08: Dates

Graphical Analysis of Biological Data

Dates and time

Carefully read R4ds Chapter 16: Dates and times sections 1-3. I suggest you run the examples but I am not requiring that you do so. The assignment will require you to use some of the functions.

You will need to use the `lubridate` package. I believe it was installed when you installed the `tidyverse` package but `lubridate` is not loaded when you load the `tidyverse` library. You will have to load `lubridate` separately. If you get an error, then install it first.

As you will learn from the text, dates are tricky, in part because of the many ways in which dates are formatted. In the US, we tend to use MM/DD/YY, such as 11/19/1863. Europeans tend to use a format that switches the order of the month and year, like 19/11/1863. Other variations are common, such as 19 November 1863, Nov. 11th, 1863, and 19-11-63. If a date is given as 11/5/1605, is it November 5th or May 11th?

Different date formats have been developed for computers, but R and `lubridate` follow the ISO 8601 format of YYYY-MM-DD, e.g., 1863-11-19. However, this is just a way of representing the date, but not how dates are stored internally. Usually, dates are stored as the number of days since some particular reference point. For example, some UNIX systems use 1970-01-01 00:00:00 (hour minute seconds) as the reference point.

Computer systems also have to make assumptions about the century if it is not explicitly stated. For example, load lubridate and run this code.

```
mdy(c("1/2/54","12/31/68","1/1/69","1/2/99","1/2/04"))
```

```
## [1] "2054-01-02" "2068-12-31" "1969-01-01" "1999-01-02" "2004-01-02"
```

We would naturally assume that the second and third dates in the vector are only one day apart. However, the POSIXct standard assumes that years less than or equal to "68" belong to the 21st century and years greater than 68 belong to the 20th century.

Despite the complexities of dates (and time), lubridate date provides several convenience functions for wrangling dates. Lubridate can convert date strings, such as "19 November 1863" and "11/19/1863" to date the proper ISO formats using the `dmy()` and `mdy()` functions. Notice the names of the functions. You can change the order of `d`, `m`, and `y` to convert nearly any date string to the ISO format.

For example,

```
# Attestation clause, witnessing the U.S. constitution.
# Actual text spells out all of the numbers so here modified so it works.
attestation <- "17th Day of September in the Year of Our Lord 1787"

dmy(attestation)
```

```
## [1] "1787-09-17"
```

You can use `make_date` to assemble dates from numeric components, as shown here.

```
gf <- tibble(Day = 5, Month = 11, Year = 1605)

gf <- gf %>% mutate(Date = make_date(Year, Month, Day))
gf
```

```
## # A tibble: 1 x 4
##     Day Month  Year Date
##   <dbl> <dbl> <dbl> <date>
## 1     5    11  1605 1605-11-05
```

```r
# Date stored as date format.
str(gf)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    1 obs. of  4 variables:
##  $ Day  : num 5
##  $ Month: num 11
##  $ Year : num 1605
##  $ Date : Date, format: "1605-11-05"
```

You can extract months, weekdays, and more, too. Notice that the months are treated as ordered factors. Factored months ensures the months will appear in the proper order in plots.

```r
# Short month name
month(gf$Date, label = TRUE)
```

```
## [1] Nov
## 12 Levels: Jan < Feb < Mar < Apr < May < Jun < Jul < Aug < Sep < ... < Dec
```

```r
# Full month name
month(gf$Date, label = TRUE, abbr = FALSE)
```

```
## [1] November
## 12 Levels: January < February < March < April < May < June < ... < December
```

I've touched on some but not all of the functions you will need to complete this assignment. Read chapter 16, sections 1-3, then do the assignment.