Apply, Purrr, Functions, API Queries, and Other Advanced R Programming Tools

Task One: Conceptual Questions

Question 1: What is the purpose of the `lapply()` function? What is the equivalent `purrr` function?

The `lapply()` function is used for list objects. It applies the specified function to every element of the list and then returns a list. The equivalent `purrr` function is `map()`.

Question 2: Suppose we have a list called `my_list`. Each element of the list is a numeric data frame (all columns

are numeric). We want use `lapply()` to run the code `cor(numeric_matrix, method = "kendall")` on each element of the list. Write code to do this below! (I'm really trying to ask you how you specify `method = "kendall"` when calling `lapply()`)

```
# lapply(X = my_list,
#        FUN = cor,
#        method = "kendall")
```

Question 3: What are two advantages of using `purrr` functions instead of the `BaseR` apply family?

One advantages of the `purrr` functions is that they are generally more consistent (additional arguments can be specified under the same naming conventions, the order of arguments is consistent, etc.). Another advantage is that the `purrr` functions contain useful helper functions (like `partial()`) and variants (like `map2()`).

Question 4: What is a side-effect function?

A side-effect function uses the data to perform a task, but when the task is complete it returns the original data unmodified.

Question 5: Why can you name a variable `sd` in a function and not cause any issues with the `sd` function?

Variables that are created in functions are only stored temporarily in the function environment. They are not kept after the function call is finished, and therefore never stored in the global environment.

Task Two: Writing R Functions

Question 1: Writing RMSE function

```
#Writing the RMSE function as taking in two vectors
getRMSE <- function(resp_vec, pred_vec, ...) {
  sq_error <- (resp_vec - pred_vec)^2 #finding square error
  mean <- mean(sq_error, ...) #finding mean square error (MSE)
  rmse <- sqrt(mean) #finding root MSE
  return(rmse) #returning the calculated RMSE value
  }
```

Question 2: Testing RMSE function

```
#Generating test data
set.seed(10) #setting random seed so that data can be replicated
n <- 100 #we want 100 generated data points
x <- runif(n) #randomly generating 100 data points from the uniform distribution
resp <- 3 + 10*x + rnorm(n) #generating response data
pred <- predict(lm(resp ~ x), data.frame(x)) #generating prediction data
```

Testing my RMSE function

```
getRMSE(resp, pred)
```

```
[1] 0.9581677
```

Replacing several observations with missing values

```
resp[3] <- NA_real_ #setting the third response to missing
resp[55] <- NA_real_ #setting the 55th response to missing
```

Testing my RMSE function without specifying missing values behavior

```
getRMSE(resp, pred)
```

```
[1] NA
```

Testing my RMSE function with removing missing values

```
getRMSE(resp, pred, na.rm = TRUE)
```

```
[1] 0.9568069
```

Question 3: Writing Mean Absolute Deviation Function

```
#Writing the MAE function as taking in two vectors
getMAE <- function(resp_vec, pred_vec, ...){
  diff <- resp_vec - pred_vec #finding the difference between prediction and response
  abs_diff <- abs(diff) #finding the absolute difference
  mae <- mean(abs_diff, ...) #finding MAE
  return(mae) #returning MAE
}
```

Question 4: Testing MAE Function

Generating Test Data

```
set.seed(10) #setting random seed so that data can be replicated
n <- 100 #we want 100 generated data points
x <- runif(n) #randomly generating 100 data points from the uniform distribution
resp <- 3 + 10*x + rnorm(n) #generating response data
pred <- predict(lm(resp ~ x), data.frame(x)) #generating prediction data
```

Testing my MAE function

```
getMAE(resp, pred)
```

```
[1] 0.8155776
```

Replacing several observations with missing values

```
resp[3] <- NA_real_ #setting the third response to missing
resp[55] <- NA_real_ #setting the 55th response to missing
```

Testing my MAE function without specifying missing values behavior

```
getMAE(resp, pred)
```

```
[1] NA
```

Testing my MAE function with removing missing values

```
getMAE(resp, pred, na.rm = TRUE)
```

```
[1] 0.812853
```

Question 5: Writing a Wrapper Function to Calculate RMSE and MAE

```
wrapper <- function(resp_vec, pred_vec, metric = c("RMSE", "MAE"), ...) {
  if (!is.vector(resp_vec) | !is.vector(pred_vec) | #checking vectors
      !is.numeric(resp_vec) | !is.numeric(pred_vec) | #checking numeric
      !is.atomic(resp_vec) | !is.atomic(pred_vec)) { #checking atomic
    print("ERROR: the responses and/or the predictions are not a numeric(atomic) vector")
    } else if (all(c("RMSE","MAE") %in% metric)) { #checking for both metrics
      rmse <- getRMSE(resp_vec, pred_vec, ...) #calculating RMSE
      mae <- getMAE(resp_vec, pred_vec, ...) #calculating MAE
      return(c(paste("RMSE =", rmse), paste("MAE =", mae)))
      } else if (metric == "rmse" | metric == "RMSE") { #checking for RMSE
        rmse <- getRMSE(resp_vec, pred_vec, ...) #calculating RMSE
        return(paste("RMSE =", rmse))
        } else if (metric == "mae" | metric == "MAE") { #checking for MAE
          mae <- getMAE(resp_vec, pred_vec, ...) #calculating MAE
          return(paste("MAE =", mae))
        }
  }
```

Question 6: Testing the Wrapper Function

```
#Generating Test Data
set.seed(10) #setting random seed so that data can be replicated
n <- 100 #we want 100 generated data points
x <- runif(n) #randomly generating 100 data points from the uniform distribution
resp <- 3 + 10*x + rnorm(n) #generating response data
pred <- predict(lm(resp ~ x), data.frame(x)) #generating prediction data
```

Testing the Wrapper Function Under Metric Default (Both)

```
wrapper(resp, pred)
```

```
[1] "RMSE = 0.958167655151933" "MAE = 0.815577593682669"
```

Testing the Wrapper Function Specifying RMSE as the Metric

```
wrapper(resp, pred, "rmse")
```

```
[1] "RMSE = 0.958167655151933"
```

Testing the Wrapper Function Specifying MAE as the Metric

```
wrapper(resp, pred, "mae")
```

```
[1] "MAE = 0.815577593682669"
```

Replacing several observations with missing values

```
resp[3] <- NA_real_ #setting the third response to missing
resp[55] <- NA_real_ #setting the 55th response to missing
```

Testing the Wrapper Function Under Metric Default (Both) with Missing Data

```
wrapper(resp, pred)
```

```
[1] "RMSE = NA" "MAE = NA"
```

Testing the Wrapper Function Under Metric Default (Both) with Missing Data Removed

```
wrapper(resp, pred, na.rm = TRUE)
```

```
[1] "RMSE = 0.956806850171415" "MAE = 0.812853024472129"
```

Testing the Wrapper Function Specifying RMSE as the Metric with Missing Data

```
wrapper(resp, pred, "rmse")
```

```
[1] "RMSE = NA"
```

Testing the Wrapper Function Specifying RMSE as the Metric with Missing Data Removed

```
wrapper(resp, pred, "rmse", na.rm = TRUE)
```

```
[1] "RMSE = 0.956806850171415"
```

Testing the Wrapper Function Specifying MAE as the Metric with Missing Data

```
wrapper(resp, pred, "mae")
```

```
[1] "MAE = NA"
```

Testing the Wrapper Function Specifying MAE as the Metric with Missing Data Removed

```
wrapper(resp, pred, "mae", na.rm = TRUE)
```

```
[1] "MAE = 0.812853024472129"
```

Testing the Wrapper Function When Both Arguments are not Vectors

```
resp_df <- as.data.frame(resp) #converting responses to a data frame
pred_df <- as.data.frame(pred) #converting predictions to a data frame

wrapper(resp_df, pred_df)
```

```
[1] "ERROR: the responses and/or the predictions are not a numeric(atomic) vector"
```

Testing the Wrapper Function When Response Argument is not a Vector

```
wrapper(resp, pred_df)
```

```
[1] "ERROR: the responses and/or the predictions are not a numeric(atomic) vector"
```

Testing the Wrapper Function When Prediction Argument is not a Vector

```
wrapper(resp_df, pred)
```

```
[1] "ERROR: the responses and/or the predictions are not a numeric(atomic) vector"
```

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

This information sometimes get cut off, but my API key is 0617778f612b4946859a211593d3efda.

Question 1: Using `GET` from the `httr` Package to Pull News Info

```
#Loading required packages
library("httr")

#Using httr::GET to collect information about the Pacers in the news
pacers_info <- httr::GET("https://newsapi.org/v2/everything?q=pacers&from=2025-06-22&apiKey=0
pacers_info
```

```
Response [https://newsapi.org/v2/everything?q=pacers&from=2025-06-22&apiKey=0617778f612b49468
  Date: 2025-06-24 21:19
  Status: 200
  Content-Type: application/json; charset=utf-8
  Size: 83.1 kB
```

Question 2: Parsing the News Info to Pull Articles

```
#Loading required packages
library("tidyverse")
library("jsonlite")

#Parsing the news information about the Pacers
parsed_pacers_info <- fromJSON(rawToChar(pacers_info$content))
pacers_data <- as_tibble(parsed_pacers_info$articles) #pulling articles
pacers_data #displaying the article information
```

```
# A tibble: 96 x 8
   source$id $name author title description url   urlToImage publishedAt content
   <chr>     <chr> <chr>  <chr> <chr>       <chr> <chr>      <chr>       <chr>
 1 <NA>      NPR   Becky~ Afte~ Led by poi~ http~ https://n~ 2025-06-23~ "The O~
 2 espn      ESPN  ESPN ~ Jale~ The respec~ http~ https://a~ 2025-06-22~ "It's ~
 3 espn      ESPN  NBA i~ Game~ The Pacers~ http~ https://a~ 2025-06-22~ "Jun 2~
 4 espn      ESPN  David~ Game~ Multiple s~ http~ https://a~ 2025-06-22~ "The I~
 5 abc-news  ABC ~ TIM R~ Thun~ Shai Gilge~ http~ https://i~ 2025-06-23~ "OKLAH~
```

```
 6 abc-news  ABC ~ ABC N~ WATC~ The Oklaho~ http~ https://i~ 2025-06-23~ "<ul><~
 7 le-monde  Le M~ Valen~ NBA ~ Portée par~ http~ https://i~ 2025-06-23~ "Alex ~
 8 <NA>      HYPE~ info@~ The ~ SummaryThe~ http~ https://i~ 2025-06-23~ "Summa~
 9 <NA>      CNET  Matt ~ NBA ~ Discover t~ http~ https://w~ 2025-06-22~ "The N~
10 die-zeit  Die ~ ZEIT ~ Bask~ Hier finde~ http~ https://i~ 2025-06-23~ "Natio~
# i 86 more rows
```

Question 3: Writing a Function to Query the API

```
api_query <- function(subject, time, key){
  url = paste0("https://newsapi.org/v2/everything?q=", #base of URL
               subject, #adding subject to URL
               "&from=", time, #adding time to URL
               "&apiKey=", key) #adding API key to URL
  subject_info <- httr::GET(url) #extracting news info
  parsed_subject_info <- fromJSON(rawToChar(subject_info$content)) #parsing content of news
  subject_data <- as_tibble(parsed_subject_info$articles) #pulling articles
  return(subject_data)
}
```

Testing my Short Function to Query the API

```
api_query("gamestop", "2025-05-24", "0617778f612b4946859a211593d3efda")
```

```
# A tibble: 99 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 ~ David~ A ni~ I'm standi~ http~ https://p~ 2025-06-05~ "Body ~
 2 the-verge The ~ Brand~ The ~ Amazon's m~ http~ https://p~ 2025-06-20~ "Amazo~
 3 business~ Busi~ fdemo~ Game~ GameStop a~ http~ https://i~ 2025-05-28~ "GameS~
 4 <NA>      Gizm~ Kyle ~ Targ~ Check to m~ http~ https://g~ 2025-06-03~ "The S~
 5 <NA>      Slas~ msmash Game~ GameStop i~ http~ https://a~ 2025-06-13~ "Cohen~
 6 <NA>      Yaho~ Brad ~ Bitc~ The 2025 a~ http~ https://s~ 2025-05-28~ "Bitco~
 7 <NA>      Gizm~ James~ Did ~ Maybe orde~ http~ https://g~ 2025-06-05~ "When ~
 8 <NA>      Hipe~ Gabri~ Desa~ El gran dí~ http~ https://i~ 2025-06-05~ "El gr~
 9 <NA>      Kota~ Ethan~ Stat~ Imagine yo~ http~ https://i~ 2025-06-05~ "Imagi~
10 <NA>      Gizm~ James~ Some~ There's on~ http~ https://g~ 2025-06-18~ "The S~
# i 89 more rows
```