

Learn Bayes book club

Session 1

2026-10-14

Intro and Menti

Different schools, same goal

All statistical frameworks use data y to learn about parameters θ :

1. Observe y
2. Estimate θ

All statistical frameworks use data y to learn about parameters θ :

1. Observe y
2. Estimate θ

Frequentist statistics treats this as an **optimisation problem**.

Bayesian statistics uses **probability calculus**.

All statistical frameworks use data y to learn about parameters θ :

1. Observe y
2. Estimate θ

Frequentist statistics treats this as an **optimisation problem**.

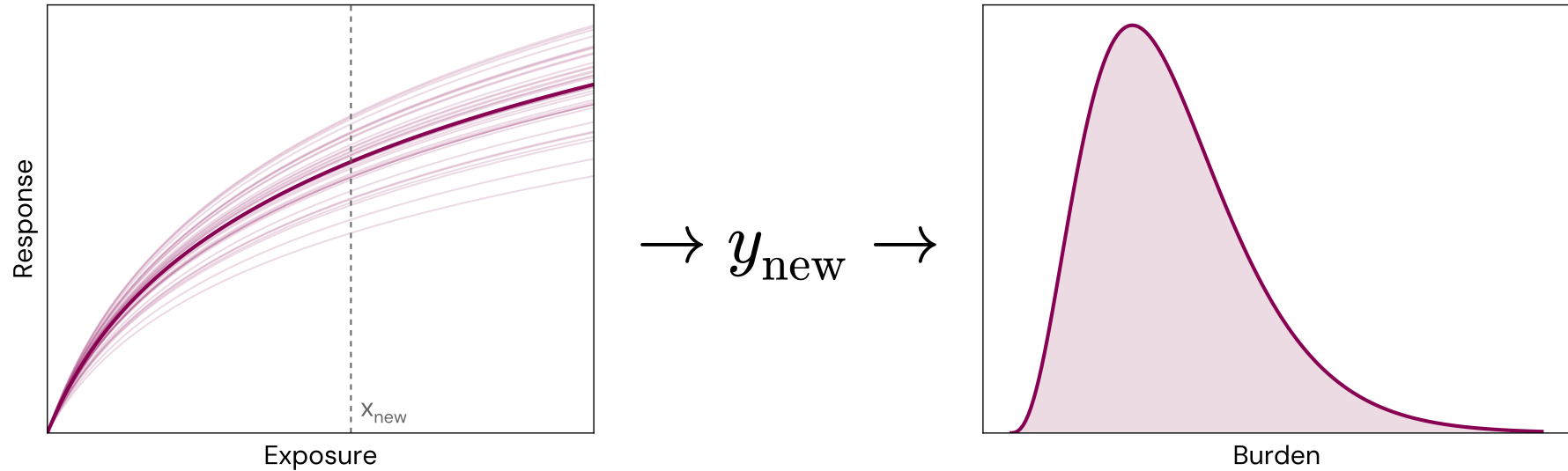
Bayesian statistics uses **probability calculus**.

Bayesian models are generative; going from θ to y is intuitive, too:

1. Assume θ
2. Draw y

Why Bayes

Uncertainty of parameters is captured in probability distributions.



Prior distributions can stabilise models that would otherwise break.

Prior distributions can stabilise models that would otherwise break.

```
1 library(lme4)
2
3 # We know `dose` effects vary by clinic
4 # but some clinics have very little data
5 fit <- lmer(
6   response ~ dose + (dose | clinic),
7   data = patients
8 )
9 #> boundary (singular) fit: see help('isSingular')
10
```



Sometimes our data alone can't identify the model we want to fit.

Prior distributions can stabilise models that would otherwise break.

```
1 library(lme4)
2
3 # We know `dose` effects vary by clinic
4 # but some clinics have very little data
5 fit <- lmer(
6   response ~ dose + (dose | clinic),
7   data = patients
8 )
9 #> boundary (singular) fit: see help('isSingular')
10
```



Sometimes our data alone can't identify the model we want to fit.

But priors are often hotly debated! What are your thoughts?

Bring your own model

You're allowed to write custom models.¹

The book encourages this because it's educational.

I also found it liberating and fun!

¹ (but don't have to thanks to packages like brms)

How Bayes

$$p(\theta \mid \text{data}) \propto p(\text{data} \mid \theta) \times p(\theta)$$

posterior $\propto$ likelihood $\times$ prior

Probabilistic programming languages allow you to “spell out” the math:

$$y_i \sim \text{Normal}(\mu_i, \sigma)$$

$$\mu_i = \alpha + \beta x_i$$

$$\alpha \sim \text{Normal}(0, 10)$$

$$\beta \sim \text{Normal}(0, 10)$$

$$\sigma \sim \text{Exponential}(1)$$

```
1 library(rethinking)
2
3 linear_regression <- ulam(
4   alist(
5     y ~ dnorm(mu, sigma),
6     mu <- alpha + beta * x,
7     alpha ~ dnorm(0, 10),
8     beta ~ dnorm(0, 10),
9     sigma ~ dexp(1)
10  ),
11  data = list(x = x, y = y)
12 )
13
```



What Bayes

Appendix

```

1 data {
2   int<lower=0> N;
3   vector[N] x, y;
4 }
5 parameters {
6   real alpha;
7   real beta;
8   real<lower=0> sigma;
9 }
10 model {
11   alpha ~ normal(0, 10);
12   beta ~ normal(0, 10);
13   sigma ~ exponential(1);
14   y ~ normal(alpha + beta * x, sigma);
15 }
16

```



```
1 import pymc as pm
2
3 with pm.Model() as linear_regression:
4     alpha = pm.Normal("alpha", mu=0, sigma=10)
5     beta = pm.Normal("beta", mu=0, sigma=10)
6     sigma = pm.Exponential("sigma", lam=1)
7     pm.Normal("y", mu=alpha + beta * x, sigma=sigma, observed=y)
8
```



```
1 using Turing
2
3 @model function linear_regression(x, y)
4      $\alpha$  ~ Normal(0, 10)
5      $\beta$  ~ Normal(0, 10)
6      $\sigma$  ~ Exponential(1)
7     for i in eachindex(x, y)
8         y[i] ~ Normal( $\alpha$  +  $\beta$  * x[i],  $\sigma$ )
9     end
10 end
11
```

