Tools for Data Science Workshop

Dates and Times with Lubridate

By Monserrat López and
Anzhelika Belozerova

Hertie School

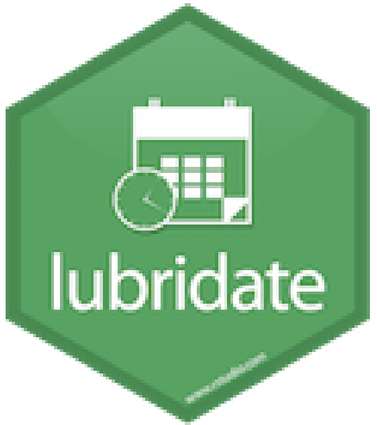
Introduction

Today's Session Goals:

- Understand the core components of Lubridate.
- Identify the different **date and time formats** we can work with.
- To recognize the **essential functions and operators in Lubridate,** which enable us to efficiently manipulate date and time data in R.

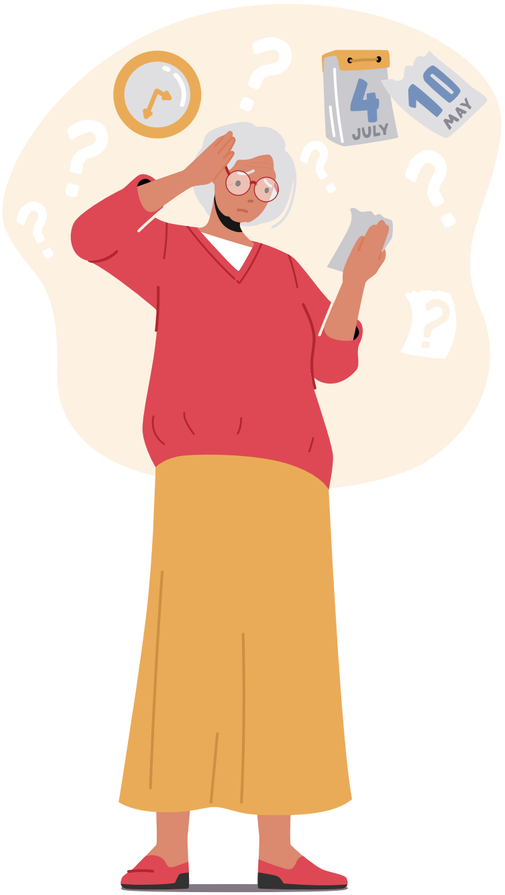
What is Lubridate?

It is a R package that **simplifies date and time data manipulation in R.** This package offers a set of standardized functions for parsing dates, performing arithmetic with date-time data, and extracting specific components of date and time formats (e.g., year, month, day).



Why use lubridate?

When working with data and time datasets, we may encounter various challenges. One of them is that **date formats may differ across countries**. For instance, the United States commonly uses the format month, day, and year (MM/DD/YYYY), while in Europe, it's often day, month, and year (DD/MM/YYYY). These variations can lead to errors and misunderstandings.



- The international standard recommends writing the date as year, followed by month and day: YYYY-MM-DD, and this is the recommended format when working with R. You will often need to **convert different date formats** into this universal standard.

Why use lubridate?

Lubridate help addressing such problems, helping to manipulate date and time data with intuitive functions for parsing dates, handling time zones, and extracting components. It **recognizes various date formats, supports leap years and irregular dates,** and is integrated to Tidyverse!

- Note that Lubridate **is not automatically loaded by the Tidyverse package,** despite being part of its ecosystem. Although it is installed with Tidyverse, **you should load it separately** by typing `library(lubridate)` in your console.

Date and POSIXct Classes

1. Date Class

- Represents **dates without a specific time component.**
- Note that with the functions year(), month() and day() **we can extract components** of the dates.

```
> # 1. Creating a Date object
> date_obj <- ymd("2023-10-26")
> class(date_obj)
[1] "Date"
> # Extracting components
> year(date_obj)  # Returns the year (e.g., 2023)
[1] 2023
> month(date_obj) # Returns the month (e.g., 10)
[1] 10
> day(date_obj)   # Returns the day (e.g., 26)
[1] 26
```

Lubridate Basics: Date and POSIXct

2. POSIXct Class

- Represents date and time values **with precision up to seconds and time zones.** You can use it for tasks where both date and time information are crucial, such as time series analysis.
- Note that with this date and time class, **we can extract additional components** as, hour, minute, seconds and time zones.

```
> datetime_obj <- ymd_hms("2023-10-26 14:30:00")
> class(datetime_obj)
[1] "POSIXct" "POSIXt"
> # Extracting components
> year(datetime_obj)    # Returns the year (e.g., 2023)
[1] 2023
> month(datetime_obj)   # Returns the month (e.g., 10)
[1] 10
> day(datetime_obj)     # Returns the day (e.g., 26)
[1] 26
> hour(datetime_obj)    # Returns the hour (e.g., 14)
[1] 14
> minute(datetime_obj)  # Returns the minute (e.g., 30)
[1] 30
> second(datetime_obj)  # Returns the second (e.g., 0)
[1] 0
> tz(datetime_obj)      # Returns the time zone (e.g., "UTC")
[1] "UTC"
```

Parsing Dates and Times with Lubridate

Lubridate helps to **convert text-based** date and time information (strings) **into proper date-time objects** that can be easily managed within R.

1. Dates. Functions like ymd(), mdy(), and dmy() are used for parsing date-only strings in different formats.

- Note that **the order in which you specify the year, month, and day components doesn't affect the outcome** because lubridate can interpret and rearrange them accordingly.

```
> #### Parsing dates
> ymd("2023-10-26")
[1] "2023-10-26"
> ymd("23.01.26")
[1] "2023-01-26"
> ymd("2023 October 26")
[1] "2023-10-26"
> mdy("10/26/2023")
[1] "2023-10-26"
> mdy("10.26.2023")
[1] "2023-10-26"
> mdy("October 26 2023")
[1] "2023-10-26"
> dmy("26/10/2023")
[1] "2023-10-26"
> dmy("26.10.2023")
[1] "2023-10-26"
> dmy("26 October 2023")
[1] "2023-10-26"
```

Parsing Dates and Times with Lubridate

2. Dates and Times
Functions like ymd_hms(), mdy_hm(), and dmy_hms() are used for parsing **date-time strings** in various formats.

```
> # Parsing dates and times
> ymd_hms("2023-10-26 14:30:00")
[1] "2023-10-26 14:30:00 UTC"
> mdy_hm("10/26/2023 14:30")
[1] "2023-10-26 14:30:00 UTC"
> dmy_hms("26/10/2023 14:30:00")
[1] "2023-10-26 14:30:00 UTC"
```

3. Custom Date-Time Strings
Use parse_date_time() for **flexibility in parsing date-time strings** with custom formats.

- Specify the format using format codes (e.g., %Y, %m, %d, %H, %M, %S) in the orders argument.

```
> # Parsing Custom Date-Time Strings
> date_string <- "2023.10.26 14:30:00"
> format <- "%Y-%m-%d %H:%M:%S"
> parse_date_time(date_string, orders = format)
[1] "2023-10-26 14:30:00 UTC"
```

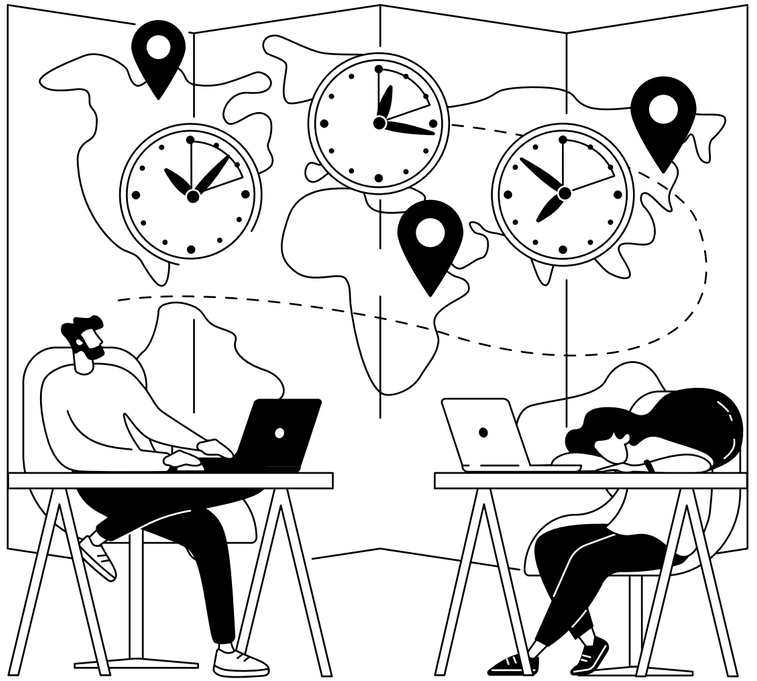
Parsing Dates and Times with Lubridate

4. Time zones

Time zones exist due to the Earth's rotation, and they are defined as **regions of the Earth that share the same standard time.**

- We can use function OlsonNames() to explore the approximately 600 time zones.

```
> OlsonNames()
  [1] "Africa/Abidjan"
  [2] "Africa/Accra"
  [3] "Africa/Addis_Ababa"
  [4] "Africa/Algiers"
  [5] "Africa/Asmara"
  [6] "Africa/Asmera"
```



Parsing Dates and Times with Lubridate

4. Time zones

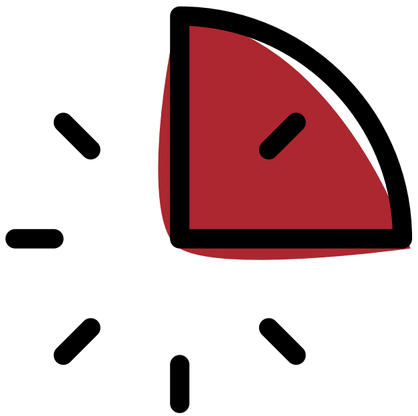
Lubridate provides two helper functions for **handling time zones:**

- The **with_tz() function** simply **changes the time zone** in which the current time is printed.
- The **force_tz() function** force moving the time to a different time zone.

```
> # Handle Time Zones
> # Get the date and time rigth now
> time <- now()
> time
[1] "2023-10-27 10:18:40 CEST"
> # Display the current time in America/Chicago Time (CDT)
> with_tz(time, "America/Chicago")
[1] "2023-10-27 03:18:40 CDT"
> # Force the current time to America/Chicago Time (CDT)
> force_tz(time, "America/Chicago")
[1] "2023-10-27 10:18:40 CDT"
```

Time Spans with Lubridate: Durations, Periods, and Intervals

Lubridate also **expands the type of mathematical operations** that can be performed with date-time objects, using three time span classes:



- **Durations:** represent an exact amount of time, typically measured in hours, minutes, seconds, and fractions of a second.
- **Periods:** represents a time span in terms of years, months, days, hours, minutes, and seconds
- **Intervals:** represents the time span between two specific date-time points.

Time Spans with Lubridate: Durations, Periods, and Intervals

1. Durations

The duration() function can be used to create a Duration object and perform operations.

```
> # Durations
> # Creating a duration of 2 hours, 30 minutes, and 45 seconds
> dur <- duration(hours = 2, minutes = 30, seconds = 45)
> dur
[1] "9045s (~2.51 hours)"
> # Adding a duration to a date-time object
> start_time <- ymd_hms("2023-10-26 08:00:00")
> end_time <- start_time + dur
> end_time
[1] "2023-10-26 10:30:45 UTC"
```

Time Spans with Lubridate: Durations, Periods, and Intervals

2. Periods

The period() function is used to create a period object, which represents a time span in terms of years, months, days, hours, minutes, seconds, and other clock-based units.

```
> # Periods
> # Creating a period of 2 years, 5 months, and 15 days
> per <- period(years = 2, months = 5, days = 15)
> per
[1] "2y 5m 15d 0H 0M 0S"
> # Adding a period to a date-time object
> start_date <- ymd("2023-01-01")
> end_date <- start_date + per
> end_date
[1] "2025-06-16"
```

Time Spans with Lubridate: Durations, Periods, and Intervals

3. Intervals

An interval represents the **time span between two specific date-time points**. This can be useful for calculating the duration between two events, summarizing the time information between them, or comparing and performing operations with date-time objects.

```
> # Intervals
> # Creating an interval between two date-time objects
> start_datetime <- ymd_hms("2023-10-26 08:00:00")
> end_datetime <- ymd_hms("2023-10-26 14:30:00")
> time_span <- interval(start_datetime, end_datetime)
> time_span
[1] 2023-10-26 08:00:00 UTC--2023-10-26 14:30:00 UTC
```

Time Spans with Lubridate: Durations, Periods, and Intervals

- Use dminutes(), dhours(), dyears(), and similar functions to parse durations, and months(), years(), quarters(), etc., to parse periods.

```
> # Parsing duratons
> duration_example <- dhours(2) + dminutes(30)
> duration_example
[1] "9000s (~2.5 hours)"
> period_example <- months(2) + days(15)
> period_example
[1] "2m 15d 0H 0M 0S"
```

Resources

- Dates and times made easy with lubridate
- Lubridate CheatSheet
- R for Data Science, Chapter 16: Dates and times
- Learn how to work with dates and times using the package 'lubridate'.

Workshop Dates and Times with Lubridate

Thank you!

Hertie School