# Linear and ridge regressions

Anna Errichiello MSc Data Science and Economics, University of Milan email: [anna.errichiello@studenti.unimi.it](mailto:anna.errichiello@studenti.unimi.it)

## Abstract

This contribution develops an analysis on the dataset ‘bad-drivers.csv’ (taken from github.com/fivethirtyeight/data) by considering two models: the linear regression and the ridge regression. In particular, the analysis is focused con the computation of the predictions of the variable y, following the partition of the dataset into training set and test set. Finally, the study aims to find the best value of lambda used in the ridge regression, passing through the computation of the regularization parameter β.

## Goal of the analysis

This study wants to give an overview of the difference between the linear model and the ridge one, underlying the main element that defines a dissimilarity between the two: the presence of the tuning parameter lambda in the ridge regression that helps us in controlling the bias of the algorithm. In this case there is an improvement of the accuracy of the model for the dataset. The main point is lambda, to which is associated the ridge parameter β and thanks to this it is possible to find the predictions and check which is the best value of lambda among those values that have been previously chosen.

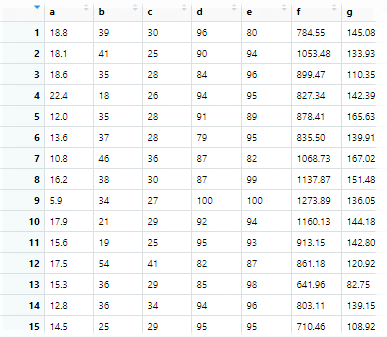
Finally, the keypoints of this study are:

* difference between linear regression and ridge regression (analysis of the main characteristics),
* improvement of the accuracy of the ridge regression model,
* the tuning parameter lambda and the dependence of the ridge parameter on lambda, considering that the predictions Y’ = Xβ.

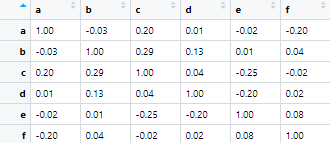
## Description of the dataset

I choose the dataset ‘bad-drivers.csv’ (taken from github.com/fivethirtyeight/data), that considers data whose sources are: National Highway Traffic Safety Administration and National Association of Insurance Commissioners. This folder contains data behind the story ‘Dear Mona, Which State Has The Worst Drivers?’, where Mona Chalabi, a British data journalist answer some questions about drivers and car accidents in the USA. She says that historic data that could indicate where America’s worst drivers are: the number of car crashes in each state (especially those where the driver was negligent in some way), how much insurance companies pay out, and how much insurance companies charge drivers. The reported variables are:

* State (a specific country in the USA);
* Number of drivers involved in fatal collisions per billion miles;
* Percentage of drivers involved in fatal collisions who were speeding;
* Percentage of drivers involved in fatal collisions who were alcohol impaired;
* Percentage of drivers involved in fatal collisions who were not distracted;
* Percentage of drivers involved in fatal collisions who had not been involved in any previous accidents;
* Car insurance premiums;
* Losses incurred by insurance companies for collisions per insured driver.

I report the first fiftheen rows of the dataset:

## Data partitioning

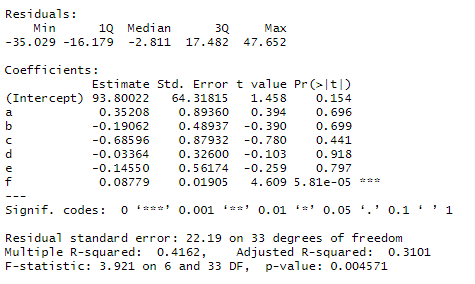
After importing the dataset, I renamed the columns from the second one to the last one and I choose not to consider the first column since each row refers to a different state and it is easier to run the analysis without it. I started my study by splitting the dataset into training set and test set. I choose to consider as training set the 80% of the dataset and, the remaining 20%, as the test set. It is important to define the variables x and y for the data set, the training set and the test set. I choose as x the matrix that contains all the variables except for the one that I considered as the ‘response’ variable y, which is ‘Losses incurred by insurance companies for collisions per insured driver’ that I renamed ‘g’. I run also a correlation test in order to check the measure of some kind of correlation between couple of variables. The correlation test is the following:

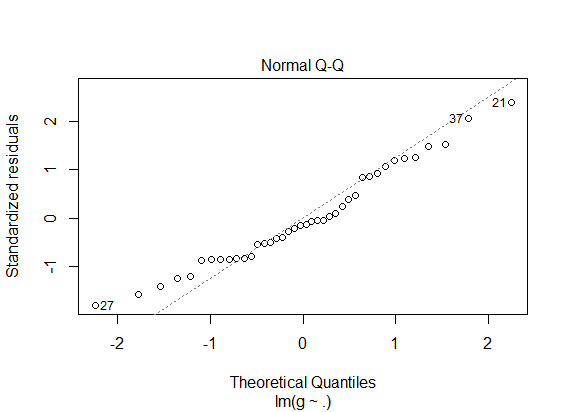
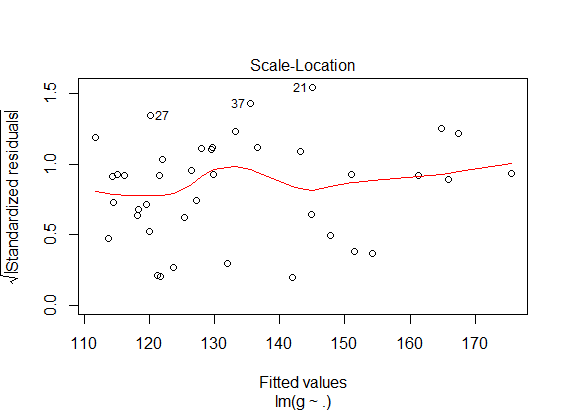
The interpretation of this test, in which the absolute values are between 0 and 1 is the following: if the correlation is equal to 1 then the relationship is linear, otherwise, in case the value is close to 0, it is non-linear. Furthermore, if both values tend to increase or decrease together the coefficient is positive, and the line that represents the correlation slopes upward, otherwise the coefficient is negative.

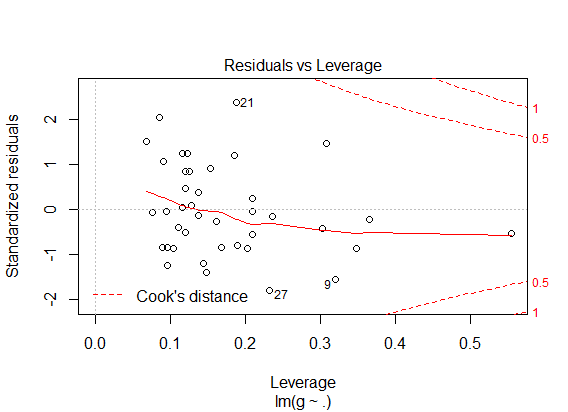
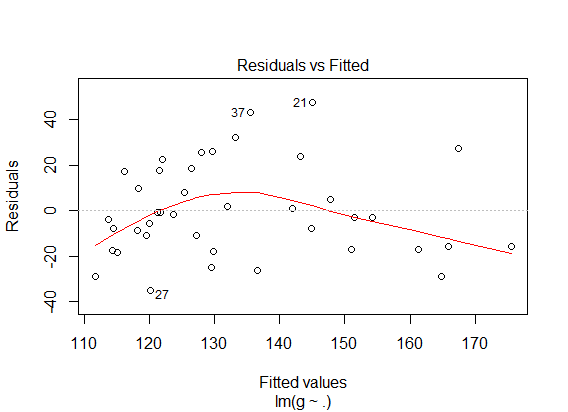
## Linear Regression and Ridge Regression

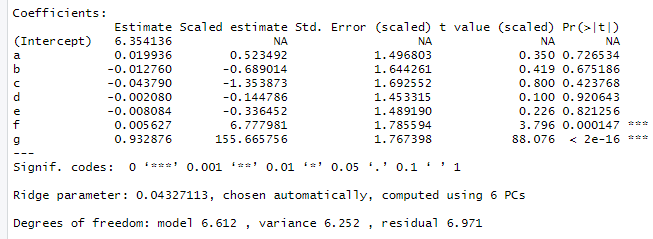
I run the linear model and the ridge one respectively with the commands ‘lm()’ and ‘linearRidge’. The linear model has an equation of the form Y = a + bX, in which ‘X’ is the explanatory variable and ‘Y’ is the dependent variable. The slope of the line is b, and a is the intercept (the value of y when x = 0). However the ridge regression can be represented by the equation Y = Xβ + e where ‘X’ is a high-dimensional matrix,; ‘β’, which is the regression coefficient to be estimated; ‘Y’ is the dependent ‘response’ variable. In fact, as I reported at the beginning the main difference between the linear regression and the ridge one is given by the presence in the latter of the tuning parameter lambda. The coefficient estimates β= (β0,β1,…,βp) are given by: β≡argminβ||Y-Xβ||^(2)+λβ’Dβ where D is a diagonal matrix with 0 in the [1,1] position and ones in the rest of the diagonal. The part of the object of the argmin function where it’s contained lambda is called ’penalty’ and if λ=0 then we get a linear regression, whereas if λ grows then the estimates of β will get closer and closer to 0. Using the matrix notations, it is possible to write the ridge parameter as β=((t(X)X + λ\*I)^(-1))t(X)Y where t(x) is the transpose matrix of x. By running the two models on the training set and then computing the predictions on the test sets, it is possible calculate the accuracy of the model by running the min/max ratio that measure how far is the model’s prediction from the real value. The better the prediction the higher it will be. In this case there is an improvement of accuracy by switching from linear regression to ridge regression, in fact for the linear model the accuracy is around 90%, that is very high, whereas for ridge regression is more or less 99%, which is even higher. Even the Root Squared Error is around 99% and it confirms the goodness of the ridge model.

The output of the linear model is:



and the associated graphs are:

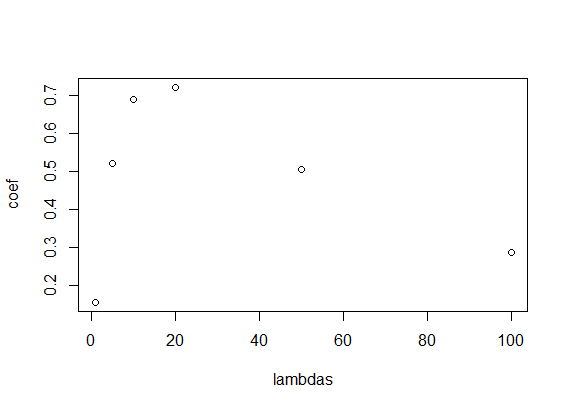
The output for the Ridge Regression is:

## 

## Regularization and tuning parameters

The ridge parameter is given by the equation β=((t(X)X + λ\*I)^(-1))t(X)Y and predictions of y are given by Y’=X((t(X)X+λI)^(-1))t(X)Y: this implies that Y’=Xβ.

I decided to give lambdas the values 1,5,10,20,50,100. Then I built a function where I run the ‘lm.ridge’ function from the R package ‘MASS’ with each value I choose for lambda, I take the first coefficient and use it as the ridge parameter in order to compute the predictions Y’=Xβ and, finally, I calculated the Mean Squared Error for each value of β, generated by the function ‘lm.ridge’. By running this function, the result shows that the lowest Mean Squared Error is obtained by using 1 as the value of lambda.

The relation between λ and β is non-linear as the following graph shows:

## Conclusions

In conclusion, both the linear model and the ridge one are good models for the analysis of the dataset, since they show very high values of accuracy. The introduction of lambda in the ridge regression contributes in some changes and helps in the improvement of the model used. As previously reported, if λ=0, then a linear regression would occur, whereas, if the value of lambda is strictly greater than 0, it will lead to the computation of the ridge parameter.

The ridge regression helps in facing the problem of multicollinearity that refers to data that are affected by a non-linear relationships between independent variables: in case of least squares regressions, this would cause unbiased estimates, however the variances are very large and far away from the true value.

The choice of lambda is crucial, since an optimal value, as reported in the analysis, generates a lower error. Furthermore, as shown in the last graph, the ridge parameter and λ have a non-linear relationship and if λ=1, β is very low, then it grows very fast in the interval where the values of λ are between 5 and 20; after that value, β starts decreasing. The computation of β leads to the calculation of the predictions of Y since they are equal to Y’=Xβ. Thanks to this, it is possible to define which is the optimal value of λ among those that have been chosen.

## Appendix

library(tidyverse)

library(car)

library(corrplot)

library(ridge)

library(MASS)

#import the dataset  
bad\_drivers <- read.csv("https://raw.githubusercontent.com/fivethirtyeight/data/master/bad-drivers/bad-drivers.csv", sep = ",")

head(bad\_drivers)

dir.create("data")

save(bad\_drivers, file=file.path("data","bad\_drivers.rda"))

#rename the columns (for simplicity)  
a <- bad\_drivers$Number.of.drivers.involved.in.fatal.collisions.per.billion.miles

b <- bad\_drivers$Percentage.Of.Drivers.Involved.In.Fatal.Collisions.Who.Were.Speeding

c <- bad\_drivers$Percentage.Of.Drivers.Involved.In.Fatal.Collisions.Who.Were.Alcohol.Impaired

d <- bad\_drivers$Percentage.Of.Drivers.Involved.In.Fatal.Collisions.Who.Were.Not.Distracted

e <- bad\_drivers$Percentage.Of.Drivers.Involved.In.Fatal.Collisions.Who.Had.Not.Been.Involved.In.Any.Previous.Accidents

f <- bad\_drivers$Car.Insurance.Premiums....

g <- bad\_drivers$Losses.incurred.by.insurance.companies.for.collisions.per.insured.driver....

#dataset  
bad\_drivers <- data.frame(a,b,c,d,e,f,g)  
view(bad\_drivers)  
  
x <- bad\_drivers[,-7] %>% as.matrix()

y <- bad\_drivers$g

view(round(cor(x), 2)) # Correlation Test  
corrplot(round(cor(x), 2)) #correlation graph  
  
trainingIndex <- sample(nrow(bad\_drivers), 0.80\*nrow(bad\_drivers)) # index for 80%  
trainingData <- bad\_drivers[trainingIndex, ] # training data  
x\_train <- trainingData [,-7] %>% as.matrix()  
y\_train <- trainingData$g  
  
testData <- bad\_drivers[-trainingIndex, ] # test data  
x\_test <- testData[,-7] %>% as.matrix()  
y\_test <- testData$g  
  
lmMod <- lm(g ~ ., trainingData) # the linear reg model  
summary (lmMod) # get summary  
plot(lmMod)  
  
predicted <- predict(lmMod, testData) # predict on test data  
compare <- cbind(actual=testData$g, predicted) # combine actual and predicted  
show (compare)  
  
mean (apply(compare, 1, min)/apply(compare, 1, max)) # calculate accuracy

# the ridge regression model  
ridge\_regr <- linearRidge(y\_train ~ ., data = trainingData)   
summary(ridge\_regr)  
  
y\_predicted <- predict(ridge\_regr, testData) # predict on test data  
compare <- cbind (actual=testData$g, y\_predicted) # combine  
compare  
mean (apply(compare, 1, min)/apply(compare, 1, max))  
  
# Sum of Squares Total and Error  
sst <- sum((y\_test - mean(y\_test))^2)  
sse <- sum((y\_predicted - y\_test)^2)  
  
# R squared  
rsq <- 1 - sse / sst  
rsq  
  
lambdas <- c(1,5,10,20,50,100)  
table <- data.frame()  
{coef <- c((ridge1 <- lm.ridge(y\_train ~ ., lambda = 1, data = trainingData))$coef[1],   
 (ridge2 <- lm.ridge(y\_train ~ ., lambda = 5, data = trainingData))$coef[1],  
 (ridge3 <- lm.ridge(y\_train ~ ., lambda = 10, data = trainingData))$coef[1],  
 (ridge4 <- lm.ridge(y\_train ~ ., lambda = 20, data = trainingData))$coef[1],  
 (ridge5 <- lm.ridge(y\_train ~ ., lambda = 50, data = trainingData))$coef[1],  
 (ridge6 <- lm.ridge(y\_train ~ ., lambda = 100, data = trainingData))$coef[1])  
  
plot(lambdas,coef)  
  
   
 pred <- c(pred1 <- x\_test %\*% matrix(c(ridge1$coef[1]), ncol=1,nrow=6),  
 pred2 <- x\_test %\*% matrix(c(ridge2$coef[1]), ncol=1,nrow=6),  
 pred3 <- x\_test %\*% matrix(c(ridge3$coef[1]), ncol=1,nrow=6),  
 pred4 <- x\_test %\*% matrix(c(ridge4$coef[1]), ncol=1,nrow=6),  
 pred5 <- x\_test %\*% matrix(c(ridge5$coef[1]), ncol=1,nrow=6),  
 pred6 <- x\_test %\*% matrix(c(ridge6$coef[1]), ncol=1,nrow=6))

mse <- c(mse1 <- mean((y\_test-pred1)^(2)),  
 mse2 <- mean((y\_test-pred2)^(2)),  
 mse3 <- mean((y\_test-pred3)^(2)),  
 mse4 <- mean((y\_test-pred4)^(2)),  
 mse5 <- mean((y\_test-pred5)^(2)),  
 mse6 <- mean((y\_test-pred6)^(2)))  
table <- data.frame(lambdas,mse=mse)  
}  
show(table)