Factors and Date

박찬영

2024-08-29

tidyverse와 forcats, lubridate library를 사용합니다. 팩터형은 forcats, 날짜형은 lubridate를 쓰는게 좋다. (tidyverse에 없음)

## 팩터형

팩터형 데이터는 범주형 변수를 나타내는 자료형이다

```
month_levels = c(
    "Jan","Feb","Mar","Apr","May","Jun",
    "Jul","Aug","Sep","Oct","Nov","Dec"
    ) #가질 수 있는 모든 값의 집합 (level 이라고 함)

x1= factor(c("Mar","Nov","Sep","Jan"), levels=month_levels)
x1
```

```
## [1] Mar Nov Sep Jan
## Levels: Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
```

```
#기본적으로 벡터를 가지고 변환하는 형식이다

sort(x1) #이러면 알파벳순이 아닌 levels 기준으로 정렬해준다
```

```
## [1] Jan Mar Sep Nov
## Levels: Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
```

```
x2=c("Jan","zz","Dec")
factor(x2,levels=month_levels) #없는 값은 NA로
```

```
## [1] Jan  <NA> Dec
## Levels: Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
```

```
levels(x1) #가질수 있는 모든 수준 표시
```

```
##  [1] "Jan" "Feb" "Mar" "Apr" "May" "Jun" "Jul" "Aug" "Sep" "Oct" "Nov" "Dec"
```

```
gss_cat
```

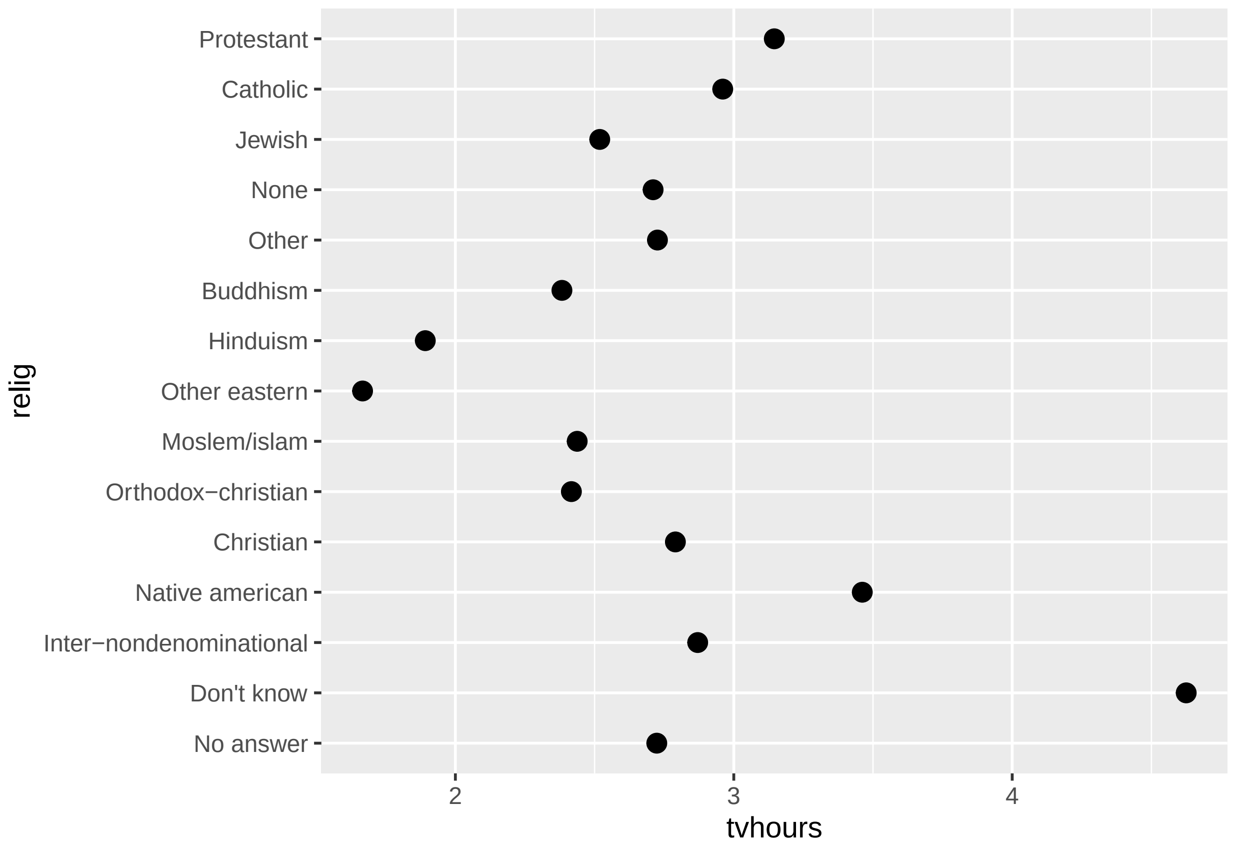
```
## # A tibble: 21,483 x 9
##     year marital         age race  rincome        partyid    relig denom tvhours
##    <int> <fct>         <int> <fct> <fct>          <fct>      <fct> <fct>   <int>
##  1  2000 Never married    26 White $8000 to 9999  Ind,near ~ Prot~ Sout~      12
##  2  2000 Divorced         48 White $8000 to 9999  Not str r~ Prot~ Bapt~      NA
##  3  2000 Widowed          67 White Not applicable Independe~ Prot~ No d~       2
##  4  2000 Never married    39 White Not applicable Ind,near ~ Orth~ Not ~       4
##  5  2000 Divorced         25 White Not applicable Not str d~ None  Not ~       1
##  6  2000 Married          25 White $20000 - 24999 Strong de~ Prot~ Sout~      NA
##  7  2000 Never married    36 White $25000 or more Not str r~ Chri~ Not ~       3
##  8  2000 Divorced         44 White $7000 to 7999  Ind,near ~ Prot~ Luth~      NA
##  9  2000 Married          44 White $25000 or more Not str d~ Prot~ Other       0
## 10  2000 Married          47 White $25000 or more Strong re~ Prot~ Sout~       3
## # i 21,473 more rows
```

```
#몇몇 변수들은 fct 형으로 저장되어있다
gss_cat %>% count(race) #수준은 3개
```

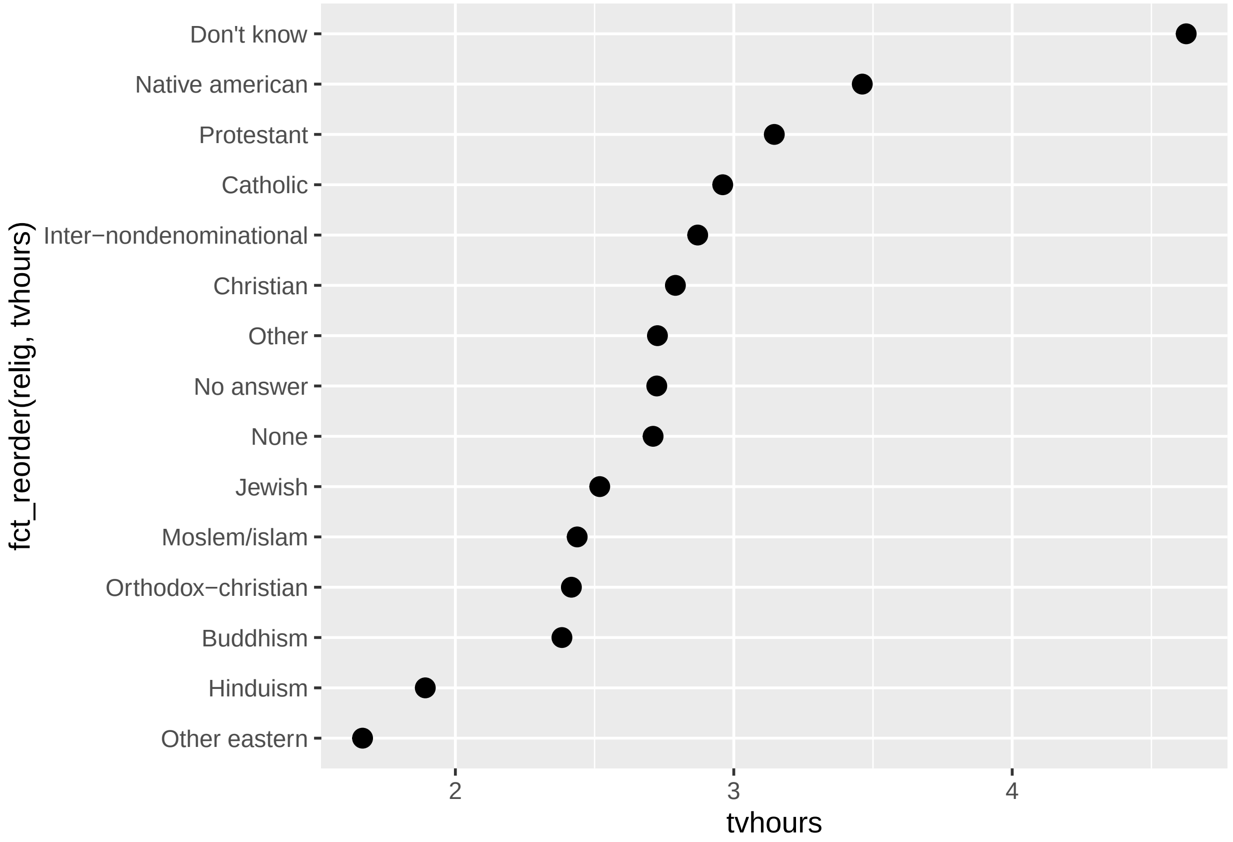
```
## # A tibble: 3 x 2
##   race      n
##   <fct> <int>
## 1 Other  1959
## 2 Black  3129
## 3 White 16395
```

팩터형 자료를 정렬하고 수정해보자

```
relig_summary = gss_cat %>%
    group_by(relig) %>%
    summarise(
        age=mean(age,na.rm=TRUE),
        tvhours=mean(tvhours, na.rm=TRUE),
        n=n()
        )



ggplot(relig_summary, aes(tvhours, relig)) +geom_point(size=3)
```



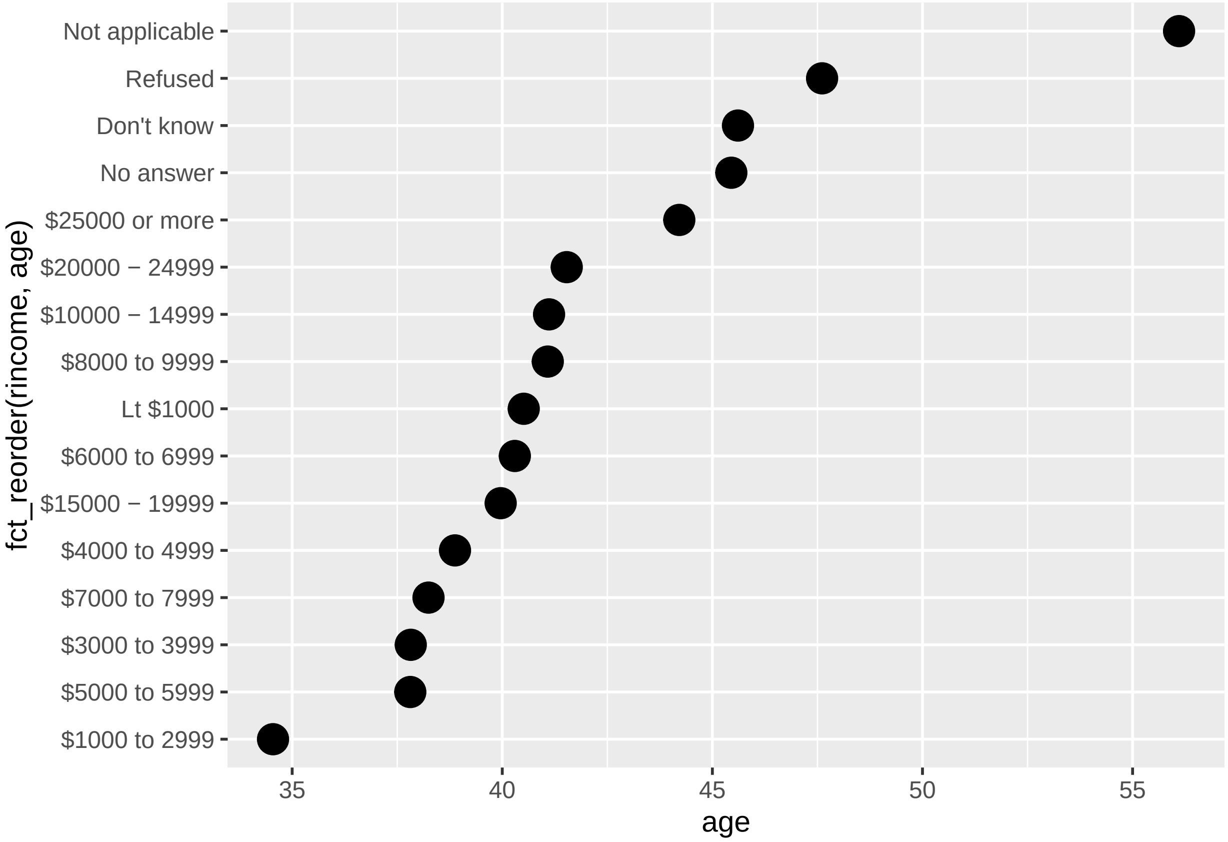
```
ggplot(relig_summary, aes(tvhours, fct_reorder(relig,tvhours))) +geom_point(size=3)
```



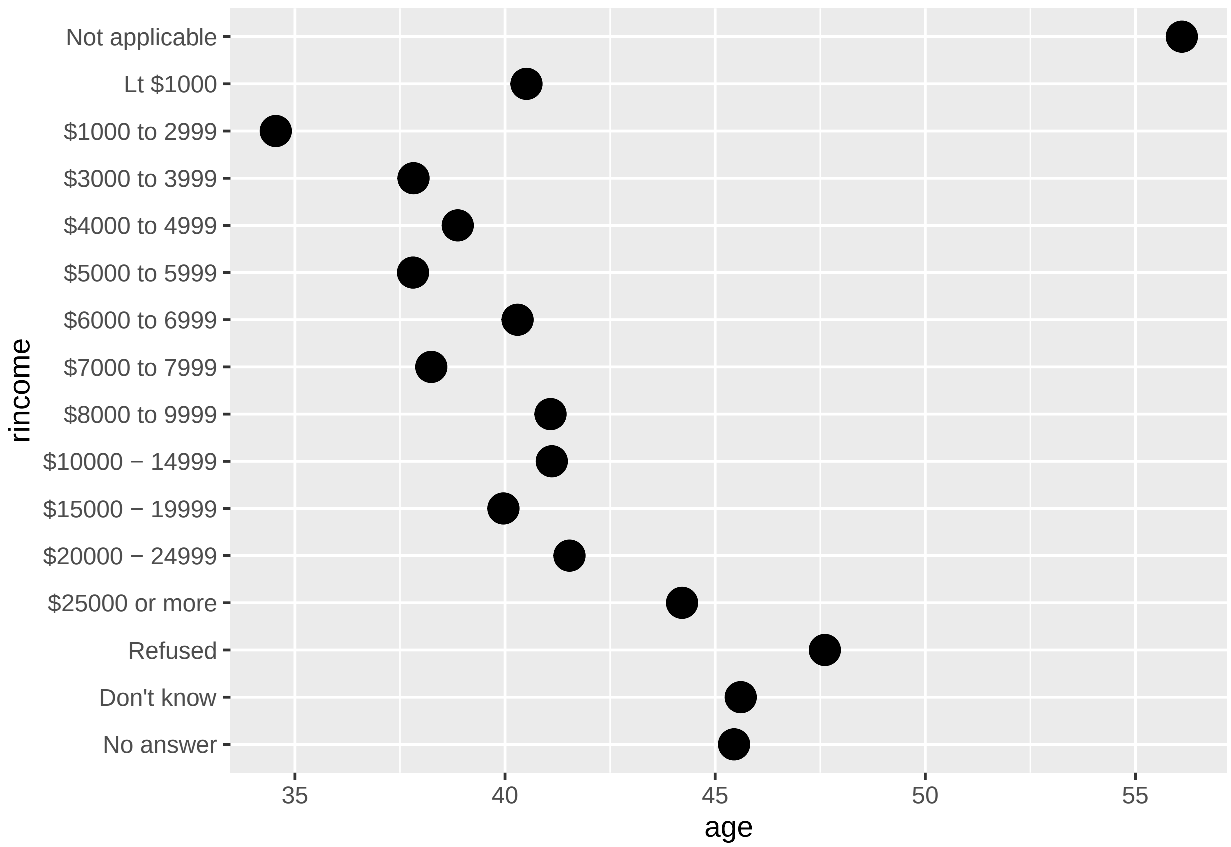
```
fct_reorder(relig_summary$relig, relig_summary$tvhours)
```

```
##  [1] No answer               Don't know              Inter-nondenominational
##  [4] Native american         Christian               Orthodox-christian
##  [7] Moslem/islam            Other eastern           Hinduism
## [10] Buddhism                Other                   None
## [13] Jewish                  Catholic                Protestant
## 16 Levels: Other eastern Hinduism Buddhism Orthodox-christian ... Not applicable
```

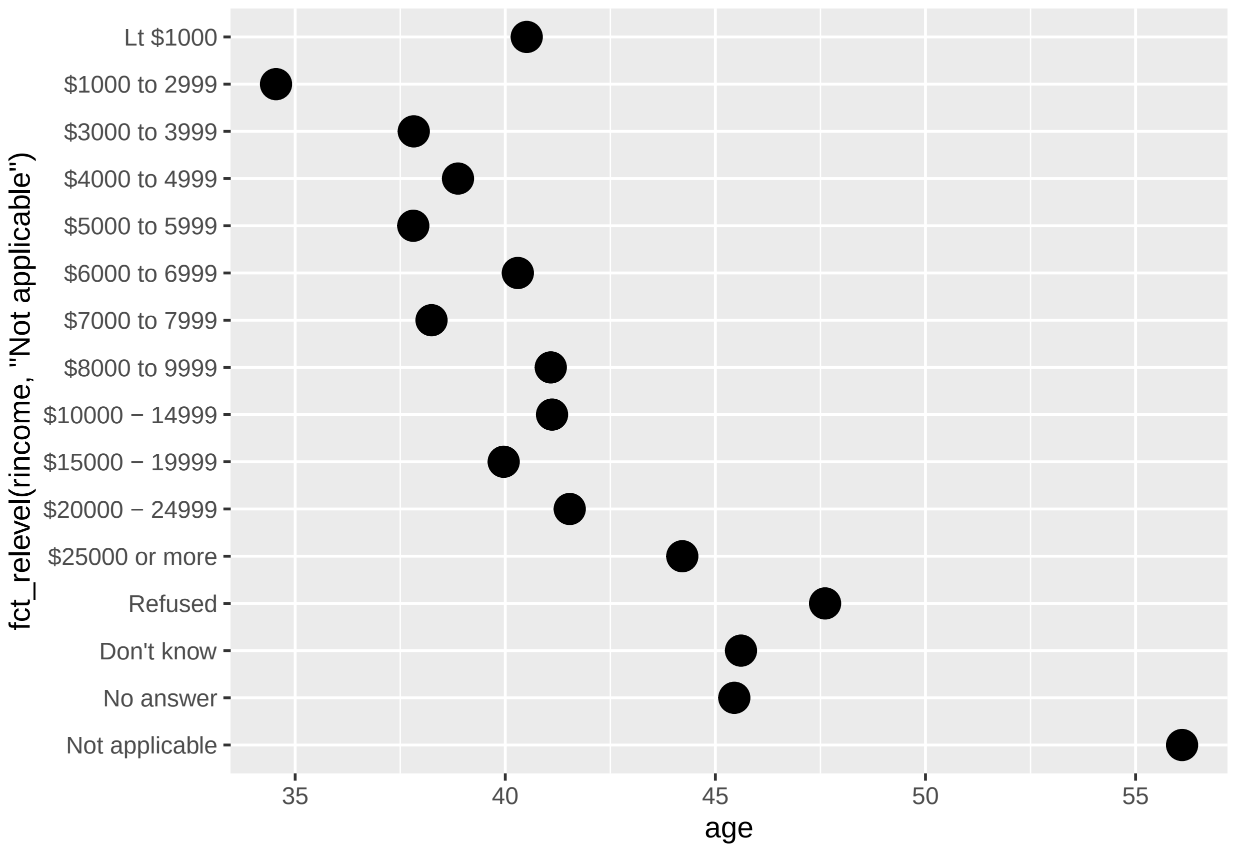
```
#fct_reorder는 팩터를 많은 순서대로 정렬한다

rincome_summary= gss_cat %>%
    group_by(rincome) %>%
    summarise(
        age=mean(age,na.rm=TRUE),
        tvhours=mean(tvhours,na.rm=TRUE),
        n=n()
        )

rincome_summary %>%
    ggplot(aes(age,fct_reorder(rincome,age)))+geom_point(size=5)
```



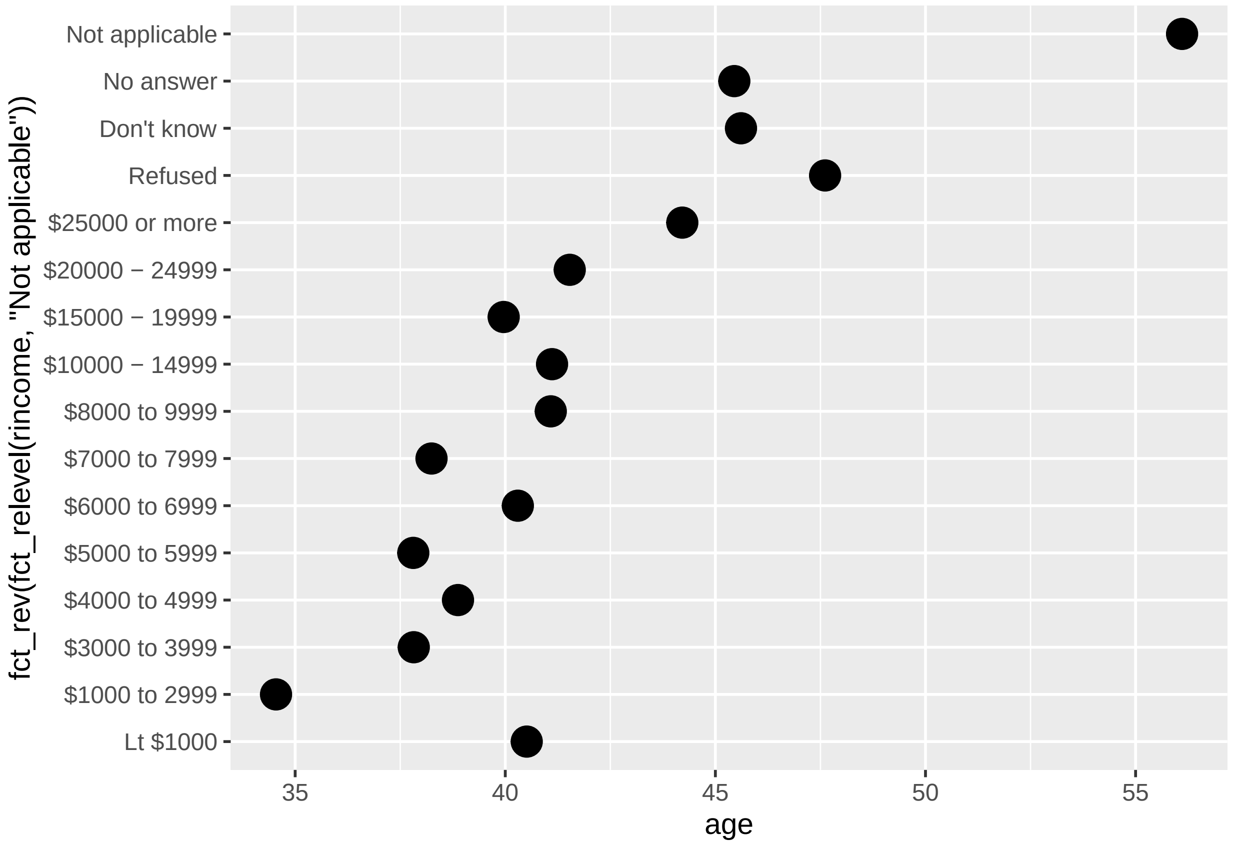
```
#다른 예제, 하지만 얘는 정렬 안하느게 좋음


rincome_summary %>%
    ggplot(aes(age,rincome))+geom_point(size=5)
```



```
#근데 Not applicable 위치를 내리고 싶다


rincome_summary %>%
    ggplot(aes(age,fct_relevel(rincome, "Not applicable")))+geom_point(size=5)
```



```
#fct_relevel은 수준의 순서변경, 먼저쓴게 맨 첫순서가 됨


rincome_summary %>%
    ggplot(aes(age,fct_rev(fct_relevel(rincome, "Not applicable"))))+geom_point(size=5)
```



```
#fct_rev는 역순서배치


gss_cat %>% count(partyid) #수준이 다양함
```

```
## # A tibble: 10 x 2
##    partyid                n
##    <fct>              <int>
##  1 No answer            154
##  2 Don't know             1
##  3 Other party          393
##  4 Strong republican   2314
##  5 Not str republican  3032
##  6 Ind,near rep        1791
##  7 Independent         4119
##  8 Ind,near dem        2499
##  9 Not str democrat    3690
## 10 Strong democrat     3490
```

```
gss_cat %>%
  mutate(partyid = fct_recode(partyid,
    "Republican, strong"    = "Strong republican",
    "Republican, weak"      = "Not str republican",
    "Independent, near rep" = "Ind,near rep",
    "Independent, near dem" = "Ind,near dem",
    "Democrat, weak"        = "Not str democrat",
    "Democrat, strong"      = "Strong democrat"
  )) %>%
  count(partyid) #약간 수정해보자
```

```
## # A tibble: 10 x 2
##    partyid                   n
##    <fct>                 <int>
##  1 No answer               154
##  2 Don't know                1
##  3 Other party             393
##  4 Republican, strong     2314
##  5 Republican, weak       3032
##  6 Independent, near rep  1791
##  7 Independent            4119
##  8 Independent, near dem  2499
##  9 Democrat, weak         3690
## 10 Democrat, strong       3490
```

```
#fct_recode는 수준을 조정하는 함수



gss_cat %>%
  mutate(partyid = fct_recode(partyid,
    "Republican, strong"    = "Strong republican",
    "Republican, weak"      = "Not str republican",
    "Independent, near rep" = "Ind,near rep",
    "Independent, near dem" = "Ind,near dem",
    "Democrat, weak"        = "Not str democrat",
    "Democrat, strong"      = "Strong democrat",
    "Other"                 = "No answer",
    "Other"                 = "Don't know",
    "Other"                 = "Other party"
  )) %>%
  count(partyid) #이렇게하면 Other로 통합시키기 가능
```

```
## # A tibble: 8 x 2
##   partyid                  n
##   <fct>                <int>
## 1 Other                  548
## 2 Republican, strong    2314
## 3 Republican, weak      3032
## 4 Independent, near rep 1791
## 5 Independent           4119
## 6 Independent, near dem 2499
## 7 Democrat, weak        3690
## 8 Democrat, strong      3490
```

```
gss_cat %>%
  mutate(partyid = fct_collapse(partyid,
    other = c("No answer", "Don't know", "Other party"),
    rep = c("Strong republican", "Not str republican"),
    ind = c("Ind,near rep", "Independent", "Ind,near dem"),
    dem = c("Not str democrat", "Strong democrat")
  )) %>%
  count(partyid) #이런 방식도 가능
```

```
## # A tibble: 4 x 2
##   partyid     n
##   <fct>   <int>
## 1 other     548
## 2 rep      5346
## 3 ind      8409
## 4 dem      7180
```

```
#fct_collapse을 사용 벡터를 이용해 파괴해서 새롭게 할당

gss_cat %>%
  mutate(relig = fct_lump(relig, n = 10)) %>%
  count(relig, sort = TRUE) %>%
  print(n = Inf)
```

```
## # A tibble: 10 x 2
##    relig                        n
##    <fct>                    <int>
##  1 Protestant              10846
##  2 Catholic                 5124
##  3 None                     3523
##  4 Christian                 689
##  5 Other                     458
##  6 Jewish                    388
##  7 Buddhism                  147
##  8 Inter-nondenominational   109
##  9 Moslem/islam              104
## 10 Orthodox-christian         95
```

```
#fct_lump는 소규모 그룹들을 알아서 묶어준다
```

## 날짜 및 시간

날짜와 시간 데이터를 다뤄보자

날짜 시간 자료형으로는 datetime형과 date형이 있다. date형은 연월일, datetime형은 연월일시분초이다.

```
today() #오늘 날짜, 데이트형
```

```
## [1] "2024-09-02"
```

```
now() #시간까지, 데이트타임형
```

```
## [1] "2024-09-02 03:02:12 KST"
```

```
ymd("2017.01.31") #문자열을 날짜로
```

```
## [1] "2017-01-31"
```

```
myd("12/2003/13") #myd냐 ymd냐에 따라 순서를 자유롭게
```

```
## [1] "2003-12-13"
```

```
ymd_hms("2024/08/29/15/22/15") #연월일 시분초
```

```
## [1] "2024-08-29 15:22:15 UTC"
```

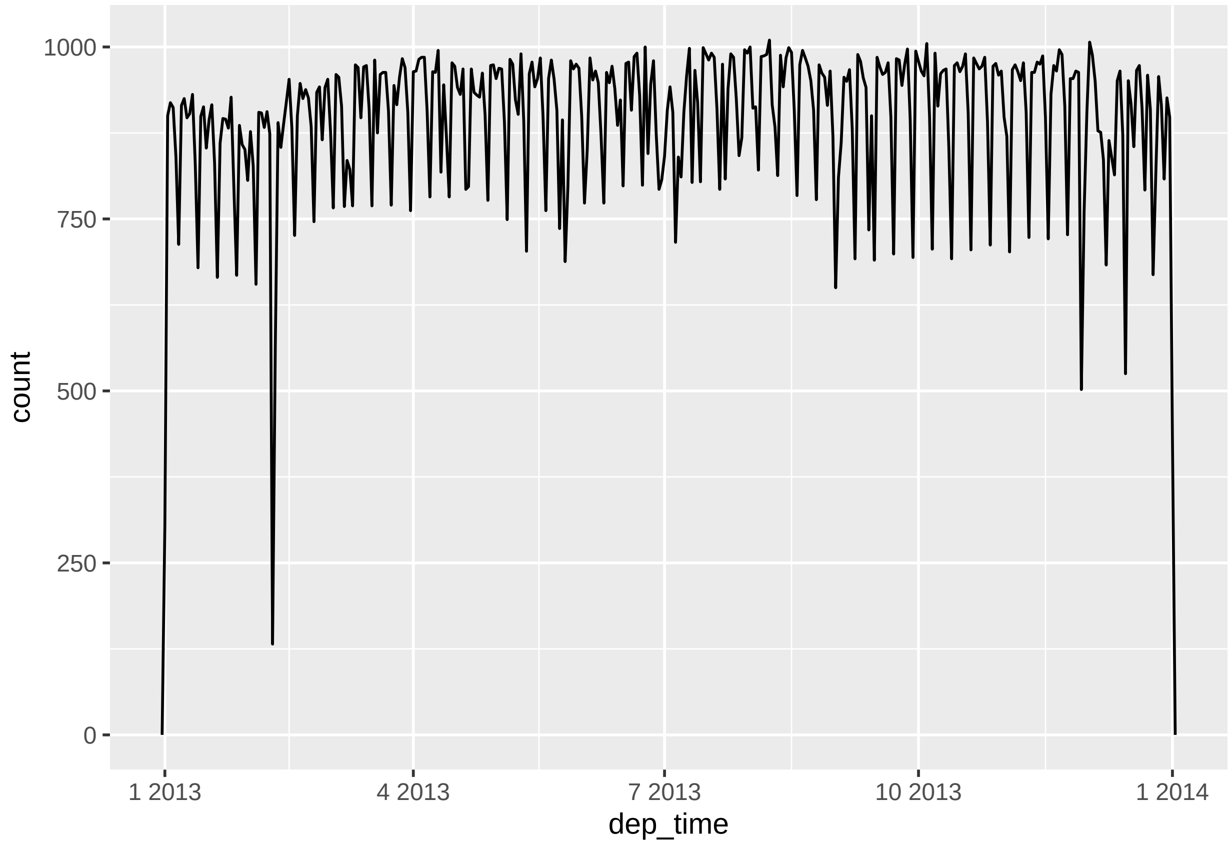
```
#얘들은 다 조합가능 mdy_ms dy_hm 등등


#쪼개진 데이터로 날짜 만들기


fl2= flights %>% select(year,month,day,hour,tailnum)


fl2 %>% mutate(depaprture = make_datetime(year, month, day,hour))
```

```
## # A tibble: 336,776 x 6
##     year month   day  hour tailnum depaprture
##    <int> <int> <int> <dbl> <chr>   <dttm>
##  1  2013     1     1     5 N14228  2013-01-01 05:00:00
##  2  2013     1     1     5 N24211  2013-01-01 05:00:00
##  3  2013     1     1     5 N619AA  2013-01-01 05:00:00
##  4  2013     1     1     5 N804JB  2013-01-01 05:00:00
##  5  2013     1     1     6 N668DN  2013-01-01 06:00:00
##  6  2013     1     1     5 N39463  2013-01-01 05:00:00
##  7  2013     1     1     6 N516JB  2013-01-01 06:00:00
##  8  2013     1     1     6 N829AS  2013-01-01 06:00:00
##  9  2013     1     1     6 N593JB  2013-01-01 06:00:00
## 10  2013     1     1     6 N3ALAA  2013-01-01 06:00:00
## # i 336,766 more rows
```

```r
#make_datetime은 날짜형을 만들어줌



make_datetime_100 <- function(year, month, day, time) {
  make_datetime(year, month, day, time %/% 100, time %% 100)
} #함수선언을 통해 만들기 함수는 나중에 배움

flights_dt <- flights %>%
  filter(!is.na(dep_time), !is.na(arr_time)) %>%
  mutate(
    dep_time = make_datetime_100(year, month, day, dep_time),
    arr_time = make_datetime_100(year, month, day, arr_time),
    sched_dep_time = make_datetime_100(year, month, day, sched_dep_time),
    sched_arr_time = make_datetime_100(year, month, day, sched_arr_time)
  ) %>%
  select(origin, dest, ends_with("delay"), ends_with("time"))

flights_dt #출발지 목적지 지연시간 예정시간 실제시간 데이터
```

```
## # A tibble: 328,063 x 9
##    origin dest  dep_delay arr_delay dep_time            sched_dep_time
##    <chr>  <chr>     <dbl>     <dbl> <dttm>              <dttm>
##  1 EWR    IAH           2        11 2013-01-01 05:17:00 2013-01-01 05:15:00
##  2 LGA    IAH           4        20 2013-01-01 05:33:00 2013-01-01 05:29:00
##  3 JFK    MIA           2        33 2013-01-01 05:42:00 2013-01-01 05:40:00
##  4 JFK    BQN          -1       -18 2013-01-01 05:44:00 2013-01-01 05:45:00
##   5 LGA     ATL            -6        -25 2013-01-01 05:54:00 2013-01-01 06:00:00
##   6 EWR     ORD            -4         12 2013-01-01 05:54:00 2013-01-01 05:58:00
##   7 EWR     FLL            -5         19 2013-01-01 05:55:00 2013-01-01 06:00:00
##   8 LGA     IAD            -3        -14 2013-01-01 05:57:00 2013-01-01 06:00:00
##   9 JFK     MCO            -3         -8 2013-01-01 05:57:00 2013-01-01 06:00:00
##  10 LGA     ORD            -2          8 2013-01-01 05:58:00 2013-01-01 06:00:00
## # i 328,053 more rows
## # i 3 more variables: arr_time <dttm>, sched_arr_time <dttm>, air_time <dbl>
```

```
flights_dt %>%
    ggplot(aes(dep_time)) +
    geom_freqpoly(binwidth=86400)
```



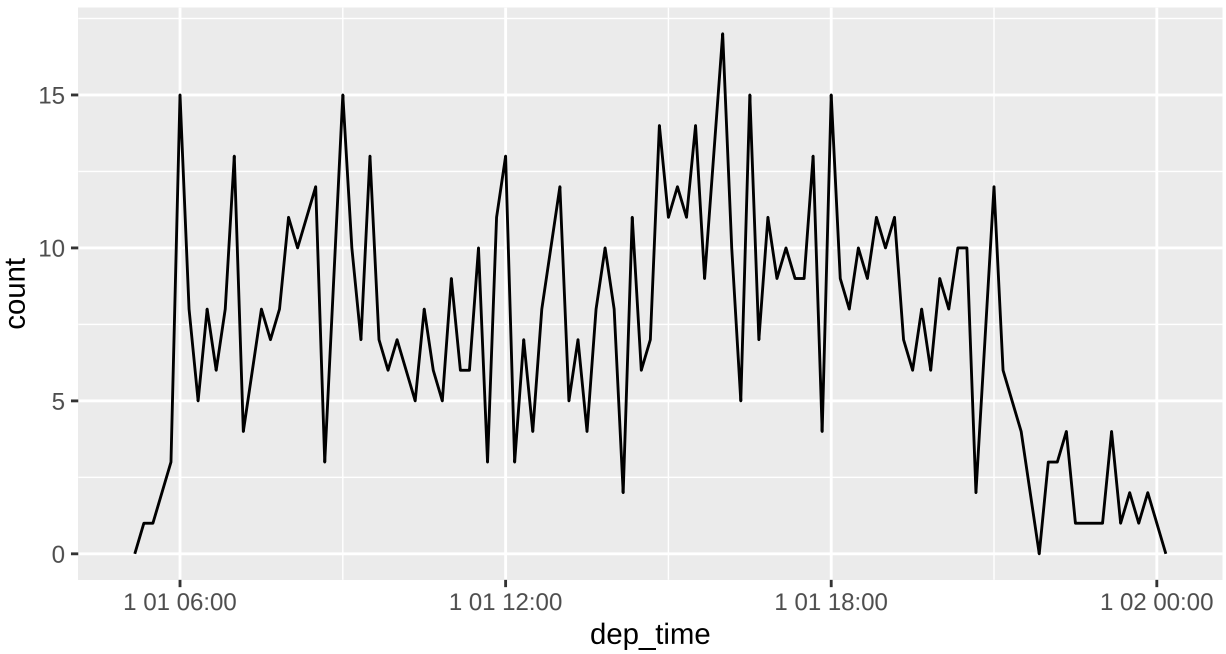
```
#하루 단위로 쪼개는 빈도선그래프

as_date(now()) #얘는 데이트-타임형을 데이트로 바꿈
```

```
## [1] "2024-09-02"
```

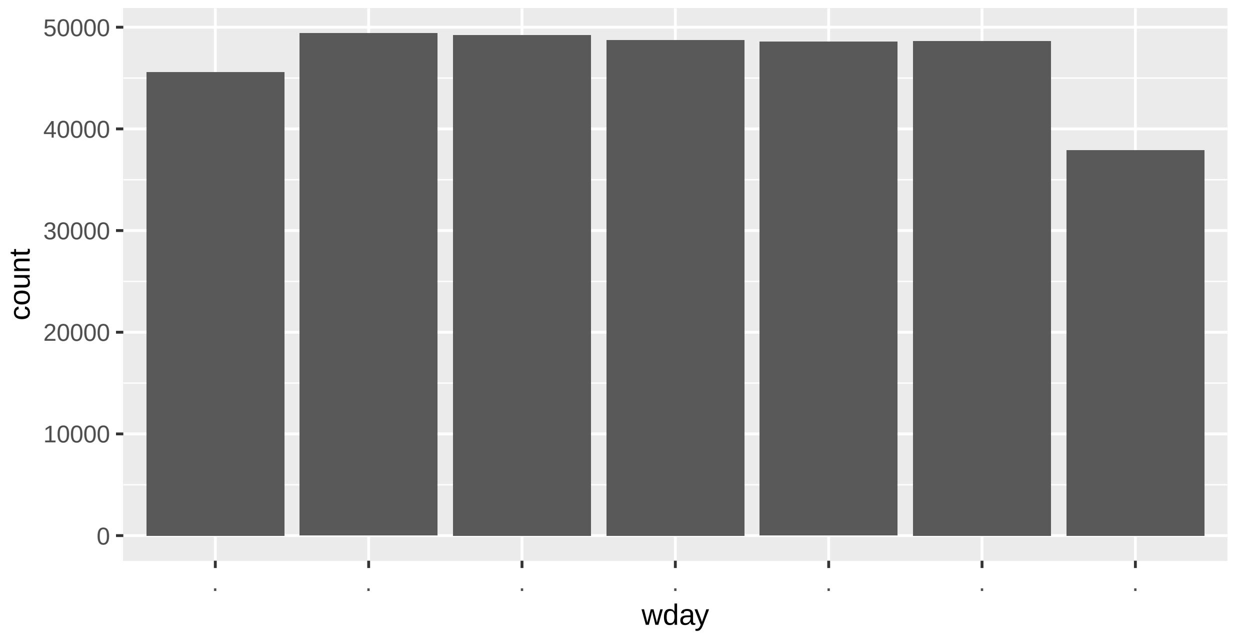
```
as_datetime(today()) #얘는 데이트형을 데이트 타임형으로 바꿈
```

```
## [1] "2024-09-02 UTC"
```

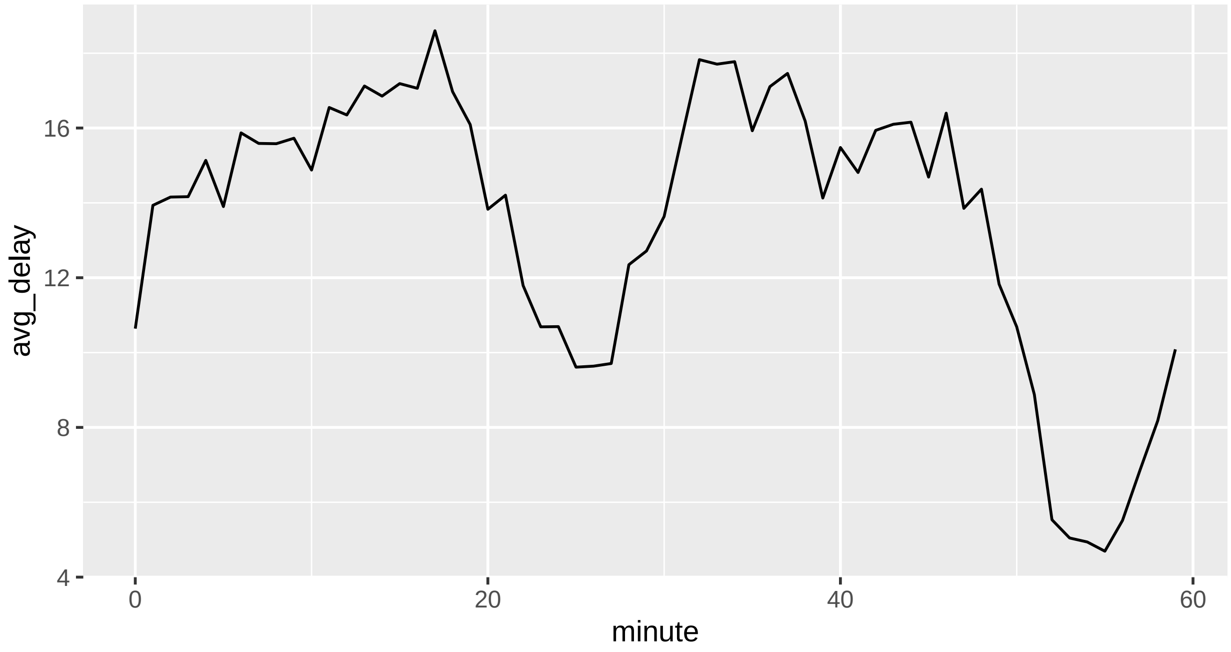
```
flights_dt %>%
    filter(dep_time < ymd(20130102)) %>% #1월1일만 보기
    ggplot(aes(dep_time)) +
    geom_freqpoly(binwidth=600) + #10분 단위로 끊기
    theme(aspect.ratio = 1/2)
```



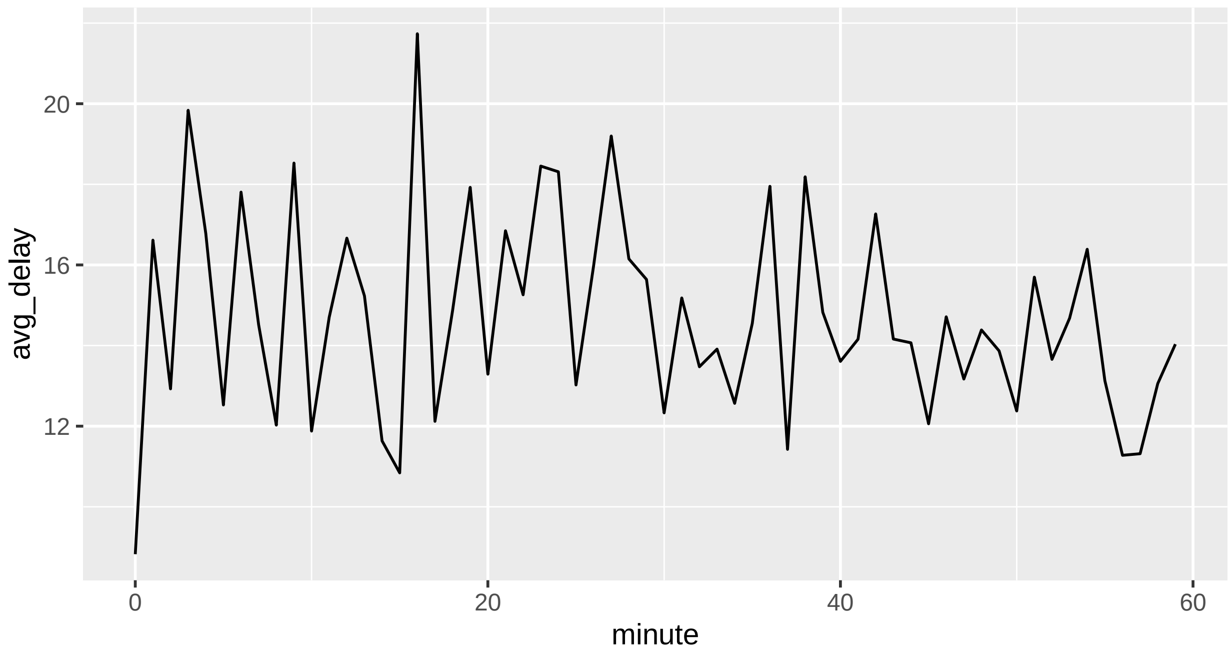
```
flights_dt %>%
    mutate(wday=wday(dep_time, labe=TRUE)) %>%
    ggplot(aes(wday)) +
    geom_bar() +
    theme(aspect.ratio = 1/2)
```



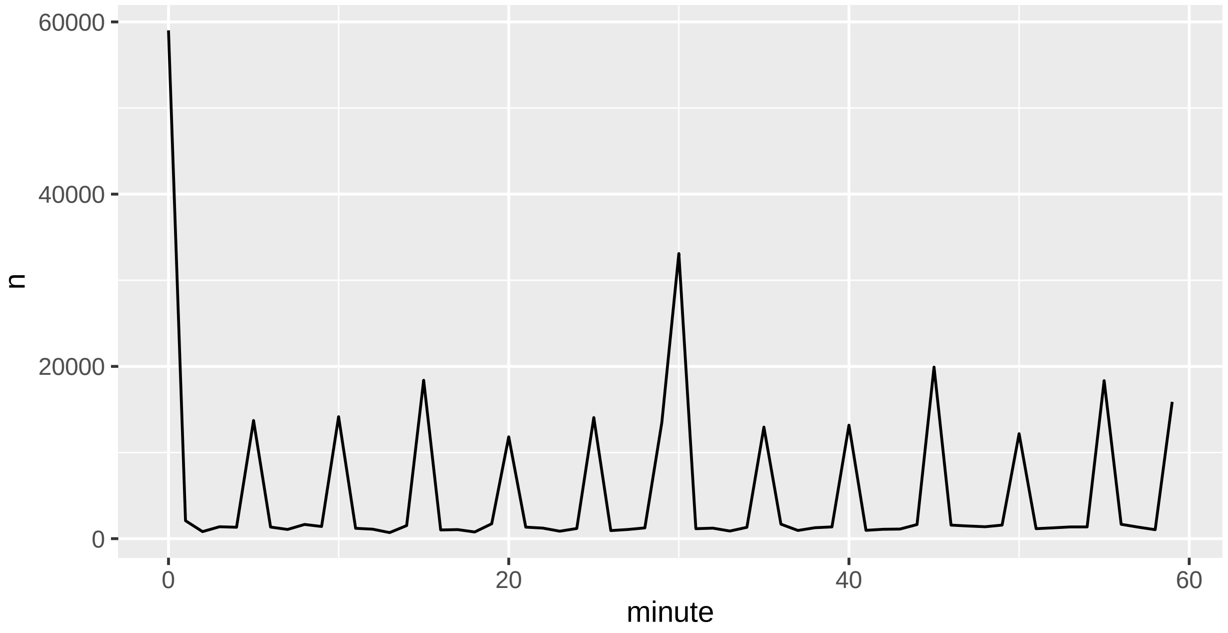
```
#wday는 날짜데이터를 넣으면 요일로 바꿔준다

flights_dt %>%
    mutate(minute=minute(dep_time)) %>%
    group_by(minute) %>%
    summarise(avg_delay=mean(dep_delay, na.rm=TRUE)) %>%
    ggplot(aes(minute, avg_delay)) +
    geom_line() +
    theme(aspect.ratio = 1/2)
```



```
#minute은 분만 남기기, 실제 출발 분에 따른 딜레이 평균

flights_dt %>%
    mutate(minute=minute(sched_dep_time)) %>%
    group_by(minute) %>%
    summarise(avg_delay=mean(dep_delay, na.rm=TRUE)) %>%
    ggplot(aes(minute, avg_delay)) +
    geom_line() +
    theme(aspect.ratio = 1/2)
```



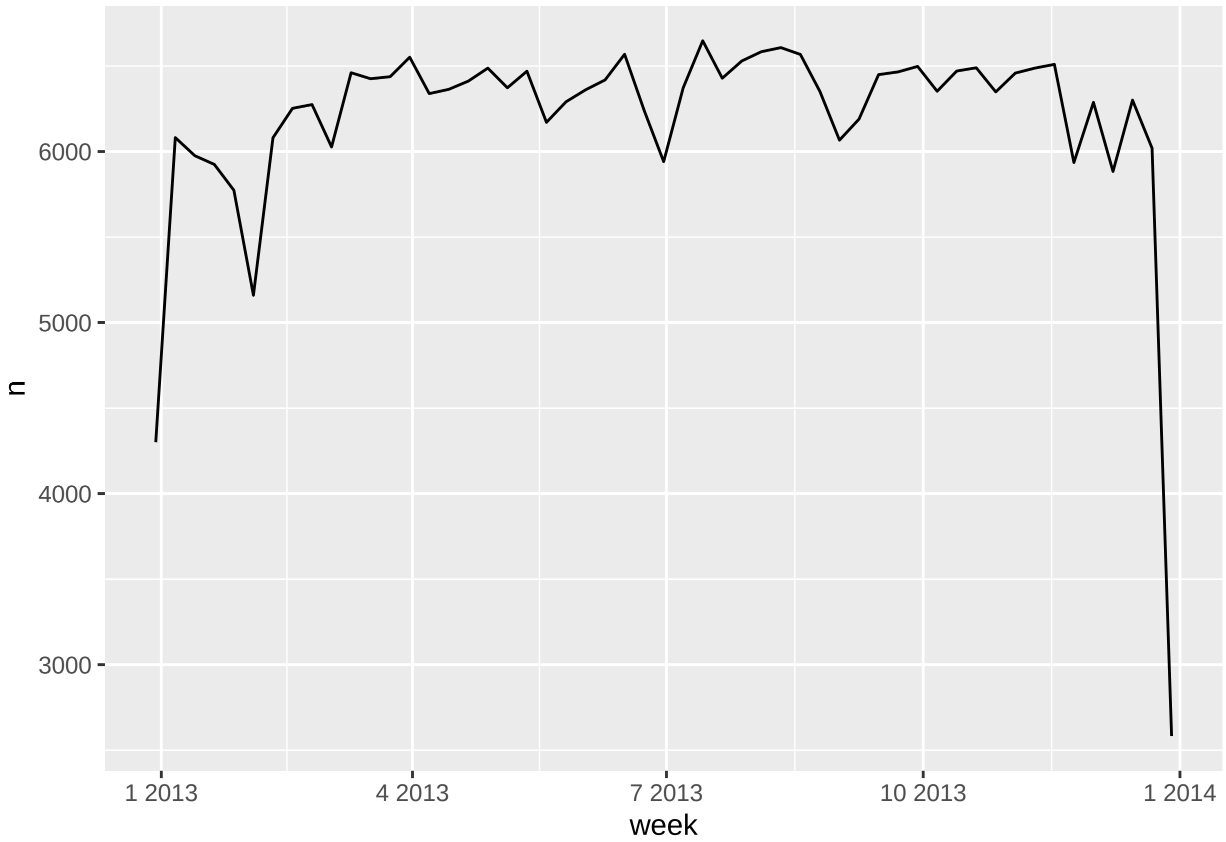
```
#예정 출발 분에 따른 딜레이 평균

flights_dt %>%
    mutate(minute=minute(sched_dep_time)) %>%
    group_by(minute) %>%
    summarise(avg_delay=mean(dep_delay, na.rm=TRUE), n=n()) %>%
    ggplot(aes(minute,n)) +
    geom_line() +
    theme(aspect.ratio = 1/2)
```



```
#딜레이 된 수가 제각기 달라서 바로 앞의 그래프에서 패턴찾기가 어려움



#floor, round, ceiling은 각각 내림 반올림 올림으로
#floor_date는 날짜를 내림하는 함수로 어느단위로 버리고 올릴지 인수로 전달해줘야한다


flights_dt
```

```
## # A tibble: 328,063 x 9
##    origin dest  dep_delay arr_delay dep_time            sched_dep_time
##    <chr>  <chr>     <dbl>     <dbl> <dttm>              <dttm>
##  1 EWR    IAH           2        11 2013-01-01 05:17:00 2013-01-01 05:15:00
##  2 LGA    IAH           4        20 2013-01-01 05:33:00 2013-01-01 05:29:00
##  3 JFK    MIA           2        33 2013-01-01 05:42:00 2013-01-01 05:40:00
##  4 JFK    BQN          -1       -18 2013-01-01 05:44:00 2013-01-01 05:45:00
##  5 LGA    ATL          -6       -25 2013-01-01 05:54:00 2013-01-01 06:00:00
##  6 EWR    ORD          -4        12 2013-01-01 05:54:00 2013-01-01 05:58:00
##  7 EWR    FLL          -5        19 2013-01-01 05:55:00 2013-01-01 06:00:00
##  8 LGA    IAD          -3       -14 2013-01-01 05:57:00 2013-01-01 06:00:00
##  9 JFK    MCO          -3        -8 2013-01-01 05:57:00 2013-01-01 06:00:00
## 10 LGA    ORD          -2         8 2013-01-01 05:58:00 2013-01-01 06:00:00
## # i 328,053 more rows
## # i 3 more variables: arr_time <dttm>, sched_arr_time <dttm>, air_time <dbl>
```

```
flights_dt %>%
    count(week=floor_date(dep_time,"week")) %>%
    ggplot(aes(week,n))+
    geom_line()
```



```
#flights_dt %>%
#   ggplot(aes(dep_time))+
#   geom_freqpoly(binwidth=86400*7)
#그냥 한번 해본 freqpoly 사용 이제 알겠다

datetime=today()
year(datetime) = 2020
datetime #오늘이 2020이 되는 마법, 할당이 쉽다
```

```
## [1] "2020-09-02"
```

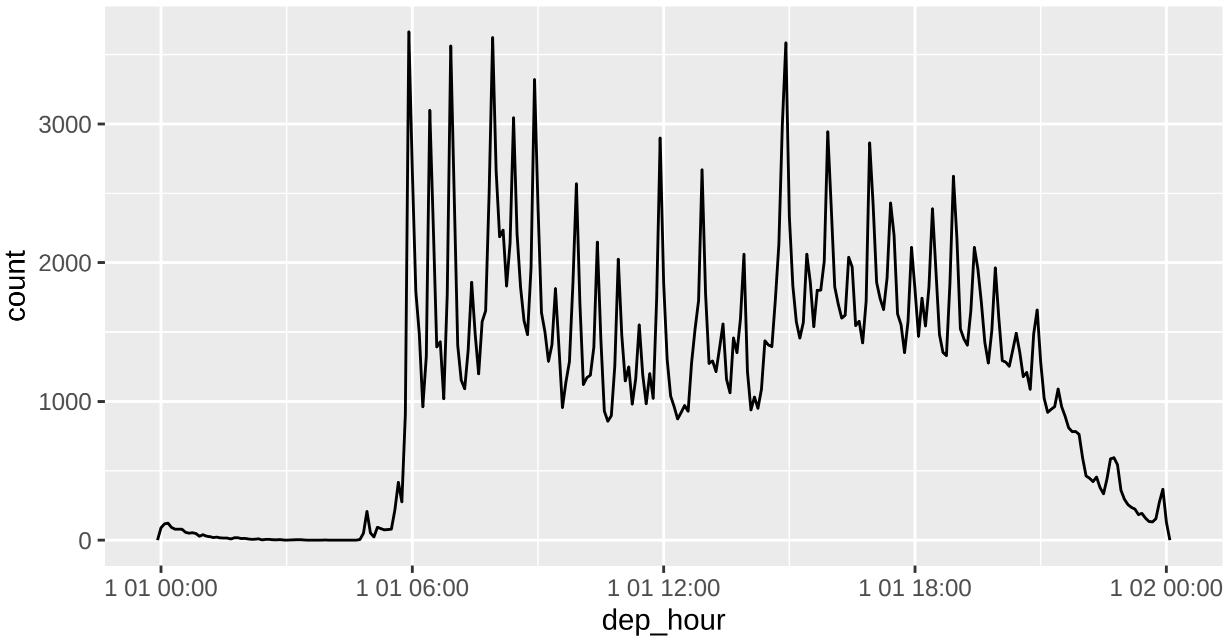
```
update(datetime, year= 2021, month=7,mday=4,hour=18)
```

```
## [1] "2021-07-04 18:00:00 UTC"
```

```
#날짜 막 바꾸기

ymd("2024/02/28") %>% update(mday=40) #자동 이월 ㄹㅇ굿
```

```
## [1] "2024-03-11"
```

```
flights_dt %>%
    mutate(dep_hour = update(dep_time, yday=1)) %>% #날짜를 다 뭉개버리기 하루로, 남는건 시간
    ggplot(aes(dep_hour)) +
    geom_freqpoly(binwidth=300) + #5분단위
    theme(aspect.ratio = 1/2)
```



```
#연월일을 상수로 뭉개서 시간데이터만 다룰 수 있다
```

시간에 대한 연산을 해봅시다

```
h_age = today()-ymd(19791014)
h_age
```

```
## Time difference of 16395 days
```

```r
#date형의 뺄셈은 잘 정의되어있다, 날짜 반환
as.duration(h_age) #초로 기간을 써줌
```

```
## [1] "1416528000s (~44.89 years)"
```

```r
dyears(1) #연을 초로 변환
```

```
## [1] "31557600s (~1 years)"
```

```r
ddays(1) #일을 초로 변환
```

```
## [1] "86400s (~1 days)"
```

```r
tomorrow = today()+ddays(1) #내일

one_pm=ymd_hms("20160312 13:00:00",tz="America/New_York")
one_pm + ddays(1)
```

```
## [1] "2016-03-13 14:00:00 EDT"
```

```r
#사실 하루는 24시간이 아니다....

#이상해요잉 periods를 써야한대

one_pm + days(1)
```

```
## [1] "2016-03-13 13:00:00 EDT"
```

```r
days(1) #하루단위
```

```
## [1] "1d 0H 0M 0S"
```

```r
#비행시간에 대한 계산
flights_dt %>%
    filter(arr_time < dep_time) %>%
    select(origin, dest, dep_time, arr_time)
```

```
## # A tibble: 10,633 x 4
##    origin dest  dep_time            arr_time
##    <chr>  <chr> <dttm>              <dttm>
##  1 EWR    BQN   2013-01-01 19:29:00 2013-01-01 00:03:00
##  2 JFK    DFW   2013-01-01 19:39:00 2013-01-01 00:29:00
##  3 EWR    TPA   2013-01-01 20:58:00 2013-01-01 00:08:00
##  4 EWR    SJU   2013-01-01 21:02:00 2013-01-01 01:46:00
##  5 EWR    SFO   2013-01-01 21:08:00 2013-01-01 00:25:00
##  6 LGA    FLL   2013-01-01 21:20:00 2013-01-01 00:16:00
##  7 EWR    MCO   2013-01-01 21:21:00 2013-01-01 00:06:00
##  8 JFK    LAX   2013-01-01 21:28:00 2013-01-01 00:26:00
##  9 EWR    FLL   2013-01-01 21:34:00 2013-01-01 00:20:00
## 10 EWR    FLL   2013-01-01 21:36:00 2013-01-01 00:25:00
## # i 10,623 more rows
```

```
#얘들은 overnight flights 이다
#수정해주자

flights_dt = flights_dt %>%
    mutate(
        overnight = arr_time < dep_time,
        arr_time = arr_time + days(overnight * 1),
        sched_arr_time = sched_arr_time + days(overnight * 1)
    )

flights_dt %>%
    filter(arr_time < dep_time) %>%
    select(origin, dest, dep_time, arr_time)
```

```
## # A tibble: 0 x 4
## # i 4 variables: origin <chr>, dest <chr>, dep_time <dttm>, arr_time <dttm>
```

```
#이제 없다

dyears(1) / ddays(365) #1년이 365일일까?
```

```
## [1] 1.000685
```

```
years(1) / days(1) #과연?
```

```
## [1] 365.25
```

```
next_year = today() + years(1)
(today() %--% next_year) / ddays(1) #인터벌형 꽤 복잡
```

```
## [1] 365
```

마지막으로 timezone에대해 다루자

```
Sys.timezone()
```

```
## [1] "Asia/Seoul"
```

```r
x1 <- ymd_hms("2015-06-01 12:00:00", tz = "America/New_York")
x2 <- ymd_hms("2015-06-01 18:00:00", tz = "Europe/Copenhagen")
x3 <- ymd_hms("2015-06-02 04:00:00", tz = "Pacific/Auckland")


x1
```

```
## [1] "2015-06-01 12:00:00 EDT"
```

```r
x2
```

```
## [1] "2015-06-01 18:00:00 CEST"
```

```r
x3
```

```
## [1] "2015-06-02 04:00:00 NZST"
```

```r
x1-x2 #이러함
```

```
## Time difference of 0 secs
```

```r
x4=ymd_hms(now())
x4
```

```
## [1] "2024-09-02 03:02:16 UTC"
```

```r
x4a=with_tz(x4, tzone="Asia/Shanghai")
x4a
```

```
## [1] "2024-09-02 11:02:16 CST"
```

```r
x4a-x4 #with_tz 단순히 시간존만 바꿈
```

```
## Time difference of 0 secs
```

```r
x4b=force_tz(x4, tzone="Asia/Shanghai")
x4b-x4 #시차를 고려해줌
```

```
## Time difference of -8 hours
```