Foundations of Bayesian Modeling with PyMC

Ulf Aslak, June 11 2024

PyMC Labs

Make Better Decisions

- Authors of leading **open-source** data science tools: PyMC, PyMC-Marketing, & CausalPy
- Decades of **field experience** across key sectors like marketing analytics, biotech and sports analytics
- Over 50% of our team holds a **PhD** & includes 5 former **professors**
- Preferred **partner** for **industry leaders** facing complex challenges: Colgate, Roche, Takeda, LiveNation, HelloFresh...

Ulf Aslak

PhD | Data Scientist

- 🇩🇰

- **Ex academic:** PhD in Complex Systems, published eight peer-reviewed papers.

- **Worked in marketing:** Built marketing mix models at a major nordic agency.

- **Now freelance:** But most of my work is with PyMC Labs.

Overview

Foundations of Bayesian Modeling with PyMC

- What is PyMC?
- Bayesian Modeling
- A modeling example
- Workshop teaser
- Q&A

What is PyMC?

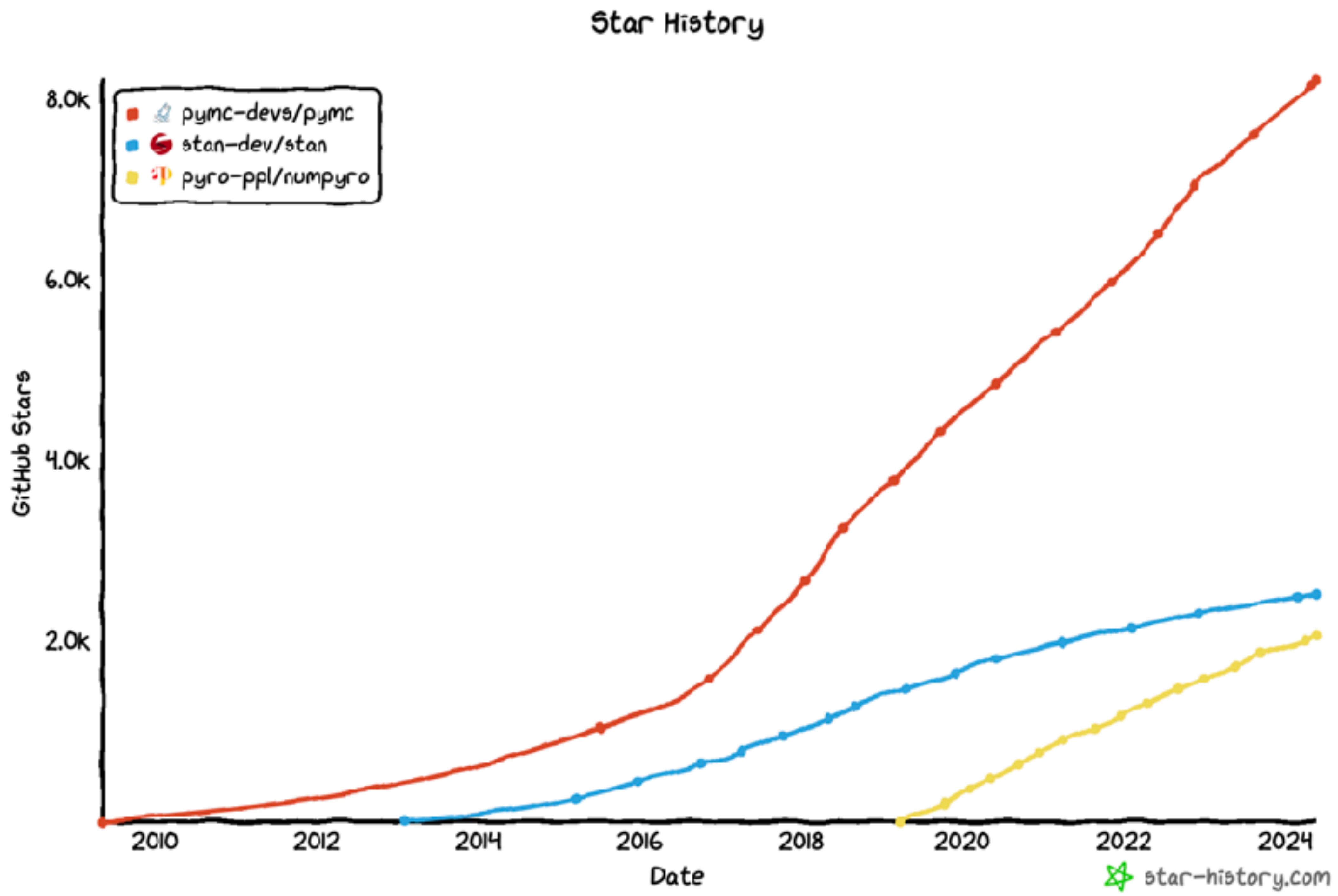
- PyMC is a **Python library** for probabilistic programming and Bayesian modeling

- Let's you specify **complex models** models **in simple syntax**

- The Inference step (aka **model fitting**) for Bayesian models used to be hard. PyMC makes that **very easy**.

- Integrates well with `numpy`, `pandas`,

- **Active community**. +400 contributors and used by +3.3k projects.



Bayesian Modeling

... you walk past an Ethiopian restaurant

🤩 You love Ethiopian food

😒 But place looks empty...

Bayesian Modeling

… you walk past an Ethiopian restaurant

Prior beliefs/preferences/knowledge

🤩 You love Ethiopian food

Likelihood given data/new information

😒 But place looks empty…

Bayesian Modeling

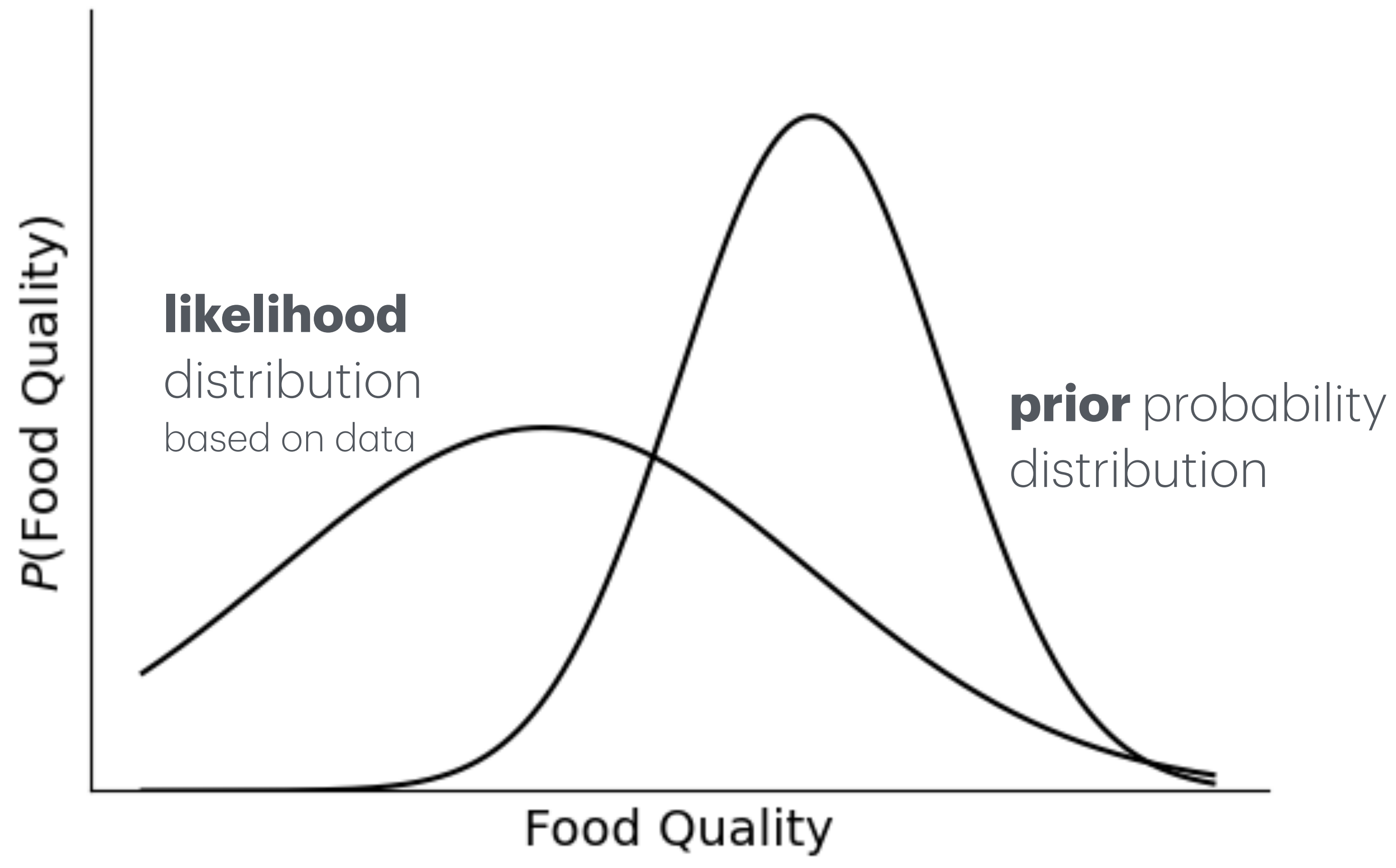
… you walk past an Ethiopian restaurant

Prior beliefs/preferences/knowledge

🤩 You love Ethiopian food

Likelihood given data/new information

😒 But place looks empty…



Bayesian Modeling

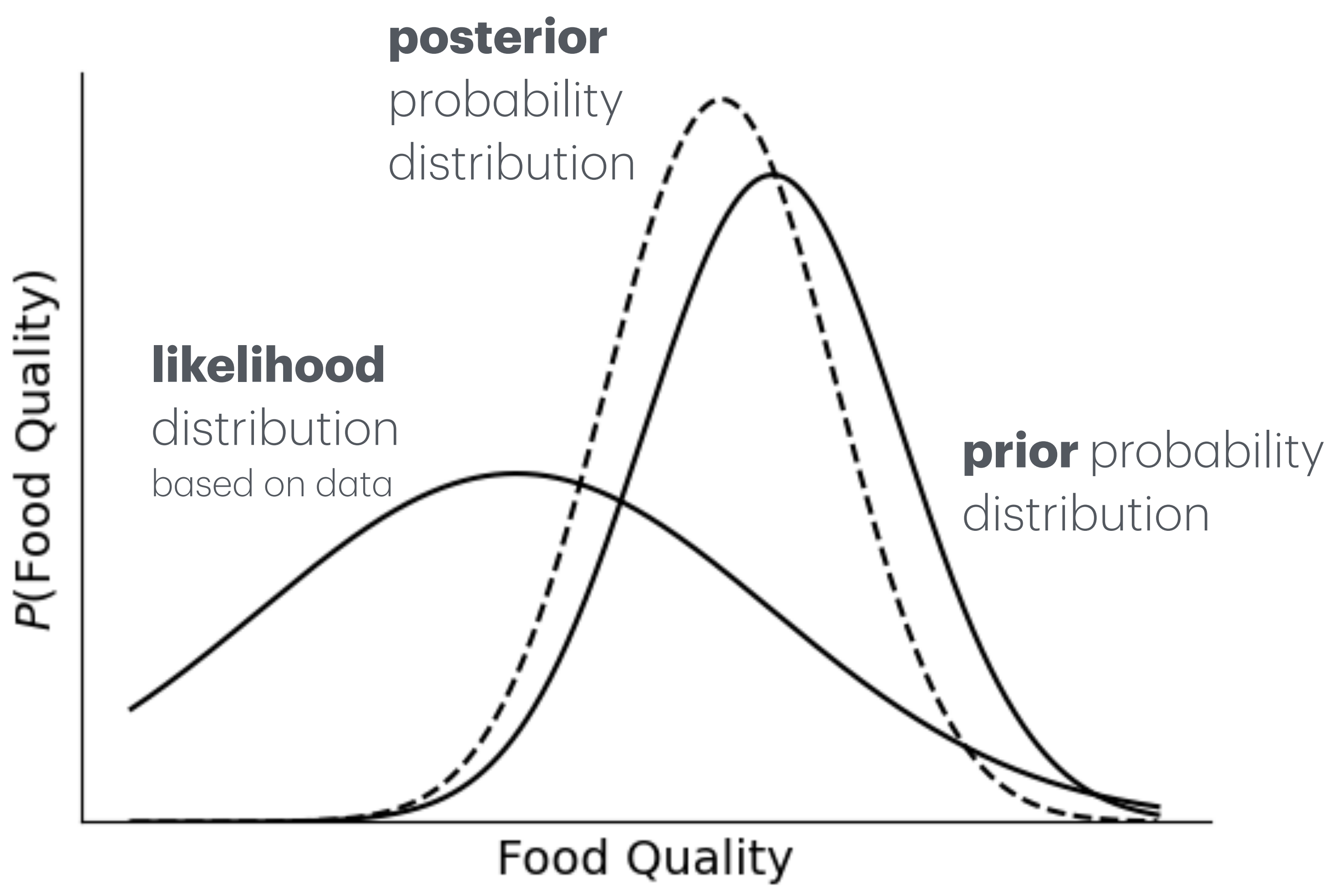
… you walk past an Ethiopian restaurant

Prior beliefs/preferences/knowledge

🤩 You love Ethiopian food

Likelihood given data/new information

😒 But place looks empty…



Bayesian Modeling

Bayes theorem computes the **Conditional Probability** *of A given B*

$$P\left(A \mid B\right) = \frac{P(A)\,P\left(B \mid A\right)}{P(B)}$$

Bayesian Modeling

... or "how likely are the model parameters are given the data"

$$P\left(\mathrm{M}|\mathrm{D}\right) = \frac{P\left(\mathrm{M}\right) P\left(\mathrm{D}|\mathrm{M}\right)}{P\left(\mathrm{D}\right)}$$

... M is just some model coefficients

$$f(\mathbf{x}) = \begin{bmatrix} M_0 \\ M_1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ x_1 \end{bmatrix}$$

$$= M_0 + M_1 x_1$$

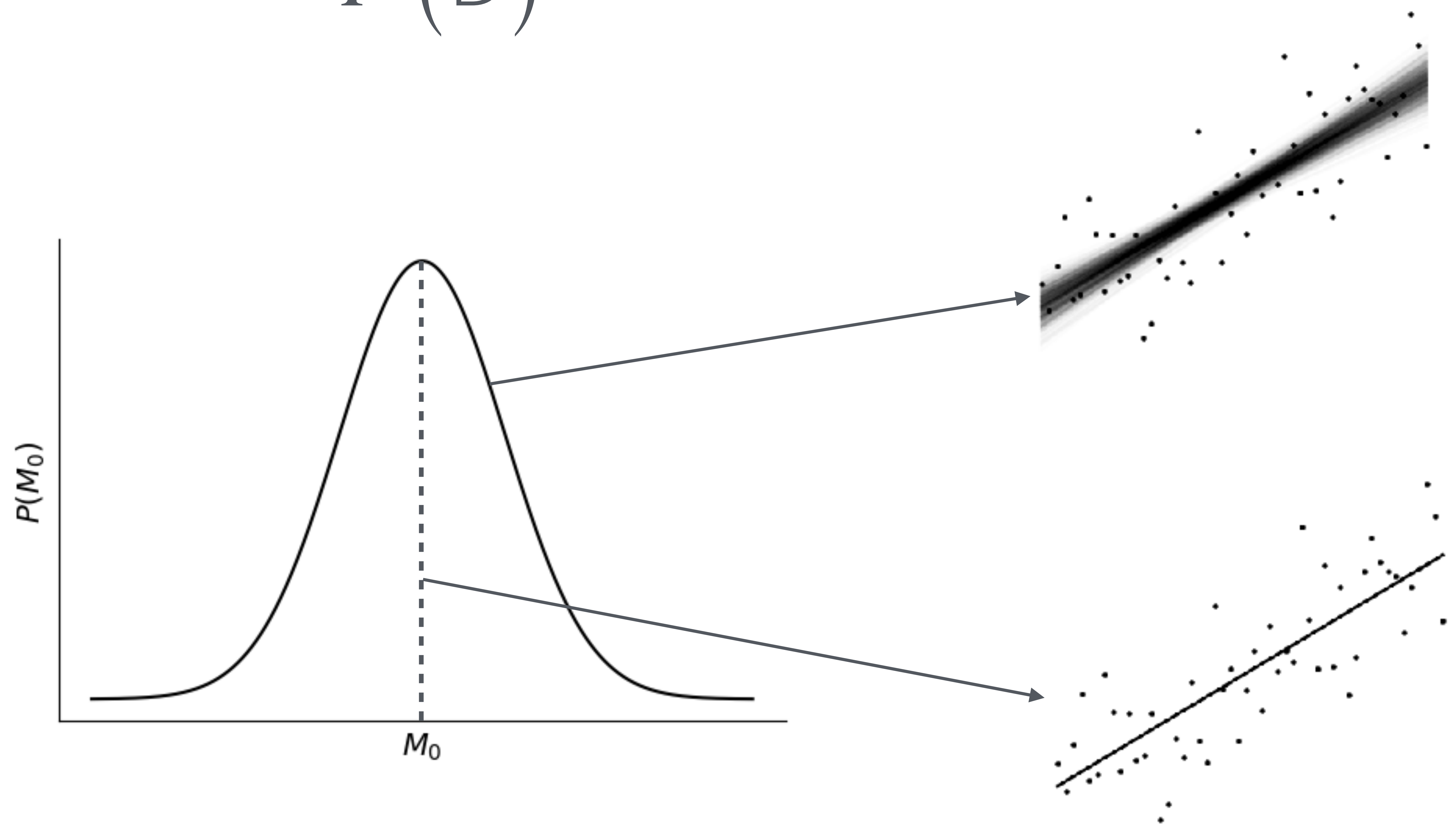
Bayesian Modeling

Each term is a distribution

$$\boxed{P(\mathrm{M}|\mathrm{D})} = \frac{P(\mathrm{M})\,P(\mathrm{D}|\mathrm{M})}{P(\mathrm{D})}$$

Posterior probability

The probability *distribution* of model **parameters** *M* given **data** *D*



Bayesian Modeling

Each term is a distribution

$$\boxed{P(M|D)} = \frac{\boxed{P(M)}\,P(D|M)}{P(D)}$$

Posterior probability

The probability *distribution* of model **parameters** *M* given **data** *D*

Prior probability

The probability *distribution* of model **parameters** *M*

(... our knowledge of what the model parameters might likely be)

Bayesian Modeling

Each term is a distribution

$$\boxed{P(\mathrm{M}\,|\,\mathrm{D})} = \frac{\boxed{P(\mathrm{M})}\,\boxed{P(\mathrm{D}\,|\,\mathrm{M})}}{P(\mathrm{D})}$$

Posterior probability

The probability *distribution* of model **parameters** *M* given **data** *D*

Prior probability

The probability *distribution* of model **parameters** *M*

(... our knowledge of what the model parameters might likely be)

Likelihood

The likelihood of the **data** *D* given **parameter** *M*

(... "If the model parameter is M, how likely is it we observe this data?")

Bayesian Modeling

In practice, no need to worry about normalisation 🤷‍♂️

$$P\left(\mathrm{M} \mid \mathrm{D}\right) \approx P\left(\mathrm{M}\right) P\left(\mathrm{D} \mid \mathrm{M}\right)$$

Bayesian Modeling

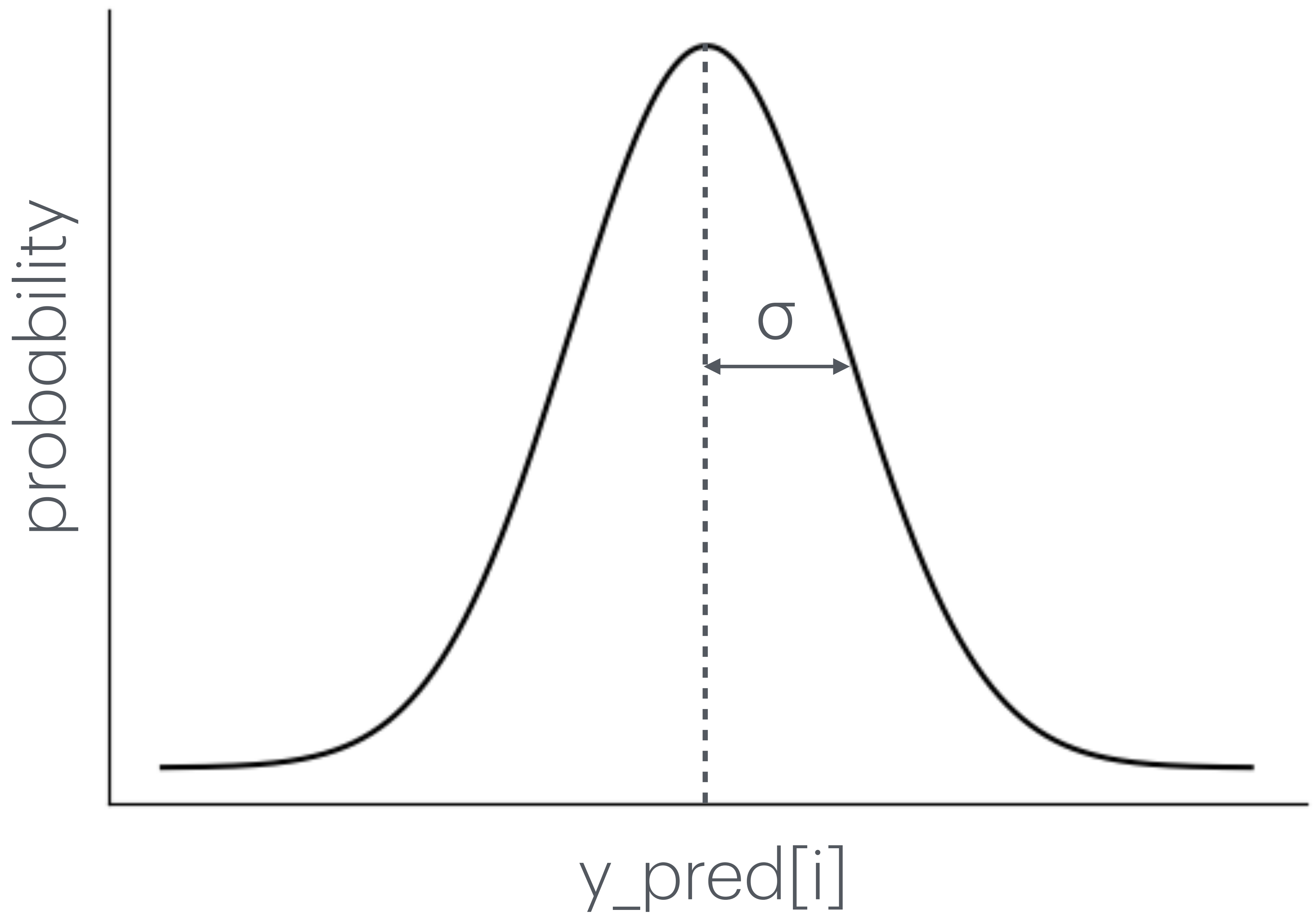
In PyMC: Specifying a Bayesian model

```python
import pymc as pm

with pm.Model():
    data = pm.Data("data", X, dims=("N", "M"))
    target = pm.Data("y", y, dims="N")

    m = pm.Normal('m', mu=1, sigma=1, dims="M")
    b = pm.Normal('b', mu=1, sigma=1)
    σ = pm.HalfNormal('σ', sigma=0.5)

    y_pred = pm.math.dot(data, m) + b

    pm.Normal('y_pred', mu=y_pred, sigma=σ, observed=y)

    idata = pm.sample(draws=4000, tune=1000, chains=4)
```

Bayesian Modeling

In PyMC: Specifying a Bayesian model

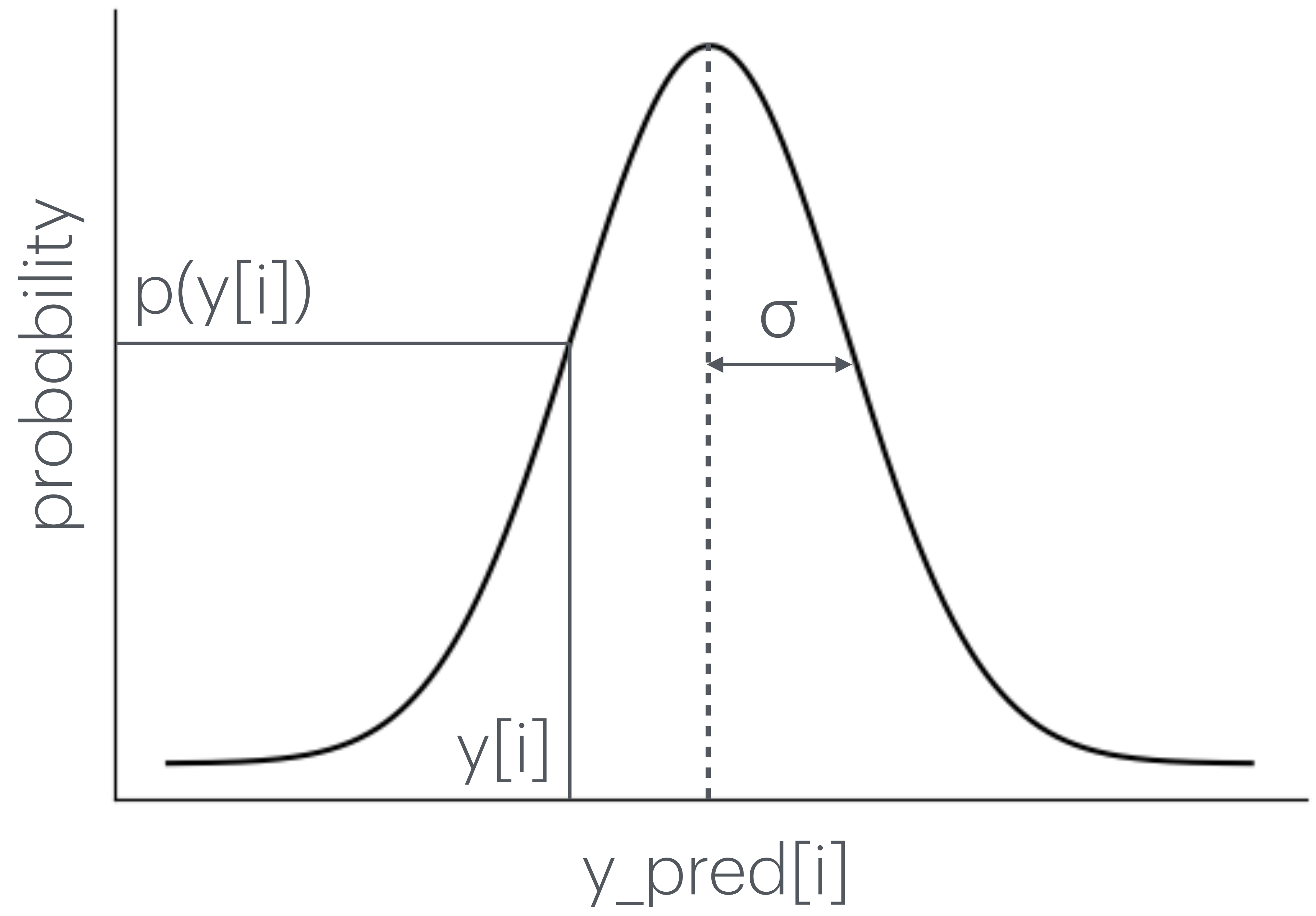
```python
import pymc as pm

with pm.Model():
    data = pm.Data("data", X, dims=("N", "M"))
    target = pm.Data("y", y, dims="N")

    m = pm.Normal('m', mu=1, sigma=1, dims="M")
    b = pm.Normal('b', mu=1, sigma=1)
    σ = pm.HalfNormal('σ', sigma=0.5)

    y_pred = pm.math.dot(data, m) + b

    pm.Normal('y_pred', mu=y_pred, sigma=σ, observed=y)

    idata = pm.sample(draws=4000, tune=1000, chains=4)
```



Bayesian Modeling

In PyMC: Specifying a Bayesian model

```
import pymc as pm

with pm.Model():
    data = pm.Data("data", X, dims=("N", "M"))
    target = pm.Data("y", y, dims="N")

    m = pm.Normal('m', mu=1, sigma=1, dims="M")
    b = pm.Normal('b', mu=1, sigma=1)
    σ = pm.HalfNormal('σ', sigma=0.5)

    y_pred = pm.math.dot(data, m) + b

    pm.Normal('y_pred', mu=y_pred, sigma=σ, observed=y)

    idata = pm.sample(draws=4000, tune=1000, chains=4)
```



A modeling example

Role modeling

Data:

- Mobile phone text messages

Key question:

- Which behaviours signal that a mobile user seeks out illegal substance?

https://thehackernews.com/2017/03/decrypt-pgp-encryption.html

A modeling example

Role modeling

Data:

- Mobile phone text messages

Key question:

- Which behaviours signal that a mobile user seeks out illegal substance?

Features (extracted):

- Network metrics (centrality, clustering, reciprocity…)
- User behaviour features (response time, bustiness…)
- Others (geo, device…)

Target:

- %–of–messages–sent *seeking to buy illegal substances*

A modeling example

… target variable has uncertainty

Target:

- %-of-messages-sent *seeking to buy illegal substances*

Has quantifiable uncertainty!

1 / 10 == 10 / 100

but which is more uncertain?

A modeling example

… target variable has uncertainty

Target:

- %–of–messages–sent *seeking to buy illegal substances*

$$\sqrt{p(1-p)/N}$$

	%-messages-on-topic	Num. messages sent
0	0.106648	122.0
1	0.006137	44.0
2	0.097083	102.0
3	0.023531	134.0
4	0.123395	169.0
...	...	...

Expected error
0.007483
0.012684
0.014947
0.018720
0.003491
...

A modeling example

... target variable has uncertainty

data_features

Degree Centrality	Betweenness Centrality	...	Message Burstiness	Reciprocity
1.322104	1.007634	...	1.051479	0.102654
1.637522	0.259430	...	0.966953	0.549651
0.489639	1.254167	...	0.328711	0.041125
1.016696	0.022683	...	0.461559	0.906399
1.287290	0.448680	...	1.271071	0.022042
0.189915	1.904019	...	0.990928	0.465488

... each row is a phone

predict →

target

%-messages-on-topic
0.106648
0.006137
0.097083
0.023531
0.123395
...

target_err

Expected error
0.007483
0.012684
0.014947
0.018720
0.003491
...

A modeling example

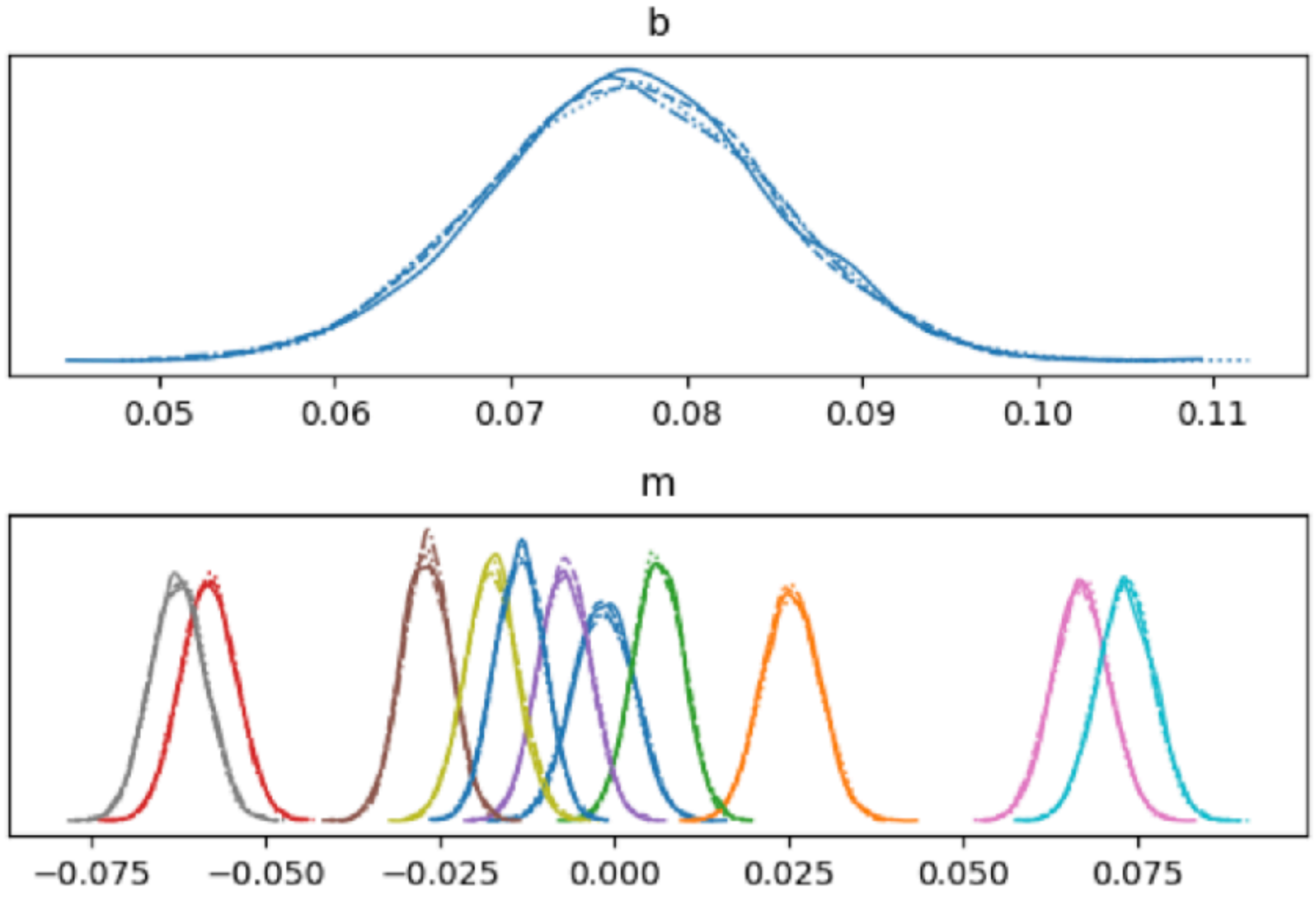
… target variable has uncertainty

```python
import pymc as pm

def standard_error(p, N):
    return np.sqrt(p * (1 - p) / N) + 1e-1

target_err = standard_error(target, num_messages_sent)

with pm.Model() as model:
    X = pm.Data("X", data_features, dims=("N", "M"))
    y = pm.Data("y", target, dims="N")
    y_err = pm.Data("y_err", target_err, dims="N")

    m = pm.Normal("m", mu=0, sigma=1, dims="M")
    b = pm.Normal("b", mu=0, sigma=1)
    σ = pm.HalfNormal("σ", sigma=1) * y_err

    y_pred = pm.math.dot(X, m) + b

    pm.Normal("y_pred", mu=y_pred, sigma=σ, observed=y)

    idata = pm.sample(draws=4000, tune=1000, chains=4)
```

A modeling example

… target variable has uncertainty

```python
import pymc as pm

def standard_error(p, N):
    return np.sqrt(p * (1 - p) / N) + 1e-1

target_err = standard_error(target, num_messages_sent)

with pm.Model() as model:
    X = pm.Data("X", data_features, dims=("N", "M"))
    y = pm.Data("y", target, dims="N")
    y_err = pm.Data("y_err", target_err, dims="N")

    m = pm.Normal("m", mu=0, sigma=1, dims="M")
    b = pm.Normal("b", mu=0, sigma=1)
    σ = pm.HalfNormal("σ", sigma=1) * y_err

    y_pred = pm.math.dot(X, m) + b

    pm.Normal("y_pred", mu=y_pred, sigma=σ, observed=y)

    idata = pm.sample(draws=4000, tune=1000, chains=4)
```



A modeling example

*Now imagine you had **new encrypted messages** with known source/target*

Extract features:

- Network metrics (centrality, clustering, reciprocity...)
- User behaviour features (response time, bustiness...)
- Others (geo, device...)

Degree Centrality	Betweenness Centrality	...	Message Burstiness	Reciprocity
1.322104	1.007634	...	1.051479	0.102654
1.637522	0.259430	...	0.966953	0.549651
0.489639	1.254167	...	0.328711	0.041125
1.016696	0.022683	...	0.461559	0.906399
1.287290	0.448680	...	1.271071	0.022042
0.189915	1.904019	...	0.990928	0.465488
2.045263	1.312755	...	0.817728	0.312907
0.305075	0.594418	...	0.168234	1.869163
0.512102	0.633174	...	1.351275	0.624492
0.570922	0.123635	...	1.034214	0.967696

A modeling example

*Now imagine you had **new encrypted messages** with known source/target*

Sample the posterior predictive given new data

```
with model:
    # Update model data
    pm.set_data(
        {
            "X": data_features_new
        }
    )

    # Sample posterior predictive for new data
    posterior_predictive = pm.sample_posterior_predictive(
        trace=idata,
        var_names=["y_pred"],
    )
```

A modeling example

Now imagine you had ***new encrypted messages*** *with known source/target*

Extract predictions and credible intervals

```
pd.DataFrame(
    np.hstack(
        [
            # Posterior predictive mean
            posterior_predictive.posterior_predictive.y_pred.mean(
                dim=["chain", "draw"]
            ).values.reshape(-1, 1),

            # Posterior predictive 95% HDI
            az.hdi(
                posterior_predictive.posterior_predictive.y_pred,
                hdi_prob=0.95
            ).y_pred.values*0.1,
        ]
    ),
    columns=["y_pred_new", "lower_95% CI", "upper_95% CI"],
)
```

outputs

y_pred_new	lower_95% CI	upper_95% CI
-0.002815	-0.115217	0.123960
-0.104019	-0.130470	0.107570
0.080498	-0.111952	0.126296
0.070727	-0.110344	0.129070
0.023705	-0.119801	0.118341
0.115565	-0.110362	0.128785
0.131232	-0.103064	0.136775
0.028444	-0.116820	0.123254
0.100259	-0.109692	0.128704
-0.057094	-0.126024	0.113270

Workshop teaser

Q&A

... and let's connect

- github.com/pymc-labs
- twitter.com/pymc_labs
- linkedin.com/company/pymc-labs/
- pymc-labs.com/
- info@pymc-labs.com

- github.com/ulfaslak
- twitter.com/ulfaslak
- linkedin.com/in/ulfaslak/
- ulfaslak.dk/
- ulf.aslak@pymc-labs.com
- ulfaslak@gmail.com