Lab - Linear Regression

A linear regression is a statistical model that analyses the relationship between a response variable (often called y) and one or more variables and their interactions (often called x or explanatory variables). You make this kind of relationships in your head all the time.

For example, when you calculate the age of a child based on her height, you are assuming the older she is, the taller she will be.

Linear regression is one of the most basic statistical models out there, its results can be interpreted by almost everyone, and it has been around since the 19th century.

This is precisely what makes linear regression so popular. It's simple, and it has survived for hundreds of years. Even though it is not as sophisticated as other algorithms like artificial neural networks or random forests, according to a survey made by KD Nuggets, regression was the algorithm most used by data scientists in 2016 and 2017. It's even predicted it's still going to be the used in year 2118!

In [ ]:

```
#Here's the data we will use, one year of marketing spend and company sales by
m onth.
dataset = read.csv("data-marketing-budget-12mo.csv")
```

Month	Spend	Sales
1	1000	9914
2	4000	40487
3	5000	54324
4	4500	50044
5	3000	34719
6	4000	42551
7	9000	94871
8	11000	118914
9	15000	158484
10	12000	131348
11	7000	78504
12	3000	36284

First use ggplot to plot a scatter plot between variables:

In [ ]:

```
ggplot(data = dataset, aes(x = Sales, y = Spend))
  + geom_point(alpha= 0.3, color= "blue")
```

Q1: Write a command to plot sales for each month?

Simple (One Variable) and Multiple Linear Regression Using lm()

The predictor (or independent) variable for our linear regression will be Spend and the dependent variable (the one we're trying to predict) will be Sales (.

The lm function really just needs a formula (Y~X) and then a data source. We'll use Sales~Spend, data=dataset and we'll call the resulting linear model "fit".

One variable:

In [ ]:

```
simple.fit = lm(Sales~Spend, data=dataset)
summary(simple.fit)
```

Multiple variables:

Notices on the multi.fit line the Spend variables is accompanied by the Month variable and a plus sign (+). The plus sign includes the Month variable in the model as a predictor (independent) variable.

The summary function outputs the results of the linear regression model.

In [ ]:

```
multi.fit = lm(Sales~Spend+Month, data=dataset)
summary(multi.fit)
```

Let's look at the output:

Simple Output

```
Call:
lm(formula = Sales ~ Spend, data = dataset)

Residuals:
   Min     1Q Median     3Q    Max
 -3385  -2097    258   1726   3034

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 1383.4714  1255.2404   1.102    0.296
Spend         10.6222     0.1625  65.378 1.71e-14 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2313 on 10 degrees of freedom
Multiple R-squared:  0.9977, Adjusted R-squared:  0.9974
F-statistic:  4274 on 1 and 10 DF,  p-value: 1.707e-14
```

Multiple Regression Output

```
Call:
lm(formula = Sales ~ Spend + Month, data = dataset)

Residuals:
      Min       1Q   Median       3Q      Max
-1793.73 -1558.33    -1.73  1374.19  1911.58

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -567.6098  1041.8836  -0.545  0.59913
Spend         10.3825     0.1328  78.159 4.65e-14 ***
Month        541.3736   158.1660   3.423  0.00759 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1607 on 9 degrees of freedom
Multiple R-squared:  0.999, Adjusted R-squared:  0.9988
F-statistic:  4433 on 2 and 9 DF,  p-value: 3.368e-14
```

Both models have significant models (see the F-Statistic for Regression) and the Multiple R-squared and Adjusted R-squared are both exceptionally high (keep in mind, this is a simplified example). We also see that all of the variables are significant (as indicated by the "**")

Interpreting R's Regression Output

- **Residuals:** The section summarizes the residuals, the error between the prediction of the model and the actual results. Smaller residuals are better.
- **Coefficients:** For each variable and the intercept, a weight is produced and that weight has other attributes like the standard error, a t-test value and significance.
- **Estimate:** This is the weight given to the variable. In the simple regression case (one variable plus the intercept), for every one dollar increase in Spend, the model predicts an increase of 10.6222.
- **Std. Error:** Tells you how precisely was the estimate measured. It's really only useful for calculating the t-value.
- **Residual Standard Error:** This is the standard deviation of the residuals. Smaller is better.
- **Multiple / Adjusted R-Square:** For one variable, the distinction doesn't really matter. R-squared shows the amount of variance explained by the model. Adjusted R-Square takes into account the number of variables and is most useful for multiple-regression.

In [ ]:

```
# Display each: # capture model summary as an object
modelSummary <- summary(simple.fit)

# model coefficients
modelCoeffs <- modelSummary$coefficients

# get beta estimate for Spend - 10.6222
beta.estimate <- modelCoeffs["Spend", "Estimate"]

# get std.error for Spend - 0.1624745
std.error <- modelCoeffs["Spend", "Std. Error"]

# get t value for Spend - 65.37761
t_value <- modelCoeffs["Spend", "t value"]

# get model F-statistic - 4274 1 10
f <- modelSummary$fstatistic
f_statistic <- modelSummary$fstatistic[1]

# get model p-value - 1.707e-14
model_p <- pf(f[1], f[2], f[3], lower=FALSE)

# get model R-squared - 0.9976659
r_2 <- modelSummary$r.squared
```

Q2: Based on residual, which model is better? Why?

Coefficients:

You can see in your results the values of the intercept (“b0” value) and the slope (“b1” value) for the spend. These “b0” and “b1” values plot a line between all the points of the data. So, in this case, if there is a spend that is 200, b0 is 1383.4714 and b1 is 10.6222, the model predicts (on average) that its sales is around 1383.4714 + (10.6222* 200) = $3508.18.

When a regression takes into account two or more predictors to create the linear regression, it’s called multiple linear regression. By the same logic you used in the simple example before, the sales value is going to be measured by:

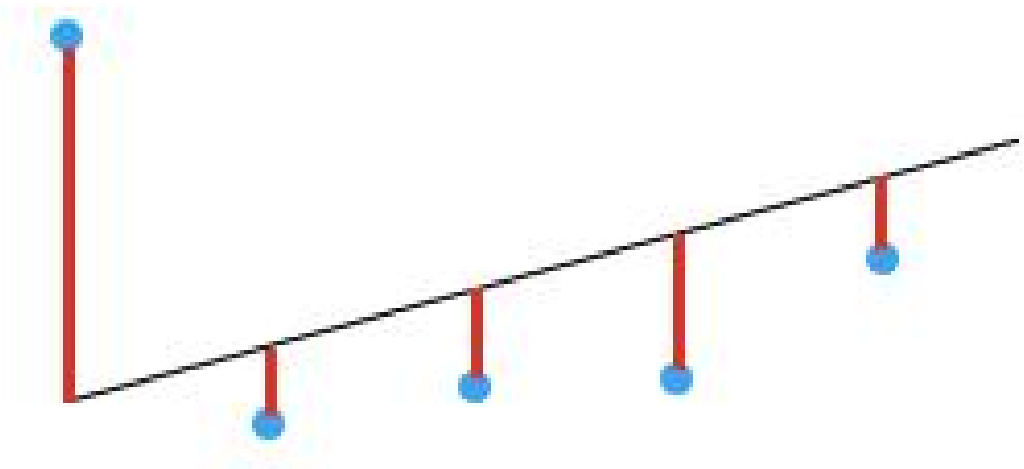
Sales = b0 + Spend × b1 + (Month) × b2

You are now looking at the sales as a function of the spend and month.

Q3: Try to write the multiple regression equation based on the numbers in the output (round to 1 decimal place).

Analysing Residuals

A good way to test the quality of the fit of the model is to look at the residuals or the differences between the real values and the predicted values. The straight line in the below image represents the predicted values. The red vertical line from the straight line to the observed data value is the residual.

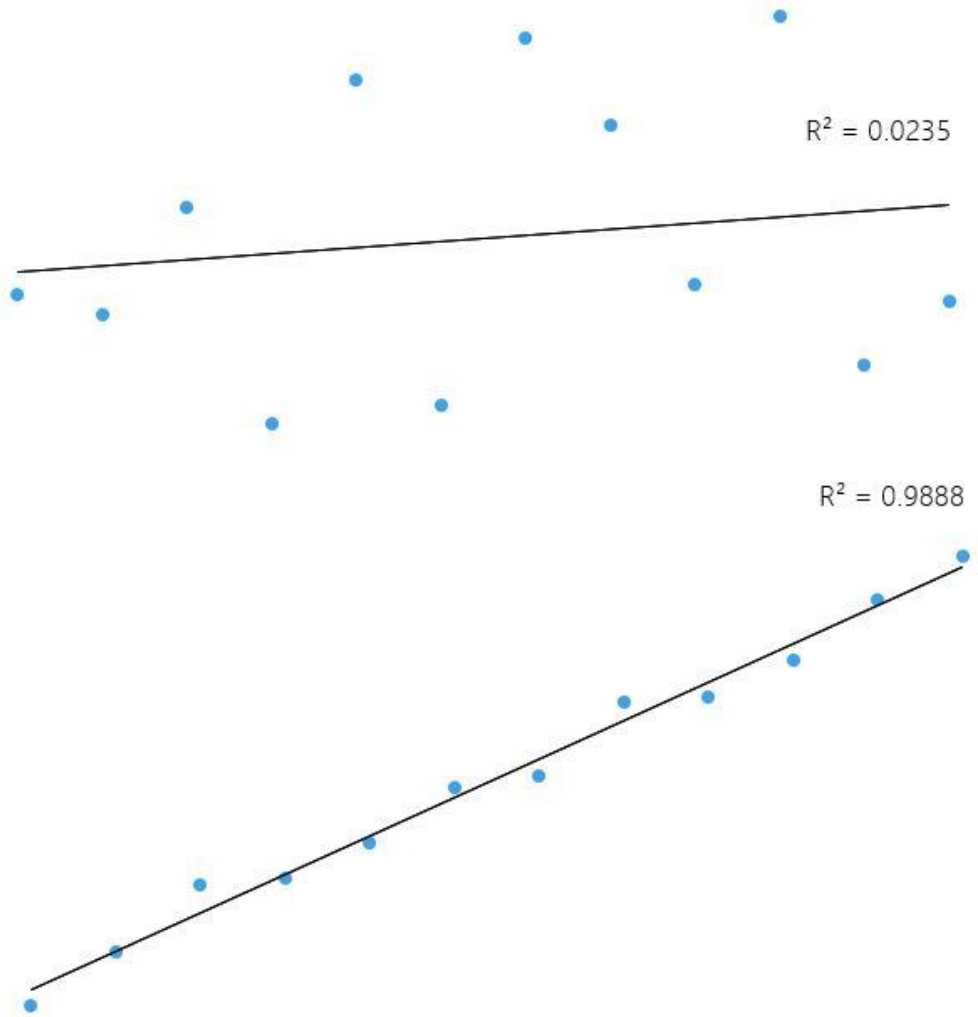


The idea in here is that the sum of the residuals is approximately zero or as low as possible. In real life, most cases will not follow a perfectly straight line, so residuals are expected. In the R summary of the lm function, you can see descriptive statistics about the residuals of the model.

How to test if your linear model has a good fit?

One measure very used to test how good is your model is the coefficient of determination or R^2. This measure is defined by the proportion of the total variability explained by the regression model.

This can seem a little bit complicated, but in general, for models that fit the data well, R^2 is near 1. Models that poorly fit the data have R^2 near 0. In the examples below, the first one has an R^2 of 0.02; this means that the model explains only 2% of the data variability. The second one has an R^2 of 0.99, and the model can explain 99% of the total variability.



Don’t forget to look at the residuals!

R2 and residual:

You can have a pretty good R^2 in your model, but let's not rush to conclusions here. Let's see an example. You are going to predict the pressure of a material in a laboratory based on its temperature.

Let's plot the data (in a simple scatterplot) and add the line you built with your linear model. In this example, let R read the data first, again with the read_excel command, to create a dataframe with the data, then create a linear regression with your new data. The command plot takes a data frame and plots the variables on it. In this case, it plots the pressure against the temperature of the material. Then, add the line made by the linear regression with the command abline.

In [ ]:

```
pressure <- read_excel("pressure.xlsx") #Upload the data
lmTemp = lm(Pressure~Temperature, data = pressure) #Create the linear regression
plot(pressure, pch = 16, col = "blue") #Plot the results
abline(lmTemp) #Add a regression line
```

If you see the summary of your new model, you can see that it has pretty good results (look at the $R^2$and the adjusted R^2)

In [ ]:

```
summary(lmTemp)
```

Ideally, when you plot the residuals, they should look random. Otherwise means that maybe there is a hidden pattern that the linear model is not considering. To plot the residuals, use the command plot(lmTemp$residuals).

In [ ]:

```
plot(lmTemp$residuals, pch = 16, col = "red")
```

In [ ]:

```
hist(lmTemp$resid, main="Histogram of
 Residuals", ylab="Residuals")
```

Is this random?

If not, that means there is a hidden pattern. Can be solved by advanced analytics

Use linear regression to predict:

In [ ]:

```
a <- data.frame(Temperature = 170)
result <-  predict(lmTemp,a)
print(result)
```

Q4: Use the linear regression to predict the pressure for temperature 40. Write your result.

Linear regression to impute missing values:

Consider the follwoing data:

In [ ]:

```
x <- 1:10
y <- c(11,12,18,14,17, NA,NA,19,NA,27)
z <- sample(1:20, 10)
w <- c(seq(1,10,3), 3,5,7,6,6,9)
```

In [ ]:

```
data <- data.frame(x,y,z,w)

data
```

In [ ]:

```
summary(data)
```

1. Creating a dummy variable that will indicate missing data:

In [ ]:

```
#For that let us create a function that will make our task easier
#The following function takes a vector as an argument and returns a binary vecto r
of 0 corresponding the missing value in the argument vector and 1 otherwise.

missDummy <- function(t)
{
  x <- dim(length(t))
  x[which(!is.na(t))] = 1
  x[which(is.na(t))] = 0
  return(x)
}

#Now we will use this function to create a dummy variable that will indicate
mis sing value using 0, otherwise willtake the value 1.

data$dummy <- missDummy(data$y)

#Let us take a look at the data
data
```

2. Next let us split data to 2sets (train and test):

In [ ]:

```
# Pick the instances with known y values as training data
TrainData<- data[data['dummy']==1,]
# Pick the instances with unknown (NA) y values as testing data
TestData<- data[data['dummy']==0,]
```

In [ ]:

```
# Exclude last column
TrainData<- TrainData[,-5]
TestData<- TestData[,-5]
```

3. Let's then fit a linear model with y as dependent variable and x as independent variable.

In [ ]:

```
model<- lm(y~x+z+w, TrainData)
```

4. Predict missing values based on the model:

In [ ]:

```
pred<-  predict(model, TestData)
pred
```

5. Insert it back in the original

In [ ]:

```
# Where are NAs?
data$y[is.na(y)]

# Replace with predicted
data$y[is.na(y)]<- pred
```