DSO 530: Simple Linear Regression

Abbass Al Sharif

Predicting House Value from Percent of Low Income Household

We are going to use a dataset called Boston which is part of the `MASS` package. It recordes the median value of houses for 506 neighborhoods around Boston. Our task is to predict the median house value (`medv`) using only one predictor (`lstat`: percent of households with low socioeconomic status).

```
# load MASS package
library(MASS)

#split the data by using the first 400 observations as the training data and the remaining as the testi
train = 1:400
test = -train

# speficy that we are going to use only two variables (lstat and medv)
variables = which(names(Boston) ==c("lstat", "medv"))
training_data = Boston[train, variables]
testing_data = Boston[test, variables]

# check the dimensions of the new data sets
dim(training_data)
```

STEP 1: Split the dataset

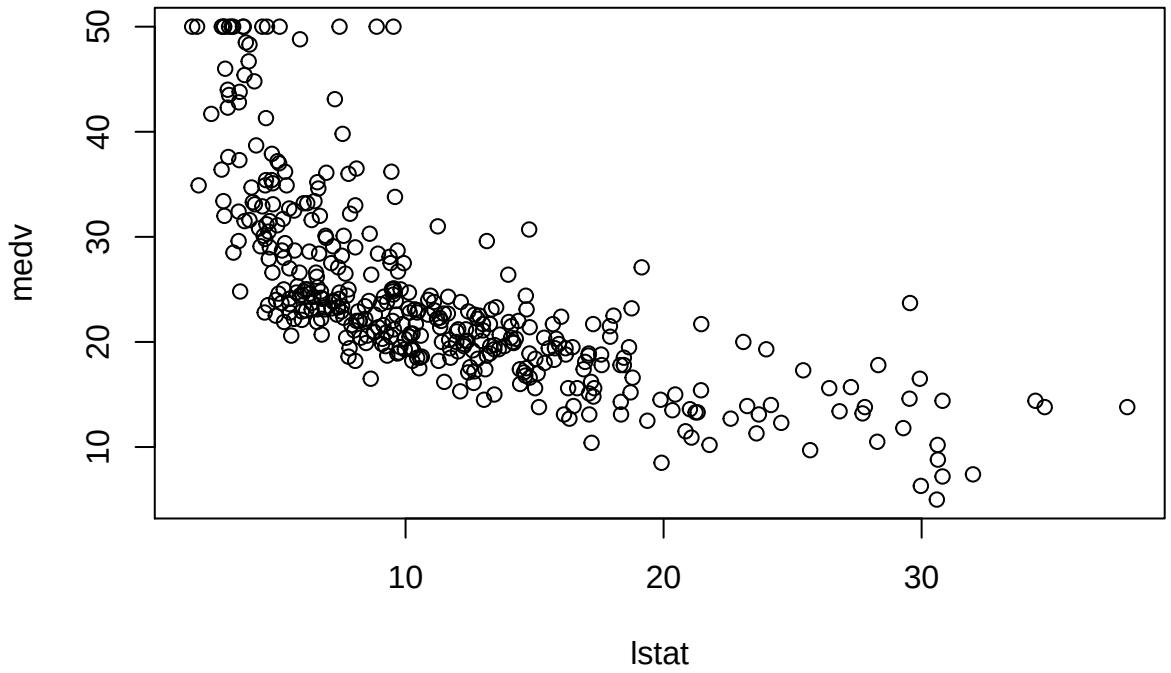
```
## [1] 400   2
```

```
dim(testing_data)
```

```
## [1] 106   2
```

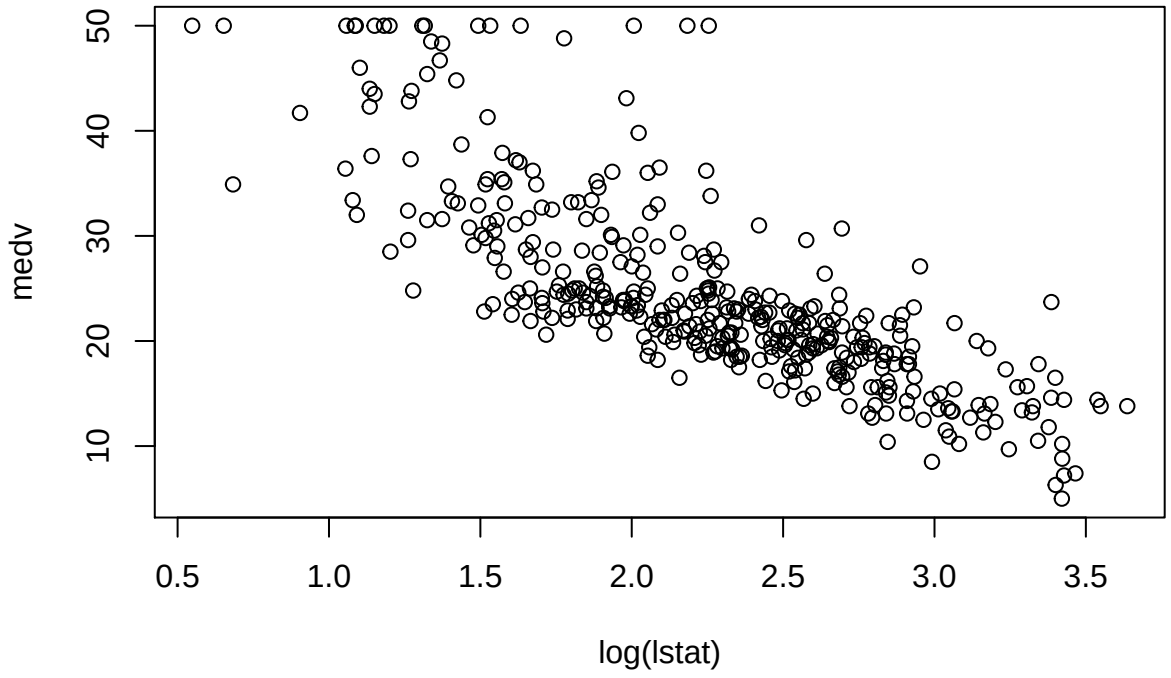
STEP 2: Check for Linearity In order to perfom linear regression in R, we will use a function called `lm()` which stands for linear models. We will use it to fit a simple linear regression with `medv` as the response (dependent variable) and `lstat` as the predictor or independent variable, and then save it in a place called `model`. Notice that the formula for `lm` in R is in this format `y~x`, where `y` is the dependent variable, and `x` is the indepenent variable. But before we run our model, let's visually check if the relationship between `x` and `y` is linear.

```
#attach training data
attach(training_data)
#scatterplot of lstat vs. medv

plot(lstat, medv)
```



According to the plot, we see that the relationship is a not linear. Let's transorm our explanatory variable `lstat`.

```
#scatterplot of log(lstat) vs. medv

plot(log(lstat), medv)
```



STEP 3: Run the linear regression model Look at the plot, it is more linear, so we can proceed and perform `lm()`:

```
model = lm(medv ~ log(lstat), data = training_data)
model
```

```
##
## Call:
## lm(formula = medv ~ log(lstat), data = training_data)
##
## Coefficients:
## (Intercept)   log(lstat)
##        51.8        -12.2
```

Notice that some basic information is shown to us when we print "model". This only gave us the slope (-12.2) and the intercept (51.8) of the linear model. Note that here we are looking at `log(lstat)` and not `lstat` anymore. So for every one unit increase in 'lstat', the median value of the house will decrease by eˆ(12.2). For more detailed information, we can use the "summary()" function:

```
summary(model)
```

```
##
## Call:
## lm(formula = medv ~ log(lstat), data = training_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -11.385  -3.908  -0.779   2.245  25.728
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   51.783      1.097    47.2   <2e-16 ***
## log(lstat)   -12.203      0.472   -25.9   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.6 on 398 degrees of freedom
## Multiple R-squared:  0.627,  Adjusted R-squared:  0.626
## F-statistic:  669 on 1 and 398 DF,  p-value: <2e-16
```

Now, we have access to p-values and standard errors for the coefficients, as well as the Rˆ2. The output states that the slope is statistically significant and different from 0 and with a t-value= -25.9 (p-value <0.05), which means that there is a significant relationship between the percentage of households with low socioeconomic income and the median house value. This relationship is negative. That is as the percantage of household with low socioeconomic income increases, the median house value decreases.

Looking at Rˆ2, we can deduce that 62.7% of the model variation is being explained by the predictor `log(lstat)`. This is probably low, but indeed it would increase if we had more independent (explanatory) variables.

We can use the "names()" function to see what other peices of information are stored in our linear model (model).

```
names(model)
```

```
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
```

```
model$coefficients
```

```
## (Intercept)  log(lstat)
##       51.78      -12.20
```

To obtain the confidence intervel for the linear model (model), we can use the "confint()" function:

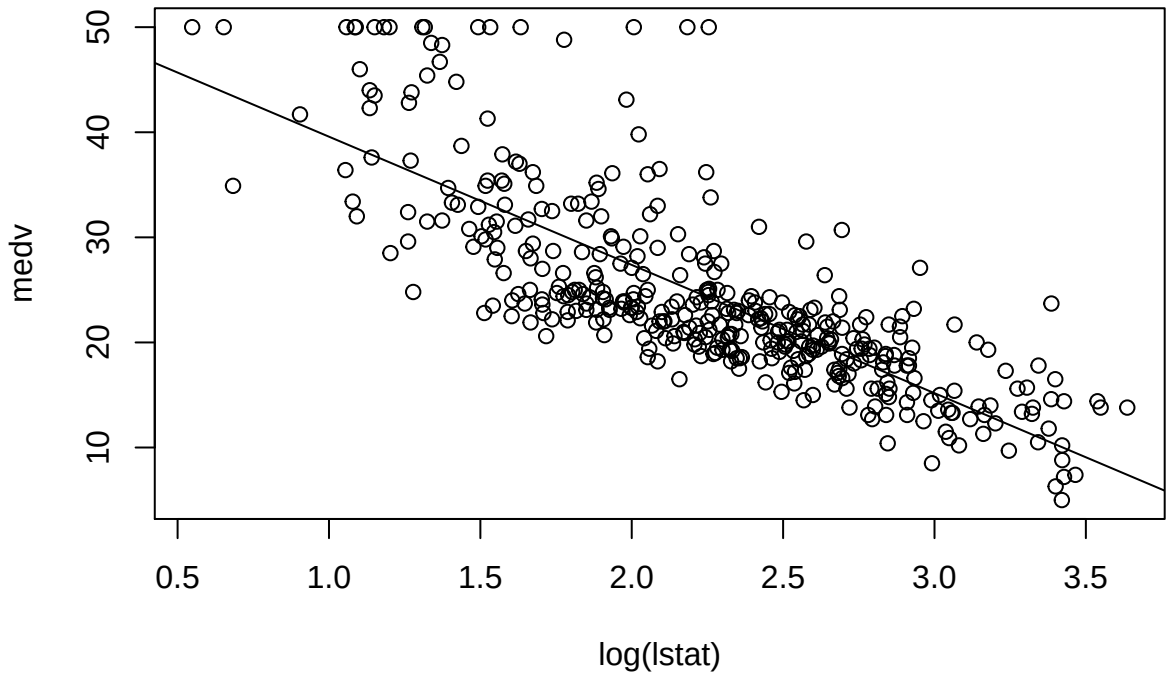
```
confint(model, level = 0.95)
```

```
##              2.5 % 97.5 %
## (Intercept)  49.63  53.94
## log(lstat)  -13.13 -11.28
```

So, a 95% confidence interval for the slope of `log(lstat)` is (-13.13, -11.28). Notice that this confidence interval gives us the same result as the hypothesis test performed earlier, by stating that we are 95% confident that the slope of lstat is not zero (in fact it is less than zero, which means that the relationship is negative.)

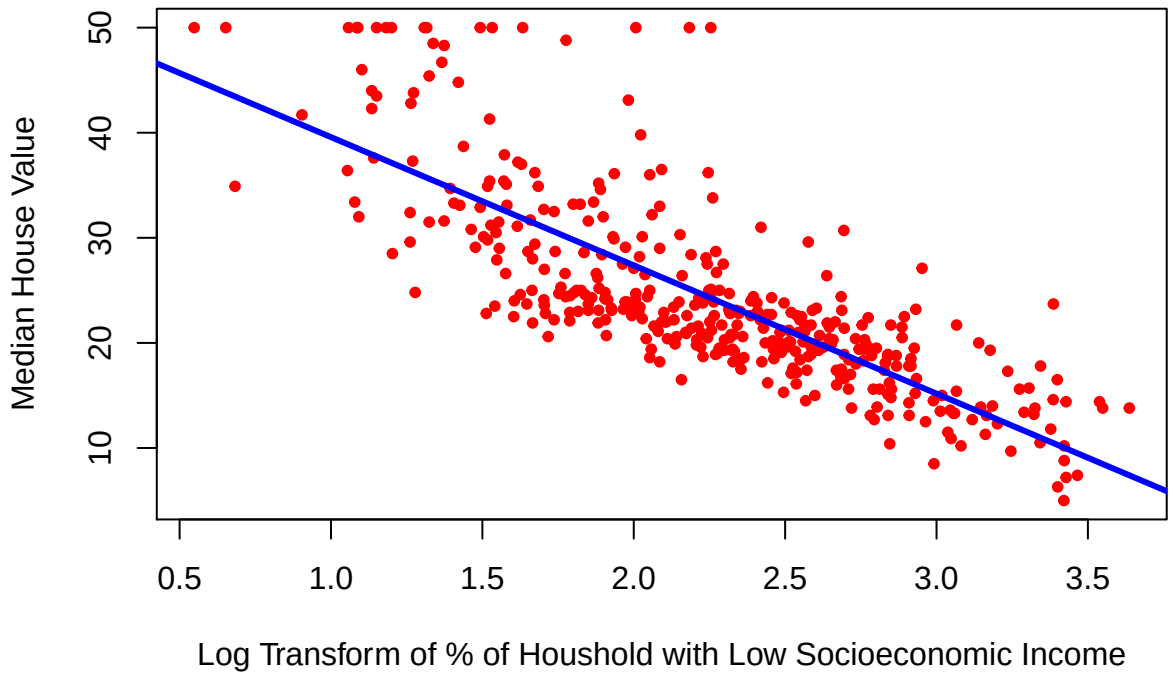
STEP 4: Plot the regression model Now, let's plot our regression line on top of our data.

```
#scatterplot of lstat vs. medv

plot(log(lstat), medv)

# Add the regression line to the existing scatterplot
abline(model)
```



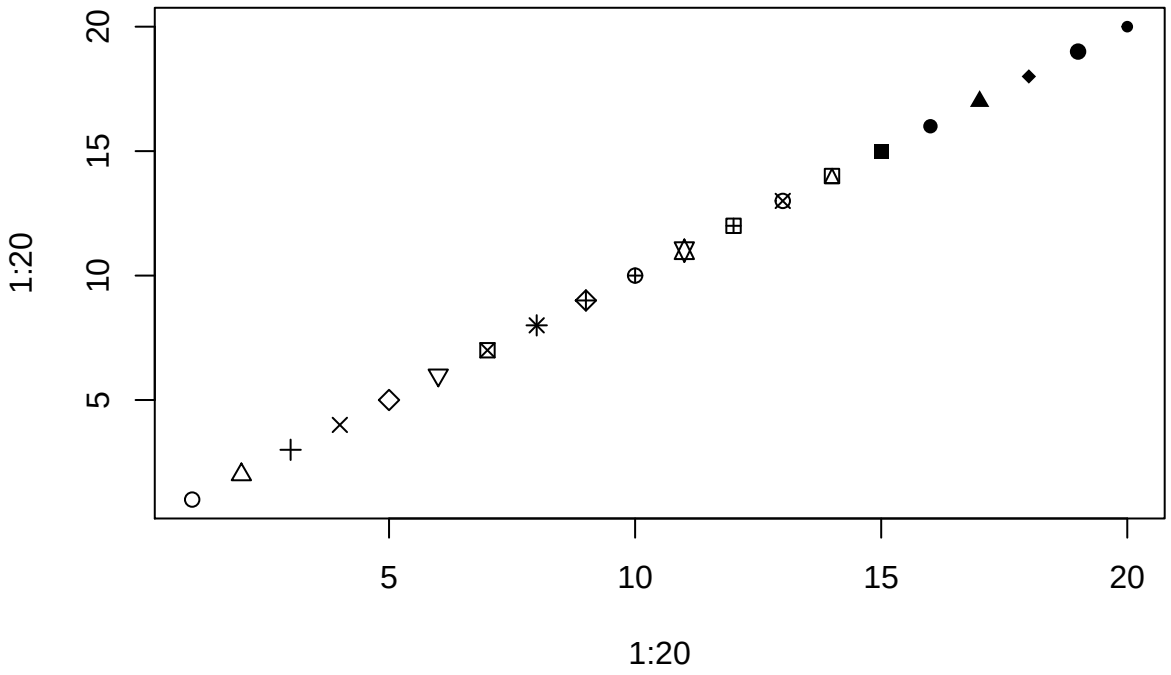
Let's play with the look of the plot, and makes it perttier!

```r
#scatterplot of lstat vs. medv

plot(log(lstat), medv,
     xlab = "Log Transform of % of Houshold with Low Socioeconomic Income",
     ylab = "Median House Value",
     col = "red",
     pch = 20)

# Make the line color blue, and the line's width =3 (play with the width!)
abline(model, col = "blue", lwd =3)
```



Do you want to see what point character (pch) you want to choose? Try this:

```r
plot(1:20, 1:20, pch = 1:20)
```



STEP 5: Assess your model! Final thing to do would be to predict using our fitted model. We can use the "predict()" function for this purpose:

```
#predict what is the median value of the house with lstat= 5%
predict(model, data.frame(lstat = c(5)))
```

```
##     1
## 32.14
```

```
#predict what is the median values of houses with lstat= 5%, 10%, and 15%
predict(model, data.frame(lstat = c(5,10,15), interval = "prediction"))
```

```
##     1     2     3
## 32.14 23.68 18.74
```

Now let's assess our model, by computing th mean squared error (MSE). To assess the model we created, then we will be using the test data!

```
#save the testing median values for houses (testing y) in y
y = testing_data$medv

#compute the predicted value for this y (y hat)
y_hat = predict(model, data.frame(lstat = testing_data$lstat))

#Now we have both y and y_hat for our testing data. let's find the mean square error

error = y-y_hat
error_squared = error^2
MSE = mean(error_squared)
MSE
```

```
## [1] 17.7
```

The mean squared error for our model is 17.7. This number will be useful later on, when we create another model using the same data but multiple variables. Do you think this MSE would go lower or higher for this dataset? That would be our next tast!