Homework4

Jessie Heise

Task 1: Conceptual Questions

Question 1

The purpose of the lapply() function is to apply a function to every element of a list. The equivalent purrr function is map().

Question 2

lapply(x = my_list, FUN = cor(numeric_matrix), method = "kendall")

Question 3

One advantage of using purrr functions instead of the BaseR apply family is that purrr functions provide a more consistent and clean way to apply functions to objects. Another advantage is that purrr functions allow for shorthand and have additional helper functions.

Question 4

A side-effect function is a type of pipeable function that doesn't actually try to change the data, it just tries to produce something. The print() and plot() functions are examples of side-effect functions.

Question 5

You can name a variable sd in a function and not cause any issues with the sd function because of lexical scoping. Lexical scoping is how R looks up where to get the object from.

Task 2: Writing R Functions

Question 1

```r
#Write getRMSE function
getRMSE <- function(response_input,prediction_input,...){
  RMSE <- sqrt(mean((response_input - prediction_input)^2,...))
}
```

Question 2

```r
#Run code from assignment to create some response values and predictions
set.seed(10)
n <- 100
x <- runif(n)
resp <- 3 + 10*x + rnorm(n)
pred <- predict(lm(resp ~ x), data.frame(x))

#Test RMSE function using this data
testRMSE <- getRMSE(resp,pred)

#Replace two of the response values with missing values (NA_real_)
resp2 <- resp
resp2[c(1,2)] <- c(NA_real_,NA_real_)

#Test getRMSE with specifying behavior to deal with missing values
testRMSE2 <- getRMSE(resp2,pred,na.rm=TRUE)

#Test getRMSE without specifying behavior to deal with missing values
testRMSE3 <- getRMSE(resp2,pred)
```

Question 3

```r
#getMAE() function that evaluates predictions using mean absolute deviation
getMAE <- function(response_input, prediction_input,...){
  MAE <- sqrt(mean(abs(response_input - prediction_input),...))
}
```

Question 4

```r
#Run code from assignment to create some response values and predictions
set.seed(10)
n <- 100
x <- runif(n)
resp <- 3 + 10*x + rnorm(n)
pred <- predict(lm(resp ~ x), data.frame(x))

#Test MAE function using this data
testMAE <- getMAE(resp,pred)

#Replace two of the response values with missing values (NA_real_)
resp2 <- resp
resp2[c(1,2)] <- c(NA_real_,NA_real_)

#Test getMAE with specifying behavior to deal with missing values
testMAE2 <- getMAE(resp2,pred,na.rm=TRUE)

#Test getMAE without specifying behavior to deal with missing values
testMAE3 <- getMAE(resp2,pred)
```

Question 5

```r
#Create a wrapper function to get either or both metrics returned with a
#single function call
wrapper <- function(vector1,vector2,RMSE=TRUE,MAE=TRUE,...){
  if(is.vector(vector1)==FALSE | is.vector(vector2)==FALSE){
    stop("One or more inputs are not in vector form")
  } else{
    if(RMSE==TRUE && MAE==FALSE){
      RMSE <- getRMSE(vector1,vector2,...)
      names(RMSE) <- "RMSE"
      print(RMSE)
    }else if(MAE==TRUE && RMSE==FALSE){
      MAE <- getMAE(vector1,vector2,...)
      names(MAE) <- "MAE"
      print(MAE)
    }else if(MAE==TRUE && RMSE==TRUE){
      RMSE <- getRMSE(vector1,vector2,...)
      MAE <- getMAE(vector1,vector2,...)
      results <- c(RMSE,MAE)
      names(results) <- c("RMSE","MAE")
      print(results)
    }else{print("Specify a metric")}
  }
}
```

Question 6

```r
#Run code from assignment to create some response values and predictions
set.seed(10)
n <- 100
x <- runif(n)
resp <- 3 + 10*x + rnorm(n)
pred <- predict(lm(resp ~ x), data.frame(x))

#Test wrapper function using this data
#Call once asking for each metric individually-RMSE
testwrapper1 <- wrapper(resp,pred,RMSE=TRUE,MAE=FALSE)
```

```
     RMSE
0.9581677
```

```r
#Call once asking for each metric indvidually-MAE
testwrapper2 <- wrapper(resp,pred,RMSE=FALSE,MAE=TRUE)
```

```
      MAE
0.9030933
```

```r
#Call once specifying both metrics
testwrapper3 <- wrapper(resp,pred,RMSE=TRUE,MAE=TRUE)
```

```
     RMSE       MAE
0.9581677 0.9030933
```

```
#Replace two of the response values with missing values (NA_real_)
resp2 <- resp
resp2[c(1,2)] <- c(NA_real_,NA_real_)

#Test wrapper with specifying behavior to deal with missing values
testwrapper4 <- wrapper(resp2,pred,na.rm=TRUE)
```

```
     RMSE       MAE
0.9661699 0.9078106
```

```
#Test wrapper without specifying behavior to deal with missing values
testwrapper5 <- wrapper(resp2,pred)
```

```
RMSE  MAE
  NA   NA
```

```
#Test wrapper by passing it incorrect data
bad_data_1 <- c(1,2,3)
bad_data_2 <- c(4,5,6)
bad_data_frame <- data.frame(bad_data_1,bad_data_2)
#testwrapper6 <- wrapper(bad_data_frame,bad_data_2) #commented out due to error
```

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

Question 1

```
#Use GET() from the httr package to return information about protests in the
#news in the last 30 days.
news_URL_protests <- "https://newsapi.org/v2/everything?q=protests&from=2025-05-19&sortBy=pul
news_protests_return <- httr::GET(news_URL_protests)
```

Question 2

```
#Parse what is returned in question 1 and find the data frame that has the
#actual article information
parsed_news_protests <- jsonlite::fromJSON(rawToChar(news_protests_return$content))
protests <- as_tibble(parsed_news_protests)
```

Question 3

```r
#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), and an API key
API_query <- function(subject,time_period,key){
  news_URL <- "https://newsapi.org/v2/everything?q="
  query_URL <- paste0(news_URL,subject,"&from=",time_period,
                      "&sortBy=publishedAt&apiKey=",key)
  news_return <- httr::GET(query_URL)
  parsed_news <- jsonlite::fromJSON(rawToChar(news_return$content))
  news <- as_tibble(parsed_news)
}

query_test <- API_query("gamestop","2025-05-19","24187cd558dc45b9a6b10d1891bdd396")
```