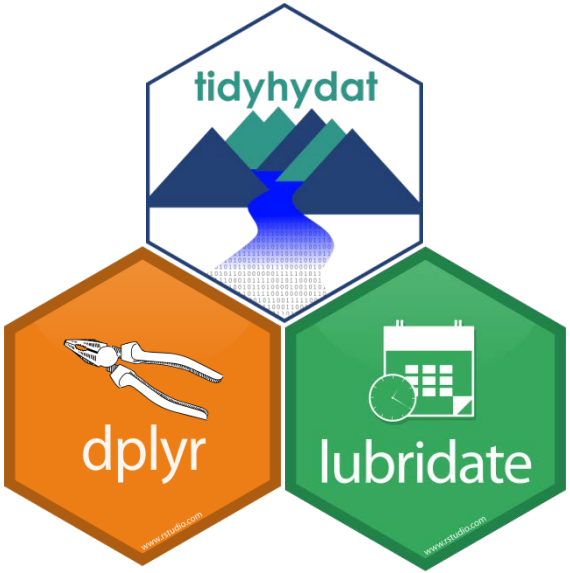
Working with Dates and Joins



Dates and Times

lubridate

Functions for working with dates and time spans

```
# install.packages("tidyverse")
library(lubridate)
```

ymd() family

To parse strings as dates, use a y, m, d, h, m, s combo

```
ymd("2017/01/11")
mdy("January 11, 2017")
ymd_hms("2017-01-11  01:30:55")
```

Parsing functions

function	parses to
ymd_hms(), ymd_hm(), ymd_h() ydm_hms(), ydm_hm(), ydm_h() dmy_hms(), dmy_hm(), dmy_h() mdy_hms(), mdy_hm(), mdy_h()	POSIXct
ymd(), ydm(), mdy() myd(), dmy(), dym(), yq()	Date (POSIXct if tz specified)
hms(), hm(), ms()	Period

Your turn

For each of the following formats (of the same date), pick the right `ymd()` function to parse them:

- "2018 Feb 01"
- "2-1-18"
- "01/02/2018"

```
ymd("2018 Feb 02")
# [1] "2018-02-02"

mdy("2-1-18")
# [1] "2018-02-01"

dmy("01/02/2018")
# [1] "2018-02-01"
```

```
fraser_flow <- hy_daily_levels(STATION_NUMBER == "08MF005"))
```

```
No start and end dates specified. All dates available will be returned.
All station successfully retrieved
# A tibble: 37,075 x 5
   STATION_NUMBER Date       Parameter Value Symbol
   <chr>          <date>     <chr>     <dbl> <chr>
 1 08MF005        1912-03-01 Level        NA NA
 2 08MF005        1912-03-02 Level        NA NA
 3 08MF005        1912-03-03 Level        NA NA
 4 08MF005        1912-03-04 Level        NA NA
 5 08MF005        1912-03-05 Level        NA NA
 6 08MF005        1912-03-06 Level      3.05 NA
 7 08MF005        1912-03-07 Level      2.96 NA
 8 08MF005        1912-03-08 Level      2.96 NA
 9 08MF005        1912-03-09 Level      2.93 NA
10 08MF005        1912-03-10 Level      2.93 NA
# ... with 37,065 more rows
```

Accessing components

Accessing components

Extract components by name with a **singular** name

```
date <- ymd("2018-02-01")
year(date)
## 2018
```

Accessing components

function	extracts	extra arguments
year()	year	
month()	month	label = FALSE, abbr = TRUE
week()	week	
day()	day of month	
wday()	day of week	label = FALSE, abbr = TRUE
qday()	day of quarter	
yday()	day of year	
hour()	hour	
minute()	minute	
second()	second	

Your turn

Fill in the blanks to for `fraser_flows` data:

- Extract the month from date.
- Extract the year from date.
- Calculate the mean flow with a measurement for each year/month.
- Filter the results for only the month of June

```
fraser_flow %>%
    mutate(year = year(Date),
           month = month(Date)) %>%
    group_by(month, year) %>%
    summarise(mean_monthly_flow = mean(Value, na.rm = TRUE)) %>%
    filter(month == 6)
```

realtime data – dealing with time

```
mackenzie_realtime <- realtime_dd(STATION_NUMBER == "10LC014"))
```

```
# A tibble: 17,536 x 8
   STATION_NUMBER PROV_TERR_STATE_LOC Date                Parameter Value Grade Symbol Code
   <chr>          <chr>               <dttm>              <chr>     <dbl> <chr> <chr>  <chr>
 1 10LC014        NT                  2018-04-17 07:00:00 Flow       2800 NA    NA     1
 2 10LC014        NT                  2018-04-17 07:05:00 Flow       2800 NA    NA     1
 3 10LC014        NT                  2018-04-17 07:10:00 Flow       2810 NA    NA     1
 4 10LC014        NT                  2018-04-17 07:15:00 Flow       2810 NA    NA     1
 5 10LC014        NT                  2018-04-17 07:20:00 Flow       2800 NA    NA     1
 6 10LC014        NT                  2018-04-17 07:25:00 Flow       2800 NA    NA     1
 7 10LC014        NT                  2018-04-17 07:30:00 Flow       2810 NA    NA     1
 8 10LC014        NT                  2018-04-17 07:35:00 Flow       2810 NA    NA     1
 9 10LC014        NT                  2018-04-17 07:40:00 Flow       2800 NA    NA     1
10 10LC014        NT                  2018-04-17 07:45:00 Flow       2800 NA    NA     1
# ... with 17,526 more rows
```

realtime data – dealing with time

```
mackenzie_realtime %>%
    mutate(seconds = second(Date),
           minute = minute(Date),
           hour = hour(Date)) %>%
    filter(hour == 7)
```

realtime data – dealing with time

```
mackenzie_realtime %>%
    mutate(seconds = second(Date),
           minute = minute(Date),
           hour = hour(Date)) %>%
    filter(hour == 7)
```

or

```
mackenzie_realtime %>%
    filter(hour(Date) == 7)
```

filter() and dates

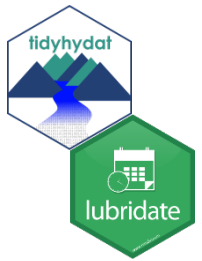
Subsetting with time

Extract rows that meet a certain time criteria

- Have to create a logical test

```
> "01-01-1950"
[1] "01-01-1950"

> dmy("01-01-1950")
[1] "1950-01-01"
```



Subsetting with time

Extract rows that meet a certain time criteria

- Have to create a logical test

```
fraser_flow %>%
    filter(Date >= "01-01-1950")

fraser_flow %>%
    filter(Date >= dmy("01-01-1950"))
```

Challenge

Compute the average flow for every discharge station in Nunavut for August using dates only after the year 1960

Hint

Use tidyhydat function hy_stations() and hy_daily_flows()

Challenge

```
hy_daily_flows(prov_terr_state_loc = "NU") %>%
    mutate(Year = year(Date), Month = month(Date, label = TRUE)) %>%
    filter(Month == "Aug") %>%
    group_by(STATION_NUMBER, Year, Month) %>%
    summarise(mean_flow = mean(Value, na.rm = TRUE))
```

Joining datasets

In R4DS

Relational Data

Joins

Mutating joins use information from one data set to add variables to another data set (like mutate())

Filtering joins use information from one data set to extract cases from another data set (like filter())

Common Syntax

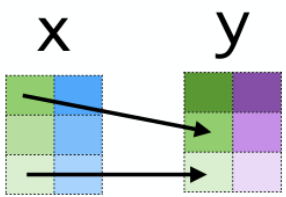
Each join function returns a data frame / tibble.

```
left_join(x, y, by = NULL, ... )
```


```
x %>% left_join(y, by = NULL, ... )
```

In pipe form

Two table verbs

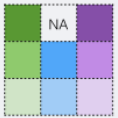


Mutating joins

Columns from x and y

All rows in x

`x %>% left_join(y)`



All rows in y

`x %>% right_join(y)`

Only rows in x with matches in y

`x %>% inner_join(y)`

All rows from x and y

`x %>% full_join(y)`

Filtering joins

Columns from x

Rows in x that have matches in y

`x %>% semi_join(y)`

Rows in x that don't have matches in y

`x %>% anti_join(y)`

left_join

Join metadata to streamflow information

```
bc_stations <- hy_stations(prov_terr_state_loc = "BC") %>%
     select(STATION_NUMBER, STATION_NAME)

bc_mad <- hy_annual_stats(prov_terr_state_loc = "BC") %>%
     filter(Sum_stat == "MEAN")

bc_stations %>%
     left_join(bc_mad, by = c("STATION_NUMBER"))
```

Your turn

Find the Unit flow (Flow per watershed area) for all BC stations for all years

left_join

Find the Unit flow

```
bc_stations_drainage_area <- hy_stations(prov_terr_state_loc = "BC") %>%
    select(STATION_NUMBER, DRAINAGE_AREA_GROSS)

hy_annual_stats(prov_terr_state_loc = "BC") %>%
    filter(Sum_stat == "MEAN", Parameter == "Flow") %>%
    left_join(bc_stations_drainage_area, by = c("STATION_NUMBER")) %>%
    mutate(unit_flow = Value/DRAINAGE_AREA_GROSS) %>%
    select(STATION_NUMBER, Parameter, Year, Value, DRAINAGE_AREA_GROSS,
    unit_flow)
```