

Homework: Lubridate and Purrr

Jesse Brandt and Collin Brown

2025-02-25

```
library(tidyverse)

## -- Attaching core tidyverse packages ----- tidyverse 2.0.0 --
## v dplyr      1.1.4      v readr      2.1.5
## v forcats    1.0.0      v stringr   1.5.1
## v ggplot2    3.5.1      v tibble    3.2.1
## v lubridate  1.9.4      v tidyr     1.3.1
## v purrr      1.0.2
## -- Conflicts ----- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()     masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

Exercise 1 Question 1

Generate a sequence of dates from January 1, 2015 to December 31, 2025, spaced by every two months. Extract the year, quarter, and ISO week number for each date.

Interpretation: every 2 months starting January 1, 2015. This will not include December 31, 2025. The last date within the date range is November 1, 2025.

```
start <- mdy("01/01/2015") #start date
end <- mdy("12/31/2025") # end date
intrv <- interval(start, end) #interval between start and end date
period <- (as.period(intrv)) #convert to period

months_total <- (12*period@year + period@month) #find number of full months in period
#this is simple arithmetic, but doing it this way will make it possible to do this for any 2 dates

date <- map_vec(seq(0, months_total, by = 2), \(x)start + months(x)) #create date vector
year <- map_int(date, ~year(.)) #Create year column

df <- (data.frame(date, year)) %>% #bind those two columns together
  mutate(quarter = quarter(date), iso_week = isoweek(date)) #add the rest of the columns

df

##           date year quarter iso_week
## 1 2015-01-01 2015         1         1
## 2 2015-03-01 2015         1         9
## 3 2015-05-01 2015         2        18
```

| | | | | |
|-------|------------|------|---|----|
| ## 4 | 2015-07-01 | 2015 | 3 | 27 |
| ## 5 | 2015-09-01 | 2015 | 3 | 36 |
| ## 6 | 2015-11-01 | 2015 | 4 | 44 |
| ## 7 | 2016-01-01 | 2016 | 1 | 53 |
| ## 8 | 2016-03-01 | 2016 | 1 | 9 |
| ## 9 | 2016-05-01 | 2016 | 2 | 17 |
| ## 10 | 2016-07-01 | 2016 | 3 | 26 |
| ## 11 | 2016-09-01 | 2016 | 3 | 35 |
| ## 12 | 2016-11-01 | 2016 | 4 | 44 |
| ## 13 | 2017-01-01 | 2017 | 1 | 52 |
| ## 14 | 2017-03-01 | 2017 | 1 | 9 |
| ## 15 | 2017-05-01 | 2017 | 2 | 18 |
| ## 16 | 2017-07-01 | 2017 | 3 | 26 |
| ## 17 | 2017-09-01 | 2017 | 3 | 35 |
| ## 18 | 2017-11-01 | 2017 | 4 | 44 |
| ## 19 | 2018-01-01 | 2018 | 1 | 1 |
| ## 20 | 2018-03-01 | 2018 | 1 | 9 |
| ## 21 | 2018-05-01 | 2018 | 2 | 18 |
| ## 22 | 2018-07-01 | 2018 | 3 | 26 |
| ## 23 | 2018-09-01 | 2018 | 3 | 35 |
| ## 24 | 2018-11-01 | 2018 | 4 | 44 |
| ## 25 | 2019-01-01 | 2019 | 1 | 1 |
| ## 26 | 2019-03-01 | 2019 | 1 | 9 |
| ## 27 | 2019-05-01 | 2019 | 2 | 18 |
| ## 28 | 2019-07-01 | 2019 | 3 | 27 |
| ## 29 | 2019-09-01 | 2019 | 3 | 35 |
| ## 30 | 2019-11-01 | 2019 | 4 | 44 |
| ## 31 | 2020-01-01 | 2020 | 1 | 1 |
| ## 32 | 2020-03-01 | 2020 | 1 | 9 |
| ## 33 | 2020-05-01 | 2020 | 2 | 18 |
| ## 34 | 2020-07-01 | 2020 | 3 | 27 |
| ## 35 | 2020-09-01 | 2020 | 3 | 36 |
| ## 36 | 2020-11-01 | 2020 | 4 | 44 |
| ## 37 | 2021-01-01 | 2021 | 1 | 53 |
| ## 38 | 2021-03-01 | 2021 | 1 | 9 |
| ## 39 | 2021-05-01 | 2021 | 2 | 17 |
| ## 40 | 2021-07-01 | 2021 | 3 | 26 |
| ## 41 | 2021-09-01 | 2021 | 3 | 35 |
| ## 42 | 2021-11-01 | 2021 | 4 | 44 |
| ## 43 | 2022-01-01 | 2022 | 1 | 52 |
| ## 44 | 2022-03-01 | 2022 | 1 | 9 |
| ## 45 | 2022-05-01 | 2022 | 2 | 17 |
| ## 46 | 2022-07-01 | 2022 | 3 | 26 |
| ## 47 | 2022-09-01 | 2022 | 3 | 35 |
| ## 48 | 2022-11-01 | 2022 | 4 | 44 |
| ## 49 | 2023-01-01 | 2023 | 1 | 52 |
| ## 50 | 2023-03-01 | 2023 | 1 | 9 |
| ## 51 | 2023-05-01 | 2023 | 2 | 18 |
| ## 52 | 2023-07-01 | 2023 | 3 | 26 |
| ## 53 | 2023-09-01 | 2023 | 3 | 35 |
| ## 54 | 2023-11-01 | 2023 | 4 | 44 |
| ## 55 | 2024-01-01 | 2024 | 1 | 1 |
| ## 56 | 2024-03-01 | 2024 | 1 | 9 |
| ## 57 | 2024-05-01 | 2024 | 2 | 18 |

```
## 58 2024-07-01 2024      3      27
## 59 2024-09-01 2024      3      35
## 60 2024-11-01 2024      4      44
## 61 2025-01-01 2025      1       1
## 62 2025-03-01 2025      1       9
## 63 2025-05-01 2025      2      18
## 64 2025-07-01 2025      3      27
## 65 2025-09-01 2025      3      36
## 66 2025-11-01 2025      4      44
```

Exercise 2 Question 2

Given the following dates, compute the difference in months and weeks between each consecutive pair.

```
sample_dates <- c("2018-03-15", "2020-07-20", "2023-01-10", "2025-09-05") %>%
  ymd() #convert sample dates from strings to Dates
dates <- sample_dates %>%
  data.frame() %>%
  select(date = ".") %>% #rename column
  mutate(date_diff = (date - lag(date))%>%as.period(), #find difference between dates
         diff_months = date_diff / months(1), diff_weeks = date_diff / weeks(1)) #convert to weeks and
# interpretation question - is difference in months and weeks referring to 2 differences?
# or is it referring to 1 difference, expressed in months,weeks?
# choosing the former because the differences involve non-integer numbers of weeks
dates
```

```
##      date      date_diff diff_months diff_weeks
## 1 2018-03-15      <NA>         NA         NA
## 2 2020-07-20  858d 0H 0M 0S    28.18891    122.5714
## 3 2023-01-10  904d 0H 0M 0S    29.70021    129.1429
## 4 2025-09-05  969d 0H 0M 0S    31.83573    138.4286
```

Exercise 3 Question 3

Using `map()` and `map_dbl()`, compute the mean, median, and standard deviation for each numeric vector in the following list

```
num_lists <- list(c(4, 16, 25, 36, 49), c(2.3, 5.7, 8.1, 11.4), c(10, 20, 30, 40, 50))
.f = function(l){ #create function to compute mean, median, and standard deviation
  return(c(mean(l), median(l), sd(l)))
}
map(num_lists, .f) # apply function to each vector in list
```

```
## [[1]]
## [1] 26.00000 25.00000 17.42125
##
## [[2]]
## [1] 6.8750 6.9000 3.8422
##
## [[3]]
## [1] 30.00000 30.00000 15.81139
```

*#why use map and map_dbl? there is only one list which needs functions applied to each item
#the inner lists need multiple functions applied to the whole list instead*

Exercise 4 Question 4

Given a list of mixed date formats, use `map()` and `possibly()` from `purrr` to safely convert them to `Date` format and extract the month name.

Assumptions: possible formats are `ymd` and `dmy`. (Without assumption, date 1 is ambiguous).

```
date_strings <- list("2023-06-10", "2022/12/25", "15-Aug-2021", "InvalidDate")
d = date_strings[1]
date_convert = function(d){ #function to convert dates in either format
  date = ifelse(is.na(ymd(d)), dmy(d), ymd(d)) %>%
    as.Date()
  return(date)
}
possibly_convert <- possibly(date_convert) #safe version of convert function
possibly_month <- possibly(~month(., label = T)) #safe version of month function
formatted_dates <- map_vec(date_strings, possibly_convert) #convert all to dates
months <- map_vec(formatted_dates, possibly_month) #extract all month names
data.frame(formatted_dates, months) #present as data frame
```

```
##   formatted_dates months
## 1      2023-06-10     Jun
## 2      2022-12-25     Dec
## 3      2021-08-15     Aug
## 4              <NA>  <NA>
```