Lubridate Vignette

Daniel Moscoe

4/6/2021

Introduction Lubridate contains functions that make it easier to work with dates and times. In this vignette we'll use Lubridate to assist with the following tasks:

(1) Create date/time objects from strings;
(2) Create date/time objects from individual components;
(3) Use accessors to get/set individual components of a date/time object;
(4) Use durations to perform arithmetic on date/times.

Lubridate is part of the Tidyverse ecosystem of packages. Because Lubridate is not one of the core packages of Tidyverse, it needs to be imported separately.

```
library(tidyverse)
library(lubridate)
```

About the data We'll work with a subset of the data from "NY Bus Breakdown and Delays", referred to here as `delays`. The data have been modified slightly to better suit the instructional purpose of this vignette.

`delays` contains information on 499 school bus delays during the school year beginning in 2018.

```
delays <- read_csv(
  url("https://raw.githubusercontent.com/dmoscoe/SPS/main/DATA607/delays.csv"),
  col_names = TRUE)
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_character(),
##   year = col_double(),
##   breakdown_id = col_double(),
##   occurred_on_yr = col_double(),
##   occurred_on_month = col_double(),
##   occurred_on_day = col_double(),
##   occurred_on_time = col_time(format = ""),
##   created_on = col_double(),
##   passengers = col_double(),
##   opt_notified_on = col_datetime(format = ""),
##   updated_on = col_datetime(format = ""),
##   breakdown = col_logical()
## )
## i Use `spec()` for the full column specifications.
```

```
head(delays)
```

```
## # A tibble: 6 x 22
##    year breakdown_id run_type        bus_no route reason          occurred_on_yr
##   <dbl>        <dbl> <chr>           <chr>  <chr> <chr>                    <dbl>
## 1  2018      1456303 Pre-K/EI        2630   ER    Heavy Traffic             2018
## 2  2018      1456346 Special Ed AM ~ 44291  K887  Mechanical Pro~           2018
## 3  2018      1471198 Special Ed AM ~ 804    K546  Accident                  2018
## 4  2018      1471203 General Ed AM ~ 479    R1097 Heavy Traffic             2018
## 5  2018      1471210 Special Ed AM ~ 1413   M676  Heavy Traffic             2018
## 6  2018      1471216 Special Ed AM ~ 41103  K301  Other                     2018
## # ... with 15 more variables: occurred_on_month <dbl>, occurred_on_day <dbl>,
## #   occurred_on_time <time>, created_on <dbl>, boro <chr>, company <chr>,
## #   delay_est <chr>, passengers <dbl>, schools_notified <chr>,
## #   parents_notified <chr>, opt_notified <chr>, opt_notified_on <dttm>,
## #   updated_on <dttm>, breakdown <lgl>, school_age <chr>
```

Create date/time objects from strings `delays` contains date/times in a variety of formats. The column `created_on` gives a date for the creation of the record in the table. The date is given in yyyymmdd format, and R interprets each entry as a number.

```
str(delays$created_on[1:5])
```

```
##  num [1:5] 20180904 20180904 20181015 20181015 20181015
```

Lubridate includes functions to parse date/times from a variety of formats. The arrangement of the date information in the string to be parsed helps determine the Lubridate function to use. In the case of the strings in `delays$created_on`, the year occurs first, followed by the month, then the day. We use the corresponding Lubridate function, `ymd()`.

```
delays$created_on <- ymd(delays$created_on)
delays$created_on[1:5]
```

```
## [1] "2018-09-04" "2018-09-04" "2018-10-15" "2018-10-15" "2018-10-15"
```

```
str(delays$created_on[1:5])
```

```
##  Date[1:5], format: "2018-09-04" "2018-09-04" "2018-10-15" "2018-10-15" "2018-10-15"
```

Lubridate includes analogous parsing functions for date/times given in different orders. It can even accept months given as words.

```
dmy('11 July 1999')
```

```
## [1] "1999-07-11"
```

Create date/time objects from individual components The columns `occurred_on_yr`, `occurred_on_month`, `occurred_on_day`, and `occurred_on_time` contain date/time information given as individual components.

```
delays %>%
  select(occurred_on_yr, occurred_on_month, occurred_on_day, occurred_on_time)
```

```
## # A tibble: 499 x 4
##    occurred_on_yr occurred_on_month occurred_on_day occurred_on_time
##             <dbl>             <dbl>           <dbl> <time>
##  1           2018                 9               4 07:13
##  2           2018                 9               4 07:40
##  3           2018                10              15 06:19
##  4           2018                10              15 06:35
##  5           2018                10              15 06:36
##  6           2018                10              15 06:30
##  7           2018                10              15 06:38
##  8           2018                10              15 06:40
##  9           2018                 9              25 07:18
## 10           2018                10              15 06:48
## # ... with 489 more rows
```

After parsing the times in `occurred_on_time`, We can use `make_date()` to create dates that combine the information in these several components.

```
delays$occurred_on_time <- delays$occurred_on_time %>%
  hms()

delays <- delays %>%
  mutate(occurred_on = make_datetime(
    year = occurred_on_yr,
    month = occurred_on_month,
    day = occurred_on_day,
    hour = hour(occurred_on_time),
    min = minute(occurred_on_time),
    tz = 'US/Eastern')
    )

delays$occurred_on[1:5]
```

```
## [1] "2018-09-04 07:13:00 EDT" "2018-09-04 07:40:00 EDT"
## [3] "2018-10-15 06:19:00 EDT" "2018-10-15 06:35:00 EDT"
## [5] "2018-10-15 06:36:00 EDT"
```

Note the keyword argument `tz` in `make_datetime()`. To view a list of the almost 600 valid time zones, use `OlsonNames()`.

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

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

Use accessors to get/set individual components of a date/time object In datasets containing times recorded by humans, it's common to see more "nice" times than expected. People have a tendency

to approximate times by rounding to the nearest ten or fifteen minutes. Is this the case in `delays`? Let's investigate the minute component of the times in `occurred_on` to find out.

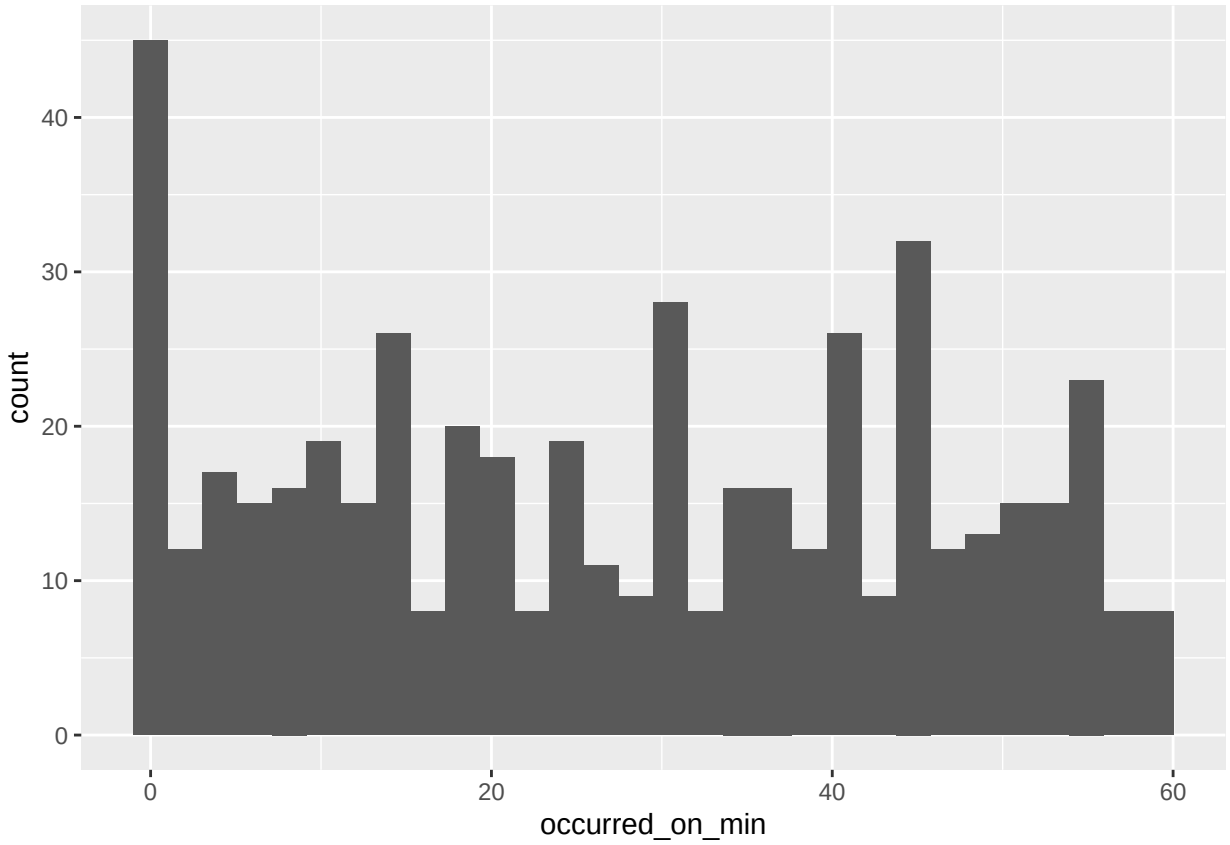
We use the accessor function `minute()` to create a list of the minutes components of times in `occurred_on`.

```
delays <- delays %>%
  mutate(occurred_on_min = minute(occurred_on))

delays$occurred_on_min[1:5]
```

```
## [1] 13 40 19 35 36
```

```
ggplot(data = delays, mapping = aes(x = occurred_on_min)) +
  geom_histogram(bins = 30)
```



The histogram shows that delays are more commonly reported to have occurred at times ending in :00, :15, :30, or :45.

We can also use accessors to set components of a date/time. The column `updated_on` contains many entries of 1/1/1900, which indicate missing data.

```
delays$updated_on[1:5]
```

```
## [1] "1900-01-01 UTC" "1900-01-01 UTC" "1900-01-01 UTC" "1900-01-01 UTC"
## [5] "1900-01-01 UTC"
```

For demonstration purposes, let's change the month component of these first five 1/1/1900 entries to August. In order to do this, we first change the type of these dates from the default type for date/times, `POSIXct`, to Tidyverse `dates`.

```
delays$updated_on <- date(delays$updated_on)
month(delays$updated_on[1:5]) <- 8
delays[1:10,c(1,2,3,20)]
```

```
## # A tibble: 10 x 4
##     year breakdown_id run_type          updated_on
##    <dbl>        <dbl> <chr>             <date>
##  1  2018      1456303 Pre-K/EI          1900-08-01
##  2  2018      1456346 Special Ed AM Run 1900-08-01
##  3  2018      1471198 Special Ed AM Run 1900-08-01
##  4  2018      1471203 General Ed AM Run 1900-08-01
##  5  2018      1471210 Special Ed AM Run 1900-08-01
##  6  2018      1471216 Special Ed AM Run 1900-01-01
##  7  2018      1471220 Special Ed AM Run 1900-01-01
##  8  2018      1471244 General Ed AM Run 1900-01-01
##  9  2018      1463347 Pre-K/EI          1900-01-01
## 10  2018      1471268 Special Ed AM Run 1900-01-01
```

Lubridate includes accessor functions for every component of a date/time.

Use durations to perform arithmetic on date/times A duration is an amount of time in seconds. Durations help us perform arithmetic on date/times. Constructors for durations are as expected: `dseconds()`, `dminutes()`, `dhours()`, `ddays()`, `dweeks()`, and `dyears()`. Each constructor takes a value in the units indicated by the constructur, and returns the number of seconds.

```
a <- dseconds(45)
a
```

```
## [1] "45s"
```

```
b <- dhours(3)
b
```

```
## [1] "10800s (~3 hours)"
```

```
c <- dweeks(2)
c
```

```
## [1] "1209600s (~2 weeks)"
```

We can use durations to combine lengths of time in different units. How long is 10 weeks, a month, and 19 days?

```
dweeks(10) + dmonths(1) + ddays(19)
```

```
## [1] "10319400s (~17.06 weeks)"
```

Summary Lubridate contains functions that make it easier to work with dates and times. Working with dates and times is essential but, at times, surprisingly complex, because calendars and time zones contain many inconsistencies. Some fundamental tasks when working with date/times include those described in this vignette: creating date/time objects from strings and from individual components; using accessors to get and set components of date/times; and performing computations using durations. Lubridate contains other functionality for working with time zones and for dealing with time spans under different assumptions.

Valuable references drawn on for this vignette include Chapter 13 of *R for Data Science* by Hadley Wickham, and the Lubridate cheat sheet.