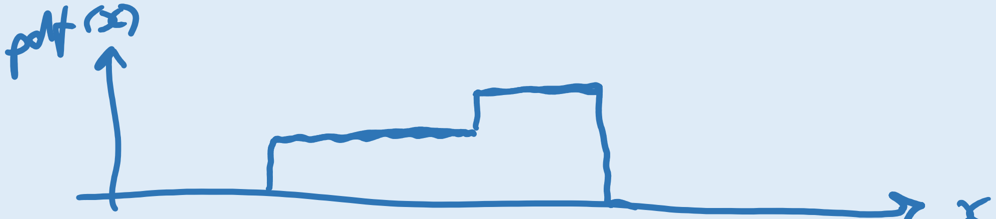
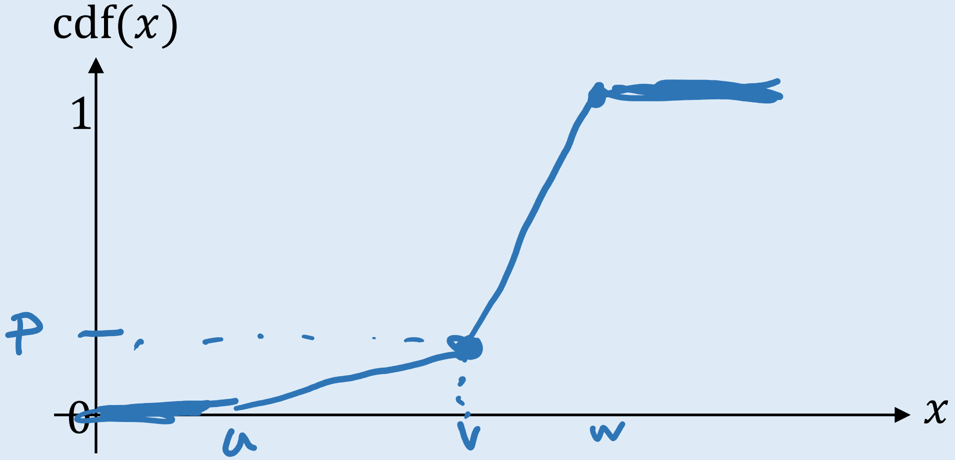
EXERCISE

What's the cdf for this random variable?

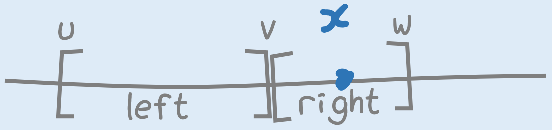



```
def rx(u,v,w,p):
    # preconditions: u < v < w, and 0 < p < 1
    k = np.random.choice(["left","right"], [p,1-p])
    if k == "left":
        return np.random.uniform(u,v)
    else:
        return np.random.uniform(v,w)
```



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

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



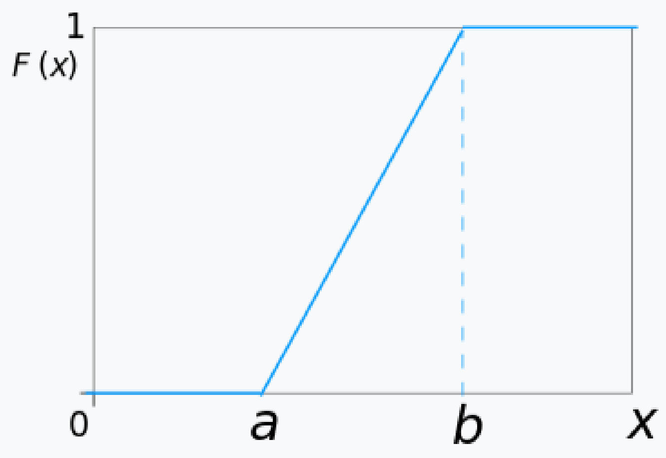
$\mathbb{P}(X \le x) = \mathbb{P}(X \le x | K = \text{left}) \times \mathbb{P}(K = \text{left}) + \mathbb{P}(X \le x | K = \text{right}) \times \mathbb{P}(K = \text{right})$ by the Law of Total Probability

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

Note: at $x = v$, both these cases agree, $\mathbb{P}(X \le v) = p$.

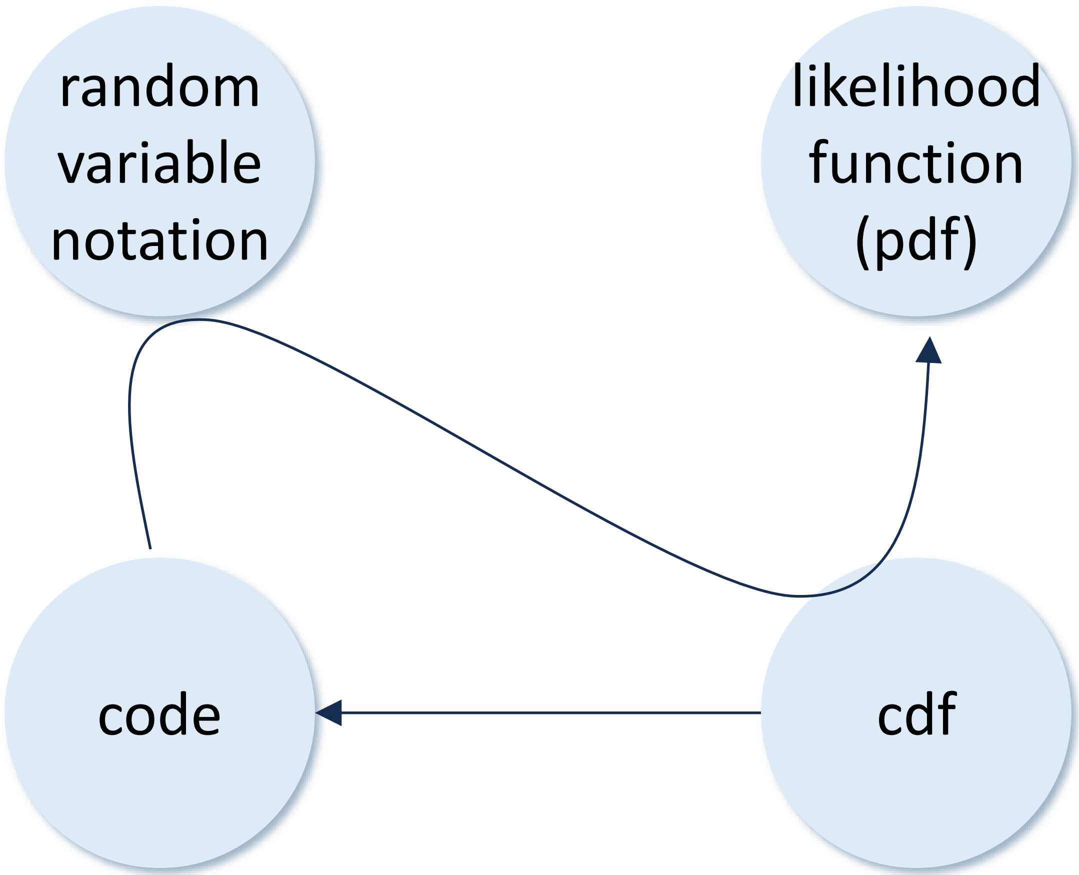
Wikipedia: Uniform distribution

Cumulative distribution function

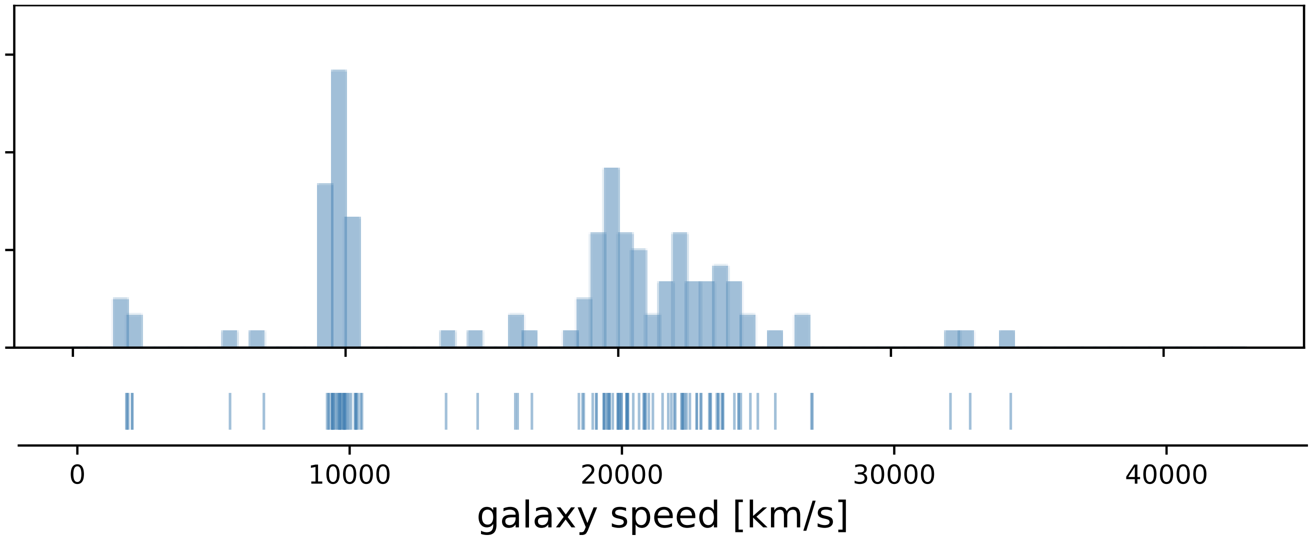


Notation	$\mathcal{U}_{[a,b]}$
Parameters	$-\infty < a < b < \infty$
Support	$x \in [a, b]$
PDF	$\begin{cases} \frac{1}{b-a} & \text{for } x \in [a,b] \\ 0 & \text{otherwise} \end{cases}$
CDF	$\begin{cases} 0 & \text{for } x < a \\ \frac{x-a}{b-a} & \text{for } x \in [a,b] \\ 1 & \text{for } x > b \end{cases}$

Bespoke probability distributions



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)

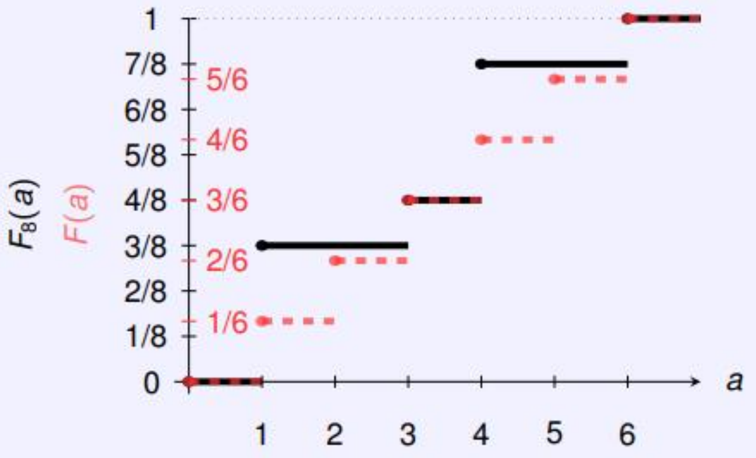
Example 1

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

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

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

Answer



Empirical Distribution Functions (Example 2/2)

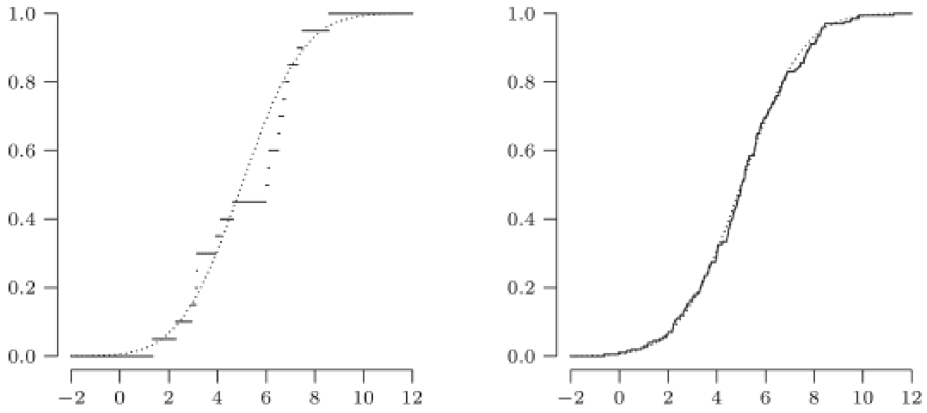


Fig. 17.1. Empirical distribution functions of normal samples.

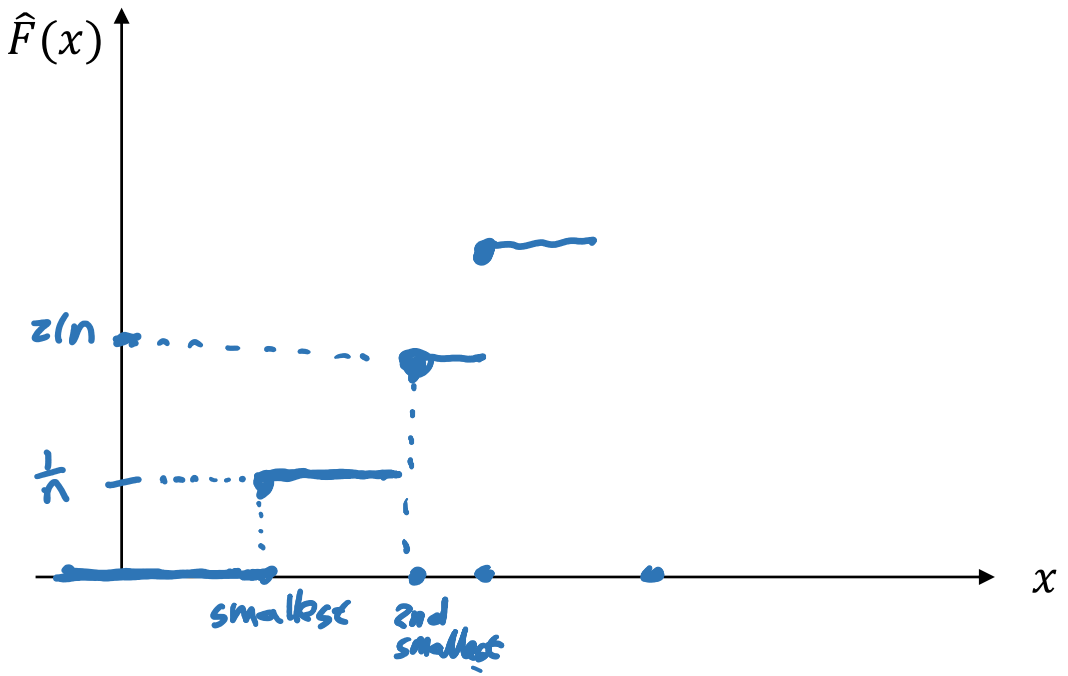
Source: Modern Introduction to Statistics

Figure: Empirical Distribution Functions of samples from a Normal Distribution $\mathcal{N}(5, 4)$ ($n = 20$ left, $n = 200$ right)

ECDF

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

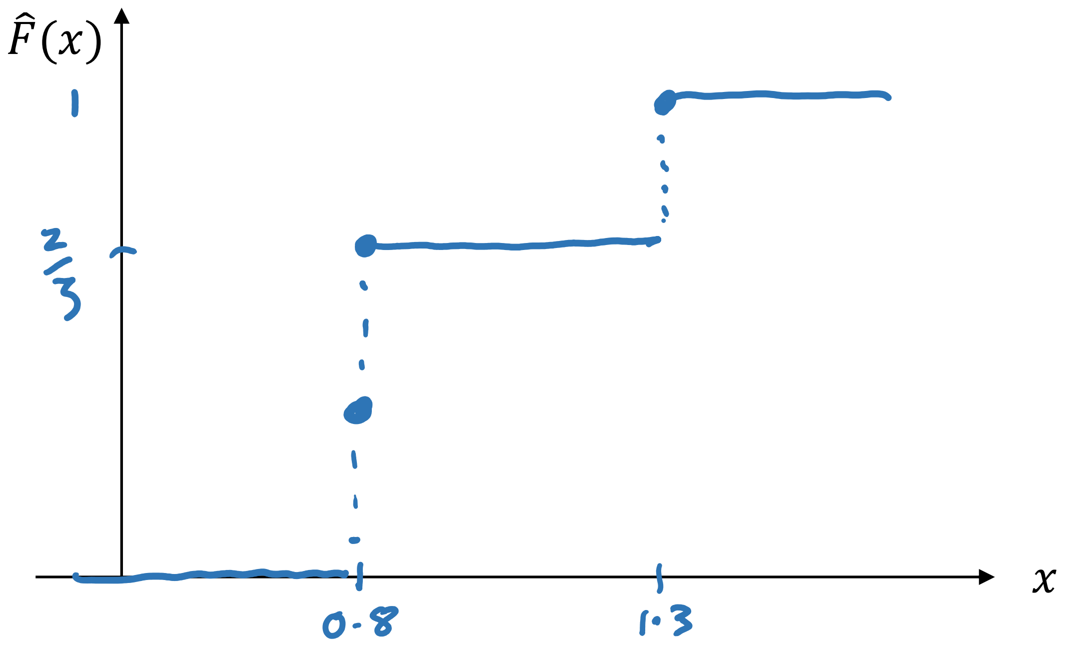
$$\hat{F}(x) = \frac{1}{n}\begin{pmatrix}\text{how many datapoints} \\ \text{there are } \leq x\end{pmatrix}$$




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

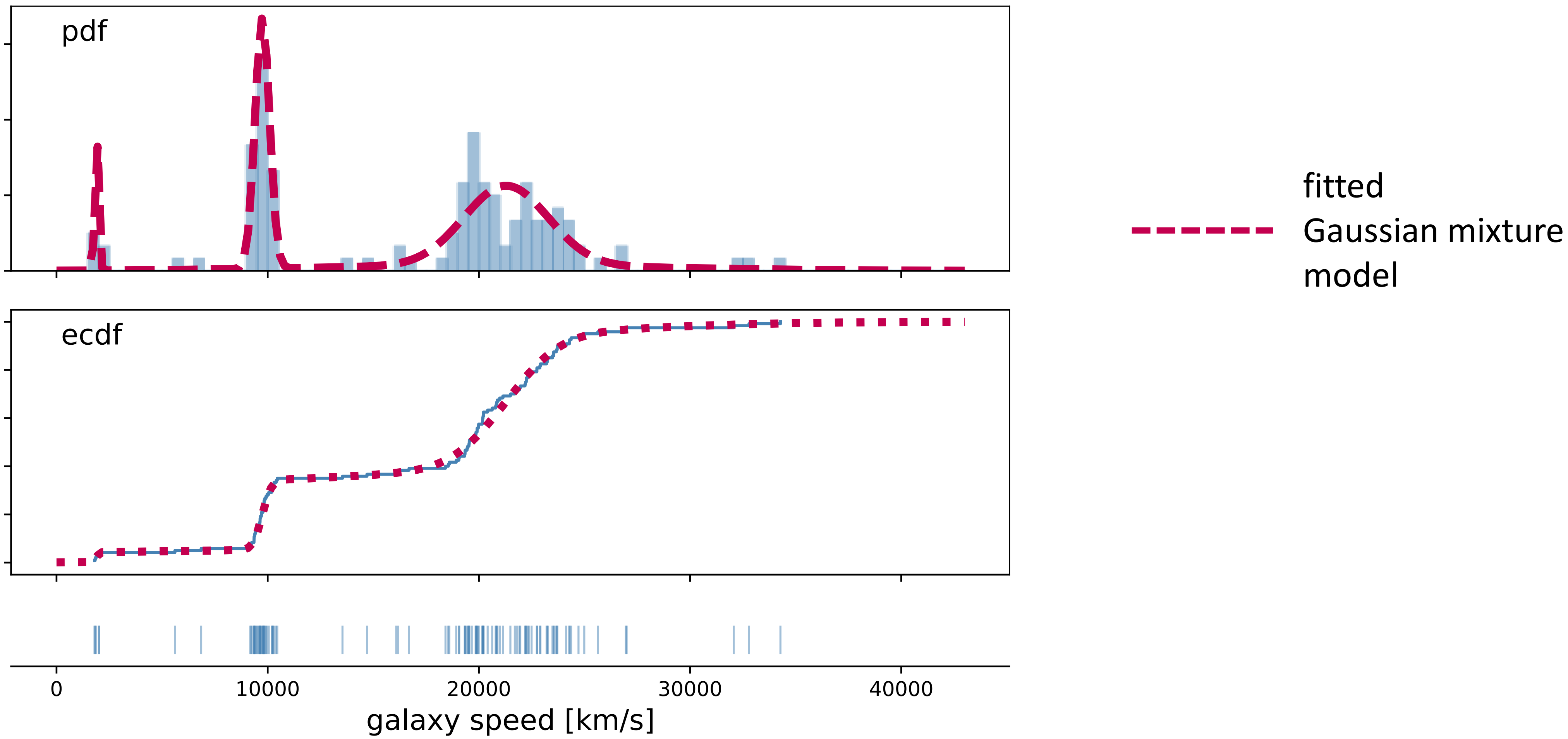
What if there are repeated values in the dataset, e.g.

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

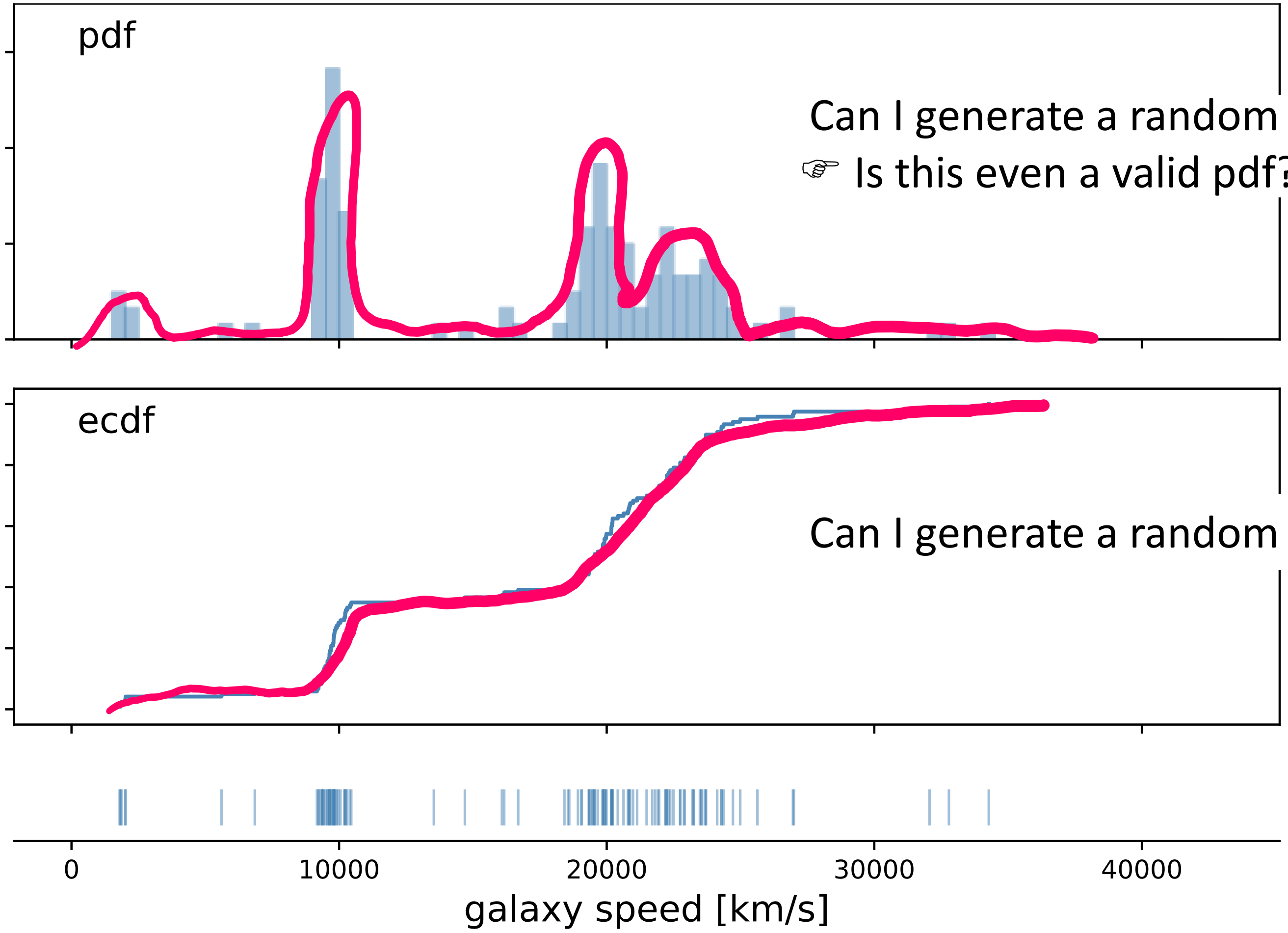



```
x = [...]
F = np.arange(1, len(x)+1) / len(x)
plt.plot(np.sort(x), F, drawstyle='steps-post')
```

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



But can I find a better-fitting distribution?



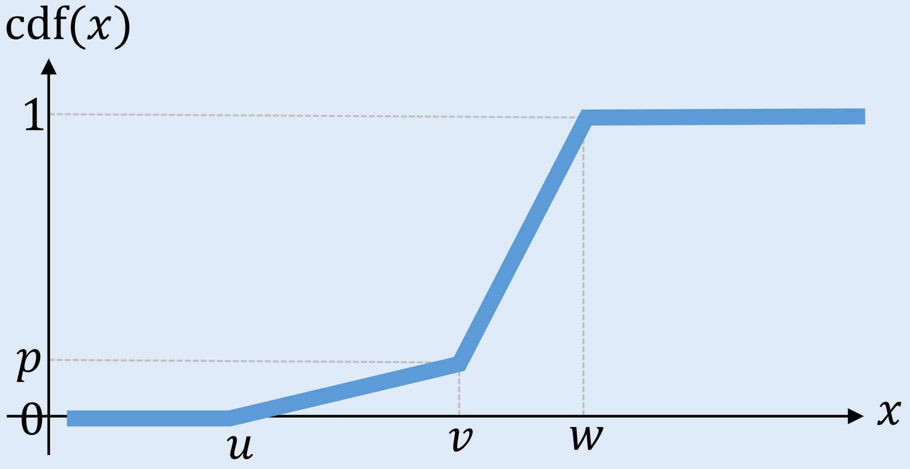
Can I generate a random variable with this pdf?

☞ Is this even a valid pdf?

Can I generate a random variable with this cdf?

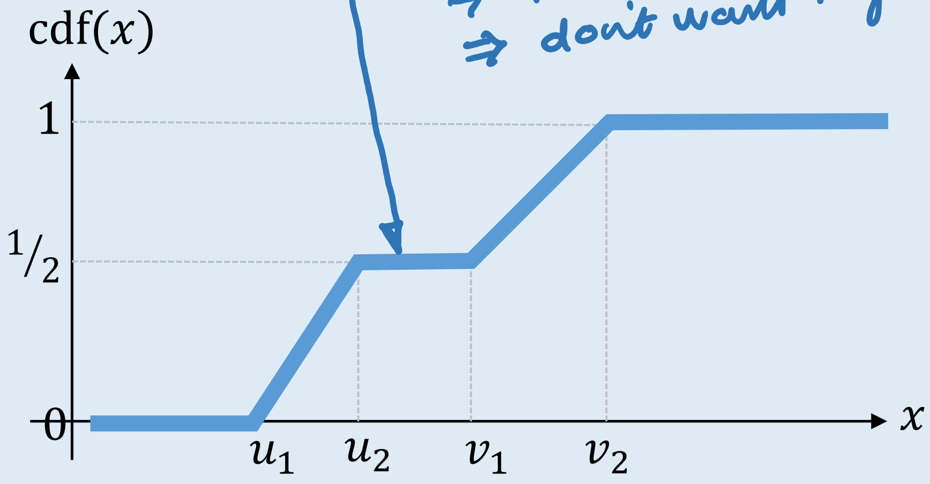
It's certainly a valid cdf: it starts at 0, goes to 1, and is non-decreasing.

But can I find a better-fitting distribution?

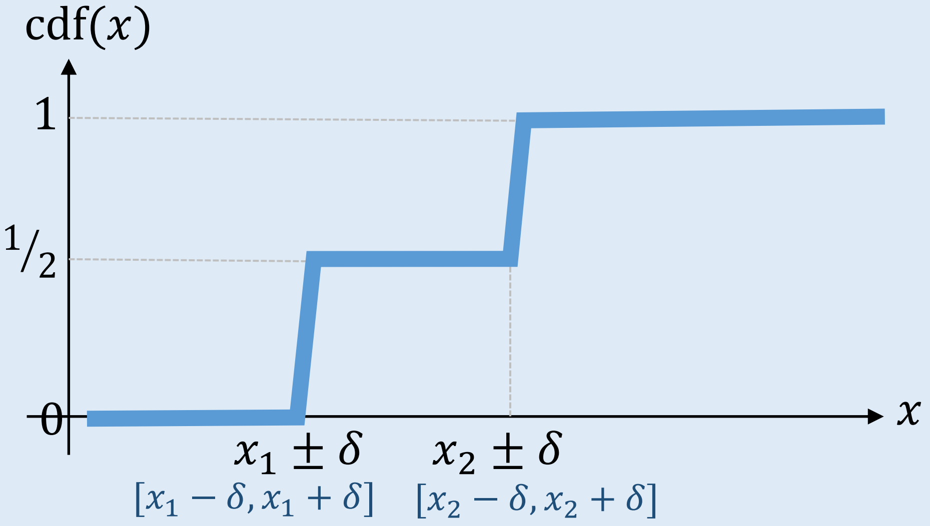



```
def rx(u,v,w,p):
    k = np.random.choice(["left","right"], [p,1-p])
    if k == "left":
        return np.random.uniform(u,v)
    else:
        return np.random.uniform(v,w)
```

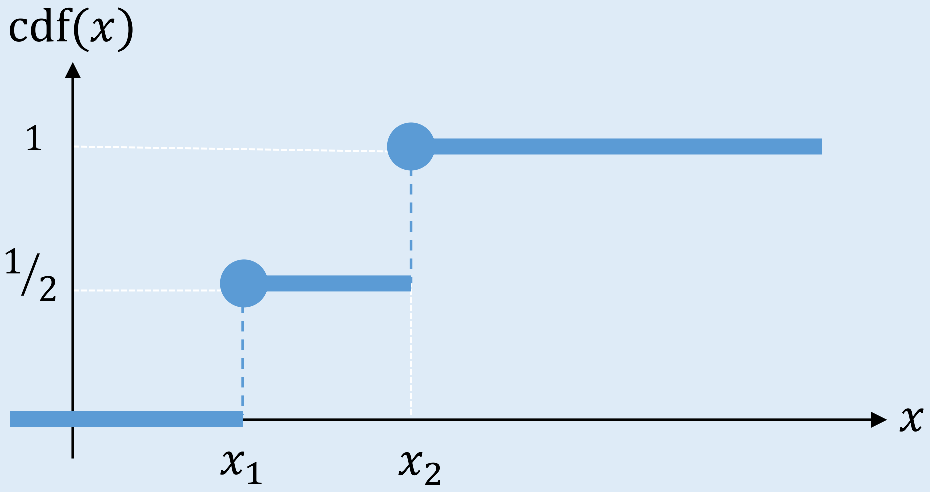
cdf = const
⇒ pdf = 0
⇒ don't want to generate any values in $[u_2, v_1]$




```
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)
```

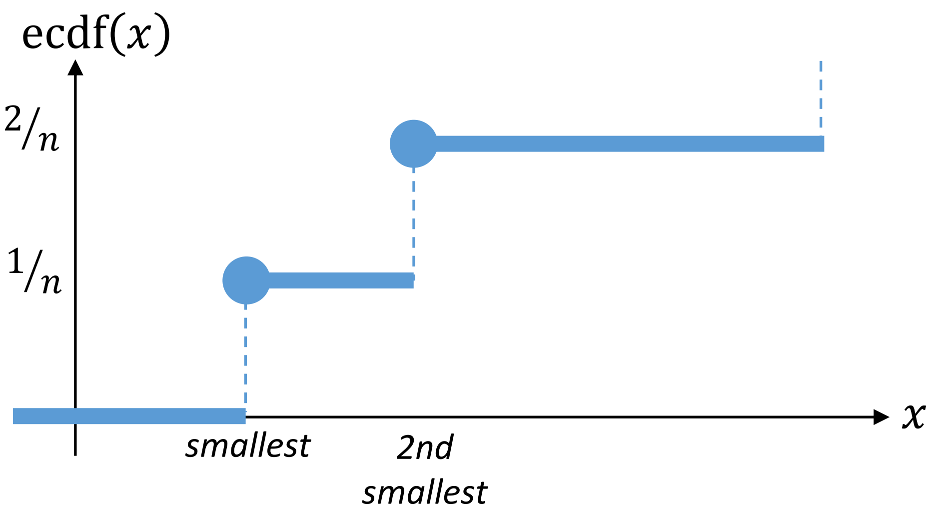



```
def rx(x1,x2,δ):
    k = np.random.choice(["left","right"])
    if k == "left":
        return np.random.uniform(x1 − δ,x1 + δ)
    else:
        return np.random.uniform(x2 − δ,x2 + δ)
```



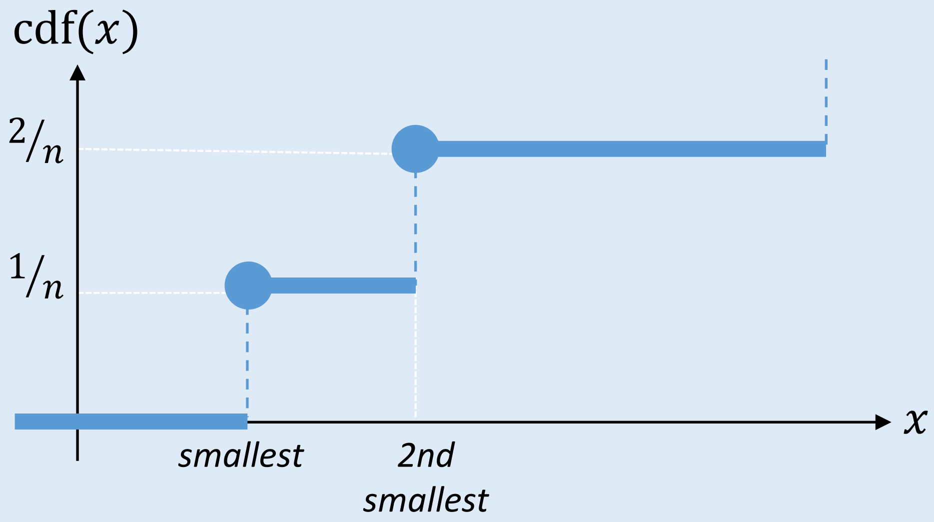

```
def rx(x1,x2):
    k = np.random.choice(["left","right"])
    if k == "left":
        return x1
    else:
        return x2
```

np.random.choice([x_1, x_2])



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 $[x_1, x_2, \dots, x_n]$
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

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

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

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

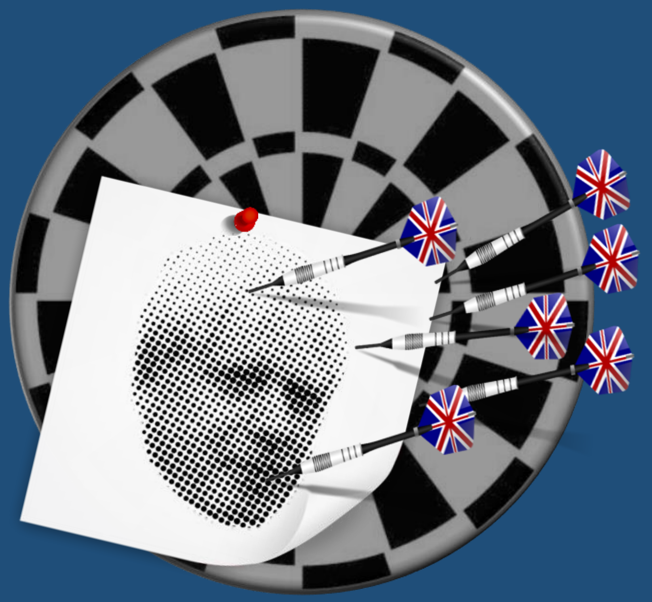
Francis Bacon, 1561–1626

- Empirical modelling
  The empirical distribution is a perfect fit for a dataset. Why bother fitting a parametric probability model?

Monte Carlo

Let $[x_1, \ldots, x_n]$ be sampled from a random variable X.
For any real-valued readout function h,

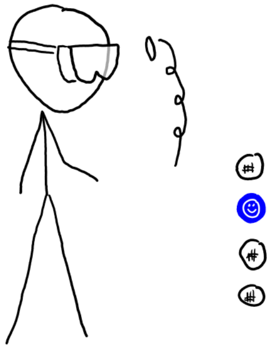
$$\mathbb{E}\, h(X) \approx \frac{1}{n}\sum_{i=1}^{n} h(x_i) = \mathbb{E}\, h(\hat{X})$$



- Empirical calculations
  Don't bother doing maths with a tricky random variable X, just take a sample and use its empirical distribution $\hat{X}$!

The challenge of induction

induction = inferring general truths from finite data



I tossed four coins and got one head.
What is it reasonable to infer about the probability of heads (call it θ)?

- "The maximum likelihood estimator is $\hat{\theta} = 25\%$, thus the true probability of heads is 25%"
  (hence if I tossed millions more coins that's the fraction of heads I'd see)
  unjustified!
- "All we know for certain is that $0 < \theta < 1$"
  logical, but useless!
- Let it be random with prior distribution $\Theta \sim U[0,1]$.
  Then $\mathbb{P}(\Theta \in [3\%, 72\%] \mid \text{data}) = 95\%$
  justifiable, useful, subjective.
- ???

Bayes's rule:

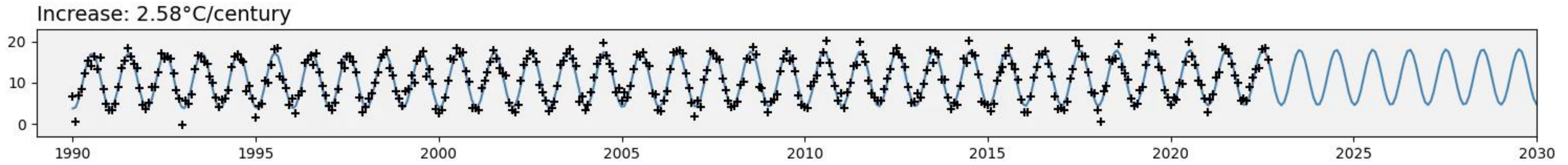
$$\Pr_{\Theta}(\theta|x) = \kappa \Pr_{\Theta}(\theta) \Pr_X(x|\Theta=\theta)$$

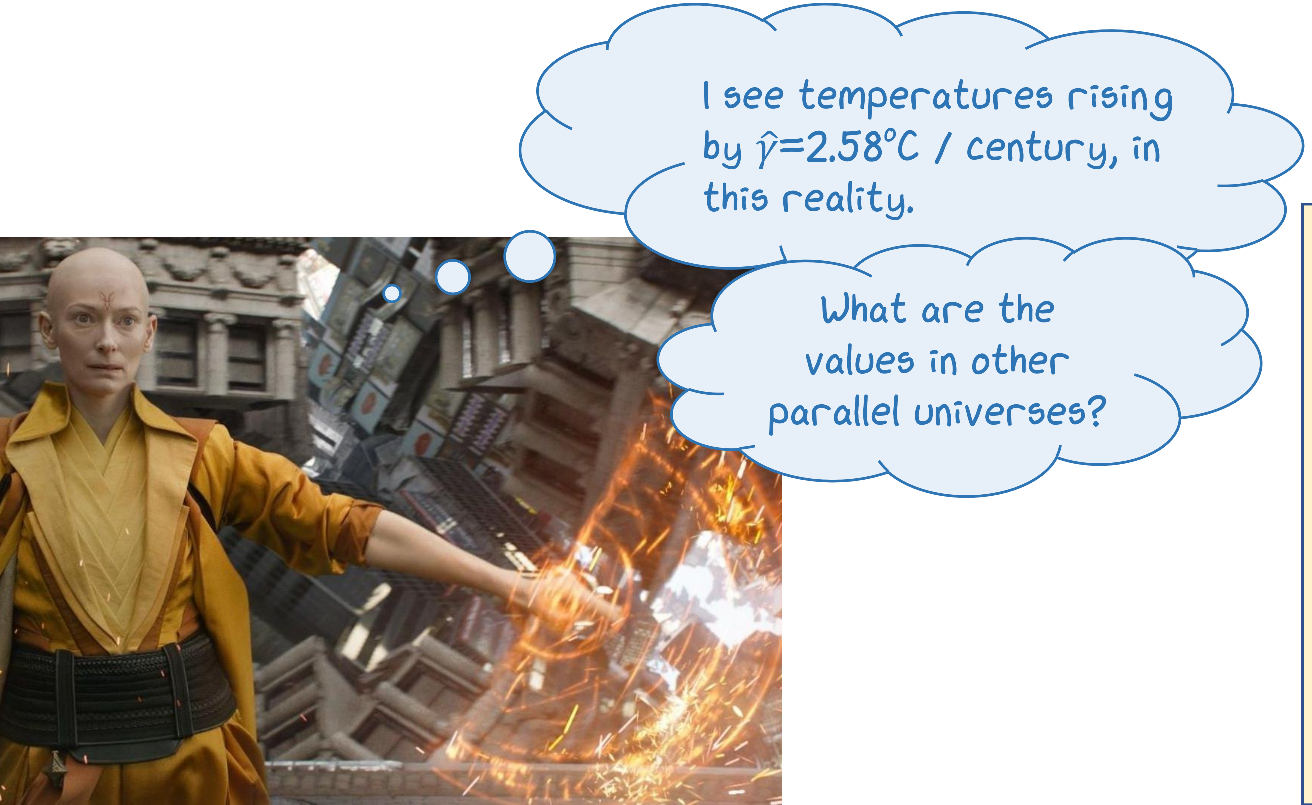
Frequentism

I’m not so bothered about knowing whether $\hat{\theta} \in [\mathrm{lo}, \mathrm{hi}]$ in *this* universe.

I’m interested in the *frequency* with which $\hat{\theta} \in [\mathrm{lo}, \mathrm{hi}]$ across the multiverse.

How might I simulate 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.)

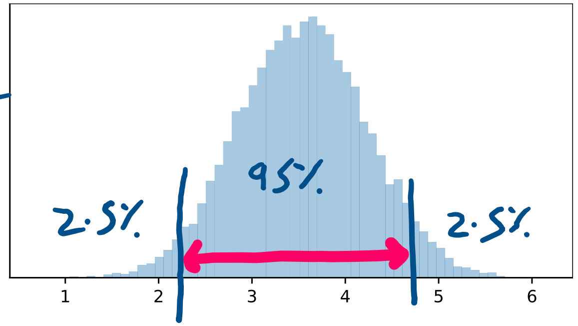
Please submit your answer on Moodle
by Monday 6 November

Confidence intervals via resampling

Given a dataset x,

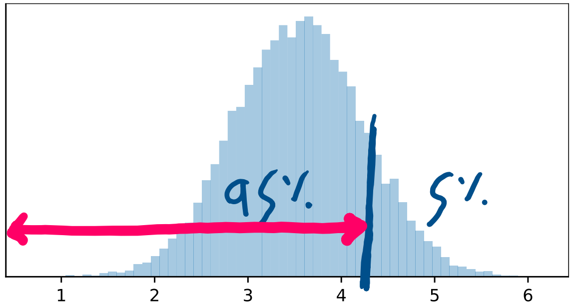
1. Decide on a readout function $t(x)$
2. *"Simulate a multiverse of datasets."*
   - Fit a model for the dataset
   - Let X^* be a random synthetic dataset, generated from the fitted model
   - Simulate many synthetic datasets
3. Compute t for each dataset, and report the spread of t

   for example with a histogram or a confidence interval



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 $\hat{\mu}$.

This problem is over-specified. It might as well just say "Find a 95% confidence interval for the mean of the dataset."

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

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

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

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

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

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 $\hat{\mu}$.

```python
# 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, we can 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 $\Pr(x;\theta)$

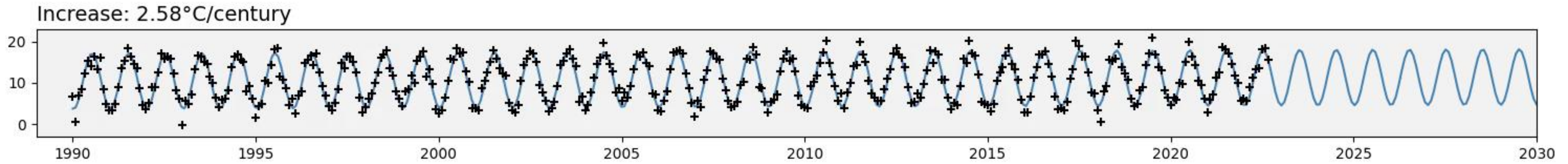
all the data

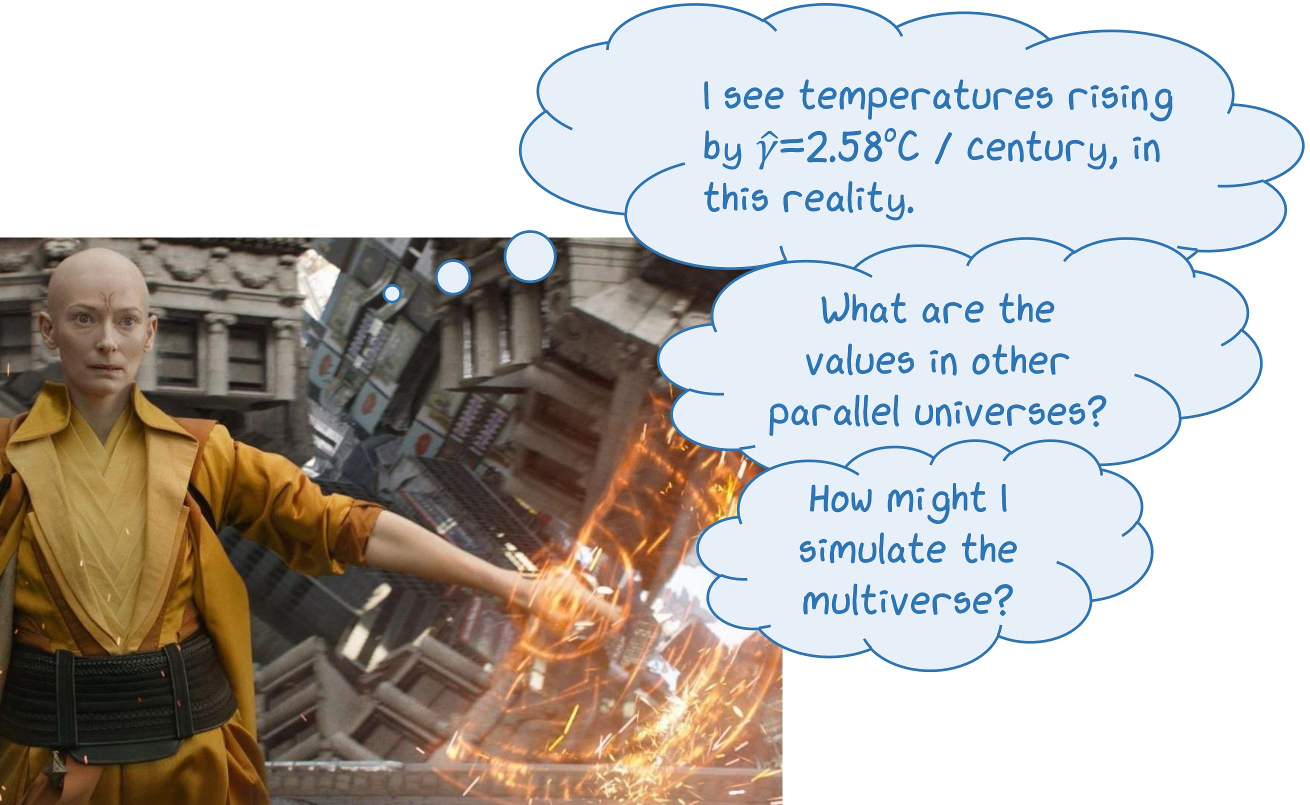
all the parameters

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

Parametric resampling





The model we fitted:

$$\text{Temp}_i \sim \alpha \sin\big(2\pi(t_i + \phi)\big) + c + \gamma t_i + N(0, \sigma^2)$$

Simple way to simulate a new dataset:

Fit $\hat{\alpha}, \hat{c}, \hat{\gamma}, \hat{\sigma}$ from the observed data, then generate n new datapoints Temp_i, $i = 1, \dots, n$, by

$$\text{Temp}_i \sim \hat{\alpha} \sin\left(2\pi\left(t_i + \hat{\phi}\right)\right) + \hat{c} + \hat{\gamma} t_i + N(0, \hat{\sigma}^2)$$

Exercise 9.2.3 (Comparing groups).
We are given data $x = [x_1, \ldots, x_m]$ which we believe is $N(\mu, \sigma^2)$ and further data $y = [y_1, \ldots, y_n]$ 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 I question 5:

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

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

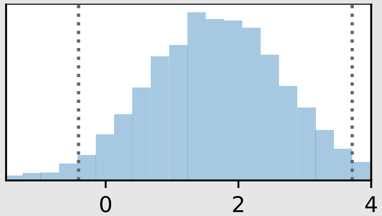
$\hat{\sigma} = \cdots$

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

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


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

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



There is only ever ONE dataset, consisting of ALL the observations.

$$\Pr(x_1, \ldots, x_m, y_1, \ldots, y_n \; ; \; \mu, \delta, \sigma) = \cdots$$

To simulate it, we need to estimate ALL the unknown parameters.

- This resampling approach requires us to simulate the multiverse.
- Which is better, parametric resampling or resampling from the empirical distribution?
- Simulating the multiverse is modelling, not maths. There is no right answer. We just have to invent something we can argue is plausible.
- We can't possibly *deduce* "what might have been" from "what was".