Data 312: Lubridate and Purr assigment

Alasya Zeweldi

2025-02-25

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(purrr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
# First, I need to generate the date sequence using seq.i sued theof dates from January 1, 2015 to Dece
dates_seq <- seq(from = ymd("2015-01-01"),
                 to = ymd("2025-12-31"),
                 by = "2 months")

head(dates_seq)
```

```
## [1] "2015-01-01" "2015-03-01" "2015-05-01" "2015-07-01" "2015-09-01"
## [6] "2015-11-01"
```

```
# Now I extracted the components using lubridate functions, here I extracted the year, quarter, and ISO
date_info <- data.frame(
  date = dates_seq,
  year = year(dates_seq),
  quarter = quarter(dates_seq),
  iso_week = isoweek(dates_seq))

head(date_info)
```

```
##         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
```

```
# I defined the sample dates and then i converted strings to date objects, then i converted strings to
sample_dates <- c("2018-03-15", "2020-07-20", "2023-01-10", "2025-09-05")
date_objects <- ymd(sample_dates)


head(sample_dates)
```

```
## [1] "2018-03-15" "2020-07-20" "2023-01-10" "2025-09-05"
```

```
# then i calculated differences between consecutive dates pairs in both months and weeks. The head() an
date_diffs <- data.frame(
  start_date = head(date_objects, -1),
  end_date = tail(date_objects, -1),
  diff_months = interval(head(date_objects, -1), tail(date_objects, -1)) %/% months(1),
  diff_weeks = interval(head(date_objects, -1), tail(date_objects, -1)) %/% weeks(1))

head(date_diffs)
```

```
##   start_date   end_date diff_months diff_weeks
## 1 2018-03-15 2020-07-20          28        122
## 2 2020-07-20 2023-01-10          29        129
## 3 2023-01-10 2025-09-05          31        138
```

```
# heree is out numeric vector and head list the output, These varied vector types help demonstrate how

num_lists <- list(c(4, 16, 25, 36, 49), c(2.3, 5.7, 8.1, 11.4), c(10, 20, 30, 40, 50))

head(num_lists)
```

```
## [[1]]
## [1]  4 16 25 36 49
##
## [[2]]
## [1]  2.3  5.7  8.1 11.4
##
## [[3]]
## [1] 10 20 30 40 50
```

```
# here i used map() to compute statistics for each vector. I used purrr's map_dfr function to calculate
stats_results <- map_dfr(num_lists, function(x) {
  tibble(
    mean_value = mean(x),
    median_value = median(x),
    sd_value = sd(x))})
```

```r
head(stats_results)
```

```
## # A tibble: 3 x 3
##   mean_value median_value sd_value
##        <dbl>        <dbl>    <dbl>
## 1      26            25      17.4
## 2       6.88          6.9     3.84
## 3      30            30      15.8
```

```r
# i added identifier for each list to clarify which statistics belong to which vector. Using mutate wit
stats_results <- stats_results %>%
  mutate(list_id = paste0("list_", row_number()))

head(stats_results)
```

```
## # A tibble: 3 x 4
##   mean_value median_value sd_value list_id
##        <dbl>        <dbl>    <dbl> <chr>
## 1      26            25      17.4  list_1
## 2       6.88          6.9     3.84 list_2
## 3      30            30      15.8  list_3
```

```r
# here i used Alternative using map_dbl() for individual statistics.  This creates individual vectors f
mean_values <- map_dbl(num_lists, mean)
median_values <- map_dbl(num_lists, median)
sd_values <- map_dbl(num_lists, sd)

head(mean_values)
```

```
## [1] 26.000  6.875 30.000
```

```r
stats_results_alt <- tibble(
  list_id = paste0("list_", 1:length(num_lists)),
  mean_value = mean_values,
  median_value = median_values,
  sd_value = sd_values)

head(stats_results_alt)
```

```
## # A tibble: 3 x 4
##   list_id mean_value median_value sd_value
##   <chr>        <dbl>        <dbl>    <dbl>
## 1 list_1       26            25      17.4
## 2 list_2        6.88          6.9     3.84
## 3 list_3       30            30      15.8
```

```r
#here i safely convert mixed date formats to Date format and extract the month name. I created a list o

date_strings <- list("2023-06-10", "2022/12/25", "15-Aug-2021", "InvalidDate")
head(date_strings)
```

```
## [[1]]
## [1] "2023-06-10"
##
## [[2]]
## [1] "2022/12/25"
##
## [[3]]
## [1] "15-Aug-2021"
##
## [[4]]
## [1] "InvalidDate"
```

```
# here i created a function to parse dates in various formats and extract month
parse_date_get_month <- function(date_str) {
  # Try to parse the date with multiple formats
  parsed_date <- NA

  # i tried each format, one at a time
  if(is.na(parsed_date)) {
    parsed_date <- suppressWarnings(try(ymd(date_str), silent = TRUE))
  }


  if(is.na(parsed_date) || inherits(parsed_date, "try-error")) {
    parsed_date <- suppressWarnings(try(dmy(date_str), silent = TRUE))
  }

  if(is.na(parsed_date) || inherits(parsed_date, "try-error")) {
    parsed_date <- suppressWarnings(try(mdy(date_str), silent = TRUE))
  }

  # and then returned month name if successful, "Invalid" otherwise
  if(!is.na(parsed_date) && !inherits(parsed_date, "try-error")) {
    return(month(parsed_date, label = TRUE, abbr = FALSE))
  } else {
    return("Invalid")}}

  head(parse_date_get_month)
```

```
##
## 1 function (date_str)
## 2 {
## 3     parsed_date <- NA
## 4     if (is.na(parsed_date)) {
## 5         parsed_date <- suppressWarnings(try(ymd(date_str), silent = TRUE))
## 6     }
```

```
  # This function handles multiple date formats and errors gracefully. It tries different formats seque
```

```
#  I used a simple for loop instead of map functions to apply our parsing function. This makes debuggin
month_names <- character(length(date_strings))
for(i in 1:length(date_strings)) {
  month_names[i] <- parse_date_get_month(date_strings[[i]])}
```

```r
head(month_names)
```

```
## [1] "6"       "12"      "8"       "Invalid"
```

```r
# here i created a clean table showing result. This created a tibble showing both original strings and

date_results <- tibble(
  Date_String = unlist(date_strings),
  Month_Name = month_names)

head(date_results)
```

```
## # A tibble: 4 x 2
##   Date_String Month_Name
##   <chr>       <chr>
## 1 2023-06-10  6
## 2 2022/12/25  12
## 3 15-Aug-2021 8
## 4 InvalidDate Invalid
```

```r
#Conclusion:


#Exercise 1 demonstrates date sequence generation and extraction of date components using lubridate. Ex
```