Practice Building the Best Linear Prediction Rule

eCornell

10/01/2021

Step 1: Load the data and define colors.

```
library(carData) # The Prestige data set is available in the carData library
data(Prestige)
# Exclude any observations that do not have an entry in the type column:
Prestige = Prestige[!is.na(Prestige$type),]

#eCornell Hex Codes:
crimson = '#b31b1b'   #crimson
lightGray = '#cecece' #lightGray
darkGray = '#606366'  #darkGray
skyBlue = '#92b2c4'   #skyblue
gold = '#fbb040'      #gold
ecBlack = '#393f47'   #ecBlack
```

##Step 2: Use a grid search to find the best fit line. Use R to systematically search over a wide range of possible slopes and intercepts, then calculate the MSE for each combination to find the best fit line.

```
# Create two vectors to form the grid
# a.grid represents the intercept values. This vector goes from -12 to -8 and by 0.01
a.grid = seq(-12, -8, by = 0.01)
# Examine the a.grid vector
a.grid
```

```
##   [1] -12.00 -11.99 -11.98 -11.97 -11.96 -11.95 -11.94 -11.93 -11.92 -11.91
##  [11] -11.90 -11.89 -11.88 -11.87 -11.86 -11.85 -11.84 -11.83 -11.82 -11.81
##  [21] -11.80 -11.79 -11.78 -11.77 -11.76 -11.75 -11.74 -11.73 -11.72 -11.71
##  [31] -11.70 -11.69 -11.68 -11.67 -11.66 -11.65 -11.64 -11.63 -11.62 -11.61
##  [41] -11.60 -11.59 -11.58 -11.57 -11.56 -11.55 -11.54 -11.53 -11.52 -11.51
##  [51] -11.50 -11.49 -11.48 -11.47 -11.46 -11.45 -11.44 -11.43 -11.42 -11.41
##  [61] -11.40 -11.39 -11.38 -11.37 -11.36 -11.35 -11.34 -11.33 -11.32 -11.31
##  [71] -11.30 -11.29 -11.28 -11.27 -11.26 -11.25 -11.24 -11.23 -11.22 -11.21
##  [81] -11.20 -11.19 -11.18 -11.17 -11.16 -11.15 -11.14 -11.13 -11.12 -11.11
##  [91] -11.10 -11.09 -11.08 -11.07 -11.06 -11.05 -11.04 -11.03 -11.02 -11.01
## [101] -11.00 -10.99 -10.98 -10.97 -10.96 -10.95 -10.94 -10.93 -10.92 -10.91
## [111] -10.90 -10.89 -10.88 -10.87 -10.86 -10.85 -10.84 -10.83 -10.82 -10.81
## [121] -10.80 -10.79 -10.78 -10.77 -10.76 -10.75 -10.74 -10.73 -10.72 -10.71
## [131] -10.70 -10.69 -10.68 -10.67 -10.66 -10.65 -10.64 -10.63 -10.62 -10.61
## [141] -10.60 -10.59 -10.58 -10.57 -10.56 -10.55 -10.54 -10.53 -10.52 -10.51
## [151] -10.50 -10.49 -10.48 -10.47 -10.46 -10.45 -10.44 -10.43 -10.42 -10.41
## [161] -10.40 -10.39 -10.38 -10.37 -10.36 -10.35 -10.34 -10.33 -10.32 -10.31
## [171] -10.30 -10.29 -10.28 -10.27 -10.26 -10.25 -10.24 -10.23 -10.22 -10.21
## [181] -10.20 -10.19 -10.18 -10.17 -10.16 -10.15 -10.14 -10.13 -10.12 -10.11
```

```
## [191] -10.10 -10.09 -10.08 -10.07 -10.06 -10.05 -10.04 -10.03 -10.02 -10.01
## [201] -10.00  -9.99  -9.98  -9.97  -9.96  -9.95  -9.94  -9.93  -9.92  -9.91
## [211]  -9.90  -9.89  -9.88  -9.87  -9.86  -9.85  -9.84  -9.83  -9.82  -9.81
## [221]  -9.80  -9.79  -9.78  -9.77  -9.76  -9.75  -9.74  -9.73  -9.72  -9.71
## [231]  -9.70  -9.69  -9.68  -9.67  -9.66  -9.65  -9.64  -9.63  -9.62  -9.61
## [241]  -9.60  -9.59  -9.58  -9.57  -9.56  -9.55  -9.54  -9.53  -9.52  -9.51
## [251]  -9.50  -9.49  -9.48  -9.47  -9.46  -9.45  -9.44  -9.43  -9.42  -9.41
## [261]  -9.40  -9.39  -9.38  -9.37  -9.36  -9.35  -9.34  -9.33  -9.32  -9.31
## [271]  -9.30  -9.29  -9.28  -9.27  -9.26  -9.25  -9.24  -9.23  -9.22  -9.21
## [281]  -9.20  -9.19  -9.18  -9.17  -9.16  -9.15  -9.14  -9.13  -9.12  -9.11
## [291]  -9.10  -9.09  -9.08  -9.07  -9.06  -9.05  -9.04  -9.03  -9.02  -9.01
## [301]  -9.00  -8.99  -8.98  -8.97  -8.96  -8.95  -8.94  -8.93  -8.92  -8.91
## [311]  -8.90  -8.89  -8.88  -8.87  -8.86  -8.85  -8.84  -8.83  -8.82  -8.81
## [321]  -8.80  -8.79  -8.78  -8.77  -8.76  -8.75  -8.74  -8.73  -8.72  -8.71
## [331]  -8.70  -8.69  -8.68  -8.67  -8.66  -8.65  -8.64  -8.63  -8.62  -8.61
## [341]  -8.60  -8.59  -8.58  -8.57  -8.56  -8.55  -8.54  -8.53  -8.52  -8.51
## [351]  -8.50  -8.49  -8.48  -8.47  -8.46  -8.45  -8.44  -8.43  -8.42  -8.41
## [361]  -8.40  -8.39  -8.38  -8.37  -8.36  -8.35  -8.34  -8.33  -8.32  -8.31
## [371]  -8.30  -8.29  -8.28  -8.27  -8.26  -8.25  -8.24  -8.23  -8.22  -8.21
## [381]  -8.20  -8.19  -8.18  -8.17  -8.16  -8.15  -8.14  -8.13  -8.12  -8.11
## [391]  -8.10  -8.09  -8.08  -8.07  -8.06  -8.05  -8.04  -8.03  -8.02  -8.01
## [401]  -8.00
```

```r
# b.grid represents the slope values. This vector goes from 4 to 7 and by 0.01
b.grid = seq(4, 7, by = 0.01)

# We don't know which line will be best, so start with a slope of 0 and an intercept of 0. As we search
a.best = 0 # This is the starting intercept value
b.best = 0 # This is the starting slope value
MSE.best = mean(Prestige$prestige^2)
```

##Step 3: Use a nested for loop to search for the best intercept and slope. Use a nested for loop (one for loop nested inside another for loop) to search through the entire grid. This for loop will keep the intercept (a) the same and search through each slope (b) value in the b.grid vector, and then move to the next a value and search through the b.grid vector, etc.

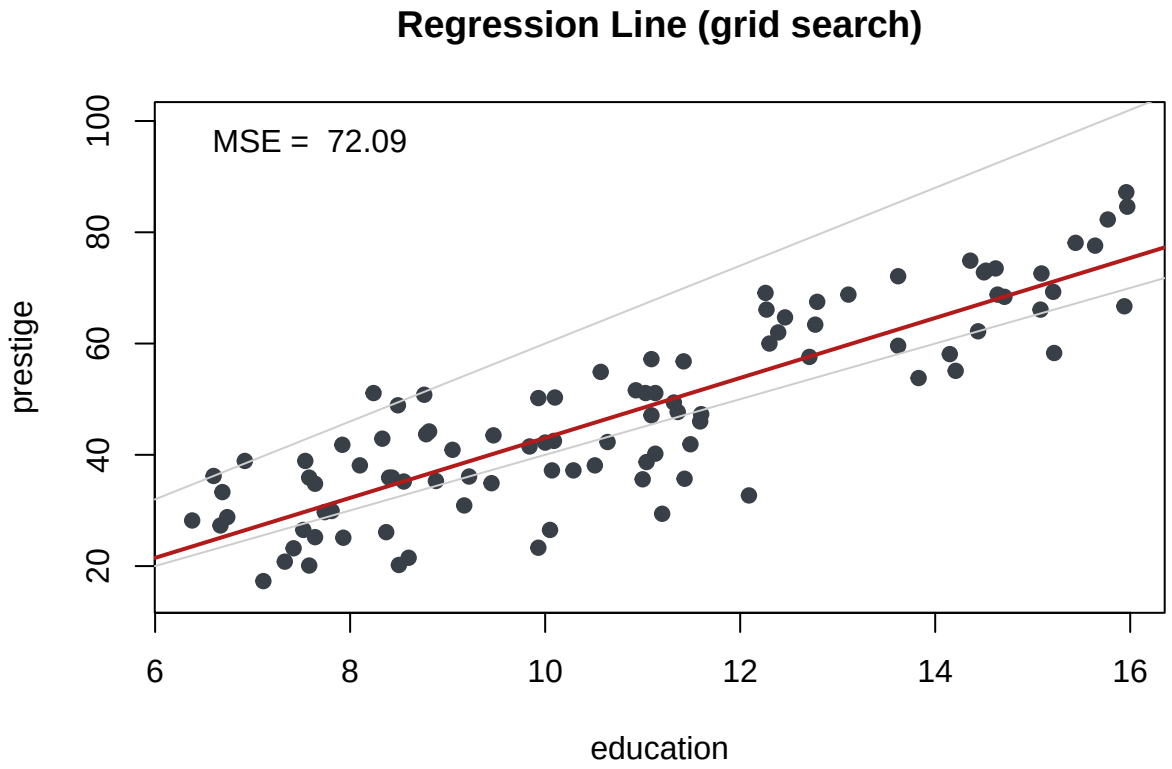
```r
for (a in a.grid){
  for (b in b.grid){
    # Find the predicted prestige value at a, b:
    prestige.fit = a + b * Prestige$education
    # Find the error of the predicted value by comparing it with the observed value:
    prestige.error = Prestige$prestige - prestige.fit
    # Calculate MSE at a,b:
    MSE = mean(prestige.error^2)
    # Determine whether this MSE value is better than the previous best value. If it is, update the bes
    if (MSE < MSE.best){
      a.best = a
      b.best = b
      MSE.best = MSE
    }
  }
}

# Print the best values
print(c(a.best, b.best, MSE.best))
```

```
## [1] -10.86000   5.39000  72.08578
```

##Step 4: Plot the best fit line with the data. Plot the best fit line you found in the previous step, and compare it to the other lines you examined.

```
plot(prestige ~ education, data = Prestige,
     pch = 19, col = ecBlack,
     main = 'Regression Line (grid search)',
     ylim = c(15, 100))
abline(a = a.best, b = b.best,
       col = crimson, lwd = 2)
legend('topleft', legend =
       paste('MSE = ', round(MSE.best, 2)),
       bty = 'n')
abline(a = -10, b = 5,
       col = lightGray, lwd = 1)
abline(a = -10, b = 7,
       col = lightGray, lwd = 1)
```



##Step 5: Use and interpret a regression analysis. Use R function lm() to find the regression line. The key outputs from this function are: regression coefficients: intercept and slope residuals: prediction errors of the fitted line

```
fit.prestige <- lm(prestige ~ education, data = Prestige)
# An object to store all the outputs of your regression (slope, intercept, predicted values, errors etc
fit.prestige
```

```
##
## Call:
## lm(formula = prestige ~ education, data = Prestige)
##
## Coefficients:
```

```
## (Intercept)    education
##     -10.841        5.388
```

```
fit.prestige$coefficients[1]   # Intercept
```

```
## (Intercept)
##   -10.84086
```

```
fit.prestige$coefficients[2]   # Slope
```

```
## education
##  5.388408
```

```
mean(fit.prestige$residuals^2) # MSE
```

```
## [1] 72.08576
```

In regression analysis, we are primarily interested in slope b. "b = 5.39": for every additional year of education, average prestige score of workers in a job increases by 5.39 points.

##Step 6: Visualize the best fit line. Look at how the regression line fits the data.

```
# Create a plot:
plot(prestige ~ education, data = Prestige, pch = 19, col = ecBlack,
     main = 'Regression line (using lm())', ylim = c(15, 100))
abline(fit.prestige, col = crimson, lwd = 3) # the regression object is an argument in abline() so you
# Add a legend:
legend('topleft', legend = paste('MSE = ',
       round(mean(fit.prestige$residuals^2), 2)), bty = 'n',
       lwd = 3, col = crimson)
```