Working with Dates and Time in R

Introduction

- R has a range of built-in functions that allow us to work with dates and time.
- Working on dates and time can be tedious when the data come with date values in different format.
- The base function of R, `as.Date()` converts a variety of character date formats into R dates. Once converted to dates, the following functions will return information about dates: `weekdays(), months(), quarters(), seq()`.
- `as.Date()` handles only dates.
- For handling both dates and time there is a package called lubridate. The inbuilt function of this package offers a nice way to make easy parsing in dates and times.

In this tutorial we will see how base function `as.Date()` & other related functions and package lubridate works and also learn date manipulation tasks.

Base Package Functions

`as.Date()` converts dates entered as strings into numeric dates.

```
as.Date(x, format="%Y-%m-%d")
```

x is a string object to be converted

`format=` is the format (in which the date appears within the string) composed of codes such as:

Day	day as a number (01-31)	%d
	abbreviated weekday (Mon)	%a
	full weekday name (Monday)	%A
Month	abbreviated month (Jan)	%b
	full month name (January)	%B
	month as a number (01-12)	%m
Year	2-digit year (16)	%y
	4-digit year (2016)	%Y

Base Package Functions

```
# Formatting a date
```

```
sdate1<-"5jan2010"
ndate1<-as.Date(sdate1,format="%d%b%Y")
ndate1
```

```
[1] "2010-01-05"
```

Default format for dates in as.Date() is YYYY-MM-DD — four digits for year, and two digits for month and day, separated by a hyphen.

```
class(ndate1)
```

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

```
# Using format argument to extract parts of date
```

```
format(ndate1,format="%Y")
```

```
[1] "2010"
```

Format codes can also be used to extract parts of dates using format()

```
format(ndate1,format="%Y%B")
```

```
[1] "2010January"
```

Base Package Functions

Once you have converted to dates, the following functions will return information about dates:

```
#To extract Day of the week.
weekdays(ndate1)
[1] "Monday"
```

```
#To extract Month of the Year.
months(ndate1)
[1] "August"
```

```
#To extract Quarter no.
quarters(ndate1)
[1] "Q3"
```

```
#Generates dates sequences for Date object by 2 months.
x<-seq(ndate1,by="2 months",length.out=5)
x
[1] "2016-08-08" "2016-10-08" "2016-12-08" "2017-02-08" "2017-04-08"
```

Base Package Functions

```
# Capture current date
```

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

```
[1] "2017-01-05"
```

Sys.Date() returns the your system’s current date

```
# Using operators with dates
```

```
d1<-as.Date("20101201",format="%Y%m%d")
d2<-as.Date("10/7/04",format="%m/%d/%y")
d1
d2
```

```
[1] "2010-12-01“
[1] "2004-10-07"
```

```
d1-d2
```

```
Time difference of 2246 days
```

```
d1-d2>365
```

```
[1] TRUE
```

Different operators can be used with date objects

Package lubridate

- This package is developed by Garrett Grolemund and Hadley Wickham.
- lubridate offers many useful functions to work with date-times and timespans which makes basic date-time manipulations much more straightforward.
- It works for most of the popular date-time object classes (Date, POSIXt, chron, etc.), which is not always true for base R functions.

```r
# Install and load package lubridate
install.packages("lubridate")
library(lubridate)
```

Package lubridate Functions

```
#To parse character strings into dates.


mdy("12-01-2015")
[1] "2015-12-01"
```

The letters y, m, and d correspond to the year, month, and day elements of a date-time. To read in a date, choose the function name that matches the order of elements in your date-time object.

```
Other functions are: ymd(), ydm(), dmy(), hm(), hms() and ymd_hms()
```

```
#To capture the Current date and time.
date<-now()
date
[1] "2019-04-11 12:48:00 IST“
```

Package lubridate Functions

```
#To extract the hour component from the date object.
hour(date)
[1] 12
```

```
#To extract the minute component from the date object.
minute(date)
[1] 38
```

```
#To extract the second component from the date object.
second(date)
[1] 59.24696
```

Merge Three Different Columns Into a Date in R

```
# My Dataframe
```

```
EmpID<-c(101,102,103,104,105)
year<-c(1977,1989,2000,2012,2015)
month<-c(2,5,10,1,11)
day<-c(2,3,1,1,5)

datedf<-data.frame(EmpID,year,month,day)
datedf
```

```
  EmpID year month day
1   101 1977     2   2
2   102 1989     5   3
3   103 2000    10   1
4   104 2012     1   1
5   105 2015    11   5
```

Data: Employee ID (EmpID) and joining date (split into 3 columns: year month & day)

Merge Three Different Columns Into a Date in R

We are having 3 separate columns as year, month, and day in our dataframe **datedf**.

```
# Merge 3 columns into one date column
```

```
datedf$date<-as.Date(paste(datedf$year,datedf$month,datedf$day,
sep='-'),format="%Y-%m-%d")
datedf
```

```
  EmpID year month day       date
1   101 1977     2   2 1977-02-02
2   102 1989     5   3 1989-05-03
3   103 2000    10   1 2000-10-01
4   104 2012     1   1 2012-01-01
5   105 2015    11   5 2015-11-05
```

- ❑ new column date is created using $
- ❑ paste() combines the columns
- ❑ as.date() converts date column into a date type

Format a Vector With Inconsistent Date Formats

Converting dates entered as strings into numeric dates in R is a little tricky if the date information is not represented consistently. Let's see how to deal with this kind of situation.

```
dates<-c("12aug08","01sep09","7august06","9august2007","20july1999")
ndates<-as.Date(dates,format="%d%b%y")
```

```
ndates
[1] "2008-08-12" "2009-09-01" "2006-08-07" "2020-08-09" "2019-07-20"
```

*In Vector **dates**, there are multiple date formats. We have specified a date format that appears appropriate for the first few dates including **%d**, which allows for days within a month to optionally have a leading zero when less than 10, and **%b**, which can match either an entire or an abbreviate month name. Unfortunately, we have some years that appear as 2 digits and some that appear as 4. As a result, we can see in our results that the first three dates are correct and the last two are not*

Format a Vector With Inconsistent Date Formats

In the loop below, we go through our vector of numeric dates and see if any appear later than 2018. For these, we assume the date is presented with 4 digits and reread the string with the appropriate format.

```
for (i in 1:length(ndates)){
  if ((ndates[i])> as.Date("2018-01-01")){
    ndates[i] <- as.Date(dates[i],format="%d%B%Y")
  }
}
ndates
```

```
[1] "2008-08-12" "2009-09-01" "2006-08-07" "2007-08-09" "1999-07-20"
```

length() returns the length of an object
for loop is executed from 1 to the length of ndates i.e. 5

Format a Vector With Inconsistent Date Formats

The code we used here cannot be applied to many situations, but the steps we used can be:

- Use a format that is appropriate for as many dates as possible, focusing on the more flexible formats offered by R.
- Determine when the format did not work properly and define a rule for finding such cases.
- Use a format appropriate to the misread dates.

Quick Recap

In this session, we learnt how to deal with dates and time using base package functions in R & package lubridate, how to merge 3 different columns into one date column and how to format a vector with inconsistent dates. Here is a quick recap:

Base functions	• **`as.date(), weekdays(), months(), quarters, seq()., Sys.Date()`**
Package lubridate functions	• **`ymd() series, now(), hour(), minute(), second()`**
Date manipulation tasks	• Merge Three Different Columns Into a Date in R using **`paste()`**