Tutorial 4

Statistical Computation and Analysis

Spring 2025

Tutorial Outline

- Probabilistic Programming Languages
- Analyzing the posterior
  - Arviz
  - KDE
  - HDI
  - Savage-Dickey Density Ratio
  - ROPE
- PyTensor

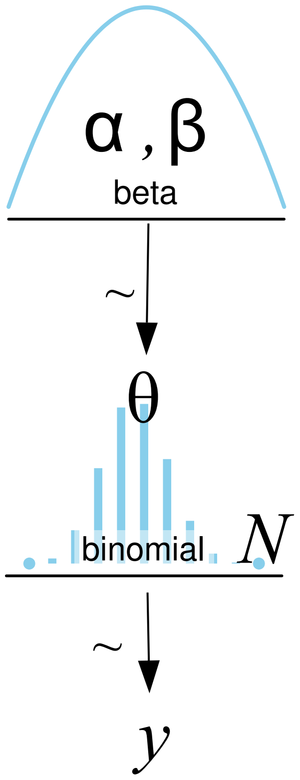
Probabilistic Programming Languages

- Challenge
  - Fully probabilistic models often lead to analytically intractable expressions
- Probabilistic Programming Languages
  - Allow clear separation between model creation and inference
  - Users specify a full probabilistic model by writing a few lines of code, and then inference follows automatically

Translating a Graphical Model to a PPL

- Coin flipping

Graphical model



Equations

$$\theta \sim \mathrm{Beta}(\alpha, \beta)$$

$$y \sim \mathrm{Binom}(\theta, N)$$

PyMC (a PPL)

```
with pm.Model() as our_first_model:
    θ = pm.Beta('θ', alpha=1., beta=1.)
    y = pm.Bernoulli('y', p=θ, observed=data)
    idata = pm.sample(1000, random_seed=4591)
```

Translating a Graphical Model to a PPL

- Coin flipping

```
with pm.Model() as our_first_model:
    θ = pm.Beta('θ', alpha=1., beta=1.)
    y = pm.Bernoulli('y', p=θ, observed=data)
    idata = pm.sample(1000, random_seed=4591)
```

Line 1: creates a container for our model. Everything inside the with block will be automatically added to our_first_model.

Line 2: prior

Line 3: likelihood

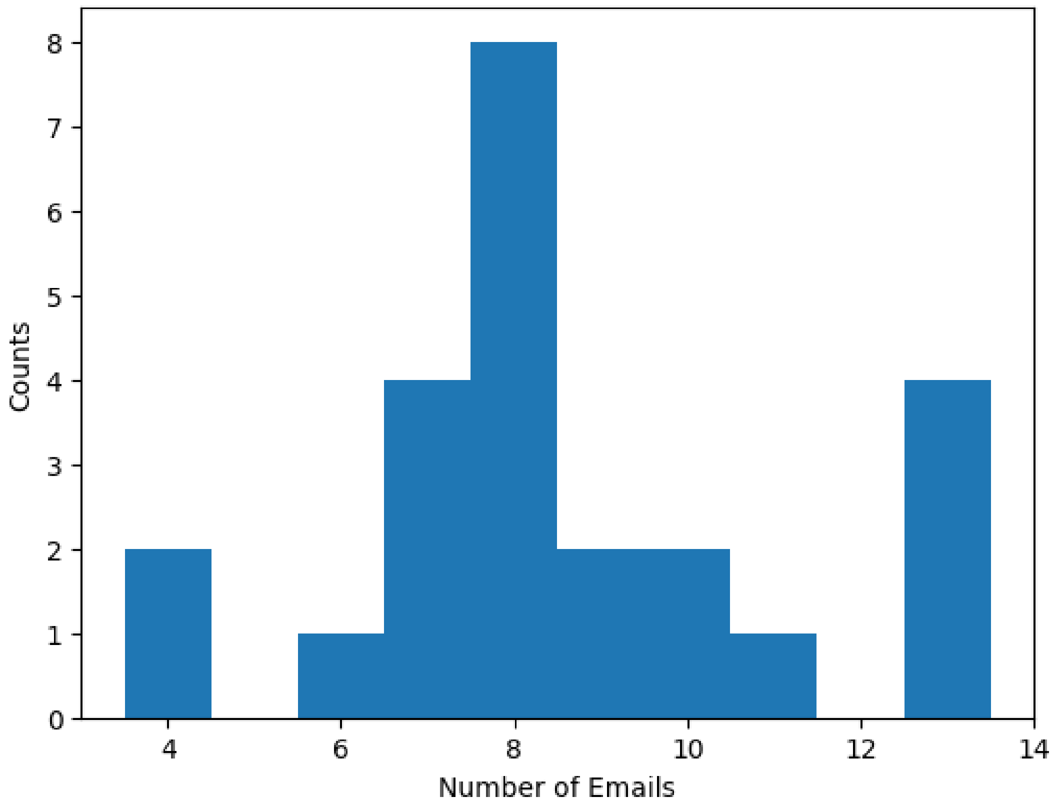
Line 4: computes the posterior using numerical methods.

Translating a Graphical Model to a PPL

- The use of numerical methods solves the challenge of posteriors we cannot compute.
- However, what we get are samples from the posterior, rather than the posterior itself.
- Every time we run the inference, the samples will change.
- **If the inference process works as expected, the samples will be representative of the posterior distribution, and we will obtain the same conclusion from any of those samples.**

Translating a Graphical Model to a PPL

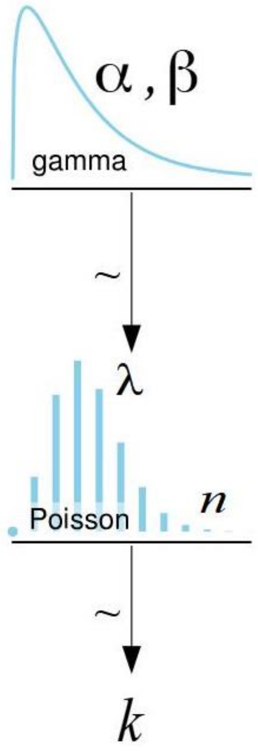
- Number of emails
  - Poisson likelihood
  - Gamma prior on λ



Translating a Graphical Model to a PPL

- Number of emails

Graphical model



Equations

$$\lambda \sim Gamma(\alpha, \beta)$$
$$k \sim Poisson(\lambda)$$

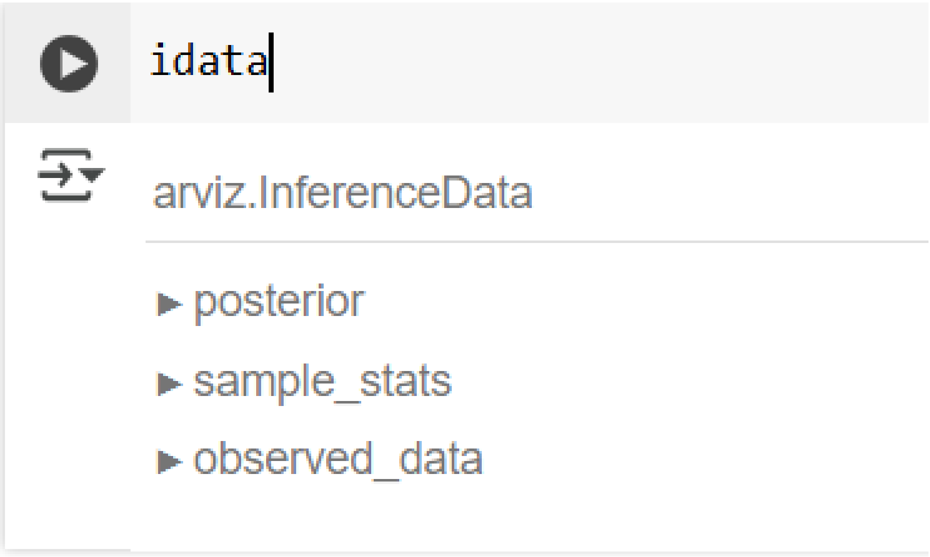
PyMC (a PPL)

```
coords = {"data": np.arange(n)}

with pm.Model(coords = coords) as our_first_model:
    lambda_ = pm.Gamma('lam', alpha = 1.68, beta = 0.0569)
    k = pm.Poisson('k', mu = lambda_, observed=data, dims = 'data')
    idata = pm.sample(1000, chains = 4)
```

Inference Data Object

- idata – what we get back from the inference process
- Container for all the data generated by PyMC.



- Xarray object
- We will use the Arviz library to analyze the posterior

Inference Data Object

- Observed data

▸ posterior

▸ sample_stats

▾ observed_data

xarray.Dataset

▸ Dimensions:	(**data**: 24)		
▾ Coordinates:			
data	(data)	int64	0 1 2 3 4 5 6 ... 18 19 20 21 22 23
▾ Data variables:			
k	(data)	int64	10 5 2 11 9 7 8 ... 9 6 12 8 6 8 6
▸ Indexes: (1)			
▸ Attributes: (4)			

Number of samples collected (24)

Their coordinates – here just 0-23

The values – the numbers of emails received by each student

Inference Data Object

- Posterior

```
idata
```

arviz.InferenceData

▼ posterior

xarray.Dataset

▶ Dimensions: (**chain**: 4, **draw**: 1000)

4 chains of 1000 samples (draws) each

▼ Coordinates:

chain	(chain)	int64	0 1 2 3
draw	(draw)	int64	0 1 2 3 4 5 ... 995 996 997 998 999

▼ Data variables:

lambda	(chain, draw)	float64	9.073 9.011 7.855 ... 8.935 8.721

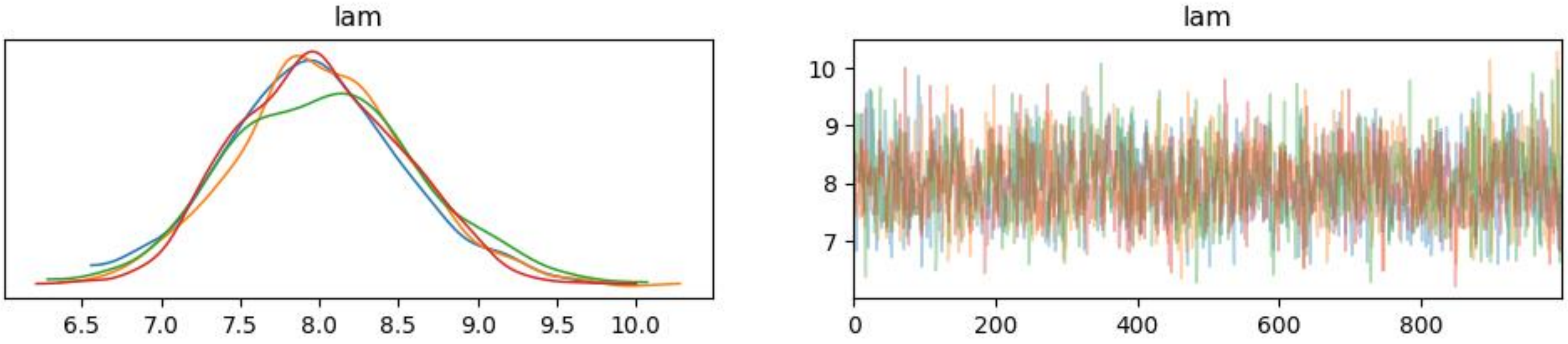
▶ Indexes: (2)

▶ Attributes: (6)

The lambda values (4, 1000)

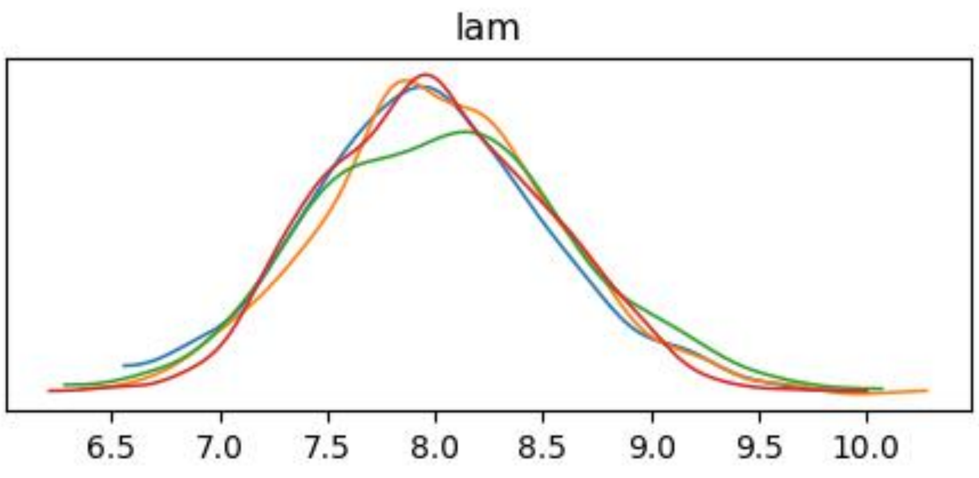
Analyzing the Posterior

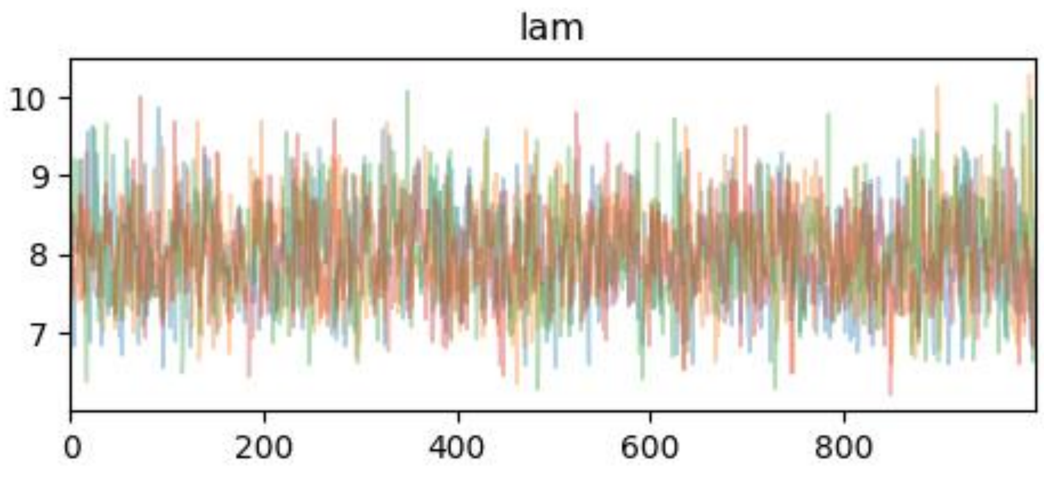
- Look at the results
  - az.plot_trace(idata)



Analyzing the Posterior

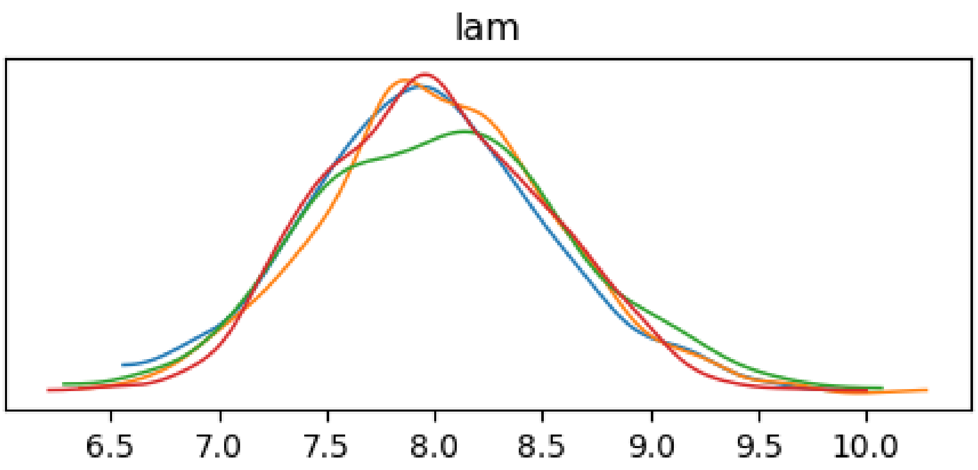
- Left: the four chains showing the values of the posterior distribution **(KDE plot = Kernel Density Estimation)**
- Right: For now – the overlap between the four colors shows that the sampler is working.

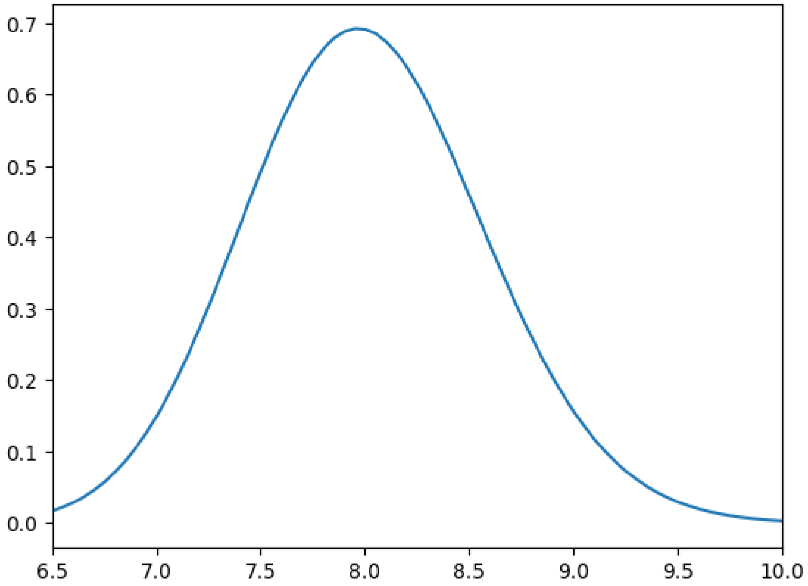




Analyzing the Posterior

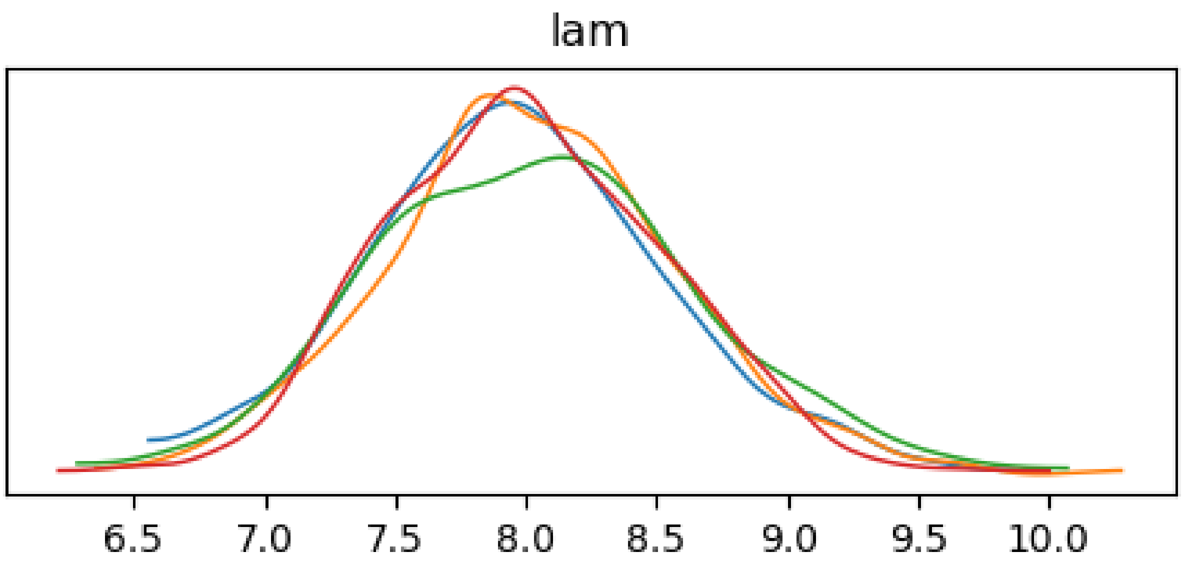
- Compare our analytical results (from Tutorial 3) to these:





Analyzing the Posterior

- We want all the chains to overlap as much as possible.
    - How much?
        - We'll learn statistics with thresholds we can calculate
    - For now, It looks pretty good.

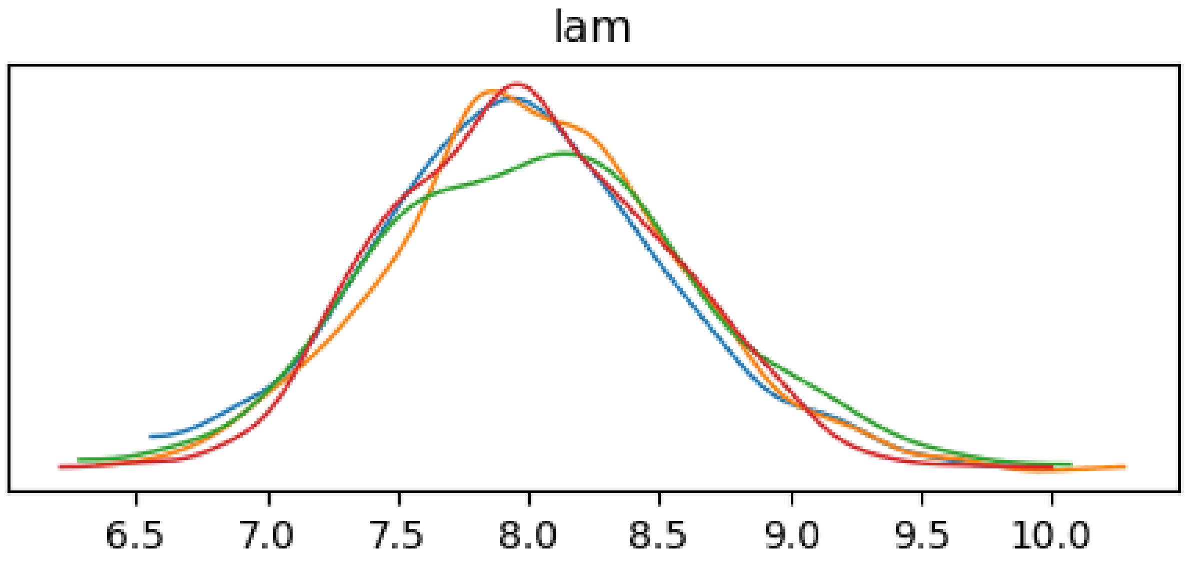


Analyzing the Posterior

- How close are the means of the different chains?

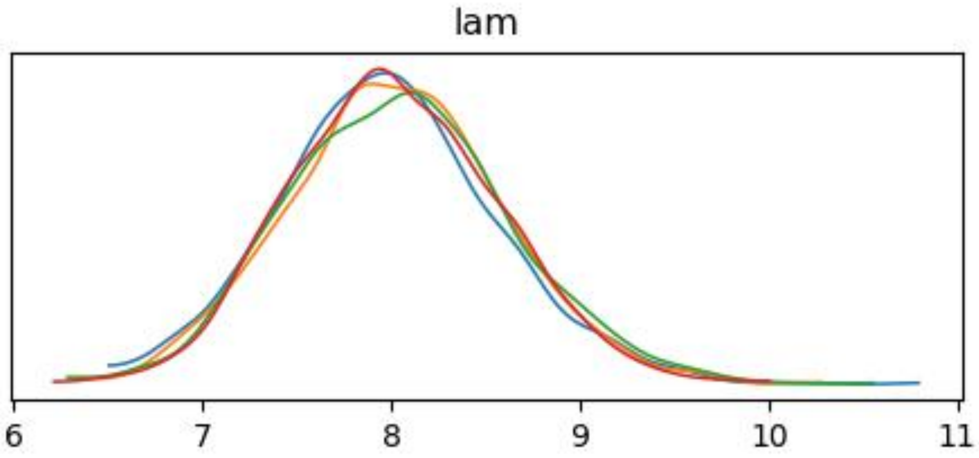
```
ms = idata.posterior.mean(dim = 'draw')
print(f'The means of the 4 chains are: {np.round(ms.lam.to_numpy(), 3)}')
```

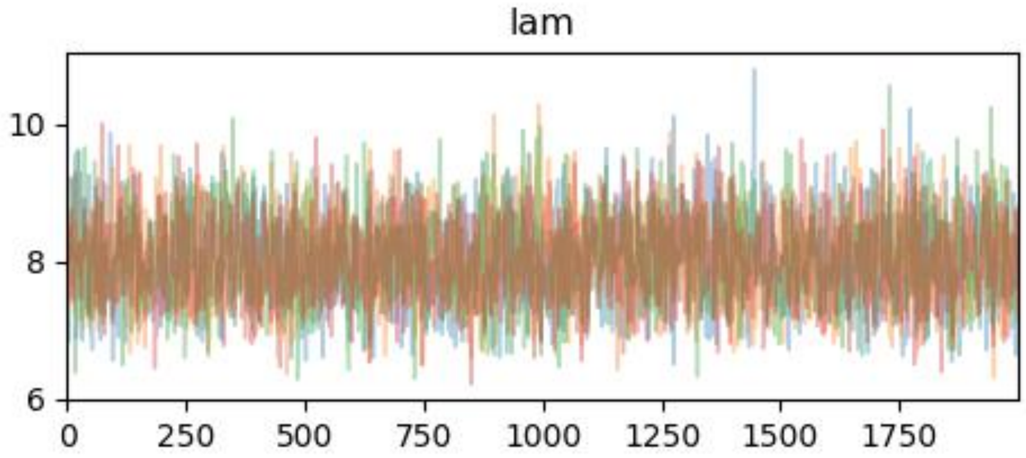
```
The means of the 4 chains are: [7.97  8.028 8.036 7.998]
```



Analyzing the Posterior

- How can we get the chains to look more similar?
  - Sample more (2000)

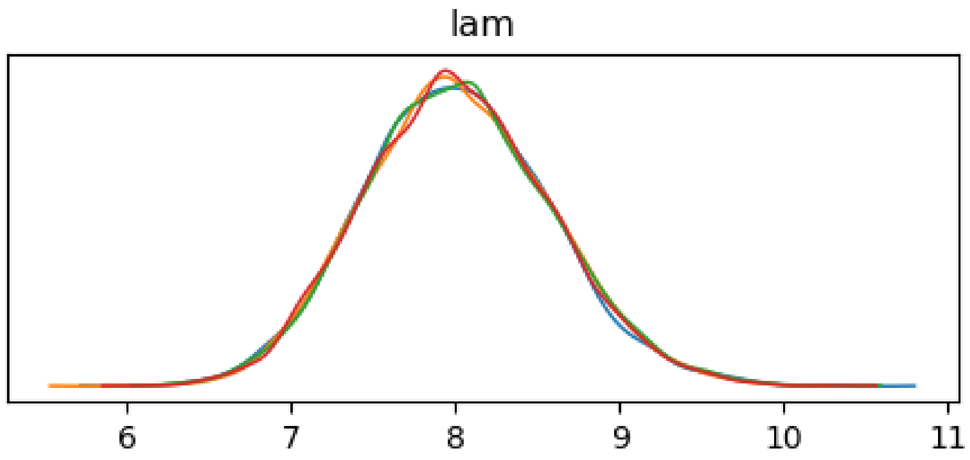


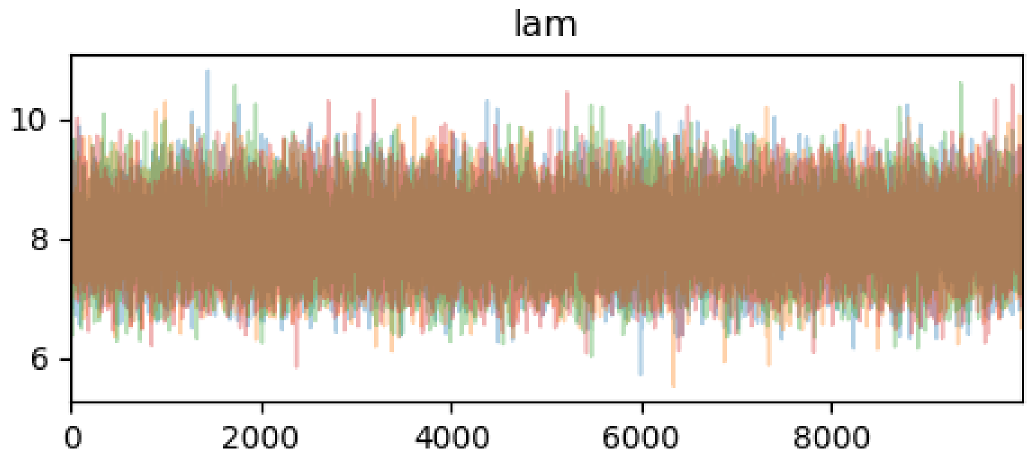



```
The means of the 4 chains are: [7.984 8.029 8.043 8.016]
```

Analyzing the Posterior

- How can we get the chains to look more similar?
  - Sample more (10,000)






```
The means of the 4 chains are: [8.002 8.013 8.018 8.016]
```

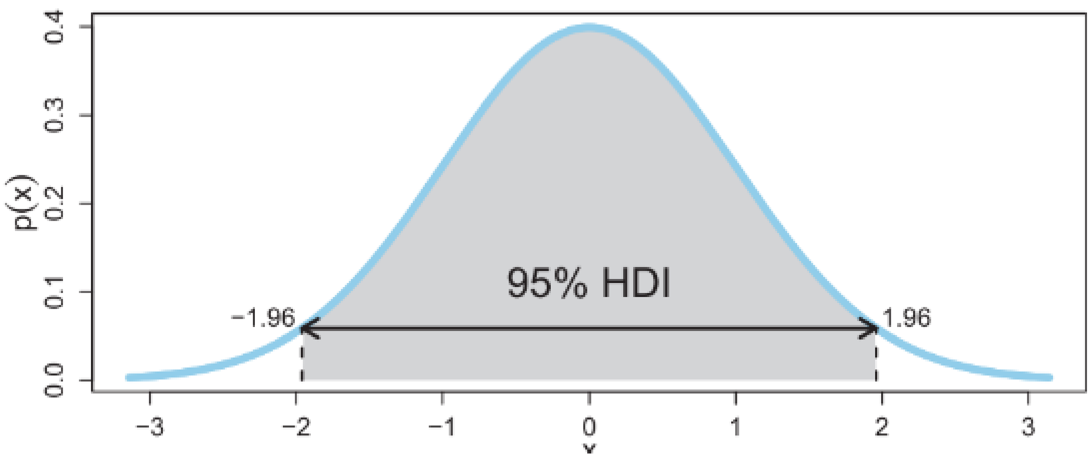
Highest Density Interval (HDI)

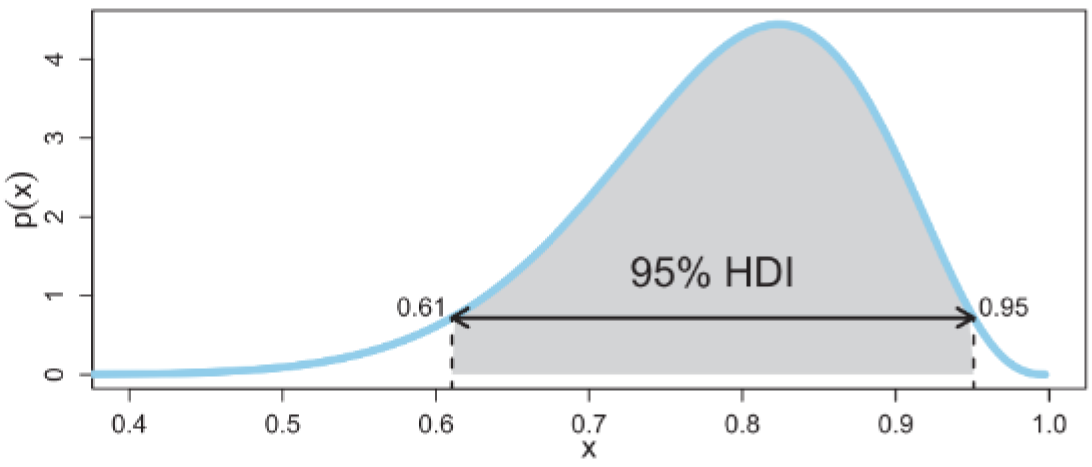
- We computed the mean
- We can also compute other statistics, such as the HDI
- The Z% HDI is the interval of values containing Z% of the probability

$$\int_{x \in Z\%\,\mathrm{HDI}} dx\; p\left(x\right) = \frac{Z}{100}$$

Highest Density Interval (HDI)

- There are a lot of intervals like this.
- The HDI is the shortest interval containing Z% of the distribution.
- Equivalent: the probability of all values inside the HDI are higher than those out of it.

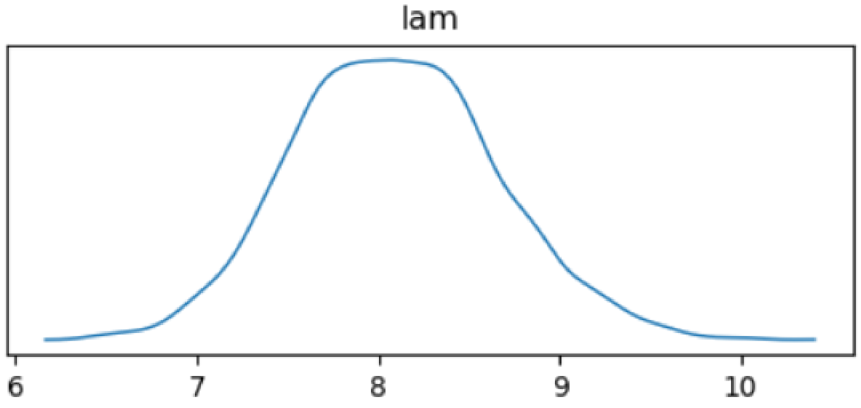




Highest Density Interval (HDI)

- The default in Arviz is the 94% HDI, so that's what we'll generally use.
- Often, the 50% HDI is also reported
- Before we plot, we can in general combine the 4 chains to 4000 samples:

```
The total mean of the 4000 samples is: 8.095
```



Highest Density Interval (HDI)

- We can also get the data as samples instead of as separate chains using az.extract

```
#we can also get all the data as 4000 samples using extract, instead of as four seperate chains
az.extract(idata)

#if you specify number of samples, you get random sampling out of the 4000
az.extract(idata, num_samples = 3500)
```

xarray.Dataset

▸ Dimensions: (**sample**: 3500)

▾ Coordinates:

sample	(sample)	object	MultiIndex
chain	(sample)	int64	2 1 2 0 1 1 2 0 ... 2 1 1 3 3 2 0 1
draw	(sample)	int64	961 68 332 587 ... 208 289 484 635

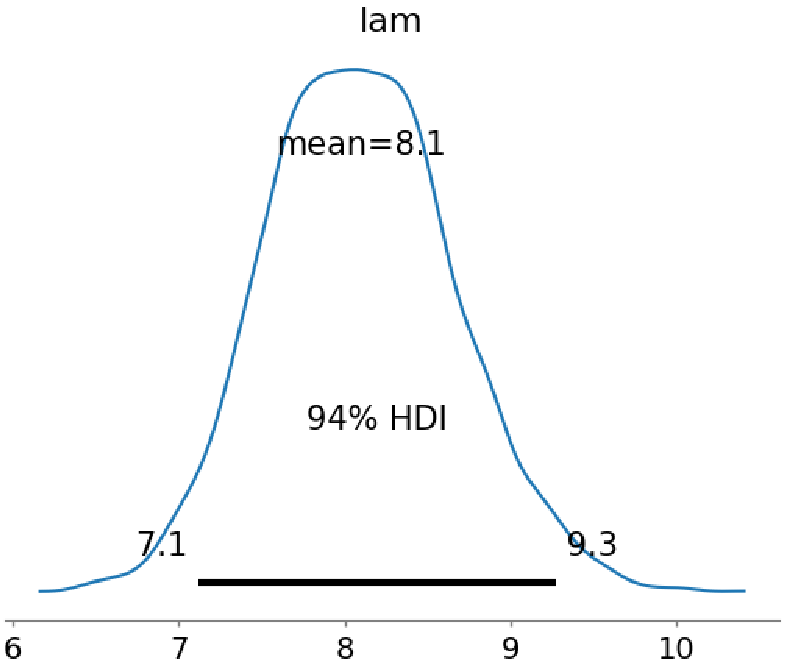
▾ Data variables:

lam	(sample)	float64	9.237 8.037 7.363 ... 8.934 7.618

Highest Density Interval (HDI)

- We can plot HDI and compute the values

```
#plotting the posterior with the HDI
az.plot_posterior(idata)
```




```
#and getting the values in a table
az.summary(idata, kind = 'stats').round(2)
```

	mean	sd	hdi_3%	hdi_97%
lam	8.1	0.57	7.12	9.27

Lower limit

Upper limit

Help Pages

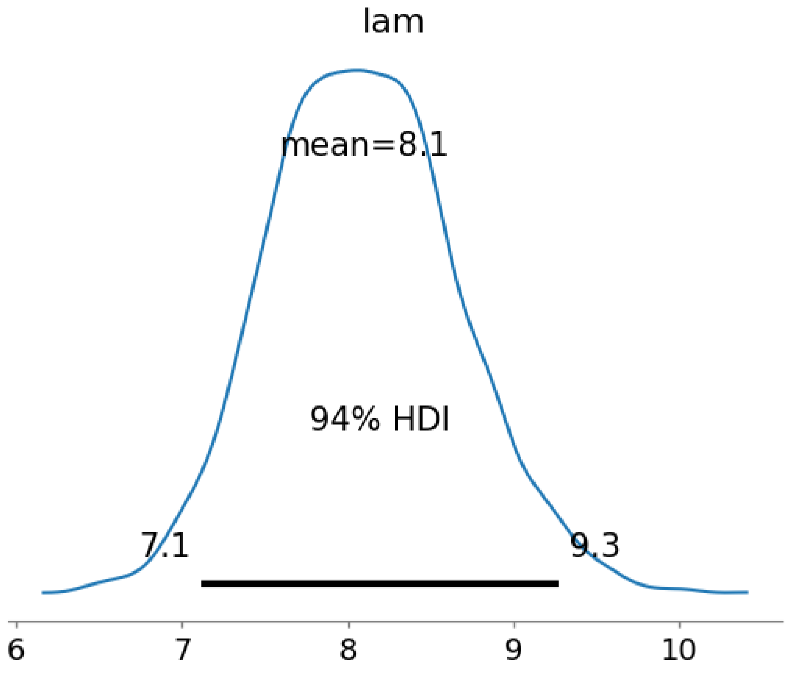
- Arviz: https://python.arviz.org/en/stable/api/index.html
- PyMC: https://www.pymc.io/projects/docs/en/stable/api.html
- Preliz: https://preliz.readthedocs.io/en/latest/index.html\

Posterior Based Decisions

- Sometimes we would like to do more than describe the posterior
- Sometimes we'd like to make a decision
  - Is the coin fair?
  - Is the person sick?
- Do biomedical students receive 9 emails a day?

Posterior Based Decisions

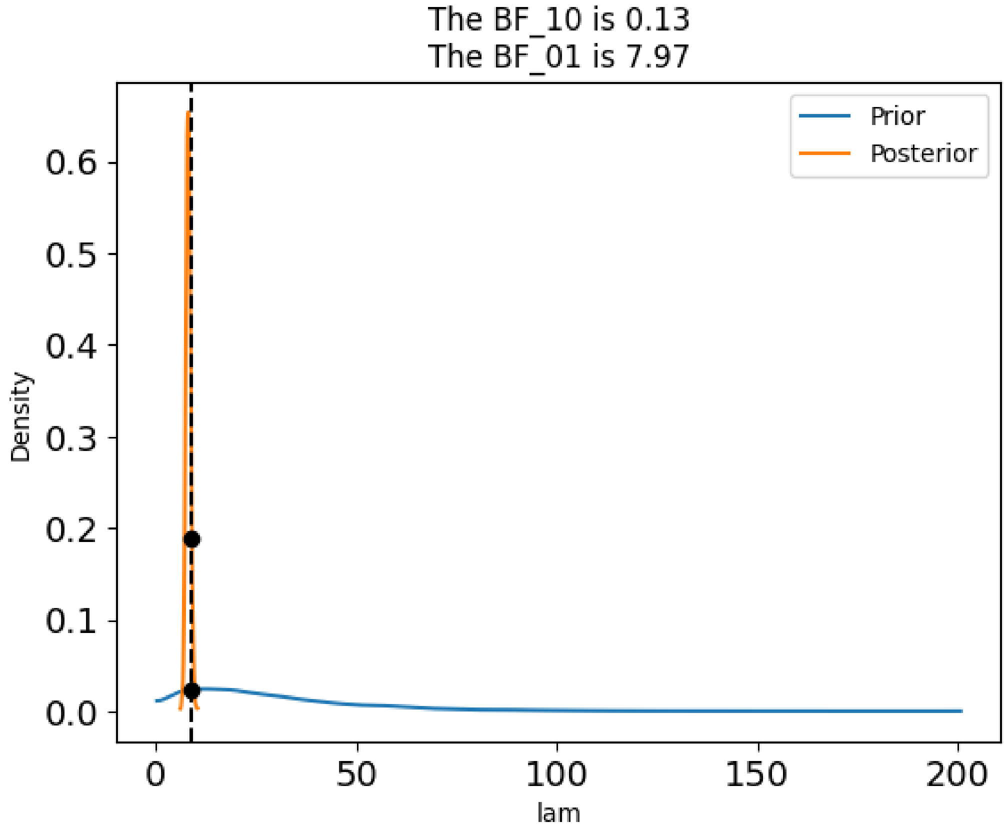
- 9 is inside the 94% HDI.



- We can compute a value called the **Savage-Dickey density ratio**.

Savage-Dickey Density Ratio

- The ratio of the posterior and prior densities at that value.
  - Evaluates how much support the posterior provides for a given value
- BF_01 = 7.97
  - The value of 9 emails is 7.97 times more likely under the posterior than under the prior.
- BF_10 = 1/BF_01



Savage-Dickey Density Ratio

- How do we a decision?
  - Rule of thumb

BF_{01}	Interpretation
< 3.2	Not worth mentioning
3.2 to 10	Substantial
10 to 100	Strong
> 100	Decisive

ROPE

- Region of Practical Equivalence
  - For example, students will not each get exactly 9 emails a day.
  - Let’s say we think between 8-10 is pretty much the same as 9.
  - ROPE is defined using domain knowledge.

ROPE

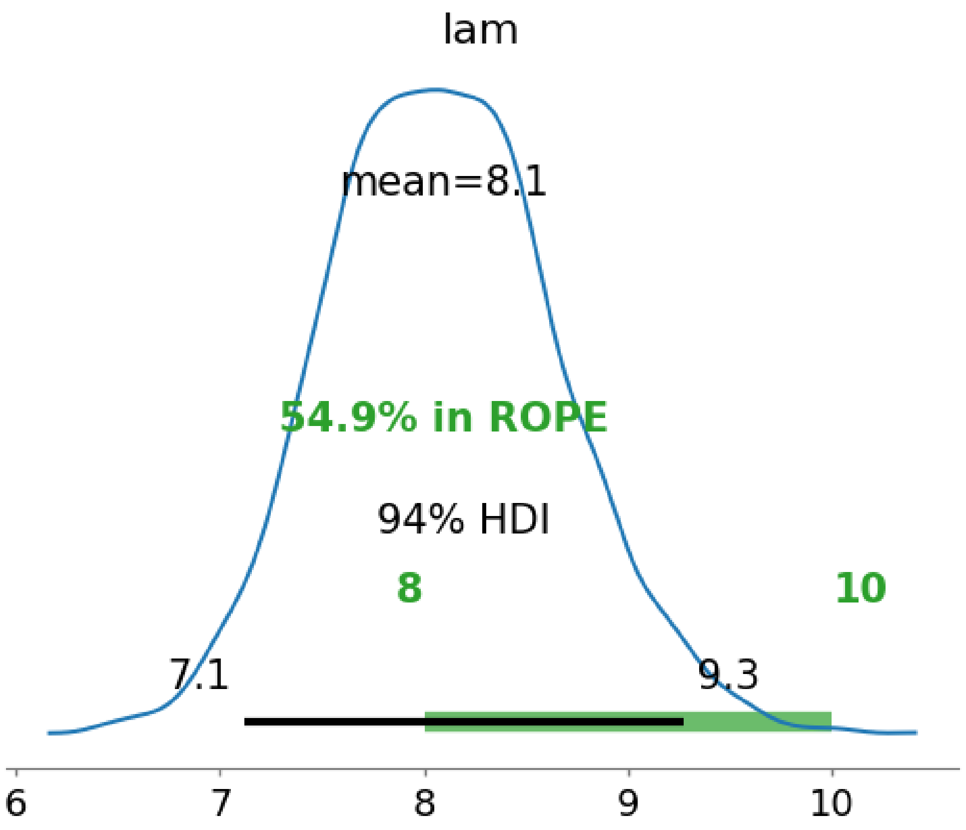
- If the ROPE and HDI do not overlap
  - Students don’t receive 9 emails a day
- If the ROPE contains the entire HDI
  - Students receive 9 emails a day
- If the ROPE and HDI partially overlap
  - We can’t say if students do or do not receive 9 emails a day

ROPE

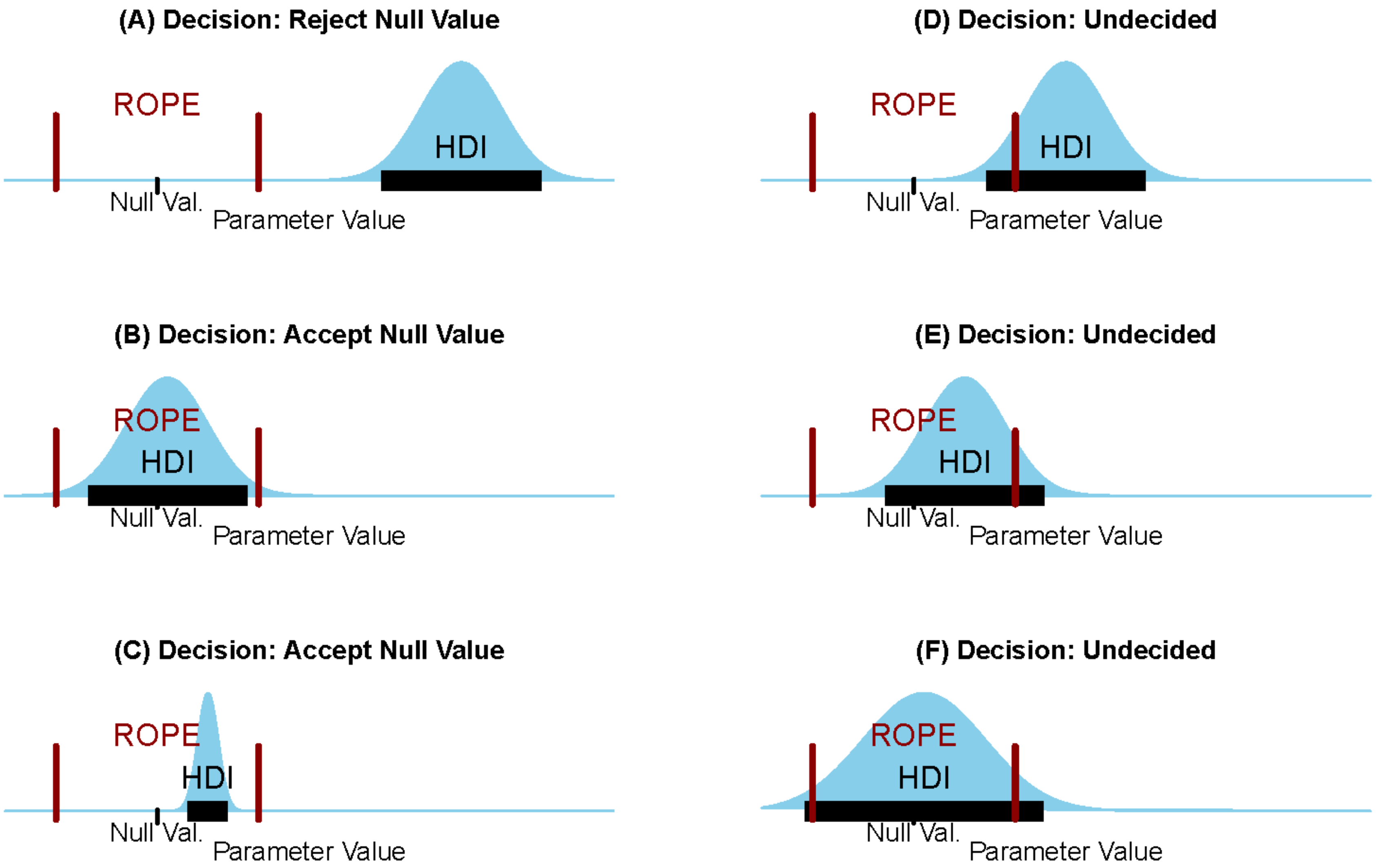
- If the ROPE and HDI do not overlap
  - Students don’t receive 9 emails a day
- If the ROPE contains the entire HDI
  - Students receive 9 emails a day
- If the ROPE and HDI partially overlap
  - We can’t say if students do or do not receive 9 emails a day
- If the definition of the ROPE changes our conclusions – maybe question it.

ROPE

- Can we conclude if students receive 9 emails a day?
  - No



ROPE


(A) Decision: Reject Null Value
ROPE
HDI
Null Val.
Parameter Value
(B) Decision: Accept Null Value
ROPE
HDI
Null Val.
Parameter Value
(C) Decision: Accept Null Value
ROPE
HDI
Null Val.
Parameter Value
(D) Decision: Undecided
ROPE
HDI
Null Val.
Parameter Value
(E) Decision: Undecided
ROPE
HDI
Null Val.
Parameter Value
(F) Decision: Undecided
ROPE
HDI
Null Val.
Parameter Value

PyTensor

- Allows for defining, optimizing, and efficiently evaluating mathematical expressions involving multi-dimensional arrays.
- A tensor is a generalized mathematical structure that can represent scalars, vectors, matrices, and higher-dimensional arrays.
- Unlike other variables, tensors don’t have to have a defined shape to be manipulated.
- This can be useful for cases in which tensors exist without a concrete shape until execution or have dimensions that vary per sample.
  - Used in deep learning.
- Computational graphs enable automatic differentiation for efficient gradient computation using the chain rule.