# Homework 4

201982188 통계학과 박현주

2020-10-14

## 1. Which plane (tailnum) has the worst on-time record?

Consider two metrics:

1. proportion of flights not delayed or canceled.
2. mean arrival delay.

```
library(tidyverse)
library(nycflights13)
```

```
df <- flights

num_na <- function(x){
  sum(is.na(x))
}

df_na <- df %>% apply(.,2,num_na)

df_na[df_na != 0]%>% t()%>% knitr::kable(.)
```

| dep_time | dep_delay | arr_time | arr_delay | tailnum | air_time |
|---:|---:|---:|---:|---:|---:|
| 8255 | 8255 | 8713 | 9430 | 2512 | 9430 |

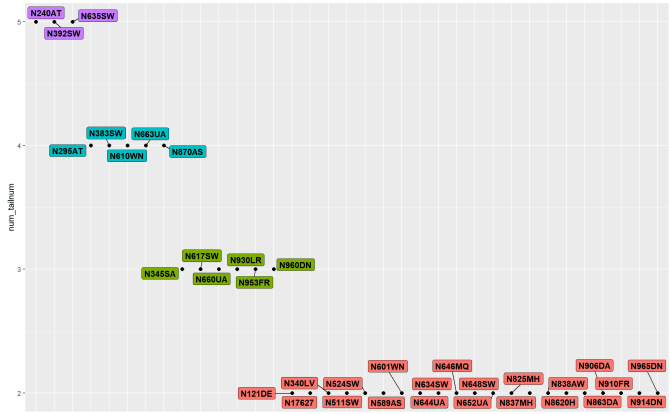
결측값을 포함하고 있는 변수는 위 테이블과 같다. 그 중에서 air_time이 NA일 때 그 비행기는 취소됐다고 생각할 수 있다.

```
df %>% filter(!is.na(tailnum)) %>%
  mutate(no_delay = ifelse(!is.na(air_time) & arr_delay<=0,1,0)) %>%
  group_by(tailnum) %>%
  summarise(num_tailnum = n() ,num_no_delay = sum(no_delay),no_delay_rate = num_no_delay/num_tailnum) %>%
  arrange(no_delay_rate) -> prob1

knitr::kable(head(prob1,15))
```

| tailnum | num_tailnum | num_no_delay | no_delay_rate |
|---|---:|---:|---:|
| N121DE | 2 | 0 | 0 |
| N136DL | 1 | 0 | 0 |
| N143DA | 1 | 0 | 0 |
| N17627 | 2 | 0 | 0 |
| N240AT | 5 | 0 | 0 |
| N26906 | 1 | 0 | 0 |
| N295AT | 4 | 0 | 0 |
| N302AS | 1 | 0 | 0 |
| N303AS | 1 | 0 | 0 |
| N32626 | 1 | 0 | 0 |
| N340LV | 2 | 0 | 0 |
| N345SA | 3 | 0 | 0 |
| N347SW | 1 | 0 | 0 |
| N383SW | 4 | 0 | 0 |
| N392SW | 5 | 0 | 0 |

tailnum이 결측값을 가지는 경우가 있으니 이를 제외하고 취소되거나 지연되지 않은 항공만을 고려해보자. 이와 같은 변수를 no_delay로 만들어 지연 또는 취소되지 않은 항공편 비율(no_delay_rate)을 계산했다. no_delay_rate가 0의 값을 가지는 plane이 정시 기록이 좋지 않다.




no_delay_rate가 0인 tailnum들의 운행 비행기 수는 1개 이상 5개 이하였다. 운행 비행기 수가 상대적으로 많은데 지연되지 않은 적이 없는 tailnum이 더욱 안좋은 정시 기록을 가진다. 위 그림은 no_delay_rate이 0인 tailnum 중 num_tailnum에 대한 그림이다. 해당 tailnum에는 N3925W, N240AT, N6365W가 있다.

## 2. What time of day should you fly if you want to avoid delays as much as possible?

- Hint : The earlier the flight is scheduled, the lower its expected delay.

2.1 각 시간에 따른 운행 비행기 수

| sd_time | 8 | 6 | 17 | 15 | 16 | 7 | 18 | 14 | 19 | 9 | 13 | 12 | 20 | 10 | 11 | 21 | 22 | 5 | 23 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| n | 27242 | 25951 | 24426 | 23888 | 23002 | 22821 | 21783 | 21706 | 21441 | 20312 | 19956 | 18181 | 16739 | 16708 | 16033 | 10933 | 2639 | 1953 | 1061 | 1 |

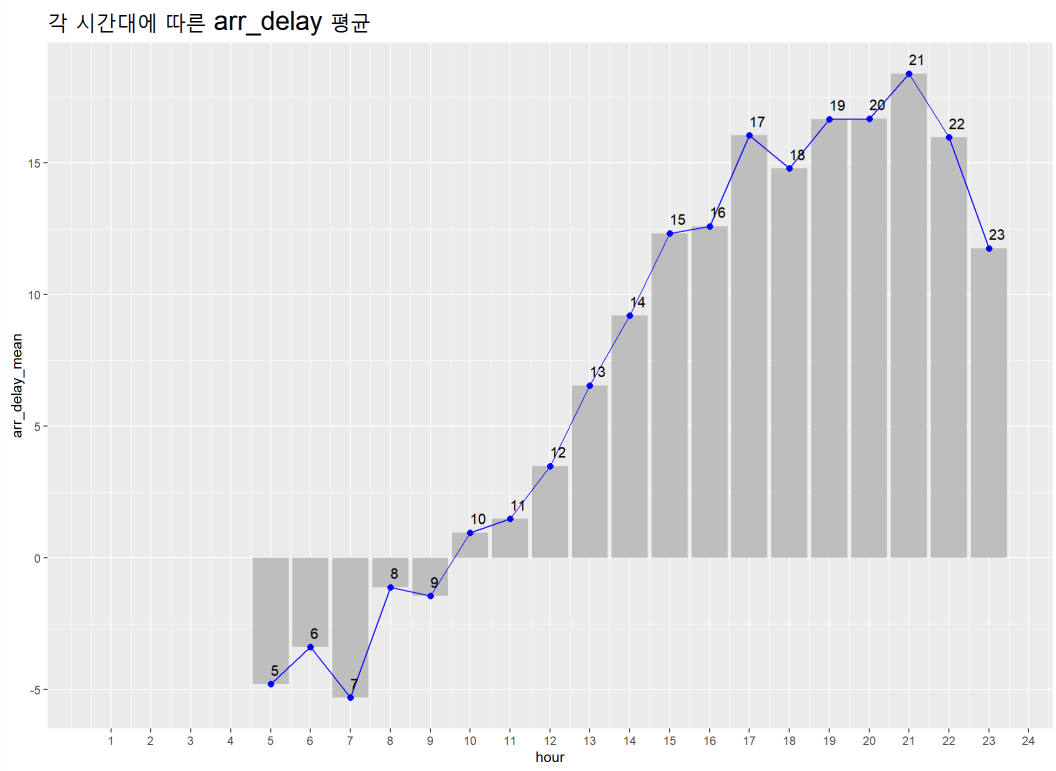
각 시간에 운행 비행기 수를 알아보았다. 운행 수가 많은 시간은 차례로 8AM, 6AM, 5PM,3PM가 있다. 운행 수가 적은 시간은 차례로 1AM, 11PM, 5AM등이 있다.

2.2 각 시간에 따른 arr_delay_mean

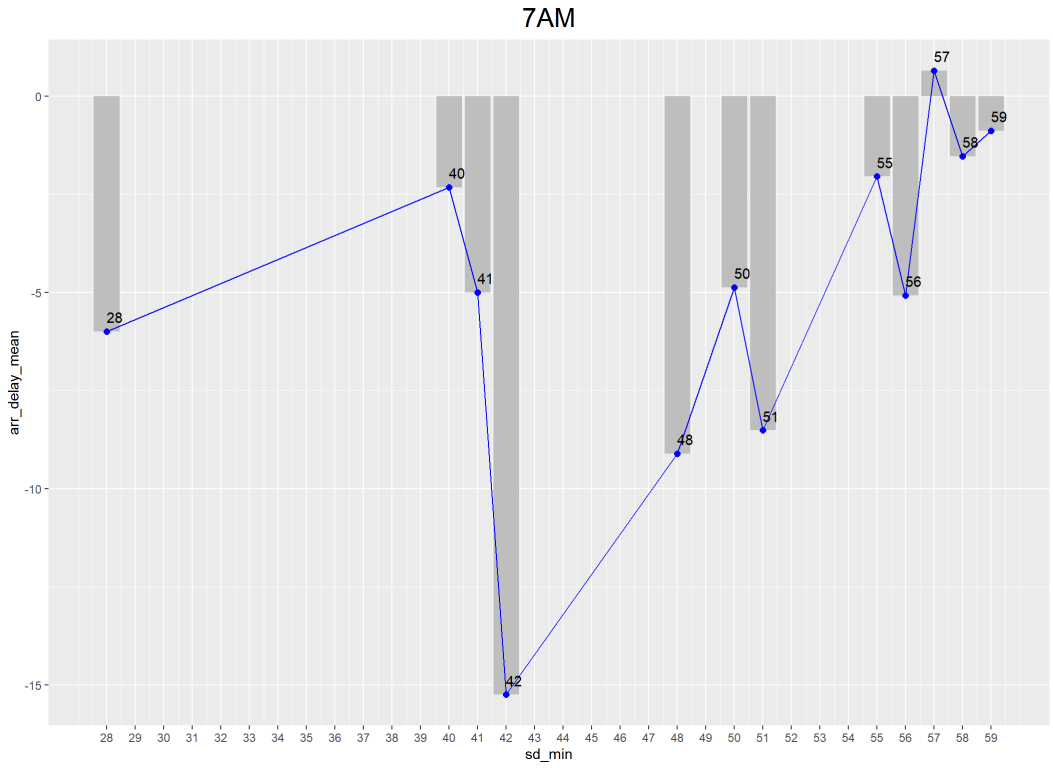
| hour | 7.000000 | 5.000000 | 6.000000 | 9.000000 | 8.000000 | 10.0000000 | 11.00000 | 12.00000 | 13.00000 | 14.00000 | 23.00000 | 15.00000 | 16.00000 | 18.00000 | 22.00000 | 17.00000 | 19.00000 | 20.0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| arr_delay | -5.304472 | -4.796907 | -3.384485 | -1.451407 | -1.113227 | 0.9539401 | 1.48193 | 3.48901 | 6.54474 | 9.19765 | 11.75528 | 12.32419 | 12.59764 | 14.78872 | 15.96716 | 16.04027 | 16.65587 | 16.6 |

각 시간에 따른 arr_delay_mean은 위와 같다. arr_delay_mean이 작은 시간대는 차례로 7AM, 5AM, 6AM, 9AM이 있고 arr_delay_mean이 큰 시간대는 차례로 21PM, 20PM, 19PM이 있다. Hint와 비슷한 양상을 보인다.

2.3 시간에 따른 arr_delay 평균 그림

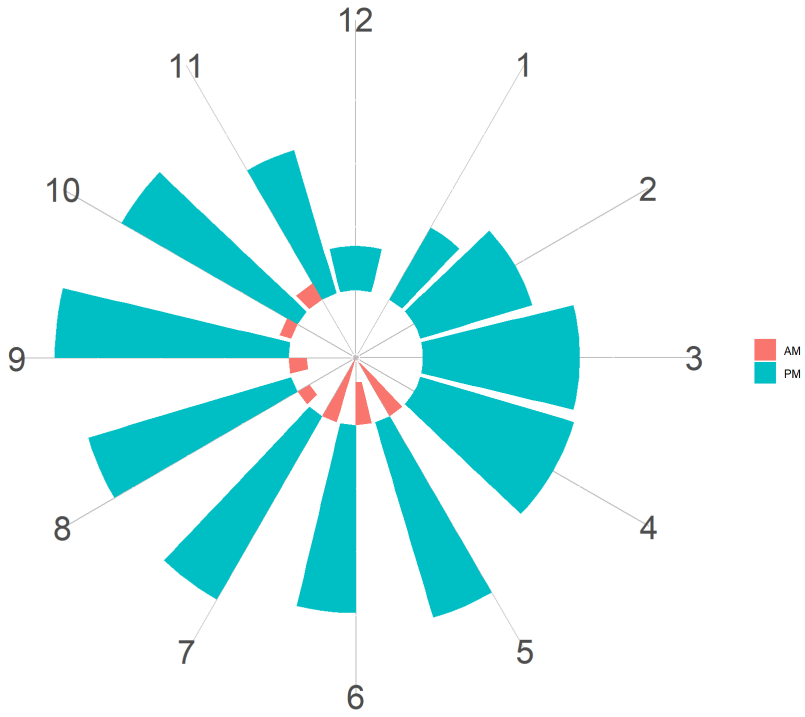



2.5 7시 분에 따른 arr_delay 평균




7시 중에서 구체적으로 arr_delay_mean이 가장 낮은 분(minute)을 확인했다. 그 결과 42분, 48분, 41분에서 가장 값이 작았다. 따라서 오전 *7시 40분 ~ 오전 7시 50분* 사이에서 비행기 지연을 최대한 피할 수 있다.

## Result



## 3-1. For each destination, compute the total minutes of delay

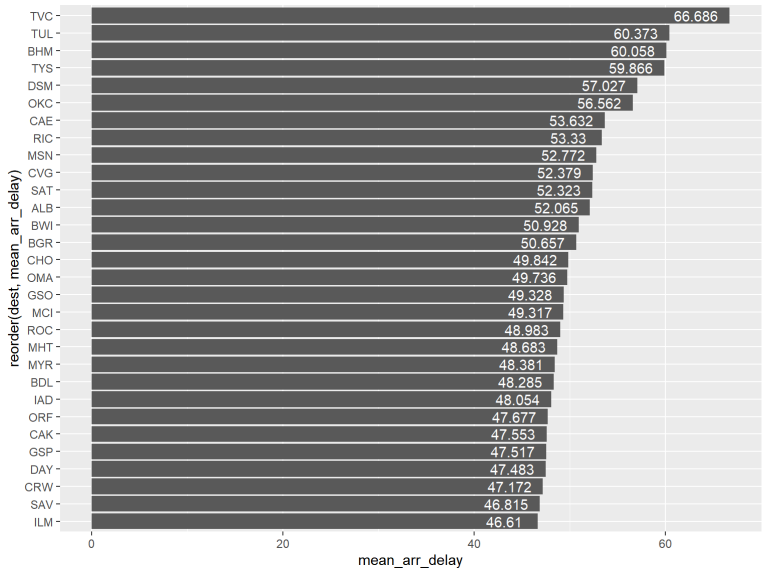
```
library(tidyverse)
df %>%
  group_by(dest) %>%
  filter(arr_delay > 0) %>%
  mutate(sum_arr_delay = sum(arr_delay,na.rm = TRUE),
           rate_arr_delay = arr_delay/sum_arr_delay) %>%
  select(flight,dest, sum_arr_delay, rate_arr_delay) %>%
  arrange(-rate_arr_delay) -> prob3

knitr::kable(head(prob3,15)) #rate_delay_mean에 따라 정렬
```

| flight dest | sum_arr_delay | rate_arr_delay |
|---:|---:|---:|
| 887 ANC | 62 | 0.6290323 |
| 385 MTJ | 170 | 0.5941176 |
| 55 PSP | 36 | 0.4722222 |
| 5383 SBN | 125 | 0.4240000 |
| 5383 SBN | 125 | 0.4000000 |
| 441 HDN | 119 | 0.3613445 |
| 568 BZN | 491 | 0.3136456 |
| 1506 JAC | 619 | 0.2827141 |
| 355 HDN | 119 | 0.2689076 |
| 5325 CHO | 947 | 0.2407603 |
| 1873 EYW | 194 | 0.2319588 |
| 55 PSP | 36 | 0.2222222 |
| 2453 EYW | 194 | 0.2164948 |
| 4412 MYR | 1016 | 0.2037402 |
| 486 MTJ | 170 | 0.1823529 |

rate_arr_delay에 따라 정렬하면 그 결과 위와 같다. *ANC로 가는 887 flight*의 rate_arr_delay값이 가장 크다.

| dest | sum_arr_delay | mean_arr_delay |
|---|---:|---:|
| TVC | 2334 | 66.68571 |
| TUL | 11652 | 60.37306 |
| BHM | 7267 | 60.05785 |
| TYS | 18379 | 59.86645 |
| DSM | 14827 | 57.02692 |
| OKC | 11369 | 56.56219 |
| CAE | 4666 | 53.63218 |
| RIC | 63783 | 53.33027 |
| MSN | 15040 | 52.77193 |
| CVG | 87997 | 52.37917 |
| SAT | 13447 | 52.32296 |
| ALB | 9580 | 52.06522 |
| BWI | 33918 | 50.92793 |
| BGR | 6940 | 50.65693 |
| CHO | 947 | 49.84211 |




mean_arr_delay에 따라 정렬하면 그 결과 위와 같다. *dest가 TVC일 경우* mean_arr_dealy가 가장 컸다.

## 3-2. For each flight, compute the proportion of the total delay for its destination

| flight dest | sum_arr_delay | arr_delay | rate_arr_delay |
|---:|---:|---:|---:|
| 887 ANC | 62 | 39 | 0.6290323 |
| 385 MTJ | 170 | 101 | 0.5941176 |

| flight dest | sum_arr_delay | arr_delay | rate_arr_delay |
|---|---|---|---|
| 55 PSP | 36 | 17 | 0.4722222 |
| 5383 SBN | 125 | 53 | 0.4240000 |
| 5383 SBN | 125 | 50 | 0.4000000 |
| 441 HDN | 119 | 43 | 0.3613445 |
| 568 BZN | 491 | 154 | 0.3136456 |
| 1506 JAC | 619 | 175 | 0.2827141 |
| 355 HDN | 119 | 32 | 0.2689076 |
| 5325 CHO | 947 | 228 | 0.2407603 |
| 1873 EYW | 194 | 45 | 0.2319588 |
| 55 PSP | 36 | 8 | 0.2222222 |
| 2453 EYW | 194 | 42 | 0.2164948 |
| 4412 MYR | 1016 | 207 | 0.2037402 |
| 486 MTJ | 170 | 31 | 0.1823529 |

결과는 위 rate_arr_delay로 정렬했을 때와 같다.

## 4. Delays are typically temporally correlated: even once the problem that caused the initial delay has been resolved, later flights are delayed to allow earlier flights to leave.

Draw a the scatter plot between mean of dep_delay of flights and the dep_delay of the immediately preceding flight.

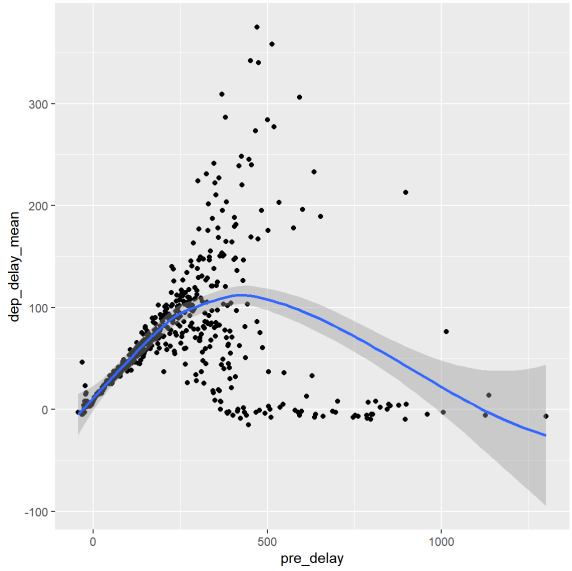
- Hint 1:Using lag(), explore how the delay of a flight is related to the delay of the immediately preceding flight. Use lag() to get the info of dep_delay of a previous flight in the same origin. Before using lag(),the data should be ordered by origin, month, day, dep_time.

- Hint 2:group by the dep_delay of the immediately preceding flight and find the mean of dep_delay of current flights.

| pre_delay | dep_delay_mean |
|---|---|
| -43 | -3.000000 |
| -33 | -5.000000 |
| -32 | 46.000000 |
| -30 | -5.000000 |
| -27 | 4.000000 |
| -26 | 1.000000 |
| -25 | 8.000000 |
| -24 | -3.000000 |
| -23 | 23.333333 |
| -22 | 14.545454 |
| -21 | 7.750000 |
| -20 | 16.250000 |
| -19 | 2.611111 |
| -18 | 6.912500 |
| -17 | 7.906542 |

lag()를 사용해 previous dep_delay를 의미하는 pre_delay를 계산해주었다. pre_delay를 기준으로 dep_delay의 평균을 계산한 결과 표는 위와 같다.

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




pre_delay에 따른 dep_delay_mean의 산점도를 그린 결과 *pre_delay > 250* 인 경우 두 변수가 양의 상관관계를 가진다. smooth line을 보면 *pre_delay > 480* 인 경우 양의 상관을 가지고 *pre_delay <= 480* 인 경우 음의 상관을 가지는 것처럼 보인다.

## 5. Look at each destination. Can you find flights that are suspiciously fast? (i.e. flights that represent a potential data entry error). Compute the air time of a flight relative to the shortest flight to that destination. Which flights were most delayed in the air?

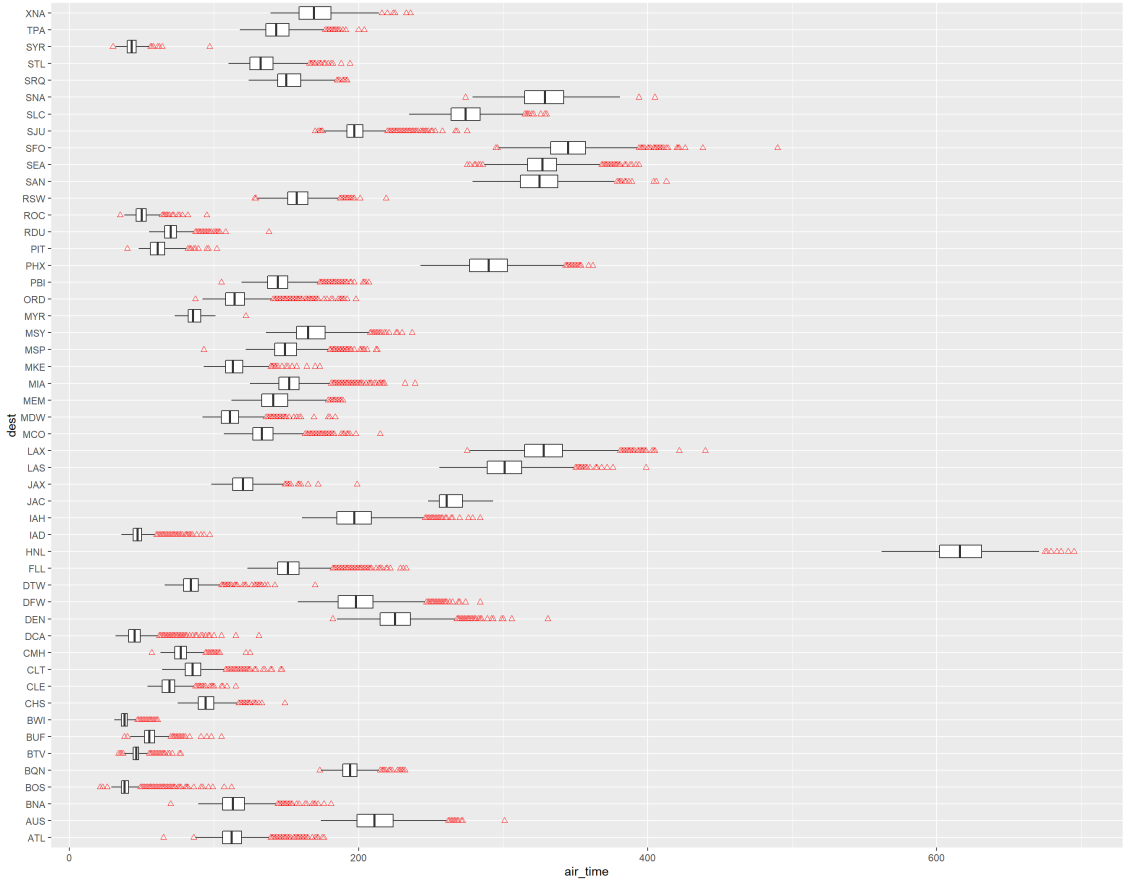
### Boxplot

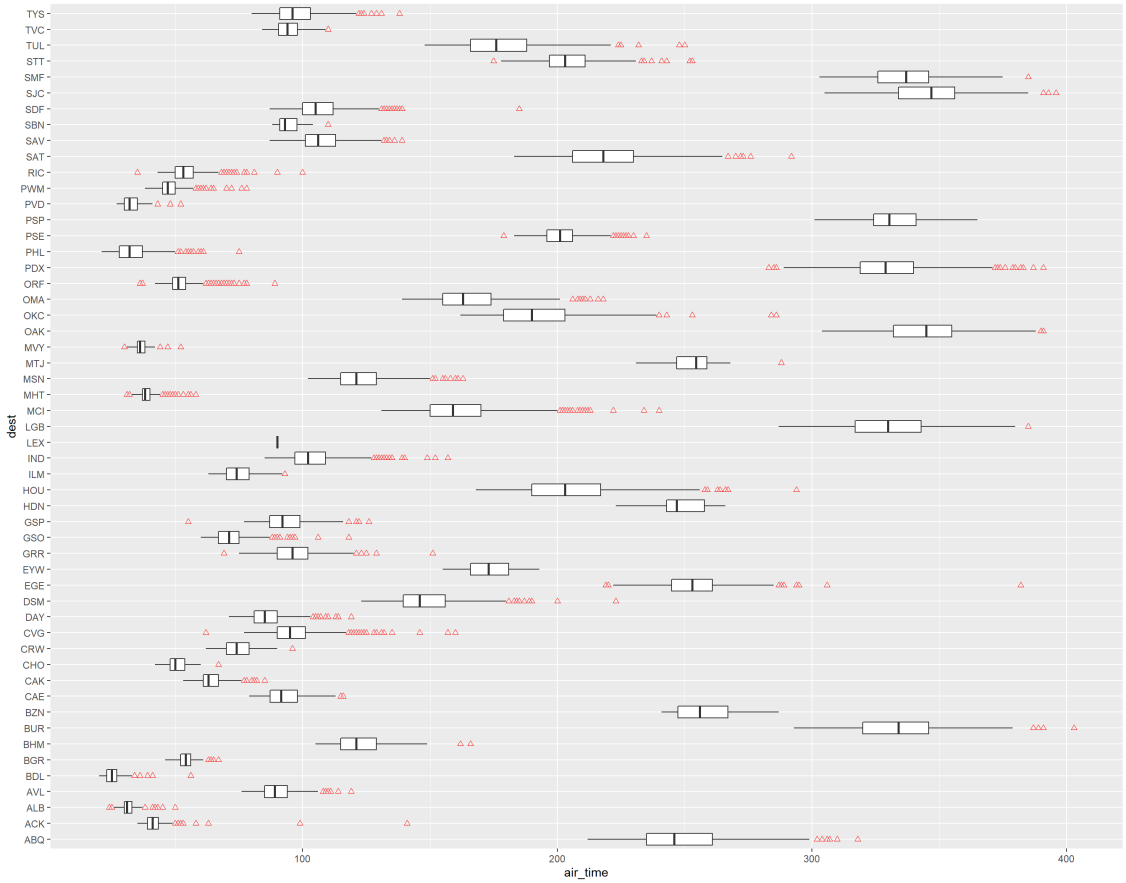
Boxplot을 이용해 이상치를 찾아보았다.

```
uni_dest<- df$dest %>% unique

df %>% # dest : 1 - 50
  select(dest,air_time)  %>%
  group_by(dest) %>%
  arrange(dest) %>%
  filter(dest %in% uni_dest[1:50]) %>%
  ggplot(aes(x=dest,y=air_time))+
  geom_boxplot(outlier.color = 'red',outlier.shape = 2)+
  coord_flip() -> plot5_1

df %>% # dest : 51 - 103 (total : 103)
  select(dest,air_time)  %>%
  group_by(dest) %>%
  arrange(dest) %>%
  filter(dest %in% uni_dest[51:103]) %>%
  ggplot(aes(x=dest,y=air_time))+
  geom_boxplot(outlier.color = 'red',outlier.shape = 2)+
  coord_flip() -> plot5_2
plot5_1; plot5_2
```






각 dest의 air_time에 대한 Boxplot을 그려보면 결과는 위와 같다. 빨간색 세모로 표시된 점이 IQR에 기반을 둔 이상치이다. 우리가 관심있는 건 *suspiciously fast*?이므로 Q1 - 1.5 * (Q3 - Q1)를 의미하는 아래 울타리 이하의 값을 살펴보자.

| dest | air_time | carrier | flight | origin | upper | lower | IQR | lowerOutlier | is_lowerOutlier |
|---|---|---|---|---|---|---|---|---|---|
| ALB | 25 | EV | 3811 | EWR | 33 | 30 | 3 | 25.5 | TRUE |
| ALB | 24 | EV | 4162 | EWR | 33 | 30 | 3 | 25.5 | TRUE |
| BOS | 26 | US | 2136 | LGA | 41 | 36 | 5 | 28.5 | TRUE |
| BOS | 23 | US | 2142 | LGA | 41 | 36 | 5 | 28.5 | TRUE |
| BOS | 21 | US | 2132 | LGA | 41 | 36 | 5 | 28.5 | TRUE |
| BOS | 26 | DL | 1597 | JFK | 41 | 36 | 5 | 28.5 | TRUE |
| MVY | 30 | B6 | 1438 | JFK | 38 | 35 | 3 | 30.5 | TRUE |
| MVY | 30 | B6 | 1338 | JFK | 38 | 35 | 3 | 30.5 | TRUE |
| SYR | 30 | EV | 4249 | EWR | 46 | 40 | 6 | 31.0 | TRUE |
| MHT | 32 | EV | 3842 | EWR | 40 | 37 | 3 | 32.5 | TRUE |
| MHT | 32 | EV | 3813 | EWR | 40 | 37 | 3 | 32.5 | TRUE |
| MHT | 32 | EV | 5223 | LGA | 40 | 37 | 3 | 32.5 | TRUE |
| MHT | 32 | 9E | 3965 | LGA | 40 | 37 | 3 | 32.5 | TRUE |
| MHT | 32 | EV | 4969 | LGA | 40 | 37 | 3 | 32.5 | TRUE |
| MHT | 31 | EV | 5258 | LGA | 40 | 37 | 3 | 32.5 | TRUE |

우선 각 dest에 따라 Q2, Q4, IQR을 계산해주고 이를 이용해 lowerOutlier를 계산한다. lowerOutlier값보다 더 작은지 확인하는 is_lowerOutlier라는 변수를 추가로 만들어 TRUE인 값만 추출해서 확인한다. 그 결과 100개의 비행기 운행 정보가 출력된다.

## 6. Find all destinations that are flown by at least two carriers. Use that information to rank the carriers.

| carrier | EV | 9E | UA | DL | B6 | AA | MQ | WN | OO | US | VX | YV | FL | AS | F9 | HA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| num_dest | 51 | 48 | 42 | 39 | 35 | 19 | 19 | 10 | 5 | 5 | 4 | 3 | 2 | 1 | 1 | 1 |

carrier의 순위는 위와 같다. EV라는 carrier가 두 개 이상의 항공사가 운항하는 공항만을 고려해 가장 많은 목적지로 비행한다.