Fitting Lines with Error in Both Variables

Ian Hunter, MIT, 1 March 2018

Consider a data set consisting of n values of x_i and y_i where i = 1 .. n. The linear relation (or model) predicting the y values from the x values is:

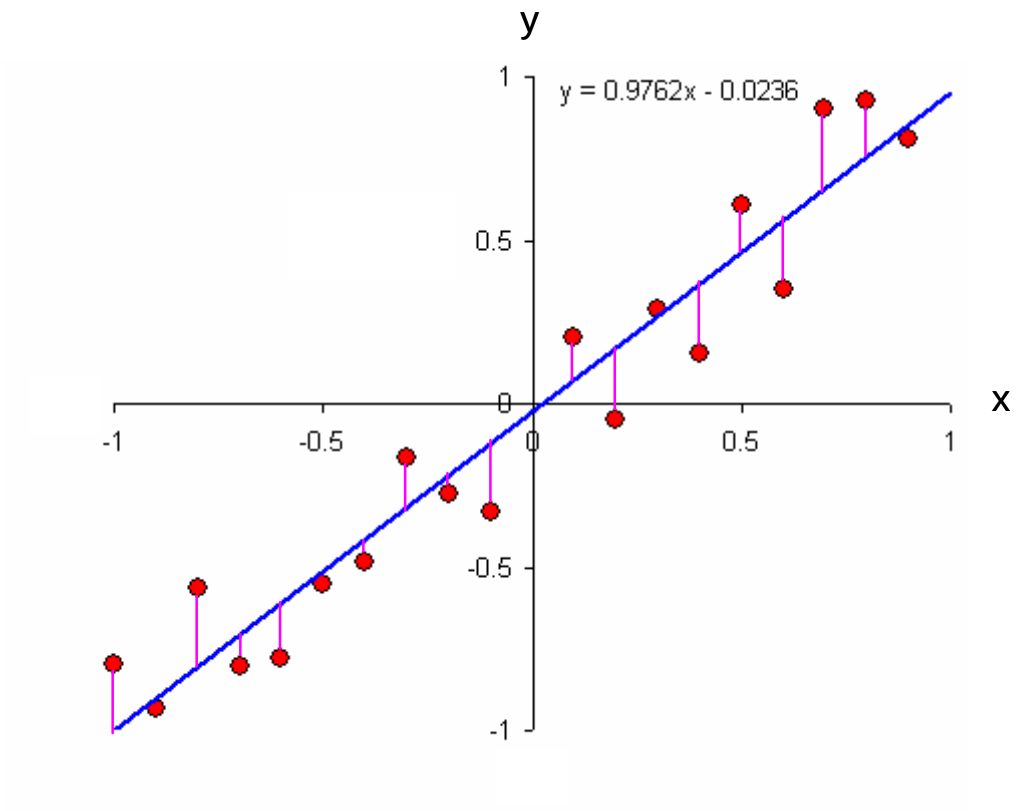
$$y = a_1.x + a_0$$

This function is sometimes called a first-order polynomial or simply the equation of a straight line. The linear model has two free parameters, a_1 (slope) and a_0 (intercept).

Minimizing the Sum of Squared Vertical Deviations

Fitting such a straight line to a x,y data set might seem like a straight forward proposition. The usual line fitting functions found in Mathcad, MatLab, Excel and hand-held calculators estimate the two parameters of the line (slope and intercept) by minimizing the sum of the squared **vertical** deviations between the line and the data points. As mentioned in other notes the "sum of squares" function to be minimized is called the objective function.

The plot below (obtained using Excel) shows an example of the "best" fit line (blue) obtained by minimizing the sum of the squared vertical deviations (purple) between the line and the data points (red).



It can be proved (see Straight Line Fitting notes) that the minimum possible value of the sum of the squared vertical deviations occurs when the slope (a_1) and intercept (a_0) are determined using the following parameter estimators.

$$\text{Parameters}_{vert}(x, y) := \left| \begin{array}{l} n \leftarrow \text{length}(x) - 1 \\ S_x \leftarrow \sum_{i=1}^{n} x_i \\ S_y \leftarrow \sum_{i=1}^{n} y_i \\ S_{xx} \leftarrow \sum_{i=1}^{n} \left(x_i \cdot x_i\right) \\ S_{xy} \leftarrow \sum_{i=1}^{n} \left(x_i \cdot y_i\right) \\ \text{Slope} \leftarrow \dfrac{n \cdot S_{xy} - S_x \cdot S_y}{n \cdot S_{xx} - S_x \cdot S_x} \\ \text{Intercept} \leftarrow \dfrac{S_y \cdot S_{xx} - S_x \cdot S_{xy}}{n \cdot S_{xx} - S_x \cdot S_x} \\ \text{return } (\text{Slope} \quad \text{Intercept}) \end{array} \right.$$

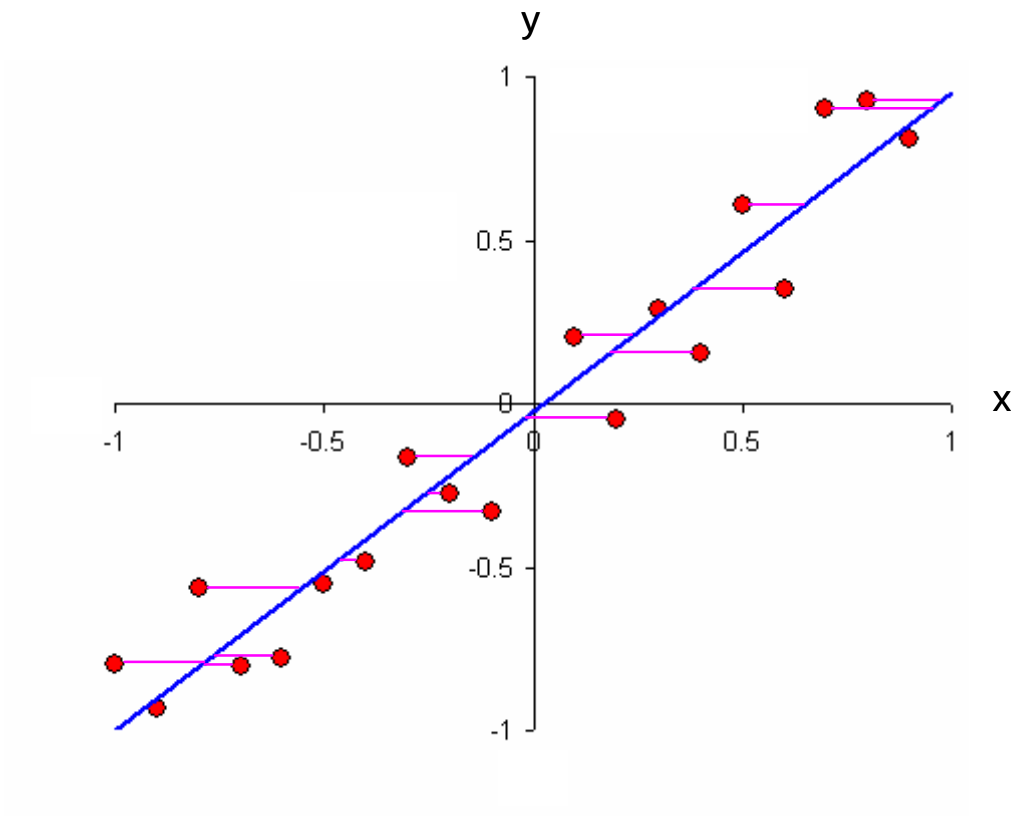
the data sets are x_i and y_i where $i = 1 .. n$

As mentioned in other notes these slope and intercept estimators can exhibit numerical problems under some conditions and may be replaced by better (and sometimes much more complex) estimators (some of which employ a beautiful algorithm called singular value decomposition).

Minimizing the sum of the squared vertical differences between the line and the data points is a good strategy when the y data are corrupted with additive Gaussian white noise (with a constant standard deviation) and the x data are noise free. This is the typical situation when the x data are values of the independent variable which is manipulated experimentally and the y data are values of the measured dependent variable which often may be considered to have additive measurement error (or some other source of additive noise) which is Gaussian and uncorrelated (white).

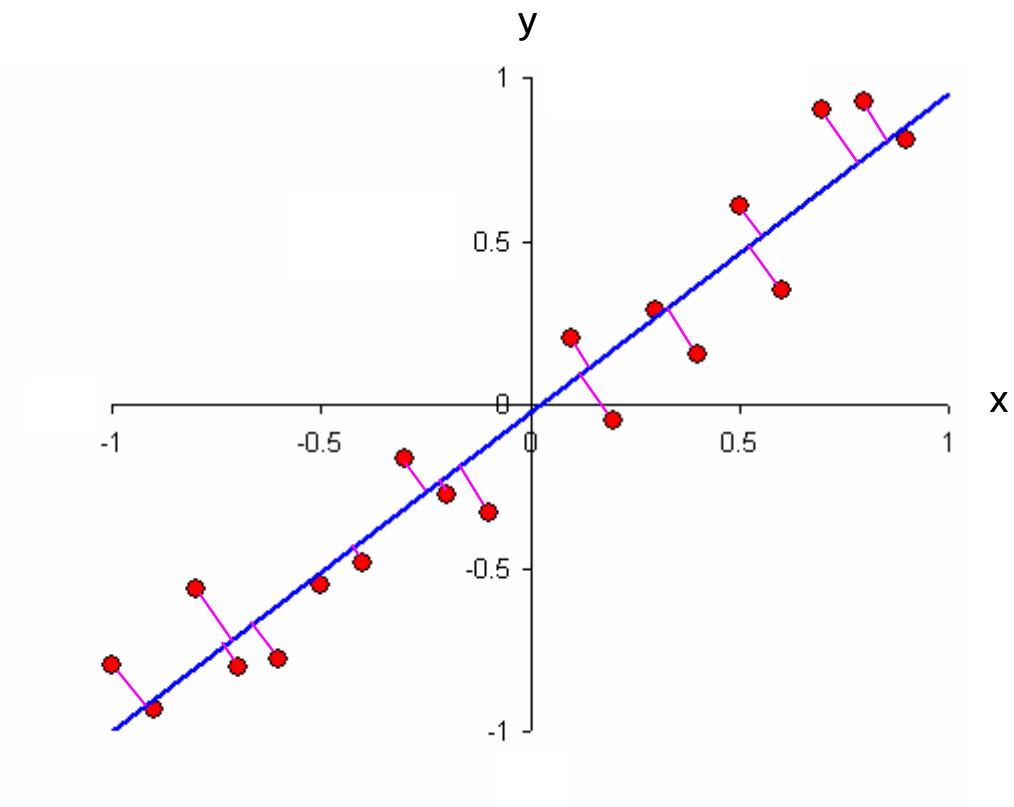
Minimizing the Sum of Squared Horizontal Deviations

If the y data are noise free and the x data are assumed to have been corrupted by additive Gaussian white noise (having constant standard deviation) then a good strategy is to fit a line by minimizing the sum of squared **horizontal** deviations between the data points and the line. The plot below shows this case. However this is not a typical situation and under most circumstances you would simply exchange data so that the noise free data were plotted on the x-axis. Once exchanged the data now conform to the usual case detailed above.



Minimizing the Sum of Squared Perpendicular Deviations

However if the x and y data both may be considered to be corrupted with additive Gaussian white noise then a better strategy is to minimize the sum of the squared distances between the line and the data points. If the standard deviations of the noise in both the x and y data are approximately equal then the distance to be minimized is best defined as the perpendicular (i.e. shortest) distance from the line to the data points as shown in the plot below (note that maximum likelihood theory gives a firm theoretical foundation for deriving optimal line (and more generally function and model) fitting techniques).



I am not aware of any predefined methods in common data analysis packages such as Mathcad, MatLab and Excel to determine the line for which the sum of the squared perpendicular deviations is minimized. It is possible to use general purpose iterative minimization techniques (such as Levenberg-Marquardt, steepest decent, etc.) to find the parameters of such a line. However in the usual case of minimizing the sum of squared vertical deviations iterative techniques are not required as direct analytic equations for the slope and intercept are readily available as shown above (the straight line "trend" fitting capability in Excel uses them, as do most calculators, Matlab. Mathcad, etc.). Such direct (explicit, non-iterative) equations for estimating the slope and intercept of the line which minimizes the sum of the squared perpendicular deviations are also available (though rarely presented) and may be written

$\text{Parameters}_{perp}(x, y) :=$

$n \leftarrow \text{length}(x) - 1$

$\mu_x \leftarrow \frac{1}{n} \cdot \sum_{i=1}^{n} x_i$

$\mu_y \leftarrow \frac{1}{n} \cdot \sum_{i=1}^{n} y_i$

$S_{xx} \leftarrow \sum_{i=1}^{n} \left(x_i - \mu_x\right)^2$

$S_{yy} \leftarrow \sum_{i=1}^{n} \left(y_i - \mu_y\right)^2$

$S_{xy} \leftarrow \sum_{i=1}^{n} \left[\left(x_i - \mu_x\right) \cdot \left(y_i - \mu_y\right)\right]$

$\text{Slope} \leftarrow \frac{2 \cdot S_{xy}}{S_{xx} - S_{yy} + \sqrt{\left(S_{xx} - S_{yy}\right)^2 + 4 \cdot S_{xy}^2}}$

$\text{Intercept} \leftarrow \frac{2 \cdot S_{xy}}{S_{yy} - S_{xx} - \sqrt{\left(S_{xx} - S_{yy}\right)^2 + 4 \cdot S_{xy}^2}} \cdot \mu_x + \mu_y$

return (Slope Intercept)

where x_i and y_i have i = 1 .. n

Example of Linearly Related Data in Which Both x and y "Measurements" are Corrupted by Additive Gaussian White Noise

In order to illustrate the difference between line fitting using vertical and perpendicular minimization consider the following example.

We start by generating a data set in which y is a linear function of x.

$n := 101$ $\quad i := 1 .. n$

$x_i := 0.1 \cdot (i - 1)$

We set the slope and intercept to be $\quad Slope := 1.0 \quad Intercept := 0.0$

$y_i := Slope \cdot x_i + Intercept$

We now generate two different Gaussian white noise values

$$e_{x_i} := 0.8 \cdot \sum_{j=1}^{12} (rnd(1) - 0.5) \qquad e_{y_i} := 0.8 \cdot \sum_{j=1}^{12} (rnd(1) - 0.5)$$

And then add them to x and y $\quad x_i := x_i + e_{x_i}$ $\quad$ and $\quad y_i := y_i + e_{y_i}$

We then pretend that we do not know the values of the slope and intercept and proceed to estimate them via the two different estimation techniques.

$\left(slope_{vert} \quad intercept_{vert}\right) := Parameters_{vert}(x, y)$

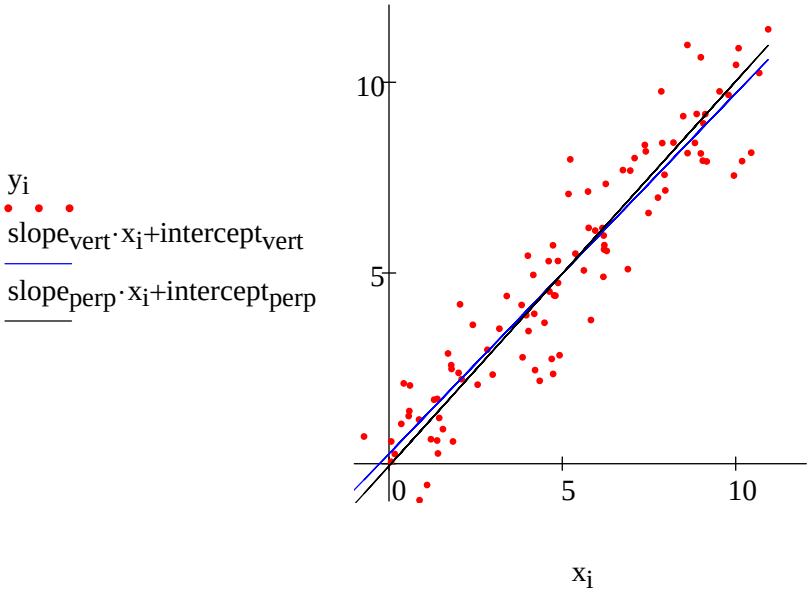
$slope_{vert} = 0.947$

$intercept_{vert} = 0.239$

$\left(slope_{perp} \quad intercept_{perp}\right) := Parameters_{perp}(x, y)$

$slope_{perp} = 1.011$

$intercept_{perp} = -0.08$



Clearly the perpendicular estimators are closer to the original slope and intercept values (1.0 and 0.0 respectively).

Simulation of the Difference Between the Vertical and Perpendicular Estimators

We now simulate m repeated experiments to evaluate whether there is any substantial difference between the estimates of slope and intercept using the two estimation methods.

$$\begin{aligned}
&\text{Simulate}(m) := \\
&\quad \text{Slope} \leftarrow 1.0 \\
&\quad \text{Intercept} \leftarrow 0.0 \\
&\quad \text{for } j \in 1 .. m \\
&\quad\quad \text{for } i \in 1 .. 101 \\
&\quad\quad\quad x_i \leftarrow 0.1 \cdot (i - 1) \\
&\quad\quad\quad y_i \leftarrow \text{Slope} \cdot x_i + \text{Intercept} \\
&\quad\quad\quad e_{x_i} \leftarrow 0.8 \cdot \sum_{j=1}^{12} (\text{rnd}(1) - 0.5) \\
&\quad\quad\quad e_{y_i} \leftarrow 0.8 \cdot \sum_{j=1}^{12} (\text{rnd}(1) - 0.5) \\
&\quad\quad\quad x_i \leftarrow x_i + e_{x_i} \\
&\quad\quad\quad y_i \leftarrow y_i + e_{y_i} \\
&\quad\quad \left(\text{slope}_{\text{vert}_j} \quad \text{intercept}_{\text{vert}_j} \right) \leftarrow \text{Parameters}_{\text{vert}}(x, y) \\
&\quad\quad \left(\text{slope}_{\text{perp}_j} \quad \text{intercept}_{\text{perp}_j} \right) \leftarrow \text{Parameters}_{\text{perp}}(x, y) \\
&\quad \text{return } \left(\text{slope}_{\text{vert}} \quad \text{intercept}_{\text{vert}} \quad \text{slope}_{\text{perp}} \quad \text{intercept}_{\text{perp}} \right)
\end{aligned}$$

where

$\text{slope}_{\text{vert}}$ stores the m estimates of the slope obtained using the vertical deviation based objective function, and

$\text{intercept}_{\text{vert}}$ stores the m estimates of the intercept obtained using the vertical deviation based objective function, and

$\text{slope}_{\text{perp}}$ stores the m estimates of the slope obtained using the perpendicular deviation based objective function, and

$\text{intercept}_{\text{perp}}$ stores the m estimates of the intercept obtained using the perpendicular deviation based objective function.

We now run m simulations and save the resulting m estimates of slope and intercept.

$m := 1000 \qquad j := 1 .. m$

$$\left(\text{slope}_{\text{vert}} \quad \text{intercept}_{\text{vert}} \quad \text{slope}_{\text{perp}} \quad \text{intercept}_{\text{perp}} \right) := \text{Simulate}(m)$$

Analysis of the Slope Estimates

We now calculate the mean of the slope estimates

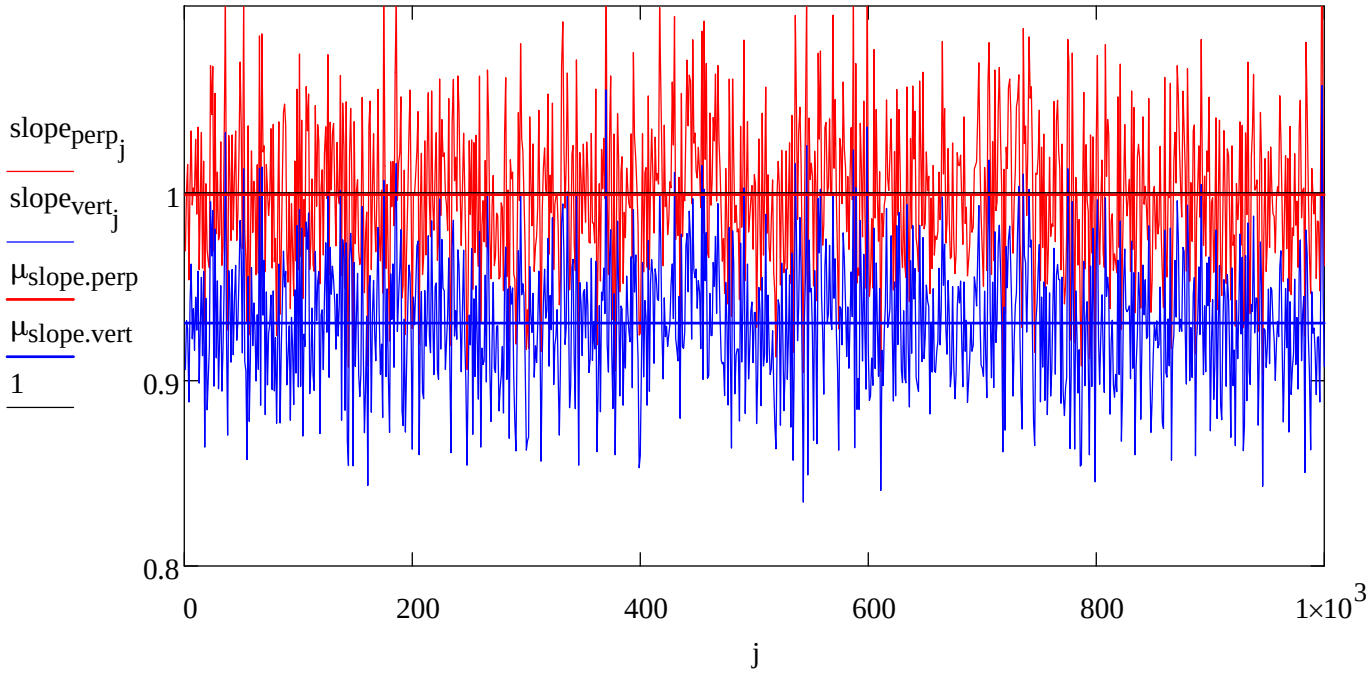
$$\mu_{slope.vert} := \frac{1}{m} \cdot \sum_{j=1}^{m} slope_{vert_j} \qquad \mu_{slope.vert} = 0.93 \qquad \text{vertical deviations}$$

$$\mu_{slope.perp} := \frac{1}{m} \cdot \sum_{j=1}^{m} slope_{perp_j} \qquad \mu_{slope.perp} = 0.999 \qquad \text{perpendicular deviations}$$

and the standard deviation of the slope estimates

$$\sigma_{slope.vert} := \sqrt{\frac{1}{m} \cdot \sum_{j=1}^{m} \left(slope_{vert_j} - \mu_{slope.vert}\right)^2} \qquad \sigma_{slope.vert} = 0.036 \qquad \text{vertical deviations}$$

$$\sigma_{slope.perp} := \sqrt{\frac{1}{m} \cdot \sum_{j=1}^{m} \left(slope_{perp_j} - \mu_{slope.perp}\right)^2} \qquad \sigma_{slope.perp} = 0.041 \qquad \text{perpendicular deviations}$$



Notice that the perpendicular deviation based estimates of the slope (red) vary about a mean of almost exactly the true value of 1.0 (such an estimator is called an **unbiased estimator**) . In contrast note that the vertical deviation estimates of the slope (blue) are systematically different from the "true" value of 1.0 (such an estimator is called a **biased estimator**).

The standard deviations of the slope parameter estimates are similar.

Probability Density and Distribution Functions of the Slope Estimates

It is of interest to ask how these estimates are distributed. Are they typically distributed according to a uniform, Gaussian, Laplacian, exponential, etc. probability density function? It turns out that for additive Gaussian white noise measurement (or otherwise) noise the parameter estimates are typically Gaussian (and incidentally white). The Gaussian nature of the parameter estimates is shown below where the computed probability density function are shown together with Gaussian density functions having the same mean (μ) and standard deviations (σ). We compute the probability density as follows

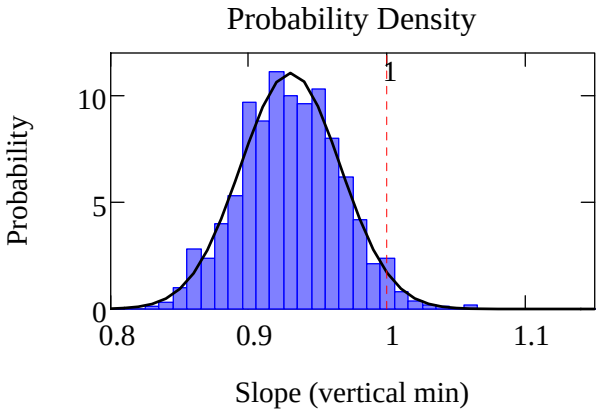
$$P(z,\alpha,\Delta) := \frac{1}{\Delta \cdot m} \cdot \sum_{i=1}^{m} \text{if}\left[\left(\alpha - \frac{\Delta}{2}\right) < z_i \le \left(\alpha + \frac{\Delta}{2}\right), 1, 0\right]$$

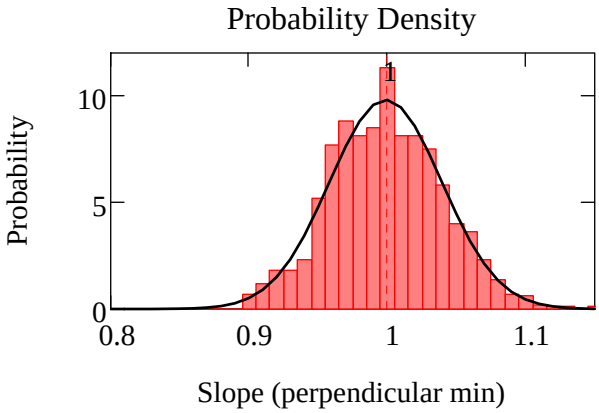
where z are the m values (slopes or intercept estimates in this case), and Δ is the bin width. The probability density is determined at the signal amplitude, α, plus and minus $\Delta/2$.

The theoretical Gaussian probability density is

$$P_{Gauss}(z,\mu,\sigma) := \frac{1}{\sigma \cdot \sqrt{2 \cdot \pi}} \cdot e^{-\frac{(z-\mu)^2}{2 \cdot \sigma^2}}$$

$\alpha := 0.8, 0.81 .. 1.15$



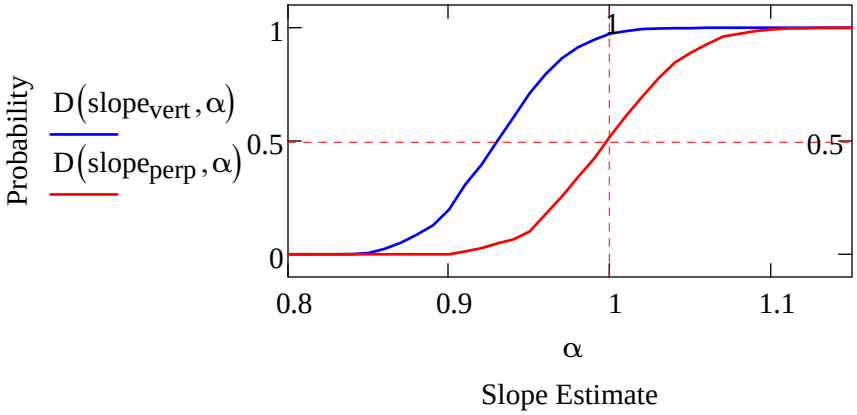


Notice that the estimates are approximately Gaussian.

We compute the probability distribution as

$$D(z,\alpha) := \frac{1}{m} \cdot \sum_{j=1}^{m} \text{if}\left(z_j \le \alpha, 1, 0\right)$$

$\alpha := 0.8, 0.81 .. 1.15$



It is very clear from both the density and distribution functions that the vertical deviation objective function based slope estimates are biased (blue) and the perpendicular deviation objective function based slope estimates (red) are unbiased.

Analysis of the Intercept Estimates

We now calculate the mean of the intercept estimates

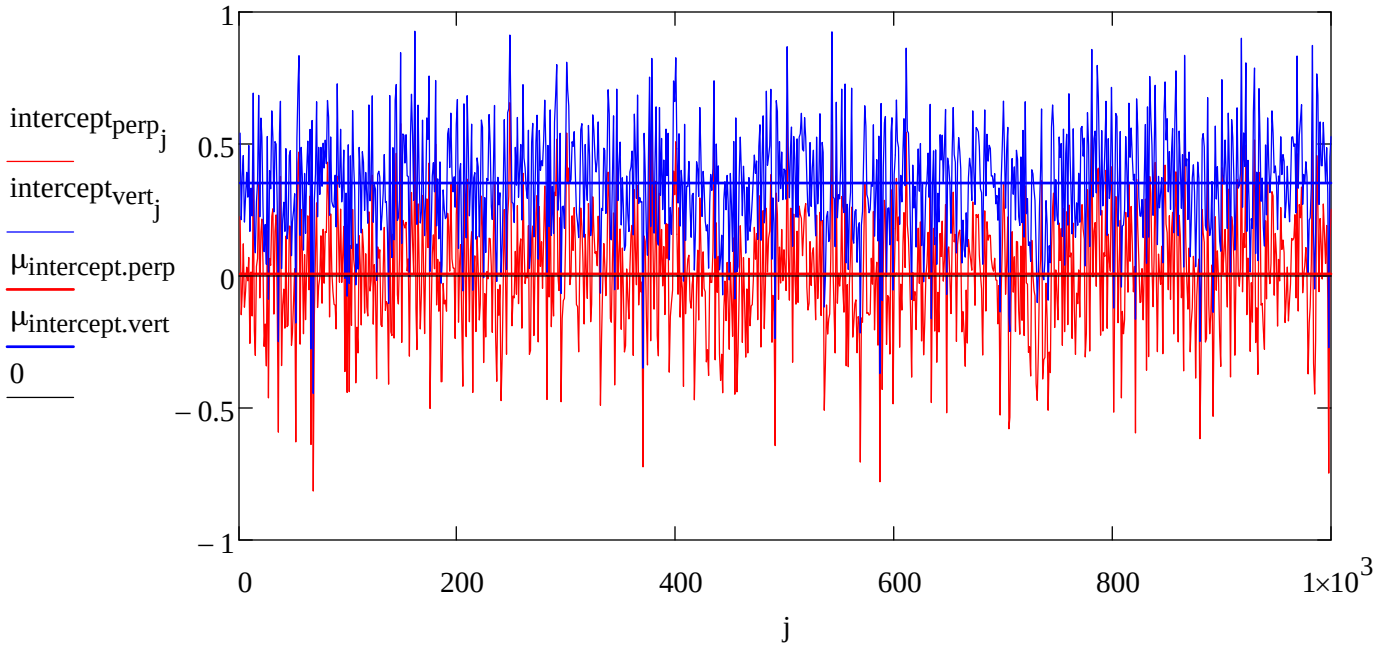
$$\mu_{intercept.vert} := \frac{1}{m} \cdot \sum_{j=1}^{m} intercept_{vert_j} \qquad \mu_{intercept.vert} = 0.352 \qquad \text{vertical deviations}$$

$$\mu_{intercept.perp} := \frac{1}{m} \cdot \sum_{j=1}^{m} intercept_{perp_j} \qquad \mu_{intercept.perp} = 0.008 \qquad \text{perpendicular deviations}$$

and the standard deviation of the intercept estimates

$$\sigma_{intercept.vert} := \sqrt{\frac{1}{m} \cdot \sum_{j=1}^{m} \left(intercept_{vert_j} - \mu_{intercept.vert}\right)^2} \qquad \sigma_{intercept.vert} = 0.212 \qquad \text{vertical deviations}$$

$$\sigma_{intercept.perp} := \sqrt{\frac{1}{m} \cdot \sum_{j=1}^{m} \left(intercept_{perp_j} - \mu_{intercept.perp}\right)^2} \qquad \sigma_{intercept.perp} = 0.233 \qquad \text{perpendicular deviations}$$

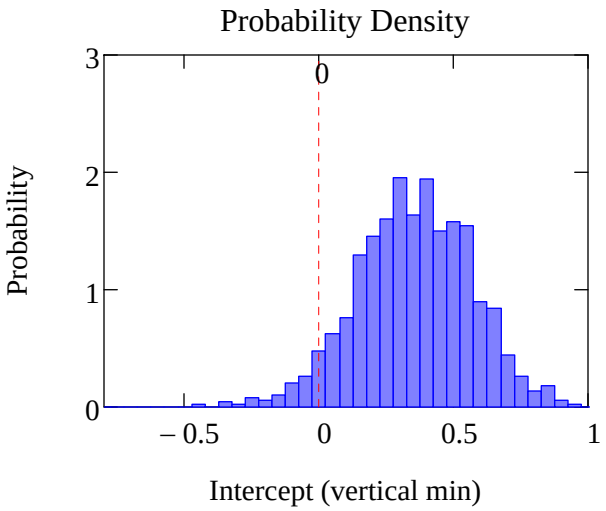


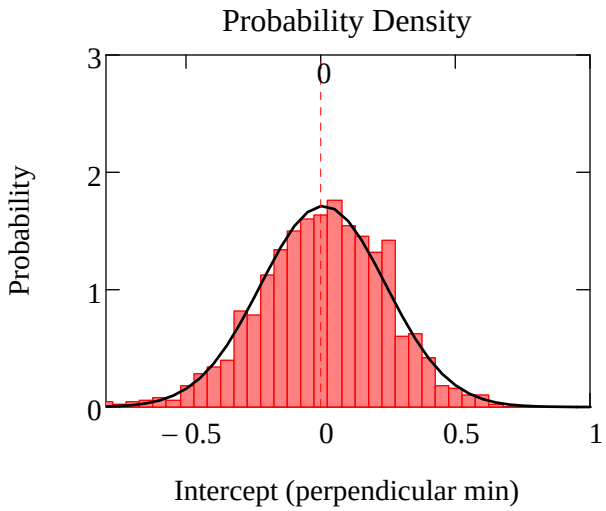
Notice that the perpendicular deviation based estimates of the intercept (red) vary about a mean of almost exactly the true value of 0.0 (such an estimator is called an **unbiased estimator**) . In contrast note that the vertical deviation estimates of the intercept (blue) are systematically different from the "true" value of 0.0 (such an estimator is called a **biased estimator**).

The standard deviations of the intercept parameter estimates are similar.

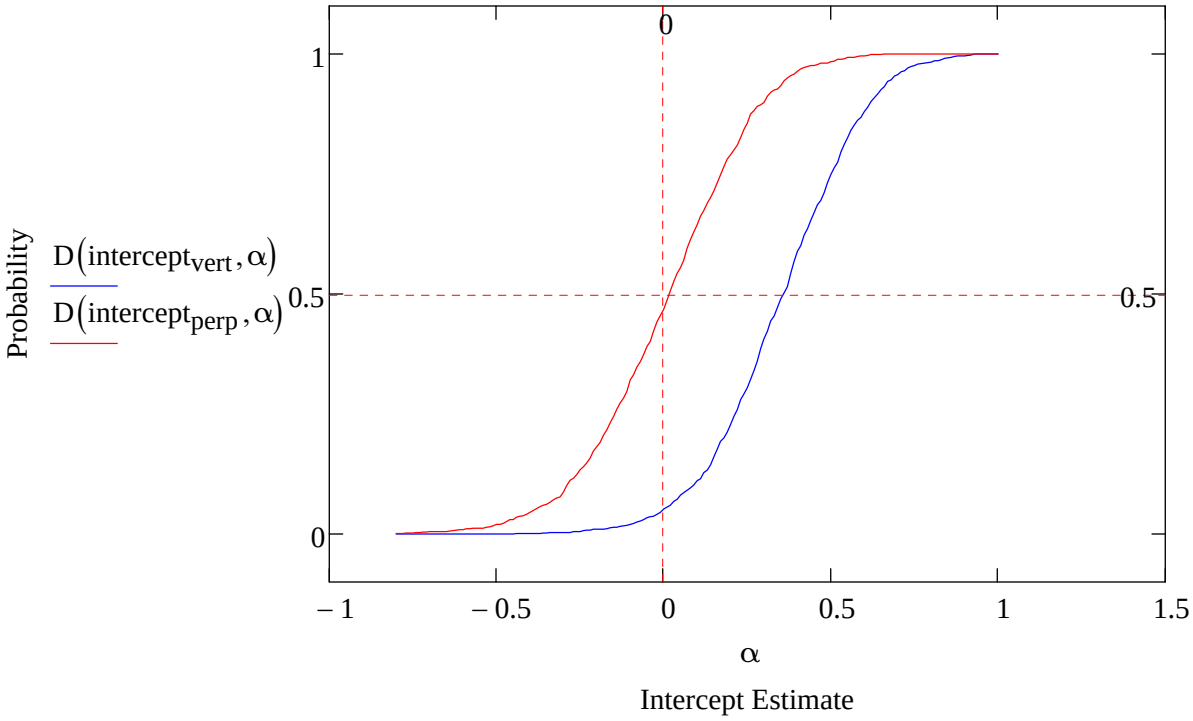
Probability Density and Distribution Functions of the Intercept Estimates

$\alpha := -0.8, -0.75 .. 1.0$





$\alpha := -0.8, -0.79 .. 1.0$



Once again it is very clear from both the density and distribution functions that the vertical deviation objective function based intercept estimates are biased (blue) and the perpendicular deviation objective function based intercept estimates (red) are unbiased.