TSA - Sp22 - Date Objects

Luana Lima

1/19/2022

Date or time period

The data sets we will work with will be index by time, remember we are doing TIME series analysis. After importing your data set make sure that you have your dates right.

Formatting of dates in R:

%d day as number (0-31) %m month (00-12, can be e.g., 01 or 1) %y 2-digit year %Y 4-digit year %a abbreviated weekday %A unabbreviated weekday %b abbreviated month %B unabbreviated month

```
# Adjust date formatting for today
# Write code for three different date formats.
# An example is provided to get you started.
# (code must be uncommented)
today <- Sys.Date()
format(today, format = "%B")
```

```
## [1] "January"
```

```
format(today, format = "%y")
```

```
## [1] "22"
```

```
#format(today, format = "")
#format(today, format = "")

#as.Date()
```

Package lubridate

The best package to handle date conversion in R is lubridate. Let's see how we can use lubridate functions to combine those two columns into one date object. For more info on lubridate functions refer to tho this file file also available on our Sakai lessons page for M3.

Install and load the package lubridate into your R session. Lubridate offers fast and user friendly parsing of date-time data. Create a string for today's data and then convert it to R date object using lubridate.

```
#install.packages("lubridate")
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
#Ex1
str_today <- "2022-jan-19"
#Since the format is year-month-day we will use function ymd()
date_obj_today <- ymd(str_today)
date_obj_today
```

```
## [1] "2022-01-19"
```

```
#Ex2
str_today <- "2022-jan-19"
#Sine the format is year-month-day we will use function ymd()
date_obj_today <- ymd(str_today)
date_obj_today
```

```
## [1] "2022-01-19"
```

```
#there are other similar functions ydm(), mdy(), etc
```

```
#century issue
str_past <- "55-jan-19"
date_obj_past <- ymd(str_past)
date_obj_past
```

```
## [1] "2055-01-19"
```

```
#Build a function to fix year that is more general than the one discussed in the lesson
fix.early.dates <- function(d, cutoff) {
       m <- year(d) %% 100  #operator %% is a modular division i.e. integer-divide year(d) by 100 and r
       year(d) <- ifelse(m > cutoff, 1900+m, 2000+m)  #this will update year(d), year() is a function t
       return(d)
}

fixed_date_obj_past <- fix.early.dates(date_obj_past,cutoff=21) #cutoff could be the current year to be
fixed_date_obj_past
```

```
## [1] "1955-01-19"
```

```
#Fix for century issue
str_past <- "55-jan-19"
#Alternative 1
date_obj_past <- fast_strptime(str_past,"%y-%b-%d",cutoff_2000=21)
date_obj_past
```

```
## [1] "1955-01-19 UTC"
```

```r
#Alternative 2 - easiest
date_obj_past2 <- parse_date_time2(str_past,"ymd",cutoff_2000=21)
date_obj_past2
```

```
## [1] "1955-01-19 UTC"
```

```r
#Functions ymd(), mdy(), ydm() do not take argument cutoff_2000
```

In some cases when dates are provided as integers, you may need to provide an origin for your dates. For example, excel date could be given as number of days since an origin date. Origin date can be different. When R looks at dates as integers, its origin is January 1, 1970. Check if that is true on your machine.

```r
#Check if "1970-01-01" is your origin date.
lubridate::origin
```

```
## [1] "1970-01-01 UTC"
```