Linear Regression - Part 1 - Edx Analytical Edge

FdR

3 January 2015

This document is inspired and use the *EDx course - Analytical Edge* resources.

Linear regression is one of the easiest tool in the predicitve analytics field. This paper intends to show by example how to make it works with R using the great libraries of dplyr, ggplot2 whenever necessary.

## Single variable regression.

The general equation for a linear regression model

where:

* is the observation of the dependent variable
* is the intercept coefficient
* is the regression coefficient for the dependent variable
* is the observation of the independent variable
* is the error term for the observation. It basically is the difference in therm of y between the observed value and the estimated value. It is also called the residuals. A good model minimize these errors.

Some ways to assess how good our model is to:

1. compute the SSE (the sum of squared error)
   * SSE = =
   * A good model will minimize SSE
   * problem: SSE is dependent of N. SSE will naturally increase as N increase
2. compute the RMSE (the root mean squared error)
   * RMSE =
   * Also a good model will minimize SSE
   * It depends of the unit of the dependent variable. It is like the average error the model is making (in term of the unit of the dependent variable)
3. compute
   * It compare the models to a baseline model
   * is **unitless** and **universaly** interpretable
   * SST is the sum of the squared of the difference between the observed value and the mean of all the observed value

### First example. Predicting wine price.

The wine.csv file is used in the class.

Let's load it and then have a quick look at its structure.

wine = read.csv("wine.csv")  
str(wine)

## 'data.frame': 25 obs. of 7 variables:  
## $ Year : int 1952 1953 1955 1957 1958 1959 1960 1961 1962 1963 ...  
## $ Price : num 7.5 8.04 7.69 6.98 6.78 ...  
## $ WinterRain : int 600 690 502 420 582 485 763 830 697 608 ...  
## $ AGST : num 17.1 16.7 17.1 16.1 16.4 ...  
## $ HarvestRain: int 160 80 130 110 187 187 290 38 52 155 ...  
## $ Age : int 31 30 28 26 25 24 23 22 21 20 ...  
## $ FrancePop : num 43184 43495 44218 45152 45654 ...

head(wine)

## Year Price WinterRain AGST HarvestRain Age FrancePop  
## 1 1952 7.4950 600 17.1167 160 31 43183.57  
## 2 1953 8.0393 690 16.7333 80 30 43495.03  
## 3 1955 7.6858 502 17.1500 130 28 44217.86  
## 4 1957 6.9845 420 16.1333 110 26 45152.25  
## 5 1958 6.7772 582 16.4167 187 25 45653.81  
## 6 1959 8.0757 485 17.4833 187 24 46128.64

We use the lm function to find our linear regression model. We use *AGST* as the independent variable while the *price* is the dependent variable.

model1 = lm(Price ~ AGST, data = wine)  
summary(model1)

##   
## Call:  
## lm(formula = Price ~ AGST, data = wine)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.78450 -0.23882 -0.03727 0.38992 0.90318   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -3.4178 2.4935 -1.371 0.183710   
## AGST 0.6351 0.1509 4.208 0.000335 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.4993 on 23 degrees of freedom  
## Multiple R-squared: 0.435, Adjusted R-squared: 0.4105   
## F-statistic: 17.71 on 1 and 23 DF, p-value: 0.000335

The summary function applied on the model is giving us a bunch of important information

* the stars next to the predictor variable indicated how significant the variable is for our regression model
* it also gives us the value of the R coefficient

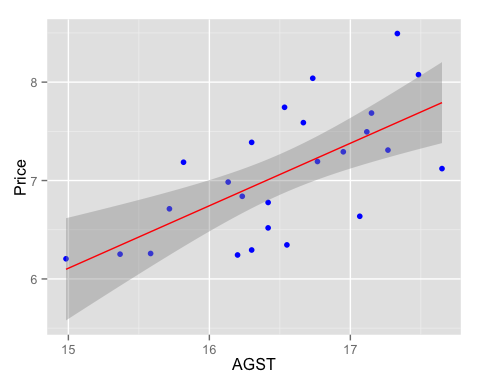
We could have calculated the R value ourselves:

SSE = sum(model1$residuals^2)  
SST = sum((wine$Price - mean(wine$Price))^2)  
r\_squared = 1 - SSE/SST  
r\_squared

## [1] 0.4350232

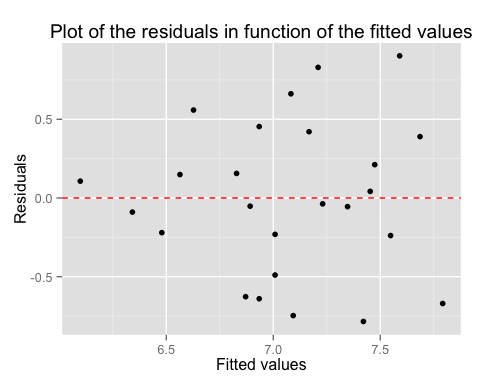
We can now plot the observations and the line of regression; and see how the linear model fits the data.

library(ggplot2)  
ggplot(wine, aes(AGST, Price)) + geom\_point(col = "blue") + geom\_smooth(method = "lm", col = "red")



It is always nice to see how our residuals are distributed.  
We use the ggplot2 library and the fortify function which transform the summary(model1) into a data frame usable for plotting.

model1 <- fortify(model1)  
p <- ggplot(model1, aes(.fitted, .resid)) + geom\_point() + geom\_hline(yintercept = 0, col = "red", linetype = "dashed")   
p <- p + xlab("Fitted values") + ylab("Residuals") + ggtitle("Plot of the residuals in function of the fitted values")  
p



## Multi-variables regression.

Instead of just considering one variable as predictor, we'll add a few more variables to our model with the idea to increase its predictive ability.

We have to be cautious in adding more variables. Too many variable might give a high on our training data, but this not be the case as we switch to our testing data.

The general equations can be expressed as

when there are k predictors variables.

There are a bit of trials and errors to make while trying to fit mutliple variables into a model, but a rule of thumb would be to include most of the variable (all these that would make sense) and then take out the ones that are not very significant using the summary(modelx)

### First example. Predicting wine price.

We continue here with the same dataset, *wine.csv*.  
First, we can see how each variable is correlated with each other ones, using

cor(wine)

## Year Price WinterRain AGST HarvestRain  
## Year 1.00000000 -0.4477679 0.016970024 -0.24691585 0.02800907  
## Price -0.44776786 1.0000000 0.136650547 0.65956286 -0.56332190  
## WinterRain 0.01697002 0.1366505 1.000000000 -0.32109061 -0.27544085  
## AGST -0.24691585 0.6595629 -0.321090611 1.00000000 -0.06449593  
## HarvestRain 0.02800907 -0.5633219 -0.275440854 -0.06449593 1.00000000  
## Age -1.00000000 0.4477679 -0.016970024 0.24691585 -0.02800907  
## FrancePop 0.99448510 -0.4668616 -0.001621627 -0.25916227 0.04126439  
## Age FrancePop  
## Year -1.00000000 0.994485097  
## Price 0.44776786 -0.466861641  
## WinterRain -0.01697002 -0.001621627  
## AGST 0.24691585 -0.259162274  
## HarvestRain -0.02800907 0.041264394  
## Age 1.00000000 -0.994485097  
## FrancePop -0.99448510 1.000000000

by default, R uses the Pearson coefficient of correlation.  
So let's start by using all variables.

model2 <- lm(Price ~ Year + WinterRain + AGST + HarvestRain + Age + FrancePop, data = wine)  
summary(model2)

##   
## Call:  
## lm(formula = Price ~ Year + WinterRain + AGST + HarvestRain +   
## Age + FrancePop, data = wine)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.48179 -0.24662 -0.00726 0.22012 0.51987   
##   
## Coefficients: (1 not defined because of singularities)  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 7.092e-01 1.467e+02 0.005 0.996194   
## Year -5.847e-04 7.900e-02 -0.007 0.994172   
## WinterRain 1.043e-03 5.310e-04 1.963 0.064416 .   
## AGST 6.012e-01 1.030e-01 5.836 1.27e-05 \*\*\*  
## HarvestRain -3.958e-03 8.751e-04 -4.523 0.000233 \*\*\*  
## Age NA NA NA NA   
## FrancePop -4.953e-05 1.667e-04 -0.297 0.769578   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.3019 on 19 degrees of freedom  
## Multiple R-squared: 0.8294, Adjusted R-squared: 0.7845   
## F-statistic: 18.47 on 5 and 19 DF, p-value: 1.044e-06

While doing so, we notice that the variable *Age* has NA (issues wiht missing data?) and that the variable *FrancePop* isn't very predictive of the price of wine. So we can refine our models, by taking out these 2 variables, and as we'll see, it won't affect much our value. Note that with multiple variables regression, it is important to look at the **Adjusted R-squared** as it take into consideration the amount of variables in the model.

model3 <- lm(Price ~ Year + WinterRain + AGST + HarvestRain, data = wine)  
summary(model3)

##   
## Call:  
## lm(formula = Price ~ Year + WinterRain + AGST + HarvestRain,   
## data = wine)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.45470 -0.24273 0.00752 0.19773 0.53637   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 44.0248601 16.4434570 2.677 0.014477 \*   
## Year -0.0239308 0.0080969 -2.956 0.007819 \*\*   
## WinterRain 0.0010755 0.0005073 2.120 0.046694 \*   
## AGST 0.6072093 0.0987022 6.152 5.2e-06 \*\*\*  
## HarvestRain -0.0039715 0.0008538 -4.652 0.000154 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.295 on 20 degrees of freedom  
## Multiple R-squared: 0.8286, Adjusted R-squared: 0.7943   
## F-statistic: 24.17 on 4 and 20 DF, p-value: 2.036e-07

Although it isn't now feasible to graph in 2D the *Price* in function of the other variables, we can still graph our residuals.

model3 <- fortify(model3)  
ggplot(model3, aes(.fitted, .resid)) + geom\_point() + geom\_hline(yintercept = 0, col = "red", linetype = "dashed") + xlab("Fitted values") + ylab("Residuals") + ggtitle("Plot of the residuals in function of the fitted values (multiple variables)")

