Applied Statistical Programming - Functions

2/2/2022

Names: Alma, Cassandra, Messi, Zion

Write the R code to answer the following questions. Write the code and then show what the computer returns when that code is run. Thoroughly comment your code describing your thought process when answering the questions.

You have until the beginning of class 2/7 at 10:00am to answer all of the questions below. You may use R, but not any online documentation. Submit the Rmarkdown and the knitted PDF to Canvas. Only one member of your group needs to submit the in class exercise.

1. Use the following code block to generate data x and y along with a fitted model.

```
set.seed(12345)
n <- 20  # 20 observations in the sample
x <- runif(n, min = -1, max = 1)  # Sample uniformly from [-1,1]
noise <- rnorm(n, mean = 0, sd = 0.3)  # We have a noisy sample
y <- 1 + 3 * x + noise  # generate the response variable
df <- as.data.frame(matrix(data = cbind(y, x), nrow = n, ncol = 2))
colnames(df) <- c("y", "x")
model <- lm(y ~ x, data = df)
summary(model)
```

```
##
## Call:
## lm(formula = y ~ x, data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.51908 -0.20634 -0.00541  0.23767  0.49204
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.05461    0.06806   15.50 7.47e-12 ***
## x            3.06821    0.11258   27.25 4.36e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3043 on 18 degrees of freedom
## Multiple R-squared:  0.9763, Adjusted R-squared:  0.975
## F-statistic: 742.7 on 1 and 18 DF,  p-value: 4.36e-16
```

a. Create a function that reports the root mean squared error (RMSE) of model predictions. Your function must only take vectors the x, y, and `lm()` objects as inputs. For N observations indexed by i, the equation for RMSE is provided.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (\hat{y}_i - y_i)^2}{N}}$$

```r
rmse <- function(x, y, model) {
    N <- length(x)
    rmse <- sqrt((sum((model$fitted.values - y)^2))/N)
    return(list(RMSE = rmse))
}

rmse(x, y, model)
```

```
## $RMSE
## [1] 0.2886809
```

b. If x and y are not the same length, report an error instead of calculating the RMSE.

```r
rmse <- function(x, y, model) {
    N = length(x)
    length_test <- N == length(y)
    if (!length_test) {
        stop("Error. Length of x and y not same.")
    } else {
        rmse <- sqrt((sum((model$fitted.values - y)^2))/N)
        return(list(RMSE = rmse))
    }
}


x.1 <- x[-1]
rmse(x.1, y)
```

c. If either x or y are non-numeric, report an error instead of calculating the RMSE.

```r
rmse <- function(x, y, model) {
    N = length(x)
    length_test <- N == length(y)
    class_test <- is.numeric(x) & is.numeric(y)

    if (!length_test) {
        stop("Error. Length of x and y not same.")
    } else if (!class_test) {
        stop("Error. Both x and y must be numeric")
    } else {
        rmse <- sqrt((sum((model$fitted.values - y)^2))/N)
        return(list(RMSE = rmse))
    }
}


x.2 <- as.character(x)
rmse(x.2, y, model)
```

d. Add a parameter `likelihood` to your function so that when set to `TRUE` the function also returns the likelihood of the data in the list.

$$L(\hat{y}_1, ..., \hat{y}_N, \sigma^2; y_1, ..., y_N) = \left(2\pi\sigma^2\right)^{-N/2} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2\right)$$

Use the y provided, the fitted y values from the `lm()` model object, and use $\sigma^2 = 0.09$. Be sure to set a default value for `likelihood`.

```
rmse <- function(x, y, model, likelihood = FALSE, sigma_squared = 0.09) {
    N = length(x)
    length_test <- N == length(y)
    class_test <- is.numeric(x) & is.numeric(y)

    if (!length_test) {
        stop("Error. Length of x and y not same.")
    } else if (!class_test) {
        stop("Error. Both x and y must be numeric")
    } else {
        rmse = sqrt(sum((model$fitted.values - y)^2)/N)
        if (likelihood) {
            lh.value <- (2 * pi * sigma_squared)^(-N/2) * exp((-1/(2 * sigma_squared) *
                sum((model$fitted.values - y)^2)))
            return(list(RMSE = rmse, Likelihood = lh.value))
        }
        return(list(RMSE = rmse))
    }
}


rmse(x, y, model, TRUE)
```

```
## $RMSE
## [1] 0.2886809
##
## $Likelihood
## [1] 0.02846882
```

e. Add a parameter `logL` to your function so that when set to `TRUE` the function also returns the log likelihood of the data in the list.

$$\ln L(\hat{y}_1, ..., \hat{y}_N, \sigma^2; y_1, ..., y_N) = -\frac{N}{2}\ln(2\pi) - \frac{N}{2}\ln(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

Use the y provided, the fitted y values from the `lm()` model object, and use $\sigma^2 = 0.09$. Be sure to set a default value for `logL`.

```
rmse <- function(x, y, model, likelihood = FALSE, logL = FALSE, sigma_squared = 0.09) {
    N = length(x)
    length_test <- N == length(y)
    class_test <- is.numeric(x) & is.numeric(y)

    if (!length_test) {
        stop("Error. Length of x and y not same.")
    } else if (!class_test) {
        stop("Error. Both x and y must be numeric")
    } else {
        rmse = sqrt(sum((model$fitted.values - y)^2)/N)
        if (likelihood) {
            lh.value <- (2 * pi * sigma_squared)^(-N/2) * exp((-1/(2 * sigma_squared) *
                sum((model$fitted.values - y)^2)))
            if (logL) {
                log.lh.value <- -(N/2) * log(2 * pi) - (N/2) * log(sigma_squared) -
```

```
                  1/(2 * sigma_squared) * sum((model$fitted.values - y)^2)
                return(list(RMSE = rmse, Likelihood = lh.value, Log_Likelihood = log.lh.value))
            }
            return(list(RMSE = rmse, Likelihood = lh.value))
        }
        return(list(RMSE = rmse))
    }
}



rmse(x, y, model, TRUE, TRUE)
```

```
## $RMSE
## [1] 0.2886809
##
## $Likelihood
## [1] 0.02846882
##
## $Log_Likelihood
## [1] -3.558946
```