code: Azure notebook

Exercise 2.4 (Confounded features).
The UK Home Office makes available a dataset of police stop-and-search incidents. We wish to investigate whether there is racial bias in police decisions to stop-and-search. Using the binary response vector

$$y = 1[\texttt{outcome} \neq \text{"False"}]$$

and the linear model

$$y \approx \alpha + \beta_{\texttt{eth}}$$

analyse whether there is racial bias in policing actions.

$$y_i = \begin{cases} 1 & \text{if record } i \text{ has } \texttt{outcome} \neq \text{"False"}\text{, i.e. if the police find something} \\ 0 & \texttt{outcome} = \text{"False"} \qquad \text{nothing} \end{cases}$$

$$\text{mean of } y = \frac{\#\text{records where } y_i = 1}{\text{total } \#\text{records}} = \frac{\#\text{stops where police find sthg}}{\#\text{stops}} = \mathbb{P}(\text{police find sthg})$$

This model was designed so that its predicted responses correspond to $\mathbb{P}(\text{police find sthg})$.
If this probability is different for different values of $\texttt{eth}$, it suggests racial bias.

Fitting the linear model $y \approx \alpha + \beta_{\texttt{eth}}$ using one-hot coding to extract the β coefficients:

code: Azure notebook

```
ethnicity_levels = numpy.unique(df['eth'])
eth_onehot = [df['eth']==i for i in ethnicity_levels]

model = sklearn.linear_model.LinearRegression()
model.fit(numpy.column_stack(eth_onehot), df['y'])
α,βs = model.intercept_, model.coef_

print(f'α = {α}')
for i,β in zip(ethnicity_levels, βs):
    print(f'β[{i}] = {β}')
```

```
α = 171497104.17042407
β[Asian] = -171497103.3887985
β[Black] = -171497103.40795302
β[Mixed] = -171497103.2539072
β[Other] = -171497103.40197015
β[White] = -171497103.41569293
```

The Guardian

Scientists report clear evidence of ...
Something must be done.

Because the features for this model are linearly dependent ("confounded"), the fitted coefficients are an arbitrary choice. They may differ from one library to another, one version to another.

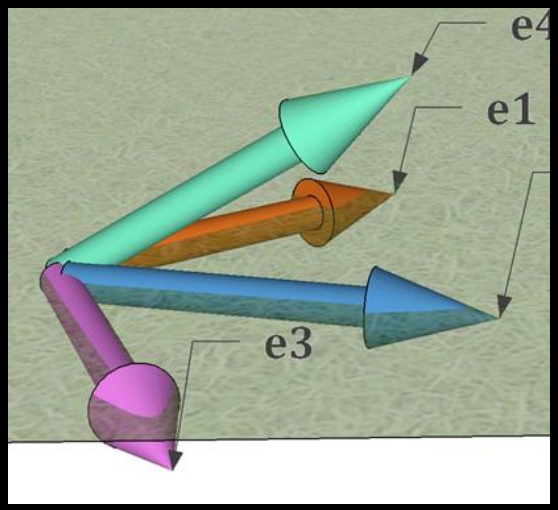
Daily Mail

WEDNESDAY, NOVEMBER 28, 2018 www.dailymail.co.uk

Doubt thrown on earlier claims.
Can we trust the scientists?

To interpret coefficients, (1) select linearly independent features, (2) write out predictions for generic individuals.

It's up to us which features to use.



Our first model, with linearly dependent features:

$$y \approx \alpha\, 1 + \beta_{\text{As}} e_{\text{As}} + \beta_{\text{Bl}}\, e_{\text{Bl}} + \beta_{\text{Mi}}\, e_{\text{Mi}} + \beta_{\text{Oth}}\, e_{\text{Oth}} + \beta_{\text{Wh}}\, e_{\text{Wh}}$$

Revised model, with linearly independent features that span the same featurespace:

$$y \approx \alpha\, 1 + \beta_{\text{As}} e_{\text{As}} + \beta_{\text{Bl}}\, e_{\text{Bl}} + \beta_{\text{Mi}}\, e_{\text{Mi}} + \beta_{\text{Oth}}\, e_{\text{Oth}}$$

Write out predictions for generic individuals:

For a person with eth = "Asian" predict $y = \alpha + \beta_{As}$

"Black" $y = \alpha + \beta_{Bl}$

⋮

"white" $y = \alpha$

```
want_levels = ['Asian','Black','Mixed','Other']
eth_onehot = [df['eth']==i for i in want_levels]

model = sklearn.linear_model.LinearRegression()
model.fit(numpy.column_stack(eth_onehot), df['y'])
α,βs = model.intercept_, model.coef_

print(f'α = {α:.3}')
for i,β in zip(want_levels, βs):
    print(f'β[{i}] = {β:.3}')
```

```
α = 0.7547
β[Asian] = 0.02745
β[Black] = 0.00775
β[Mixed] = 0.1625
β[Other] = 0.0135
```

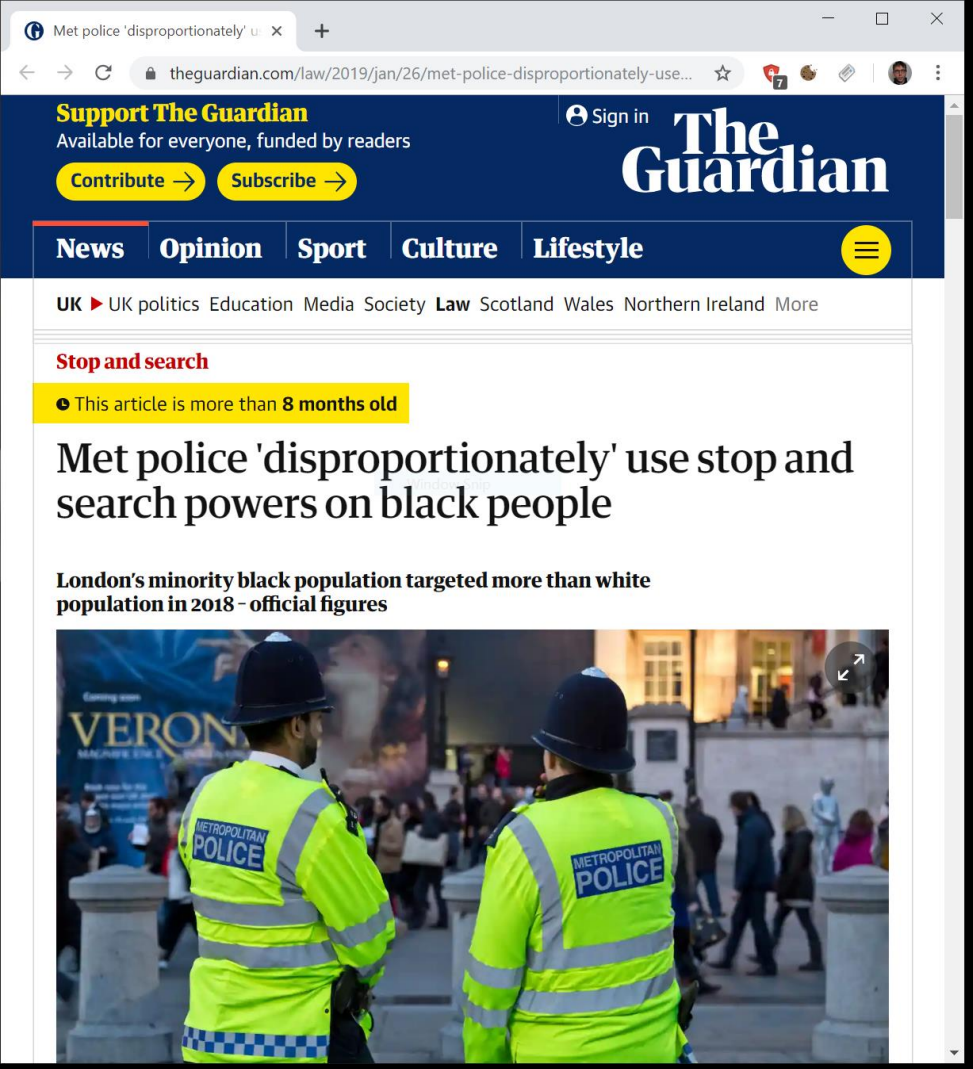
predicted y (eth = "white")

predicted y (eth = "Asian") is 0.02745 larger than predicted y (eth = "white").

code: Azure notebook

The predicted y value, i.e. our estimated probability that the police find something in a stop-and-search, is 2.7 %points higher for people with `eth="Asian"` than for `eth="White"`.
So the police are stopping more *innocent* people with `eth="White"` than `eth="Asian"`.

— Or, they're stopping just as many guilty people in each class, but letting more `eth="White"` off the hook.



Vikram Dodd *Police and crime correspondent*

Sat 26 Jan 2019 06.00 GMT

The Metropolitan police increased its use of stop and search last year, with a 19% rise among London's minority black population, which was targeted more than the white population, official figures show.

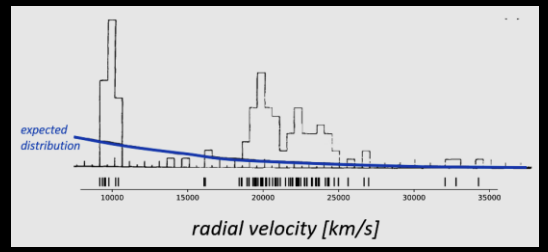
Analysis commissioned by the Guardian also shows that searches of black people were less likely to detect crime than those conducted on white people, and most stops found no wrongdoing.

Black people make up 15.6% of London's population while white people make up 59.8%. In 2018, 43% of searches were of black people, while 35.5% were of white people , according to official figures from the London Mayor's Office for Policing and Crime (MOPAC).

Practical 2

Reproduce the Guardian's analysis using a linear model.

Explain why their findings differ from ours.

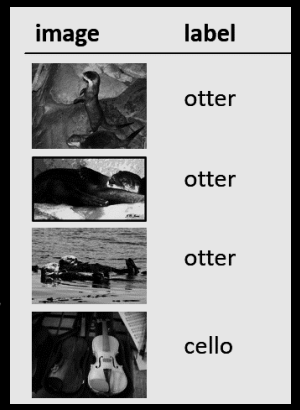


clustering of galaxies (lecture 2) five parameters

science = finding patterns in nature

- meaningful parameters
- interpretable patterns

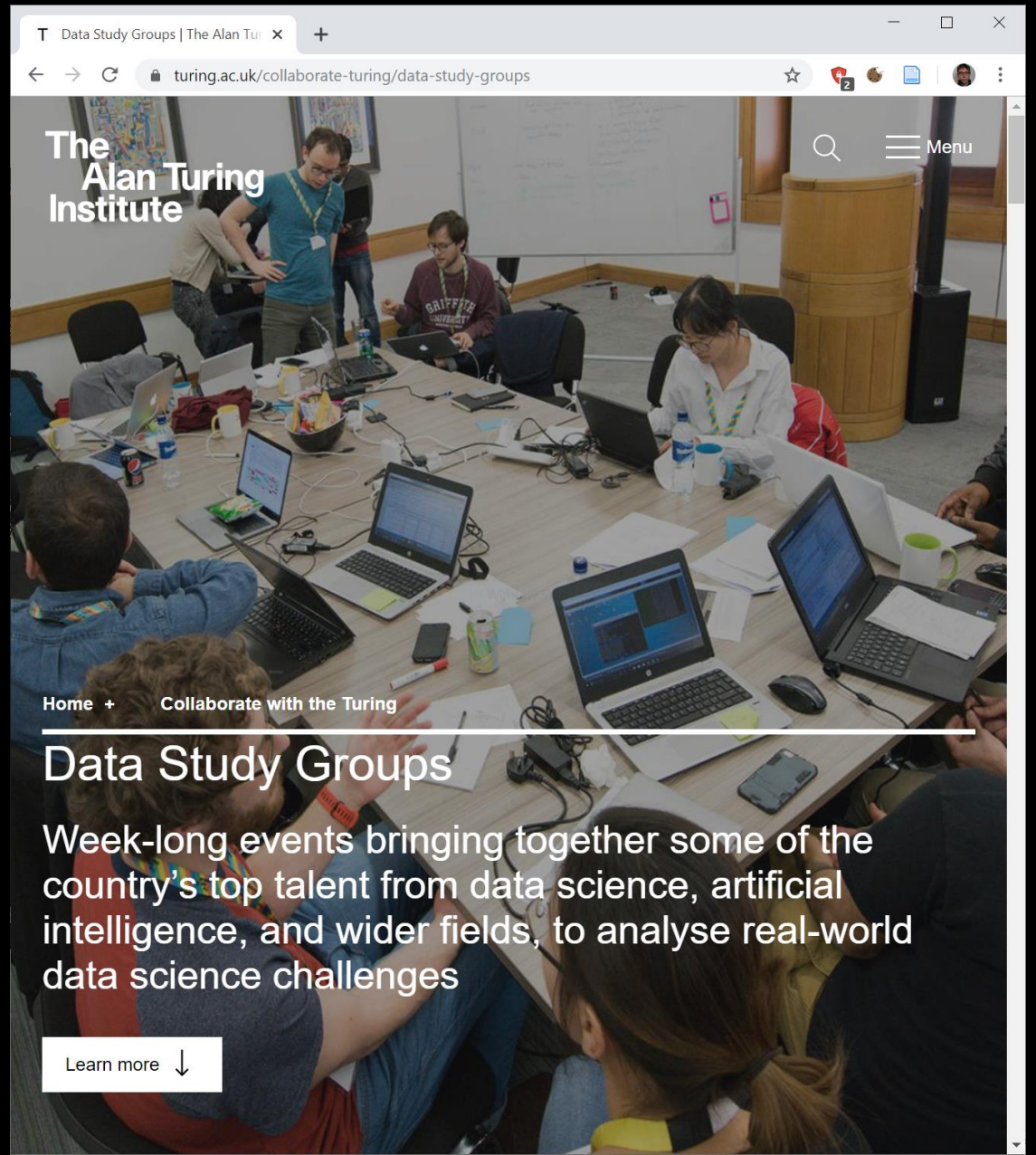
image classification (lecture 3) using a black-box function with billions of parameters



modern machine learning = finding patterns in nature/data

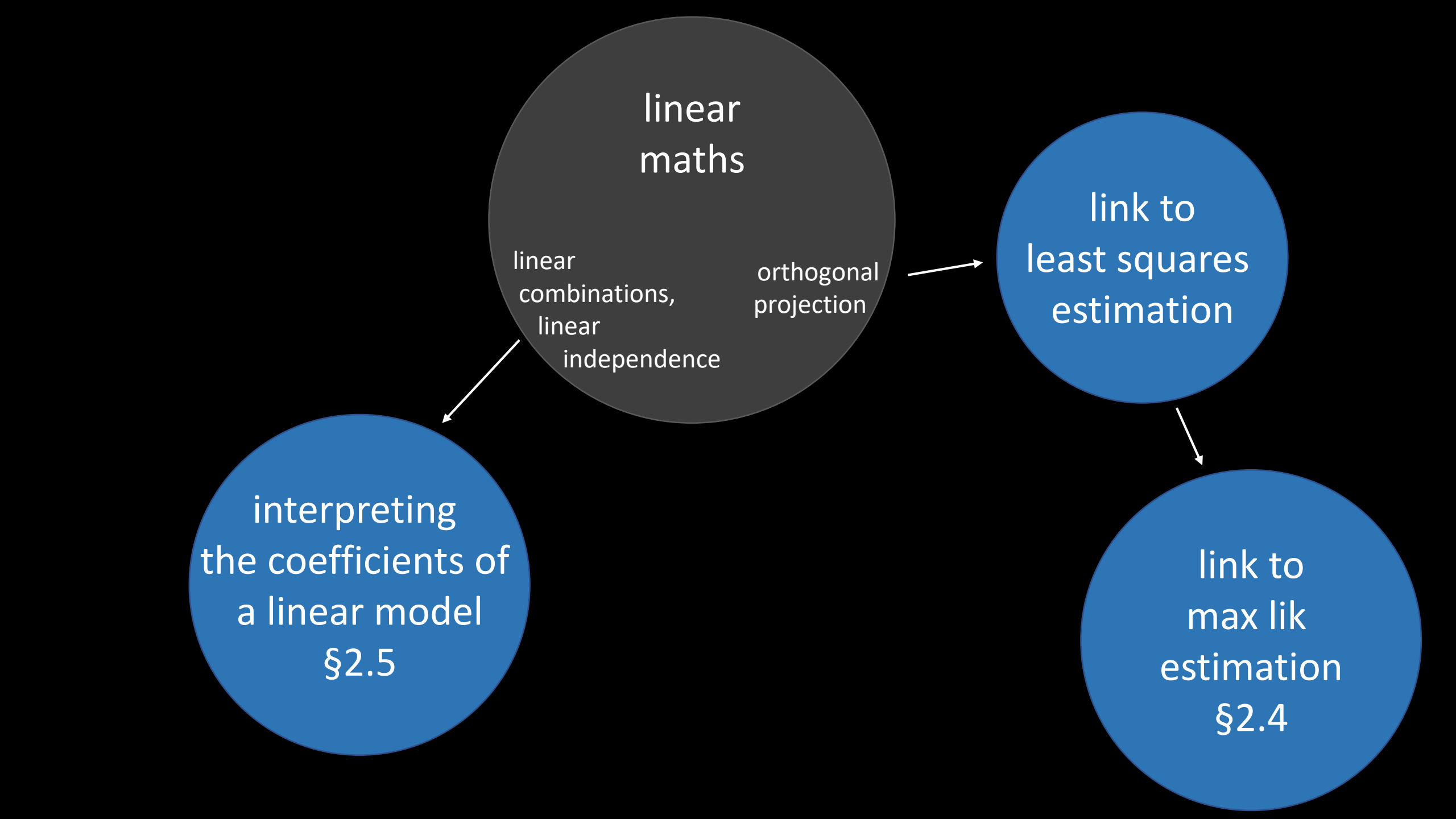
- training a neural net = maximum likelihood estimation
- billions of parameters
- patterns beyond our comprehension

Scientists and social scientists have so far rejected neural networks. It's a crisis for science, that the models that have the potential to work best—these models are uninterpretable hence unreliable.



If you're thinking of internships in data science, and want some experience to put in your application letter, consider a Data Study Group at the Alan Turing Institute.

The next is in December, and applications are open.

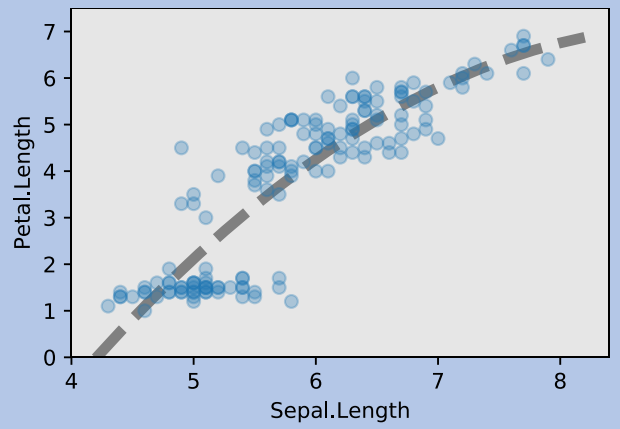

linear maths
linear combinations, linear independence
orthogonal projection
link to least squares estimation
interpreting the coefficients of a linear model §2.5
link to max lik estimation §2.4

A *linear model* is a model of the form

$$y = \beta_1 e_1 + \cdots + \beta_K e_K + \varepsilon$$

- y is the response vector $[y_1, y_2, \ldots, y_n]$
- $e_1, \ldots, e_K$ are features, each a vector of length n
- $\varepsilon = [\varepsilon_1, \ldots, \varepsilon_n]$ is the *residuals* vector / error / noise
- After fitting the model the *predicted values* are

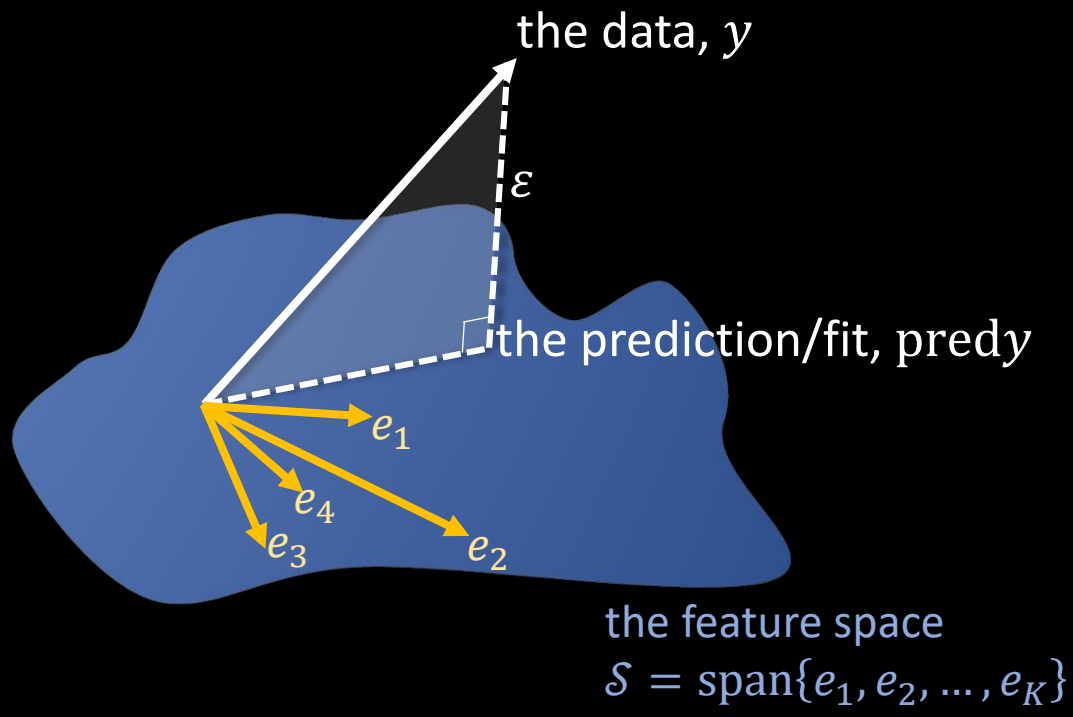
$$\text{pred}y = \hat{\beta}_1 e_1 + \cdots + \hat{\beta}_K e_K$$



A sensible way to fit a linear model is to pick the β coefficients so as to minimize the *mean square error*

$$\text{mse} = \frac{1}{n}\sum_{i=1}^{n} \varepsilon_i^2$$

$$= \frac{1}{n}\|y - \text{pred}y\|^2$$



2.4 Linear regression and least squares

Consider the probability model

$$\text{Petal.Length}_i = \alpha + \beta\ \text{Sepal.Length}_i + \gamma(\text{Sepal.Length}_i)^2 + \text{Normal}(0, \sigma^2)$$

$$Y_i = \alpha + \beta e_i + \gamma f_i + N(0, \sigma^2)$$

$$Y_i \sim N(\alpha + \beta e_i + \gamma f_i,\ \sigma^2)$$

Supervised Learning setup:
Y_i is the random response/label
e_i, f_i are non-random predictors

(the algebra of the Normal)

let's find max. lik. estimators for $\alpha, \beta, \gamma, \sigma$.

$$\text{loglik}(\alpha, \beta, \gamma, \sigma \mid y_1, \cdots y_n) = -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum [y_i - (\alpha + \beta e_i + \gamma f_i)]^2$$

To maximize this,

1. to find $\hat{\alpha}, \hat{\beta}, \hat{\gamma}$: minimize $\sum [y_i - (\alpha + \beta e_i + \gamma f_i)]^2$
2. find $\hat{\sigma}$ by $\frac{\partial}{\partial \sigma}\text{loglik} = 0 \Rightarrow \hat{\sigma} = \cdots$

i.e. solve Least Squares estimation.

Maximum likelihood estimation for a Gaussian regression model is *the same thing* as least squares estimation

(except that the probability model also reports $\hat{\sigma}$).

- If you don't believe the random variables are Gaussian, don't use least squares estimation.

- All the ideas about linear independence and interpretation of coefficients carry across to Gaussian regression.

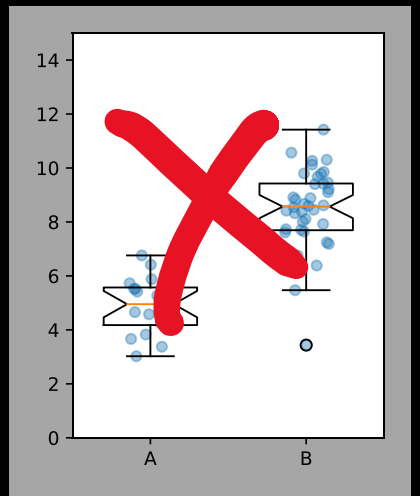
Carl Friedrich Gauss, the prince of mathematicians

1. In many situations, random quantities can be approximated by Normal random variables
2. Likelihood is nothing more than probability, so maximum likelihood estimation is inherently reasonable
3. Maximum likelihood estimation for Normal random variables is equivalent to least squares estimation
4. Thus least squares estimation is sensible, not just an heuristic kludge.

- There's no reason to believe linear models are *true*—
  but they're useful because they are flexible and interpretable.
- If the data suggests that a Gaussian model is inappropriate,
  don't do least squares estimation.

- George Box, one of the greatest statistical minds of the 20th century, wrote that

  "All models are wrong—but some are useful"



- Box didn't invent the Box plot—
  Mary Eleanor Spear did.

Part II
Handling probability models

3.6. Bayes's rule for random variables

Exercise 3.15 (Bayes's rule for binary outcomes).
A screening test is 99% effective in detecting a certain disease when applied to a person who has the disease. The test yields a 'false positive' for 0.5% of healthy persons tested. Suppose 0.2% of the population has the disease. What is the probability that a person whose test is positive has the disease?

Let $X \in \{\text{healthy}, \text{sick}\}$ be a person's true health.

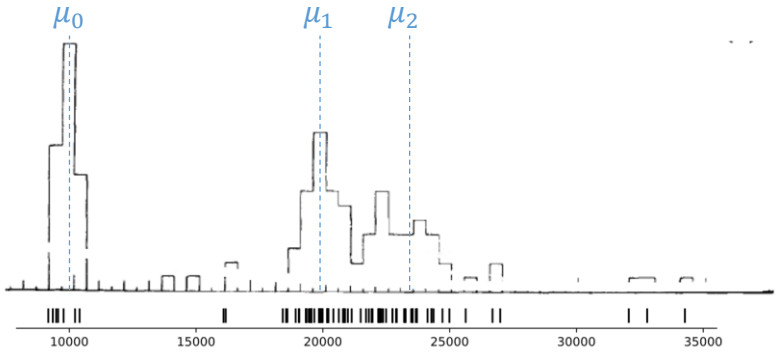
Let $Y \in \{\text{pos}, \text{neg}\}$ be the test outcome.

Then Bayes's rule says

$$\mathbb{P}(X = \text{sick} \mid Y = \text{pos}) = \frac{\mathbb{P}(X = \text{sick})\, \mathbb{P}(Y = \text{pos} | X = \text{sick})}{\mathbb{P}(Y = \text{pos})}$$

(Bayes's rule requires $\mathbb{P}(Y = \text{pos}) > 0$, which is true for this question.)

How does Bayes's rule apply to continuous random variables?



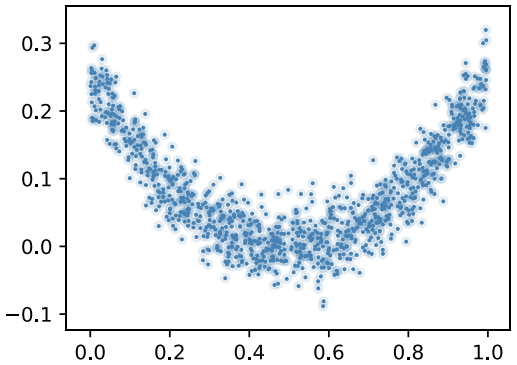
For the galaxies dataset from lecture 2, we proposed a Gaussian mixture model:

```
# parameters p = (p0,p1,p2), μ = (μ0,μ1,μ2), σ = (σ0,σ1,σ2)
k = numpy.random.choice([0,1,2], p=p)
x = numpy.random.normal(loc=μ[k], scale=σ[k])
```

For a galaxy with x=22000, which cluster does it belong to?

$$\mathbb{P}(K=1|X=22000) = \frac{\mathbb{P}(K=1)\,\mathbb{P}(X=22000|K=1)}{\mathbb{P}(X=22000)} = \frac{p_1 \times 0}{0}$$???

because X is a continuous r.v.



What about this pair of continuous random variables?

```
x = numpy.random.uniform(0,1)
y = numpy.random.normal(loc=x*(x-1)+1/4, scale=.03)
```

They're not independent. If I'm told y=0.2, what can I deduce about x?

$$\mathbb{P}(X=x|Y=0.2) = \frac{\mathbb{P}(X=x)\,\mathbb{P}(X=x|Y=0.2)}{\mathbb{P}(Y=0.2)}$$

$0 = \frac{0 \times 0}{0}$ because X, Y both cts

Bayes's rule. For two random variables X and Y,

$$\Pr_X(x|Y=y) = \frac{\Pr_X(x)\Pr_Y(y|X=x)}{\Pr_Y(y)} \quad \text{when } \Pr_Y(y) > 0$$

For this course, you need to

- ~~be able to derive this and related equations~~
- ~~be able to define the terms in these equations~~
- understand intuitively what the terms mean
- not be flummoxed by notation overload
- be able to translate such equations into code
- be able to do some calculations based on such equations

Bayes's rule. For two random variables X and Y,

$$\Pr_X(x|Y=y) = \frac{\Pr_X(x)\Pr_Y(y|X=x)}{\Pr_Y(y)} \quad \text{when } \Pr_Y(y) > 0$$

For a discrete random variable, $\Pr_X(x)$ is defined to be $\mathbb{P}(X=x)$.

For a continuous random variable, $\Pr_X(x)$ is defined to be $\text{pdf}(x)$.

We haven't yet defined $\Pr_X(x|\dots)$

- When X and Y are discrete, this equation is equivalent to the familiar version of Bayes's rule

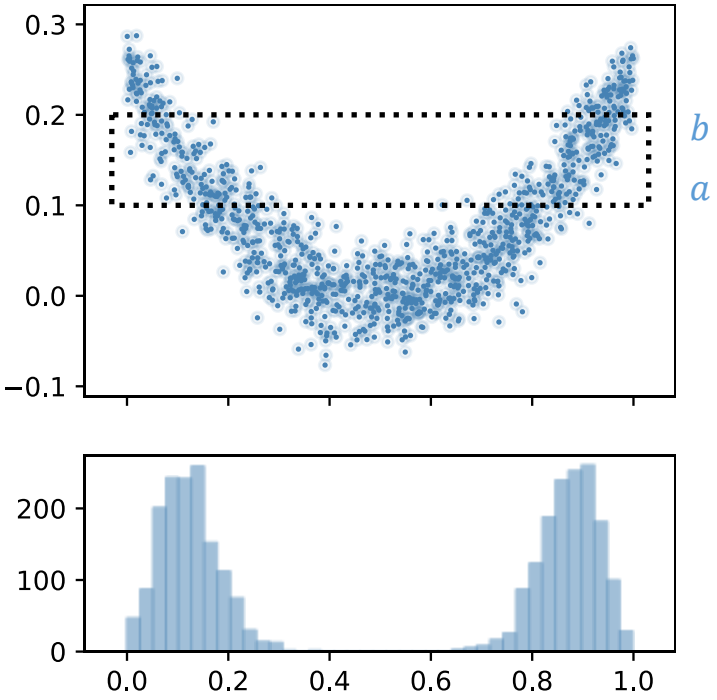
$$\mathbb{P}(X=x \mid Y=y) = \frac{\mathbb{P}(X=x)\,\mathbb{P}(Y=y|X=x)}{\mathbb{P}(Y=y)} \quad \text{when } \mathbb{P}(Y=y) > 0$$

- When X or Y is continuous, it gives sensible answers (plus, it can be proved!)

But what is $\Pr_X(x|Y=y)$ *even meant to be?*
What does it mean to condition on an event with probability zero?

3.5. Conditional random variables

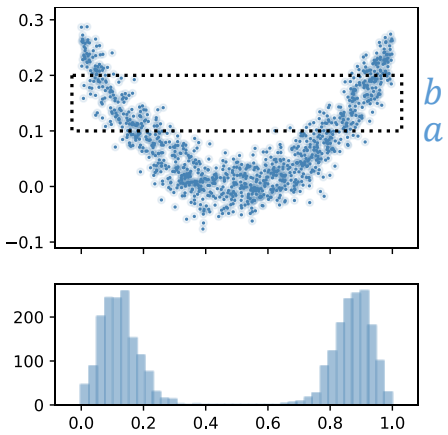
For a random variable X and an event C, we write $(X|C)$ for X *conditioned on* C.



For example, here's code to simulate $(X|Y \in [a,b])$

```
def rxy():
    x = numpy.random.uniform(0,1)
    y = numpy.random.normal(loc=x*(x-1)+1/4, scale=.03)
    return (x,y)

def rx_given_yrange(a,b):
    while True:
        x,y = rxy()
        if y>=a and y<=b:
            break
    return x
```



For example, here's code to simulate $(X|Y \in [a,b])$

```
def rxy():
    x = numpy.random.uniform(0,1)
    y = numpy.random.normal(loc=x*(x-1)+1/4, scale=.03)
    return (x,y)

def rx_given_yrange(a,b):
    while True:
        x,y = rxy()
        if y>=a and y<=b:
            break
    return x
```

We've given code to generate $(X|Y \in [a,b])$.

Our function `rx_given_yrange(a,b)` is a random number generator, i.e. a random variable: it produces a new output each time we invoke it.

What's the sample space of this random variable? What's its density?

Sample space of $(X \mid Y \in [a,b])$?

rxy() generates a pair of values in the sample space $[0,1] \times \mathbb{R}$.

rx_given_yrange(a,b) generates a value in the sample space $[0,1]$.

To work out its density, we need some notation....

The density of a conditional random variable

1. Given a random variable X and an event C, we can define a function
$$F(x) = \mathbb{P}(X \leq x \mid C).$$
This is of course not the same thing as the cdf of X, which is
$$G(x) = \mathbb{P}(X \leq x).$$

Clearly $F(x)$ is an increasing function, with range $[0,1]$

2. Given a function $F(x)$, if it's increasing and has range $[0,1]$, we can in principle create a random variable from it. In other words, we can design a random variable Z such that
$$\mathbb{P}(Z \leq x) = F(x)$$

See Example sheet 1 question 2: which gave you F and asked you for code

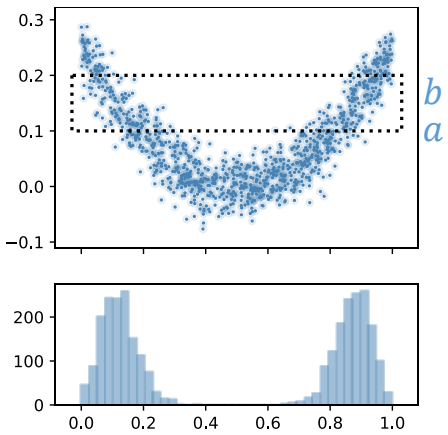
3. This is what $(X|C)$ is defined to be.

4. If F is differentiable, then $(X|C)$ is a continuous random variable, with density
$$\Pr_{(X|C)}(x) = F'(x)$$

5. As usual,
$$\mathbb{P}\big((X|C) \in A\big) = \int_{x \in A} \Pr_{(X|C)}(x)\, dx$$

more commonly written $\Pr_X(x \mid C)$.

Read it as "the density of X conditioned on C".



For example, here's code to simulate $(X|Y \in [a,b])$

```
def rxy():
    x = numpy.random.uniform(0,1)
    y = numpy.random.normal(loc=x*(x-1)+1/4, scale=.03)
    return (x,y)

def rx_given_yrange(a,b):
    while True:
        x,y = rxy()
        if y>=a and y<=b:
            break
    return x
```

We've given code to generate the random variable $(X|Y \in [a,b])$.

And now we know what is meant by its density

$$\Pr_X(x|Y \in [a,b])$$

If I want $(X \mid Y = 0.2)$ the code is no use.

And what is $F(x) = \mathbb{P}(X \le x \mid Y = 0.2)$?

How can we condition on an event with probability zero?