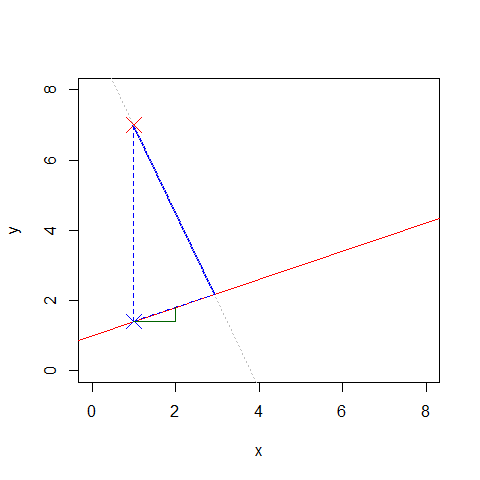
Calculating the Distance from a Point to a Line

Bob Horton

Thursday, March 10, 2016

This is a "literate calculation" document describing how to compute the distance from a point described by x and y coordinates to a line described in the form

Consider the figure below. We are given a red line, described in terms of its slope and y-intercept. We are also given the x and y coordinates of a point, shown as a red "X". Our task is to find the perpendicular distance from the point to the line.



Let's start with something easier; the vertical distance from the point to the line is just the difference between the y-coordinate of the point and the y-value on the line at that same x-value. This is shown as a dashed blue line, whose length is given by

The solid blue line shows the distance we really want, but this perpendicular distance is proportional to the vertical distance, with the ratio between them determined by the triangle outlined in blue.

The small green triangle has a horizontal side starting from the point where our dashed blue vertical line meets the red line. Convince yourself that this green triangle is geometrically similar to the blue triangle, that is, the angles and the ratios of its sides are the same in both the green and the blue triangle.

The horizintal side of the green triangle was drawn to have a length of 1, so its vertical side has a length equal to m, the slope of the original red line. Using the Pythagorian theorem and the fact that 1^2 equals 1, we find that the hypotenuse of the green triangle has a length of

Finally, our equation for the perpendicular distance from the point to the line comes from taking a proportion of the vertical distance, based on the ratio of the sides of the triangle:

For the values in this example, this works out to a distance of 5.2 units.

Here is the R code to draw the figure:

# Define the line by slope and intercept, and the point  
m <- 2/5  
b <- 1  
  
P <- c(x=1, y=7)  
  
# Convenience functions to work with points represented   
# by two element vectors containing x and y values  
plot\_point <- function(point, pch=4, cex=2, asp=1, ...)  
 points(point['x'], point['y'], pch=pch, cex=cex, ...)  
  
connect\_points <- function(pt1, pt2, ...)  
 lines(c(pt1[['x']], pt2[['x']]), c(pt1[['y']], pt2[['y']]), ...)  
  
# Create blank plot with x and y axis ranges set.  
# Make the x and y ranges the same so the aspect ratio is 1.  
xy\_range <- c(0, 8)  
plot(x=xy\_range, y=xy\_range, xlab='x', ylab='y', type='n')  
  
abline(b, m, col="red")  
  
plot\_point(P, col="red")  
  
P1 <- c(x=P[['x']], y=m \* P[['x']] + b)  
connect\_points(P, P1, lty=2, col="blue")  
  
plot\_point(P1, col="blue")  
  
m2 <- -1/m  
b2 <- P[['y']] - P[['x']] \* m2  
  
  
x2 <- (b2 - b)/(m - m2)  
P2 <- c(x=x2, y=P1[['y']] + m \* (x2 - P[['x']]))  
connect\_points(P, P2, col="blue", lwd=2)  
abline(b2, m2, lty=3, col="gray")  
  
connect\_points(P1, P2, col="blue", lty=2)  
  
# Draw lower triangle  
P3 <- c(x=P[['x']] + 1, y=P1[['y']])  
P4 <- c(x=P[['x']] + 1, y=P1[['y']] + m)  
connect\_points(P1, P3, col="darkgreen")  
connect\_points(P3, P4, col="darkgreen")

This equation is used in my blog post entitled "[Fun with Simpson's Paradox](http://blog.revolutionanalytics.com/2015/11/fun-with-simpsons-paradox-simulating-confounders.html)". See also the Wikipedia article on "[distance from a point to a line](https://en.wikipedia.org/wiki/Distance_from_a_point_to_a_line)".