# MASTER CLASS: Uncertainty Quantification (UQ) Demystified

Justin Brown — Sandia National Laboratories

Beth Lindquist — Los Alamos National Laboratory EST. 1943

Jason Bernstein — Lawrence Livermore National Laboratory

*ChatGPT's interpretation of our class title*

# Agenda

1. Justin Brown : A Shock Physics Perspective on UQ
2. Beth Lindquist: Potential Pitfalls of UQ for Physics Problems
3. Jason Bernstein: Interactive Python Examples

   https://github.com/LLNL/SHOCK-UQ

**Please help make the class as interactive as possible**

- Justin and Beth have some prepared material. We targeted ~20 minutes each so there should be plenty of time for questions or comments.
- Jason's interactive examples will take up the remainder of the class.

Sandia National Laboratories

# A Shock Physics Perspective on Uncertainty Quantification (UQ)

*SCCM Masterclass: Uncertainty Quantification Demystified*

*SCCM 2025, Washington DC*

*June 25, 9:15 – 10:45*

Justin Brown, Beth Lindquist, Jason Bernstein

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

SAND2025-07391C

# My Perspective on Practical Applications for UQ

## Putting error bars on your data

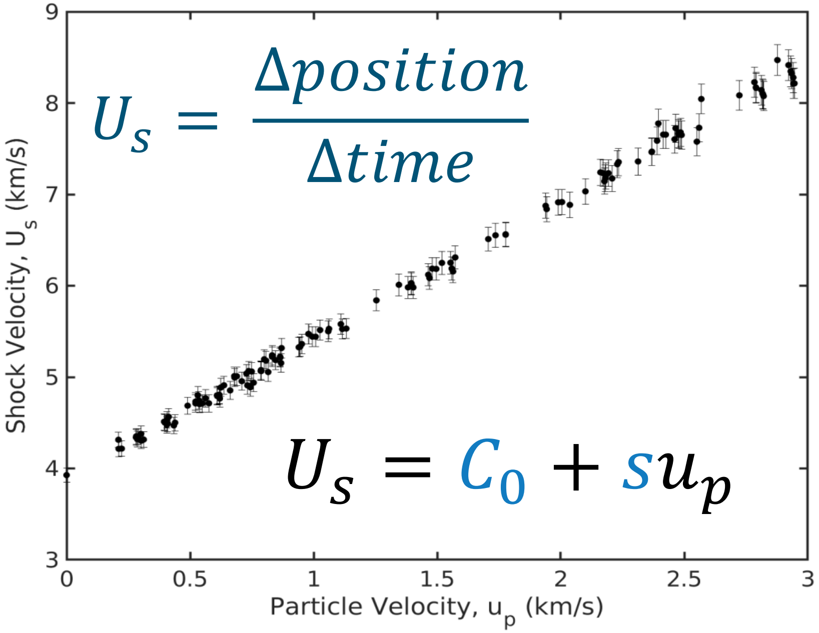
- Measure sample thickness and shock arrival time, what is the error in shock speed?

## Calibrating models

- Fit a linear shock speed – particle velocity data set

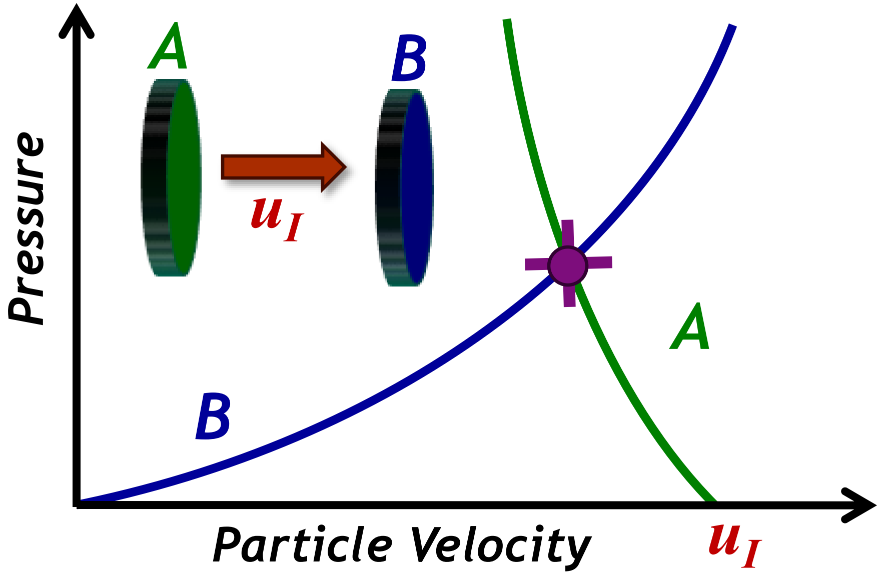
## Propagating error

- Impedance matching with uncertainty



$$U_s = \frac{\Delta position}{\Delta time}$$

$$U_s = C_0 + s u_p$$



# There are 3 methods commonly used for UQ

1 ~~Analytic~~

- ~~Regression, derivative expansions, etc.~~

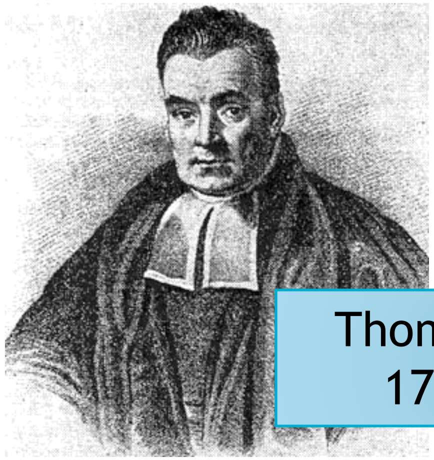
**Won’t be covered during this section. Jason will have an example later.**

2 Monte Carlo Sampling / Bootstrapping

Stanislaw Ulam, 1909-1984

https://en.wikipedia.org/wiki/Stanislaw_Ulam

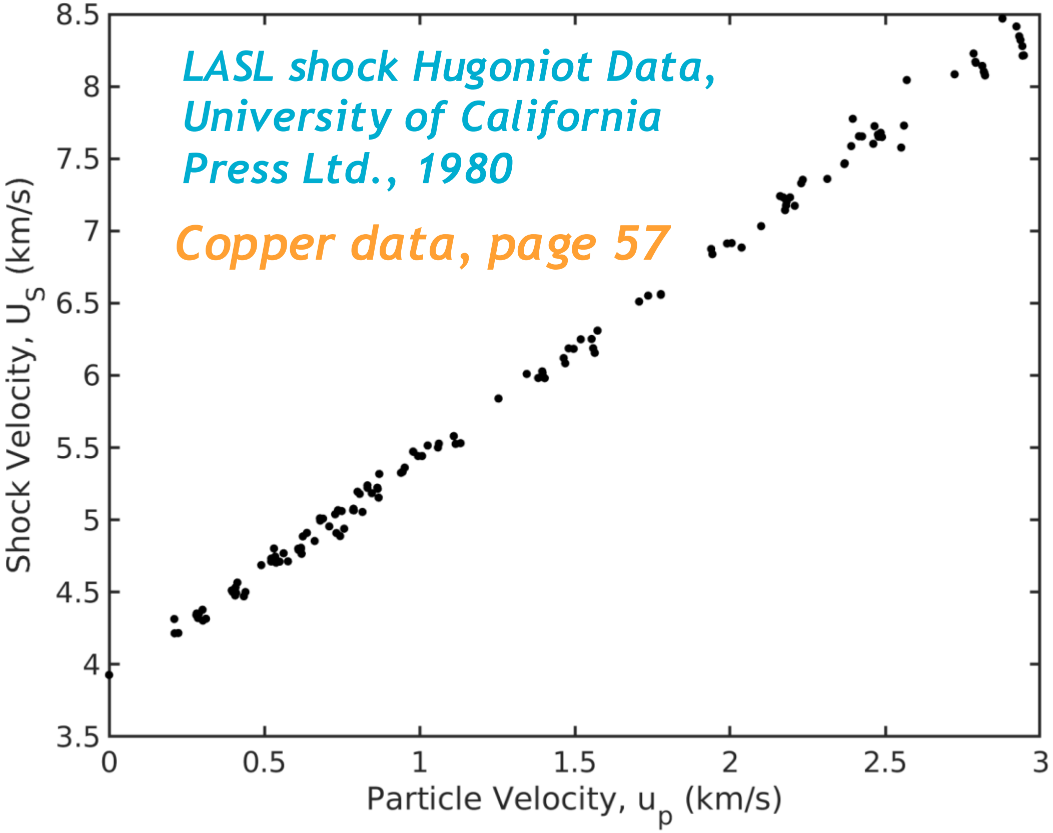
3 Bayesian



Thomas Bayes, 1701-1761

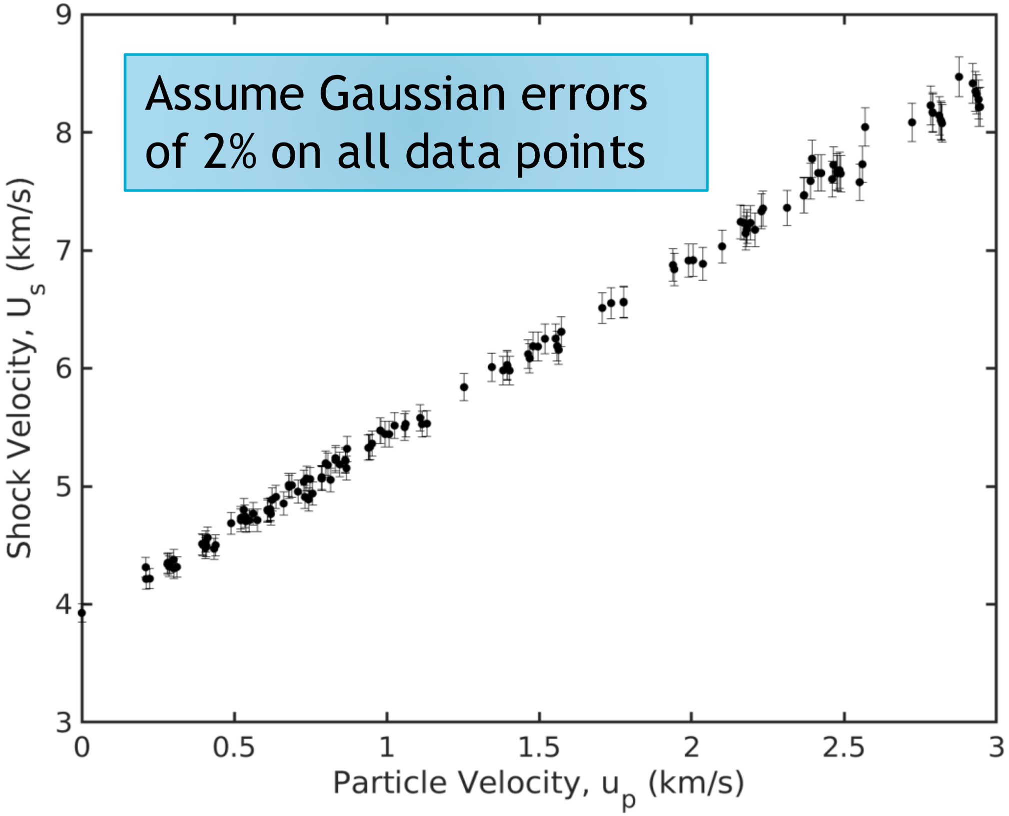
https://en.wikipedia.org/wiki/Thomas_Bayes

# Today's exemplar will be fitting a Hugoniot model



*LASL shock Hugoniot Data, University of California Press Ltd., 1980*

*Copper data, page 57*



Assume Gaussian errors of 2% on all data points

$$U_s = C_0 + s u_p$$

**Objective**: determine the probability distribution describing $C_0$ and s

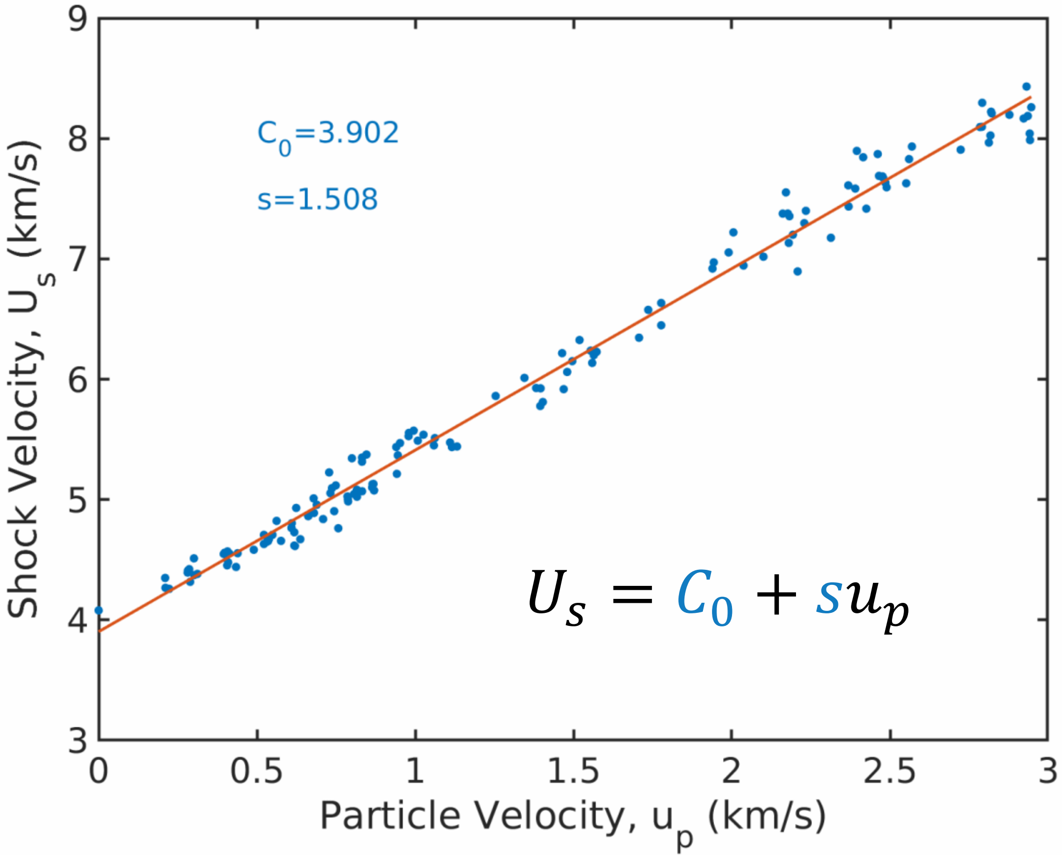
# Method 1: Monte Carlo or Bootstrap Sampling

$Convergence: O\left(\frac{1}{\sqrt{N}}\right)$

1. Randomly sample data points
2. Fit the model to each instantiation
3. Examine the distribution of fit parameters

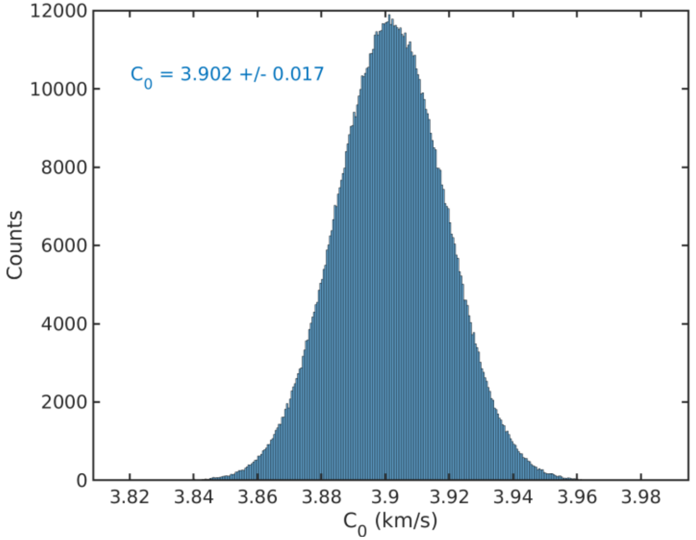
Blue points: current value of $U_s$

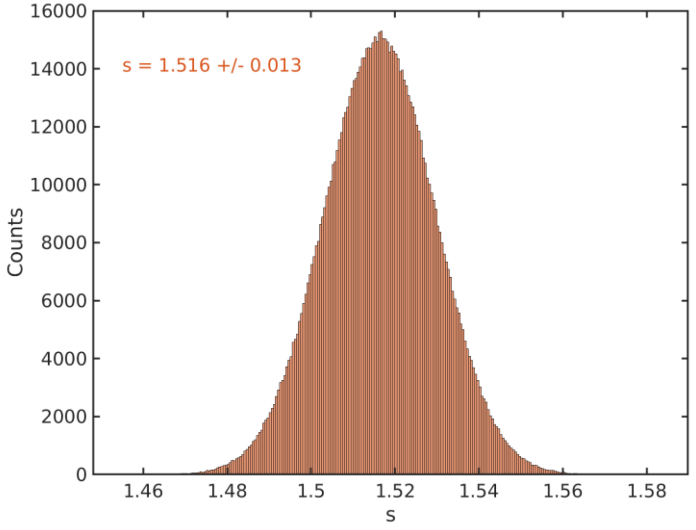
Orange line: current fit to model



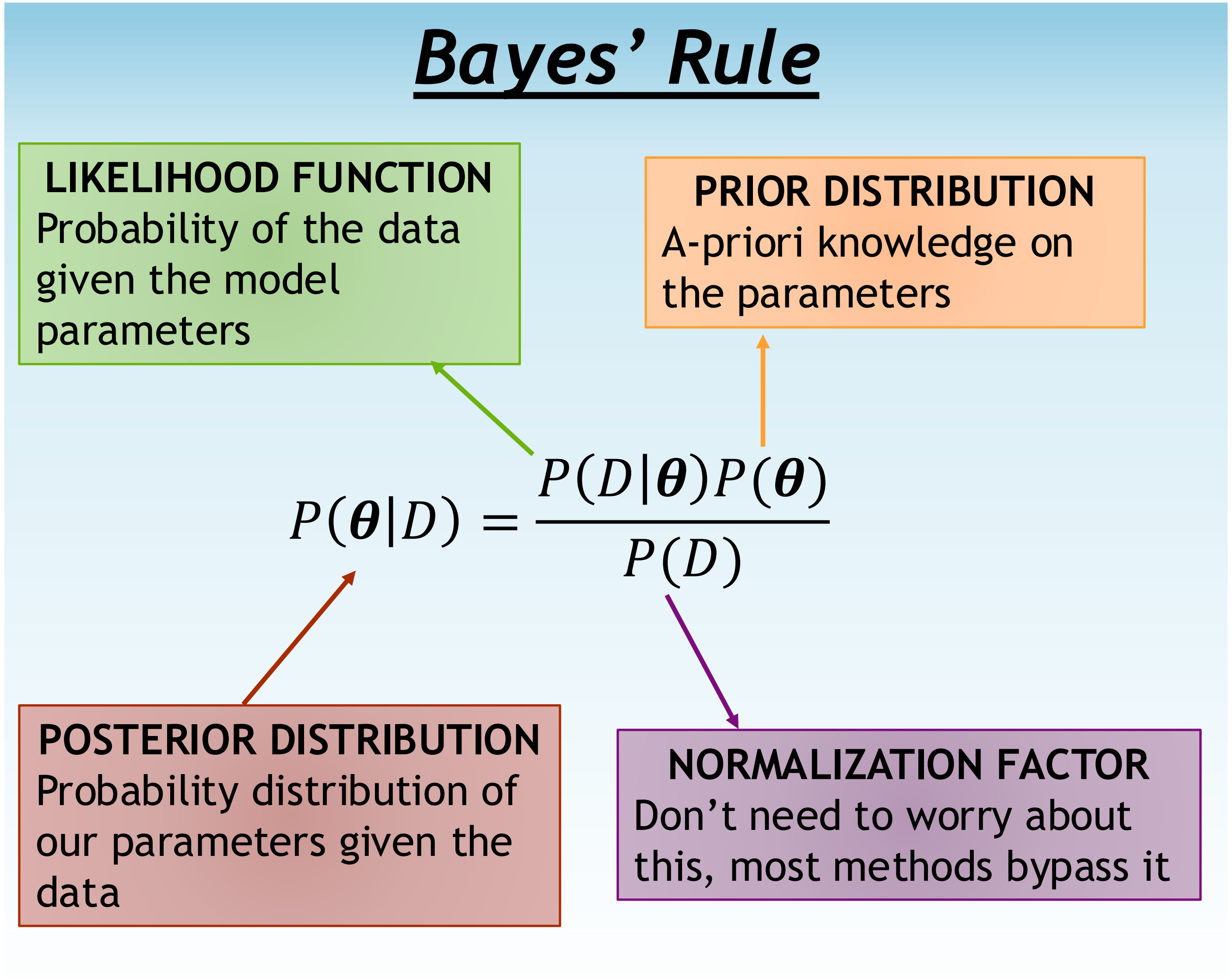
$$U_s = C_0 + s u_p$$

RESULTS FOR $10^6$ SAMPLES





# Method 2: Bayesian Calibration



## ***Bayes' Rule***

**LIKELIHOOD FUNCTION**
Probability of the data given the model parameters

**PRIOR DISTRIBUTION**
A-priori knowledge on the parameters

$$P(\boldsymbol{\theta}|D) = \frac{P(D|\boldsymbol{\theta})P(\boldsymbol{\theta})}{P(D)}$$

**POSTERIOR DISTRIBUTION**
Probability distribution of our parameters given the data

**NORMALIZATION FACTOR**
Don't need to worry about this, most methods bypass it

$P(D|\theta)$: how well a particular set of parameters explains the observed data

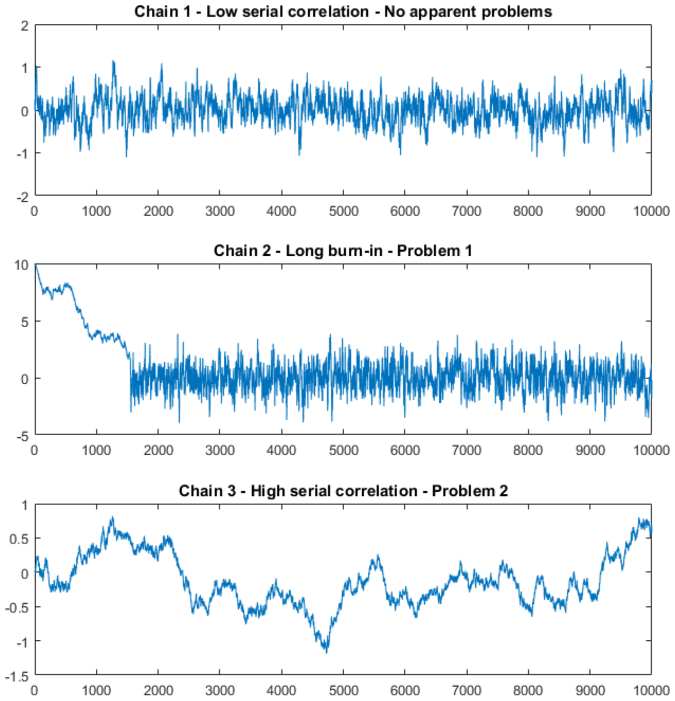
Once Bayes' rule is defined there are methods to directly sample the posterior distribution.

Typically Markov Chain Monte Carlo (MCMC) based on the Metropolis-Hastings algorithm

- Markov process: stochastic process where the probability of each event depends only on the state of the previous event (Andrey Markov, 1906)
- In the limit, samples being generated by the MCMC method will be samples from the desired target distribution
- *Nicholas Metropolis, Arianna Rosenbluth, Marshall Rosenbluth, Augusta Teller, Edward Teller, "Equation of State Calculations by Fast Computing Machines", JCP, 21 (6), 1953*

# High level overview of Bayesian Calibration

1. Define vectors of data, data errors, and model evaluation
2. Define prior distributions
3. Tell the code how long you want the chain to run for and let it do all the heavy lifting
4. Evaluate the Markov chains for burn-in and convergence
   - “Look for the hairy caterpillar”



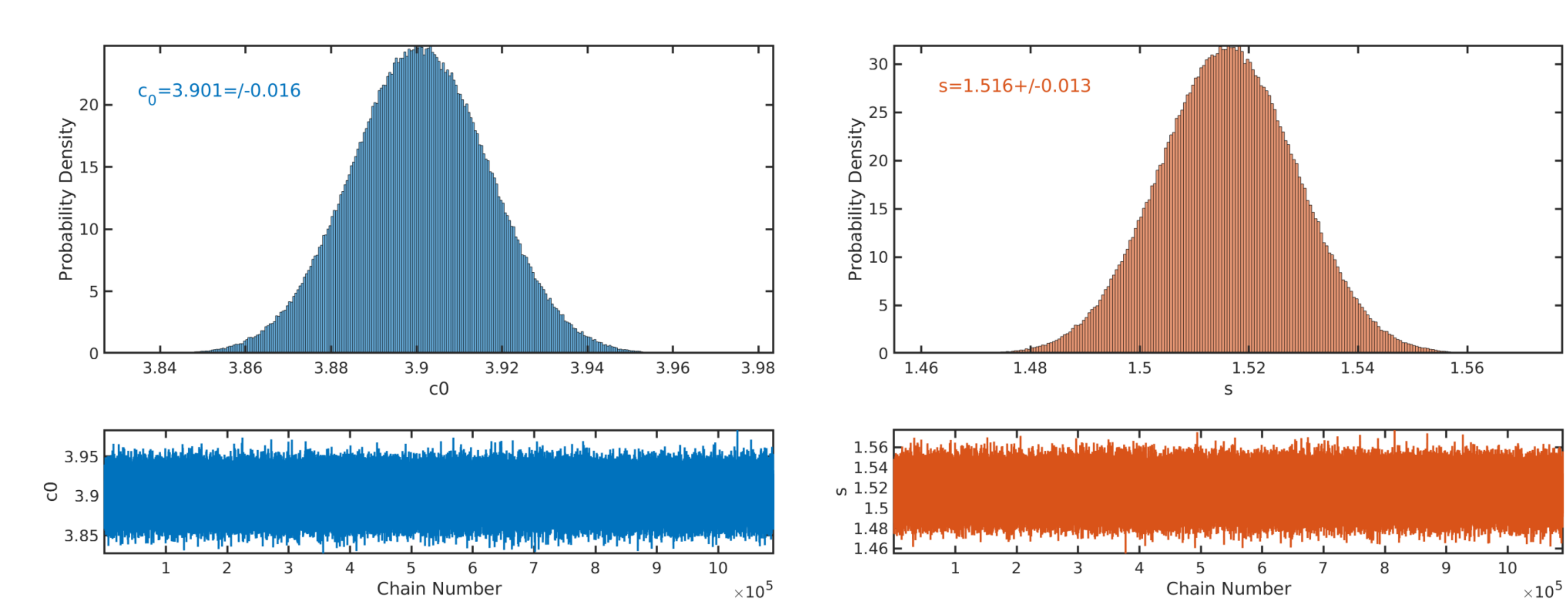
https://www.statlect.com/fundamentals-of-statistics/Markov-Chain-Monte-Carlo-diagnostics

# Bayes has some things to consider

Need to be thoughtful about setup

- Distribution on the priors
- Everything has error. If you can quantify it, you can include it.

**Noninformative priors = same as Monte Carlo**

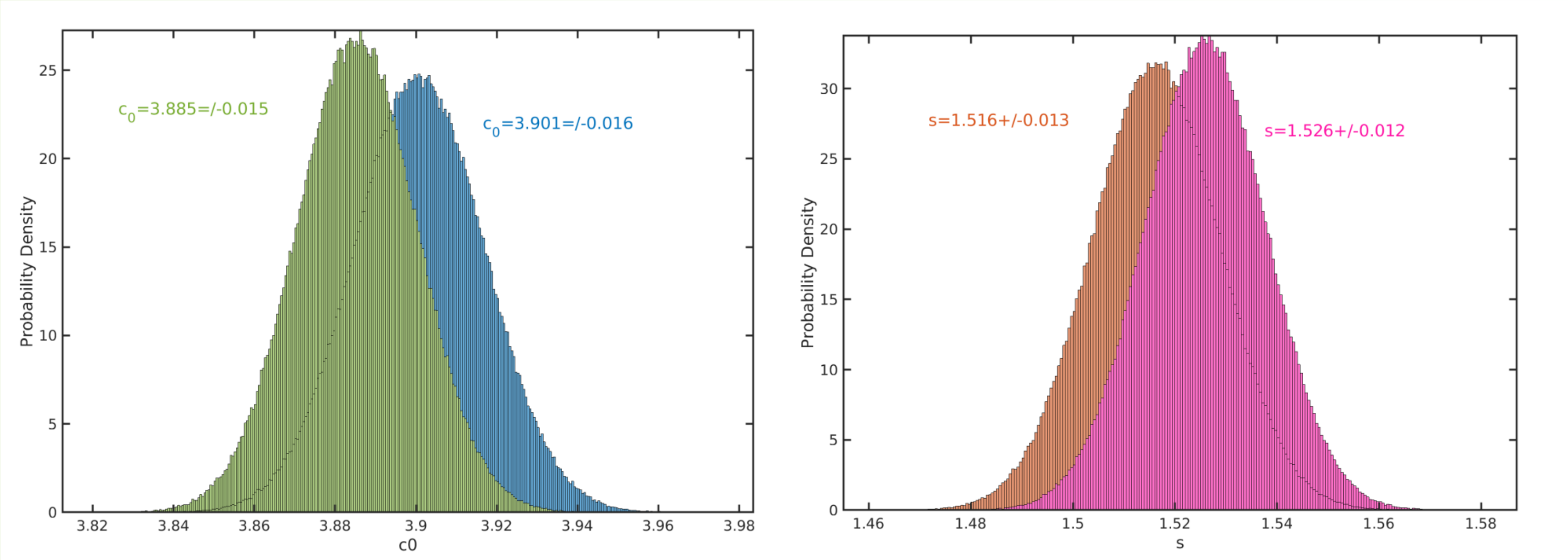


# Bayes has some things to consider

Need to be thoughtful about setup

- Distribution on the priors
- Everything has error. If you can quantify it, you can include it.

**Informative prior on c0:**
**3.80 +/- 1%**
**(Static Ultrasonic Measurement)**



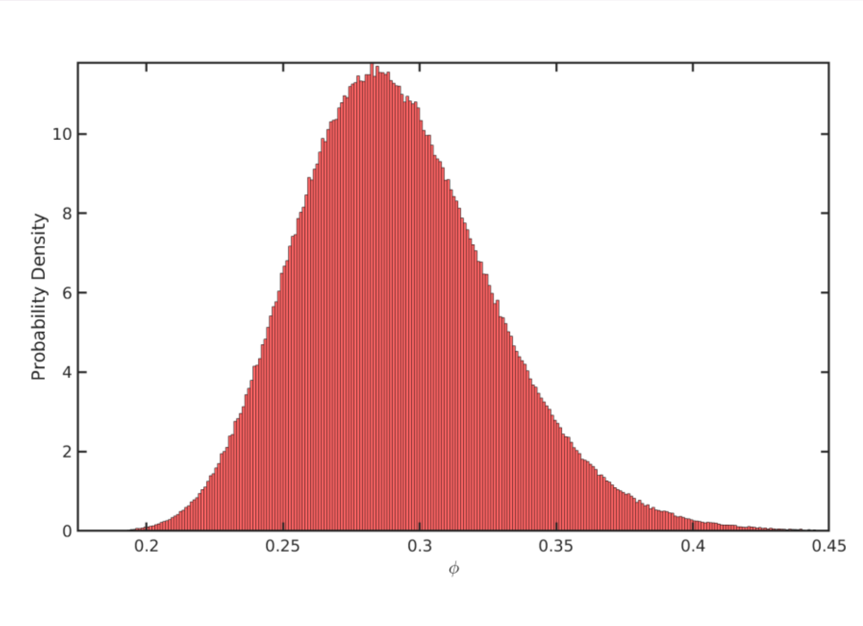
# Bayes has some things to consider

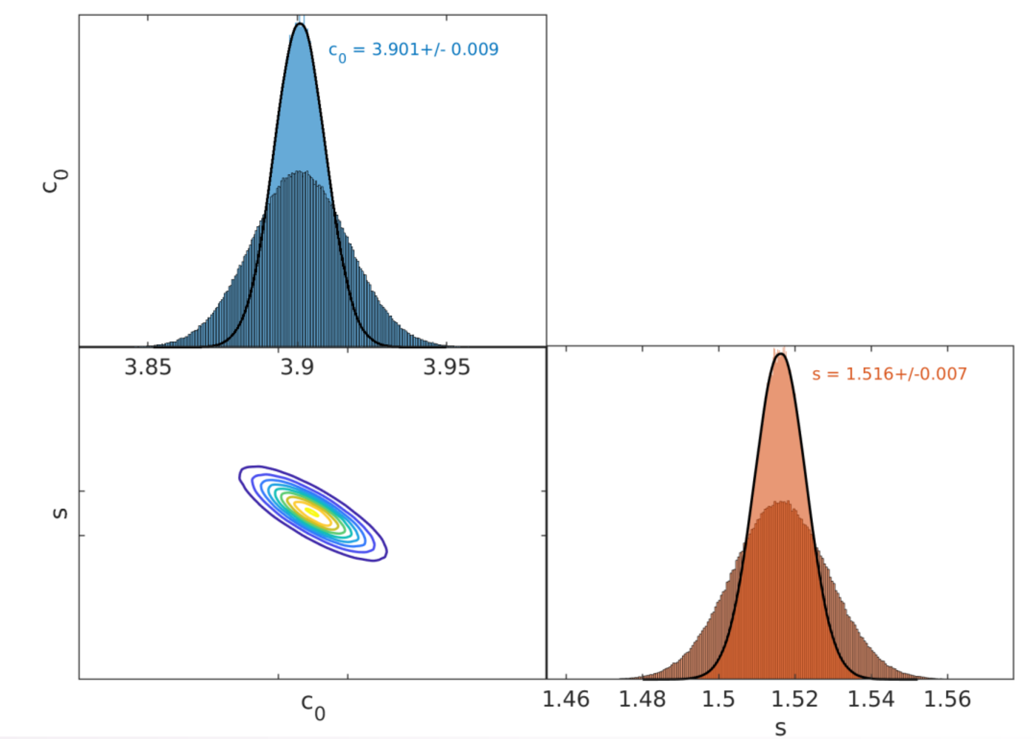
Need to be thoughtful about setup

- Distribution on the priors
- Everything has error. If you can quantify it, you can include it.

$$U_s\left(u_{p_i}\right) = c_0 + s\left(u_{p_i}\right) + \phi\epsilon_i$$

**Back to noninformative priors, but assume there is uncertainty in the data errors**





# Bootstrapping vs Bayes

## ***Bootstrap***

- Relatively intuitive and simple
- Unambiguous
  - Not many choices to make in the presence of well defined data
- Can be computationally expensive for high-dimensional problems
- Can be difficult to intractable to implement for some inverse problems

## ***Bayes***

- Generally requires more user expertise
  - Higher learning curve
  - Include all sources of error
  - Incorporate prior knowledge
  - Possible to model discrepancy
    - Be very careful about bias-variance tradeoffs
- Well-suited for high-dimensional and inverse problems
- Popular in the sciences today

# Conclusions

Each method has its pros and cons and the best approach likely depends on your application

My advice: use both if possible!

- Bootstrapping: well-suited for most problems, generally an unbiased conservative estimator
- Bayes: has the potential to reduce errors or learn about other aspects of the problem but need to be thoughtful about the setup

Both methods are generally available in any programming language

- The energy barrier for implementing UQ in your own work may not be as daunting as it seems
- Once you learn how to implement UQ in your favorite code, you may be surprised at how easy and generalizable these numerical methods are
- If you're familiar with Python, Jason's examples have you covered: https://github.com/LLNL/SHOCK-UQ