Physics 509: Bootstrap and Robust Parameter Estimation

Scott Oser
Lecture #20

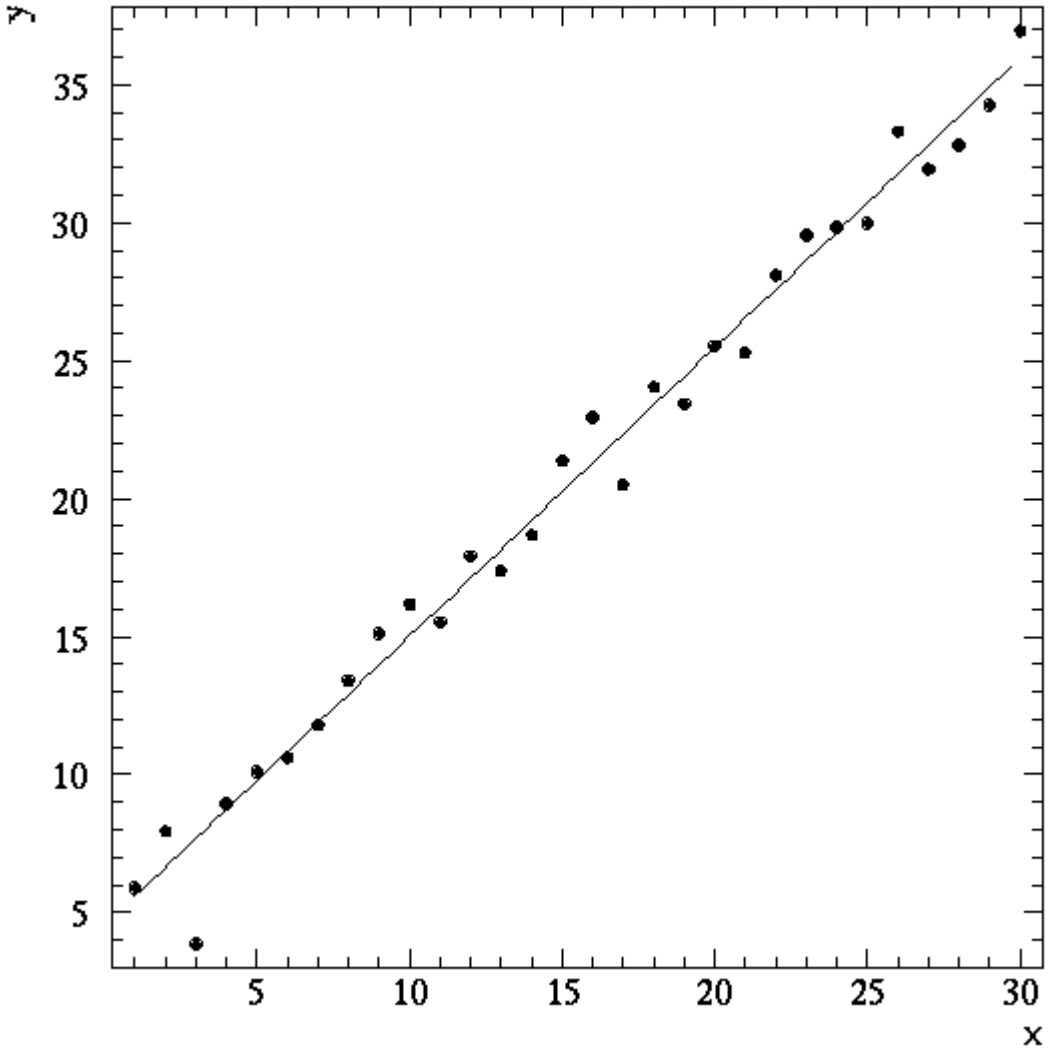
Nonparametric parameter estimation

Question: what error estimate should you assign to the slope and intercept from this fit?

You are not given the error bars.

You are not told the distribution of the errors.

In fact, all you are told is that all residuals are independent and identically distributed.



Nonparametric parameter estimation

This sounds like an impossible problem. For either an ML estimator or a Bayesian solution to the problem we need to be able to write down the likelihood function:

$$L = \prod_{i=1}^{N} f\left(y_i - y\left(x_i\right)\right)$$

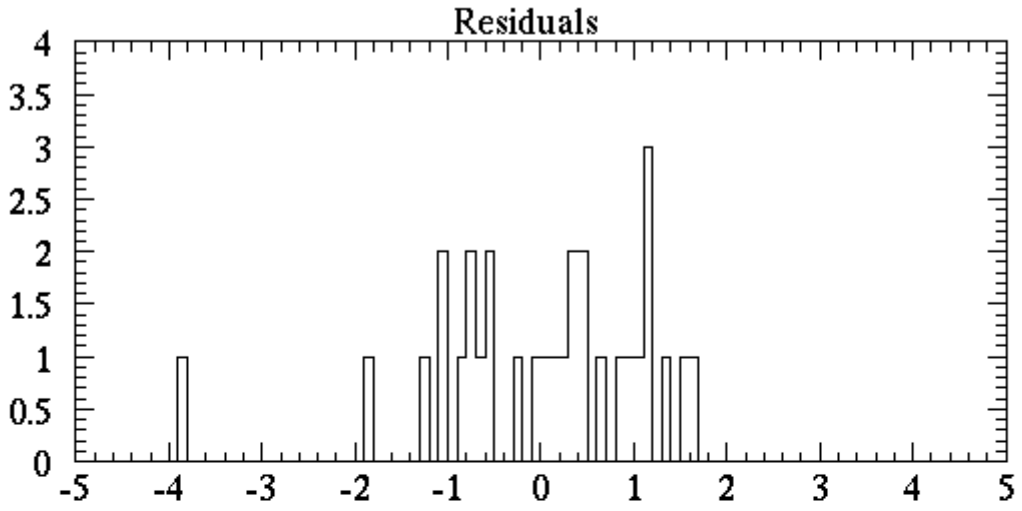
Here f(Δ) is the distribution of the residuals between the data and the model. If for example f(Δ) is a Gaussian, then the ML estimators becomes the least-squares estimator.

If you don't know f(Δ) then you're seemingly screwed.

Bootstrap

The bootstrap method is an attempt to calculate the distributions of the errors from the data itself, and to use these to calculate the errors on the fit.

After all, the data contains a lot of information about the errors:



Description of the bootstrap

It's a very simple technique: You start with a set of N independent and identically distributed observations, and calculate some estimator $\theta(x_1...x_N)$ of the data. To get the error on θ do the following:

1) Make a new data set by selecting N random observations from the data, *with replacement*. Some data points will get selected more than once, others not at all. Call this new data set X'.
2) Calculate $\theta(X')$.
3) Repeat the procedure many times (at least 100).

The width of the distribution of the θ calculated from the resampled data sets gives you your error on θ.

Effectively you use your own data to make “Monte Carlo” data sets.

Justification for the bootstrap

This sounds like cheating, and it has to be conceded that the procedure doesn't always work. But it has some intuitiveness.

To calculate the real statistical error on θ you'd need to know the true distribution F(x) that the data is drawn through.

Given that you don't know the form of F(x), you could try to estimate it with its nonparametric maximum likelihood estimator. This of course is just the observed distribution of x. You basically assume that your observed distribution of x is a fair measure of the true distribution F(x) and can be used to generate “Monte Carlo” data sets.

Obviously this is going to work better when N is large, since the better your estimate of F(x) the more accurate your results are going to be.

Advantages of the Bootstrap

The bootstrap has a number of advantages:

1) Like Monte Carlo, you can use it to estimate errors on parameters that depend in a complicated way on the data.
2) You can use it even when you don't know the true underlying error distribution. It's “nonparametric” in this sense.
3) You don't have to generate zillions of Monte Carlo data sets to use it---it simply uses the data itself.

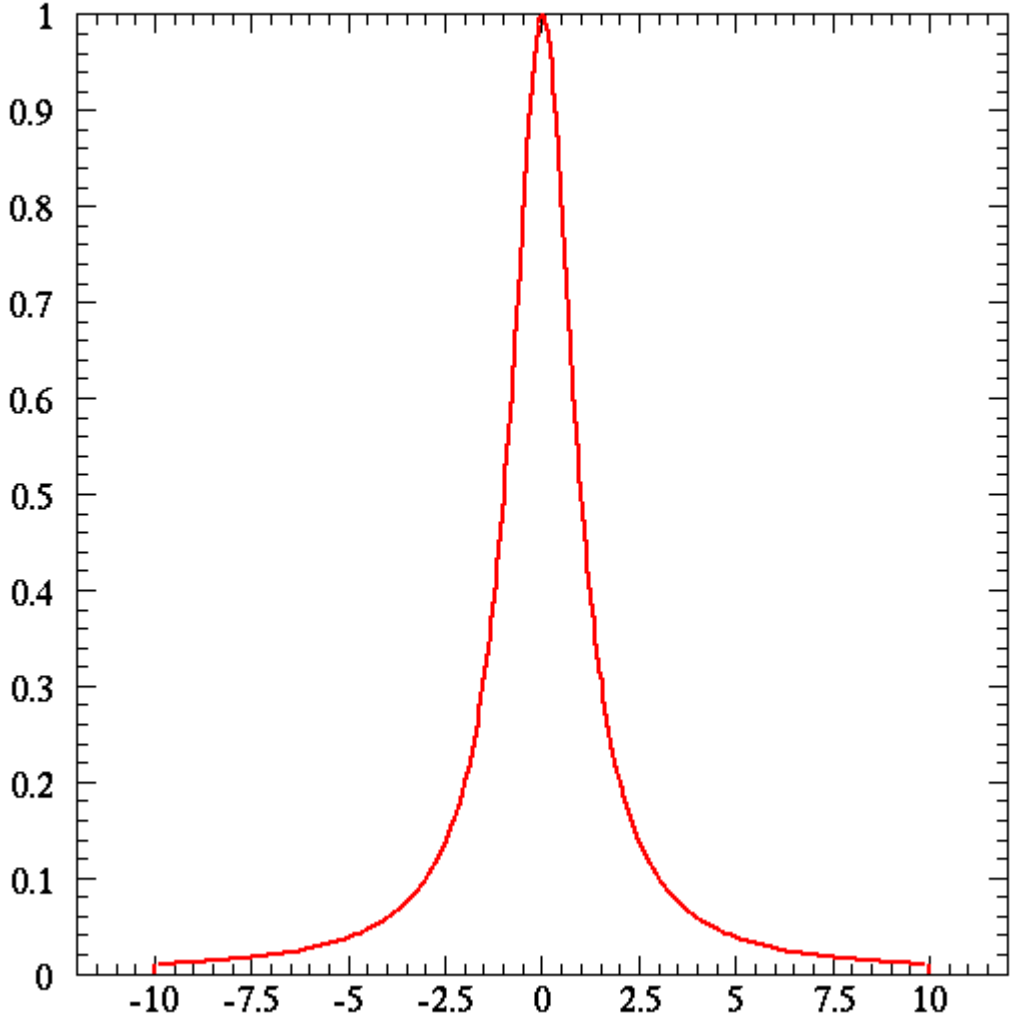
Bootstrap example #1

Consider the following setup:

1000 data points are drawn from the distribution to the right.

We sort them, and return the width between the 75% and 25% percentile points. We use this as a measure of the width of the distribution.

We want to determine the uncertainty on our width parameter.

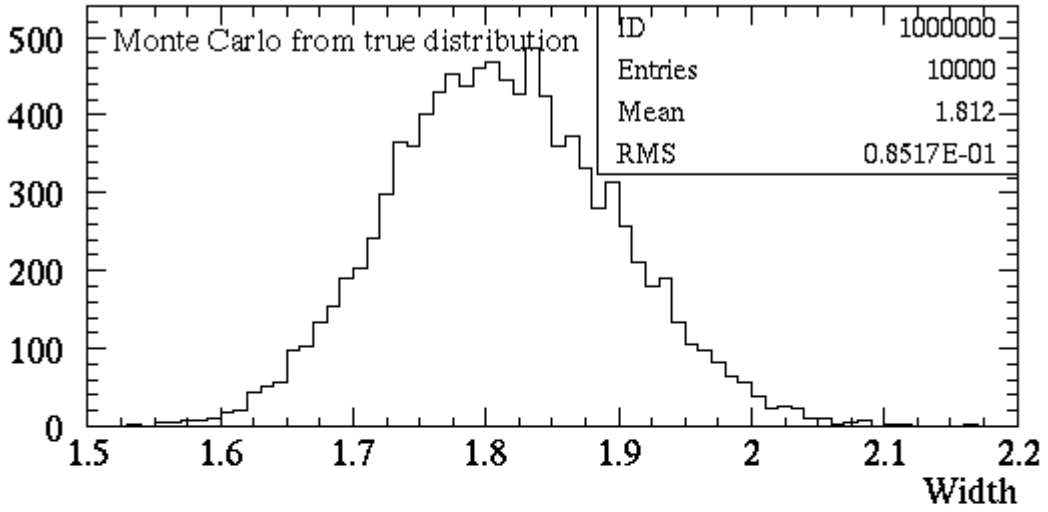


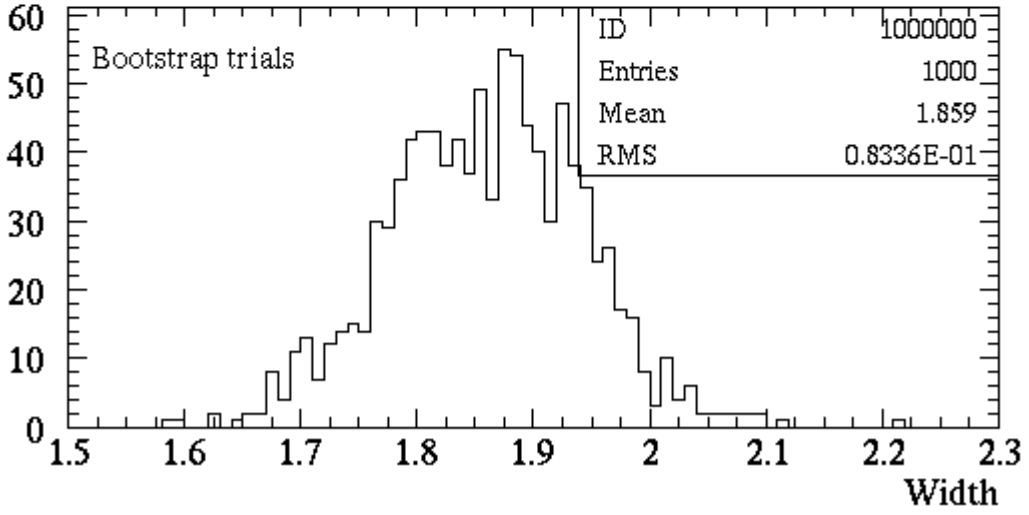
Bootstrap example #1: error on width

If we knew the true distribution, we could Monte Carlo it. Suppose we don't, but are only handed the set of 1000 data points.

The histograms on the right show:

top: width parameter distribution from 10000 independent Monte Carlo data sets
bottom: width parameter distribution from bootstrap resampling of original data



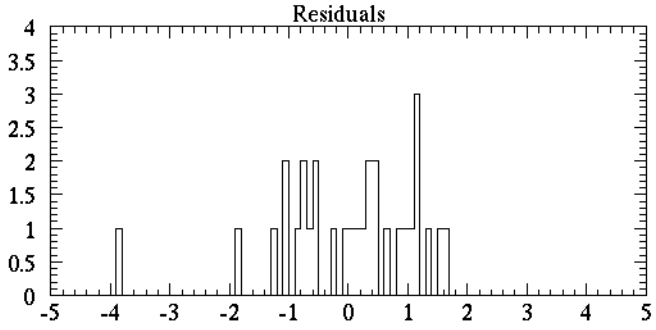


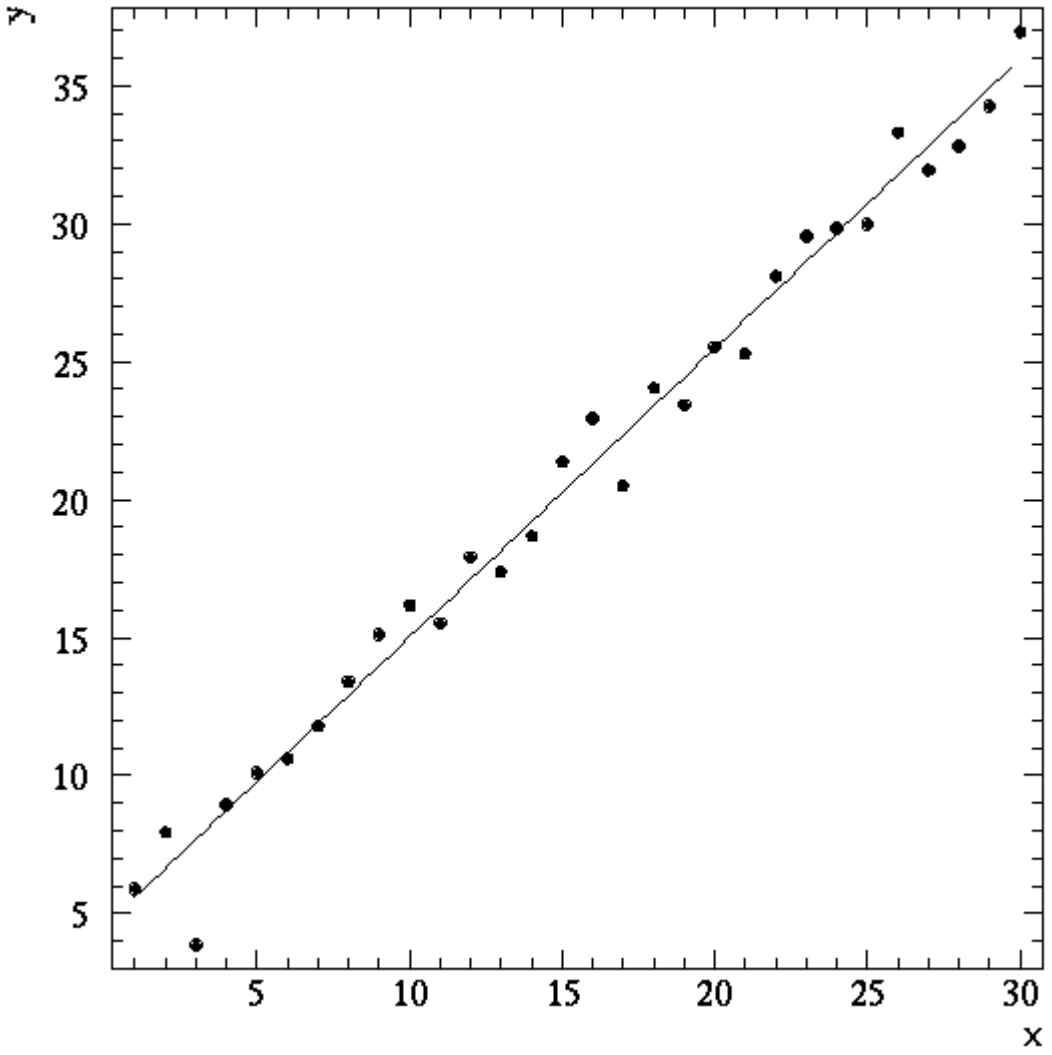
Bootstrap example #2

What are the errors on the slopes and intercepts of this data?

First fit the data for your best-fit line (as shown), using whatever estimator you like for the line (e.g. ML, least squares, etc.)

Now calculate residuals





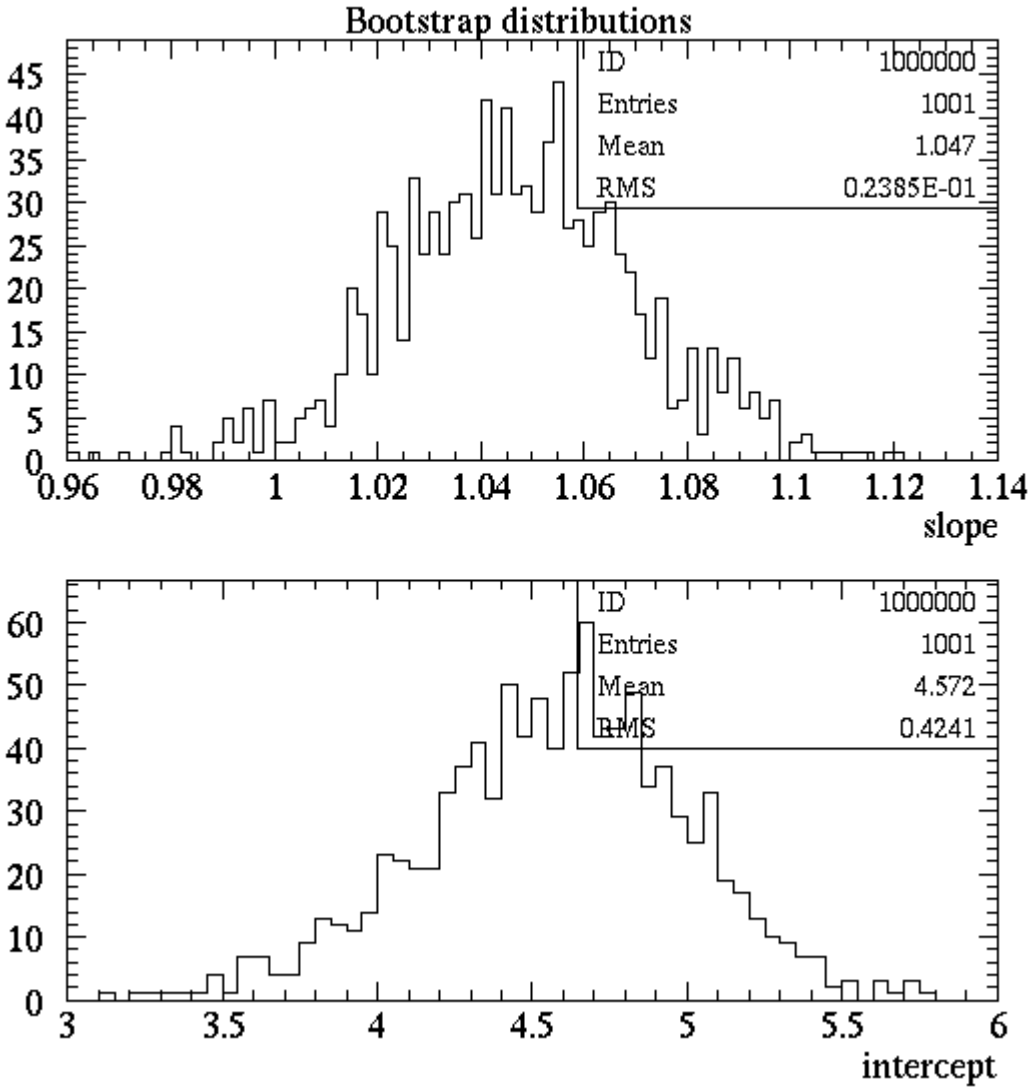
Bootstrap example #2: a line fit

Generate new "bootstrap" data sets according to:

$$y(x_i) = \hat{m}\, x_i + \hat{b}_i + \text{residual}_j$$

where we randomly pick one of the N residual values to add to the best-fit value.

We get m=1.047±0.024
and b= 4.57 ± 0.42



Evaluation of bootstrap example #2: a line fit

The bootstrap data sets gave us:

$m = 1.047 \pm 0.024$

$b = 4.57 \pm 0.42$

But Monte Carlo data sets sampled from the true distribution give

$m = 1.00 \pm 0.038$

$b = 5.02 \pm 0.68$

The error estimates in this case don't agree well. The problem is that we got kind of unlucky in the real data---with only 30 data points, the observed residual distribution happens to be narrower than the true residual distribution.

This is an example of a case where the bootstrap didn't work well.

When not to use the bootstrap

Regrettably the conditions in which the bootstrap gives bad results are not fully understood yet. Some circumstances to be wary of:

1) Small sample sizes (N<50)
2) Distributions with infinite moments
3) Using bootstraps to estimate extreme values (eg. random numbers are drawn between [*a,b*], and you want to estimate *a* and *b*.)
4) Any case where you have reason to suspect that the data don't accurately sample the underlying distribution (eg. you expect the residual to be symmetric based on physics but for the sample you have it isn't very symmetric.)

Robust parameter estimation

Almost every statistical test makes some implicit assumptions. Some make more than others. Example of assumptions that are often made and sometimes questionable:

- independent errors
- Gaussian errors
- some other specific model of the errors

When these assumptions are violated, the test may “break”. Data outliers are a good example---if you're trying to compare the average incomes of college graduates with people who don't have college degrees, the accidental presence of drop-out Bill Gates in your random sample will really mess you up!

A “robust” test is a test that isn't too sensitive to violations of the model assumptions.

“Breakdown point”

The “breakdown point” of a test is the proportion of incorrect observations with arbitrarily large errors the estimator can accept before giving an arbitrarily large result.

The mean of a distribution has a breakdown point of zero: even one wildly wrong data point will pull the mean!

The median of a distribution is very robust, however---up to half of the data points can be corrupted before the median changes (although this seemingly assumes in turn that the corrupted data are equally likely to lie above and below the median).

Non-parametric tests as robust estimators

You've already seen a number of “robust” estimators and tests:

- the median as a measure of the location of a distribution
- rank statistics as a measure of a distribution's width: for example, the data value of the 75% percentile minus the value at the 25% percentile
- Kolmogorov-Smirnov “goodness of fit” test
- Spearman's correlation coefficient

The non-parametric tests we studied in Lecture 19 are in general going to be more robust, although less powerful, than more specific tests.

M-estimator

In a maximum likelihood estimator, we start with some PDF for the error residuals:

$$f\left(y_{data} - y_{model}\right)$$

We then seek to minimize the negative log likelihood:

$$-\sum_i \ln f\left(y_i - y_{model}\left(x_i|\vec{\alpha}\right)\right)$$

This is all well justified in terms of probability theory. Therefore we can use this likelihood in Bayes' theorem, etc.

Most commonly f() is a Gaussian distribution.

An M-estimator is a generalization of the ML estimator. Rather than using a Gaussian distribution, or another distribution as dictated by your error model, use a different distribution designed to be more robust.

How M-estimators work

Define some “fit function” you want to minimize:

$$\sum_{i=1}^{N} \rho\left(\frac{y_i - y(x_i|\vec{\alpha})}{\sigma_i}\right)$$

After taking the derivative with respect to the fit parameters α we get:

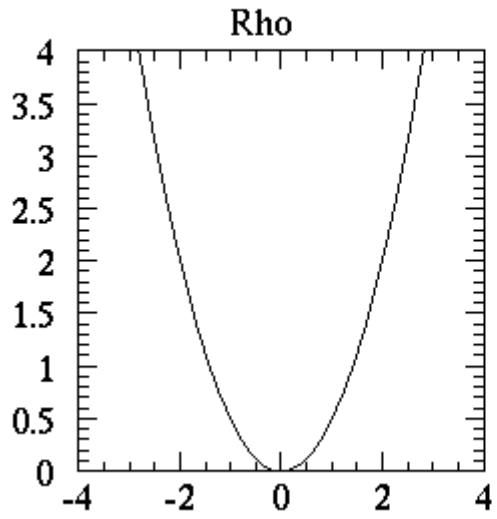
$$0 = \sum_{i=1}^{N} \frac{1}{\sigma_i} \psi\left(\frac{y_i - y(x_i|\vec{\alpha})}{\sigma_i}\right)\left(\frac{\partial\, y(x_i|\vec{\alpha})}{\partial \alpha_k}\right) \quad \text{for } k = 1,\ 2,..,M$$

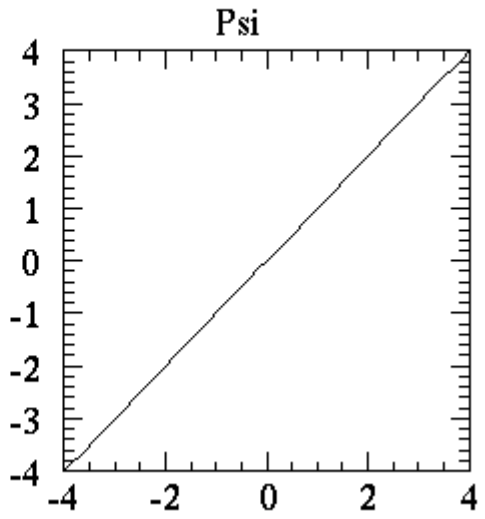
where $\psi(x) \equiv d\rho/dx$. This function ψ is a weighting function that dictates how much deviant points are weighted.

Let's see some examples ...

Examples of some M-estimators

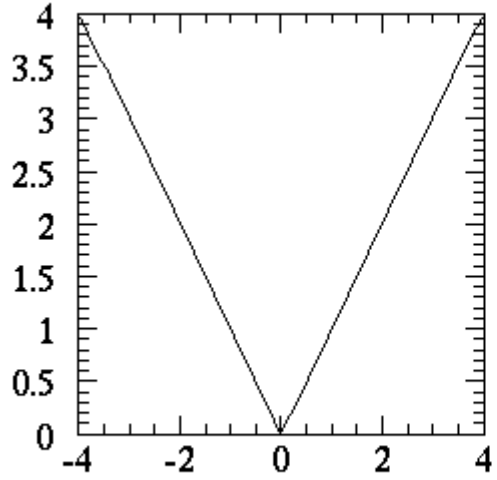
Gaussian errors: deviation enters as quantity squared, bigger deviants weighted more

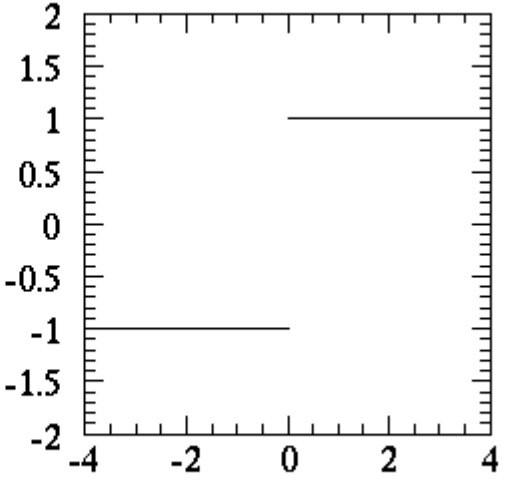




Absolute value: equivalent to minimizing

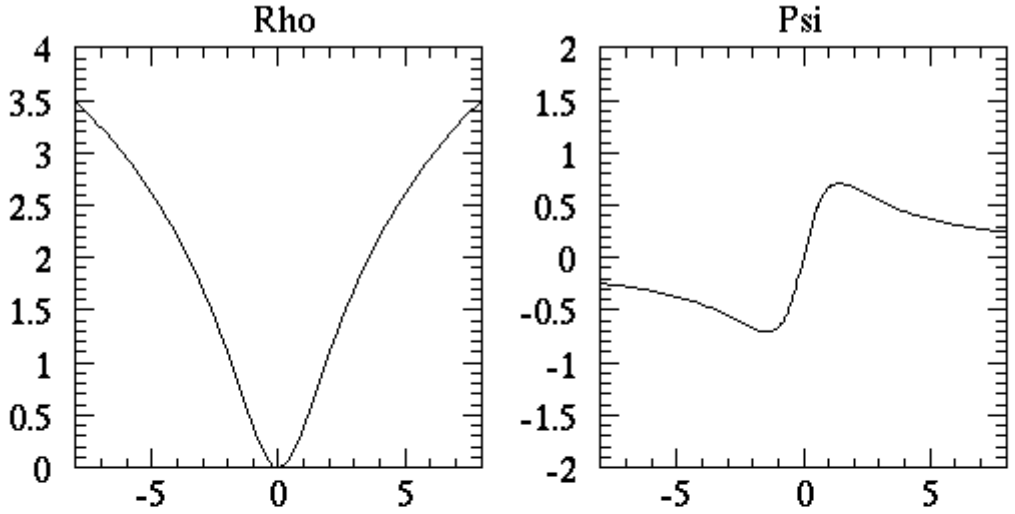
$$\sum_i |(y_i - y(x_i|\alpha))|$$





Examples of some M-estimators

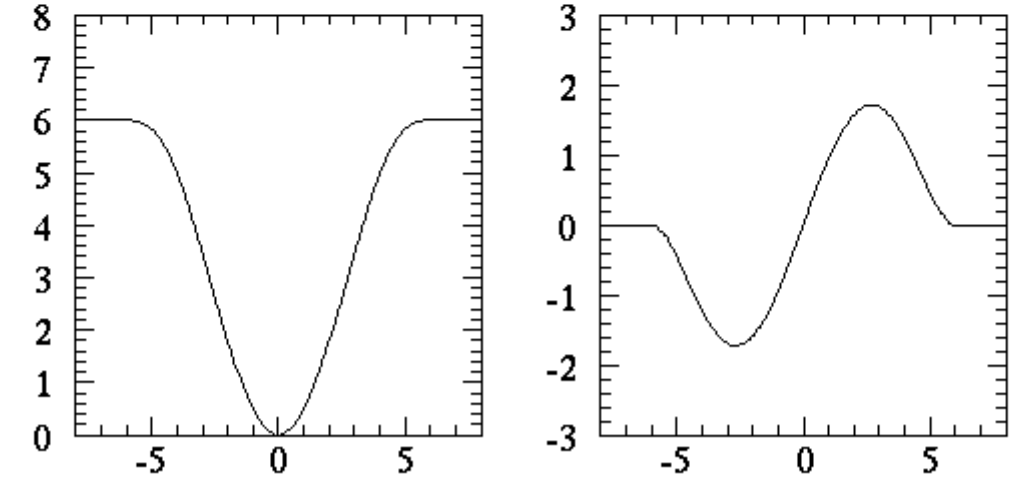
Cauchy distribution: if errors follow a Lorentzian. Weighting function goes to zero.



Tukey's biweight:

$$\psi(x) = x(1 - x^2/36)^2$$

for |x|<6. All >6σ deviations are ignored completely!



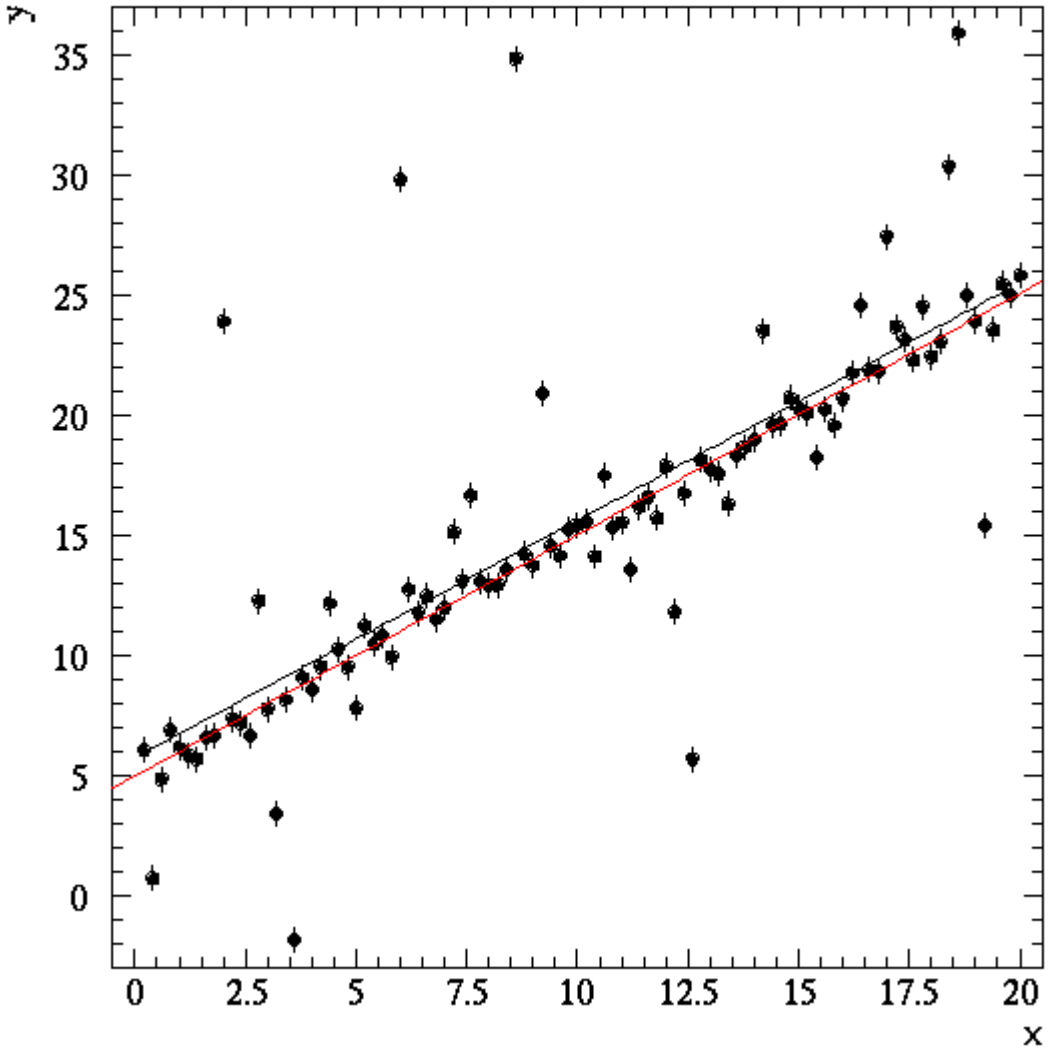
An example: linear fit with an M-estimator

Minimizing the sum:

$$\sum_i |(y_i - y(x_i|\alpha))|$$

The black line is from a χ^2 fit, while the red line is from using the absolute value version. The red point goes much closer to the majority of data points.

Fit may be more robust, but you don't get an error estimate easily. If you know the true underlying distribution, use MC, else try bootstrap to get the variance on the best fit.



What M-estimators don't do ...

You'd only use an M-estimator if you don't know the true probability distribution of the residuals, or to cover yourself in case you have the wrong error model.

You cannot easily interpret the results in terms of probability theory---Bayes' theorem is out, for example, as are most frequentist tests.

About the only thing you can do to determine the errors on your parameters is to try the bootstrap method to estimate the errors (remembering that this assumes that the measured data has accurately measured the true underlying probability distribution).

(Of course if you know the underlying probability distribution you can still use an M-estimator, and use probability theory to determine the PDFs for your estimator, but it would be a little strange to do so.)

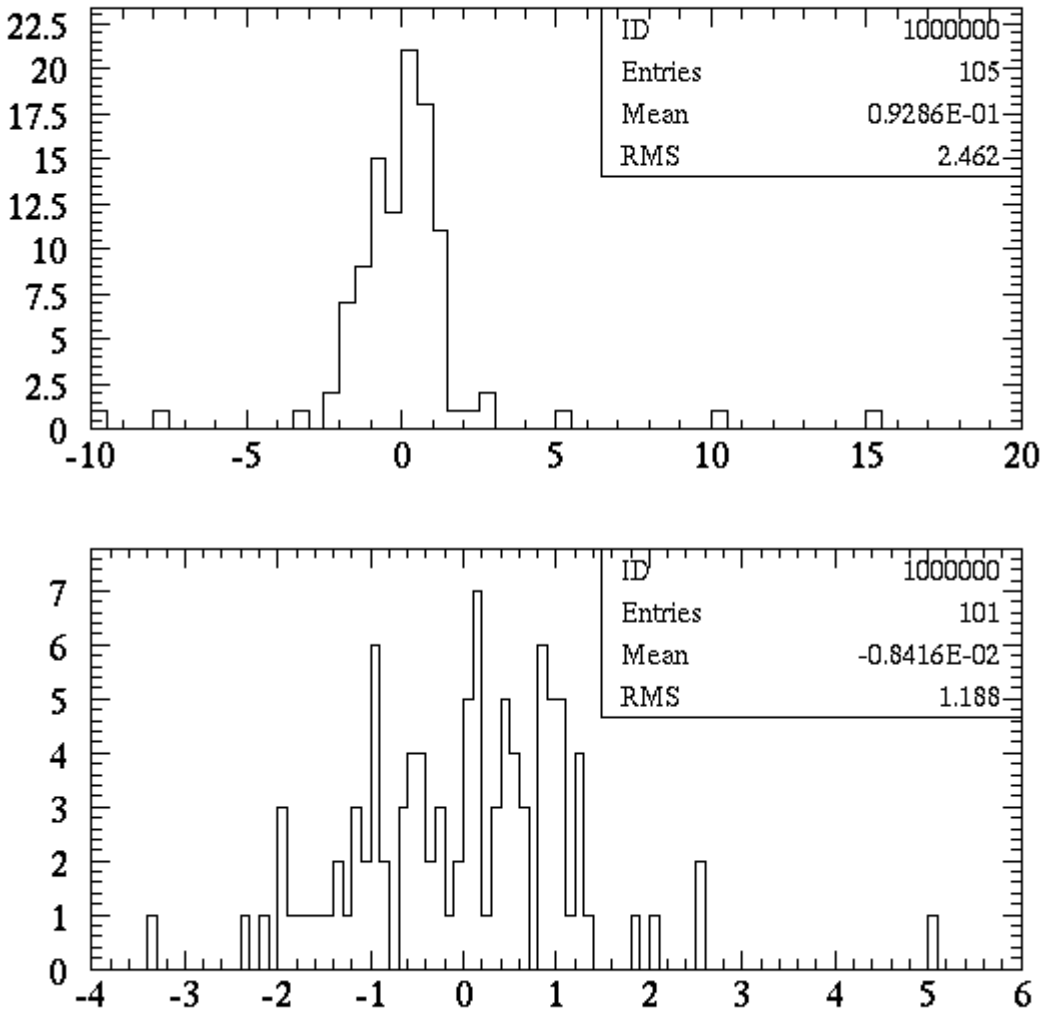
Masking outliers

Outliers can be very tricky. A really big outlier can mask the presence of other outlying points.

For example, suppose we decide in advance we'll throw out any point that is >3σ from the mean. That rejects 4 points from the top plot.

But if we look at what remains, the point at +5.0 is also pretty far out, although it passed our first cut.

Do we iterate?

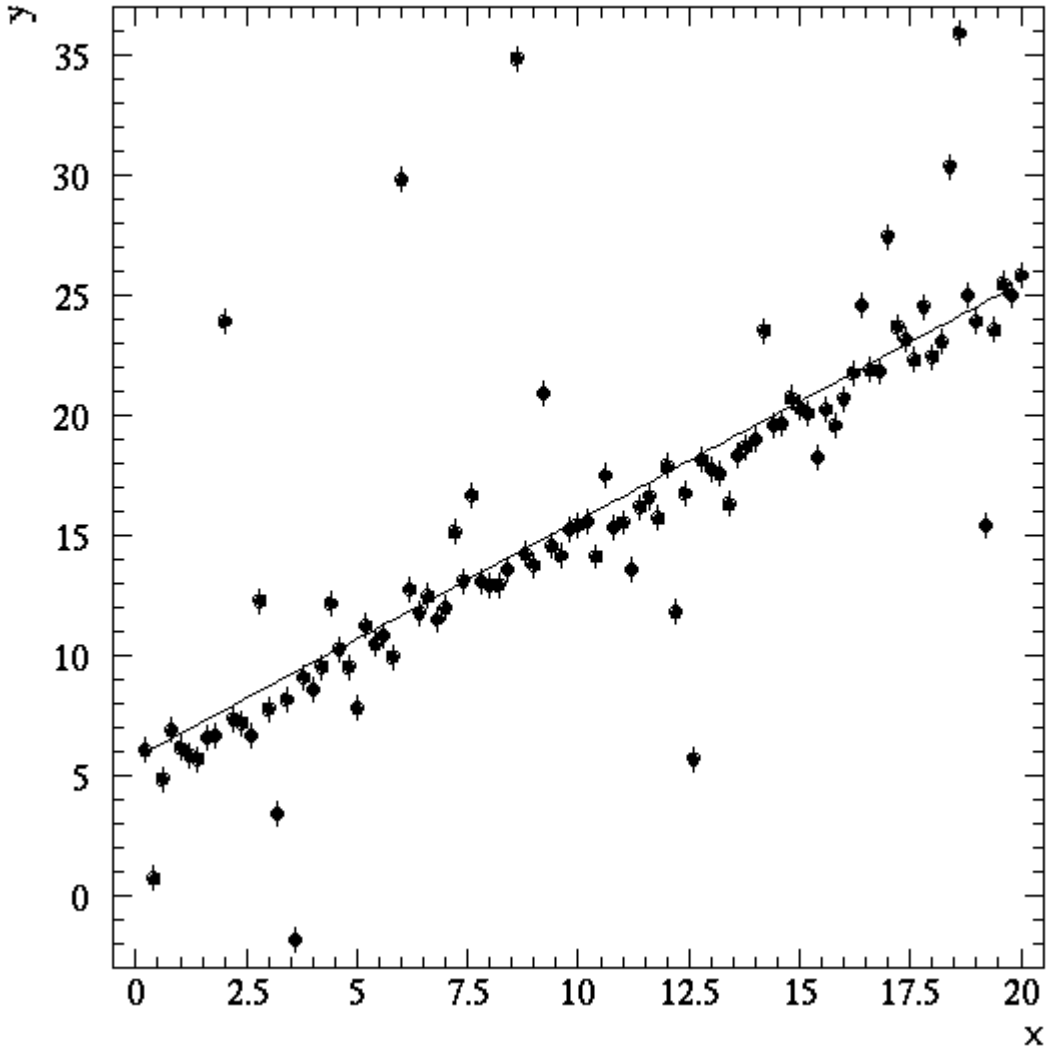


Robust Bayesian estimators

There really isn't any such thing as a “non-parametric” Bayesian calculation. Bayesian analyses need a clearly defined hypothesis space.

But Bayesian analyses can be made more robust by intelligently parametrizing the error distribution itself with some free parameters.

Consider fitting a straight line to this data.



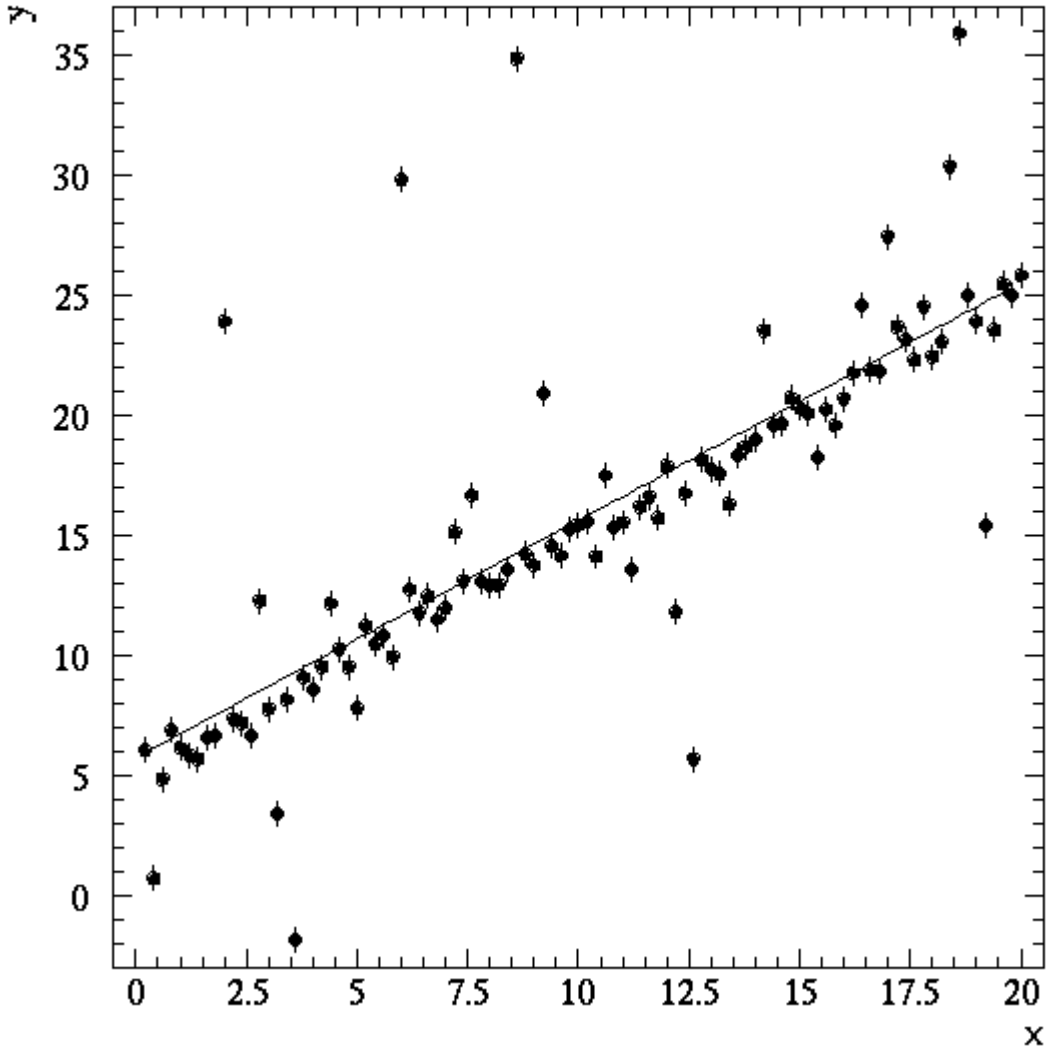
Naïve Bayesian result

Fit using Gaussian errors with nominal errors of 0.5.

68% 1D credible regions:
m= 0.9894 ± 0.0084
b = 5.731 ± 0.097

True data was drawn from m=1.0, b=5.0

Because the error estimates are just not realistic, we get an absurd result.



“Conservative” Bayesian result

Instead of fixing the errors at 0.5, make σ a free parameter. Give it a Jeffrey's prior between 0.1 and 20.0.

$$P(m,b,\sigma|D,I) \propto \frac{1}{\sigma}\prod_{i=1}^{N}\frac{1}{\sigma\sqrt{(2\pi)}}\exp\left(-\frac{1}{2}\left(\frac{y_i - mx_i - b}{\sigma}\right)^2\right)$$

I used a Markov Chain Monte Carlo to calculate the joint PDF of these three parameters and to generate 1D PDFs for each by marginalizing over the other two parameters.

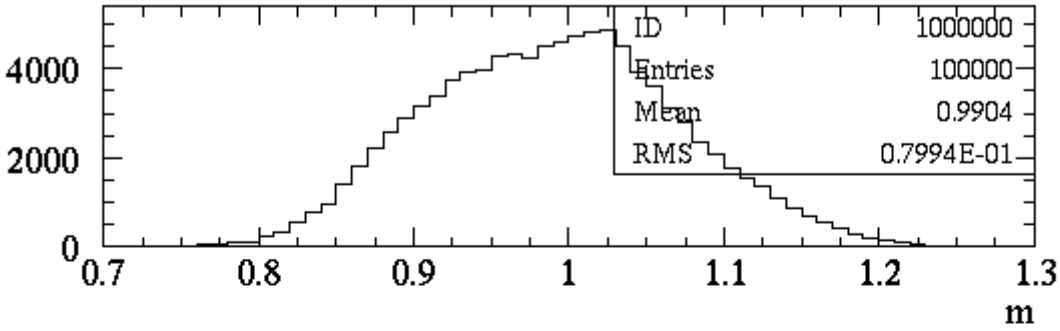
“Conservative” Bayesian result

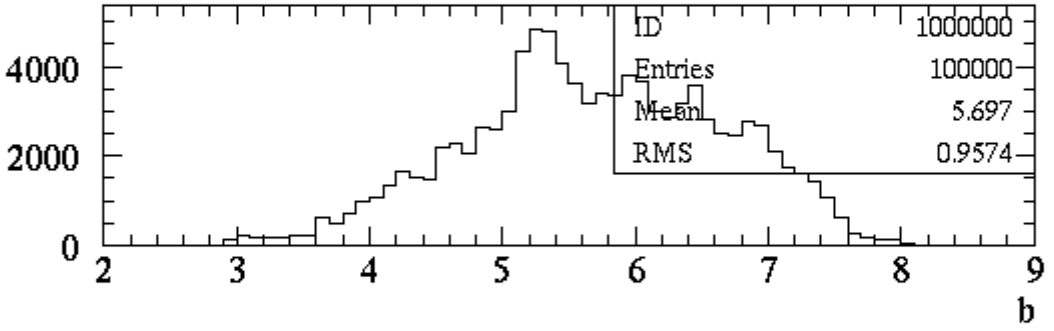
m= 0.9904 ± 0.0799
b = 5.697 ± 0.957
σ = 4.41 ± 0.32

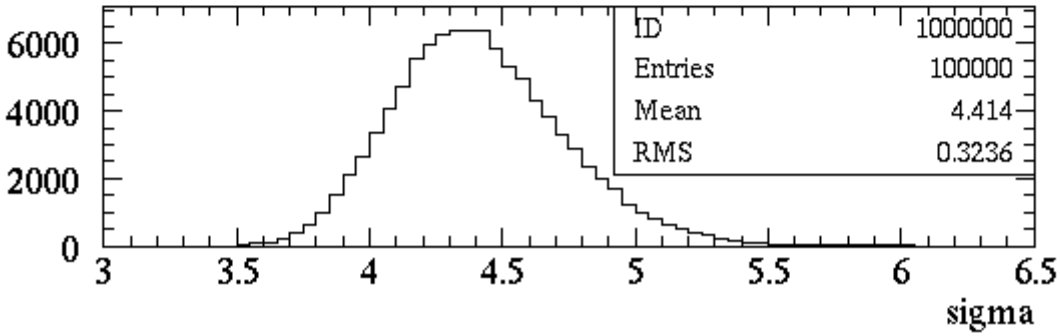
Results are consistent with true values, but with much bigger error.

Remember: by Maximum Entropy principle, a Gaussian error assumption contains the least information of any error assignment assumption.

Can we do better?







Two-component Bayesian fit

What if we model the errors as if some fraction are Gaussian and the others are from a Cauchy distribution (to give wide tails)?

$$g(\Delta) = f \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\frac{\Delta^2}{\sigma^2}\right) + (1-f)\frac{1}{\pi\tau}\frac{1}{1+(\Delta/\tau)^2}$$

Now there are 5 free parameters: m, b, σ, *f*, and τ.

Again, Markov Chain Monte Carlo is the most efficient way to calculate this. I use uniform priors on *f*, which ranges from 0 to 1, and on the width τ of the Cauchy distribution (between 0.1 and 10).

Results: m= 1.009 ± 0.015
b= 4.933 ± 0.168

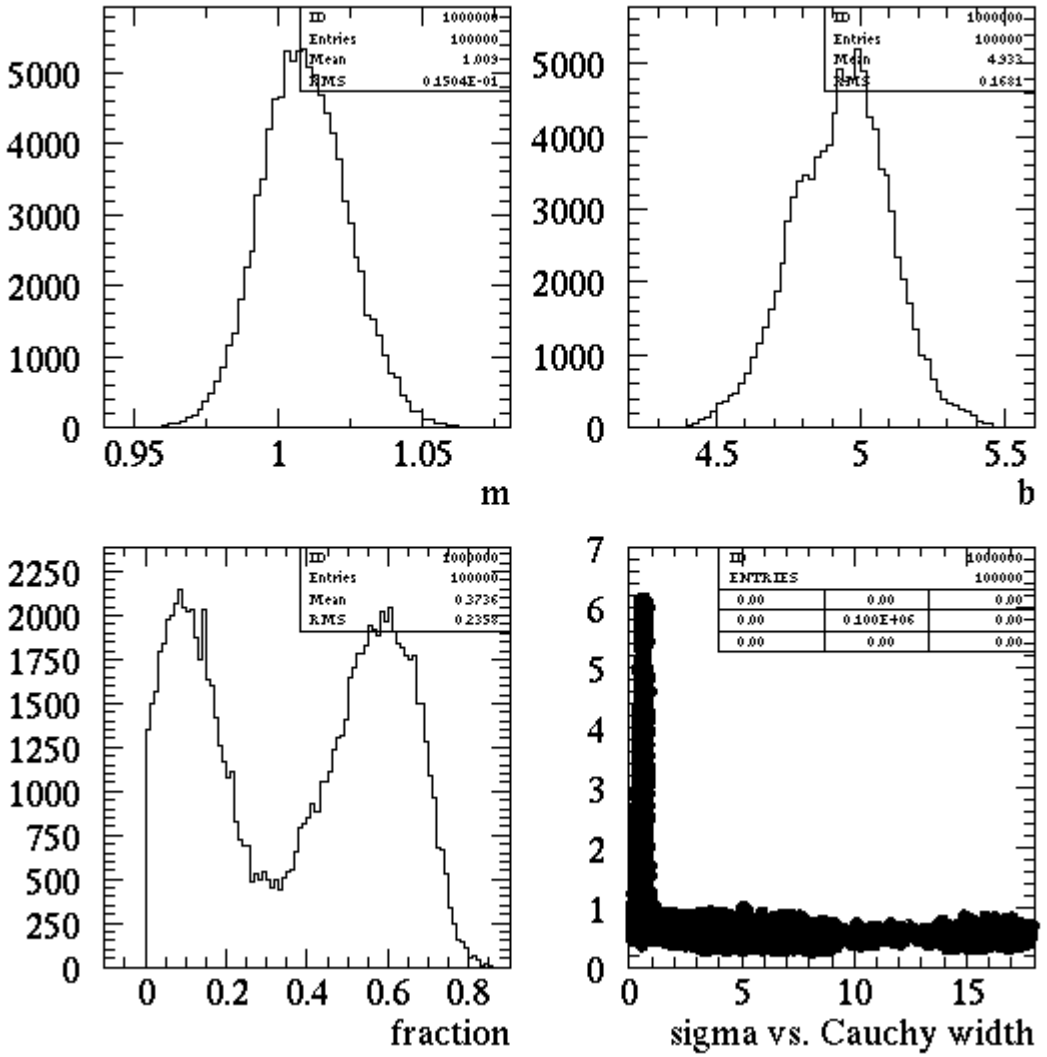
Consistent with true values, and much small uncertainties than Gaussian fit. This comes at the price of some increased model dependence.

Two-component Bayesian fit: graphs

In doing this problem I noticed a weird bimodal behaviour in the error parameters.

In retrospect the answer is obvious: the data is equally well fit by a narrow Gaussian error and a wide Cauchy error as by a narrow Cauchy error and a wide Gaussian error.

Doesn't affect the final answer, but perhaps I should have broken this degeneracy in my prior.



Notes of caution

Obviously, be VERY CAUTIOUS about throwing out any data points or deweighting outliers, especially if you don't understand what causes them!

If you look at the final answer before excluding outliers, you are certainly not doing a blind analysis, and there's an excellent chance you're badly biasing your result! (But you may be able to do *blind* outlier rejection if you plan for it in advance.)

Almost all standard results and tests in probability theory fail for censored data. You're forced to use MC or bootstrap.

Is that “outlier” actually a new discovery you're throwing out? For example, Mossbauer effect showed up as “noise” in Mossbauer's PhD thesis. He wasn't looking for it, and had he rejected it with an outlier cut would he still have won the Nobel Prize for his PhD?