Assignment B-1

Charlotte Edgar

2023-10-24

```
library(testthat)
```

Function

In this assignment we will make a function to find the mean and standard deviation of multiple columns of a dataset at once and print them in a table. This function can be used in an exploratory data analysis and can save time by looking at all numeric columns of a dataset at once instead of individually.

```
#' @title Mean of columns
#' @details This function takes the mean and standard deviation of each numeric column in a dataset
#' @param your_data the dataset containing only numeric columns, named your_data since data is a functi
#' @return returns a data frame with one column for the column means and one for the column standard de

mean_sd_of_cols <- function(your_data){
   means <- c()
   sds <- c()
   your_data <- as.data.frame(your_data)

  for(i in 1:ncol(your_data)){
    if(!is.numeric(your_data[,i])) {
    stop('I am so sorry, but this function only works for numeric input!\n',
         'Your provided dataset has at least one column of class: ', class(your_data[,i])[1])
  }
    means[i] <- mean(your_data[,i], na.rm= TRUE)
    sds[i] <- sd(your_data[,i],na.rm=TRUE)
  }
   result <- data.frame(cbind(means,sds))
   colnames(result) <- c("Mean", "Std. Dev.")
   rownames(result) <- paste0("Column", 1:ncol(your_data))
   return(result)
}
```

Examples

Examples to show how our new function works.

We begin with a simple example where the input is a 3 x 3 matrix.

```
dataset1 <- matrix(c(1,2,3,4,5,6,7,8,9),3,3)
dataset1
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

We apply our function to this matrix.

```
mean_sd_of_cols(dataset1)
```

```
##         Mean Std. Dev.
## Column1    2         1
## Column2    5         1
## Column3    8         1
```

The output is as expected where the means are in the first column and the standard deviations are in the second column. Each row is a column from the dataset, hence we have 3 rows.

The second example starts by creating a new dataset with the first column being odd numbers starting at 1 and ending at 99. The second column is created using seq() from 25 to 50 and the third using seq() from 1 to 4. All columns have length 50. This gives us a variety of ranges in the columns. Note that the input in this example is now a data frame instead of a matrix to show the different inputs that our function can handle.

```
dataset2 <- data.frame(cbind( seq(1,100,by = 2), seq(25,50, length = 50), seq(1,4, length = 50)))
head(dataset2)
```

```
##   X1       X2       X3
## 1  1 25.00000 1.000000
## 2  3 25.51020 1.061224
## 3  5 26.02041 1.122449
## 4  7 26.53061 1.183673
## 5  9 27.04082 1.244898
## 6 11 27.55102 1.306122
```

We will use our newly created function to see the means and standard deviations of each column.

```
mean_sd_of_cols(dataset2)
```

```
##         Mean  Std. Dev.
## Column1 50.0 29.1547595
## Column2 37.5  7.4374386
## Column3  2.5  0.8924926
```

The output of our second example is also as expected where the means are in the first column and the standard deviations are in the second column. Each row is a new column of the dataset, hence we have 3 rows.

The third example will demonstrate what happens if a dataset including a non-numeric column is inputted as the argument into our new function. This simple dataset is made with 2 columns where the first column is numeric using seq() from 1 to 10 and the second is a vector of characters.

```
dataset3 <- data.frame(numbers = seq(1,10,length = 26), letters = paste0(letters))
head(dataset3)
```

```
##   numbers letters
## 1    1.00       a
## 2    1.36       b
## 3    1.72       c
## 4    2.08       d
## 5    2.44       e
## 6    2.80       f
```

We apply our function to this dataset.

```
mean_sd_of_cols(dataset3)
```

```
## Error in mean_sd_of_cols(dataset3): I am so sorry, but this function only works for numeric input!
## Your provided dataset has at least one column of class: character
```

We get an error as expected because the input has to have all numeric columns. The error message explains this. This would encourage the user to subset their dataset so that only numeric columns are inputted as an argument to the new function.

Testing the function

```
test_that("Testing numeric examples", {
  expect_equal(mean_sd_of_cols(cbind(rep(1,10), rep(5,10)))$Mean, c(1,5) )
  expect_equal(mean_sd_of_cols(cbind(rep(1,10), rep(5,10)))$"Std. Dev.", c(0,0) )
})
```

```
## Test passed
```

```
test_that("Testing number of columns and rows printed",{
  expect_equal(ncol(mean_sd_of_cols(cbind(rep(1,10), rep(5,10)))), 2 )
  expect_equal(nrow(mean_sd_of_cols(cbind(rep(1,10), rep(5,10)))), ncol(cbind(rep(1,10), rep(5,10))) )
})
```

```
## Test passed
```

```
test_that("Testing number of columns and rows printed",{
  expect_equal(ncol(mean_sd_of_cols(matrix(rnorm(100), 25, 4))), 2 )
  expect_equal(nrow(mean_sd_of_cols(matrix(rnorm(100), 25, 4))), 4)
})
```

```
## Test passed
```

```
test_that("Testing a dataset with a cateogrical variable",{
  expect_error(mean_sd_of_cols(cbind("red","blue","yellow")), "I am so sorry, but this function only wo
})
```

```
## Test passed
```

```
data_w_1_NAs <- cbind(rep(1,10), rep(5,10))
data_w_1_NAs[2,2] <- NA

test_that("Testing a dataset with NA's",{
  expect_equal(mean_sd_of_cols(data_w_1_NAs)$Mean, c(1,5) )
})
```

```
## Test passed
```

```
data_w_all_NAs <- cbind(rep(NA, 10), rep(5,10))

test_that("Testing a dataset with all NA's (mean)",{
  expect_equal(mean_sd_of_cols(data_w_all_NAs)$Mean, c(NA,5) )
})
```

```
## Test passed
```

```
test_that("Testing a dataset with all NA's (Std. Dev.)",{
  expect_equal(mean_sd_of_cols(data_w_all_NAs)$"Std. Dev.", c(NA,0) )
})
```

```
## Test passed
```

The tests all pass as expected. Notice that if we have an NA in one of our columns the function we made removes this value when calculating the mean and standard deviation. However, if the a column of all NA values is entered, then the mean and standard deviation for this column is also NA. However, no error is returned as the function still returns the mean and standard deviations of other numeric columns inputted.