Dates and Times

Solutiuons to Swirl's R Programming Exercises

07-09-2022

Acknowledgements: R Language Concepts and code questions (with minor modifications) are used here from the swirl package. https://www.r-project.org/nosvn/pandoc/swirl.html

Important note: We don't require to use library(swirl) and swirl() here because we are not going to run the R script in RStudio Console.

R has a special way of representing dates and times, which can be helpful if you're working with data that show how something changes over time (i.e. time-series data) or if your data contain some other temporal information, like dates of birth.

Dates are represented by the 'Date' class and times are represented by the 'POSIXct' and 'POSIXlt' classes. Internally, dates are stored as the number of days since 1970-01-01 and times are stored as either the number of seconds since 1970-01-01 (for 'POSIXct') or a list of seconds, minutes, hours, etc. (for 'POSIXlt').

Let's start by using d1 <- Sys.Date() to get the current date and store it in the variable d1. (That's the letter 'd' and the number 1.)

```
d1 <- Sys.Date()
```

Use the class() function to confirm d1 is a Date object.

```
class(d1)
```

```
## [1] "Date"
```

We can use the unclass() function to see what d1 looks like internally. Try it out.

```
unclass(d1)
```

```
## [1] 19182
```

That's the exact number of days since 1970-01-01!

However, if you print d1 to the console, you'll get today's date – YEAR-MONTH-DAY. Give it a try.

```
d1
```

```
## [1] "2022-07-09"
```

What if we need to reference a date prior to 1970-01-01? Create a variable d2 containing as.Date("1969-01-01").

```
d2 <- as.Date("1969-01-01")
```

Now use unclass() again to see what d2 looks like internally.

```
unclass(d2)
```

```
## [1] -365
```

As you may have anticipated, you get a negative number. In this case, it's -365, since 1969-01-01 is exactly one calendar year (i.e. 365 days) BEFORE 1970-01-01.

Now, let's take a look at how R stores times. You can access the current date and time using the Sys.time() function with no arguments. Do this and store the result in a variable called t1.

```
t1 <- Sys.time()
```

```
t1
```

```
## [1] "2022-07-09 01:37:54 EDT"
```

And check the class() of t1.

```
class(t1)
```

```
## [1] "POSIXct" "POSIXt"
```

As mentioned earlier, POSIXct is just one of two ways that R represents time information. (You can ignore the second value above, POSIXt, which just functions as a common language between POSIXct and POSIXlt.) Use unclass() to see what t1 looks like internally – the (large) number of seconds since the beginning of 1970.

```
unclass(t1)
```

```
## [1] 1657345075
```

By default, Sys.time() returns an object of class POSIXct, but we can coerce the result to POSIXlt with as.POSIXlt(Sys.time()). Give it a try and store the result in t2.

```
t2 <- as.POSIXlt(Sys.time())
```

```
class(t2)
```

```
## [1] "POSIXlt" "POSIXt"
```

Now view its contents.

```
t2
```

```
## [1] "2022-07-09 01:37:54 EDT"
```

The printed format of t2 is identical to that of t1. Now unclass() t2 to see how it is different internally.

```
unclass(t2)
```

```
## $sec
## [1] 54.89726
##
## $min
## [1] 37
##
## $hour
## [1] 1
##
## $mday
## [1] 9
##
## $mon
## [1] 6
##
## $year
## [1] 122
##
## $wday
## [1] 6
##
## $yday
## [1] 189
##
## $isdst
## [1] 1
##
## $zone
## [1] "EDT"
##
## $gmtoff
## [1] -14400
##
## attr(,"tzone")
## [1] ""    "EST" "EDT"
```

t2, like all POSIXlt objects, is just a list of values that make up the date and time. Use str(unclass(t2)) to have a more compact view.

```
str(unclass(t2))
```

```
## List of 11
##  $ sec   : num 54.9
##  $ min   : int 37
##  $ hour  : int 1
##  $ mday  : int 9
##  $ mon   : int 6
##  $ year  : int 122
##  $ wday  : int 6
##  $ yday  : int 189
```

```
##  $ isdst : int 1
##  $ zone  : chr "EDT"
##  $ gmtoff: int -14400
##  - attr(*, "tzone")= chr [1:3] "" "EST" "EDT"
```

ust the minutes from the time stored in t2, we can access them with t2$min. Give it a try.

```
t2$min
```

```
## [1] 37
```

Now that we have explored all three types of date and time objects, let's look at a few functions that extract useful information from any of these objects – weekdays(), months(), and quarters(). The weekdays() function will return the day of week from any date or time object. Try it out on d1, which is the Date object that contains today's date.

```
weekdays(d1)
```

```
## [1] "Saturday"
```

The months() function also works on any date or time object. Try it on t1, which is the POSIXct object that contains the current time (well, it was the current time when you created it).

```
months(t1)
```

```
## [1] "July"
```

The quarters() function returns the quarter of the year (Q1-Q4) from any date or time object. Try it on t2, which is the POSIXlt object that contains the time at which you created it.

```
quarters(t2)
```

```
## [1] "Q3"
```

Often, the dates and times in a dataset will be in a format that R does not recognize. The strptime() function can be helpful in this situation.

strptime() converts character vectors to POSIXlt. In that sense, it is similar to as.POSIXlt(), except that the input doesn't have to be in a particular format (YYYY-MM-DD).

To see how it works, store the following character string in a variable called t3: "October 17, 1986 08:24" (with the quotes).

```
t3 <- "October 17, 1986 08:24"
```

Now, use strptime(t3, "%B %d, %Y %H:%M") to help R convert our date/time object to a format that it understands. Assign the result to a new variable called t4. (You should pull up the documentation for strptime() if you'd like to know more about how it works.)

```
t4 <- strptime(t3, "%B %d, %Y %H:%M")
```

Print the contents of t4.

```
t4
```

```
## [1] "1986-10-17 08:24:00 EDT"
```

Now, let's check its class().

```
class(t4)
```

```
## [1] "POSIXlt" "POSIXt"
```

Finally, there are a number of operations that you can perform on dates and times, including arithmetic operations (+ and -) and comparisons (<, ==, etc.)

The variable t1 contains the time at which you created it (recall you used Sys.time()). Confirm that some time has passed since you created t1 by using the 'greater than' operator to compare it to the current time: Sys.time()

```
t1
```

```
## [1] "2022-07-09 01:37:54 EDT"
```

```
Sys.time() >  t1
```

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

So we know that some time has passed, but how much? Try subtracting t1 from the current time using Sys.time() - t1. Don't forget the parentheses at the end of Sys.time(), since it is a function.

```
Sys.time() - t1
```

```
## Time difference of 0.1195521 secs
```

The same line of thinking applies to addition and the other comparison operators. If you want more control over the units when finding the above difference in times, you can use difftime(), which allows you to specify a 'units' parameter.

Use difftime(Sys.time(), t1, units = 'days') to find the amount of time in DAYS that has passed since you created t1.

```
difftime(Sys.time(), t1, units = 'days')
```

```
## Time difference of 1.452107e-06 days
```

In this lesson, you learned how to work with dates and times in R. While it is important to understand the basics, if you find yourself working with dates and times often, you may want to check out the lubridate package by Hadley Wickham.