Airline Data Challenge

Haile Endeshaw

1/16/2022

Part 1: Data Import, Quality Check, and Cleaning

```
library(tidyverse)
library(ggplot2)
library(readxl)
library(data.table)
library(plotly)
library(knitr)
```

Data import:

```
#function to load data into R
read_dat <- function(file_name){
  read_csv(file_name)
}
#call function to import data
#airport_codes <- read_dat("Airport_Codes.csv")
airport_codes <- fread("Airport_Codes.csv")
flights <- fread("Flights.csv")
tickets <- fread("tickets.csv")
```

Data Quality Checks: Preview data and feature data types

```
as_tibble(airport_codes)
```

```
## # A tibble: 55,369 x 8
##    TYPE          NAME  ELEVATION_FT CONTINENT ISO_COUNTRY MUNICIPALITY IATA_CODE
##    <chr>         <chr>        <dbl> <chr>     <chr>       <chr>        <chr>
##  1 heliport      Tota~           11 ""        US          Bensalem     ""
##  2 small_airport Aero~         3435 ""        US          Leoti        ""
##  3 small_airport Lowe~          450 ""        US          Anchor Point ""
##  4 small_airport Epps~          820 ""        US          Harvest      ""
##  5 closed        Newp~          237 ""        US          Newport      ""
##  6 small_airport Fult~         1100 ""        US          Alex         ""
##  7 small_airport Cord~         3810 ""        US          Cordes       ""
##  8 small_airport Gold~         3038 ""        US          Barstow      ""
##  9 small_airport Will~           87 ""        US          Biggs        ""
## 10 heliport      Kitc~         3350 ""        US          Pine Valley  ""
## # ... with 55,359 more rows, and 1 more variable: COORDINATES <chr>
```

```
as_tibble(flights)
```

```
## # A tibble: 1,915,886 x 16
##    FL_DATE    OP_CARRIER TAIL_NUM OP_CARRIER_FL_NUM ORIGIN_AIRPORT_ID ORIGIN
##    <chr>      <chr>      <chr>    <chr>                          <int> <chr>
##  1 2019-03-02 WN         N955WN   4591                           14635 RSW
##  2 2019-03-02 WN         N8686A   3231                           14635 RSW
##  3 2019-03-02 WN         N201LV   3383                           14635 RSW
##  4 2019-03-02 WN         N413WN   5498                           14635 RSW
##  5 2019-03-02 WN         N7832A   6933                           14635 RSW
##  6 2019-03-02 WN         N492WN   3960                           14635 RSW
##  7 2019-03-02 WN         N8634A   3826                           14635 RSW
##  8 2019-03-02 WN         N210WN   4014                           14635 RSW
##  9 2019-03-02 WN         N232WN   4492                           14635 RSW
## 10 2019-03-02 WN         N7889A   4899                           14635 RSW
## # ... with 1,915,876 more rows, and 10 more variables: ORIGIN_CITY_NAME <chr>,
## #   DEST_AIRPORT_ID <int>, DESTINATION <chr>, DEST_CITY_NAME <chr>,
## #   DEP_DELAY <dbl>, ARR_DELAY <dbl>, CANCELLED <dbl>, AIR_TIME <chr>,
## #   DISTANCE <chr>, OCCUPANCY_RATE <dbl>
```

```
as_tibble(tickets)
```

```
## # A tibble: 1,167,285 x 12
##     ITIN_ID  YEAR QUARTER ORIGIN ORIGIN_COUNTRY ORIGIN_STATE_ABR ORIGIN_STATE_NM
##     <int64> <int>   <int> <chr>  <chr>          <chr>            <chr>
##  1     2e11  2019       1 ABI    US             TX               Texas
##  2     2e11  2019       1 ABI    US             TX               Texas
##  3     2e11  2019       1 ABI    US             TX               Texas
##  4     2e11  2019       1 ABI    US             TX               Texas
##  5     2e11  2019       1 ABI    US             TX               Texas
##  6     2e11  2019       1 ABI    US             TX               Texas
##  7     2e11  2019       1 ABI    US             TX               Texas
##  8     2e11  2019       1 ABI    US             TX               Texas
##  9     2e11  2019       1 ABI    US             TX               Texas
## 10     2e11  2019       1 ABI    US             TX               Texas
## # ... with 1,167,275 more rows, and 5 more variables: ROUNDTRIP <dbl>,
## #   REPORTING_CARRIER <chr>, PASSENGERS <dbl>, ITIN_FARE <chr>,
## #   DESTINATION <chr>
```

```
glimpse(airport_codes) #airport_codes datasets
```

```
## Rows: 55,369
## Columns: 8
## $ TYPE         <chr> "heliport", "small_airport", "small_airport", "small_airp~
## $ NAME         <chr> "Total Rf Heliport", "Aero B Ranch Airport", "Lowell Fiel~
## $ ELEVATION_FT <dbl> 11, 3435, 450, 820, 237, 1100, 3810, 3038, 87, 3350, 4830~
## $ CONTINENT    <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "~
## $ ISO_COUNTRY  <chr> "US", "US", "US", "US", "US", "US", "US", "US", "US", "US~
## $ MUNICIPALITY <chr> "Bensalem", "Leoti", "Anchor Point", "Harvest", "Newport"~
## $ IATA_CODE    <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "~
## $ COORDINATES  <chr> "-74.93360137939453, 40.07080078125", "-101.473911, 38.70~
```

```
glimpse(flights) #flights dataset
```

```
## Rows: 1,915,886
## Columns: 16
## $ FL_DATE           <chr> "2019-03-02", "2019-03-02", "2019-03-02", "2019-03-0~
## $ OP_CARRIER        <chr> "WN", "WN", "WN", "WN", "WN", "WN", "WN", "WN", "WN"~
## $ TAIL_NUM          <chr> "N955WN", "N8686A", "N201LV", "N413WN", "N7832A", "N~
## $ OP_CARRIER_FL_NUM <chr> "4591", "3231", "3383", "5498", "6933", "3960", "382~
## $ ORIGIN_AIRPORT_ID <int> 14635, 14635, 14635, 14635, 14635, 14635, 14635, 146~
## $ ORIGIN            <chr> "RSW", "RSW", "RSW", "RSW", "RSW", "RSW", "RSW", "RS~
## $ ORIGIN_CITY_NAME  <chr> "Fort Myers, FL", "Fort Myers, FL", "Fort Myers, FL"~
## $ DEST_AIRPORT_ID   <int> 11042, 11066, 11066, 11066, 11259, 11986, 12339, 123~
## $ DESTINATION       <chr> "CLE", "CMH", "CMH", "CMH", "DAL", "GRR", "IND", "IN~
## $ DEST_CITY_NAME    <chr> "Cleveland, OH", "Columbus, OH", "Columbus, OH", "Co~
## $ DEP_DELAY         <dbl> -8, 1, 0, 11, 0, -2, 11, -5, -9, 0, -7, 5, -2, -2, 1~
## $ ARR_DELAY         <dbl> -6, 5, 4, 14, -17, -8, 17, 8, -16, 11, -5, 8, -4, 11~
## $ CANCELLED         <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ AIR_TIME          <chr> "143.0", "135.0", "132.0", "136.0", "151.0", "162.0"~
## $ DISTANCE          <chr> "1025.0", "930.0", "930.0", "930.0", "1005.0", "1147~
## $ OCCUPANCY_RATE    <dbl> 0.97, 0.55, 0.91, 0.67, 0.62, 0.49, 0.49, 0.37, 0.31~
```

The distance airtime features is given as a character vector. These need to be converted into numeric

```
flights$AIR_TIME <- as.numeric(flights$AIR_TIME)
flights$DISTANCE <- as.numeric(flights$DISTANCE)
flights$OCCUPANCY_RATE <- as.numeric(flights$OCCUPANCY_RATE)
```

Check for missing values (NAs) and invalid values on the flights dataset

```
sum(is.na(flights$ORIGIN_AIRPORT_ID))
```

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

```
sum(is.na(flights$DEST_AIRPORT_ID))
```

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

```
sum(is.na(flights$ORIGIN))
```

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

```
sum(is.na(flights$DESTINATION))
```

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

```
sum(is.na(flights$DEP_DELAY)) #NAs should be eliminated from consideration
```

```
## [1] 50351
```

```
sum(is.na(flights$ARR_DELAY)) #NAs should be eliminated from consideration
```

```
## [1] 55991
```

```
sum(is.na(flights$CANCELLED))
```

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

```
sum(is.na(flights$DISTANCE))
```

```
## [1] 2740
```

```
sum(is.na(flights$OCCUPANCY_RATE) | flights$OCCUPANCY_RATE <= 0)
```

```
## [1] 310
```

Find rows where NAs are available and check to see if the data is available in other rows, but the same routes. First let's see the DISTANCE vector in the flights dataset.

```
#Flights dataset - DISTANCE
ind <- which(is.na(flights$DISTANCE) | flights$DISTANCE <= 0) #indices where distance da
ta is missing or invalid
inv_dist <- data.frame(ORIGIN = flights$ORIGIN[ind], DESTINATION = flights$DESTINATION[i
nd]) %>%
  unique() #extract routes where distance data is missing or invalid
#Extract valid distance from other rows of the same routes
v_dist <- flights[-ind, ] %>%
  filter(inv_dist[, 1] == ORIGIN & inv_dist[, 2] == DESTINATION) %>%
  select(ORIGIN, DESTINATION, DISTANCE) %>% unique() %>%
  arrange(ORIGIN, DESTINATION)
#It can be seen that distance info for all missing or invalid data is already available
 from other rows
new_dist <- left_join(inv_dist, v_dist, by = c("ORIGIN", "DESTINATION"))
new_dist
```

```
##    ORIGIN DESTINATION DISTANCE
## 1     RSW         MDW     1105
## 2     RSW         MDW     2312
## 3     BWI         DFW     1217
## 4     DFW         SAT      247
## 5     PHX         SJC      621
## 6     SJC         PHX      621
## 7     ATL         LAX     1947
## 8     JFK         ORD      740
## 9     DFW         BDL     1471
## 10    PIT         ORD      413
## 11    BOS         ORD      867
## 12    DFW         SFO     1464
## 13    DTW         PHX     1671
## 14    LAX         ORD     1744
## 15    SNA         DFW     1205
## 16    TUL         CLT      842
```

It can be seen that RSW to MDW has two different distance values. Therefore, the second (incorrect) should be eliminated

```
new_dist <- new_dist[-2, ] #remove the wrong distance
#Substitute the distance values into the missing values into the flights dataset
new_df <- data.frame(ORIGIN = flights$ORIGIN[ind], DESTINATION = flights$DESTINATION[in
d]) #collect missing values
dist <- left_join(new_df, new_dist, by = c("ORIGIN", "DESTINATION")) #join distance data
to routes
flights$DISTANCE[ind] <- dist$DISTANCE #substitute distance data back to the flights dat
abase
sum(is.na(flights$DISTANCE) | flights$DISTANCE <= 0) #check for invalid or missing dista
nce
```

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

The DISTANCE vector represents the one-way distance between origin and destination airports. For roundtrip analysis, distance values must be doubled.

```
flights$DISTANCE <- 2 * flights$DISTANCE
```

Now let's look into the "OCCUPANCY_RATE" feature in the flights dataset

```
#Flights dataset - OCCUPANCY RATE
#similarly substitute missing values in the OCCUPANCY_RATE vector
ind <- which(is.na(flights$OCCUPANCY_RATE) | flights$OCCUPANCY_RATE <= 0) #indices where
distance data is missing or invalid
inv_o_rate <- data.frame(ORIGIN = flights$ORIGIN[ind],
                         DESTINATION = flights$DESTINATION[ind],
                         OP_CARRIER = flights$OP_CARRIER[ind]) %>%
  unique()             #extract routes and carriers where fare data is missing or invali
d
#Extract valid fares from other tickets of the same routes and carriers
v_o_rate <- flights[-ind, ] %>%
  filter(inv_o_rate[, 1] == ORIGIN & inv_o_rate[, 2] == DESTINATION & inv_o_rate[, 3] ==
OP_CARRIER) %>%
  select(ORIGIN, DESTINATION, OP_CARRIER, OCCUPANCY_RATE) %>% unique() %>%
  arrange(ORIGIN, DESTINATION, OP_CARRIER)
nrow(v_o_rate)
```

```
## [1] 169
```

It can be seen that valid occupancy-rate data for all missing or invalid data is already available from other rows

```
new_o_rate <- left_join(inv_o_rate, v_o_rate, by = c("ORIGIN", "DESTINATION", "OP_CARRIE
R")) %>%
  group_by(ORIGIN, DESTINATION, OP_CARRIER) %>%
  summarise(ave_OCCUPANCY_RATE = mean(OCCUPANCY_RATE))
as_tibble(new_o_rate)
```

```
## # A tibble: 2 x 4
##   ORIGIN DESTINATION OP_CARRIER ave_OCCUPANCY_RATE
##   <chr>  <chr>       <chr>                   <dbl>
## 1 BOS    ORD         AA                      0.645
## 2 DFW    SFO         AA                      0.673
```

Substitute the distance values into the missing values into the flights dataset

```
new_df <- data.frame(ORIGIN = flights$ORIGIN[ind], DESTINATION = flights$DESTINATION[in
d],
                     OP_CARRIER = flights$OP_CARRIER[ind]) #collect missing values
o_rate <- left_join(new_df, new_o_rate, by = c("ORIGIN", "DESTINATION", "OP_CARRIER")) #
join distance data to routes
flights$OCCUPANCY_RATE[ind] <- o_rate$OP_CARRIER #substitute occ-rate data back to the f
lights database
sum(is.na(flights$OCCUPANCY_RATE) | flights$OCCUPANCY_RATE <= 0) #check for invalid or m
issing distance
```

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

Tickets dataset: Check missing and invalid features in the tickets dataset

```
#check preview of data features and their data-types
glimpse(tickets)
```

```
## Rows: 1,167,285
## Columns: 12
## $ ITIN_ID           <int64> 201912723049, 201912723085, 201912723491, 20191272~
## $ YEAR              <int> 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019~
## $ QUARTER           <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
## $ ORIGIN            <chr> "ABI", "ABI", "ABI", "ABI", "ABI", "ABI", "ABI", "AB~
## $ ORIGIN_COUNTRY    <chr> "US", "US", "US", "US", "US", "US", "US", "US", "US"~
## $ ORIGIN_STATE_ABR  <chr> "TX", "TX", "TX", "TX", "TX", "TX", "TX", "TX", "TX"~
## $ ORIGIN_STATE_NM   <chr> "Texas", "Texas", "Texas", "Texas", "Texas", "Texas"~
## $ ROUNDTRIP         <dbl> 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0~
## $ REPORTING_CARRIER <chr> "MQ", "MQ", "MQ", "MQ", "MQ", "MQ", "MQ", "MQ", "MQ"~
## $ PASSENGERS        <dbl> 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
## $ ITIN_FARE         <chr> "736.0", "570.0", "564.0", "345.0", "309.0", "303.0"~
## $ DESTINATION       <chr> "DAB", "COS", "MCO", "LGA", "MGM", "MSY", "MSY", "LA~
```

ITIN_Fare is listed as character and needs to be converted into numeric for analysis

```
tickets$ITIN_FARE <- as.numeric(tickets$ITIN_FARE)
```

check for missing or invalid values on important features that are used for analysis

```
sum(is.na(tickets$ORIGIN))
```

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

```
sum(is.na(tickets$DESTINATION))
```

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

```
sum(is.na(tickets$ROUNDTRIP))
```

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

```
sum(is.na(tickets$ITIN_FARE) | tickets$ITIN_FARE <= 0)
```

```
## [1] 17412
```

Find rows where NAs are available and check to see if the data is available in other rows, but the same routes and carriers

```
ind <- which(is.na(tickets$ITIN_FARE) | tickets$ITIN_FARE <= 0) #indices where distance
 data is missing or invalid
inv_fare <- data.frame(ORIGIN = tickets$ORIGIN[ind],
                       DESTINATION = tickets$DESTINATION[ind],
                       REPORTING_CARRIER = tickets$REPORTING_CARRIER[ind]) %>%
  unique()             #extract routes and carriers where fare data is missing or invali
d
```

Extract valid fares from other tickets of the same routes and carriers

```
v_fare <- tickets[-ind, ] %>%
  filter(inv_fare[, 1] == ORIGIN & inv_fare[, 2] == DESTINATION & inv_fare[, 3] == REPOR
TING_CARRIER) %>%
  select(ORIGIN, DESTINATION, REPORTING_CARRIER, ITIN_FARE) %>% unique() %>%
  arrange(ORIGIN, DESTINATION, REPORTING_CARRIER)
nrow(v_fare)
```

```
## [1] 33
```

Only a very small percentage of of the missing values are found this way. Assumption: Use only route information and average fare information to these routes to fill the missing fare data, instead of removing the whole rows

```
ave_fares <- tickets %>% group_by(ORIGIN, DESTINATION) %>%
  summarise(fare = mean(ITIN_FARE, na.rm = TRUE))
as_tibble(ave_fares)
```

```
## # A tibble: 48,202 x 3
##    ORIGIN DESTINATION  fare
##    <chr>  <chr>       <dbl>
##  1 ABE    ABQ          530
##  2 ABE    AGS          299
##  3 ABE    ALB          284
##  4 ABE    AMA          654
##  5 ABE    ASE          742
##  6 ABE    ATL          460.
##  7 ABE    ATW          576.
##  8 ABE    AUS          340.
##  9 ABE    AZO          331
## 10 ABE    BDL          380.
## # ... with 48,192 more rows
```

```
new_fare <- left_join(inv_fare, ave_fares, by = c("ORIGIN", "DESTINATION"))
as_tibble(new_fare)
```

```
## # A tibble: 10,932 x 4
##    ORIGIN DESTINATION REPORTING_CARRIER  fare
##    <chr>  <chr>       <chr>             <dbl>
##  1 ABR    SLC         OO                 411.
##  2 ABR    SJU         OO                 344.
##  3 ABR    MCO         OO                 515.
##  4 ABR    HNL         OO                1259.
##  5 ABR    OGG         OO                 587.
##  6 ABR    RSW         OO                 459.
##  7 ABR    LAS         OO                 352.
##  8 ABR    PHX         OO                 462.
##  9 ABR    SFO         OO                 666.
## 10 ABR    FLL         OO                 488.
## # ... with 10,922 more rows
```

Check if there are any missing or invalid values

```
new_fare %>% filter(is.na(fare)) %>% nrow()
```

```
## [1] 29
```

The number of missing values are now greatly reduced.Substitute the fare values in place of the missing data

```
new_df <- data.frame(ORIGIN = tickets$ORIGIN[ind],
                     DESTINATION = tickets$DESTINATION[ind],
                     REPORTING_CARRIER = tickets$REPORTING_CARRIER[ind]) #collect missin
g rows
fare <- left_join(new_df, new_fare, by = c("ORIGIN", "DESTINATION", "REPORTING_CARRIER"
)) #join distance data to routes
tickets$ITIN_FARE[ind] <- fare$fare #substitute distance data back to the tickets databa
se
```

check for invalid or missing fare

```
sum(is.na(tickets$ITIN_FARE) | tickets$ITIN_FARE <= 0)
```

```
## [1] 465
```

A few hundred routes don't have any valid fare data reported and must be removed from consideration since cost analysis for these routes can not be performed.

```
tickets <- tickets %>% filter(!is.na(ITIN_FARE) & !(ITIN_FARE <= 0))
sum(is.na(tickets$ITIN_FARE) | tickets$ITIN_FARE <= 0)  #check missing values
```

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

```
nrow(tickets) #No missing or invalid fare values
```

```
## [1] 1166820
```

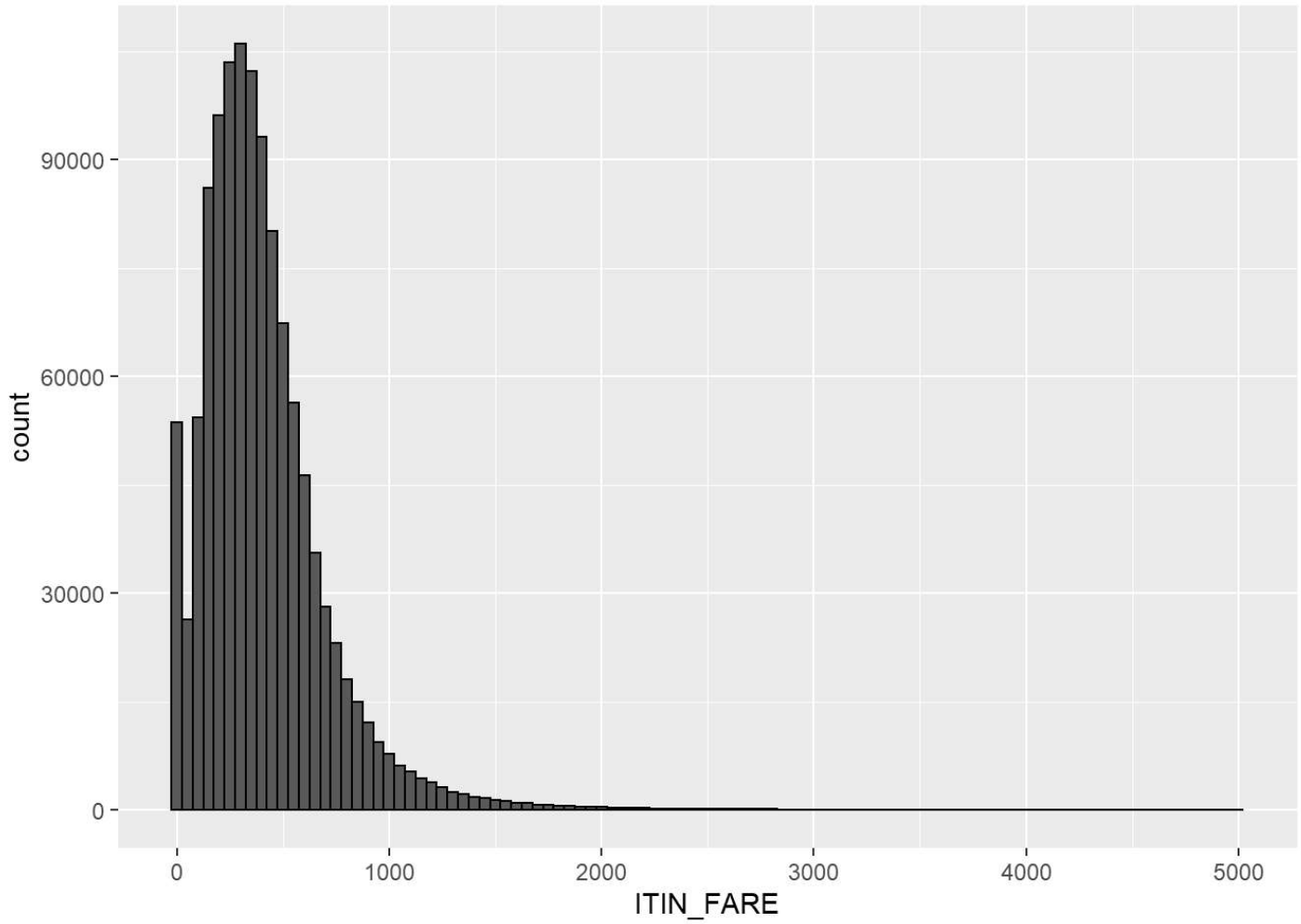
Checking the range of fares reveals that some fare values may not be accurate (are too high or too low)

```
range(na.omit(as.numeric(tickets$ITIN_FARE)))
```

```
## [1]      1 377400
```

Removing fares the distribution of fares can be shown

```
tickets %>% filter(ITIN_FARE <= 5000) %>% #use fares < $30,000
  ggplot(aes(ITIN_FARE)) + geom_histogram(binwidth = 50, color = "black")
```



The very high prices are very small in number and chances are they are incorrect values. Therefore, fares above $5000 are ignored in this analysis. Prices less than $100 are also assumed to be incorrect and, therefore, are ignored.

```
tickets <- tickets %>% filter(ITIN_FARE <= 5000 | ITIN_FARE > 100)
```

To limit the size of the resulting dataframes, select only important features that could be used for analysis.

```
#select only important features
levels(as_factor(airport_codes$TYPE)) #levels of airport types
```

```
## [1] "heliport"       "small_airport"  "closed"         "seaplane_base"
## [5] "balloonport"    "medium_airport" "large_airport"
```

```
airport_codes <- airport_codes %>%
  filter(TYPE %in% c("medium_airport", "large_airport")) %>% #select only medium and lar
ge airports
  mutate(origin_IATA_code = IATA_CODE) %>%  #new features
  select(TYPE, origin_IATA_code)
levels(as_factor(airport_codes$TYPE)) #levels of airport types
```

```
## [1] "medium_airport" "large_airport"
```

Select flights that are not canceled and rename a few columns in the df for joining

```
flights <- flights %>%
  select(-c(FL_DATE, TAIL_NUM, OP_CARRIER_FL_NUM, AIR_TIME)) %>%
  filter(CANCELLED == 0) %>%
  rename(origin_IATA_code = ORIGIN, dest_IATA_code = DESTINATION)
```

select tickets that are round-trip and rename a few columns in the df for joining

```
tickets <- tickets %>%
  select(-c(YEAR, QUARTER, PASSENGERS)) %>%
  filter(ROUNDTRIP == 1) %>%
  rename(origin_IATA_code = ORIGIN, dest_IATA_code = DESTINATION)
```

Join airport_codes and flights data frames by origin and destination airport codes

```
flights <- left_join(flights, airport_codes, by = "origin_IATA_code") #
#join airport_codes and tickets data frames by origin and destination airport codes
tickets <- left_join(tickets, airport_codes, by = "origin_IATA_code") #type col---not ri
ght
airport_codes <- airport_codes %>%
  rename(dest_IATA_code = origin_IATA_code)
```

Join airport_codes and flights data frames by destination airport codes

```
flights <- left_join(flights, airport_codes, by = "dest_IATA_code") %>%
  rename(origin_airport_type = TYPE.x, des_airport_type = TYPE.y)
```

join airport_codes and tickets data frames by destination airport codes

```
tickets <- left_join(tickets, airport_codes, by = "dest_IATA_code") %>% #type col---not
 right
  rename(origin_airport_type = TYPE.x, des_airport_type = TYPE.y)
```

Extract fares for each route. Fares per carrier were averaged

```
t1 <- tickets %>% rename(OP_CARRIER = REPORTING_CARRIER) %>%
  group_by(origin_IATA_code, dest_IATA_code, OP_CARRIER) %>%
  summarise(fare = mean(ITIN_FARE)) %>%
  arrange(desc(fare))
```

Join t1 and the flights dataframes to Create a new dataframe that contains flights information as well as fares

```
flights2 <- inner_join(flights, t1, by = c("origin_IATA_code", "dest_IATA_code", "OP_CAR
RIER")) %>%
  group_by(origin_IATA_code, dest_IATA_code, OP_CARRIER)
```

Part 2: Analysis

Top Busiest Round-trip Routes

```
top10_busiest_routes <- flights %>%
  group_by(origin_IATA_code, dest_IATA_code) %>%
  summarize(num_routes = n()) %>% arrange(desc(num_routes)) %>% #count the num of rows w
ith same routes and arrange in desc order
  head(10)
top10_busiest_routes
```

```
## # A tibble: 10 x 3
## # Groups:   origin_IATA_code [7]
##    origin_IATA_code dest_IATA_code num_routes
##    <chr>            <chr>               <int>
##  1 SFO              LAX                  4176
##  2 LAX              SFO                  4164
##  3 ORD              LGA                  3580
##  4 LGA              ORD                  3576
##  5 LAX              LAS                  3257
##  6 LAS              LAX                  3254
##  7 LAX              JFK                  3162
##  8 JFK              LAX                  3158
##  9 LAX              SEA                  2502
## 10 SEA              LAX                  2497
```

Top Profitable Round-Trip Routes

Cost Analysis The parameters used to calculate cost are

- OM_cost - operation and maintenance cost per mile (fuel, oil, maintenance, and crew)
- medium_airport_use_cost = medium airport cost
- large_airport_use_cost = large airport cost
- dep_delay_rate = departure delay fee per minute for each delay greater than 15 min
- arr_delay_rate = arrival delay fee per minute for each delay greater than 15 min
- free_delay_minutes = maximum free delay possible in minutes
- plane_capacity = capacity of each plane
- baggage_rate = fee for each checked bag for a round trip
- p_bag_passengers = proportion of passengers with checked-in bags
- Miscellaneous cost = depreciation, insurance, etc. costs per mile

```
#Known parameters
OM_rate <- 8 #O&M cost per mile
misc_rate <- 1.18 # miscellaneous (depreciation, insurance, and other) costs per mile
medium_airport_use_cost <- 5000 #medium airport cost
large_airport_use_cost <- 10000 #large airport cost
dep_delay_rate <- 75 #departure delay fee per minute for each delay greater than 15 min
arr_delay_rate <- 75 #arrival delay fee per minute for each delay greater than 15 min
free_delay_minutes <- 15 #maximum free delay possible in minutes
plane_capacity <- 200 #capacity of each plane
baggage_rate <- 70 #fee for each checked bag for a round trip
p_bag_passengers <- 0.5 #proportion of passengers with checked-in bags
flights2$OCCUPANCY_RATE <- as.numeric(flights2$OCCUPANCY_RATE) #convert occupancy rate i
nto numeric
```

Create “cost” df and include features from flights and other calculated features

```
cost <- flights2 %>%
  mutate(origin_airport_use_cost = ifelse(origin_airport_type == "medium_airport", mediu
m_airport_use_cost, large_airport_use_cost),
         dest_airport_use_cost = ifelse(des_airport_type == "medium_airport", medium_air
port_use_cost, large_airport_use_cost),
         OM_cost = 2 * DISTANCE * OM_rate,
         dep_delay_penalty_min = ifelse(DEP_DELAY - free_delay_minutes > 0, DEP_DELAY -
 free_delay_minutes, 0),
         arr_delay_penalty_min = ifelse(ARR_DELAY - free_delay_minutes > 0, ARR_DELAY -
 free_delay_minutes, 0),
         dep_delay_cost = dep_delay_rate * dep_delay_penalty_min,
         arr_delay_cost = arr_delay_rate * arr_delay_penalty_min,
         misc_cost = round(2 * DISTANCE * misc_rate))
```

Calculate the revenue and add the revenue and cost data together. Calculate profit for each route.

```
cost_revenue <- cost %>%
  mutate(n_passengers = round(plane_capacity * OCCUPANCY_RATE),              #n_passenge
rs = # of passengers
         baggage_fee = baggage_rate * p_bag_passengers * n_passengers,
         total_revenue = round(baggage_fee + n_passengers * fare),
         total_cost = round(origin_airport_use_cost + dest_airport_use_cost +
                              OM_cost + dep_delay_cost + arr_delay_cost + misc_cost),
         profit = total_revenue - total_cost) %>%
  arrange(desc(profit))
```

Unite origin and destination airports to create a route feature that contains the names of both the departure and destination airports

```
cost_revenue <- cost_revenue %>%
  unite(origin_IATA_code, dest_IATA_code, col = "route", sep = "-")
as_tibble(cost_revenue) #show preview
```

```
## # A tibble: 1,758,951 x 27
##    OP_CARRIER ORIGIN_AIRPORT_ID route   ORIGIN_CITY_NAME   DEST_AIRPORT_ID
##    <chr>                  <int> <chr>   <chr>                        <int>
##  1 OO                     14869 SLC-TWF Salt Lake City, UT           15389
##  2 OO                     14869 SLC-TWF Salt Lake City, UT           15389
##  3 OO                     14869 SLC-TWF Salt Lake City, UT           15389
##  4 OO                     14869 SLC-TWF Salt Lake City, UT           15389
##  5 OO                     14869 SLC-TWF Salt Lake City, UT           15389
##  6 OO                     14869 SLC-TWF Salt Lake City, UT           15389
##  7 OO                     14869 SLC-TWF Salt Lake City, UT           15389
##  8 OO                     14869 SLC-TWF Salt Lake City, UT           15389
##  9 OO                     14869 SLC-TWF Salt Lake City, UT           15389
## 10 OO                     14869 SLC-TWF Salt Lake City, UT           15389
## # ... with 1,758,941 more rows, and 22 more variables: DEST_CITY_NAME <chr>,
## #   DEP_DELAY <dbl>, ARR_DELAY <dbl>, CANCELLED <dbl>, DISTANCE <dbl>,
## #   OCCUPANCY_RATE <dbl>, origin_airport_type <chr>, des_airport_type <chr>,
## #   fare <dbl>, origin_airport_use_cost <dbl>, dest_airport_use_cost <dbl>,
## #   OM_cost <dbl>, dep_delay_penalty_min <dbl>, arr_delay_penalty_min <dbl>,
## #   dep_delay_cost <dbl>, arr_delay_cost <dbl>, misc_cost <dbl>,
## #   n_passengers <dbl>, baggage_fee <dbl>, total_revenue <dbl>, ...
```

Joining the cost and fare_dat results in missing values since not all routes are available in both flights and tickets databases. The best practice would be to join the flights, tickets, and airport_codes datasets at the beginning of the analysis and to eliminate routes that do not exist in all the three databases. However, joining the datasets this way was not possible because of the lack of computer memory.

The number of distinct routes for the tickets and flights datasets can be shown as

```
tickets %>% group_by(origin_IATA_code, dest_IATA_code) %>%
  select(origin_IATA_code, dest_IATA_code) %>% unique() %>% nrow()
```

```
## [1] 40145
```

```
flights %>% group_by(origin_IATA_code, dest_IATA_code) %>%
  select(origin_IATA_code, dest_IATA_code) %>% unique() %>% nrow()
```

```
## [1] 5917
```

Delete rows containing NAs which are generated because of joining of the tickets database (from fare_dat df) and a portion of the flights dataset (the cost df).

```
cost_revenue <- cost_revenue %>% drop_na()
```

Part 3: Visualization

Before starting to visualize data, it's important to check for the number of datapoints available for each route. This is because some flights may be assumed profitable but only have a small number of data. Therefore, it will be difficult to make reliable prediction.

```
#First we find the number of times there were flights in each routes is computed.
n_routes <- cost_revenue %>%
  group_by(route) %>%
  summarise(n = n(), profit) %>%
  arrange(desc(profit))
```

The frequency of flights in each route ranges as follows:

```
range(n_routes$n)
```

```
## [1]    1 4175
```

Routes where frequency of flights was below 20 were eliminated from consideration.

```
s <- 20 #minimum sample size
top_profitable_routes <- n_routes %>% filter(n >= s)  #disregard data with small sample
 size
```

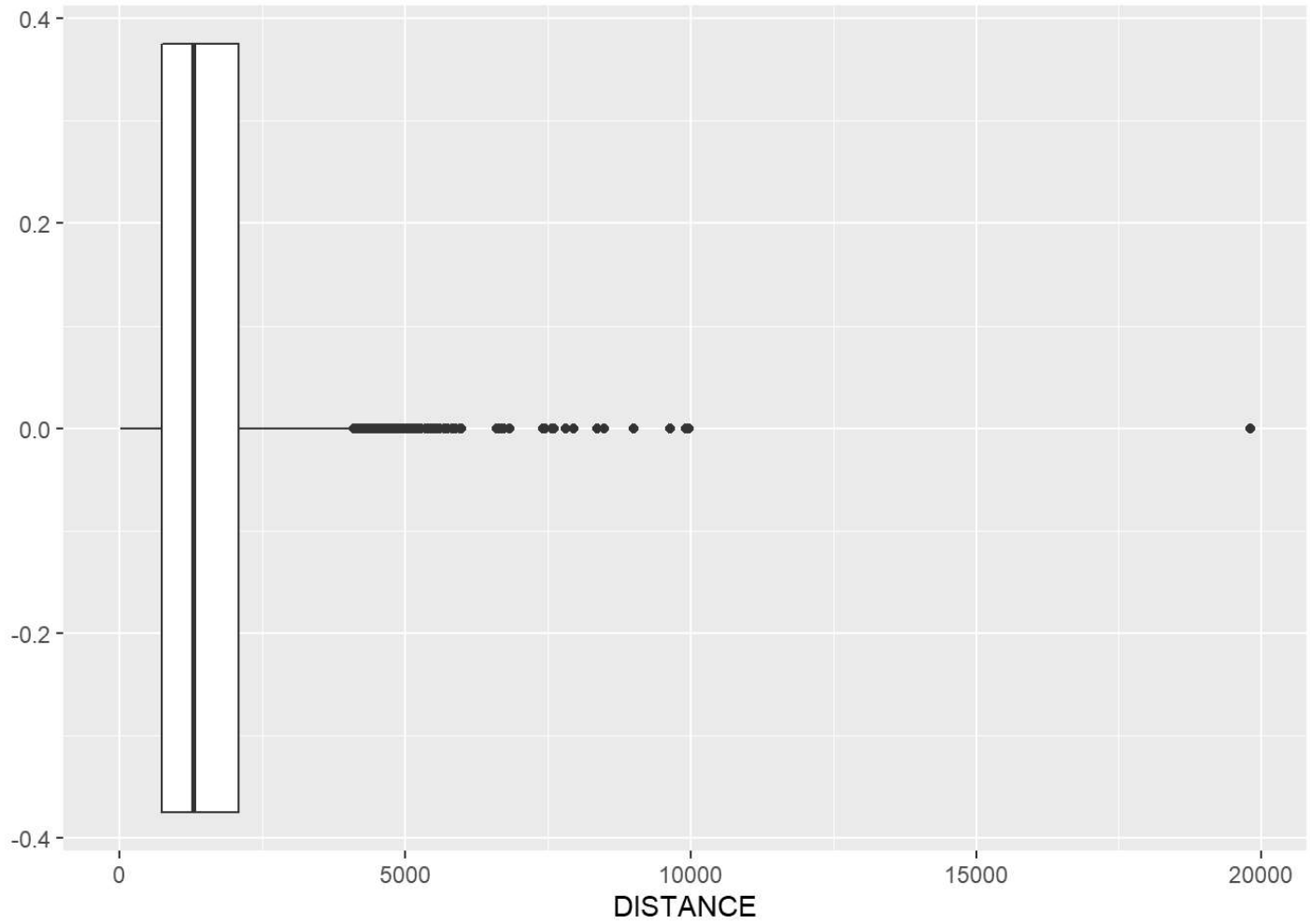
The top 15 routes based on average profitability

```
top_10_routes <- top_profitable_routes %>% group_by(route) %>%
  summarize(ave_profit = round(mean(profit)), n = mean(n)) %>%
  arrange(desc(ave_profit)) %>%
  head(10)
top_10_routes
```

```
## # A tibble: 10 x 3
##    route   ave_profit     n
##    <chr>        <dbl> <dbl>
##  1 SLC-TWF    1303294   256
##  2 EGE-JFK     193260    57
##  3 PIH-SLC     165108   270
##  4 EYW-ORD     149671    65
##  5 HNL-GUM     134005    88
##  6 EGE-DEN      99279   219
##  7 EWR-XNA      98066    78
##  8 EYW-DFW      90615    89
##  9 IAH-MOB      88566   413
## 10 GSP-CLT      88471   772
```

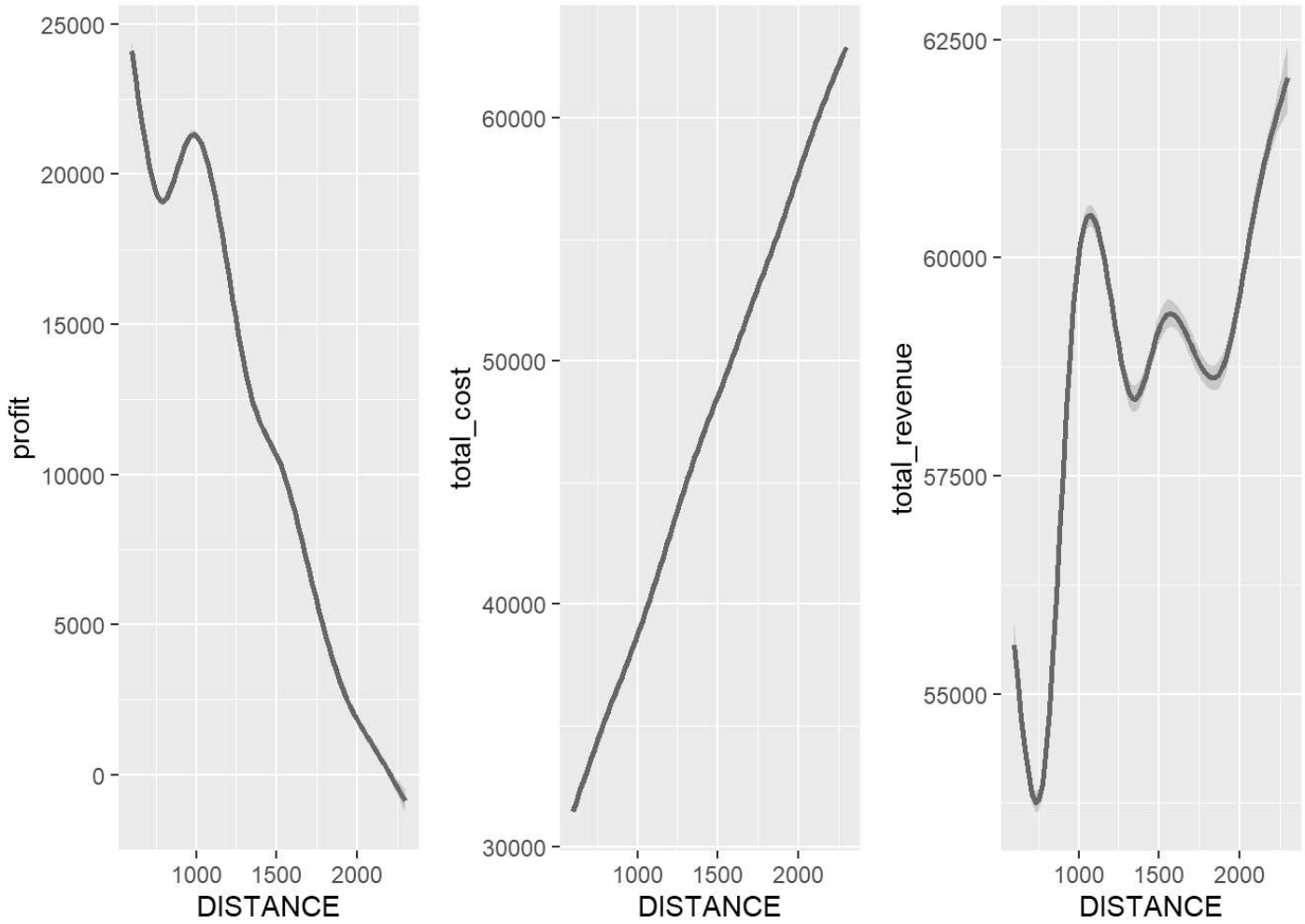
Effect of distance Distance is a major factor that contributes towards the cost as well as the profit. With an increase in distance, there will be an increase in fare and cost. The following graphs show how distance impacts cost, revenue, and overall profit. First it may be useful to see the range of distance values for the majority of the flights

```
cost_revenue %>% ggplot(aes(DISTANCE)) + geom_boxplot() #boxplot of distance
```



It can be seen that the majority of the distances lie in the range of about 600 and 2300 miles. Effect of distance was investigated using these range of values.

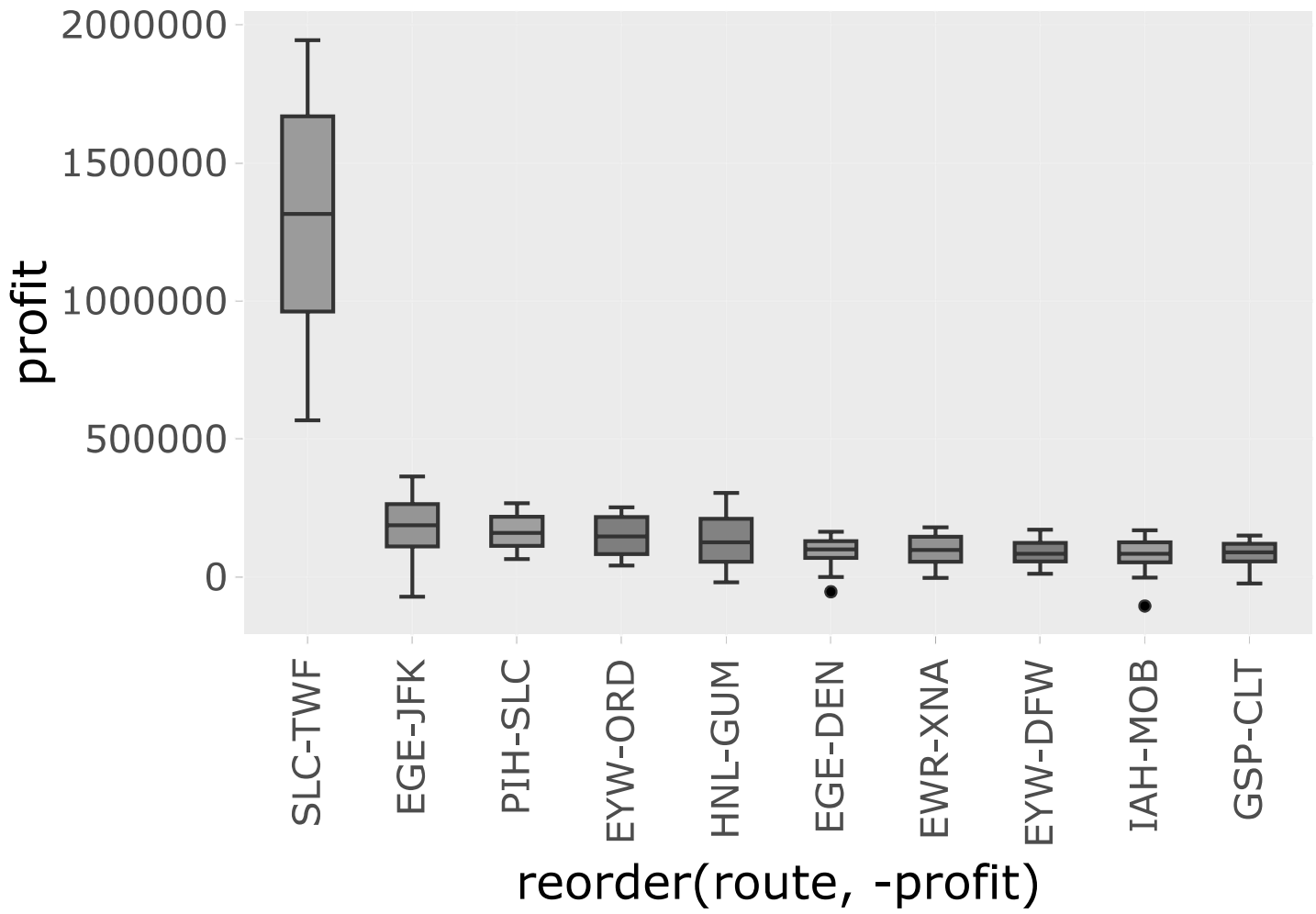
```
r1 <- cost_revenue %>% filter(DISTANCE > 600 & DISTANCE < 2300) %>%
  ggplot(aes(DISTANCE, profit)) + geom_smooth()
r2 <- cost_revenue %>% filter(DISTANCE > 600 & DISTANCE < 2300) %>%
  ggplot(aes(DISTANCE, total_cost)) + geom_smooth()
r3 <- cost_revenue %>% filter(DISTANCE > 600 & DISTANCE < 2300) %>%
  ggplot(aes(DISTANCE, total_revenue)) + geom_smooth()
gridExtra::grid.arrange(r1, r2, r3, nrow = 1)
```



The graphs show that, generally, profit decreases with distance.

Show the distribution of profits of the top most profitable routes

```
p1 <- top_profitable_routes %>%
  filter(route  %in% top_10_routes$route) %>%
  arrange(desc(profit)) %>%
  ggplot(aes(x = reorder(route, -profit), profit, fill = route)) +
  geom_boxplot() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5),
        text = element_text(size = 18), legend.position = "none")
ggplotly(p1, tooltip = "all")
```



The boxplots show the top profitble routes and their median and ranges of profits. It can be clearly seen that the route SLC-TWF seems that it is significantly profitable than the rest. The main factor of high revenue is high fare. Let's look at the fare for this route.

```
t2 <- cost_revenue %>% filter(route == "SLC-TWF")
range(t2$fare) #Range of fare for the flight
```

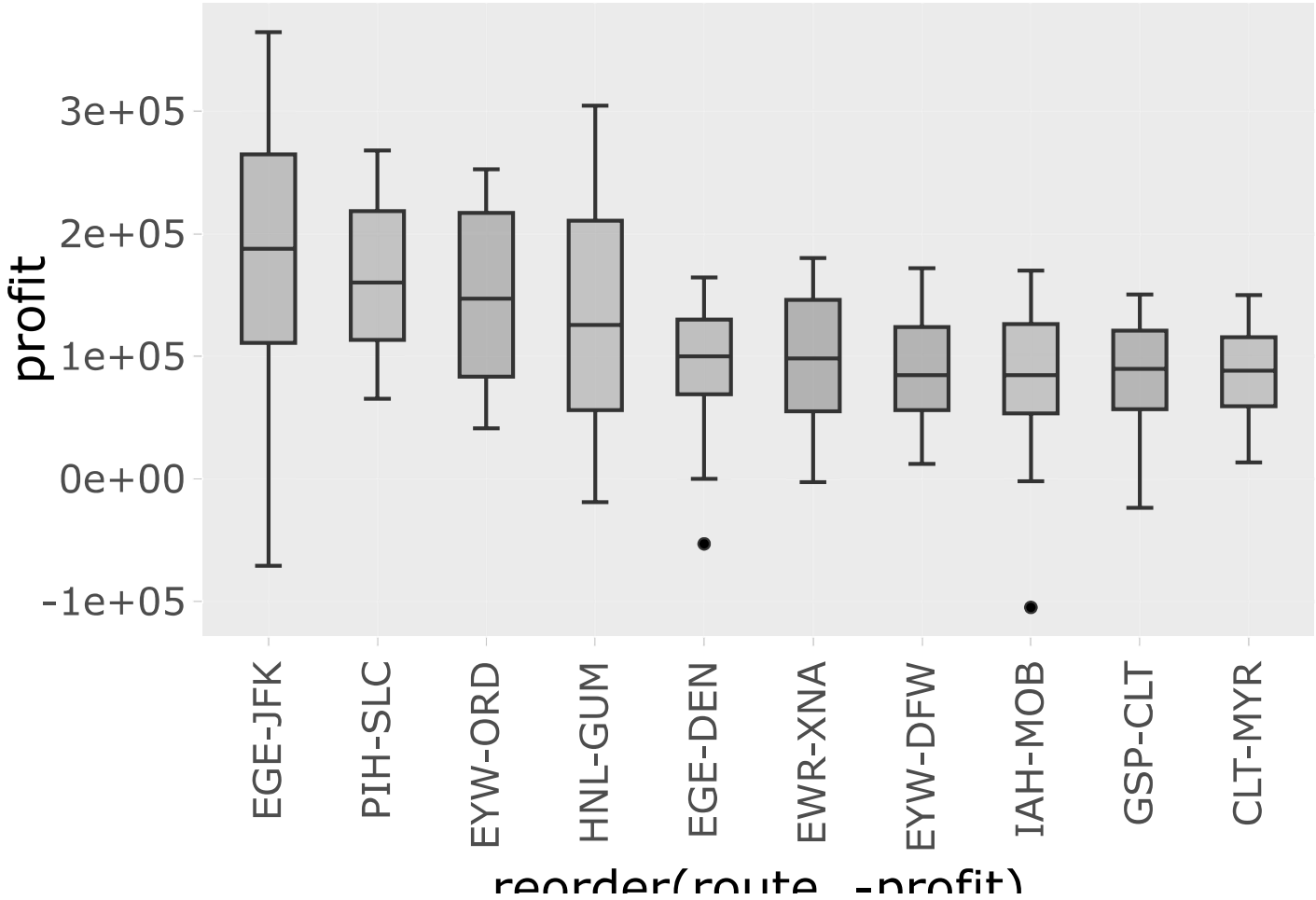
```
## [1] 9794.75 9794.75
```

The fare is too high considering the flight is relatively short. This, therefore is considered an outlier resulted because of incorrect fare entry in the dataset.The route information should, therefore, be eliminated from consideration. Update the top 10 most profitable routes

```
cost_revenue <- cost_revenue %>% filter(!(route == "SLC-TWF")) #remove outlier
top_profitable_routes <- top_profitable_routes %>% filter(!(route == "SLC-TWF"))
n_routes <- n_routes %>% filter(!(route == "SLC-TWF"))
top_10_routes <- top_profitable_routes %>% group_by(route) %>%
  summarize(ave_profit = round(mean(profit)), n = mean(n)) %>%
  arrange(desc(ave_profit)) %>%
  head(10)
```

Updated boxplot of the top profitable routes is shown below.

```
p2 <- top_profitable_routes %>%
  filter(route  %in% top_10_routes$route) %>%
  arrange(desc(profit)) %>%
  ggplot(aes(x = reorder(route, -profit), profit, fill = route)) +
  geom_boxplot(alpha = 0.5) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5),
        text = element_text(size = 18), legend.position = "none")
ggplotly(p2, tooltip = "all")
```

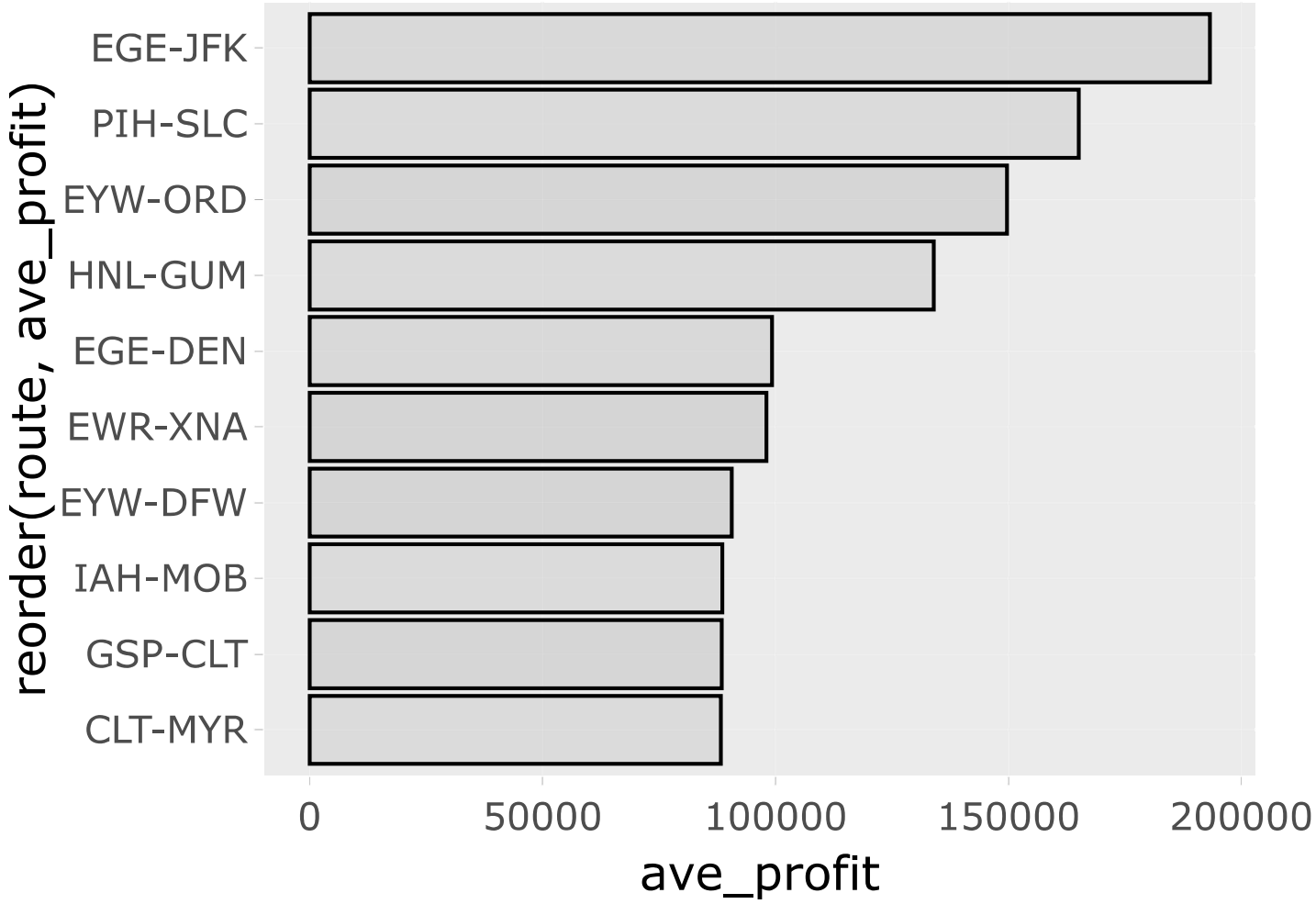


reorder(route, ave_profit)

Top routes by average profitability can also be shown.

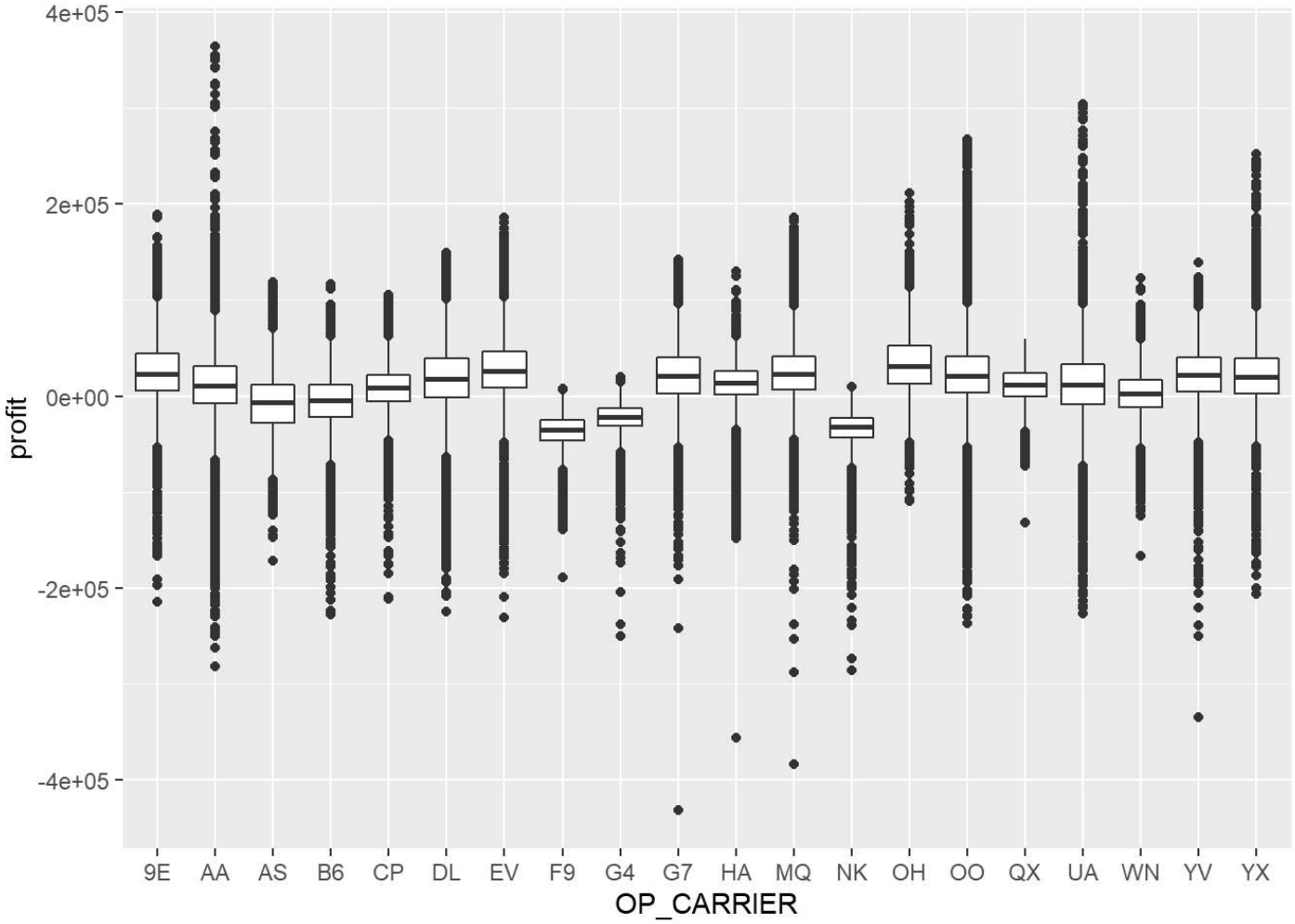
```
# profit of different routes
p21 <- top_profitable_routes %>% group_by(route) %>%
  summarise(ave_profit = mean(profit)) %>%
  arrange(desc(ave_profit)) %>% head(10) %>%
  ggplot(aes(x = reorder(route, ave_profit), ave_profit, fill = route)) +
     geom_bar(stat = "identity", color = "black", alpha = 0.2) +
     coord_flip() +
  theme(text = element_text(size = 18), legend.position = "none") +
  scale_color_brewer("Dark2")

ggplotly(p21, tooltip = "all")
```



Average occupancy rate versus profit for top profitable carriers

```
cost_revenue %>%
  group_by(OP_CARRIER) %>%
  arrange(desc(profit)) %>%
  ggplot(aes(OP_CARRIER, profit)) +
  geom_boxplot()
```



#Based on the information obtained from average profitability and the distribution of profitability, the top best 5 routes for investment are * EGE-JFK, * PIH-SLC, * EIY-ORD, * HNL-GUM, and * EGE-DEN.

Breakeven Analysis

Here, the number of round-trips required for the aircraft costs to be returned is calculated.

```
cap_cost <- 90e6 #initial cost of aircraft
top_10_routes %>% mutate(n_breakeven_routes = ceiling(cap_cost/ave_profit)) %>%
  select(route, ave_profit, n_breakeven_routes)
```

```
## # A tibble: 10 x 3
##    route   ave_profit n_breakeven_routes
##    <chr>        <dbl>               <dbl>
##  1 EGE-JFK     193260                 466
##  2 PIH-SLC     165108                 546
##  3 EYW-ORD     149671                 602
##  4 HNL-GUM     134005                 672
##  5 EGE-DEN      99279                 907
##  6 EWR-XNA      98066                 918
##  7 EYW-DFW      90615                 994
##  8 IAH-MOB      88566                1017
##  9 GSP-CLT      88471                1018
## 10 CLT-MYR      88280                1020
```

Part 4: Conclusions

#Based on the information obtained from average profitability and the distribution of profitability, the top best 5 routes for investment are

- EGE-JFK, (Eagle county regional, Gypsum, CO - JFK Airport, Queens, NY)
- PIH-SLC, (Pocatello regional, Pocatello, ID - Salt lake city Intl, UT)
- EIY-ORD, (Key West Intl, KeyWest FL - Chigaco OHare, Chicago, IL)
- HNL-GUM, (Inouye Intl, Honolulu, HI - Won Pat Intl, Guam)
- EGE-DEN. (Eagle county regional, Gypsum, CO - Denver Intl, Denver, CO)

The top key parameters that affect profitability are

- fare,
- distance, and
- occupancy rate

Distance is a big factor in profitability since other parameters such as fuel, crew, oil, maintenance, etc. are directly affected by this parameter. Therefore, the longer the distance is, the higher the cost will be. On the other hand, the biggest factors in profitability are fare and occupancy rate.

Part 5: Future Work

In order to compute breakeven analysis more accurately, it's important to consider the frequency of flights to the most profitable routes. Currently, the analysis was solely based on profitability per round-trip. It doesn't consider profitability in the quarter year. Profitability in the quarter year is affected by how often an aircraft flies in its assigned route. The frequency of flights, in turn, is determined by the need, i.e. availability of passengers (occupancy rate).

The following table shows some of the most profitable routes, frequency flights in those routes, and profit.

```
top_10_routes <- top_profitable_routes %>% group_by(route) %>%
  summarize(ave_profit = round(mean(profit)), n = mean(n)) %>%
  arrange(desc(ave_profit)) %>%
  head(10)
top_10_routes
```

```
## # A tibble: 10 x 3
##    route   ave_profit     n
##    <chr>        <dbl> <dbl>
##  1 EGE-JFK     193260    57
##  2 PIH-SLC     165108   270
##  3 EYW-ORD     149671    65
##  4 HNL-GUM     134005    88
##  5 EGE-DEN      99279   219
##  6 EWR-XNA      98066    78
##  7 EYW-DFW      90615    89
##  8 IAH-MOB      88566   413
##  9 GSP-CLT      88471   772
## 10 CLT-MYR      88280   601
```

For example, further analysis may be required to determine the overall profitability of the first route (EGE-JFK) compared to the second one (PIH-SLC). Although the first one is the most profitable per round-trip, the second route is likely to allow more frequent flights, and, therefore, may result in a higher overall profit per year.