Regression Models - Galton

Jeffrey Strickland

2/23/2022

Regression models are the workhorse of data science. They are the most well described, practical and theoretically understood models in statistics. A data scientist well versed in regression models will be able to solve an incredible array of problems.

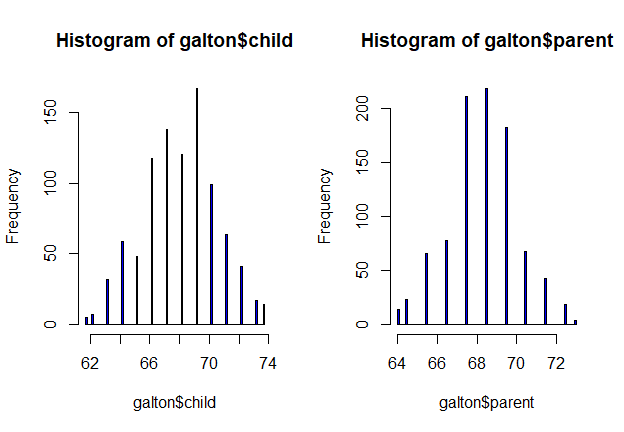
Perhaps the key insight for regression models is that they produce highly interpretable model fits. This is unlike machine learning algorithms, which often sacrifice interpretability for improved prediction performance or automation. These are, of course, valuable attributes in their own rights. However, the benefit of simplicity, parsimony and interpretability offered by regression models (and their close generalizations) should make them a first tool of choice for any practical problem.

Let’s look at the data first, used by Francis Galton in 1885. Galton was a statistician who invented the term and concepts of regression and correlation, founded the journal Biometrika, and was the cousin of Charles Darwin. Galton is a data set from tabulated data set used by Galton in 1885 to study the relationship between a parent’s height and their childrens.

The midparent’s height is an average of the father’s height and 1.08 times the mother’s. In the data there are 205 different parents and 928 children. The data here is truncated at the ends for both parents and children so that it can be treated as numeric data. The data were tabulated and consequently made discrete. The father.son data set is similar data used by Galton and is continuous.

You may need to run install.packages(“UsingR”) if the UsingR library is not installed. Let’s look at the marginal (parents disregarding children and children disregarding parents) distributions first. \* Parent distribution is all heterosexual couples. \* Correction for gender via multiplying female heights by 1.08. \* Overplotting is an issue from discretization.

library(UsingR)  
data(galton)   
par(mfrow=c(1,2))   
hist(galton$child, col="blue", breaks=100)   
hist(galton$parent, col="blue", breaks=100)



### Finding the middle via least squares

In statistics, ordinary least squares (OLS) or linear least squares is a method for estimating the unknown parameters in a linear regression model. This method minimizes the sum of squared vertical distances between the observed responses in the dataset and the responses predicted by the linear approximation. The resulting estimator can be expressed by a simple formula, especially in the case of a single regressor on the right-hand side.

Consider only the children’s heights.

### How could one describe the “middle”?

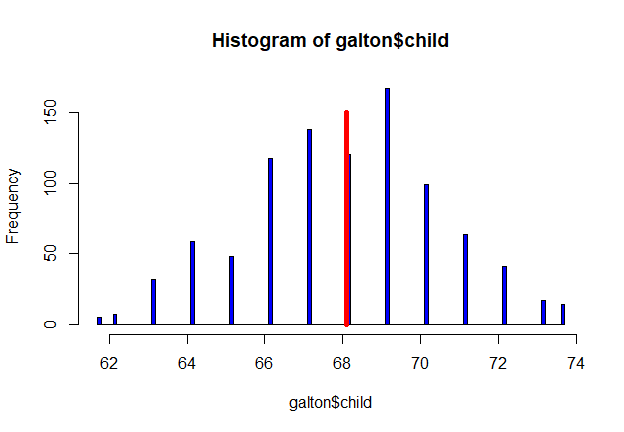
One definition, let Y be the height of child i for i= 1…, n = 928, then define the middle as the value of μ that minimizes

This is physical center of mass of the histogram. You might have guessed that the answer μ= X-bar.

### Experiment

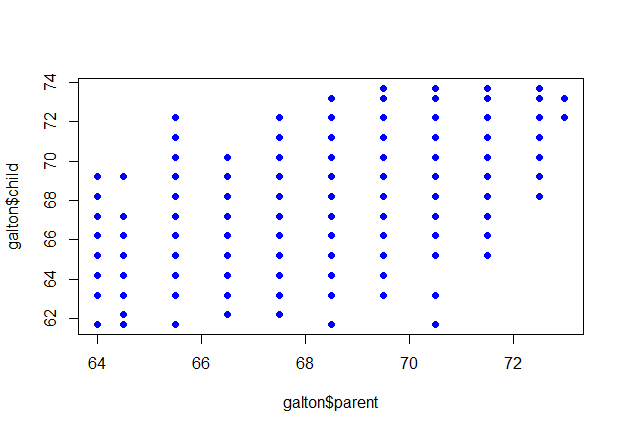
Use RStudio’s manipulate to see what value of μ minimizes the sum of the squared deviations

library(manipulate)   
myHist <- function(mu){  
 hist(galton$child, col= "blue", breaks=100)   
 lines(c(mu, mu), c(0, 150), col="red", lwd=5)   
 mse <- mean((galton$child - mu)^2)  
 text(63, 150, paste("mu = ", mu))  
 text(63, 140, paste("MSE = ", round(mse, 2)))  
}  
#manipulate(myHist(mu), mu = slider(62, 74, step = 0.5))  
hist(galton$child, col = "blue", breaks = 100)   
meanChild <- mean(galton$child)   
lines(rep(meanChild,100), seq(0,150,length=100),col="red", lwd=5)



Comparing childrens’ heights and their parents’ heights is shown in the scatter plot

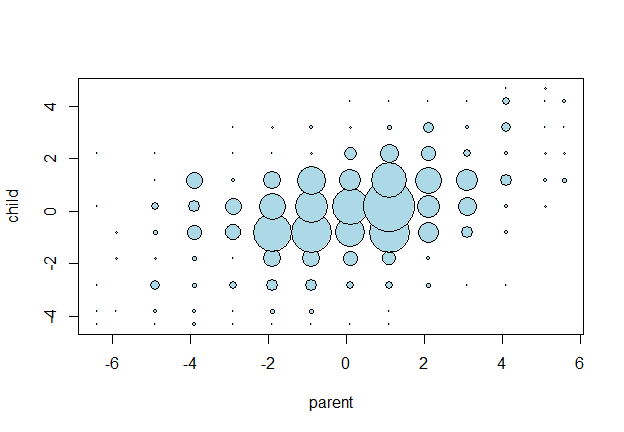
plot(galton$parent, galton$child, pch=19, col = "blue")



### General least squares for linear equations

Consider again the parent and child height data from Galton. In Figure 7 5, we present a scatterplot with point size a function of frequency. That is, the more record with equal heights, the larger the bubble, i.e., 12 records with height equal to 53 inches, appears as a larger bubble that 2 records with height equal to 60 inches.

myPlot <- function(beta){  
 y <- galton$child - mean (galton$child) # Child weight  
 x <- galton$parent - mean (galton$parent) # Parent weight   
 freqData <- as.data.frame(table(x, y))   
 names(freqData) <- c("child", "parent", "freq")   
 plot(  
 as.numeric(as.vector(freqData$parent)),   
 as.numeric(as.vector(freqData$child)),  
 pch = 21, col = "black", bg = "lightblue",  
 cex = .15 \* freqData$freq, # Bubble size  
 xlab = "parent",   
 ylab = "child"  
 )  
}  
myPlot(beta)



### Fitting the best line

Since we are interested in summarizing the trend between two quantitative variables, the natural question arises — “what is the best fitting line?” At some point in your education, you were probably shown a scatter plot of (x, y) data and were asked to draw the “most appropriate” line through the data. Even if you weren’t, you can try it now on a set of parent’s heights (x) and children’s heights (y) of 928 pairs, (Galton dataset). Now, what best summarizes the trend between height and weight?

In order to answer the question of better fit, we first need to introduce some common notation: \* y\_i denotes the observed response for experimental unit i \* x\_i denotes the predictor value for experimental unit i \* y ̂\_i is the predicted response (or fitted value) for experimental unit i

Then, the equation for the best fitting line is:

Incidentally, recall that an “experimental unit” is the object or person on which the measurement is made. In our Galton example, the experimental units are the children, i.e., can we predict the height of the child based on the height of the parent?

### Regression through the origin

To put this in terms of the simple linear regression model, suppose that X\_i are the parents’ heights, and Y\_i are the heights of the children. Then, consider picking the slope β that minimizes. In other words, we want to find the slope that minimizes the squared distance between the heights of the parents and children.

This is exactly using the origin as a pivot point picking the line that minimizes the sum of the squared vertical distances of the points to the line. We will use R’s manipulate() function to experiment. And, we’ll subtract the means so that the origin is the mean of the parent and children’s heights. We make three plots (Figure 7 6, Figure 7 7, Figure 7 8) at three different values of β (0.62, 0.64, and 0.66, respectively) using the manipulate() function.

myPlot <- function(beta){  
 y <- galton$child - mean(galton$child)  
 x <- galton$parent - mean(galton$parent)   
 freqData <- as.data.frame(table(x, y))   
 names(freqData) <- c("child", "parent", "freq")   
 plot(  
 as.numeric(as.vector(freqData$parent)) ,   
 as.numeric(as.vector(freqData$child)),  
 pch = 21, col = "black", bg = "lightblue",  
 cex = .15 \* freqData$freq,   
 xlab = "parent",   
 ylab = "child"  
 )  
 abline(0, beta, lwd = 3) # Added for best fit line  
 points(0, 0, cex = 2, pch = 19) # Added for center  
 mse <- mean((y - beta \* x)^2) # Added for metric  
 title(paste("beta = ", beta, "mse = ", round(mse, 3)))  
 }  
#manipulate(myPlot(beta), beta=slider(0.6, 1.2,step=0.02))

# The solution

library(ModelMetrics)  
fit <- lm(I(child - mean(child))~ I(parent - mean(parent)) - 1, data = galton)  
yhat <- predict(fit, type = 'response')   
mse(fit, yhat)

## [1] 5.000294

So, the beta that minimizes is 0.6463, which we approximated as 0.64 with our manipulation of the abline plots. And, the MSE is 5.000294, which we estimated as 5.0.

Now, we double check our approximations using R.

y <- galton$child  
x <- galton$parent  
beta1 <- cor(y, x) \* sd(y) / sd(x)   
beta0 <- mean (y) - beta1 \* mean(x)  
rbind (c(beta0, beta1), coef(lm(y ~ x)))

## (Intercept) x  
## [1,] 23.94153 0.6462906  
## [2,] 23.94153 0.6462906

# Reversing the outcome/predictor relationship

beta1 <- cor(y, x) \* sd(x) / sd(y)  
beta0 <- mean(x) - beta1 \* mean(y)  
rbind(c(beta0, beta1), coef(lm(x ~ y)))

## (Intercept) y  
## [1,] 46.13535 0.3256475  
## [2,] 46.13535 0.3256475

Regression through the origin yields an equivalent slope if you center the data first

yc <- y - mean(y)  
xc <- x - mean(x)   
beta1 <- sum(yc \* xc) / sum(xc ^ 2)   
c(beta1, coef(lm(y ~ x))[2])

## x   
## 0.6462906 0.6462906

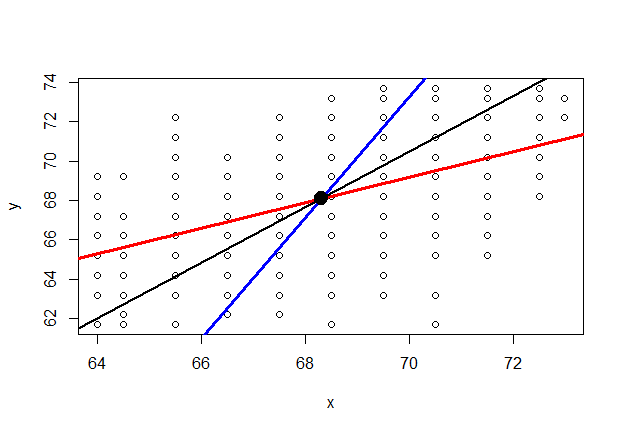
# Normalizing variables results in the slope being the correlation

yn <- (y - mean(y))/sd(y)  
xn <- (x - mean(x))/sd(x)  
c(cor(y, x), cor(yn, xn), coef(lm(yn ~ xn))[2])

## xn   
## 0.4587624 0.4587624 0.4587624

The code to add the lines, with the black line representing the best fit line

plot(x,y)  
abline(mean(y) - mean(x) \* cor(y, x) \* sd(y) / sd(x),  
 sd(y) / sd(x) \* cor(y, x), lwd = 3, col = "red")  
abline(mean(y) - mean(x) \* sd(y) / sd(x) / cor(y, x) ,  
 sd(y) / cor(y, x) / sd(x), lwd = 3, col = "blue")  
abline (mean(y) - mean(x) \* sd(y) / sd(x),  
 sd(y) / sd(x), lwd = 2)  
  
points(mean(x), mean(y), cex = 2, pch = 19)



### Interpreting regression coefficients, the slope

The parameter β\_1 is the expected change in response for a 1-unit change in the predictor.

# Residuals

Now, any time we perform regression on random samples, we introduce a rando error epsilon or ε, whereby our regression model is

$$Y\_i=β\_0+β\_1 X\_i+ε\_i$$

where ε\_i is follows a normal distribution N(0,σ^2).

Now, least squares minimize the sum of the squared residuals, that is least squares minimize∑\_(i=1)n▒e\_i2 , where e\_i are the observed errors. That it is the observed and the predicted outcome:

$$e\_i=Y\_i-Ŷ$$

So, is an estimate of .

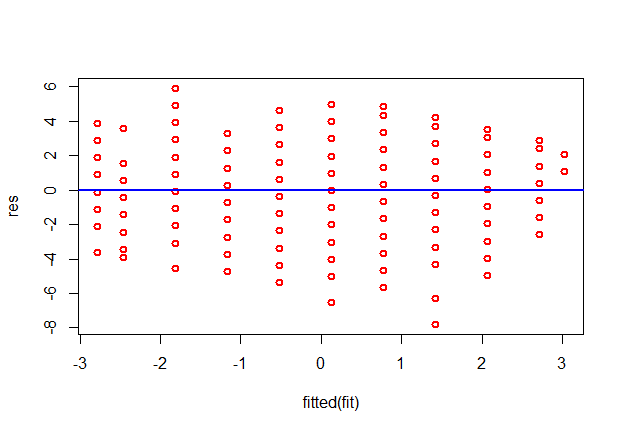
# To calculate the fitted model residuals, we use the resid() function.

res <- resid(fit)

Now, to produce the residual plot, we generate a scatterplot of the fitted model, using the fitted() function, versus the residuals. fitted() is a generic function which extracts fitted values from objects returned by modeling functions.

# Produce residual vs. fitted plot

plot(fitted(fit), res, lwd=2, col="red")  
# Add a horizontal line at 0   
abline(0, 0, lwd=2, col="blue")



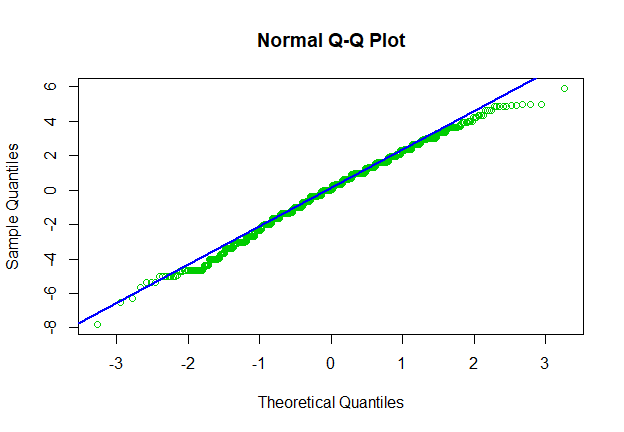
### Quantile-Quantile Plots

We can also produce a Q-Q plot, which is useful for determining if the residuals follow a normal distribution. If the data values in the plot fall along a roughly straight diagonal line, then the data is normally distributed.

The qqnorm() function the default method for producing a normal QQ plot of the values in y. The qqline() function adds a line to a “theoretical” (normal by default), quantile-quantile plot, which passes through the probability quantiles (first and third quartiles by default).

### Create Q-Q plot for residuals

qqnorm(res, col="green3")  
#Add a straight diagonal line to the plot  
qqline(data.frame(res), lwd=2, col="blue")



### Density Plot of the Residuals

We can also produce a density plot, which is also useful for visually checking whether or not the residuals are normally distributed. If the plot is roughly bell-shaped, then the residuals likely follow a normal distribution

### Create density plot of residuals

plot(density(res), lwd=2, col="purple")

