IB Foundations of Data Science

Damon Wischik, Computer Laboratory, Cambridge University

Michaelmas Term 2017

- There will be 12 lectures (held in Lecture Theatre 1 at 11am). The timetabling is erratic; the calendar below may help.
- There will be three example sheets, combining pen-and-paper work with practical work. The material covered in the practicals may be tested in the exam, but the practicals themselves won't be graded. Example sheets will be handed out in advance of the material they cover.
- There will be four practical help sessions (held in the Intel Laboratory at 11am). These are optional. You can ask your college supervisor for any help you need, or you can come to the help sessions, or you can do both.

	Mon	Tue	Wed	Thu	Fri
Oct				5	6 FDS Ex.0
	9 FDS Ex.1	10	11 FDS	12	13 FDS
	16 FDS	17	18 prac	19	20 FDS Ex.2
	23 FDS	24	25 FDS	26	27 prac
Nov	30 prac	31	1 FDS Ex.3	2	3 FDS
	6	7	8	9	10 FDS
	13	14	15	16	17 FDS
	20	21	22	23	24 prac
	27	28	29		

0. What is data science?

The Harvard Business Review called data science "the sexiest job of the 21st century"[1], and the Economist says "The world's most valuable resource is no longer oil, but data"[2]. So it is no surprise that *data science* is a label that many people have seized upon, to mean many different things. Here is my attempt at a definition:

> *Data science is any field of study where you can be surprised by data.*

What do you think data science is? Note down your answer now, and again at the end of the course.

0.1. 'Surprised by data': reasoning about uncertainty

Here are some results from a survey of undergraduates, 15 female, 94 male, 4 other / didn't specify.

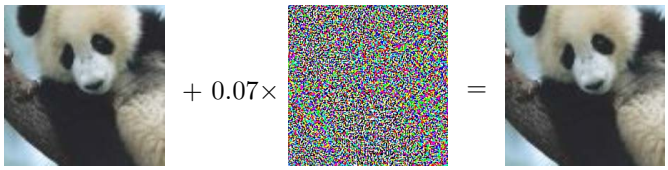
		F	M	X
I am treated fairly in lectures	yes	14	86	·
	no	1	8	·
I'm comfortable asking questions	str.agree	2	36	·
	str.disagree	10	36	·

Simple percentages say that female students are more likely to say they're treated fairly in lectures than male students, and yet they're less comfortable asking questions, which seems surprising—but when we look at actual numbers rather than percentages we intuit that the numbers are so small we might expect some mixed signals. Is this intuition correct? In other words, what counts as surprising, and what counts as chance variation?

Data science is about fields of study where you can be surprised by data. To say what's suprising, we need to be able to *reason about uncertainty*. Here is a quote from an astronomer, about possible detection of an exomoon, which illustrates that reasoning about uncertainty does not come naturally:

> *The work by Dr Kipping, his Columbia colleague Alex Teachey and citizen scientist Allan R Schmitt, assigns a confidence level of four sigma to the signal from the distant planetary system. The confidence level describes how unlikely it is that an experimental result is simply down to chance. If you express it in terms of tossing a coin, it's equivalent to tossing 15 heads in row.*
>
> *But Dr Kipping said this is not the best way to gauge the potential detection. He told BBC News: "We're excited about it... statistically, formally, it's a very high probability. But do we really trust the statistics? That's something unquantifiable. Until we get the measurements from Hubble, it may as well be 50–50 in my mind."*[3]

Today's neural networks for image classification also have trouble reasoning about uncertainty, as this adversarial panda illustrates:[4]



`panda`, 57.7% confidence | `nematode`, 8.2% confidence | `gibbon`, 99.3% confidence

The main goal of this course is to learn how to express uncertainty in equations and computer programs, so that 'what you know in your mind' and 'what your probability computations say' support each other.

---

[1] `https://hbr.org/2012/10/data-scientist-the-sexiest-job-of-the-21st-century`

[2] `https://www.economist.com/news/leaders/21721656-data-economy-demands-new-approach-antitrust-rules-worlds-most-valuable-resource`

[3] `http://www.bbc.co.uk/news/science-environment-40741545`

[4] I. J. Goodfellow, J. Shlens, and C. Szegedy. "Explaining and Harnessing Adversarial Examples". In: *ArXiv e-prints* (Dec. 2014). arXiv: `1412.6572 [stat.ML]`.

0.2. 'Field of study': scientific modeling

Given a crime and policing dataset[5], here are some questions that spring to mind:

1. What is the typical crime rate in each neighbourhood?
   Keywords: description, estimation.
2. How many police officers should we allocate to each neighbourhood this week?
   Keywords: prediction accuracy, working system.
3. Do our policing strategies exacerbate racial tension?
   Keywords: science, hypothesis, policy, counterfactual.

You'll sometimes see *machine learning* used to mean "an operational engineering discipline, for designing algorithms that predict outputs when fed with inputs, and are evaluated according to the accuracy of their predictions". This is the spirit, for example, of competitions at `kaggle.com`, and it has led to some remarkably clever algorithms. It also leads to offensive mistakes:

> *Google came under fire this week after its new Photos app categorized photos in one of the most racist ways possible. On June 28th [2015], computer programmer Jacky Alciné found that the feature kept tagging pictures of him and his girlfriend as "gorillas." He tweeted at Google asking what kind of sample images the company had used that would allow such a terrible mistake to happen.*
>
> *Google's chief social architect Yonatan Zunger responded quickly, apologizing for the feature.*[6]

A machine learning system isn't value neutral, it's a reflection of the choices that went into the training dataset. Data investigation is a required skill for responsible machine learning. Conversely, whenever you analyse data, the tools you use to test your hypotheses are estimation and prediction. So 'data science' and 'machine learning' are complementary and intertwined.

This course is about data *science*, i.e. about building models and hypotheses and understanding, using data. It's *not* an introduction to the machine learning toolbox. The range of algorithms out there is exciting and growing rapidly. There are courses in Part II and III that will introduce you to some of them, and there are others that you will pick up in your own reading. The goal of this course is to give you practice at asking the right questions and finding the right tools, so that when you read about a new algorithm you can quickly pick it up and decide where it is and isn't appropriate.

What is data science modeling?

> *All models are wrong but some are useful [...] there is no need to ask the question "Is the model true?". If "truth" is to be the "whole truth" the answer must be "No". The only question of interest is "Is the model illuminating and useful?".*[7]

> *Since no model is to be believed in, no optimization for a single model can offer more than distant guidance. What is needed, and is never more than approximately at hand, is guidance about what to do in a sequence of ever more realistic situations. The analyst of data is lucky if he has some insight into a few terms of this sequence, particularly those not yet mathematized. [...] The main tasks of pictures are then: to reveal the unexpected, to make the complex easier to perceive. Either may be effective for that which is important above all: suggesting the next step in analysis, or offering the next insight. In doing either of these there is much room for mathematics and novelty.*[8]

[5] e.g. `https://data.police.uk/data/`

[6] `https://www.theverge.com/2015/7/1/8880363/google-apologizes-photos-app-tags-two-black-people-gorillas`

[7] G. E. P. Box. "Robustness in the Strategy of Scientific Model Building". In: *Robustness in Statistics*. Vol. 1. May 1979, p. 40. URL: `http://www.dtic.mil/docs/citations/ADA070213`.

[8] John W Tukey. "Mathematics and the picturing of data". In: *Proceedings of the international congress of mathematicians*. Vol. 2. 1975, pp. 523–531. URL: `http://www.mathunion.org/ICM/ICM1974.2/Main/icm1974.2.0523.0532.ocr.pdf`.

> *You've got to have models in your head. And you've got to array your experience—both vicarious and direct—on this latticework of models. You may have noticed students who just try to remember and pound back what is remembered. Well, they fail in school and in life. You've got to hang experience on a latticework of models in your head.*
>
> *What are the models? Well, the first rule is that you've got to have multiple models— because if you just have one or two that you're using, the nature of human psychology is such that you'll torture reality so that it fits your models, or at least you'll think it does.*[9]

0.3. The foundations

The foundations of data science are probability, computing, and statistics.

You can learn enough probability theory in a term, though it takes practice practice practice. You'll pick up the relevant computing skills over the course of your degree: you need to be able to think algorithmically and write fast code, and to understand databases and distributed systems for big data. You can pick up some statistics ideas in a term, but to really understand what it means to learn from data you will need a lifetime of experience, including either a stint in a startup or a degree in philosophy.

Warren Buffet's business partner says in colourful language how important it is to get practice at working with probability:

> *If you don't get this elementary, but mildly unnatural, mathematics of elementary probability into your repertoire, then you go through a long life like a one-legged man in an ass kicking contest. You're giving a huge advantage to everybody else. One of the advantages of a fellow like Buffett, whom I've worked with all these years, is that he automatically thinks in terms of decision trees and the elementary math of permutations and combinations.*[10]

This course assumes you know the basic rules for manipulating probability, such as Bayes's rule. We'll use probability for modelling, and we'll ask what we can learn from data via our models. Here's a taster, the naïve Bayes classifier. Suppose we have the data

Emails labeled `spam`: "buy this viagra", "cheap online pharma", "cheap viagra today".
Emails labeled `genuine`: "will you buy the present or will I", "I will buy it online today".
Test email 1: "buy viagra today".
Test email 2: "buy viagra as a present".

Here's a simple model to start with: Each word in an email is chosen independently, with a probability that depends on the label of the email. In mathematical notation, let θ_w be the probability of word w in spam emails, and ϕ_w be the probability in genuine emails, and let

$$\mathbb{P}(\text{words } w_1 w_2 \dots w_n | \texttt{spam}) = \prod_i \theta_{w_i},$$

$$\mathbb{P}(\text{words } w_1 w_2 \dots w_n | \texttt{genuine}) = \prod_i \phi_{w_i}.$$

Based on the labeled data, the obvious parameter estimates are

w	buy	this	viagra	cheap	online	pharma	today	will	you	the	present	or	I	it
θ_w	1/9	1/9	2/9	2/9	1/9	1/9	1/9	0	0	0	0	0	0	0
ϕ_w	2/14	0	0	0	1/14	0	1/14	3/14	1/14	1/14	1/14	1/14	2/14	1/14

According to Bayes's rule,

$$\mathbb{P}(\texttt{spam}|\text{words}) = \frac{\mathbb{P}(\text{words}|\texttt{spam})\mathbb{P}(\texttt{spam})}{\mathbb{P}(\text{words}|\texttt{spam})\mathbb{P}(\texttt{spam}) + \mathbb{P}(\text{words}|\texttt{genuine})\mathbb{P}(\texttt{genuine})}$$

[9] Charles Munger. *A lesson on elementary, worldly wisdom as it relates to investment management & business.* Speech given at USC Business School. 1994. URL: `http://www.safalniveshak.com/wp-content/uploads/2012/08/Lesson-on-Elementary-Worldly-Wisdom-Charlie-Munger.pdf`.

[10] Ibid.

where $\mathbb{P}(\texttt{spam})$ and $\mathbb{P}(\texttt{genuine})$ are prior beliefs about the label; prior beliefs might be chosen based on the anticipated statistics of the test document collection, e.g. 50% and 50%. When we evaluate the formula on the test documents,

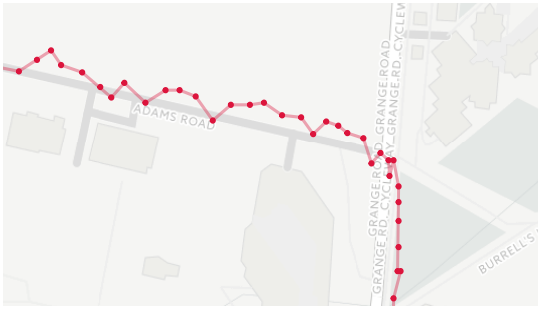
$$\mathbb{P}(\texttt{spam}|\text{test email 1}) = 1$$
$$\mathbb{P}(\texttt{spam}|\text{test email 2}) = \text{divide by zero error.}$$

What does the divide by zero error mean? The simple naïve Bayes model might be inaccurate but it's not impossible, so something must have gone wrong with the way be applied it to the data. The statistics component of this course will give you practice at debugging this sort of 'inference bug'.

1. Probabilistic models

Goals. Refresh your memory of IA Maths for NST, where you were taught some basic probability, and practice on some harder questions. Learn about the four major types of probabilistic model, through examples.

Application. Suppose we wanted to write an app to detect if the user is cycling, running, or driving, and which records or assists as appropriate. The GPS readings might look something like this (showing one sample per second).



GPS readings are noisy, so the user's exact location is unknown, but it probably doesn't jerk about as wildly as the GPS readings do. The user is probably cycling given the speed, which suggests how smooth the trajectory is likely to be. If we had a large dataset of traces, we should be able to learn typical GPS noisiness, as well as typical statistics about speed and smoothness for different modes of transport.

We often face these generic issues in data science applications:

- observations are noisy;
- there is hidden state (true position and transport mode) that we'd like to reason about;
- it's a dynamical system we're observing, and if we know how it typically evolves then this tells us something about the hidden state;
- we want to learn system parameters from a large dataset.

In this section we'll look at four common types of model, which showcase these issues. The actual GPS smoothing problem is too involved for lectures, so we'll look at simpler idealized models.

1.1. Random samples

Application. I've developed a new load-balancing algorithm for my web server. I want to test my algorithm, by means of simulation. My simulator needs a random number generator (RNG) to generate file sizes, request times etc. The performance of my load balancer will surely depend on the random number generator I use. How should I program this random number generator?

We use RNGs in situations like this because the real world is too complicated to model in a Newtonian cause-and-effect way. We use random numbers to say "There is variability, and I can quantify the degree of variability, but I'm not going to look in excruciating detail for causes for every little variation." It's up to the modeler to draw the line between 'causes of variation that it's worth including explicitly' and 'residual variation that we'll label noise'.

Typically we take real-world measurements, we look at the data, and we pick a RNG that produces output consistent with the data. Many standard RNGs come with tuneable parameters. Typically we look at the data to estimate what values to use for the tuning parameters (and, in this application, to figure out how the parameters vary with time of day, request type, etc.)

Working definitions. A *random variable* is a function that can give different answers, e.g. a function that calls a random number routine. We say it *takes values in* S to mean that the return value of the function is in the set S. A *random sample* is a random variable that returns a list, in which the individual elements are chosen independently. A *dataset* is a collection of numbers.

Example. Here is a random variable:

```
1 def rgeom(p):
2     x = 1
3     while random.random() > p:
4         x = x + 1
5     return x
```

Let X be the output of rgeom(1/3). To find the distribution of X, we can use simple probability calculation. To get $X = 1$ we need the random.random()>p test to fail on the first pass, which has probability p. To get $X = 2$ we need the test to succeed on the first pass then fail on the second, which has probability $(1-p)p$. Generalizing, $\mathbb{P}(X = k) = (1-p)^{k-1}p$.

Example. Let X be the set of birthdays of n people, assuming all days are equally likely, that people are independent, and ignoring leap years. Then X is a random variable, and so is $|X|$. As you saw in IA Maths for NST,

$$\mathbb{P}(\text{all } n \text{ have different birthdays}) = \mathbb{P}(|X| = n) = 1 \times \frac{364}{365} \times \cdots \times \frac{365 - n + 1}{365}.$$

Example. What's the chance that two or more people present in the first lecture for this course share a birthday?

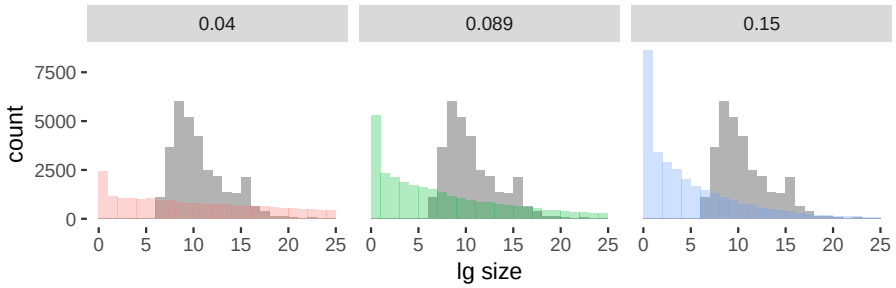
This is badly put, since it's describing a dataset not a random variable. Either there is a shared birthday, or there isn't, so the probability is either 1 or 0. So what on earth did the example on page 1 mean by "`panda`, 57.7% confidence"? This is a hard question, and we'll come back to it in Section 3.

When we write for example "Let X be rexp(3)", remember that X doesn't *have* any particular value. It's a stand-in for all the possible values that the random variable might produce, weighted by their probabilities.

* * *

For a random variable that takes parameters, such as rgeom(p), the obvious question is "How should I pick the value of the parameter so that the output of a random variable matches my dataset?". A very simple (and perfectly good) answer is the eyeball method: plot the histogram of your dataset, and superimpose the histogram of a random sample from the RNG, and tune the parameter until they look close.

Example. I collected 31,078 lines from the request log of a web server, and extracted the size in bytes of the contents. The sizes vary hugely, from 1 byte to 8,944,270 bytes, so to see them on a sensible scale I computed $\log_2(\text{size})$ and rounded down. I then generated 31,078 random variables using rgeom(p) for different values of p. This plot shows the results, with one panel for each value of p I tried.



It looks like rgeom(p) is a bad choice for this dataset, for the three values of p we tried. This prompts two questions: Is there a systematic way to pick the best p? and Is there a systematic way to pick the best RNG? The answer to the former is Yes. The response to the latter is That's a fundamentally wrong-headed question, as we'll discover in Section 2.

Definitions. A random variable is *discrete* if it takes values in some countable space. The *density function* for a discrete random variable X is $f(x) = \mathbb{P}(X = x)$. If the density function depends on some parameter θ, then the *likelihood* of a sample $x_1, \ldots, x_n$ is

$$\text{lik}(\theta | x_1, \ldots, x_n) = f(x_1) f(x_2) \cdots f(x_n).$$

Procedure. A sensible way to estimate an unknown parameter, given a dataset, is to find the value of the parameter that maximizes the likelihood. This is called *maximum likelihood estimation.*

Example. Suppose I have a dataset $x_1, \ldots, x_n$, all integers, and my model is the rgeom(p) random variable. The likelihood is

$$\begin{aligned} \text{lik}(p | x_1, \ldots, x_n) &= (1-p)^{x_1 - 1} p \times \cdots \times (1-p)^{x_n - 1} p \\ &= (1-p)^{s-n} p^n \quad \text{where } s = x_1 + \cdots + x_n. \end{aligned}$$

It's usually easier to maximize $\log \text{lik}()$ rather than lik(), because talking logs turns products into sums. Thus

$$\begin{aligned} \log \text{lik}(p | x_1, \ldots, x_n) &= (s-n) \log(1-p) + n \log p, \\ \frac{d}{dp} \log \text{lik}(p | x_1, \ldots, x_n) &= -\frac{s-n}{1-p} + \frac{n}{p}. \end{aligned}$$

To maximize this, we solve $d/dp = 0$, giving the maximum likelihood estimator $\hat{p} = n/s$. For the dataset of web server content sizes shown above, it evaluates to $\hat{p} = 0.089$.

* * *

Maximum likelihood estimation is nearly universal. It's simple to explain. It's easy to compute (or at least no harder than any other method). It's got some nice properties:

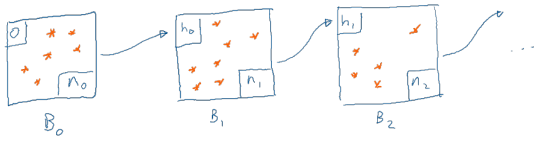
- It's intuitively plausible. For simple problems like "Toss n biased coins, get x heads, estimate the the probability of heads" it gives the sensible answer x/n.
- If the RNG truly is the correct RNG for the dataset, then one can prove that as the size of the dataset increases, the maximum likelihood estimator is guaranteed to approach the true value of the parameter. We'll learn more about the properties of big datasets in Section 2.

But there's nothing necessarily 'true' about maximum likelihood estimation. You'll learn from exercises on the example sheet that the maximum likelihood estimator can be biased, and that it performs poorly when there are lots of parameters to estimate.

Pay close attention to the style of reasoning behind the two bullet points. "Here's a method, M. How good is M? Let's take an RNG, consider a random sample generated from that RNG, and see how close M gets to the truth." This is a good sanity check, and if the method didn't pass this test then we wouldn't want to use it. But, as the quotes on page 2 remind us, any model we use is wrong. What we need to ask is "How robust is this method, i.e. does it give a useful answer when my model is wrong?". This is much harder to answer. As you gain experience of data science, you will develop the skill to look at a method and quickly see what sorts of modeling errors will trick it, and then measure by how much.

1.2. Markov models

Application. Bitcoin[11] is a decentralized electronic cash system, introduced by Satoshi Nakamoto in 2008. It has been a wild success, arguably because of the ingenious way it balances incentives[12]. An important part, and a focus of Nakamoto's original paper, was how to solve the 'double spend' problem in a decentralized system. To understand what this problem is, and how Bitcoin solves it, let's start with some background.

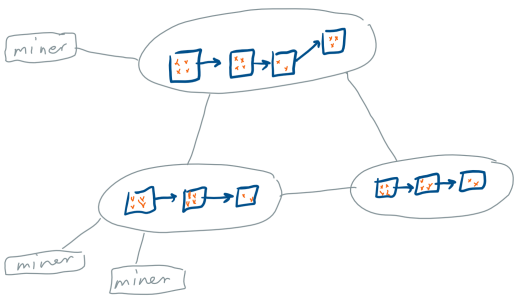


$$
\begin{aligned}
&n_0 \text{ chosen such that } \operatorname{hash}(0, B_0.\text{records}, n_0) < \text{threshold},\\
&h_0 = \operatorname{hash}(0, B_0.\text{records}),\\
&n_1 \text{ chosen such that } \operatorname{hash}(h_0, B_1.\text{records}, n_1) < \text{threshold},\\
&h_1 = \operatorname{hash}(h_0, B_1.\text{records}),\\
&\vdots
\end{aligned}
$$

Bitcoin is a system is for storing and verify transaction records, which are depicted as stars in the diagram. A transaction record might be e.g. Tx_1 ="Alice transfers coin 314 to Bob", cryptographically signed. Transaction records are assembled into blocks $B_0, B_1, \ldots$, and each block additionally includes two values: the hash of the previous block, and a nonce which solves a computationally demanding inequality.

In the simplest world, there might be a central bank which publishes blocks, say one block every 10 minutes. If Alice wants to pay Bob, she asks the bank to record Tx_1, the bank checks that previous blocks confirm that Alice owns the coin, Bob waits until the bank publishes a new block containg Tx_1, and then he posts the widget to Alice. The nonces are unnecessary, in a centralized system.

Bitcoin is a decentralized system with roughly 9750 nodes[13], each of which keeps a copy of the entire blockchain. When Alice wants to record the transaction, she sends it to one of these nodes, which broadcasts it to the rest of the network; it takes 1.5 seconds to reach 50% of the nodes. Other machines work to mine blocks, i.e. to find a nonce with a suitable hash. The time between blocks depends on the threshold; Nakamoto specified an algorithm to dynamically adapt the threshold so that a new block is mined every 10 minutes on average, regardless of the number of miners. Roughly 3.9×10^{21} hashes must be tested to find a block, and each block contains around 2000 transactions. When a block has been mined, the nodes broadcast it to each other, and it takes roughly 5 seconds to reach 50% of the nodes.



[11] Satoshi Nakamoto. *Bitcoin: A peer-to-peer electronic cash system*. 2008. URL: `https://bitcoin.org/bitcoin.pdf`.

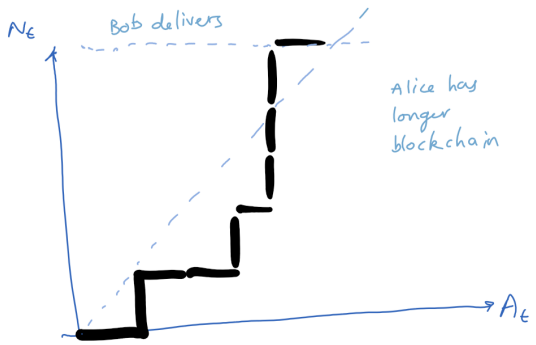
[12] Simon Barber et al. "Bitter to Better : How to Make Bitcoin a Better Currency". In: *Financial Cryptography—FC 2012*. Vol. 7397. Lecture Notes in Computer Science. 2012, pp. 399–414. URL: `http://www.cs.stanford.edu/~xb/fc12/`.

[13] Bitcoin statistics are from September 2017. Current statistics can be found at `bitnodes.21.co/nodes/live-map`, `bitcoinstats.com/network/propagation`, `data.bitcoinity.org/bitcoin/block_time`, `statoshi.info/dashboard/db/mining`, `www.bitcoinmining.com/bitcoin-mining-hardware`, `blockchain.info/charts/n-transactions-per-block`.

What we've described so far allows money to be double-spent. Suppose Alice broadcasts Tx_1 "Alice transfers coin 314 to Bob", Bob sees this, and sends Alice the widget. Suppose Alice also creates a new transaction Tx_2 "Alice transfers coin 314 to Alicia" (her alter-ego), mines a block containing Tx_2, and broadcasts it. Now Alice has the widget, and if Alice's block gets spread widely then everyone accepts that Alicia owns the money.

The Bitcoin strategy to prevent double-spend attacks is for nodes to use the rule "If there are two possible chains, discard the shorter", and for Bob to use the rule "Wait for 6 blocks (1 containing Tx_1, then 5 more chained after it)" before sending Alice the widget. To double-spend, Alice would need to create an alternative history with Tx_2 rather than Tx_1, and get it accepted by the rest of the nodes. The chance of a successful double-spend attack should be small, assuming Alice doesn't own too much of the worldwide block mining power. What is the chance of this? And why did Nakamoto come up with "wait for 6 blocks"?

Nakamoto's calculation was as follows. Assume that Alice controls a fraction p of the worldwide block mining power. Let A_t be the number of blocks that Alice has mined at time t after her attempted double-spend, and let N_t be the number of blocks mined by everyone else, so they start[14] at $A_0 = N_0 = 0$. At any point in time, the probability that the next block comes from Alice is p, and the probability it comes from someone else is $1 - p$. We want to calculate the probability that, at any time after reach $N_t = 6$, we later hit $A_s > N_s$.



The starting point is to split $\mathbb{P}(\text{Alice double-spends})$ using conditional probability, conditioning on how it might have happened.

$$\mathbb{P}\begin{pmatrix}\text{Alice}\\ \text{double-}\\ \text{spends}\end{pmatrix} = \sum_a \mathbb{P}\left(\begin{matrix}\text{Alice}\\ \text{double-}\\ \text{spends}\end{matrix} \;\middle|\; \begin{matrix}A_t = a \text{ when}\\ \text{Bob delivers}\end{matrix}\right)\mathbb{P}\begin{pmatrix}A_t = a \text{ when}\\ \text{Bob delivers}\end{pmatrix}.$$

For the first term, with a flash of insight, we spot that all that matters is the gap $N_t - A_t$. At the instant Bob delivers, this is equal to $6 - a$, and thereafter it may go up or down, and if it ever hits -1 then Alice double-spends. Let

$$\pi_x = \mathbb{P}\left(\begin{matrix}\text{eventually hit}\\ N_s - A_s = -1\end{matrix} \;\middle|\; \begin{matrix}\text{currently at}\\ N_t - A_t = x\end{matrix}\right)$$

and split this using conditional probability, conditioning on who mines the next block:

$$\begin{aligned}\pi_x &= \sum_{m\in\{\text{others, Alice}\}} \mathbb{P}\left(\begin{matrix}\text{eventually hit}\\ N_s - A_s = -1\end{matrix} \;\middle|\; \begin{matrix}\text{currently at } N_t - A_t = x,\\ \text{and next block mined by } m\end{matrix}\right)\\ &\qquad\qquad \times \mathbb{P}\left(\begin{matrix}\text{next block}\\ \text{mined by } m\end{matrix} \;\middle|\; \begin{matrix}\text{currently}\\ \text{at } N_t - A_t = x\end{matrix}\right)\\ &= \mathbb{P}\left(\begin{matrix}\text{eventually hit}\\ N_s - A_s = -1\end{matrix} \;\middle|\; \begin{matrix}\text{now at}\\ N_t - A_t = x+1\end{matrix}\right)(1-p)\\ &\qquad\qquad + \mathbb{P}\left(\begin{matrix}\text{eventually hit}\\ N_s - A_s = -1\end{matrix} \;\middle|\; \begin{matrix}\text{now at}\\ N_t - A_t = x-1\end{matrix}\right)p \qquad (1)\\ &= \pi_{x+1}(1-p) + \pi_{x-1}p\end{aligned}$$

[14] What if Alice prepares some malicious blocks in advance, and only launches her attack when she has enough blocks stored? This is a problem, called the selfish miner attack. See for example Yonatan Sompolinsky and Aviv Zohar. "Bitcoin's Security Model Revisited". In: *CoRR* (2016). URL: `http://arxiv.org/abs/1605.09193`.

except for $x \le -1$ where $\pi_x = 1$. This is now a pure maths recurrence equation, and we can solve it and discover

$$\pi_x = \begin{cases} \left(\frac{p}{1-p}\right)^{x+1} & \text{if } x \ge 0, \\ 1 & \text{if } x \le -1. \end{cases}$$

(You should check that this does indeed solve the recurrence equation. We'll see many more calculations along these lines in Section 5, and we'll discuss what happens when the recurrence equation has more than one solution.)

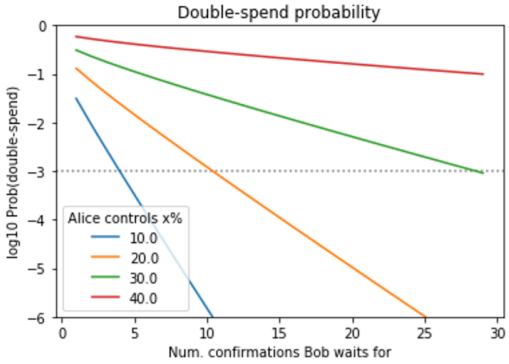
We still have to work out $\mathbb{P}(A_t = a$ when Bob delivers). Call it ξ_a. By simply counting up all the ways that we might reach $N_t = 6$, along the lines of Example 1, we find

$$\begin{aligned} \xi_0 &= (1-p)^6 \\ \xi_1 &= p(1-p)^2 + (1-p)p(1-p)^5 + \cdots + (1-p)^5 p(1-p) = \binom{6}{1} p(1-p)^6 \\ \xi_a &= \binom{a+5}{a} p^a (1-p)^6. \end{aligned}$$

Putting everything together,

$$\begin{aligned} \mathbb{P}\begin{pmatrix}\text{Alice}\\ \text{double-}\\ \text{spends}\end{pmatrix} &= \sum_{a\in\{0,\ldots,6\}} \binom{a+5}{a} p^a (1-p)^6 \left(\frac{p}{1-p}\right)^{6-a+1} + \sum_{a\ge 7} \binom{a+5}{a} p^a (1-p)^6 \\ &= \sum_{a\in\{0,\ldots,6\}} \binom{a+5}{a} p^a (1-p)^6 \left(\frac{p}{1-p}\right)^{6-a+1} + \left(1 - \sum_{a\in\{0,\ldots,6\}} \binom{a+5}{a} p^a (1-p)^6\right). \end{aligned}$$

Here are some numbers, showing the probability that Alice successfully double-spends, depending on the proportion p that of block mining power that she controls, and the number of confirmations that Bob waits for. This is the data that Nakamoto used to choose the rule "wait for 6 confirmations".



* * *

In this problem, we studied a dynamical process (N_t, A_t), $t \ge 0$. There were several key steps in how we analysed it:

Conditioning. To find $\mathbb{P}$(event), we conditioned on how the event happened. We chose the conditioning strategically. We applied it in two ways: we conditioned on A_t at the instant that Bob delivers the widget, and we conditioned on who mines the next block. This conditioning is an application of the *law of total probability*, which states

$$\mathbb{P}(A) = \sum_i \mathbb{P}(A|B_i)\mathbb{P}(B_i)$$

where A is an event and $B_1, \ldots, B_n$ are mutually exclusive events that cover all possible ways that A can occur.

Conditioning might remind you of dynamic programming, the algorithmic strategy of decomposing a problem into smaller sub-problems. There is a substantial difference, though. We came up with the equation

$$\pi_x = \pi_{x+1}(1-p) + \pi_{x-1}p$$

which does not lend itself to a recursive algorithm because there is no base case. In dynamic programming, on the other hand, the goal is to turn the problem into a recursive algorithm. In Section 5 we'll look at computational methods for solving equations like this.

Memorylessness. The most important step in the calculation was (1). It expresses the idea "What happens in the future depends only on where you are now, not on how you got here." Why is this true for bitcoin? The way the bitcoin hash calculation works, finding a nonce is like winning the lottery: if you haven't won one so far, it doesn't mean you're more likely to win next time. If we're at state $N - A = x + 1$, it's immaterial whether we reached there from $N - A = x$ or from $N - A = x + 2$, the future looks exactly the same either way. This is called *memorylessness*, or alternatively the *Markov property* after the Russian mathematician Andrey Markov, who invented the theory of memoryless processes, which we will study in much more detail in Section 5.

There are very many non-Markov processes, but they're often much harder to analyse. Even in the bitcoin problem, we can question whether it truly is memoryless. If for example the number of mining machines was slowly varying, then the fact "$N - A$ used to be $x + 2$ before it became $x + 1$" gives a slight hint that Alice might have a slightly higher number of miners than she started with, which would impact our estimate of what happens in the future. To make the problem memoryless, we assumed that Alice controls precisely p of the worldwide mining power, and that p doesn't change.

Embedded chain. The process (N_t, A_t) evolves in continuous time, but all we chose to look at is the instants where it jumps. This is called *finding an embedded Markov chain*. The word *chain* here means "discrete sequence of events".

1.3. Descriptive models

A group of four friends, A, B, C, and D, are deciding how to vote in the Brexit referendum. There are 16 possible outcomes. Based on survey statistics for similar groups, the estimated chance of each of the 16 possible outcomes for (A, B, C, D) is

$$\begin{array}{ll} (0,0,0,0) & 17.109\% \\ (0,0,0,1),(0,0,1,0),(0,1,0,0),(1,0,0,0) & 6.095\% \\ (0,0,1,1),(0,1,1,0),(1,1,0,0),(1,0,0,1) & 7.821\% \\ (0,1,0,1),(1,0,1,0) & 0.295\% \\ (1,1,1,0),(1,1,0,1),(1,0,1,1),(0,1,1,1) & 3.069\% \\ (1,1,1,1) & 14.361\% \end{array} \tag{2}$$

where 0 means remain and 1 means exit.

The voter dataset is called *multivariate*, meaning that each item is a vector, in this case of length four. It's simple to simulate a random outcome in Python:

How would you implement random.choices() if your language provides only simple uniform random numbers in $[0, 1]$? There is an obvious method, and a clever trick called the Alias Method that you'll explore in the example sheet.

```
def rvote():
    outcomes = [(0,0,0,0), (0,0,0,1), ...]
    probs = [0.17109, 0.060905, ...]
    return random.choices(outcomes, weights=probs)
```

There are many other ways we could implement the simulator. For example, simply adding up the probabilities we get

$$\begin{aligned} &\mathbb{P}(A = 0) = 54.4\% \\ &\mathbb{P}(B = 0 \mid A = 0) = 68.2\%, \\ &\mathbb{P}(B = 0 \mid A = 1) = 37.8\% \\ &\mathbb{P}(C = 0 \mid A = 0, B = 0) = 62.5\%, \\ &\mathbb{P}(C = 0 \mid A = 0, B = 1) = 37.0\%, \quad \ldots \end{aligned}$$

which suggests the code

```
def rvote2():
    pA = 0.544
    A = random.choices([0,1], weights=[pA, 1-pA])
    pB = {0: 0.682, 1: 0.378}[A]
    B = random.choices([0,1], weights=[pB, 1-pB])
    pC = {(0,0): 0.625, (0,1): 0.370, ...}[(A,B)]
    C = random.choices([0,1], weights=[pC, 1-pC])
    ...
    return (A, B, C, D)
```

These two random simulators are equivalent—they generate exactly the same distribution of outcomes.

When we're presented with a multivariate dataset, we typically start by investigating how one variable depends on the others, or on some summary of the others. This is called the *marginal distribution* of the variable we're investigating. For example, pick any one of the four friends, and call their vote Y, and let X be the total vote of the other three. Then

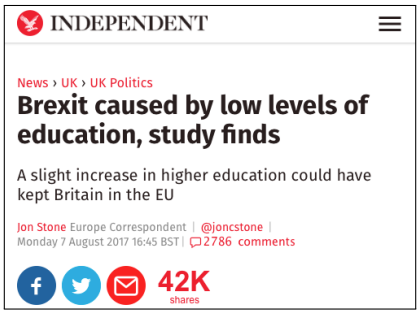
$$\mathbb{P}(Y = 0 \mid X = x) = \begin{cases} 73.7\%, & \text{if } x = 0 \\ 53.4\%, & \text{if } x = 1 \\ 63.4\%, & \text{if } x = 2 \\ 17.6\%, & \text{if } x = 3. \end{cases} \tag{3}$$

We might write the model this way if we want to predict the behaviour of an individual voter. The terms that we're conditioning on are called *features*, and it's an art to find useful features. We'll discuss this further in Section 4.

Definitions. A *descriptive* or *observational* model specifies the distribution of the data, without specifying the mechanism by which it was generated. A *discriminative* model is a type of

descriptive model, written as the marginal distribution of some variable of interest conditional on *features*.

The 'outrage media' generates catchy headlines when it muddles descriptive models and causal mechanisms. Here's a headline[15] describing a study that was purely observational:



In Section 1.4 we'll look harder at causal versus descriptive models.

* * *

A group of four friends are deciding how to vote in the Brexit referendum. For each of them, the probability of voting Remain given the number of friends n who vote Leave is 73.7% (if $n = 0$), or 53.4% (if $n = 1$), or 63.4% (if $n = 2$), or 17.6% (if $n = 3$).

Sometimes, we're only told marginal distributions, as in this second dataset. This invites questions:

- Is there a descriptive model that's consistent with these marginals, i.e. is it possible to define a random variable (A, B, C, D) that yields these marginals? In this case, yes, because the marginal probabilities are exactly what's specified in equation (3), which we know were derived from the full descriptive model in equation (2).
- Is there always a consistent descriptive model? No. Suppose we wrote down a contrarian model, $\mathbb{P}(A = a|B = b) = 1_{a=b}$ and $\mathbb{P}(B = b|A = a) = 1_{a\neq b}$. There's no distribution for (A, B) that has these two marginals.
- But isn't that contrarian case pathological? Yes, it is. We can always just define the probability of a particular outcome to be the *product* or *factor distribution*, i.e.

The notation 1_Q is shorthand for "1 if Q is true, 0 if Q is false".

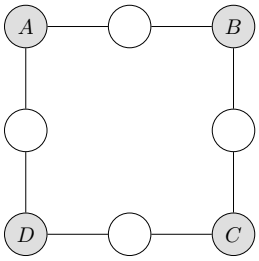
$$\mathbb{P}(A = a, B = b, C = c, D = d) \\ = \kappa\, \mathbb{P}(A = a \mid \dots)\mathbb{P}(B = b \mid \dots)\mathbb{P}(C = c \mid \dots)\mathbb{P}(D = d \mid \dots)$$

  and pick κ so that the probabilities sum to 1. In the pathological case, all the probabilities are zero so it's impossible to pick κ. In all other cases, we can.
- Is there a unique descriptive model that's consistent with a given set of marginal distributions? No. In this case, the original descriptive model (2) turns out to be different to the product distribution.

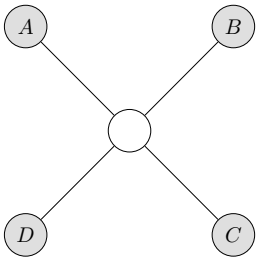
You often see factor distributions drawn as undirected graphs, with a node for each variable and edges to indicate which variables depend on which other variables. They are also called *Markov random fields*, where the word 'Markov' here means 'depends only on neighbours'. The original voter distribution in equation (2) was actually produced from a factor distribution, with some extra hidden variables corresponding to pairs of friends.

[15] http://www.independent.co.uk/news/uk/politics/brexit-education-higher-university-study-university-leave-eu-remain-voters-educated-a7881441.html

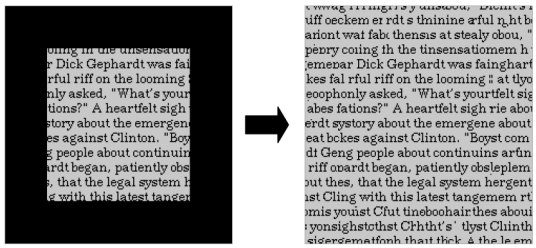
Graph behind the first voter model:



Graph of the second voter model:



Factor distributions are popular with physicists, e.g. the Ising model for magnetism. They can also be used in computer graphics, for synthesizing texture. Take a source image with a sample of the texture, pick a window size, e.g. 11×11 pixels, and work out the marginal distribution of the center pixel conditional on the other $11^2 - 1$ pixels. Consider the entire area we want to texturize to be a multivariate random variable, with the factor distribution, and generate a single sample of this random variable. (In Section 5 we'll learn how to generate samples from factor distributions, using Markov chains.) Here is an example[16].



[16]Taken from lecture notes by Steven Seitz at the University of Washington, `https://courses.cs.washington.edu/courses/csep576/05wi/lectures/texture.pdf`

1.4. Causal models

The goal of data science is often to answer policy questions.

> *Should we cut tuition fees? What does the data tell us will happen?*

This is a hypothetical question, about a situation that hasn't yet been observed. Such questions may be dressed up to look like descriptive questions

> *In places or times where tuition fees were cut, what happened?*

but there's an implicit claim

> *If we were to cut tuition fees now, the same would surely happen.*

Sometimes it's expressed as a counterfactual question:

> *If we had cut tuition fees, what state would we be in now?*

Definition. A model is called *causal* or *generative* if the steps in the code represent mechanisms, such that if we intervene and alter some value or mechanism then the rest of the mechanism still works as before. It's convenient to draw causal models as directed acyclic graphs, where the edges show which variables are used to generate which other variables. A variable is called *latent* if it is not observed.

Descriptive models don't tell you the order in which variables are generated. In Section 1.3, for example, we saw two different mechanisms `rvote()` and `rvote2()` for simulating votes, which both correspond to exactly the same descriptive model. But if we intervene and force D to vote remain (by adding a line of code), then the two mechanisms will produce different outcome distributions. So, if all we know are descriptive summaries, it's generally impossible to answer causal questions.

A (fictional) drug is taken by some members of the population, and it leads to better survival outcomes. A zealous health minister wants to add the drug to drinking water. Before this is approved, a heroic scientist runs a controlled trial, which show the drug actually leads to worse outcomes.

	population		trial	
	drug	clean	drug	clean
$\mathbb{P}$(death)	.002	.028	.016	.014
$\mathbb{P}$(survival)	.998	.972	.984	.986

This numbers are taken from Tian and Pearl[17], who put them instead in a much more interesting legal counterfactual context:

> *A lawsuit is led against the manufacturer of drug x, charging that the drug is likely to have caused the death of Mr A, who took the drug to relieve symptom S associated with disease D.*
>
> *The manufacturer claims that experimental data on patients with symptom S show conclusively that drug x may cause only minor increase in death rates. The plaintiff argues, however, that the experimental study is of little relevance to this case, because it represents the effect of the drug on all patients, not on patients like Mr A who actually died while using drug x. Moreover, argues the plaintiff, Mr A is unique in that he used the drug on his own volition, unlike subjects in the experimental study who took the drug to comply with experimental protocols. To support this argument, the plaintiff furnishes nonexperimental data indicating that most patients who chose drug x would have been alive if it were not for the drug. The manufacturer counter-argues by stating that: (1) counterfactual speculations regarding whether patients would or would not have died are purely metaphysical and should be avoided, and (2) nonexperimental data should be*

[17] Jin Tian and Judea Pearl. "Probabilities of Causation: Bounds and Identification". In: *Proc. of the 16th Conference on Uncertainty in Artificial Intelligence*. 2000. URL: `https://arxiv.org/ftp/arxiv/papers/1301/1301.3898.pdf`.

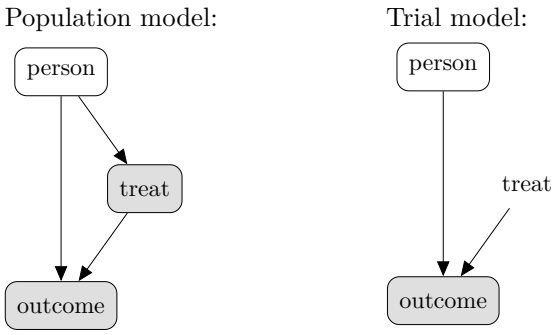
dismissed a priori, on the ground that such data may be highly biased; for example, incurable terminal patients might be more inclined to use drug x if it provides them greater symptomatic relief. The court must now decide, based on both the experimental and non-experimental studies, what the probability is that drug x was in fact the cause of Mr A's death.

Let's invent a probabilistic model, that can generate outcomes like those in the table. We'll imagine there are two types of patient, regular and terminal. Greek letters indicate parameters of the model.

```
def population():
    person = random.choices(['regular','terminal'], weights=[π,1−π])
    p_drug = {'regular': θ, 'terminal': ϕ}[person]
    treat = random.choices(['drug','clean'], weights=[p_drug, 1−p_drug])
    p_death = ξ_{person, treat}
    return random.choices(['die','survive'], weights=[p_death, 1−p_death])

def trial(treat):
    person = random.choices(['regular','terminal'], weights=[π,1−π])
    p_death = ξ_{person, treat}
    return random.choices(['die','survive'], weights=[p_death, 1−p_death])
```

Drawn as directed acyclic graphs, with white boxes for latent random variables and shaded boxes for observed random variables,



If we simply count up the probability of each outcome, we get the formulae in the table below. In the population table all four probabilities sum to 1 (since the table describes the characteristics of a random person), whereas in the trial table each column sums to 1 (since the table describes the outcome for each type of patient in the trial). Parameters that correspond to the percentages given earlier are $\pi = 0.1183$, $\theta = 0$, $\phi = 0.125$, $\xi_{\text{regular,drug}} = 0$, $\xi_{\text{regular,clean}} = 0.1184$, $\xi_{\text{terminal,drug}} = 0.0181$, $\xi_{\text{terminal,clean}} = 0$.

population:	drug	clean
$\mathbb{P}$(death)	$\pi\theta\xi_{r,d} + (1-\pi)\phi\xi_{t,d}$	$\pi(1-\theta)\xi_{r,c} + (1-\pi)(1-\phi)\xi_{t,c}$
$\mathbb{P}$(survival)	$\pi\theta(1-\xi_{r,d}) + (1-\pi)\phi(1-\xi_{t,d})$	$\pi(1-\theta)(1-\xi_{r,c}) + (1-\pi)(1-\phi)(1-\xi_{t,c})$

trial:	drug	clean
$\mathbb{P}$(death)	$\pi\xi_{r,d} + (1-\pi)\xi_{r,d}$	$\pi\xi_{r,c} + (1-\pi)\xi_{r,c}$
$\mathbb{P}$(survival)	1 - $\pi\xi_{r,d} + (1-\pi)\xi_{r,d}$	1 - $\pi\xi_{r,c} + (1-\pi)\xi_{r,c}$

The legal counterfactual question is this: We know Mr A took the drug and died; given this, what's the probability he would have died if he hadn't taken the drug? Let's use Bayes's rule to first work out the probability that Mr A is regular or terminal given these facts.

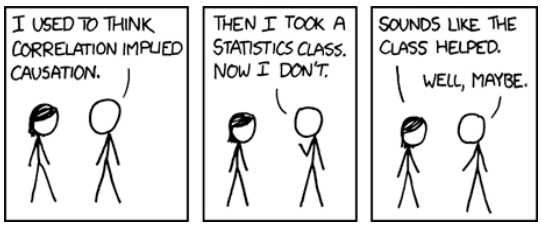
$$\mathbb{P}(\text{regular} \mid \text{drug, dead}) = \frac{\mathbb{P}(\text{drug, dead} \mid \text{regular})\mathbb{P}(\text{regular})}{\mathbb{P}(\text{drug, dead} \mid \text{regular})\mathbb{P}(\text{regular}) + \mathbb{P}(\text{drug, dead} \mid \text{terminal})\mathbb{P}(\text{terminal})}$$

and this is 0 because regular people do not die on the drug, $\xi_{\text{regular,drug}} = 0$. Therefore Mr A is terminal. But $\xi_{\text{terminal,clean}} = 0$, i.e. terminal people do not die when they're not on the drug, therefore Mr A's death was caused by the drug.

We have worked through this calculation with an explicit and invented probabilistic model, as expressed in the code and the parameter values. The remarkable contribution of Tian and Pearl was to show that the same conclusion must hold *whatever* the model.

* * *

You sometimes come across the glib remark 'correlation does not imply causation', but causality theory goes far beyond this. It is a relatively new area, still barely developed, and still contentious. Machine learning algorithms are getting very good at descriptive and discriminative models, but I think it will be 15 years or more before AIs can manage general causal reasoning. Even today, according to Pearl[18], many human statisticians still find it perplexing that we can draw counterfactual conclusions from observational data. Here's a last word from the inimitable Randall Munroe.[19]



* * *

What is modelling for?

- Sometimes we put forwards a model, even though we don't really believe it's accurate, because it lets us make useful inferences about hidden variables. Any reasonably sensible model for human motion should let us infer walking / cycling / driving from GPS traces, and probabilistic modeling is a convenient and interpretable tool that won't lead to crazy answers.
- Sometimes we have detailed observational data and just we want to find a reasonably good and simple approximation, either for storage or speed reasons. Here we're looking for a reasonably good fit for the distribution, and we want to find useful features and parameters.
- Sometimes we want to build a product that makes predictions, e.g. that classifies new images. Probabilistic models are one way to build predictors, and they're often useful when we want to assess how well our product might perform on data of a type it hasn't seen before.
- Sometimes we're interested in building a simulator of a system that we understand well. It's helpful to be familiar with a range of probability models, to be able to pull 'off the shelf' whatever is appropriate and has good library support.
- Sometimes we want to answer policy questions, or to build our scientific understanding. This, in my opinion, is the truest data science, and by far the most challenging.

[18] Judea Pearl. *Causality: Models, Reasoning and Inference*. 2nd ed. Cambridge University Press, 2009.
[19] `https://xkcd.com/552/`

1.5. Common random variables

It's useful to have a range of common random variables at our fingertips.

Definition. A random variable X is *discrete* if it takes values in some countable space, such as the integers. The *density* is $f(x) = \mathbb{P}(X = x)$. The density must be everywhere ≥ 0 and must sum to 1.

Definition. A random variable X is *continuous* if it takes values in a continuous set, such as the real numbers, or $[0, 1]$. A continuous random variable has a *density function* $f(x)$ such that

$$\mathbb{P}(X \in A) = \int_{x \in A} f(x)\, dx \quad \text{for all sets } A.$$

The density must be everywhere ≥ 0 and must integrate to 1.

VARIABLES ASSOCIATED WITH WAITING AND COUNTING

Geometric: If we're playing a lottery, and each week the chance of winning is p, then our first win happens on week $X \sim \text{Geom}(p)$. This random variable takes values in $\{1, 2, \dots\}$, and

$$\mathbb{P}(X = r) = (1-p)^{r-1}p, \quad \mathbb{P}(X \geq r) = (1-p)^{r-1}.$$

We came across it in Section 1.1. In Python, numpy.random.geometric(p).

Exponential: The Exponential random variable is a continuous-time version of the Geometric. It's used to model the time until an event, for many natural processes: for example the time until a lump of radioactive matter emits its next particle, or the time until a lightbulb blows, or the time until the next web request arrives. If $X \sim \text{Exp}(\lambda)$ then it takes values in $[0, \infty)$, and

$$f(x) = \lambda e^{-\lambda x}, \quad \mathbb{P}(X \geq x) = e^{-\lambda x}, \quad \mathbb{E}X = \frac{1}{\lambda}.$$

The parameter λ is called the *rate*. The chance of an event in a short interval of time $[t, t+\delta]$ is

$$\mathbb{P}(X \leq t + \delta \mid X \geq t) = \frac{\mathbb{P}(X \in [t, t+\delta])}{\mathbb{P}(X \geq t)} = \frac{\int_t^{t+\delta} \lambda e^{-\lambda x}\, dx}{e^{-\lambda t}} \approx \delta\lambda.$$

In Python, numpy.random.exponential(scale=$1/\lambda$).

Binomial: If we toss a biased coin n times, and each coin has chance p of heads, the total number of heads is $X \sim \text{Bin}(n, p)$. This random variable takes values in $\{0, 1, \dots\}$, and

$$\mathbb{P}(X = r) = \binom{n}{r} p^r (1-p)^r.$$

In Python, numpy.random.binomial(n,p). When $n = 1$, i.e. a single coin toss, it's called a Bernoulli random variable. There is a related random variable called the negative binomial, which arose in Section 1.2 when we calculated $\mathbb{P}(A_t = a$ when Bob delivers).

Multinomial: If we have n individuals each of whom falls into one of k categories, and the probability of falling into category i is p_i, then the total number in each category is a multivariate random variable $X \sim \text{Multinom}(n, p)$. We used it for counting outcomes in the drug model in Section 1.4. It takes values in $\{0, 1, \dots, n\}^k$, and

$$\mathbb{P}(X = x) = \frac{n!}{x_1! x_2! \cdots x_k!} p_1^{x_1} p_2^{x_2} \cdots p_k^{x_k}.$$

In Python, numpy.random.multinomial(n,p). (The binomial distribution is the special case when $k = 2$.)

Poisson: The random variable $X \sim \text{Poisson}(\lambda)$ takes values in $\{0, 1, \dots\}$, and

$$\mathbb{P}(X = r) = \frac{\lambda^r e^{-\lambda}}{r!}.$$

In Python, numpy.random.poisson(λ). Suppose we're counting the number of events in a fixed interval of time, for example the number of buses passing a spot on the street, or the number of web requests, or the number of particles emitted by a lump of radioactive matter. If the time between events is $\text{Exp}(\lambda)$, then the total number of events in time t is $X \sim \text{Poisson}(\lambda t)$.

VARIABLES ASSOCIATED WITH SIZES

Normal / Gaussian: This distribution is a very popular choice for data analysis because it's often a good model for things that are the aggregate of many small pieces, for example height which is the aggregate of many influences from genetics and the environment. It's also easy to do probability calculations with it. If $X \sim N(\mu, \sigma^2)$ then X is a continuous random variable taking values in the entire real line, and

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}, \quad \mathbb{E}X = \mu, \quad \text{Var}\, X = \sigma^2.$$

In Python, numpy.random.normal(loc=μ, scale=σ) (and watch out for σ versus σ^2). If $X \sim N(\mu, \sigma^2)$ and $Y \sim N(\nu, \rho^2)$ and they are independent, then

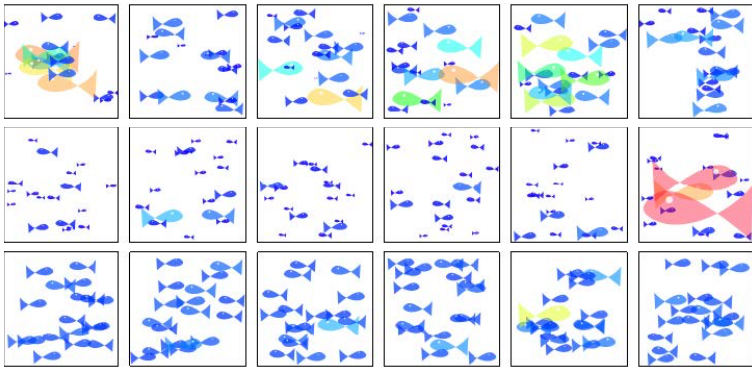
- $aX + b \sim N(a\mu + b, a^2\sigma^2)$
- $(X - \mu)/\sigma \sim N(0, 1)$
- $X + Y \sim N(mu + \nu, \sigma^2 + \rho^2)$.

There is also a multivariate version, called the multivariate normal.

Pareto and lognormal: Some natural phenomena, like sizes of forest fires, or insurance claims, or Internet traffic volumes, or stock market crashes, have the characteristic that there are events of wildly different sizes. This tends to cause problems for simulations and forecasting, since the entire outcome can hinge on one 'black swan' event[20]. A common random variable with this characteristic is the Pareto distribution, $X \sim \text{Pareto}(\alpha)$, named after the Italian economist Vilfredo Pareto who studied extreme wealth inequality. It is a continuous random variable taking values in $[1, \infty)$, and

$$f(x) = \alpha x^{-(\alpha+1)}, \quad \mathbb{P}(X \geq x) = x^{-\alpha}, \quad \mathbb{E}X = \begin{cases} \infty \text{ if } \alpha \leq 1 \\ \alpha/(\alpha - 1) \text{ otherwise.} \end{cases}$$

For $\alpha < 2$ it tends to produce many small values ('mice') and very occasional huge values ('elephants'). To illustrate, here are some samples drawn from three different distributions, all with mean value 1.



$X \sim \text{Exp}(1)$, $\mathbb{E}X = 1$

$X \sim \frac{\alpha-1}{\alpha}\,\text{Pareto}(\alpha)$ with $\alpha = 1.1$, $\mathbb{E}X = 1$

$X \sim \frac{\alpha-1}{\alpha}\,\text{Pareto}(\alpha)$ with $\alpha = 5$, $\mathbb{E}X = 1$

The lognormal distribution $X \sim e^{N(\mu,\sigma^2)}$ has similar characteristics to the Pareto but is not quite as extreme. It was invented by the Cambridge senior wrangler and medic Donald MacAlister.

[20] Nassim Nicholas Taleb. *The Black Swan: The Impact of the Highly Improbable.* 2nd ed. Random House, 2010.

Zipf: The random variable $X \sim \text{Zipf}(n, s)$ takes values in $\{1, 2, \ldots, n\}$ and

$$\mathbb{P}(X = r) = \frac{r^{-s}}{1 + 2^{-s} + \cdots + n^{-s}}.$$

It is named after the American linguist Goerge Zipf, who used it to describe frequencies of words in texts[21]. Take a large piece of text, and count the number of occurrences of each word, and rank the words from most common to least common. Say that the most common word has rank 1, the next most common has rank 2, and so on. Zipf observed that the number of occurrences of the rth ranked word is roughly const $\times r^{-s}$ where $s \approx 1$ in English texts. Another way of putting this: if we pick a word at random from the entire body of text, then the rank of that word is $\text{Zipf}(n, s)$, where n is the size of the vocabulary. The same phenomenon happens with cities: if we take a person at random from the entire population, and look at which city they come from, and rank cities by size, then the rank of that person's city is $\text{Zipf}(n, s)$ where n is the number of cities and s is roughly 1.07.

There is a direct link between the $\text{Pareto}(\alpha)$ and $\text{Zipf}(n, 1/\alpha)$ distributions. First, create a 'pseudo-random' sample of n city sizes, to match the $\text{Pareto}(\alpha)$ distribution. Make the largest city have size $x_{(1)}$ such that $x_{(1)}^{-\alpha} = 1/N$, make the second-largest city have size $x_{(2)}$ such that $x_{(2)}^{-\alpha} = 2/N$, etc. This is a deterministic equivalent of the $\text{Pareto}(\alpha)$ distribution, in which $\mathbb{P}(X \geq x) = x^{-\alpha}$. Then, the city of rank r has size const $\times r^{-1/\alpha}$, which fits with $\text{Zipf}(n, 1/\alpha)$.

[21] See the IA course *Machine Learning and Real-World Data*

1.6. Independence and joint distributions

The concept of independent random variables is fundamental in modeling. Informally it means "knowing the value of one of them gives no information about the other." We've used the word several times so far, but we haven't defined it.

Definition. Two random variables X and Y are *independent* if

$$\mathbb{P}(X \in A,\, Y \in B) = \mathbb{P}(X \in A)\,\mathbb{P}(Y \in B) \quad \text{for all } A \text{ and } B.$$

For discrete random variables it's sufficient to check

$$\mathbb{P}(X = x,\, Y = y) = \mathbb{P}(X = x)\,\mathbb{P}(Y = y) \quad \text{for all } x \text{ and } y,$$

and for continuous random variables with joint density function $f_{X,Y}(x, y)$, it's sufficient to check

$$f_{X,Y}(x, y) = f_X(x) f_Y(y) \quad \text{for all } x \text{ and } y.$$

We can also write the definition in terms of conditional probability:

$$\mathbb{P}(X \in A \mid Y \in B) = \mathbb{P}(X \in A) \quad \text{for all } A \text{ and } B \text{ such that } \mathbb{P}(Y \in B) > 0.$$

A collection of independent random variables drawn from the same distribution, such as we investigated in Section 1.1, are said to be *independent and identically distributed*, abbreviated IID.

Example. I throw a fair die. Let Z be the result. Let $X = Z \bmod 2$ and let $Y = Z \operatorname{div} 3$, so for example $Z = 3$ gives $X = 1$ and $Y = 1$. Are X and Y independent? The definition gives a condition that has to be satisfied for all x and y. Let's try some:

- Try $x = 0$, $y = 0$. For these, $\mathbb{P}(X = 0, Y = 0) = \mathbb{P}(Z = 4) = 1/6$, and $\mathbb{P}(X = 0) = 1/2$ and $\mathbb{P}(Y = 0) = 1/3$. So this pair passes the test.
- Try $x = 0$, $y = 1$. For these, $\mathbb{P}(X = 0, Y = 1) = \mathbb{P}(Z = 4) = 1/6$, and $\mathbb{P}(X = 0) = 1/2$ and $\mathbb{P}(Y = 1) = 1/2$. So the test fails.

Thus X and Y are not independent.

Exercise 1.1. I throw a fair die. Let Z be the result. Let $X = (z - 1) \bmod 2$ and $Y = (Z - 1) \operatorname{div} 2$. Show that X and Y are independent.

Example. In the voting example in Section 1.3, equation (2), are A and $N = B + C + D$ independent? No: when we added up the probabilities we saw $\mathbb{P}(A = 0 \mid N = 0) = 0.737$ and $\mathbb{P}(A = 0 \mid N = 1) = 0.534$, so they can't be independent.

Example. Let X and Y be independent $\mathrm{Bin}(1, p)$ random variables, so

$$\mathbb{P}(X = x,\ Y = y) = p^x (1 - p)^{1-x}\, p^y (1 - p)^{1-y},$$

and suppose p is fixed but unknown. Obviously, learning the value of X tells us something about p (exercise: show that the maximum likelihood estimator for p given X is $\hat{p} = X$). That doesn't mean that X and Y are independent: the joint probability still factorizes into an x-part and a y-part, so the definition of independence is satisfied.

Whenever you hear "independent random variables", it's a good idea to whisper to yourself the coda "given their parameters", so you don't confuse 'unrelated' and 'independent'.

Exercise 1.2. In this code snippet,

```
1 def PXY():
2     P = random.random()  # generates a random number in [0,1]
3     X = numpy.random.binomial(1, P)
4     Y = numpy.random.binomial(1, P)
5     return (P,X,Y)
```

show that X and Y are not independent. Note however that

$$\mathbb{P}(X = x,\, Y = y \mid P = p) = p^x (1 - p)^{1-x}\, p^y (1 - p)^{1-y}$$

which we describe as "X and Y are conditionally independent given P".

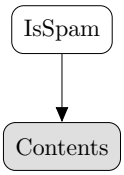
* * *

If we ever try to compute a probability or expectation and we end up with a random variable on the right hand side, we've made a mistake. Probabilities are numbers in $[0, 1]$, and random variables are functions, and we should be hyper-vigilant about which is which. In machine learning we want to write things like

$$\mathbb{P}(\text{email is spam}) = \text{some function of email contents.}$$

It's usually intuitively clear what is meant, but when we come across such statements deep in the middle of a problem with 15 other moving parts it's sometimes befuddling. Are the email contents random? If so, what are they doing on the right hand side of a probability equation? If not, how can the spam-nature be a random variable yet the email's contents be non-random?

It often helps to draw out a belief graph, of the sort we drew in Section 1.4, and to label all random variables with capital letters, and values with lower case letters. Here we really mean



and the probability we want is

$$\mathbb{P}(\text{IsSpam} = \mathsf{true} \mid \text{Contents} = c) = \text{function}(c).$$

As a shorthand for this, we write

$$\mathbb{P}(\text{IsSpam} = \mathsf{true} \mid \text{Contents}) = \text{function}(\text{Contents}).$$

FOR MATHEMATICIANS ONLY

At the beginning of Section 1 we used the working definition "a random variable is a function that can return different answers". Now let's give a better account, which lets us talk about the joint distribution of two random variables X and Y. This will help clear up some difficulties with continuous random variables. It's usually intuitively clear what is needed, so you should treat this section as background reading, for interest only.

A random variable is a function that can return different answers, in the following sense:

```
ω = simulate_experiment(parameters)
def X():
    return some function of ω
def Y():
    return some other function of ω
```

Even for Markov chains, we should think of ω as a complete trace of the entire process, run forever. (Mathematicians don't worry about simple things like finite memory.) A joint distribution like $\mathbb{P}(X = x, Y = y)$ really means $\mathbb{P}(\{\omega : X(\omega) = x,\ Y(\omega) = y\})$. Most of the time we don't have to worry about this level of detail, but it's useful in some tricky cases.

Definitions. A pair of continuous random variables X and Y has a *joint density function* $f_{X,Y}(x, y)$ such that

$$\mathbb{P}\big((X, Y) \in A\big) = \int_{(x,y)\in A} f_{X,Y}(x, y)\, dx\, dy$$

for all sets A in the real number line squared. The density must be everywhere ≥ 0 and must integrate to 1. The marginal density of one of them is

$$f_Y(y) = \int_x f_{X,Y}(x, y)\, dy$$

and the conditional density is

$$f_{Y|X}(y \mid X = x) = \frac{f_{X,Y}(x, y)}{f_X(x)}, \quad \text{assuming } f_X(x) > 0.$$

Definition. Conditional probability, as in $\mathbb{P}(\text{IsSpam} = \text{true} \,|\, \text{Contents})$, is fairly intuitive. It's also intuitive that we should define

$$\mathbb{E}(X \mid Y = y) = \sum_x x\, \mathbb{P}(X = x \mid Y = y) = \sum_x x \frac{\mathbb{P}(X = x,\, Y = y)}{\mathbb{P}(Y = y)},$$

and interpret $\mathbb{E}(X|Y)$ accordingly. To be precise, $\mathbb{E}(X|Y)$ is called a *conditional expectation*, and it is a random variable (because it depends on Y which is itself a random variable). When Y is a continuous random variable, though, $\mathbb{P}(Y = y) = 0$ so the conditioning doesn't make sense—it's a divide-by-zero error—but all we need to do is replace terms like $\mathbb{P}(X = x|Y = y)$ by densities $f_{X|Y}(x|Y = y)$, and replace sums by integrals.