Frequentism and Bayesianism: What's the Big Deal?

Jake VanderPlas

SciPy 2014

What this talk is...

An introduction to the essential differences between frequentist & Bayesian analyses.

A brief discussion of tools available in Python to perform these analyses

A thinly-veiled argument for the use of Bayesian methods in science.

What this talk is not...

A complete discussion of frequentist/ Bayesian statistics & the associated examples.

(For more detail, see the accompanying SciPy proceedings paper & references within)

The frequentist/Bayesian divide is fundamentally a question of philosophy: the definition of probability.

What is probability?

Fundamentally related to the frequencies of repeated events.

- *Frequentists*

Fundamentally related to our own certainty or uncertainty of events.

- *Bayesians*

Thus we analyze…

Variation of data & derived quantities in terms of fixed model parameters.

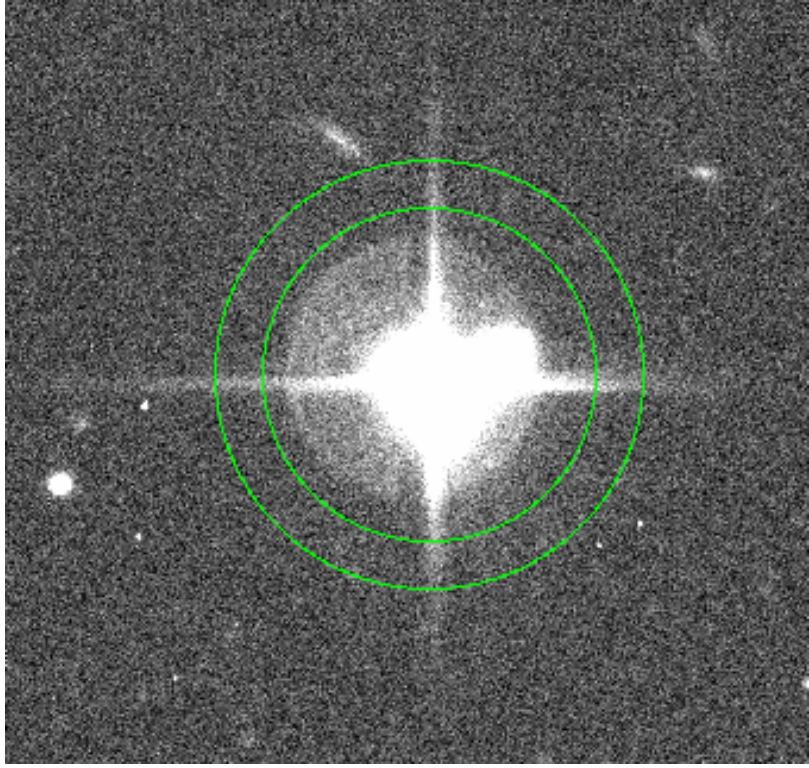
- *Frequentists*

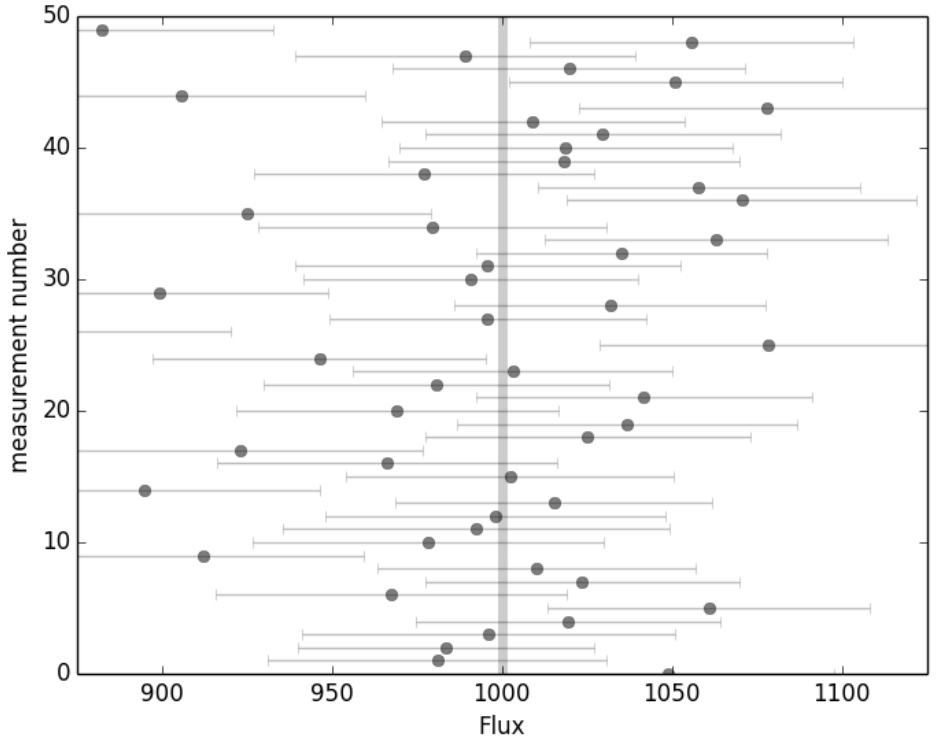
Variation of beliefs about parameters in terms of fixed observed data.

- *Bayesians*

Simple Example: Photon Flux

```
import numpy as np

np.random.seed(2)  # for repeatability
e = np.random.normal(30, 3, 50)
F = np.random.normal(1000, e)
```

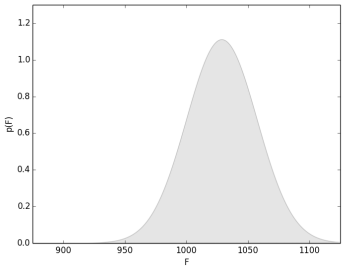




Given the observed data, what is the best estimate of the true value?

Frequentist Approach: Maximum Likelihood

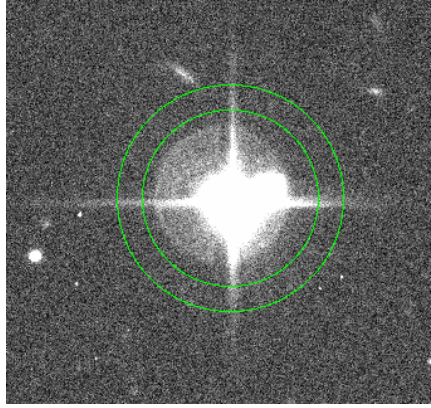
Model: each observation F_i drawn from a Gaussian of width e_i

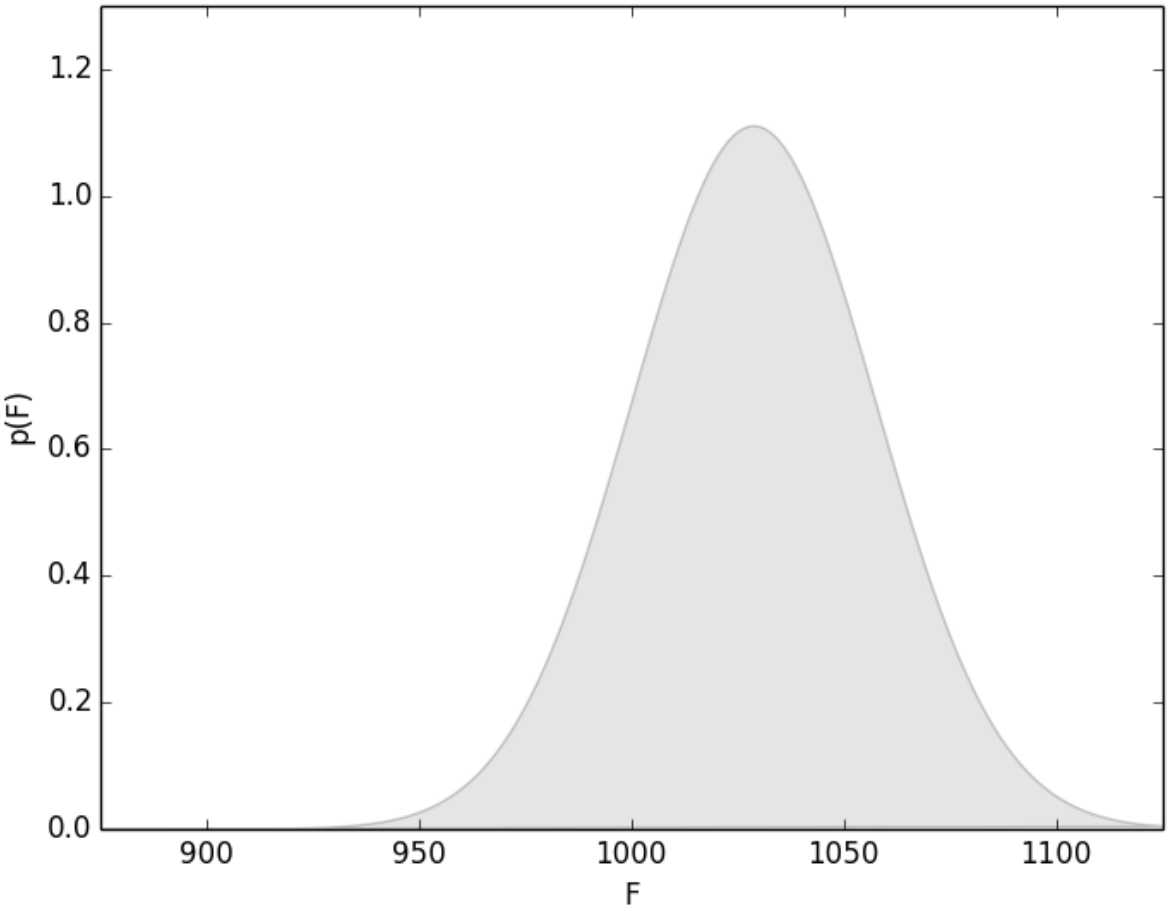


$$P(D_i \mid F_{\text{true}}) = \frac{1}{\sqrt{2\pi e_i^2}} \exp\left[\frac{-(F_i - F_{\text{true}})^2}{2e_i^2}\right]$$

Building the Likelihood…

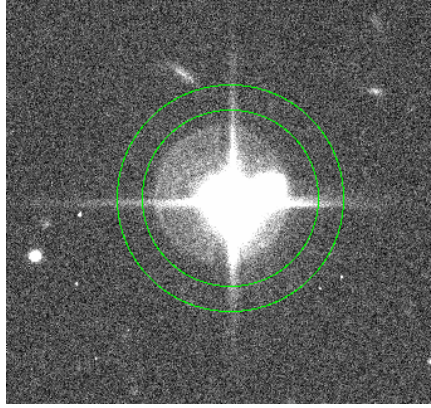
$$\mathcal{L}(D \mid F_{\text{true}}) = \prod_{i=1}^{N} P(D_i \mid F_{\text{true}})$$

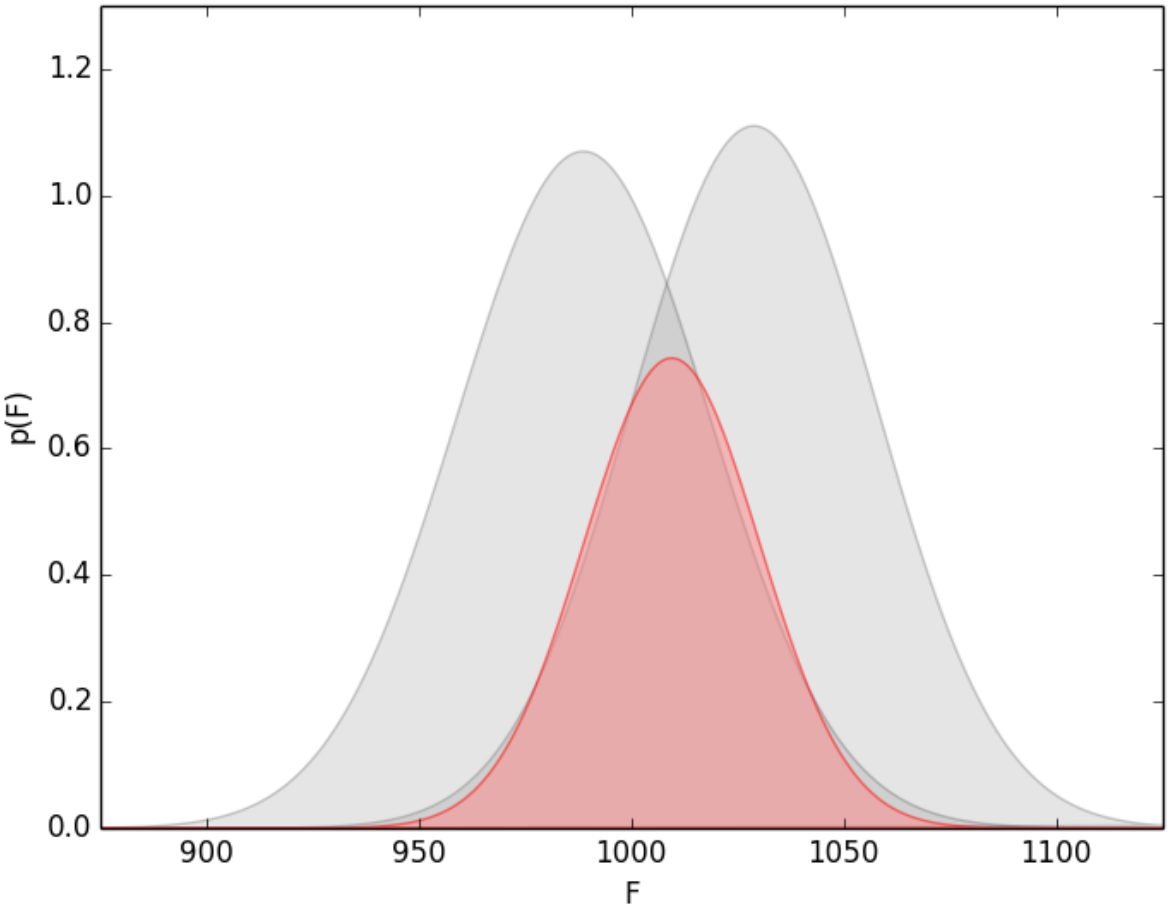




Building the Likelihood…

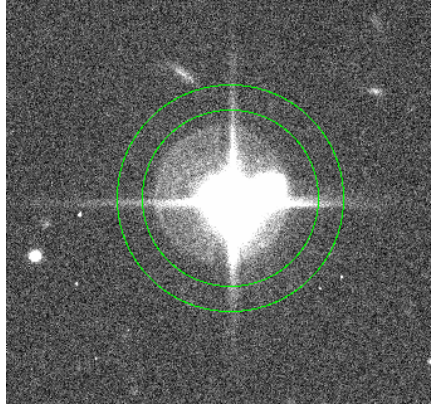
$$\mathcal{L}(D \mid F_{\text{true}}) = \prod_{i=1}^{N} P(D_i \mid F_{\text{true}})$$

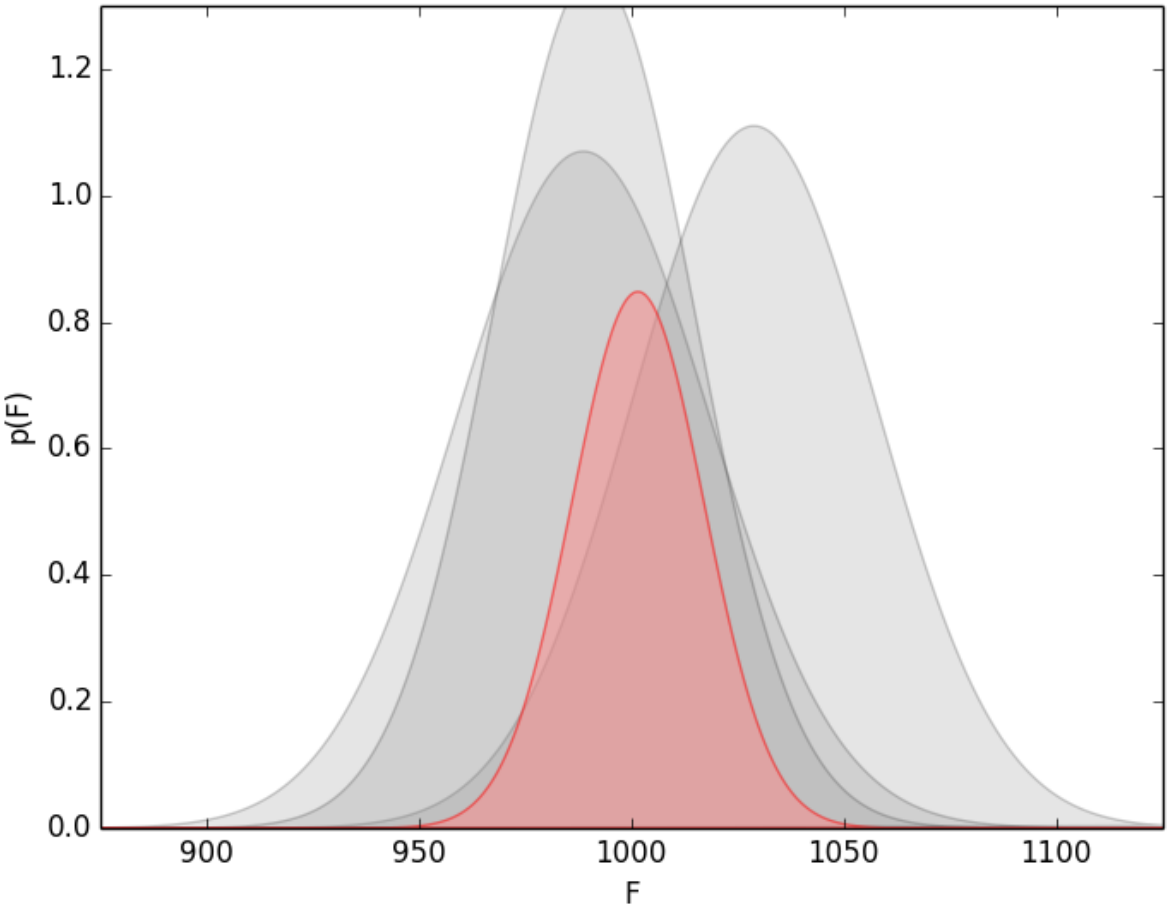




Building the Likelihood…

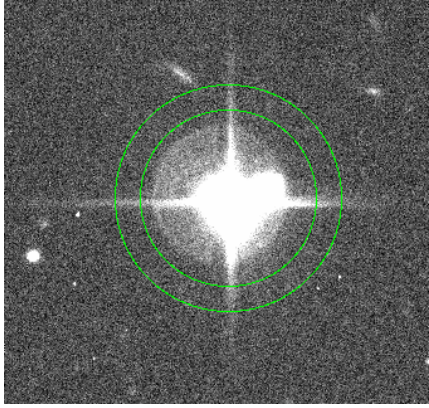
$$\mathcal{L}(D \mid F_{\text{true}}) = \prod_{i=1}^{N} P(D_i \mid F_{\text{true}})$$

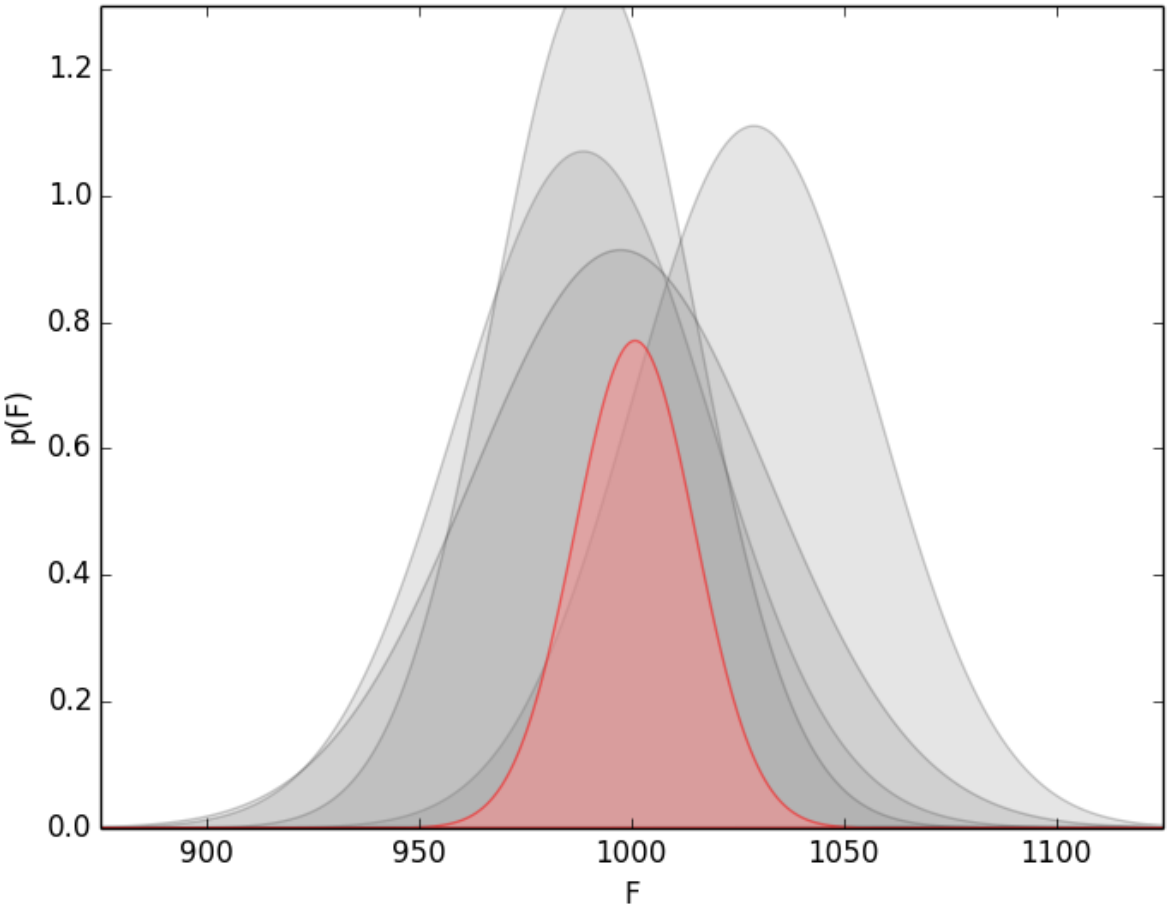




Building the Likelihood…

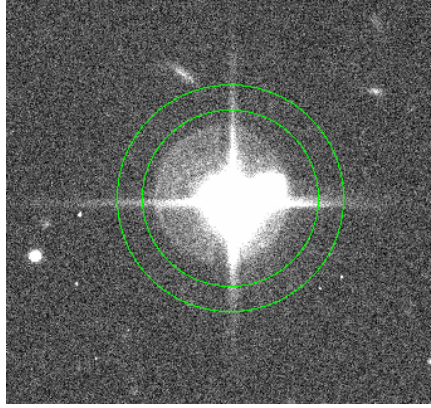
$$\mathcal{L}(D \mid F_{\text{true}}) = \prod_{i=1}^{N} P(D_i \mid F_{\text{true}})$$

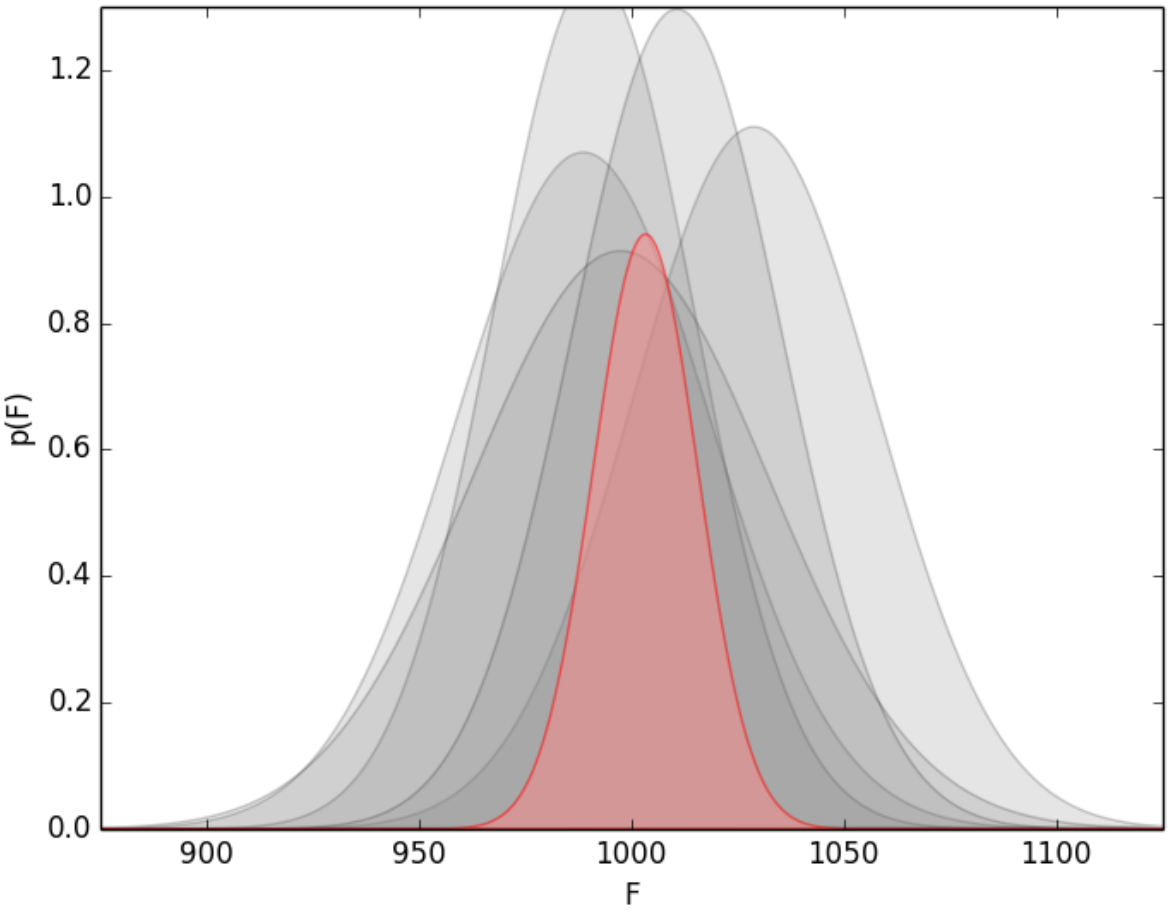




Building the Likelihood…

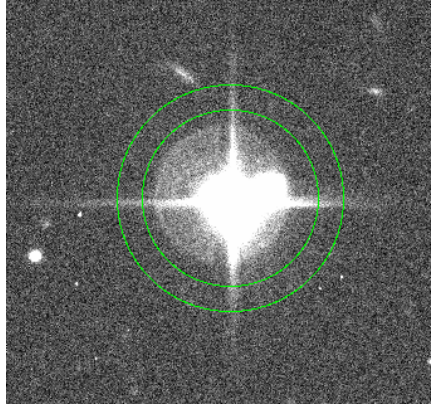
$$\mathcal{L}(D \mid F_{\text{true}}) = \prod_{i=1}^{N} P(D_i \mid F_{\text{true}})$$

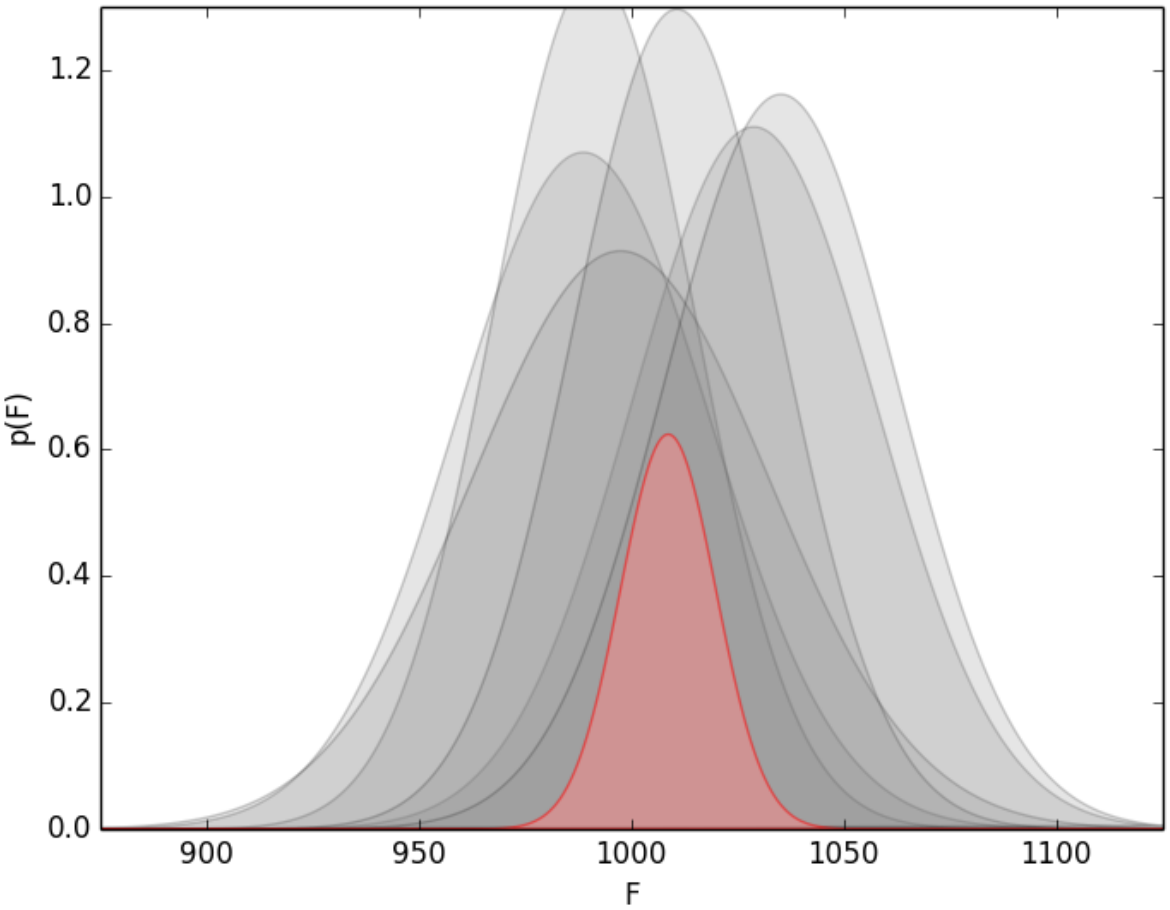




Building the Likelihood…

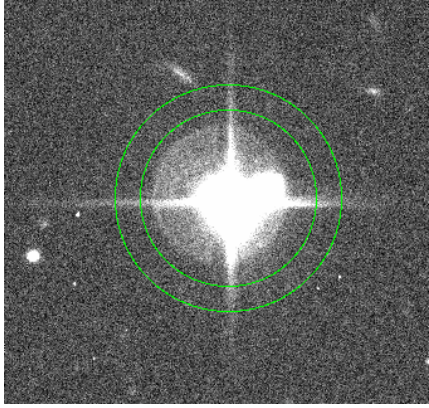
$$\mathcal{L}(D \mid F_{\text{true}}) = \prod_{i=1}^{N} P(D_i \mid F_{\text{true}})$$

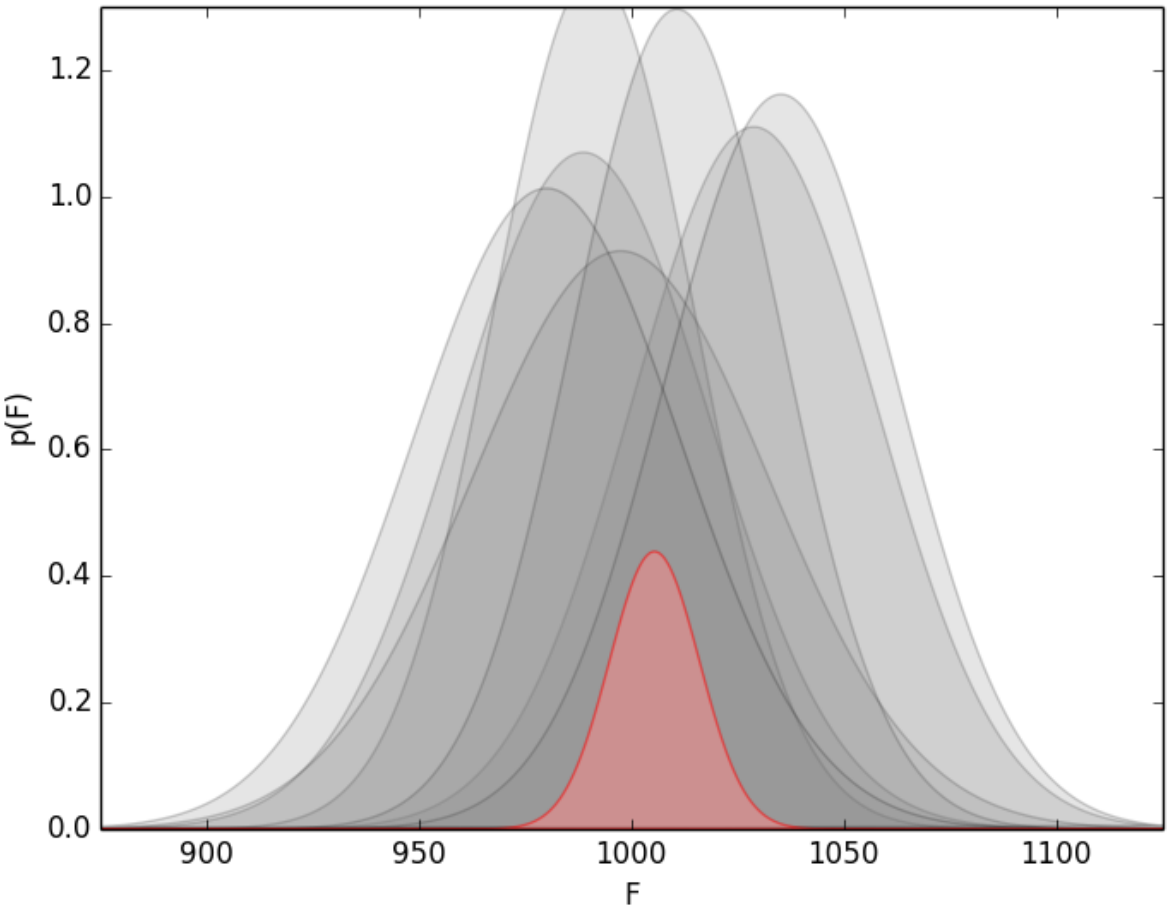




Building the Likelihood…

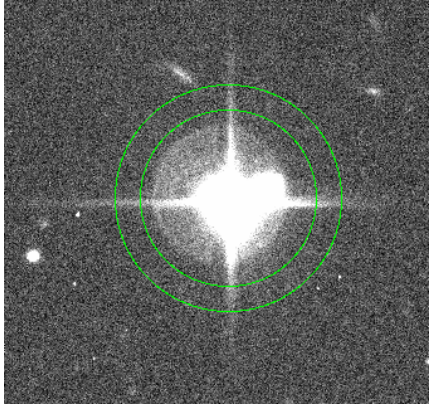
$$\mathcal{L}(D \mid F_{\text{true}}) = \prod_{i=1}^{N} P(D_i \mid F_{\text{true}})$$

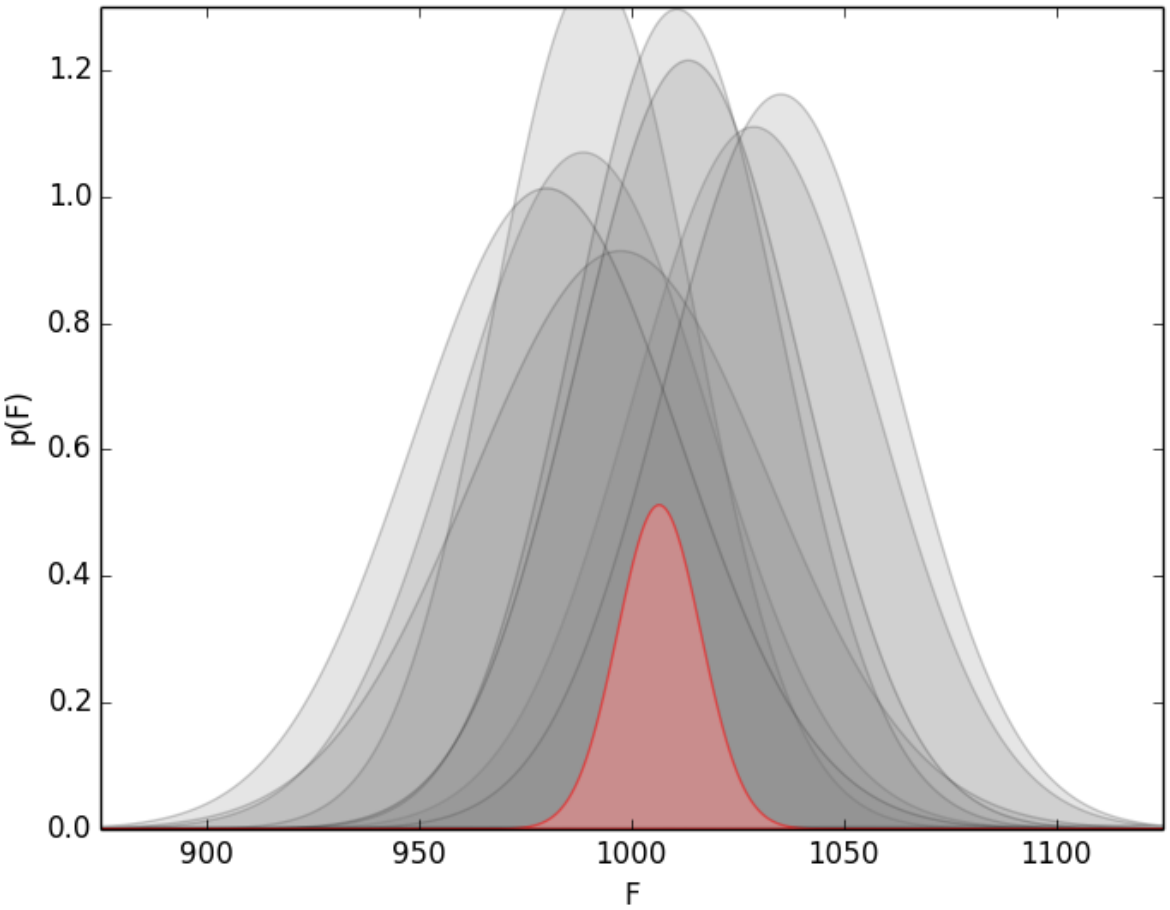




Building the Likelihood…

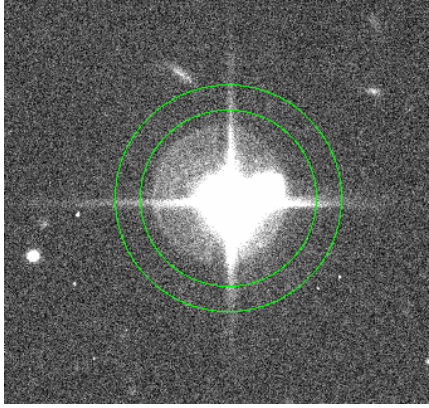
$$\mathcal{L}(D \mid F_{\text{true}}) = \prod_{i=1}^{N} P(D_i \mid F_{\text{true}})$$

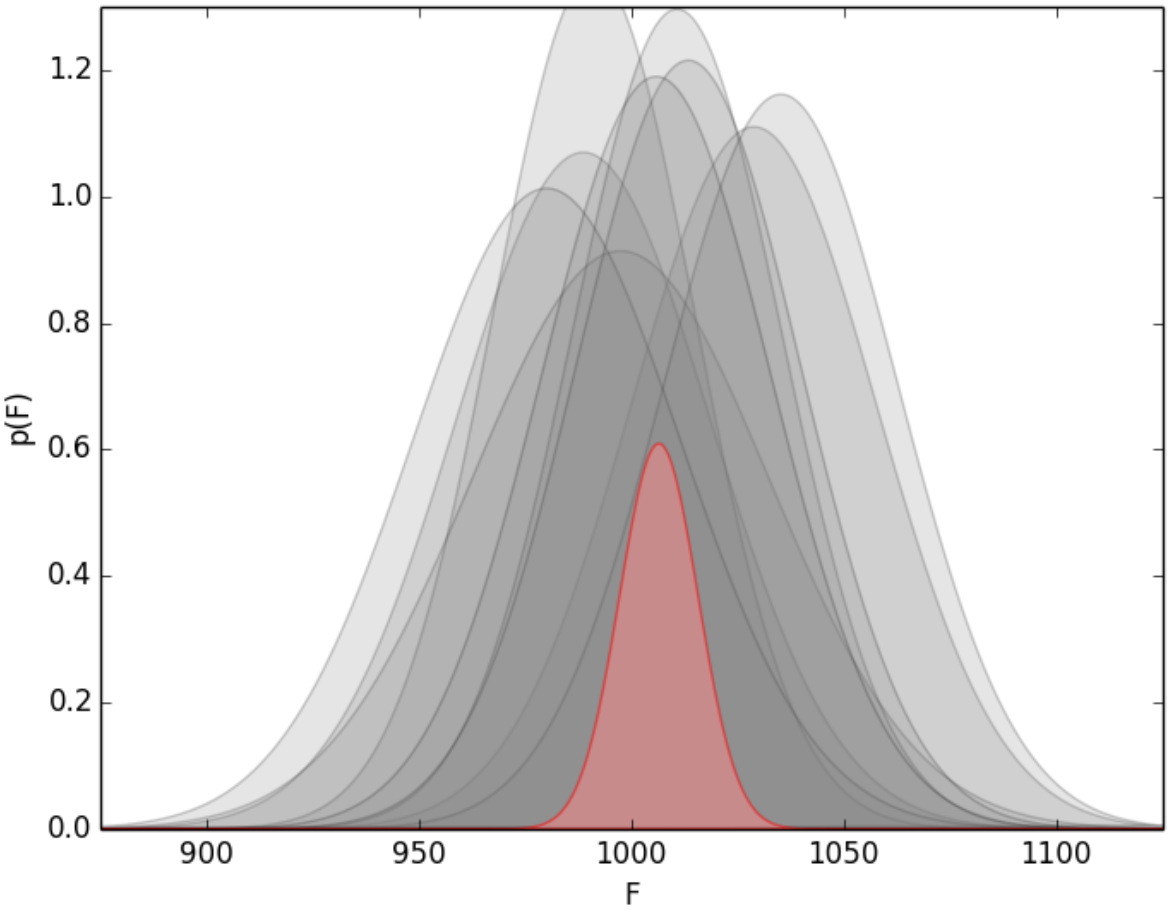




Building the Likelihood…

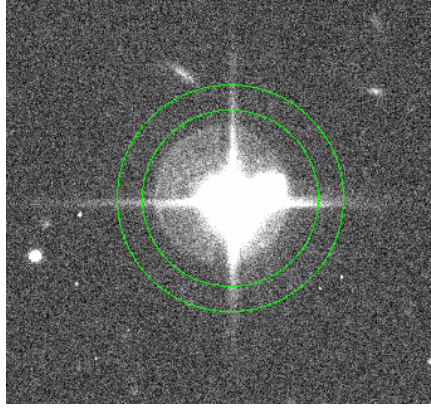
$$\mathcal{L}(D \mid F_{\text{true}}) = \prod_{i=1}^{N} P(D_i \mid F_{\text{true}})$$

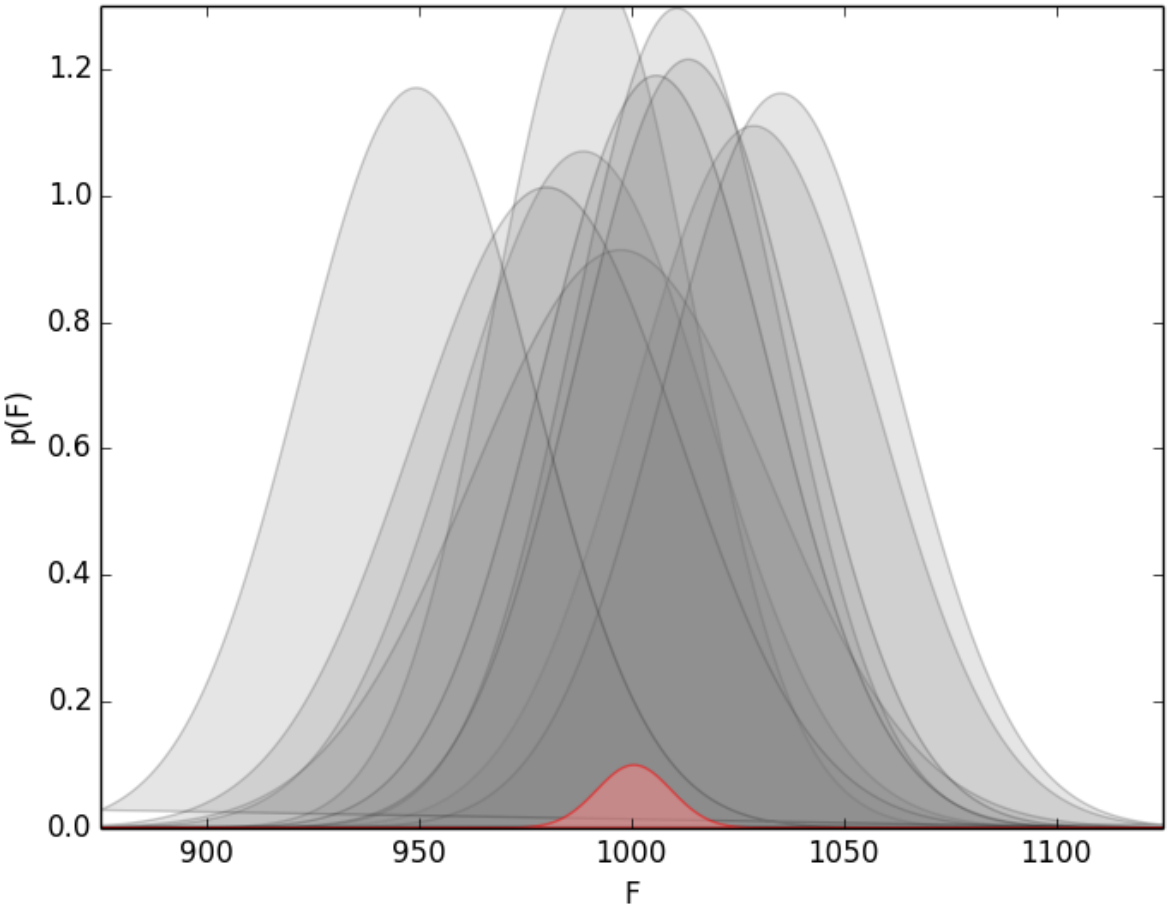




Building the Likelihood…

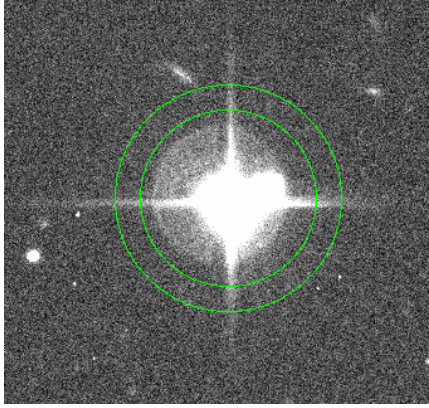
$$\mathcal{L}(D \mid F_{\text{true}}) = \prod_{i=1}^{N} P(D_i \mid F_{\text{true}})$$

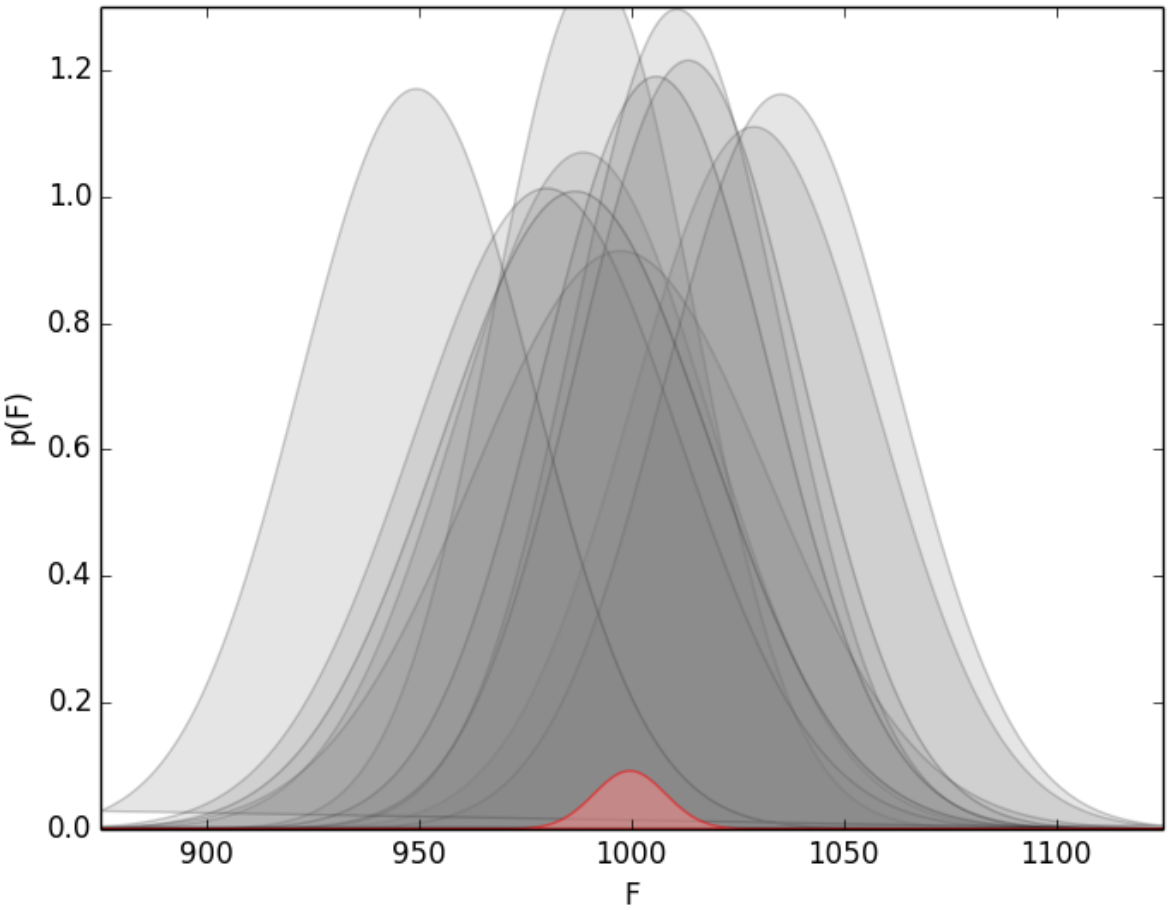




Building the Likelihood…

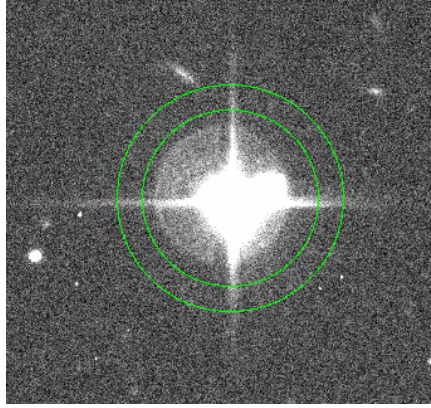
$$\mathcal{L}(D \mid F_{\text{true}}) = \prod_{i=1}^{N} P(D_i \mid F_{\text{true}})$$

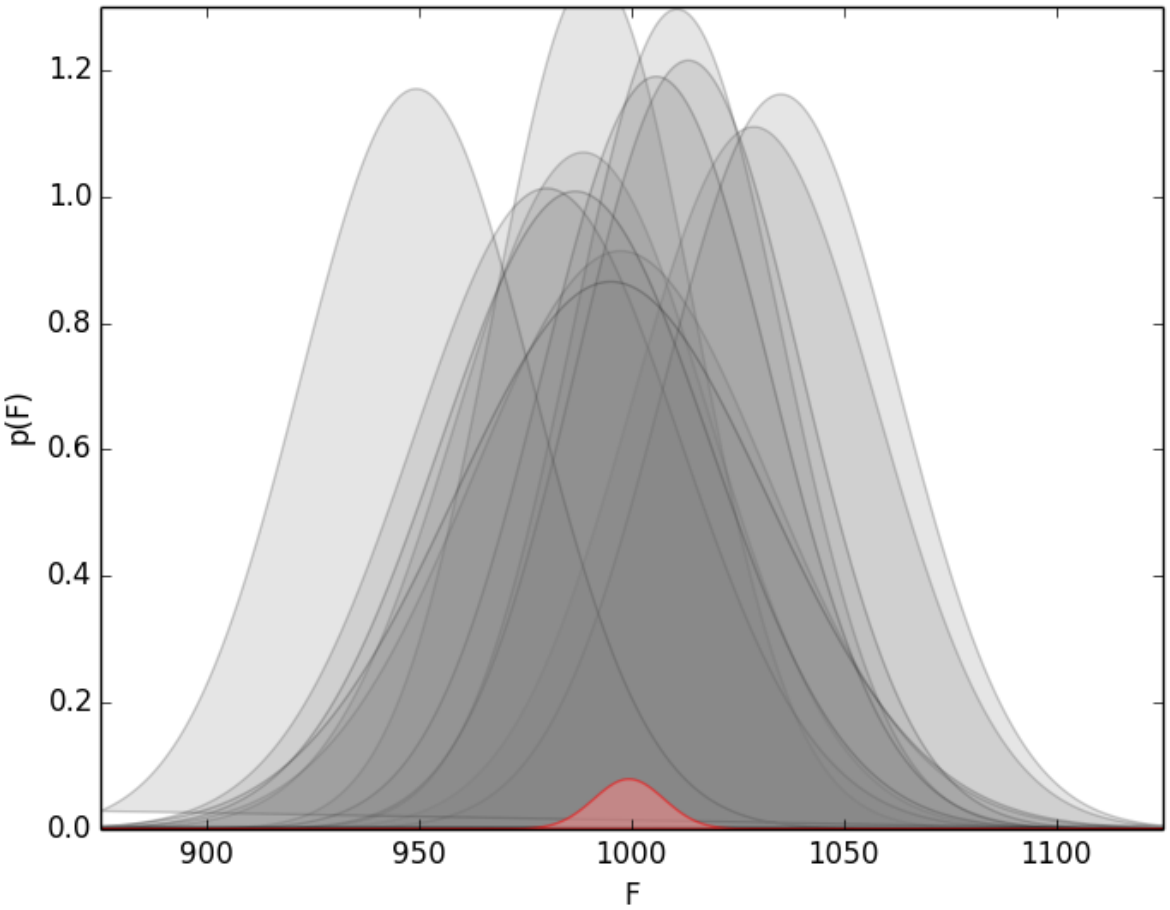




Building the Likelihood…

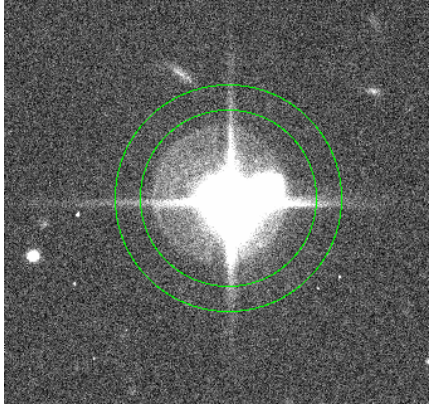
$$\mathcal{L}(D \mid F_{\text{true}}) = \prod_{i=1}^{N} P(D_i \mid F_{\text{true}})$$

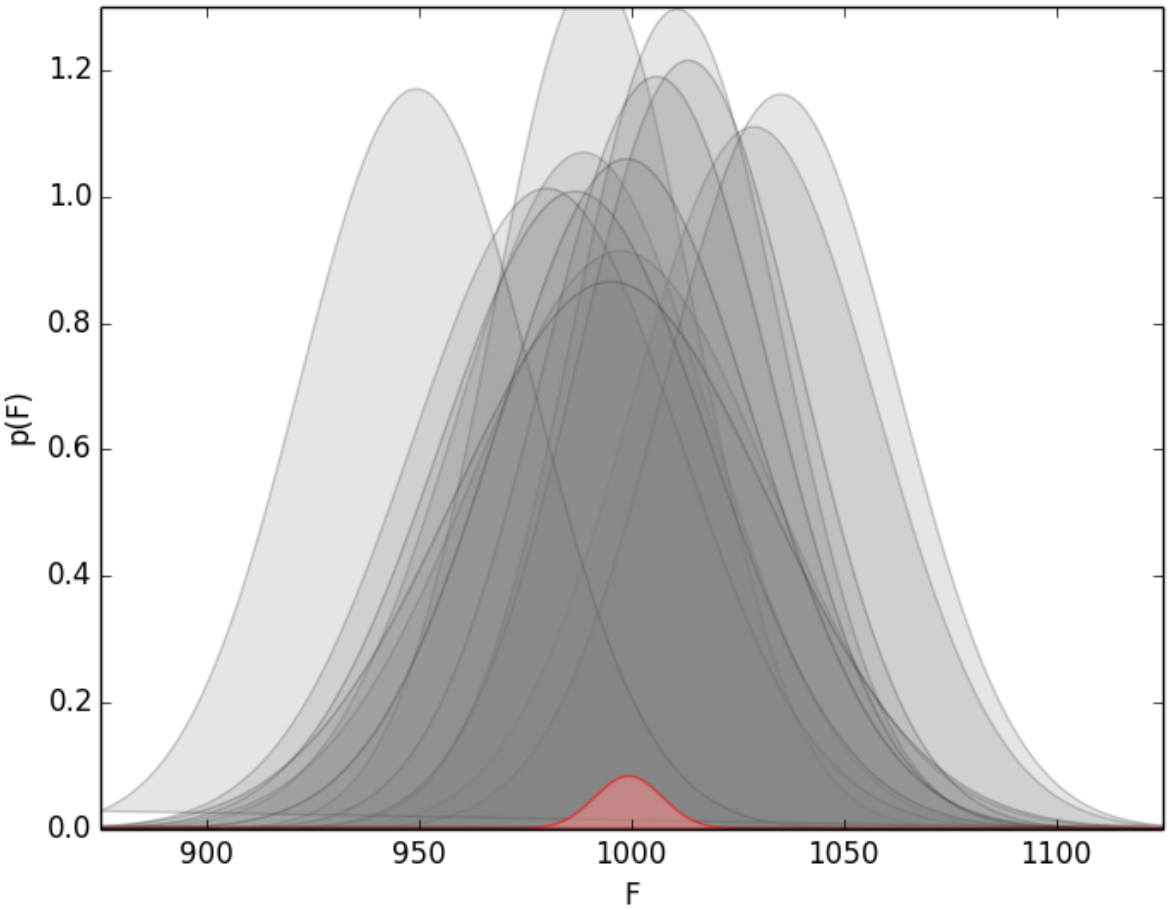




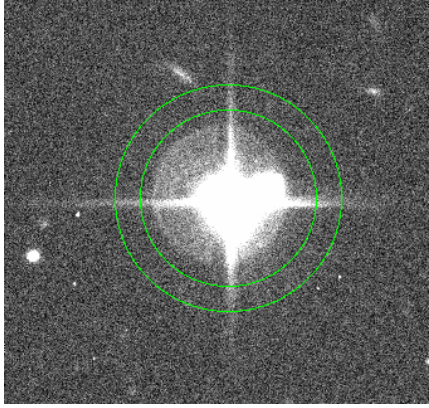
Building the Likelihood…

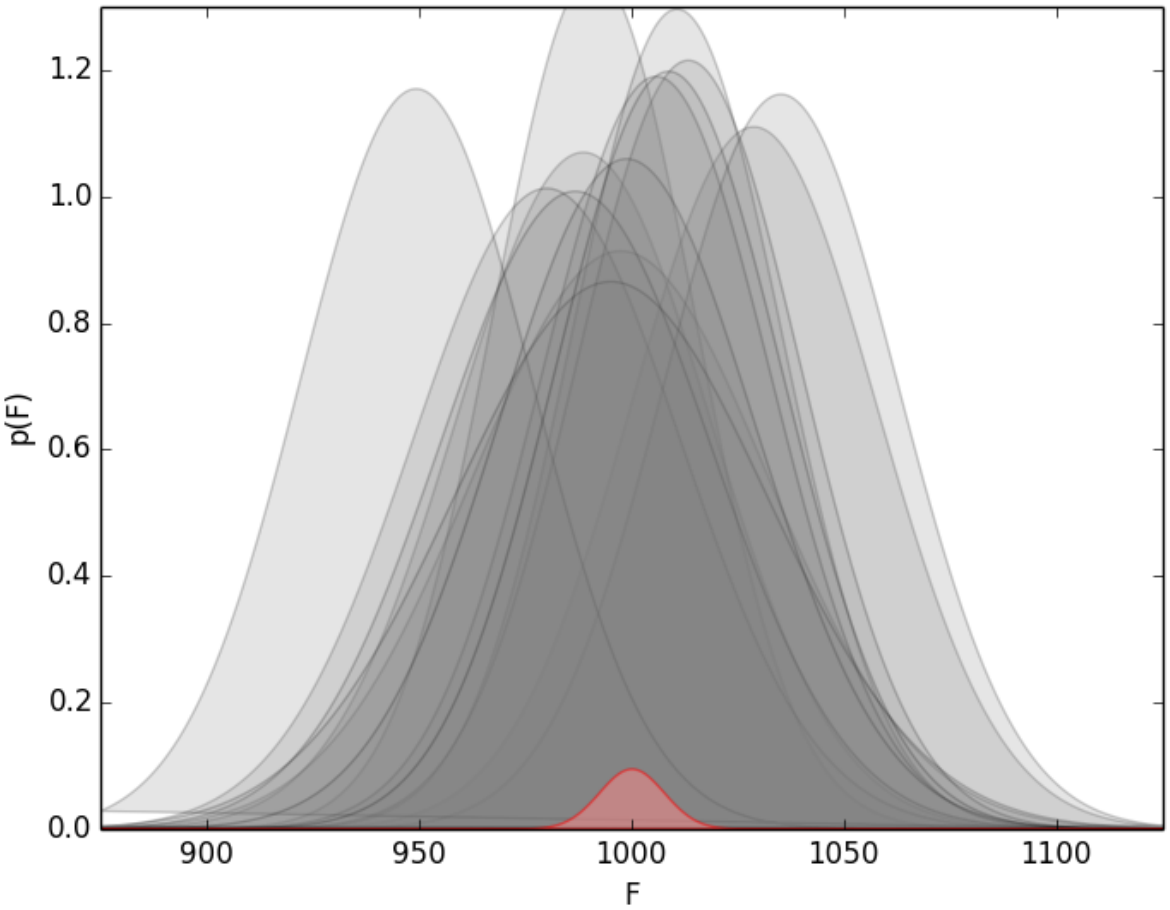
$$\mathcal{L}(D \mid F_{\text{true}}) = \prod_{i=1}^{N} P(D_i \mid F_{\text{true}})$$





“Maximum Likelihood” estimate…





Frequentist Point Estimate:

Analytically maximize $\mathcal{L}(D \mid F_{\text{true}})$ to find:

$$F_{\text{est}} = \frac{\sum w_i F_i}{\sum w_i}; \quad w_i = 1/e_i^2$$

$$\sigma_{\text{est}} = \left(\sum_{i=1}^{N} w_i \right)^{-1/2}$$

Frequentist Point Estimate:

Analytically maximize $\mathcal{L}(D \mid F_{\text{true}})$ to find:

$$F_{\text{est}} = \frac{\sum w_i F_i}{\sum w_i}; \quad w_i = 1/e_i^2$$

$$\sigma_{\text{est}} = \left(\sum_{i=1}^{N} w_i \right)^{-1/2}$$

In Python:

```
w = 1. / e ** 2
best_guess = (w * F).sum() / w.sum()
error = 1. / np.sqrt(w.sum())
```

For our 30 data points, we have 999 +/- 4

Bayesian Approach: Posterior Probability

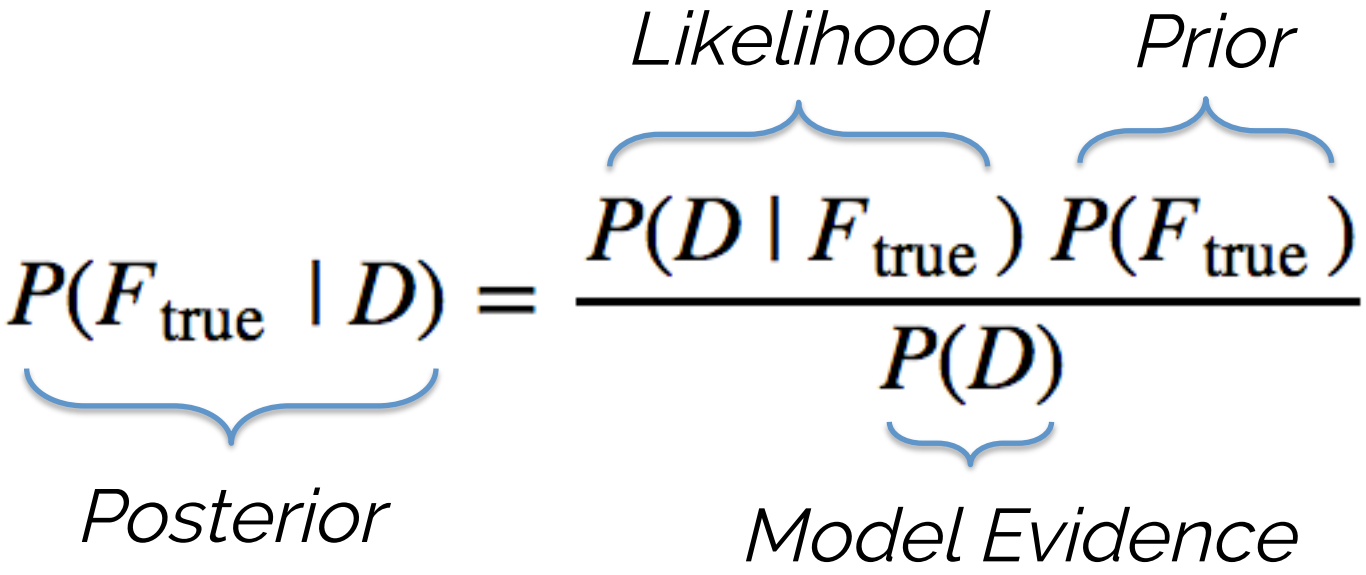
Compute our knowledge of F given the data, encoded as a probability:

$$P(F_{\text{true}} \mid D)$$

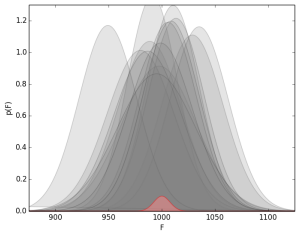
To compute this, we use Bayes' Theorem

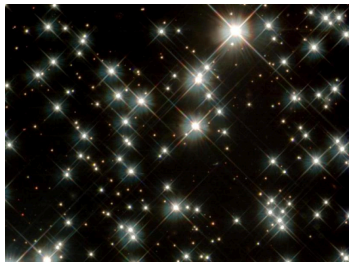
$$P(F_{\text{true}} \mid D) = \frac{P(D \mid F_{\text{true}})\, P(F_{\text{true}})}{P(D)}$$

Bayes' Theorem



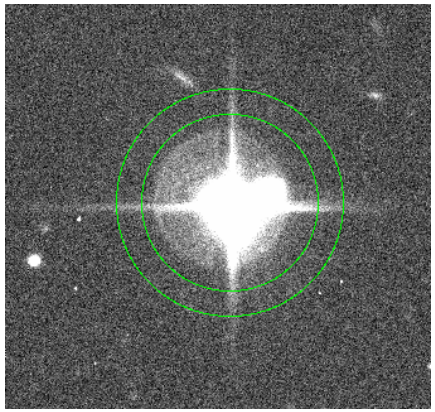
Bayes' Theorem





Likelihood *Prior*

$$\underbrace{P(F_{\text{true}} \mid D)}_{\text{Posterior}} = \frac{P(D \mid F_{\text{true}})\, P(F_{\text{true}})}{\underbrace{P(D)}_{\text{Model Evidence}}}$$



(Often simply a normalization, but useful for model evaluation, etc.)

Again, we find 999 +/- 4

For very simple problems, frequentist & Bayesian results are often practically indistinguishable

The difference becomes apparent in more complicated situations...

- Handling of nuisance parameters
- Interpretation of Uncertainty
- Incorporation of prior information
- Comparison & evaluation of Models
- etc.

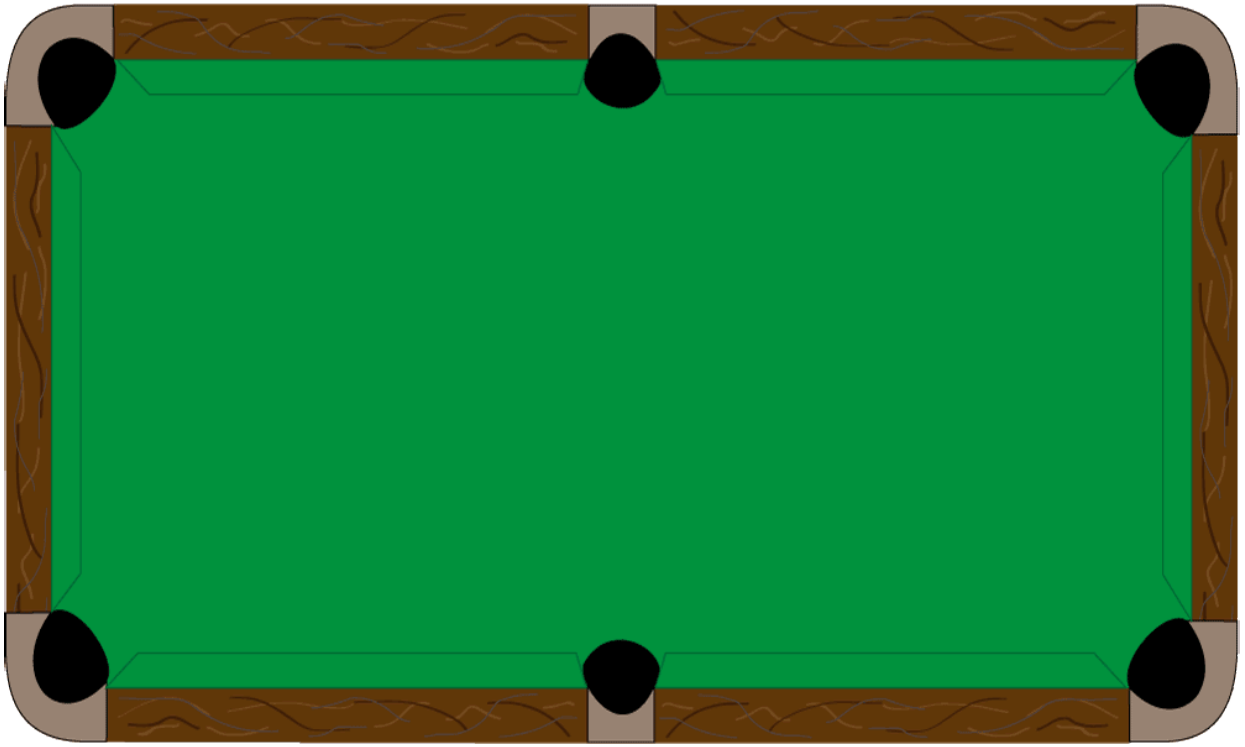
The difference becomes apparent in more complicated situations...

- Handling of nuisance parameters
- Interpretation of Uncertainty
- Incorporation of prior information
- Comparison & evaluation of Models
- etc.

Example 1: Nuisance Parameters

Nuisance Parameters: Bayes' Billiard Game

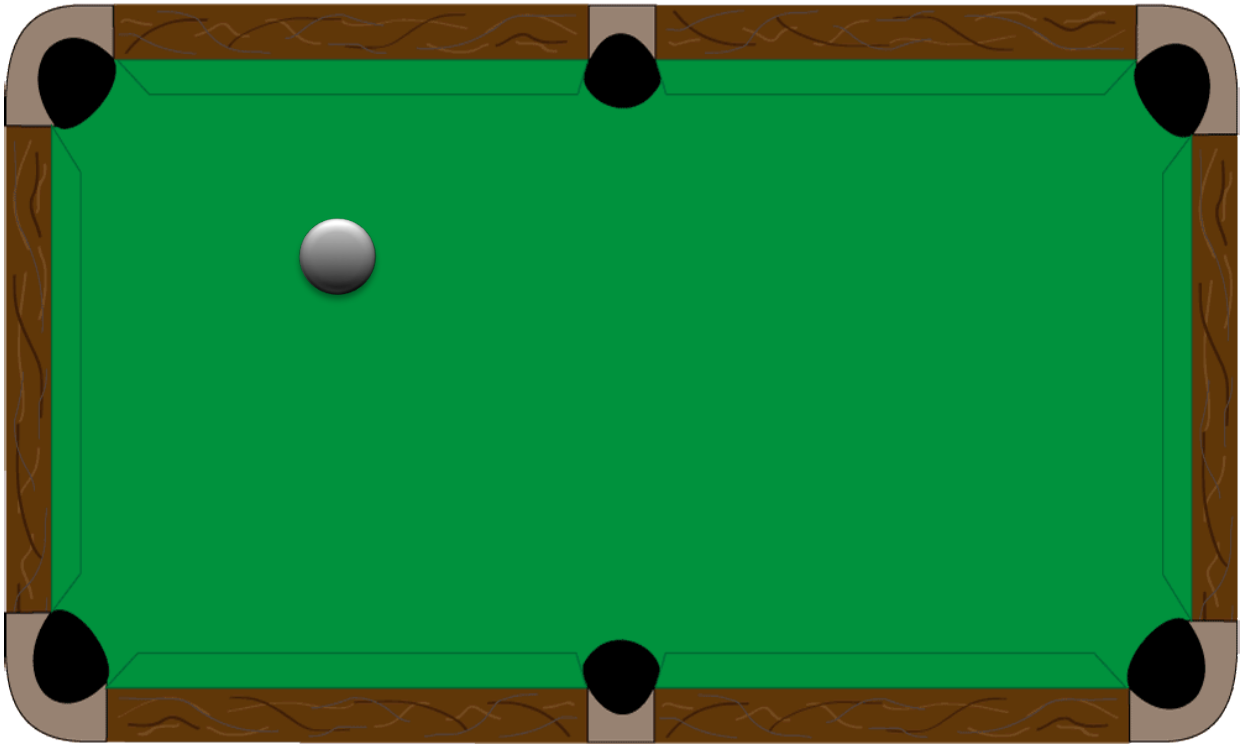
Alice and Bob have a gambling problem…



Bayes 1763
Eddy 2004

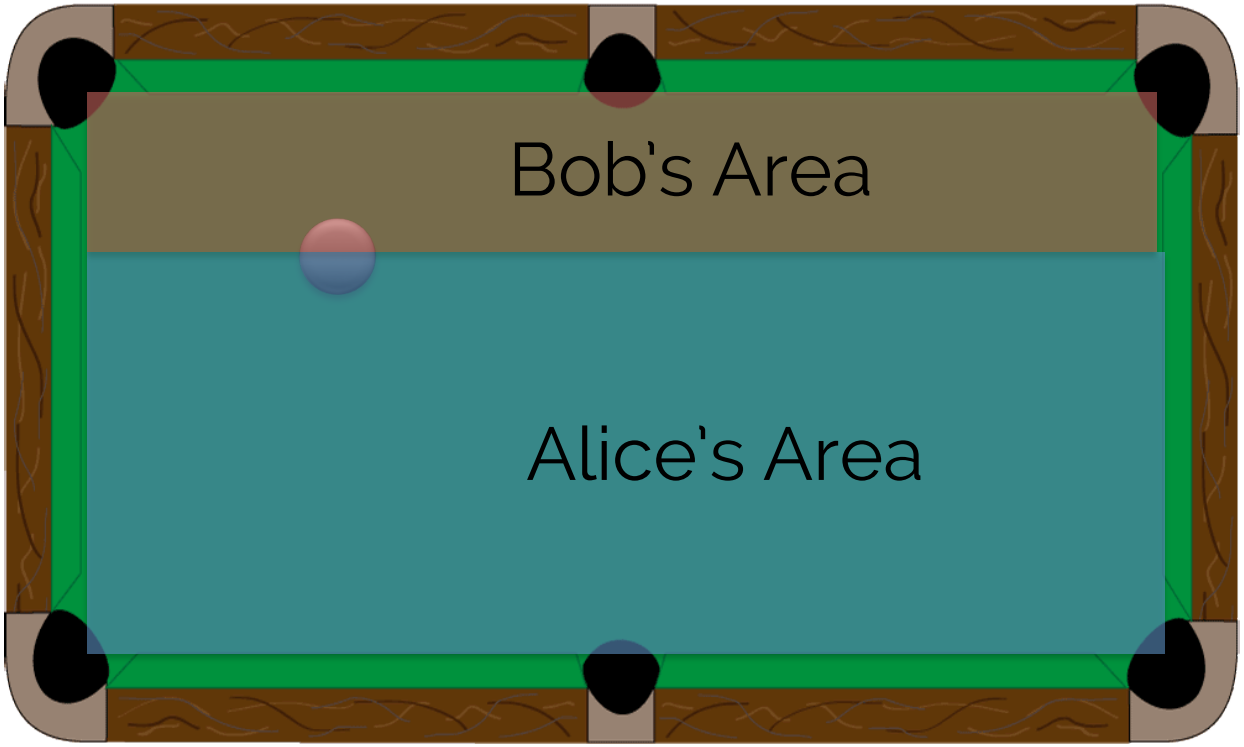
Nuisance Parameters: Bayes' Billiard Game

Carol has designed a game
for them to play…



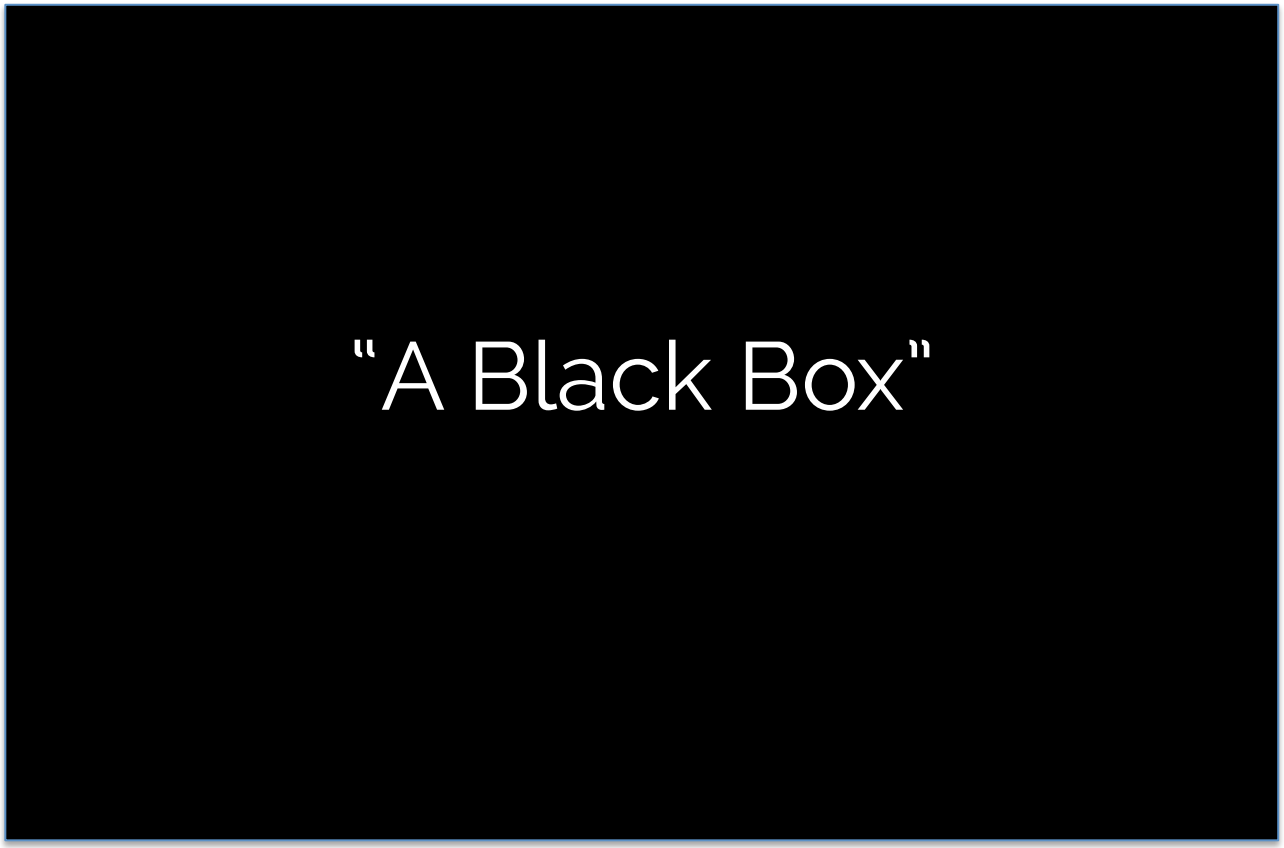
Nuisance Parameters: Bayes' Billiard Game

- The first ball divides the table
- Additional balls give a point to A or B
- First person to six points wins



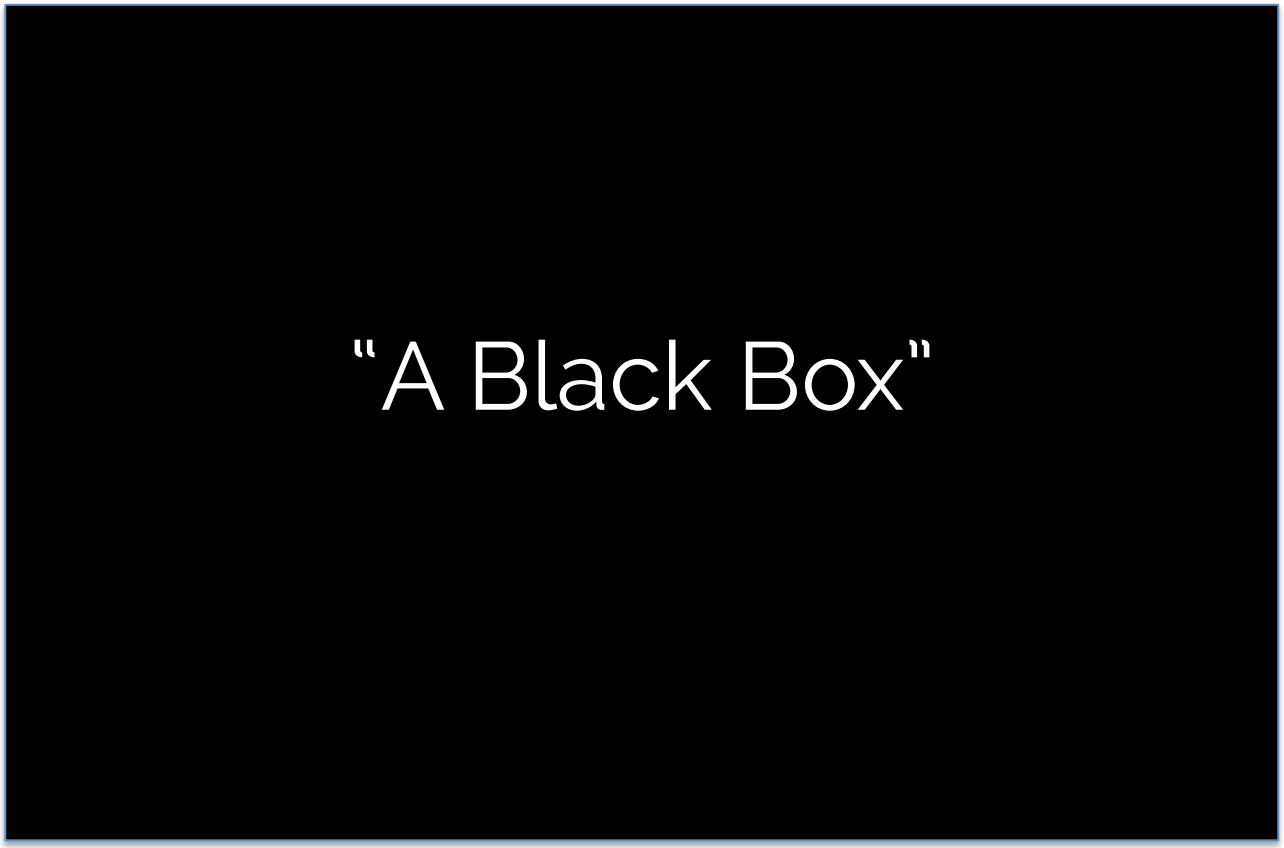
Nuisance Parameters: Bayes' Billiard Game

- The first ball divides the table
- Additional balls give a point to A or B
- First person to six points wins



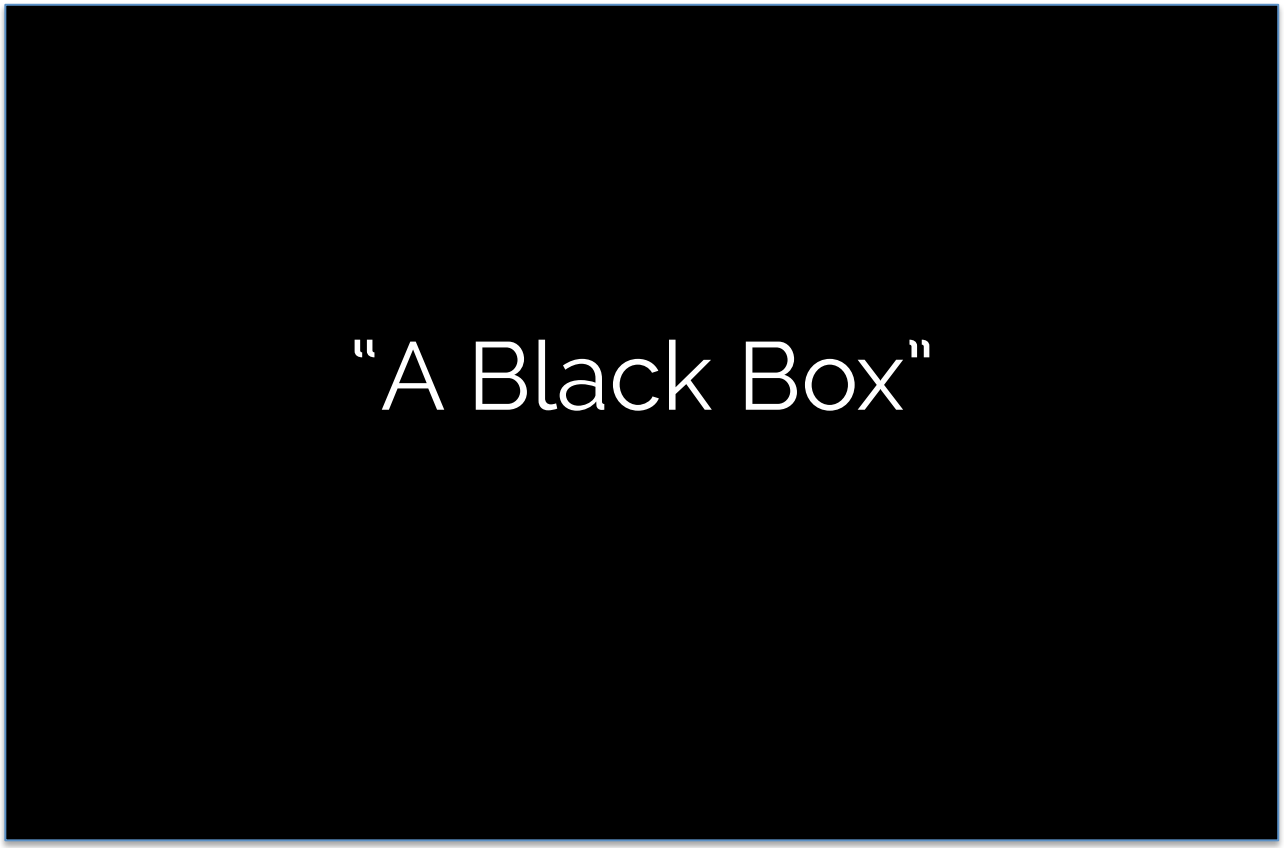
Nuisance Parameters: Bayes' Billiard Game

Question: in a certain game, Alice has 5 points and Bob has 3. What are the odds that Bob will go on to win?

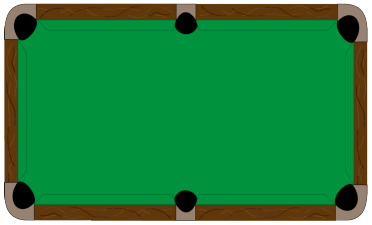


Nuisance Parameters: Bayes' Billiard Game

Note: the division of the table is a **nuisance parameter**: a parameter which affects the problem and must be accounted for, but is not of immediate interest.



A Frequentist Approach



p = probability of Alice winning any roll
(nuisance parameter)

Maximum likelihood estimate gives

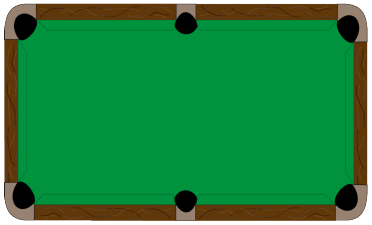
$$\hat{p} = 5/8$$

Probability of Bob winning (he needs 3 points):

$$P(B) = (1 - \hat{p})^3$$

P(B) = 0.053; Odds of 18 to 1 against

A Bayesian Approach



Marginalization:

$$P(B|D) \equiv \int_{-\infty}^{\infty} P(B, p|D)\mathrm{d}p$$

B = Bob wins
D = observed data

Some algebraic manipulation…

$$P(B|D) = \int P(B|p, D)P(p|D)dp$$

$$P(B|D) = \int P(B|p, D)\frac{P(D|p)P(p)}{P(D)}dp$$

Find $P(B|D)$ = 0.091; odds of 10 to 1 against

Bayes' Billiard Game Results:

Frequentist: 18 to 1 odds

Bayesian: 10 to 1 odds

Bayes' Billiard Game Results:

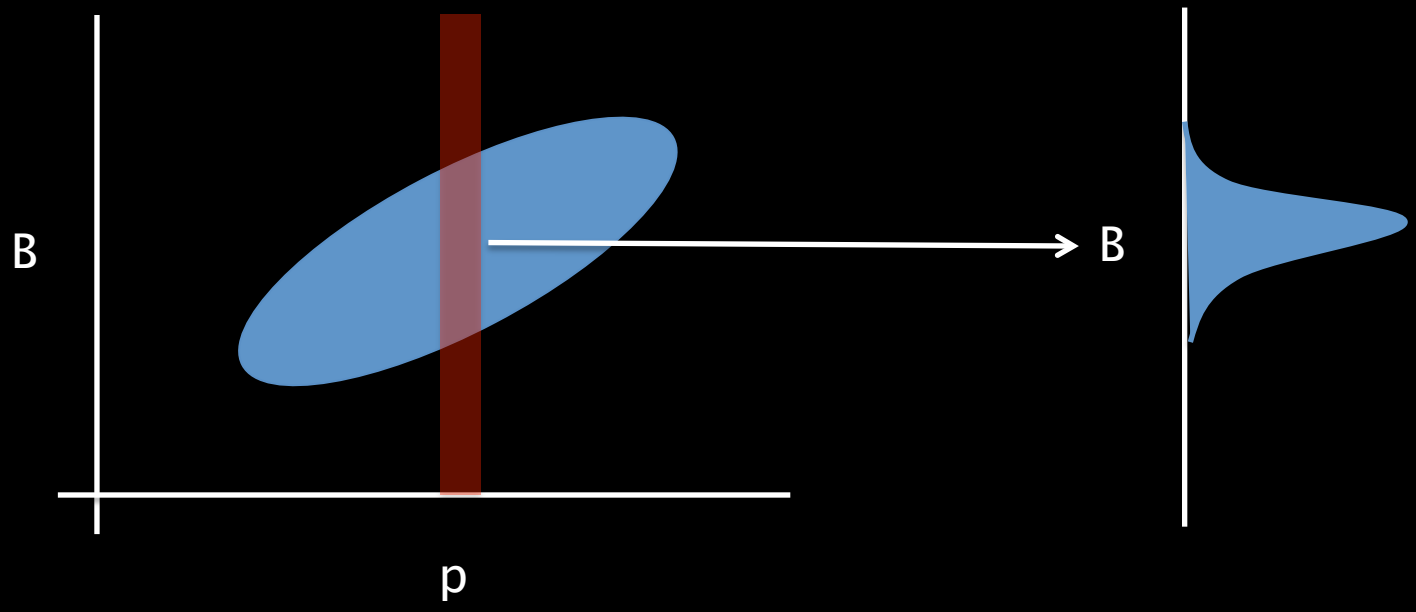
Frequentist: 18 to 1 odds

Bayesian: 10 to 1 odds

Difference: Bayes approach allows nuisance parameters to vary, through *marginalization.*

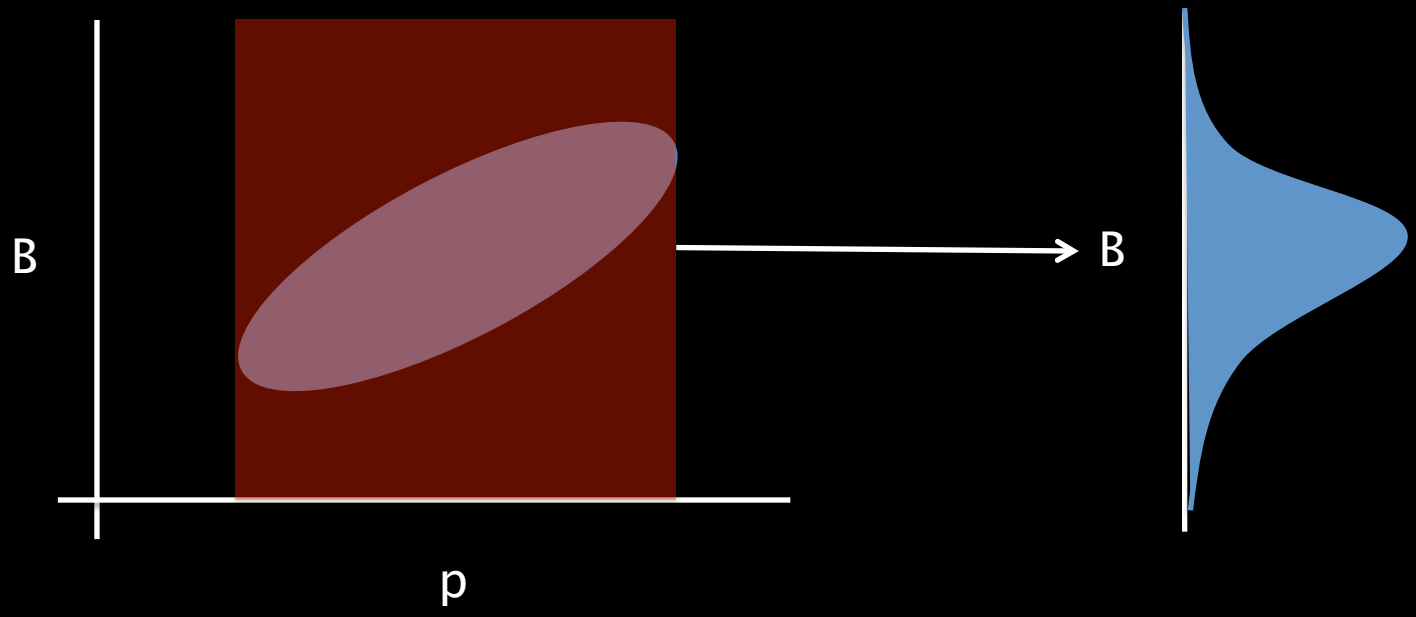
Conditioning vs. Marginalization

Conditioning (akin to Frequentist here)



Conditioning vs. Marginalization

Marginalization (Bayesian approach here)



Example 2: Uncertainties

Uncertainties: “Confidence” vs “Credibility”

“If this experiment is repeated many times, in 95% of these cases the computed confidence interval will contain the true θ.”

- Frequentists

Uncertainties: “Confidence” vs “Credibility”

“If this experiment is repeated many times, in 95% of these cases the computed confidence interval will contain the true θ.”

- Frequentists

“Given our observed data, there is a 95% probability that the value of θ lies within the credible region”.

- Bayesians

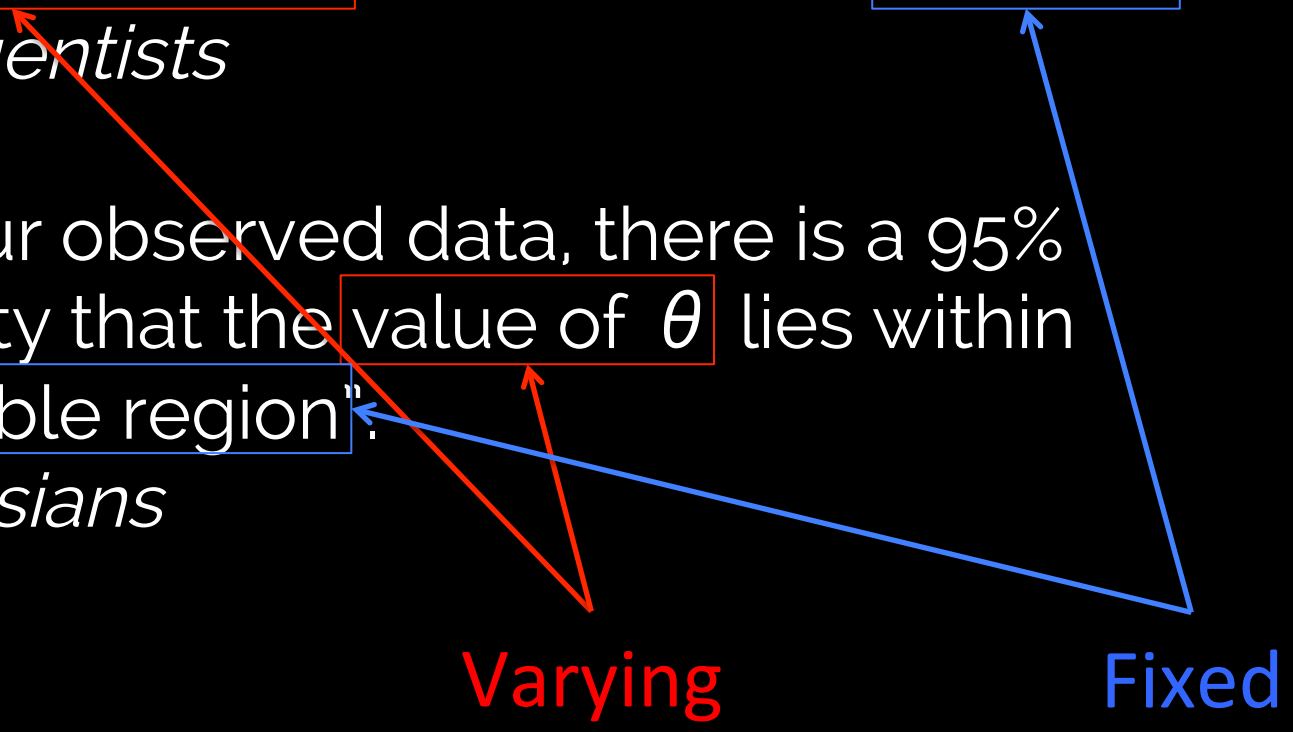
Uncertainties: "Confidence" vs "Credibility"

"If this experiment is repeated many times, in 95% of these cases the computed confidence interval will contain the true θ."

- Frequentists

"Given our observed data, there is a 95% probability that the value of θ lies within the credible region".

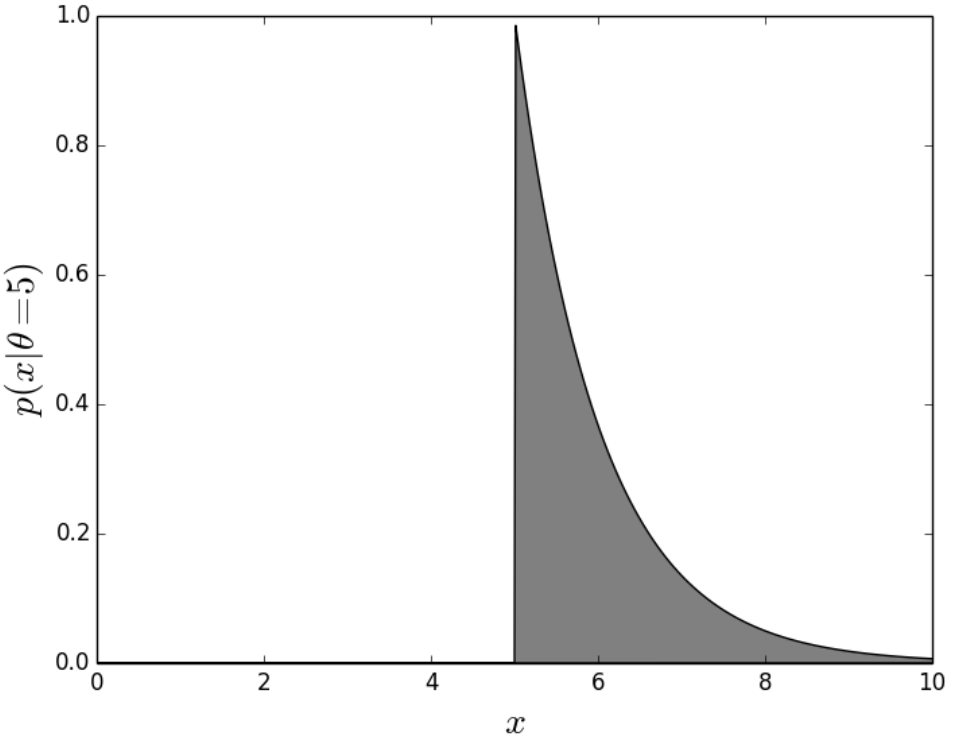
- Bayesians



Uncertainties: Jaynes' Truncated Exponential

Consider a model: $p(x \mid \theta) = \left\{ \begin{array}{ll} \exp(\theta - x) & , \quad x > \theta \\ 0 & , \quad x < \theta \end{array} \right\}$

We observe D = {10, 12, 15}
What are the 95% bounds on Θ?



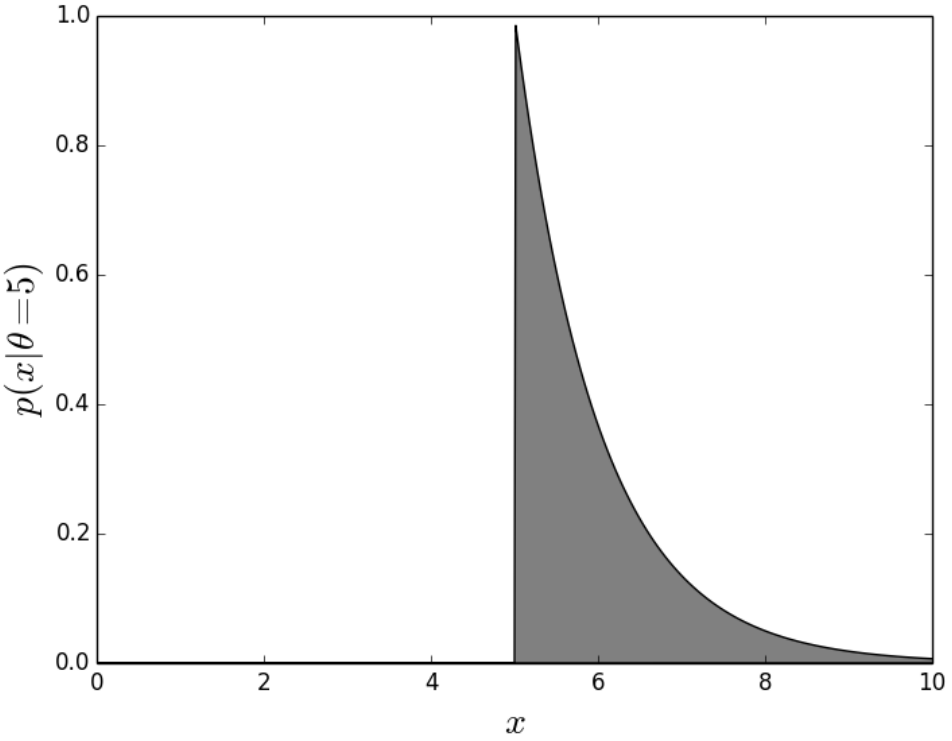
Jaynes 1976

Common-sense Approach

D = {10, 12, 15}

Each point must be greater than Θ, and the smallest observed point is x = 10.

Therefore we can immediately write the common-sense bound Θ < 10



Frequentist Approach

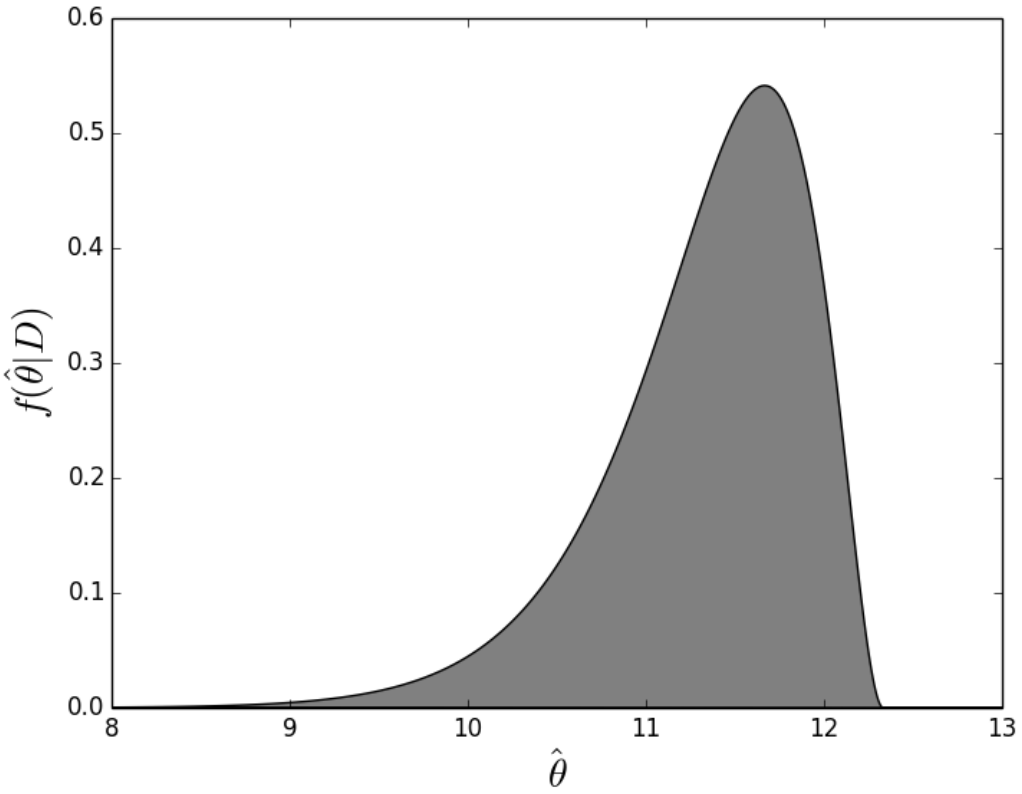
The expectation of x is:

$$E(x) = \int_0^\infty xp(x)dx = \theta + 1$$

So an unbiased estimator is:

$$\hat{\theta} = \frac{1}{N}\sum_{i=1}^{N} x_i - 1$$

Now we compute the *sampling distribution of the mean* for p(x):



Frequentist Approach

The expectation of x is:

$$E(x) = \int_0^\infty$$

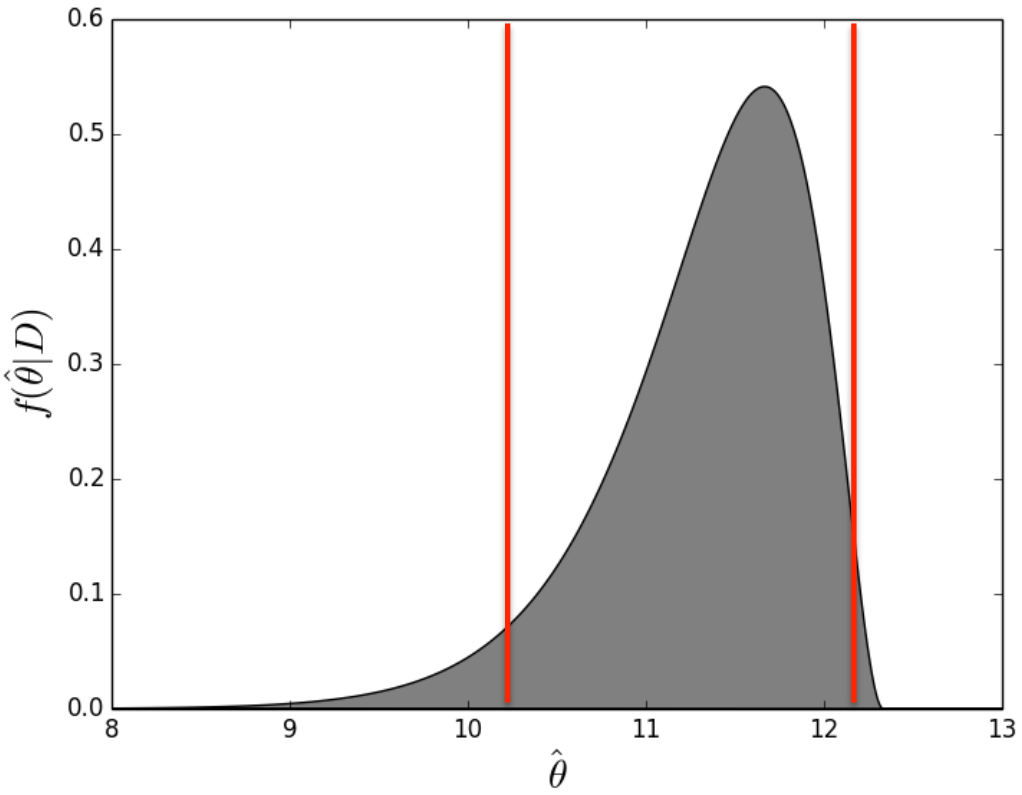
So an unbiased

$$\hat{\theta} = \frac{1}{N}\sum_{i=1}^{N} x_i - 1$$

Now we compute the *sampling distribution of the mean* for p(x):

95% confidence interval:

$10.2 < \Theta < 12.2$



Bayesian Approach

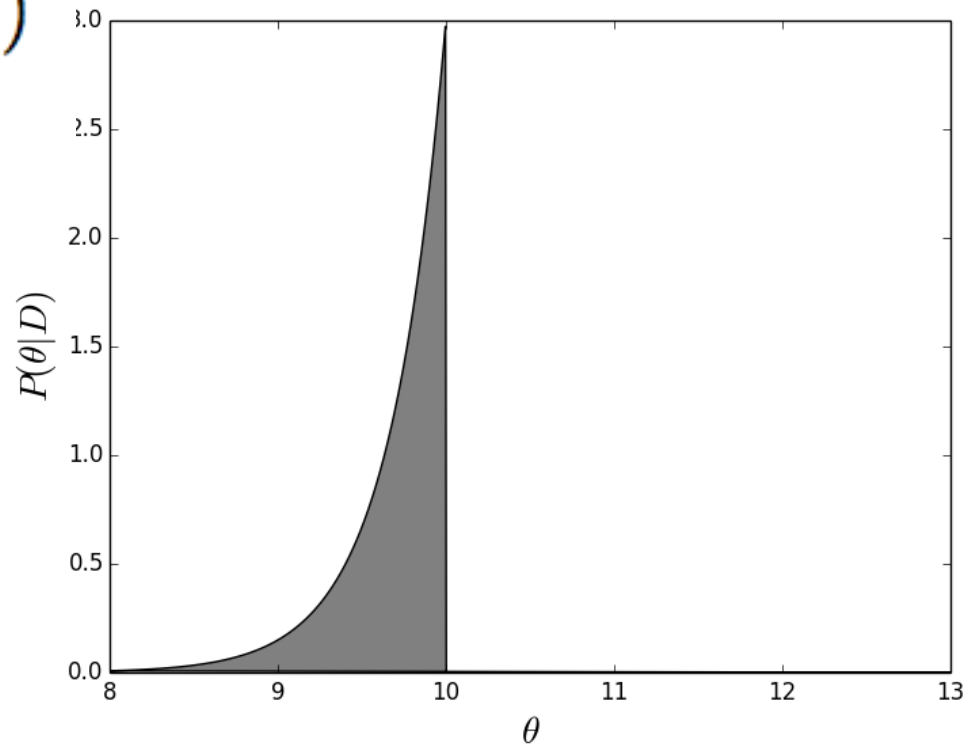
Bayes' Theorem:

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

Likelihood:

$$P(D|\theta) \propto \prod_{i=1}^{N} P(x_i|\theta)$$

With a flat prior, we get this posterior:



Bayesian Approach

Bayes' Theorem:

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

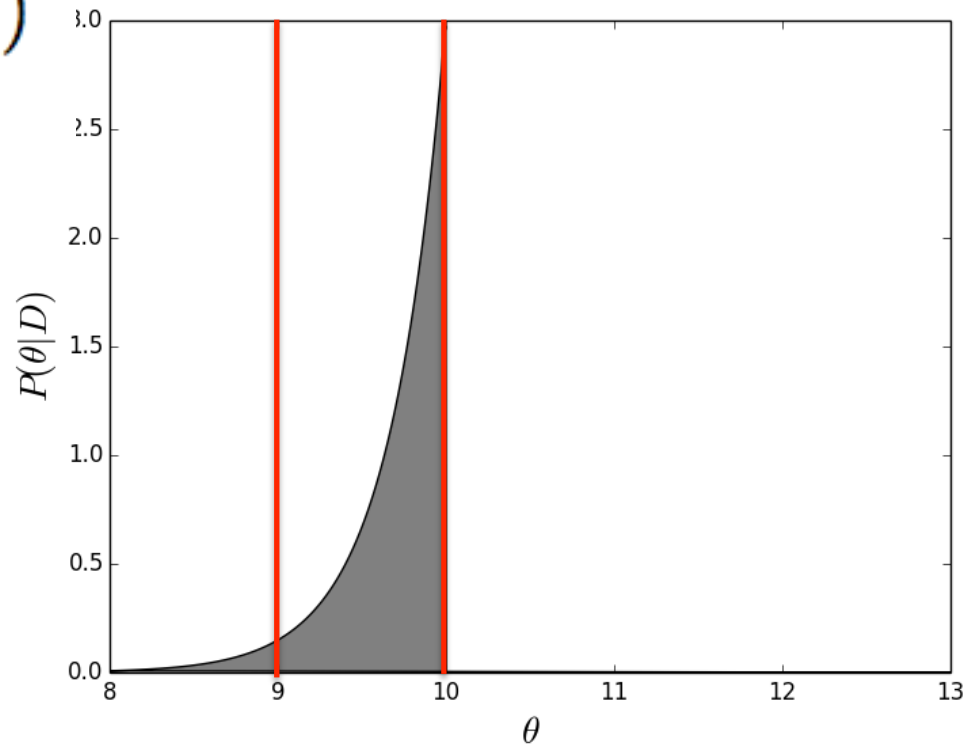
95% credible region:

$9.0 < \Theta < 10.0$

Likelihood:

$$P(D|\theta) \propto \prod_{i=1}^{N} P(x_i|\theta)$$

With a flat prior, we get this posterior:



Jaynes' Truncated Exponential Results:

Common Sense Bound:

$\Theta < 10$

Frequentist unbiased 95% confidence interval:

$10.2 < \Theta < 12.2$

Bayesian 95% credible region:

$9.0 < \Theta < 10.0$

Frequentism is not wrong!

It's just answering a different question than we might expect.

Confidence vs. Credibility

Bayesianism: probabilisitic statement about *model parameters* given a *fixed credible region*

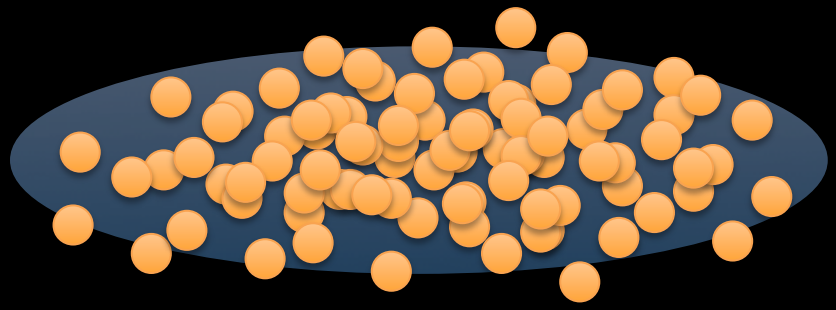
Frequentism: probabilistic statement about a *recipe for generating confidence intervals* given a *fixed model parameter*

Confidence vs. Credibility

= Parameter

= Interval

Bayesian Credible Region:

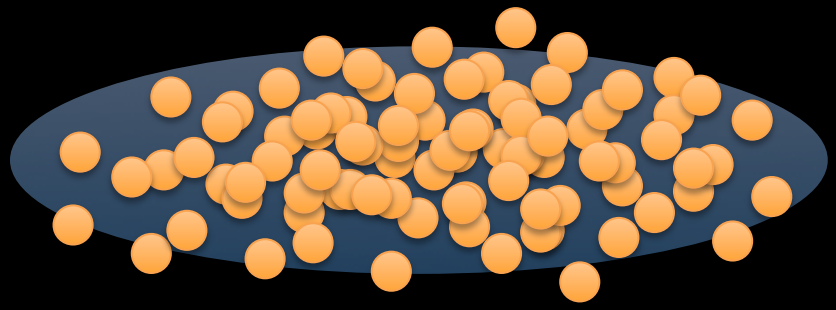


Confidence vs. Credibility

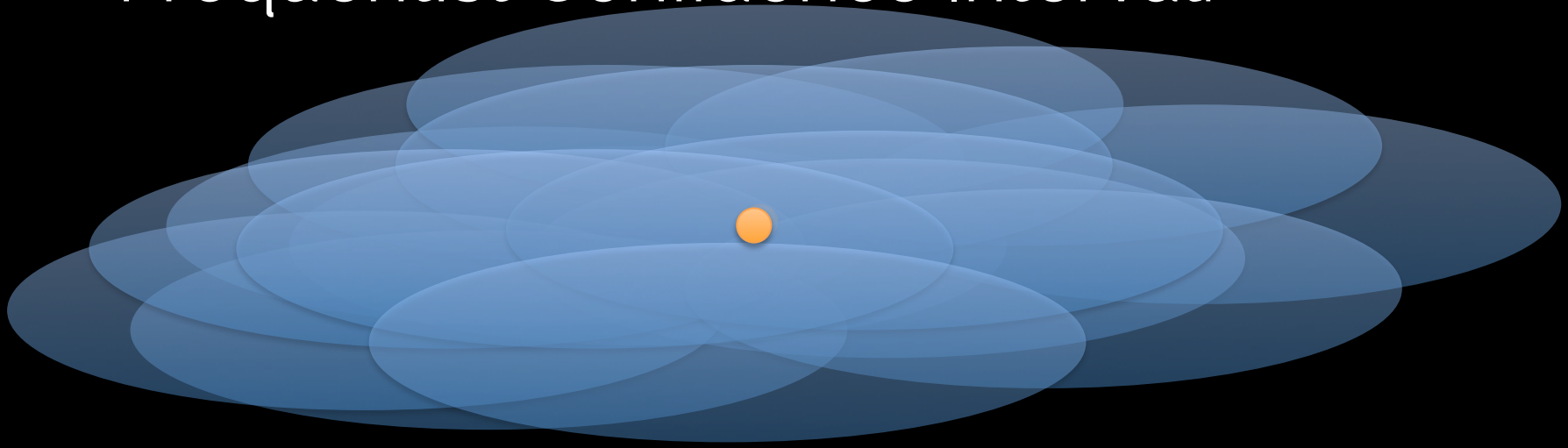
= Parameter

= Interval

Bayesian Credible Region:



Frequentist Confidence Interval:

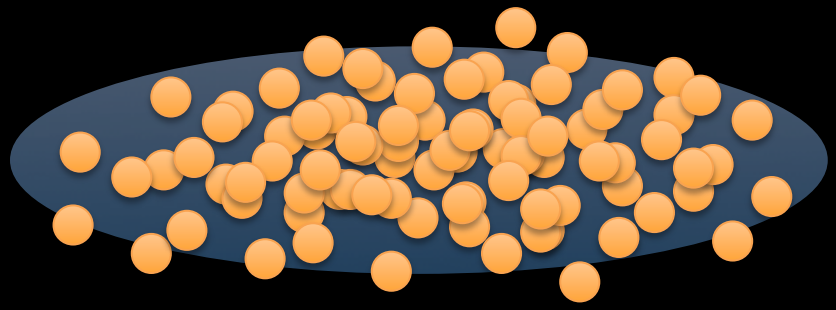


Confidence vs. Credibility

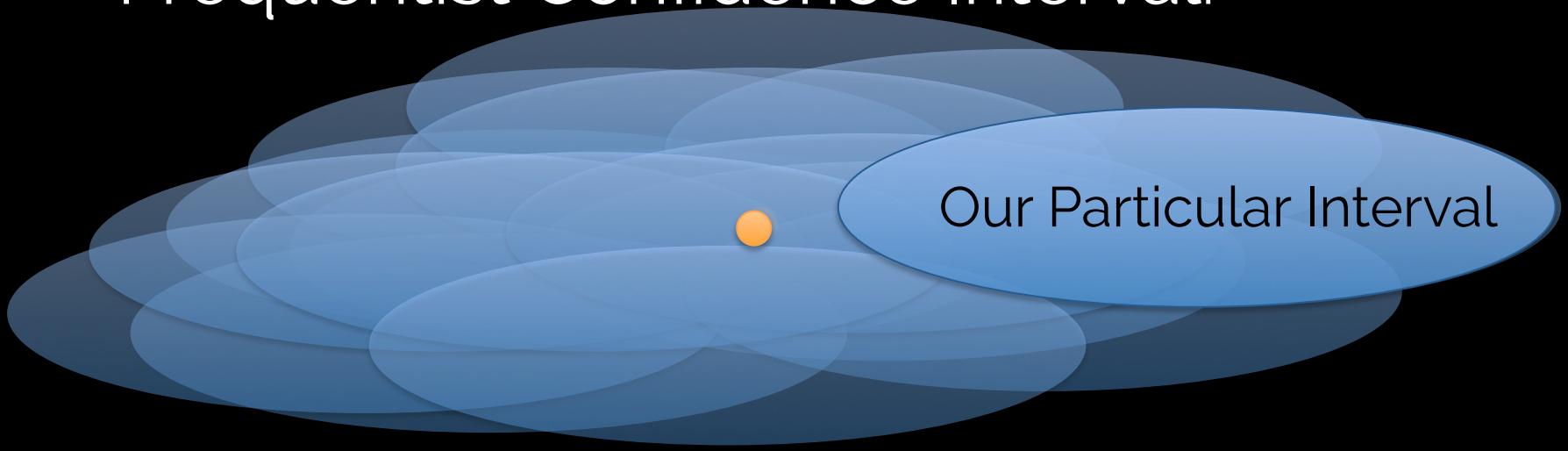
= Parameter

= Interval

Bayesian Credible Region:



Frequentist Confidence Interval:



Please Remember This:

In general, a frequentist 95% Confidence Interval is *not* 95% likely to contain the true value!

This very common mistake is a *Bayesian interpretation* of a *frequentist construct*.

Typical Conversation:

Statistician: “95% of such confidence intervals in repeated experiments will contain the true value”

Scientist: “So there’s a 95% chance that the value is in this interval?”

Typical Conversation:

Statistician: “No: you see, parameters by definition can’t vary, so referring to chance in that context is meaningless. The 95% refers to the interval itself.”

Scientist: “Oh, so there’s a 95% chance that the value is in this interval?”

Typical Conversation:

Statistician: “No. It’s this: the long-term limiting frequency of the procedure for constructing this interval ensures that 95% of the resulting ensemble of intervals contains the value.

Scientist: “Ah, I see: so there’s a 95% chance that the value is in this interval, right?”

Typical Conversation:

Statistician: “No… it’s that… well… just write down what I said, OK?”

Scientist: “OK, got it. The value is 95% likely to be in the interval.”

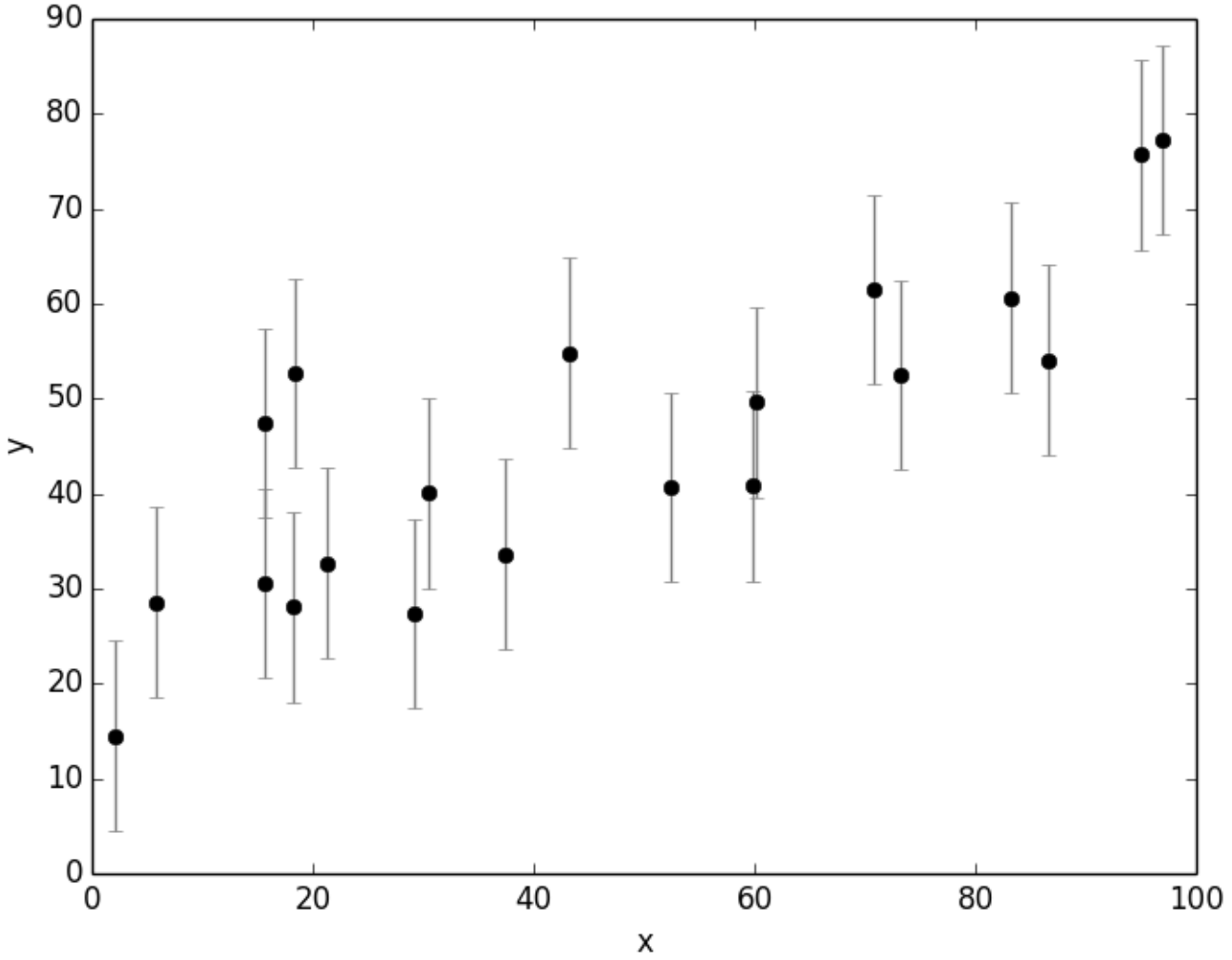
(Editorial aside…)

Non-statisticians naturally understand uncertainty in a Bayesian manner.

Wouldn't it be less confusing if we simply used Bayesian methods?

A more practical example...

Final Example: Line of Best Fit



Final Example: Line of Best Fit

The Model:

$$\hat{y}_i = \alpha + \beta x_i$$

$$P(x_i, y_i \mid \alpha, \beta, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[\frac{-(y_i - \hat{y}_i)^2}{2\sigma^2}\right]$$

Bayesian Approach uses Bayes' Theorem:

$$P(\alpha, \beta, \sigma \mid D) = \frac{P(D \mid \alpha, \beta, \sigma) P(\alpha, \beta, \sigma)}{P(D)}$$

Final Example: Line of Best Fit

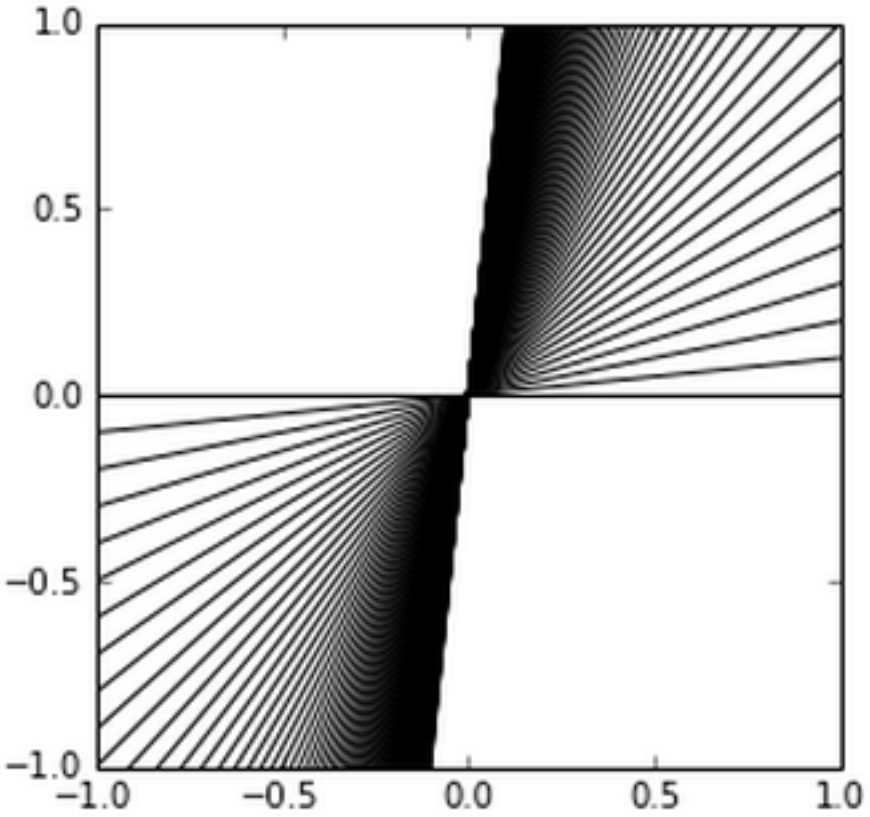
The Prior: $P(\alpha, \beta, \sigma)$

Is a flat prior on the slope appropriate?

Final Example: Line of Best Fit

The Prior: $P(\alpha, \beta, \sigma)$
Is a flat prior on the slope appropriate?

No!



Final Example: Line of Best Fit

By symmetry arguments, we can motivate the following *uninformative prior:*

$$P(\alpha, \beta, \sigma) \propto \frac{1}{\sigma}(1+\beta^2)^{-3/2}$$

Or equivalently, a flat prior on these:

$$\theta = \tan^{-1}\beta$$

$$\alpha_\perp = \alpha\cos\theta$$

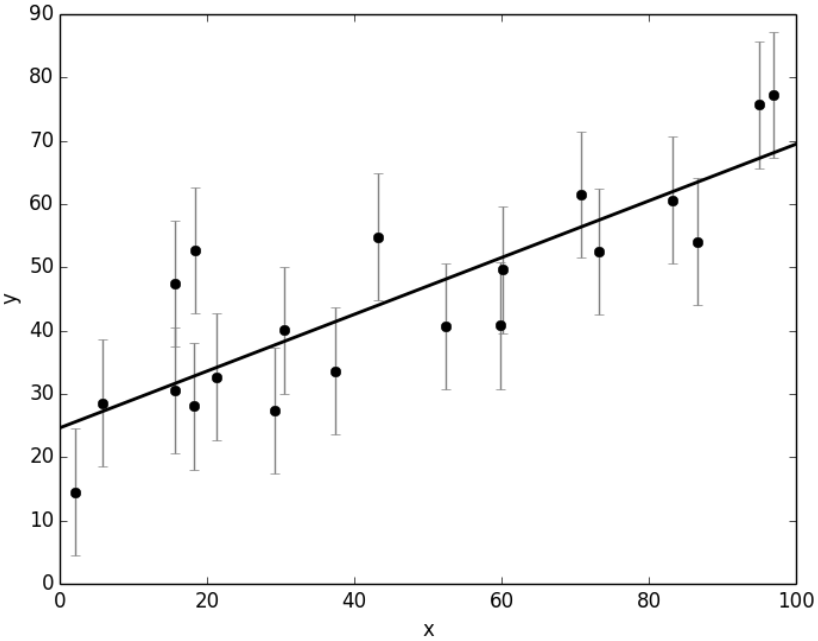
$$ln\sigma = \log(\sigma)$$

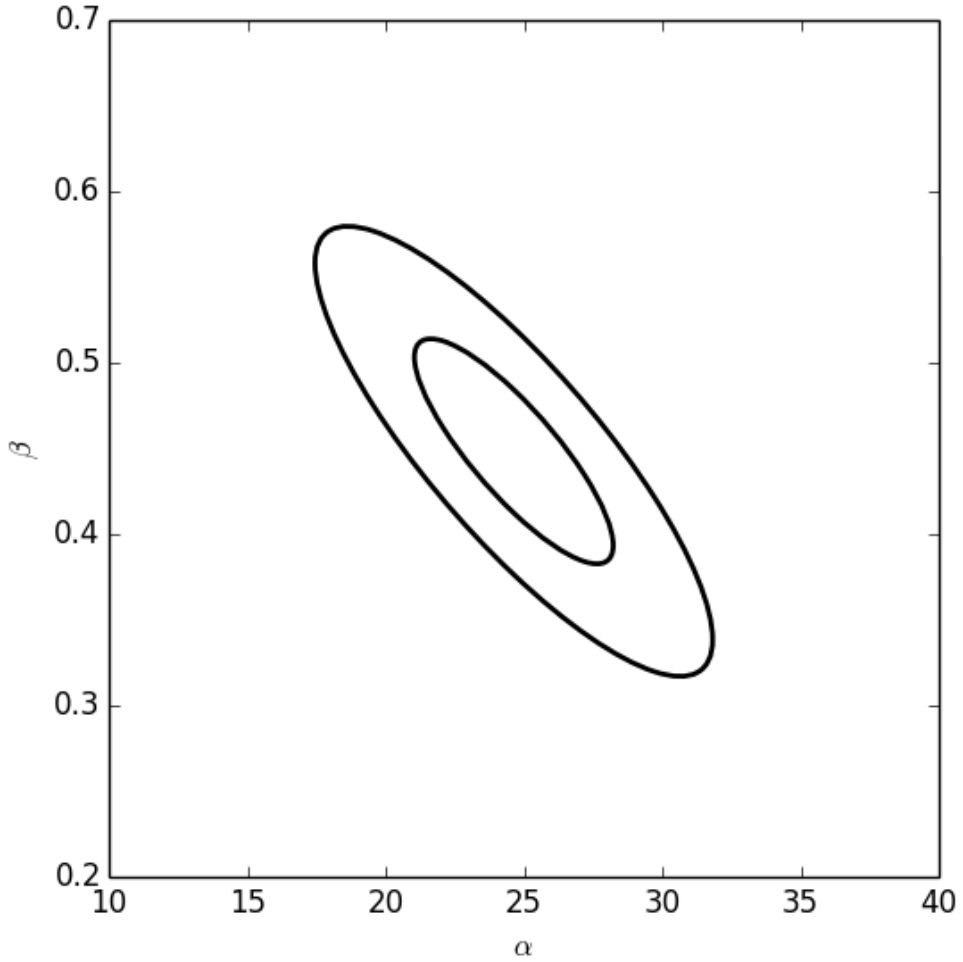
Frequentist Result: StatsModels

```
import statsmodels.api as sm
X = sm.add_constant(xdata)
result = sm.OLS(ydata, X).fit()
print(result.summary2())
```

```
            Results: Ordinary least squares
=================================================================
Model:                OLS    AIC:                     147.7737
Dependent Variable:   y      BIC:                     149.7651
No. Observations:     20     Log-Likelihood:          -71.887
Df Model:             1      F-statistic:             41.97
Df Residuals:         18     Prob (F-statistic):      4.30e-06
R-squared:            0.700  Scale:                   86.157
Adj. R-squared:       0.683
-----------------------------------------------------------------
          Coef.   Std.Err.    t     P>|t|    [0.025    0.975]
-----------------------------------------------------------------
const    24.6361    3.7871  6.5053  0.0000  16.6797  32.5924
x1        0.4483    0.0692  6.4782  0.0000   0.3029   0.5937
-----------------------------------------------------------------
Omnibus:              1.996     Durbin-Watson:          2.758
Prob(Omnibus):        0.369     Jarque-Bera (JB):       1.634
Skew:                 0.651     Prob(JB):               0.442
Kurtosis:             2.486     Condition No.:          100
=================================================================
```

frequentist





Bayesian Result: emcee

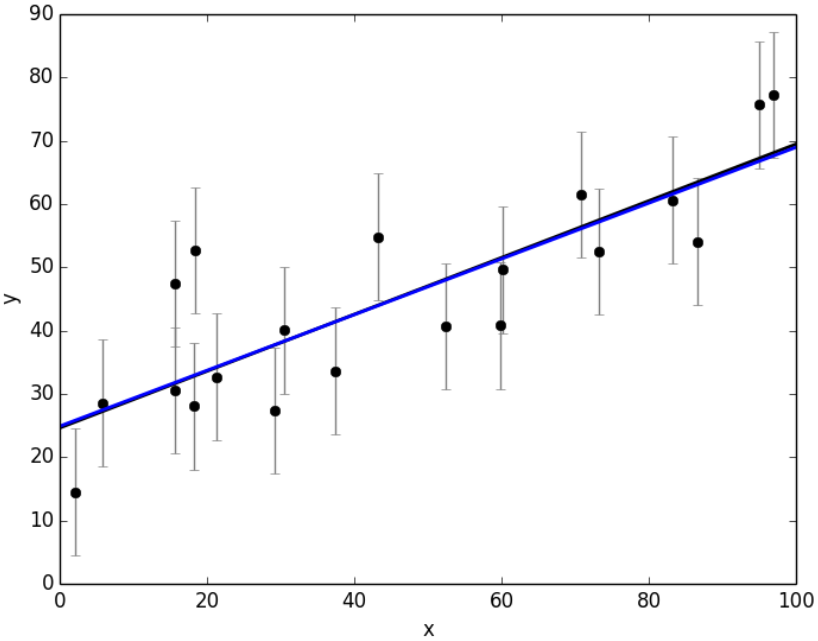
```
import emcee
print("Emcee version {0}".format(emcee.__version__))

def log_prior(theta):
    alpha, beta, sigma = theta
    if sigma < 0:
        return -np.inf  # log(0)
    else:
        return -1.5 * np.log(1 + beta ** 2) - np.log(sigma)

def log_likelihood(theta, x, y):
    alpha, beta, sigma = theta
    y_model = alpha + beta * x
    return -0.5 * np.sum(np.log(2 * np.pi * sigma ** 2)
                         + (y - y_model) ** 2 / sigma ** 2)

def log_posterior(theta, x, y):
    return log_prior(theta) + log_likelihood(theta, x, y)
```

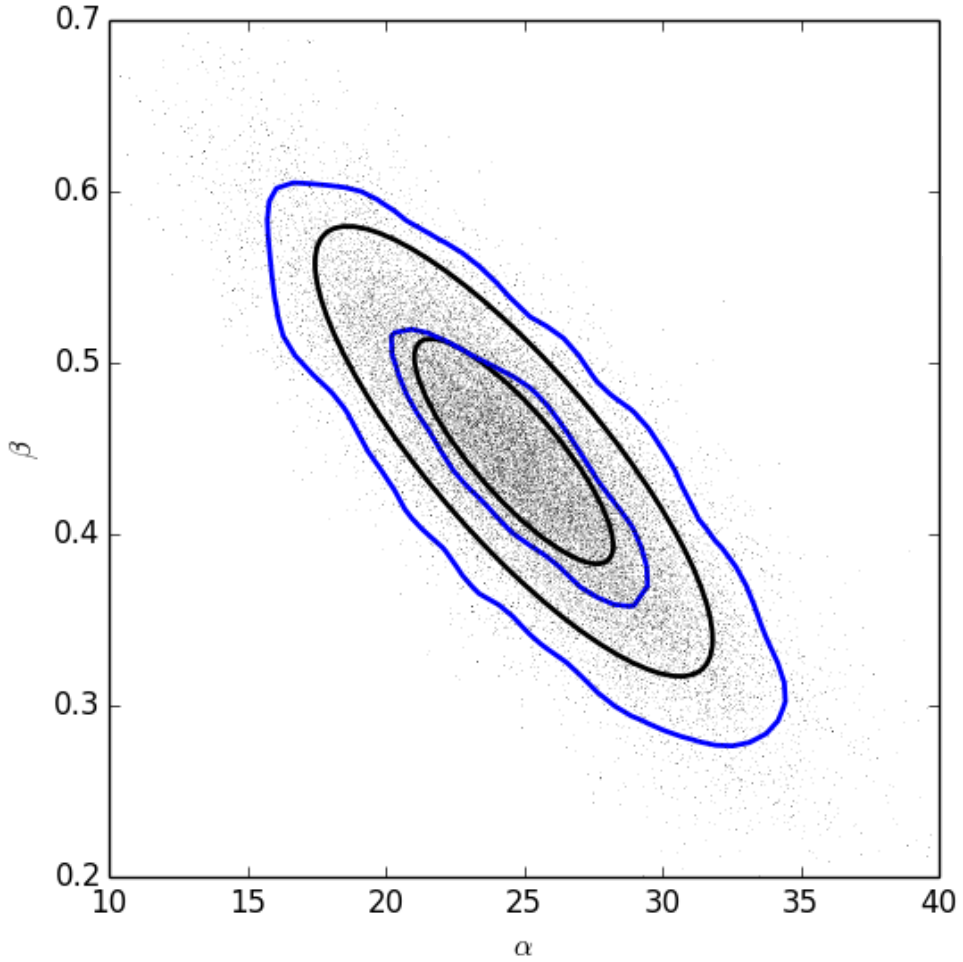
Bayesian Result: emcee

```
ndim = 3  # number of parameters in the model
nwalkers = 50  # number of MCMC walkers
nburn = 1000  # "burn-in" period to let chains stabilize
nsteps = 2000  # number of MCMC steps to take

# set theta near the maximum likelihood, with
np.random.seed(0)
starting_guesses = np.random.random((nwalkers, ndim))

print("  running emcee sampler...")
sampler = emcee.EnsembleSampler(nwalkers, ndim, log_posterior,
                                args=[xdata, ydata])
sampler.run_mcmc(starting_guesses, nsteps)
emcee_trace = sampler.chain[:, nburn:, :].reshape(-1, ndim).T
print("  done")
```

frequentist

emcee





Bayesian Result: pymc

```
import pymc

alpha = pymc.Uniform('alpha', -100, 100)

@pymc.stochastic(observed=False)
def beta(value=0):
    return -1.5 * np.log(1 + value ** 2)

@pymc.stochastic(observed=False)
def sigma(value=1):
    return -np.log(abs(value))

@pymc.deterministic
def y_model(x=xdata, alpha=alpha, beta=beta):
    return alpha + beta * x
```

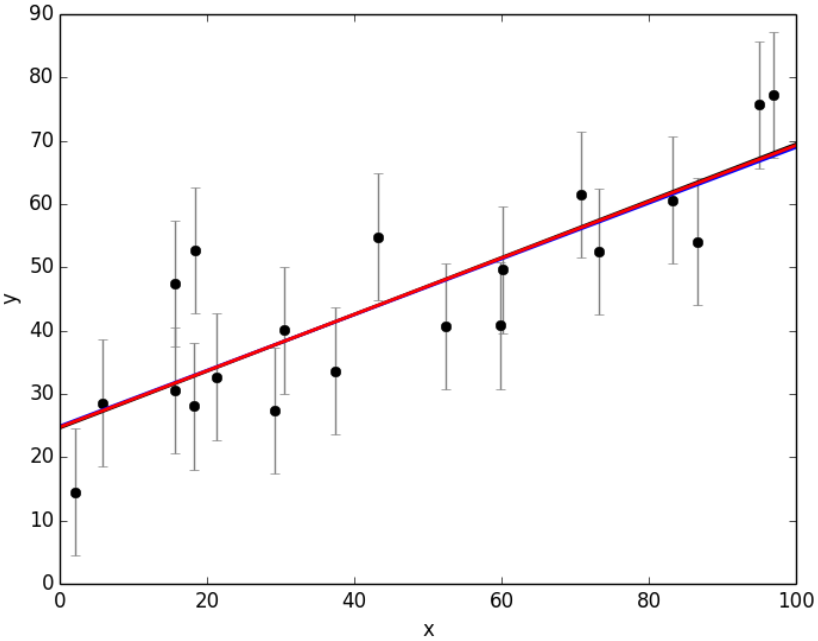
Bayesian Result: pymc

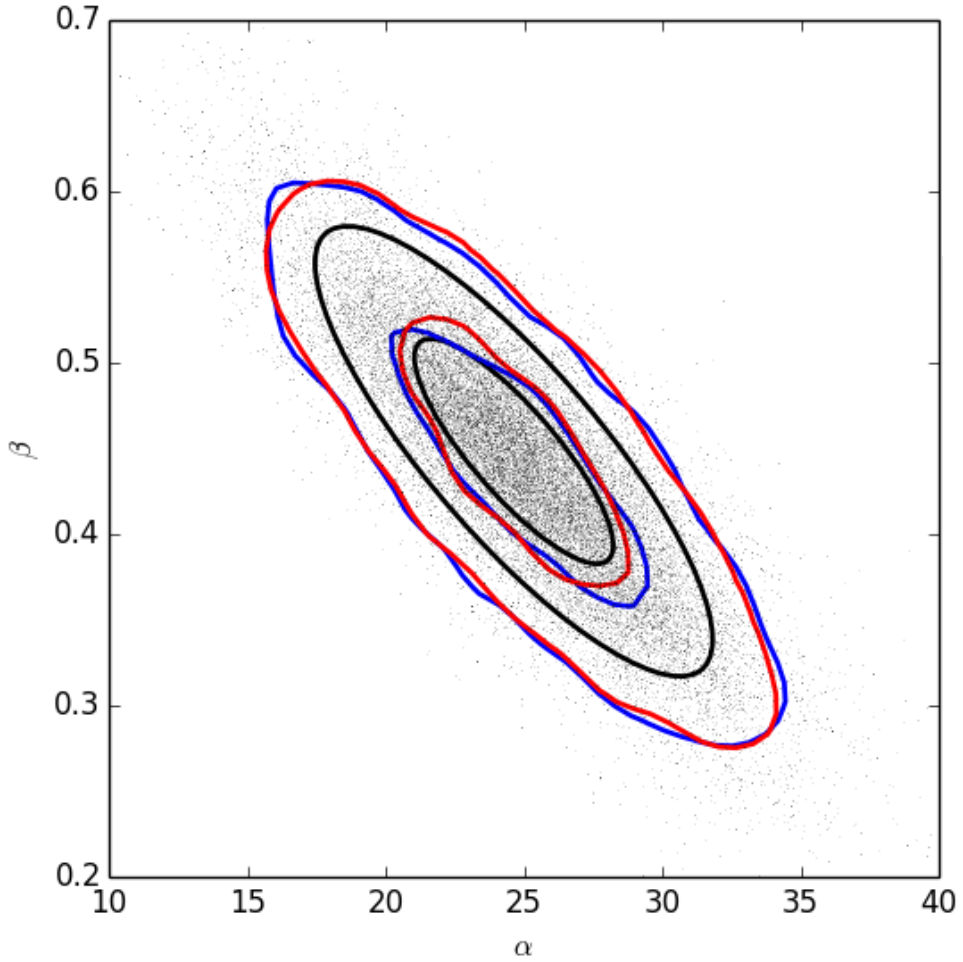
```
y = pymc.Normal('y', mu=y_model, tau=1. / sigma ** 2,
                observed=True, value=ydata)

model1 = dict(alpha=alpha, beta=beta, sigma=sigma,
              y_model=y_model, y=y)

S = pymc.MCMC(model1)
S.sample(iter=100000, burn=50000)

pymc_trace = [S.trace('alpha')[:],
              S.trace('beta')[:],
              S.trace('sigma')[:]]
```

frequentist

emcee

pyMC





Bayesian Result: PyStan

```
import pystan

model_code = """
data {
    int<lower=0> N; // number of points
    real x[N]; // x values
    real y[N]; // y values
}
parameters {
    real alpha_perp;
    real<lower=-pi()/2, upper=pi()/2> theta;
    real log_sigma;
}
transformed parameters {
    real alpha;
    real beta;
    real sigma;
    real ymodel[N];

    alpha <- alpha_perp / cos(theta);
    beta <- sin(theta);
    sigma <- exp(log_sigma);
    for (j in 1:N)
    ymodel[j] <- alpha + beta * x[j];
}
model {
    y ~ normal(ymodel, sigma);
}
"""
```

Bayesian Result: PyStan

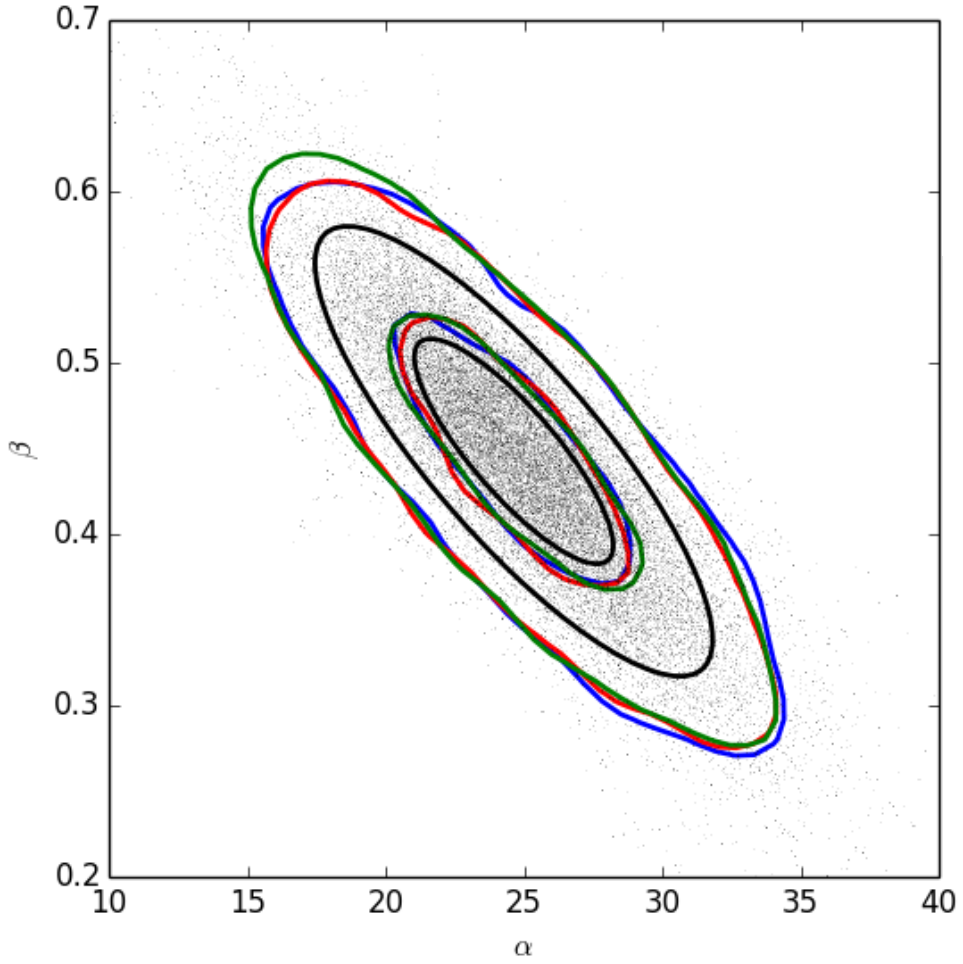
```
data = {'N': len(xdata), 'x': xdata, 'y': ydata}
fit = pystan.stan(model_code=model_code, data=data,
                  iter=25000, chains=4)

tr = fit.extract()
pystan_trace = [tr['alpha'], tr['beta'], tr['sigma']]
```

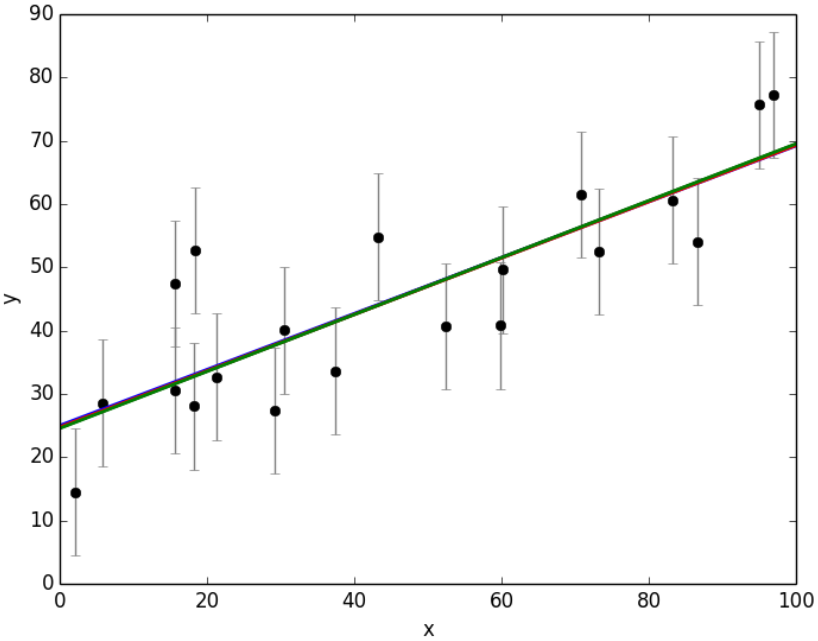
frequentist

emcee

pyMC

pyStan





Conclusion:

- Frequentism & Bayesianism fundamentally differ in their *definition of probability.*

- Results are similar for simple problems, but often differ for more complicated problems.

- Bayesianism provides a more natural handling of nuisance parameters, and a more natural interpretation of errors.

- Both paradigms are useful in the right situation, but *be careful to interpret the results (especially frequentist results) correctly!*

Thank You!


jakevdp@cs.washington.edu


@jakevdp


jakevdp


http://jakevdp.github.io

For more details on this topic, see the accompanying proceedings paper, or the blog posts at the above site