lec1101

박효선 1585063

2018년 12월 3일

#############  
# 11 월 1 일  
#############  
  
# 기준 조건 이상의 그룹 찾기  
popular\_dest <- flights %>%   
 group\_by(dest) %>% # 목적지별로   
 filter(n() > 500) # 운행건수가 500 이상인 목적지

## Warning: package 'bindrcpp' was built under R version 3.5.1

popular\_dest

## # A tibble: 330,573 x 19  
## # Groups: dest [72]  
## 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 330,563 more rows, and 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>

# 각 그룹마다 새로운 변수 생성하기  
popular\_dest %>% # 목적지별로 (시간의 비율)  
 filter(arr\_delay > 0) %>% # 진짜 지연된 운행만 뽑아서  
 mutate(prop\_delay = arr\_delay / sum(arr\_delay)) %>% # 해당운행 지연된시간/목적지별총지연된시간합  
 select(year:day, dest, arr\_delay, prop\_delay)

## # A tibble: 130,294 x 6  
## # Groups: dest [72]  
## year month day dest arr\_delay prop\_delay  
## <int> <int> <int> <chr> <dbl> <dbl>  
## 1 2013 1 1 IAH 11 0.000111   
## 2 2013 1 1 IAH 20 0.000201   
## 3 2013 1 1 MIA 33 0.000235   
## 4 2013 1 1 ORD 12 0.0000424  
## 5 2013 1 1 FLL 19 0.0000938  
## 6 2013 1 1 ORD 8 0.0000283  
## 7 2013 1 1 LAX 7 0.0000344  
## 8 2013 1 1 DFW 31 0.000282   
## 9 2013 1 1 ATL 12 0.0000400  
## 10 2013 1 1 DTW 16 0.000116   
## # ... with 130,284 more rows

popular\_dest %>% # mean(조건문) :비율 # 목적지별 (지연여부의 비율)  
 mutate(prop\_delay = mean(arr\_delay>0, na.rm = T)) %>% # 실제 지연여부 비율  
 select(year:day, dest, arr\_delay, prop\_delay)

## # A tibble: 330,573 x 6  
## # Groups: dest [72]  
## year month day dest arr\_delay prop\_delay  
## <int> <int> <int> <chr> <dbl> <dbl>  
## 1 2013 1 1 IAH 11 0.407  
## 2 2013 1 1 IAH 20 0.407  
## 3 2013 1 1 MIA 33 0.333  
## 4 2013 1 1 BQN -18 0.467  
## 5 2013 1 1 ATL -25 0.472  
## 6 2013 1 1 ORD 12 0.374  
## 7 2013 1 1 FLL 19 0.438  
## 8 2013 1 1 IAD -14 0.441  
## 9 2013 1 1 MCO -8 0.397  
## 10 2013 1 1 ORD 8 0.374  
## # ... with 330,563 more rows

popular\_dest %>% # mean(data[조건문]) :평균 # 목적지별 (지연된 시간의 평균)  
 mutate(prop\_delay = mean(arr\_delay[arr\_delay>0], na.rm = T)) %>% #실제로 딜레이된 시간의 평균  
 select(year:day, dest, arr\_delay, prop\_delay)

## # A tibble: 330,573 x 6  
## # Groups: dest [72]  
## year month day dest arr\_delay prop\_delay  
## <int> <int> <int> <chr> <dbl> <dbl>  
## 1 2013 1 1 IAH 11 34.5  
## 2 2013 1 1 IAH 20 34.5  
## 3 2013 1 1 MIA 33 36.4  
## 4 2013 1 1 BQN -18 31.0  
## 5 2013 1 1 ATL -25 37.8  
## 6 2013 1 1 ORD 12 45.7  
## 7 2013 1 1 FLL 19 38.9  
## 8 2013 1 1 IAD -14 48.1  
## 9 2013 1 1 MCO -8 37.2  
## 10 2013 1 1 ORD 8 45.7  
## # ... with 330,563 more rows

##########################################################  
# 각 그룹에서 최하위 찾기   
# 일별로 도착연착이 큰 순서대로 top10  
flights %>% group\_by(year, month, day) %>% # 날짜별로  
 filter(rank(desc(arr\_delay)) <= 10 ) # 도착지연시간이 큰것부터 순서대로 top 10 비행

## # A tibble: 3,609 x 19  
## # Groups: year, month, day [365]  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int>  
## 1 2013 1 1 848 1835 853 1001  
## 2 2013 1 1 1815 1325 290 2120  
## 3 2013 1 1 1842 1422 260 1958  
## 4 2013 1 1 1938 1703 155 2109  
## 5 2013 1 1 1942 1705 157 2124  
## 6 2013 1 1 2006 1630 216 2230  
## 7 2013 1 1 2115 1700 255 2330  
## 8 2013 1 1 2205 1720 285 46  
## 9 2013 1 1 2312 2000 192 21  
## 10 2013 1 1 2343 1724 379 314  
## # ... with 3,599 more rows, and 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 %>% group\_by(year, month, day) %>% # 날짜별로   
 filter(rank(desc(arr\_delay)) <= 5) %>% # 도착지연시간이 큰 순서대로 top5  
 print(n = 15)

## # A tibble: 1,805 x 19  
## # Groups: year, month, day [365]  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int>  
## 1 2013 1 1 848 1835 853 1001  
## 2 2013 1 1 1815 1325 290 2120  
## 3 2013 1 1 1842 1422 260 1958  
## 4 2013 1 1 2115 1700 255 2330  
## 5 2013 1 1 2343 1724 379 314  
## 6 2013 1 2 1332 904 268 1616  
## 7 2013 1 2 1412 838 334 1710  
## 8 2013 1 2 1607 1030 337 2003  
## 9 2013 1 2 2131 1512 379 2340  
## 10 2013 1 2 2225 1930 175 231  
## 11 2013 1 3 1624 1330 174 1900  
## 12 2013 1 3 1834 1540 174 2040  
## 13 2013 1 3 2008 1540 268 2339  
## 14 2013 1 3 2012 1600 252 2314  
## 15 2013 1 3 2056 1605 291 2239  
## # ... with 1,790 more rows, and 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>

# filter는 select 안해줘도 출력된다.  
# 운항건수가 365회 이상인 목적지로 간 운항만 추출하기  
flights %>% group\_by(dest) %>%   
 filter( n() > 365 ) %>%  
 select(year:day, dest, everything())

## # A tibble: 332,577 x 19  
## # Groups: dest [77]  
## year month day dest dep\_time sched\_dep\_time dep\_delay arr\_time  
## <int> <int> <int> <chr> <int> <int> <dbl> <int>  
## 1 2013 1 1 IAH 517 515 2 830  
## 2 2013 1 1 IAH 533 529 4 850  
## 3 2013 1 1 MIA 542 540 2 923  
## 4 2013 1 1 BQN 544 545 -1 1004  
## 5 2013 1 1 ATL 554 600 -6 812  
## 6 2013 1 1 ORD 554 558 -4 740  
## 7 2013 1 1 FLL 555 600 -5 913  
## 8 2013 1 1 IAD 557 600 -3 709  
## 9 2013 1 1 MCO 557 600 -3 838  
## 10 2013 1 1 ORD 558 600 -2 753  
## # ... with 332,567 more rows, and 11 more variables: sched\_arr\_time <int>,  
## # arr\_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,  
## # origin <chr>, air\_time <dbl>, distance <dbl>, hour <dbl>,  
## # minute <dbl>, time\_hour <dttm>

flights %>% group\_by(dest) %>%   
 filter( n() > 365 ) %>% summarise(n= n()) %>% arrange(desc(n))

## # A tibble: 77 x 2  
## dest n  
## <chr> <int>  
## 1 ORD 17283  
## 2 ATL 17215  
## 3 LAX 16174  
## 4 BOS 15508  
## 5 MCO 14082  
## 6 CLT 14064  
## 7 SFO 13331  
## 8 FLL 12055  
## 9 MIA 11728  
## 10 DCA 9705  
## # ... with 67 more rows

# 목적지별 운항 건수가 많은 것부터 정렬  
flights %>% group\_by(dest) %>%  
 mutate( n = n() ) %>% arrange(desc(n)) %>%  
 select(year:day, dest, n, everything())

## # A tibble: 336,776 x 20  
## # Groups: dest [105]  
## year month day dest n dep\_time sched\_dep\_time dep\_delay  
## <int> <int> <int> <chr> <int> <int> <int> <dbl>  
## 1 2013 1 1 ORD 17283 554 558 -4  
## 2 2013 1 1 ORD 17283 558 600 -2  
## 3 2013 1 1 ORD 17283 608 600 8  
## 4 2013 1 1 ORD 17283 629 630 -1  
## 5 2013 1 1 ORD 17283 656 700 -4  
## 6 2013 1 1 ORD 17283 709 700 9  
## 7 2013 1 1 ORD 17283 715 713 2  
## 8 2013 1 1 ORD 17283 739 745 -6  
## 9 2013 1 1 ORD 17283 749 710 39  
## 10 2013 1 1 ORD 17283 828 830 -2  
## # ... with 336,766 more rows, and 12 more variables: arr\_time <int>,  
## # sched\_arr\_time <int>, arr\_delay <dbl>, carrier <chr>, flight <int>,  
## # tailnum <chr>, origin <chr>, air\_time <dbl>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

# flights 자료에서 목적지별로 365 회 이상 운항한 data 를 popular\_dest  
popular\_dest <- flights %>% group\_by(dest) %>%   
 filter(n() > 365) # 인기있는 목적지 추출  
  
# 도착지연된 운항 시간 중에 해당운항이 딜레이된 시간비율?  
popular\_dest %>% filter(arr\_delay > 0) %>%   
 mutate(prop\_delay = arr\_delay / sum(arr\_delay)) %>%   
 select(year, month, day, dest, arr\_delay, prop\_delay)

## # A tibble: 131,106 x 6  
## # Groups: dest [77]  
## year month day dest arr\_delay prop\_delay  
## <int> <int> <int> <chr> <dbl> <dbl>  
## 1 2013 1 1 IAH 11 0.000111   
## 2 2013 1 1 IAH 20 0.000201   
## 3 2013 1 1 MIA 33 0.000235   
## 4 2013 1 1 ORD 12 0.0000424  
## 5 2013 1 1 FLL 19 0.0000938  
## 6 2013 1 1 ORD 8 0.0000283  
## 7 2013 1 1 LAX 7 0.0000344  
## 8 2013 1 1 DFW 31 0.000282   
## 9 2013 1 1 ATL 12 0.0000400  
## 10 2013 1 1 DTW 16 0.000116   
## # ... with 131,096 more rows

popular\_dest %>% summarise(a = sum(arr\_delay)) # NA 가 있는데 arregate 하면 결측

## # A tibble: 77 x 2  
## dest a  
## <chr> <dbl>  
## 1 ALB NA  
## 2 ATL NA  
## 3 AUS NA  
## 4 BDL NA  
## 5 BGR NA  
## 6 BNA NA  
## 7 BOS NA  
## 8 BQN NA  
## 9 BTV NA  
## 10 BUF NA  
## # ... with 67 more rows

popular\_dest %>% summarise(a = sum(arr\_delay, na.rm = TRUE)) # NA 제외 후 계산

## # A tibble: 77 x 2  
## dest a  
## <chr> <dbl>  
## 1 ALB 6018  
## 2 ATL 190260  
## 3 AUS 14514  
## 4 BDL 2904  
## 5 BGR 2874  
## 6 BNA 71867  
## 7 BOS 43780  
## 8 BQN 7322  
## 9 BTV 22467  
## 10 BUF 40883  
## # ... with 67 more rows

popular\_dest %>% filter(arr\_delay > 0) %>% # NA 는 제외된다  
 mutate(prop\_delay = arr\_delay / sum(arr\_delay)) %>% # 여기서는 왜 na.rm =TRUE 안하는지? 아 하네  
 select(year, month, day, dest, arr\_delay, prop\_delay)

## # A tibble: 131,106 x 6  
## # Groups: dest [77]  
## year month day dest arr\_delay prop\_delay  
## <int> <int> <int> <chr> <dbl> <dbl>  
## 1 2013 1 1 IAH 11 0.000111   
## 2 2013 1 1 IAH 20 0.000201   
## 3 2013 1 1 MIA 33 0.000235   
## 4 2013 1 1 ORD 12 0.0000424  
## 5 2013 1 1 FLL 19 0.0000938  
## 6 2013 1 1 ORD 8 0.0000283  
## 7 2013 1 1 LAX 7 0.0000344  
## 8 2013 1 1 DFW 31 0.000282   
## 9 2013 1 1 ATL 12 0.0000400  
## 10 2013 1 1 DTW 16 0.000116   
## # ... with 131,096 more rows

flights %>% summarise(a = sum(arr\_delay, na.rm = TRUE))

## # A tibble: 1 x 1  
## a  
## <dbl>  
## 1 2257174

11/30046

## [1] 0.0003661053

11/2257174

## [1] 4.87335e-06

# 각 그룹마다 새로운 변수 생성하기  
A <- popular\_dest %>% # 목적지별로   
 filter(arr\_delay > 0) %>% # 실제로 도착이 지연된 운항  
 mutate(prop\_delay = arr\_delay / sum(arr\_delay)) %>% # 지연시간 비율  
 select(year, month, day, dest, arr\_delay, prop\_delay)   
A

## # A tibble: 131,106 x 6  
## # Groups: dest [77]  
## year month day dest arr\_delay prop\_delay  
## <int> <int> <int> <chr> <dbl> <dbl>  
## 1 2013 1 1 IAH 11 0.000111   
## 2 2013 1 1 IAH 20 0.000201   
## 3 2013 1 1 MIA 33 0.000235   
## 4 2013 1 1 ORD 12 0.0000424  
## 5 2013 1 1 FLL 19 0.0000938  
## 6 2013 1 1 ORD 8 0.0000283  
## 7 2013 1 1 LAX 7 0.0000344  
## 8 2013 1 1 DFW 31 0.000282   
## 9 2013 1 1 ATL 12 0.0000400  
## 10 2013 1 1 DTW 16 0.000116   
## # ... with 131,096 more rows

A %>% summarise(n = sum(arr\_delay, na.rm = TRUE)) # 목적지별로 지연시간 총합

## # A tibble: 77 x 2  
## dest n  
## <chr> <dbl>  
## 1 ALB 9580  
## 2 ATL 300299  
## 3 AUS 39940  
## 4 BDL 6953  
## 5 BGR 6940  
## 6 BNA 122152  
## 7 BOS 190190  
## 8 BQN 12872  
## 9 BTV 41288  
## 10 BUF 76478  
## # ... with 67 more rows

class(A)

## [1] "grouped\_df" "tbl\_df" "tbl" "data.frame"

class(flights)

## [1] "tbl\_df" "tbl" "data.frame"

class(popular\_dest) # 그룹화된 data frame

## [1] "grouped\_df" "tbl\_df" "tbl" "data.frame"

y <- data.frame(x = c(1,2,NA))  
y %>% filter(x > 1)

## x  
## 1 2

##################  
# tibble, readr, and tidyr  
  
# tibbles   
# - data frame 을 좀 더 편하게 사용할 수 있도록 변형시켜 놓은 class  
# - data frame 과 상호교환 가능  
  
# tibbles 만들기  
# as\_tibble() : data frame 을 tibble 로 바꿔주는 함수   
  
head(iris)

## Sepal.Length Sepal.Width Petal.Length Petal.Width Species  
## 1 5.1 3.5 1.4 0.2 setosa  
## 2 4.9 3.0 1.4 0.2 setosa  
## 3 4.7 3.2 1.3 0.2 setosa  
## 4 4.6 3.1 1.5 0.2 setosa  
## 5 5.0 3.6 1.4 0.2 setosa  
## 6 5.4 3.9 1.7 0.4 setosa

as\_tibble(iris)

## # A tibble: 150 x 5  
## Sepal.Length Sepal.Width Petal.Length Petal.Width Species  
## <dbl> <dbl> <dbl> <dbl> <fct>   
## 1 5.1 3.5 1.4 0.2 setosa   
## 2 4.9 3 1.4 0.2 setosa   
## 3 4.7 3.2 1.3 0.2 setosa   
## 4 4.6 3.1 1.5 0.2 setosa   
## 5 5 3.6 1.4 0.2 setosa   
## 6 5.4 3.9 1.7 0.4 setosa   
## 7 4.6 3.4 1.4 0.3 setosa   
## 8 5 3.4 1.5 0.2 setosa   
## 9 4.4 2.9 1.4 0.2 setosa   
## 10 4.9 3.1 1.5 0.1 setosa   
## # ... with 140 more rows

# tibble() 을 이용하여 만들기  
  
# data frame 과의 차이점   
# - character 가 factor 로 바뀌지 않음  
# - R 에서 허용하지 않는 형태의 변수 이름 가능   
# - printing 에서 처음 10줄과 화면에 맞는 변수만을 보여줌  
  
tb <- tibble( `:)` = 'smile',   
 ` ` = 'space',  
 `2000` = 'number' )  
tb

## # A tibble: 1 x 3  
## `:)` ` ` `2000`  
## <chr> <chr> <chr>   
## 1 smile space number

tb$`:)`

## [1] "smile"

iris.tb <- as\_tibble(iris)  
  
# tribble() : SAS 의 cards 문과 같은 형태의 자료 입력도 허용  
# Create tibbles using an easier to read row-by-row layout.  
tribble(  
 ~x, ~y, ~z,  
 #--/--/--  
 "a", 2, 3.6,  
 "b", 1, 8.5  
)

## # A tibble: 2 x 3  
## x y z  
## <chr> <dbl> <dbl>  
## 1 a 2 3.6  
## 2 b 1 8.5

tribble(~x,~y,~z, #--/--/-- "a",2,3.6,"b",1,8.5))  
) # A tibble: 0 x 3

## # A tibble: 0 x 3  
## # ... with 3 variables: x <lgl>, y <lgl>, z <lgl>

tribble(~x,~y,~z, #--/--/--  
 "a",2,3.6,"b",1,8.5)

## # A tibble: 2 x 3  
## x y z  
## <chr> <dbl> <dbl>  
## 1 a 2 3.6  
## 2 b 1 8.5

tribble(~x,~y,~z,"a",2,3.6,"b",1,8.5)

## # A tibble: 2 x 3  
## x y z  
## <chr> <dbl> <dbl>  
## 1 a 2 3.6  
## 2 b 1 8.5

######################################  
# Printing  
# - data frame 모든 자료를 보여줌  
# - tibble : 처음 10 줄과 화면에 맞는 만큼의 변수만을 보여주며 변수의 type 도 함께 보여줌  
A <- tibble( a = lubridate::now() + runif(1000)\*86400, # <dttm>   
 b = lubridate::today() + runif(1000)\*30, # <date>   
 c = 1:1000,  
 d = runif(1000), # random number of uniform distribution. n = 1000, min = 0, max = 1  
 e = sample(letters, 1000, replace = TRUE)) # should sampling be with replacement? 복원추출  
A

## # A tibble: 1,000 x 5  
## a b c d e   
## <dttm> <date> <int> <dbl> <chr>  
## 1 2018-12-03 21:54:57 2018-12-26 1 0.547 e   
## 2 2018-12-04 09:55:13 2018-12-12 2 0.151 d   
## 3 2018-12-04 01:23:32 2018-12-09 3 0.565 t   
## 4 2018-12-03 22:03:23 2018-12-04 4 0.00371 m   
## 5 2018-12-04 02:16:39 2018-12-06 5 0.942 q   
## 6 2018-12-04 13:16:46 2018-12-05 6 0.797 n   
## 7 2018-12-04 17:23:17 2018-12-09 7 0.0529 e   
## 8 2018-12-04 08:08:28 2018-12-11 8 0.268 j   
## 9 2018-12-04 08:42:30 2018-12-06 9 0.414 q   
## 10 2018-12-04 11:26:48 2018-12-04 10 0.775 z   
## # ... with 990 more rows

lubridate::now() # 현재 시각

## [1] "2018-12-03 17:52:48 KST"

lubridate::today() # 오늘 날짜

## [1] "2018-12-03"

sample(letters); sample(LETTERS)

## [1] "k" "c" "h" "p" "w" "a" "r" "d" "u" "b" "l" "v" "q" "i" "s" "o" "f"  
## [18] "t" "m" "x" "z" "j" "n" "g" "e" "y"

## [1] "S" "F" "M" "U" "K" "T" "A" "X" "L" "N" "B" "V" "E" "H" "J" "O" "C"  
## [18] "P" "D" "R" "Y" "Z" "Q" "G" "I" "W"

86400/60 #초 -> 분

## [1] 1440

86400/60/60 #초 -> 분 -> 시간

## [1] 24

540%/%100 # 몫 (시간)

## [1] 5

540%%100 # 나머지 (분)

## [1] 40

# n, width 옵션을 이용하여 자료 전체를 볼 수 있음  
# defualt print option 을 바꿀 수도 있음  
# - options(tibble.print\_max = n, tibble.print\_min = m) : 자료가 m 줄 이상인 경우 처음 n 줄만을 인쇄  
# - options(dplyr.print\_min = Inf) : 항상 모든 자료를 인쇄  
# - options(tibble.width = Inf) : 항상 모든 변수를 인쇄  
# - View() : 자료 전체를 보여줌  
  
print(A)

## # A tibble: 1,000 x 5  
## a b c d e   
## <dttm> <date> <int> <dbl> <chr>  
## 1 2018-12-03 21:54:57 2018-12-26 1 0.547 e   
## 2 2018-12-04 09:55:13 2018-12-12 2 0.151 d   
## 3 2018-12-04 01:23:32 2018-12-09 3 0.565 t   
## 4 2018-12-03 22:03:23 2018-12-04 4 0.00371 m   
## 5 2018-12-04 02:16:39 2018-12-06 5 0.942 q   
## 6 2018-12-04 13:16:46 2018-12-05 6 0.797 n   
## 7 2018-12-04 17:23:17 2018-12-09 7 0.0529 e   
## 8 2018-12-04 08:08:28 2018-12-11 8 0.268 j   
## 9 2018-12-04 08:42:30 2018-12-06 9 0.414 q   
## 10 2018-12-04 11:26:48 2018-12-04 10 0.775 z   
## # ... with 990 more rows

print(A, n = 20) # n = Inf 모든 행 인쇄

## # A tibble: 1,000 x 5  
## a b c d e   
## <dttm> <date> <int> <dbl> <chr>  
## 1 2018-12-03 21:54:57 2018-12-26 1 0.547 e   
## 2 2018-12-04 09:55:13 2018-12-12 2 0.151 d   
## 3 2018-12-04 01:23:32 2018-12-09 3 0.565 t   
## 4 2018-12-03 22:03:23 2018-12-04 4 0.00371 m   
## 5 2018-12-04 02:16:39 2018-12-06 5 0.942 q   
## 6 2018-12-04 13:16:46 2018-12-05 6 0.797 n   
## 7 2018-12-04 17:23:17 2018-12-09 7 0.0529 e   
## 8 2018-12-04 08:08:28 2018-12-11 8 0.268 j   
## 9 2018-12-04 08:42:30 2018-12-06 9 0.414 q   
## 10 2018-12-04 11:26:48 2018-12-04 10 0.775 z   
## 11 2018-12-04 10:29:36 2018-12-26 11 0.184 i   
## 12 2018-12-03 23:07:12 2018-12-19 12 0.850 a   
## 13 2018-12-04 07:07:02 2018-12-30 13 0.273 e   
## 14 2018-12-04 12:28:48 2018-12-03 14 0.528 m   
## 15 2018-12-04 01:13:20 2018-12-19 15 0.418 t   
## 16 2018-12-04 00:13:36 2018-12-07 16 0.495 s   
## 17 2018-12-03 20:38:11 2018-12-13 17 0.937 h   
## 18 2018-12-03 17:58:19 2018-12-09 18 0.803 g   
## 19 2018-12-03 22:39:21 2019-01-01 19 0.635 u   
## 20 2018-12-04 06:52:59 2018-12-30 20 0.478 p   
## # ... with 980 more rows

flights %>% print(n = 5, width = Inf) # width = Inf 모든 변수 인쇄

## # A tibble: 336,776 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  
## sched\_arr\_time arr\_delay carrier flight tailnum origin dest air\_time  
## <int> <dbl> <chr> <int> <chr> <chr> <chr> <dbl>  
## 1 819 11 UA 1545 N14228 EWR IAH 227  
## 2 830 20 UA 1714 N24211 LGA IAH 227  
## 3 850 33 AA 1141 N619AA JFK MIA 160  
## 4 1022 -18 B6 725 N804JB JFK BQN 183  
## 5 837 -25 DL 461 N668DN LGA ATL 116  
## distance hour minute time\_hour   
## <dbl> <dbl> <dbl> <dttm>   
## 1 1400 5 15 2013-01-01 05:00:00  
## 2 1416 5 29 2013-01-01 05:00:00  
## 3 1089 5 40 2013-01-01 05:00:00  
## 4 1576 5 45 2013-01-01 05:00:00  
## 5 762 6 0 2013-01-01 06:00:00  
## # ... with 3.368e+05 more rows

summary(lm(Sepal.Length ~ Sepal.Width, data =iris))

##   
## Call:  
## lm(formula = Sepal.Length ~ Sepal.Width, data = iris)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -1.5561 -0.6333 -0.1120 0.5579 2.2226   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 6.5262 0.4789 13.63 <2e-16 \*\*\*  
## Sepal.Width -0.2234 0.1551 -1.44 0.152   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.8251 on 148 degrees of freedom  
## Multiple R-squared: 0.01382, Adjusted R-squared: 0.007159   
## F-statistic: 2.074 on 1 and 148 DF, p-value: 0.1519

summary(iris)

## Sepal.Length Sepal.Width Petal.Length Petal.Width   
## Min. :4.300 Min. :2.000 Min. :1.000 Min. :0.100   
## 1st Qu.:5.100 1st Qu.:2.800 1st Qu.:1.600 1st Qu.:0.300   
## Median :5.800 Median :3.000 Median :4.350 Median :1.300   
## Mean :5.843 Mean :3.057 Mean :3.758 Mean :1.199   
## 3rd Qu.:6.400 3rd Qu.:3.300 3rd Qu.:5.100 3rd Qu.:1.800   
## Max. :7.900 Max. :4.400 Max. :6.900 Max. :2.500   
## Species   
## setosa :50   
## versicolor:50   
## virginica :50   
##   
##   
##

# print(iris)  
# print(iris.tb)  
  
  
# tibble()  
df <- tibble(x = runif(5),   
 y = runif(5))  
df

## # A tibble: 5 x 2  
## x y  
## <dbl> <dbl>  
## 1 0.301 0.360  
## 2 0.289 0.395  
## 3 0.877 0.826  
## 4 0.953 0.735  
## 5 0.979 0.134

###########################  
# Subsetting  
# data frame 과 동일한 방법 이용 가능  
# tibble 일 때 출력되는 형식  
set.seed(20181101) # 고정  
df <- tibble(x = runif(5),   
 y = runif(5))  
df

## # A tibble: 5 x 2  
## x y  
## <dbl> <dbl>  
## 1 0.779 0.0479  
## 2 0.771 0.768   
## 3 0.500 0.771   
## 4 0.616 0.674   
## 5 0.618 0.783

# Extract by name  
df$x # $ vector

## [1] 0.7789209 0.7710086 0.5000731 0.6163632 0.6183271

df[['x']] # [[]] 열을 vector 로 뽑는다

## [1] 0.7789209 0.7710086 0.5000731 0.6163632 0.6183271

# Extract by position  
# Often what we will require is just the vector representing the values in the variable.   
# This is achieved using a different sort of indexing that uses double square brackets,  
df[[2]]

## [1] 0.0478566 0.7675562 0.7709332 0.6744193 0.7834728

# If a single index is specified when subsetting a data frame with single square brackets,   
# the effect is to extract the appropriate columns of the data frame and all rows are returned.  
df[1]

## # A tibble: 5 x 1  
## x  
## <dbl>  
## 1 0.779  
## 2 0.771  
## 3 0.500  
## 4 0.616  
## 5 0.618

df[1]

## # A tibble: 5 x 1  
## x  
## <dbl>  
## 1 0.779  
## 2 0.771  
## 3 0.500  
## 4 0.616  
## 5 0.618

#When subsetting using square brackets,   
# it is possible to leave the row or column index completely empty.   
# The result is that all rows or all columns, respectively, are returned.   
df[,'x'] # [, 'x']

## # A tibble: 5 x 1  
## x  
## <dbl>  
## 1 0.779  
## 2 0.771  
## 3 0.500  
## 4 0.616  
## 5 0.618

# data frame 일 때 출력되는 형식  
df1 <- data.frame(x = runif(5),   
 y = runif(5))  
df1 # 소수점 8 자리수

## x y  
## 1 0.3454878 0.9759598  
## 2 0.9242048 0.2710545  
## 3 0.6260116 0.5472805  
## 4 0.5492716 0.6151937  
## 5 0.6461395 0.6347806

df1$x # vector 로 출력

## [1] 0.3454878 0.9242048 0.6260116 0.5492716 0.6461395

df1[['x']] # vector 로 출력

## [1] 0.3454878 0.9242048 0.6260116 0.5492716 0.6461395

df1[,'x'] # vector 로 출력 "numeric"

## [1] 0.3454878 0.9242048 0.6260116 0.5492716 0.6461395

df1[,1] # vector 로 출력

## [1] 0.3454878 0.9242048 0.6260116 0.5492716 0.6461395

df1[,c('x','y')] # data frame 으로 출력

## x y  
## 1 0.3454878 0.9759598  
## 2 0.9242048 0.2710545  
## 3 0.6260116 0.5472805  
## 4 0.5492716 0.6151937  
## 5 0.6461395 0.6347806

df1[,'x', drop = FALSE] # data frame 으로 출력

## x  
## 1 0.3454878  
## 2 0.9242048  
## 3 0.6260116  
## 4 0.5492716  
## 5 0.6461395

df.tbl <- tibble( xx = runif(5),   
 y = runif(5))  
df.DF <- data.frame( xx = runif(5),  
 y = runif(5))  
df.tbl

## # A tibble: 5 x 2  
## xx y  
## <dbl> <dbl>  
## 1 0.407 0.975  
## 2 0.802 0.566  
## 3 0.233 0.457  
## 4 0.455 0.762  
## 5 0.235 0.730

df.DF

## xx y  
## 1 0.39965603 0.31311762  
## 2 0.96659746 0.94363788  
## 3 0.55862231 0.92070164  
## 4 0.14372857 0.05881271  
## 5 0.08510631 0.70900576

df.tbl$x # tibble 변수가 없으면 안뽑아줌

## Warning: Unknown or uninitialised column: 'x'.

## NULL

df.DF$x # data frame x 와 유사한 변수 불러옴

## [1] 0.39965603 0.96659746 0.55862231 0.14372857 0.08510631

df.DF$xx

## [1] 0.39965603 0.96659746 0.55862231 0.14372857 0.08510631

df.tbl$xx

## [1] 0.4074070 0.8016004 0.2328311 0.4552677 0.2347319

as.data.frame(df.tbl)

## xx y  
## 1 0.4074070 0.9752163  
## 2 0.8016004 0.5662764  
## 3 0.2328311 0.4573373  
## 4 0.4552677 0.7619291  
## 5 0.2347319 0.7303479

# partial matching 은 불가능 (data frame 에서는 가능).  
  
# as.dataframe() : tibble 을 data frame 으로 바꾸기   
class(as.data.frame(df.tbl))

## [1] "data.frame"