Homework 4

Task 1: Conceptual Questions

1. What is the purpose of the lapply() function? What is the equivalent purrr function?

The lapply() function applies any function to a list. The equivalent purr function is map().

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())

```
#Create a numeric data frame
df1 <- data.frame(a = runif(20), b = runif(20), c = runif(20))
df2 <- data.frame(a = runif(10), b = runif(10), c = runif(10))
df3 <- data.frame(a = runif(30), b = runif(30), c = runif(30))

my_list <- list(df1, df2, df3)

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

```
[[1]]
           a          b          c
a  1.0000000  0.2526316 -0.3473684
b  0.2526316  1.0000000 -0.1894737
c -0.3473684 -0.1894737  1.0000000

[[2]]
            a           b           c
a  1.00000000 -0.06666667 -0.02222222
```

```
b -0.06666667  1.00000000  0.15555556
c -0.02222222  0.15555556  1.00000000

[[3]]
          a            b            c
a 1.0000000  0.140229885  0.140229885
b 0.1402299  1.000000000 -0.002298851
c 0.1402299 -0.002298851  1.000000000
```

3. What are two advantages of using purrr functions instead of the BaseR apply family?

The two advantage of using the purrr function is the consistency between functions (i.e the first argument to all map functions is the data which is not the same as a apply function).The purr function also has more functionality such as modify(), map2(), and imap().

4. What is a side-effect function?

A side-effect functions don't return the modified argumen. Example functions would be print(), write_csv().

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

This is because when you write a new function, it creates it's own temporary environment when it executes the code. When the function is complete, the environment is gone. Therefore, in the temporary enviroment you can have a variable called sd and not cause issues with the sd function.

Task 2: Writing R functinos

1. Write a basic function (call it getRMSE()) that takes in a vector of responses and a vector of predictions and outputs the RMSE.If a value is missing for the vector of responses (i.e. an NA is present), allow for additional arguments to the mean() function (elipses) that removes the NA values in the computation.

```r
getRMSE<-function(n, resp, pred,...){
SE <- (resp-pred)^2
mean <- mean(SE,...)
RMSE <- sqrt(mean)
return(RMSE)
}
```

2. Test your RMSE function using this data. Repeat after replacing two of the response values with missing values (NA_real_).

```r
#Without Missing Values
set.seed(10)
n <- 100
x <- runif(n)
resp <- 3 + 10*x + rnorm(n)
pred <- predict(lm(resp ~ x), data.frame(x))
getRMSE(n,resp,pred)
```

```
[1] 0.9581677
```

```r
#With Missing Values
set.seed(10)
n <- 100
x <- runif(n)
resp2 <- 3 + 10*x + rnorm(n)
pred2 <- predict(lm(resp ~ x), data.frame(x))
random_na <- sample(length(resp2),2)
resp2[random_na]<- NA_real_
pred2 <- predict(lm(resp2 ~ x), data.frame(x))
getRMSE(n,resp2,pred2,)
```

```
[1] NA
```

```r
getRMSE(n,resp2,pred2,na.rm = TRUE)
```

```
[1] 0.9659839
```

3.Write a function called getMAE() that follows the specifications of the getRMSE() function.

```r
getMAE<-function(n, resp, pred,...){
dif <- abs(resp-pred)
MAE <- mean(dif,...)
return(MAE)
}
```

4. Run the following code to create some response values and predictions.Test your MAE function using this data.

```r
#Without Missing Values
getMAE(n,resp,pred)
```

```
[1] 0.8155776
```

```r
#With Missing Values
getMAE(n,resp2,pred2)
```

```
[1] NA
```

```r
getMAE(n,resp2,pred2,na.rm = TRUE)
```

```
[1] 0.8238812
```

5. Let's create a wrapper function that can be used to get either or both metrics returned with a single function call. Do not rewrite your above two functions, call them inside the wrapper function (we would call the getRMSE() and getMAE() functions helper functions). When returning your values, give them appropriate names.

```r
calculating_wrapper <- function(n,resp,pred,getRMSE=TRUE, getMAE=TRUE,...){
  if(!is.numeric(resp) || !is.vector(resp) || !is.atomic((resp))){
    stop("resp needs to be numeric vector")
  }
    if(!is.numeric(pred) || !is.vector(pred) || !is.atomic(pred)){
   stop("pred needs to be a numeric vector")
    }
  results <-list()
  if(getRMSE){
    results$RMSE<-getRMSE(n,resp,pred,...)
  }
  if(getMAE){
    results$MAE<- getMAE(n,resp,pred,...)
  }
  return(results)
}
```

6. Run the following code to create some response values and predictions.

```
#Without Missing Values
calculating_wrapper(n,resp,pred)
```

```
$RMSE
[1] 0.9581677

$MAE
[1] 0.8155776
```

```
calculating_wrapper(n,resp,pred,getRMSE=FALSE)
```

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

```
calculating_wrapper(n,resp,pred,getMAE=FALSE)
```

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

```
#With Missing Values
calculating_wrapper(n,resp2,pred2)
```

```
$RMSE
[1] NA

$MAE
[1] NA
```

```r
calculating_wrapper(n,resp2,pred2,na.rm = TRUE)
```

```
$RMSE
[1] 0.9659839

$MAE
[1] 0.8238812
```

```r
#Random Data Frame
df <-data.frame(replicate(10,sample(0:1, 1000, rep = TRUE)))
df2 <-data.frame(replicate(10,sample(0:1, 1000, rep = TRUE)))

calculating_wrapper(1000,df,df2)
```

```
Error in calculating_wrapper(1000, df, df2): resp needs to be numeric vector
```

```r
calculating_wrapper(1000,resp,df2)
```

```
Error in calculating_wrapper(1000, resp, df2): pred needs to be a numeric vector
```

Task 3: Querying an API and tidy-style function

1. Use GET() from the httr package to return information about a topic that you are interested in that has been in the news lately (store the result as an R object).

```r
URL_ids <- "https://newsapi.org/v2/everything?q=NBA&apiKey=6b3329c07b8a4a5081c52478c13ed007"
id_info <-httr::GET(URL_ids)
str(id_info, max.level = 1)
```

```
List of 10
 $ url        : chr "https://newsapi.org/v2/everything?q=NBA&apiKey=6b3329c07b8a4a5081c52478
 $ status_code: int 200
 $ headers    :List of 15
  ..- attr(*, "class")= chr [1:2] "insensitive" "list"
 $ all_headers:List of 1
 $ cookies    :'data.frame':    0 obs. of  7 variables:
 $ content    : raw [1:84163] 7b 22 73 74 ...
 $ date       : POSIXct[1:1], format: "2025-06-25 00:41:57"
 $ times      : Named num [1:6] 0 0.0025 0.0332 0.0664 0.2026 ...
  ..- attr(*, "names")= chr [1:6] "redirect" "namelookup" "connect" "pretransfer" ...
 $ request    :List of 7
  ..- attr(*, "class")= chr "request"
 $ handle     :Class 'curl_handle' <externalptr>
 - attr(*, "class")= chr "response"
```

2. Parse what is returned and find your way to the data frame that has the actual article information in it (check content).

```
-- 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 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter()  masks stats::filter()
x purrr::flatten() masks jsonlite::flatten()
x dplyr::lag()     masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becor
```

```
parsed <- fromJSON(rawToChar(id_info$content))
content <- as_tibble(parsed$articles)
content
```

```
# A tibble: 100 x 8
   source$id $name author title description url   urlToImage publishedAt content
   <chr>     <chr> <chr>  <chr> <chr>       <chr> <chr>      <chr>       <chr>
 1 <NA>      BBC ~ David~ NBA ~ Two-time N~ http~ https://i~ 2025-06-22~ "Two-t~
 2 <NA>      BBC ~ Mande~ Will~ Jalen Will~ http~ https://i~ 2025-06-17~ "Jalen~
 3 the-verge The ~ Jay P~ The ~ Thursday m~ http~ https://p~ 2025-06-12~ "You c~
 4 <NA>      CNET  Matt ~ NBA ~ Discover t~ http~ https://w~ 2025-06-20~ "The N~
 5 <NA>      CNET  Matt ~ NBA ~ Discover t~ http~ https://w~ 2025-06-16~ "The N~
 6 <NA>      CNET  Matt ~ NBA ~ The New Yo~ http~ https://w~ 2025-05-29~ "Can t~
 7 <NA>      CNET  Matt ~ NBA ~ Are the Ok~ http~ https://w~ 2025-05-28~ "Back ~
 8 <NA>      CNET  Matt ~ NBA ~ Discover t~ http~ https://w~ 2025-05-23~ "How w~
 9 <NA>      NPR   Becky~ The ~ It would b~ http~ https://n~ 2025-06-01~ "Oklah~
10 <NA>      CNET  Gael ~ Toda~ Here's tod~ http~ https://w~ 2025-06-04~ "Looki~
# i 90 more rows
```

3. Now write a quick function that allows the user to easily query this API. The inputs to the function should be the title/subject to search for (string), a time period to search from (string - you'll search from that time until the present), and an API key.

```r
API_function <- function(subject,date,key){
  URL <- paste0("https://newsapi.org/v2/everything?q=",
               as.character(subject),
               "&from=",
               date,
               "&apiKey=",
               as.character(key))
  print(URL)
  URL_ids <- URL
id_info <-httr::GET(URL_ids)
str(id_info, max.level = 1)
parsed <- fromJSON(rawToChar(id_info$content))
API_tibble <- as_tibble(parsed$articles)
return(API_tibble)
}

API_function("nhl",2025-06-12,"6b3329c07b8a4a5081c52478c13ed007")
```

```
[1] "https://newsapi.org/v2/everything?q=nhl&from=2007&apiKey=6b3329c07b8a4a5081c52478c13ed0
List of 10
 $ url        : chr "https://newsapi.org/v2/everything?q=nhl&from=2007&apiKey=6b3329c07b8a4a
 $ status_code: int 200
 $ headers    :List of 15
  ..- attr(*, "class")= chr [1:2] "insensitive" "list"
 $ all_headers:List of 1
 $ cookies    :'data.frame':    0 obs. of  7 variables:
 $ content    : raw [1:84458] 7b 22 73 74 ...
 $ date       : POSIXct[1:1], format: "2025-06-25 00:41:57"
 $ times      : Named num [1:6] 0.0 1.3e-05 0.0 1.1e-04 5.6e-02 ...
  ..- attr(*, "names")= chr [1:6] "redirect" "namelookup" "connect" "pretransfer" ...
 $ request    :List of 7
  ..- attr(*, "class")= chr "request"
 $ handle     :Class 'curl_handle' <externalptr>
 - attr(*, "class")= chr "response"

# A tibble: 100 x 8
   source$id $name author title description url   urlToImage publishedAt content
   <chr>     <chr> <chr>  <chr> <chr>       <chr> <chr>      <chr>       <chr>
 1 espn      ESPN  <NA>   Toew~ "Jonathan ~ http~ https://a~ 2025-05-29~ "May 2~
 2 espn      ESPN  Krist~ Jets~ "Winnipeg'~ http~ https://a~ 2025-06-12~ "Winni~
 3 espn      ESPN  Krist~ Over~ "The first~ http~ https://a~ 2025-06-10~ "Expec~
 4 espn      ESPN  <NA>   Foll~ "Live cove~ http~ http://s.~ 2025-06-13~ ""
 5 espn      ESPN  Greg ~ Who ~ "\"Best Vi~ http~ https://a~ 2025-06-18~ "As th~
 6 business~ Busi~ insid~ Less~ "ESPN Plus~ http~ https://i~ 2025-06-16~ "ESPN ~
 7 espn      ESPN  <NA>   Team~ "Buffalo S~ http~ https://a~ 2025-05-25~ "May 2~
 8 espn      ESPN  <NA>   Capi~ "Capitals ~ http~ https://a~ 2025-06-07~ "Spenc~
 9 espn      ESPN  Krist~ Pens~ "The Pitts~ http~ https://a~ 2025-06-04~ "The P~
10 espn      ESPN  ESPN ~ Pant~ "For the s~ http~ https://a~ 2025-05-30~ "The 2~
# i 90 more rows
```