assignment_2

```r
library(tidyverse)

library(lubridate)
```

Question 1:

Write a function `pos_na()` that takes two vectors of equal length and returns the positions where **both** vectors contain `NA`.
If the vectors do **not** have the same length, the function should return the message:

> `"The vectors are not the same length."`

Use the following pairs of vectors to test your function:

1. `c(1, NA, 2)` and `c(NA, NA)`
2. `c(NA, NA, 2)` and `c(NA, NA, 2)`
3. `c(NA, 5, NA)` and `c(NA, NA, NA)`
4. `c(NA, NA, NA, 2, NA, 4, NA)` and `c(NA, NA, 2, 4, 4, NA, NA)`
5. `c(1, NA, NA, 2, NA, 4, NA)` and `c(NA, -5, 2, 4, 4, NA, 11)`

Here's a polished, professional way to label those two parts of the assignment. It reads cleanly and sets expectations without adding extra wording.

(1i) Using an `if` statement

Write a version of `pos_na()` that checks whether the two input vectors have the same length using an `if` statement.

- If they do, return the positions where both vectors contain `NA`.

- If they do not, return the message:

> `"The vectors are not the same length."`

```r
pos_na_if <- function(x, y) {
  if (length(x) != length(y)) {
    return("The vectors are not the same length.")
  }
  which(is.na(x) & is.na(y))
}
```

(1ii) Without using an `if` statement

Write a second version of `pos_na()` that performs the same task **without** using an `if` statement.

- You may use any other valid R mechanism (e.g., `stop()`, `stopifnot()`, or logical short-circuiting) to handle the case where the vectors differ in length.

```r
pos_na_noif <- function(x, y) {
  (length(x) == length(y)) || return("The vectors are not the same length.")
  which(is.na(x) & is.na(y))
}
```

```r
pos_na_if(c(1, NA, 2), c(NA, NA))
```

```
[1] "The vectors are not the same length."
```

```r
pos_na_if(c(NA, NA, 2), c(NA, NA, 2))
```

```
[1] 1 2
```

```r
pos_na_if(c(NA, 5, NA), c(NA, NA, NA))
```

```
[1] 1 3
```

```r
pos_na_if(c(NA, NA, NA, 2, NA, 4, NA), c(NA, NA, 2, 4, 4, NA, NA))
```

```
[1] 1 2 7
```

```r
pos_na_if(c(1, NA, NA, 2, NA, 4, NA), c(NA, -5, 2, 4, 4, NA, 11))
```

```
integer(0)
```

```r
pos_na_noif(c(1, NA, 2), c(NA, NA))
```

```
[1] "The vectors are not the same length."
```

```r
pos_na_noif(c(NA, NA, 2), c(NA, NA, 2))
```

```
[1] 1 2
```

```r
pos_na_noif(c(NA, 5, NA), c(NA, NA, NA))
```

```
[1] 1 3
```

```r
pos_na_noif(c(NA, NA, NA, 2, NA, 4, NA), c(NA, NA, 2, 4, 4, NA, NA))
```

```
[1] 1 2 7
```

```r
pos_na_noif(c(1, NA, NA, 2, NA, 4, NA), c(NA, -5, 2, 4, 4, NA, 11))
```

```
integer(0)
```

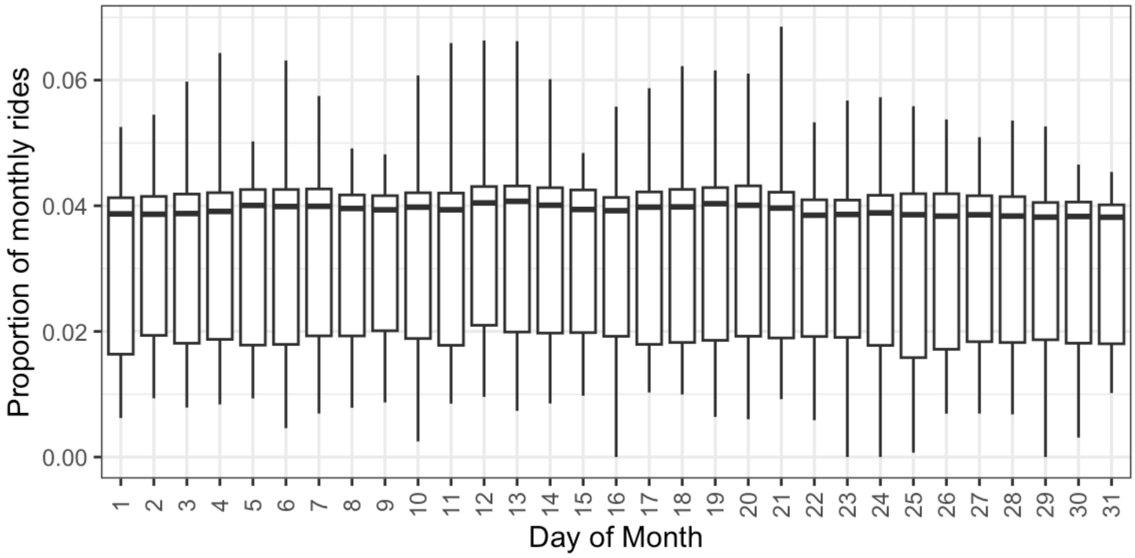
Question 2

Load the `wmata_ridership` data frame into R from https://dcgerard.github.io/stat_412_612/data/wmata_ridership.csv. For each month, calculate the proportion of rides made on a given day of the month. Then make box plots of the proport

```
library(tidyverse)
library(lubridate)

wmata <- read_csv("https://dcgerard.github.io/stat_412_612/data/wmata_ridership.csv")
```

```
Rows: 5469 Columns: 2
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
dbl  (1): Total
date (1): Date

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# Find the date column (common names: date, Date)
date_col <- names(wmata)[tolower(names(wmata)) %in% c("date")]
wmata <- wmata %>%
  mutate(date = as.Date(.data[[date_col]]))

# Find the ridership column (first numeric column other than date)
num_cols <- names(wmata)[sapply(wmata, is.numeric)]
ride_col <- setdiff(num_cols, "date")[1]

wmata_props <- wmata %>%
  mutate(
    month = floor_date(date, "month"),
    dom = day(date),
    rides = .data[[ride_col]]
  ) %>%
  group_by(month, dom) %>%
  summarise(rides = sum(rides, na.rm = TRUE), .groups = "drop") %>%
  group_by(month) %>%
  mutate(prop = rides / sum(rides)) %>%
  ungroup()

# Boxplots of proportions by day-of-month across months
ggplot(wmata_props, aes(x = factor(dom), y = prop)) +
  geom_boxplot() +
  labs(x = "Day of Month", y = "Proportion of monthly rides") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
```



Question 3:

Write R code that extracts the elements “Bears”, “Dolphins”, and “Bengals” from the vector `V` shown below, and display the result.

```
V <- c("Bears", "Lions", "Dolphins", "Eagles", "Bengals")
```

(3i) Subsetting by index

```
V[c(1, 3, 5)]
```

```
[1] "Bears"    "Dolphins" "Bengals" 
```

(3ii) Subsetting using a `for` loop

```
idx <- c(1, 3, 5)
out <- character(0)
for (i in idx) out <- c(out, V[i])
out
```

```
[1] "Bears"    "Dolphins" "Bengals" 
```

(3iii)Returning all values in a single vector (one line of code)

(a) Using positive indices

```
V[c(1, 3, 5)]
```

```
[1] "Bears"    "Dolphins" "Bengals" 
```

(b) Using negative indices

```
V[-c(2, 4)]
```

```
[1] "Bears"    "Dolphins" "Bengals"
```

(c) Using logical indexing

```
V[c(TRUE, FALSE, TRUE, FALSE, TRUE)]
```

```
[1] "Bears"    "Dolphins" "Bengals"
```

(d) Attempting to subset by names

```
V[c("Bears", "Dolphins", "Bengals")]
```

```
[1] NA NA NA
```

Question 4:

You are given a character vector containing messy employee records. Each entry includes:

- an employee name (string),
- a hire date in inconsistent formats,
- and a department code that should be treated as a factor.

```
records <- c(
  "Steve McQueen | 2020-01-15 | HR",
  "B. Smith | 15/02/2021 | FIN",
  "Carlos M | March 3, 2019 | IT",
  "D. Lee | 2018/07/30 | HR",
  "Alain Delon | 04-12-2020 | MKT"
)
records
```

```
[1] "Steve McQueen | 2020-01-15 | HR" "B. Smith | 15/02/2021 | FIN"
[3] "Carlos M | March 3, 2019 | IT"   "D. Lee | 2018/07/30 | HR"
[5] "Alain Delon | 04-12-2020 | MKT"
```

1. **Parse each record** into three separate variables:

   - `name`
   - `hire_date`
   - `dept`

2. **Convert `hire_date` into a proper `Date` object**, correctly handling all of the different date formats.

3. **Convert `dept` into an ordered factor** with the levels: `HR`, `FIN`, `IT`, `MKT`.

4. **Construct a clean data frame** containing the parsed and converted variables.

5. **Add a new variable** called `years_worked` that reports the number of full years each employee has worked as of today.

6. **Return the final cleaned data frame.**

```r
library(tidyverse)
library(lubridate)

records <- c(
  "Steve McQueen | 2020-01-15 | HR",
  "B. Smith | 15/02/2021 | FIN",
  "Carlos M | March 3, 2019 | IT",
  "D. Lee | 2018/07/30 | HR",
  "Alain Delon | 04-12-2020 | MKT"
)

clean_df <- tibble(raw = records) %>%
  separate(raw, into = c("name", "hire_date", "dept"), sep = "\\s*\\|\\s*") %>%
  mutate(
    hire_date = parse_date_time(
      hire_date,
      orders = c("ymd", "dmy", "mdy", "Y/m/d", "m-d-Y"),
      tz = "UTC"
    ) %>% as.Date(),
    dept = factor(dept, levels = c("HR", "FIN", "IT", "MKT"), ordered = TRUE),
    years_worked = floor(time_length(interval(hire_date, Sys.Date()), "years"))
  )

clean_df
```

```
# A tibble: 5 × 4
  name          hire_date  dept  years_worked
  <chr>         <date>     <ord>        <dbl>
1 Steve McQueen 2020-01-15 HR               6
2 B. Smith      2021-02-15 FIN              4
3 Carlos M      2019-03-03 IT               6
4 D. Lee        2018-07-30 HR               7
5 Alain Delon   2020-12-04 MKT              5
```