Foundations of Data Science for Biologists

Dates and times

BIO 724D

2024-NOV-25

Instructors: Greg Wray and Paul Magwene

Challenges in working with dates and times

Many different representations

November 25th, 2024 25 Nov 2024 24-11-25 *and many more*

3:15pm 15:15 15h 15m *and many more*

Requires conversion to a standard format, can easily produce errors

Non-decimal units

1000 msec / sec 60 sec / min 24 hr / day ? day / month ~365 day / year

Leap years and leap seconds add exceptions in certain situations

Makes math with dates and times complicated

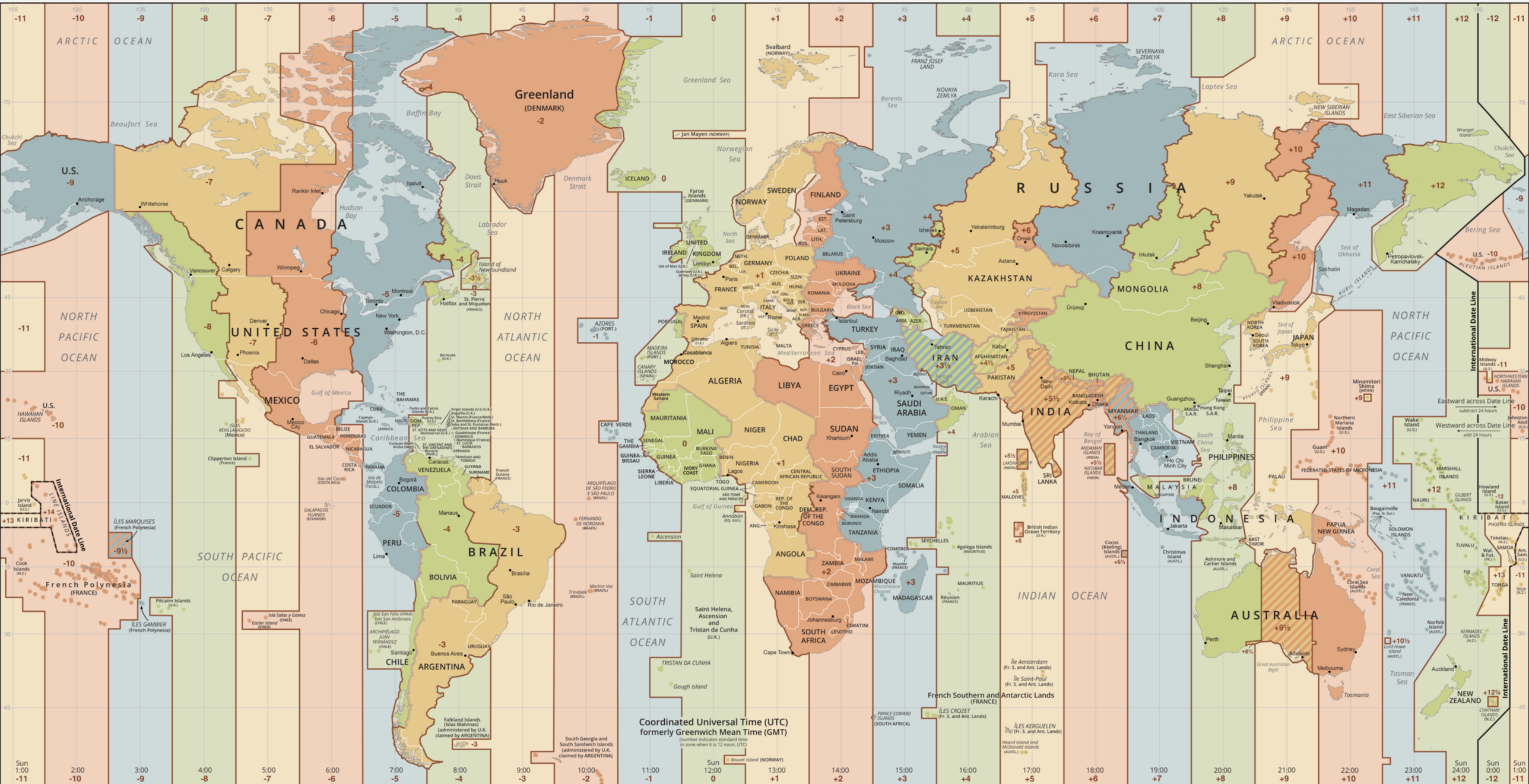
Time zones

No regular organization, many odd exceptions, local standards can change

Daylight savings time introduces further exceptions in certain situations

Complicates coordination of dates and times, and further complicates math

Time zones of the world



How do computers store dates and times?

Computers store a single date-time value and your local time zone

- The date-time value is very accurate and precise
- Local time zone is automatically or manually updated if you move to a different place

Date-time is stored in units of seconds

- The value is the time elapsed since 1970-01-01 00:00:00 UTC
- UTC (universal coordinated time) is longitude 0 (Greenwich, England)
- Values are only converted to local date and local time for humans as needed

Computers are not designed to be clocks

- Eventually, the internal date-time value will drift from true time
- The OS uses NTP (network time protocol) to synchronize the value with a time server

Date and date-time data types

Value: seconds (OS) or string (data objects)

Representation: string

Format: ISO 8601

Date

Representation example: `"2024-11-26"`

Data type: `"Date"`

Date-time

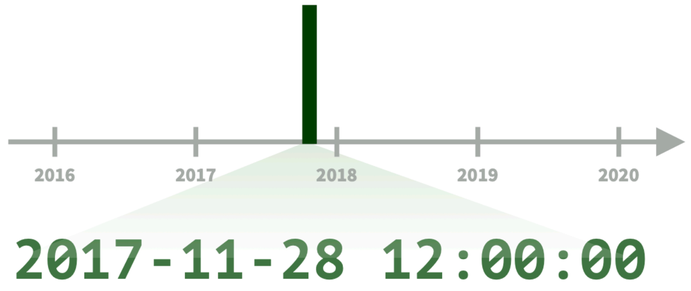
Representation example: `"2024-11-26 15:30:22 UTC”`

Data type: `"POSIXct"`

Time

No time-only data type in base R

`lubridate` provides 3 different **time span** data types



date and date-time are unique points along the time-line

time alone is not a unique point along the time-line

Outline for dates and times

Introducing the `lubridate` package

Date and date-time objects

- How to create date and date-time objects
- How to retrieve the current date and date-time for anywhere in the world
- How to convert existing non-ISO 8601 strings into proper date and date-time objects
- How to extract and modify specific components of a date or date-time object

Time spans

- The three time span classes supported by `lubridate` and their use cases
- How to create each class of time span object
- How to modify and carry out mathematical operations with time spans

Time span classes: duration

Definition

A time span defined only by its length

Not anchored to the time-line, an abstract time span

Think of it as: “stopwatch time”

Computationally

Stored as: numeric

Representation: `"79200s (~22 hours)”` , `"5702400s (~9.43 weeks)"`

Ignores the calendar when used in math with date-time objects

Uses

You want to record how long one baboon spends grooming another baboon

You want to compare how long it takes wild type and mutant cells to divide

You want to record time and do math in absolute units and ignore the calendar

Time span classes: period

Definition

A time span defined only by its length

Aware of time-line irregularities

Think of it as: “calendar time”

Computationally

Stored as: numeric from a defined start date-time

Representation: `“2M 0S”`, `"5d 4H 12M 47S"`

Consults the calendar when used in math with date-time objects (“intuitive” results)

Period

You are taking field observations every 9 days and want to know the next time point

You are planning a 5-week experiment that requires time points every 30 hours

You want to do date-time math using absolute units, but get results in human units

Time span classes: interval

Definition

A time span defined by its end points

Anchored to 2 date(-time)s, aware of time-line irregularities

Think of it as: “bounded time”

Computationally

Stored as: start date(-time) and end date(-time)

Representation: `2024-05-01 12:00:00 UTC--2024-05-01 15:32:17 UTC`

Time difference and other attributes can be extracted with functions

Interval

You have two events recorded in human units and want know the time difference

You want to know if a specific date-time falls within a defined interval

You want to work with time spans defined by their end points rather than duration