

rtweet パッケージのあれこレシピ

@anemptyarchive*

2019/11/21-2020/05/12

Contents

・ ツイート数のヒートマップを作成	1
・ ツイート収集	1
・ 年・月・日別に可視化	2
・ 時別に可視化	7

・ ツイート数のヒートマップを作成

rtweet パッケージを利用して、指定したアカウントのタイムラインを収集し、設定した期間 (年・月・日・時) ごとのツイート数を集計し、ヒートマップによる可視化を行います。

```
# 利用パッケージ
library(rtweet) # ツイート収集: get_timeline()
library(dplyr)  # データフレーム操作
library(tidyr)  # データフレーム操作: replace_na()
library(lubridate) # 時間データ操作: floor_date(), as_date(), now()
library(ggplot2) # 作図
```

利用するパッケージを読み込みます。

・ ツイート収集

まずは rtweet パッケージを利用して、ツイートを集めます。

```
# アカウントを指定
screen_name <- "anemptyarchive"

# アカウント指定でツイートを収集
tw_data <- get_timeline(screen_name, n = 10000, include_rts = FALSE)
```

get_timeline() にアカウント (@*** の ***) を指定して、ツイートを取得します。引数 n は収集するツイート数、include_rts は取得ツイートにリツイートを含めるかです (デフォルトは TRUE)。

*<https://www.anarchive-beta.com/>

特定の単語を含むツイートを収集するのであれば、`search_tweets("検索ワード")` を使います。

どちらも取得できるツイートの数に制限があったりします。

ツイート数の推移をみるにはツイート日時の情報があればいいので、取得したツイートデータのうち `created_at` 列のみを使います。それでは `created_at` 列のデータを確認してみましょう。

```
tw_data[["created_at"]][1:10]
```

```
## [1] "2020-05-10 17:23:28 UTC" "2020-05-10 15:26:52 UTC"
## [3] "2020-05-10 14:11:34 UTC" "2020-05-09 05:34:26 UTC"
## [5] "2020-05-08 10:57:19 UTC" "2020-05-08 08:01:25 UTC"
## [7] "2020-05-08 01:01:32 UTC" "2020-05-08 00:59:04 UTC"
## [9] "2020-05-07 13:48:48 UTC" "2020-05-07 08:07:00 UTC"
```

取得したツイート日時のタイムゾーンが協定世界時 (UTC) です。これを日本標準時 (JST) に変換にする必要があります。

ツイート日時データを変換

```
tw_time <- tw_data[["created_at"]] %>% # ツイート日時を抽出
  as.POSIXct(tz = "Etc/GMT") %>% # POSIXct 型の協定世界時を明示
  as.POSIXlt(tz = "Asia/Tokyo") # POSIXlt 型の日本標準時に変換
```

`as.POSIXlt()` で `POSIXct` 型から `POSIXlt` 型に変換します。また `tz` 引数に "Japan" あるいは "Asia/Tokyo" を指定して、タイムゾーンを日本標準時に変更します。(多分 2 行目はいらない)

変換後のデータを確認しておきましょう。

```
tw_time[1:10]
```

```
## [1] "2020-05-11 02:23:28 JST" "2020-05-11 00:26:52 JST"
## [3] "2020-05-10 23:11:34 JST" "2020-05-09 14:34:26 JST"
## [5] "2020-05-08 19:57:19 JST" "2020-05-08 17:01:25 JST"
## [7] "2020-05-08 10:01:32 JST" "2020-05-08 09:59:04 JST"
## [9] "2020-05-07 22:48:48 JST" "2020-05-07 17:07:00 JST"
```

これでツイートデータの取得と前処理は完了です。次からは可視化のための処理を行っていきます。

・年・月・日別に可視化

ここからは、指定した期間 (年・月・日) ごとにツイート数を集計し、ヒートマップを作図していきます。

セルの単位 (区切る期間) を指定

```
term <- "year"
term <- "mon"
term <- "day"
```

ヒートマップにする際のタイル (セル?) の単位 (期間) を指定します。「年」単位であれば `year`、「月」単位なら `mon`、「日」単位なら `day` とします。これは主に `floor_date()` の `unit` 引数に指定するためのものなので、このままの文字列を使用してください。

指定した期間ごとのツイートを数を集計します。

指定した期間ごとにツイート数を集計

```
tw_count1 <- tw_time %>%  
  floor_date(unit = term) %>% # 指定した期間に切り捨て  
  as_date() %>% # Date 型に変換  
  tibble(terms = .) %>% # データフレームに変換  
  count(terms) # ツイート数をカウント
```

`floor_date()` で指定した期間ごとに日時を丸めて (切り捨て) ます。
時刻情報も不要なので `as_date()` で POSIXlt 型から Date 型に変換することで落とします。
次に `tibble()` でデータフレームに変換して、`count()` で各期間のツイート数を集計します。

```
tail(tw_count1, 10)
```

```
## # A tibble: 10 x 2  
##   terms      n  
##   <date>   <int>  
## 1 2020-05-01     1  
## 2 2020-05-03     2  
## 3 2020-05-04     6  
## 4 2020-05-05     2  
## 5 2020-05-06     6  
## 6 2020-05-07     7  
## 7 2020-05-08     4  
## 8 2020-05-09     1  
## 9 2020-05-10     1  
## 10 2020-05-11     2
```

このままだと、ツイートがなかったタイミングが (ツイート数 0 ではなく) 抜け落ちてしまいます (5 月 2 日そのものがない)。その対処として、全ての期間が含まれるデータフレームを作成し、そこにツイート数の情報を移します。

ツイートがない期間が欠落する対策

```
term_df <- seq(  
  floor_date(tail(tw_time, 1), term), # 一番古い日時  
  floor_date(now(), term), # 現在日時  
  by = term  
) %>% # 指定した期間刻みのベクトルを作成  
  as_date() %>% # Date 型に変換  
  tibble(terms = .) # データフレームに変換
```

集計結果を結合

```
tw_count2 <- left_join(term_df, tw_count1, by = "terms") %>%  
  mutate(n = replace_na(n, 0)) # NA を 0 に置換
```

`seq()` で、一番古いツイート日時から現在の日時までのベクトルを作ります。このときベクトルの要素は、引数 `by` 指定した間隔となります。一番古いツイート日時は `tw_time` の最後の要素なので、`tail()` または `tw_time[length(tw_time)]` で取り出して渡します。現在時刻は `now()` で取得できます。

これを先ほどと同様に、日時データを Date 型に変換し、また全体をデータフレームに変換します。

これを受け皿として、`left_join()` で先ほどのデータフレームのツイート数と結合します。

このときツイートがなかった期間は NA となるので、それを `mutate()` と `replace_na()` を使って 0 に置換します。

集計結果を確認しましょう。

```
tail(tw_count2, 10)
```

```
## # A tibble: 10 x 2
##   terms      n
##   <date>    <dbl>
## 1 2020-05-03    2
## 2 2020-05-04    6
## 3 2020-05-05    2
## 4 2020-05-06    6
## 5 2020-05-07    7
## 6 2020-05-08    4
## 7 2020-05-09    1
## 8 2020-05-10    1
## 9 2020-05-11    2
## 10 2020-05-12    0
```

5月2日のツイート数が0という行が含まれていることが確認できます。

次はこのデータフレームを作図用に加工します。ただし、指定した期間によって処理が多少変わるため、`if()` で場合分けしています。

```
# 軸ラベル用にデータフレームを整形
if(term == "day") {

  tw_count2 <- tw_count2 %>%
    mutate(year_mon = format(terms, "%Y-%m")) %>% # 年月を抽出
    mutate(day = format(terms, "%d")) # 日を抽出

} else if(term == "mon") {

  tw_count2 <- tw_count2 %>%
    mutate(year = format(terms, "%Y")) %>% # 年を抽出
    mutate(mon = format(terms, "%m")) # 月を抽出

} else if(term == "year") {

  tw_count2 <- tw_count2 %>%
    mutate(year = format(terms, "%Y")) # 年を抽出

}
```

指定した期間を y 軸ラベル、それより大きな単位を x 軸目盛とするために、それぞれ `format()` でツイート日の列 (`terms`) から必要な情報を取り出し、`mutate()` で新しい列とします。

作図用のデータフレームができたので、これを用いて作図します。作図の処理も指定した期間によって異なるため、場合分けします。

```
# ヒートマップを作図
if(term == "day") {

  ggplot(tw_count2, aes(x = year_mon, y = day, fill = n)) +
    geom_tile() + # ヒートマップ
    scale_fill_gradient(low = "white", high = "#00A968") + # 塗りつぶしの濃淡
    theme(axis.text.x = element_text(angle = 90, hjust = 1)) + # 軸目盛の傾き

}
```

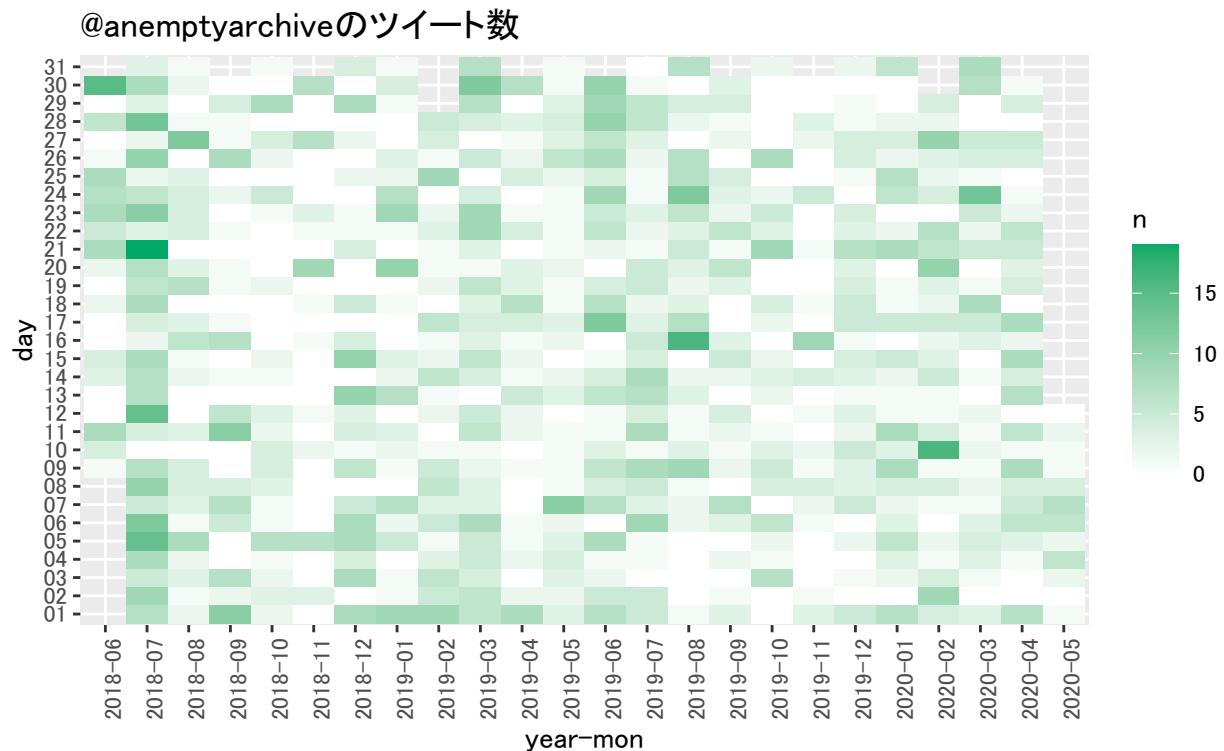
```

labs(title = paste0("@", screen_name, " のツイート数"),
     x = "year-mon", y = "day") # ラベル
} else if(term == "mon") {

  ggplot(tw_count2, aes(x = year, y = mon, fill = n)) +
    geom_tile() + # ヒートマップ
    scale_fill_gradient(low = "white", high = "#00A968") + # 塗りつぶしの濃淡
    labs(title = paste0("@", screen_name, " のツイート数"),
         x = "year", y = "mon") # ラベル
} else if(term == "year") {

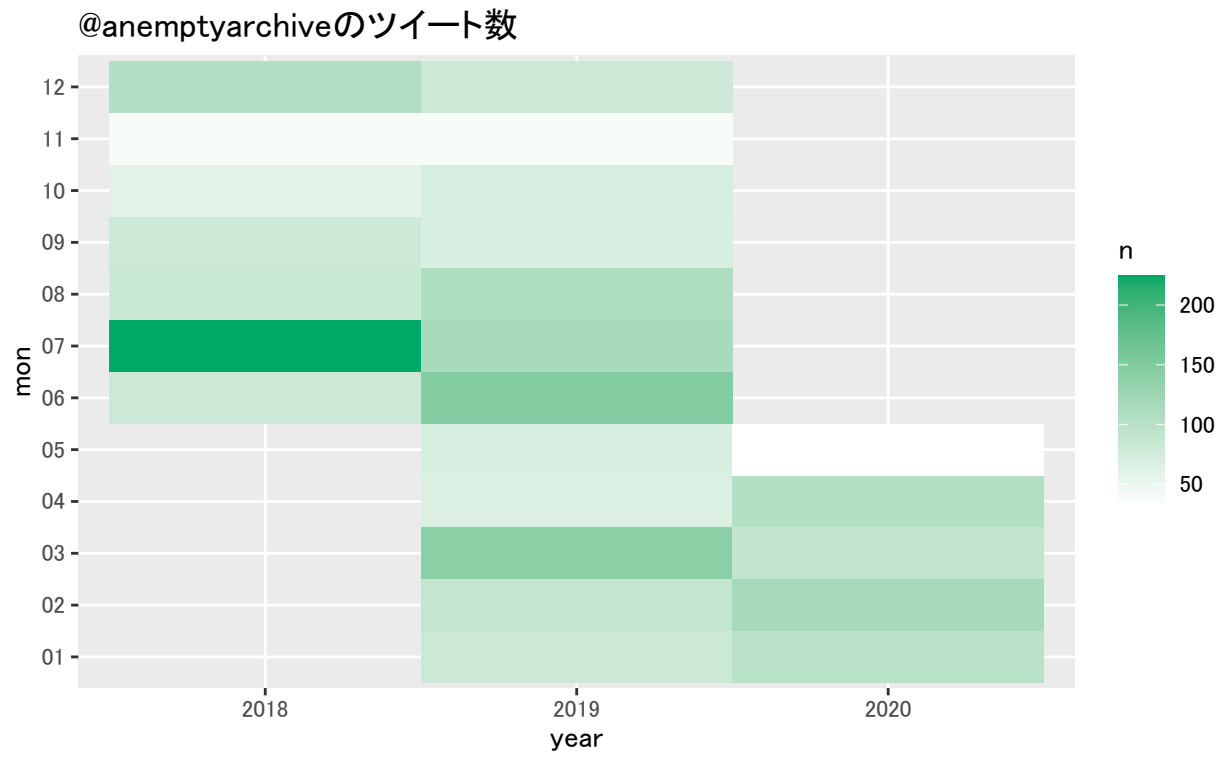
  ggplot(tw_count2, aes(x = year, y = 0, fill = n)) +
    geom_tile() + # ヒートマップ
    scale_fill_gradient(low = "white", high = "#00A968") + # 塗りつぶしの濃淡
    ylim(c(-1, 1)) + # y軸の範囲
    labs(title = paste0("@", screen_name, " のツイート数"),
         x = "year", y = "") # ラベル
}

```

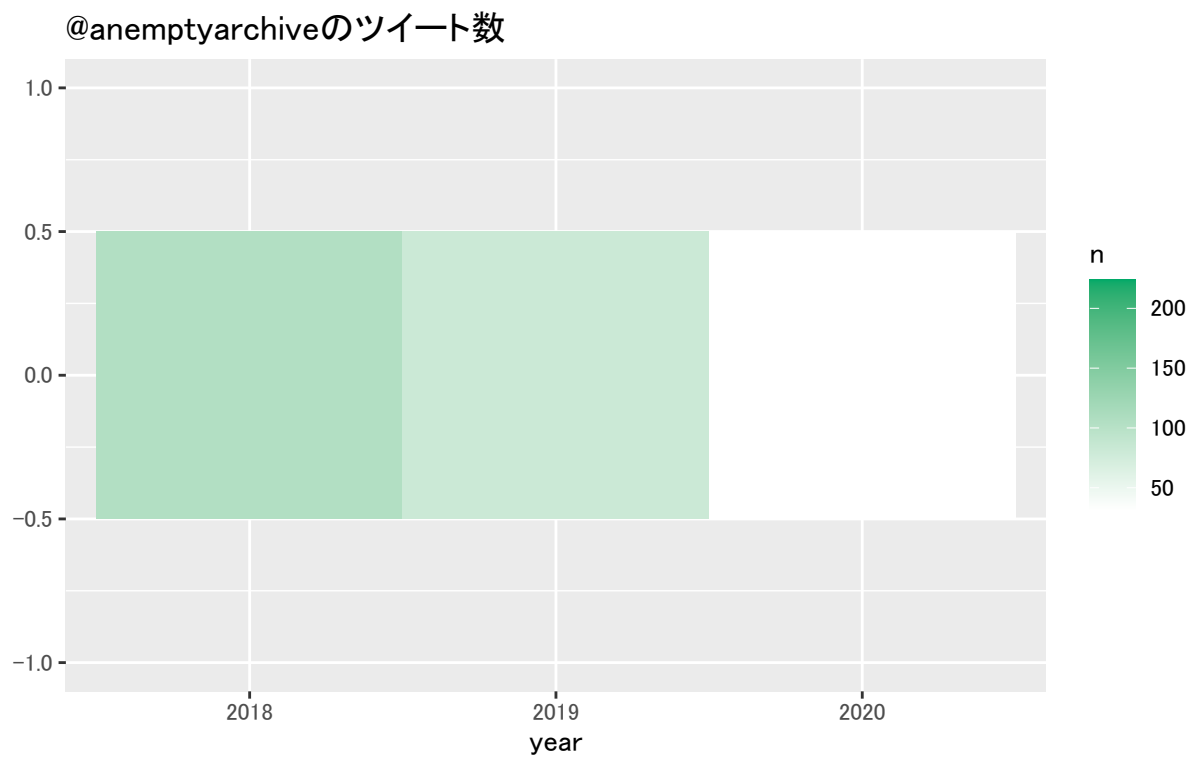


geom_tile() でヒートマップを作図します。
 タイルの色は scale_fill_gradient() の low と high にそれぞれ色を指定することで、ツイート数 (fill 引数の値) に応じたグラデーションとなります。

・ 月ごとの場合



・ 年ごとの場合



・ 時別に可視化

続いて、1時間ごとのツイート数をヒートマップ化します。
基本的な処理は同様です。こちらは可視化に時刻データを用いるため、ツイート日時を `POSIXct` 型で扱うところが異なります。

```
# 1時間ごとにツイート数を集計
tw_count1 <- tw_time %>%
  floor_date(unit = "hour") %>% # 1時間単位で切り捨て
  as.POSIXct() %>% # POSIXct 型に変換
  tibble(terms = .) %>% # データフレームに変換
  count(terms) # ツイート数をカウント

# ツイートがない期間が欠落する対策
term_df <- seq(
  floor_date(tw_time[length(tw_time)], "hour"), # 一番古い時刻
  floor_date(now(), "hour"), # 現在時刻
  by = "hour"
) %>% # 1時間刻みのベクトルを作成
  tibble(terms = .) # データフレームに変換

# 作図用にデータフレームを加工
tw_count2 <- left_join(term_df, tw_count1, by = "terms") %>% # 集計結果を結合
  mutate(n = replace_na(n, 0)) %>% # NA を 0 に置換
  mutate(year_mon_day = as_date(terms)) %>% # 年月日を抽出
  mutate(hour = format(terms, "%H")) # 時間を抽出
```

`floor_date(., unit = "hour")` で1時間単位で丸めて (切り捨て)、`as.POSIXct()` で `POSIXct` 型に変換します。

また `tibble()` でデータフレームに変換して、`count()` で各期間のツイート数を集計します。

期間内にツイートがなかった場合に欠落する問題の対処や、作図用のデータフレームの加工も概ね同じ流れです。

期間を指定する際に "hour" とする点に注意しましょう。

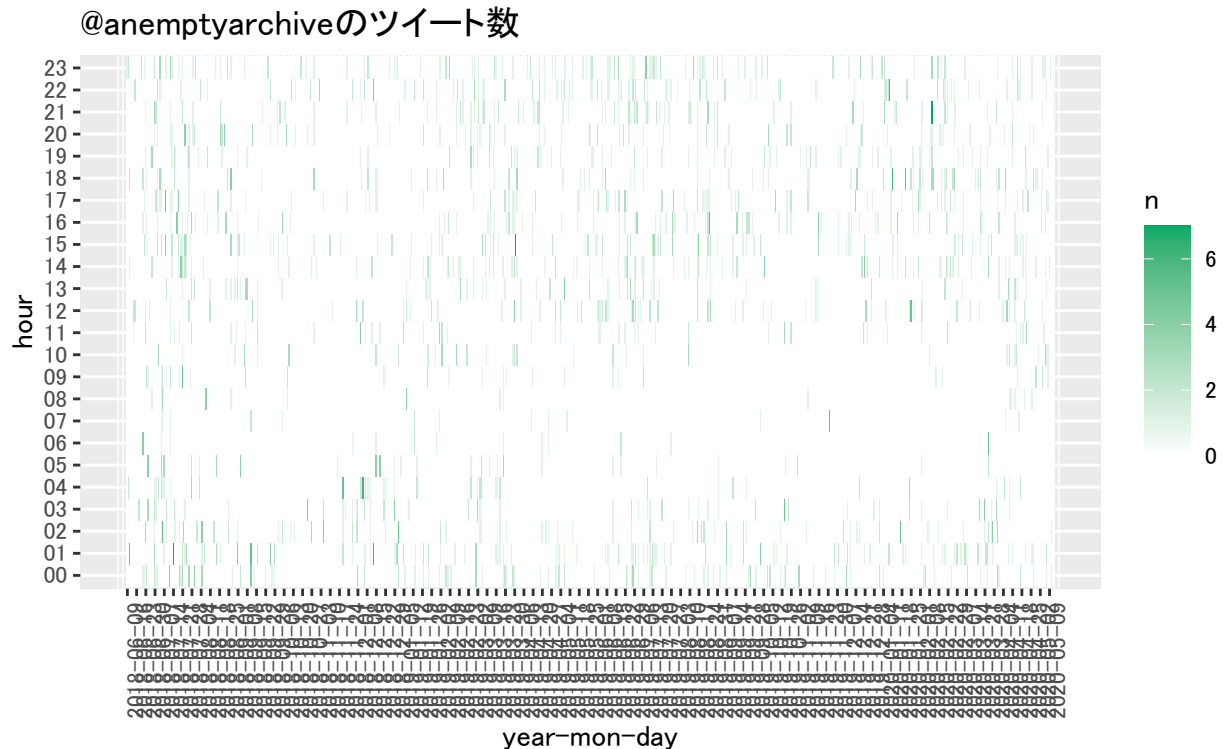
では、完成したデータフレームを確認しておきましょう。

```
tail(tw_count2, 10)

## # A tibble: 10 x 4
##   terms          n year_mon_day hour
##   <dtm>      <dbl> <date>    <chr>
## 1 2020-05-12 11:00:00    0 2020-05-12    11
## 2 2020-05-12 12:00:00    0 2020-05-12    12
## 3 2020-05-12 13:00:00    0 2020-05-12    13
## 4 2020-05-12 14:00:00    0 2020-05-12    14
## 5 2020-05-12 15:00:00    0 2020-05-12    15
## 6 2020-05-12 16:00:00    0 2020-05-12    16
## 7 2020-05-12 17:00:00    0 2020-05-12    17
## 8 2020-05-12 18:00:00    0 2020-05-12    18
## 9 2020-05-12 19:00:00    0 2020-05-12    19
## 10 2020-05-12 20:00:00    0 2020-05-12    20
```

これを用いて作図します。

```
# ヒートマップを作図
ggplot(tw_count2, aes(x = year_mon_day, y = hour, fill = n)) +
  geom_tile() + # ヒートマップ
  scale_fill_gradient(low = "white", high = "#00A968") + # 塗りつぶしの濃淡
  scale_x_date(breaks = seq(tw_count2[["year_mon_day"]][1],
                             tw_count2[["year_mon_day"]][nrow(tw_count2)],
                             by = "1 week")) + # x軸目盛 (日付)
  theme(axis.text.x = element_text(angle = 90)) + # x軸目盛の傾き
  labs(title = paste0("@", screen_name, " のツイート数"),
       x = "year-mon-day", y = "hour") # ラベル
```



x 軸に表示するデータ数が多くなるため、必要に応じて `scale_x_date()` で、x 軸ラベルを表示する位置を間引きます (ここに表示されるラベルデータは文字列型なので自動調整されない)。`scale_x_date()` の `breaks` 引数に、表示する位置のベクトルを渡すことで表示数を絞ることができます。そのベクトルは、`seq()` に `tw_count2` の `year_mon_day` 列の 1 行目と最終行の要素を渡して作成します。`by` 引数に指定する間隔を変更して調整してください。