicted_class <- ifelse(predicted_prob > 0.5, 1, 0)
table(Actual = no_na$delayed, Predicted = predicted_class)
```

```
##       Predicted
## Actual   0   1
##      0 118   4
##      1   7  19
```

```
confusionMatrix(as.factor(predicted_class), as.factor(no_na$delayed))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 118   7
##          1   4  19
##
##                Accuracy : 0.9257
##                  95% CI : (0.8709, 0.9623)
##     No Information Rate : 0.8243
##     P-Value [Acc > NIR] : 0.0003122
##
##                   Kappa : 0.7312
##
##  Mcnemar's Test P-Value : 0.5464936
##
##             Sensitivity : 0.9672
##             Specificity : 0.7308
##          Pos Pred Value : 0.9440
##          Neg Pred Value : 0.8261
##              Prevalence : 0.8243
##          Detection Rate : 0.7973
##    Detection Prevalence : 0.8446
##       Balanced Accuracy : 0.8490
##
```

```
##        'Positive' Class : 0
##
```

There are three key takeaways from this model. Overall, we see a balanced accuracy of 85%, meaning that the predictors we chose are good predictors of having a delay. The sensitivity and specificity values also reveal important model characteristics. The 'positive' class in our model is 0 (no delay), so that is the class to which our sensitivity score explains. Our model's sensitivity is 0.9672, meaning that the model correctly identified 97% of cases where there was no delay. The other 3% of no-delay cases were incorrectly labeled as delay. Our model's specificity is 0.7308, meaning that the model correctly identified 73% of cases where there was a delay. The other 27% of delay cases were misclassified as no delay.

Because our sensitivity and specificity values were high, our model was fairly good at predicting both classes of delay. It was not very much skewed toward one factor or another; i.e. this model could be useful in predicting whether a flight will be delayed.

Linear Models We then ran some linear models to predict the wait time once at Chicago O'Hare airport (after landing). A linear model can use both factor (categorical) and numerical predictors, and it creates a simple, understandable model for the outcome variable. Using a linear model means that we assume a linear relationship between our predictors and outcome variable (wait time in minutes). When analyzing these models, there are several important metrics. The r-squared value shows the strength of our model to predict our outcome variables. This value ranges from 0 - 1, with a value of 1 representing a model that predicts perfectly. Looking at our predictors, the intercept represents the increase/decrease in our outcome variable (wait time) for a change of one unit in the specific predictor. The significance codes of these predictors reveal how well they serve as predictors. Our goal is the lowest level of significance code (0.001), meaning that the variable is a strong predictor in our model.

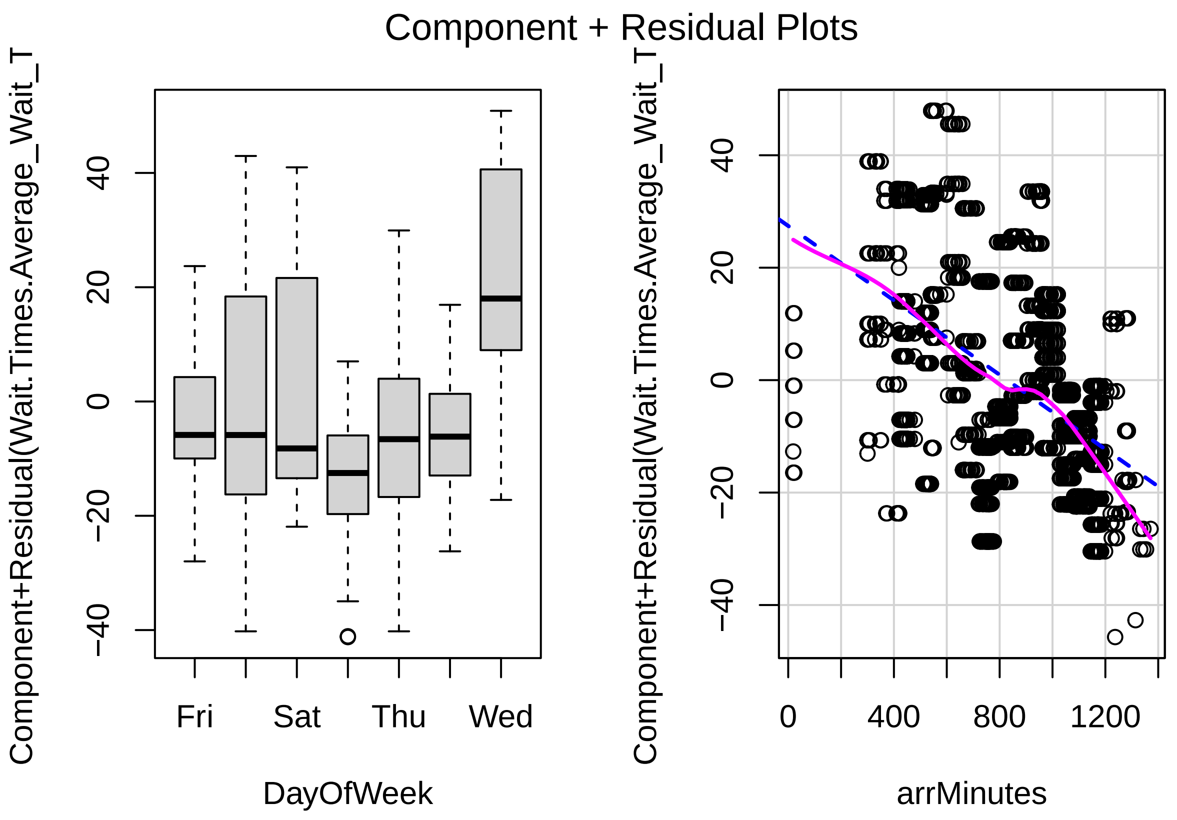
The predictors for all models were: day of week and arrival time. The outcome variables for the 3 models were: average wait time, US citizen average wait time, and Non-US citizen average wait time.

```r
wait_mod <- lm(Wait.Times.Average_Wait_Time ~ DayOfWeek + arrMinutes,
               data = result)
summary(wait_mod)
```

```
##
## Call:
## lm(formula = Wait.Times.Average_Wait_Time ~ DayOfWeek + arrMinutes,
##     data = result)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -43.263 -10.654  -1.705  10.572  40.299
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   59.44814    1.26244  47.090  < 2e-16 ***
## DayOfWeekMon   5.44438    1.12712   4.830 1.44e-06 ***
## DayOfWeekSat   3.08433    1.13336   2.721  0.00654 **
## DayOfWeekSun -11.03072    1.12919  -9.769  < 2e-16 ***
## DayOfWeekThu  -3.95218    1.28476  -3.076  0.00212 **
## DayOfWeekTue  -2.25026    1.15294  -1.952  0.05107 .
## DayOfWeekWed  24.67996    1.15783  21.316  < 2e-16 ***
## arrMinutes    -0.03311    0.00116 -28.533  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 15.84 on 2564 degrees of freedom
## Multiple R-squared:  0.4443, Adjusted R-squared:  0.4428
## F-statistic: 292.8 on 7 and 2564 DF,  p-value: < 2.2e-16
```

```
crPlots(wait_mod)
```



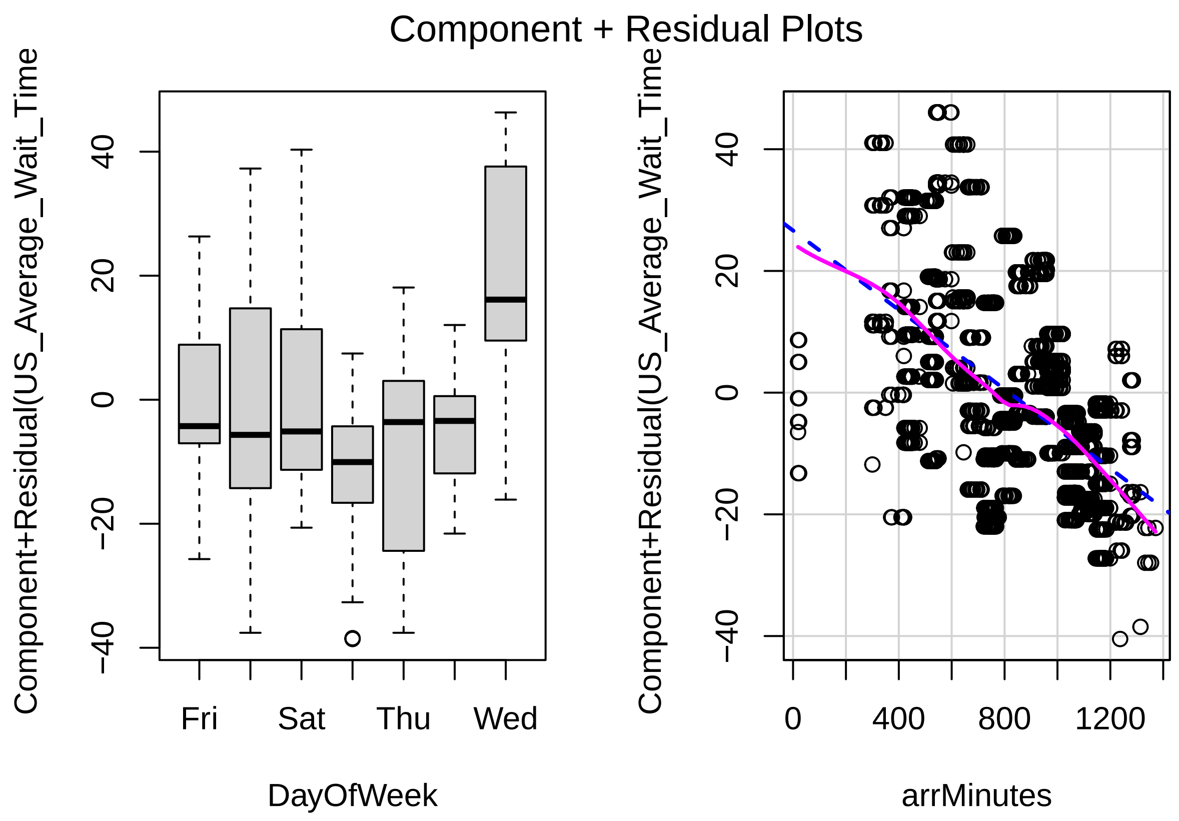
In our first wait model, we predict overall average wait time (for both US and Non-US citizens). Our model has an adjusted r-squared value of 0.4428, meaning that this model is not the greatest in terms of predicting overall wait time. Our model has two significant predictors: day of week and arrival time. Arrival time (in minutes) is significant at the 0.001 level. It has a slight negative relationship with wait time: adding one minute to arrival time (later in the day) decreases the average wait time by 0.033 minutes. Because day of week is a categorical variable, we can analyze the significance of the different levels. Sunday, Monday, and Wednesday are significant at the highest level (0.001). All have a positive relationship with wait time. If the flight occurs on a Sunday, the wait time increases by -11 minutes. If the flight occurs on a Monday, the wait time increases by 5 minutes. Finally, if the flight occurs on a Wednesday, the wait time increases by 25 minutes.

The next two models subset our original dataset. The first looks at only flights departing from within the US and predicts average wait time for a US citizen. The second model subsets to only non-US departing airports, and predicts average wait time for a Non-US citizen.

```
us_data <- result %>%
  filter(deptCountry == 'US')
us_mod <- lm(US_Average_Wait_Time ~ DayOfWeek + arrMinutes,
             data = us_data)
summary(us_mod)
```

```
##
## Call:
## lm(formula = US_Average_Wait_Time ~ DayOfWeek + arrMinutes, data = us_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -39.255  -9.593  -1.042   9.492  38.930
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   57.615771   1.145671  50.290  < 2e-16 ***
## DayOfWeekMon   2.258511   1.039970   2.172 0.029978 *
## DayOfWeekSat   1.981770   1.047772   1.891 0.058693 .
## DayOfWeekSun -11.050270   1.042775 -10.597  < 2e-16 ***
## DayOfWeekThu  -6.166795   1.179797  -5.227 1.88e-07 ***
## DayOfWeekTue  -3.614182   1.063086  -3.400 0.000686 ***
## DayOfWeekWed  19.507813   1.067109  18.281  < 2e-16 ***
## arrMinutes    -0.032601   0.001054 -30.931  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.94 on 2335 degrees of freedom
## Multiple R-squared:  0.4685, Adjusted R-squared:  0.4669
## F-statistic:   294 on 7 and 2335 DF,  p-value: < 2.2e-16
```

```
crPlots(us_mod)
```

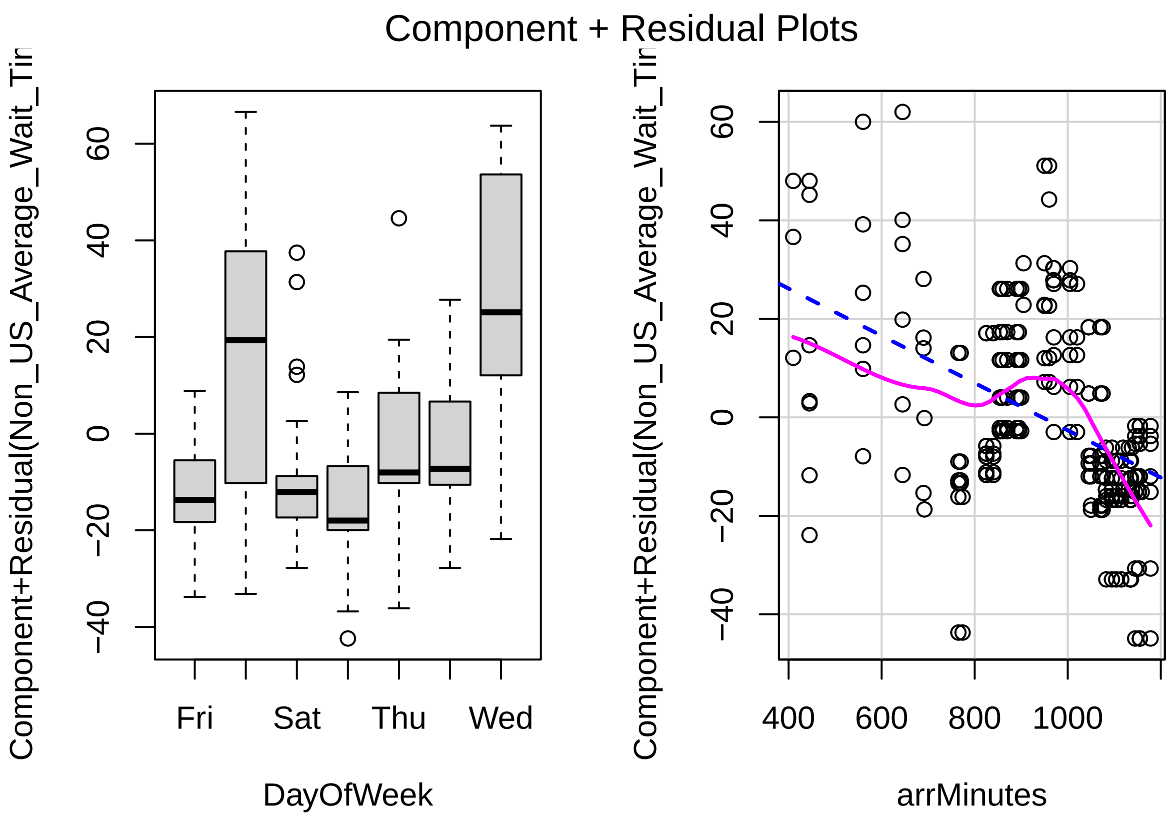


For this US citizen model, we see an adjusted r-squared of 0.4669, meaning that our model is not the greatest predictor for US wait time. Once again, arrival time is significant at the 0.001 level, and it has a slight negative relationship with US citizen wait time. An increase in one minute in the arrival time decreases the wait time by 0.03 minutes. In terms of day of the week - Sunday, Tuesday, Wednesday, and Thursday are significant at the 0.001 level. Sunday, Tuesday, and Thursday have a negative relationship with wait time. On Sunday, the wait time decreases by 11 minutes; on Tuesday, the wait time decreases by 4 minutes; on Thursday, the wait time decreases by 6 minutes. Wednesday has a positive relationship with wait time, as it increases wait time by 20 minutes.

```
non_us_data <- result %>%
  filter(deptCountry != 'US')
non_us_mod <- lm(Non_US_Average_Wait_Times ~ DayOfWeek + arrMinutes,
             data = non_us_data)
summary(non_us_mod)
```

```
##
## Call:
## lm(formula = Non_US_Average_Wait_Times ~ DayOfWeek + arrMinutes,
##     data = non_us_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -52.346  -9.569  -1.916   9.784  51.806
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  70.162836   6.761593  10.377  < 2e-16 ***
## DayOfWeekMon 27.098011   4.262266   6.358 1.16e-09 ***
## DayOfWeekSat  2.178254   4.204698   0.518   0.6049
## DayOfWeekSun -2.445628   4.232901  -0.578   0.5640
## DayOfWeekThu 11.951663   5.172202   2.311   0.0218 *
## DayOfWeekTue  9.348294   4.392339   2.128   0.0344 *
## DayOfWeekWed 42.893276   4.430557   9.681  < 2e-16 ***
## arrMinutes   -0.047987   0.006396  -7.502 1.52e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 18.08 on 221 degrees of freedom
## Multiple R-squared:  0.509,  Adjusted R-squared:  0.4935
## F-statistic: 32.73 on 7 and 221 DF,  p-value: < 2.2e-16
```

```
crPlots(non_us_mod)
```



In this model, our outcome variable is the average wait time for Non-US citizens. We see an adjusted r-squared of 0.4953, meaning that our model is not the best at prediction. However, we see some significant variables. Like in our previous models, arrival time is significant at the 0.001 level, and it once again has a slight negative relationship with wait time. In terms of days of the week, only Monday and Wednesday are significant at the 0.001 level. Both had a positive relationship with wait time. Monday increases the wait time by 27 minutes, and Wednesday increases the wait time by 43 minutes.

All of our models provide actionable insights for travelers, which we will explain in the following section.

Discussion

With our data analysis, we hoped to be able to predict delays and airport wait times for American Airlines flights into Chicago O'Hare.

In our binary model, we predicted whether a flight will be delayed using departure airport, departure gate estimated variation, and the day of the week. Our model had a 85% accuracy, meaning that this model would be useful to predict a flight's delay status.

In the following code, we test our binary delay model on a flight. We chose a scheduled flight that was hours from taking off with the goal of predicting whether the flight would be delayed. Our predictors were the departure airport (London), the day of the week of the flight (Wednesday), and the estimated out gate variation (the flight was projected to leave 9 minutes before scheduled). Our model predicted that the flight would not be delayed (delayed = 0). After the flight landed, we checked it's status; we were correct in our prediction.

```
test_data <- data.frame(deptAirport = c('LHR'), DayOfWeek = c('Wed'), outGateVarEst = c(-9))
predicted_prob_test <- predict(delay_mod, type = "response", newdata = test_data)
```

```r
predicted_class_test <- ifelse(predicted_prob_test > 0.5, 1, 0)

actual_delay = 0
table(Actual = actual_delay, Predicted = predicted_class_test)
```

```
##       Predicted
## Actual 0
##      0 1
```

```r
#predicted_class_test[1]
```

We then ran three linear models, all of which provided similar insights about wait time. In all three models, we found that a later arrival time led to a slight decrease in wait time, for both US and Non-US citizens. As a result, we would recommend that people fly later in the day to decrease their wait time at the airport. All our linear models also reveal information about wait time related to the day of the flight. For all people, Wednesday is the worst day to fly, as it increases the wait time by about 30 minutes. Flying on a Monday also leads to a longer wait time, although not as lengthy as a Wednesday. There were several days that shorten wait time: Sunday, Tuesday, and Thursday (for US travelers). We would recommend people fly on Sunday, Tuesday, and Thursday, and do not fly on Monday or Wednesday. Below is a summary of these outcomes for day of the week: 1. All travelers: - Sunday: -11 minutes - Monday: +5 minutes - Wednesday: +25 minutes

2. US travelers:

- Wednesday: +20 minutes
- Sunday: -11 minutes
- Tuesday: -4 minutes
- Thursday: -6 minutes

3. Non-US travelers:

- Monday: +27 minutes
- Wednesday: +43 minutes

Our main data limitations is the time period of the data. With our free API, we could only do a limited number of requests; using two accounts only allowed for a week's worth of data. To further validate our analyses, we would want to gather many more weeks of data and run our models again. Additionally, we would want to gather data on more than one airline and analysis waits/delays for these different airlines. Because our results include only one airline and arrival airport, we cannot generalize our results beyond these.

References

API Websites/Documentation: - A Form API of wait times for customs from international flights arriving into Chicago O'Hare, one of the major US airports. - https://awt.cbp.gov/ - A REST API of historical flight arrivals data - https://rapidapi.com/oag-oag-default/api/flight-info-api

Test Flight: - https://www.aa.com/travelInformation/flights/status/detail?search=AA%7C87%7CLHR%7CORD%7C2024,10,2&ref=results