When I was first learning R, working with dates was one of the hardest and most time consuming tasks I dealt with. There are so many things to learn! What do I do with as.POSIXct(), as.POSIXlt(), strftime(), strptime(), format(), and as.Date()? R date formats were confusing, and it seemed no matter what I did I was always running into issues.

And I’ll be honest, working with dates in R still trips me up from time to time. It can be confusing. But I’ve learned to follow a procedure to guide me through any date manipulation task with ease.

Today I’m going to help you learn that same procedure so you’ll never have to worry about R date format issues again!

**The goals of most R date format exercises**

When working with R date formats, you’re generally going to be trying to accomplish one of two different but related goals:

1. Converting a character string like “Jan 30 1989” to a Date type
2. Getting an R Date object to print in a specific format for a graph or other output

You may need to handle both of these goals in the same analysis, but it’s best to think of them as two separate exercises. Knowing which goal you are trying to accomplish is important because you will need to use different functions to accomplish each of these. Let’s tackle them one at a time.

**Converting a character string to a date**

A common scenario is that you have read in a .csv file where one of the columns contains dates. Often you will find this column is read in as a character vector.

If you don’t need to use this column for anything, this might not be a problem.

But, often, you will need to do things with it! You’ll want to sort a graph by date, or calculate the number of days between two dates, or format your dates in a specific way.

To accomplish any of those things, you’ll first need to convert your character vector to a Date vector.

R has 3 main object types for working with dates: Date, POSIXct, and POSIXlt. Date objects can only work with dates, while POSIXct and POSIXlt objects can work with both dates and times.

Before you do any conversion, you need to first decide whether you want to keep any time data (if available) or if you only are working with dates.

If you’re only working with dates, you’ll want the as.Date() function which produces objects of type Date.

If you want both dates and times, you’ll want the as.POSIXct() function which produces objects of type POSIXct.

Luckily, you’ll find that these functions operate very similarly to one another, so you won’t need to worry about memorizing little idiosyncracies between them!

Side note: In this post, I’m going to ignore the POSIXlt type which is very similar to POSIXct with some implementation differences beyond the scope of this post.

**Converting a character string to a date using the as.Date() R function**

The main function for converting from a character string to a Date (*without* time information) is the as.Date() function. as.Date() accepts a date vector and a format specification. The format specification identifies what date information is contained in the character string you are providing. Let’s look at an example:

my\_date <- "01/30/1989" # Input character string

as.Date(my\_date, format = "%m/%d/%Y")

## [1] "1989-01-30"

Our input character string was in the format month/day/year, and we used the R format specification that corresponds to this, %m/%d/%Y, to convert this character string to a date.

**Converting a character string to a POSIXct datetime using the as.POSIXct() R function**

The main function for converting from a character string to a POSIXct datetime object (*with* time information) is the as.POSIXct() function. Just like as.Date(), as.POSIXCT() accepts a date vector and a format specification, which identifies what date and time information is contained in the character string you are providing. Here’s an example:

my\_date\_time <- "01/30/1989 23:40:00" # Input character string with time information

as.POSIXct(my\_date\_time, format = "%m/%d/%Y %H:%M:%S")

## [1] "1989-01-30 23:40:00 EST"

Here we had the added the time string 23:40:00 on top of the same date we processed previously. As before, we used the R format specification for this string, %m/%d/%Y %H:%M:%S, to convert to a datetime POSIXct object.

**Discarding unnecessarily time data**

We’ll get more into how these format specifications work in a minute, but first I want to make a quick aside. Sometimes your dates will contain time information, but you won’t actually need that for your analysis. This can sometimes be annoying to keep around, and it’s often cleaner if you just get rid of it. Luckily, this is quite easy. Instead of using the as.POSIXct() function, we can simply use the as.Date() function and ignore the trailing timestamp information, as follows:

my\_date\_time <- "01/30/1989 23:40:00"

as.Date(my\_date\_time, format = "%m/%d/%Y")

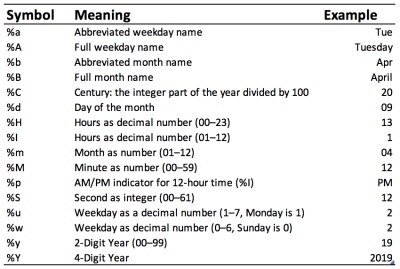
## [1] "1989-01-30"

This will convert the character strings to Date objects, dropping the extraneous time data from our dataset. Remember: if you need time data, use as.POSIXct(), but if you don’t, just use as.Date()!

**More on R date format specifications**

Above, we reviewed examples of how as.Date() and as.POSIXct() can convert character strings to dates, given the right *format specification*. Now, it’s time to review what the different format specifications are, and how we can use them to convert character strings formatted in all different ways to dates.

We’ve already briefly seen the symbols %m, %d, %Y, %H, %M, and %S, and you probably have some idea that these correspond to month, day, year, hour, minute, and second. I’d now like to introduce the list of the most commonly used R date format specifications:



Look through this table to identify the different date formats we worked through previously. Pay special attention to the difference between %b, %B, and %m, as well as %y and %Y!

**Standard procedure for converting a character string to date or datetime object**

1. Identify the key variables we need to map
   * Generally month, day, and year for dates
   * Add in hour, minute, and second for times
2. For each key variable, identify the appropriate mapping
3. Construct the format specification string
4. Construct the as.Date() or as.POSIXct() function call

Let’s work through a few examples!

Say we have a string in the format Jan 30th, 1989 23:40.

1. The variables we need to map are month, day, year, hour, and minute
2. Let’s find the appropriate mappings:
   * This string uses an abbreviated month, which maps to %b
   * Day of the month maps to %d
   * 4-digit years map to %Y
   * 24-hour-clock hour maps to %H
   * Minutes map to %M
3. The format specification string should exactly match the input string, simply substituting in our mappings from above. In this case: Jan 30th, 1989 23:40 becomes %b %dth, %Y %H:%M
4. Because we’re dealing with both dates and times here, we know we’re going to need as.POSIXct() if we want to maintain that time data. Let’s give this a try:

my\_date\_time <- "Jan 30th, 1989 23:40"

as.POSIXct(my\_date\_time, format = "%b %dth, %Y %H:%M")

## [1] "1989-01-30 23:40:00 EST"

Awesome! It processed the data exactly as we needed it. Let’s go through one more exercise to make sure we have this down before we move onto date formatting for output.

In this case, say we have the string 30 January 1989 11:40 PM.

1. The variables we need to map are day, month, year, hour, minute, and AM/PM
2. Let’s again find the appropriate mappings:
   * Day of the month maps to %d
   * This string uses a full month, which maps to %B
   * 4-digit years map to %Y
   * 12-hour-clock hour maps to %I
   * Minutes map to %M
   * AM/PM indicator maps to %p
3. Substituting, 30 January 1989 11:40 PM becomes %d %B %Y %I:%M %p
4. Again, because we’re dealing with both dates and times we need as.POSIXct to maintain the time data

my\_date\_time <- '30 January 1989 2:24 PM'

as.POSIXct(my\_date\_time, format = "%d %B %Y %I:%M %p")

## [1] "1989-01-30 14:24:00 EST"

There we have it! You should now be equipped to take a given character string, determine the format of that string, and then use either the as.Date() R function (in the case of only dates) or the as.POSIXct() R function (in the case of dates and times) to convert that character string to a date or datetime representation.

Now, let’s turn to our second challenge: formatting an R date or datetime object for output, cleaning things up to be publication-ready. Luckily, you’ll find there are many similarities in the approach to what you’ve just learned!

**Formatting a date for publication-ready output**

In this case, you have an R object that is already stored as one of several R date formats (Date, POSIXct, or POSIXlt), and now you’d like to clean up that date for graphing or publication. I find that I perform this type of transformation most often when I’m making graphs, but this is useful for creating RMarkdown reports and other output as well.

While earlier we used the as.POSIXct() and as.Date() R functions to convert from characters to dates, we’ll now be using the format() R function to convert from dates to characters!

As before, we’ll want to decide what information is important for mapping, select the appropriate format specification, and then build our function call. Luckily, this process looks very similar to what we did before, we’re just working in reverse! Using the same table from above, we can find the variables we need to map our date to a specific format. Review the examples below to see how we convert a date or datetime variable to different character formats for output!

my\_date <- as.Date("01/30/1989", "%m/%d/%Y")

my\_date # Unformatted date

## [1] "1989-01-30"

format(my\_date, '%B %d %Y') # Date format 'January 30 1989'

## [1] "January 30 1989"

format(my\_date, '%B %dth, %Y') # Date format 'January 30th, 1989'

## [1] "January 30th, 1989"

format(my\_date, '%d %b %Y') # Date format '30 Jan 1989'

## [1] "30 Jan 1989"

format(my\_date, '%A %B %d %Y') # Date format 'Monday January 30 1989'

## [1] "Monday January 30 1989"

format(my\_date, '%m/%d/%y') # Date format '01/30/90'

## [1] "01/30/89"

my\_date\_time <- Sys.time() # Function that generates the current time

my\_date\_time # Unformatted datetime

## [1] "2019-04-09 13:28:12 EDT"

format(my\_date\_time, '%B %d %Y') # Datetime format 'April 09 1989' (Discard time)

## [1] "April 09 2019"

format(my\_date\_time, '%B %d %Y %H:%M') # Datetime format 'April 09 1989 12:14'

## [1] "April 09 2019 13:28"

format(my\_date\_time, '%H:%M on %B %d %Y') # Datetime format '12:14 on April 09 1989'

## [1] "13:28 on April 09 2019"

There we have it! You should now be able to easily perform nearly any date operation you need in R. You can take character strings and convert them to dates using the as.POSIXct() and as.Date() R functions. You can also take date or datetime objects and use the format() function to clean them up for publication-ready graphs and papers!