Naïve Bayesians

Hierarchical Modelling | Intro

Goal

Developing the Bayesian muscle to solve a wide range of problems

Naïve Bayesian Philosophy

Intuitive (Visual) Understanding of the Bayesian Reasoning

Ability to model real world problems in a Bayesian Setting

Starting from Simple Probabilistic modelling

Adapting it in a a Bayesian setting
And moving towards ML models

PyMC3

Fluency in the Calculus of Bayesian Stats & ML model

Bayes Rule

Posterior Likelihood Prior

$$P(\theta \mid D) = \frac{P(D \mid \theta)\, P(\theta)}{P(D)}$$

Normalising Constant

Bayes Rule

Posterior Likelihood Prior

$$P(\theta_i \mid D) = \frac{P(D \mid \theta_i)\, P(\theta_i)}{\sum_{all\ j} P(D \mid \theta_j)\, P(\theta_j)}$$

Normalising Constant

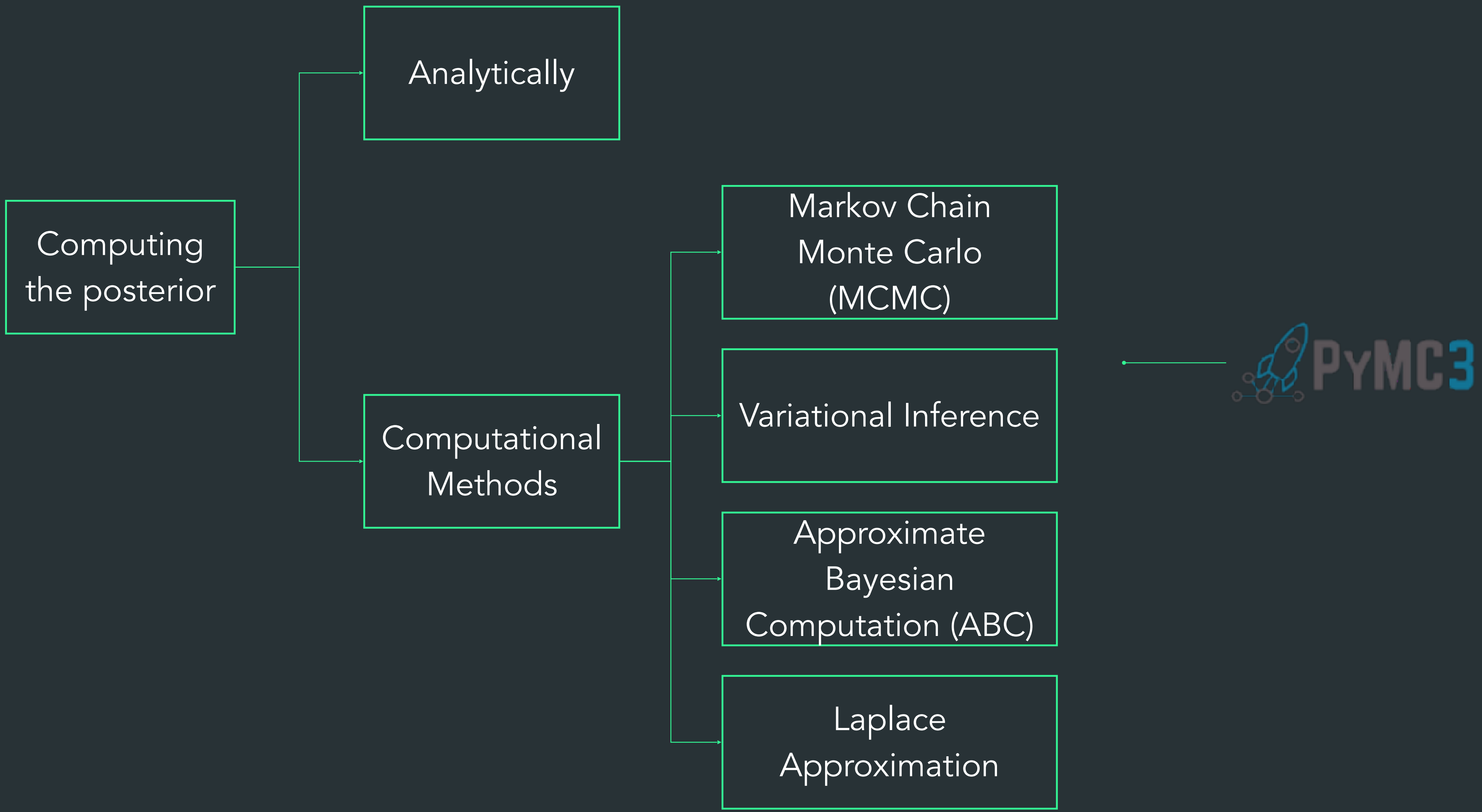
Bayes Rule

Posterior Likelihood Prior

$$P(\theta \mid D) = \frac{P(D \mid \theta)\, P(\theta)}{\int P(D \mid \theta')\, P(\theta')\, d\theta'}$$

Normalising Constant

Ways to get to the Posterior



Generating samples from a random variable

$$x_k \sim \mathcal{N}\left(\mu = 0, \sigma = 1\right)$$

Generating samples from a random variable

$$x_k \sim \mathcal{N}\left(\mu = 0, \sigma = 1\right)$$

```
from scipy import stats

# Sample the distribution
x = stats.norm.rvs(
    size=1000,
    loc=0,
    scale=1,
)
```

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_K \end{bmatrix}$$

Generating samples from a random variable

$$x_k \sim \mathcal{N}(\mu = 0, \sigma = 1)$$

```python
from scipy import stats

# Sample the distribution
x = stats.norm.rvs(
    size=1000,
    loc=0,
    scale=1,
)
```

```python
import pymc3 as pm

with pm.Model() as model:
    # Specify the distribution
    x_n = pm.Normal(name="x", mu=0, sigma=1)
    # Sample
    trace = pm.sample(draws=1000)

x_pm = trace.get_values(varname="x")
```

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_K \end{bmatrix}$$

Fitting a probability distribution

$$x_k \sim \mathcal{N}(\mu = 0, \sigma = 1)$$

```
from scipy import stats

# Fitting the distribution
mu_fit, sigma_fit = stats.norm.fit(data=x)
```

```
import pymc3 as pm

with pm.Model() as mod:
    # Prior
    mu = pm.Normal(
        name="mu",
        mu=0,
        sigma=10,
    )
    # Likelihood
    obs = pm.Normal(
        name="x",
        observed=x,
        mu=mu,
        sigma=1,
    )
    # Posterior
    trace = pm.sample(draws=1000)

mu_pm = trace.get_values(varname="mu")
```

Fitting a probability distribution

$$x_k \sim \mathcal{N}\left(\mu = 0, \sigma = 1\right)$$

```
from scipy import stats

# Fitting the distribution
mu_fit, sigma_fit = stats.norm.fit(data=x)
```

```
import pymc3 as pm

with pm.Model() as mod:
    # Prior
    mu = pm.Normal(
        name="mu",
        mu=0,
        sigma=10,
    )
    # Likelihood
    obs = pm.Normal(
        name="x",
        observed=x,
        mu=mu,
        sigma=1,
    )
    # Posterior
    trace = pm.sample(draws=1000)

mu_pm = trace.get_values(varname="mu")
```

Fitting a probability distribution

```python
with pm.Model() as mod:
    # Prior
    mu = pm.Normal(
        name="mu",
        mu=0,
        sigma=10,
    )
    # Likelihood
    obs = pm.Normal(
        name="x",
        observed=x,
        mu=mu,
        sigma=1,
    )
    # Posterior
    trace = pm.sample(draws=1000)

mu_pm = trace.get_values(varname="mu")
```

Initialise Model

Fitting a probability distribution

```python
with pm.Model() as mod:
    # Prior
    mu = pm.Normal(
        name="mu",
        mu=0,
        sigma=10,
    )
    # Likelihood
    obs = pm.Normal(
        name="x",
        observed=x,
        mu=mu,
        sigma=1,
    )
    # Posterior
    trace = pm.sample(draws=1000)

mu_pm = trace.get_values(varname="mu")
```

Define priors

$$\mu \sim \mathcal{N}\left(\mu = 0, \sigma = 10\right)$$

Fitting a probability distribution

```
with pm.Model() as mod:
    # Prior
    mu = pm.Normal(
        name="mu",
        mu=0,
        sigma=10,
    )
    # Likelihood
    obs = pm.Normal(
        name="x",
        observed=x,
        mu=mu,
        sigma=1,
    )
    # Posterior
    trace = pm.sample(draws=1000)

mu_pm = trace.get_values(varname="mu")
```

Define likelihood

$$x_k \sim \mathcal{N}\left(\mu = \mu, \sigma = 1\right)$$

Fitting a probability distribution

```python
with pm.Model() as mod:
    # Prior
    mu = pm.Normal(
        name="mu",
        mu=0,
        sigma=10,
    )
    # Likelihood
    obs = pm.Normal(
        name="x",
        observed=x,
        mu=mu,
        sigma=1,
    )
    # Posterior
    trace = pm.sample(draws=1000)

mu_pm = trace.get_values(varname="mu")
```

$$\mu \sim \mathcal{N}\left(\mu = 0, \sigma = 10\right)$$

$$x_k \sim \mathcal{N}\left(\mu = \mu, \sigma = 1\right)$$

Fitting a probability distribution

```
with pm.Model() as mod:
    # Prior
    mu = pm.Normal(
        name="mu",
        mu=0,
        sigma=10,
   )
    # Likelihood
    obs = pm.Normal(
        name="x",
        observed=x,
        mu=mu,
        sigma=1,
    )
    # Posterior
    trace = pm.sample(draws=1000)

mu_pm = trace.get_values(varname="mu")
```

Pass in the data you have

Fitting a probability distribution

```
with pm.Model() as mod:
    # Prior
    mu = pm.Normal(
        name="mu",
        mu=0,
        sigma=10,
    )
    # Likelihood
    obs = pm.Normal(
        name="x",
        observed=x,
        mu=mu,
        sigma=1,
    )
    # Posterior
    trace = pm.sample(draws=1000)

mu_pm = trace.get_values(varname="mu")
```

Compute the posterior

Fitting a probability distribution

```
with pm.Model() as mod:
    # Prior
    mu = pm.Normal(
        name="mu",
        mu=0,
        sigma=10,
    )
    # Likelihood
    obs = pm.Normal(
        name="x",
        observed=x,
        mu=mu,
        sigma=1,
    )
    # Posterior
    trace = pm.sample(draws=1000)

mu_pm = trace.get_values(varname="mu")
```

Get samples from your posterior distribution

Fitting a probability distribution | Vectorising the parameters

```
with pm.Model() as mod:
    # Prior
    mu = pm.Normal(
        name="mu",
        mu=0,
        sigma=10,
    )
    # Likelihood
    obs = pm.Normal(
        name="x",
        observed=x,
        mu=mu,
        sigma=1,
    )
    # Posterior
    trace = pm.sample(draws=1000)

mu_pm = trace.get_values(varname="mu")
```

Fitting a probability distribution | Vectorising the parameters

```python
with pm.Model() as mod:
    # Prior
    mu = pm.Normal(
        name="mu",
        mu=0,
        sigma=10,
        shape=1,
    )
    # Likelihood
    obs = pm.Normal(
        name="x",
        observed=x,
        mu=mu,
        sigma=1,
    )
    # Posterior
    trace = pm.sample(draws=1000)

mu_pm = trace.get_values(varname="mu")
```

Vectorise the prior parameters

$$\boldsymbol{\mu} = \left[\mu \sim \mathcal{N}\left(\mu = 0, \sigma = 10\right)\right]$$

Fitting a probability distribution | Vectorising the parameters

```
with pm.Model() as mod:
    # Prior
    mu = pm.Normal(
        name="mu",
        mu=0,
        sigma=10,
        shape=1,
    )
    # Likelihood
    obs = pm.Normal(
        name="x",
        observed=x,
        mu=mu,
        sigma=1,
    )
    # Posterior
    trace = pm.sample(draws=1000)

mu_pm = trace.get_values(varname="mu")
```

Vectorise the prior parameters

$$\boldsymbol{\mu} = \left[\mu \sim \mathcal{N}\left(\mu = 0, \sigma = 10\right)\right]$$

Fitting a probability distribution | Vectorising the parameters

```
index = np.zeros(len(x), dtype=np.int8)

with pm.Model() as mod:
    # Prior
    mu = pm.Normal(
        name="mu",
        mu=0,
        sigma=10,
        shape=1,
    )
    # Likelihood
    obs = pm.Normal(
        name="x",
        observed=x,
        mu=mu,
        sigma=1,
    )
    # Posterior
    trace = pm.sample(draws=1000)

mu_pm = trace.get_values(varname="mu")
```

Fitting a probability distribution | Vectorising the parameters

```
index = np.zeros(len(x), dtype=np.int8)

with pm.Model() as mod:
    # Prior
    mu = pm.Normal(
        name="mu",
        mu=0,
        sigma=10,
        shape=1,
    )
    # Likelihood
    obs = pm.Normal(
        name="x",
        observed=x,
        mu=mu,
        sigma=1,
    )
    # Posterior
    trace = pm.sample(draws=1000)

mu_pm = trace.get_values(varname="mu")
```

$$x = \begin{bmatrix} \mathcal{N}\left(\mu = \mu, \sigma = 1\right) \\ \vdots \\ \mathcal{N}\left(\mu = \mu, \sigma = 1\right) \end{bmatrix}$$

Fitting a probability distribution | Vectorising the parameters

```
index = np.zeros(len(x), dtype=np.int8)

with pm.Model() as mod:
    # Prior
    mu = pm.Normal(
        name="mu",
        mu=0,
        sigma=10,
        shape=1,
    )
    # Likelihood
    obs = pm.Normal(
        name="x",
        observed=x,
        mu=mu,
        sigma=1,
    )
    # Posterior
    trace = pm.sample(draws=1000)

mu_pm = trace.get_values(varname="mu")
```

Fitting a probability distribution | Vectorising the parameters

```python
index = np.zeros(len(x), dtype=np.int8)

with pm.Model() as mod:
    # Prior
    mu = pm.Normal(
        name="mu",
        mu=0,
        sigma=10,
        shape=1,
    )
    # Likelihood
    obs = pm.Normal(
        name="x",
        observed=x,
        mu=mu[index],
        sigma=1,
    )
    # Posterior
    trace = pm.sample(draws=1000)

mu_pm = trace.get_values(varname="mu")
```

"Explode" the parameters

$$\begin{bmatrix} \mu \\ \vdots \\ \mu \end{bmatrix}$$

Length of the data

Fitting a probability distribution | Vectorising the parameters

```
index = np.zeros(len(x), dtype=np.int8)

with pm.Model() as mod:
    # Prior
    mu = pm.Normal(
        name="mu",
        mu=0,
        sigma=10,
        shape=1,
    )
    # Likelihood
    obs = pm.Normal(
        name="x",
        observed=x,
        mu=mu[index],
        sigma=1,
    )
    # Posterior
    trace = pm.sample(draws=1000)

mu_pm = trace.get_values(varname="mu")
```

Moving to hierarchical models

```python
# Generate synthetic hierarchical data
x_a = stats.norm.rvs(
    size=1000,
    loc=0,
    scale=1,
)
x_b = stats.norm.rvs(
    size=100,
    loc=2,
    scale=1,
)
x_hierarchy = np.append(x_a, x_b)
```

$$\boldsymbol{x} = \begin{bmatrix} x_1^a \\ \vdots \\ x_K^a \\ x_1^b \\ \vdots \\ x_{K'}^b \end{bmatrix}$$

Moving to hierarchical models | Pooled Model

```
# Generate synthetic hierarchical data
x_a = stats.norm.rvs(
    size=1000,
    loc=0,
    scale=1,
)
x_b = stats.norm.rvs(
    size=200,
    loc=2,
    scale=1,
)
x_hierarchy = np.append(x_a, x_b)

# Fitting the distribution
mu_fit, sigma_fit = stats.norm.fit(data=x_hierarchy)
```

$$\boldsymbol{x} = \begin{bmatrix} x_1^a \\ \vdots \\ x_K^a \\ x_1^b \\ \vdots \\ x_{K'}^b \end{bmatrix}$$

Moving to hierarchical models | Unpooled Model

```
index_a = np.zeros(len(x_a), dtype=np.int8)
index_b = np.ones(len(x_b), dtype=np.int8)
index = np.append(index_a, index_b)

with pm.Model() as mod:
    # Prior
    mu = pm.Normal(
        name="mu",
        mu=0,
        sigma=10,
        shape=2,
    )
    # Likelihood
    obs = pm.Normal(
        name="x",
        observed=x_hierarchy,
        mu=mu[index],
        sigma=1,
    )
    # Posterior
    trace = pm.sample(draws=1000)

mu_pm = trace.get_values(varname="mu")
```

$$\boldsymbol{\mu} = \begin{bmatrix} \mu_a \sim \mathcal{N}\left(\mu = 0, \sigma = 1\right) \\ \mu_b \sim \mathcal{N}\left(\mu = 0, \sigma = 1\right) \end{bmatrix}$$

Moving to hierarchical models | Unpooled Model

```python
index_a = np.zeros(len(x_a), dtype=np.int8)
index_b = np.ones(len(x_b), dtype=np.int8)
index = np.append(index_a, index_b)

with pm.Model() as mod:
    # Prior
    mu = pm.Normal(
        name="mu",
        mu=0,
        sigma=10,
        shape=2,
    )
    # Likelihood
    obs = pm.Normal(
        name="x",
        observed=x_hierarchy,
        mu=mu[index],
        sigma=1,
    )
    # Posterior
    trace = pm.sample(draws=1000)

mu_pm = trace.get_values(varname="mu")
```

$$i = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

Moving to hierarchical models | Unpooled Model

```
index_a = np.zeros(len(x_a), dtype=np.int8)
index_b = np.ones(len(x_b), dtype=np.int8)
index = np.append(index_a, index_b)

with pm.Model() as mod:
    # Prior
    mu = pm.Normal(
        name="mu",
        mu=0,
        sigma=10,
        shape=2,
    )
    # Likelihood
    obs = pm.Normal(
        name="x",
        observed=x_hierarchy,
        mu=mu[index],
        sigma=1,
    )
    # Posterior
    trace = pm.sample(draws=1000)

mu_pm = trace.get_values(varname="mu")
```

$$\boldsymbol{x} = \begin{bmatrix} x_1^a \\ \vdots \\ x_K^a \\ x_1^b \\ \vdots \\ x_{K'}^b \end{bmatrix}$$

Moving to hierarchical models | Unpooled Model

```
index_a = np.zeros(len(x_a), dtype=np.int8)
index_b = np.ones(len(x_b), dtype=np.int8)
index = np.append(index_a, index_b)

with pm.Model() as mod:
    # Prior
    mu = pm.Normal(
        name="mu",
        mu=0,
        sigma=10,
        shape=2,
    )
    # Likelihood
    obs = pm.Normal(
        name="x",
        observed=x_hierarchy,
        mu=mu[index],
        sigma=1,
    )
    # Posterior
    trace = pm.sample(draws=1000)

mu_pm = trace.get_values(varname="mu")
```

$$\begin{bmatrix} \mu_a \\ \vdots \\ \mu_a \\ \mu_b \\ \vdots \\ \mu_b \end{bmatrix}$$

Moving to hierarchical models | Unpooled Model

```python
with pm.Model() as hierarchical_mod:



    # Prior
    mu = pm.Normal(
        name="mu",
        mu=0,
        sigma=0.1,
        shape=2,
    )
    # Likelihood
    obs = pm.Normal(
        name="x",
        observed=x_hierarchy,
        mu=mu[index],
        sigma=1,
    )
    # Posterior
    trace = pm.sample(draws=1000)
```

$$\boldsymbol{\mu} = \begin{bmatrix} \mu_a \sim \mathcal{N}\left(\mu = 0, \sigma = 1\right) \\ \mu_b \sim \mathcal{N}\left(\mu = 0, \sigma = 1\right) \end{bmatrix}$$

Moving to hierarchical models

```python
with pm.Model() as hierarchical_mod:
    # Set a global prior for the parameters
    mu_glob = pm.Normal(
        name="mu_glob",
        mu=1,
        sigma=0.1,
    )

    # Prior
    mu = pm.Normal(
        name="mu",
        mu=mu_glob,
        sigma=0.1,
        shape=2,
    )
    # Likelihood
    obs = pm.Normal(
        name="x",
        observed=x_hierarchy,
        mu=mu[index],
        sigma=1,
    )
    # Posterior
    trace = pm.sample(draws=1000)
```

$$\mu_{\text{glob}} \sim \mathcal{N}\left(\mu = 1, \sigma = 0.1\right)$$

Moving to hierarchical models

```python
with pm.Model() as hierarchical_mod:
    # Set a global prior for the parameters
    mu_glob = pm.Normal(
        name="mu_glob",
        mu=1,
        sigma=0.1,
    )

    # Prior
    mu = pm.Normal(
        name="mu",
        mu=mu_glob,
        sigma=0.1,
        shape=2,
    )
    # Likelihood
    obs = pm.Normal(
        name="x",
        observed=x_hierarchy,
        mu=mu[index],
        sigma=1,
    )
    # Posterior
    trace = pm.sample(draws=1000)
```

$$\mu_{\text{glob}} \sim \mathcal{N}\left(\mu = 1, \sigma = 0.1\right)$$

$$\boldsymbol{\mu} = \begin{bmatrix} \mu_a \sim \mathcal{N}\left(\mu = \mu_{\text{glob}}, \sigma = 0.1\right) \\ \mu_b \sim \mathcal{N}\left(\mu = \mu_{\text{glob}}, \sigma = 0.1\right) \end{bmatrix}$$

Moving to hierarchical models