Probabilistic Programming

Marius Popescu

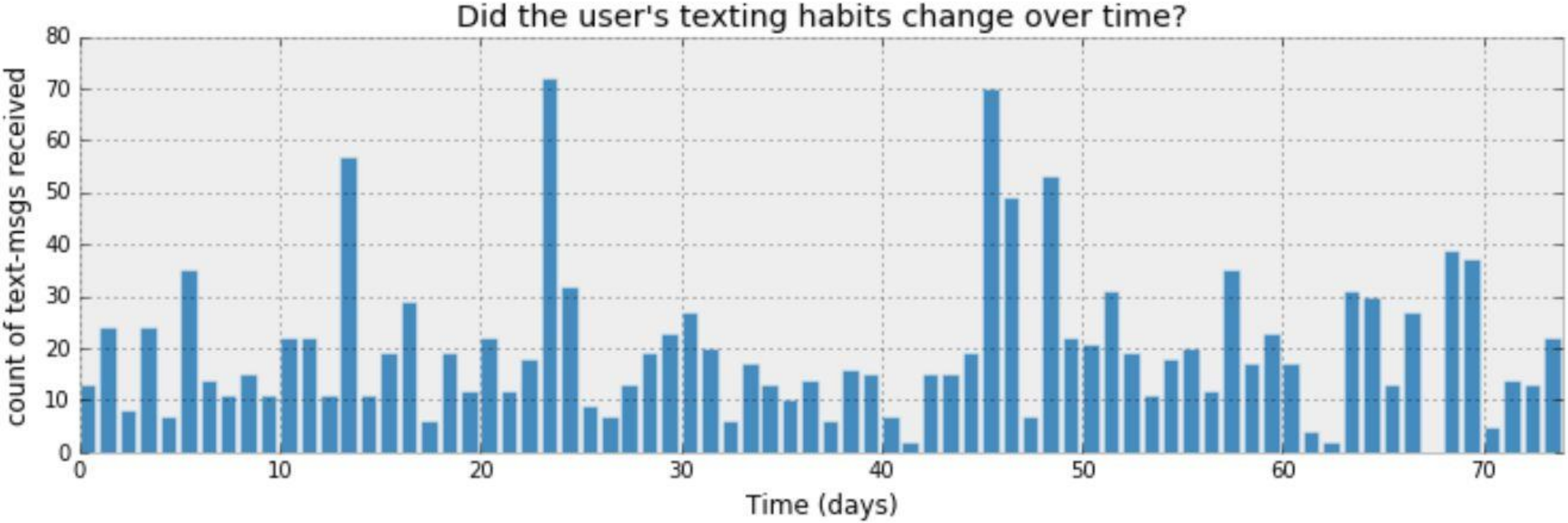
popescunmarius@gmail.com

2019 - 2020

A More Elaborate “Hello World!”

Your most humble servant and most faithful friend

Inferring behaviour from text-message data



How can we start to model this?

Denoting day *i*'s text-message count by C_i

↓

C is a random variable

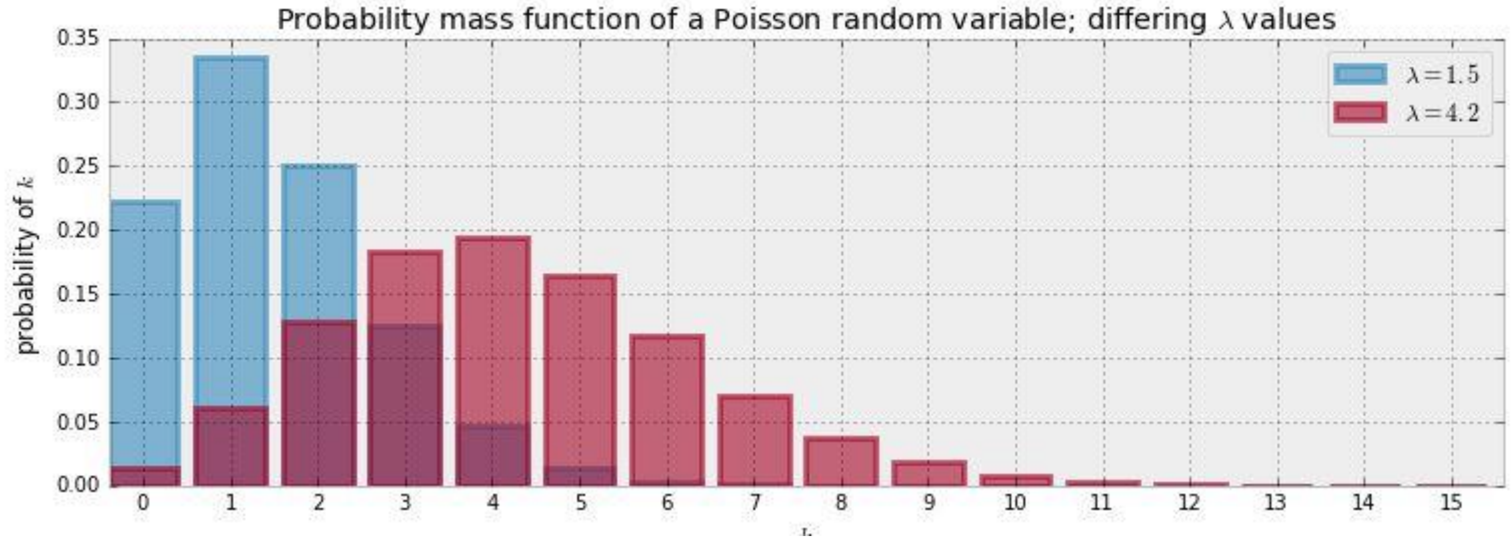
What can be the distribution of C ?

Probability Distributions (Memento)

Let Z be some random variable. Then associated with Z is a probability distribution function that assigns probabilities to the different outcomes Z can take

- **Discrete Case**: If Z is discrete, then its distribution is called a *probability mass function*, which measures the probability Z takes on the value k, denoted $P(Z = k)$
- **Continuous Case**: Instead of a probability mass function, a continuous random variable$Z$ has a *probability density function*, denoted f_Z and $P(a < z < b) = \int_a^b f_Z(z)dz$

Poisson Distribution (Memento)

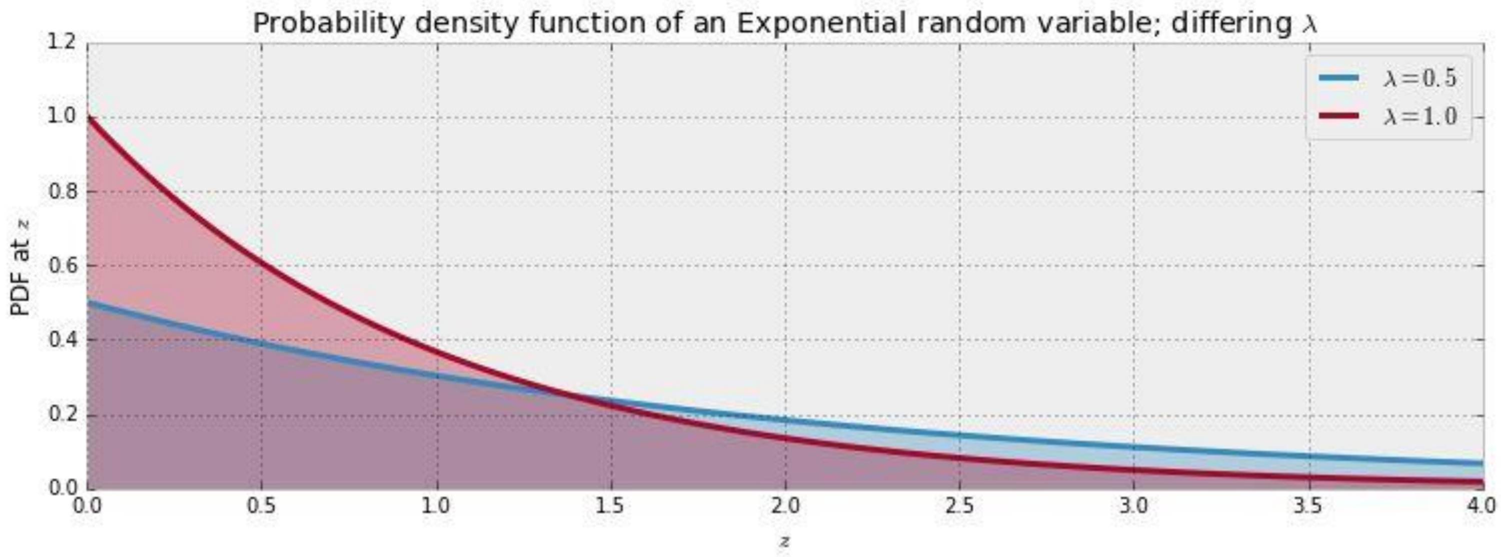


$$Z \sim \mathrm{Poi}(\lambda)$$

$$P(Z = k) = \frac{\lambda^k e^{-\lambda}}{k!}, \qquad k = 0,1,2, \dots , \qquad \lambda \in \mathbb{R}_{>0}$$

$$E(Z|\lambda) = \sum_{k=0}^{\infty} kP(Z = k) = \lambda$$

Exponential Distribution (Memento)



$$Z \sim \text{Exp}(\lambda)$$

$$f_Z(z|\lambda) = \lambda e^{-\lambda z}, \qquad z \geq 0$$

$$E(Z|\lambda) = \int_{-\infty}^{\infty} z f_Z(z|\lambda) dz = \frac{1}{\lambda}$$

Modeling

How can we start to model this?

Denoting day *i*'s text-message count by C_i

C is a random variable

What can be the distribution of C ?

A Poisson random variable is a very appropriate model for this type of count data

$$C \sim \mathrm{Poi}(\lambda)$$

Inferring behaviour from text-message data: prior distributions

$$C \sim \mathrm{Poi}(\lambda)$$

We are not sure what the value of the λ parameter really is, however. Looking at the chart above, it appears that the rate might become higher late in the observation period, which is equivalent to saying that λ increases at some point during the observations. (Recall that a higher value of λ assigns more probability to larger outcomes. That is, there is a higher probability of many text messages having been sent on a given day)

Inferring behaviour from text-message data: prior distributions

How can we represent this observation mathematically? Let's assume that on some day during the observation period (call it τ), the parameter λ suddenly jumps to a higher value. So we really have two λ parameters: one for the period before τ, and one for the rest of the observation period. In the literature, a sudden transition like this would be called a switch point:

$$\lambda = \begin{cases} \lambda_1 & \text{if } t < \tau \\ \lambda_2 & \text{if } t \geq \tau \end{cases}$$

If, in reality, no sudden change occurred and indeed $\lambda_1 = \lambda_2$, then the λs posterior distributions should look about equal.

Inferring behaviour from text-message data: prior distributions

We are interested in inferring the unknown λs. To use Bayesian inference, we need to assign prior probabilities to the different possible values of λ. What would be good prior probability distributions for λ_1 and λ_2? Recall that λ can be any positive number. As we saw earlier, the exponential distribution provides a continuous density function for positive numbers, so it might be a good choice for modeling λ_i. But recall that the exponential distribution takes a parameter of its own, so we'll need to include that parameter in our model. Let's call that parameter α.

$$\lambda_1 \sim \text{Exp}(\alpha)$$

$$\lambda_2 \sim \text{Exp}(\alpha)$$

Inferring behaviour from text-message data: prior distributions

α is called a hyper-parameter or parent variable. In literal terms, it is a parameter that influences other parameters. Our initial guess at α does not influence the model too strongly, so we have some flexibility in our choice. A good rule of thumb is to set the exponential parameter equal to the inverse of the average of the count data

$$\frac{1}{N}\sum_{i=0}^{N} C_i \approx E[\lambda|\alpha] = \frac{1}{\alpha}$$

Inferring behaviour from text-message data: prior distributions

What about τ? Because of the noisiness of the data, it's difficult to pick out a priori when τ might have occurred. Instead, we can assign a *uniform prior belief* to every possible day. This is equivalent to saying:

$$P(\tau = k) = \frac{1}{74}$$

Bayesian modeling is to think about how your data might have been generated

- We started by thinking "what is the best random variable to describe this count data?" A Poisson random variable is a good candidate because it can represent count data. So we model the number of sms's received as sampled from a Poisson distribution.
- Next, we think, "Ok, assuming sms's are Poisson-distributed, what do I need for the Poisson distribution?" Well, the Poisson distribution has a parameter λ.
- Do we know λ? No. In fact, we have a suspicion that there are two λ values, one for the earlier behaviour and one for the later behaviour. We don't know when the behaviour switches though, but call the switchpoint τ.
- What is a good distribution for the two λ s? The exponential is good, as it assigns probabilities to positive real numbers. Well the exponential distribution has a parameter too, call it α.

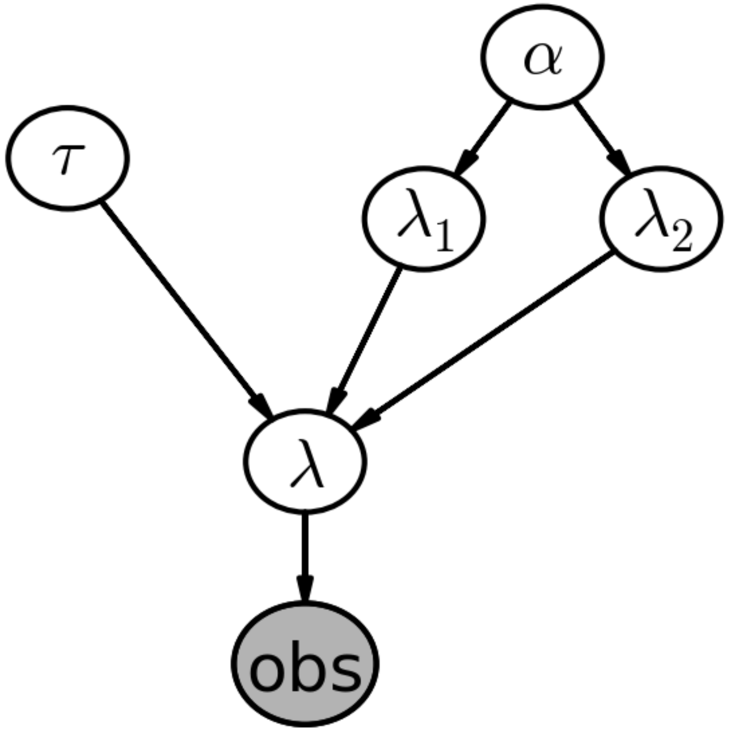
Bayesian modeling is to think about how your data might have been generated

- Do we know what the parameter α might be? No. At this point, we could continue and assign a distribution to α, but it's better to stop once we reach a set level of ignorance: whereas we have a prior belief about λ, ("it probably changes over time", "it's likely between 10 and 30", etc.), we don't really have any strong beliefs about α. So it's best to stop here.

  What is a good value for α then? We think that the λ s are between 10-30, so if we set α really low (which corresponds to larger probability on high values) we are not reflecting our prior well. Similar, a too-high alpha misses our prior belief as well. A good idea for α as to reflect our belief is to set the value so that the mean of λ, given α, is equal to our observed mean.

- We have no expert opinion of when τ might have occurred. So we will suppose τ is from a discrete uniform distribution over the entire timespan.

The Model



Generative Models

Generative models are statistical models that describe a joint distribution over three types of random variables:

- The "observed" random variables (often the "input space"), which are the random variables we have data for.
- The "latent" random variables, which are the random variables that play a role in the statistical model, but which are never observed (in the Bayesian setting, these are usually, at the very least, the parameters of the model).
- The "predicted" random variables, which are the random variables that represent the target predictions.

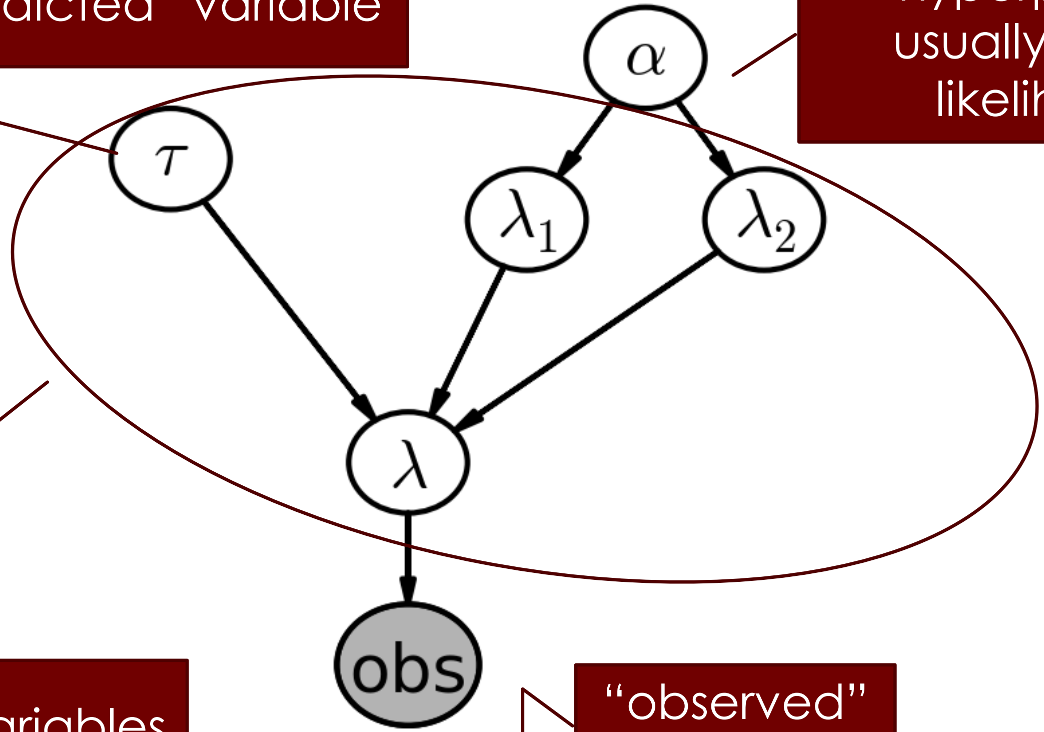
This categorization for the random variables in the joint distribution is not mutually exclusive (though observed random variables are never latent). For example, the predicted random variables can be also latent, such as in the unsupervised setting

Generative Stories

- A generative stories identify a joint distribution over the variables in the model, where this joint distribution is a product of several factors.
- The generative story picks an ordering for the random variables, and the *chain rule* is applied using that order to yield the joint distribution.
- Each factor can theoretically condition on all possible random variables that were generated before, but the independence assumptions in the model make some of these variables unnecessary to condition on.

$$p\left(X^{(1)}, X^{(2)}, \dots, X^{(n)}\right) = p\left(X^{(1)}\right) \prod_{1=2}^{n} p\left(X^{(i)} | X^{(1)}, X^{(2)}, \dots, X^{(i-1)}\right)$$

The Model



"predicted" variable

"hyperparameter" (not random) usually estimated by maximum likelihood (*empirical Bayes*)

"latent" variables

"observed" variables

PyMC Modeling

```
import numpy as np
import pymc as pm

count_data = np.loadtxt("txtdata.csv")
n_count_data = len(count_data)

alpha = 1.0 / count_data.mean()  # Recall count_data is the
                                 # variable that holds our txt counts
lambda_1 = pm.Exponential("lambda_1", alpha)
lambda_2 = pm.Exponential("lambda_2", alpha)
tau = pm.DiscreteUniform("tau", lower=0, upper=n_count_data)
```

PyMC Modeling

```
@pm.deterministic
def lambda_(tau=tau, lambda_1=lambda_1, lambda_2=lambda_2):
    out = np.zeros(n_count_data)
    out[:tau] = lambda_1  # lambda before tau is lambda1
    out[tau:] = lambda_2  # lambda after (and including) tau is lambda2
    return out
```

Simulation

PyMC Simulation

```
n_count_data = 80
alpha = 1.0 / 20.0

observation = pm.Poisson("obs", lambda_, size=80)
model = pm.Model([observation, lambda_1, lambda_2, tau])

mcmc = pm.MCMC(model)
mcmc.sample(40000, 10000, 1)

tau_samples = mcmc.trace('tau')[:]
obs_samples = mcmc.trace('obs')[:]

data = obs_samples[10000][:]
tau = tau_samples[10000]
```

PyMC Simulation

Having the model, we can simulate a possible realization of the dataset:

1. Specify when the user's behaviour switches by sampling from DiscreteUniform(0, 80):

```
tau = pm.rdiscrete_uniform(0, 80)
print(tau)

[Output]:
21
```

2. Draw λ_1 and λ_2 from an $\mathrm{Exp}(\alpha)$ distribution:

```
alpha = 1. / 20.
lambda_1, lambda_2 = pm.rexponential(alpha, 2)
print(lambda_1, lambda_2)

[Output]:
20.7789591495 62.1938883352
```

PyMC Simulation

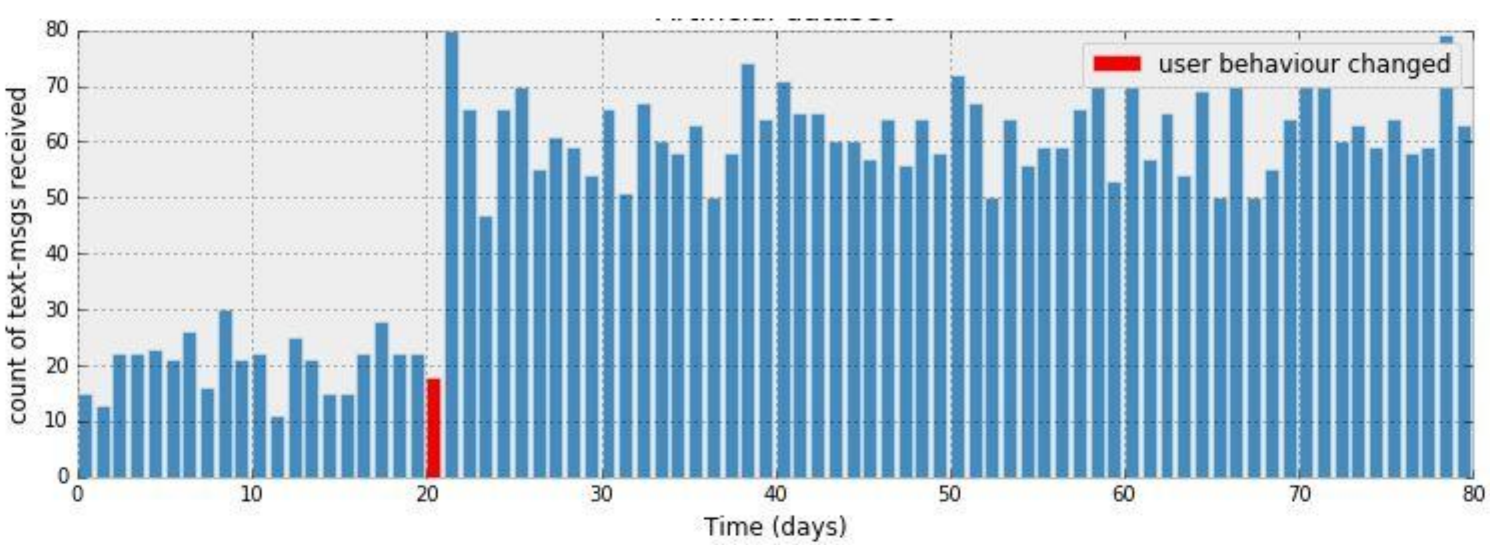
3. For days before τ, represent the user's received SMS count by sampling from $\mathrm{Poi}(\lambda_1)$, and sample from $\mathrm{Poi}(\lambda_2)$ for days after τ :

```
data = np.r_[pm.rpoisson(lambda_1, tau), pm.rpoisson(lambda_2, 80 - tau)]
```

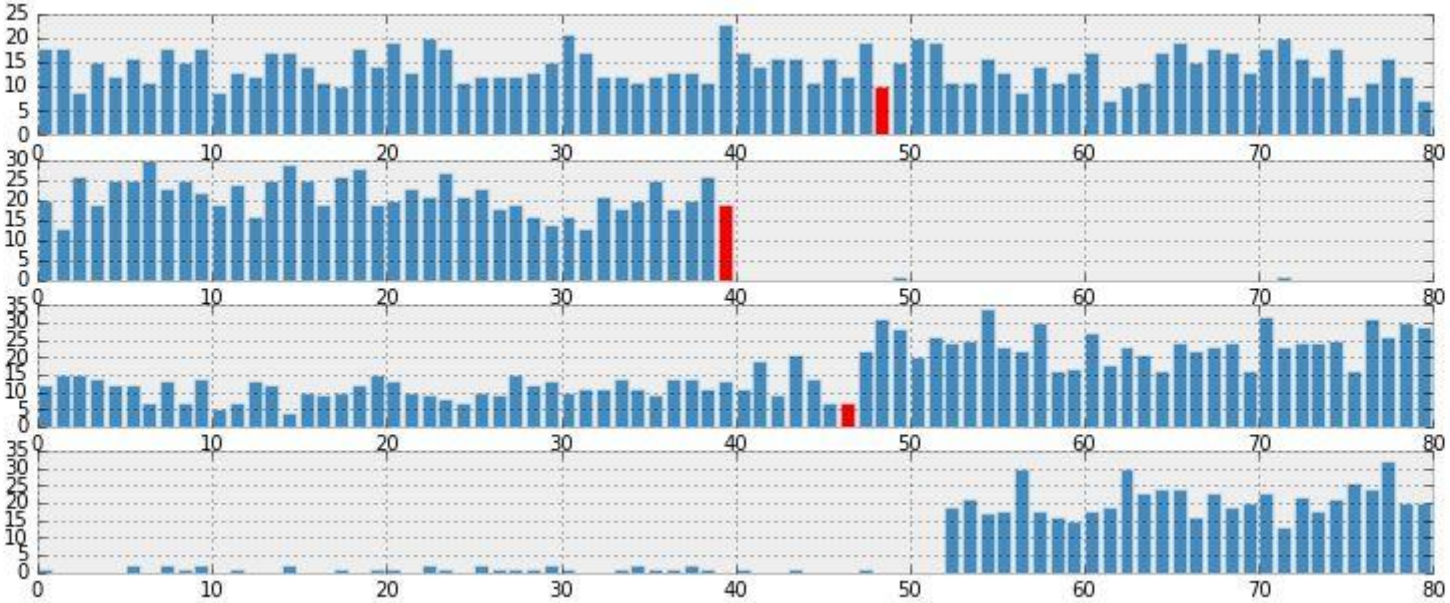
4. Plot the artificial dataset:

```
plt.bar(np.arange(80), data, color="#348ABD")

plt.bar(tau - 1, data[tau - 1], color="r", label="user behaviour changed")

plt.xlabel("Time (days)")

plt.ylabel("count of text-msgs received")

plt.title("Artificial dataset")

plt.xlim(0, 80)

plt.legend();
```

Artificial Dataset



More Examples of Artificial Datasets



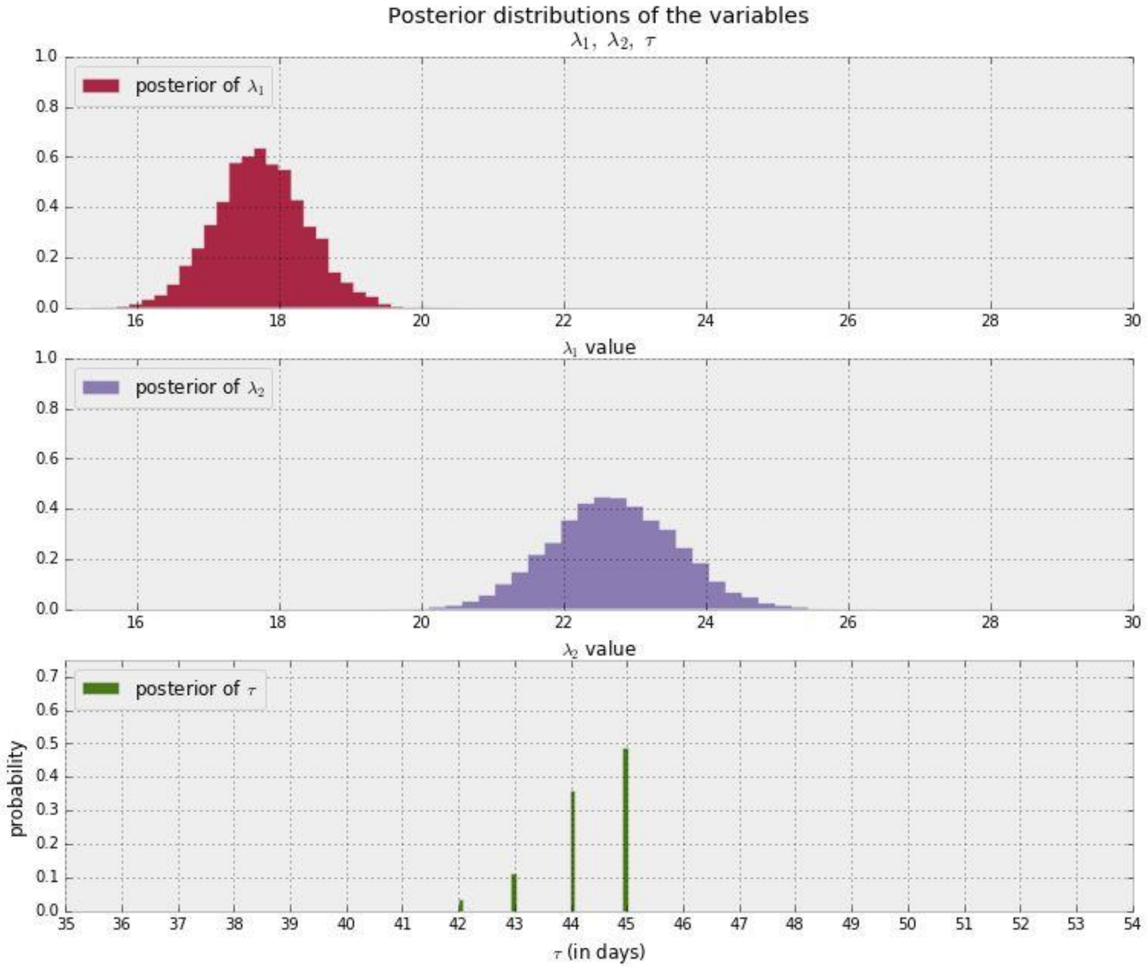
Inference

PyMC Inference

```
observation = pm.Poisson("obs", lambda_, value=count_data, observed=True)

model = pm.Model([observation, lambda_1, lambda_2, tau])

mcmc = pm.MCMC(model)
mcmc.sample(40000, 10000, 1)

lambda_1_samples = mcmc.trace('lambda_1')[:]
lambda_2_samples = mcmc.trace('lambda_2')[:]
tau_samples = mcmc.trace('tau')[:]
```

Criticism of the Model

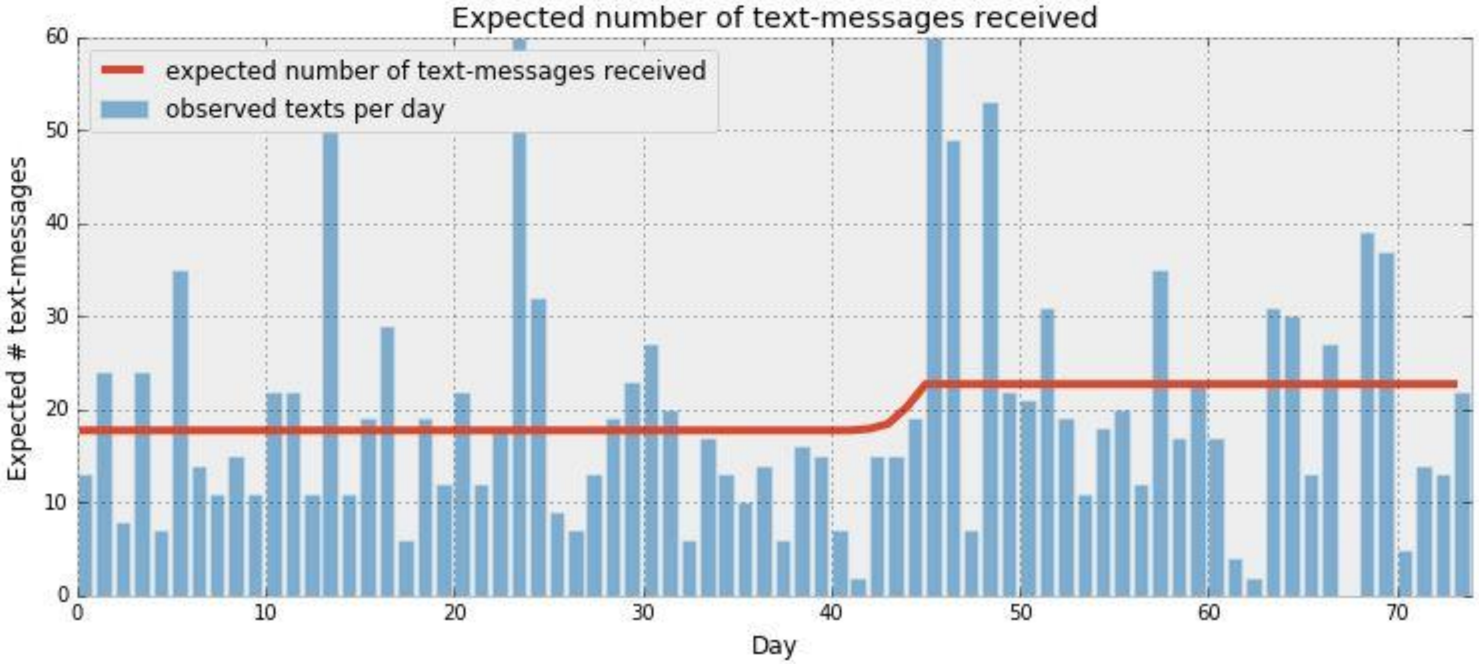
Inferring behaviour from text-message data: posterior distributions



Expected number of texts per day

```
# tau_samples, lambda_1_samples, lambda_2_samples contain
# N samples from the corresponding posterior distribution
N = tau_samples.shape[0]
expected_texts_per_day = np.zeros(n_count_data)
for day in range(0, n_count_data):
    # ix is a bool index of all tau samples corresponding to
    # the switchpoint occurring prior to value of 'day'
    ix = day < tau_samples
    # Each posterior sample corresponds to a value for tau.
    # for each day, that value of tau indicates whether we're "before"
    # (in the lambda1 "regime") or
    #  "after" (in the lambda2 "regime") the switchpoint.
    # by taking the posterior sample of lambda1/2 accordingly, we can average
    # over all samples to get an expected value for lambda on that day.
    # As explained, the "message count" random variable is Poisson distributed,
    # and therefore lambda (the poisson parameter) is the expected value of
    # "message count".
    expected_texts_per_day[day] = (lambda_1_samples[ix].sum()
                                   + lambda_2_samples[~ix].sum()) / N
```

Expected number of texts per day



Determining statistically if the two λs are indeed different

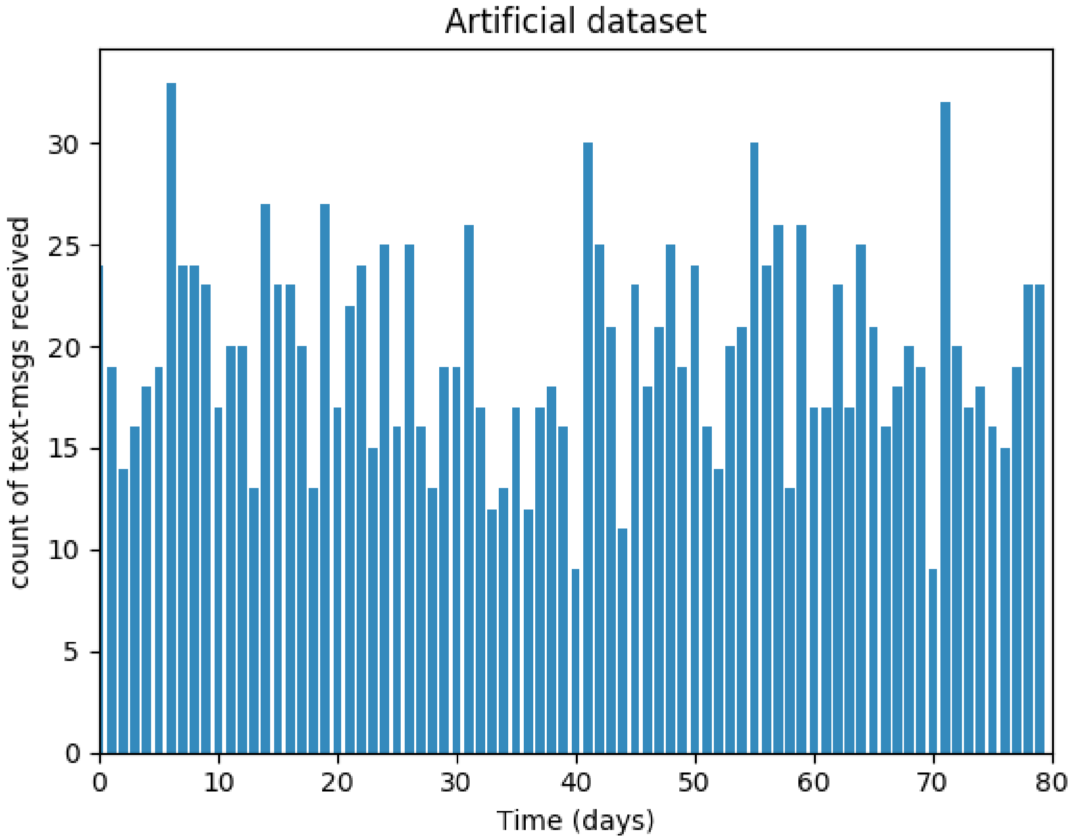
- We visually inspected the posteriors of λ_1 and λ_2 to declare them different. How can we make this decision more formal?
- One way is to compute $P(\lambda_1 < \lambda_2|\text{data})$; that is, what is the probability that the true value of λ_1 is smaller than λ_2 given the data we observed. Using samples from the posteriors, this computation is very simple – we compute the fraction of times that a sample from the posterior of λ_1 is less than one from λ_2:

```
print((lambda_1_samples < lambda_2_samples).mean())
```

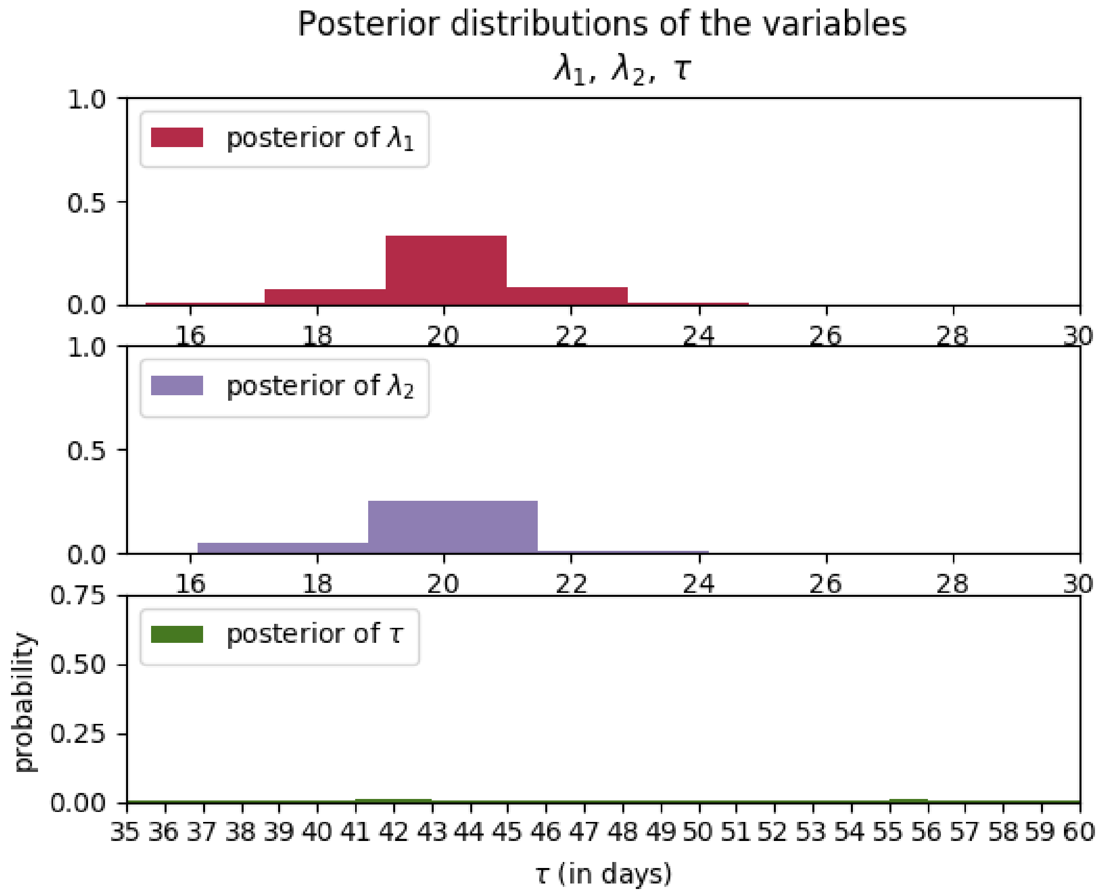
```
[Output]:

0.9998
```

Artificial Dataset

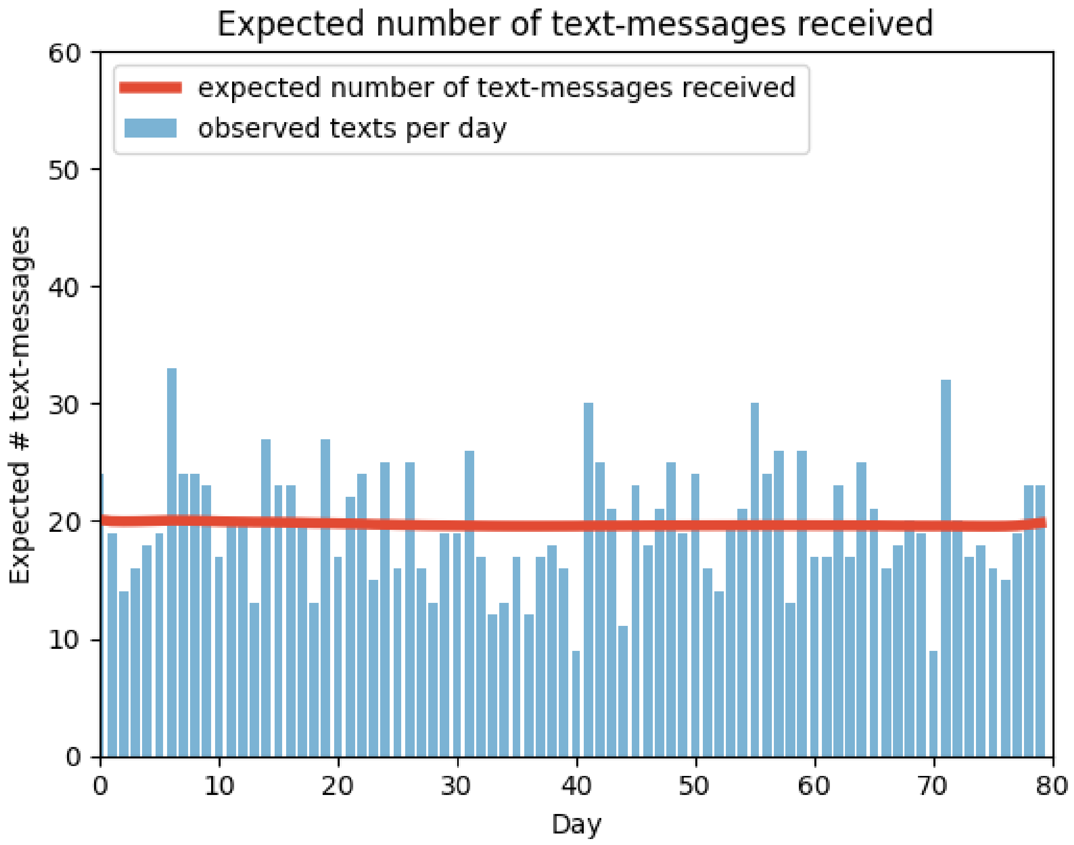
Artificial data set was generated from a Poisson distribution with $\lambda = 19.255$



Inferring behaviour from artificial data: posterior distributions



Expected number of texts per day



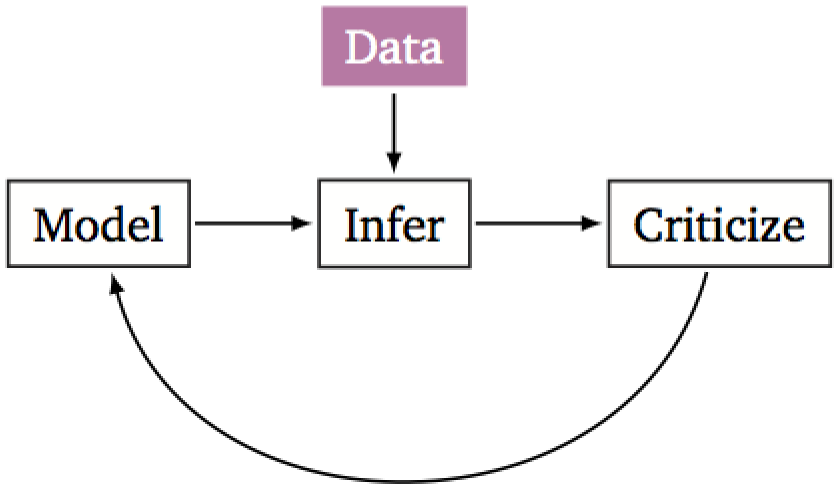
Determining statistically if the two λs are indeed different

```
print((lambda_1_samples < lambda_2_samples).mean())
```

```
[Output]:

0.3962
```

Probabilistic Modeling

- Build a probabilistic model of the phenomena.
- Reason about the phenomena given model and data.
- Criticize the model, revise and repeat.


Data
Model
Infer
Criticize