NSE
Nuclear Science & Engineering at MIT
science : systems : society

Bayesian Inference using Python

PSAM 18

Curtis L. Smith

Supporting Material

- **Kelly and Smith, *Bayesian Inference for Probabilistic Risk Assessment***
- ***Bayesian Inference for NASA Probabilistic Risk and Reliability Analysis*, NASA/SP-2009-569**
  - https://ntrs.nasa.gov/citations/20090023159
- **Google Colab (python)**
- **PyMC**
  - https://www.pymc.io/projects/docs/en/stable/learn.html

curtis@mit.edu

Topics

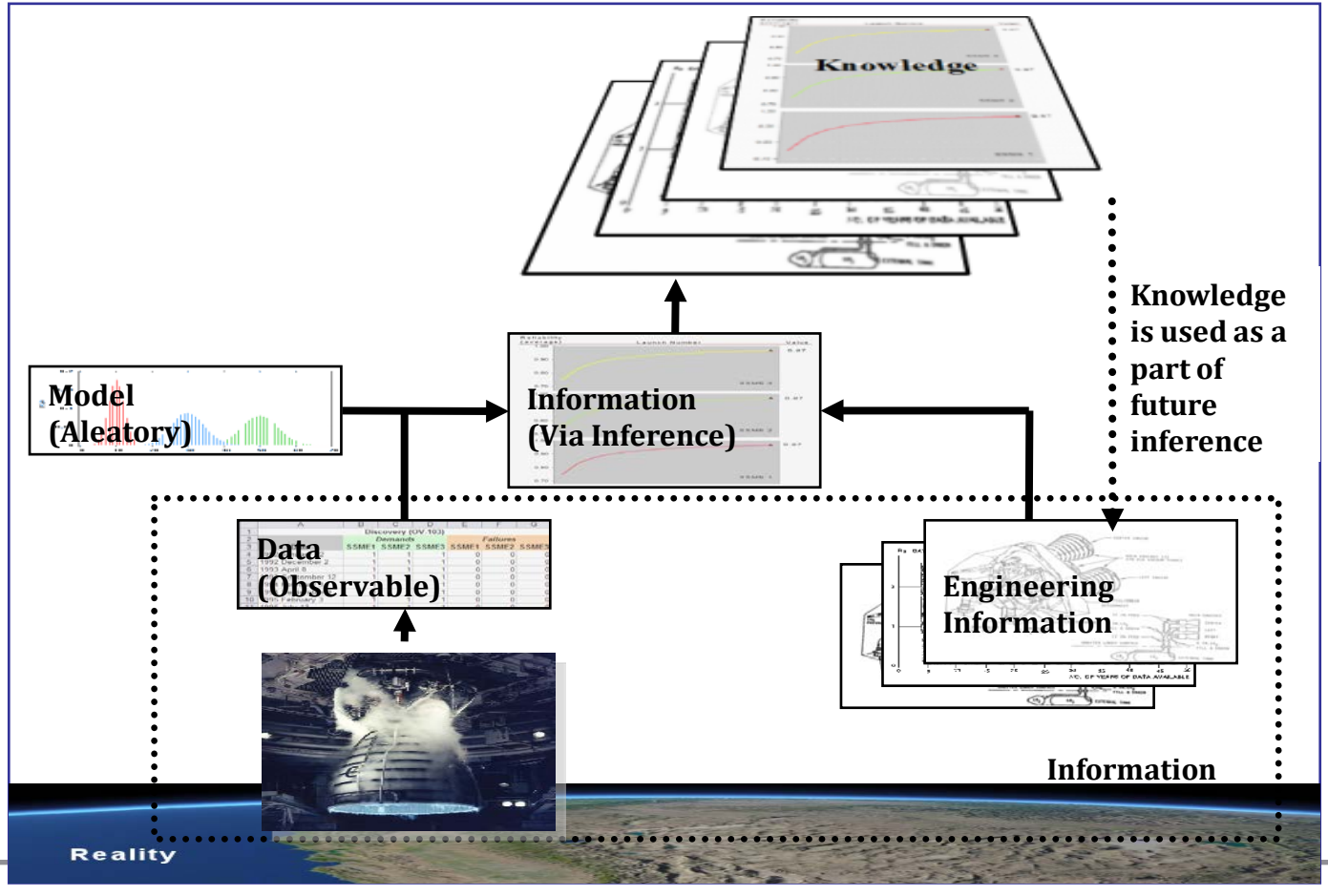
- Overview of Bayesian Parameter Estimation
- Parameters in Failure Models
- Conjugate Priors
- Noninformative Priors
- Nonconjugate Priors
- Bayesian Parameter Estimation in Python
- Uncertain Data

curtis@mit.edu

Overview of Bayesian Parameter Estimation

curtis@mit.edu

Visual Taxonomy of Bayesian Inference



curtis@mit.edu

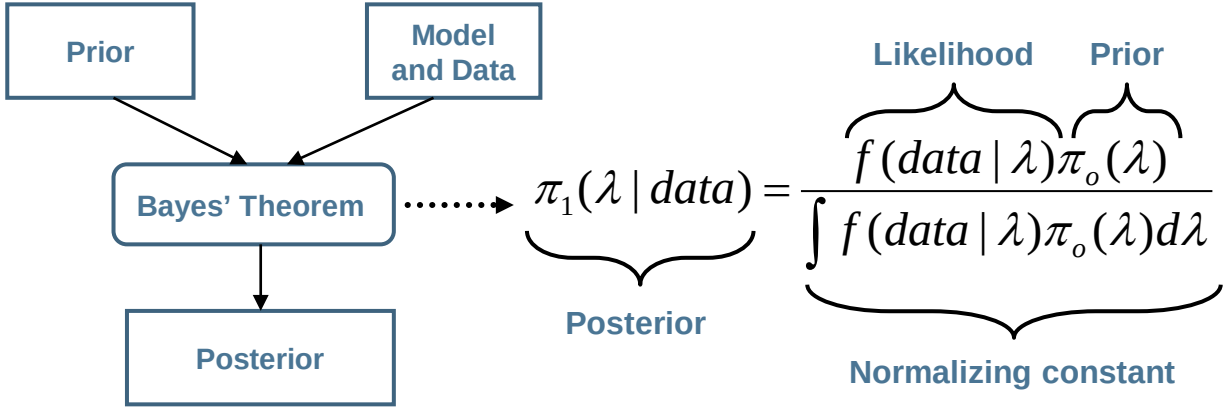
Bayesian Parameter Estimation

- **The general procedure is:**
  - Begin with an aleatory model for the process of interest
  - Specify a prior distribution for parameter(s) in this model, quantifying epistemic uncertainty, i.e., quantifying state of knowledge about the possible parameter values
  - Collect data
  - Obtain the posterior (i.e., updated) distribution for the parameter(s) of interest
  - Check validity of model
- **We follow this process to make inferences, that is, to determine the probability that a model or hypothesis is reasonable, conditional on all available evidence**

curtis@mit.edu

Bayesian Updating – Process Overview

- **The types of information available for parameter values include:**
  - Prior knowledge such as general engineering knowledge and historical information from similar events
  - The observed data for the event of interest in the system under study
- **The Bayesian approach provides a formal mechanism for combing all available information**
  - Including engineering and qualification test data, field experience, expert judgment, and data from similar systems



curtis@mit.edu

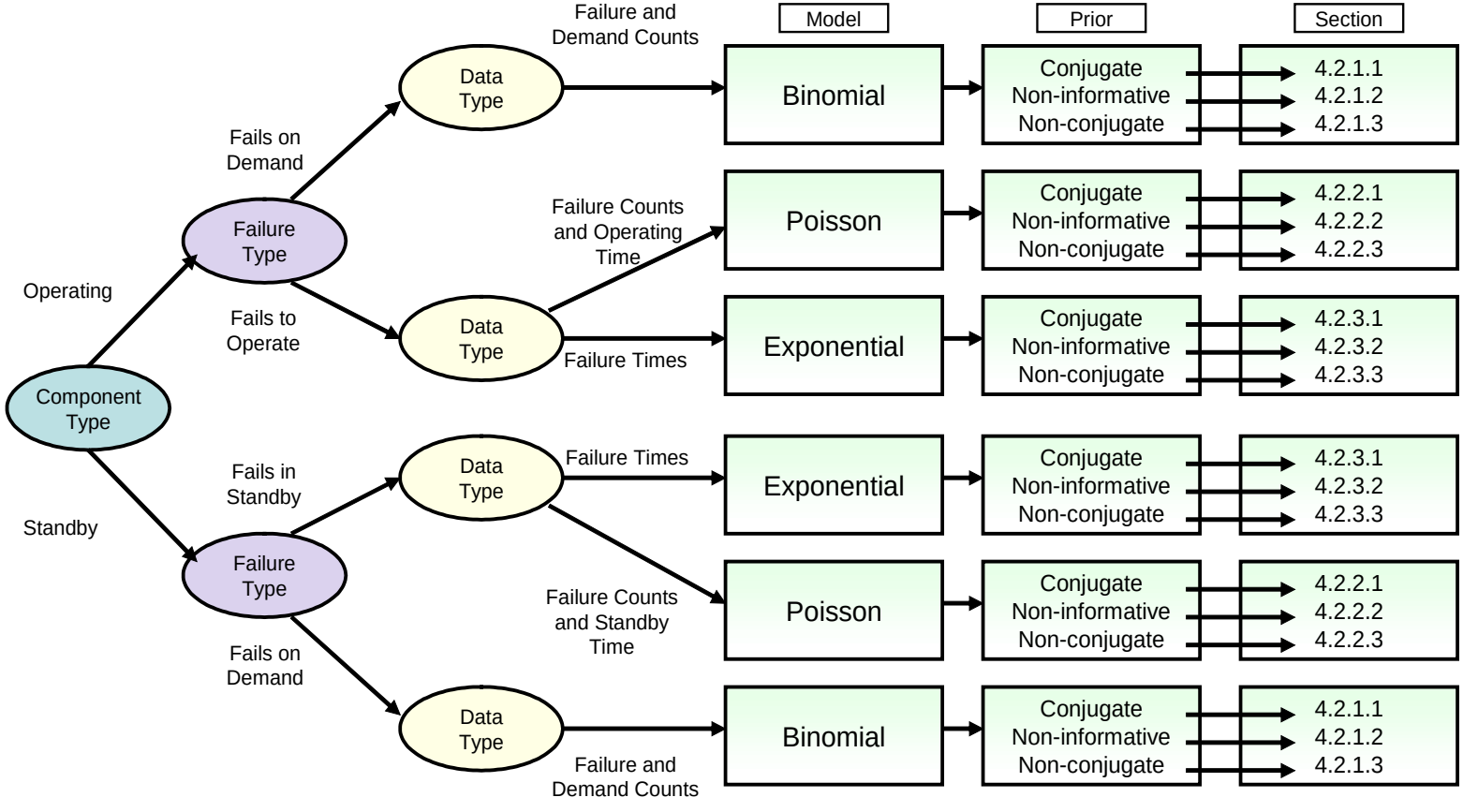
Common Aleatory Models in Failure Modeling

- **Binomial**
- **Poisson**
- **Exponential**

- **We use these models to count failures or estimate failure times**

curtis@mit.edu

A Modeling "Roadmap" (from NASA/SP-2009-569)



curtis@mit.edu

Parameters in Failure Models

curtis@mit.edu

Binomial Distribution: Functional Form

$$\binom{n}{x} = \frac{n!}{k!\,(n-k)!}$$

- **The random variable X has a binomial(n, p) distribution:**

$$f(x) = \mathrm{P}(X = x) = \binom{n}{x} p^x (1-p)^{n-x}$$

for x = 0, 1, …, n
(x = number of failures)

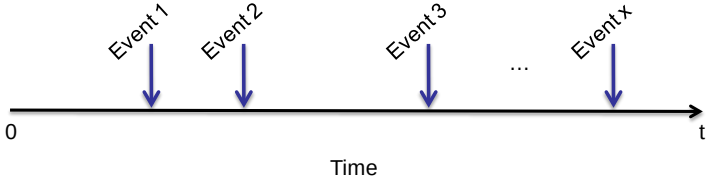
- **Distribution parameters are p (unknown) and n (specified)**
- **For Bayesian inference, we write f(x) as f(x|p), called the likelihood function, sometimes denoted L(p)**
  - For example, if we see n=10 and x = 2, then $\mathrm{L}(p) = \binom{10}{2} p^2 (1-p)^8$
  - Leads to frequentist maximum likelihood estimate (MLE) for p of x/n
- **X is observed (failures) and p is unknown (focus of inference)**

curtis@mit.edu

Poisson Distribution: Functional Form

- **The random variable X has a Poisson(λt) distribution:**

$$f(x) = \frac{(\lambda t)^x e^{-\lambda t}}{x!}$$



for x = 0, 1, 2, …
(x = number of events)

- **The distribution depends on one quantity, λt, (λ unknown, t specified)**
  - Therefore, product λt is sometimes written as μ (or even λ), and the distribution is called Poisson(μ)
- **For Bayesian inference, we write f(x) as f(x|λ), called the likelihood function, sometimes denoted L(λ)**
  - Leads to frequentist MLE for λ of x/t

curtis@mit.edu

Prior Distributions

- **We can have different types of prior information**
  - Discrete priors (not discussed)
  - Conjugate priors
    - Informative
    - Noninformative
  - Nonconjugate priors
  - Hybrid priors (a combination of discrete and continuous, also not discussed)
- **In any situation though, the prior is intended to represent knowledge about a parameter independent of the data collected for that parameter**

curtis@mit.edu

Conjugate Priors

curtis@mit.edu

Prior Distributions

- **In risk and reliability assessment, we tend to use one of two types of priors**
  1. Conjugate (distributions where the posterior and prior have the same distribution type)
     - These can be informative or noninformative
  2. Nonconjugate
     - These can be informative or noninformative
- **For example, to estimate P(component fails to start) we could**
  - Use a binomial model with a beta prior → beta(0.5, 0.5)
    - This would be a conjugate using a noninformative prior (i.e., Jeffreys)
  - Use a binomial model with a beta prior → beta(10, 55)
    - This would be a conjugate using an informative prior
  - Use a binomial model with a gamma prior → gamma(1, 5)
    - This would be a nonconjugate using an informative prior
- **In all cases, the prior should represent what we know about the problem at hand, independent of data**

curtis@mit.edu

Conjugate Priors for Common Failure Models

- **If X is binomial(n, p) and $g_{prior}(p)$ is beta(α_{prior}, β_{prior})**
  - Then **posterior** distribution of p is
    - beta(α_{post}, β_{post})

      $\alpha_{post} = \alpha_{prior} + x$ (x = # events)

      $\beta_{post} = \beta_{prior} + n - x$ (n = total # trials)
- **If X is Poisson(λ t) and $g_{prior}(\lambda)$ is gamma(α_{prior} , β_{prior})**
  - Then **posterior** distribution of λ is
    - gamma(α_{post}, β_{post})

      $\alpha_{post} = \alpha_{prior} + x$ (x = # events)

      $\beta_{post} = \beta_{prior} + t$ (t = observation time)
- **If $T_1,\ldots,T_n$ are times from exponential(λ) distribution and $g_{prior}(\lambda)$ is gamma(α_{prior}, β_{prior})**
  - Then **posterior** distribution of λ is
    - gamma(α_{post}, β_{post})

      $\alpha_{post} = \alpha_{prior} + n$ (n = # events)

      $\beta_{post} = \beta_{prior} + \Sigma t_i$ (t_i = observed times of n events)

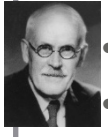
curtis@mit.edu

Noninformative Priors

curtis@mit.edu

Noninformative (Formal) Prior Distributions

- **The original intent of "noninformative" priors was to answer question**
  - How do we find a prior representing complete ignorance?
- **Rev. Bayes suggested a uniform prior**
  - Laplace used this in his activities with great success
  - But, there are philosophical & mathematical problems with this
- **Sir Harold Jeffreys suggested a prior that was invariant to changes in**
  - Scale
  - Location
- **Consequently, a "noninformative" prior is typically *not* uniform for the parameter of interest**
  - It is uniform (or approximately so) for some transformed parameter
  - It depends on the process generating the data
  - Has property that the Bayes posterior intervals are approximately equal to frequentist confidence interval
    - Formal priors "let the data speak for themselves"



curtis@mit.edu

Jeffreys Prior Distributions for Our Failure Models

- **For binomial(n, p) aleatory model**
  - Jeffreys noninformative prior for p is beta(1/2, 1/2), which is a proper prior (integral equals 1)
- **For Poisson(λt) aleatory model**
  - Jeffreys noninformative prior for λ is gamma(1/2, 0) (improper prior, integral diverges)
- **For exponential(λ) aleatory model**
  - Jeffreys noninformative prior for λ is gamma(0, 0) (improper prior, integral diverges)

curtis@mit.edu

Updating Jeffreys Prior Distributions

- **For binomial(n, p) aleatory model**
  - Jeffreys noninformative prior for p is beta(½, ½)
  - Posterior is beta(x + ½, n – x + ½)
- **For Poisson(λt) aleatory model**
  - Jeffreys noninformative prior for λ is gamma(½, 0)
  - Posterior is gamma(x + ½, t)
- **For exponential(λ) aleatory model**
  - Jeffreys noninformative prior for λ is gamma(0, 0)
  - Posterior is gamma(n, Σt_i)

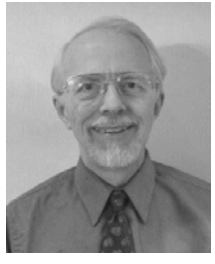
curtis@mit.edu

Constrained Noninformative (CNI) Prior

- **Recall that Jeffreys prior for binomial likelihood is beta(½, ½)**
  - The prior mean is 0.5 (quite large if a failure probability)
- **With sparse data, prior mean can influence results too much**
- **To overcome this, can use prior which has specified mean, but is close to Jeffreys prior otherwise**
  - Goal is to maximize uncertainty but still start with an acceptable mean
  - Specified mean might be industry average value
- **Result is called "constrained noninformative" (CNI) prior**

Left to right: Claude Shannon, Edward Jaynes, Corwin Atwood





curtis@mit.edu

Constrained Noninformative Prior

- **Cannot be written in form of standard distribution for case of binomial likelihood**
  - Approximated well by beta distribution with α = 0.5
  - For the beta, the mean = α / (α + β), thus β can be found to be
    - β = α*(1-mean) / mean
- **For Poisson likelihood, CNI prior is gamma(½, 1/(2*mean))**
- **For exponential likelihood, cannot define CNI prior, as it is not proper, therefore cannot have finite mean**
  - Alternative is maximum entropy prior, which is
    - Gamma(1, 1/mean)
    - This is an exponential distribution

curtis@mit.edu

Nonconjugate Priors

curtis@mit.edu

Nonconjugate Priors

- **For each aleatory model, there is at *most* one conjugate prior type**
  - All other distributional forms are nonconjugate
  - Integral in denominator of Bayes' Theorem must (typically) be done numerically
- **A common nonconjugate prior in risk assessment is lognormal distribution**
  - Originally used in WASH-1400 to represent plant-to-plant variability in parameter values
  - Very convenient when uncertainty spans several orders of magnitude
  - Useful in expert elicitation (e.g., seismic analysis)
    - Elicit median and upper bound, 5th and 95th percentiles, etc.
  - Generic failure rate and failure probability databases often express uncertainty in terms of lognormal distribution
- **Definition of a lognormal distribution**
  - X is lognormal(μ, σ) if ln(X) is normal(μ, σ)

curtis@mit.edu

Nonconjugate Prior with Binomial

- **If we use a lognormal distribution as a prior for a binomial failure model**
  - This is nonconjugate
- **Bayes' theorem looks like this with the integration**

$$P(p|x) = \frac{\binom{n}{x} p^x (1-p)^{n-x} \frac{1}{\sqrt{2\pi}\sigma p} \exp\left(-\frac{(\ln p - \mu)^2}{2\sigma^2}\right)}{\int_0^\infty \binom{n}{x} p^x (1-p)^{n-x} \frac{1}{\sqrt{2\pi}\sigma p} \exp\left(-\frac{(\ln p - \mu)^2}{2\sigma^2}\right) dp}$$

- **The problem is even more complicated if we want the expected value of the posterior**
  - That involves another integration

$$E[p] = \int_0^\infty p\, P(p|x) dp$$

curtis@mit.edu

How to Solve the Example

- **The more general case is to solve Bayes numerically**
- **One approach is Markov Chain Monte Carlo (MCMC)**
  - The idea is to estimate the posterior distribution directly
  - Sample from Pr(p | x)
- **Different algorithms exist**
  - Simplest is Random Walk Metropolis
    - Guess value of p → call it p'
    - Check a branching condition r
    - Get random value from ordinate of Pr(p' | x)'s CDF
      - This is a pick from uniform distribution from 0 to 1
    - Check against branching condition r to keep p' (or not)
- **Solution tools**
  - MultiBUGS or Python (e.g., pymc library)

curtis@mit.edu

Bayesian Parameter Estimation in Python

curtis@mit.edu

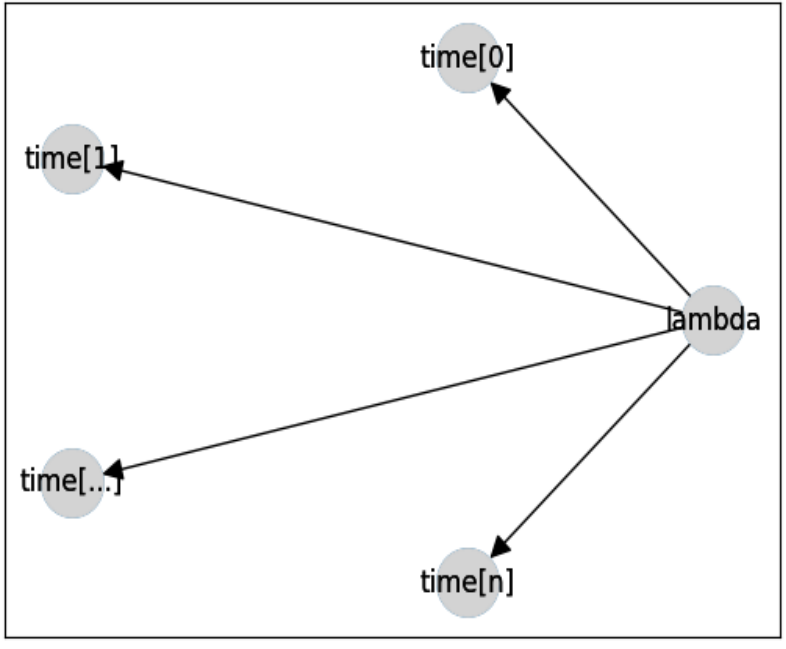
Models for Time to _____ (e.g., recovery, failure, …)

- **How do we represent a duration?**
- **We create a probability distribution that represents the time of interest**
  - These times could represent any process, repairing components, failures, suppressing a fire, an adversary breaching a barrier, …
- **One of the *simplest* aleatory model for time is the exponential distribution**
  - $T \sim \exp(\lambda)$ Recall that E[T] = Mean Time to ___ = $1/\lambda$
    - The unknown parameter is λ, a rate
- **Our example data is series of times for recovery, assumed to be represented by exp(λ)**

curtis@mit.edu

Influence Diagram Representation

- **Assume a facility records the time it takes (in minutes) to restore a system**
  105, 5, 1263, 72, 37, 814, 11.5, 211, 330, 7929, 296, 31, 120, 441



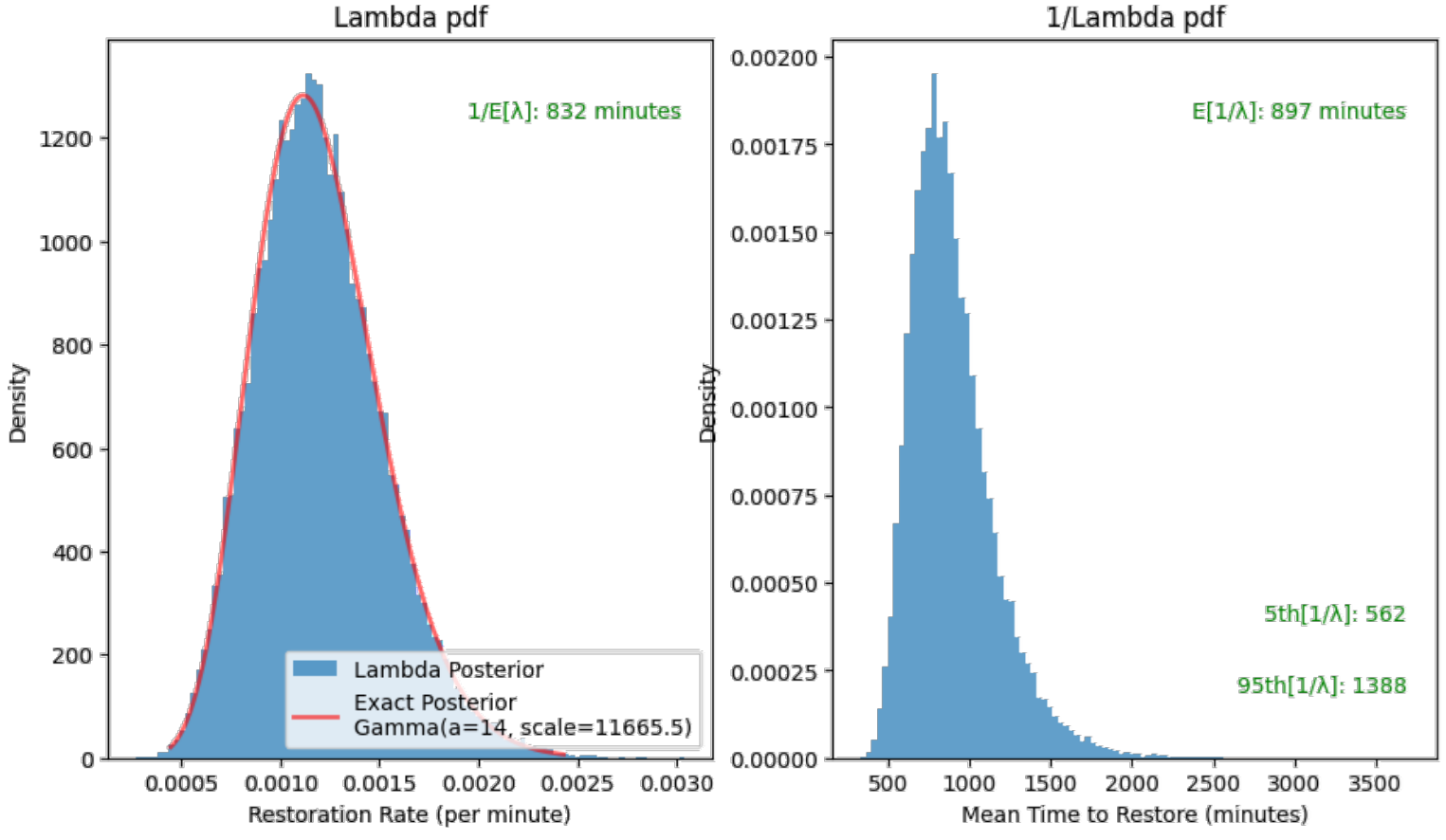
curtis@mit.edu

The Python Model

```
import pymc as pm
number_samples = 20_000
tune_samples = int(number_samples * 0.1)
restoration_times = [105, 5, 1263, 72, 37, 814, 11.5, 211, 330, 7929, 296, 31, 120, 441] # Minutes
n = len(restoration_times)
sum_times = np.sum(restoration_times)
# Define the PyMC Model
with pm.Model() as model:
    # Prior for the restoration rate Jeffreys Gamma(0,0)
    lambda_param = pm.Gamma('lambda_param', alpha=0.001, beta=0.001)
    # Likelihood: Exponential time observation
    obs = pm.Exponential("obs", lam=lambda_param, observed=restoration_times)
    # Inference: Sample from posterior
    idata = pm.sample(num_samples, tune=tuning_samples, cores=2, target_accept=0.9)
# Get posterior samples
lambda_samples = idata.posterior['lambda_param'].values.flatten()
```


curtis@mit.edu

Running the Script in Python Provides



curtis@mit.edu

Lambda versus MTTR versus Predicted Restoration Times

- **The posterior results**

Variable		*Mean*	*5th*	*95th*
MTTR (min.)	900	560	1400	
Lambda (/min)	1.2E-3	7.2E-4	1.8E-3	

- **Okay, how might we use this information in risk or reliability?**
  - MTTR only represents the prediction of *mean time* to restore, it is not a predictive model of possible restoration times
  - The Python posterior samples represent uncertainty in Lambda
- **We could also produce a posterior predictive distribution of times**
  - This is accomplished by using the aleatory model (exponential in this case) with the uncertainty in Lambda

curtis@mit.edu

What is the Posterior Predictive Distribution?

- **Posterior predictive distribution**
  - Where we (via Bayes) create a distribution using the aleatory model + posterior distributions for parameters in that model
  - This predictive distribution represents a way to make future predictions about the process represented by the aleatory model
- **For T ~ exp(λ)**
  - Predictions of a future time (t_{pred}) is obtained by averaging over the posterior distribution for all model parameters (only λ in this case)

$$f(t_{pred}|t_1,\dots,t_N) = \int_0^\infty f(t_{new}|\lambda)\, p(\lambda|t_1,\dots,t_N)\, d\lambda$$

$$= \int_0^\infty \lambda e^{-\lambda t_{new}}\, g(\lambda|\, t_1,\dots,t_N)\, d\lambda$$

curtis@mit.edu

Determining the Posterior Predictive Distribution

- **Since we have the aleatory model & associated parameter distributions, it is easy to calculate the posterior predictive distribution**
  - Add a new line to the Python script

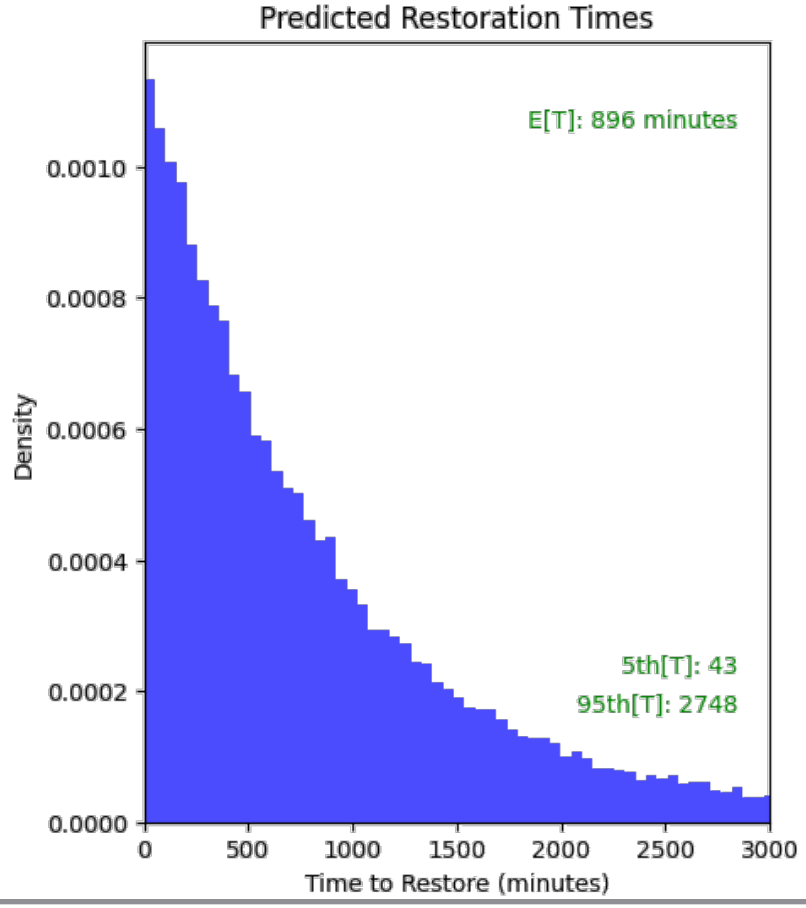
    exp_times = [expon.rvs(scale=1/lam) for lam in lambda_samples]
  - It uses the distribution on lambda to predict restoration times
    - These predicted times are stored in the exp_times variable
- **The posterior results**

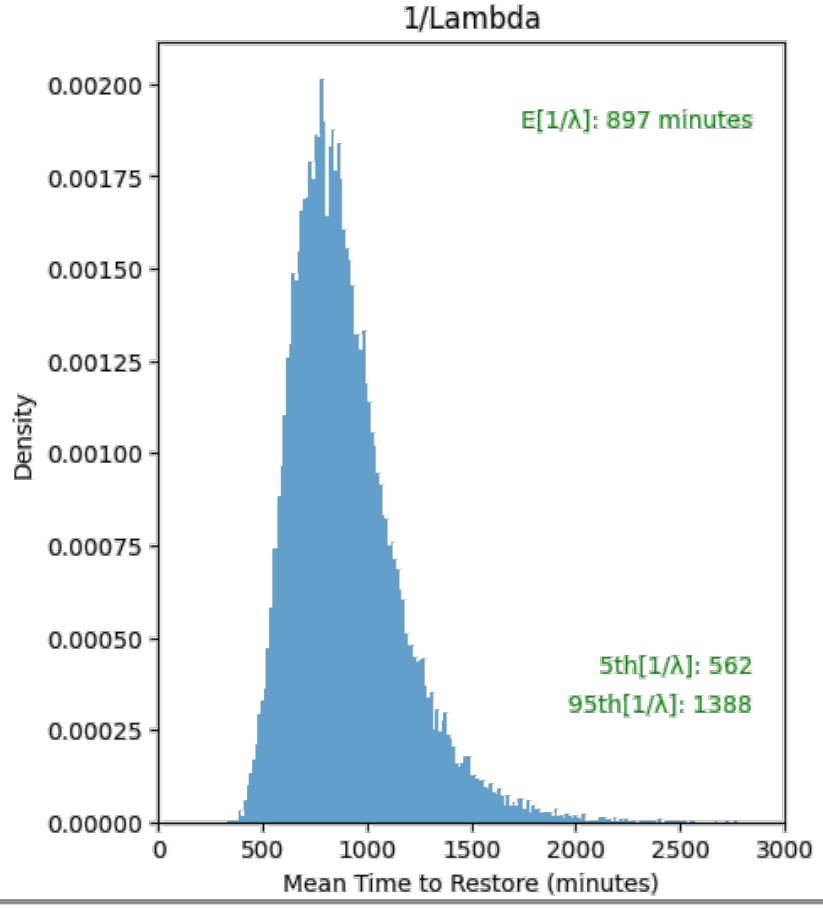
Variable		*Mean*	*5th*	*95th*
MTTR (min.)	900	560	1400	
exp_times (min.)		900	43	2700

- **Note the larger uncertainty for exp_times compared to MTTR**
  - The predictive distribution uses *all* the uncertainty contained in λ

curtis@mit.edu

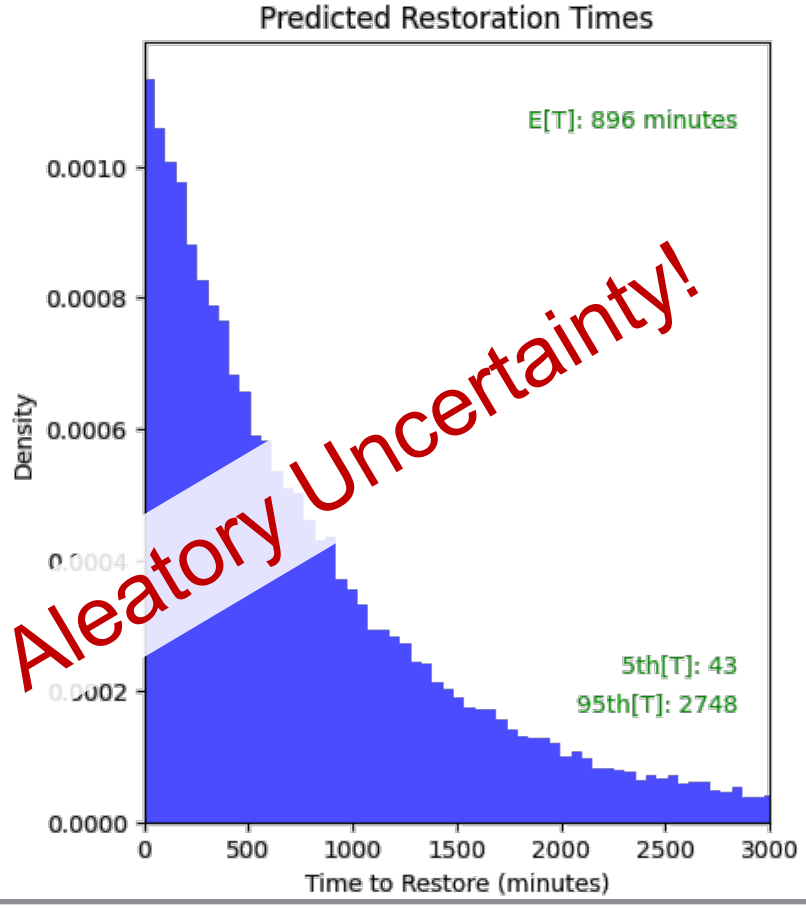
Plotting the Posterior Predictive Distribution



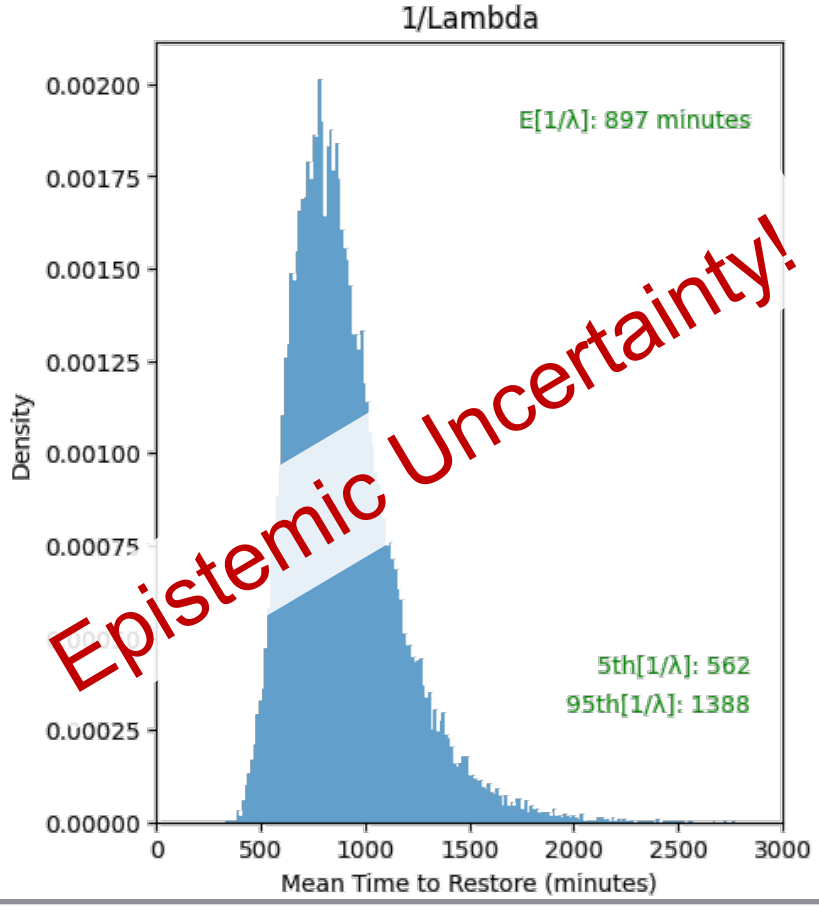


curtis@mit.edu

Plotting the Posterior Predictive Distribution



Aleatory Uncertainty!



Epistemic Uncertainty!

curtis@mit.edu

Uncertain Data

curtis@mit.edu

What is Uncertain Data?

- **Occasionally, observed data are uncertain or "fuzzy"**
- **For example, we see**
  - Various types of censoring of data
  - Unclear whether failure occurred (or not)
  - Number of demands not known with certainty
- **We can treat these complications using a Bayesian approach to directly account for the uncertainty in the data**
- **While accounting for unknown information will lead to more imprecise parameter estimation**
  - Not accounting for this information may lead to false inference

curtis@mit.edu

Three Cases can be Considered

- **We could consider the following cases**
  - Uncertain durations
    - For example, data is collected related to the time it takes to recover power to a facility once it is lost
    - The actual duration of real events may not be known exactly
  - Uncertain demands
    - For example, a facility may occasionally test a component, but poor record keeping leads to uncertainty in the actual number of demands recorded
  - Uncertain failures
    - For example, an actual event is recorded as a failure but upon review we are not certain that the component was not able to perform its intended function

curtis@mit.edu

When Durations are Uncertain

- **Consider following data for fire suppression time**

Event	*Duration (mins.)*
1	*< 5*
2	*< 5*
3	*< 15*
4	*14*
5	*15*
6	*15*
7	*10 - 30*
8	*15*
9	*150*
10	*100 - 300*

curtis@mit.edu

When Durations are Uncertain

- **Treat the data (observations) as uncertain**
- **Assume a simple model for T (fire duration) as exponential**
  - Tsupp ~ exp(λ) where λ is suppression rate, units of min^{-1}
- **Some data are left-censored, some are interval-censored (only have an interval for some points)**
- **Handled in Python by using censored likelihood approach**
- **Specifying the data**

```
# Set up the data type
# KEY: 0: Observed
#      1: Right Censored (time is greater than t)
#      2: Interval Censored (time between high and low t)
#      3: Left Censored (time is less than t)
type = np.array([3,3,3,0,0,0,2,0,0,2])
low  = np.array([0.01,0.01, 0.01,14,15,15,10,15,150,100])
high = np.array([   5,   5,   15,14,15,15,30,15,150,300])
```


curtis@mit.edu

When Durations are Uncertain

- **Excerpt from the Python program in the Colab file**

```
with pm.Model() as mixed_model:
    # Specify the prior on the RATE of the exponential model
    lambda_param = pm.Gamma('lambda_param', alpha=0.001, beta=0.001)
    # Define the exponential distribution representing times
    dist = pm.Exponential.dist(scale=1/lambda_param)
 # Custom logp for mixed censoring
    def mixed_logp(value, lower, upper, type, dist):
        logp_obs = pm.logp(dist, lower) # TYPE 0:
        logp_right = pm.logccdf(dist, lower) # Type 1
        logp_int = pm.logcdf(dist, lower) + at.log(at.expm1(pm.logcdf(dist, upper) -
pm.logcdf(dist, lower))) # Type 2
        logp_left = pm.logcdf(dist, upper) # Type 3
        # Combine based on type indicator
        return at.switch(at.eq(type, 0), logp_obs,
               at.switch(at.eq(type, 1), logp_right,
               at.switch(at.eq(type, 2), logp_int, logp_left)))
```

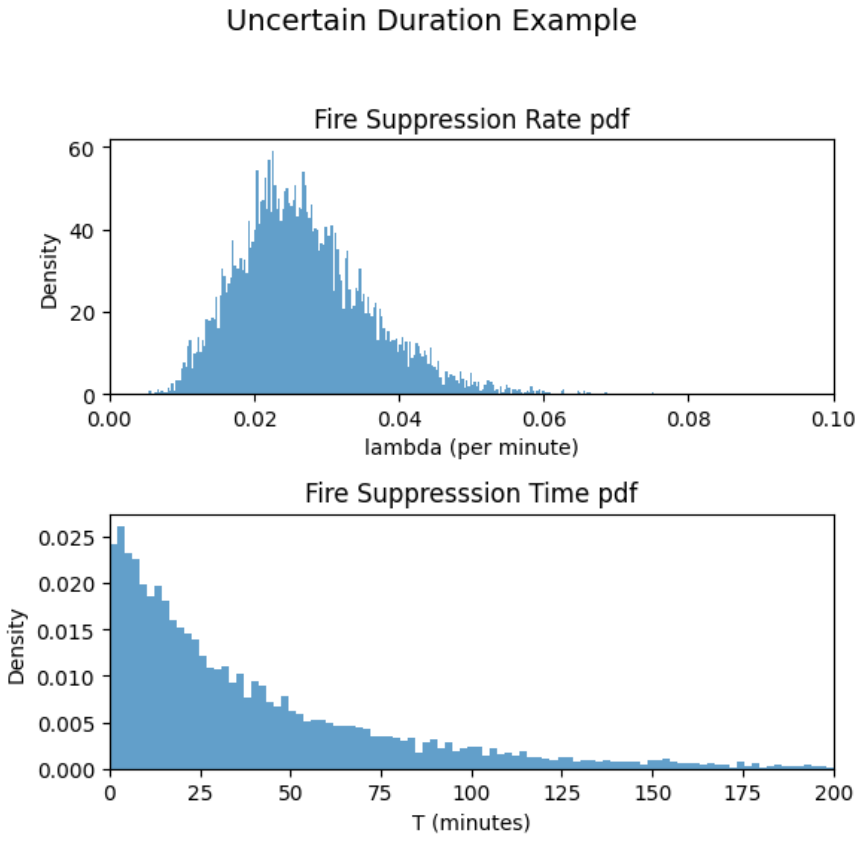

curtis@mit.edu

When Durations are Uncertain

- ** OpenBUGS results **
- Lambda Mean: 2.7E-02
- Lambda 5th Percentile: 1.4E-02
- Lambda 95th Percentile: 4.2E-02

- ** PyMC results **
- Lambda Mean: 2.7E-02
- Lambda 5th Percentile: 1.4E-02
- Lambda 95th Percentile: 4.3E-02

- Fire Suppression Time Mean: 4.2E+01
- Fire Suppression 5th Percentile: 2.1E+00
- Fire Suppression 95th Percentile: 1.3E+02



curtis@mit.edu

Example When Demands are Uncertain

- **Assumptions:**
  - Prior distribution for MOV fails to open (FTO) probability is lognormal with mean = 0.003 and EF = 10
    - Convert (via DSW) mean and EF to mu (-6.79) and sigma (1.4)
  - One failure is seen
  - Demands are not recorded, but are known to be between 12 and 40
- **Excerpt from the Python model**

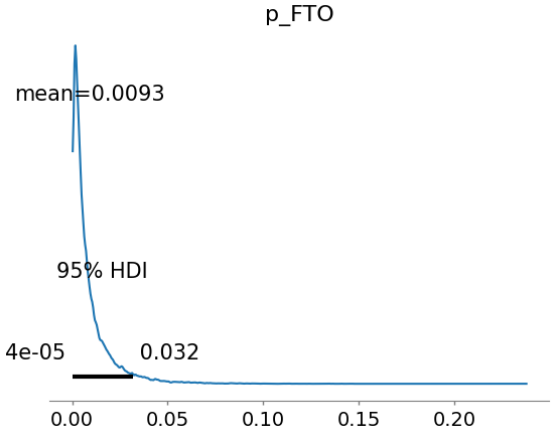
```
with pm.Model() as model:
    # Distribution for uncertain n
    n = pm.DiscreteUniform("n", lower=demands_low, upper=demands_high)
    # Prior for probability FTO p
    p_FTO = pm.LogNormal("p_FTO", mu=-6.79, sigma=1.4)
    # Binomial Likelihood with uncertain n
    y = pm.Binomial("y", n=n, p=p_FTO, observed=num_FTO)
    # Sampling
    idata = pm.sample(num_samples, tune=tuning_samples, cores=2, target_accept=0.9)
```

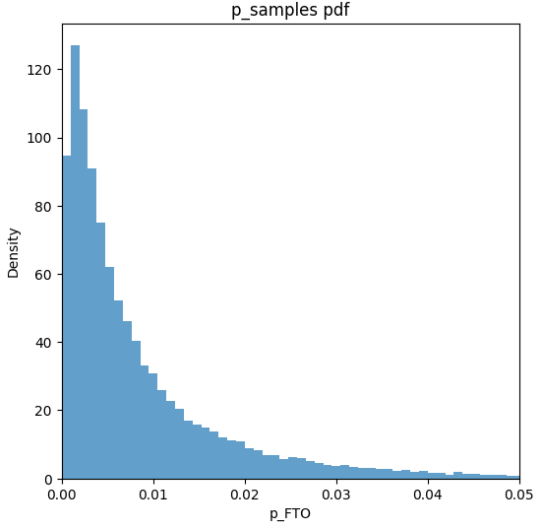

curtis@mit.edu

When Demands are Uncertain

- **Posterior results**

- **p_FTO Mean: 9.16E-03**
- **p_FTO 5th Percentile: 6.66E-04**
- **p_FTO 95th Percentile: 3.12E-02**





curtis@mit.edu

When Failures Are Uncertain

- **Information from event reports and other information sources may not be clear enough to ascertain exact number of failures**
- **Sloppy record-keeping may result in imprecise estimates**
  - Analogous to rounding of value
- **Two approaches**
  1. Posterior-averaging
     - Allows weighting of each posterior
       - For example, if the failure count might be 4, 5, or 6, we can assign a weight to each possible outcome
  2. Likelihood-based
     - Similar to posterior-averaging, but assumes all possible outcomes have equal weight

curtis@mit.edu

Posterior-Averaging Approach

- **Analyst must develop subjective prior distribution for number of observed events**
- **Example 1: plugging of service water strainers**
  - Inspection Report (IR) describes 7 plugging events over time period of interest
  - Three of these may not have been complete plugging events from perspective of a risk scenario
  - Therefore, actual number of events is 4, 5, 6, or 7
    - We are just uncertain as to which is “correct”
  - The total operation time was 48,180 hours

curtis@mit.edu

Posterior-Averaging Approach

- **By poring over the IR, and applying expert knowledge, we come up with following distribution for actual number of complete plugging events:**

  Pr(x = 4) = 0.75

  Pr(x = 5) = 0.15

  Pr(x = 6) = 0.075

  Pr(x = 7) = 0.025
- **Note that probabilities must sum to unity**
- **What Python will do is run four cases (where x=4, x=5, x=6, and x=7) then weight the posterior distributions**
  - Results in one overall distribution though

curtis@mit.edu

Posterior-Averaging Approach

- **Excerpt of Python program for the example**

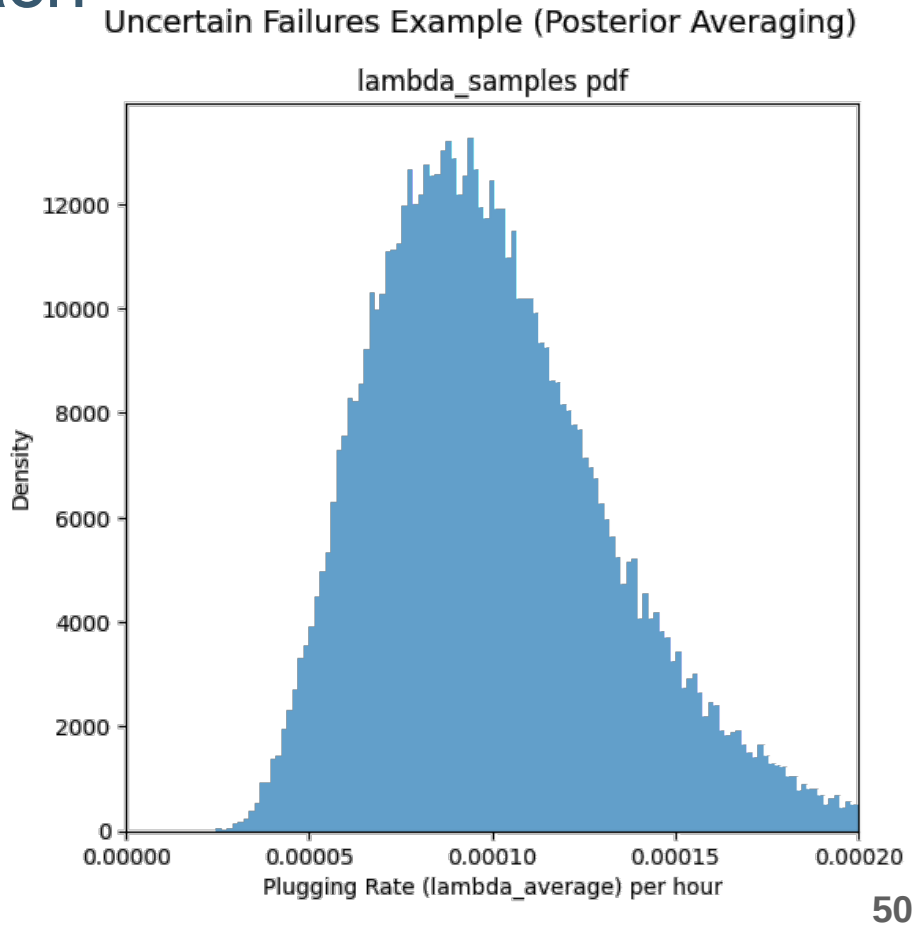
```
# Use a Jeffreys prior
a_prior = 0.5
b_prior = 0
t = 48180 # hours
num_samples = 50000
posteriors = [
    np.random.gamma(shape=a_prior + 4, scale=1/(b_prior+t), size=num_samples),
    np.random.gamma(shape=a_prior + 5, scale=1/(b_prior+t), size=num_samples),
    np.random.gamma(shape=a_prior + 6, scale=1/(b_prior+t), size=num_samples),
    np.random.gamma(shape=a_prior + 7, scale=1/(b_prior+t), size=num_samples)
]
# Assign weights to each model (must sum to 1)
weights = [0.75, 0.15, 0.075, 0.025]
# Perform weighted averaging using the posteriors
lambda_samples = posteriors[0] * weights[0] + posteriors[1] * weights[1] +
posteriors[2] * weights[2] + posteriors[3] * weights[3]
```


curtis@mit.edu

Posterior-Averaging Approach

- **Posterior results for example**

- **Lambda Mean: 1.01E-04**
- **Lambda 5th: 5.48E-05**
- **Lambda 95th: 1.64E-04**



curtis@mit.edu

Conclusions

- **Thank you for your attention!**

- **Send me an email if you would like a copy of the slides or have questions**

curtis@mit.edu