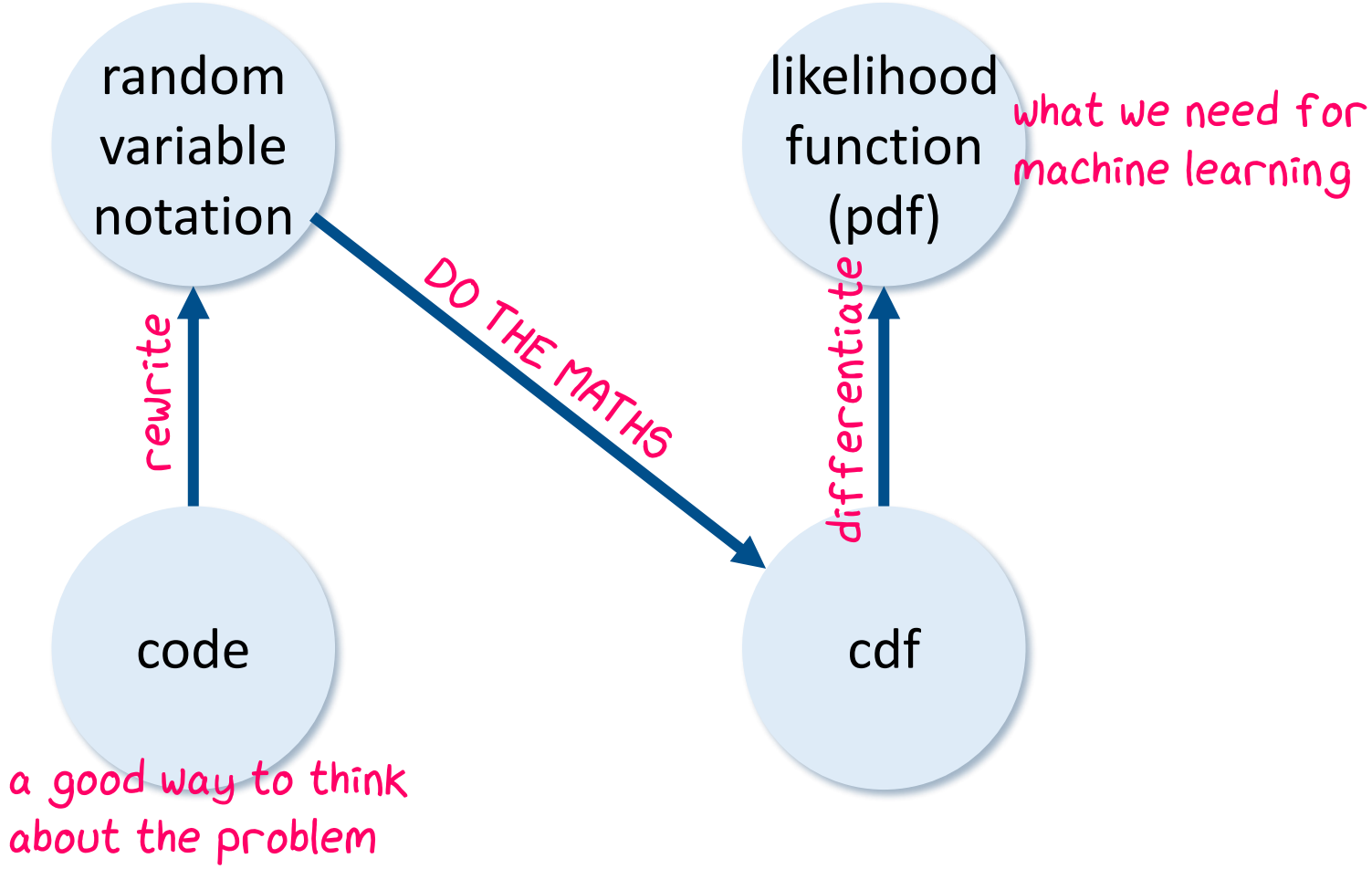
LAST LECTURE

Bespoke probability distributions: from code to likelihood

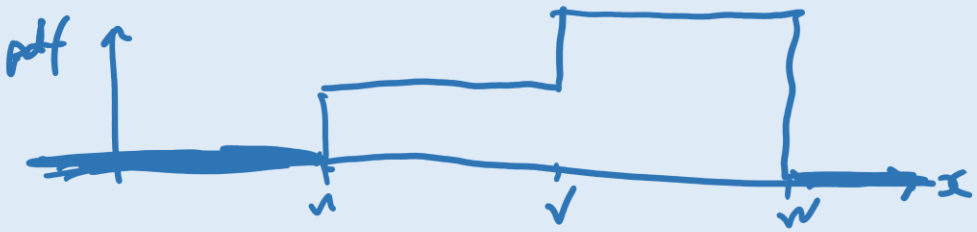
There are four ways to specify a distribution:

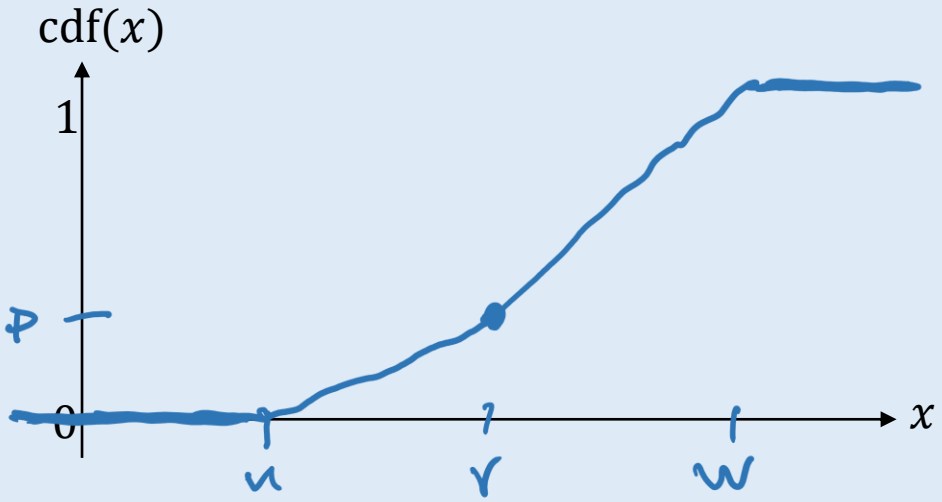


EXERCISE

What's the pdf for this random variable?

```
def rx(u,v,w,p):
    # preconditions: u < v < w, and 0 < p < 1
    k = np.random.choice(["left","right"], [p,1-p])
    if k == "left":
        return np.random.uniform(u,v)
    else:
        return np.random.uniform(v,w)
```

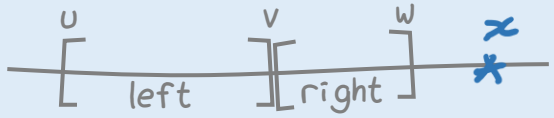




① Rewrite it in random variable notation

$$\text{Let } K = \begin{cases} \text{left} & \text{with prob. } p \\ \text{right} & \text{with prob. } 1-p \end{cases}$$

$$\text{Let } X \sim \begin{cases} U[u,v] & \text{if } K = \text{left} \\ U[v,w] & \text{if } K = \text{right} \end{cases}$$



② Do the maths: simplify the cdf into elementary building blocks

$$\mathbb{P}(X \le x) = \mathbb{P}(X \le x | K = \text{left}) \times \mathbb{P}(K = \text{left}) + \mathbb{P}(X \le x | K = \text{right}) \times \mathbb{P}(K = \text{right})$$

by the Law of Total Probability

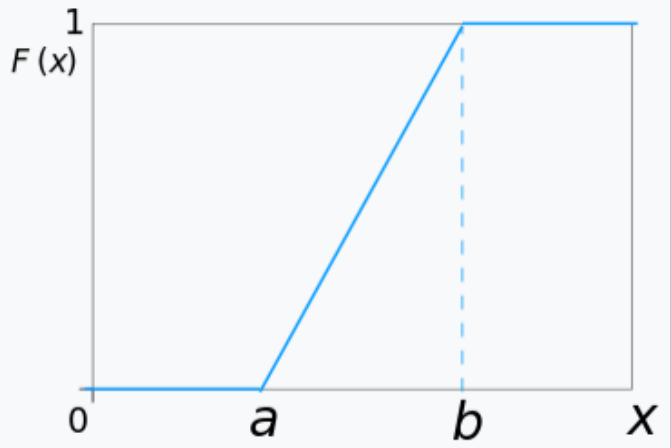
$$\mathbb{P}(X = x) = \sum_y \mathbb{P}(X = x | Y = y)\mathbb{P}(Y = y)$$

$$= p\,\mathbb{P}(U[u,v] \le x) + (1-p)\,\mathbb{P}(U[v,w] \le x)$$

$$= \begin{cases} \text{if } x < u: & p \times 0 + (1-p) \times 0 = 0 \\ \text{if } x \in [u,v]: & p \times \frac{x-u}{v-u} + (1-p) \times 0 \\ \text{if } x \in [v,w]: & p \times 1 + (1-p) \times \frac{x-v}{w-v} \\ \text{if } x > w: & 1 \end{cases}$$

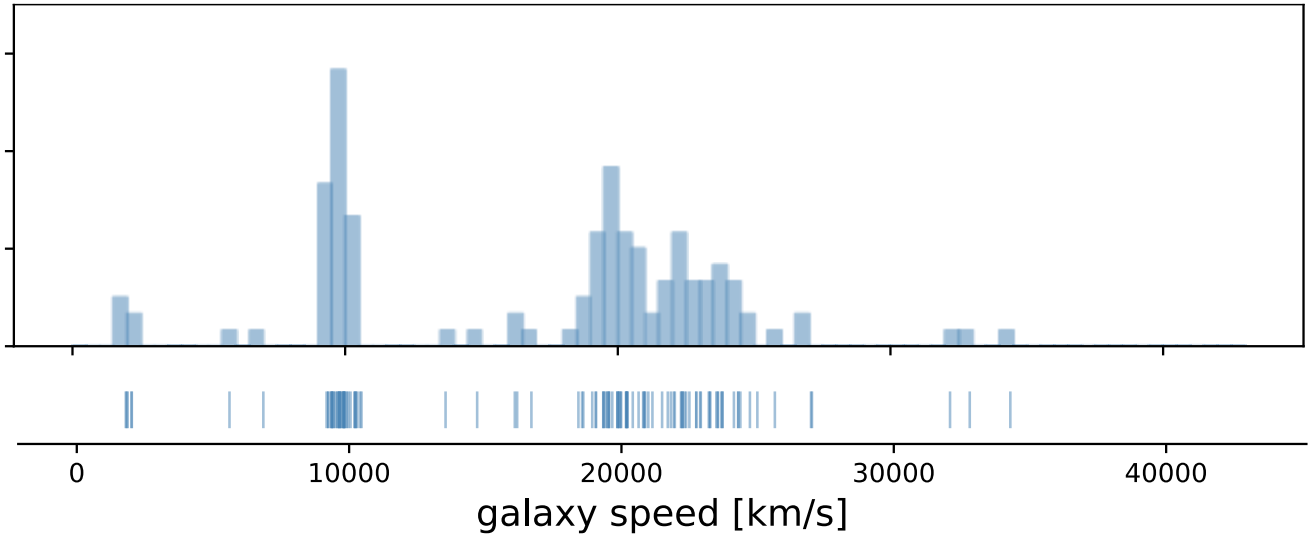
Sanity check: at $x = v$, $\begin{cases} p \\ p. \end{cases}$ ✓

Wikipedia: Uniform distribution

Cumulative distribution function


Notation	$\mathcal{U}_{[a,b]}$
Parameters	$-\infty < a < b < \infty$
Support	$x \in [a, b]$
PDF	$\begin{cases} \frac{1}{b-a} & \text{for } x \in [a, b] \\ 0 & \text{otherwise} \end{cases}$
CDF	$\begin{cases} 0 & \text{for } x < a \\ \frac{x-a}{b-a} & \text{for } x \in [a, b] \\ 1 & \text{for } x > b \end{cases}$

Our goal:
to find the best distribution we can to fit this dataset.



IA Probability lecture 10

Empirical cumulative distribution functions

Empirical Distribution Functions (Example 1/2)

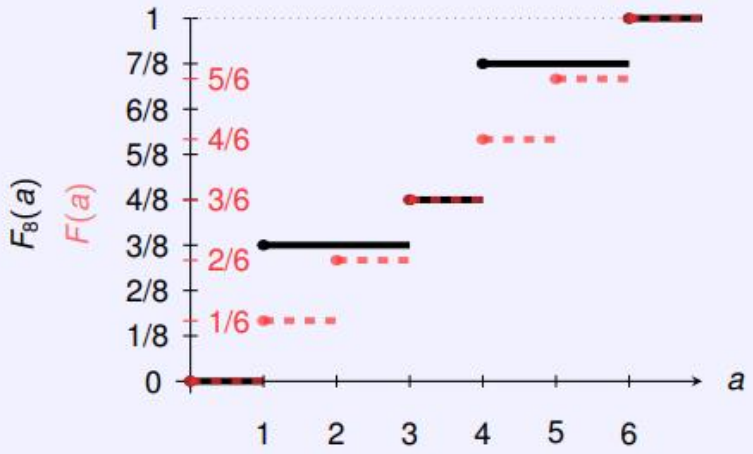
Example 1

Consider throwing an unbiased dice 8 times, and let the realisation be:

$$(x_1, x_2, \ldots, x_8) = (4, 1, 5, 3, 1, 6, 4, 1).$$

What is the Empirical Distribution Function $F_8(a)$?

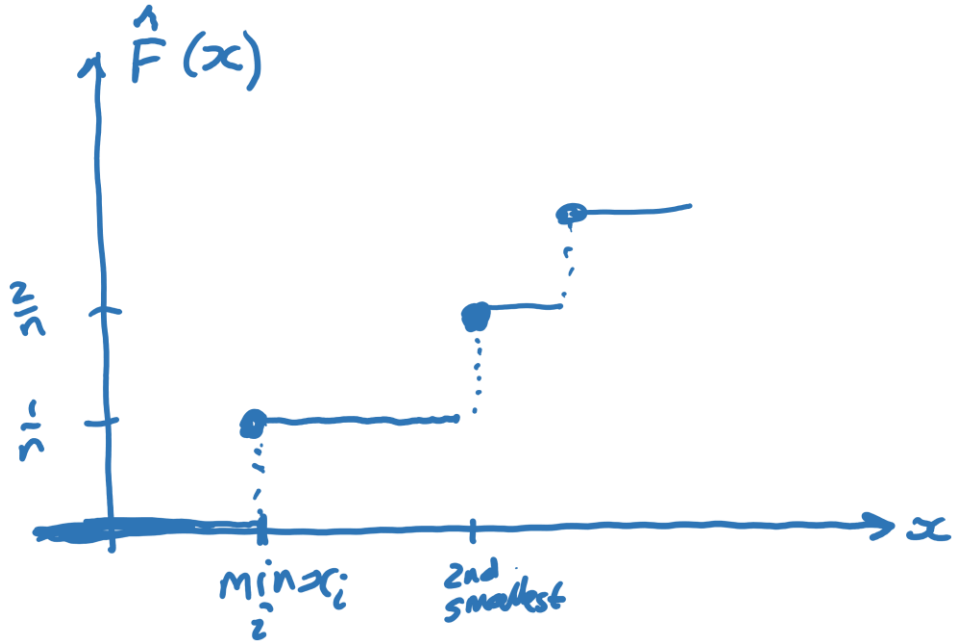
Answer



ECDF

Given a dataset of numerical values $[x_1, x_2, \ldots, x_n]$, the empirical cumulative distribution function or ecdf is

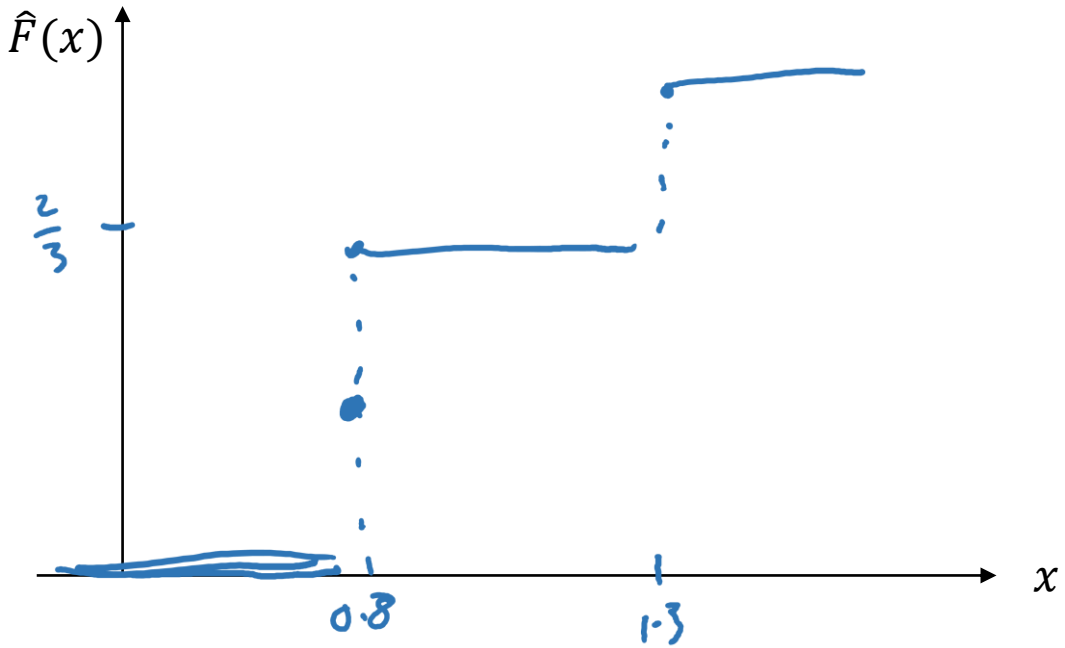
$$\hat{F}(x) = \frac{1}{n}\begin{pmatrix}\text{how many datapoints} \\ \text{there are } \leq x\end{pmatrix}$$




```
x = [...]
F = np.arange(1, len(x)+1) / len(x)
plt.plot(np.sort(x), F, drawstyle='steps-post')
```

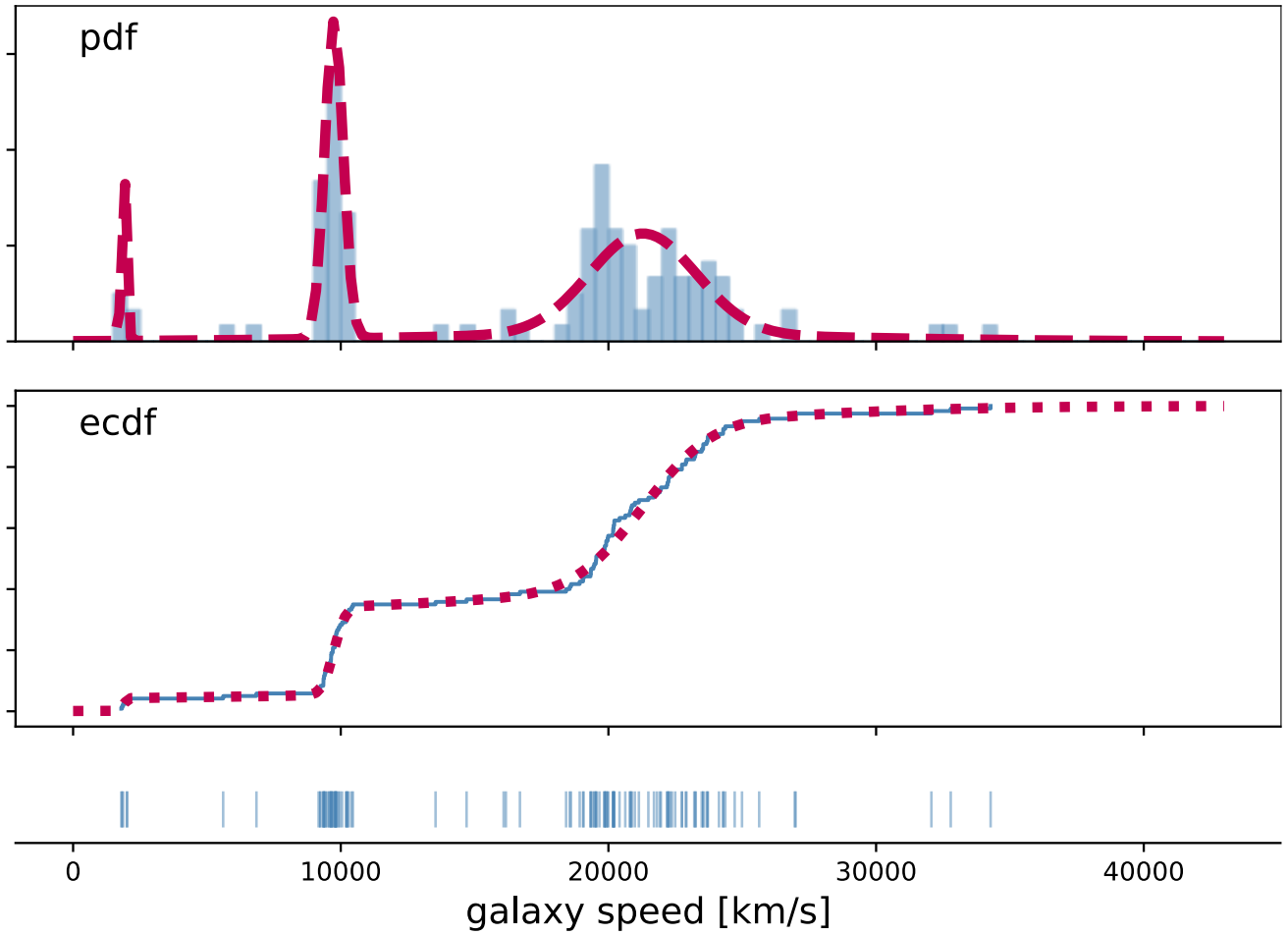
What if there are repeated values in the dataset, e.g.

```
x = [0.8, 0.8, 1.3]
```



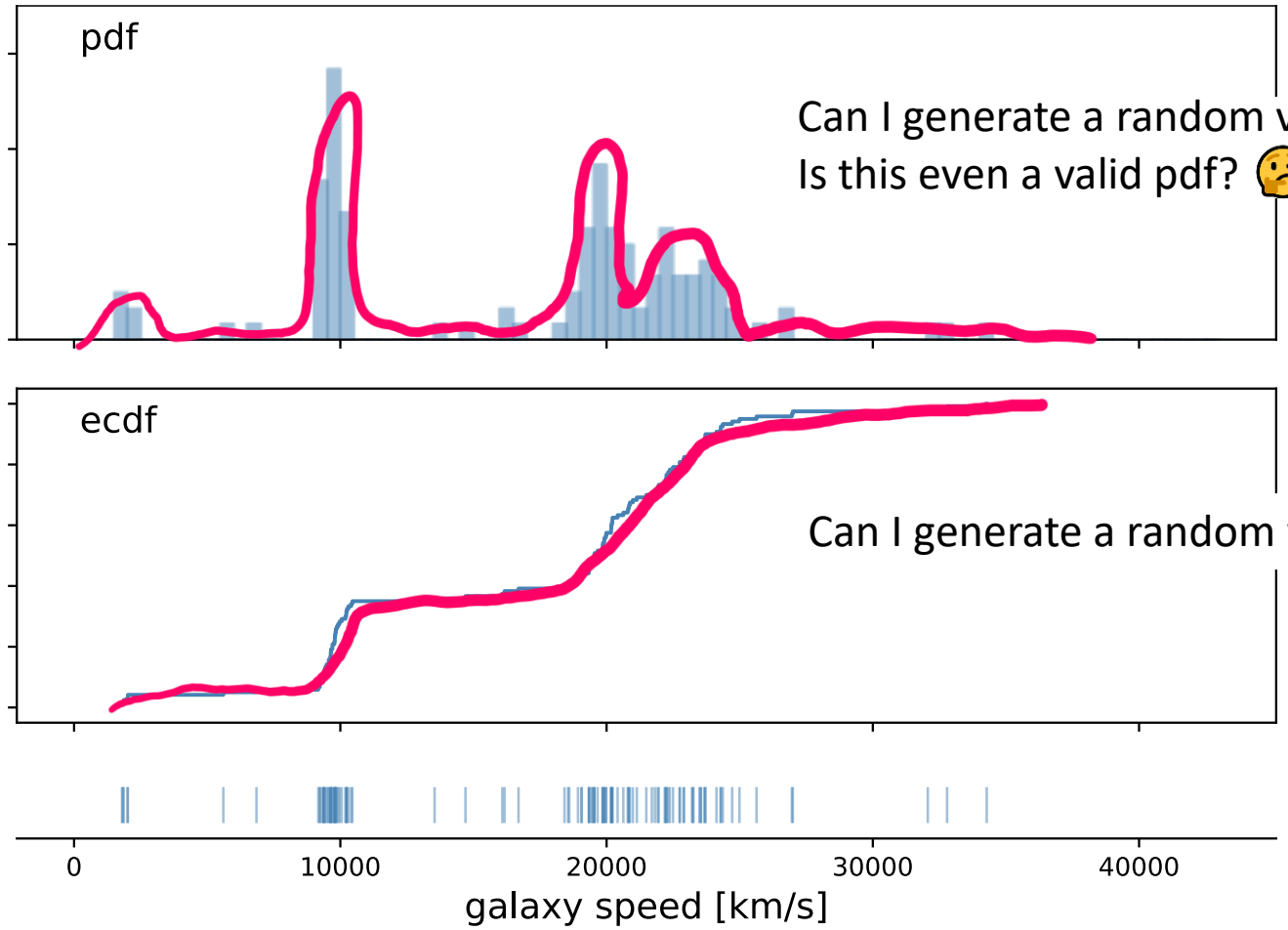

```
x = [...]
F = np.arange(1, len(x)+1) / len(x)
plt.plot(np.sort(x), F, drawstyle='steps-post')
```

(This code will plot an extra point at (0.8, 1/3), but who cares? The plot is still correct.)



fitted Gaussian mixture model

But can I find a better-fitting distribution?



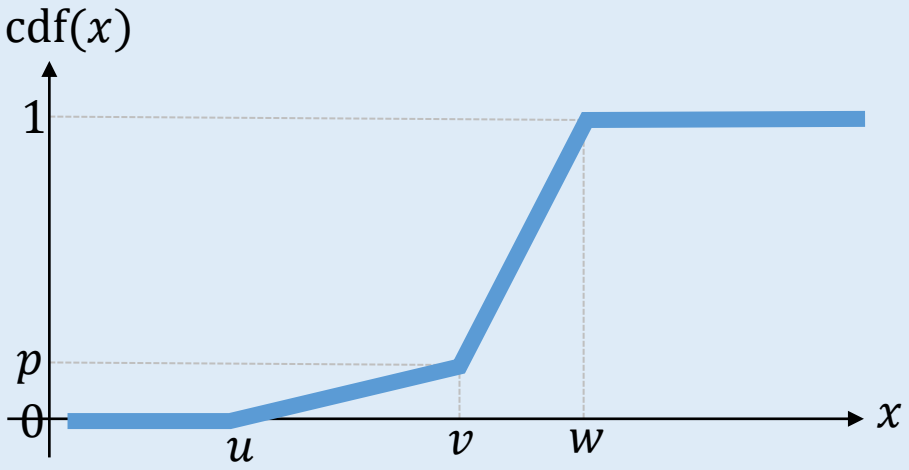
Can I generate a random variable with this pdf?
Is this even a valid pdf? 🤔 🤷‍♀️

Can I generate a random variable with this cdf?

It's certainly a valid cdf:
it starts at 0, goes to 1,
and is non-decreasing.

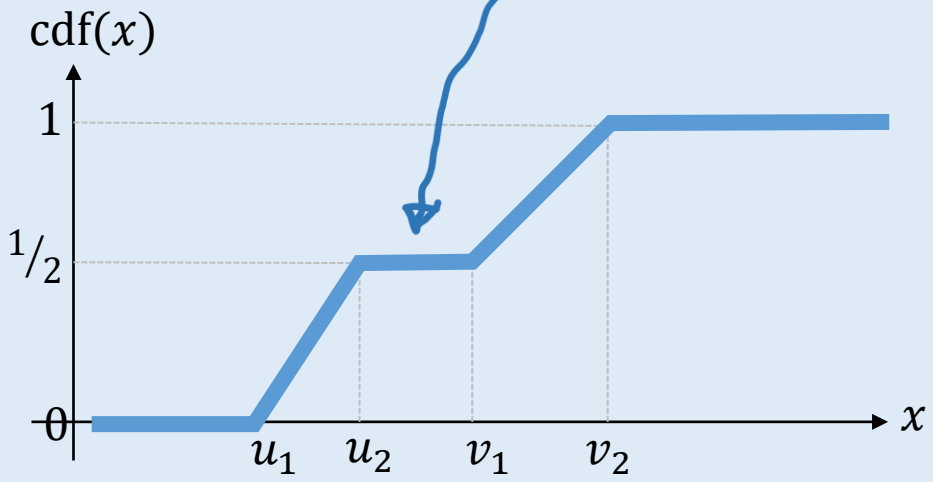
Let's build up our skills at turning cdf plots into code.

But can I find a better-fitting distribution?



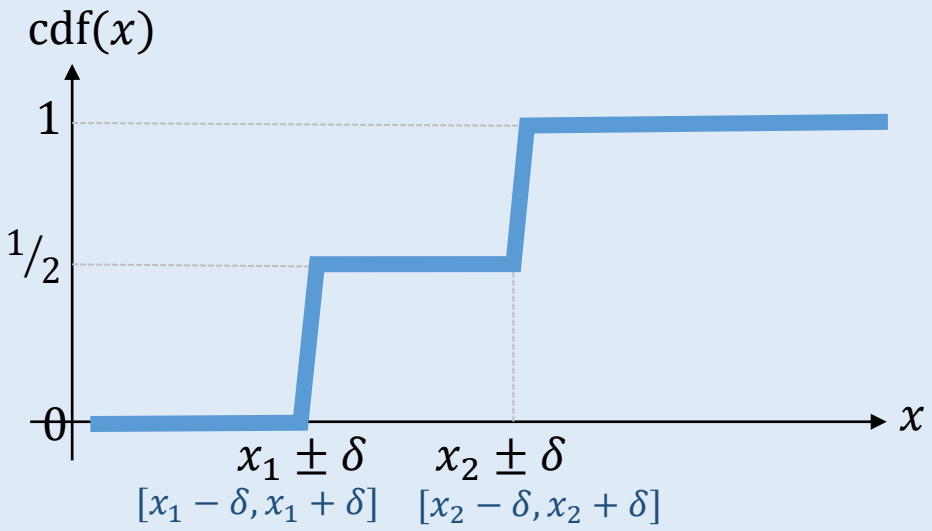

```
def rx(u,v,w,p):
    k = np.random.choice(["left","right"], [p,1-p])
    if k == "left":
        return np.random.uniform(u,v)
    else:
        return np.random.uniform(v,w)
```

cdf is flat ⇒ pdf=0 ⇒ don't generate any values in $[u_2, v_1]$



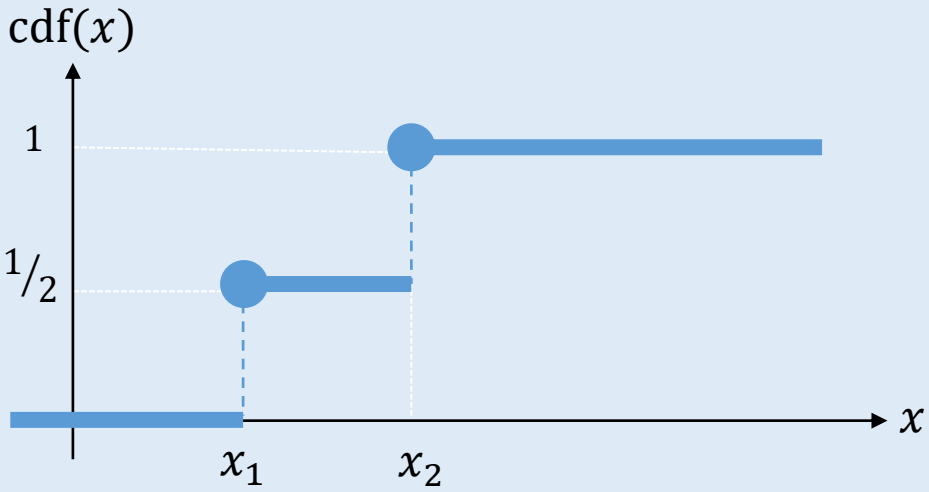

```
def rx(u1,u2,v1,v2):
    # pick either left or right, with equal probability
    k = np.random.choice(["left","right"])
    if k == "left":
        return np.random.uniform(u1,u2)
    else:
        return np.random.uniform(v1,v2)
```

This cdf has equal probabilities ($p = \frac{1}{2}$). That's the default for np.random.choice — no need to specify $[\frac{1}{2}, \frac{1}{2}]$.




```
def rx(x1,x2,δ):
    k = np.random.choice(["left","right"])
    if k == "left":
        return np.random.uniform(x1 − δ,x1 + δ)
    else:
        return np.random.uniform(x2 − δ,x2 + δ)
```

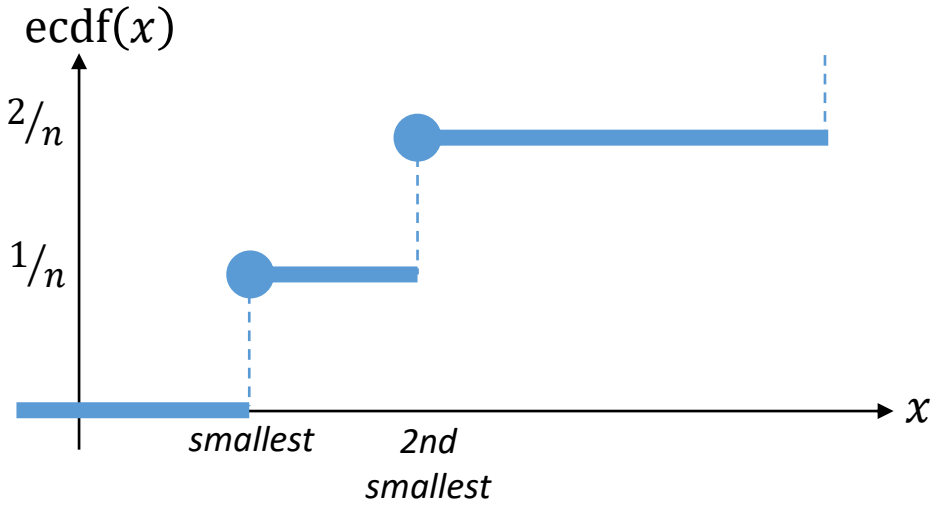
Extreme case as $\delta \to 0$




```
def rx(x1,x2):
    k = np.random.choice(["left","right"])
    if k == "left":
        return x1
    else:
        return x2
```

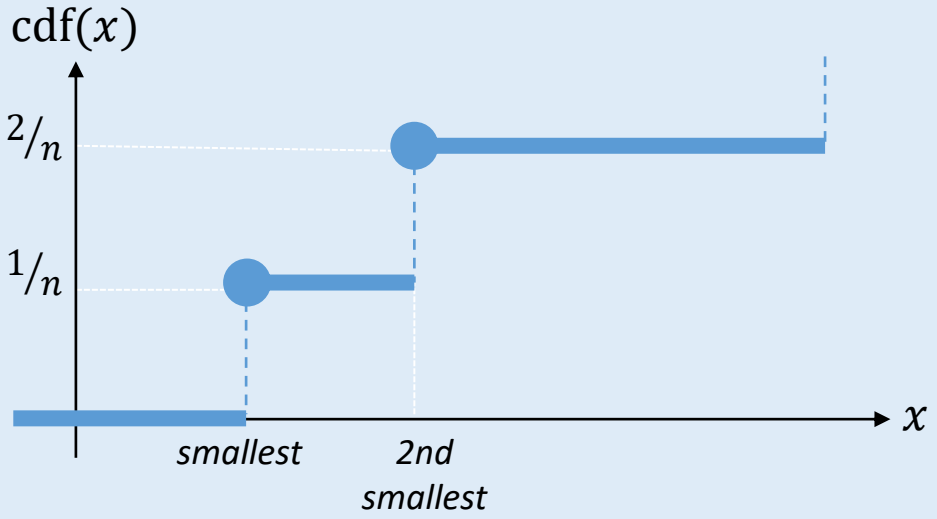
return np.random.choice([x1,x2])

obviously generalizes to lists.



Recall the empirical distribution for a dataset $\vec{x} = (x_1, x_2, \ldots, x_n)$:

$$\text{ecdf}(x) = \frac{1}{n}(\#\text{points} \leq x)$$



To generate a random variable $\hat{X}$ whose cdf matches exactly this step function:

```
def rxhat([x_1, ..., x_n]):
    return np.random.choice([x_1, ..., x_n])
```

This is a perfect fit to the dataset!

The empirical distribution

Given a dataset $[x_1, x_2, \ldots, x_n]$
let $\hat{X}$ be the random variable obtained by picking one of the x_i at random.
(This is a discrete random variable.)

We say this random variable has *the empirical distribution of the dataset*.

The ecdf only applies to real-valued random variables, whereas this definition makes sense for any type of data (text, images, etc.)

Instead of saying "the cdf of $\hat{X}$ matches the ecdf of the data", we can say

$$\mathbb{P}(\hat{X} \in A) = \frac{1}{n}\sum_{i=1}^{n} 1_{x_i \in A}$$

$$\mathbb{E}\, h(\hat{X}) = \frac{1}{n}\sum_{i=1}^{n} h(x_i)$$

Applications of the empirical distribution

"Whenever you want to work with a true distribution, you can just bung in an empirical distribution instead."

Monte Carlo

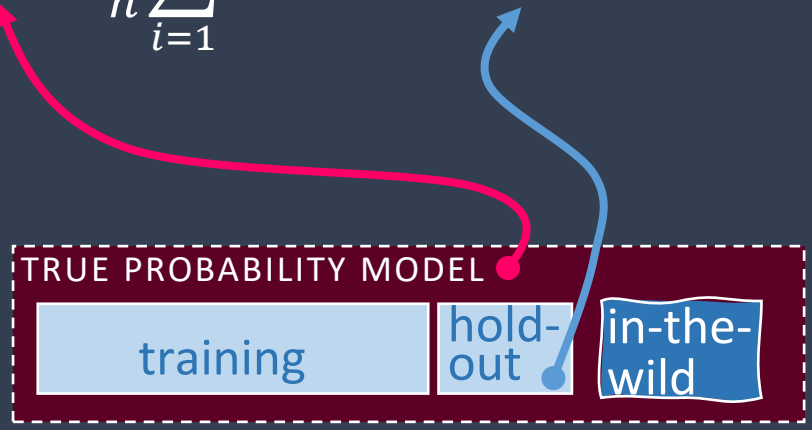
When we want probabilities / expectations
but the maths is too hard
generate a sample and work with it instead.

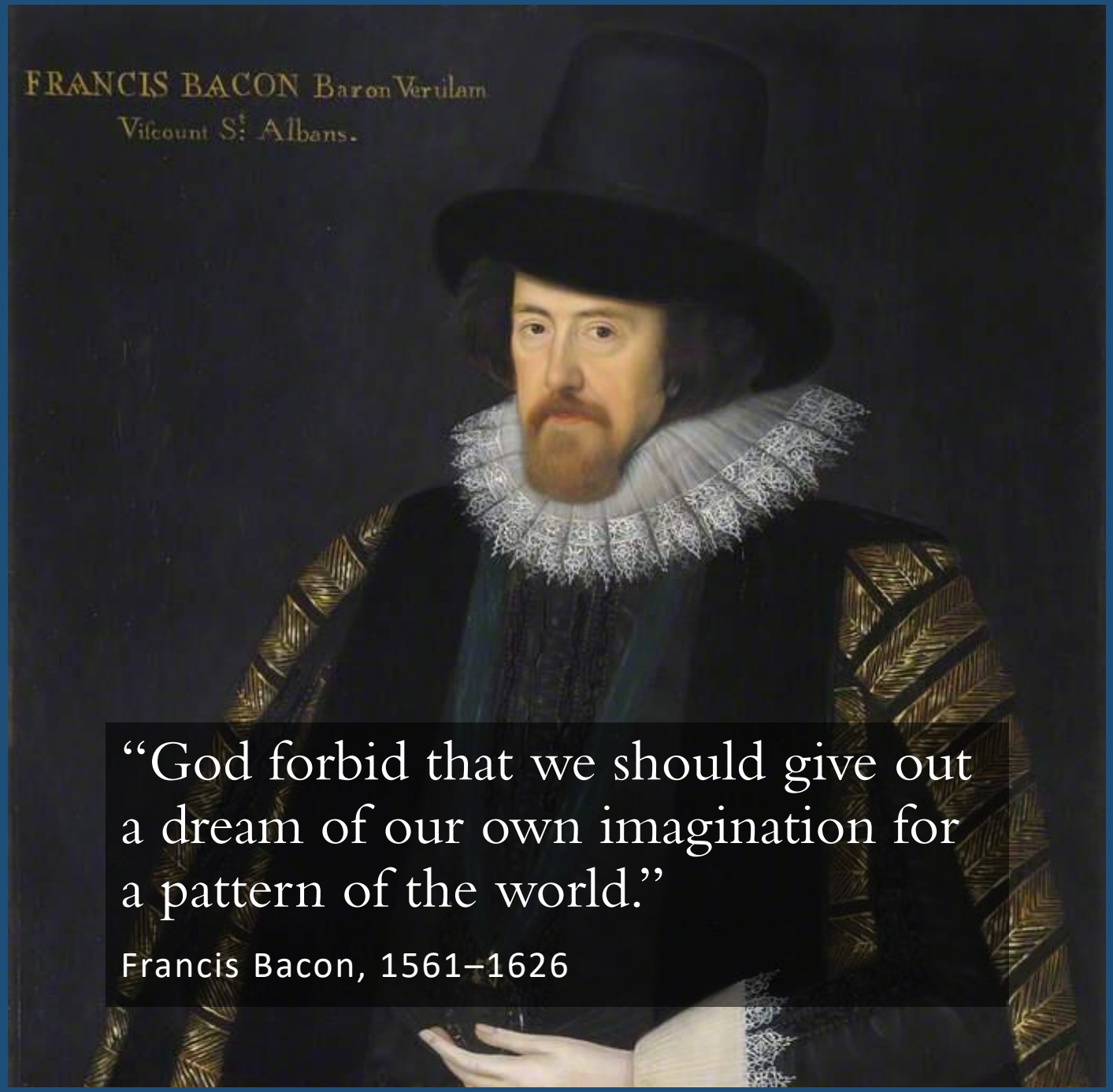
$$\mathbb{E}\,h(X) \approx \frac{1}{n}\sum_{i=1}^{n} h(x_i) \;= \mathbb{E}\,h(\hat{X})$$

Holdout approximation

When we want probabilities / expectations
but we don't know the true distribution,
find a sample and work with it instead.

$$\mathbb{E}\,h(X) \approx \frac{1}{n}\sum_{i=1}^{n} h(x_i) = \mathbb{E}\,h(\hat{X})$$



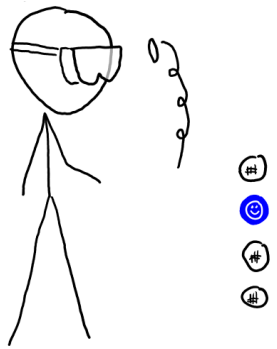


"God forbid that we should give out a dream of our own imagination for a pattern of the world."

Francis Bacon, 1561–1626

The empirical distribution is a perfect fit for a dataset. Why bother fitting a parametric probability model at all?

§9. The frequentist approach to generalization



I tossed four coins
and got $x = 1$ head.
My data model is

$$X \sim \text{Bin}(4, \theta)$$

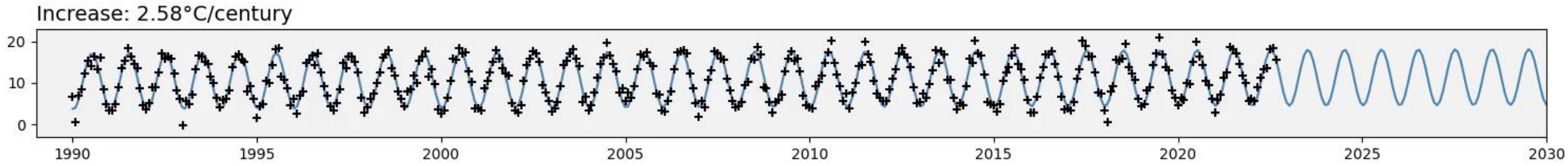
What can I say about the **laws of nature**, i.e. about the true value of θ?

I saw x=1. Let me go figure out how likely is each possible explanation $\Theta=\theta$.

Bayes's rule:

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

I saw x=1, $\hat{\theta}=1/4$, **IN THIS REALITY.** What was $\hat{\theta}$ in other dimensions of the multiverse?



Climate confidence challenge.

Find a 95% confidence interval for γ, for Cambridge from 1985 to the present. (It's your choice how to simulate the multiverse.)

Please submit your answer on Moodle by Tuesday 11 November

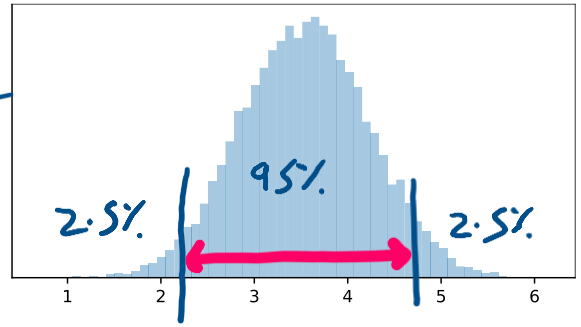
Confidence intervals via resampling

Given a dataset x,

1. Decide on a readout function $t(x)$
2. *"Simulate a multiverse of datasets."*
   - Fit a model for the dataset
   - Let X^* be a random synthetic dataset, generated from the fitted model
   - Simulate many synthetic datasets
3. Compute t for each dataset, and report the spread of t

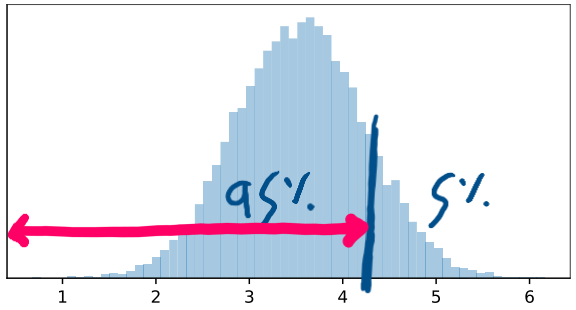
   for example with a histogram or a confidence interval

This has to be a computable function of x, i.e. it's not allowed to have any unknown parameters. Such a function is called a statistic.



Two-sided 95% confidence interval

```
np.quantile(tsamples, [.025, .975])
```



One-sided 95% confidence interval

```
np.quantile(tsamples, [0,.95])
```

Example.
We are given a dataset

$$x = [4.3, 5.1, 6.1, 6.8, 7.4, 8.8, 9.9]$$

which we decide to model as independent samples from $N(\mu, \sigma^2)$. Find a 95% confidence interval for $\hat{\mu}$.

```
# 1. Define a readout statistic
def t(x): return np.mean(x)

# 2. To generate a synthetic dataset ...
def rx_star():
    return np.random.choice(x, size=len(x))


# 3. Sample the readout statistic, and report its spread
t_ = [t(rx_star()) for _ in range(10000)]
lo,hi = np.quantile(t_, [.025, .975])
```

since the MLE $\hat{\mu}$ is just the sample mean

i.e. to simulate what the dataset might have been, simply sample n values from the empirical distribution (which is a perfect fit to the data)

Example 9.2.1.

We are given a dataset

$$x = [4.3, 5.1, 6.1, 6.8, 7.4, 8.8, 9.9]$$

which we decide to model as independent samples from $N(\mu, \sigma^2)$. Find a 95% confidence interval for $\hat{\mu}$.

```
# 1. Define a readout statistic
def t(x): return np.mean(x)

# 2. To generate a synthetic dataset ...
μhat = np.mean(x)
σhat = np.sqrt(np.mean((x-μhat)**2))
def rx_star():
    return np.random.normal(loc=μhat, scale=σhat, size=len(x))

# 3. Sample the readout statistic, and report its spread
t_ = [t(rx_star()) for _ in range(10000)]
lo,hi = np.quantile(t_, [.025, .975])
```

i.e. to simulate what the dataset might have been,
fit the probability model $N(\mu,\sigma^2)$, then sample n values from it

Confidence intervals via parametric resampling

Given a dataset x
and a parametric probability model $\Pr(x;\theta)$

1. Decide on a readout function $t(x)$
2. *"Simulate a multiverse of datasets."*
   - Fit this model, i.e. estimate $\hat{\theta}$
   - Let X^* be a random synthetic dataset, generated from the fitted model
   - Simulate many synthetic datasets
3. Compute t for each dataset, and report the spread of t

   for example with a histogram
   or a confidence interval

Exercise 9.2.3 (Comparing groups).

We are given data $x = [x_1, \dots, x_m]$ which we believe is $N(\mu, \sigma^2)$ and further data $y = [y_1, \dots, y_n]$ which we believe is $N(\mu + \delta, \sigma^2)$. Find a 95% confidence interval for $\hat{\delta}$.

The MLEs for μ, δ, σ are what you calculated in Example Sheet 1 question 4:

$\hat{\mu} = \bar{x}$

$\hat{\delta} = \bar{y} - \bar{x}$

$\hat{\sigma} = \cdots$

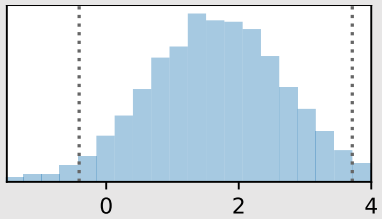
```
1  x = [4.3, 5.1, 6.1, 6.8, 7.4, 8.8, 9.9]
2  y = [8.3, 8.5, 8.9]
3  m,n = len(x), len(y)

4  # 1. Define the readout statistic
5  def t(x,y): return np.mean(y) - np.mean(x)

6

7  # 2. To generate a synthetic dataset ...
8  μ̂,δ̂ = np.mean(x), np.mean(y) - np.mean(x)
9  σ̂ = np.sqrt((np.sum((x-μ̂)**2 + np.sum((y-μ̂-δ̂)**2))/(m+n))
10 def rxy_star():
11     return (np.random.normal(loc=μ̂, scale=σ̂, size=m),
12             np.random.normal(loc=μ̂ + δ̂, scale=σ̂, size=n))

13 # 3. Sample the readout statistic, and report its spread
14 t_ = [t(*rx_star()) for _ in range(10000)]
15 lo,hi = np.quantile(t_, [.025, .975])
16 plt.hist(t_)
```

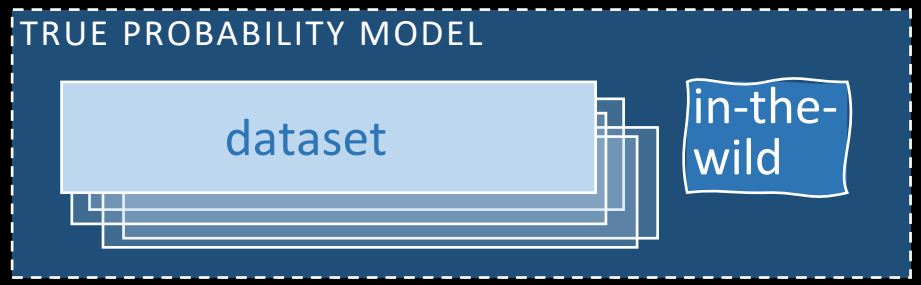


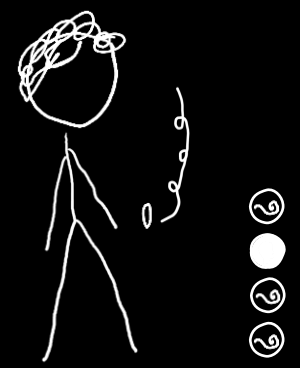
When you fit a model by maximizing $\Pr(\text{data}; \text{params})$ use ALL the data and ALL the parameters.

When you resample, resample the FULL data.

GENERALIZATION, ACCORDING TO FREQUENTISTS

I trained my ML system on a dataset. How will it work on in-the-wild data?





- The job of a ML system is to report an output, e.g. a parameter or a prediction.
- I want to know what output my system might report on in-the-wild data.
- Let me consider an *ensemble* of systems, trained on many possible datasets. What range of outputs do they report?
- This range says how confident I can be about my system's output (assuming I've even got the probability model right).

My system reports a parameter estimate $\hat{\theta}$. What's the frequency of $\hat{\theta} \in [lo, hi]$ across the multiverse?

Office hours
1–1.30pm in the cafe area today

No lecture on Friday
Only Mon+Wed from now on