CHAPTER 13. Dates and Times with lubridate

Prerequisites

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.4.4
```

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

```
## √ ggplot2 3.0.0     √ purrr   0.2.5
## √ tibble  1.4.2     √ dplyr   0.7.6
## √ tidyr   0.8.1     √ stringr 1.3.1
## √ readr   1.1.1     √ forcats 0.3.0
```

```
## Warning: package 'ggplot2' was built under R version 3.4.4
```

```
## Warning: package 'tibble' was built under R version 3.4.4
```

```
## Warning: package 'tidyr' was built under R version 3.4.4
```

```
## Warning: package 'readr' was built under R version 3.4.4
```

```
## Warning: package 'purrr' was built under R version 3.4.4
```

```
## Warning: package 'dplyr' was built under R version 3.4.4
```

```
## Warning: package 'stringr' was built under R version 3.4.4
```

```
## Warning: package 'forcats' was built under R version 3.4.4
```

```
## -- Conflicts ------------------------------------ tidyverse_conflicts() -
-
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(lubridate)
```

```
## Warning: package 'lubridate' was built under R version 3.4.4
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

```
library(nycflights13)
```

```
## Warning: package 'nycflights13' was built under R version 3.4.4
```

Creating Date/Times

날짜를 다루는 데이터 종류 3 가지 - date : 날짜 - time : day 를 포함 - date-time : time 을 포함한 date. 기본 R 에서는 POSIXct 타임.

현재 날짜와 시간을 얻는 방법

```
# not show Warning Message
options(warn=0)

today()

## [1] "2019-03-22"

now()

## [1] "2019-03-22 11:43:38 KST"
```

날짜와 시간을 생성하는 3 가지 다른 방법 - From a string. - From individual date-time components. - From an existing date/time object.

From Strings

ubridate 함수안에는 날짜/시간 관련 문자열을 파싱해줌. 함수 이름에 y(년), m(월), d(일) 순서를 변경해서 다양한 문자열을 파싱

```
ymd("2017-01-31")

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

mdy("January 31st, 2017")

## [1] "2017-03-01"

dmy("31-Jan-2017")

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

하이픈(-)을 생략 가능

```
ymd("20170131")

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

시간은 h(시), m(분), s(초) 순서에 따라 파싱

```
ymd_hms("2017-01-31 20:11:59")
```

```
## [1] "2017-01-31 20:11:59 UTC"
```

```
mdy_hm("01/31/2017 08:01")
```

```
## [1] "2017-01-31 08:01:00 UTC"
```

time zone 지정

```
ymd(20170131, tz = "UTC")
```

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

From Individual Components

개별적인 년월일 요소로부터 날짜/시간 데이터 타임 생성

```
flights %>%
    select(year, month, day, hour, minute)
```

```
## # A tibble: 336,776 x 5
##     year month   day  hour minute
##    <int> <int> <int> <dbl>  <dbl>
##  1  2013     1     1     5     15
##  2  2013     1     1     5     29
##  3  2013     1     1     5     40
##  4  2013     1     1     5     45
##  5  2013     1     1     6      0
##  6  2013     1     1     5     58
##  7  2013     1     1     6      0
##  8  2013     1     1     6      0
##  9  2013     1     1     6      0
## 10  2013     1     1     6      0
## # ... with 336,766 more rows
```

make_date() 또는 make_datetime() 함수 사용해서 생성 가능

```
flights %>%
    select(year, month, day, hour, minute) %>%
    mutate(
      departure = make_datetime(year, month, day, hour, minute)
    )
```

```
## Warning: package 'bindrcpp' was built under R version 3.4.4
```

```
## # A tibble: 336,776 x 6
##     year month   day  hour minute departure
##    <int> <int> <int> <dbl>  <dbl> <dttm>
##  1  2013     1     1     5     15 2013-01-01 05:15:00
##  2  2013     1     1     5     29 2013-01-01 05:29:00
##  3  2013     1     1     5     40 2013-01-01 05:40:00
```

```
##  4  2013     1     1     5     45 2013-01-01 05:45:00
##  5  2013     1     1     6      0 2013-01-01 06:00:00
##  6  2013     1     1     5     58 2013-01-01 05:58:00
##  7  2013     1     1     6      0 2013-01-01 06:00:00
##  8  2013     1     1     6      0 2013-01-01 06:00:00
##  9  2013     1     1     6      0 2013-01-01 06:00:00
## 10  2013     1     1     6      0 2013-01-01 06:00:00
## # ... with 336,766 more rows
```

flights 데이터안에 dep_time 와 arr_time 이 이상한 형식으로 표현되어 있어서 나머지 연산자를 사용하여 시간 및 분을 계산함.

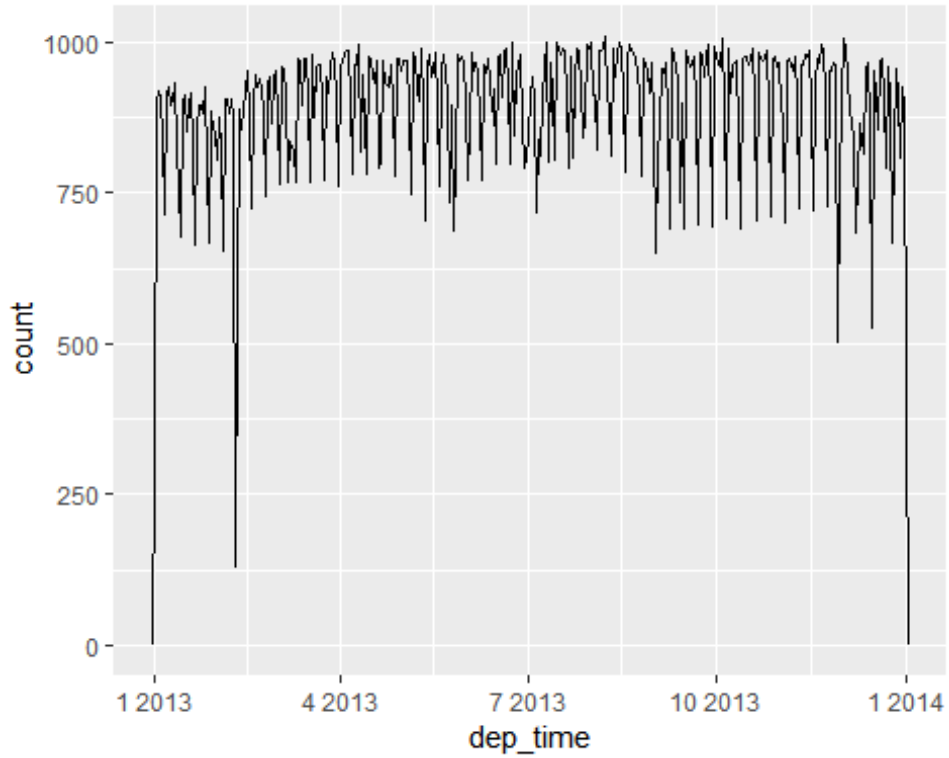
```
make_datetime_100 <- function(year, month, day, time) {
    make_datetime(year, month, day, time %/% 100, time %% 100)
}

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
```

```
## # 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 05:15:0
0
##  2 LGA    IAH           4        20 2013-01-01 05:33:00 2013-01-01 05:29:0
0
##  3 JFK    MIA           2        33 2013-01-01 05:42:00 2013-01-01 05:40:0
0
##  4 JFK    BQN          -1       -18 2013-01-01 05:44:00 2013-01-01 05:45:0
0
##  5 LGA    ATL          -6       -25 2013-01-01 05:54:00 2013-01-01 06:00:0
0
##  6 EWR    ORD          -4        12 2013-01-01 05:54:00 2013-01-01 05:58:0
```

```
0
##  7 EWR    FLL          -5        19 2013-01-01 05:55:00 2013-01-01 06:00:0
0
##  8 LGA    IAD          -3       -14 2013-01-01 05:57:00 2013-01-01 06:00:0
0
##  9 JFK    MCO          -3        -8 2013-01-01 05:57:00 2013-01-01 06:00:0
0
## 10 LGA    ORD          -2         8 2013-01-01 05:58:00 2013-01-01 06:00:0
0
## # ... with 328,053 more rows, and 3 more variables: arr_time <dttm>,
## #   sched_arr_time <dttm>, air_time <dbl>
```

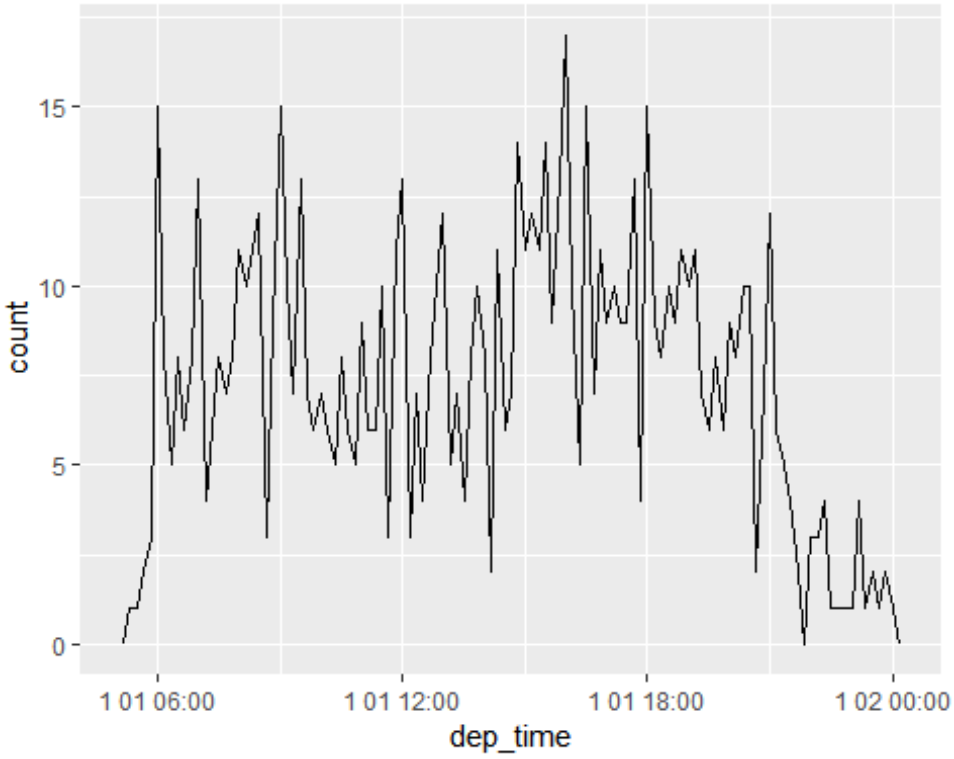
년도에 따른 도착 시간의 분포를 시각화

```
flights_dt %>%
    ggplot(aes(dep_time)) +
    geom_freqpoly(binwidth = 86400) # 86400 seconds = 1 day
```



하루동안의 도착시간의 분포

```
flights_dt %>%
    filter(dep_time < ymd(20130102)) %>%
    ggplot(aes(dep_time)) +
    geom_freqpoly(binwidth = 600) # 600 s = 10 minutes
```



From Other Types

date-time <-> date 는 as_datetime()함수와 as_date()함수를 사용해서 상호 변환

```
as_datetime(today())
## [1] "2019-03-22 UTC"
as_date(now())
## [1] "2019-03-22"
```

Unix 의 기준시간(1970-01-01)을 0 으로 계산하여 정수로 표현 가능

```
as_datetime(60 * 60 * 10)
## [1] "1970-01-01 10:00:00 UTC"
as_date(365 * 10 + 2)
## [1] "1980-01-01"
```

Date-Time Components

Getting Components

year(), month(), mday() (day of the month), yday() (day of the year), wday() (day of the week), hour(), minute(), second()

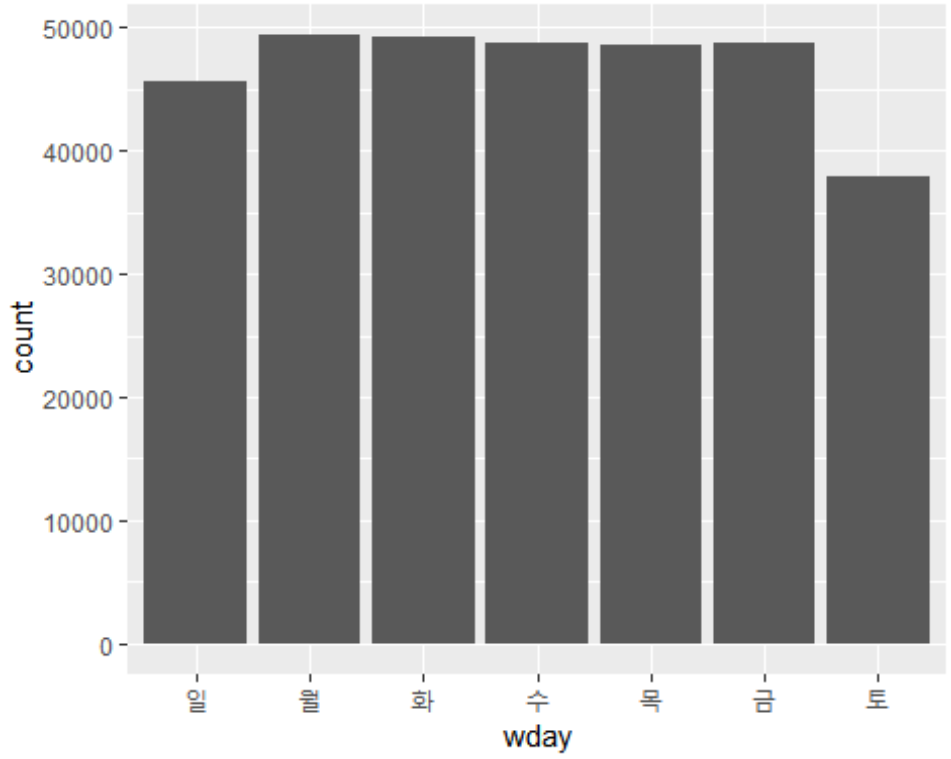
```
datetime <- ymd_hms("2016-07-08 12:34:56")

year(datetime)

## [1] 2016

month(datetime)

## [1] 7

mday(datetime)

## [1] 8

yday(datetime)

## [1] 190

wday(datetime)

## [1] 6
```

month() 와 wday()은 label = TRUE 설정해서 라벨로 변환

```
month(datetime, label = TRUE)

## [1] 7
## Levels: 1 < 2 < 3 < 4 < 5 < 6 < 7 < 8 < 9 < 10 < 11 < 12

wday(datetime, label = TRUE, abbr = FALSE)

## [1] 금요일
## 7 Levels: 일요일 < 월요일 < 화요일 < 수요일 < 목요일 < ... < 토요일
```

wday()을 사용해서 요일별 도착건수를 시각화

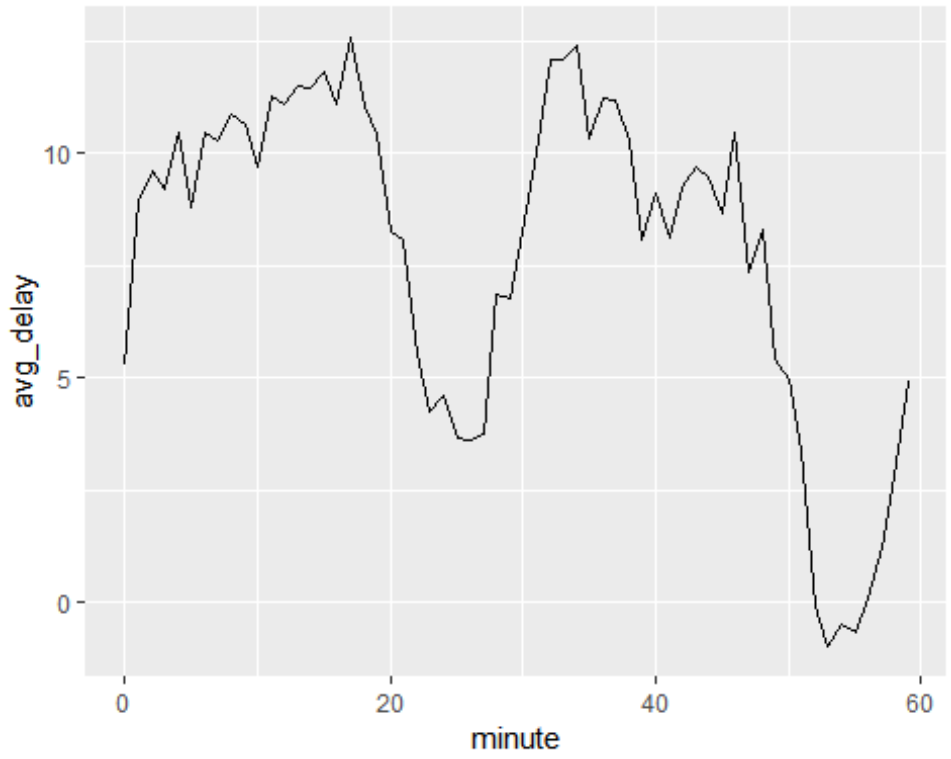
```
flights_dt %>%
   mutate(wday = wday(dep_time, label = TRUE)) %>%
   ggplot(aes(x = wday)) +
   geom_bar()
```



한 시간 동안 분단위로 도착지연 평균을 시각화

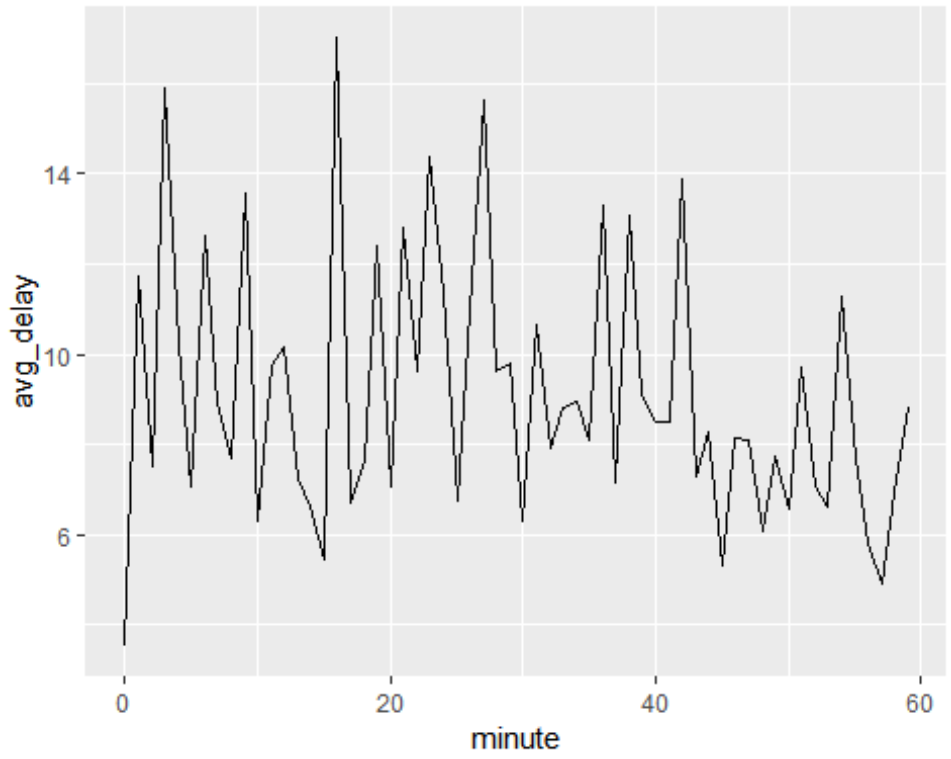
20-30 와 50-60 분 사이에 도착하는 여객기가 나머지 시간대의 여객기보다 지연이 많이 작음.

```
flights_dt %>%
   mutate(minute = minute(dep_time)) %>%
   group_by(minute) %>%
   summarize(
      avg_delay = mean(arr_delay, na.rm = TRUE),
      n = n()) %>%
   ggplot(aes(minute, avg_delay)) +
   geom_line()
```



scheduled 도착시간은 강한 패턴이 없음.

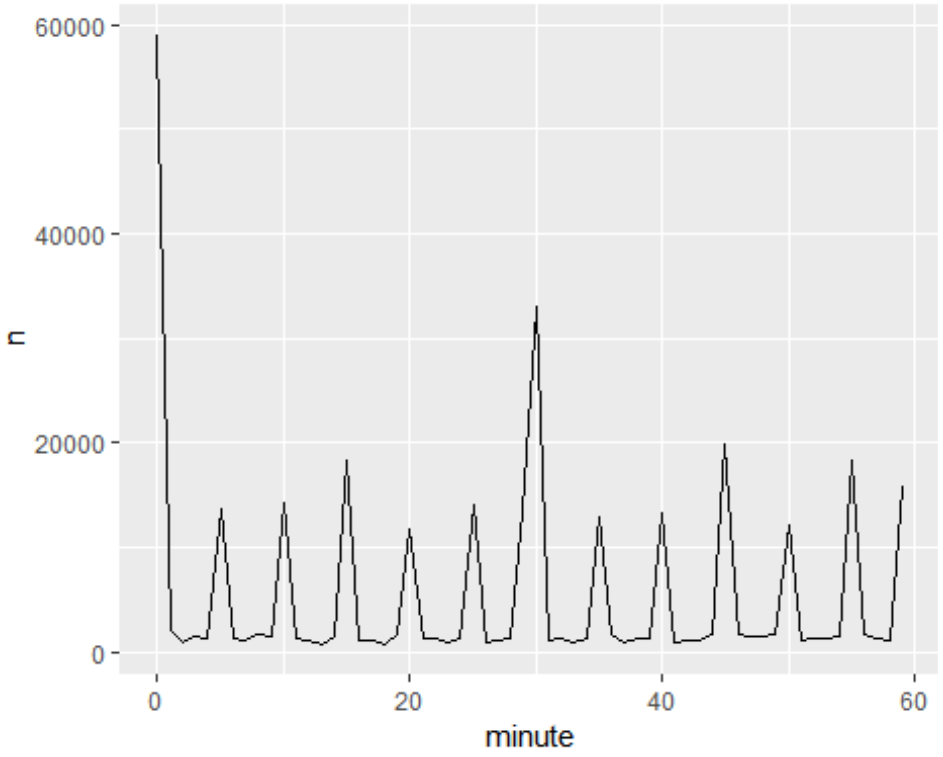
```
sched_dep <- flights_dt %>%
    mutate(minute = minute(sched_dep_time)) %>%
    group_by(minute) %>%
    summarize(
        avg_delay = mean(arr_delay, na.rm = TRUE),
        n = n())
    ggplot(sched_dep, aes(minute, avg_delay)) +
    geom_line()
```



why do we see that pattern with the actual departure times?

아래 그래프를 어떻게 해석이 가능할까요?

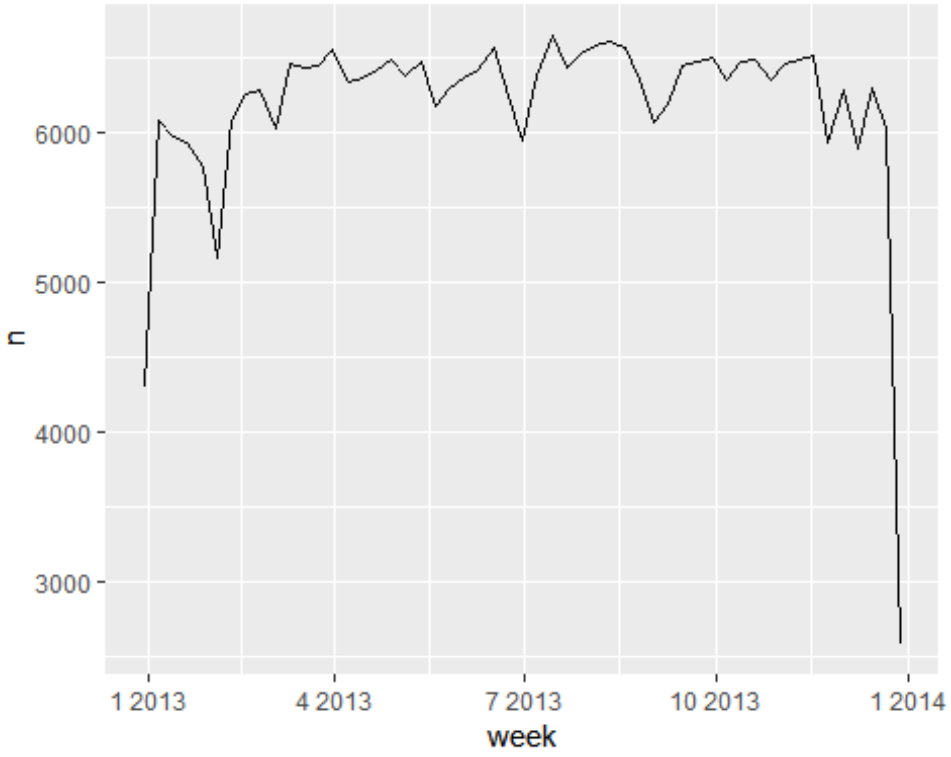
```
    ggplot(sched_dep, aes(minute, n)) +
    geom_line()
```



Rounding

날짜의 개별요소를 시각화하는 또 다른 방법은 날짜를 가까운 시간 단위로 반올림하는 것.
floor_date(), round_date(), ceiling_date()

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



Setting Components

```
(datetime <- ymd_hms("2016-07-08 12:34:56"))

## [1] "2016-07-08 12:34:56 UTC"

year(datetime) <- 2020
datetime

## [1] "2020-07-08 12:34:56 UTC"

month(datetime) <- 01
datetime

## [1] "2020-01-08 12:34:56 UTC"

hour(datetime) <- hour(datetime) + 1
```

update 함수로 한번에

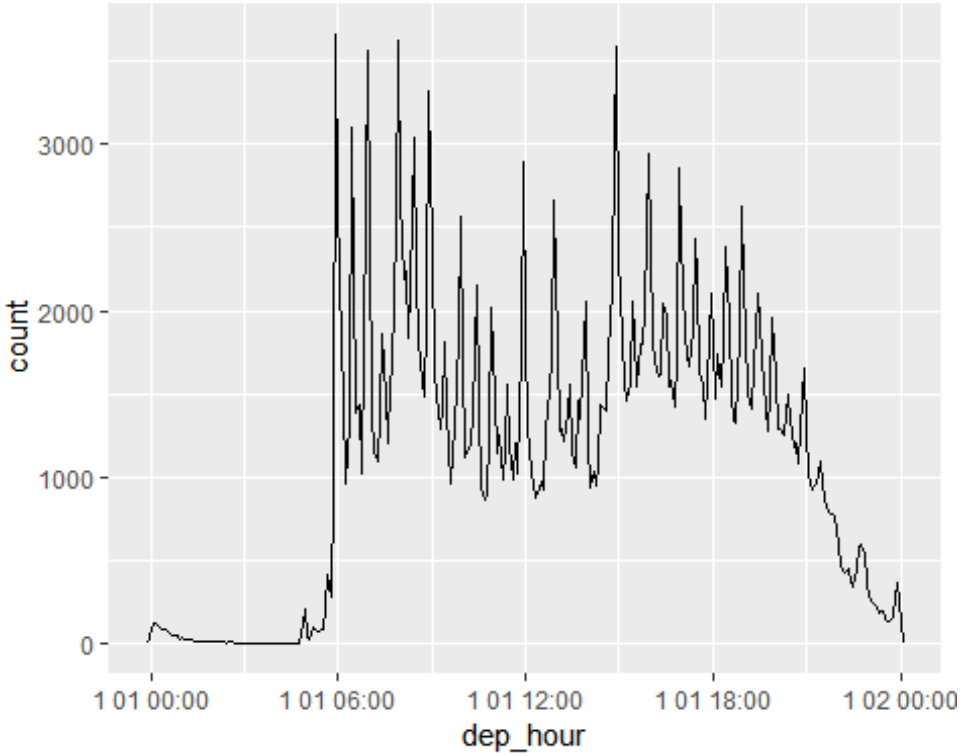
```
update(datetime, year = 2020, month = 2, mday = 2, hour = 2)

## [1] "2020-02-02 02:34:56 UTC"
```

단위보다 큰값을 설정하면, 상위 단위로 넘어감.

```r
ymd("2015-02-01") %>%
    update(mday = 30)

## [1] "2015-03-02"

ymd("2015-02-01") %>%
    update(hour = 400)

## [1] "2015-02-17 16:00:00 UTC"
```

날짜를 모든 1 월 1 일로 변경되어 도착시간을 매시간 마다 비행편의 분포를 시각

```r
flights_dt %>%
    mutate(dep_hour = update(dep_time, yday = 1)) %>%
    ggplot(aes(dep_hour)) +
    geom_freqpoly(binwidth = 300)
```



Time Spans

시간의 범위를 표현하는 3 가지 클래스

- Durations : 정확한 초 수를 나타냄.
- Periods : 주 단위와 달 단위 같은 인간 단위를 나타냄.
- Intervals : 시작과 끝 지점을 나타냄.

Durations

```
# How old is Hadley?
h_age <- today() - ymd(19791014)
h_age

## Time difference of 14404 days
```

lubridate 패키지에서는 duration 은 초 단위로 표

```
as.duration(h_age)

## [1] "1244505600s (~39.44 years)"
```

Durations 과 관련된 함수들

```
dseconds(15)

## [1] "15s"

dminutes(10)

## [1] "600s (~10 minutes)"

dhours(c(12, 24))

## [1] "43200s (~12 hours)" "86400s (~1 days)"

ddays(0:5)

## [1] "0s"                 "86400s (~1 days)"  "172800s (~2 days)"
## [4] "259200s (~3 days)" "345600s (~4 days)" "432000s (~5 days)"

dweeks(3)

## [1] "1814400s (~3 weeks)"

dyears(1)

## [1] "31536000s (~52.14 weeks)"
```

duration 을 더하기와 곱셈 가능

```
2 * dyears(1)

## [1] "63072000s (~2 years)"

dyears(1) + dweeks(12) + dhours(15)

## [1] "38847600s (~1.23 years)"
```

day 로부터 duration 을 더하고 뺄셈 가능.

```
tomorrow <- today() + ddays(1)
tomorrow
```

```
## [1] "2019-03-23"
```

```
last_year <- today() - dyears(1)
last_year
```

```
## [1] "2018-03-22"
```

durations 은 정확히 초 단위로 계산되기 때문에 항상 원하는 결과가 나오지 않음.

```
one_pm <- ymd_hms(
    "2016-03-12 13:00:00",
    tz = "America/New_York"
)

one_pm
```

```
## [1] "2016-03-12 13:00:00 EST"
```

```
one_pm + ddays(1)
```

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

Because of DST, March 12 only has 23 hours, so if we add a full day's worth of seconds we end up with a different

Periods

위의 문제를 해결하기 위해서, lubridate 패키지에서는 periods 을 제공. 사람이 사용하는 "하루 또는 달"을 사용해서 작업함.

```
one_pm
```

```
## [1] "2016-03-12 13:00:00 EST"
```

```
one_pm + ddays(1)
```

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

Periods 관련 함수들

```
seconds(15)
```

```
## [1] "15S"
```

```
minutes(10)
```

```
## [1] "10M 0S"
```

```
hours(c(12, 24))

## [1] "12H 0M 0S" "24H 0M 0S"

days(7)

## [1] "7d 0H 0M 0S"

months(1:6)

## [1] "1m 0d 0H 0M 0S" "2m 0d 0H 0M 0S" "3m 0d 0H 0M 0S" "4m 0d 0H 0M 0S"
## [5] "5m 0d 0H 0M 0S" "6m 0d 0H 0M 0S"

weeks(3)

## [1] "21d 0H 0M 0S"

years(1)

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

더하기와 곱셈

```
10 * (months(6) + days(1))

## [1] "60m 10d 0H 0M 0S"

days(50) + hours(25) + minutes(2)

## [1] "50d 25H 2M 0S"
```

윤년일때 durations 과 periods 비교

```
# A leap year
ymd("2016-01-01") + dyears(1)

## [1] "2016-12-31"

ymd("2016-01-01") + years(1)

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

# Daylight Savings Time
one_pm + ddays(1)

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

one_pm + days(1)

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

periods 은 사용해서 항공편 날짜와 관련된 관련된 괴상한 점을 수정해보자.

출발하기 이전에 도착하는 항공편이 나타냄.

```
flights_dt %>%
    filter(arr_time < dep_time)
```

```
## # A tibble: 10,633 x 9
##    origin dest  dep_delay arr_delay dep_time            sched_dep_time
##    <chr>  <chr>     <dbl>     <dbl> <dttm>              <dttm>
##  1 EWR    BQN           9        -4 2013-01-01 19:29:00 2013-01-01 19:20:0
0
##  2 JFK    DFW          59        NA 2013-01-01 19:39:00 2013-01-01 18:40:0
0
##  3 EWR    TPA          -2         9 2013-01-01 20:58:00 2013-01-01 21:00:0
0
##  4 EWR    SJU          -6       -12 2013-01-01 21:02:00 2013-01-01 21:08:0
0
##  5 EWR    SFO          11       -14 2013-01-01 21:08:00 2013-01-01 20:57:0
0
##  6 LGA    FLL         -10        -2 2013-01-01 21:20:00 2013-01-01 21:30:0
0
##  7 EWR    MCO          41        43 2013-01-01 21:21:00 2013-01-01 20:40:0
0
##  8 JFK    LAX          -7       -24 2013-01-01 21:28:00 2013-01-01 21:35:0
0
##  9 EWR    FLL          49        28 2013-01-01 21:34:00 2013-01-01 20:45:0
0
## 10 EWR    FLL          -9       -14 2013-01-01 21:36:00 2013-01-01 21:45:0
0
## # ... with 10,623 more rows, and 3 more variables: arr_time <dttm>,
## #   sched_arr_time <dttm>, air_time <dbl>
```

밤샘 비행으로, 밤샘 비행일때는 days(1)을 추가.

```
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(overnight, arr_time < dep_time)
```

```
## # A tibble: 0 x 10
## # ... with 10 variables: origin <chr>, dest <chr>, dep_delay <dbl>,
## #   arr_delay <dbl>, dep_time <dttm>, sched_dep_time <dttm>,
```

```
## #   arr_time <dttm>, sched_arr_time <dttm>, air_time <dbl>,
## #   overnight <lgl>
```

Intervals

초 단위이기 때문에 동일한 초 라서 정확 1.

```
dyears(1) / ddays(365)
```

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

Well, if the year was 2015 it should return 365, but if it was 2016, it should return 366!

```
years(1) / days(1)
```

```
## estimate only: convert to intervals for accuracy
```

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

보다 정확한 측정을 원하면 interval 을 사용해야합니다.interval 은 시작점이있는 기간입니다. 정확하기 때문에 정확히 얼마나 오래 있는지 확인할 수 있습니다.

```
next_year <- today() + years(1)
(today() %--% next_year) / ddays(1)
```

```
## [1] 366
```

정확성을 확인하기 위해서 정수의 나누기 연산자를 사용함.

```
(today() %--% next_year) %/% days(1)
```

```
## Note: method with signature 'Timespan#Timespan' chosen for function '%/%',
##  target signature 'Interval#Period'.
##  "Interval#ANY", "ANY#Period" would also be valid
```

```
## [1] 366
```

Summary

서로 다른 데이터 유형 간의 허용 된 산술 연산을 요약

	date				date time				duration				period				interval				number			
date	-								-	+			-	+							-	+		
date time					-				-	+			-	+							-	+		
duration	-	+			-	+			-	+		/									-	+	×	/
period	-	+			-	+							-	+							-	+	×	/
interval												/				/								
number	-	+			-	+			-	+	×		-	+	×		-	+	×		-	+	×	/

Time Zones

시스템 타임존 확인

```
Sys.timezone()
```

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

전체 타임존 확인

```
length(OlsonNames())
```

```
## [1] 592
```

```
head(OlsonNames())
```

```
## [1] "Africa/Abidjan"     "Africa/Accra"       "Africa/Addis_Ababa"
## [4] "Africa/Algiers"     "Africa/Asmara"      "Africa/Asmera"
```

```
(x1 <- ymd_hms("2015-06-01 12:00:00", tz = "America/New_York"))
```

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

```
(x2 <- ymd_hms("2015-06-01 18:00:00", tz = "Europe/Copenhagen"))
```

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

```
(x3 <- ymd_hms("2015-06-02 04:00:00", tz = "Pacific/Auckland"))
```

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

```
x1 - x2
```

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

```
x1 - x3
```

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

lubridate 에서는 타임존을 지정하지 않으면 ,UTC 를 항상 사용.

c()와 같은 date-times 값들을 묶을때는 time zone 을 삭제되고 local time zone 으로 표시됨.

```
x4 <- c(x1, x2, x3)
x4

## [1] "2015-06-02 01:00:00 KST" "2015-06-02 01:00:00 KST"
## [3] "2015-06-02 01:00:00 KST"
```

같은 시간값(기준시간)을 표현만 다르게 할때.

```
x4a <- with_tz(x4, tzone = "Australia/Lord_Howe")
x4a

## [1] "2015-06-02 02:30:00 +1030" "2015-06-02 02:30:00 +1030"
## [3] "2015-06-02 02:30:00 +1030"

x4a - x4

## Time differences in secs
## [1] 0 0 0
```

특정시간의 타임존을 변경할때

```
x4b <- force_tz(x4, tzone = "Australia/Lord_Howe")
x4b

## [1] "2015-06-02 01:00:00 +1030" "2015-06-02 01:00:00 +1030"
## [3] "2015-06-02 01:00:00 +1030"

x4b - x4

## Time differences in hours
## [1] -1.5 -1.5 -1.5
```