Linear Regression with Tensorflow in R - Comparison with *lm* model

Dario H. Romero, MSc CS - Oil & Gas Data Scientist

A proof of concept (PoC) analysis utilizing tensorflow in RStudio for a linear regression modeling, including a comparison with *lm* basic function for linear regression in R.

right click for the Github project

```r
# function to add some noise to data
noise <- function(x, decibel = 1) {
  n <- length(x)
  set.seed(123)
  noise <- runif(n, min(x)*decibel, max(x)*decibel)
}
# Create 100 x, y data points, y = x * 0.14 + 0.65
x_data <- runif(100, min=0, max=1)
y_data <- x_data * 0.14 + 0.65

# Adding some noise to data
y_data <- y_data + noise(y_data, 5*sd(y_data))
```

```r
# Try to find values for W and b that compute y_data = W * x_data + b
# (We know that W should be 0.14 and b 0.65, but TensorFlow will
# figure that out for us.)
W <- tf$Variable(tf$random_uniform(shape(1L), -1.0, 1.0), name = 'W')
b <- tf$Variable(tf$zeros(shape(1L)), name = 'b')
y <- W * x_data + b

# Minimize the mean squared errors.
loss <- tf$reduce_mean((y - y_data) ^ 2)
optimizer <- tf$train$GradientDescentOptimizer(0.5)
train <- optimizer$minimize(loss)
```

```r
# Launch the graph and initialize the variables.
sess = tf$Session()
sess$run(tf$global_variables_initializer())
```

Fit the line and print tf model parameters W: Weights and b: bias

```r
# Fit the line (Learns best fit is W: 0.14, b: 0.65)
for (step in 1:206) {
  sess$run(train)
  if (step %% 20 == 0)
    cat(step, " - W:", sess$run(W),  " - b:", sess$run(b), "\n")
  Wfinal <- sess$run(W)
  bfinal <- sess$run(b)
}
```

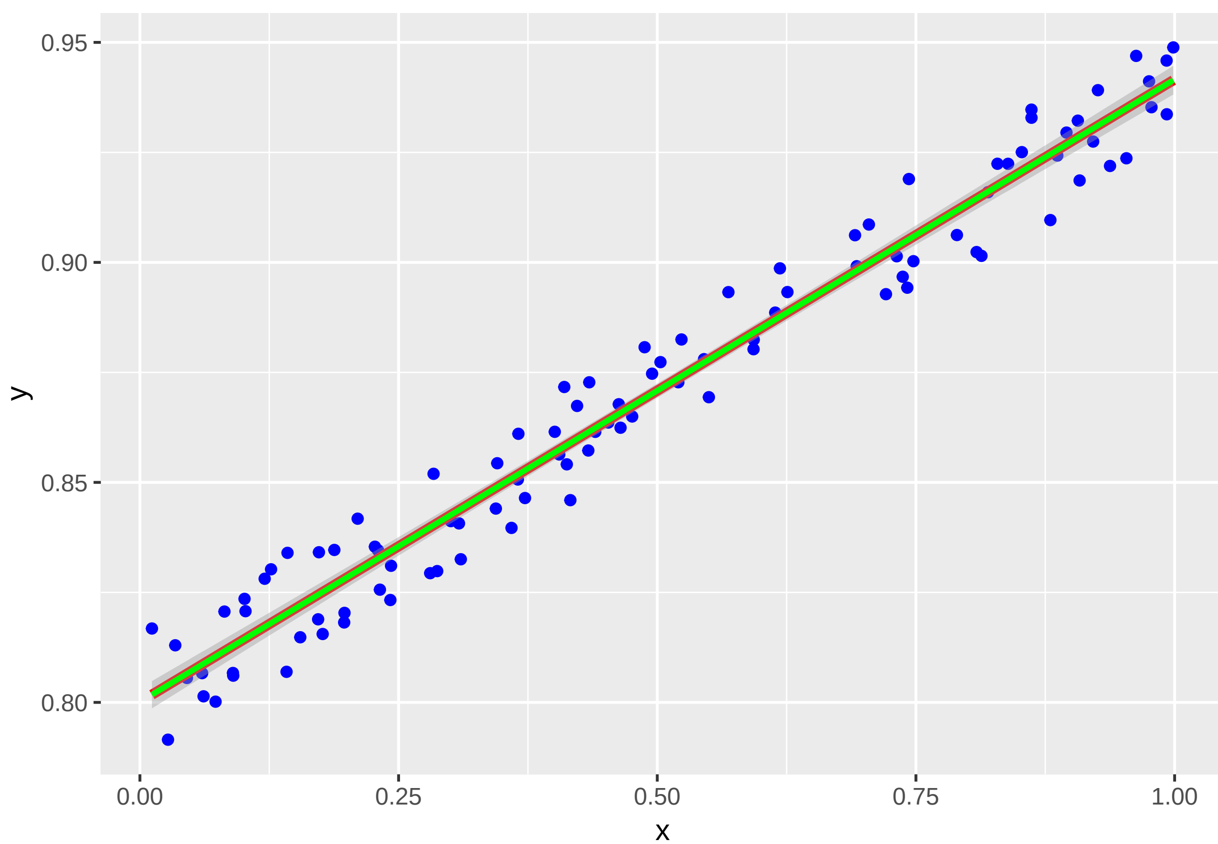
```
## 20  - W: 0.2230116  - b: 0.7572598
```

```
## 40  - W: 0.1603969  - b: 0.7901621
## 60  - W: 0.1458706  - b: 0.7977954
## 80  - W: 0.1425005  - b: 0.7995662
## 100  - W: 0.1417187  - b: 0.799977
## 120  - W: 0.1415374  - b: 0.8000723
## 140  - W: 0.1414953  - b: 0.8000944
## 160  - W: 0.1414855  - b: 0.8000996
## 180  - W: 0.1414833  - b: 0.8001007
## 200  - W: 0.1414828  - b: 0.800101
```

```
# Print Tensorflow Model parameters W: weights and b: bias
cat("Weights: ", round(Wfinal, 7), " - bias: ", round(bfinal, 7))
```

```
## Weights:  0.1414828  - bias:  0.800101
```

```
data <- tibble(x = x_data, y = y_data)
datam <- tibble(x = x_data, y = x_data * Wfinal + bfinal)
g <- ggplot(data = data, aes(x, y)) + geom_point(colour = 'blue') +
  geom_line(data = datam, colour = "red", size = 1.8) +
  geom_smooth(method = "lm", colour = "green")
g
```



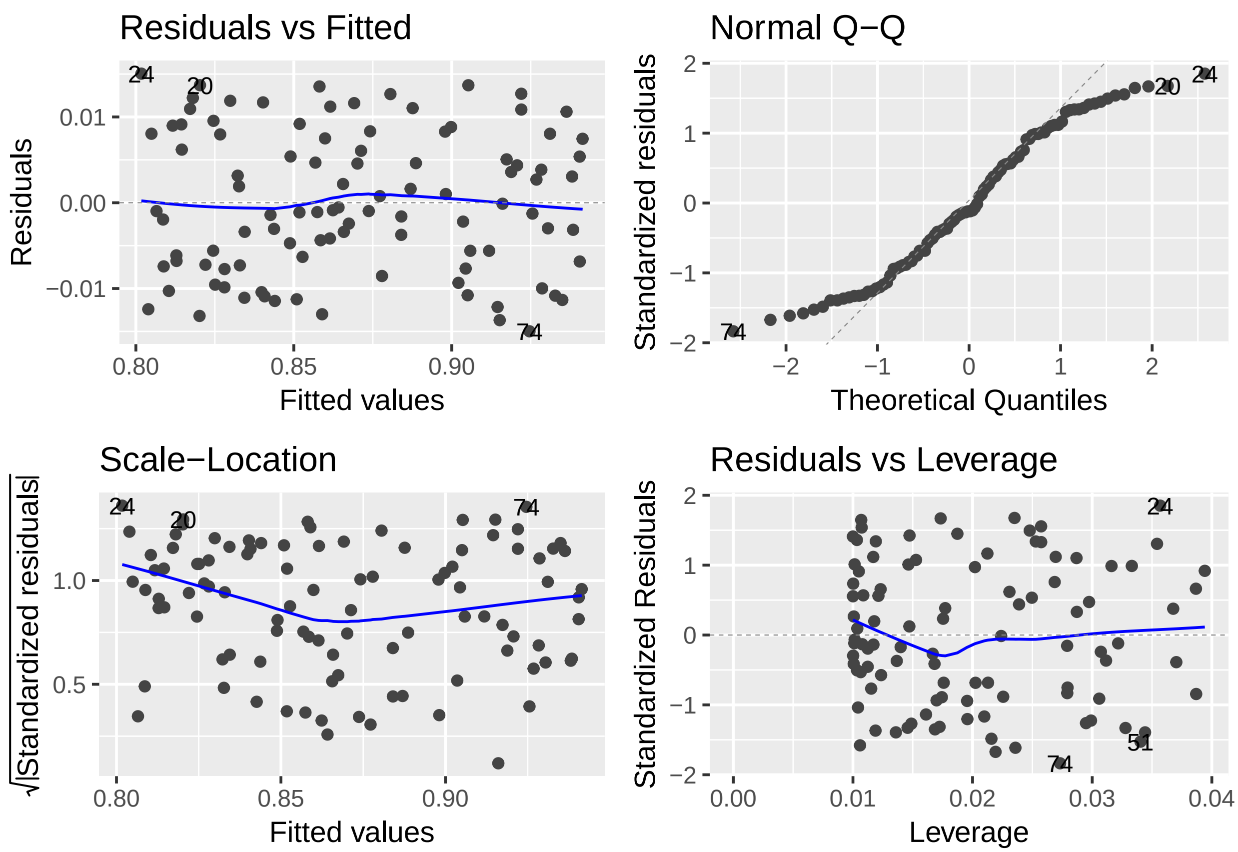
Regression diagnostics plots An important part of creating regression models is evaluating how well they fit the data. We can use the package **ggfortify** to let *ggplot2* interpret *lm* objects and create diagnostic plots.

```r
linearModel <- lm(formula = y ~ x, data = data)

# Print Linear Model $lm$ parameters $W: weights$ and $b: bias$
cat("Weights: ", round(linearModel$coefficients[2], 7),
    " - bias: ", round(linearModel$coefficients[1], 7))
```

```
## Weights:  0.1414826  - bias:  0.8001011
```

```r
autoplot(linearModel, label.size = 3)
```



Comments:

- We observe on the Residuals vs Fitted plot a random pattern in the distribution of residuals, suggesting that a linear regression is valid to represent this relationship with the dependent variable.
- Linear regression therefore works the best in this case where a linear relationship between the dependent and non-dependent variables exists.