

Durations, intervals, and periods

## Packages for this section

```
library(tidyverse)
```

Dates and times live in a package called `lubridate`, but this is now part of the `tidyverse`.

## Exact time intervals

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

```
my_url <- "http://ritsokiguess.site/datafiles/hospital.csv"
stays <- read_csv(my_url)
stays %>% mutate(stay_days = (discharge - admit) / ddays(1))
```

```
# A tibble: 3 x 3
```

	admit		discharge		stay_days
	<dtm>		<dtm>		<dbl>
1	1981-12-10 22:00:00		1982-01-03 14:00:00		23.7
2	2014-03-07 14:00:00		2014-03-08 09:30:00		0.812
3	2016-08-31 21:00:00		2016-09-02 17:00:00		1.83

but what if we wanted days, hours and minutes?

# Intervals

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

```
# A tibble: 3 x 3
```

	admit <dtm>	discharge <dtm>	stay <Interval>
1	1981-12-10 22:00:00	1982-01-03 14:00:00	1981-12-10 22:00:00 - 1982-01-03 14:00:00
2	2014-03-07 14:00:00	2014-03-08 09:30:00	2014-03-07 14:00:00 - 2014-03-08 09:30:00
3	2016-08-31 21:00:00	2016-09-02 17:00:00	2016-08-31 21:00:00 - 2016-09-02 17:00:00

- ▶ 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))
```

```
# A tibble: 3 x 3
```

	admit		discharge		stay
	<dtm>		<dtm>		<Period>
1	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
3	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))
```

# A tibble: 3 x 4

	admit <dtm>	discharge <dtm>	stay <Period>	days_of_ <
1	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	
3	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))
```

```
# A tibble: 3 x 4
```

	admit		discharge		stay	hours_of
	<dtm>		<dtm>		<Period>	
1	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	
3	2016-08-31 21:00:00		2016-09-02 17:00:00		1d 20H 0M 0S	

► 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))
```

```
# A tibble: 3 x 4
```

	admit <dtm>	discharge <dtm>	stay <Period>	hours_of
1	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	
3	2016-08-31 21:00:00	2016-09-02 17:00:00	1d 20H 0M 0S	



## Durations

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

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

```
# A tibble: 3 x 3
```

	admit <dtm>	discharge <dtm>	stay <Duration>
1	1981-12-10 22:00:00	1982-01-03 14:00:00	2044800s (~3.38 w)
2	2014-03-07 14:00:00	2014-03-08 09:30:00	70200s (~19.5 hou)
3	2016-08-31 21:00:00	2016-09-02 17:00:00	158400s (~1.83 da)

- ▶ A duration is always a number of *seconds*.
- ▶ Also shown is an approx equivalent on a more human scale (calculated from 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))
```

```
# A tibble: 2 x 4
```

	start	end	period	duration
	<date>	<date>	<Period>	<Duration>
1	2020-01-15	2020-03-15	2m 0d 0H 0M 0S	5184000s (~8.57 weeks)
2	2020-07-15	2020-09-15	2m 0d 0H 0M 0S	5356800s (~8.86 weeks)

## Comments

- ▶ Both periods are exactly two months
- ▶ but they have a different duration in seconds
- ▶ the first two-month period is shorter because it contains the short month February
- ▶ the second two-month period is longer because both July and August have 31 days.

# 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/datafiles/manu.csv"  
read_csv(my_url) %>%  
  select(name, date_of_birth, age) -> man_united
```

# The data

```
man_united
```

```
# A tibble: 29 x 3
```

	name <chr>	date_of_birth <chr>	age <dbl>
1	David de Gea Quintana	7 November 1990	29
2	Lee Grant	27 January 1983	36
3	Sergio Germán Romero	22 February 1987	32
4	Victor Nilsson Lindelöf	17 July 1994	25
5	Eric Bertrand Bailly	12 April 1994	25
6	Phil Jones	21 February 1992	27
7	Harry Maguire	5 March 1993	26
8	Faustino Marcos Alberto Rojo	20 March 1990	29
9	Ashley Young	9 July 1985	34
10	José Diogo Dalot Teixeira	18 March 1999	20

```
# i 19 more rows
```

# 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 and work out ages

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(age_period = as.period(dob %--% guess)) %>%
  mutate(age_years = year(age_period)) -> d
```

## Results (just the ages)

```
d %>% select(name, age, age_years)
```

```
# A tibble: 29 x 3
```

	name	age	age_years
	<chr>	<dbl>	<dbl>
1	David de Gea Quintana	29	29
2	Lee Grant	36	36
3	Sergio Germán Romero	32	32
4	Victor Nilsson Lindelöf	25	25
5	Eric Bertrand Bailly	25	25
6	Phil Jones	27	27
7	Harry Maguire	26	26
8	Faustino Marcos Alberto Rojo	29	29
9	Ashley Young	34	34
10	José Diogo Dalot Teixeira	20	20

```
# i 19 more rows
```



## Which ones are different?

```
d %>% filter(age != age_years) %>%  
  select(name, date_of_birth, age, age_years)
```

```
# A tibble: 3 x 4
```

	name	date_of_birth	age	age_years
	<chr>	<chr>	<dbl>	<dbl>
1	Timothy Evans Fosu-Mensah	2 January 1998	21	22
2	Jesse Lingard	15 December 1992	26	27
3	Andreas Hoelgebaum Pereira	1 January 1996	23	24

- ▶ these three players were calculated wrong: we got one year too many.
- ▶ Our guessed date, January 10, was too *late*.
- ▶ These three players had a birthday since the actual download date
- ▶ actual download date must have been before Dec 15.

## Try an earlier date

► say Dec 5:

```
guess <- ymd("2019-12-05")
man_united %>%
  mutate(dob = dmy(date_of_birth)) %>%
  mutate(age_period = as.period(dob %--% guess)) %>%
  mutate(age_years = year(age_period)) %>%
  filter(age != age_years) %>%
  select(name, date_of_birth, age, age_years) -> d2
```

## Results

```
d2
```

```
# A tibble: 1 x 4
```

	name	date_of_birth	age	age_years
	<chr>	<chr>	<dbl>	<dbl>
1	Scott McTominay	8 December 1996	23	22

- ▶ Dec 5 was too early for the download date
- ▶ must have been later than Dec 8 (to get McTominay's age right)
- ▶ so must have been between Dec 8 and Dec 15 (Lingard's birthday)
- ▶ Actually I downloaded the data on Dec 10.