cleaning_dates

October 19, 2020

Cleaning Dates in R

1 1. The *lubridate* Package

- Extremely powerful R package for working with dates and timestamps
- Part of the *tidyverse* family of packages (e.g., *dplyr*, *ggplot*, *stringr*)

```
In [2]: # load pacakges
        # read in data

        library(dplyr)
        library(lubridate)

        arrests_df <- read.csv("./data/pvd_arrests_2020-10-03.csv")
```

1.1 1.1 Working with Timestamps

- The *lubridate* package has many built-in functions for timestamp data
- Also often easily recognizes when a string *is* a timestamp

```
In [3]: ts <- "2020-10-11 02:30:59"      # ISO 8601 format

        year(ts)
```

2020

```
In [4]: month(ts)
```

10

```
In [6]: day(ts)
```

11

1.1.1 1.1.1 Extracting Time

```
In [7]: ts <- "2020-10-11 02:30:59"

        hour(ts)
        minute(ts)
        second(ts)
```

2
30
59

```
In [10]: am(ts)              # is it AM time (i.e., morning)?

         dst(ts)
```

TRUE
TRUE

1.1.2 1.1.2 Extracting Day-of-Week

```
In [11]: ts <- "2020-10-11 02:30:59"

         wday(ts)
```

1

```
In [13]: toString(wday(ts, label = TRUE))
```

'Sun'

1.2 1.2 Other Timestamp Formats

```
In [14]: ts2 <- "2020-10-11"

         toString(wday(ts2, label = TRUE))
```

'Sun'

```
In [21]: ts3 <- as_datetime("20201011")

         toString(wday(ts3, label = TRUE))
```

'Sun'

1.2.1 1.2.1 Non ISO 8601 Format

- We can also tell *lubridate* package how to parse non-obvious timestamps

```
In [23]: ts3 <- "October 11, 2020"

         month(ts3)


        Error in as.POSIXlt.character(x, tz = tz(x)): character string is not in a standard una
    Traceback:


        1. month(ts3)

        2. month.default(ts3)

        3. month(as.POSIXlt(x, tz = tz(x))$mon + 1, label, abbr, locale = locale)

        4. as.POSIXlt(x, tz = tz(x))

        5. as.POSIXlt.character(x, tz = tz(x))

        6. stop("character string is not in a standard unambiguous format")
```

```
In [25]: mdy(ts3)              # Month-day-year format (also dmy(), ymd(), and others)
```

2020-10-11

```
In [26]: month(mdy(ts3))
```

10

2 2. Math with Timestamps

- The *lubridate* pacakge also makes it easy to do math with dates and times

```
In [29]: time1 <- as_datetime("2020-10-11 03:45:52")
         time2 <- as_datetime("2020-10-13 23:41:09")

         time2 - time1
```

```
Time difference of 2.830058 days
```

2.1 2.1 Date/Time Intervals

```
In [30]: time1 <- as_datetime("2020-10-12")
         time2 <- as_datetime("2020-10-15")


         dt_intr <- interval(time1, time2)
```

```
In [37]: as_datetime("2020-10-13") %within% dt_intr
```

TRUE

```
In [38]: now() %within% dt_intr
```

FALSE

3 3. Arrests by Day-of-Week

- Suppose we want to explore the number of arrests by the day of the week

3.1 3.1 Create `day_of_week()` Function

```
In [39]: day_of_week <- function(timestamps) {

             n <- length(timestamps)  # get length of input column
             day <- rep("", n)        # allocate vector for day of week

             # iterate over elements of input column and return
             # the day of the week for each timestamp

             for (i in 1:n) {
                 day[i] <- toString(wday(timestamps[i], label = TRUE))
             }
             return(day)
         }
```

3.1.1 3.1.1 Creating `weekday` Column

- Now we can use our newly created `day_of_week()` function to add a new column

```
In [40]: # use out `day_of_week()` function to create new column
         # in our original dataframe

         arrests_df$weekday <- day_of_week(arrests_df$arrest_date)
```

```
In [42]: # use head() to examine updated dataframe

         head(arrests_df)
```

A data.frame: 6 Œ 19

	arrest_date <chr>	year <int>	month <int>	gender <chr>	race <chr>	ethnicity <chr>	year_of <int>
1	2019-08-24T02:23:00.0	2019	8	Male	White	NonHispanic	1981
2	2019-08-24T02:02:00.0	2019	8				1994
3	2019-08-24T02:02:00.0	2019	8	Female	Black	NonHispanic	1984
4	2019-08-24T02:02:00.0	2019	8	Female	Black	NonHispanic	1984
5	2019-08-24T02:02:00.0	2019	8	Female	Black	Unknown	2001
6	2019-08-24T02:02:00.0	2019	8	Female	Black	Unknown	2001

3.1.2 3.1.2 Counts by `weekday`

We can now obtain thee counts by day of the week using the `table()` function. We simply pass it the column of the dataframe for which we want to create a tabular summary.

```
In [43]: # use table() to get counts of arrests by `weekday`

         table(arrests_df$weekday)
```

```
 Fri  Mon  Sat  Sun  Thu  Tue  Wed 
1278 1164 1277 1293 1178 1323 1242 
```