FUNCTION CODE IN R

I created a function to run multiple linear regression in R called **mylm()** and the function code is as follows:

```
mylm <- function(data, outcome, predictors) {
  ## checks if data inputted is a data frame
  if (!is.data.frame(data)) {
    stop("argument 'data' must be a data frame")
  }
  ## checks if outcome exists in data
  if (!(outcome %in% names(data))) {
    stop(paste("outcome", outcome, "is not a column name found in the data\n",
               "(syntax is case-sensitive)"))
  }
  ## checks if predictors exist in data
  if (!all(predictors %in% names(data))) {
    stop(paste("at least one of the predictors",
               "is not a column name found in the data\n",
               "(syntax is case-sensitive)"))
  }
  ## checks for NAs in outcome and predictors
  if (any(is.na(data[[outcome]]))) {
    stop(paste("outcome", outcome, "contains missing values",
               "\nremove NAs to run function"))
  }
  for (onepred in predictors) {
    if (any(is.na(data[[onepred]]))) {
      stop(paste("predictor", onepred, "contains missing values",
                 "\nremove NAs to run function"))
    }
  }

  ## inputting the variables as y and x
  y <- data[[outcome]]
  x <- data.frame(Intercept=rep(1,nrow(data)))
  for (onepred in predictors) {
    x[[onepred]] <- data[[onepred]]
  }
  x <- as.matrix(x)

  ## print formula
  formula <- paste(outcome, "~", paste(predictors, collapse = " + "))
  cat("Linear Regression Formula:", formula, "\n")

  ## beta = ((X'X)^-1)X'Y
  beta <- solve(t(x)%*%x)%*%(t(x)%*%y)
  colnames(beta) <- c("Beta Estimate")
```

```r
## RSS = y'y - b'X'y
RSS <- t(y)%*%y - t(beta)%*%t(x)%*%y
## s2 = RSS/(n-p-1)
s2 <- RSS/(length(y)-ncol(x))
## variance matrix
var <- as.numeric(s2)*(solve(t(x)%*%x))
## standard error
se <- sqrt(diag(var))
se <- data.frame(se)
colnames(se) <- c("Standard Error")

## t-test
tstat <- beta / se
tstat <- data.frame(tstat)
colnames(tstat) <- c("T-Stat")
## p-value
tstat2 <- unlist(tstat)
df <- length(y) - ncol(x)
pval <- 2*pt(as.numeric(abs(tstat2)), df=df, lower.tail=F)
pval <- data.frame(pval)
colnames(pval) <- c("P-Value")

## print coefficients
coeff <- data.frame(round(beta, digits = 5), round(se, digits = 5),
                    round(tstat, digits = 5), pval)
cat("\nCoefficients:\n")
print(coeff)

## print degrees of freedom
cat("\nDegrees of Freedom for T-Test:", df)

## finding R2 and adj R2
## SYY = y'y - n*y_bar^2
SYY <- t(y)%*%y - length(y)*mean(y)**2
## R^2 = 1 - RSS/SYY
R2 <- 1 - RSS/SYY
## R^2 adjusted = ((n-1)R2 - p) / (n-p-1)
R2.adj <- ((length(y)-1)*R2 - (ncol(x)-1)) / (length(y)-ncol(x))

## print R2 and adj R2
cat("\nR-Squared:", round(R2, digits = 5), "and",
    "Adjusted R-Squared:", round(R2.adj, digits = 5), "\n")
```

```
  ## f-test
  df1 = (ncol(x)-1)
  df2 = (length(y)-ncol(x))
  num <- (SYY - RSS) / df1
  den <- RSS / df2
  fstat <- num/den
  pval_f <- 1-pf(fstat, df1, df2)

  ## print f-stat and p-value
  cat("F-Statistic:", round(fstat, digits = 5), "on", df1, "and", df2, "DF,",
      "p-value:", pval_f, "\n")

}
```

FUNCTION ARGUMENTS (INSTRUCTIONS)
To run the mylm() function, there are three arguments that must be inputted:

- **data** – a data frame of the data
- **outcome** – a character string of the column name from the data that is the outcome variable of interest
    - only one outcome variable can be inputted
- **predictors** – a character vector of the column name(s) from the data that is/are the predictor variable(s) of interest
    - any number of predictor variables can be inputted

FUNCTION RETURNS
The mylm() function returns the following output (additional functionalities are in red):

- The inputted outcome and predictor variables written as a formula
- Coefficients including the beta estimate, standard error, t-test statistic, and p-value of t-test statistic for all predictors and the intercept
- Degrees of freedom for the t-test
- R-squared goodness of fit measure and adjusted R-squared goodness of fit measure
- F-test results including the F-statistic, degrees of freedom, and p-value
- Error messages for if data is not a data frame, outcome does not exist in data, predictors does not exist in data, there are missing values in outcome or predictors
    - See page 6 for examples (“FUNCTION CHECKS FOR ERRORS”)

FUNCTION EXAMPLE

As an example run, I ran the mylm() function with a built-in dataset in R called USArrests:

```
> data(USArrests)
> mylm(data = USArrests, outcome = "UrbanPop", predictors = c("Murder", "Assault", "Rape"))
Linear Regression Formula: UrbanPop ~ Murder + Assault + Rape

Coefficients:
          Beta.Estimate Standard.Error   T.Stat      P.Value
Intercept      52.84187        4.82483 10.95207 2.086265e-14
Murder         -1.41154        0.71954 -1.96173 5.586128e-02
Assault         0.05190        0.04161  1.24742 2.185567e-01
Rape            0.69841        0.26776  2.60833 1.223030e-02

Degrees of Freedom for T-Test: 46
R-Squared: 0.2337 and Adjusted R-Squared: 0.18373
F-Statistic: 4.67629 on 3 and 46 DF, p-value: 0.006207528
```

To run a multiple linear regression on this dataset, I inputted the following arguments:

- data = USArrests
  - The dataset is USArrests.
- outcome = “UrbanPop”
  - The column name of my outcome variable of interest in the dataset is called “UrbanPop”.
  - It must be inputted as a character string, meaning the column name of the outcome variable should be in quotations.
- predictors = c(“Murder”, “Assault”, “Rape”)
  - The column names of my predictor variables of interest in the dataset are called “Murder”, “Assault”, and “Rape”.
  - It must be inputted as a character vector, meaning the column names of the predictor variables should each be in quotations and inside of the concatenate function c().

After running the function, the output shows:

- The inputted outcome and predictor variables written as a formula
  - `Linear Regression Formula: UrbanPop ~ Murder + Assault + Rape`
- Coefficients including the beta estimate, standard error, t-test statistic, and p-value of t-test statistic for all predictors and the intercept
  -
```
Coefficients:
          Beta.Estimate Standard.Error   T.Stat      P.Value
Intercept      52.84187        4.82483 10.95207 2.086265e-14
Murder         -1.41154        0.71954 -1.96173 5.586128e-02
Assault         0.05190        0.04161  1.24742 2.185567e-01
Rape            0.69841        0.26776  2.60833 1.223030e-02
```
- Degrees of freedom for the t-test
  - `Degrees of Freedom for T-Test: 46`
- R-squared goodness of fit measure and adjusted R-squared goodness of fit measure
  - `R-Squared: 0.2337 and Adjusted R-Squared: 0.18373`
- F-test results including the F-statistic, degrees of freedom, and p-value
  - `F-Statistic: 4.67629 on 3 and 46 DF, p-value: 0.006207528`

FUNCTION COMPARISON TO LM()

To compare mylm() function to the existing multiple linear regression function lm() in R, I ran a linear model on the same dataset (USArrests) with the same outcome (UrbanPop) and same predictors (Murder, Assault, Rape). I ran a summary of that linear model using lm().

Existing lm() Output:

```
> summary(lm(UrbanPop ~ Murder + Assault + Rape, data = USArrests))

Call:
lm(formula = UrbanPop ~ Murder + Assault + Rape, data = USArrests)

Residuals:
    Min      1Q  Median      3Q     Max
-35.456  -6.950   0.077   7.770  25.221

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 52.84187    4.82483  10.952 2.09e-14 ***
Murder      -1.41154    0.71954  -1.962   0.0559 .
Assault      0.05190    0.04161   1.247   0.2186
Rape         0.69841    0.26776   2.608   0.0122 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 13.08 on 46 degrees of freedom
Multiple R-squared:  0.2337,    Adjusted R-squared:  0.1837
F-statistic: 4.676 on 3 and 46 DF,  p-value: 0.006208
```

mylm() Output:

```
> mylm(data = USArrests, outcome = "UrbanPop", predictors = c("Murder",
"Assault", "Rape"))
Linear Regression Formula: UrbanPop ~ Murder + Assault + Rape

Coefficients:
          Beta.Estimate Standard.Error   T.Stat      P.Value
Intercept      52.84187        4.82483 10.95207 2.086265e-14
Murder         -1.41154        0.71954 -1.96173 5.586128e-02
Assault         0.05190        0.04161  1.24742 2.185567e-01
Rape            0.69841        0.26776  2.60833 1.223030e-02

Degrees of Freedom for T-Test: 46
R-Squared: 0.2337 and Adjusted R-Squared: 0.18373
F-Statistic: 4.67629 on 3 and 46 DF, p-value: 0.006207528
```

The function calling are similar except the existing lm() function does not need to have its variables inputted as strings like the mylm() function. The outputs of the two functions are very

similar. All the coefficients, degrees of freedom, R-squared, adjusted R-squared, and F-test results are the same values except for differences in rounding. The formatting between the two functions is also very similar. The existing lm() function has a few more outputs than the mylm() function, including the residuals, significance codes for p-values, and residual standard error.

FUNCTION CHECKS FOR ERRORS

I added a few checks for errors in the mylm() function, including:

- Checking if the data inputted is a data frame

```
> wrong <- c(1, 2, 3)
> mylm(data = wrong, outcome = "UrbanPop", predictors = c("Murder",
"Assault", "Rape"))
Error in mylm(data = wrong, outcome = "UrbanPop", predictors = c("Mur
der",  :
  argument 'data' must be a data frame
```

  - In this example, the data inputted is a numeric vector so the function stops and outputs an error message that the data must be a data frame.
- Checking if the outcome exists in the data

```
> mylm(data = USArrests, outcome = "Urban", predictors = c("Murder",
"Assault", "Rape"))
Error in mylm(data = USArrests, outcome = "Urban", predictors = c("Mur
der",  :
  outcome Urban is not a column name found in the data
 (syntax is case-sensitive)
```

  - In this example, the outcome inputted is purposely entered in wrong as “Urban” instead of “UrbanPop”. Since “Urban” does not exist as a column name in the data, the function stops and outputs an error message that the inputted outcome does not exist in the data.
- Checking if the predictors exist in the data

```
> mylm(data = USArrests, outcome = "UrbanPop", predictors = c("Wrong",
"Assault", "Rape"))
Error in mylm(data = USArrests, outcome = "UrbanPop", predictors = c("Wr
ong",  :
  at least one of the predictors is not a column name found in the data
 (syntax is case-sensitive)
> mylm(data = USArrests, outcome = "UrbanPop", predictors = c("Wrong",
"Wrong", "Rape"))
Error in mylm(data = USArrests, outcome = "UrbanPop", predictors = c("Wr
ong",  :
  at least one of the predictors is not a column name found in the data
 (syntax is case-sensitive)
```

  - In this example, the predictors are purposely entered in wrong as “Wrong” instead of a column name that actually exists in the data. If there is at least one predictor that is not found the data, then the function stops and outputs an error message that at least one of the predictors does not exist in the data.
- Checking if there are missing values in the outcome and predictors

```
> USArrests$UrbanPop[1] <- NA
> mylm(data = USArrests, outcome = "UrbanPop", predictors = c("Murder",
"Assault", "Rape"))
Error in mylm(data = USArrests, outcome = "UrbanPop", predictors = c("Mur
der",  :
  outcome UrbanPop contains missing values
remove NAs to run function
```

- In this example, the first observation for the outcome "UrbanPop" is purposely made to be a missing value. Trying to run the mylm() function with the same outcome and predictors as in the original example does not work because the outcome now has a missing value. Thus, the function stops and outputs an error message that the outcome contains missing values and that the missing values need to be removed to run the mylm() function.

```
> USArrests$Murder[1] <- NA
> mylm(data = USArrests, outcome = "UrbanPop", predictors = c("Murder",
"Assault", "Rape"))
Error in mylm(data = USArrests, outcome = "UrbanPop", predictors = c("Mur
der",  :
  predictor Murder contains missing values
remove NAs to run function
```

- After resetting the data back to the original dataset, this example shows an error after the first observation of the predictor "Murder" is purposely made to be a missing value. Since the predictor "Murder" now has a missing value, the function stops and outputs an error message that the predictor "Murder" contains missing values and that the missing values need to be removed to run the mylm() function.

ADDITIONAL EXAMPLE

The following displays an additional example of the mylm() function with a different built-in dataset in R called "quakes". The example uses "mag" as the outcome variable and "lat", "long", and "depth" as the predictor variables. The same data, outcome variable, and predictor variables were used to run a liner model in the existing lm() function for comparison.

mylm() Output:

```
> data(quakes)
> mylm(data = quakes, outcome = "mag", predictors = c("lat", "long", "depth"))
Linear Regression Formula: mag ~ lat + long + depth

Coefficients:
          Beta.Estimate Standard.Error   T.Stat      P.Value
Intercept       6.75327        0.37433 18.04106 3.553392e-63
lat            -0.00894        0.00262 -3.41254 6.695004e-04
long           -0.01226        0.00219 -5.59378 2.870034e-08
depth          -0.00037        0.00006 -6.51399 1.159421e-10

Degrees of Freedom for T-Test: 996
R-Squared: 0.08385 and Adjusted R-Squared: 0.08109
F-Statistic: 30.38712 on 3 and 996 DF, p-value: 0
```

Existing lm() Output:

```
> summary(lm(mag ~ lat + long + depth, data = quakes))

Call:
lm(formula = mag ~ lat + long + depth, data = quakes)

Residuals:
     Min       1Q   Median       3Q      Max
-0.74051 -0.29109 -0.06593  0.21310  1.61074

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.753e+00  3.743e-01  18.041  < 2e-16 ***
lat         -8.939e-03  2.619e-03  -3.413  0.00067 ***
long        -1.226e-02  2.192e-03  -5.594 2.87e-08 ***
depth       -3.746e-04  5.751e-05  -6.514 1.16e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.3861 on 996 degrees of freedom
Multiple R-squared:  0.08385,   Adjusted R-squared:  0.08109
F-statistic: 30.39 on 3 and 996 DF,  p-value: < 2.2e-16
```

Again, the outputs are very similar, except for the lm() function having a few more outputs compared to the mylm() function and a slight difference in rounding for the same output values.