



Bayesian Methods

CSIT 2023/24

Plan

Probability – a quick review

Basic applications of the Bayes Theorem

Bayesian Methods with PyMC

Bayesian Networks with GeNIe

Probability

A QUICK REVIEW

Probability – what is it?

Classic definition (Bernoulli, Laplace): ratio of the number of propitious outcomes to the number of all outcomes, assuming all outcomes are equally possible, $P(A) = \frac{N_A}{N}$

Frequentist: assuming infinite number of trials, it is the ratio of number of occurrences of the outcome to the number of trials (relative frequency), $P(A) = \lim_{n \rightarrow \infty} \frac{n_a}{n} = \lim_{n \rightarrow \infty} f_n(A)$

Subjectivist: one's belief in the outcome of the experiment, based on prior knowledge and experiment circumstances

Basic terms

Experiment (doświadczenie losowe): process or phenomenon that gives information

Outcome (wynik): single piece of information coming from the experiment

Sample space (przestrzeń zdarzeń): set of all possible outcomes of the experiment

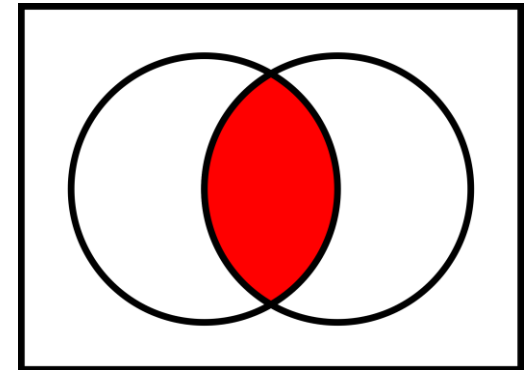
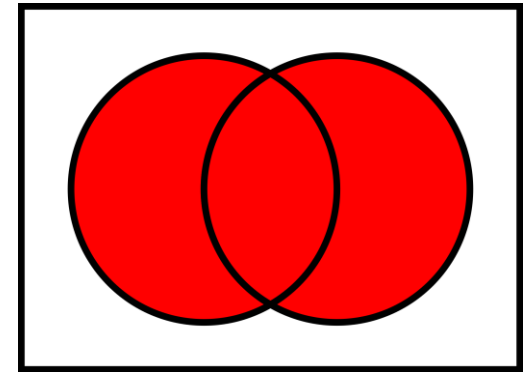
Event (zdarzenie): what comes from the experiment

- Every outcome is an event – it is an **elementary** event
- Event may contain many outcomes – it is a **composite** event

Events and sets

Union of sets: $A \cup B$ means that A occurred, or B occurred

Intersection of sets: $A \cap B$ means that A occurred, and B occurred



Probability axioms

For every event E belonging to the sample space S : $0 \leq P(E) \leq 1$

$$P(S) = 1$$

If $E \cap F = \emptyset$ then $P(E \cup F) = P(E) + P(F)$

Conditional probability

If E and F are events in S and $P(F) > 0$, then the conditional probability of E given F is defined by

$$P(E|F) = \frac{P(E \cap F)}{P(F)}$$

E.g., what is the probability of passing the exam given attending classes? It is the probability of both passing and attending divided by the probability of attending.

Independence of events

Events are independent if $P(E \cap F) = P(E) \cdot P(F)$

If $P(F) > 0$ both sides can be divided: $\frac{P(E \cap F)}{P(F)} = P(E)$

Using definition of conditional probability $P(E|F) = P(E)$

Are the events of passing an exam and eating doughnuts independent?

Bayes theorem

$$P(E_j | A) = \frac{P(A|E_j) \cdot P(E_j)}{\sum_i P(A|E_i) \cdot P(E_i)} = \frac{P(A|E_j) \cdot P(E_j)}{P(A)}$$

Events E_1, E_2, \dots, E_N partition the sample space: $E_1 \cap E_2 \cap \dots \cap E_N = S$, $E_i \cap E_j = \emptyset$ for $i \neq j$

Random variable

Function which assigns numerical value to every possible outcome of an experiment

- Denoted by big letters, e.g., X

For each repetition of an experiment only one value of the random variable – its **realization** – is observed

- Denoted by the corresponding small letter

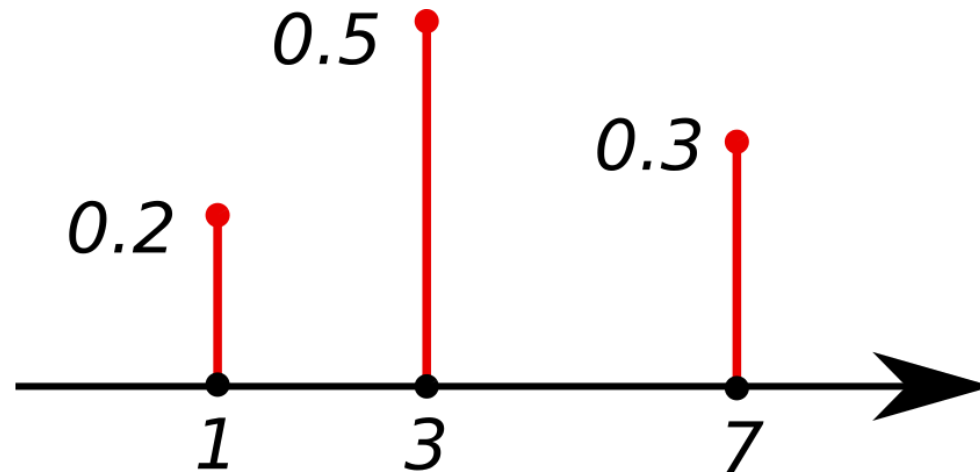
Based on the set of possible outcomes can be

- Discrete
- Continuous

Probability distribution

For a discrete random variable, it is possible for every $x \in R$ to define $p(x) = P(X = x)$

Collection of pairs $\{(x, p(x)), x \in R\}$ is called probability distribution (probability mass function)



Cumulative distribution function

For a discrete random variable, it is defined as

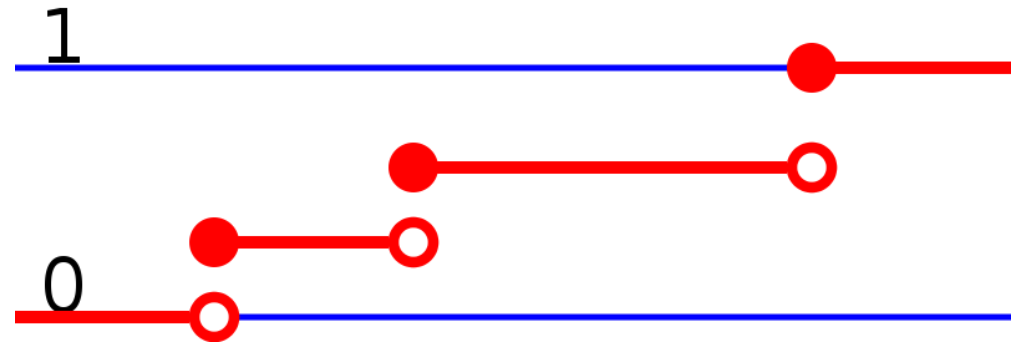
$$F(x) = P(X \leq x) = \sum_{y \in R: y \leq x} p(y)$$

Basic properties:

$$F(-\infty) = P(X \leq -\infty) = 0$$

$$F(\infty) = P(X \leq \infty) = 1$$

$$\text{for } x_1 \leq x_2, F(x_1) \leq F(x_2)$$



Expected value

For discrete random variable X with distribution $p(x)$ the expected value is defined as

$$E(X) = \sum_{x \in R} x \cdot p(x)$$

The expected value does not have to belong to the values of the random variable!

Bernoulli distribution

The random variable may take only two values: 0 or 1

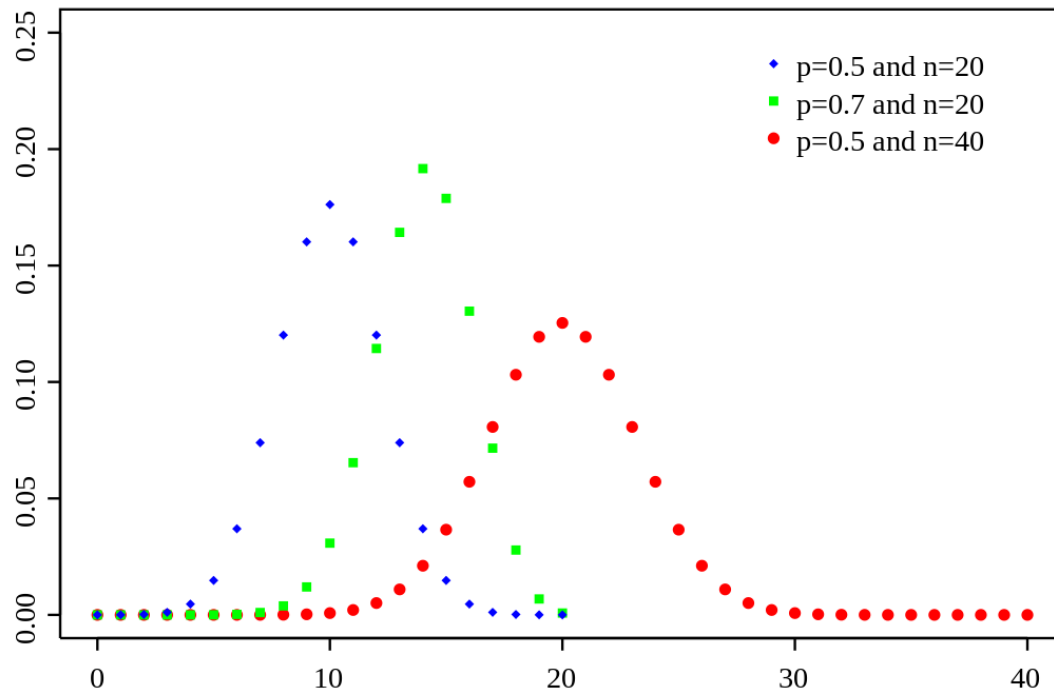
$$P(X = 1) = p$$

$$P(X = 0) = 1 - p$$

$$E(X) = p$$



Binomial distribution



Results from n Bernoulli trials:

- Each trial has only two possible outcomes: success or failure
- Results of subsequent trials are independent
- In each trial probability of success is p

Random variable X describes the number of successes in n Bernoulli trials

$$P(X = x) = \binom{n}{x} p^x (1 - p)^{n-x}$$

$$E(X) = np$$

Poisson distribution

Events occur in continuous time or space

Number of events in non-overlapping periods of time or areas is independent

Events occur individually

Events occur with constant average speed or density λ

Random variable X describes number of events in time period or area of interest

$$P(X = x) = e^{-\lambda} \lambda^x / x!$$

$$E(X) = \lambda$$

Probability density function

For a continuous random variable, for every $x \in R$ $p(x) = 0$

Probability mass function makes no sense

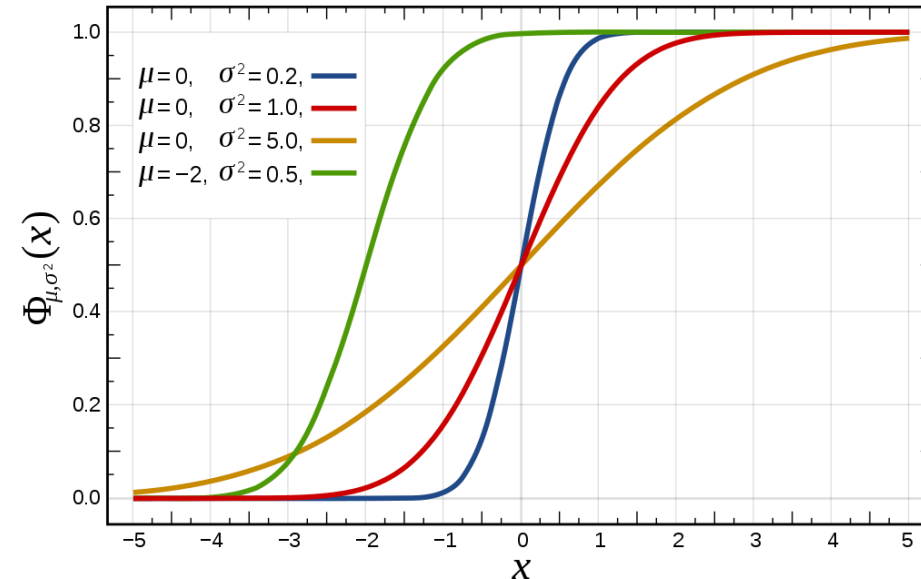
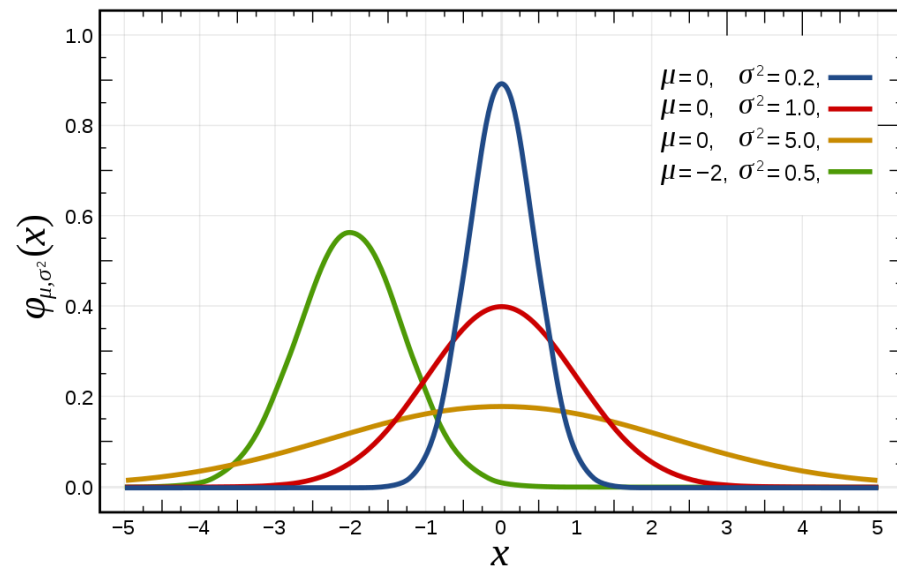
Instead, probability density function is used, derivative of the cumulative distribution function

$$f(x) = \frac{d}{dx} F(x)$$

Normal (Gauss) distribution

PDF given by $f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left[-\frac{1}{2} \left(\frac{x-\mu}{\sigma} \right)^2 \right]$

The parameters are mean and standard deviation (or variance)



Basic applications of the Bayes Theorem

THOMAS BAYES (1701 – 1761) WAS AN ENGLISH STATISTICIAN, PHILOSOPHER AND PRESBYTERIAN MINISTER WHO IS KNOWN FOR FORMULATING A SPECIFIC CASE OF THE THEOREM THAT BEARS HIS NAME (FROM WIKIPEDIA).

T. Bayes.

Reasoning backwards

Most people, if you describe a train of events to them, will tell you what the result would be. They can put those events together in their minds, and argue from them that something will come to pass. There are few people, however, who, if you told them a result, would be able to evolve from their own inner consciousness what the steps were which led up to that result. This power is what I mean when I talk of reasoning backwards, or analytically.

Sherlock Holmes
(Arthur Conan Doyle, *A Study in Scarlet*)

Reasoning backwards

If I throw a stone at this window, what are the chances that it will break?

Here's a broken window, what are the chances it was broken by a stone?

If she flies airline X, what are the chances the flight will be delayed?

Her flight was delayed, what are the chances she was flying airline X?

If he has flu, what are the chances he has cough?

He coughs, what are the chances he has flu?

Reasoning backwards

Let there be causal relationship between events C (cause) and E (effect).

$$p(E|C)p(C) = p(C|E)p(E)$$

People usually can estimate $p(E|C)$ from previous experience, it can be also obtained from typically gathered data

$p(C)$ and $p(E)$ are also often available due to previous observations

$p(C|E)$ is of particular interest, as it gives knowledge about causes when effects are observed

Reasoning backwards

Loss of smell is often present during COVID-19 infection. If a person losses smell, is she/he COVID-19 positive?

C – COVID-19 infection, L – loss of smell

$$p(C) = 237118/38268000 \approx 0.0062 \text{ (from active cases in Poland)}$$

$$p(L) = 0.01 \text{ (assumed, should be from active cases of smell loss)}$$

$$p(L|C) = 0.9 \text{ (numerical expression of the “often present” statement)}$$

$$p(C|L) = \frac{p(L|C)p(C)}{p(L)} \approx 0.558$$

Losing one's smell dramatically rises belief in being COVID-19 positive.

Reasoning backwards

And what if loss of smell was a very common complaint?

$p(L) = 0.1$ (order of magnitude greater than before)

$p(L|C) = 0.9$ (as before)

$$p(C|L) = \frac{p(L|C)p(C)}{p(L)} \approx 0.058$$

When the symptom is common (i.e., occurs also due to other complaints), its presence does not rise our confidence that much.

Reasoning backwards

And what if loss of smell was present only rarely during COVID-19 infection?

$p(L) = 0.01$ (initial value, i.e., loss of smell is not common)

$p(L|C) = 0.1$ (loss of smell is only present in 10% of COVID-19 cases)

$$p(C|L) = \frac{p(L|C)p(C)}{p(L)} \approx 0.062$$

If the symptom is not strongly tied to the disease in question, observing it does not rise our belief that much.

Reasoning backwards

Going back to the original numbers:

$$p(L) = 0.01, \text{ consequently, } p(\neg L) = 0.99$$

$$p(L|C) = 0.9, \text{ consequently, } p(\neg L|C) = 0.1$$

$$p(C|\neg L) = \frac{p(\neg L|C)p(C)}{p(\neg L)} \approx 0.00063$$

Not having symptoms lowers our belief in the illness.

Note: “not having symptoms” is fundamentally different from “having no data about presence of symptoms”!

Multiple evidence

A burglary, a dog and a silent alarm

In my country house I have a silent burglary alarm. When intrusion is detected, this alarm does not produce any output on-site, but instead calls my phone. Moreover, my country neighbour has a very watchful dog, who is good at detecting any movements or sounds and barks loudly.

Devise a model that will help me decide whether burglary occurred at my country house, taking into account all available evidence.

Independence of evidence

In this example, we may assume independence of the evidence:

- The dog is not influenced by the alarm, as it is a silent alarm.
- Clearly, the alarm is also not influenced by the dog.

Note, that this is only true when the presence of burglary is known for sure (we know that either there is burglary, or not)!

Conditional independence

Consider the following example:

If electrician working in your house by mistake connects 400 V to your single-phase installation, it is very, very probable all of your electrical equipment will break down.

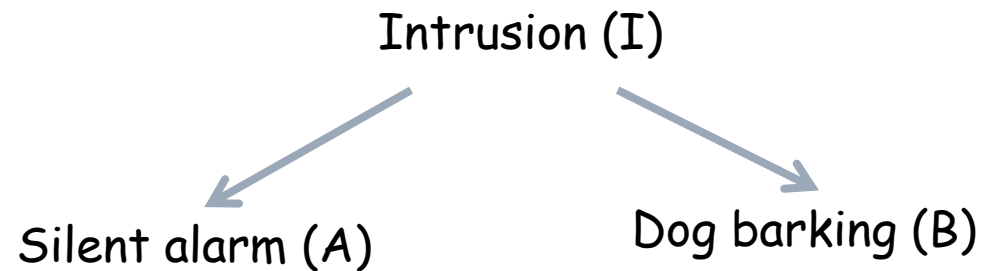
Definitely, if such a mistake is made, breaking down of, say, TV does not influence breaking down of, say, computer.

Situation A: you suspect the electrician made such a mistake (but are not sure). You observe broken TV. Does this influence your belief in the computer being broken?

Situation B: first, you learn that the electrician **did make** this mistake. Afterwards, you observe broken TV. Does this influence your belief in the computer being broken?

Depicting relationship

An informal schematic of the case may look like this:



Note lack of an arrow between (A) and (B) – this indicates conditional independence of the events.

Available data – “don’t know” is an option!

If thinking about events in term of (random) variables, each has only two possible states: *yes* and *no*

However, the model should also handle the “don’t know” situation – the variable is not observed, and we don’t know its state:

- The silent alarm calls our phone number, but we may have displaced our phone, or the battery might have died out
- It is even more obvious for the dog: we are not there, so we must call the neighbour and ask about the dog.

Model formulation

We are interested in $p(I|a, b)$ – a, b denote the observed values of events A and B

Applying definition of conditional probability $p(I|a, b) = \frac{p(I, a, b)}{p(a, b)} = \frac{p(a, b|I)p(I)}{p(a, b)}$

$p(a, b)$ – denotes the probability of events A and B *without reference to the model* – it is our **a’priori belief**. We assume that our belief in these events is independent, i.e., $p(A \cap B) = p(A)p(B)$. The values may be thought as the relative frequency of alarm going off and dog barking, respectively.

Similarly, $p(I)$ is our a’priori belief in there being intrusion.

$p(a, b|I)$ – denotes the probability of events A, B given I (intrusion). Because A and B are conditionally independent given I, $(A \perp B)|I$, $p(a, b|I) = p(a|I)p(b|I)$

Model formulation

$$p(I|a, b) = \frac{p(I, a, b)}{p(a, b)} = \frac{p(a, b|I)p(I)}{p(a, b)} = \frac{p(a|I)p(b|I)p(I)}{p(a)p(b)} = \frac{p(a|I)p(b|I)}{p(a)p(b)} p(I)$$

Let's write down conditional probability of I given *only one* piece of evidence, e.g., the alarm

$$p(I|a) = \frac{p(I, a)}{p(a)} = \frac{p(a|I)p(I)}{p(a)} = \frac{p(a|I)}{p(a)} p(I)$$

Putting into first equation $p(I|a, b) = \frac{p(b|I)}{p(b)} p(I|a)$

This means we may update our belief sequentially: we absorb one piece of evidence, then the updated belief becomes our a priori belief for the subsequent update.

Putting in some numbers

We need a priori probabilities: $p(a)$, $b(b)$, $p(i)$ and conditional probabilities $p(a|i)$, $p(b|i)$

Sample situation might look as follows:

$p(I) = 0.002$ (for my neighbourhood: average ratio of the houses that are burgled during the night to the total houses)

$p(A) = 0.01$ (for my alarm: ratio of the nights it goes off to the nights in the assumed time period; it goes off twice a night so rarely we may treat it as a single event)

$p(B) = 0.50$ (for my neighbour's dog: ratio of the days he barks during the night to the nights in the assumed time period)

$p(A|I) = 0.8$ (for my alarm: probability of it going off when burglary takes place)

$p(B|I) = 0.98$ (for my neighbour's dog: probability of it barking when burglary takes place; the dog is very watchful)

Getting results

Scenario 1

My alarm phoned me this night! What is the probability of burglary taking place?

$$p(I|A) = \frac{p(A|I)p(I)}{p(A)} = \frac{0.8 \cdot 0.002}{0.01} = 0.16 - \text{much higher than the prior of } 0.002$$

I'm nervous so I called my neighbour: he informs me that the dog barked last night

$$p(I|A, B) = \frac{p(B|I)}{p(B)} p(I|A) = \frac{0.98}{0.5} 0.16 = 0.3136 - \text{the probability did not go up that much, the dog is not reliable as he barks every other night}$$

Getting results

Scenario 2

My alarm phoned me this night! What is the probability of burglary taking place?

$$p(I|A) = \frac{p(A|I)p(I)}{p(A)} = \frac{0.8 \cdot 0.002}{0.01} = 0.16 \text{ – much higher than the prior of } 0.002$$

I'm nervous so I called my neighbour: he informs me that the dog did not bark last night.

$$p(I|A, \neg B) = \frac{p(\neg B|I)}{p(\neg B)} p(I|A) = \frac{0.02}{0.5} 0.16 = 0.0064 \text{ – the probability dropped enormously, it is very improbable the dog did not notice the burglars}$$

Getting results

Scenario 3

I just talked to my neighbour, he tells me his dog barked during the night. Should I worry?

$$p(I|B) = \frac{p(B|I)p(I)}{p(B)} = \frac{0.98 \cdot 0.002}{0.5} = 0.00392 - \text{not that different from the prior of } 0.002$$

Anyway, I double-checked the status of the alarm – it did not go off.

$$p(I|\neg A, B) = \frac{p(\neg A|I)}{p(\neg A)} p(I|B) = \frac{0.2}{0.99} 0.00392 \approx 0.00079 - \text{probability dropped, but not that much:}$$

my alarm is not really that sensitive.

Getting results

Scenario 4

I just talked to my neighbour, he tells me his dog barked during the night. Should I worry?

$$p(I|B) = \frac{p(B|I)p(I)}{p(B)} = \frac{0.98 \cdot 0.002}{0.5} = 0.00392 \text{ - not that different from the prior of } 0.002$$

Anyway, I double-checked the status of the alarm – it did go off.

$$p(I|A, B) = \frac{p(A|I)}{p(A)} p(I|B) = \frac{0.8}{0.01} 0.00392 = 0.3136 \text{ - as in scenario 1.}$$

Introducing PyMC3

[HTTPS://DOCS.PYMC.IO/](https://docs.pymc.io/)

What is it?

Python package for probabilistic programming (i.e., programming allowing to cope with distributions, random variables etc.)

Based on Theano

Employs MCMC (Markov chain Monte Carlo) methods to sample distributions – this allows to get samples from continuous variable distributions

- basic Mont Carlo methods sample distributions in completely random manner – for multi-dimensional problems this may be a problem
- MCMC is a guided way to sample distribution with emphasis on samples that most influence the computed output

Basic concepts

The basic building block is a **stochastic** variable:

```
x = pm.Normal("x", mu=0, sigma=1)
```

- defines **stochastic** variable with standard normal distribution

The variables must be contained in a model

```
with pm.Model() as model:  
    x = pm.Normal("x", mu=0, sigma=1)
```

Each variable has an initial value, that is the starting point of sampling

```
print(x.tag.test_value)
```

```
0.0
```

Basic concepts

PyMC3 works with stochastic and deterministic variables:

- value of a stochastic variable is not known even if all its parameters and components are known. Instances of distribution classes (Normal, Exponential, etc.) are examples of stochastic variables.

```
x = pm.Normal("x", mu=0, sigma=1)
```

- value of a deterministic variable is known exactly once its components are known. These are variables similar to the ones used in normal programming.

```
d = pm.Deterministic("d", x + 1)
```

- deterministic variables are also created by elementary operations, however, in such case the result is not stored

```
d2 = x + 1
```

Basic concepts

Using random method, it is possible to sample the distribution:

```
print(x.random(size=10))
```

```
[-1.32036024 -0.23443528 -1.5668316  0.15234133  0.75418639 -0.24177129  
 1.03009621  0.27025339 -0.37722446  0.30920738]
```

For any given argument(s) of a distribution, it is possible to obtain the log-probability value using logp method

- log-probability is just natural logarithm of probability, or, in case of continuous random variables, of the value of the pdf
- use of logarithms makes sense, as for the independent random variables $P(E \cap F) = P(E) \cdot P(F)$.
Logarithm transforms multiplication into addition, $\log(P(E) \cdot P(F)) = \log(P(E)) + \log(P(F))$

```
print(x.logp({'x':0}))
```

```
-0.9189385332046727
```

Example: investigate parameter of a distribution

You invented a new anti-viral drug. You want to test how it works for COVID-19 patients. You select a group of patients, give each patient the drug, wait 5 days and check whether the patient is healthy (success) or ill (failure). Taking into account the size of the group and obtained results, you want to know the probability the drug will cure a patient, and how sure you can be of this value.

Let's build a model. Question 1: what kind of distribution governs the process?

- for each patient there are two possible outcomes: success and failure
- patients are independent
- the drug works the same for every patient (pretty strong assumption, requires pre-selecting patients with similar background: age, sex, co-existing illnesses etc.)

Bernoulli distribution

Bernoulli distribution takes one parameter, p – the probability of success.

We need to put this distribution into our model, but don't know this parameter (in fact, this is the parameter we want to find out).

We must inform the model what range this parameter may take and how probable are the values of this parameter.

We use another distribution for this. Question 2: what kind of distribution will we use?

- the parameter is probability, so it may take any value from 0 to 1
- we have no idea about the efficiency of the drug, so we do not prefer any values

Uniform distribution

For this case $a = 0, b = 1$

Let's put it into a model:

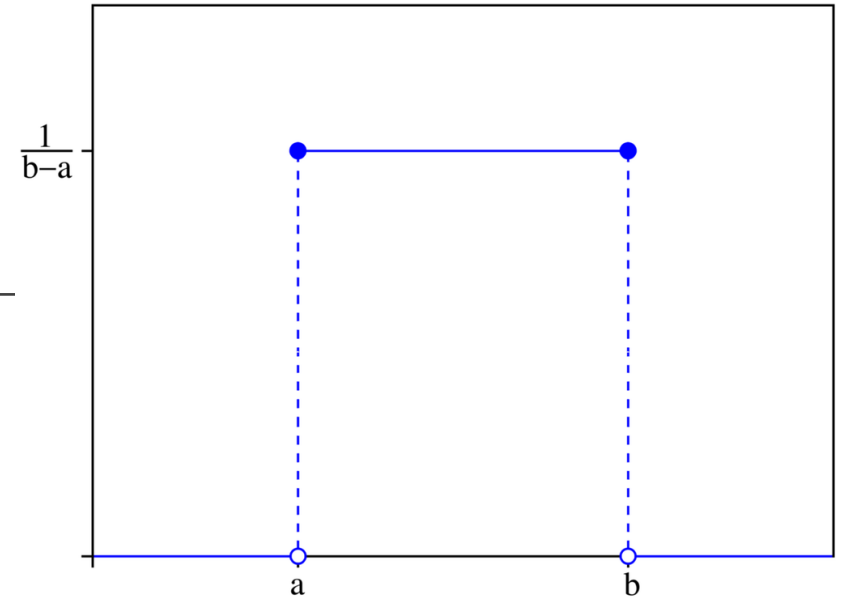
```
with pm.Model() as model:  
    p = pm.Uniform('p', lower=0, upper=1)
```

This variable is **unobserved**: we don't see its value directly in our experiment

Now we will add Bernoulli. This, on the other hand, is observed: we see its values during the experiment (the patient is either cured, or not)

```
obs = pm.Bernoulli("obs", p, observed=occurrences)
```

In order for the model to work, we need the occurrences variable: a list containing results of the experiment.



Running model

The sample method performs inference

```
idata = pm.sample(2000, tune=2500)
```

- first parameter indicates number of samples to draw
- tune indicates number of samples to draw during tune-in phase (they are discarded)

One way to analyse results is to plot the so-called trace. It can be easily done using arviz package.

Complete code:

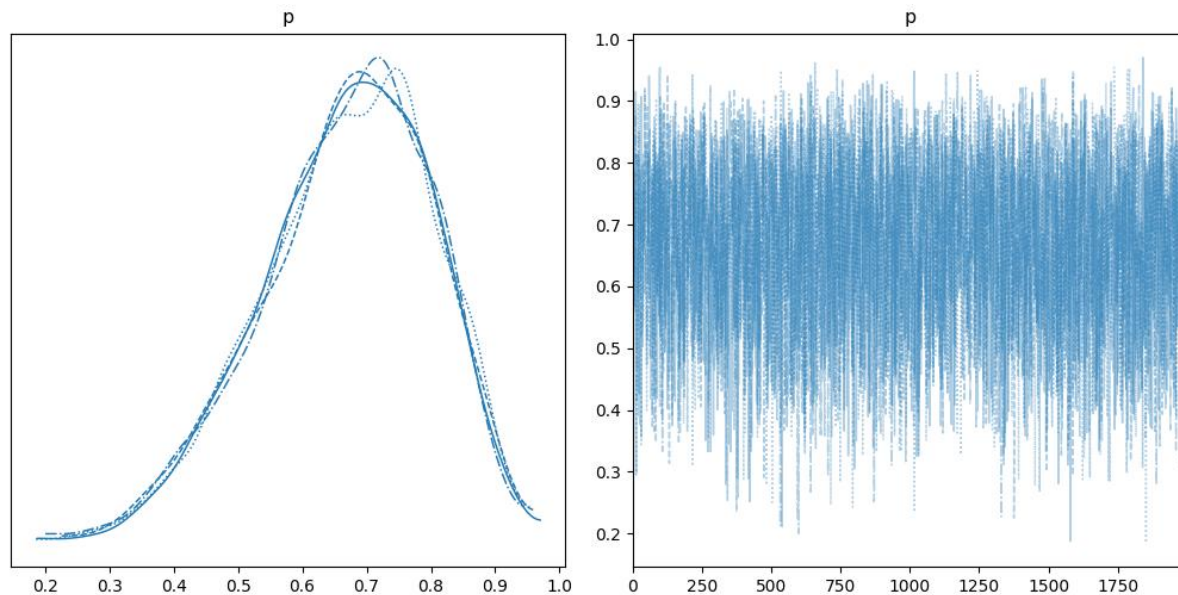
```
import arviz as az
import pymc3 as pm

with pm.Model() as model:
    p = pm.Uniform("p", lower=0, upper=1)
    obs = pm.Bernoulli("obs", p, observed=occurrences)
    idata = pm.sample(2000, tune=2500)
    az.plot_trace(idata, show=True)
```


Observations

Let's see for 10 patients, 7 of which were cured:

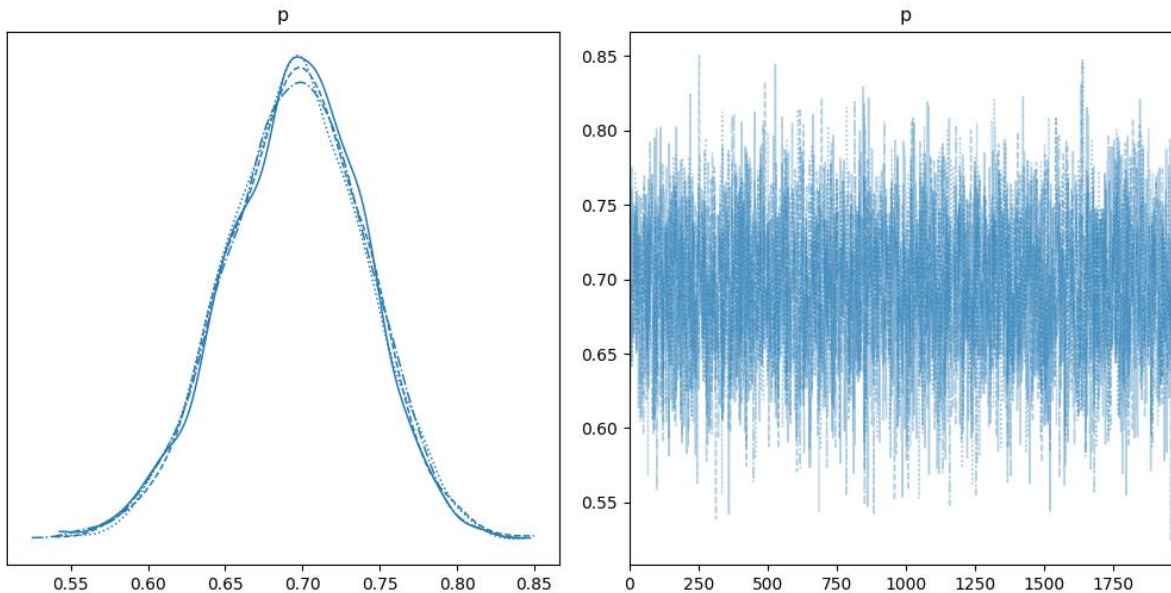
```
occurrences = [1, 1, 1, 1, 1, 1, 1, 0, 0, 0]
```



Observations

Let's see for 100 patients, 70 of which were cured. Note that the percentage of cured remains the same.

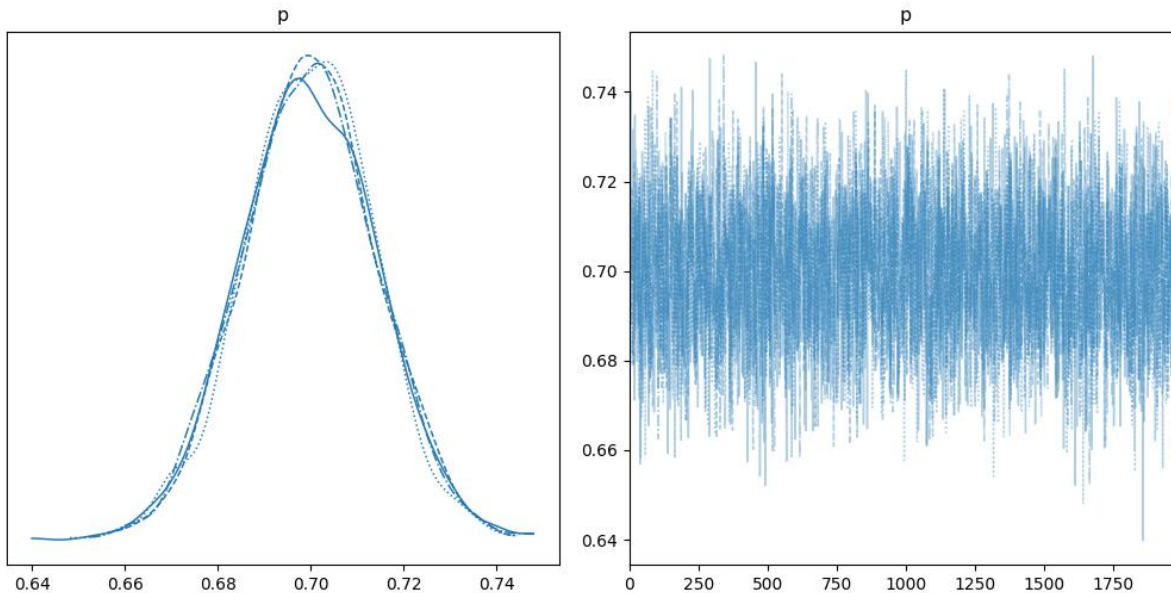
```
ones = np.ones(70)  
zeros = np.zeros(30)  
occurrences = np.concatenate((ones, zeros))
```



Observations

And now for 1000 patients, 700 of which were cured. The percentage of cured still remains the same.

```
ones = np.ones(700)  
zeros = np.zeros(300)  
occurrences = np.concatenate((ones, zeros))
```



Are you taking drugs?

You will be well paid for conducting a survey among students at the University regarding use of drugs. The condition is that the survey must be done during online meetings (e.g., in MS Teams) without any additional technical means, save the simplest everyday objects. As asking a person a point-blank question regarding drug usage is unlikely to result in credible answers, devise a probabilistic way to approximate the truth.

Possible approach

Ask the student to perform the following algorithm:

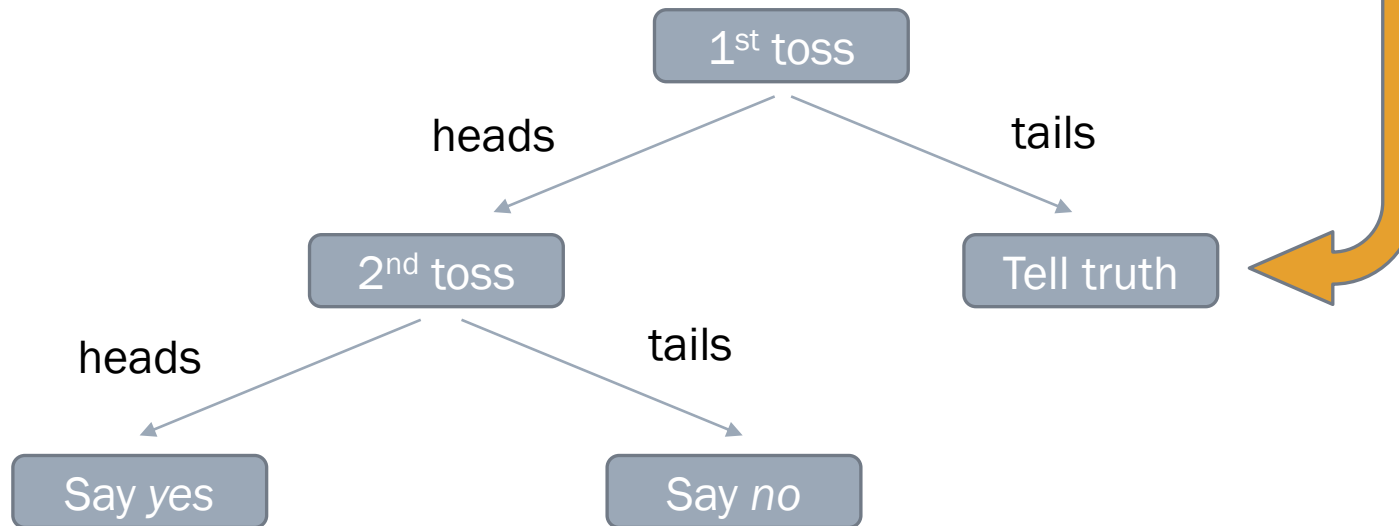
1. toss a coin. Do not tell the result. If it's heads, go to 2, if tails go to 3.
2. toss a coin. Do not tell the result. If it's heads, say that you are taking drugs, if tails, say you are not taking drugs. End.
3. Say honestly whether you are taking drugs. End.

Rationale: The person conducting survey does not know whether the answer results from the person habits or is a result of the coin toss.

Encode the approach as a model

Question 1: What is the probability of obtaining “I’m taking drugs” answer?

Suppose that of the N surveyed students n_d are taking drugs. What is the probability that a randomly chosen student takes drugs? $p_d = n_d/N$



Probability of obtaining “Yes, I’m taking” answer

$$p(Y) = \underset{\substack{\uparrow \\ \text{2nd toss heads}}}{\underset{\substack{\downarrow \\ \text{1st toss heads}}}{\frac{1}{2}}} \left(\underset{\substack{\uparrow \\ \text{2nd toss tails}}}{\frac{1}{2}} \cdot 1 + \frac{1}{2} \cdot 0 \right) + \underset{\substack{\downarrow \\ \text{1st toss tails}}}{\frac{1}{2}} p_d = \frac{1}{4} + \frac{1}{2} p_d$$

Put it into a model

The obtained formula “transforms” the probability we are interested in (p_d) into the probability we observe ($p(Y)$).

This is a deterministic process, so deterministic variable will be used:

```
p_observed = pm.Deterministic("p_observed", 0.25 + 0.5*p_d)
```


Prior distribution and input data

Question 2: what will be the prior distribution of p_d ?

If we have no prior knowledge, uniform will do:

```
p_d = pm.Uniform("p_d", 0, 1)
```

Question 3: what kind of data will be available?

Each student will provide yes/no answer.

Students are anonymous, indistinguishable.

Total number of respondents will be known after the survey is complete.

Putting in input data

The data can be treated as Bernoulli distribution:

```
occurrences = np.concatenate((np.zeros(70), np.ones(30)))  
answers = pm.Bernoulli("answers", p_observed, observed=occurrences)
```

or as binomial distribution:

```
yes_count = 70  
answers = pm.Binomial("answers", 100, p_observed, observed=yes_count)
```

And now for something completely different...

In some bank operations the transaction is rounded to the whole \$. Consequently, the real value of the transaction will differ from the value recorded in the account.

We will consider one year of transactions. We are interested in the impact of this policy on our bank account, i.e., how much can we gain or lose in one year due to rounding.

Let's say we make yearly 1200 such transactions. What can we expect?



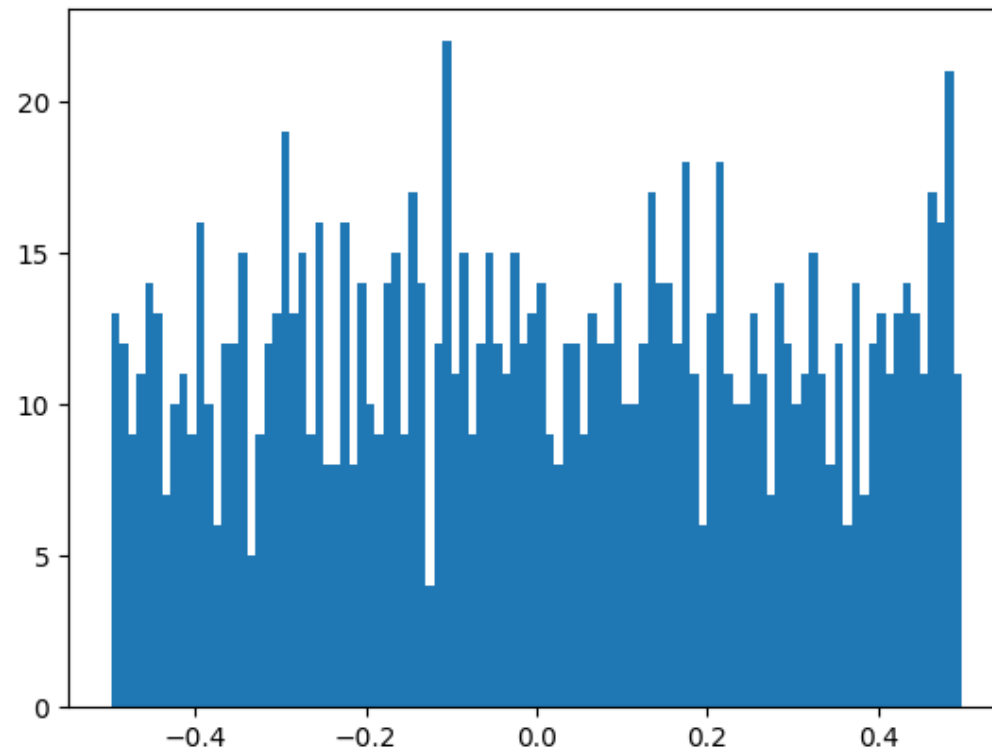
I WANT YOUR ESTIMATES

Underlying distribution

Question: What is the distribution of difference between real and recorded value for a single transaction?



Sample client



Write some code

```
result = []  
for i in range(10000):  
    u = sta.uniform.rvs(loc=-0.5, size=1200)  
    u_sum = np.sum(u)  
    result.append(u_sum)  
plt.hist(result, bins=100)  
plt.show()
```

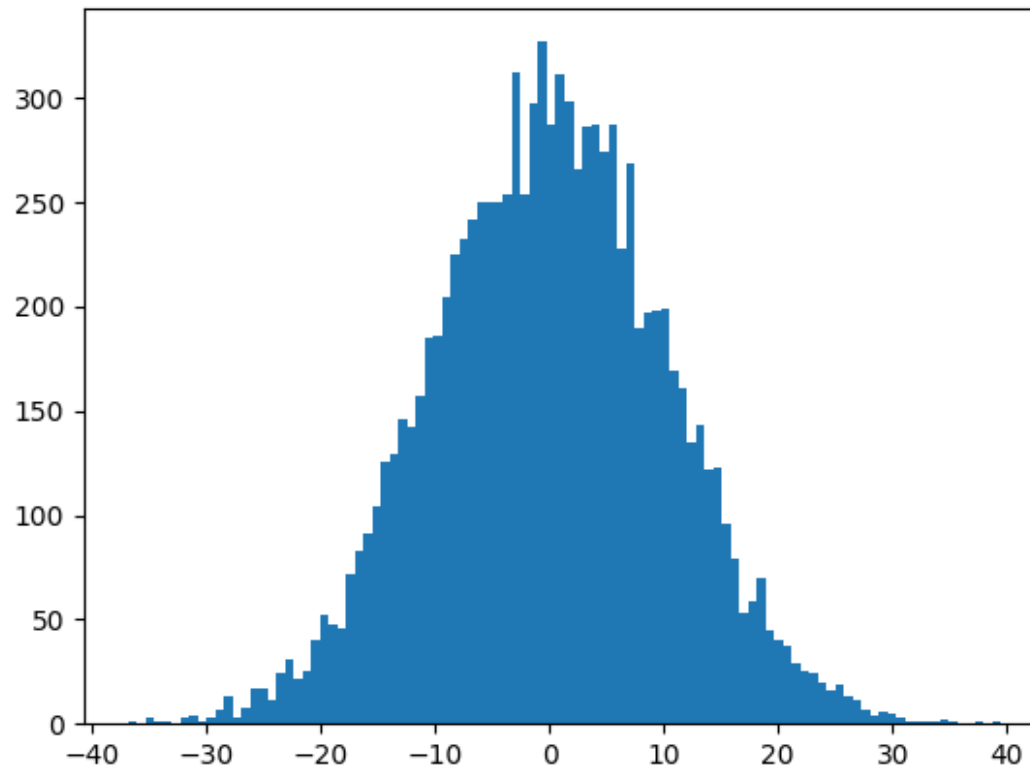
Simulate 10 000 clients.

Use uniform distribution, it defaults to 0, 1 bounds, so use the loc parameter to shift it to -0.5, 0.5. Make 1200 draws ("transactions"), rvs method will draw random numbers.

Add transactions.

Record result.

Be astonished by the result



Loss function

Loss function (a.k.a cost function) maps its arguments (events, values of variables) to a real number that represents “badness” of the situation.

The greater its value, the worse. The task is to minimize the loss function.

It may be used to compare correct decision value θ with estimated decision value $\hat{\theta}$, $L(\theta, \hat{\theta})$. In this case the task is to bring it to 0.

Commonly used loss functions include:

Quadratic (squared-error): $L(\theta, \hat{\theta}) = (\theta - \hat{\theta})^2$

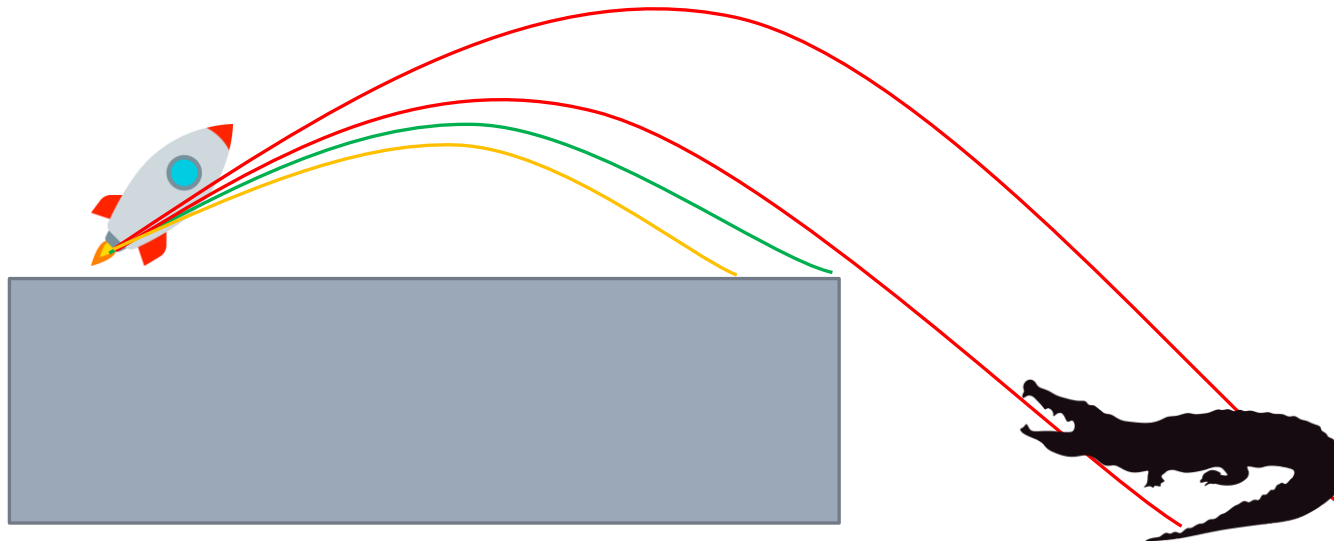
Absolute: $L(\theta, \hat{\theta}) = |\theta - \hat{\theta}|$

0-1: $L(\theta, \hat{\theta}) = \begin{cases} 0 & \text{for } \theta = \hat{\theta} \\ 1 & \text{otherwise} \end{cases}$

Custom loss functions

Quadratic loss is useful, but not always correct.

Example: you take part in a competition, in which contestants (including you) are propelled towards an edge of a cliff by a rocket charge. Your task is to compute the charge, the contestant who lands closest to the edge will win \$1000.



Linear regression

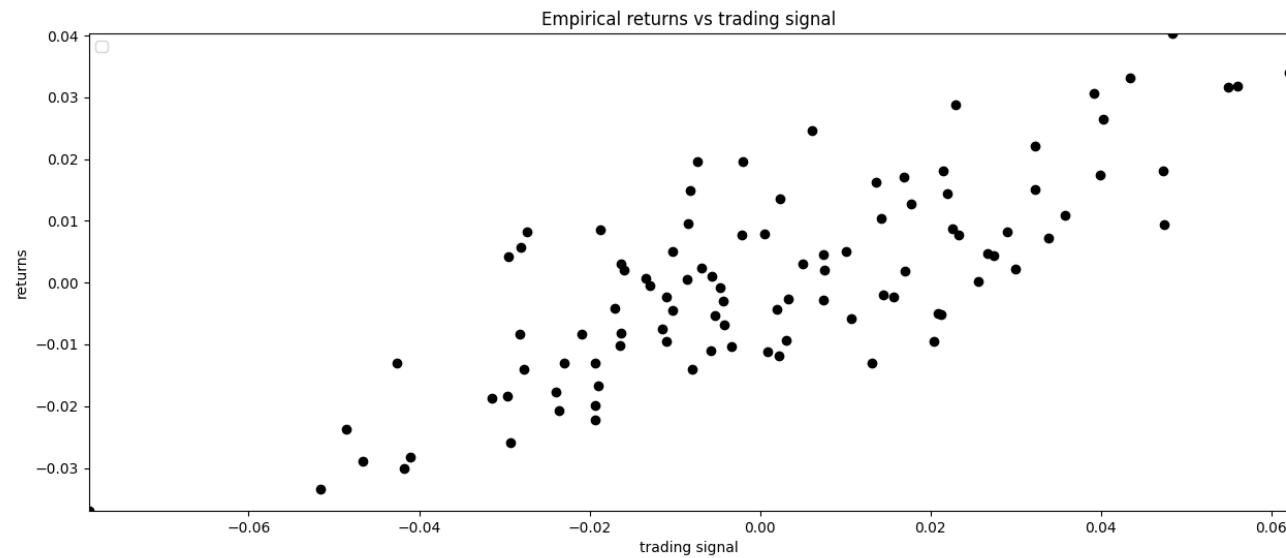
Common application of quadratic loss is in linear regression.

In two dimensions, the task is to fit a line based on a set of (x_i, y_i) value pairs. This means computing α and β of equation $y = \alpha + \beta x$

When quadratic loss is used, the fit is performed in such way as to minimize $\sum_i y_i - \alpha + \beta x_i$

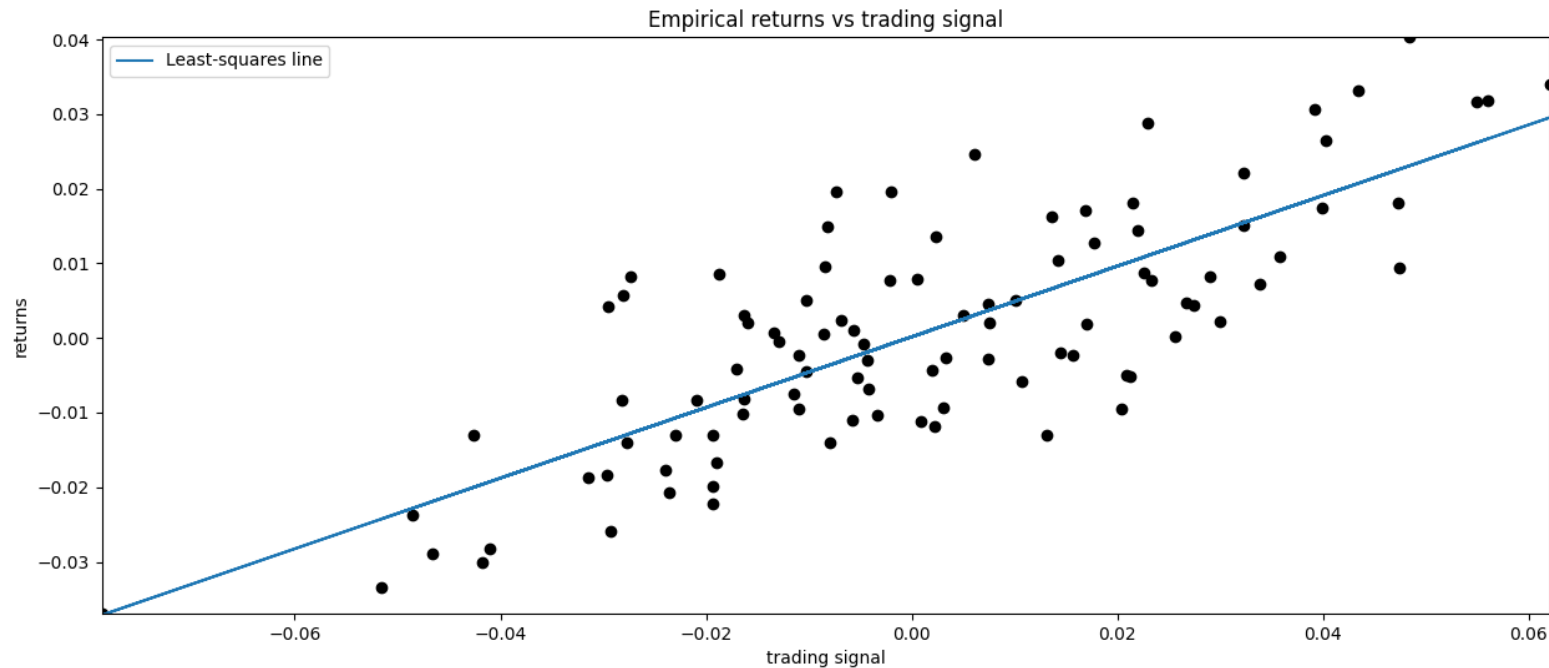
Trade signal vs. future returns

Let's assume that we identified a trade signal¹ for which the returns seem to follow linear relationship. We want to capture this relationship so that we can perform the correct investment action.



¹ A trigger for action, either to buy or sell a security or other asset, generated by analysis

Using linear regression



α 0.0001585368241228453

β 0.47349479915088166

Is quadratic loss correct?

If we prognose **positive** returns, and they turn out to be **negative**, it is **very bad**: we bought stock, and its prices dropped, so we lost money.

If we prognose **negative** returns, and they turn out to be **positive**, it is **very bad**: we sold stock, and its prices went up, so we lost money.

If we prognose **positive** returns, and they turn out to be **positive**, but different, it is **somewhat bad**: we bought stock, and its prices went up, so we earned money, but not as much as we could.

If we prognose **negative** returns, and they turn out to be **negative**, but different, it is **somewhat bad**: we sold stock, and its prices dropped, so we earned money, but not as much as we could.

Is quadratic loss correct?

Quadratic loss is symmetric and cannot follow that pattern:

$$\hat{\theta} = 0.01, \theta = -0.01, L(\theta, \hat{\theta}) = (\theta - \hat{\theta})^2 = (-0.02)^2 = 0.004$$

$$\hat{\theta} = -0.03, \theta = -0.01, L(\theta, \hat{\theta}) = (\theta - \hat{\theta})^2 = (0.02)^2 = 0.004$$

Custom loss

We want the loss to be dependent on the difference between prognosis and reality.

If prognosis and reality have the same sign, linear “penalty” is sufficient.

If the signs differ, more punishing approach should be used.

```
def stock_loss(factual, predicted, coef = 100.):
```

```
    if factual * predicted < 0:
```

```
        return coef * predicted**2 + abs(factual - predicted)
```

```
    else:
```

```
        return abs(factual - predicted)
```

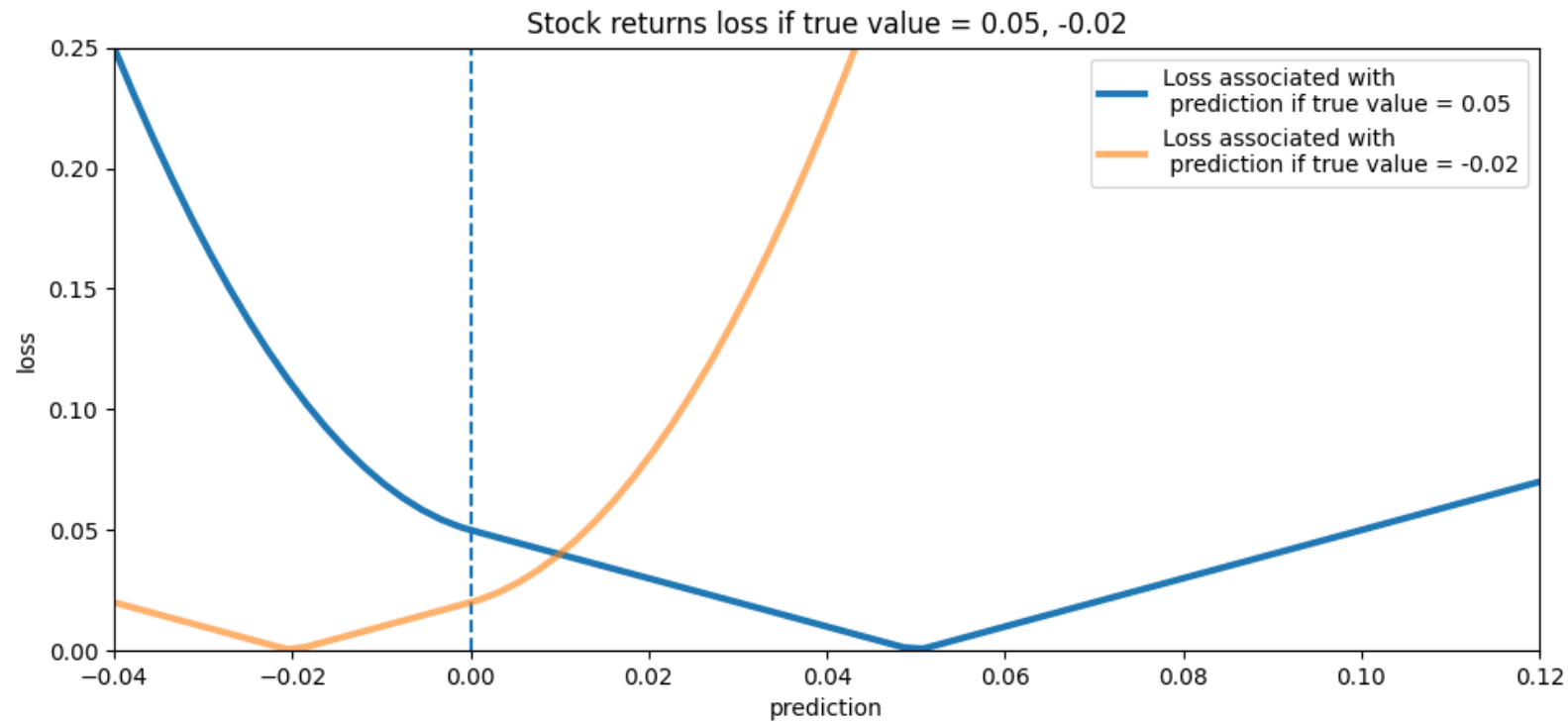
← Different signs – very bad.

← Quadratic + linear.

← Same signs – somewhat bad.

← Linear only.

Custom loss



Applying the custom loss

The devised loss function cannot be (reasonably) applied to typical linear regression – no matter what loss we use, the result will be a single straight line (albeit perhaps different from the obtained previously).

Linear regression tries to generalize the previously recorded data and ignores the currently available information: whether the current trade signal is at 0.01 or at 0.05, the prediction is performed using the **same straight line**.

Bayesian regression will fit **many straight lines**, with various degree of probability. We can then use our custom loss function to choose the line that minimizes loss for the currently observed trade signal.

Bayesian linear model

The model will take form of $y = \alpha + \beta x + \epsilon$, where $\epsilon \sim N(0, \sigma)$

Commonly used priors for α and β are normal. We will use uniform prior for σ .

```
with pm.Model() as model:
```

```
    std = pm.Uniform("std", 0, 100)
```

```
    beta = pm.Normal("beta", mu=0, sd=100)
```

```
    alpha = pm.Normal("alpha", mu=0, sd=100)
```

}

```
    mean = pm.Deterministic("mean", alpha + beta * X)
```

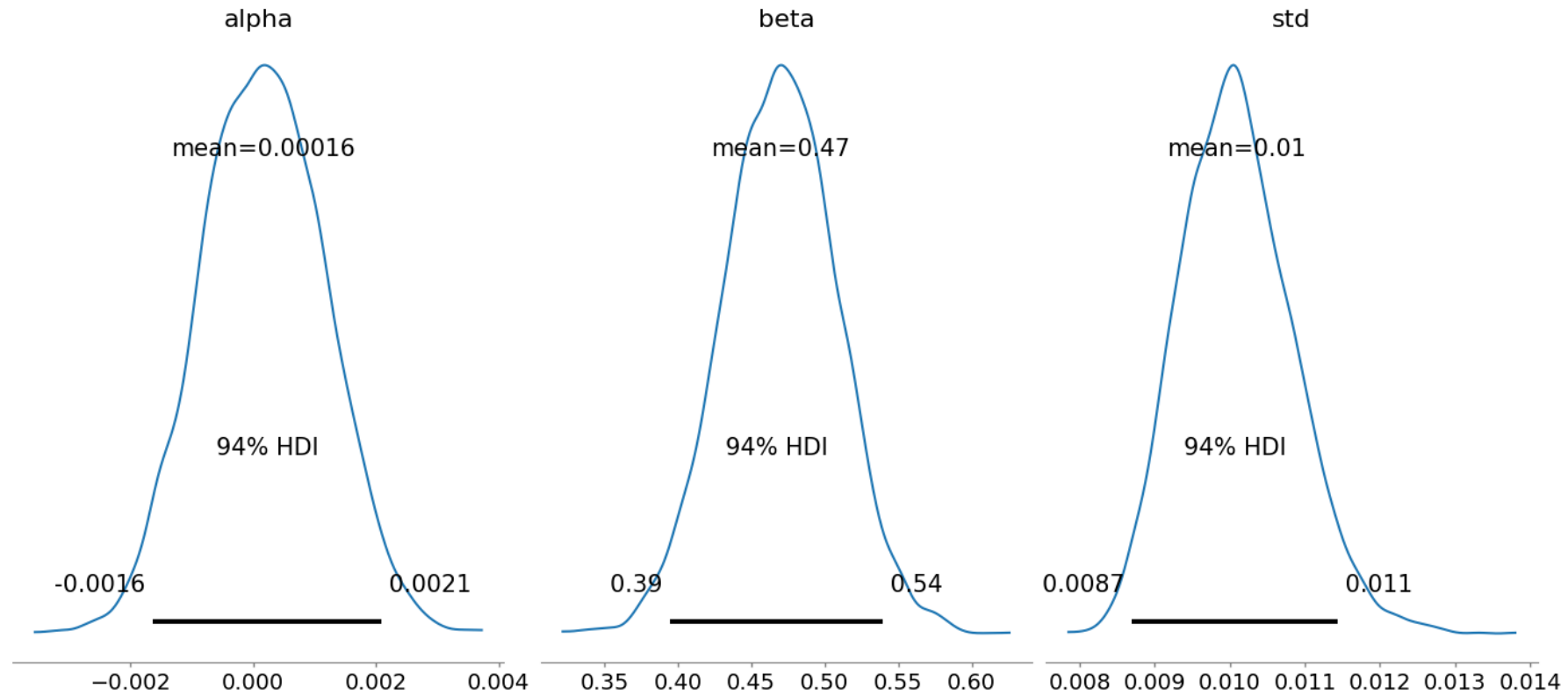
```
    obs = pm.Normal("obs", mu=mean, sd=std, observed=Y)
```

Priors

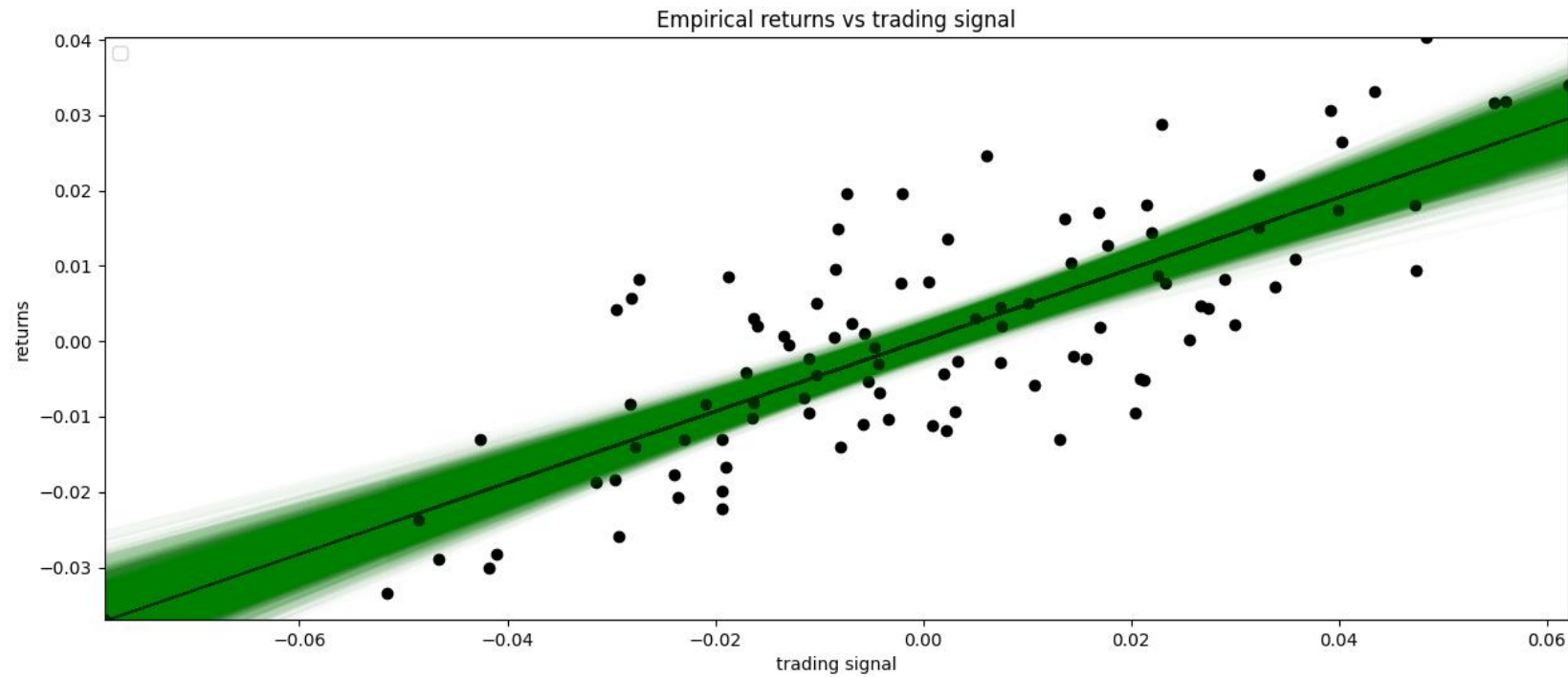
Linear regression part

Stochastic error part

Bayesian linear model



Possible regression lines



Using the model to minimize custom loss

We will be confronting our prediction with many possible real scenarios (i.e., possible lines fitted to the observed data), so it's useful to have a loss function that computes losses for all these scenarios at once:

```
def stock_loss_vect(factual_vect, predicted, coef=100):  
    loss_vect = np.zeros_like(factual_vect)  
    ix_vect = factual_vect * predicted < 0  
    loss_vect[ix_vect] = coef * predicted ** 2 + abs(factual_vect[ix_vect] - predicted)  
    loss_vect[~ix_vect] = abs(factual_vect[~ix_vect] - predicted)  
    return loss_vect.mean()
```

← Vector of “realities” and single prediction

← We want single output value, mean of losses for all possible scenarios

Optimization routine

We want to find the prediction (single value) that minimizes the loss function from the previous slide.

We do not know the functional relation tying our prediction and loss – so we need an optimization algorithm.

SciPy supplies fmin function:

scipy.optimize.fmin

```
scipy.optimize.fmin(func, x0, args=(), xtol=0.0001, ftol=0.0001, maxiter=None,  
maxfun=None, full_output=0, disp=1, retall=0, callback=None, initial_simplex=None)
```

Minimize a function using the downhill simplex algorithm.

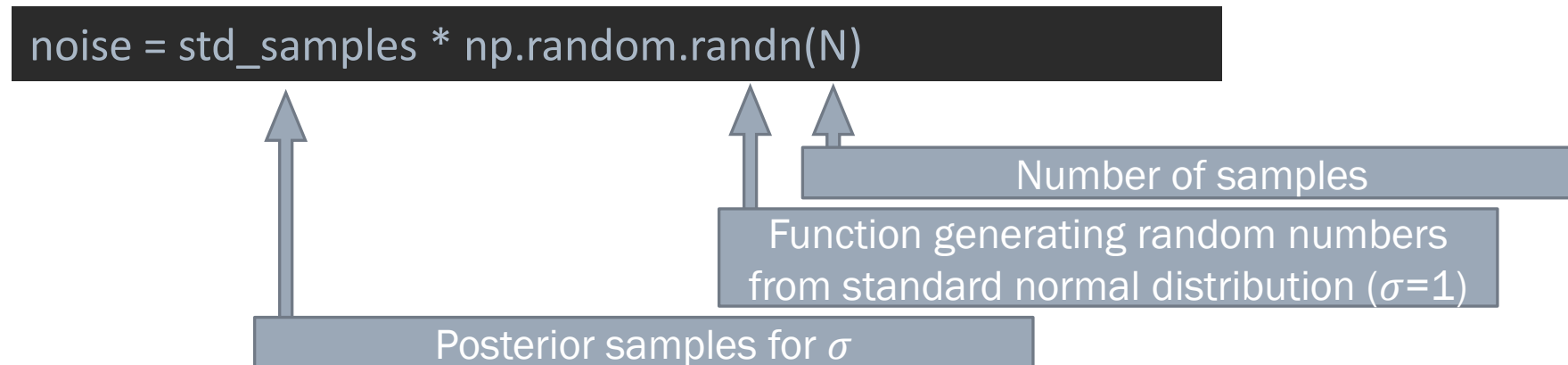
This algorithm only uses function values, not derivatives or second derivatives.

Almost there...

We will be using the posteriors generated by PyMC3: $y = \alpha + \beta x + \epsilon$, where $\epsilon \sim N(0, \sigma)$

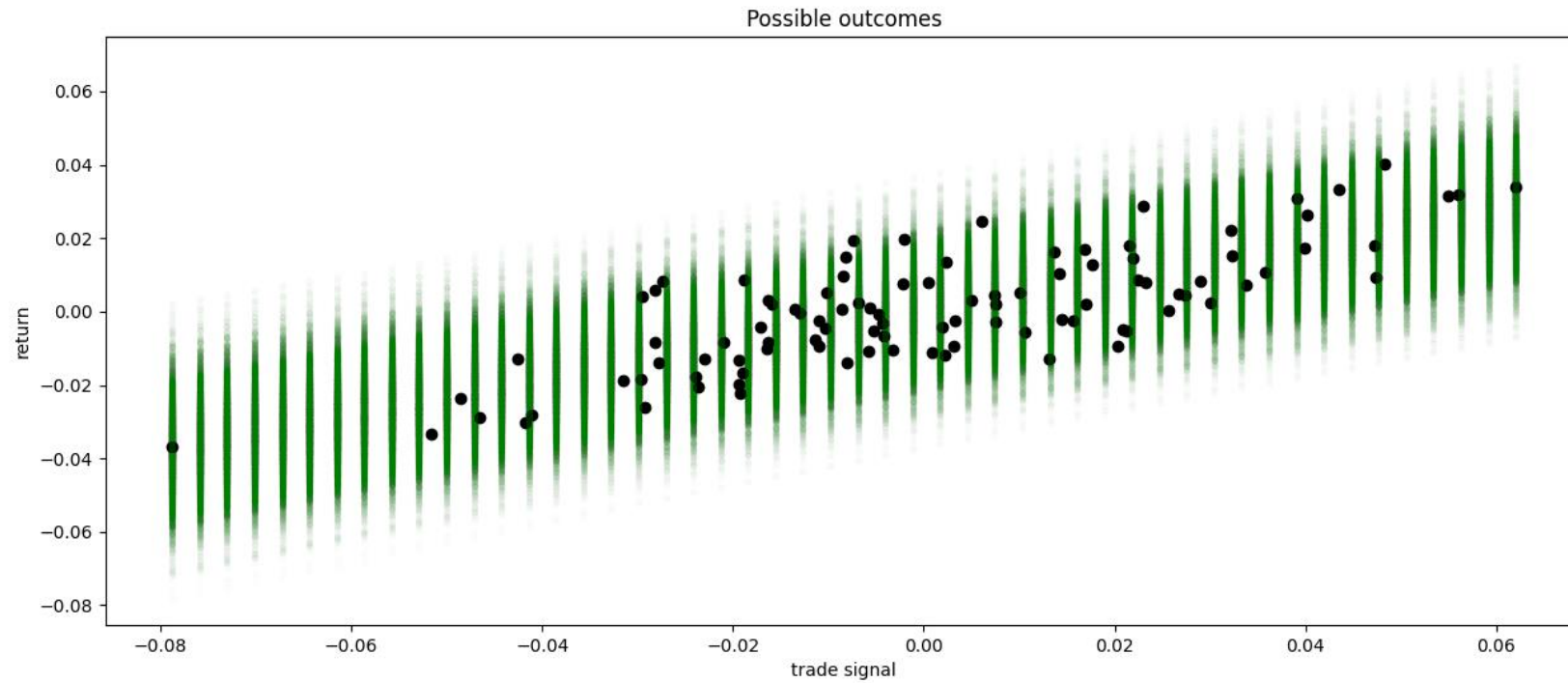
We can **directly use** posteriors for α and β .

We must **compute** ϵ using the posterior for σ :

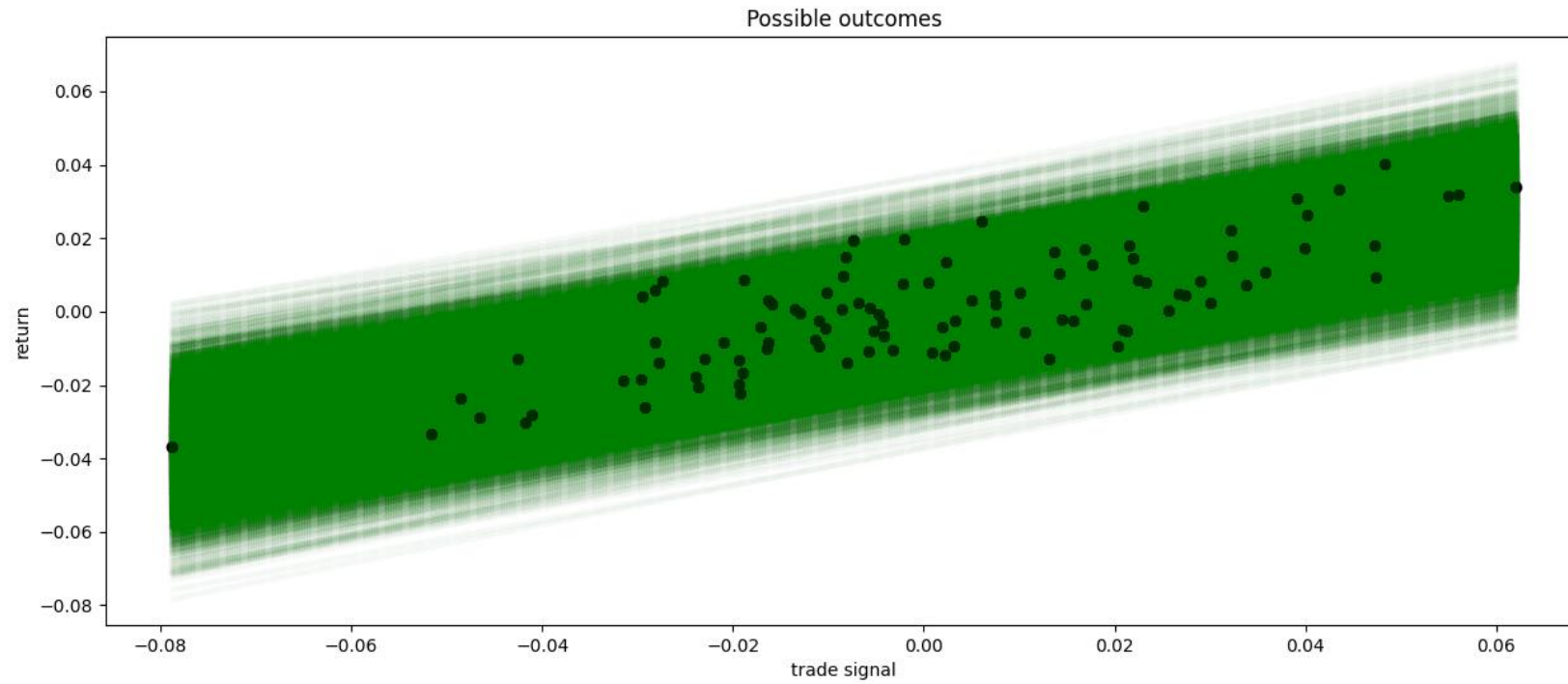


Possible outcomes

```
possible_outcomes = alpha_samples + beta_samples * signal + noise
```



Possible outcomes



Complete optimization routine

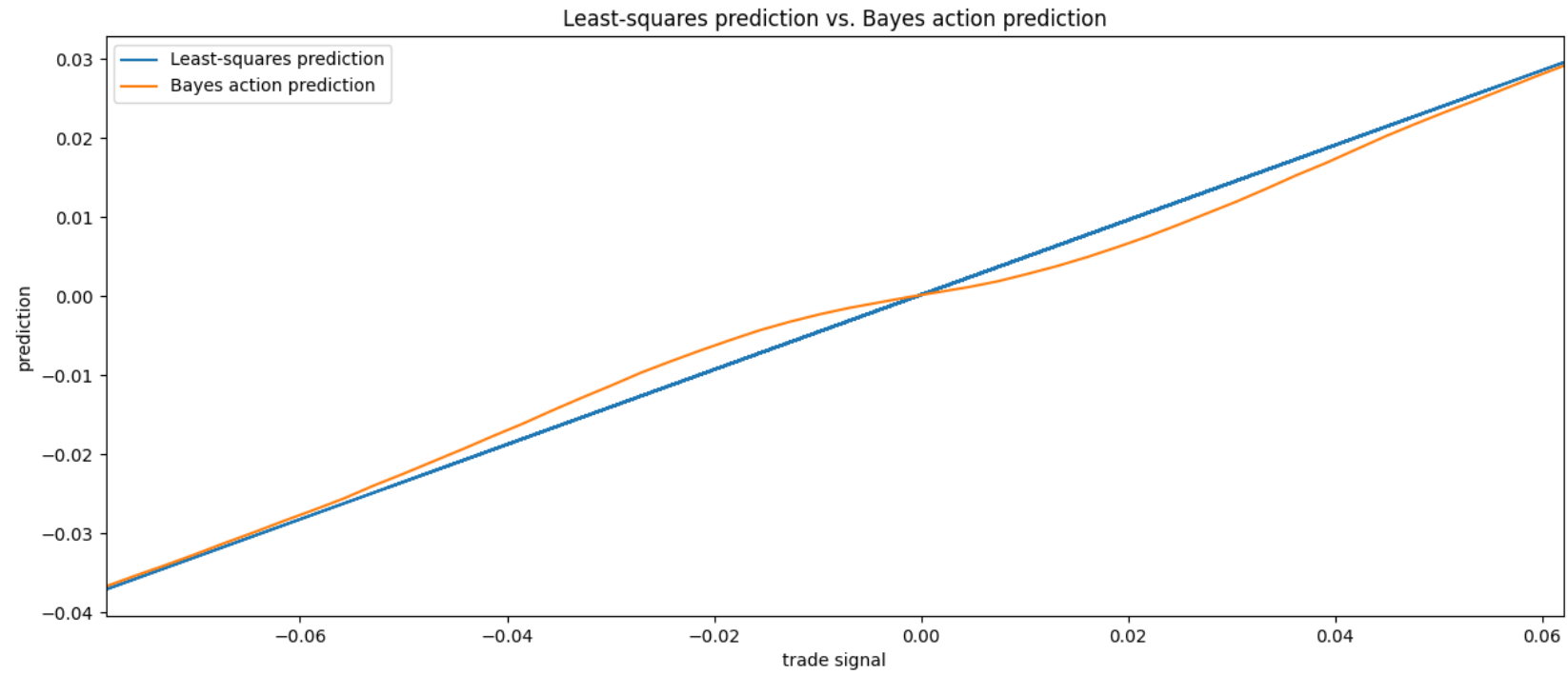
```
std_samples = idata["posterior"]["std"]
alpha_samples = idata["posterior"]["alpha"]
beta_samples = idata["posterior"]["beta"]

N = std_samples.shape[0]

noise = std_samples * np.random.randn(N)

opt_predictions = np.zeros(50)
trade_signals = np.linspace(X.min(), X.max(), 50)
for i, signal in enumerate(trade_signals):
    possible_outcomes = alpha_samples + beta_samples * signal + noise
    def tomin(pred):
        return stock_loss_vect(possible_outcomes, pred)
    opt_predictions[i] = fmin(tomin, 0, disp=False)
```

Results

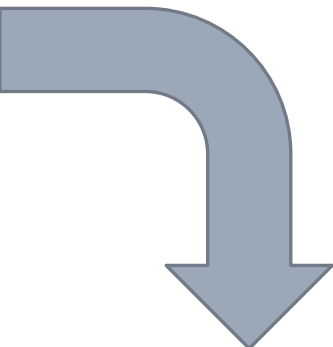


Simplified approach using GLM

```
std = pm.Uniform("std", 0, 100)
beta = pm.Normal("beta", mu=0, sd=100)
alpha = pm.Normal("alpha", mu=0, sd=100)

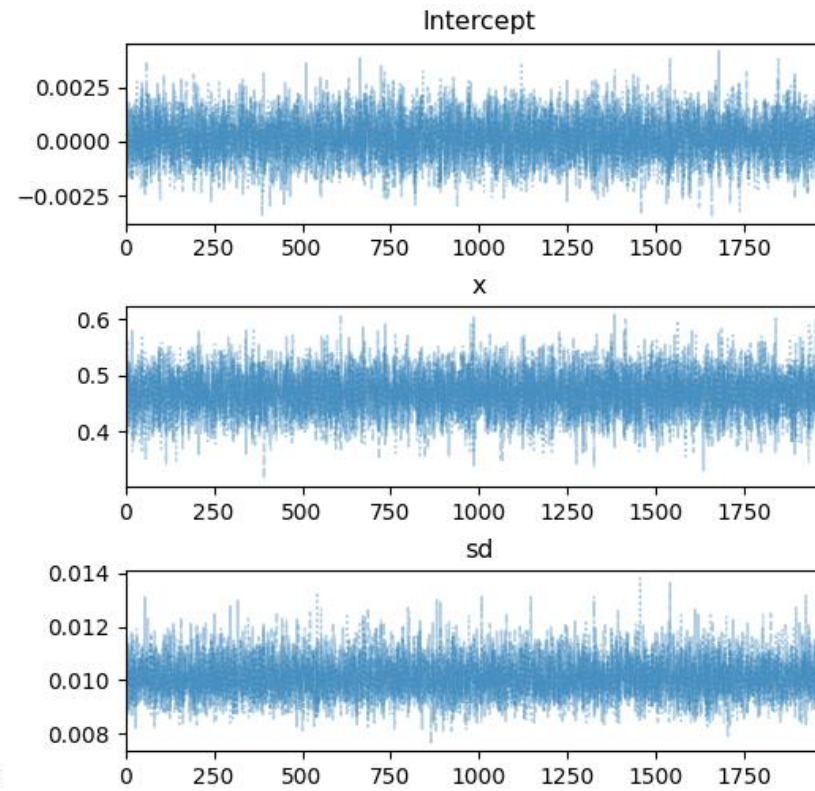
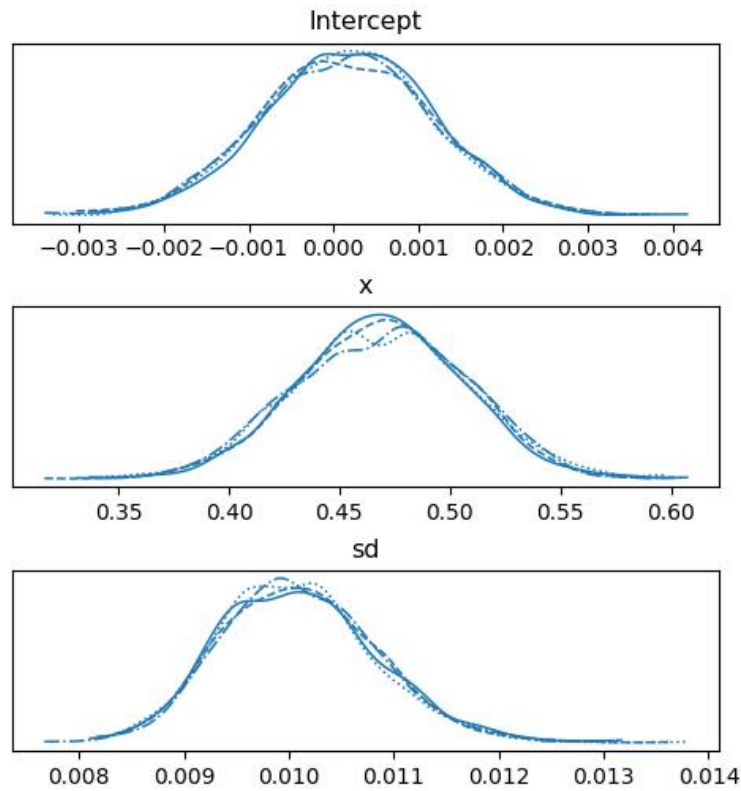
mean = pm.Deterministic("mean", alpha + beta * X)

obs = pm.Normal("obs", mu=mean, sd=std, observed=Y)
```

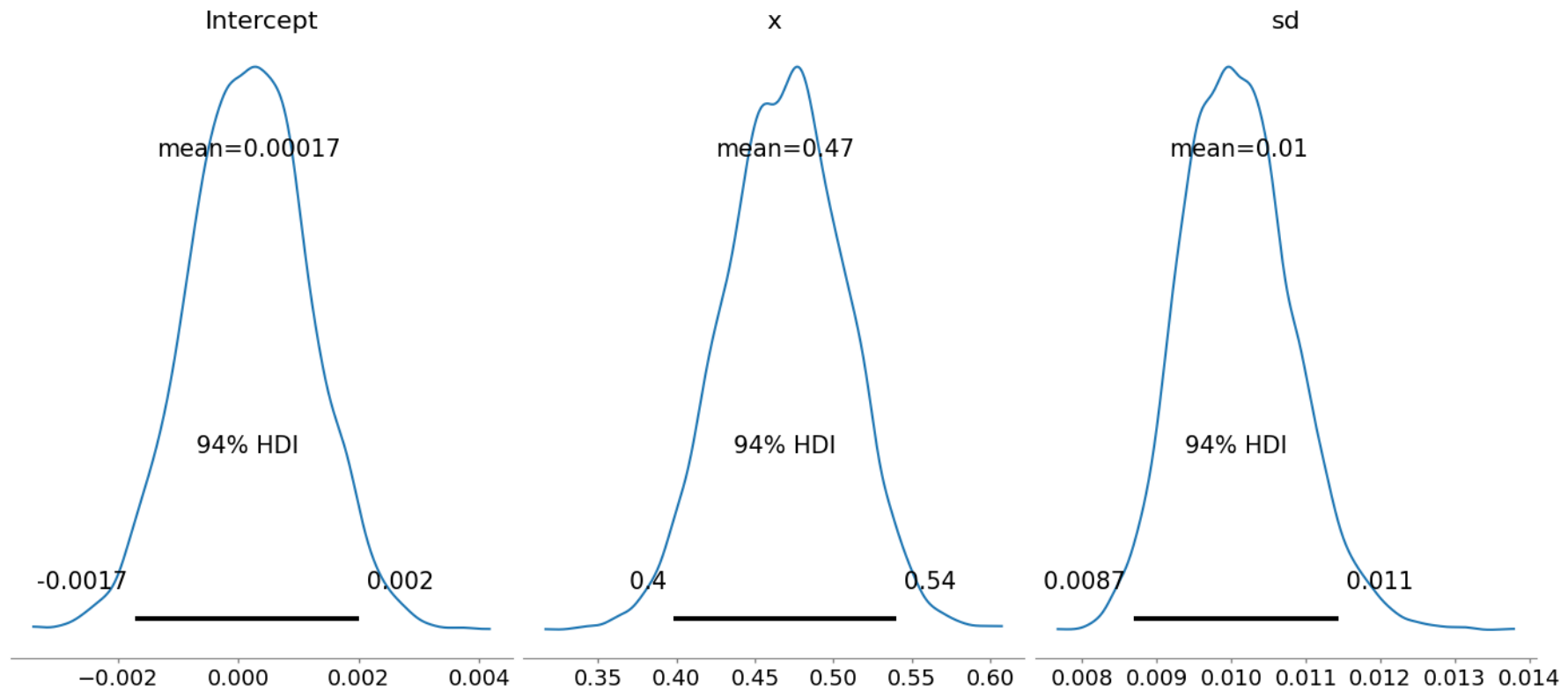


```
data = dict(x=X, y=Y)
pm.GLM.from_formula("y ~ x", data)
```

Simplified approach using GLM



Simplified approach using GLM

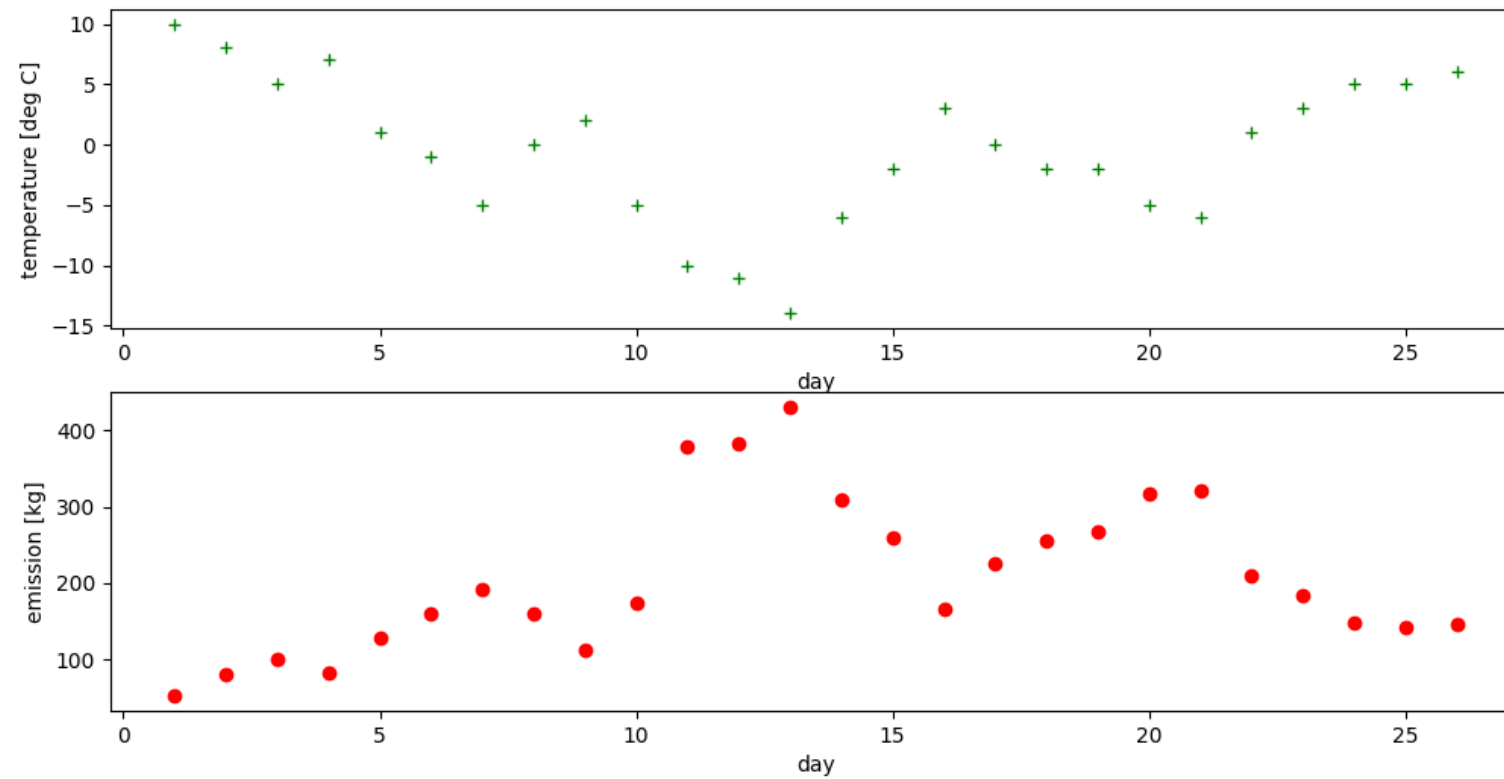


Linear relationship with changing parameter

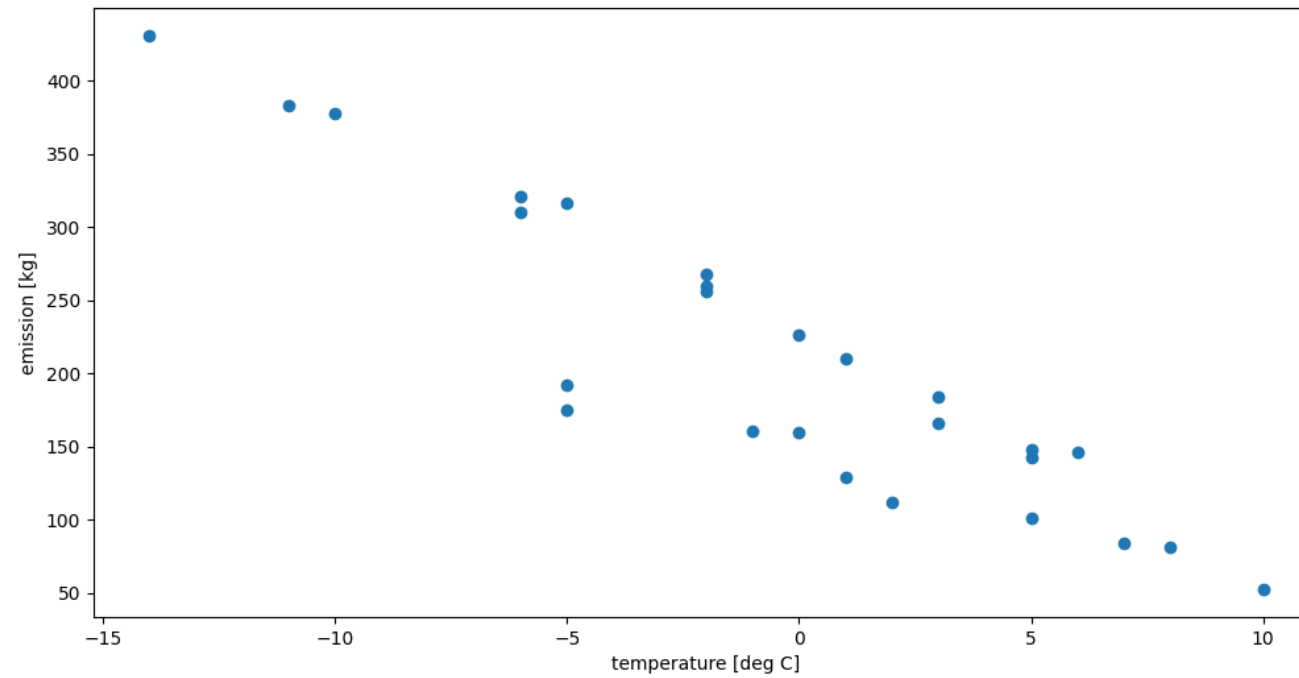
You are an investigative journalist concerned with environmental issues. You got a hint from local activists that there may be excessive emission from the city's heating plant. You don't know what the proper emission should be, but you know that the heating plant starts operating when the air temperature drops below 15°C and the emission goes up linearly as the air temperature drops further. You have access to emission measurements and temperature data for the past 26 days.

As you don't know the proper emission, you want to check whether the parameter tying emission with temperature changed during this period.

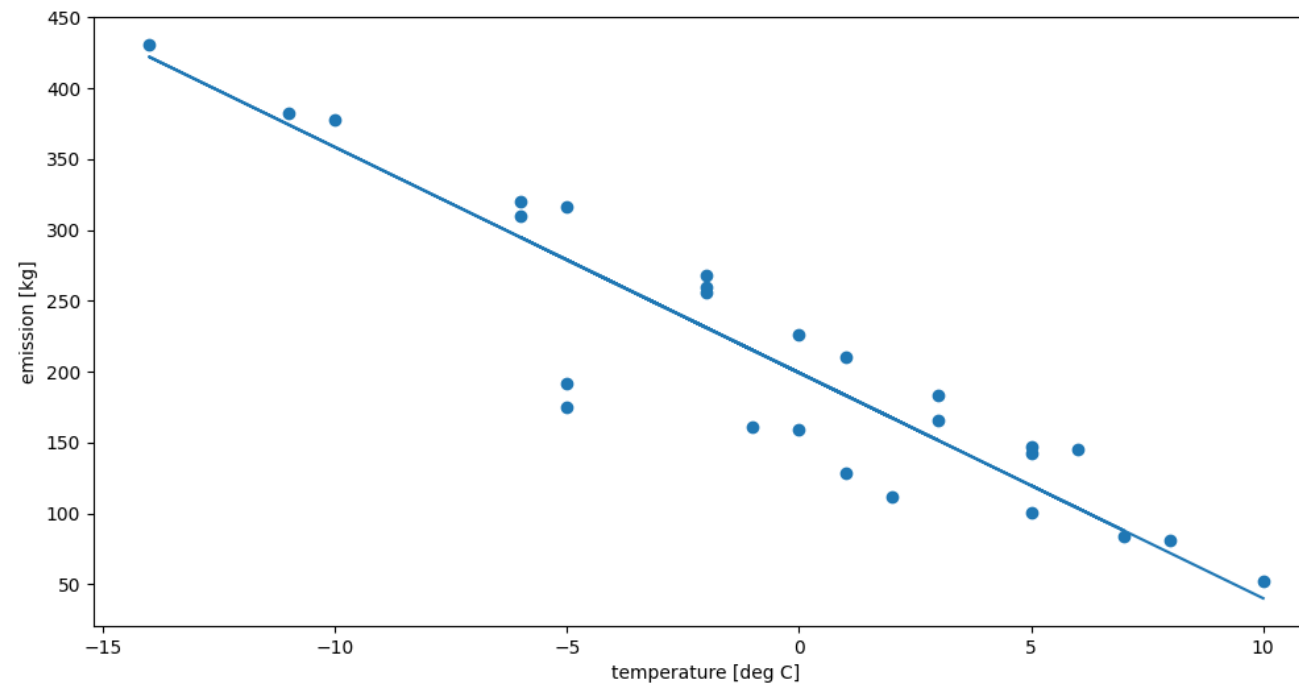
Data



Data



Linear regression

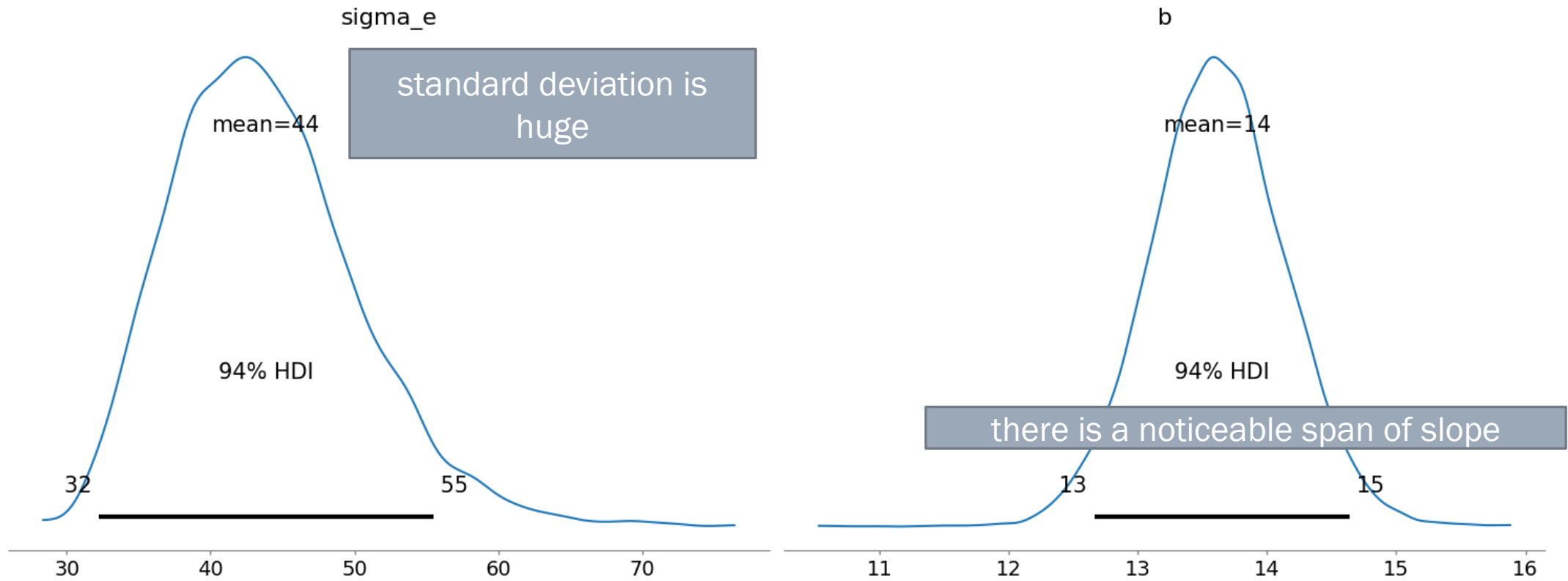


Initial model

Let's make a linear model that will cover the data:

```
with pm.Model() as model:  
    b = pm.Uniform("b", lower=-100, upper=100)  
    m_e = pm.Deterministic("m_e", b*15-b*temperatures) ←  $b \cdot (15-t)$   
    sigma_e = pm.Uniform("sigma_e", lower=0.1, upper=100)  
    emission = pm.Normal("emission", mu=m_e, sigma=sigma_e, observed=emissions)  
  
idata = pm.sample(2000, tune=2500)
```

Results



Two-part model

Let's now try to make a model that can fit two slopes, separately for two subsets of data.

We do not know the “switch point” between subsets, it also must be parametrized.

```
b_1 = pm.Uniform("b_1", lower=-100, upper=100)
b_2 = pm.Uniform("b_2", lower=-100, upper=100)
tau = pm.DiscreteUniform("tau", lower=0, upper=len(temperatures) - 1)
idx = np.arange(len(temperatures))
b = pm.math.switch(tau > idx, b_1, b_2)
m_e = pm.Deterministic("m_e", b*15-b*temperatures)

sigma_e = pm.Uniform("sigma_e", lower=0.1, upper=100)
emission = pm.Normal("emission", mu=m_e, sigma=sigma_e, observed=emissions)
```

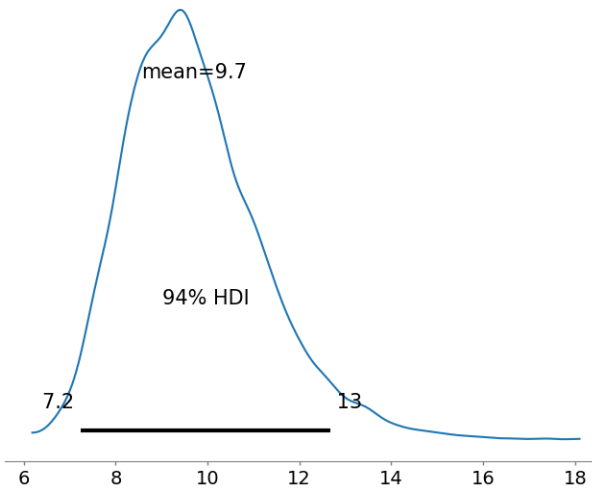
Diagram illustrating the parametrization of the two-part model:

- `b_1`: slope for the first subset
- `b_2`: slope for the second subset
- `tau`: switch point
- `idx`: index
- `b`: function returning `b_1` for `tau > idx` and `b_2` otherwise

Results

significantly reduced
standard deviation

sigma_e



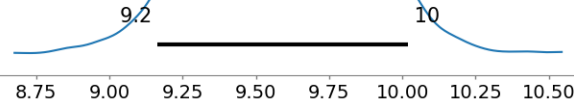
obvious difference of
slopes

b_1

mean=9.6

94% HDI

narrow span of slope

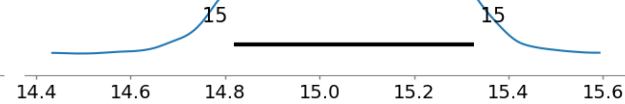


b_2

mean=15

94% HDI

narrow span of slope



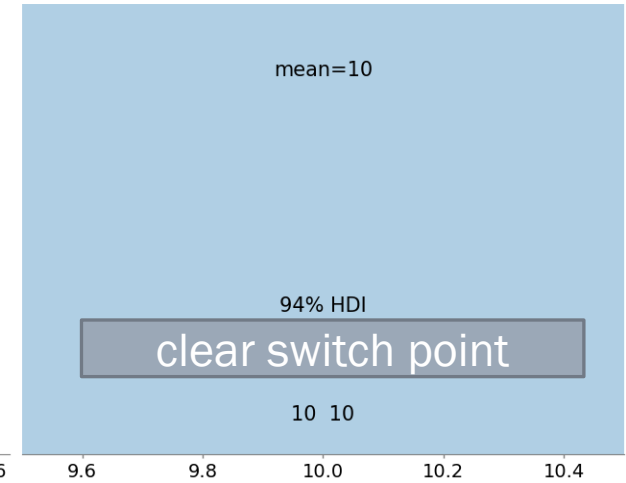
tau

mean=10

94% HDI

clear switch point

10 10



Prior distributions

When building a model, it is necessary to provide information about prior distribution, i.e., our belief about the quantity of interest when no evidence are available.

```
with pm.Model() as model:
```

```
    p = pm.Uniform("p", lower=0, upper=1)
```

```
    obs = pm.Bernoulli("obs", p, observed=occurrence)
```

```
    idata = pm.sample(2000, tune=2500)
```

← belief in probability of anti-viral drug curing a patient, before any patients were treated

```
with pm.Model() as model:
```

```
    std = pm.Uniform("std", 0, 100)
```

```
    beta = pm.Normal("beta", mu=0, sd=100)
```

```
    alpha = pm.Normal("alpha", mu=0, sd=100)
```

← belief in coefficients of linear relationship between trade signal and returns, before any historical data were available

```
    mean = pm.Deterministic("mean", alpha + beta * X)
```

```
    obs = pm.Normal("obs", mu=mean, sd=std, observed=Y)
```

Importance of priors

As a minimum, priors will influence the convergence process of the sampling algorithm.

They also often influence the final result.

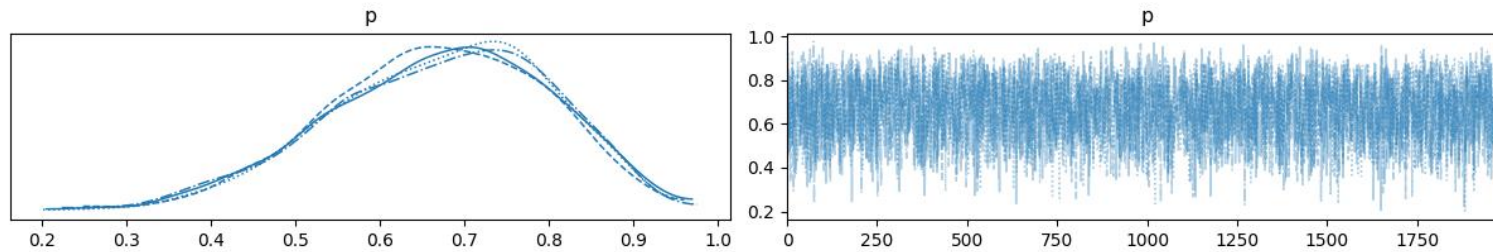
Correctly defined priors may express our lack of knowledge.

Or they may express the prior knowledge, e.g., elicited from experts.

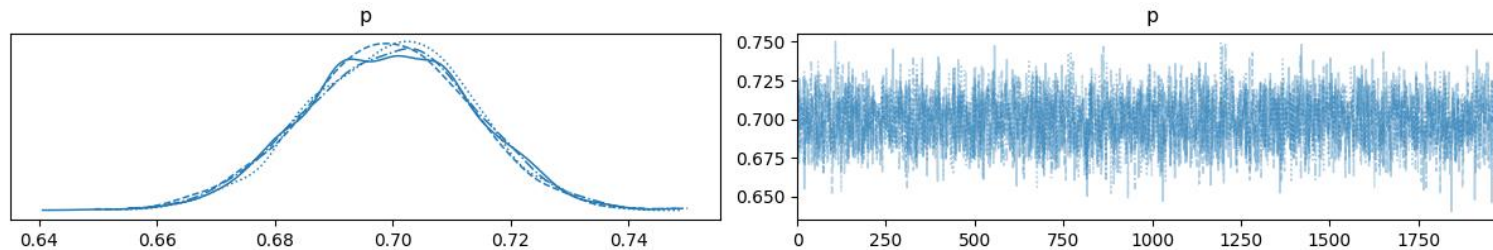
Back to anti-viral drug

The original example utilized uniform prior. Lets' recall how the results looked like:

7 cured and 3 not cured patients



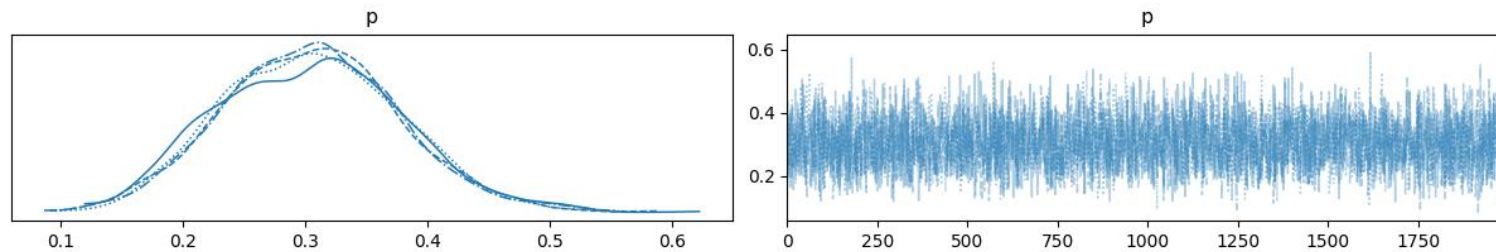
700 cured and 300 not cured patients



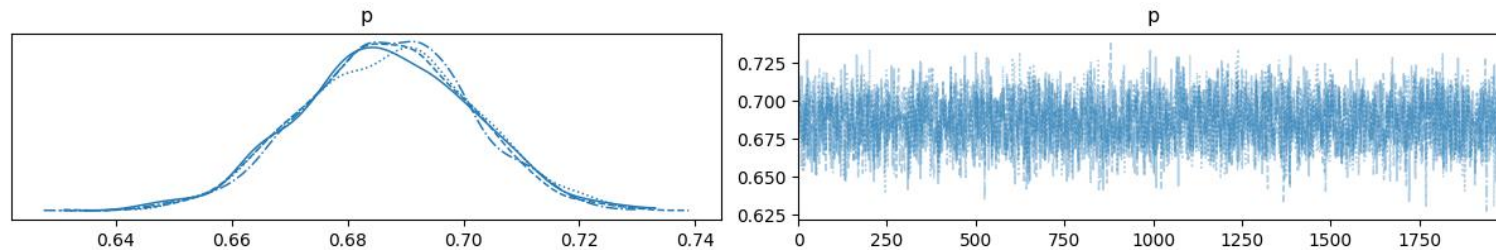
Back to anti-viral drug

And now use truncated normal with $\mu = 0.1$ and $\sigma = 0.1$:

7 cured and 3 not cured patients

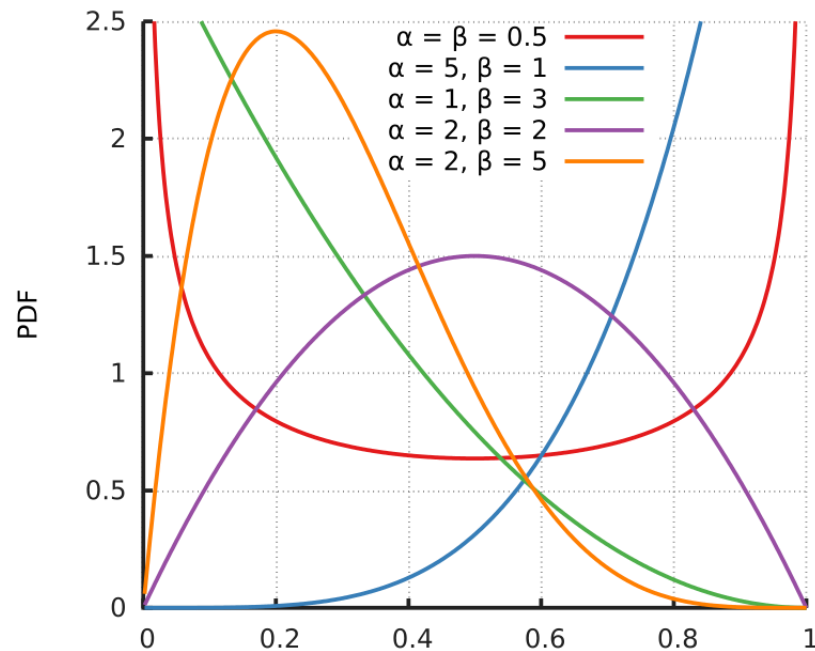


700 cured and 300 not cured patients



A very useful prior

Beta distribution is defined on the interval $[0; 1]$ and is governed by two parameters, α and β .



When $\alpha = \beta = 1$, Beta distribution becomes continuous uniform.

Conjugate prior

In Bayesian approach, we are interested in computing posterior distribution of some parameter θ given data x : $p(\theta|x)$

We do this using prior distribution of the parameter, $p(\theta)$, likelihood function returning probability of data given parameter, $p(x|\theta)$, and the probability of data, $p(x)$:

$$p(\theta|x) = \frac{p(x|\theta)}{p(x)} p(\theta)$$

The Big Question: is it possible that the posterior and the prior have the same algebraic form (are the same function, only with different parameters)?

Usefulness of Beta distribution

Beta distribution is conjugate prior if the likelihood function is Binomial or Bernoulli distribution.

	Prior Beta parameters	Observed random variable	Posterior Beta parameters
Bernoulli	α, β	X – success (1) or failure (0)	$\alpha + X, \beta + 1 - X$
Binomial	α, β	X – number of successes in N trials	$\alpha + X, \beta + N - X$

Usefulness of conjugate priors

Because the form of the posterior is known, it is not necessary to employ Markov Chain sampling approaches.

Instead, the posterior distribution is directly obtained by applying the observations to the parameters of the distribution (see table in the previous slide).

Conjugate prior can also be treated as a way of “seeding” the computations with a number of “pseudo-observations”, expressing our beliefs or real previous observations. E.g., if in some previous Bernoulli experiment we obtained 23 successes and 7 failures, we will set

$$\alpha = 23, \beta = 30 - 23 = 7$$

Choose best supplier

You run a company producing electronic systems that rely on hard-to-manufacture state-of-art component. You got batches of this component from several manufacturers. You are aware that there may be vast differences of quality of components between manufacturers but have no way of determining this quality other than building your system and checking whether it works (success) or not (failure). You want to devise a procedure which you will employ each time when building your system. It should select the batch (manufacturer) that will maximize the number of successes (working systems).

Why this task poses a problem?

At first, you have no idea about quality of batches, so the first choice will be made completely at random.

After you've built the first system, you've gained some knowledge about one of the suppliers. But this knowledge is very imprecise. As the element from the batch was chosen at random, you might have chosen the only “lemon” in the overall good batch. Or the only one functioning in a batch of “lemons”.

If the element was OK, should you stick with the batch and not test other batches? Obviously, if it was a “lemon”, you want to try another batch. But should you never return to the first one?

Possible strategies

Choose at random 😞

Select the batch that best performed so far.

If a batch returns a “lemon”, select the next batch.

Rank the N batches from best to worst based on their success ratio, select N elements from the best, $N-1$ from the next one, etc.

Rank the N batches from worst to best based on their success ratio s_i , compute ratio of success ratios of neighbouring batches $r_i = s_i/s_{i-1}$. Select 1 element from the worst, then from each next batch r_i times the amount of the previous batch.

Bayesian strategy

Set the prior distribution for each batch to $Beta(1, 1)$

Loop:

- For each batch i , sample a random variable X_i from the prior
- Select the batch for which X_i was largest, i.e., select $I = \operatorname{argmax} X_i$
- Build the system, test it, and update prior of batch I accordingly

Rationale:

- $Beta(1, 1)$ as prior expresses our lack of knowledge about batches (indifference to the choice of batch)
- Sampling allows to favour the batches that performed best so far, but (due to randomness) allows also to give a chance to other batches
- Each resulting system, whether good or bad, increases our body of knowledge

Possible implementation

Class representing a single batch:

```
class Batch:
    quality = 0.5
    alpha = 1
    beta = 1

    def __init__(self, quality):
        self.quality = quality

    def sample_distribution(self):
        return np.random.beta(self.alpha, self.beta)

    def get_element(self):
        good = np.random.rand() < self.quality
        self.alpha = self.alpha + int(good)
        self.beta = self.beta + 1 - int(good)
        return good
```

probability of drawing good element from the batch; not visible to the system producer

parameters of the Beta distribution for this batch

draws a sample from Beta distribution for the batch

gets element from the batch (Bernoulli process)

updates parameters of the Beta distribution based on the quality of the element

Possible implementation

Class representing factory:

```
class Factory:
    batches = []
    systems = 0
    good = 0

    def set_batches(self, qualities):
        self.batches = [Batch(quality) for quality in qualities]
        self.systems = 0
        self.good = 0

    def get_element(self):
        samples = [batch.sample_distribution() for batch in self.batches]
        choice = np.argmax(samples)
        result = self.batches[choice].get_element()
        self.systems = self.systems + 1
        self.good = self.good + int(result)
```

← batches of elements from different manufacturers

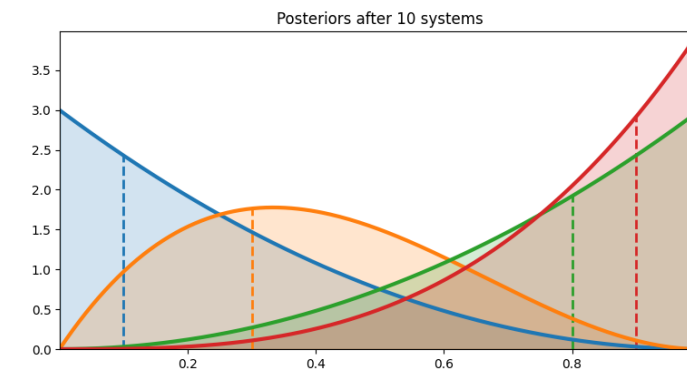
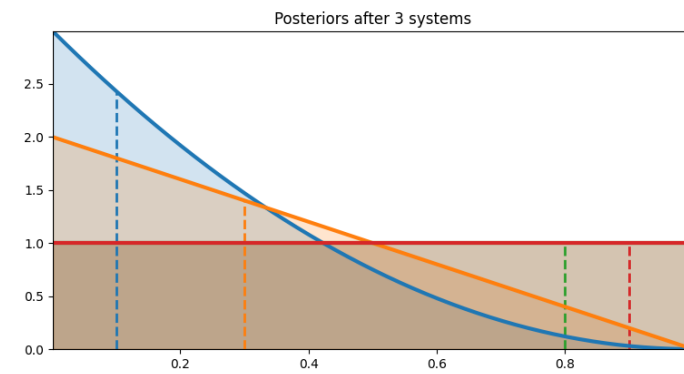
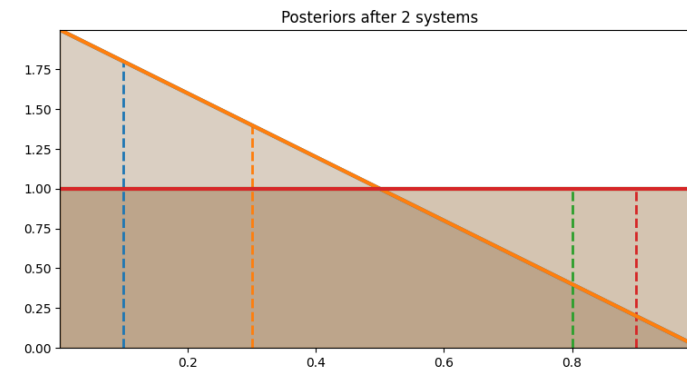
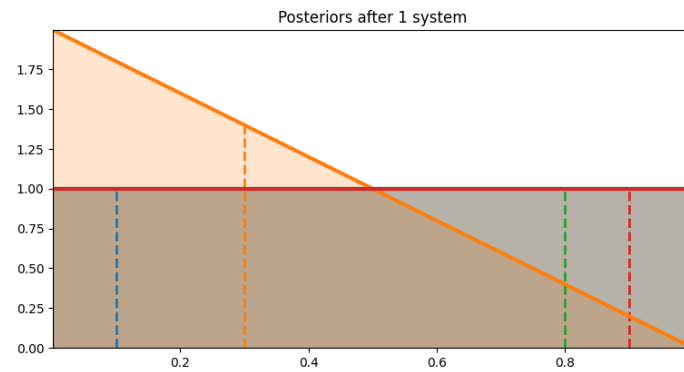
← count how many systems were produced and how many were good

← makes batches based on list of probabilities

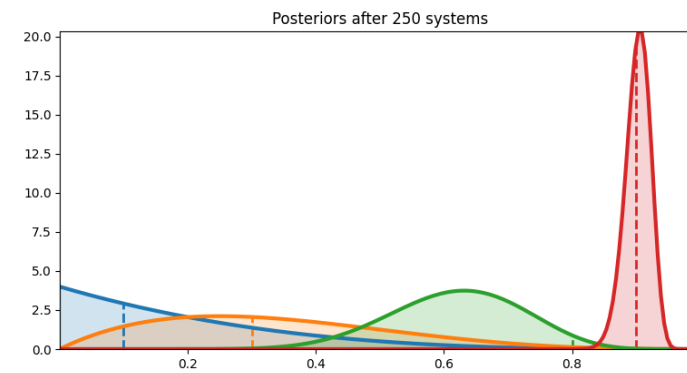
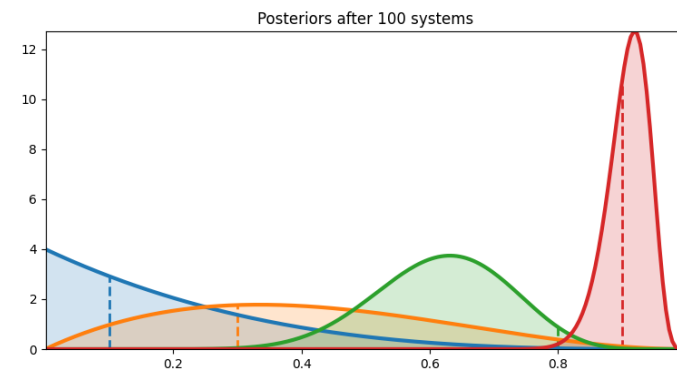
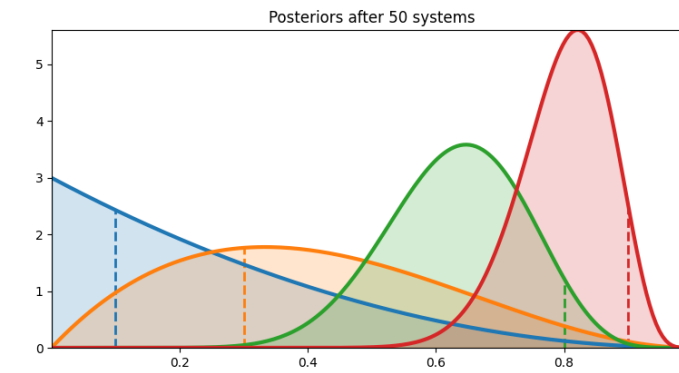
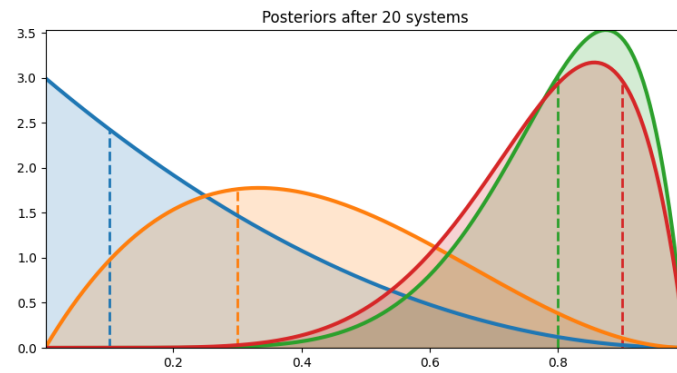
← draws one sample from prior distribution of each batch

← selects the best batch and gets element from it

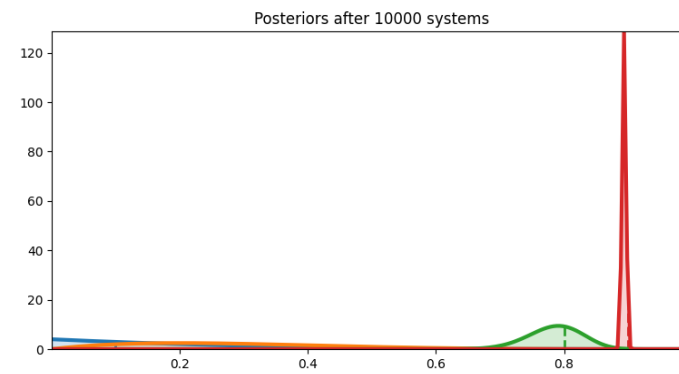
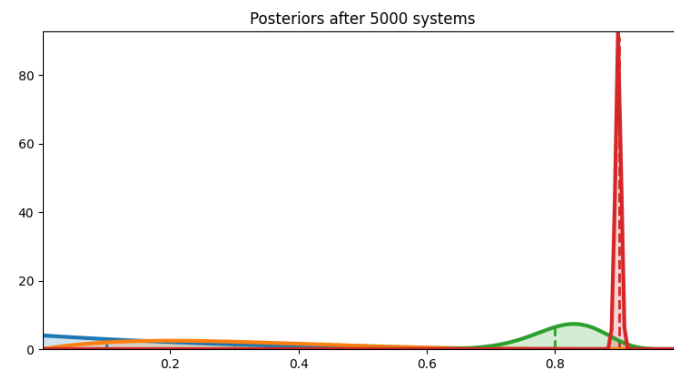
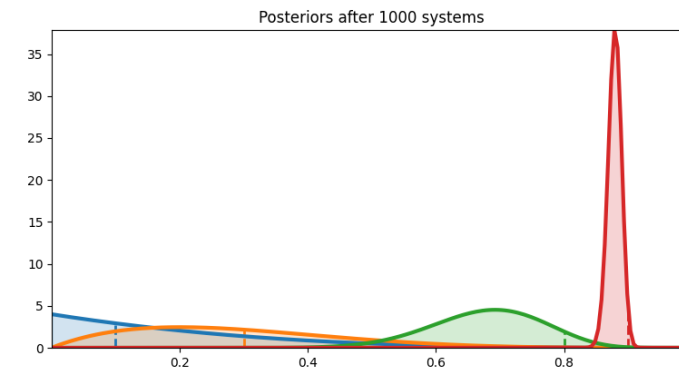
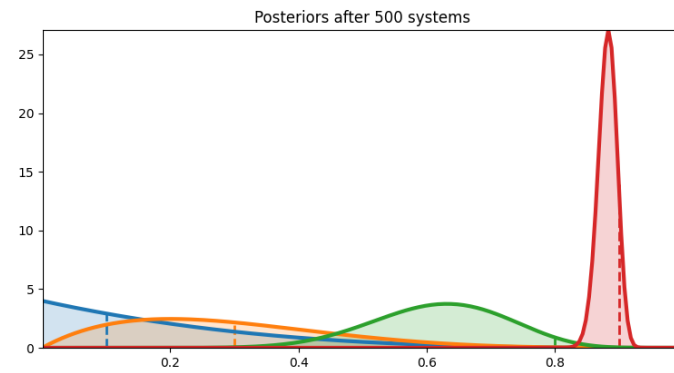
Results



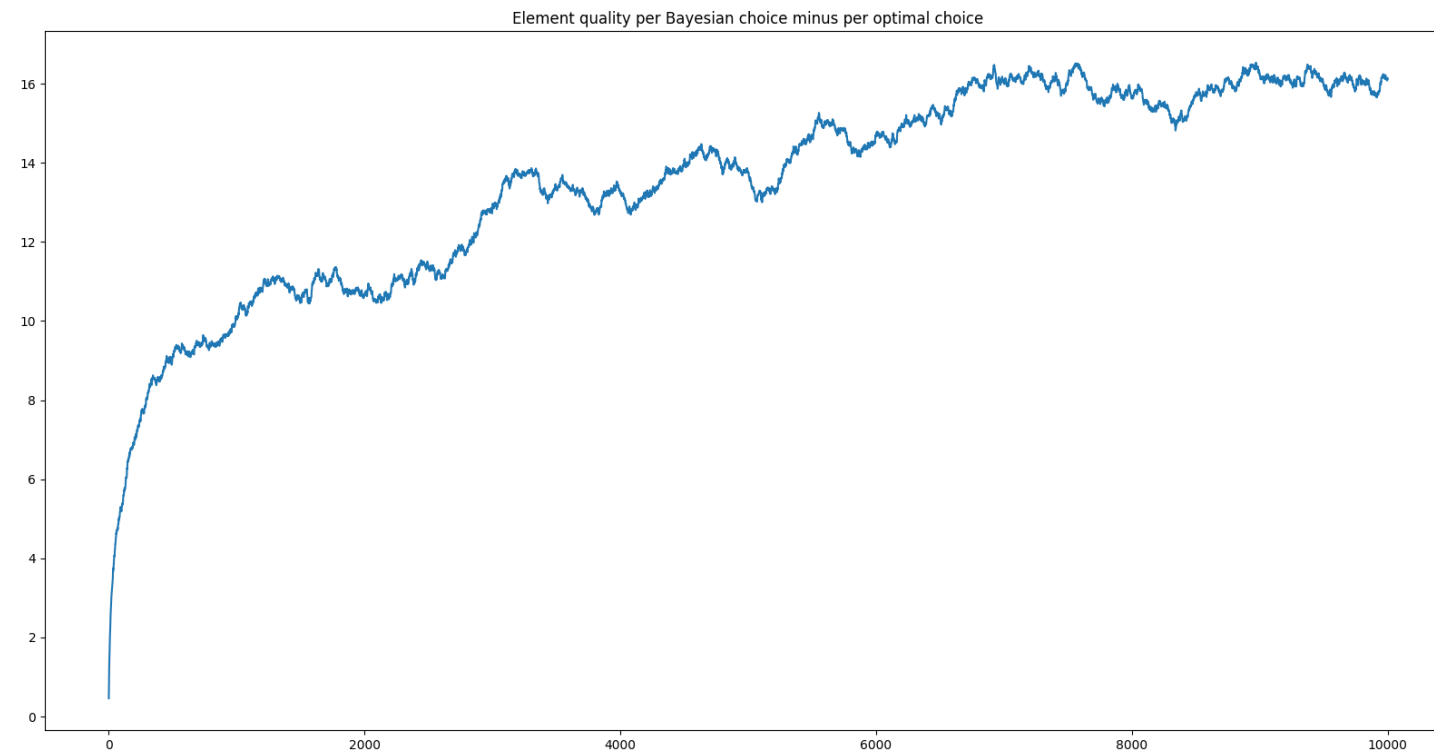
Results



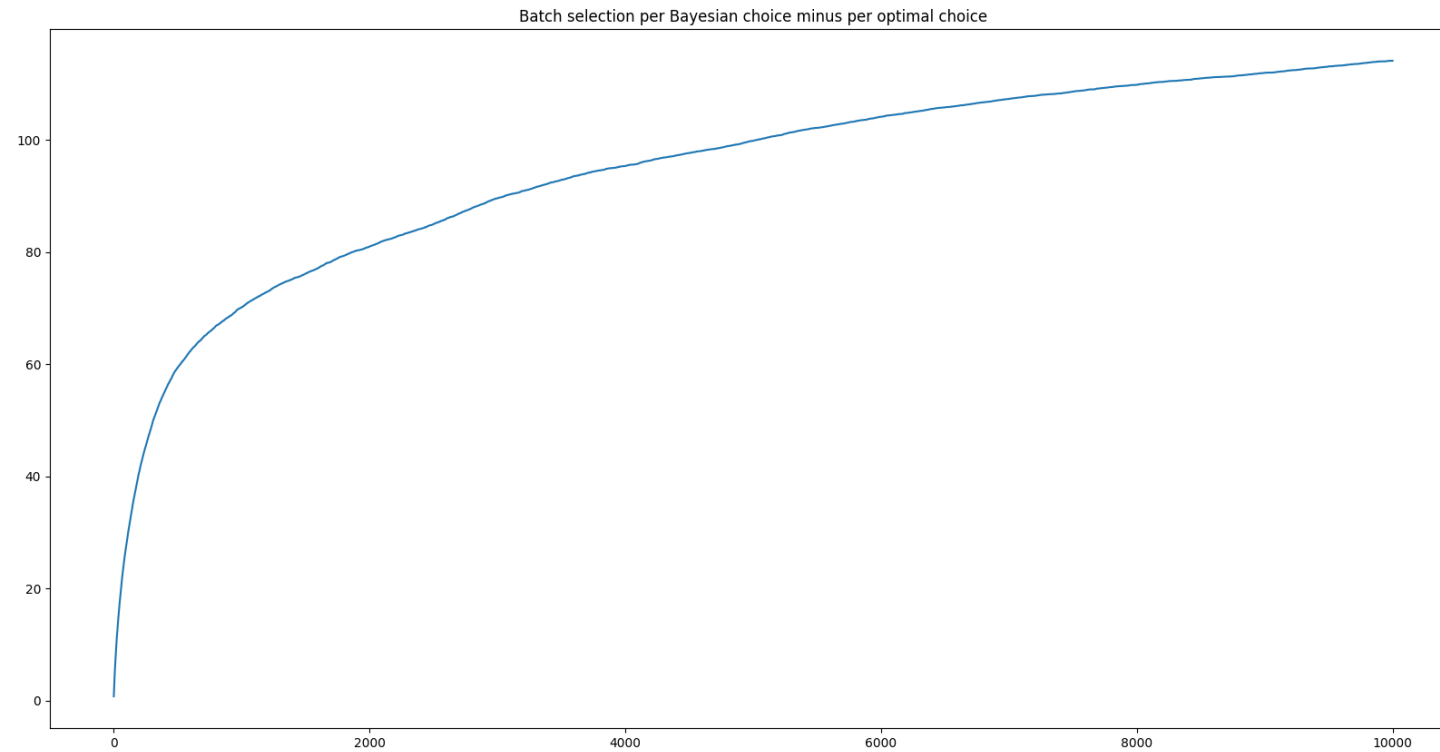
Results



Results



Results



Generalization of binomial distribution

In binomial distribution there are only two possible outcomes.

If more outcomes are possible, multinomial distribution can be used.

Binomial distribution uses one parameter, p , to encode probability of success. The probability of failure is $1 - p$.

Multinomial distribution with k outcomes requires $k - 1$ parameters. To avoid ambiguity, k parameters are given, they must sum up to 1.

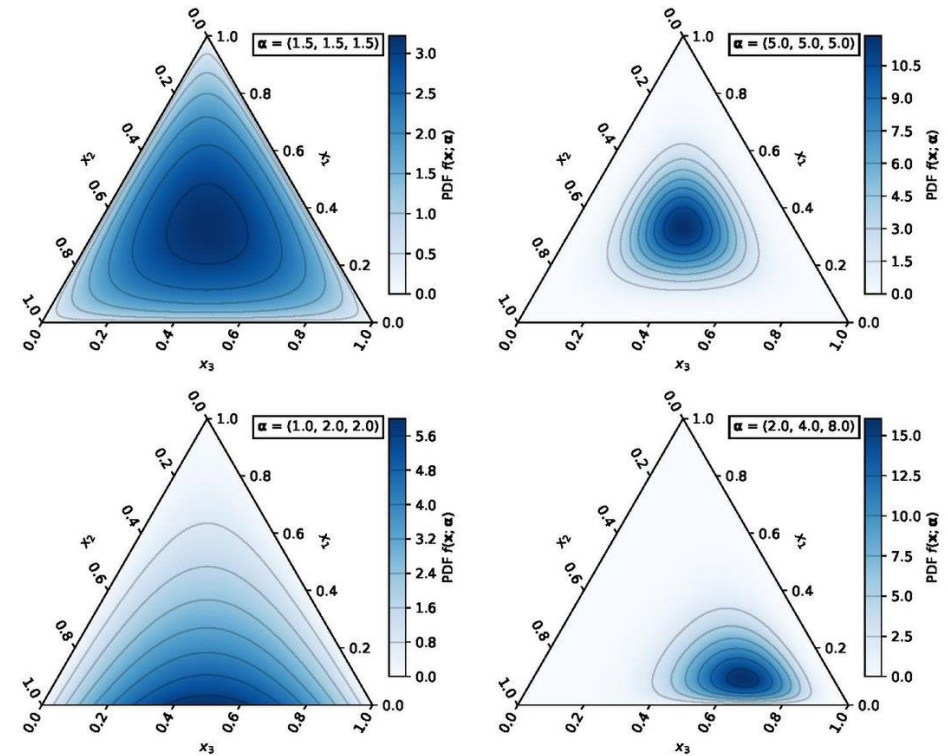
Question: why we cannot use k Binomial distributions to model situation with many outcomes?

Conjugate prior for multinomial distribution

For binomial distribution, the conjugate prior is Beta distribution.

Similarly, for multinomial distribution it is Dirichlet distribution.

For k outcomes it has k parameters.



<https://en.wikipedia.org/wiki/File:Dirichlet.pdf>

Usefulness of the Dirichlet distribution

Recall that for Beta prior and binomial likelihood it is possible to “sum up” N prior outcomes with x successes by setting prior of $Beta(x, N - x)$.

Accordingly, N prior outcomes of k -nomial process with $X = [x_1, x_2, \dots, x_k]$ outcomes, $\sum X = N$, can be summarized by $Dirichlet(X)$.

Example: mobile contracts

You work for a mobile operator. The operator has a webpage, where a user can select and settle a new contract. There are five contracts to choose from, each with different monthly payment. You analyse history of contract settlements in order to learn user preferences. In particular, you want to know what is the probability of a new user selecting particular contract.

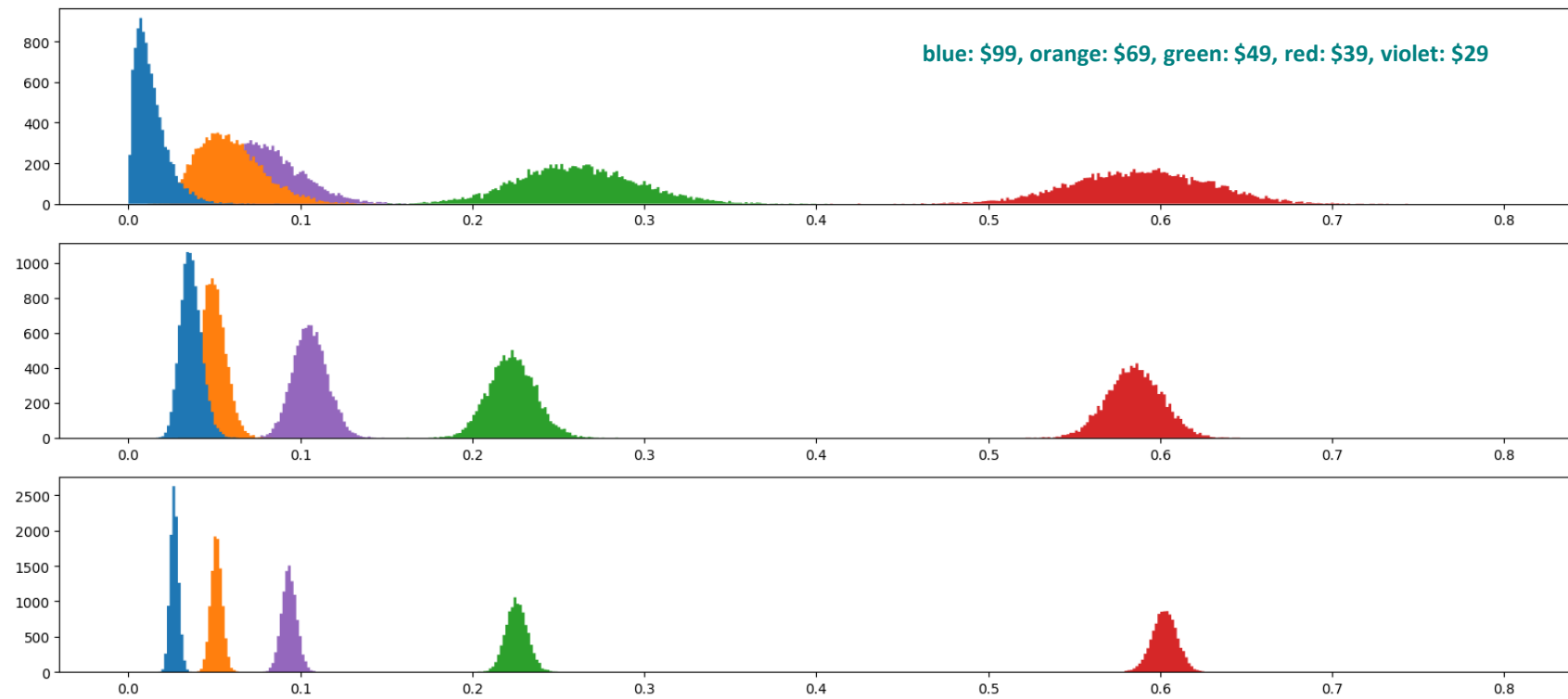
Let's denote the number of settled contracts by $X = [x_{99}, x_{69}, x_{49}, x_{39}, x_{29}]$, where the subscript indicates the value of monthly payments.

After one week worth of data is collected, you get $X = [35, 48, 215, 544, 112]$

After one month worth of data is collected, you get $X = [120, 230, 1012, 2900, 399]$

Example: mobile contracts

Distributions after one day, one week, one month.



Code

```
X1 = (2, 9, 40, 90, 12)
X2 = (35, 48, 215, 564, 102)
X3 = (120, 230, 1012, 2700, 419)

rng = np.random.default_rng()
samples1 = rng.dirichlet(X1, 10000)
samples2 = rng.dirichlet(X2, 10000)
samples3 = rng.dirichlet(X3, 10000)
```

Summing it up

Suppose you are interested in the distribution of earnings per settled contract?

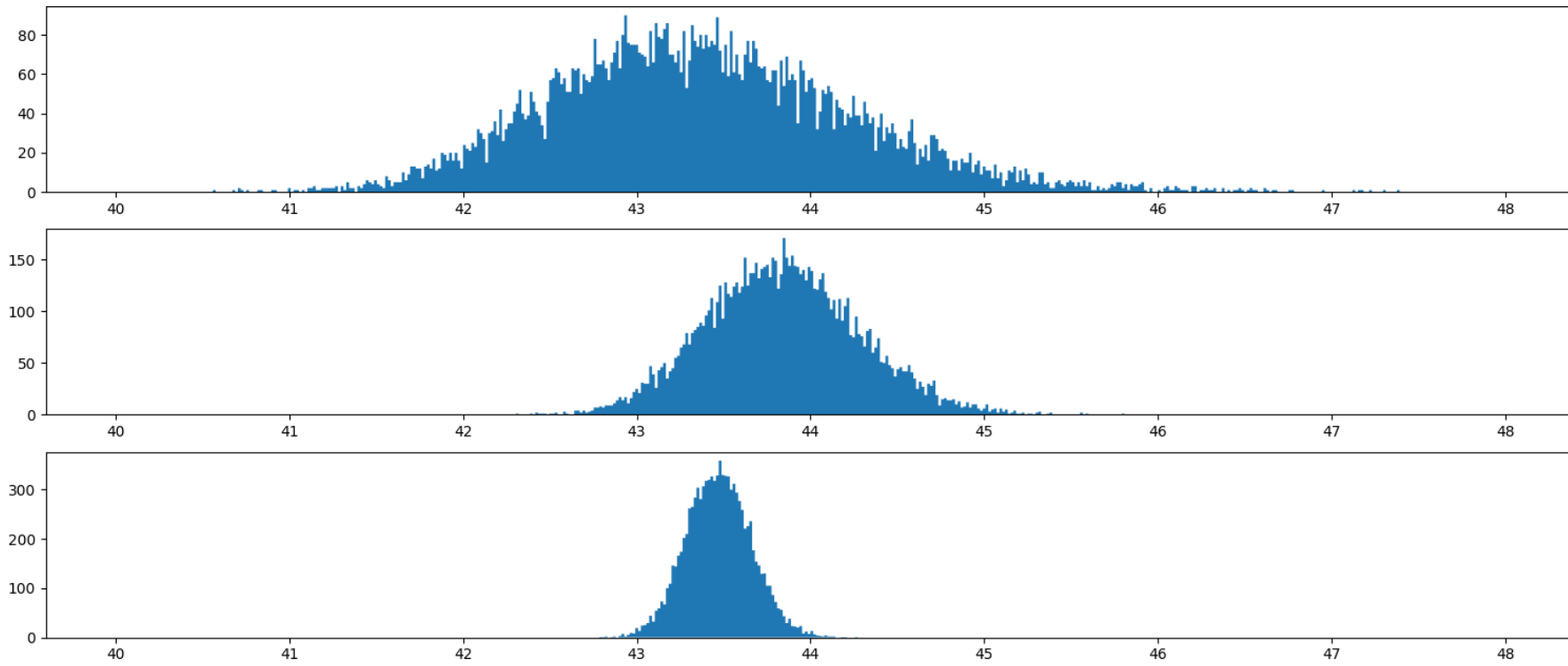
We can feed samples from distributions to a custom function that will apply \$ to probabilities:

```
def to_dollars(prices, probabilities):  
    return np.sum(prices*probabilities, axis=1)
```

```
prices = np.array([99, 69, 49, 39, 29])  
  
dollars1 = to_dollars(prices, samples1)  
dollars2 = to_dollars(prices, samples2)  
dollars3 = to_dollars(prices, samples3)
```


Summing it up

Income per customer after one day, one week, one month.

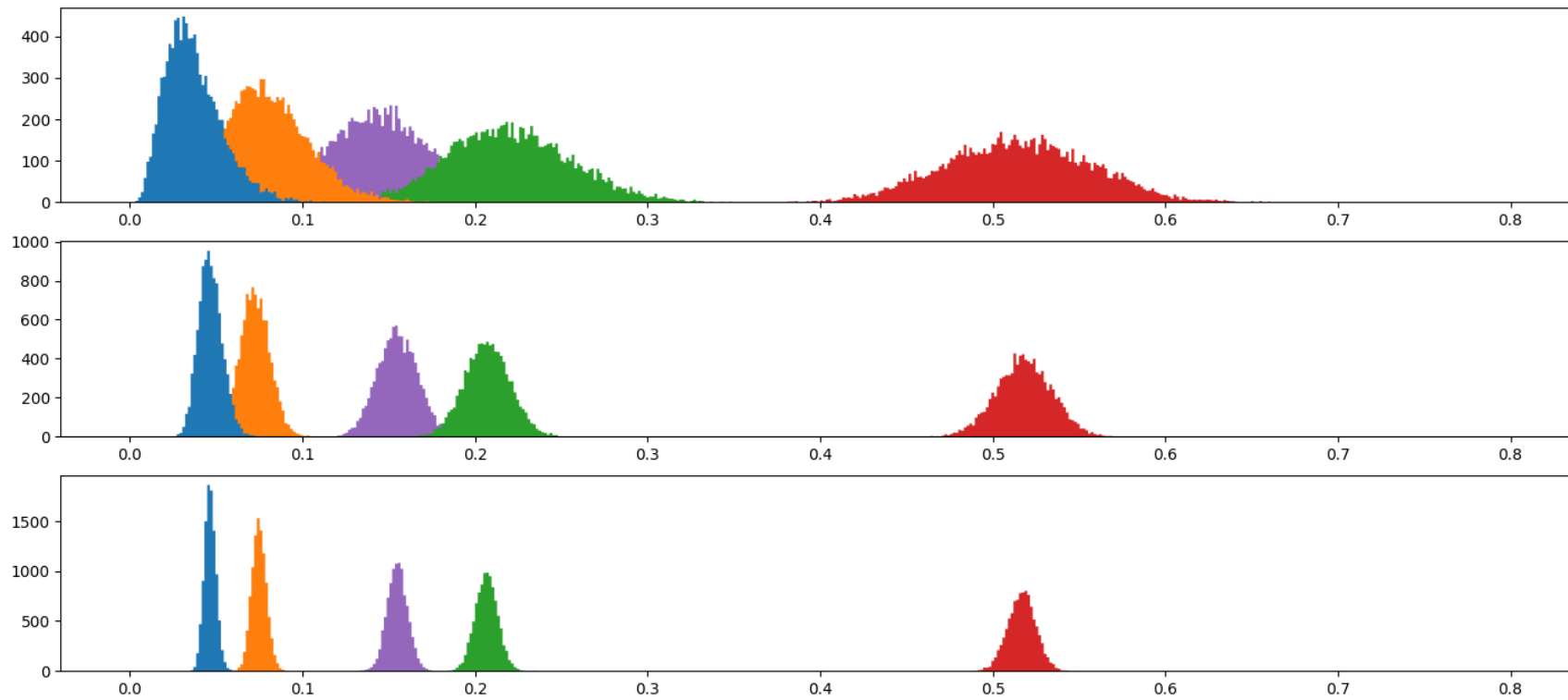


New website

The research department designed a new website, they promise the customers will be selecting more expensive plans thanks to its unique design. You want to check their claims. You allow them to set up the new website, each user is directed at random to the old or new one. You want to compare the incomes per customer.

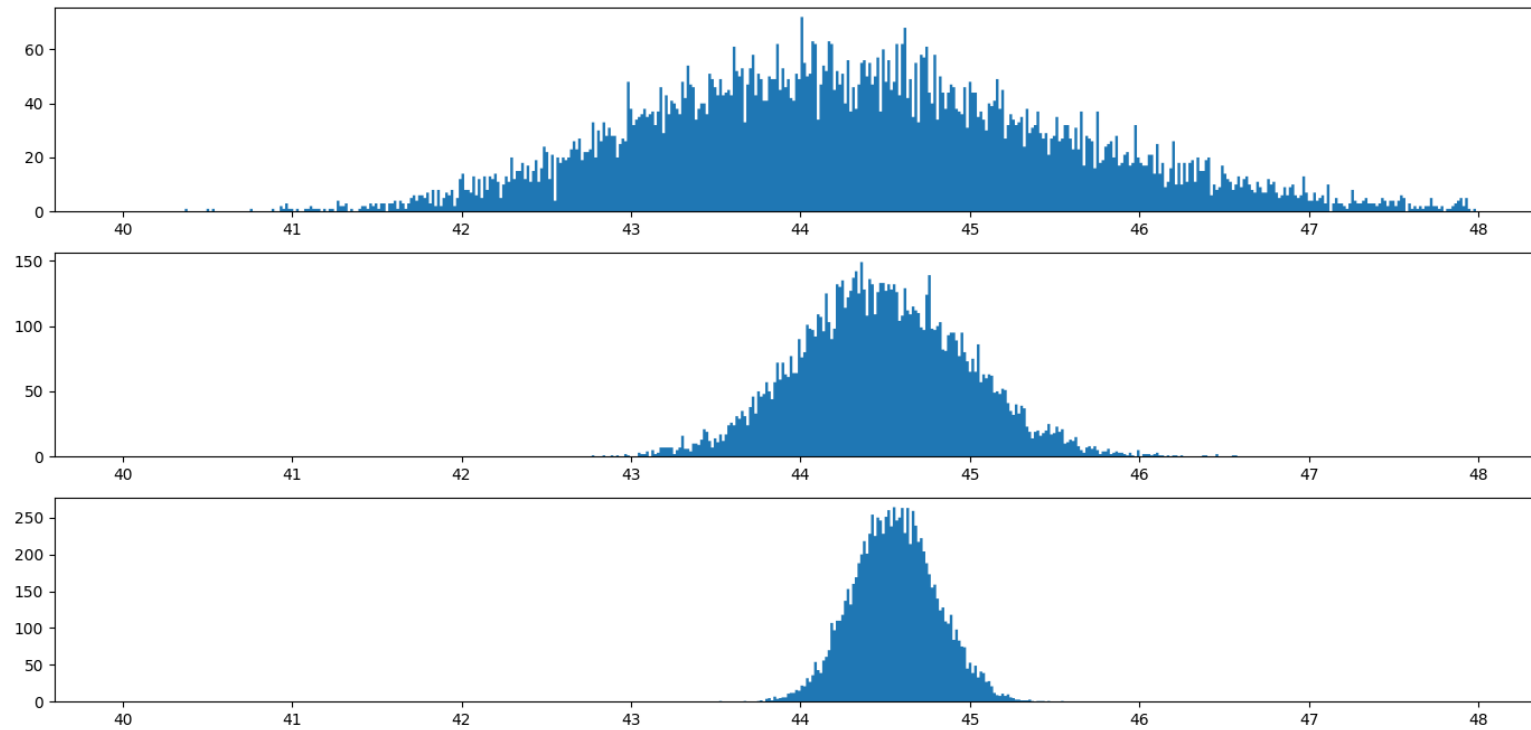
New website

Distributions after one day, one week, one month.



New website

Income per customer after one day, one week, one month.



Compare

What is the probability that the new website is better?

```
rng = np.random.default_rng()
Xsamples1 = rng.dirichlet(X1, 10000)
Ysamples1 = rng.dirichlet(Y1, 10000)

prices = np.array([99, 69, 49, 39, 29])

Xdollars1 = to_dollars(prices, Xsamples1)
Ydollars1 = to_dollars(prices, Ysamples1)

print((Ydollars1 > Xdollars1).mean())
```

After one day: 0.7452, after one week: 0.8412, after one month: 0.9997

Compare

The research department wants \$1/month during duration of the contract, for every contract, if their new website is to be used permanently. Is it worth it?

```
print(((Ydollars1-1)>Xdollars1).mean())
```

After one day: 0.4909, after one week: 0.2995, after one month: 0.617

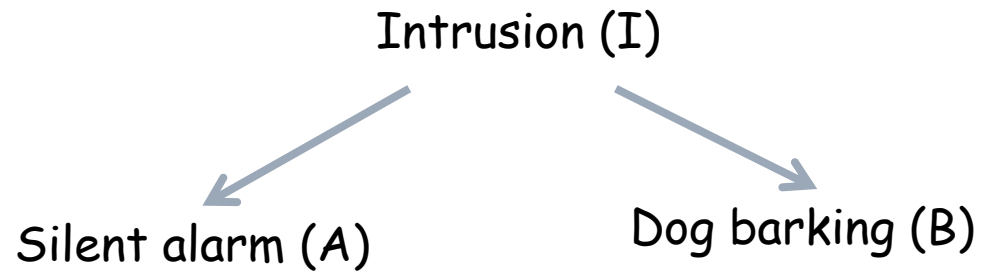
A question

What important factor we missed in our model?

Bayesian Networks

AND OTHER GRAPHICAL PROBABILISTIC MODELS

Hey, you've already seen a Bayesian network!



$$p(I) = 0.002$$

$$p(A) = 0.01$$

$$p(B) = 0.50$$

$$p(A|I) = 0.8$$

$$p(B|I) = 0.98$$

What is a Bayesian network?

- It is a probabilistic graphical model that uses Directed Acyclic Graph
 - In order to express independence between events
 - Allowing to build factorization of the joint probability distribution
 - Which enables local and, in turn, efficient computations.
-
- It can capture traits commonly associated with human reasoning.
 - It consists of qualitative and quantitative parts.

What are unique features of Bayesian networks?

- They allow to build the qualitative part based on domain expertise.
 - “Data are dumb” problem
 - Combining expertise and observations
- They handle uncertainty in a consistent manner.
- They handle missing data well.
- They are useful when the interaction between events is “localized”.

One slide about history

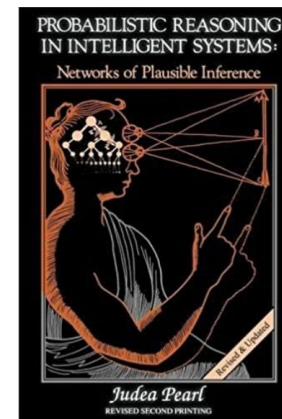
Bayesian networks are useful in constructing (normative) expert systems (i.e., systems with problem-solving ability restricted to a particular area of expertise).

One of the earliest expert systems used logical rules: if A then B. It enables to encode causality.

Soon it became evident that uncertainty should be introduced to such systems.

Early approaches included Certainty Factors and Dempster-Schaffer theory, they turned out to be not consistent with probability theory.

In response, Bayesian networks were introduced by Judea Pearl (1988).



Directed Acyclic Graphs

A **graph** is a pair $G = (V, E)$, where V is a finite set of distinct **vertices** and $E \subseteq V \times V$ is a set of **edges**.

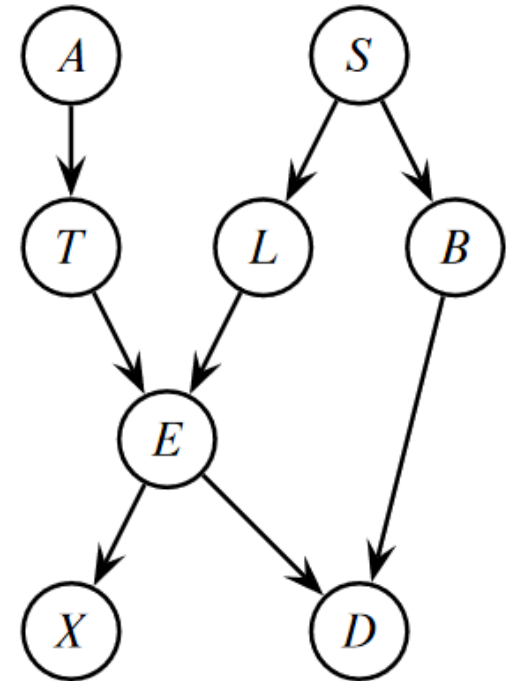
An ordered pair $(u, v) \in E$ is a **directed** edge from u to v ; u is the **parent** of v , v is the **child** of u .

☞ Directed edges are shown as arrows, undirected as lines.

If E does not contain directed edges, it is an **undirected** graph.

If E does not contain undirected edges, it is a **directed** graph.

For nodes u, v we write $u \rightarrow v$, $v \rightarrow u$ for directed edges, $u - v$ for undirected edge, $u \sim v$ for any connection between u and v .



More about graphs

A **path** is a sequence of distinct vertices such that $v_i \sim v_{i+1}$

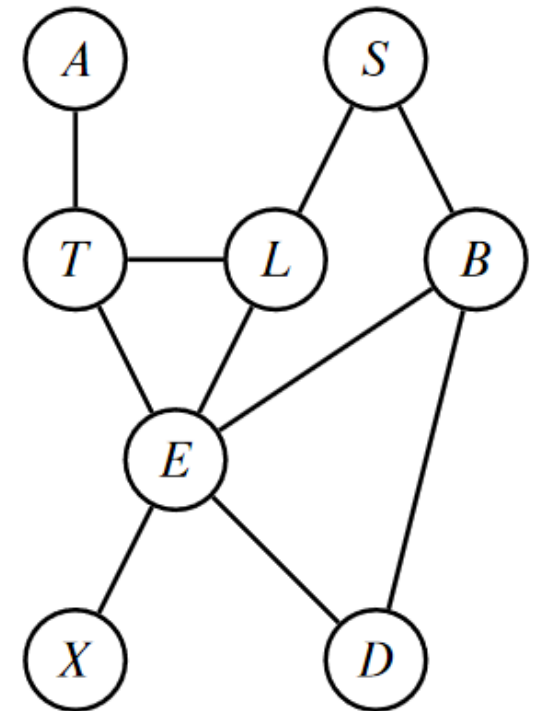
In a directed path $v_i \rightarrow v_{i+1}$, if $j > i$ then v_i is an **ancestor** of v_j and v_j is a **descendant** of v_i

A **cycle** is a path consisting of more than two vertices such that the first and the last vertex is the same.

A directed graph with no cycles is called **Directed Acyclic Graph (DAG)**.

For a DAG, if a) undirected edges are added between parents sharing common child and b) all edges are converted to undirected, the result is a **moral graph**.

In a moral graph, a set of vertices connected to v is called **Markov blanket** of v .



Vertices (nodes)

Probabilistic graphical models use vertices to describe variables and utility functions.

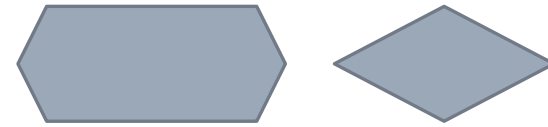
Three kind of nodes may be present:



chance variable



decision variable



value (utility function)

☞ “pure” Bayesian networks are limited to chance nodes, so *vertex* becomes a synonym for *chance variable*

Chance variables

It represents an exhaustive set of mutually exclusive events. Will be denoted by capital letters (e.g., X)

This set is the **domain** of the variable. It is denoted by $\text{dom}(X)$

The domain can be discrete or continuous.

Elements of the set are called **states** (especially for discrete domain). Will be denoted by corresponding lowercase letters (e.g., x).

☞ Similar notation may also be used for sets of variables (vectors of values)

For a set of variables $X = (X_1, X_2, \dots, X_n)$, $\text{dom}(X) = \text{dom}(X_1) \times \text{dom}(X_2) \times \dots \times \text{dom}(X_n)$

Evidence

Once the graph is constructed, for every chance variable it is possible to provide **evidence**, i.e., information (derived from real world) about states of the variable.

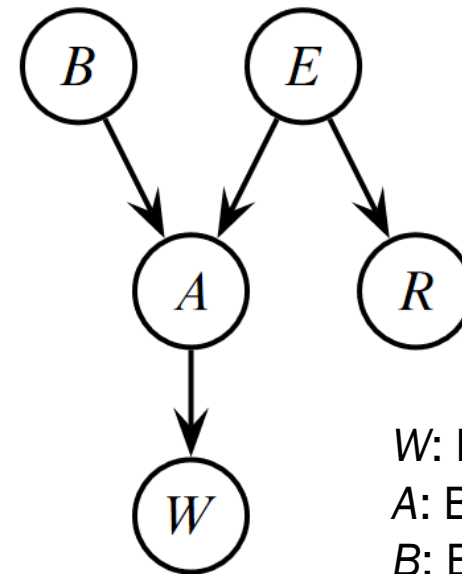
Evidence assigns values from \mathbb{R}^+ to all states of variable X .

If the assigned value is 0, it means that we are sure the variable is *not* in this state.

If the evidence assigns zeros to all but one state, it is called **hard evidence** (instantiation of X , X has been **observed**).

Different connection scenarios

Mr. Holmes is working in his office when he receives a phone call from his neighbour Dr. Watson, who tells him that Holmes' burglar alarm has gone off. Convinced that a burglar has broken into his house, Holmes rushes to his car and heads for home. On his way, he listens to the radio, and in the news, it is reported that there has been a small earthquake in the area. Knowing that earthquakes have a tendency to make burglar alarms go off, he returns to his work (*Burglary or Earthquake, Pearl 1988*)



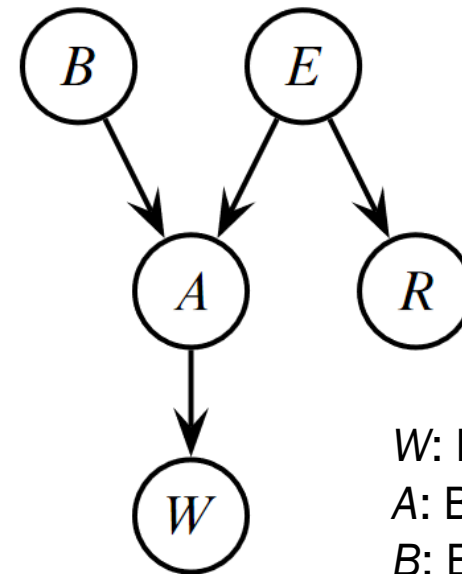
W: Phone call from Watson
A: Burglary alarm
B: Burglary
R: Radio news
E: Earthquake

Different connection scenarios

Serial connections: $B \rightarrow A \rightarrow W$, $E \rightarrow A \rightarrow W$

Diverging connection: $A \leftarrow E \rightarrow R$

Converging connection: $B \rightarrow A \leftarrow E$

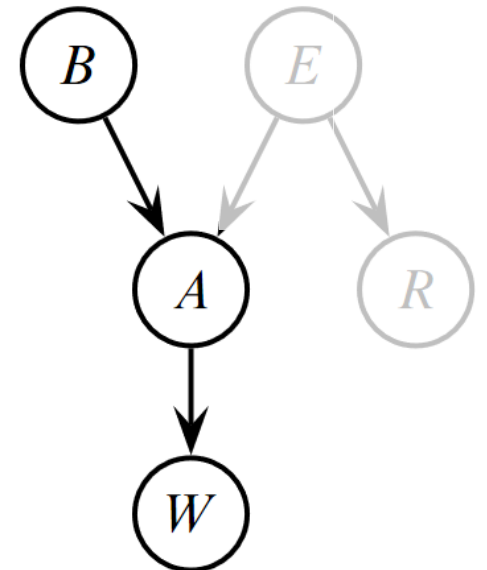


W: Phone call from Watson
A: Burglary alarm
B: Burglary
R: Radio news
E: Earthquake

Serial connection

Consider the following situations:

1. Holmes has **no information about alarm**. He learns somehow (e.g., because police contacted him) that there was a burglary in his house. Does it influence his belief that Watson will call him?
2. Holmes has **no information about alarm**. Watson calls him. Does it influence Holmes's belief that there was burglary in his house?
3. Holmes **knows that there was alarm** in his house. He learns somehow (e.g., because police contacted him) that there was a burglary in his house. Does it influence his belief that Watson will call him?
4. Holmes **knows that there was alarm** in his house. Watson calls him. Does it influence Holmes's belief that there was burglary in his house?



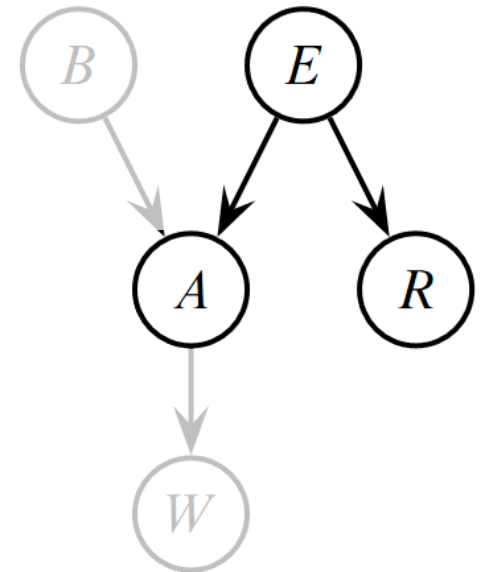
Serial connection

In serial connection, data flow between “edge” nodes is possible only if the state of the “middle” node is not known.

Diverging connection

Consider the following situations:

1. Holmes **has no information about occurrence of an earthquake**. He hears radio news about there being an earthquake. Will it influence his belief in alarm going off in his house?
2. Holmes **has no information about occurrence of an earthquake**. He learns that there was an alarm in his house. Will it influence his belief in radio news stating that there was an earthquake?
3. Holmes **knows that there was an earthquake**. He hears radio news about there being an earthquake. Will it influence his belief in alarm going off in his house?
4. Holmes **knows that there was an earthquake**. He learns that there was an alarm in his house. Will it influence his belief in radio news stating that there was an earthquake?



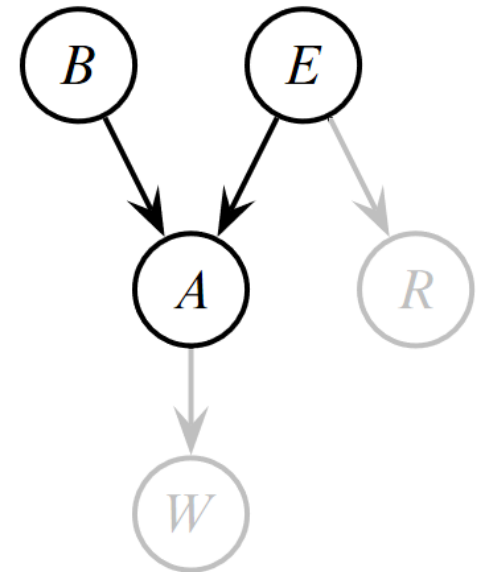
Diverging connection

In diverging connection, data flow between “edge” nodes is possible only if the state of the “middle” node is not known.

Converging connection

Consider the following situations:

1. Holmes has **no information about alarm**. He learns about a burglary in his house. Does this influence his belief in there being an earthquake?
2. Holmes has **no information about alarm**. He learns about an earthquake. Does this influence his belief in there being burglary in his house?
3. Holmes **knows that there was alarm** in his house. He learns about a burglary in his house. Does this influence his belief in there being an earthquake?
4. Holmes **knows that there was alarm** in his house. He learns about an earthquake. Does this influence his belief in there being burglary in his house?



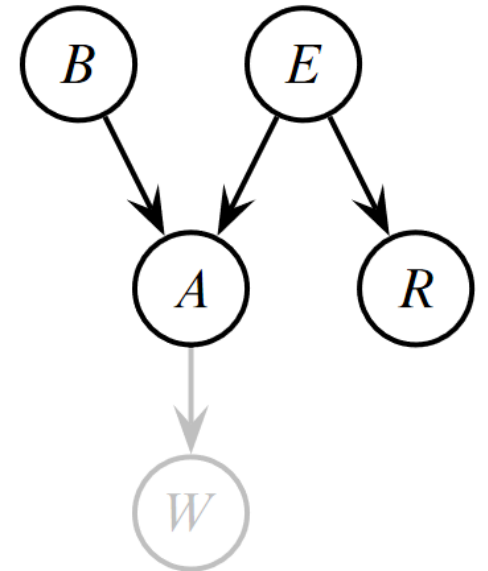
Converging connection

In converging connection, data flow between “edge” nodes is possible only if the state of the “middle” node is known.

Converging + diverging connection (explaining away)

Consider the following situation:

Holmes learns that the alarm went off in his house. He updates his beliefs about burglary and earthquake. Later, he hears the radio announcement stating that there was an earthquake. Does it change his belief in there being a burglary in his house?



Causality

Pure logic statements are useful in encoding causality: if you hit your finger with a hammer, it will hurt.

However, such statement is unidirectional: it does not allow us to conclude that if the finger hurts, it must have been hit with a hammer.

In real life, we usually think in a “bidirectional manner”: if it rained, the grass is wet. But also, if we observed wet grass, it supports our belief that it rained.

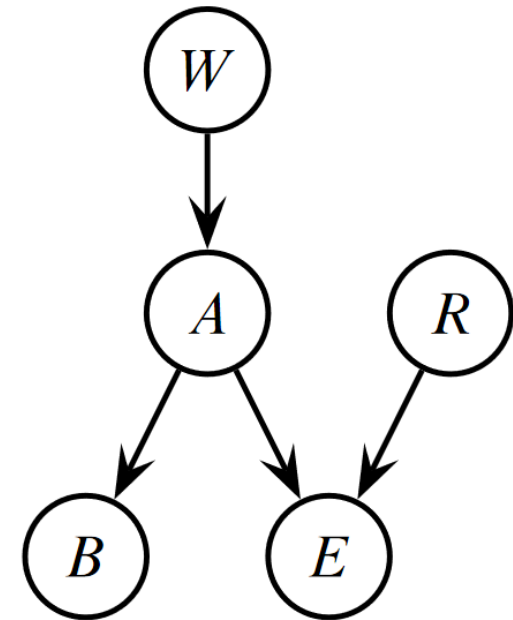
It is important to remember what is cause and effect and encode it properly.

1st reason to correctly encode causality

Assume the model has been built violating the causality principle:

Let's try to apply the “diverging edges” principle to nodes *B*, *A*, *E*. It turns out that when we don't know whether the alarm sounded, learning about burglary changes our belief in earthquake.

When we apply the “converging edges” principle to nodes *A*, *E*, *R* it turns out that when we already know there was an earthquake, hearing radio report about an earthquake changes our belief in the alarm going off.



2nd reason to correctly encode causality

It is usually quite easy (or at least feasible) to specify conditional probabilities following causal relationships:

- What is the probability of the alarm going off given there is a burglary?
- What is the probability of developing lung cancer given a person is a smoker?

If these are reversed, it is much more complicated:

- What is the probability of a burglary given the alarm went off?
- What is the probability that a person is a smoker if she/he developed lung cancer?

d-separation

d-separation (directional-separation) summarizes (in)dependence of variables, including situations when some other variables are observed

Notation: $u \perp v$, $u \perp v \mid W$. W denotes a set of nodes that has been observed

We trace all possible paths between u and v , if all are blocked, then u and v are d-separated.

A path is blocked when it contains:

- a serial connection $a \rightarrow b \rightarrow c$ or a diverging connection $a \leftarrow b \rightarrow c$ and the state of b is known ($b \in W$)
- a converging connection $a \rightarrow b \leftarrow c$ and neither the state of b , nor of its descendants is known ($b \notin W, \text{de}(b) \notin W$)

d-separation

Are B and R d-separated when no variables have been observed?

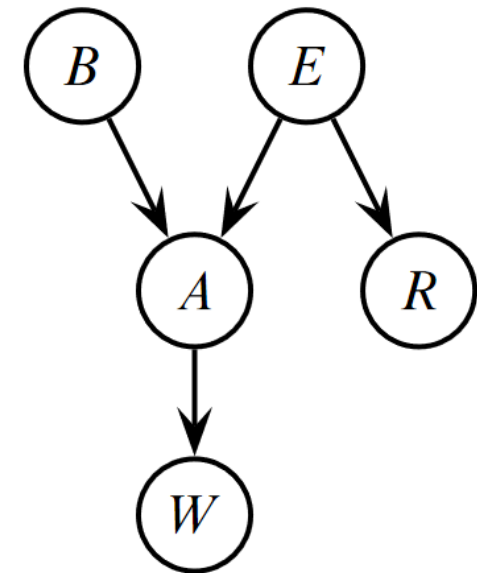
The only path between B and R is $B \rightarrow A \leftarrow E \rightarrow R$. A is a “collider”, neither its state nor the state of its descendant (W) is known. $B \perp R$

Are B and R d-separated when W has been observed?

As above, but the descendant of the collider is in the set of observed variables. $B \not\perp R$

Are W and R d-separated when no variables have been observed?

The only path between W and R is $W \leftarrow A \leftarrow E \rightarrow R$. Only serial and diverging connections with no observed variables. $W \not\perp R$



Numerical specification

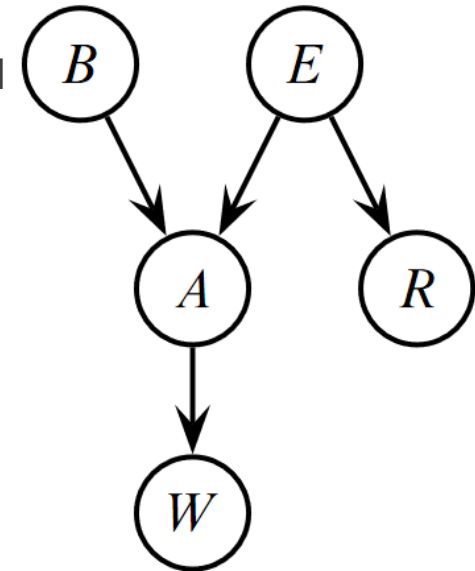
The structure of the graph determines the kind of probabilities that are needed:

- If the node has no parents, unconditional probabilities of its states are needed
- If the node has parents, conditional probabilities of its states given the states of its parents are needed.

Assuming the network is discrete, how many numbers will be needed?

For a node V with set of parents $P = \{P_1, P_2, \dots, P_n\}$ it is given by $|\text{dom}(V \cup P)|$

Assuming each variable is two-state, how many numbers are needed for nodes E , R and A ?



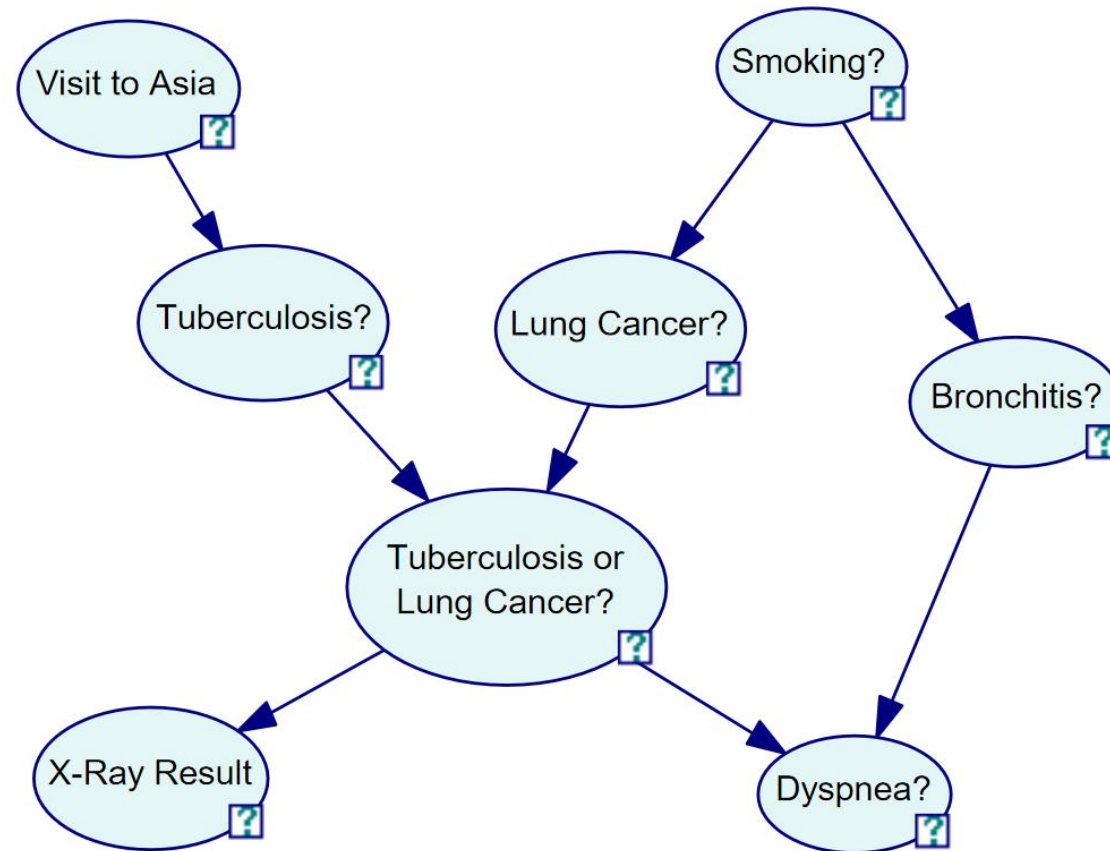
Bayesian network construction

The network can be constructed:

- Manually (both the structure and probabilities)
- Manually regarding structure, using machine learning regarding probabilities
- Completely using machine learning

Regardless of the approach, the structure is constructed first.

Visit to Asia



Rules for good structure

Rule 0: use probabilistic networks when they are well suited for the problem:

- Well defined variables (“burglary” vs. “is tall”).
- Identifiable and limited cause-effect relations, preferably static (i.e., not changing over time).
- Availability of data for the unconditional and conditional probabilities.
- Uncertainty associated with the cause-effect relations (if all conditional probabilities are at 0 or 1, use another tool).
- Needed to be used more than once.

Rules for good structure

It should not be possible to determine state of a variable based on a state of a single other variable (variables should be *unique*).

Example: dice roll. variable A: {even, not even}, variable B: {odd, not odd}. Variables should be merged into C: {even, odd}.

A variable must represent an *exhaustive* set of *mutually exclusive* events.

Example: dice roll. variable A: {1, 2, 3, 4, 5} is not exhaustive. Variable B: {1, 2, 3, 3 or more} does not describe mutually exclusive events.

The variable should be clearly defined.

Types of variables

Problem variables – for which we want to compute posterior probabilities

Information variables – for which data (evidence) may be available

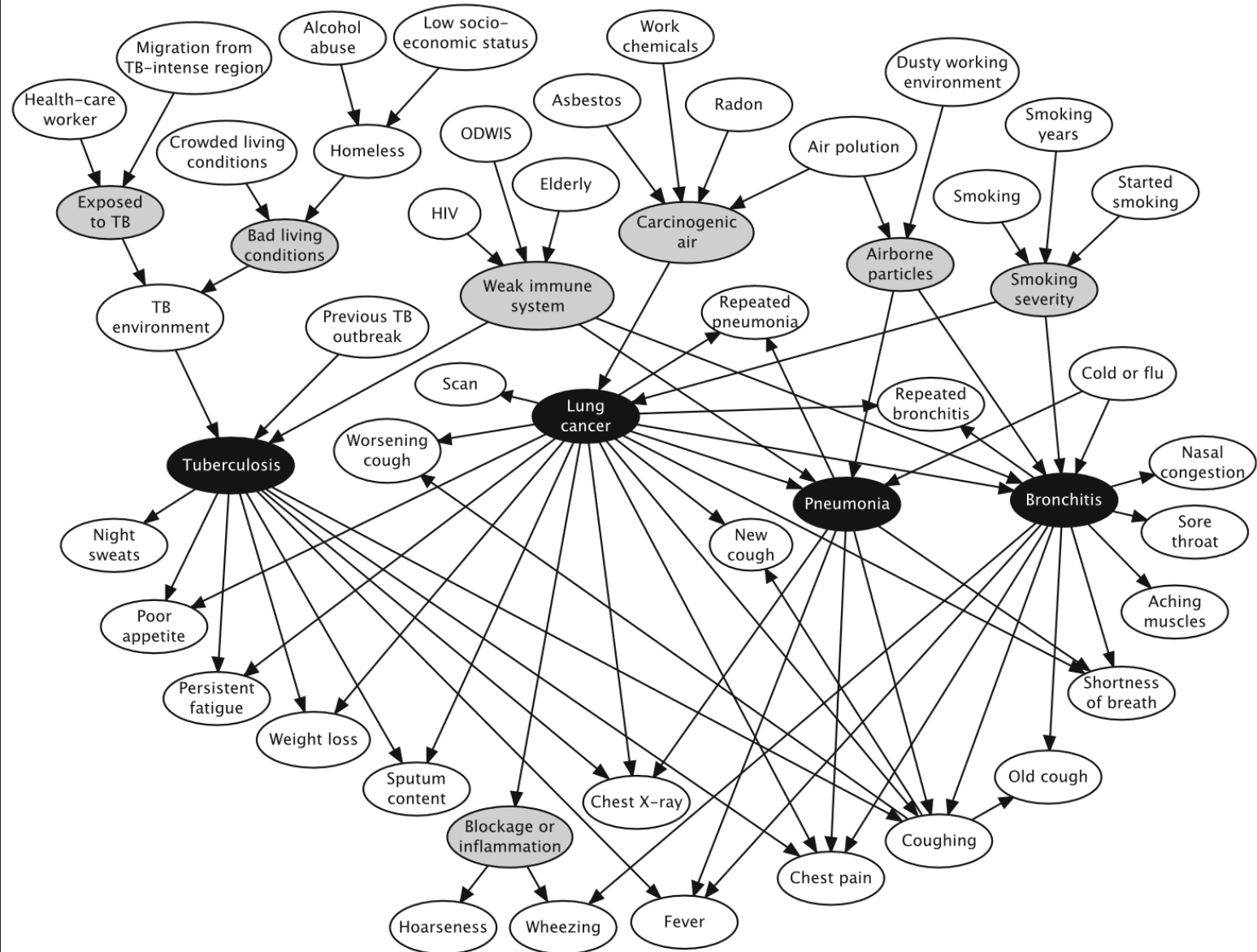
- Background information – variables having causal influence on the problem variables. Their state is usually available before the occurrence of the problem.
- Symptom information – variables that are descendants of problem (and background) variables – problem variables have causal influence on the symptom variables. Their state will change due to the occurrence of the problem.

Mediating variables – unobserved variables that are not of interest themselves, but that play important role in the structure of the network

Chest clinic model

Uffe B. Kjærulff, Anders L. Madsen

Bayesian Networks and Influence
Diagrams: A Guide to Construction
and Analysis, Second Edition



Numbers, numbers

People rarely think in terms of probabilities.

Techniques exist that “translate” the way humans express their beliefs into numbers.

These approaches can be used for unconditional and conditional probabilities.

Probability wheel

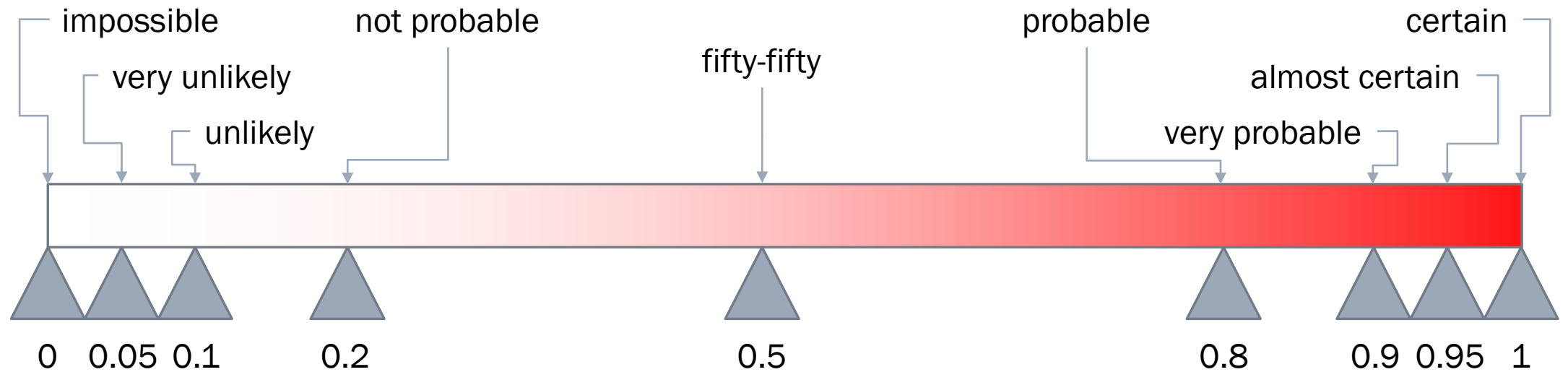
A pie-chart with number of wedges equal to the number of variable states.

The domain expert is asked to adjust the sizes of the wedges so as to best match her/his beliefs.

<https://www.nctm.org/adjustablespinner/>



Mapping of verbal statements



Game-based approach

Task: provide conditional probability of being caught by police when speeding between Łódź and Warsaw.

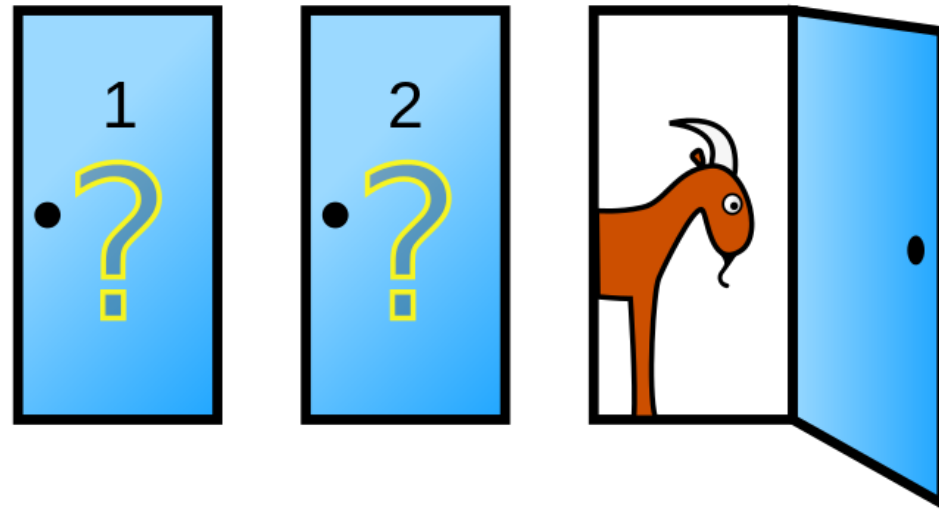
Consider two games (bets):

- Mr. Fast Guy will be speeding between Łódź and Warsaw. If he gets caught, you will receive \$100.
- An urn contains n red balls and $100 - n$ white ones. If you draw a red ball, you will receive \$100.

Adjust n so that these two games are indifferent to you.

Monty Hall problem

Suppose you're on a game show, and you're given the choice of three doors. Behind one door is a car, behind the others, goats. You pick a door, say #1, and the host, who knows what's behind the doors, opens another door, say #3, which has a goat. He says to you, "Do you want to pick door #2?" Is it to your advantage to switch your choice of doors?



Hard choice

At the grocer's we can choose a box of 5 oranges from two producers. Let X_1 and X_2 denote the number of good oranges in the box from, respectively, the first and the second producer. From previous experience we can estimate the distribution of these random variables as:

x	0	1	2	3	4	5
$p(x_1) = P(X_1 = x)$	0	0	0	0.1	0.3	0.6
$p(x_2) = P(X_2 = x)$	0.1	0	0	0	0	0.9

Which choice is better?

Utility

Let's compute the expected value for each distribution:

$$E(X_1) = 0 \cdot 0 + 1 \cdot 0 + 2 \cdot 0 + 3 \cdot 0.1 + 4 \cdot 0.3 + 5 \cdot 0.6 = 4.5$$

$$E(X_2) = 0 \cdot 0.1 + 1 \cdot 0 + 2 \cdot 0 + 3 \cdot 0 + 4 \cdot 0 + 5 \cdot 0.9 = 4.5$$

Utility function assigns to each possible state a value, representing “happiness” of the subject if this state comes true

- may be based on strict computations
- may be purely subjective
- is often expressed in currency

Examples of utility functions

Case 1: own consumption

- I want badly to eat some oranges. However, I know that if I buy a box of five, I will eat no more than 3 of them, the rest will rot.

Case 2: grandchildren

- I am a grandpa that wants to give my grandchildren a treat. There are five of them. I want to give them oranges, however, I must give an orange to everyone, as otherwise they will fight between them.

x	0	1	2	3	4	5
$U_1(x)$	0	5	10	15	15	15
$U_2(x)$	0	0	0	0	0	15

Expected utility

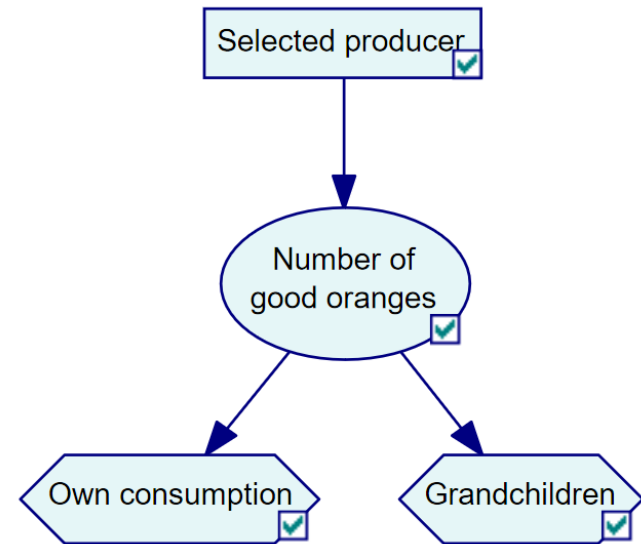
In the discrete case, the expected utility can be computed by multiplying the utility of the state by its associated probability and summing up the results.

x	0	1	2	3	4	5	
$p(x_1) = P(X_1 = x)$	0	0	0	0.1	0.3	0.6	
$p(x_2) = P(X_2 = x)$	0.1	0	0	0	0	0.9	
$U_1(x)$	0	5	10	15	15	15	
$U_2(x)$	0	0	0	0	0	15	$E(U)$
$U_1(x_1)$	0	0	0	1.5	4.5	9	15
$U_1(x_2)$	0	0	0	0	0	13.5	13.5
$U_2(x_1)$	0	0	0	0	0	9	9
$U_2(x_2)$	0	0	0	0	0	13.5	13.5

Influence diagrams

Bayesian network supplemented by decision and utility nodes.

Allows to analyse expected utility of each decision (given current evidence).

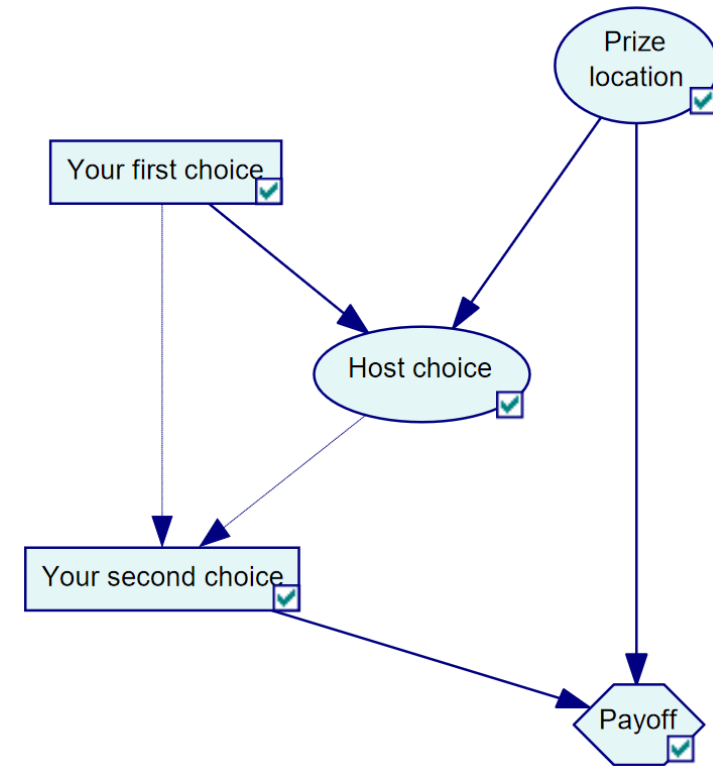


Monty Hall revisited

Monty Hall problem can be modelled in terms of influence diagrams.

Dashed-line (or thinner) arrows represent informational arcs: they show what data are available when the decision is to be made (temporal precedence)

- In particular, decision nodes (if multiple) must be connected by informational arcs to encode the order of decisions.

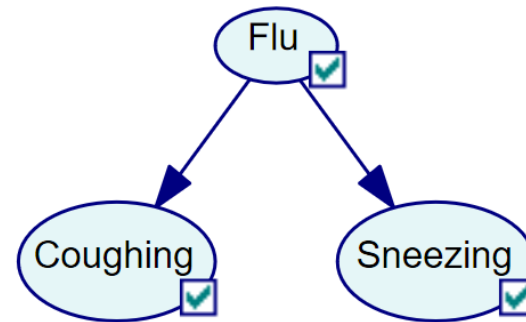


Have data, will use them

Learning probabilities from data is straightforward when evidence for all nodes of the network is available.

Unconditional probabilities will be frequencies of states occurrence in data.

Conditional probabilities – similarly, but separately for records with different combinations of parent's states.



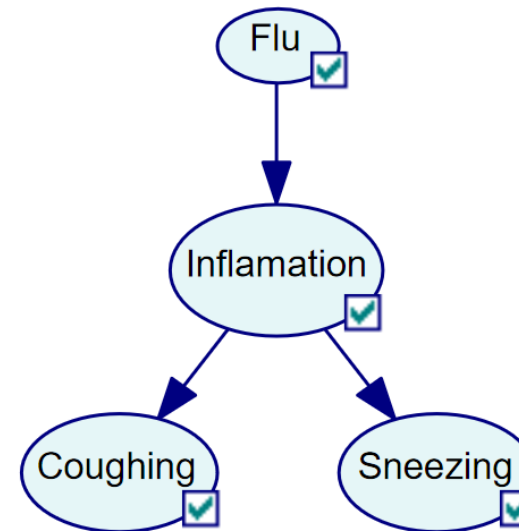
Flu	Coughing	Sneezing
1	1	1
1	1	0
1	1	1
0	0	0
1	0	0
1	1	1
0	0	0
1	1	1
0	0	0

Missing data

In many cases the records do not contain all data.

This is typical for mediating (latent) variables.

Also, for different records there some of the usually observed variables may be missing.



Flu	Coughing	Sneezing
1	1	1
1		0
1	1	1
0	0	0
1	0	0
1		1
0	0	0
1	1	1
0	0	

EM algorithm

The algorithm usually employed is called Expectation-maximization (Dempster, Laird, Rubin, 1977)

After a parameter initialization phase (using random values or values from a uniform distribution) it consists of two steps :

- compute probabilities of missing values (using current parameters)
- compute a new set of parameters (using real data along with the probabilities just computed)

The algorithm alternates between these steps until convergence. It can be proved that this procedure really finds a maximum, although it cannot be guaranteed that this will be a global maximum.

Combining previous and new knowledge

A useful feature is that the algorithm does not have to start by randomizing parameters.

- Instead, the parameters currently present in the network may be used as the starting point

This enables to combine previous knowledge (whether elicited from an expert or obtained through learning from data) with new data.

It is possible to state how important this previous knowledge is, by tweaking the “equivalent sample size” parameter.

- This corresponds to augmenting the learning data with the desired number of samples generated from the current network.

It is also possible to “freeze” some parameters, limiting learning to the desired variables.

The End

LIFE IS LIKE A BOX OF CHOCOLATES. YOU NEVER KNOW WHAT YOU'RE GONNA GET.

But you can always compute the probability.