Simulation

Raphael Carvalho

04/06/2019

Dates and Times

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

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

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

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

```
unclass(d1)
```

```
## [1] 18051
```

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] "2019-06-04"
```

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")
unclass(d2)
```

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

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()
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 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] 1559696036
```

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"
```

```
t2
```

```
## [1] "2019-06-04 21:53:56 -03"
```

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] 56.27899
##
## $min
## [1] 53
##
## $hour
## [1] 21
##
## $mday
## [1] 4
##
## $mon
## [1] 5
##
## $year
## [1] 119
##
## $wday
## [1] 2
##
## $yday
## [1] 154
##
## $isdst
## [1] 0
##
## $zone
## [1] "-03"
##
## $gmtoff
## [1] -10800
##
## attr(,"tzone")
## [1] ""    "-03" "-02"
```

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 56.3
##  $ min   : int 53
##  $ hour  : int 21
##  $ mday  : int 4
##  $ mon   : int 5
##  $ year  : int 119
##  $ wday  : int 2
##  $ yday  : int 154
##  $ isdst : int 0
##  $ zone  : chr "-03"
##  $ gmtoff: int -10800
##  - attr(*, "tzone")= chr [1:3] "" "-03" "-02"
```

If, for example, we want just the minutes from the time stored in t2, we can access them with t2$min. Give it a try.

```
t2$min
```

```
## [1] 53
```

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] "Terça Feira"
```

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] "Junho"
```

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] "Q2"
```

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")
t4
```

```
## [1] NA
```

```
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

```
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 0.0827961 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 1.028508e-06 days
```