빅데이터를이용한통계그래픽스 HW #5

1678226 유찬미 (Chanmi Yoo)

2019년 11월 15일

###### \* tidyverse 에서 제공하는 table3 자료를 이용

# Load Packages  
library(tidyverse)

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

## √ ggplot2 3.2.1 √ purrr 0.3.3  
## √ tibble 2.1.3 √ dplyr 0.8.3  
## √ tidyr 1.0.0 √ stringr 1.4.0  
## √ readr 1.3.1 √ forcats 0.4.0

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

library(dplyr)  
  
# Load Data  
data("table3")

###### 1. 각 연도별, 나라별, TB 수를 뽑아내시오.

tb1 <- table3 %>%   
 separate(rate, into=c('cases', 'population'), sep='/', convert = T) %>%   
 select(year, country, cases)  
tb1

## # A tibble: 6 x 3  
## year country cases  
## <int> <chr> <int>  
## 1 1999 Afghanistan 745  
## 2 2000 Afghanistan 2666  
## 3 1999 Brazil 37737  
## 4 2000 Brazil 80488  
## 5 1999 China 212258  
## 6 2000 China 213766

###### 2. 각 연도별, 나라별로 population 수를 뽑아내시오.

tb2 <- table3 %>%   
 separate(rate, into = c('cases', 'population'), sep = '/', convert = T) %>%   
 select(year, country, population)  
tb2

## # A tibble: 6 x 3  
## year country population  
## <int> <chr> <int>  
## 1 1999 Afghanistan 19987071  
## 2 2000 Afghanistan 20595360  
## 3 1999 Brazil 172006362  
## 4 2000 Brazil 174504898  
## 5 1999 China 1272915272  
## 6 2000 China 1280428583

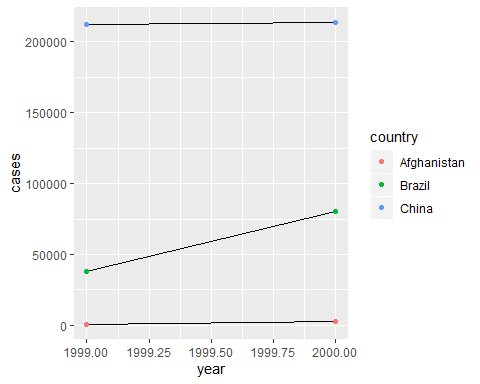
###### 3. 1,2의 결과를 이용하여 비율을 계산하시오.

tb3 <- tb1 %>%   
 left\_join(tb2, by = c('year', 'country')) %>%   
 mutate(rate = cases/population)  
tb3

## # A tibble: 6 x 5  
## year country cases population rate  
## <int> <chr> <int> <int> <dbl>  
## 1 1999 Afghanistan 745 19987071 0.0000373  
## 2 2000 Afghanistan 2666 20595360 0.000129   
## 3 1999 Brazil 37737 172006362 0.000219   
## 4 2000 Brazil 80488 174504898 0.000461   
## 5 1999 China 212258 1272915272 0.000167   
## 6 2000 China 213766 1280428583 0.000167

###### 4. 시간에 따른 TB 수의 변화를 나라별로 나타내는 그림을 그리시오.

ggplot2::ggplot(tb3, aes(year, cases)) +  
 geom\_line(aes(group=country)) +  
 geom\_point(aes(color=country))



###### 5. gather()과 spread()는 대칭적인 함수인가? 아래의 예를 실행시켜보고 이를 이용하여 설명하시오.

stocks <- tibble(   
 year = c(2015, 2015, 2016, 2016),   
 half = c( 1, 2, 1, 2),   
 return = c(1.88, 0.59, 0.92, 0.17)   
 )  
stocks

## # A tibble: 4 x 3  
## year half return  
## <dbl> <dbl> <dbl>  
## 1 2015 1 1.88  
## 2 2015 2 0.59  
## 3 2016 1 0.92  
## 4 2016 2 0.17

stocks %>%   
 spread(year, return) %>%   
 gather("year", "return", `2015`:`2016`)

## # A tibble: 4 x 3  
## half year return  
## <dbl> <chr> <dbl>  
## 1 1 2015 1.88  
## 2 2 2015 0.59  
## 3 1 2016 0.92  
## 4 2 2016 0.17

[설명] spread 함수를 적용한 뒤 gather 함수를 적용했더니, 변수들이 그대로 유지되었다. 다만, 변수명으로 쓰인 year가 double에서 character로 바뀌었으며, 열의 배열 순서가 바뀌었다.

##### \* nycflights13 패키지에 있는 flights 자료를 이용하여 각 상황에 맞는 비행을 모두 찾으시오.

library(nycflights13)  
data(flights)

##### 6. dep\_time과 sched\_dep\_time은 특수한 형태로 저장된 자료이다. (519는 5시 19분을 나타냄). 이 두 변수의 시간을 자정을 기준으로 몇 분 후인지를 나타내는 dep\_time\_min과 sched\_dep\_time\_min을 만드시오.

flights %>%   
 mutate(dep\_time, dep\_time\_min = (dep\_time%/%100)\*60 + (dep\_time%%100)) %>%   
 mutate(sched\_dep\_time, sched\_dep\_time\_min = (sched\_dep\_time%/%100)\*60 + (sched\_dep\_time%%100))

## # A tibble: 336,776 x 21  
## 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  
## 7 2013 1 1 555 600 -5 913  
## 8 2013 1 1 557 600 -3 709  
## 9 2013 1 1 557 600 -3 838  
## 10 2013 1 1 558 600 -2 753  
## # ... with 336,766 more rows, and 14 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>, dep\_time\_min <dbl>,  
## # sched\_dep\_time\_min <dbl>

##### 7. 가장 delay 건수가 많은 도착공항은? (여기서 delay는 dep\_delay, arr\_delay 모두를 고려) 가장 departure delay 가 많은 공항은? 가장 arrival delay 가 많은 공항은?

not\_cancelled <- flights %>%   
 filter(!is.na(dep\_delay), !is.na(arr\_delay))  
  
# 총 delay 건수   
not\_cancelled %>%   
 group\_by(dest) %>%   
 summarise(n\_delay = sum(dep\_delay >0) + sum(arr\_delay >0)) %>%  
 arrange(desc(n\_delay))

## # A tibble: 104 x 2  
## dest n\_delay  
## <chr> <int>  
## 1 ATL 13989  
## 2 ORD 12499  
## 3 LAX 12422  
## 4 MCO 11005  
## 5 SFO 10601  
## 6 FLL 10250  
## 7 CLT 10108  
## 8 BOS 9552  
## 9 MIA 7940  
## 10 DCA 6963  
## # ... with 94 more rows

# departure delay 건수   
not\_cancelled %>%   
 group\_by(dest) %>%   
 summarise(n\_dep\_delay = sum(dep\_delay > 0)) %>%   
 arrange(desc(n\_dep\_delay))

## # A tibble: 104 x 2  
## dest n\_dep\_delay  
## <chr> <int>  
## 1 LAX 6455  
## 2 ORD 6301  
## 3 ATL 6043  
## 4 SFO 5660  
## 5 MCO 5460  
## 6 FLL 5038  
## 7 BOS 4809  
## 8 CLT 4270  
## 9 MIA 4085  
## 10 DEN 3603  
## # ... with 94 more rows

# arrival delay 건수   
not\_cancelled %>%   
 group\_by(dest) %>%   
 summarise(n\_arr\_delay = sum(arr\_delay > 0)) %>%   
 arrange(desc(n\_arr\_delay))

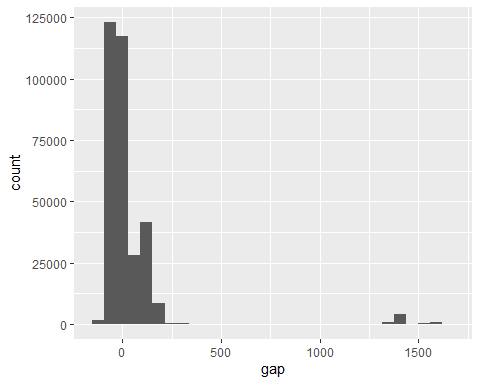
## # A tibble: 104 x 2  
## dest n\_arr\_delay  
## <chr> <int>  
## 1 ATL 7946  
## 2 ORD 6198  
## 3 LAX 5967  
## 4 CLT 5838  
## 5 MCO 5545  
## 6 FLL 5212  
## 7 SFO 4941  
## 8 BOS 4743  
## 9 DCA 4003  
## 10 MIA 3855  
## # ... with 94 more rows

[답] 전체적으로 delay가 가장 많이 일어나는 공항은 ATL이다. 가장 departure delay가 많은 공항은 LAX이고, 가장 arrival delay가 많은 공항은 ATL이다.

##### 8. 자료가 잘못 입력되었다고 의심될 만큼 빠르거나 느린 비행이 있는지 살펴보려고 한다. 이를 위해 필요한 변수를 만들고 살펴보시오.

flights%>%   
 mutate(sched\_time = (sched\_arr\_time%/%100\*60 + sched\_arr\_time%%100) - (sched\_dep\_time%/%100\*60 + sched\_dep\_time%%100)) %>% # sched\_time: 예정된 비행시간(분 단위)  
 mutate(gap = air\_time - sched\_time) %>% # gap: 예정된 비행시간과 실제 비행시간의 차이   
 ggplot2::ggplot(aes(gap)) +  
 geom\_histogram(na.rm = T)

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



[해석] 위 히스토그램은 예정 비행시간과 실제 비행시간의 차이를 나타낸 것으로, 대부분 0 근처의 값을 가진다. 즉, 예정 비행시간대로 이동한 경우가 대부분이다. 그런데 일부 자료가 1500 정도 값을 가진다. 즉, 실제 비행시간과 예정 비행시간이 약 1500분이라는 극단적 차이를 보인다. 이는 일어날 가능성이 굉장히 희박하며, 자료가 잘못 입력되었을 가능성이 높다.