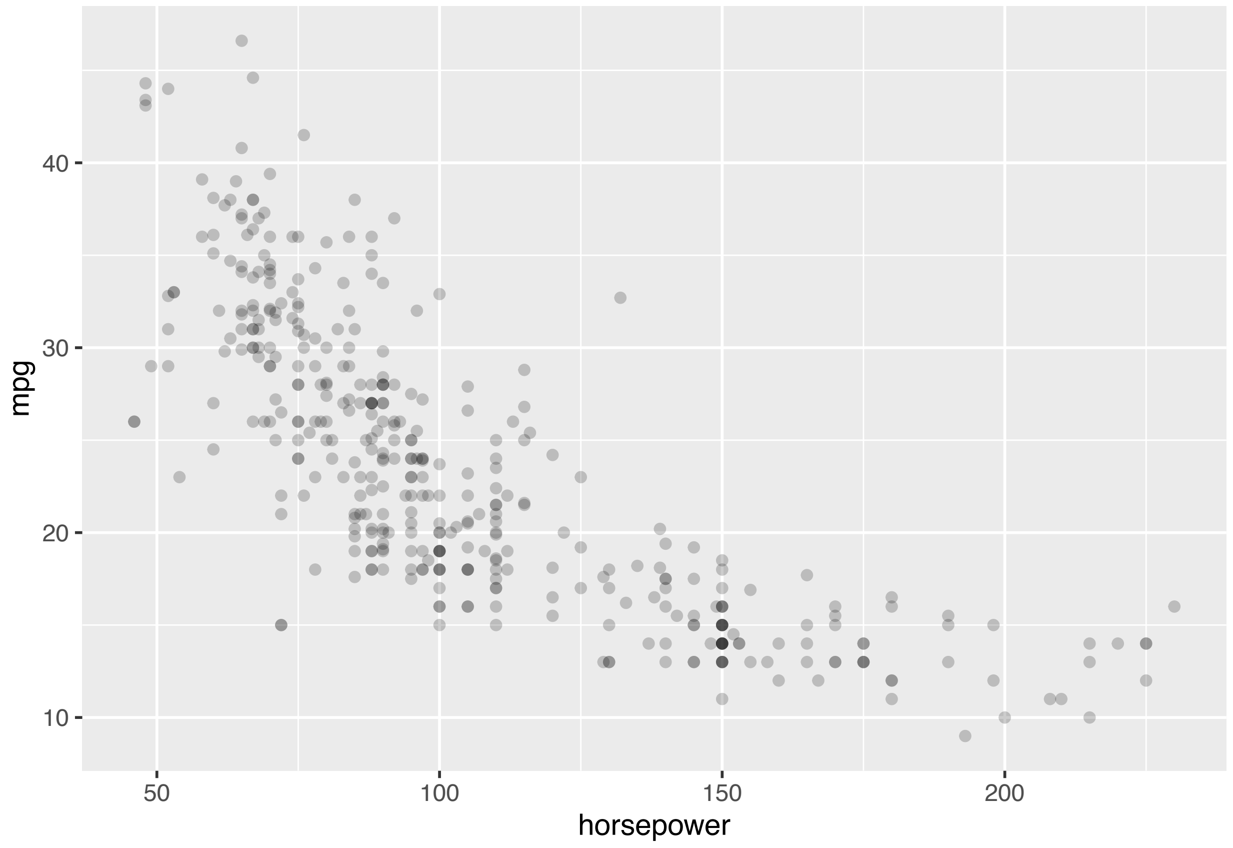
Question 3. Gradient Descent Algorithm

3.1. Get familiar

You will use horsepower as input variable and miles per gallon (mpg) as output:

1. Plot the scatterplot between `mpg` (Y) and `horsepower` (X).
    - Is the relationship positive or negative? Does mpg increase or reduce as horsepower increases?
    - Is the relationship linear?

```
ggplot(Auto, aes(x=horsepower, y=mpg)) +
  geom_point(alpha=0.2)
```

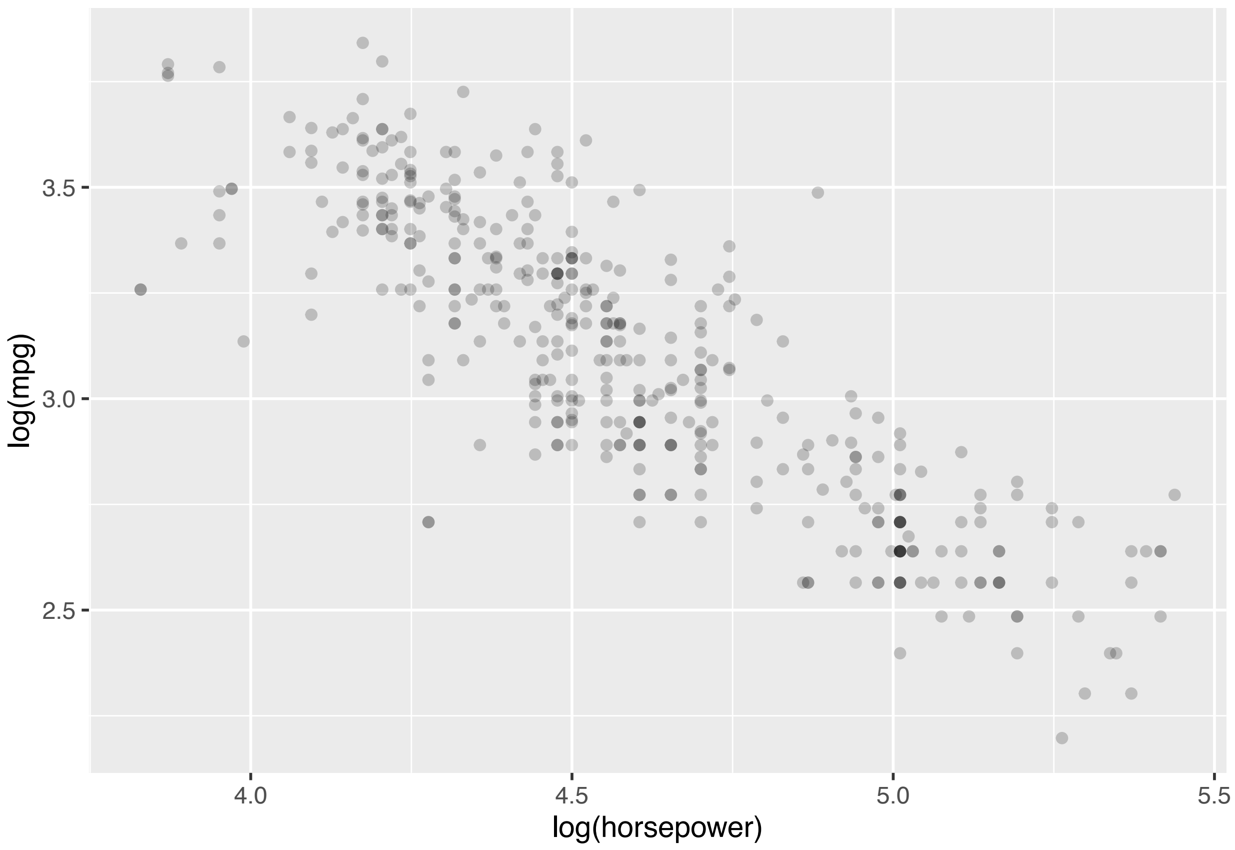


Negative relationship. MPG decreases as Horsepower increases.

The relationship seems to be non-linear, as the scatterplot depicts a curved shaped best-fit line.

2. Plot the scatterplot between `log(mpg)` and `log(horsepower)`.
    - Is the relationship positive or negative?
    - Is the relationship linear?

```
ggplot(Auto, aes(x=log(horsepower), y=log(mpg))) +
geom_point(alpha=0.2)
```



Negative relationship. log(MPG) decreases as log(Horsepower) increases.

The relation seems to linear, as the scatterplot depict a straight best-fit line.

3. Which of the two versions is better for linear regression?

The second version is better for linear regression, as the data is better predicted using a straight downward sloping, best-fit line.

3.2. Fill in the code

The code below estimates the coefficients of linear regression using gradient descent algorithm. If you are given a single linear regression model;

$$Y = \beta_0 + \beta_1 X$$

where $Y = [Y_1, \ldots, Y_N]^T$ and $X = [X_1, \ldots, X_N]^T$ are output and input vectors containing the observations.

The algorithm estimates the parameter vector $\theta = [\beta_0, \beta_1]$ by starting with an arbitrary θ_0 and adjusting it with the gradient of the loss function as:

$$\theta := \theta + \frac{\alpha}{N} X^T (Y - \theta X)$$

where α is the step size (or learning rate) and $(Y - \theta X)^T X$ is the gradient. At each step it calculates the gradient of the loss and adjusts the parameter set accordingly.

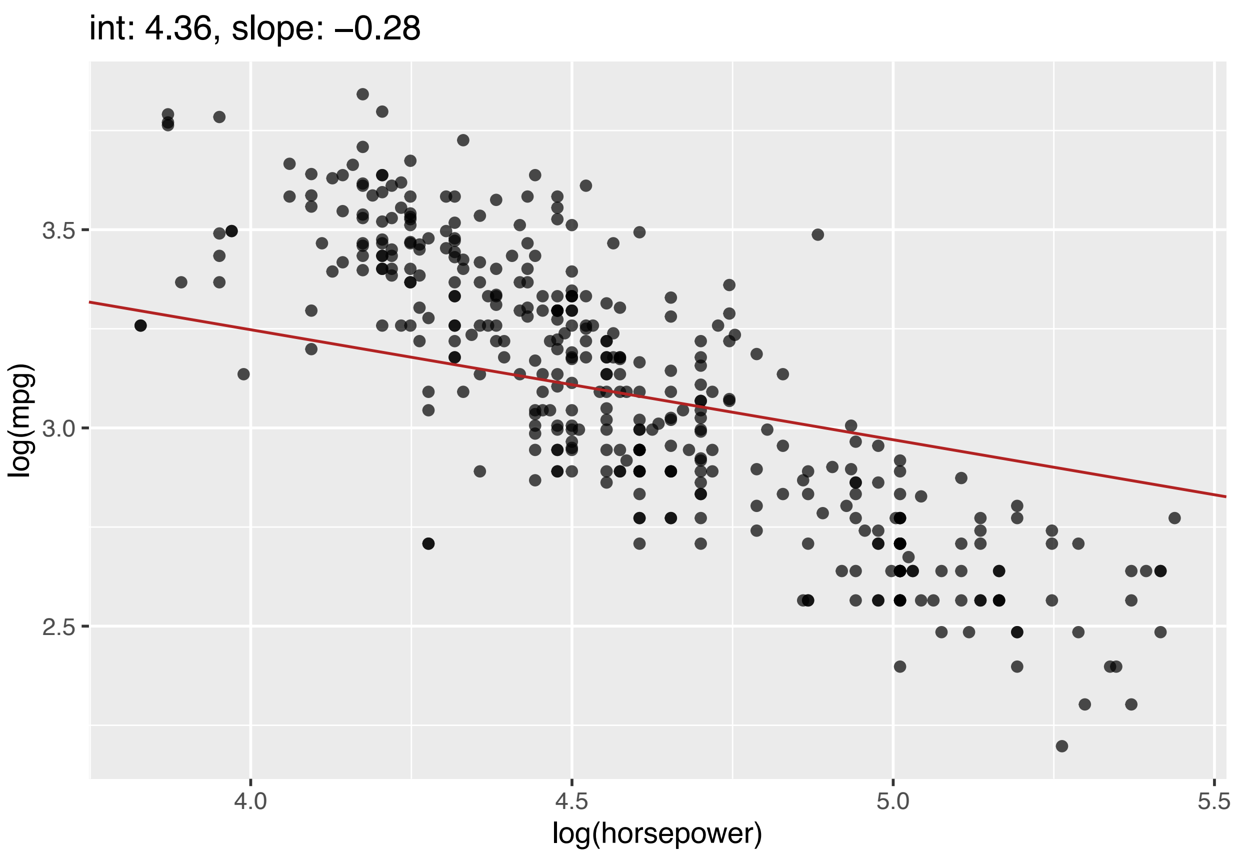
3.3. Run GDA

1. Run the code with the above parameters. How many iterations did it take to estimate the parameters?

```r
GDA <- function(x, y, theta0, alpha = 0.01, epsilon = 1e-8, max_iter=25000){

  # Inputs
  # x      : The input variables (M columns)
  # y      : Output variables    (1 column)
  # theta0 : Initial weight vector (M+1 columns)

  x     <- as.matrix(x)
  y     <- as.matrix(y)
  N     <- nrow(x)
  i     <- 0
  theta <- theta0
  x     <- cbind(1, x) # Adding 1 as first column for intercept
  imprv <- 1e10
  cost  <- (1/(2*N)) * t(x %*% theta - y) %*% (x %*% theta - y)
  delta <- 1
  while(imprv > epsilon & i < max_iter){
    i <- i + 1
    grad <- (t(x) %*% (y-x %*% theta))
    theta <- theta + (alpha / N) * grad
    cost  <- append(cost, (1/(2*N)) * t(x %*% theta - y) %*% (x %*% theta - y))
    imprv <- abs(cost[i+1] - cost[i])
    if((cost[i+1] - cost[i]) > 0) stop("Cost is increasing. Try reducing alpha.")
  }
  if (i==max_iter){print(paste0("maximum interation ", max_iter, " was reached"))} else {
    print(paste0("Finished in ", i, " iterations"))
  }

  return(theta)
}

plot_line <- function(theta) {
  ggplot(Auto, aes(x=log(horsepower),y=log(mpg))) +
    geom_point(alpha=.7) +
    geom_abline(slope = theta[2], intercept = theta[1], colour='firebrick') +
    ggtitle(paste0('int: ', round(theta[1],2), ', slope: ', round(theta[2],2)))
}

x <- log(Auto$horsepower)
y <- log(Auto$mpg)
theta0 <- c(1,1)
theta   <- GDA(x, y, theta0, alpha = 0.05, epsilon = 1e-5)
```

```
## [1] "Finished in 3193 iterations"
```

```
plot_line(theta)
```



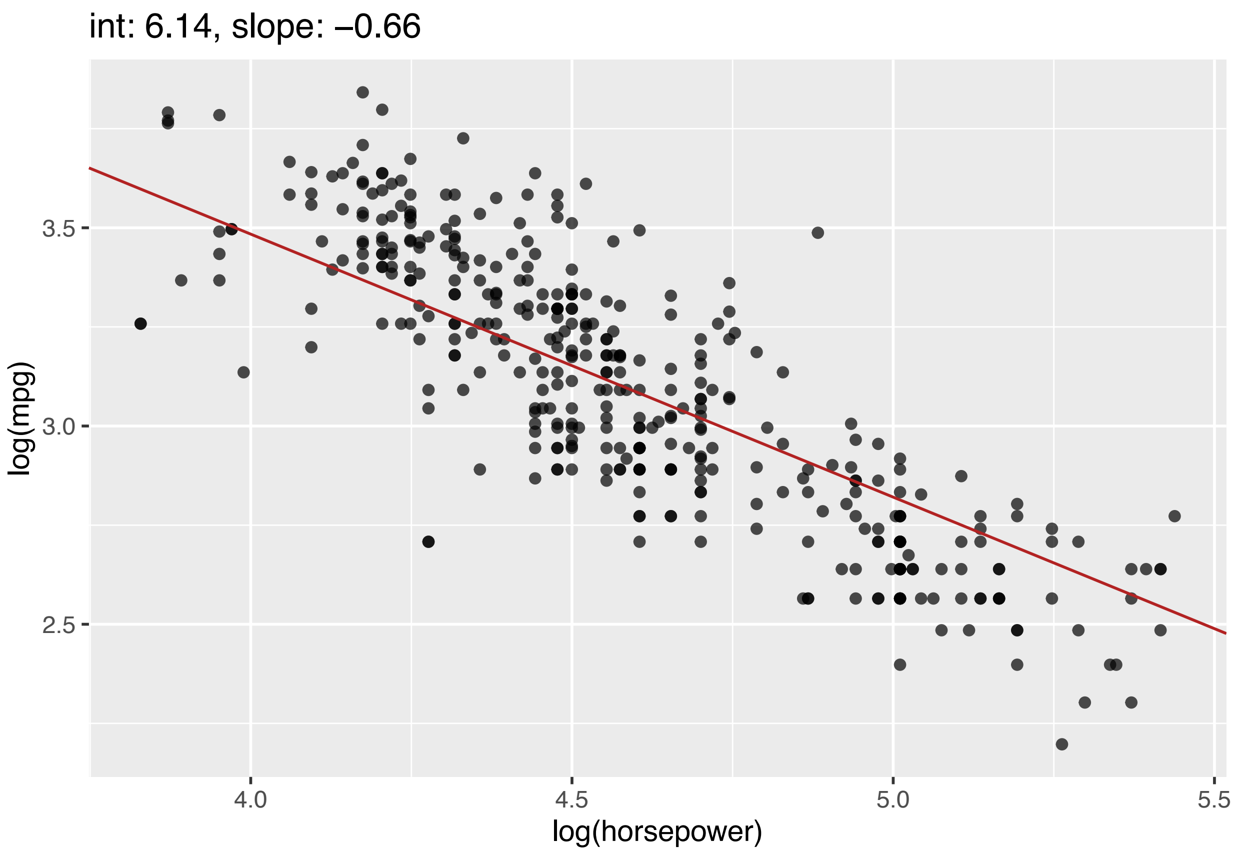
2. Reduce epsilon to `1e-6`, set `alpha=0.05` run the code.

   - How many iterations did it take to estimate the parameters?
   - Does the result improve? Why or why not?

```
x <- log(Auto$horsepower)
y <- log(Auto$mpg)
theta0 <- c(1,1)
theta   <- GDA(x, y, theta0, alpha = 0.05, epsilon = 1e-6)
```

```
## [1] "Finished in 7531 iterations"
```

```
plot_line(theta)
```



Result does not improve. The condition (imprv > epsilon) is going to take longer to terminate, as the epsilon value is now smaller.

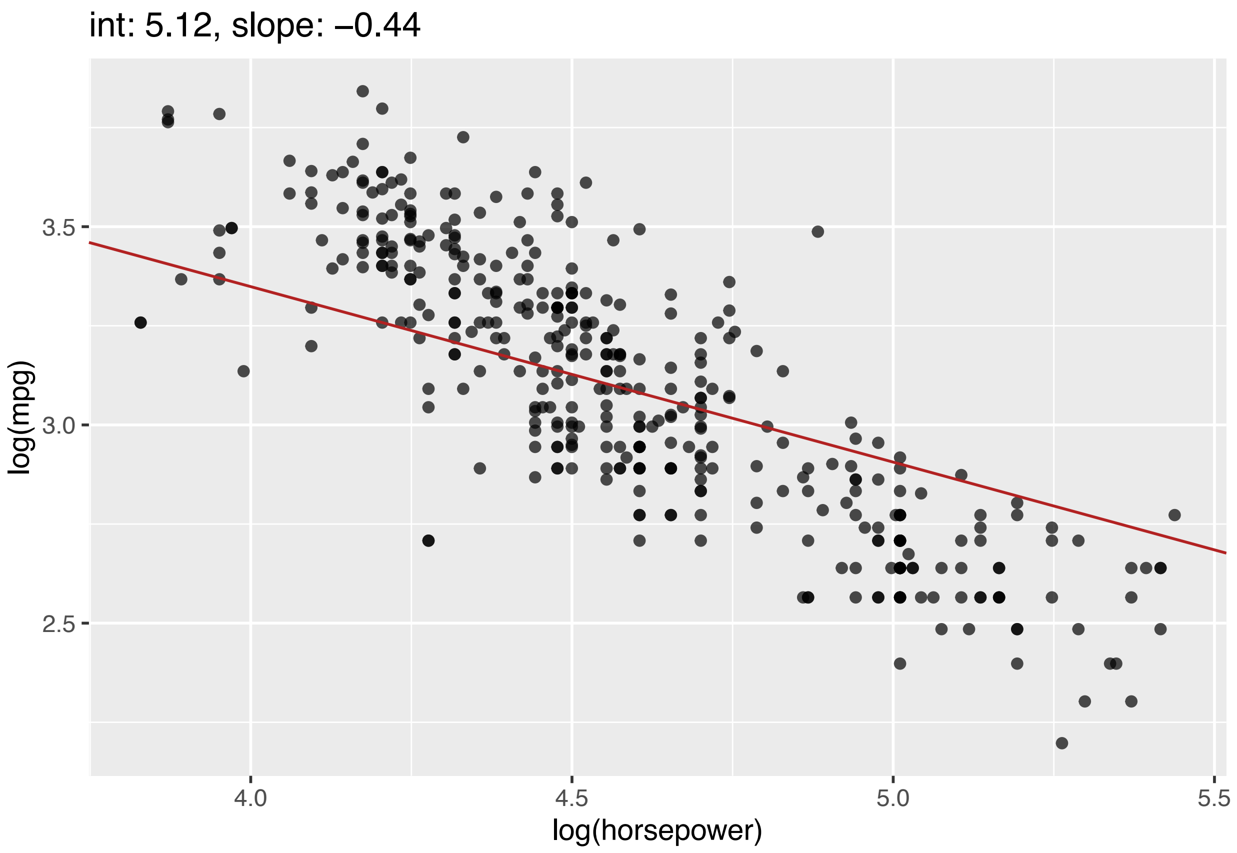
3. Reduce alpha to `alpha=0.01`

   - How many iterations did it take?
   - Did the resulting line change? Why or why not?

```
x <- log(Auto$horsepower)
y <- log(Auto$mpg)
theta0 <- c(1,1)
theta   <- GDA(x, y, theta0, alpha = 0.01, epsilon = 1e-6)
```

```
## [1] "Finished in 22490 iterations"
```

```
plot_line(theta)
```



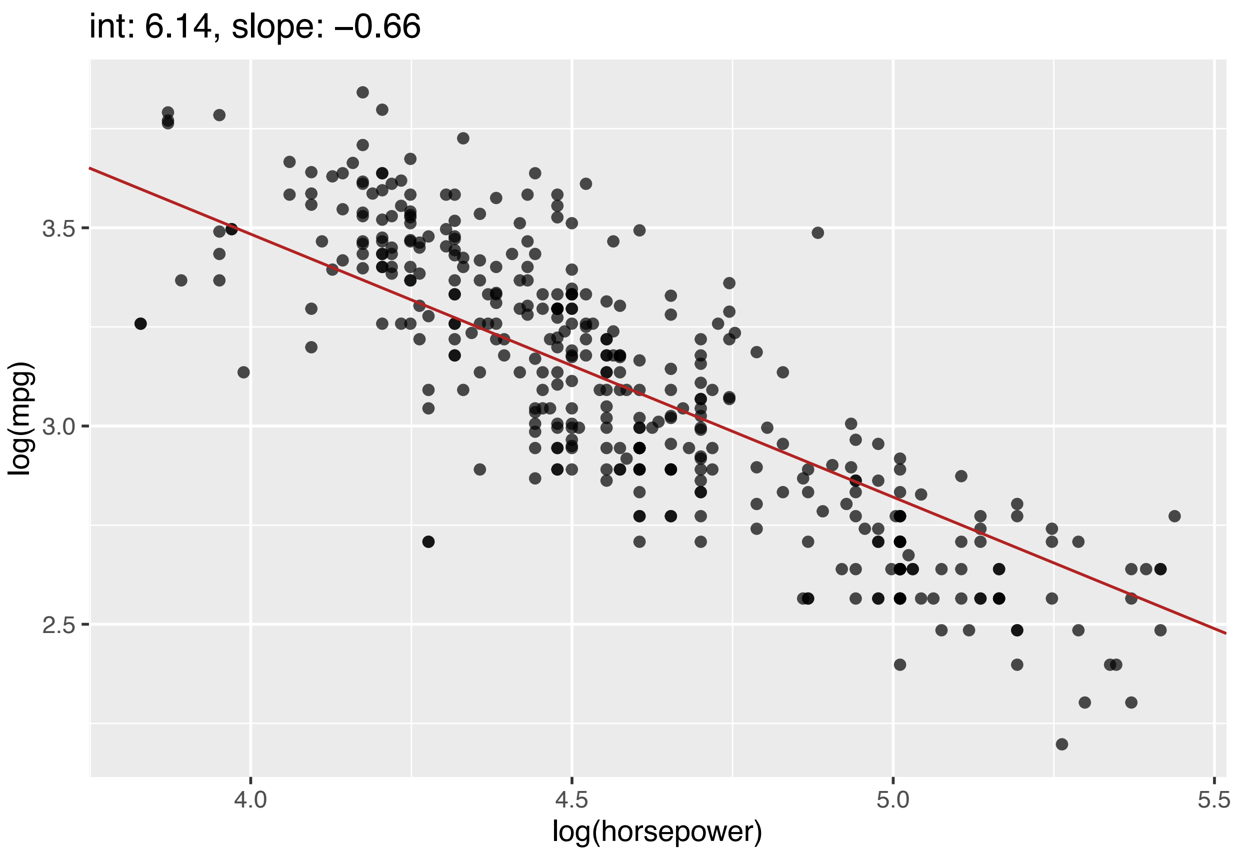
The resulting line did change. The slope increased, hence the best fit line is not a bteer representation of the data points. The increased learning rate, slightly increased our predictions, which seems to be a step in the right dirextion.

4. Set alpha back to `alpha=0.05` and try `theta0=c(1,1)` vs. `theta0=c(1,-1)`:
    - How many iterations did it take? Which is less than the other?
    - Why starting with a negative slope have this effect?

```
x <- log(Auto$horsepower)
y <- log(Auto$mpg)
theta0 <- c(1,1)
theta   <- GDA(x, y, theta0, alpha = 0.05, epsilon = 1e-6)
```

```
## [1] "Finished in 7531 iterations"
```

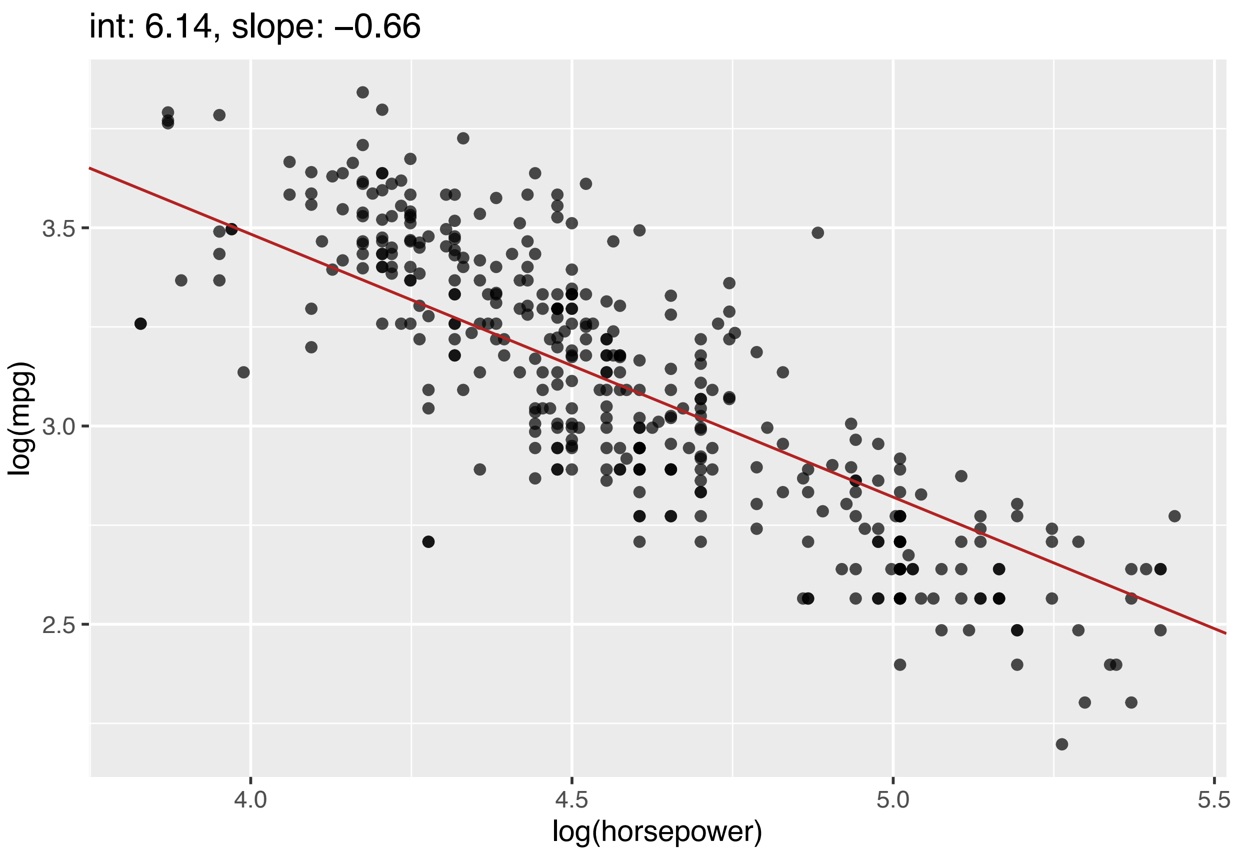
```
plot_line(theta)
```



```r
theta0 <- c(1,-1)
theta   <- GDA(x, y, theta0, alpha = 0.05, epsilon = 1e-6)
```

```
## [1] "Finished in 7265 iterations"
```

```r
plot_line(theta)
```



theta0=c(1,1) took 7531 iterations, while theta0=c(1,-1) took 7265 iterations.

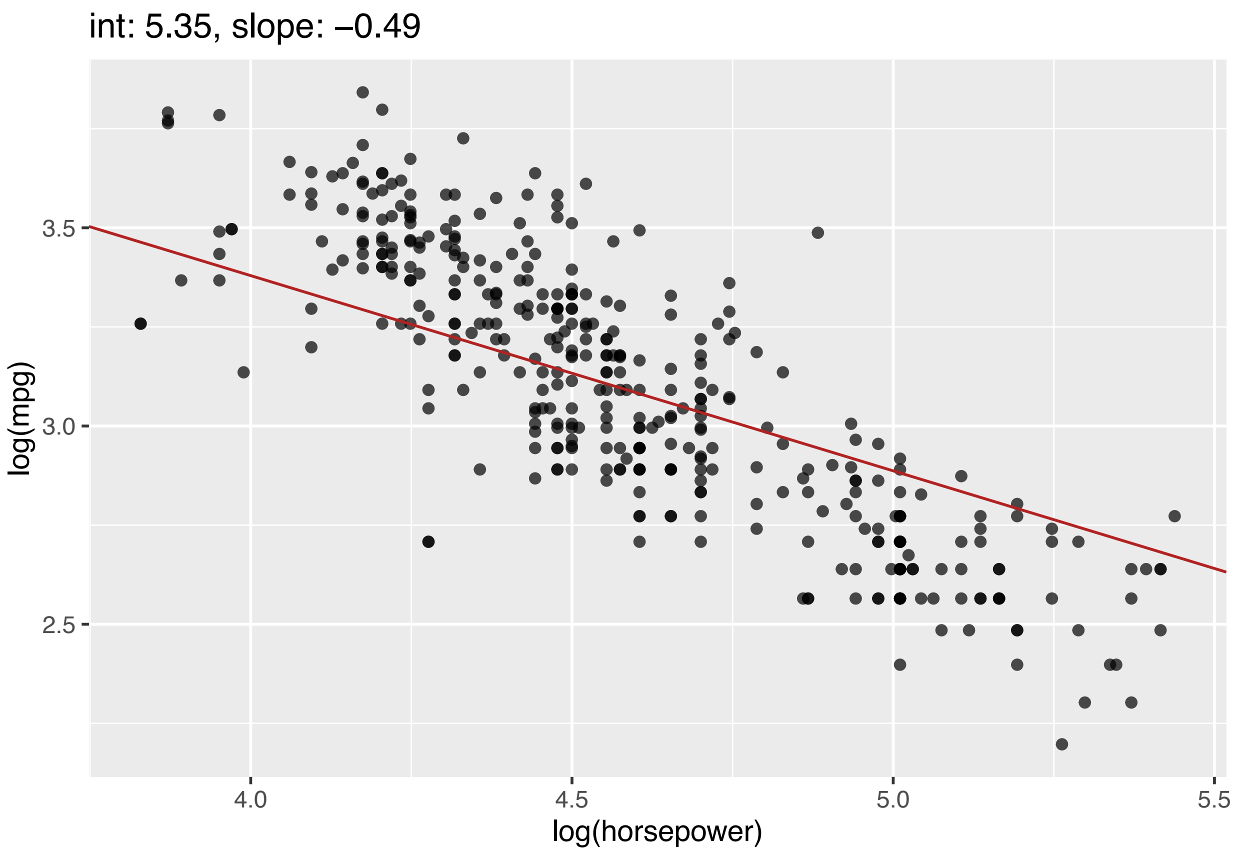
Starting with the negative slope is the accurate decision, since the data is downward sloping. Adding weight to the negative slope is rewarded with lower iterations to complete.

5. Reduce epsilon to `epsilon = 1e-8` and try `alpha=0.01`, `alpha=0.05` and `alpha=0.1`.
    - What effect does alpha have on iterations and resulting fitted line?

```
x <- log(Auto$horsepower)
y <- log(Auto$mpg)
theta0 <- c(1,1)
theta   <- GDA(x, y, theta0, alpha = 0.01, epsilon = 1e-8)
```

```
## [1] "maximum interation 25000 was reached"
```

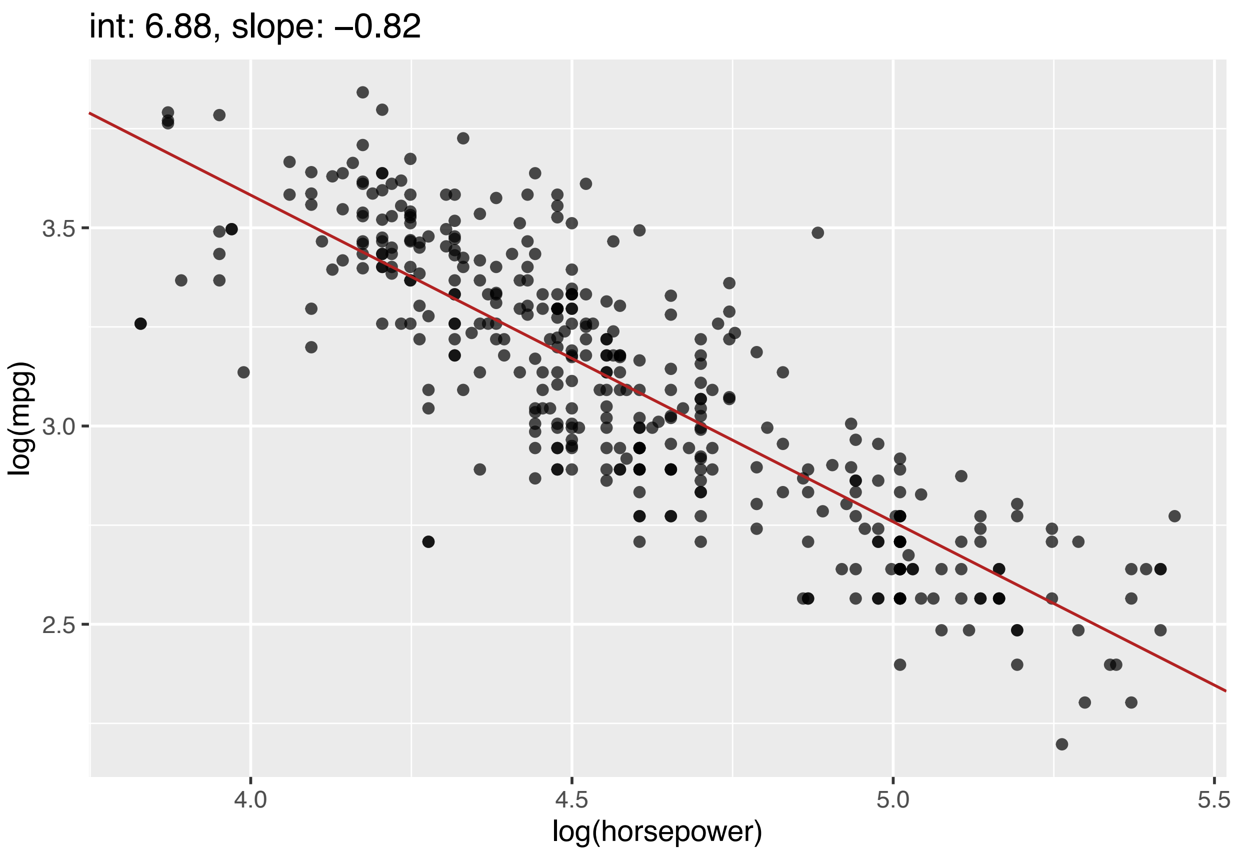
```
plot_line(theta)
```



```r
theta   <- GDA(x, y, theta0, alpha = 0.05, epsilon = 1e-8)
```

```
## [1] "Finished in 16207 iterations"
```

```r
plot_line(theta)
```



```
#Commented out, as it seems to crash the GDA function. (Error in GDA, Cost is increasing. Try redu
#theta   <- GDA(x, y, theta0, alpha = 0.1, epsilon = 1e-8)
#plot_line(theta)
```

Increasing the learning rate improves the predictions and the slope changes to better reflect the datapoints in the scatterplat. This can be clearly observed in the plots drawn for alpha = 0.01 and alpha = 0.05.