Temporal Data

We will deal with time data by creating a new type of column.

	date	datetime	hms (time)
CREATE	ymd('2021-10-10') make_date(2021, 10, 10) as_date(<datetime>)	ymd_hms('2021-10-10 03:50') make_datetime(2021, 10, 10, 3, 50) as_datetime(<date>|<hms> | <int>)	hms('03:50:00') as_hms(<datetime>)
MODIFY	+/- 1 (back/ahead by 1 day)	+/- 1 (back/ahead by 1 second) floor_date(), round_date()	+/- 1 (back/ahead by 1 second)
EXTRACT	year(), month(), day(), isoweek(), wday()	+ hour(), minute(), second()	+ hour(), minute(), second()

Loading Temporal Data

When R reads in data from a CSV file that has dates or date times stored in a ISO 8601 format (such as YYYY-MM-DD or YYYY-MM-DD HH:MM), it will automatically create a <date> or <dttm> data type.

Another common way to store time data is as an integer representing Unix time. This is the number of seconds since 1 January 1970. These won't be automatically converted; use **as_datetime()**. Often JSON data will contain Unix time but in milliseconds. Just divide by 1000 before using the function.

Hand constructed datasets may have dates in other weird formats. The functions ymd() and varients can handle most things that are consistent. Otherwise some tricky data cleaning with stringi may be needed.

Plotting Temporal Data

If you put a date, datetime, or time object as a x- or y-aesthetic in a plot, it will generally work as you expect. However, as with color in spatial plots, you will often want to modify the default scales to make the plot easier to read. This can be done with (_y_ version exist as well):

```
scale_x_date()
scale_x_datetime()
scale_x_time()
```

The first two have options date_breaks, date_labels, and date_minor_breaks that accept easy-to-use options (Note: it is always "date", never "datetime").

Times do not have the same flexibility. I find it best to use as_datetime() to squish the times into a single day and plot that instead.