Durations, intervals and periods

Packages for this section

```
library(tidyverse)
library(lubridate)
```

Exact time intervals

We previously got fractional days (of stays in hospital):

```
my_url <- "http://www.utsc.utoronto.ca/~butler/c32/hospital.cs
stays <- read_csv(my_url)

## Parsed with column specification:
## cols(
##   admit = col_datetime(format = ""),
##   discharge = col_datetime(format = "")
## )
```

```
stays %>% mutate(stay_days = (discharge - admit) / ddays(1))
```

admit	discharge	stay_days
1981-12-10 22:00:00	1982-01-03 14:00:00	23.666667
2014-03-07 14:00:00	2014-03-08 09:30:00	0.812500
2016-08-31 21:00:00	2016-09-02 17:00:00	1.833333

Intervals

```
stays %>% mutate(stay = admit %--% discharge)
```

admit	discharge	stay
1981-12-10 22:00:00	1982-01-03 14:00:00	1981-12-10 22:00:00 UTC–1982-01-03 14:00:00 UTC
2014-03-07 14:00:00	2014-03-08 09:30:00	2014-03-07 14:00:00 UTC–2014-03-08 09:30:00 UTC
2016-08-31 21:00:00	2016-09-02 17:00:00	2016-08-31 21:00:00 UTC–2016-09-02 17:00:00 UTC

- These are called *intervals*: they have a start point and an end point.

Periods

To work out the exact length of an interval, in human units, turn it into a `period`:

```
stays %>% mutate(stay = as.period(admit %--% discharge))
```

admit	discharge	stay
1981-12-10 22:00:00	1982-01-03 14:00:00	23d 16H 0M 0S
2014-03-07 14:00:00	2014-03-08 09:30:00	19H 30M 0S
2016-08-31 21:00:00	2016-09-02 17:00:00	1d 20H 0M 0S

A period is exact as long as it has a start and an end (accounting for daylight savings, leap years etc).

Completed days

Take day of the periods:

```
stays %>% mutate(stay = as.period(admit %--% discharge)) %>%
  mutate(days_of_stay = day(stay))
```

admit	discharge	stay	days_of_sta
1981-12-10 22:00:00	1982-01-03 14:00:00	23d 16H 0M 0S	2
2014-03-07 14:00:00	2014-03-08 09:30:00	19H 30M 0S	
2016-08-31 21:00:00	2016-09-02 17:00:00	1d 20H 0M 0S	

Completed hours 1/2

- Not quite what you think:

```
stays %>% mutate(stay = as.period(admit %--% discharge)) %>%
  mutate(hours_of_stay = hour(stay))
```

admit	discharge	stay	hours_of_stay
1981-12-10 22:00:00	1982-01-03 14:00:00	23d 16H 0M 0S	16
2014-03-07 14:00:00	2014-03-08 09:30:00	19H 30M 0S	19
2016-08-31 21:00:00	2016-09-02 17:00:00	1d 20H 0M 0S	20

- These are completed hours *within* days.

Completed hours 2/2

- To get total hours, count each day as 24 hours also:

```
stays %>% mutate(stay = as.period(admit %--% discharge)) %>%
  mutate(hours_of_stay = hour(stay) + 24*day(stay))
```

admit	discharge	stay	hours_of_stay
1981-12-10 22:00:00	1982-01-03 14:00:00	23d 16H 0M 0S	568
2014-03-07 14:00:00	2014-03-08 09:30:00	19H 30M 0S	19
2016-08-31 21:00:00	2016-09-02 17:00:00	1d 20H 0M 0S	44

Durations

- What's the difference between `duration` and `period`?

```
stays %>% mutate(stay = as.duration(admit %--% discharge))
```

admit	discharge	stay
1981-12-10 22:00:00	1982-01-03 14:00:00	2044800s (~3.38 weeks)
2014-03-07 14:00:00	2014-03-08 09:30:00	70200s (~19.5 hours)
2016-08-31 21:00:00	2016-09-02 17:00:00	158400s (~1.83 days)

- A duration is always a number of *seconds*.

Sometimes it matters

- Days and hours are always the same length (as a number of seconds).
- Months and years are not always the same length:
  - months have different numbers of days
  - years can be leap years or not
  - the actual length of 2 months depends *which* 2 months:

```
tribble(
  ~start, ~end,
  ymd("2020-01-15"), ymd("2020-03-15"),
  ymd("2020-07-15"), ymd("2020-09-15")
) %>% mutate(period = as.period(start %--% end)) %>%
  mutate(duration = as.duration(start %--% end))
```

start	end	period	duration
2020-01-15	2020-03-15	2m 0d 0H 0M 0S	5184000s (~8.57 weeks)
2020-07-15	2020-09-15	2m 0d 0H 0M 0S	5356800s (~8.86 weeks)

Manchester United

Sometime in December 2019 or January 2020, I downloaded some information about the players that were then in the squad of the famous Manchester United Football (soccer) Club. We are going to use the players' ages (as given) to figure out exactly when the download happened.

```
my_url <- "http://ritsokiguess.site/STAD29/manu.csv"
read_csv(my_url) %>%
  select(name, date_of_birth, age) -> man_united
```

```
## Parsed with column specification:
## cols(
##   number = col_double(),
##   name = col_character(),
##   nationality = col_character(),
##   date_of_birth = col_character(),
##   age = col_double(),
##   country_of_birth = col_character(),
##   place_of_birth = col_character(),
##   position = col_character()
```

The data

```
man_united
```

name	date_of_birth	age
David de Gea Quintana	7 November 1990	29
Lee Grant	27 January 1983	36
Sergio Germán Romero	22 February 1987	32
Victor Nilsson Lindelöf	17 July 1994	25
Eric Bertrand Bailly	12 April 1994	25
Phil Jones	21 February 1992	27
Harry Maguire	5 March 1993	26
Faustino Marcos Alberto Rojo	20 March 1990	29
Ashley Young	9 July 1985	34
José Diogo Dalot Teixeira	18 March 1999	20
Luke Shaw	12 July 1995	24
Timothy Evans Fosu-Mensah	2 January 1998	21
Aaron Wan-Bissaka	26 November 1997	22

Ages

- A player's age is the number of *completed* years since their birth
- This suggests:
  - guessing a download date
  - working out time since birth as *period*
  - extracting number of years
- After that, see if our calculations of age match actual ages

Guess download date

Guess January 10, 2020 as download date (just to pick a date):

```
guess <- ymd("2020-01-10")
man_united %>%
  mutate(dob = dmy(date_of_birth)) %>%
  mutate(per = as.period(dob %--% guess)) %>%
  mutate(calc_age = year(per)) %>%
  select(name, age, calc_age)
```

name	age	calc_age
David de Gea Quintana	29	29
Lee Grant	36	36
Sergio Germán Romero	32	32
Victor Nilsson Lindelöf	25	25
Eric Bertrand Bailly	25	25
Phil Jones	27	27
Harry Maguire	26	26

Fractional years

- guess a date for when I downloaded this,
- turn date of birth into a date,
- work out fractional years:

```
dl_date <- ymd("2020-01-05")
man_united %>%
  mutate(date_of_birth = dmy(date_of_birth)) %>%
  mutate(years = (dl_date - date_of_birth) / dyears(1))
```

name	date_of_birth	age	years
David de Gea Quintana	1990-11-07	29	29.16085
Lee Grant	1983-01-27	36	36.93908
Sergio Germán Romero	1987-02-22	32	32.86790
Victor Nilsson Lindelöf	1994-07-17	25	25.47023
Eric Bertrand Bailly	1994-04-12	25	25.73306
Phil Jones	1992-02-21	27	27.87132
Harry Maguire	1993-03-05	26	26.82641

See which ages were gotten wrong

`floor` gets the integer part of a decimal number:

```
man_united %>%
  mutate(date_of_birth = dmy(date_of_birth)) %>%
  mutate(years = (dl_date - date_of_birth) / dyears(1)) %>%
  filter(floor(years) != age)
```

name	date_of_birth	age	years
Timothy Evans Fosu-Mensah	1998-01-02	21	22.00684
Jesse Lingard	1992-12-15	26	27.05544
Andreas Hoelgebaum Pereira	1996-01-01	23	24.01095

- These three players had a birthday between the date I downloaded the data and the date I guessed.

```
man_united %>%
  mutate(date_of_birth = dmy(date_of_birth)) %>%
```

going back

- what if I go back to December 1?

```
dl_date <- ymd("2019-12-01")
man_united %>%
  mutate(date_of_birth = dmy(date_of_birth)) %>%
  mutate(calc_age = as.period(date_of_birth %--% dl_date)) %>%
  mutate(yrs = year(calc_age)) %>%
  filter(age != yrs)
```

name	date_of_birth	age	calc_age	yrs
Scott McTominay	1996-12-08	23	22y 11m 23d 0H 0M 0S	22
Anthony Martial	1995-12-05	24	23y 11m 26d 0H 0M 0S	23
Tahith Chong	1999-12-04	20	19y 11m 27d 0H 0M 0S	19

notes

- fractional years get all the ages right
- so there is no advantage to using a period