# How does R handle date values?

Authored by

**stats writer**

June 30, 2024

R is a programming language that is specifically designed for statistical computing and graphics. One of its key features is its ability to handle date values efficiently. R uses the date class to represent date values, which allows for various operations and manipulations to be performed on them. The date values in R are stored as the number of days since January 1, 1970, which is known as the "epoch date". This allows for easy computation and comparison of dates. R also has a built-in date function that allows users to convert character strings to date values, making it easier to work with different date formats. Furthermore, R offers a wide range of date-related functions and packages that enable users to perform tasks such as date arithmetic, formatting, and extraction of specific date components. Overall, R's handling of date values is efficient, versatile, and well-suited for data analysis and visualization purposes.

## How does R handle date values? | R FAQ

**Date variables can pose a challenge in data management. This is true in any package and different packages handle date values differently. This page aims to provide**
**an overview of dates in R-how to format them, how they are stored, and what**
**functions are available for analyzing them.**

### Defining variables as dates

**For a date variable stored as a vector of strings, see R FAQ: How can I format a string containing a date in R "Date" object?.**

**For a date variable stored as a vector of numbers, there is a little bit of**

detective work to be done. Look at the few of the numbers and see if there's a
clear pattern. If the numeric values are actually month, day, and
year values concatenated without separation, like 20011010 for October 10, 2001,
then these values should be converted to character strings (using
as.character) and then formatted using the tips in the link above.

If the numeric values are counting the days that have passed since some starting
date, then the as.Date function can be used with an origin date
indicated. If we need to read a numeric value that represent date from Excel to R we need to notice that Excel dates, when converted to integers, are counting from January 1,
1900. However, Excel uses 1 to represent this origin, while R uses 0, so we should first subtract 1 from each value before converting. Furthermore, Excel erroneously treats 1900 as a leap year, and so all numbers representing dates after February 28, 1900 are

incorrectly incremented by 1, so we will need to subtract one from dates that fall after February 28, 1900. After cleaning up the numbers, we can indicate Excel's origin date of January 1, 1900 in as.Date.

#These correspond to the following dates in Excel: #1: January 1, 1900 #59: February 28, 1900 #61: March 1, 1900 (remember that Excel wrongly treats 1900 as a leap year) #1000: September 26, 1902
edates <- c(1, 59, 61, 1000)edates <- edates - 1edates

0 58 60 999

edates <- edates - 1as.Date(edates, origin="1900-01-01")

"1900-01-01" "1900-02-28" "1900-03-01" "1902-09-26"

Other packages store dates using different origins. SAS, for example uses
1960 rather than 1900. When R looks at dates as integers, its origin is January 1, 1970.

as.numeric(as.Date(edates, origin = "1900-01-01"))

-25567 -25509 -25508 -24569

**startdate <- "1970-01-01"**

**as.numeric(as.Date(startdate))**

**0**

### Date objects in R

**Date objects are stored in R as integer values, allowing for dates to be compared and manipulated as you would a numeric vector. Logical comparisons are a simple. When referring to dates, earlier dates are "less than" later dates. Returning to our example above, we can compare the three dates in edates to January 1, 1970. For dates prior to this, the comparison should return TRUE. For later dates, the comparison should return FALSE. Here all dates are earlier than January 1, 1970.**

**as.Date(edates, origin = "1900-01-01") < "1970-01-01"**

**TRUE TRUE TRUE TRUE**

**Adding a week to dates can be done by simply adding**

## 7. The date format is maintained.

```
weeklater <- as.Date(edates, origin = "1900-01-01") + 7
weeklater
"1900-01-08" "1900-03-07" "1900-03-08" "1902-10-03"

class(weeklater)
"Date"
```

### R functions for dates

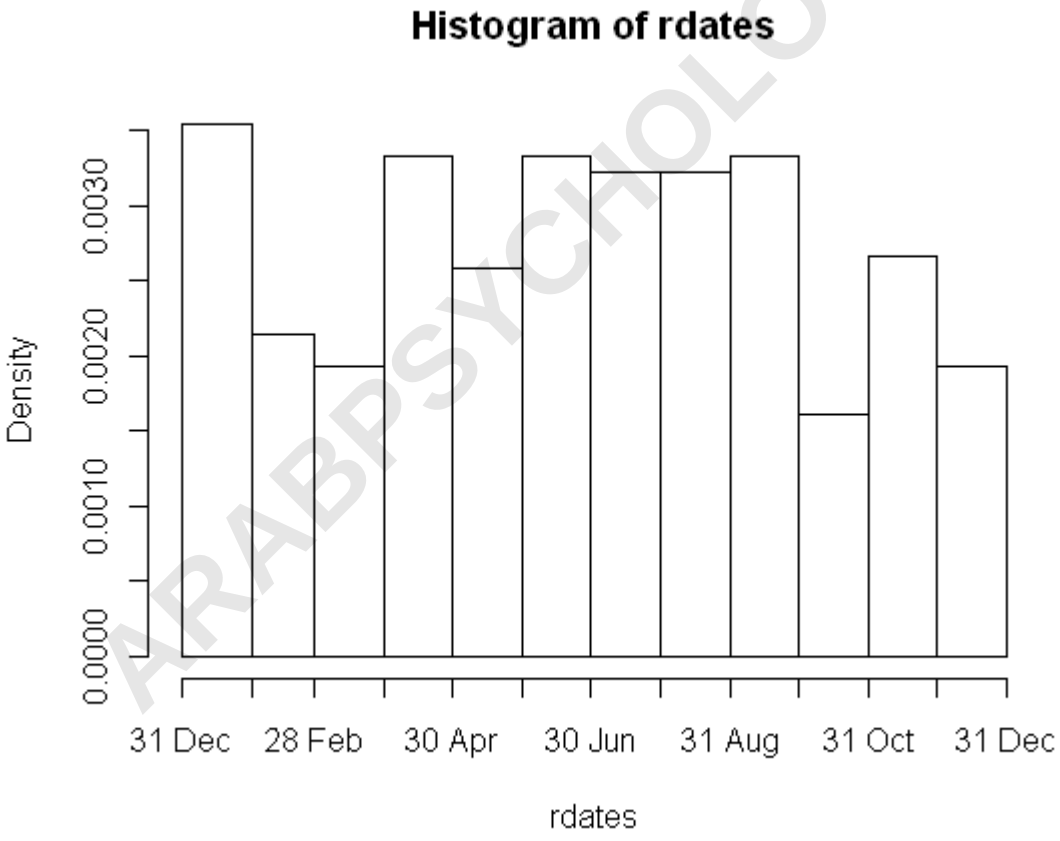
There are several functions in R specific to Date objects or for creating Date objects. The Sys.Date() function generates the value of the current date. It is easy to extract the day of the week and the month.

```
weekdays(weeklater)
"Monday" "Wednesday" "Thursday" "Friday"

months(weeklater)
"January" "March" "March" "October
```

**If you are interested in the distribution of a date variable, there are plotting functions available. Below, we randomly sample 100 dates in a year and then plot a histogram with one bar per month.**

```
rdates <- as.Date("2010/1/1") + floor(365*runif(100))hist(rdates, "months", format = "%d %b")
```






**Much, much more**

**This has been a very quick overview. R also has date-time objects and functions specific to date-time data and more far functions than were shown here. The documentation pages for Dates and DateTimeClasses, both in base R, provide more details.**