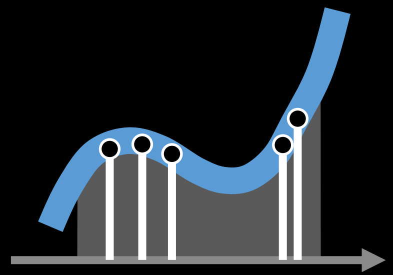
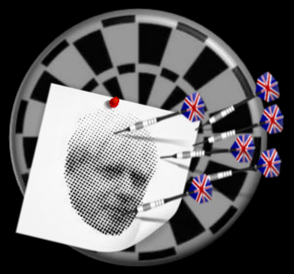
COMPUTATIONAL METHODS

- If we want $\mathbb{E}h(X)$ but the maths is too complicated, we can approximate
$$\mathbb{E}h(x) \approx n^{-1} \sum_{i=1}^{n} h(x_i)$$
where $x_1, \ldots, x_n$ are sampled from X

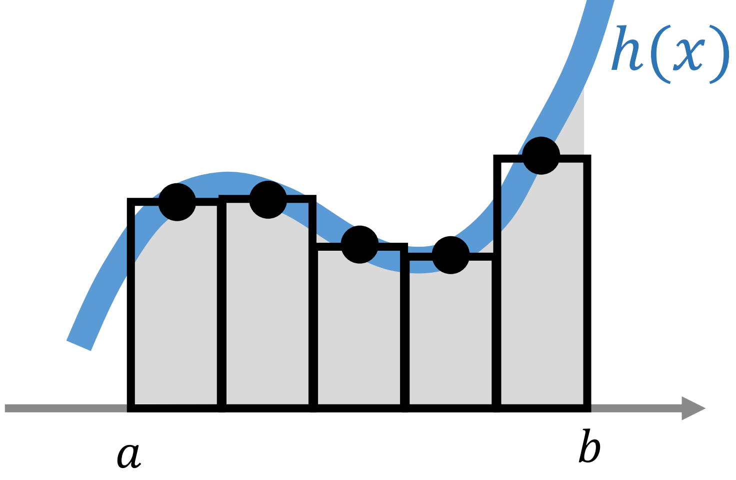
- This approximation also tells us how to estimate probabilities, since
$$\mathbb{P}(X \in A) = \mathbb{E}1_{X \in A}$$

- For computational Bayes, we need something a bit fancier: *weighted samples*





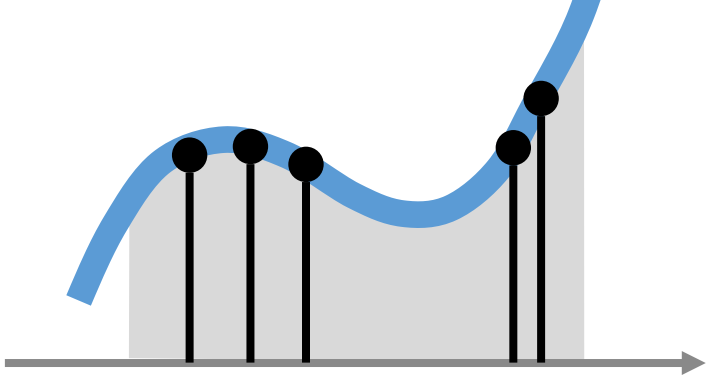
Trinity College integration



$$\int_{x=a}^{b} h(x)\,dx \approx \sum_{i=1}^{n} h(x_i)\,\frac{b-a}{n}$$

where x_i is the midpoint of interval i

Monte Carlo integration



Let's instead approximate this integral using Monte Carlo. Let $X \sim U[a,b]$.

By Monte Carlo,

$$\underbrace{\mathbb{E}h(X)}_{} \approx \frac{1}{n}\sum_{i=1}^{n} h(x_i) \quad \text{where } x_1, \ldots, x_n \text{ sampled from } X$$

$$\int_{x=a}^{b} h(x)\,\Pr_X(x)\,dx = \int_{x=a}^{b} h(x)\,\frac{1}{b-a}\,dx$$

Thus,

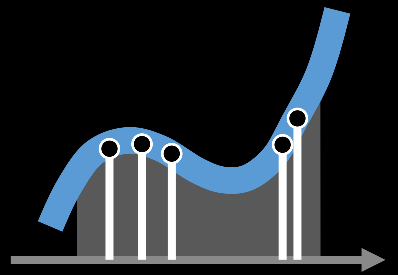
$$\int_{x=a}^{b} h(x)\,dx \approx \frac{b-a}{n}\sum_{i=1}^{n} h(x_i)$$

COMPUTATIONAL METHODS

- If we want $\mathbb{E}h(X)$ but the maths is too complicated, we can approximate
$$\mathbb{E}h(x) \approx n^{-1} \sum_{i=1}^{n} h(x_i)$$
where $x_1, \ldots, x_n$ are sampled from X

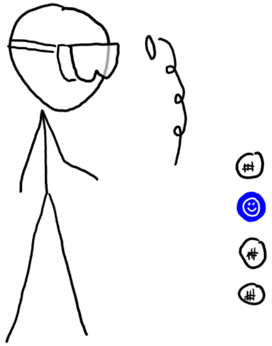
- This approximation also tells us how to estimate probabilities, since
$$\mathbb{P}(X \in A) = \mathbb{E}1_{X \in A}$$

- For computational Bayes, we need something a bit fancier: *weighted samples*



"Pasito a pasito, suave suavecito"

Luis Fonsi



probability of heads, unknown

$\Theta \sim U[0,1]$

$\downarrow$

$X \sim \mathrm{Bin}(n, \Theta)$

number of heads from 4 coin tosses

0. First write out our probability model for the data $\Pr_X(x|\Theta=\theta)$

1. Write out $\Pr_\Theta(\theta)$

2. Use the formula
$\Pr_\Theta(\theta|X=x) = \kappa \Pr_\Theta(\theta) \Pr_X(x|\Theta=\theta)$
then find κ to make this integrate to 1

… but these are usually intractable

This lets us calculate probabilities:

$$\mathbb{P}(\Theta \in \text{range}|X=x) = \int_{\theta \in \text{range}} \Pr_\Theta(\theta|X=x)\, d\theta$$

One way to do

COMPUTATIONAL BAYES

1. Generate a sample $(\theta_1, \ldots, \theta_n)$ from Θ
2. Compute weights
   $w_i = \Pr_X(x|\Theta = \theta_i)$,
   then rescale weights to sum to one

$$\mathbb{P}(\Theta \in \text{range}|X = x) \approx \sum_{i=1}^{n} w_i 1_{\theta_i \in \text{range}}$$

It's more elegant to use the generalized version

$$\mathbb{E}[h(\Theta)|X = x] \approx \Sigma_i w_i h(\theta_i)$$

ALGEBRAIC BAYES

0. First write out our probability model for the data $\Pr_X(x|\Theta = \theta)$
1. ~~Write out $\Pr_\Theta(\theta)$~~
2. Use the formula
   $\Pr_\Theta(\theta|X = x) = \kappa \Pr_\Theta(\theta) \Pr_X(x|\Theta = \theta)$
   then find κ to make this integrate to 1

… but these are usually intractable

This lets us calculate probabilities:

$$\mathbb{P}(\Theta \in \text{range}|X = x) = \int_{\theta \in \text{range}} \Pr_\Theta(\theta|X = x)\, d\theta$$

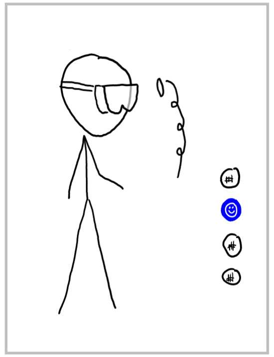
One way to do
COMPUTATIONAL BAYES

0. First write out our probability model for the data $\Pr_X(x|\Theta = \theta)$
1. Generate a sample $(\theta_1, \dots, \theta_n)$ from Θ
2. Compute weights $w_i = \Pr_X(x|\Theta = \theta_i)$, then rescale weights to sum to one

Reason about $(\Theta|X = x)$ indirectly, using

$$\mathbb{E}[h(\Theta)|X = x] \approx \Sigma_i w_i h(\theta_i)$$

Example



I got $x = 1$ head out of $n = 4$ coin tosses. I propose the probability model $X \sim \text{Bin}(n, \Theta)$. I don't know Θ, so I'll treat it as a random variable, $\Theta \sim U[0,1]$.

Plot the distribution of $(\Theta | X = x)$.

Likelihood of the data:

$X \sim \text{Bin}(n, \Theta)$

$$\Pr_X(x \mid \Theta = \theta) = \binom{n}{x} \theta^x (1-\theta)^{n-x}$$

$$= 4\, \theta\, (1-\theta)^3 \quad \text{for } n=4, x=1$$

Generate a sample $(\theta_1, \ldots, \theta_n)$ from Θ:

```
θsamp = np.random.uniform(0,1, size=1000)
```

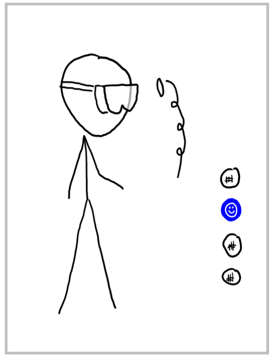
Compute weights $w_i = \Pr_X(x | \Theta = \theta_i)$, then rescale weights to sum to one:

```
w = 4 * θsamp**1 * (1-θsamp)**3
w = w / np.sum(w)
```

Reason about $(\Theta | X = x)$ indirectly, using

$$\mathbb{E}[h(\Theta) | X = x] \approx \Sigma_i w_i h(\theta_i)$$

Example



I got $x = 1$ head out of $n = 4$ coin tosses. I propose the probability model $X \sim \text{Bin}(n, \Theta)$. I don't know Θ, so I'll treat it as a random variable, $\Theta \sim U[0,1]$.

Plot the distribution of $(\Theta | X = x)$.

Reason about $(\Theta | X = x)$ indirectly, using

$$\mathbb{E}[h(\Theta)|X = x] \approx \Sigma_i w_i h(\theta_i)$$

$$\mathbb{P}(\Theta \in \text{bin} \mid \text{data}) = \mathbb{E}(1_{\Theta \in \text{bin}} \mid \text{data})$$

$$= \mathbb{E}(h(\Theta) \mid \text{data}) \quad \text{where } h(\theta) = 1_{\theta \in \text{bin}}$$

$$\approx \sum_i w_i h(\theta_i) \quad \text{where } \theta_i \text{ sampled from } \Theta \sim U[0,1]$$

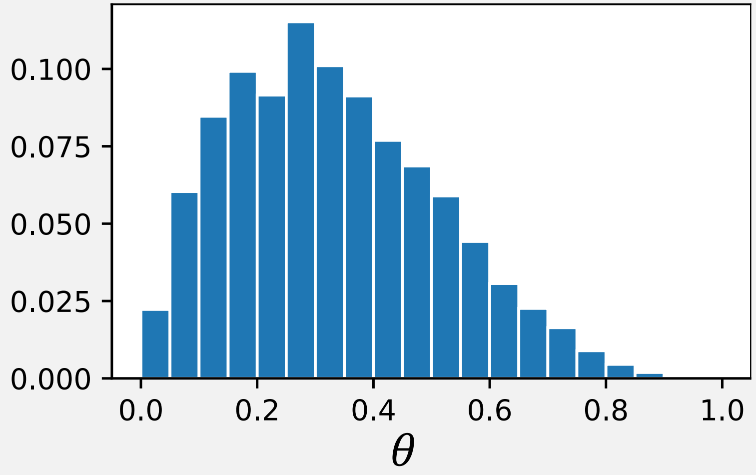
$$= \sum_i w_i 1_{\theta_i \in \text{bin}}$$

$$= \sum_{i: \theta_i \in \text{bin}} w_i$$

For each bin, sum up the weights of the θ-samples that are in that bin.

For each θ-bin, let's show a bar of height $\mathbb{P}(\theta \in \text{bin} \mid X = x)$

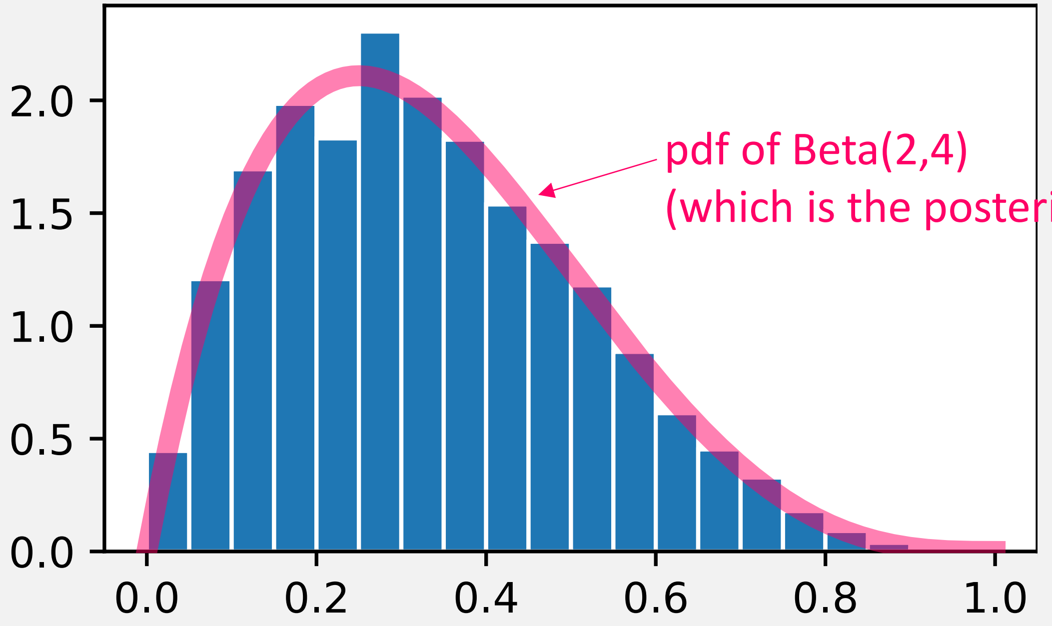
```
plt.hist(θsamp, weights=w)
```



For samples of a continuous random variable, I prefer to plot *density histograms*, where the bar heights are rescaled so that the total area is 1.

This makes them directly comparable to a pdf.

```
plt.hist(θsamp, weights=w) density=True)
```



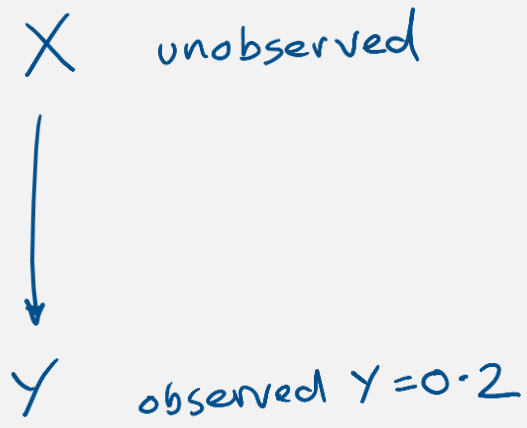
pdf of Beta(2,4)
(which is the posterior distribution we derived mathematically)

Exercise 6.2.1

Consider the probability model

```
def rxy():
    x = np.random.uniform(-1,1)
    y = np.random.normal(loc=x**2, scale=0.1)
    return (x,y)
```

Suppose we have observed $Y = 0.2$ and we want to know the likely range of X. Plot a histogram of $(X|Y = 0.2)$.



Likelihood of the data: $\Pr_Y(0.2 \mid X = x)$ = scipy.stats.norm.pdf(0.2, loc = x**2, scale = 0.1)

Generate a sample ~~$(\theta_1, \ldots, \theta_n)$~~ $x_1 \cdots x_n$ from ~~Θ~~ X:

Compute weights $w_i =$ ~~$\Pr_X(x|\Theta = \theta_i)$~~ $\Pr_Y(0.2 \mid X = x_i)$, then rescale weights to sum to one:

```
xsamp = np.random.uniform(-1, 1, size=10000)

# weight[i] = Pr_Y(0.2 | x=xsamp[i])
w = scipy.stats.norm.pdf(.2, loc=xsamp**2, scale=0.1)
w = w / np.sum(w)

plt.hist(xsamp, weights=w, density=True, bins=np.linspace(-1,1,100))
plt.show()
```

Exercise 8.3.2 (Multiple unknowns)

We have a dataset $[x_1, \dots, x_n]$. We propose to model it as independent samples from $U[A, A+B]$, where A and B are unknown parameters.

Using $A \sim \text{Exp}(0.5)$ and $B \sim \text{Exp}(1.0)$ as prior distributions for the unknown parameters, find the distribution of $(B|\text{data})$.

A, B unobserved

↓

$X_1, \dots, X_n$ observed (observed values $x_1, \dots, x_n$).

Likelihood of the data:

$$\Pr(x_1, \dots, x_n \mid A=a, B=b) = \prod_{i=1}^{n} \Pr(x_i \mid A=a, B=b)$$ since our model says they're independent

$$= \prod_{i=1}^{n} \left\{ \frac{1}{b} 1_{a \le x_i \le b} \right\}$$ the pdf of $U[a, a+b]$

$$= \frac{1}{b^n} 1_{a \le \min x_i} 1_{\max x_i \le a+b}$$

$((a_1, b_1), \dots, (a_n, b_n))$ (A, B)

Generate a sample ~~$(\theta_1, \dots, \theta_n)$~~ from ~~$\Theta$~~:

$\Pr(\text{data} \mid (A, B) = (a, b))$

Compute weights $w_i =$ ~~$\Pr_X(x|\Theta = \theta_i)$~~, then rescale weights to sum to one:

```
x = [2, 3, 2.1, 2.4, 3.14, 1.8]

# Assume that A and B are independent. To generate samples of (A,B) ...
asamp = np.random.exponential(scale=1/0.5, size=1000000)
bsamp = np.random.exponential(scale=1/1.0, size=1000000)
#absamp = zip(asamp, bsamp)

w = 1/bsamp**len(x) * np.where((asamp <= min(x)) & (max(x) <= asamp+bsamp), 1, 0)
w = w / np.sum(w)

plt.hist(bsamp, weights=w, density=True, bins=np.linspace(0,5,100))
plt.show()
```

Exercise 8.3.2 (Multiple unknowns)

We have a dataset $[x_1, \ldots, x_n]$. We propose to model it as independent samples from $U[A, A+B]$, where A and B are unknown parameters.

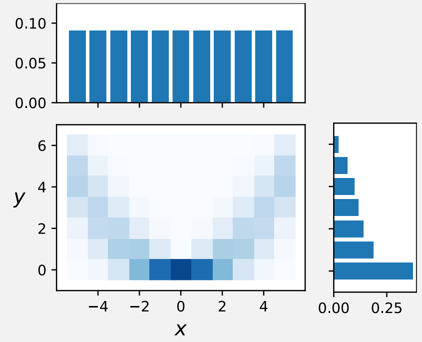
Using $A \sim \text{Exp}(0.5)$ and $B \sim \text{Exp}(1.0)$ as prior distributions for the unknown parameters, find the distribution of $(B|\text{data})$.

TIP. If n is large, you can run into underflow problems if you compute $\Pr(x_1, \ldots, x_n|\text{params})$ directly.

Be clever about rescaling the weights, using the log-sum-exp trick (exercise 8.3.4).

TIP. First find the joint posterior distribution for *all* the unknown parameters. Then, pick out just the one you're interested in.

We call this *marginalization.*



Why does computational Bayes work?

$\Theta \longrightarrow X$

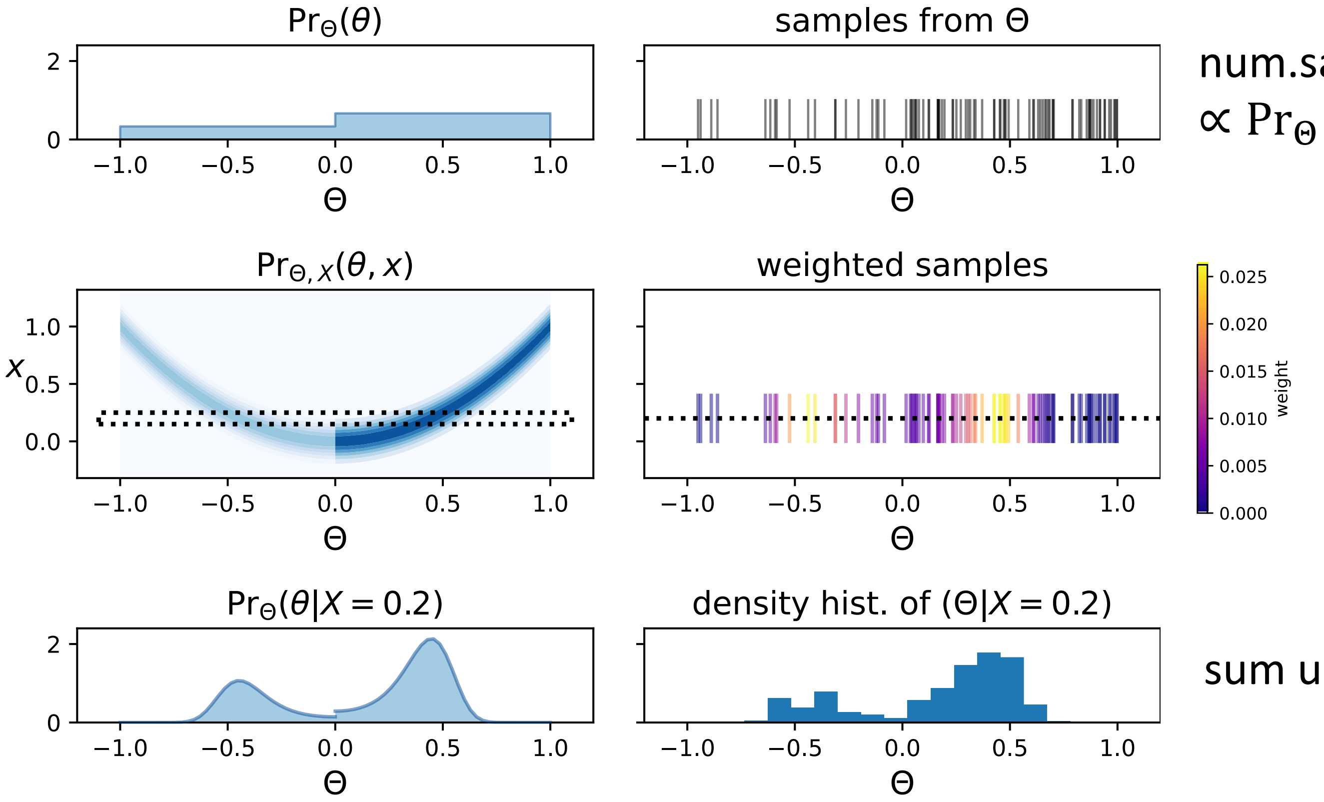
non-uniform distribution

$\sim N(\Theta^2, 0.1^2)$

Joint pdf

$\Pr_{\Theta,X}(\theta, x)$

$= \Pr_\Theta(\theta)\,\Pr_X(x|\Theta = \theta)$



num.samples near θ

$\propto \Pr_\Theta(\theta)$

weight $w_i = \Pr_X(x|\Theta = \theta_i)$

sum up the weights in each bin

$\Pr_\Theta(\theta|X = x)$

$\propto \Pr_{\Theta,X}(\theta, x)$

$\propto \Pr_\Theta(\theta)\,\Pr_X(x|\Theta = \theta)$

bin height at θ

$\propto$ num.samples × weights

$\propto \Pr_\Theta(\theta) \times \Pr_X(x|\Theta = \theta)$

Functionally speaking

$$\mathrm{o}(\mathrm{e}) = \Phi(c)$$

Observations

Effects

Causes

$$\mathrm{o} = \Psi(e)$$

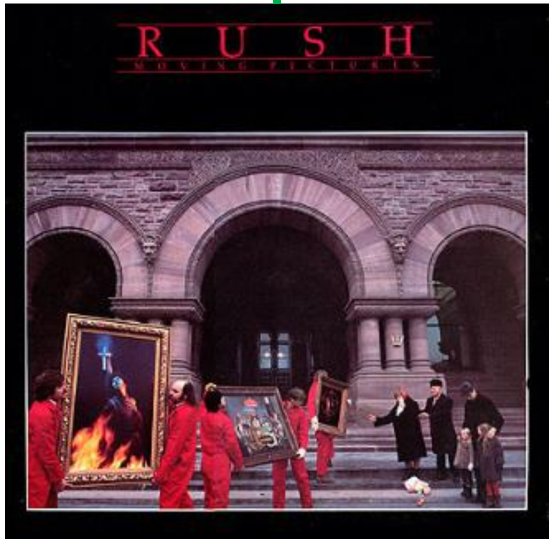
Causal AI aim to retrieve the function Φ having knowledge of the causes (and typically having some information/assumption on Ψ) $\leftrightarrow$ *inverse problems*

Several times acquired in statistical terms → computational Bayes!

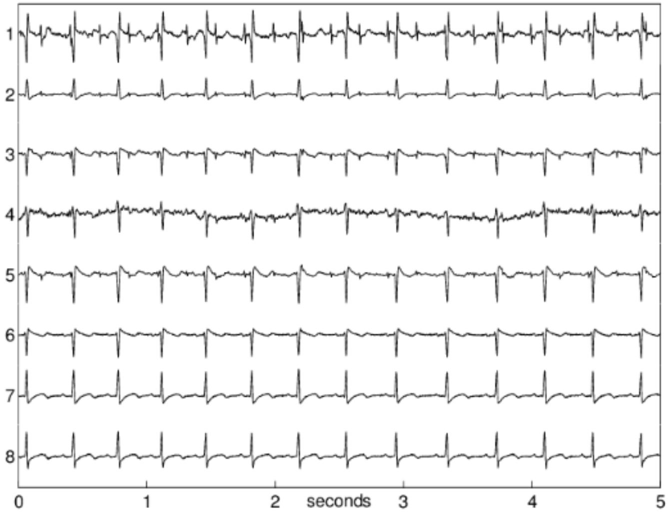
An example – remastering music from the ‘70s

$$\mathrm{o}(\mathrm{e}) = \Phi(c)$$

$$\tilde{c} \propto \Phi^{-1}(o(e))$$



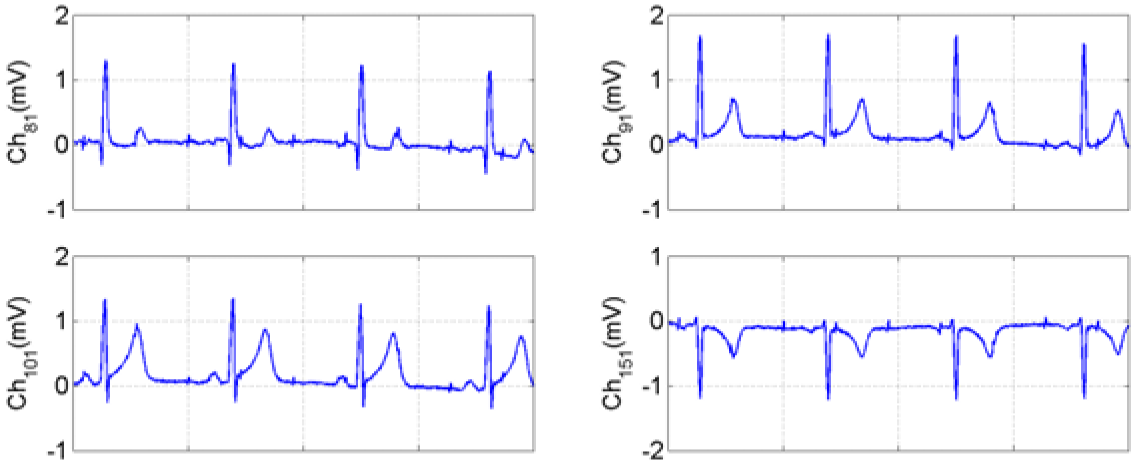
Another example – ECG of pregnant women



$$o(e) = \Phi(c)$$

$$\tilde{c} \propto \Phi^{-1}(o(e))$$

Early detection of cardiac issues for the baby and the mother



Reverend Thomas Bayes, 1701–1761

Bayes's rule for random variables

$$\Pr_X(x|Y=y) = \Pr_X(x)\,\frac{\Pr_Y(y|X=x)}{\Pr_Y(y)}$$

$$\mathbb{P}(X \in \mathrm{A}|Y=y) \approx \sum_{i=1}^{n} w_i 1_{x_i \in A}$$

X unobserved (latent) variable

↓

Y we have observed the value of Y

Bayesianism

Whenever there's an unknown parameter, you should express your uncertainty about it by treating it as a random variable.

Warning:
the physical world
is not / might not be
random!