STSCI 4520 Exercise 1

Nick Gembs

Stat Computing - Exercises 01 - Dates and Date-times

Include your answers as code chunks within this document, and render (knit) it as a pdf.

To help the graders differentiate between the question statements and your own comments, please change the formatting of the question statements to italics.

Due to the many formatting conventions for dates and times, and due to time zone considerations, dates can be tricky to work with. These exercises will give you some practice working with dates and times in R.

1. Read the documentation for `as.Date` and answer the following questions:

a. Why does `as.Date("2023-01-01")` work without supplying a format?

```
as.Date("2023-01-01")
```

```
## [1] "2023-01-01"
```

This works because if format is not specified, it will will try the formats of tryformats one by one until one works or until all have been tried. Since tryformats is not specified either, as.Date will resort to strptime() for processing, which is in "%Y-%m-%d %H:%M:%S" format. This format aligns with the input above.

b. Why doesn't `as.Date(365)` work and how can you make it work to produce the date 2023-01-01?

```
as.Date(365, origin = as.Date("2022-01-01"))
```

```
## [1] "2023-01-01"
```

as.Date will only accept numeric data if an origin date is provided. In order to make the date 365 equivalent to 2023-01-01, the origin needs to be set to as.Date("2022-01-01") as seen above.

2. Using a combination of `seq` and `as.Date`, display the dates of our Tuesday lectures throughout the semester (you can ignore the fact that we don't meet during spring break).

```
as.Date(seq(from =0, to = 98, by = 7), origin = as.Date("2023-01-24"))
```

```
##  [1] "2023-01-24" "2023-01-31" "2023-02-07" "2023-02-14" "2023-02-21"
##  [6] "2023-02-28" "2023-03-07" "2023-03-14" "2023-03-21" "2023-03-28"
## [11] "2023-04-04" "2023-04-11" "2023-04-18" "2023-04-25" "2023-05-02"
```

3. Read the documentation for `strptime` and use `format` convert 2023-01-25 to these formats. a. 01/25/2023

```
date <- as.Date("2023-01-25")
format(date, format = "%m/%d/%Y")

## [1] "01/25/2023"
```

b. 25/01/2023

```
format(date, format = "%d/%m/%Y")

## [1] "25/01/2023"
```

c. 01/25/22

```
format(date-365, format = "%m/%d/%y")

## [1] "01/25/22"
```

d. January 25, 2023

```
format(date, format = "%B %d, %Y")

## [1] "January 25, 2023"
```

e. Jan 25, 2023

```
format(date, format = "%b %d, %Y")

## [1] "Jan 25, 2023"
```

4. Use `format` to figure out what day of the week you were born on.

```
birthday <- as.Date("2002-07-17")
format(birthday, format = "%u")

## [1] "3"
```

I was born on a Wednesday. %u provides weekday as a decimal number 1-7. Monday is 1.

5. How many days are in the years 2000, 2022, 2024, and 2100? Why? Hint: R gives the right answer for all of these.

```
as.Date(366, origin = as.Date("1999-12-31"))

## [1] "2000-12-31"

as.Date(365, origin = as.Date("2021-12-31"))

## [1] "2022-12-31"

as.Date(366, origin = as.Date("2023-12-31"))

## [1] "2024-12-31"
```

```
as.Date(365, origin = as.Date("2099-12-31"))
```

```
## [1] "2100-12-31"
```

366 days in 2000 and 2024, only 365 in 2022 and 2100.

6. In a few sentences, describe the main differences between POSIXct and POSIXlt. Hint: `?POSIXlt`

POSIXct and POSIXlt are both classes used to represent dates and times. The difference is that POSIXct is the number of seconds since the beginning of 1970 while POSIXlt is a named list of vectors representing the time in seconds, minutes, hours, days, etc. POSIXct is more useful for computing in dataframes while POSIXlt is more readable for humans.

7. Convert 2023-01-01 and 2023-07-01 to POSIXct format, and print them out. What do the three-letter abbreviations stand for?

```
date <- as.Date("2023-01-01")
as.POSIXct(date)
```

```
## [1] "2022-12-31 19:00:00 EST"
```

```
date <- as.Date("2023-07-01")
as.POSIXct(date)
```

```
## [1] "2023-06-30 20:00:00 EDT"
```

The three letter abbreviations are time zones. EST = Eastern Standard Time, EDT = Eastern Daylight Time.

8. Use `as.POSIXct` to create two date time objects, one referring to 2023-01-26 at noon in Ithaca's time zone and one referring 2023-01-26 at noon in UTC time. Calculate the difference between these two objects.

```
date1 <- as.POSIXct("2023-01-26 12:00", format = "%Y-%m-%d %H:%M", tz = "EST"
)
date1
```

```
## [1] "2023-01-26 12:00:00 EST"
```

```
date2 <- as.POSIXct("2023-01-26 12:00", format = "%Y-%m-%d %H:%M", tz = "UTC"
)
date2
```

```
## [1] "2023-01-26 12:00:00 UTC"
```

```
date1 - date2
```

```
## Time difference of 5 hours
```

9. Use the `attr` function to set the `"tzone"` attribute for the UTC object to Ithaca time, and confirm that the two printed times differ by the amount in the previous question.

```
attr(date2, which = "tzone") <- "EST"
date1 - date2

## Time difference of 5 hours
```

10. Create a vector that has the datetimes 2023-01-26 12:00:00 and 2023-07-26 12:00:00 in the America/New_York time zone. Then successively change the time zone to the below time zones, and print the result. Do you notice anything unexpected?

```
America/Chicago, America/Denver, America/Phoenix, America/Los_Angeles

date1 <- as.POSIXct("2023-01-26 12:00", format = "%Y-%m-%d %H:%M", tz = "EST"
)
date2 <- as.POSIXct("2023-07-26 12:00", format = "%Y-%m-%d %H:%M", tz = "EST"
)


dates = c(date1, date2)
dates

## [1] "2023-01-26 12:00:00 EST" "2023-07-26 12:00:00 EST"

attr(dates, which = "tzone") <- "America/Chicago"
dates

## [1] "2023-01-26 11:00:00 CST" "2023-07-26 12:00:00 CDT"

attr(dates, which = "tzone") <- "America/Denver"
dates

## [1] "2023-01-26 10:00:00 MST" "2023-07-26 11:00:00 MDT"

attr(dates, which = "tzone") <- "America/Phoenix"
dates

## [1] "2023-01-26 10:00:00 MST" "2023-07-26 10:00:00 MST"

attr(dates, which = "tzone") <- "America/Los_Angeles"
dates

## [1] "2023-01-26 09:00:00 PST" "2023-07-26 10:00:00 PDT"
```

Some of the timezones undergo daylight savings time, changing their 3 letter code and altering their hour by 1.

11. Share something interesting that you learned from reading the documentation for these functions, or found online.

I found it interesting that the attr function can be used to change attributes in any object. This is a very useful setter function. I also found it interesting how many codes for time were built into R under the strptime documentation. This allows detailed description of time.