課題解説

HRY

2019年5月22日

では、[Rではじめるデータサイエンス](https://www.amazon.co.jp/R%E3%81%A7%E3%81%AF%E3%81%98%E3%82%81%E3%82%8B%E3%83%87%E3%83%BC%E3%82%BF%E3%82%B5%E3%82%A4%E3%82%A8%E3%83%B3%E3%82%B9-Hadley-Wickham/dp/487311814X)における指定問題2.と3.(p.67)の解説をします。 まず、準備段階として[GitHubのIssue](https://github.com/Ylab-Shiny/lesson_190524/issues/1)述べた install.packages(c("tidyverse", "nycflights13"))をRconsoleで実行し、外部パッケージをインストール してください。

# データ(flights)の確認

とりあえず、データセットの確認を行ってみましょう。

# パッケージの読み込み  
library(tidyverse)

## -- Attaching packages ---------------------------------------------------------------------------------------------------------- tidyverse 1.2.1 --

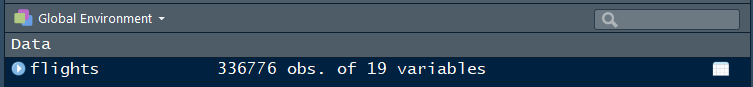
## √ ggplot2 3.0.0 √ purrr 0.2.5  
## √ tibble 1.4.2 √ dplyr 0.7.7  
## √ tidyr 0.8.1 √ stringr 1.3.1  
## √ readr 1.1.1 √ forcats 0.3.0

## -- Conflicts ------------------------------------------------------------------------------------------------------------- tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

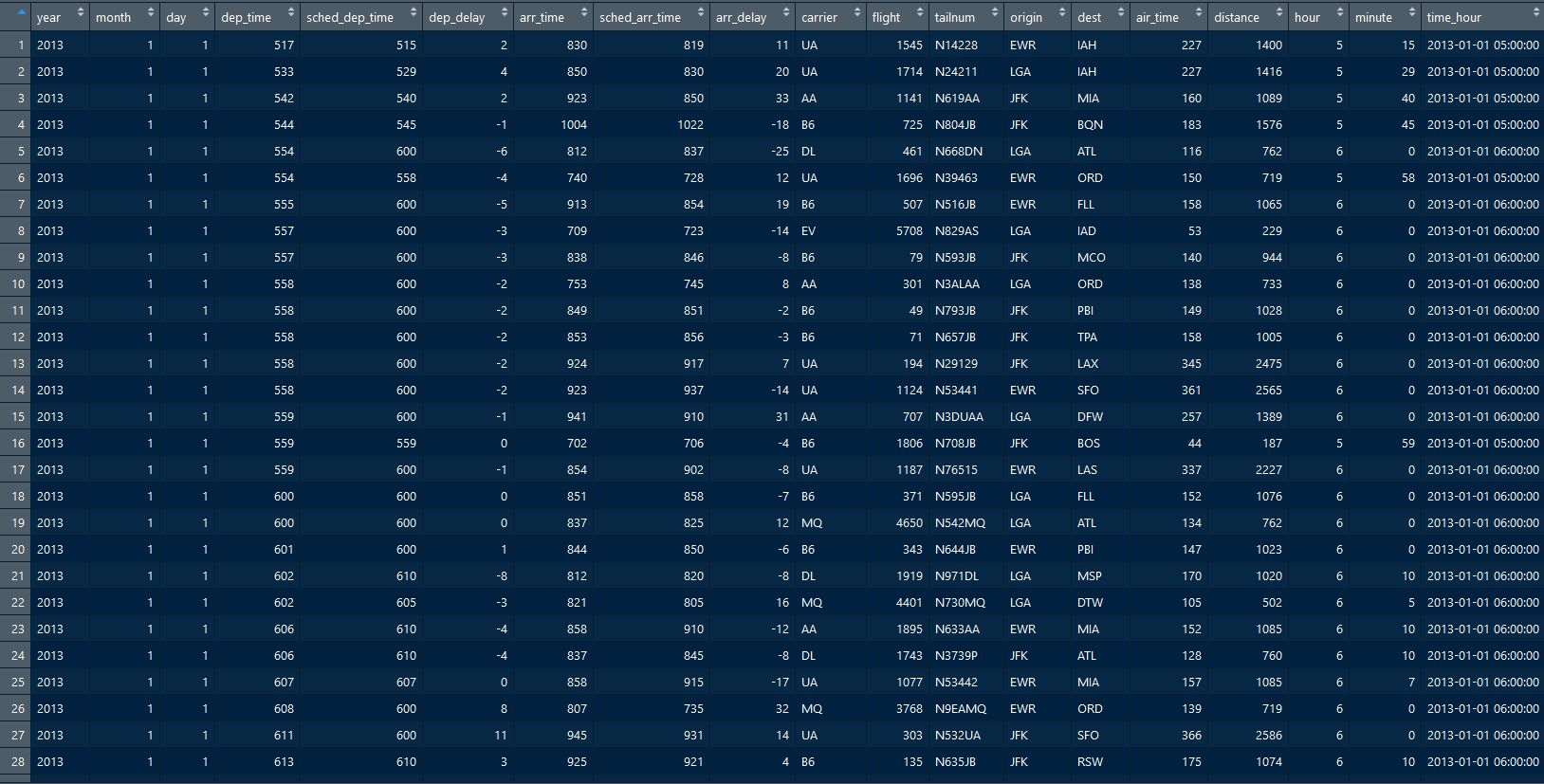
library(nycflights13)  
  
# RstudioのGlobal Environmentにflightsオブジェクトを作成  
flights <- flights  
# 上から5行までのデータの表示  
head(flights)

## # A tibble: 6 x 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int>  
## 1 2013 1 1 517 515 2 830  
## 2 2013 1 1 533 529 4 850  
## 3 2013 1 1 542 540 2 923  
## 4 2013 1 1 544 545 -1 1004  
## 5 2013 1 1 554 600 -6 812  
## 6 2013 1 1 554 558 -4 740  
## # ... with 12 more variables: sched\_arr\_time <int>, arr\_delay <dbl>,  
## # carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,  
## # air\_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,  
## # time\_hour <dttm>

flights <- flightsというコードをRstudioのConsoleで実行することによって、RstudioのGlobal Environmentにデータの行数(obs.)と列数(variables)が表示されます。



また、View(flights)を同様に実行することでRstudio上で文字通りデータを下図のように眺めることができます。



データの系列（変数）の確認については、?flightsを実行することで確認できます。



* year,month,day：文字通りの意味で出発日に関する系列(Date of departure)
* dep\_time,arr\_time：実際の出発時刻と到着時刻の系列(Actual departure and arrival times (format HHMM or HMM), local tz.)
* sched\_dep\_time,sched\_arr\_time：予定出発時刻と予定到着時刻の系列(Scheduled departure and arrival times (format HHMM or HMM), local tz.)
* dep\_delay,arr\_delay：出発と到着の分単位の遅れの系列。負値は早い出発及び到着をあらわす(Departure and arrival delays, in minutes. Negative times represent early departures/arrivals.)
* hour,minute：出発予定時刻を時間と分に分割した系列(Time of scheduled departure broken into hour and minutes.)
* carrier：2文字の業者の略語の系列(Two letter carrier abbreviation. See airlines() to get name)
* tailnum：飛行機の末尾番号の系列(Plane tail number)
* flight：フライト番号の系列(Flight number)
* origin,dest：出発地と目的地の系列(Origin and destination. See airports() for additional metadata.)
* air\_time：飛行中に費やした時間（単位：分）系列(Amount of time spent in the air, in minutes)
* distance：空港間の距離（単位：マイル）系列(Distance between airports, in miles)
* time\_hour：POSIXct型のフライト予定日時系列。出発地とともに、フライトデータを気象データに結合するために使用される(Scheduled date and hour of the flight as a POSIXct date. Along with origin, can be used to join flights data to weather data.)

# 2.どの飛行機(tailnum)が定時離着陸記録に関して最悪か。

それでは、本題に移りましょう。方針としては、飛行機ごとに遅れまたは予定の早まりを集計すればよいということです。定刻きっかりに進行することが望まれると考えられるため、数値は絶対値として扱うこととします。

# tailnumのデータ数  
data\_tailnum <- flights %>%   
 filter(!is.na(tailnum)) %>%   
 group\_by(tailnum) %>%   
 summarise(N = n()) %>%   
 arrange(desc(N))  
  
# 飛行機の最大・最小数  
max(data\_tailnum$N)

## [1] 575

min(data\_tailnum$N)

## [1] 1

# 遅れに関するデータオブジェクトdata\_delay  
data\_delay <- flights %>%   
 filter(!is.na(dep\_delay), !is.na(arr\_delay)) %>%   
 group\_by(tailnum) %>%   
 mutate(delay\_score = abs(dep\_delay) + abs(arr\_delay)) %>%   
 select(tailnum, delay\_score) %>%   
 summarise(score = sum(delay\_score), N = n()) %>%   
 mutate(mean\_score = score / N) %>%   
 arrange(desc(mean\_score))  
  
# deta\_delayの確認  
head(data\_delay)

## # A tibble: 6 x 4  
## tailnum score N mean\_score  
## <chr> <dbl> <int> <dbl>  
## 1 N844MH 617 1 617  
## 2 N911DA 562 1 562  
## 3 N922EV 550 1 550  
## 4 N587NW 536 1 536  
## 5 N851NW 452 1 452  
## 6 N654UA 412 1 412

tail(data\_delay)

## # A tibble: 6 x 4  
## tailnum score N mean\_score  
## <chr> <dbl> <int> <dbl>  
## 1 N902DA 4 1 4   
## 2 N693SW 7 2 3.5  
## 3 N27901 3 1 3   
## 4 N626AW 3 1 3   
## 5 N456UW 2 1 2   
## 6 N7BVAA 1 1 1

では、コードについて解説します。data\_tailnumでは飛行機の便数に関するデータオブジェクトを作成しています。2行目のfilter関数(p.39)でtailnumが欠損値である行を行まるごと除き（リストワイズ）、tailnumをgroup\_by関数(p.63)によってグルーピングし、summarize関数(p.55)で飛行機名毎の総便数Nを求めています。Nの最大と最小を確認すると、最大は575便、最小は1便の運航となっており、飛行機によってNに大きな差があることが分かります。

この結果を受け、1便当たりの遅れ時間mean\_scoreを求め、その値が最も大きいものを最悪な飛行機と呼ぶことにしますw

data\_delayデータオブジェクトにおいて、mean\_scoreを求めることとします。まず、filter(!is.na(dep\_delay), !is.na(arr\_delay))でキャンセル便をリストワイズします。そして、続く行のmutate関数(p.48)で定刻との差の絶対値を足し合わせることでdelay\_scoreを求め、5行目でselect関数(p.45)を用いて、必要とする系列(tailnum,delay\_score)を選択しています。そして、6行目でtailnum毎の定刻との差の総和と総便数を求め、mutateでmean\_score系列を追加し、最後にarrange(desc)を用いることで、mean\_scoreを基準とする降順に並べ替えています。

結果的に、data\_delayの先端部を確認することで、*N844MH*という飛行機が最悪であると分かります。1便だけですが、約10時間分の定刻との差はいただけませんね。逆に最高？な飛行機は、こちらも1便だけですが*N7BVAA*という結果となりました。

それと、定刻との差が0である飛行機名は存在しませんでした。飛行機運航は定刻通りに進むことはとても難しいようです。私も初めての飛行機旅行として、中部国際-新千歳行きの便を利用したとき、出発が2時間ほど遅れるというアクシデント遭遇した経験があります。

# 3. 遅延をできるだけ避けたいとすれば、どの時間に飛行するとよいか。

次の問題に移ります。まず、私の考えでは問題文で「遅延を避けたい」とされているので、遅延スコア値（dep\_delayとarr\_delay）の和delay\_valueが負であるものはリストワイズしてしまえばいいと思います。なぜ和による遅延スコア値で判断するべきかというと、例えば、出発は遅れたが到着は早まって結果的に和が0以下となる（先の遅れを取り戻した）ケース（逆もまた然り）が想定されるためです。

そして、先程の問題と同様に出発予定時間hourでグルーピングし、集計するという方針とします。

# dep\_delayまたはarr\_delayが負のものをリストワイズ  
data\_avoid\_delay <- flights %>%   
 filter(!is.na(dep\_delay), !is.na(arr\_delay)) %>%   
 mutate(delay\_value = dep\_delay + arr\_delay) %>%   
 filter(delay\_value >= 0) %>%   
 group\_by(hour) %>%   
 select(hour, delay\_value) %>%   
 summarise(sum\_delay = sum(delay\_value)) %>%   
 arrange(sum\_delay)  
  
# 最適な時間  
BestWorst\_hours <- c(data\_avoid\_delay$hour[1], data\_avoid\_delay$hour[nrow(data\_avoid\_delay)])  
print(paste0("遅延をできるだけ避けるには、", BestWorst\_hours[1], "時発の便を選択するとよいでしょう"))

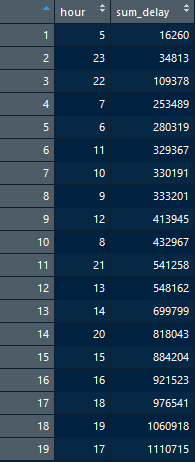
## [1] "遅延をできるだけ避けるには、5時発の便を選択するとよいでしょう"

print(paste0("一方、遅延を最も避けにくいのは、", BestWorst\_hours[2], "時発の便です"))

## [1] "一方、遅延を最も避けにくいのは、17時発の便です"

コードについては、先の問題と比べて真新しいものは存在しないので割愛させていただきます。

結果として、5時発の便を選ぶのがよいという結果となりました。各出発時刻ごとの結果を下に示します。パッと見た感じ、午前中に出発する便の方が遅延が少なそうに思われます。



もし内容について御質問やご指摘がありましたら、こちらの[GitHubのIssue]((https://github.com/Ylab-Shiny/lesson_190524/issues/1))で対応しようと思います。