HW4 - Programming in Base R

Kayla Kippes

Task 1: Conceptual Questions

Question 1

The purpose of *lapply*() is to apply functions to each element of a list and returns a list. The equivalent *purrr* function is *map*().

Question 2

```
lapply(my_list, cor, method = "kendall")
```

Question 3

1. It gives cleaner and more readable code.
2. It provides more functionality.

Question 4

A side-effect function doesn't try to change or transform the data, it just tries to produce something.

Question 5

This is because when you call a function, it creates a temporary function environment (lexical scoping).

Task 2: Writing R Functions

Question 1

```
getRMSE <- function(responses, predictions, ...) {
  rmse <- sqrt(mean((responses - predictions)^2, ...))

  return(rmse)
}
```

Question 2

```
#get sample data
set.seed(10)
n <- 100
x <- runif(n)
resp <- 3 + 10*x + rnorm(n)
pred <- predict(lm(resp ~ x), data.frame(x))

#Test function on data
getRMSE(resp, pred)
```

```
[1] 0.9581677
```

```
#Get NA data
resp_w_na <- resp
resp_w_na[c(1,2)] <- NA_real_

#Test with specifying behavior
getRMSE(resp_w_na, pred, na.rm = TRUE)
```

```
[1] 0.9661699
```

```
#Test without specifying behavior
getRMSE(resp_w_na, pred)
```

```
[1] NA
```

Question 3

```
getMAE <- function(responses, predictions, ...) {
  mae <- mean(abs(responses - predictions), ...)

  return(mae)
}
```

Question 4

```
#get sample data
set.seed(10)
n <- 100
x <- runif(n)
resp <- 3 + 10*x + rnorm(n)
pred <- predict(lm(resp ~ x), data.frame(x))

#Test function on data
getMAE(resp, pred)
```

```
[1] 0.8155776
```

```
#Get NA data
resp_w_na <- resp
resp_w_na[c(1,2)] <- NA_real_

#Test with specifying behavior
getMAE(resp_w_na, pred, na.rm = TRUE)
```

```
[1] 0.8241201
```

```
#Test without specifying behavior
getMAE(resp_w_na, pred)
```

```
[1] NA
```

Question 5

```
wrapper_metrics <- function(responses, predictions,
                            metrics = c("RMSE", "MAE"), ...) {
  #check that two numeric atomic vectors have passed
  if (!(is.vector(responses) && is.atomic(responses)
        && is.numeric(responses))) {
    return(NULL)
  }
  if (!(is.vector(predictions) && is.atomic(predictions)
        && is.numeric(predictions))) {
    return(NULL)
  }

  #store results in a list
  stored_metrics <- list()
  #check for RMSE
  if ("RMSE" %in% metrics) {
    stored_metrics$RMSE <- getRMSE(responses, predictions, ...)
  }
  #check for MAE
  if ("MAE" %in% metrics) {
    stored_metrics$MAE <- getMAE(responses, predictions, ...)
  }

  #return results
  return(stored_metrics)
}
```

Question 6

```
#get sample data
set.seed(10)
n <- 100
x <- runif(n)
resp <- 3 + 10*x + rnorm(n)
pred <- predict(lm(resp ~ x), data.frame(x))

#Test function on RMSE
wrapper_metrics(resp, pred, "RMSE")
```

```
$RMSE
[1] 0.9581677
```

```
#Test function on MAE
wrapper_metrics(resp, pred, "MAE")
```

```
$MAE
[1] 0.8155776
```

```
#Test function on both
wrapper_metrics(resp, pred)
```

```
$RMSE
[1] 0.9581677

$MAE
[1] 0.8155776
```

```
#Get NA data
resp_w_na <- resp
resp_w_na[c(1,2)] <- NA_real_

#Test function on RMSE w/ NA
wrapper_metrics(resp_w_na, pred, "RMSE", na.rm = TRUE)
```

```
$RMSE
[1] 0.9661699
```

```
#Test function on MAE w/ NA
wrapper_metrics(resp_w_na, pred, "MAE", na.rm = TRUE)
```

```
$MAE
[1] 0.8241201
```

```
#Test function on both w/ NA
wrapper_metrics(resp_w_na, pred, na.rm = TRUE)
```

```
$RMSE
[1] 0.9661699

$MAE
[1] 0.8241201
```

```
#Test using incorrect data
wrapper_metrics(iris, pred)
```

```
NULL
```

Task 3: Querying an API and a Tidy-Style Function

Question 1

```
library(httr)
```

```
Warning: package 'httr' was built under R version 4.3.3
```

```
api_key <- '6a44e855b7b643adb4a58bc5b15c26a2'

result <- GET(url = 'https://newsapi.org/v2/everything',
              query = list(q = 'economy', apiKey = api_key))
```

Question 2

```
library(jsonlite)
library(tidyverse)
parsed <- fromJSON(rawToChar(result$content))
economy_info <- as_tibble(parsed$articles)
head(economy_info)
```

```
# A tibble: 6 x 8
  source$id  $name author title description url   urlToImage publishedAt content
  <chr>      <chr> <chr>  <chr> <chr>       <chr> <chr>      <chr>       <chr>
1 the-verge  The ~ Andre~ Repu~ Things are~ http~ https://p~ 2025-06-10~ "Surpr~
2 wired      Wired Boone~ ‘Buy~ BNPL servi~ http~ https://m~ 2025-06-04~ "The e~
```

```
3 business-~ Busi~ Dan D~ The ~ A better-t~ http~ https://i~ 2025-06-09~ "REUTE~
4 business-~ Busi~ Thiba~ It's~ The OECD s~ http~ https://i~ 2025-06-03~ "The O~
5 business-~ Busi~ Kathe~ Gen ~ Gen Zers p~ http~ https://i~ 2025-06-19~ "Gen Z~
6 business-~ Busi~ Kelse~ Amer~ Plenty of ~ http~ https://i~ 2025-06-08~ "Ameri~
```

```r
query_api <- function(title, time_period, api_key) {
  #get the result
  result <- GET(url = 'https://newsapi.org/v2/everything',
                query = list(
                  q = title,
                  from = time_period,
                  apiKey = api_key
                ))
  #parse it
  parsed <- fromJSON(rawToChar(result$content))
  #go to the data frame and return
  return(head(as_tibble(parsed$articles)))
}

query_api('gamestop', '2025-06-13', '6a44e855b7b643adb4a58bc5b15c26a2')
```

```
# A tibble: 6 x 8
  source$id $name  author title description url   urlToImage publishedAt content
  <chr>     <chr>  <chr>  <chr> <chr>       <chr> <chr>      <chr>       <chr>
1 the-verge The V~ Brand~ The ~ Amazon's m~ http~ https://p~ 2025-06-20~ "Amazo~
2 <NA>      Slash~ msmash Game~ GameStop i~ http~ https://a~ 2025-06-13~ "Cohen~
3 <NA>      Gizmo~ James~ Some~ There's on~ http~ https://g~ 2025-06-18~ "The S~
4 <NA>      Yahoo~ <NA>   $1.4~ <NA>        http~ <NA>       2025-06-18~ "If yo~
5 <NA>      Andro~ Adamy~ Some~ In a high-~ http~ https://w~ 2025-06-19~ "<ul><~
6 <NA>      Xatak~ John ~ Cuan~ El robo má~ http~ https://i~ 2025-06-20~ "El ro~
```