Groups:

Group 1:
Amelia Camino
Jake Magee
Amanda Muratore
Julissa Sarmiento

Group 2:
Finian Ashmead
Francis Burk
Mykola Chernyashevskyy
Mohamed Ismail
Ehteshamul Karim

Group 3:
Cullen Abelson
Alia Dawood
Mira Salman
Yunchong Zhang

Group 4:
Nathalie Chicoine
Lauren Elicker
Yoki Salcedo
Marcos Tamargo-Arizmendi

ASTRON 3705 / PHYSICS 3704

Statistics and Data Science

Spring 2025

Goals for today: you should be able to...

- Generate Monte Carlo simulations using 2D arrays
- Explain the Bayesian definition of probability
- Apply Bayes' theorem

- **We will use the Lecture 3 notebook today!**

Let's join up in our long-term groups!

Before we do anything more, introduce yourself to other group members. Tell everyone else:

- your name/what to call you
- where your hometown is, or where you went as an undergrad
- what you're most hoping to get out of this course

Groups:

Group 1:
Amelia Camino
Jake Magee
Amanda Muratore
Julissa Sarmiento

Group 2:
Finian Ashmead
Francis Burk
Mykola Chernyashevskyy
Mohamed Ismail
Ehteshamul Karim

Group 3:
Cullen Abelson
Alia Dawood
Mira Salman
Yunchong Zhang

Group 4:
Nathalie Chicoine
Lauren Elicker
Yoki Salcedo
Marcos Tamargo-Arizmendi

Review: what do we mean by "probability"?

- In a well-controlled situation, we can define the *probability* of an event as the fraction of times it will occur if we infinitely repeat an experiment - i.e., its *frequency*.

$$P(x) = \lim_{n \to \infty} \frac{n_x}{n_t}$$

- This is the oldest definition of probability - the "frequentist" view - but not the only one possible (more on that later). Note P is at most 1.

What we just did: testing the results from coin flips

- Consider flipping two coins. The probability that both coins give the same result is 1/4+1/4= 1/2 ; if we flip coins an infinite number of times, half the time we'd get this result.
- In the Lecture 2 notebook, we made a function, `sim2coins,` which emulates this process, put in a python file (`sim2coins.py`), and imported it as a module (`s2c`)
- Today, we'll be using the Lecture 3 notebook and build on that.

Review: generating random numbers

- `numpy.random.rand()` generates random numbers, evenly distributed between 0 & 1.
- The syntax is:

```
import numpy.random as random
# only need to do that once, at the beginning of your program
randnum=random.rand(ndim1 , ndim2 , ndim3...)
```

- where `ndim1`, `ndim2`, etc. are the size of the 1st, 2nd, etc. dimensions of the array

Review: simulating many flips

- We can do `ntests` flips of 2 coins, and determine what fraction of the time both coins come up the same way, by putting the following in a file named `sim2coins.py` and then importing it:

```
import numpy.random as random
import numpy as np
def sim2coins(ntests):
# simulate ntests tosses of 2 coins
    ntests = int(ntests)
    coin1=random.rand(ntests) > 0.5
    coin2=random.rand(ntests) > 0.5
    return np.sum( coin1 == coin2 )*1./ntests
```

- We can import the module under the name `s2c` with:

```
import sim2coins as s2c
```

- Note that once coin1 or coin2 is bigger than the available RAM space, the computer will slow down as it is using the HD/SSD for memory
- If your computer has a few GB of memory, `ntests` values >~2E8 or so may be problematic

Time to move on to the lecture 3 notebook!

Suppose we make a change...

- Let's see, instead, how often coin1 is 0 (=tails) and coin2 is 1 (=heads).
- **Edit your sim2coins.py file to return the fraction of cases where that happened.** We have to do this a little differently: evaluating `(coin1==False and coin2==True)` will generate an error.
- Instead, we need to use `np.logical_and(),` which applies the `and` operation to *each element* of two arrays, element by element. So change the last bit of the function:

```
tails_and_heads = np.logical_and(coin1==False, coin2==True)
return ???
```

- Save the file
- Then, at your Python prompt do:

```
print(s2c.sim2coins(1000))
```

- and see what you get!

Why didn't that work?

- Python will automatically compile a module the first time you import it. However, to save time it won't automatically recompile a routine after that. We have to force it to.
- We can do that with the commands:

```
from importlib import reload
reload(s2c)
```

  - (Note:If you try doing `reload(sim2coins)` you'll find it doesn't work!)
- After recompiling, run it again...

How do our results depend on the number of trials?

- Let's try:

```
nsims_list=np.array([100,500,1000,5000,1E4,5E4,1E5,1E6])
result=nsims_list*0.
for i,nsims in enumerate(nsims_list):
    result[i]=s2c.sim2coins(nsims)
```

- Now, plot the fraction of successes as a function of the number of simulations.
- It would be helpful to use a logarithmic x axis: you can do this by:

```
plt.semilogx(x,y[ plot symbols, optional keywords, etc.)
```

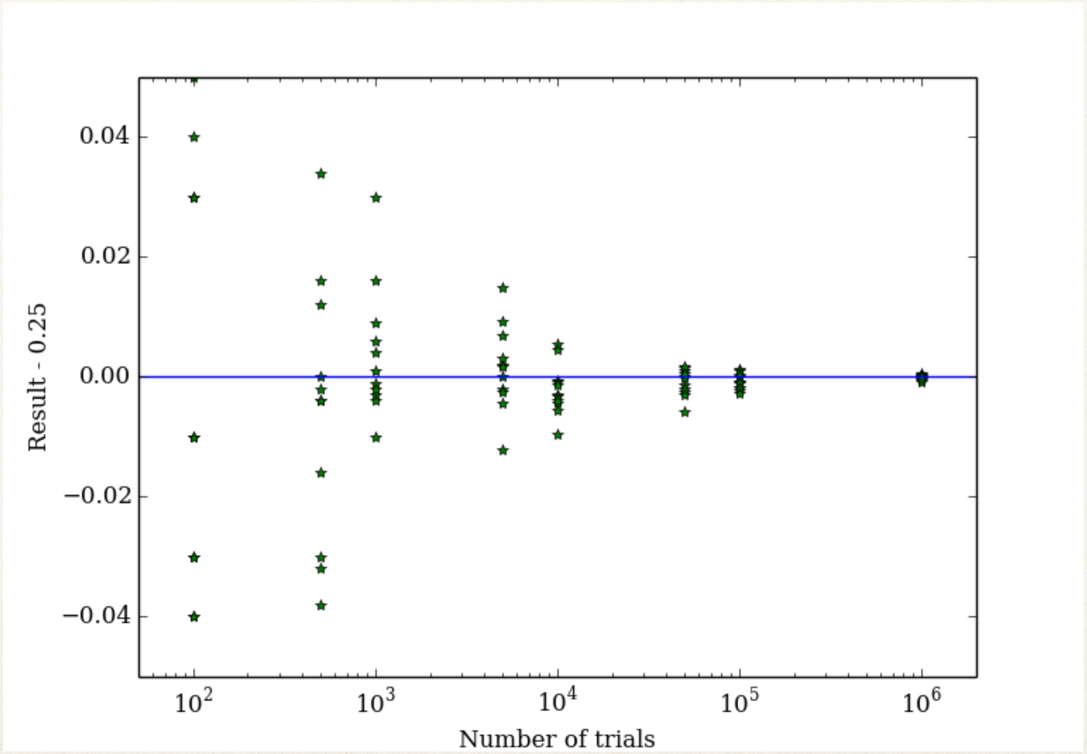
- Or, for an existing plot, you can convert to a logarithmic x axis with:

```
plt.xscale('log')
```

How do our results depend on the number of trials?

- **1) Plot the result - 0.25 (i.e., deviation from the expected probability) as a function of nsims**
  - Plot the points as green stars (look at the help on `plt.plot()`).
  - Use a logarithmic x axis.
  - Use a y axis range from -0.05 to +0.05 (look at the help on `plt.ylim()`)
- **2) By adding another, outer for loop, repeat this 20 times, overplotting all the results.**
  - If you put all the plot commands in the same code box, all the plots will be shown on the same axes, as we want.
- **3) Overplot the line y=0 to help guide the eye.**

Results from doing this 20 times:



As nsims gets larger, our results get closer & closer to 0.25

This convergent value defines the probability in the frequentist view

- As `nsims` gets larger, our results stay closer & closer to 0.25 .
- So the more trials we do, the fraction of times that coin1 is 0 and coin2 is 1 gets closer and closer to 1/4.
- In the *frequentist* definition, the *probability* of the event is the fraction of times it will occur if we repeat the experiment an infinite number of times - so that probability is 1/4, just like we all would have expected.
- This is not the only possible definition of probability - as we'll find out later.

More Monte Carlos: Simulating 6-sided dice

- An ordinary die has 6 sides, numbered from 1 to 6. How do we get that from a random number between 0 and 1?
- We want to convert the random number to an integer value. There are three ways to do this conversion:
    - `np.floor` (the largest integer below a #)
    - `np.round` (by default, rounds to nearest integer)
    - `np.ceil` (the smallest integer above a #)

- Note that the results of these routines will have data type float, not int !
- We want a result from 1 to 6...

3 ways to do this

```
nsims=1000
```

- Floor:

```
rolls_f=np.floor(random.rand(nsims)*6) + 1
```

- Round:

```
rolls_r=np.round(random.rand(nsims)*6 + 0.5)
```

- Ceil:

```
   rolls_c = np.ceil(  ???  )
```

- All 3 of these should give equivalent results (check the minimum and maximum values from each)! If you think about what numbers you get out if the random # is 0.01 or 0.99, you should be able to convince yourself of this.

Let's test it:

- How often do we 'roll' a 1, 2, 3, etc.?
  - For an ordinary die, each possibility is equally likely, and will occur one-sixth of the time.
  - In Python, the `plt.hist()` function allows us to check this easily.
- **Bring up the help on `plt.hist()` using ?.**
- To apply it:

```
plt.hist(rolls_f)
```

Adjusting the histogram

- By default, `plt.hist()` uses 10 bins ranging from the minimum to maximum of the array.
- We can use the `bins` keyword to set the number of bins, and the `range = (min,max)` keyword to set the minimum and maximum to use (note that `(min,max)` is a tuple).
- **1) Using these keywords, adjust your plot to use 6 bins, centered at 1, 2, 3, 4, 5 and 6.**

Simulating 10 dice: multidimensional arrays

- A numpy array need not have only one dimension. E.g.:

```
img = zeros( (200,200) )
```

will create a 200 x 200 array, with zeros everywhere.

- `np.zeros()` and similar routines can take a tuple of dimension sizes as input, for arbitrary numbers of dimensions.

Simulating 10 dice

- Let's start with:

```
rolls_f=np.floor(random.rand(nsims)*6) + 1
```

- Remember that `random.rand()` can create arrays of arbitrary dimensions:

```
randnum=random.rand(ndim1 , ndim2 , ndim3...)
```

so:

```
rolls=np.floor(random.rand(nsims,10)*6 ) + 1
```

would be equivalent to doing `nsims` different simulations of rolling a total of 10 dice.

Simulating 10 dice: the slow way

```
nsims = int(2E4)
rolls=np.floor(random.rand(nsims,10)*6 ) + 1
```

- is equivalent to doing `nsims` simulations of rolling 10 dice.
- If you want to find the total of the 10 rolls for each simulation, you could do:

```
total_roll=np.zeros(nsims)
for i in np.arange(nsims):
     total_roll[i]=np.sum(rolls[i,:])
```

- Then you can do `plt.hist(total_roll)` to see the results of your simulation... **adjust the number of bins and range as necessary to show all the values in the array** (you may find `np.min()` and `np.max()` helpful).

We can do this more efficiently...

- A trick using `np.sum`:
- `np.sum(array, axis = dim)` will return the total of `array` over the `dim`th dimension (where `dim` could be 0, 1, 2,etc.). If no axis is provided, it sums the entire array.
- So you could replace the lines:

```
total_roll=np.zeros(nsims)
for i in arange(nsims):
    total_roll[i]=np.sum(rolls[i,:])
```

with one short, and much quicker (by a factor of hundreds), line:

```
total_roll=np.sum(rolls,axis=1)
```

Let's try simulating 2,5,10,100 dice...

- We don't need to actually do these simulations separately - we can just simulate once, and take different subsets. If we do:

```
nsims=int(2E4)
rolls=np.floor(random.rand(nsims,100)*6 ) + 1
```

We could get the results for 5 rolls with:

```
for i in arange(nsims):
     total_roll[i]=np.sum(rolls[i,0:5])
```

or with:

```
total_roll_5=np.sum(rolls[:,0:5],axis=1)
```

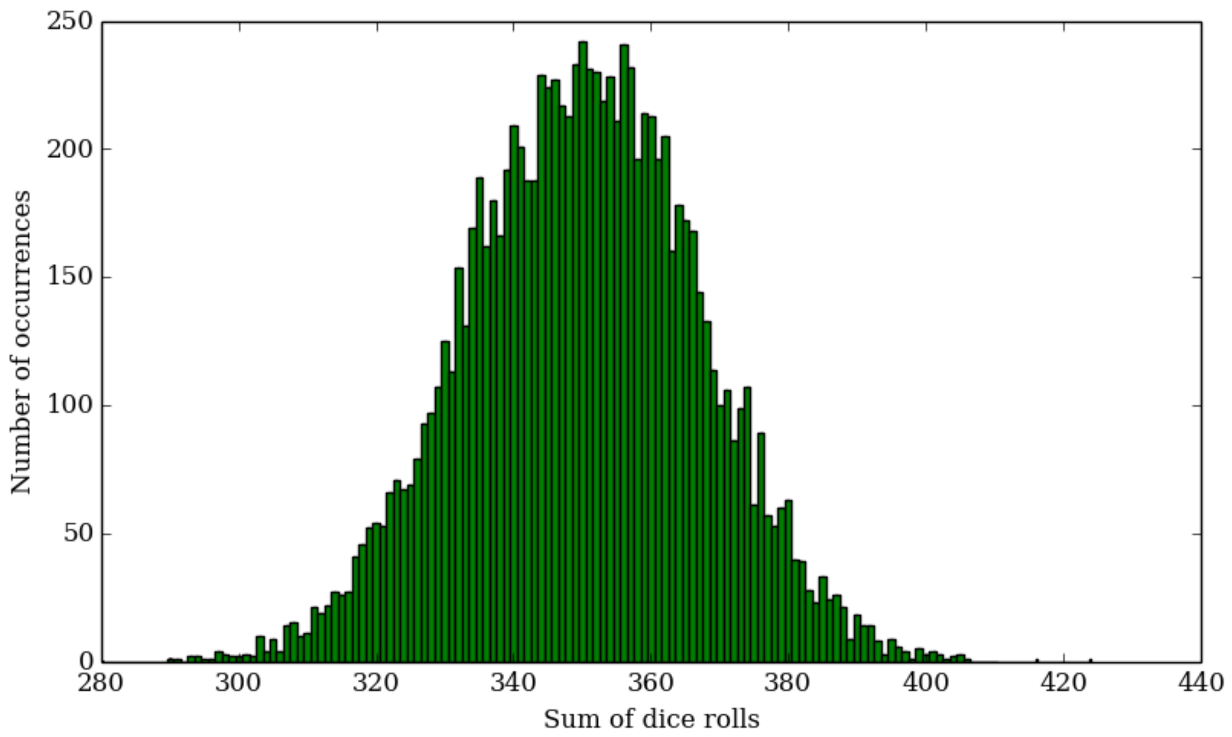
- or we could just proceed directly to plotting each case all in one line:

```
plt.hist(np.sum(rolls[:,0:5],axis=1) )
```

- **Plot up histograms for 2,5,10, & 100 rolls.**

Results

This sort of curve hopefully looks familiar - it is very close to a Gaussian distribution. We'll talk about them more later.



The Central Limit Theorem states that (with modest assumptions) the sum (or average) of a large number of random variables will approach a Gaussian distribution. Here we see it in action, starting with a totally flat, coarse-grained distribution.

Putting output in a file

- We often want to put output in a permanent file, not to the screen.
- The matplotlib routine `plt.savefig(filename)` does this.
- Depending on the filename given, savefig will write the current plot as a png, pdf, ps, eps or svg file.
- E.g.: `plt.savefig('spam.pdf')` will write the output in a PDF file named "spam.pdf"

Viewing the file

- On a Mac, you can view the file by navigating to it in the Finder (the face icon on the left side of your Dock), or else at a terminal prompt issue the command:

```
open spam.pdf
```

- Anything you can do at a prompt, you can do in jupyter/ipython (including notebooks) by putting a '!' first:

```
!open spam.pdf
```

- On a linux machine, use `gv` instead of `open` .

Homework

Homework is due next Friday, Jan. 31, via Canvas.

Problem 1 (of 1):

1A) Make a module containing a new function, `simncoins()`, that takes as input the # of coins to flip per simulation, ***ncoins,*** and the total number of simulations ***nsims***; and returns back an array containing the total number of heads from flipping *ncoins* coins, with one element for each of the *nsims* simulations (so the array contains the results of *nsims* sets of coin flips).

Not necessary, but a good idea in general: You can test your code by calculating for only 2 coins, see how often you get 2 heads (or 1 head, or 0 heads), and compare to your expectation.

Homework

1B) Consider flipping 2, 4, 8, 16, 32, 64, or 128 coins. Make separate plots with histograms of the total number of heads that you get (produced from 50,000 trials, i.e. nsims=50_000), for each number of coins.

To do this, you will need to save your plots; I would prefer PNG format (remember to provide your code as well though!)

I will set up an assignment in Canvas for you to turn in your results.

Another view of probability

- We've seen that in the frequentist view, probability is just the fraction of times an event happens out of an infinite number of trials.
- We can get very similar results with a very different definition: if we take probability to represent our ***state of knowledge*** that something will occur or that something is true; sometimes called its 'plausibility'
    - It is possible to define logic so that not just 0=false and 1=true, but values between 0 and 1 work too.
    - Alternatively, in some formulations, probability is taken to indicate our degree of belief.
- These ideas were formalized in the ~1950's, but are related to methods developed by Rev. Thomas Bayes in the 1700's; as a result, this is referred to as the ***Bayesian*** definition of probability.

Applying this concept

- Consider flipping a coin again.
- If we know it is fair, then we would expect that one-half of the time we will get heads, and one-half of the time we will get tails.
- So we would assign the same probability to the two events: 1/2 - and so would, presumably, any other person.

Odds

- Another way of expressing probability is as the *odds* on an event: the probability it occurs, divided by the probability it does not occur (i.e., $\dfrac{p}{1-p}$)
- So the odds of getting heads is 1 to 1 (or 1.0) .
- The odds of rolling a die and getting 2 would be 1 to 5 (or 0.2): it is 5 times more likely we roll something besides 2 than 2.
- This is the inverse of typical gambling odds; e.g. something that happens one-fourth of the time would "pay 3 to 1".

What role does belief have in science?

- Many people are uncomfortable with the Bayesian view of probability:
  - ☐ Our goal as scientists is to be impartial judges, right?
  - ☐ Different scientists might have different beliefs about what the probabilities might be - in the Bayesian view there may not be a single possible probability (vs. the frequentist view), but instead each of us has our own probability for event X!
  - ✓ However, we often think about data in Bayesian ways (e.g., what should I conclude about the true magnitude of an object if I observe that it is 20+/-0.1)?
  - ✓ In practice, using the Bayesian vs. frequentist definition can make little difference - e.g., the probability on a coin flip is still 1/2 in each view.

Many things work out the same in both Bayesian & Frequentist views

- R. T. Cox showed that if a definition of plausibility follows some simple assumptions - e.g. *p(A is false) = 1-p(A)*; if *p(A) > p(B)* and *p(B) > p(C)* then *p*(*A*) > *p*(C); and if plausibility depends only on the information received, not order - it will lead to the "Kolmogorov" axioms of probability theory:
  - Any random event has probability *p* between 0 and 1
  - An event that is certain to occur has *p*=1; the total probability that some event occurs is 1
  - If A and B are mutually exclusive (i.e., both A and B cannot be true simultaneously), then *p*(A OR B)=*p*(A)+*p*(B)
- These axioms allow us to manipulate probabilities, define how they combine, etc.

Independence and conditional probabilities

Commonly treated as (**important**!) definitions:

- ➡If events A and B are ***independent*** - i.e., whether A is true has no relation to whether B is true (i.e., what we know about B doesn't affect what we expect for A) - then: *p(A* AND *B)=p(A) x p(B)*
- ➡ Many things aren't independent. For instance, if it rains today, it is more likely (than if you averaged all days) that it will rain tomorrow. The ***conditional probability*** that A will be true, given that B is, turns out to be: *p(A|B)=p(A AND B)/p(B)*
- ➡Note that if we combine these, we find that *p(A|B)=p(A)* if - and only if - A and B are independent.

Manipulating probabilities

- If events A and B are mutually exclusive, then *p(A OR B) = p(A) + p(B)* .
- What if they aren't? In general, *p(A OR B) = p(A) + p(B) - p(A AND B)*
- For instance, suppose it is 50% likely that when a particular couple have a child it will be a boy, and 50% likely that any child they have will have red hair.
- Since these are independent, we calculate *p(red-haired boy)* = ½ x ½ = ¼ , and *p(red-haired OR boy)* = ½ + ½ - ¼ = ¾ .
- We could have figured this out by counting equally probable cases:

boy, red hair	girl, red hair
boy, brown hair	girl, brown hair

Marginalization

- Let's suppose there are a finite number of possible results for event B, $\{B_i\}$. E.g.: if we flip a coin, B='the coin came up heads' can be true or false, which we could call B_0 and B_1.
- It's possible to show that:

$$p(A) = \Sigma_i p(A\,|\,B_i)\ p(B_i) = \Sigma_i p(A\ \ AND\ \ B_i)$$

- We call this ***marginalization***: we have probabilities for A & B, but can use those to just get out probabilities for A.
- This can be extremely useful: e.g., if we want the probability distribution of the cosmological parameter Ω_m , irrespective of the value of another parameter *h*, when we know $p(\Omega_m\ AND\ h) = p(\Omega_m, h)$, we can get it from the continuous version of this:
  - $p(A) = \int p(A\,|\,B)\ p(B)\ dB = \int p(A, B)\ dB$

Example

- Let A= 'we roll 5 on a die', B_1='the roll of that die is even', and B_2='the roll of that die is odd'.
- Does $p(A) = \Sigma\, p(A \mid B_i)\, p(B_i)$?

- We know p(A) = 1/6. If the result of a die is even, A is impossible, so $p(A \mid B_1) = 0$; while if the result is odd, A will occur one-third of the time (out of the equally likely possibilities 1, 3, 5), so $p(A \mid B_2) = \frac{1}{3}$.
- It is equally likely that the roll is even or odd, so we'd get:

$p(A) = (0)(\frac{1}{2}) + (\frac{1}{3})(\frac{1}{2}) = \frac{1}{6}$,

- as expected!

An astronomical example

- Suppose 40% of early-type (elliptical or 'lenticular') galaxies have AGN (actively accreting supermassive black holes), and 10% of late-type (spiral/irregular) galaxies have AGN: i.e., $p(A \mid E) = 0.4$, $p(A \mid L) = 0.1$. Further assume that galaxies are an even mix of early-and late-type: $p(E) = p(L) = 0.5$.
- **What is the probability that a randomly-chosen galaxy harbors an AGN?**
- Try to reason intuitively first, and then apply the formula:

  $p(A) = \Sigma\, p(A \mid B_i)\, p(B_i)$

  If you need a calculator, use python! e.g.: `0*(1/2)+(1/3)*(1/2)` for the previous problem...

Summary: key rules of probability to remember

- ➡ If events A and B are ***independent*** then: $p(A \text{ AND } B) = p(A) \times p(B)$
- ➡ The ***conditional*** probability A is true, given that B is true: $p(A|B) = p(A \text{ AND } B)/p(B)$
- ➡ $p(A|B) = p(A)$ if - and only if - A and B are independent.
- ➡ In general, $p(A \text{ OR } B) = p(A) + p(B) - p(A \text{ AND } B)$
- ➡ ***Marginalization:*** $p(A) = \Sigma\, p(A|B_i)\, p(B_i)$

Bayes' Theorem

- An important result comes from setting:

  $p(A \text{ AND } B) = p(B \text{ AND } A)$

- so:

  $p(A \mid B)\, p(B) = p(B \mid A)\, p(A)$

- so:

  $p(B \mid A) = p(A \mid B)\, p(B) / p(A)$

- This last statement is known as Bayes' Theorem, after its discoverer (in the 1700's).
- Despite having Bayes in the name, it is equally valid in both Bayesian and frequentist views of probability.

Breaking it down

$p(B|A) = p(A|B)\,p(B)\,/\,p(A)$

- Let's let:

  B = the true value of some set of parameters (some or all of which we want to know), and

  A = the observed set of data

- Then we call:

  $p(B|A)$: the ***posterior probability*** : i.e., the probability we'd conclude for B after applying Bayes' theorem

  $p(A|B)$: the ***likelihood*** : i.e., how likely is it we'd get A in scenario B

  $p(B)$: the ***prior***: our guess at what the values of B might be, in the absence of experiment A

What about $p(A)$?

$$p(B|A) = p(A|B)\,p(B)\,/\,p(A)$$

- You may notice that the book doesn't really talk about $p(A)$ (which is sometimes called the "evidence").
- That's because it doesn't matter in many calculations - it's basically a normalization factor.
- We could construct it, though, using a definition we encountered before:

$$p(A) = \Sigma\, p(A|B_i)\, p(B_i)$$

i.e., marginalizing over all possible values of B_i.

- The evidence is sometimes used to compare the effectiveness of different models in describing data.

How does this relate to Bayesian probability?

- Let's take probability to refer to our level of belief.

- Then Bayes' theorem tells us how to **update** our beliefs based upon some set of observed data. The prior in fact represents our prior beliefs about the possible distribution of values for B.

$$p(B \mid A) = p(A \mid B)\, p(B) / p(A)$$

Bayesian vs. frequentist analyses

- Frequentist calculations often focus on how often we would get the observed result, given some presumed true situation (=hypothesis).
- A Bayesian calculation would focus on how probable we find different possible true situations to be, given the observed result.
- Notice that, if $p(B)$ and $p(A)$ are constant, we just have:
    - $p(B \mid A) \propto p(A \mid B)$
- In many cases, the inference will be same whether we work from a Bayesian or frequentist perspective!

So how do the two views of probability differ?

- Much of the difference is not in whether they accept Bayes' theorem - but in how seriously they take it.
- Bayes' theorem requires a prior - but assigning a prior generally is a subjective choice (there are some rules of thumb).
- In many cases, the choice of prior doesn't make much difference.
- There are problems that are only really solvable in the frequentist view, and others that only work out from Bayesian assumptions.
- It is often most obvious how to pose a problem in the Bayesian view, so we'll generally be following that.

Why has Bayesianism become more common recently?

- Although much of the framework was developed at about the same time as classical statistics, Bayesian methods fell by the wayside.
- This is mostly because it is computationally harder - we have to integrate over more complicated functions (thanks to the prior), which may not be Gaussian.
- These days, numerical integration can handle almost arbitrarily complex scenarios easily.

Back to our example case

- Again, let's suppose 40% of early-type galaxies have AGN, and 10% of late-type galaxies have AGN. Further assume that galaxies are an even mix of early-and late-type.
- You find a particular galaxy in an X-ray catalog, letting you know it's an AGN.
- Is it more likely to be an early-type galaxy or a late-type galaxy? If someone bets you $10 that it's a late-type galaxy (so you win $10 if it's early-type, and lose $10 if it's not), would you take the bet?

- **Discuss with your groups!**

$$p(B|A) = p(A|B)\,p(B)\,/\,p(A)$$

Bayesian (and frequentist) techniques can be applied to problems well outside of science

- Go to: https://web.archive.org/web/20201009065503/https://election.princeton.edu/2020/06/19/its-alive-2/
  - used the average of polling in each state, together with the nominal statistical uncertainties and an estimate of extra uncertainty, to predict a range of election outcomes
  - uses Monte Carlo simulations based on each poll's results to get predictions
  - basically a pure-frequentist implementation, plus a model of how uncertain polls are at some point in time at predicting results in November
- Actual result gave 306 electoral votes to Biden, https://www.270towin.com/2020_Election/interactive_map

We can think of expert opinion as being the equivalent of a strong prior:

- Go to: https://www.270towin.com/maps/cook-political-2020-electoral-ratings
    - experts look at each state race on its own
    - polls are only one ingredient used to make judgements; past experience is key
    - good track record; e.g. races listed as 'toss-ups' by Cook Report in the past have, on average, been won ~50% by Republicans, ~50% by Democrats
    - priors are one way of encoding expert knowledge: in the absence of other information, previous experience of similar years leads to judgements of how likely each candidate is to win

Hybrid techniques are also possible

- Go to: https://projects.fivethirtyeight.com/2020-election-forecast/
  - uses a model incorporating national and state polling, how similar different states are, biases of different pollsters compared to the average, etc., to predict a range of possible results (using Monte Carlo simulations of how far off each poll could be). Priors are based on economic conditions.
  - There's some amount of subjective choice in modeling. What properties of a state define 'similarity'? To what degree will this election be like past elections? The recipe the site uses has changed over time.
  - The models are tuned so that they give the right rates of success for past elections.
  - There are now a number of forecasters doing this sort of thing; see https://projects.economist.com/us-2020-forecast/president for an open-source equivalent.

Let's generalize from coins and dice to more complicated situations...

- We'll come back to applications of Bayes' theorem, but need more background to go further.
- Till now, we've mostly talked about probabilities with just a few possibilities, equally likely. However, we can instead consider arbitrary probabilities as a function of continuous variables: a ***probability density function*** or PDF (for functions of discrete variables, the term is ***probability mass function*** or PMF).
- Probability density functions just have to be nonnegative everywhere, and integrate to 1 (or have sum 1 in the discrete case). For a continuous PDF $f(x)$,

$$p(a < x < b) = \int_a^b f(x)\, dx$$

- Sometimes, the term ***probability distribution*** is used instead of PDF.

More on distributions

- It is sometimes helpful to look at the fraction of the integral of *f(x)* which is below some value. This is the ***cumulative distribution function*** or CDF, generally written *F(x)*:

$$F(x) = \int_{-\infty}^{x} f(y)\, dy$$

- Another useful thing to look at can be the expectation value of some function *g(x)*:

$$\mathbf{E}g = \int_{-\infty}^{\infty} g(x)\, f(x)\, dx$$

- The expectation value is the value of *g(x)*, weighted by the probability of each possible x.

Expectation values

- Ex.: suppose I'll pay you \$5 if a coin comes up heads, and you pay me \$5 if it comes up tails. What is the expectation value of your winnings after a coin toss?
- Some important expectation values are based on the moments of the distribution (remember basic mechanics...):
  - The ***mean*** is, simply, the first moment of f(x) - compare to the center of mass in mechanics. It provides a measure of the ***location*** or ***center*** of *f(x)*.

$$\mu = \mu_1 = \mathrm{E}x = \int_{-\infty}^{\infty} x\, f(x)\, dx$$

Variance and standard deviation

- We often are interested in the width, not just central value, of a probability distribution (e.g. an error bar). We can measure this with a second moment (compare to the moment of inertia about the center of mass):

$$\sigma^2 = \mu_2 = \mathrm{E}(x-\mu)^2 = \int_{-\infty}^{\infty} (x-\mu)^2 \; f(x) \; dx$$

- In statistics, μ_2 is called the ***variance,*** and is equal to the ***standard deviation*** squared. σ provides a measure of width in the same units as x.
- We can also construct statistics to describe other moments: the ***skewness*** = μ_3 / σ^3 = $\mathbf{E}(x-\mu)^3 \;/\; \sigma^3$ describes how asymmetric a distribution is, the ***kurtosis*** = μ^4 / σ^4 describes how flattened it is, etc. A Gaussian distribution has 0 skewness and kurtosis of 3.