Linear Regression Coding Assignment-2

```r
# Load essential libraries
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.3.2
```

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(HSAUR)
```

```
## Warning: package 'HSAUR' was built under R version 4.3.2

## Loading required package: tools
```

```r
library(ggcorrplot)
```

```
## Warning: package 'ggcorrplot' was built under R version 4.3.2
```

```r
# Load the heptathlon dataset
data(heptathlon)
str(heptathlon)
```

```
## 'data.frame':    25 obs. of  8 variables:
##  $ hurdles : num  12.7 12.8 13.2 13.6 13.5 ...
##  $ highjump: num  1.86 1.8 1.83 1.8 1.74 1.83 1.8 1.8 1.83 1.77 ...
##  $ shot    : num  15.8 16.2 14.2 15.2 14.8 ...
##  $ run200m : num  22.6 23.6 23.1 23.9 23.9 ...
##  $ longjump: num  7.27 6.71 6.68 6.25 6.32 6.33 6.37 6.47 6.11 6.28 ...
##  $ javelin : num  45.7 42.6 44.5 42.8 47.5 ...
##  $ run800m : num  129 126 124 132 128 ...
##  $ score   : int  7291 6897 6858 6540 6540 6411 6351 6297 6252 6252 ...
```

```r
# Introduce a new column called sprint highlighting slow and fast sprinters
heptathlon = heptathlon %>% mutate(sprint = ifelse(run200m <= 25 & run800m <= 129, 'fast', 'slow'))
str(heptathlon)
```

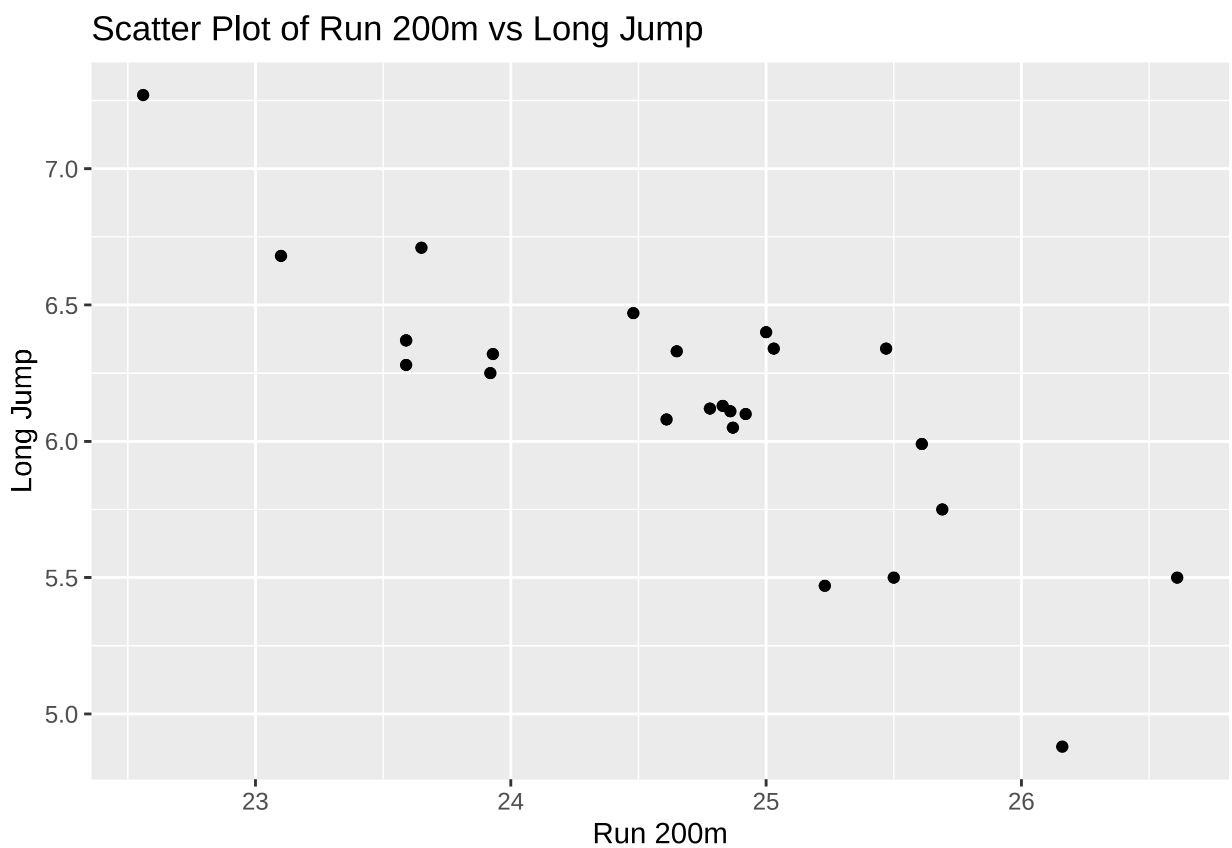
```
## 'data.frame':    25 obs. of  9 variables:
##  $ hurdles : num  12.7 12.8 13.2 13.6 13.5 ...
##  $ highjump: num  1.86 1.8 1.83 1.8 1.74 1.83 1.8 1.8 1.83 1.77 ...
##  $ shot    : num  15.8 16.2 14.2 15.2 14.8 ...
##  $ run200m : num  22.6 23.6 23.1 23.9 23.9 ...
##  $ longjump: num  7.27 6.71 6.68 6.25 6.32 6.33 6.37 6.47 6.11 6.28 ...
```

```
##  $ javelin : num  45.7 42.6 44.5 42.8 47.5 ...
##  $ run800m : num  129 126 124 132 128 ...
##  $ score   : int  7291 6897 6858 6540 6540 6411 6351 6297 6252 6252 ...
##  $ sprint  : chr  "fast" "fast" "fast" "slow" ...
```

```r
# Change sprint column to factor type
heptathlon['sprint'] = lapply(heptathlon['sprint'],as.factor)
str(heptathlon)
```

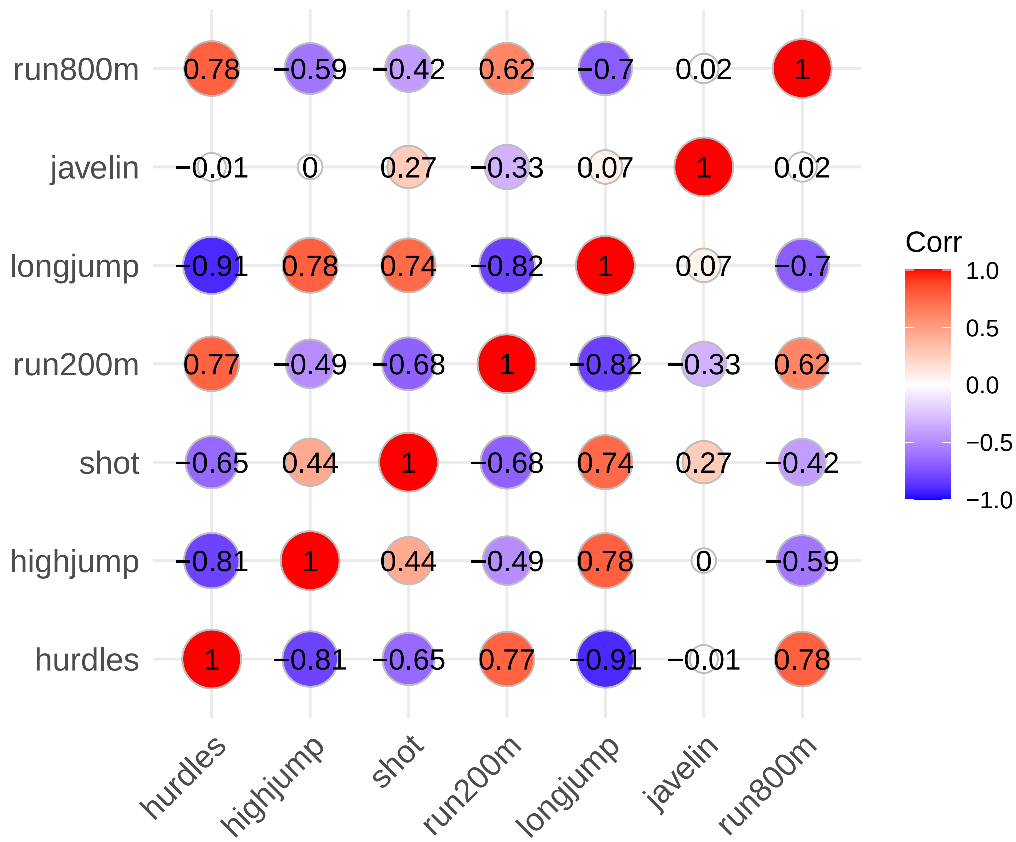
```
## 'data.frame':    25 obs. of  9 variables:
##  $ hurdles : num  12.7 12.8 13.2 13.6 13.5 ...
##  $ highjump: num  1.86 1.8 1.83 1.8 1.74 1.83 1.8 1.8 1.83 1.77 ...
##  $ shot    : num  15.8 16.2 14.2 15.2 14.8 ...
##  $ run200m : num  22.6 23.6 23.1 23.9 23.9 ...
##  $ longjump: num  7.27 6.71 6.68 6.25 6.32 6.33 6.37 6.47 6.11 6.28 ...
##  $ javelin : num  45.7 42.6 44.5 42.8 47.5 ...
##  $ run800m : num  129 126 124 132 128 ...
##  $ score   : int  7291 6897 6858 6540 6540 6411 6351 6297 6252 6252 ...
##  $ sprint  : Factor w/ 2 levels "fast","slow": 1 1 1 2 1 1 2 2 2 2 ...
```

```r
# Make a scatter plot between *run200m* (x-axis) and *longjump* (y-axis). What do you observe from this
ggplot(heptathlon, aes(x = run200m, y = longjump)) +
  geom_point() +
  labs(x = "Run 200m", y = "Long Jump",
  title = "Scatter Plot of Run 200m vs Long Jump")
```




```r
# there is no correlation between these two variables as the points are scattered without any distinct
```

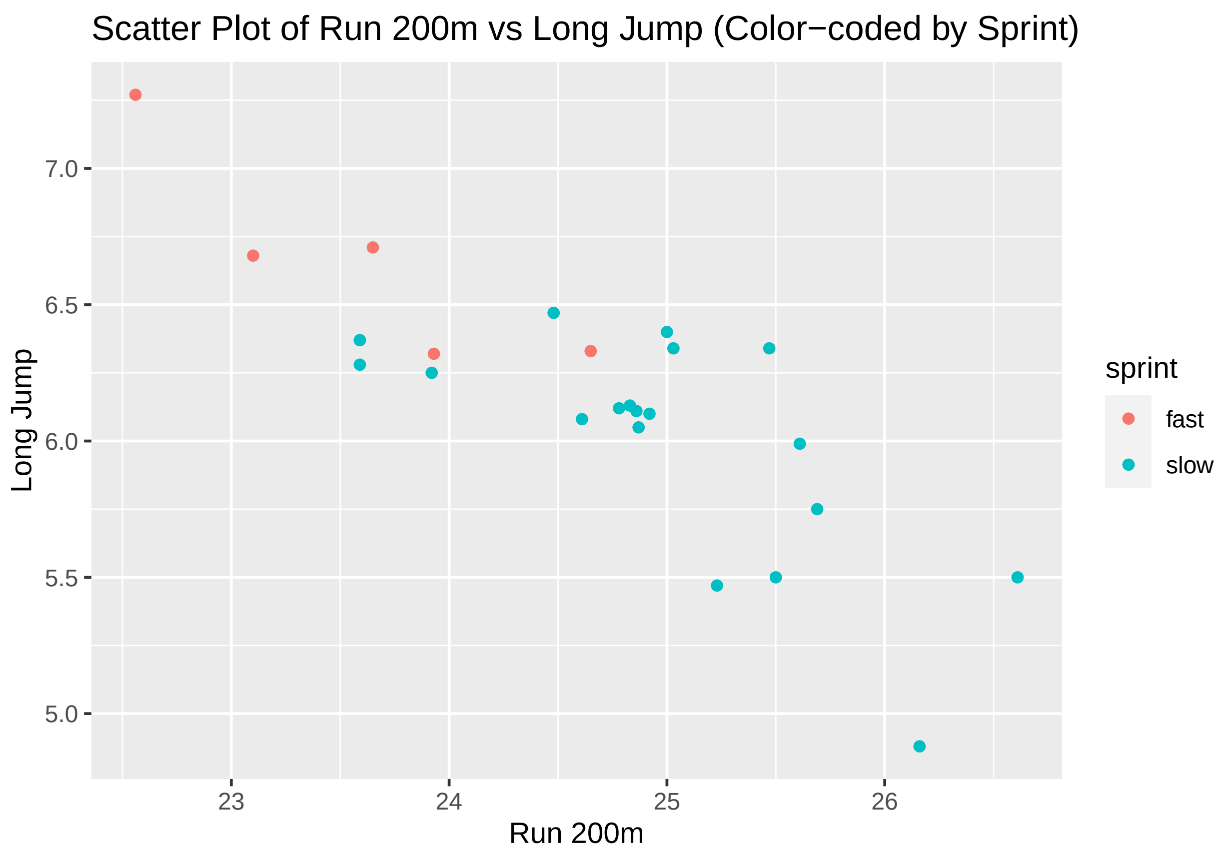
```r
# Correlation between all pairs of continuous predictors (leave out sprint and the response variable sc
cor_matrix = cor(heptathlon %>% select(-c(sprint, score)))
ggcorrplot(cor_matrix, method = 'circle', lab = TRUE)
```




```r
# there is a strong positive correlations between high jump , hurdles and long jump, run200m.
# there is strong negative correlation between run800m and shotput
```

```r
# Make a scatter plot between *run200m* (x-axis) and *longjump* (y-axis) now with the data points color

ggplot(heptathlon, aes(x = run200m, y = longjump, color = sprint)) +
  geom_point() +
  labs(x = "Run 200m", y = "Long Jump",
       title = "Scatter Plot of Run 200m vs Long Jump (Color-coded by Sprint)")
```




```r
# there are more athletes with slow sprint speed than fast sprint speed indicating that most of them ha
```

```r
# Calculate Pearson's correlation between *run200m* and *longjump*. What do you observe?
cor(heptathlon$run200m, heptathlon$longjump)
```

```
## [1] -0.8172053
```

```r
# both the variables are strongly negatively correlated indicating that if one increases the other valu
```

```r
# How many levels does the categorical variable *sprint* have? What is the reference level?
contrasts(heptathlon$sprint)
```

```
##      slow
## fast    0
## slow    1
```

```r
sprint_factor <- factor(c("slow", "fast"))

cat("Levels in 'sprint':", levels(sprint_factor), "\n")
```

```
## Levels in 'sprint': fast slow
```

```r
cat("Reference level in 'sprint':", levels(sprint_factor)[1], "\n")
```

```
## Reference level in 'sprint': fast
```

```r
# Fit a linear model for approximating *score* as a function of *sprint*. Print the model's summary. Ho
model <- lm(score ~ sprint, data = heptathlon)
summary(model)
```

```
##
## Call:
## lm(formula = score ~ sprint, data = heptathlon)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1347.4  -227.4    97.6   291.6   626.6
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   6799.4      200.3  33.939  < 2e-16 ***
## sprintslow    -886.0      224.0  -3.956 0.000628 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 448 on 23 degrees of freedom
## Multiple R-squared:  0.4049, Adjusted R-squared:  0.379
## F-statistic: 15.65 on 1 and 23 DF,  p-value: 0.0006282
```

```r
mean_slow = mean(heptathlon[heptathlon$sprint == 'slow', 'score'])
mean_fast = mean(heptathlon[heptathlon$sprint == 'fast', 'score'])
mean_fast
```

```
## [1] 6799.4
```

```r
mean_slow-mean_fast
```

```
## [1] -886
```

```r
# it deviates about 448 units from the average scores
# model explains only 40.49% of the variance in score
# p value is very low which means sprint has a impact on scores
# athletes with on slow level on an average have scores approximately 886 units less than those on fast
```

```r
# Fit a linear model for approximating *score* as a function of *shot* and *sprint*. Print the model's 

# 1. Did the addition of the new predictor *shot* improve the model accuracy?
# 2. *True/false* (explain in one line): the model suggests that there is a possible linear relationshi
# 3. For a 1 metre increase in shot put throw and with the same sprint performance, we can say with 95%
model = lm(score ~ shot + sprint, data = heptathlon)
summary(model)
```

```
##
## Call:
## lm(formula = score ~ shot + sprint, data = heptathlon)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1124.58  -164.40    35.93   207.34   496.35
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3080.0      883.0   3.488 0.002084 **
## shot           249.7       58.4   4.275 0.000308 ***
## sprintslow    -330.4      213.4  -1.548 0.135842
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 338.5 on 22 degrees of freedom
## Multiple R-squared:  0.6749, Adjusted R-squared:  0.6454
## F-statistic: 22.84 on 2 and 22 DF,  p-value: 4.282e-06
```

```r
# 1. addition of new predictor *shot* improved model accuracy as there is a increase in Multiple R squa
# 2. True, as the p value for shot is very less

conf_interval <- confint(model)["shot", ]
conf_interval
```

```
##    2.5 %   97.5 %
## 128.5552 370.7652
```

```r
# 3. For a 1 meter increase in shot put throw and with the same sprint performance, we can say with 95%
```

```r
#  Using the model built above, extract the slope and intercept for estimating the *score* of *slow* an
coefficients <- coef(model)
intercept_slow = coefficients["(Intercept)"] + coefficients["sprintslow"]
intercept_fast = coefficients["(Intercept)"]

slope_slow = coefficients["shot"] + coefficients["sprintslow"]
slope_fast = coefficients["shot"]
```

```r
# Complete the code below to build a linear model for approximating *score* as a function of *shot* and
# Split the data into 80% train and 20% test parts
set.seed(0)
train_ind = sample(seq_len(nrow(heptathlon)), size = 0.8 * nrow(heptathlon))

hDataTrain = heptathlon[train_ind, ]
hDataTest = heptathlon[-train_ind, ]

# Build linear regression model
model = lm(score ~ shot + sprint, data = hDataTrain)

# Predict on the test data
predicted_scores = predict(model, newdata = hDataTest)

# Print the true and predicted scores for the test data
cat("True Scores:\n", hDataTest$score, "\n\n")
```

```
## True Scores:
##  6858 6297 6137 5686 5289
```

```r
cat("Predicted Scores:\n", predicted_scores, "\n\n")
```

```
## Predicted Scores:
##  6549.446 6279.79 5592.081 5613.656 5389.814
```

```r
# Calculate the model error (mean-squared error for test data)
mse = mean((hDataTest$score - predicted_scores)^2)
cat("Mean-Squared Error on Test Data:", mse, "\n")
```

```
## Mean-Squared Error on Test Data: 81567.14
```

```r
# Fit a linear model for approximating *score* as a function of *shot*, *javelin*, and *sprint*. Print

#1. Did the addition of the new predictor *javelin* improve the model accuracy?
```

```r
#2. *True/false* (explain in one line): the model suggests that there is a possible linear relationship
#3. For a 1 metre increase in shot put throw and with the same javelin and sprint performance, we can s
model =  lm(score ~ shot + javelin + sprint, data = heptathlon)
summary(model)
```

```
##
## Call:
## lm(formula = score ~ shot + javelin + sprint, data = heptathlon)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1090.63  -173.25    12.63   203.29   537.00
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3349.127   1347.536   2.485  0.02144 *
## shot         249.548     59.669   4.182  0.00042 ***
## javelin       -5.996     22.297  -0.269  0.79061
## sprintslow  -354.060    235.151  -1.506  0.14705
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 345.9 on 21 degrees of freedom
## Multiple R-squared:  0.676,  Adjusted R-squared:  0.6298
## F-statistic: 14.61 on 3 and 21 DF,  p-value: 2.301e-05
```

```r
# 1. the addition of the new predictor *javelin* slightly decreased the adjusted r squared value which
# 2. False as p value is very large

conf_interval_shot <- confint(model)["shot", ]
conf_interval_shot
```

```
##    2.5 %   97.5 %
## 125.4599 373.6352
```

```r
# 3. For a 1 metre increase in shot put throw and with the same javelin and sprint performance, we can
```

```r
# Fit a linear model for approximating *score* as a function of *highjump*, and *sprint*. Print the mod
# 1. How accurate is this model?
# 2. Considering a p-value of 10% as cutoff, are there any insignificant features?
model = lm(score ~ highjump + sprint, data = heptathlon)
summary(model)
```

```
##
## Call:
## lm(formula = score ~ highjump + sprint, data = heptathlon)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -476.12 -162.88  -29.12  146.92  502.33
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -2030.8     1175.5  -1.728   0.0981 .
## highjump      4873.2      646.0   7.544 1.54e-07 ***
## sprintslow    -703.3      123.3  -5.702 9.81e-06 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 241.9 on 22 degrees of freedom
## Multiple R-squared:  0.8341, Adjusted R-squared:  0.819
## F-statistic: 55.29 on 2 and 22 DF,  p-value: 2.625e-09
```

```
# 1. the model is pretty accurate as rSe indicates that  the model deviates approximately 241.9 units o

# 2. no there are no insignificant features as both the p values are very much less than 10% or 0.1
```