Zürich (CH) Airport

Prediction of delays

Alexis Barrière

13/02/2020

Contents

Introduction

This project will use data of planned and effective flight arrival and departure-times from / to the airport of Zurich in Switzerland for the entire year 2017.
These data have been collected for the TWIST 2018 Hackdays in Zürich (https://www.twist2018.ch/).
Some work has already been done by the team to enhance and clean the data:

- add information concerning the origin or destination airport as the geographical data and the distance (source : http://ourairports.com/data/)
- add weather information concerning Zürich airport (temperature, precipitation, lightning, humidity, ...) (source : MeteoSchweiz, free only for universities)
- compute delays from effective minus planned time

The goal of this project is to find the best prediction of delays and find the best predictors of the delay (company, airplane type, weather, distance, ...)

Data Analysis

Data Loading

The data can be retrieve from Github:

```
url <- "https://github.com/tlorusso/twist_zrh/raw/master/twist_zrh.RDS"
temp <- tempfile() # create a tempfile
download.file(url, temp) # download to disk
dat <- readRDS(temp) # read the tempfile
unlink(temp) # Deletes tempfile
```

but for some reason the readRDS function is failing with this direct download. I have then dowloaded the file on my repository and then I have been able to read the file:

```
twist_zrh_clean <- readRDS("./data/twist_zrh_cleaned.RDS")
```

We can then see that our dataset have 236'351 observables and 33 variables.

The first step is to clean some of our variables, because they are factors but have not been "refactored" after the first cleaning. We will also create some more variables to make our analysis (month, hour and flight type). And then delete some uninteresting variables

```
flightdata <- twist_zrh_clean %>%
  mutate(continent=as.character(continent),
         airline_name=as.character(airline_name),
         airplane_type=as.character(airplane_type),
         origin_destination_name=as.character(origin_destination_name),
         airport_type=as.character(airport_type),
         iso_country=as.character(iso_country),
         iso_region=as.character(iso_region),
         municipality=as.character(municipality),
         month = month(date),
         hour = hour(planed_time),
         flight_type=ifelse(start_landing=="L","Landing","Take-Off"))
flightdata <- flightdata %>%
  mutate(continent=replace_na(continent,"NA"))
flightdata <- flightdata %>%
  mutate(continent=as.factor(continent),
         airline_name=as.factor(airline_name),
         airplane_type=as.factor(airplane_type),
```

```
         airport_type=as.factor(airport_type),
         iso_country=as.factor(iso_country),
         iso_region=as.factor(iso_region),
         month = as.factor(month),
         hour = as.factor(hour),
         flight_type=as.factor(flight_type)) %>%
  rename(c("delay_sec"="diff_in_secs"))
flightdata <- flightdata %>%
  select(-tde200h0) %>% select(-start_landing)
```

Unfortunately the weather data are finishing on the 30^{th} of December, so in order to have a cleaned dataset with all relevant variables, we will delete data of the 31^{st} of December.

```
flightdata <- flightdata %>% filter(!is.na(temp_avg))
```

Data Exploration

Now we can have a look at the variable we want to predict, the delay:

Table 1: Landing Delay summary statistics

Min	First.Qu.	Median	Mean	Third.Qu.	Max
-82143	-408	71	347.0833	717	40306

Table 2: Take-Off Delay summary statistics

Min	First.Qu.	Median	Mean	Third.Qu.	Max
-80590	0	300	644.4593	840	55206

We will split the data by Flight Type as it's really two different thing.

To have more relevant data, we will only take airlines with at least 500 flights (arrival or departure) during the year:

```
airline <- flightdata %>%
  group_by(airline_name) %>%
  summarize(NbFlights=n()) %>%
  filter(NbFlights>500) %>%
  select(airline_name)

flightdata <- inner_join(flightdata,airline,by="airline_name")

flightdata$airline_name <- factor(flightdata$airline_name)
flightdata$airplane_type <- factor(flightdata$airplane_type)
```

We can also add another categorization variable based on the delay which tell if a flight is early (delay in seconds < 0), if there is no delay (delay between 0 seconds and 15 minutes) or if there is a delay (more than 15 minutes of delay):

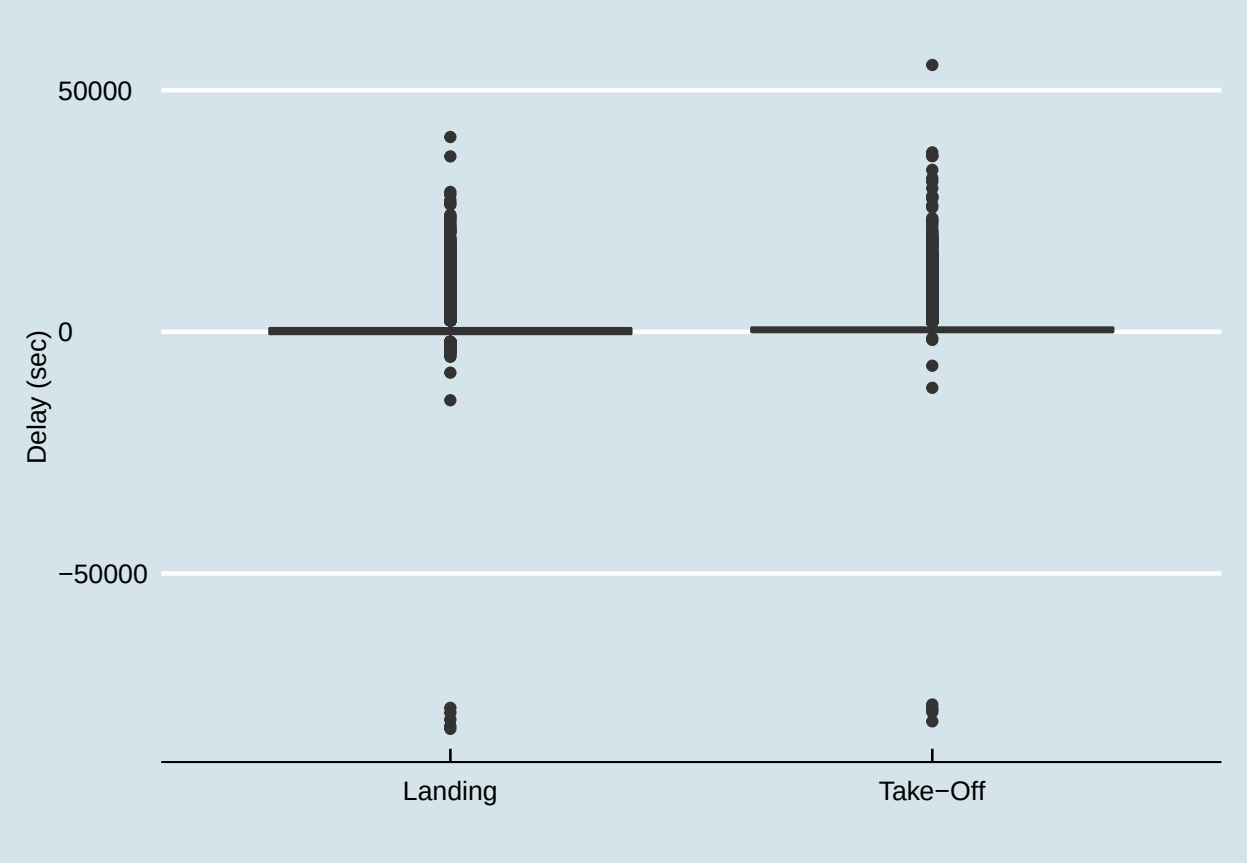
```
flightdata <- flightdata %>%
  mutate(delay_cat = cut(as.numeric(delay_sec),
                         breaks = c(-Inf,0,900,Inf),
                         labels = c("Early","No Delay","Delay")))
```

Which gives us this split :

Table 3: Number of Flights by Delay Category

Flight Type	Delay Category	Nb of Flights
Landing	Early	53'680
Landing	No Delay	37'955
Landing	Delay	23'783
Take-Off	Early	29'386
Take-Off	No Delay	59'832
Take-Off	Delay	26'346

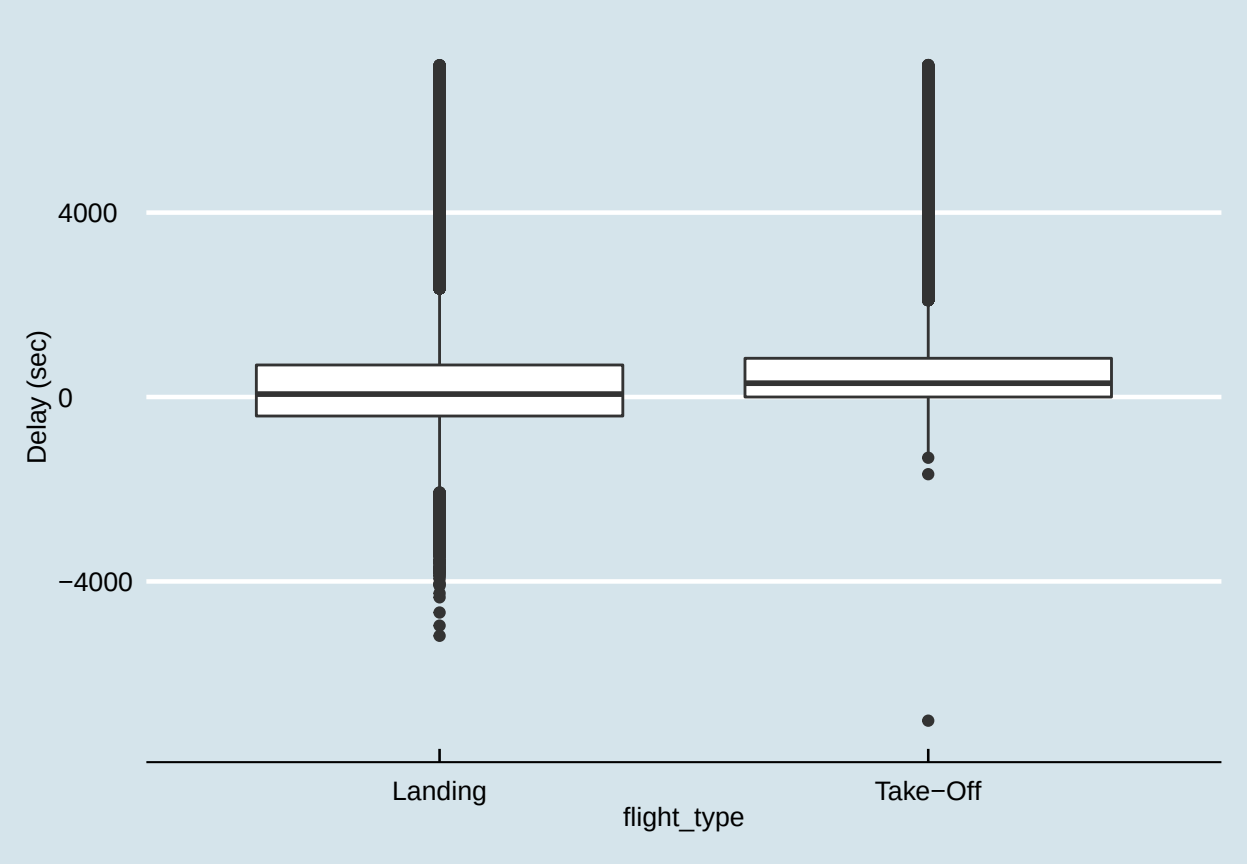
In the table 1 and 2, we see that we have extreme values that seems way off our inner quartile. Let's see it with a boxplot:



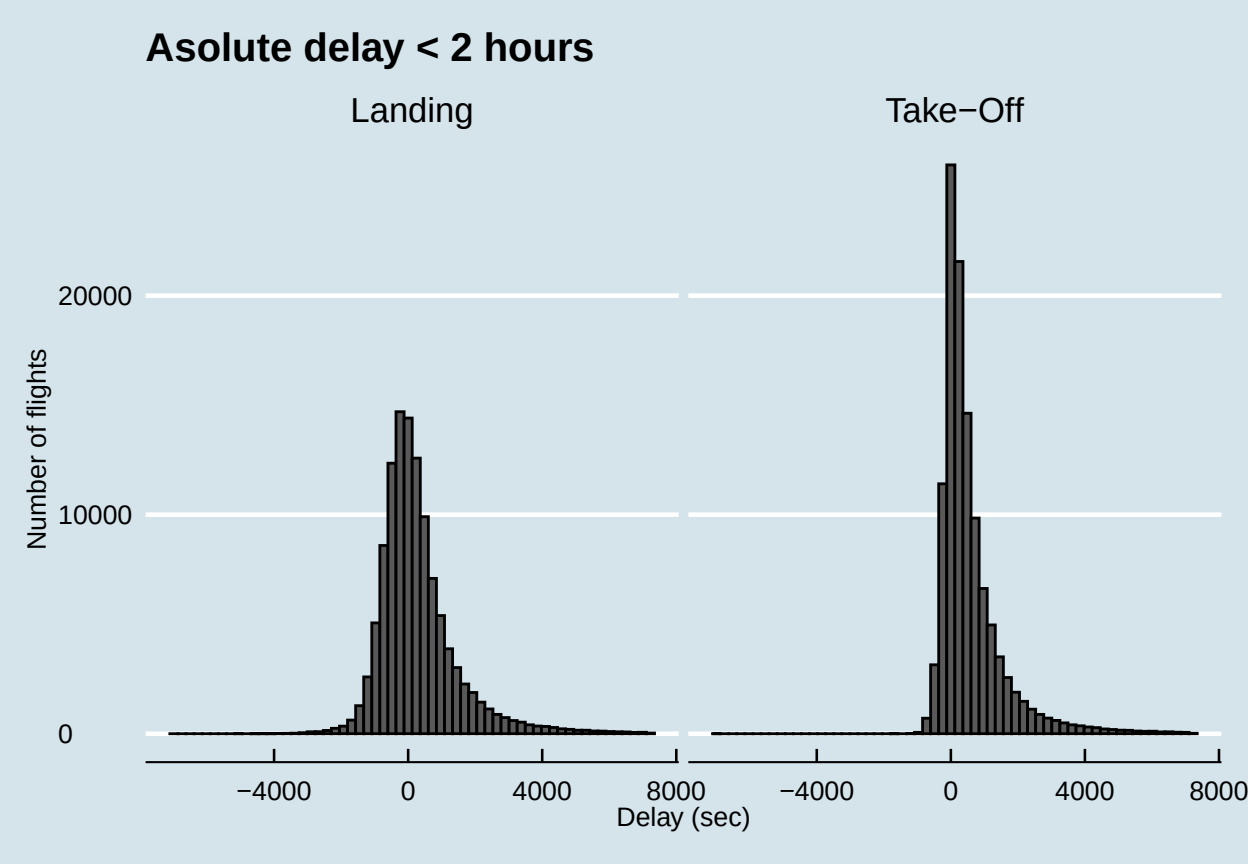
We can see that the values above 50'000 seconds and below -50'000 seconds are not representative (outliers) and should be deleted to have more relevant data:

```
flightdata <- flightdata %>% filter(abs(delay_sec)<50000)
```

Now let's just have a look closer at the values more around 0 (no delay) :



So it seems we have different patterns in our two types of flight. Let's see it whith histograms:

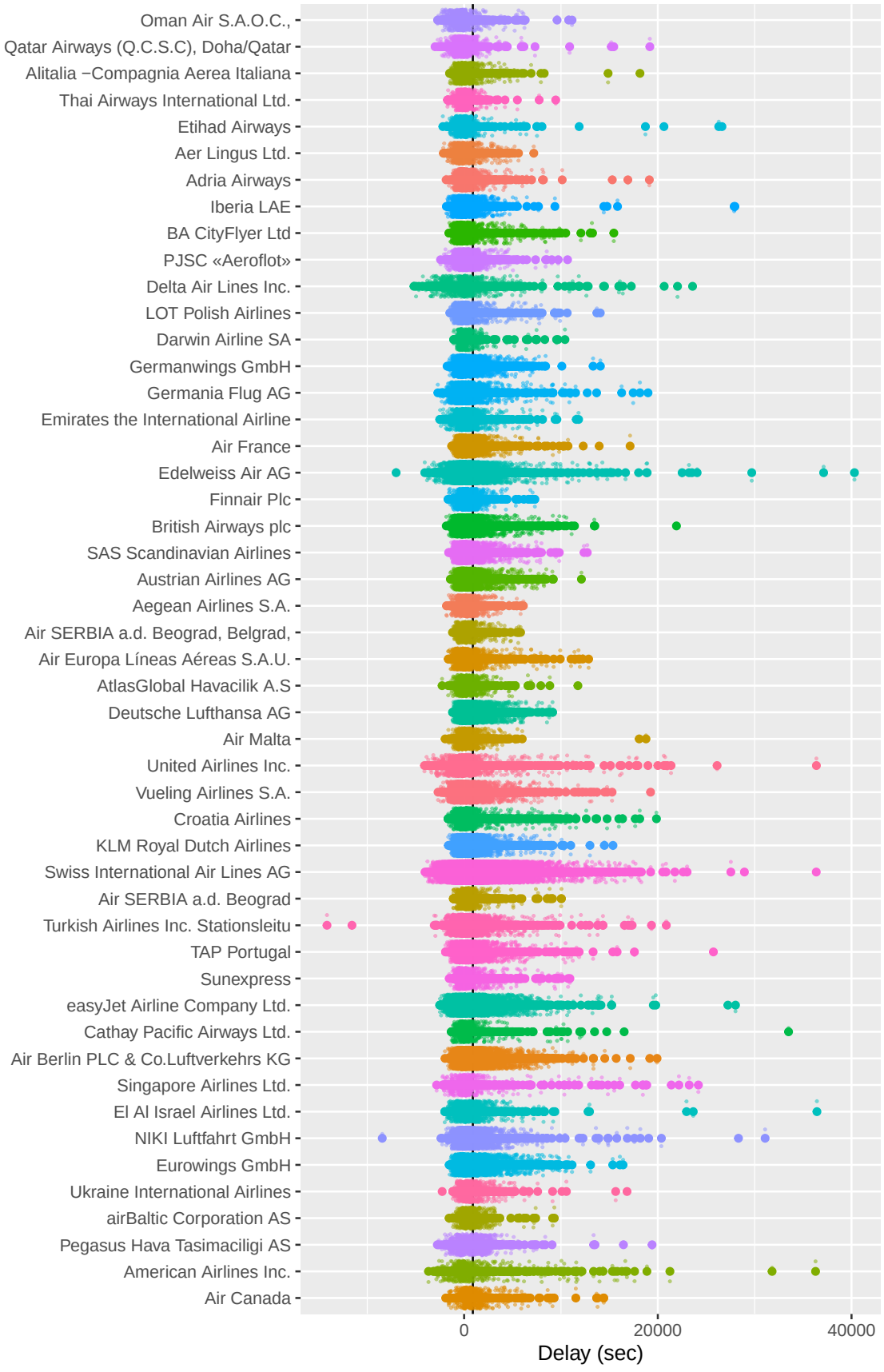


So, what we see is:

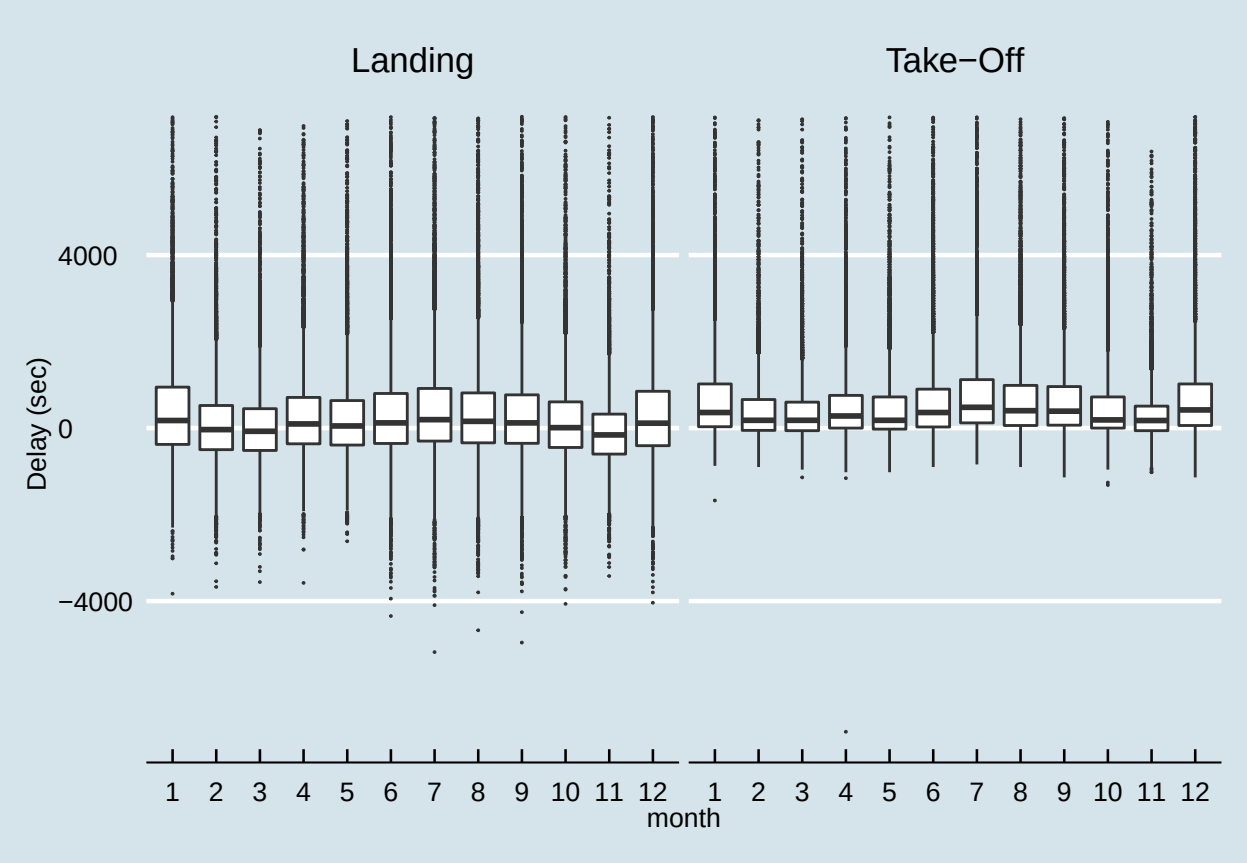
- For Landing flight type, the data seem to be more symetric aroung 0 (as much delayed than early flights)
- For Take-Off type, the data are skewed to the right (more delayed flights than early flights)

So in our analysis, we will process this two datasets separately.

We can then have a view on the spread of the delay by airlines:


Oman Air S.A.O.C.,
Qatar Airways (Q.C.S.C), Doha/Qatar
Alitalia −Compagnia Aerea Italiana
Thai Airways International Ltd.
Etihad Airways
Aer Lingus Ltd.
Adria Airways
Iberia LAE
BA CityFlyer Ltd
PJSC «Aeroflot»
Delta Air Lines Inc.
LOT Polish Airlines
Darwin Airline SA
Germanwings GmbH
Germania Flug AG
Emirates the International Airline
Air France
Edelweiss Air AG
Finnair Plc
British Airways plc
SAS Scandinavian Airlines
Austrian Airlines AG
Aegean Airlines S.A.
Air SERBIA a.d. Beograd, Belgrad,
Air Europa Líneas Aéreas S.A.U.
AtlasGlobal Havacilik A.S
Deutsche Lufthansa AG
Air Malta
United Airlines Inc.
Vueling Airlines S.A.
Croatia Airlines
KLM Royal Dutch Airlines
Swiss International Air Lines AG
Air SERBIA a.d. Beograd
Turkish Airlines Inc. Stationsleitu
TAP Portugal
Sunexpress
easyJet Airline Company Ltd.
Cathay Pacific Airways Ltd.
Air Berlin PLC & Co.Luftverkehrs KG
Singapore Airlines Ltd.
El Al Israel Airlines Ltd.
NIKI Luftfahrt GmbH
Eurowings GmbH
Ukraine International Airlines
airBaltic Corporation AS
Pegasus Hava Tasimaciligi AS
American Airlines Inc.
Air Canada
0
20000
40000
Delay (sec)

We can also have a look at the delays based on month of the year:

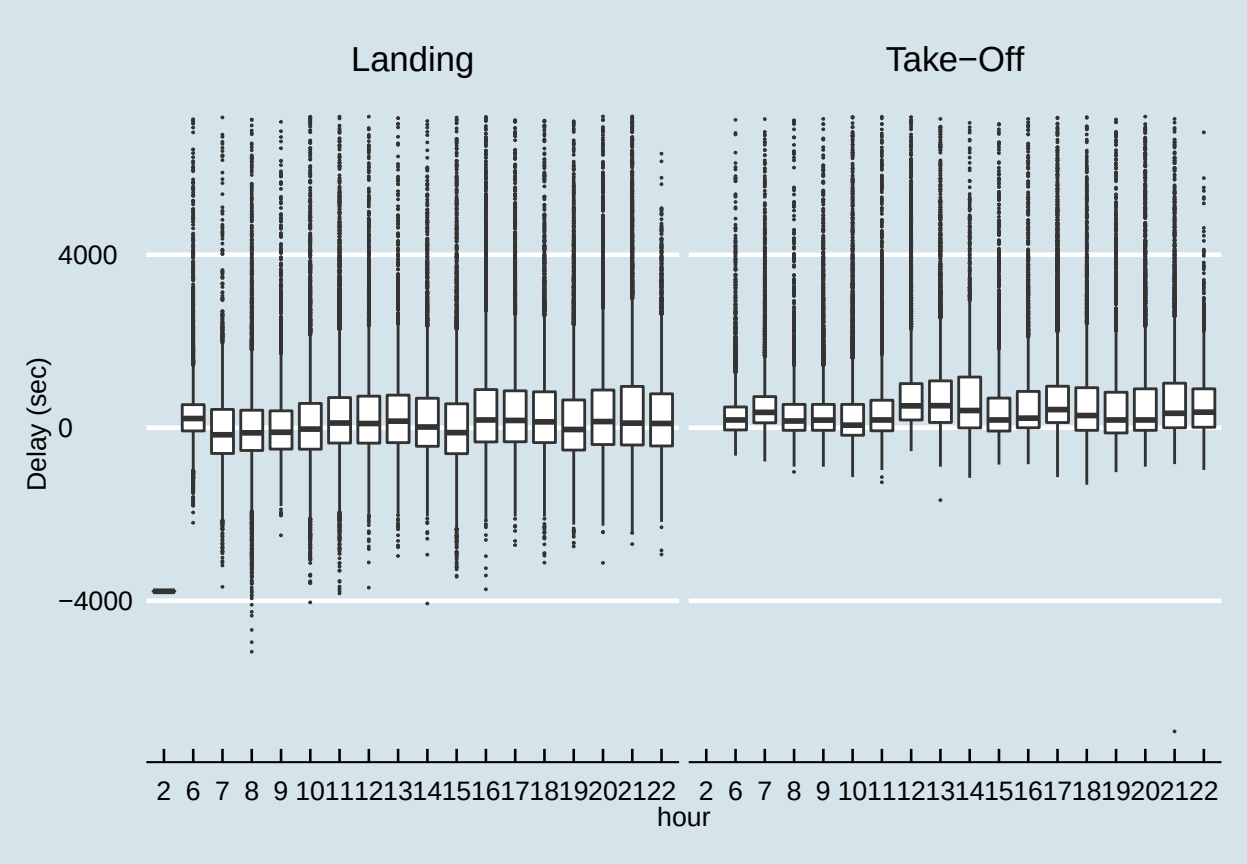


This is clearly a seasonal effect due to the fact that there is more flights during End of Year season and Summer Holiday season:

Table 4: Flights by Month detail

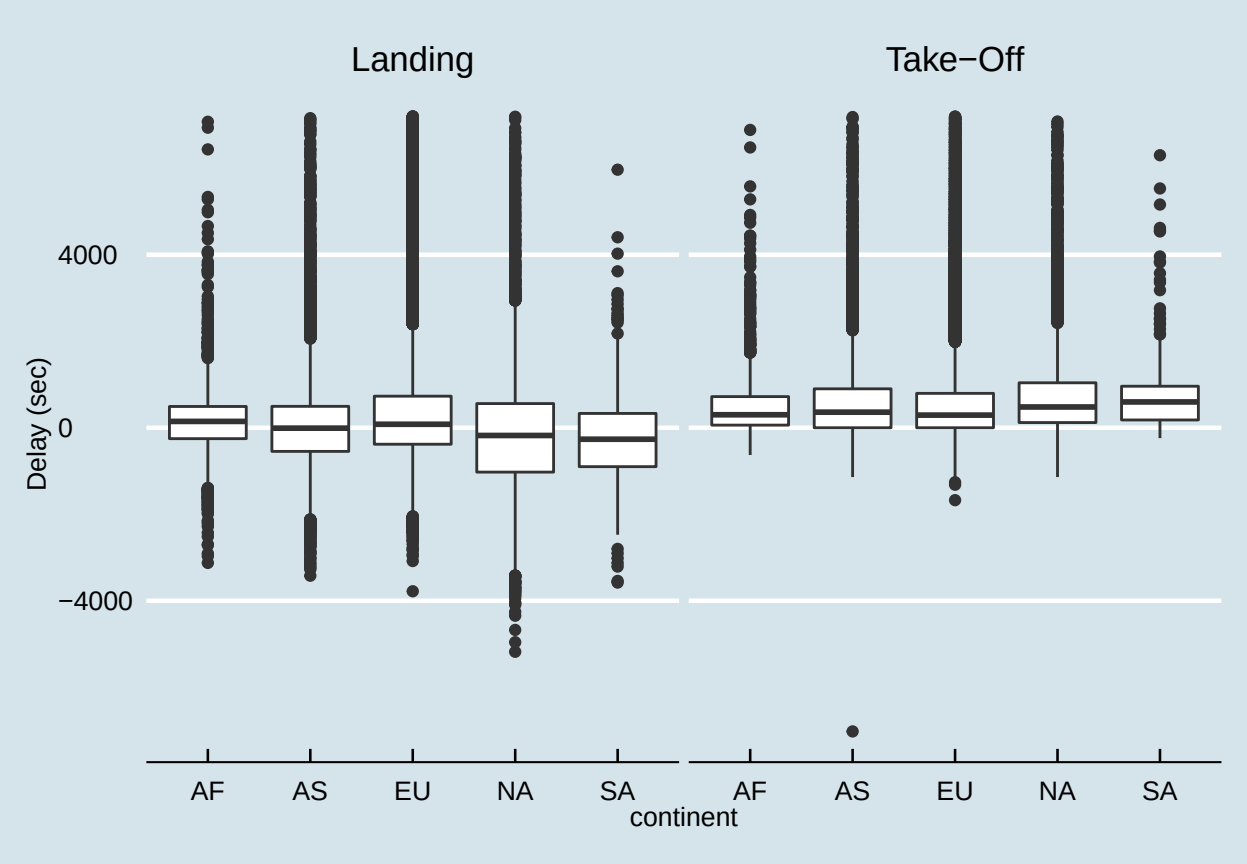
month	Nb Flights	Mean delay
1	17'330	576
2	16'067	299
3	18'480	236
4	19'356	424
5	20'423	368
6	20'107	515
7	21'320	632
8	21'323	540
9	20'596	511
10	20'608	347
11	17'352	147
12	16'733	574

The data can also be explored by hour of the day:



At least we could have expected a delay increasing as the day went on especially for Take-Off data, but the mean delay is more or less constant. The only thing we notice is that the interquartile is larger at the end of the day that at the beginning, meaning less more variability in the delays.

The last analysis that can help to understand our data is a geographical analysis of the origin or destination of the flights:



For the landing part, we clearly see that the gulf stream allows to have a majority of flights to be early. Unlike of the Take-Off flights for America that tends to be always late.

Prediction, Analysis & Results

The first step will be to split our dataset by Flight Type, Landing or Take Off. We will also select only a subset of variables to lighten the analysis by keeping only (Airline, Airplane Type, Distance, Windspeed average, Precipitation, temperature average, Air Pressure, Month and Hour) Finally, we will modify the value we want to predict to make it discrete. The goal will be to predict if a flight is delayed by more than 15 minutes or not.

```
# Create two datasets for Landing and Take-Off and select meaningfull variable
Landing <- flightdata %>% filter(flight_type=="Landing") %>%
  mutate(delay = as.numeric(delay_sec), delayYN=as.factor((delay>900))) %>%
  select(delayYN,airline_name, airplane_type,distance_km,windspeed_avg_h,
         precip,temp_avg,airpres,hour,month)
TakeOff <- flightdata %>% filter(flight_type=="Take-Off") %>%
  mutate(delay = as.numeric(delay_sec), delayYN=as.factor((delay>900))) %>%
  select(delayYN,airline_name, airplane_type,distance_km,windspeed_avg_h,
         precip,temp_avg,airpres,hour,month)
```

First we will split the Take-Off dataset in a training and a test dataset. The test dataset will have around 30% of the data, making sure all airlines, airplane type, month and hour that are in it are also in the training dataset.

```
set.seed(2020)
TakeOff_index <- createDataPartition(TakeOff$delay, times = 1, p = 0.3, list = FALSE)
```

```
TakeOff_test_set <- TakeOff[TakeOff_index, ]
TakeOff_train_set <- TakeOff[-TakeOff_index, ]

temp <- TakeOff[TakeOff_index, ]
# Make sure Ariline in test set are also in training set
TakeOff_test_set <- temp %>%
  semi_join(TakeOff_train_set, by = "airline_name") %>%
  semi_join(TakeOff_train_set, by = "airplane_type") %>%
  semi_join(TakeOff_train_set, by = "month") %>%
  semi_join(TakeOff_train_set, by = "hour")


# Add rows removed from test set back into training set
removed <- anti_join(temp, TakeOff_train_set)
TakeOff_train_set <- rbind(TakeOff_train_set, removed)

rm(TakeOff_index, temp, removed)
```

We will first try to do a prediction using the logistic regression model using the airline as predictor and then predict with our test set :

```
glm_TO_A <- train(delayYN ~ airline_name,
                  method = "glm", data = TakeOff_train_set)

delay_hat_glm_TO_A <- predict(glm_TO_A, TakeOff_test_set)
```

This gives us the following metrics:

Method	Predictors	Accuracy	Sensitivity	Specifictiy
GLM	Airline	0.7739	0.9993	0.0108

We can see that even if the Accuracy is quite correct and the Sensitivity really good, we have a very bad Specificity which means the algorithm is really bad at predicting a delay when there is one. Let's try by adding two more predictors that seems to have importance in our exploratory work, Distance and Hour:

```
glm_TO_AHD <- train(delayYN ~ airline_name+hour+distance_km,
                    method = "glm", data = TakeOff_train_set)

delay_hat_glm_TO_AHD <- predict(glm_TO_AHD, TakeOff_test_set)
```

Which gives us:

Method	Predictors	Accuracy	Sensitivity	Specifictiy
GLM	Airline	0.7739	0.9993	0.0108
GLM	Airline+Hour+Distance	0.7739	0.9993	0.0108

The results are a bit better. Now we can try with all our predictors:

```
glm_TO_Full <- train(delayYN ~ airline_name+airplane_type+distance_km+
                       windspeed_avg_h+precip+temp_avg+airpres+hour+month,
                     method = "glm", data = TakeOff_train_set)

delay_hat_glm_TO_Full <- predict(glm_TO_Full, TakeOff_test_set)
```

We can see in the following table that the results are better, especially concerning Specificity, even if it's still not good :

Method	Predictors	Accuracy	Sensitivity	Specifictiy
GLM	Airline	0.7739	0.9993	0.0108
GLM	Airline+Hour+Distance	0.7739	0.9993	0.0108
GLM	All	0.7761	0.9922	0.0445

We will then try another algorithm to see if we can have better results. So let's try thhe Random Forest algorithm with the 3 same steps :

```
randfor_TO_A <- randomForest(delayYN ~ airline_name,
                             data = TakeOff_train_set)
delay_hat_randfor_TO_A <- predict(randfor_TO_A, TakeOff_test_set)

randfor_TO_AHD <- randomForest(delayYN ~ airline_name+hour+distance_km,
                               data = TakeOff_train_set)
delay_hat_randfor_TO_AHD <- predict(randfor_TO_AHD, TakeOff_test_set)

randfor_TO_Full <- randomForest(delayYN ~ airline_name+airplane_type+distance_km+
                                  windspeed_avg_h+precip+temp_avg+airpres+hour+month,
                                data = TakeOff_train_set)
delay_hat_randfor_TO_Full <- predict(randfor_TO_Full, TakeOff_test_set)
```

This gives us the following results:

Table 8: Take-Off Results

Method	Predictors	Accuracy	Sensitivity	Specifictiy
GLM	Airline	0.7739	0.9993	0.0108
GLM	Airline+Hour+Distance	0.7739	0.9993	0.0108
GLM	All	0.7761	0.9922	0.0445
Random Forest	Airline	0.7739	0.9993	0.0108
Random Forest	Airline+Hour+Distance	0.7752	0.9962	0.0272
Random Forest	All	0.9873	0.9983	0.9499

So the first two steps give us approximately the same results as the other algorithm, but if we use all our predictors with the Random Forest, the results are really interesting. Let's look at the entire confusion matrix of this last prediction to see the details:

Table 9: Prediction Results

	FALSE	TRUE
FALSE	26719	396
TRUE	45	7508

Table 10: Model statistics

	x
Sensitivity	0.9983186
Specificity	0.9498988
Pos Pred Value	0.9853955
Neg Pred Value	0.9940421
Precision	0.9853955
Recall	0.9983186
F1	0.9918150
Prevalence	0.7720088
Detection Rate	0.7707107
Detection Prevalence	0.7821334
Balanced Accuracy	0.9741087

The first table confirms that we have still some miss predictions but really low compared to the good ones.

We can also do the same work for the Landing dataset:

```
set.seed(2020)
Landing_index <- createDataPartition(Landing$delayYN, times = 1, p = 0.3, list = FALSE)
Landing_train_set <- Landing[-Landing_index, ]

temp <- Landing[Landing_index, ]
# Make sure Ariline in test set are also in training set
Landing_test_set <- temp %>%
  semi_join(Landing_train_set, by = "airline_name") %>%
  semi_join(Landing_train_set, by = "airplane_type") %>%
  semi_join(Landing_train_set, by = "month") %>%
  semi_join(Landing_train_set, by = "hour")

# Add rows removed from test set back into training set
removed <- anti_join(temp, Landing_train_set)
Landing_train_set <- rbind(Landing_train_set, removed)

rm(Landing_index, temp, removed)


glm_LA_A <- train(delayYN ~ airline_name, method = "glm", data = Landing_train_set)
delay_hat_glm_LA_A <- predict(glm_LA_A, Landing_test_set)

glm_LA_AHD <- train(delayYN ~ airline_name+hour+distance_km, method = "glm", data = Landing_train_set)
delay_hat_glm_LA_AHD <- predict(glm_LA_AHD, Landing_test_set)

glm_LA_Full <- train(delayYN ~ airline_name+airplane_type+distance_km+windspeed_avg_h+precip+temp_avg+a
delay_hat_glm_LA_Full <- predict(glm_LA_Full, Landing_test_set)


randfor_LA_A <- randomForest(delayYN ~ airline_name, data = Landing_train_set)
delay_hat_randfor_LA_A <- predict(randfor_LA_A, Landing_test_set)

randfor_LA_AHD <- randomForest(delayYN ~ airline_name+hour+distance_km, data = Landing_train_set)
delay_hat_randfor_LA_AHD <- predict(randfor_LA_AHD, Landing_test_set)
```

```
randfor_LA_Full <- randomForest(delayYN ~ airline_name+airplane_type+distance_km+windspeed_avg_h+precip
delay_hat_randfor_LA_Full <- predict(randfor_LA_Full, Landing_test_set)
```

This gives us the following results for Landing

Table 11: Landing Results

Method	Predictors	Accuracy	Sensitivity	Specifictiy
GLM	Airline	0.7939	1.0000	0.0000
GLM	Airline+Hour+Distance	0.7939	1.0000	0.0000
GLM	All	0.7943	0.9967	0.0149
Random Forest	Airline	0.7939	1.0000	0.0000
Random Forest	Airline+Hour+Distance	0.7958	0.9976	0.0182
Random Forest	All	0.9920	0.9993	0.9638

We can see that we obtain the same type of result for Landing, confirming that the Random Forest Model is a good model for our analysis.

Conclusion

The results obtained with the Random Forest algorithm and considering all the predictors are really interesting and is a really good start to find a good algorithm to know if our flight will be delayed or not.

To improve our results, we could consider:

- tune our Random Forest algorithm with cross-validation to find the best parameters
- add weather condition of origin (for Landing flights) or destination (for Take-Off flights) airport.
- add information of number of passengers or fill percentage of airplane