20BRS1176_lab9

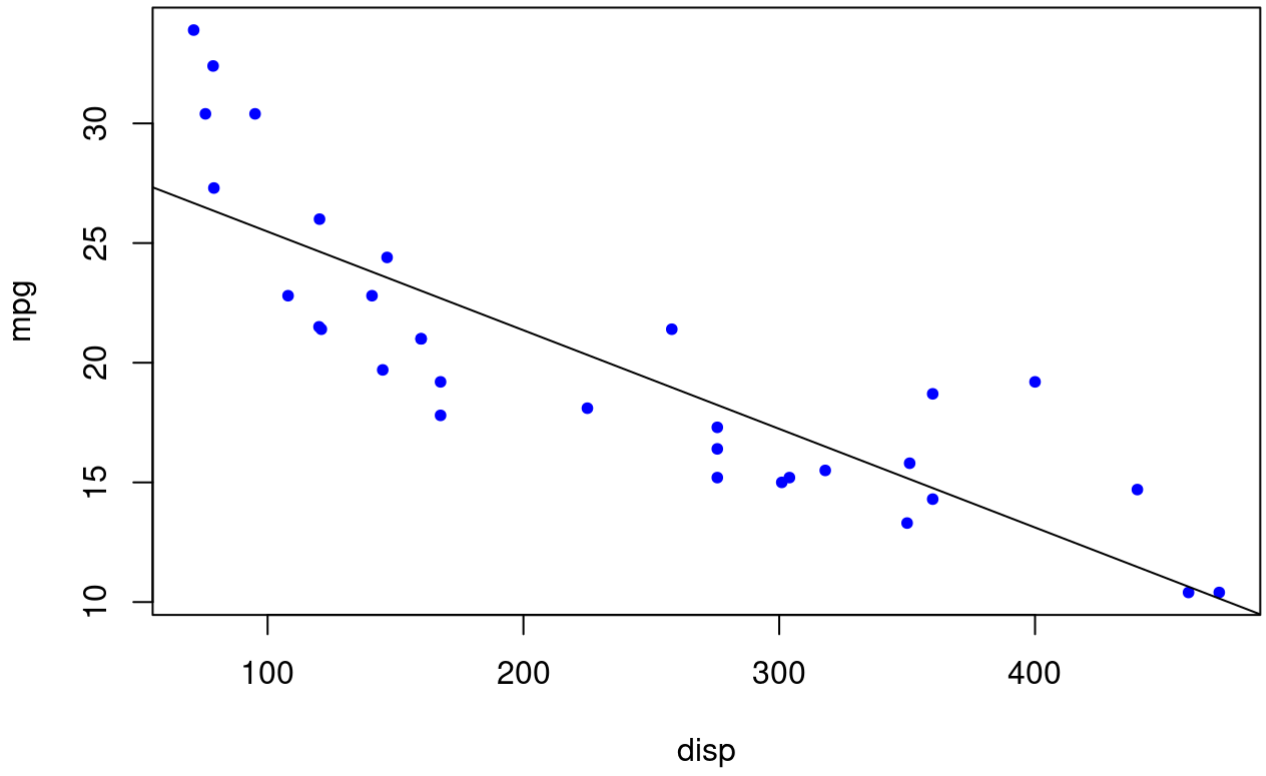
Name: George Mathew Registration No: 20BRS1176 Slot: L55+L56 Lab 9 GRADIENT DESCENT Here the mtcars dataset is used to show the implementation of Gradient Descent Algorithm

```
attach(mtcars)
plot(disp, mpg, col = "blue", pch = 20)

model <- lm(mpg ~ disp, data = mtcars)
coef(model)
```

```
## (Intercept)        disp
## 29.59985476 -0.04121512
```

```
y_preds <- predict(model)
abline(model)
```



As seen above the Intercept and slope should match the above values after gradient descent

```
errors <- unname((mpg - y_preds) ^ 2)
sum(errors) / length(mpg)
```

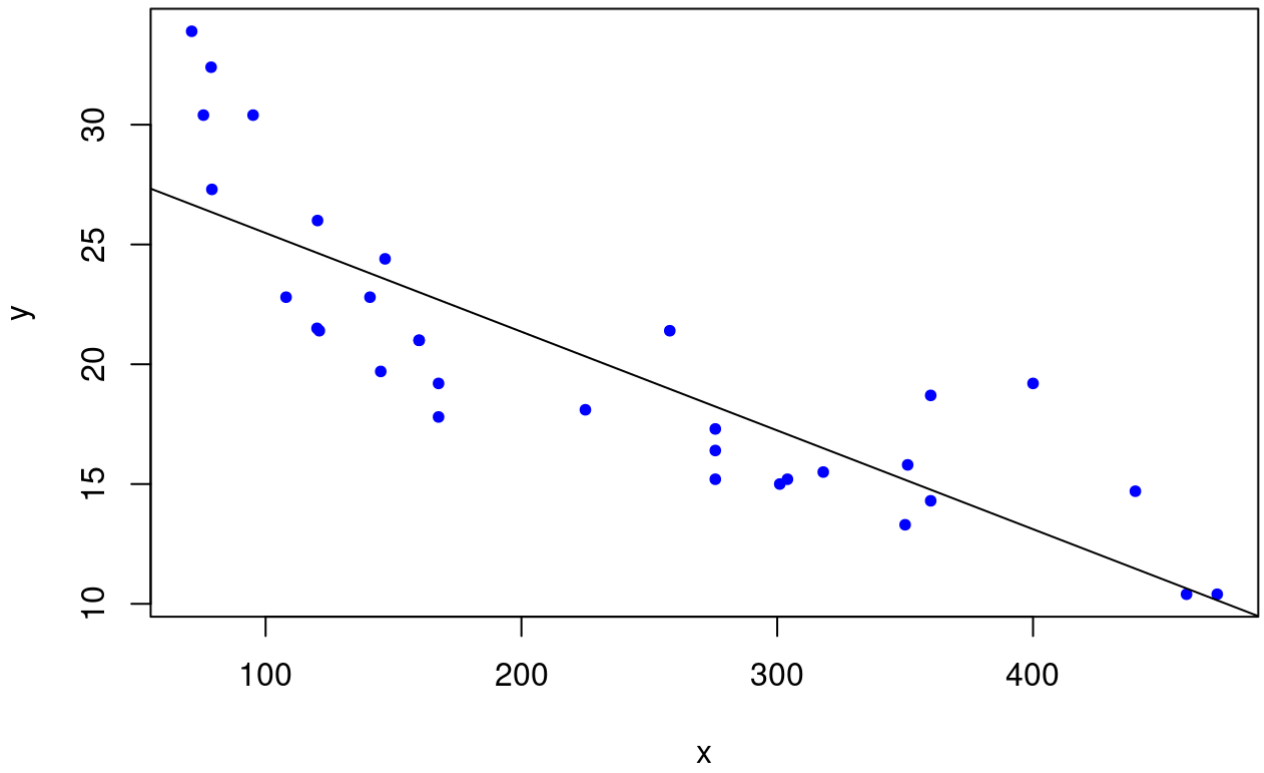
```
## [1] 9.911209
```

Executing the Gradient Descent Algorithm

```
coef(model)
```

```
## (Intercept)        disp
## 29.59985476 -0.04121512
```

```
gradientDesc <- function(x, y, learn_rate, conv_threshold, n, max_iter) {
  plot(x, y, col = "blue", pch = 20)
  m <- runif(1, 0, 1)
  c <- runif(1, 0, 1)
  yhat <- m * x + c
  MSE <- sum((y - yhat) ^ 2) / n
  converged = F
  iterations = 0
  while(converged == F) {
    ## Implement the gradient descent algorithm
    m_new <- m - learn_rate * ((1 / n) * (sum((yhat - y) * x)))
    c_new <- c - learn_rate * ((1 / n) * (sum(yhat - y)))
    m <- m_new
    c <- c_new
    yhat <- m * x + c
    MSE_new <- sum((y - yhat) ^ 2) / n
    if(MSE - MSE_new <= conv_threshold) {
      abline(c, m)
      converged = T
      return(paste("Gradient Descent Optimal intercept:", c, "Gradient Descent Optima
l slope:", m))
    }
    iterations = iterations + 1
    if(iterations > max_iter) {
      abline(c, m)
      converged = T
      return(paste("Gradient Descent Optimal intercept:", c, "Gradient Descent Optima
l slope:", m))
    }
  }
}
# Run the function
gradientDesc(disp, mpg, 0.0000293, 0.001, 32, 2500000)
```




```
## [1] "Gradient Descent Optimal intercept: 29.5998514822541 Gradient Descent Optimal
slope: -0.0412151088729638"
```

As seen above, the values are accurate and match hence Gradient Descent has been successfully implemented