Exploration librairie swirl : Least Squares Estimation

> swirl()

Welcome to swirl! Please sign in. If you've been here before, use the same name as you did then. If you are new, call yourself something unique.

What shall I call you? ***jlbellier***

Would you like to continue with one of these lessons?

1: Regression Models Least Squares Estimation
2: No. Let me start something new.

Selection: ***2***

Please choose a course, or type 0 to exit swirl.

1: Regression Models
2: Statistical Inference
3: Take me to the swirl course repository!

Selection: ***1***

Please choose a lesson, or type 0 to return to course menu.

1: Introduction
2: Residuals
3: Least Squares Estimation
4: Residual Variation
5: Introduction to Multivariable Regression
6: MultiVar Examples
7: MultiVar Examples2
8: MultiVar Examples3
9: Residuals Diagnostics and Variation
10: Variance Inflation Factors
11: Overfitting and Underfitting
12: Binary Outcomes
13: Count Outcomes

Selection: ***3***

0%

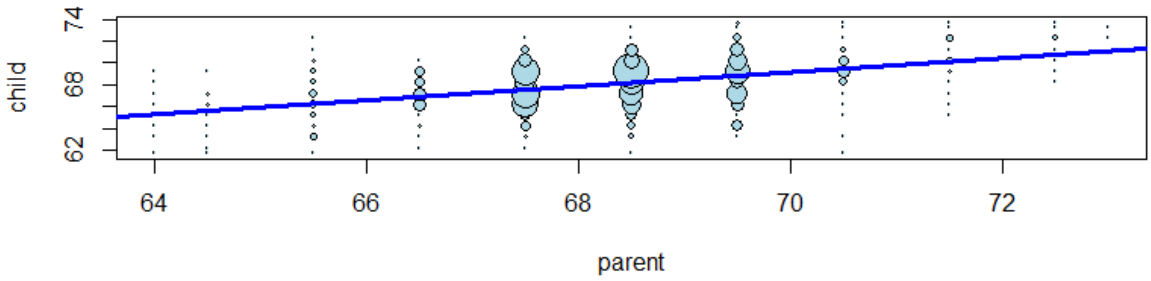
Least Squares Estimation. (Slides for this and other Data Science courses may be found at github https://github.com/DataScienceSpecialization/courses. If you care to use them, they must be downloaded as a zip file and viewed locally. This lesson corresponds to Regression_Models/01_03_ols. Galton data is from John Verzani's website, http://wiener.math.csi.cuny.edu/UsingR/)

==== ***5%***

In this lesson, if you're using RStudio, you'll be able to play with some of the code which appears in the slides. If you're not using RStudio, you can look at the code but you won't be able to experiment with the function "manipulate". We provide the code for you so you can examine it without having to type it all out. In RStudio, when the edit window displays code, make sure your flashing cursor is back in the console window before you hit "Enter" or any keyboard buttons, otherwise you might accidentally alter the code. If you do alter the file, in RStudio, you can hit Ctrl z in the editor until all the unwanted changes disappear. In other editors, you'll have to use whatever key combination performs "undo" to remove all your unwanted changes.

========= ***11%***

Here are the Galton data and the regression line seen in the Introduction. The regression line summarizes the relationship between parents' heights (the predictors) and their children's (the outcomes).



========== ***16%***

We learned in the last lesson that the regression line is the line through the data which has the minimum (least) squared "error", the vertical distance between the 928 actual children's heights and the heights predicted by the line. Squaring the distances ensures that data points above and below the line are treated the same. This method of choosing the 'best' regression line (or 'fitting' a line to the data) is known as ordinary least squares.

========== ***21%***

As shown in the slides, the regression line contains the point representing the means of the two sets of heights. These are shown by the thin horizontal and vertical lines. The intersection point is shown by the triangle on the plot. Its x-coordinate is the mean of the parents' heights and y-coordinate is the mean of the childrens' heights.

========== ***26%***

As shown in the slides, the slope of the regression line is the correlation between the two sets of heights multiplied by the ratio of the standard deviations (childrens' to parents' or outcomes to predictors).

========== ***32%***

Here we show code which demonstrates how changing the slope of the regression line effects the mean squared error between actual and predicted values. Look it over to see how straightforward it is.

```
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)
  points(0, 0, cex = 2, pch = 19)
  mse <- mean( (y - beta * x)^2 )
  title(paste("beta = ", beta, "mse = ", round(mse, 3)))
}
manipulate(myPlot(beta), beta = slider(0.4, .8, step = 0.02))
```

========== 37%

What RStudio graphics package allows the user to play with the data to see the effects of the changes?

1: plot
2: points
3: abline
4: manipulate

Selection: 4

That's a job well done!

========== 42%

Now you can actually play with the code to use R's manipulate function and find the minimum squared error. You can adjust the slider with the left mouse button or use the right and left arrow keys to see how changing the slope (beta) affects the mean squared error (mse). If the slider disappears you can call it back by clicking on the little gear in the upper left corner of the plot window.

========== 47%

Which value of the slope minimizes the mean squared error?

1: 5
2: .44
3: .64
4: .70

Selection: 3

You nailed it! Good job!

========== 53%

What was the minimum mse?

1: .64
2: 5.0

3: .66
4: 44

*Selection: **2***

You're the best!

========== ***58%***

Recall that you normalize data by subtracting its mean and dividing by its standard deviation. We've done this for the galton child and parent data for you. We've stored these normalized values in two vectors, gpa_nor and gch_nor, the normalized galton parent and child data.

========== ***63%***

Use R's function "cor" to compute the correlation between these normalized data sets.

*> **cor(gpa_nor, gch_nor)***
[1] 0.4587624

Perseverance, that's the answer.

========== ***68%***

How does this correlation relate to the correlation of the unnormalized data?

1: It is bigger.
2: It is smaller.
3: It is the same.

*Selection: **3***

Nice work!

========== ***74%***

Use R's function "lm" to generate the regression line using this normalized data. Store it in a variable called l_nor. Use the parents' heights as the predictors (independent variable) and the childrens' as the predicted (dependent). Remember, 'lm' needs a formula of the form dependent ~ independent. Since we've created the data vectors for you there's no need to provide a second "data" argument as you have previously.

> l_nor <- lm(gch_nor,gpa_nor)
Error in formula.default(object, env = baseenv()) : 'formula' incorrecte
*> **l_nor <- lm(gch_nor~gpa_nor)***

Keep up the great work!

========== ***79%***

What is the slope of this line?

1: 1.
2: I have no idea
3: The correlation of the 2 data sets

*Selection: **3***

You are really on a roll!

========== 84%

If you swapped the outcome (Y) and predictor (X) of your original (unnormalized) data, (for example, used childrens' heights to predict their parents), what would the slope of the new regression line be?

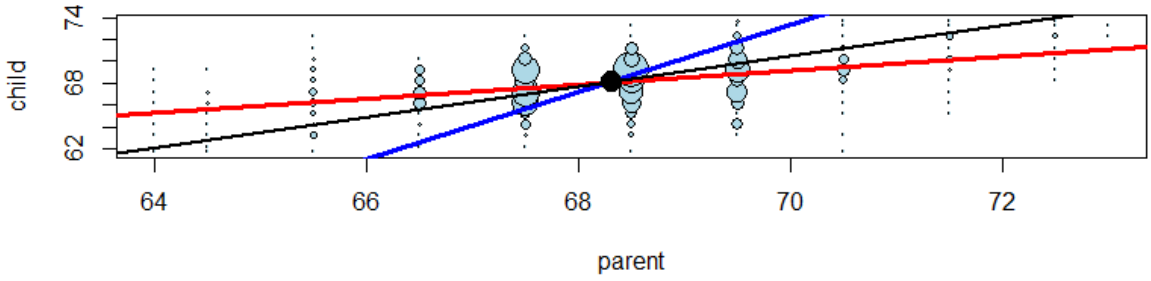
*1: correlation(X,Y) * sd(X)/sd(Y)*
2: 1.
3: I have no idea
4: the same as the original

*Selection: **1***

You nailed it! Good job!

========== 89%

We'll close with a final display of source code from the slides. It plots the galton data with three regression lines, the original in red with the children as the outcome, a new blue line with the parents' as outcome and childrens' as predictor, and a black line with the slope scaled so it equals the ratio of the standard deviations.



========== 95%

Congrats! You've concluded this lesson on ordinary least squares which are truly extraordinary!

========== 100%