Data don't speak for themselves
a Bayesian course

JOÃO P. FARIA
Institute of Astrophysics and Space Sciences, Porto
February 12, 2016

Abstract

Bayesian statistics is rising in popularity in the astrophysical literature. It is no longer a debate: "work in Bayesian statistics now focuses on applications, computations, and models. Philosophical debates [...] are fading to the background" [1]. This is happening for two main reasons: faster computers and more complex models. In order to keep up, it is important to understand the fundamentals of Bayesian statistics, but it is as important to know how to deal with data analysis applications. In this course I want to provide a brief introduction to advanced concepts in Bayesian statistics. Emphasis will be on *intuition* and *computation*. No coin tossing, only real applications that relate to our day-to-day problems.

Contents

A (tiny) primer on philosophy **3**

1 The basics of probability theory **3**
1.1 Some familiarity 4

2 Assigning probabilities **6**
2.1 The prior 6
2.1.1 An example 9
2.2 The likelihood 11
2.2.1 An example 12

3 MCMC **16**
3.1 A basic MCMC implementation 16

Before you read

This document is not finished!
What I have written here is not completely original work: I follow closely, and base the discussion on, many different sources. Because it's not finished, I have not yet made fair attribution to all these sources.

I mean no infringement of copyright and no plagiarism.

A (tiny) primer on philosophy

While preparing for this course, I asked people what it was they wanted to learn about. Much to my dismay, no one answered with "the philosophy of Bayesian statistics". Most want to learn about the "theory" and "applications" (read MCMC). They want to know how they can *use* Bayesian statistics in their work[1].

Fair enough. But bypassing phylosophy means making assumptions. Therefore, I shall introduce here Jaynes' *robot*, an imaginary being whose brain reasons according to certain definite rules [3]. These rules will simply be stated, not derived and not defended (Sect. 1). They will be taken as being true. Then, everything that follows logically from them, everything the robot does, will be true.

In the words of [2], these rules make Bayesian inference "more like integration" in that it provides a "collection of tools for finding definitive answers to well-posed problems". Posing the problem will require subjective knowledge; solving it will not.

1 The basics of probability theory

Here we present the rules of the game, following very closely (too closely!) the presentation in [4]. Using Bayesian statistics to analyse data (some might call it doing probabilistic inference) means working with likelihoods, prior probabilities, posterior probabilities and marginalization of nuisance parameters. All this will be explained later but the options we have when working with these mathematical objects are strongly constrained by the rules of probability calculus.

> *On voit, par cet Essai, que la théorie des probabilités n'est, au fond, que le bon sens réduit au calcul;*
>
> – Pierre-Simon Laplace

If we have a continuous parameter a, and a probability distribution function $p(a)$ for[2] a, it must obey the normalization condition

$$1 = \int p(a)\, da \tag{1}$$

where the integral is limited by the domain of a.

Often times we will have more than one parameter. Even if we *condition* $p(a)$ on some particular value of another parameter b, that is, we ask for

[1] I later read [2] and realised this is not the case for our institute only.

[2] I say *for* instead of *of*, but I promised no philosophy...

$p(a|b)$ (read "the pdf for a given b"), it must obey the same normalization

$$1 = \int p(a|b)\, da \tag{2}$$

If we have a probability distribution for two things, $p(a, b)$ (read "the pdf for a and b"), you can always factorize it into two distributions, one for a, and one for b given a or the other way around:

$$p(a, b) = p(a)\, p(b|a) \tag{3}$$
$$p(a, b) = p(b)\, p(a|b) \tag{4}$$

These two factorisations together lead[3] to Bayes' theorem:

$$p(a|b) = \frac{p(b|a)\, p(a)}{p(b)}. \tag{5}$$

Conditional probabilities factor just the same as unconditional ones

$$p(a, b|c) = p(a|c)\, p(b|a, c) \tag{6}$$
$$p(a, b|c) = p(b|c)\, p(a|b, c) \tag{7}$$
$$p(a|b, c) = \frac{p(b|a, c)\, p(a|c)}{p(b|c)} \tag{8}$$

where we just carried the condition c through all the terms.

You can integrate out or *marginalize* variables you want to get rid of (or *not* infer) by integrals like

$$p(a|c) = \int p(a, b|c)\, db \tag{9}$$
$$p(a|c) = \int p(a|b, c)\, p(b|c)\, db \tag{10}$$

where the second is a factorized version of the first. Remember that, since b can be a very high-dimensional mathematical object (a set of parameters), integrals like these can be extremely difficult to calculate in practice.

1.1 Some familiarity

Let us write some of the preceding equations with more familiar terms[4]. We have data D and a set of parameters θ that we are interested in learning about. In all our analyses we condition on information $\mathcal{I}$ that we have about the world. Then we can write Eq. (8) as

$$p(\theta|D, \mathcal{I}) = \frac{p(\theta|\mathcal{I})\, p(D|\theta, \mathcal{I})}{p(D|\mathcal{I})} \tag{11}$$

[3] Bayes' theorem is a consequence of the previous equations, it is not assumed as a rule. That's why it's called a theorem, actually.

[4] Only if you have heard about Bayesian statistics, otherwise just different terms.

The terms in this equation are usually called

$p(\theta|D,\mathcal{I})$ the posterior distribution

$p(\theta|\mathcal{I})$ the prior distribution

$p(D|\theta,\mathcal{I})$ the likelihood

$p(D|\mathcal{I})$ the evidence, sometimes denoted with a $\mathcal{Z}$.

but calling them this may hide something important (see Section 2.2).

From Eq. (2) we can derive[5] that

$$p(D|\mathcal{I}) = \mathcal{Z} = \int p(\theta|\mathcal{I})\, p(D|\theta,\mathcal{I})\, d\theta \tag{12}$$

Because $\mathcal{Z}$ does not depend on θ, you might see many times Eq. (11) written as

$$p(\theta|D,\mathcal{I}) \propto p(\theta|\mathcal{I})\, p(D|\theta,\mathcal{I}), \tag{13}$$

so **the posterior is proportional to the likelihood times the prior**. Nevertheless, try to stick with Eq. (11); $\mathcal{Z}$ contains in it a big deal of information.

think about it

Eq. (13) means that the likelihood and the prior can be defined up to an arbitrary multiplicative constant (i.e. not a function of θ).

For example, imagine we are interested in comparing two models M_1 and M_2 which have parameters θ_1 and θ_2, respectively. We still only have data D. Bayes' theorem works the same:

$$p(M_i|D,\mathcal{I}) = \frac{p(M_i|\mathcal{I})\, p(D|M_i,\mathcal{I})}{p(D|\mathcal{I})} \tag{14}$$

and the ratio of model probabilities is

$$\frac{p(M_1|D,\mathcal{I})}{p(M_2|D,\mathcal{I})} = \frac{p(M_1|\mathcal{I})\, p(D|M_1,\mathcal{I})}{p(M_2|\mathcal{I})\, p(D|M_2,\mathcal{I})} \tag{15}$$

$$= \frac{p(M_1|\mathcal{I}) \int p(D,\theta_1|M_1,\mathcal{I})\, d\theta_1}{p(M_2|\mathcal{I}) \int p(D,\theta_2|M_2,\mathcal{I})\, d\theta_2} \tag{16}$$

$$= \frac{p(M_1|\mathcal{I}) \int p(\theta_1|M_1,\mathcal{I})\, p(D|\theta_1,M_1,\mathcal{I})\, d\theta_1}{p(M_2|\mathcal{I}) \int p(\theta_2|M_2,\mathcal{I})\, p(D|\theta_2,M_2,\mathcal{I})\, d\theta_2} \quad . \tag{17}$$

See how the evidence (of both models) turns out to be important!

[5] Multiply both sides of Eq. (8) by $p(D|\mathcal{I})$ and integrate over θ, noting that $p(D|\mathcal{I})$ does not depend on θ and that the posterior obeys Eq. (2).

The term

$$\frac{p(M_1|\mathcal{I})}{p(M_2|\mathcal{I})}$$

is the ratio of prior probabilities for the two models. If we believe they are equally likely at the start, then this is equal to 1.

think about it

θ_1 and θ_2 can be any set of parameters and, in particular, they can have different sizes. Say model M_1 has 2 parameters $\theta_1 = (a, b)$ and model M_2 only has one parameter $\theta_2 = (c)$. Then the integrals in Eq. (17) are of different dimensionality. That's fine, probability theory doesn't care. Enough of that "divide by the degrees of freedom" nonsense! [5]

2 Assigning probabilities

Now that we have a set of rules with which we can manipulate our distributions, it is useful to learn how we can assign values to all these terms and actually start calculating things.

2.1 The prior

The dreadful prior, we wish we didn't have to deal with it. But having to specify priors is a good excuse to learn about different probability distributions. See Fig. 1, which shows the general form of many continuous probability distributions.

We can start with the Normal distribution. A variable x is normally distributed if its pdf is

$$\text{pdf}(x|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right] \tag{18}$$

where μ and σ are the two parameters of the Normal distribution. We will use the following notation to say that x follows a Normal distribution:

$$x \sim \mathcal{N}(\mu,\sigma)$$

The Normal distribution is defined on the real line, so x can take any value between $-\infty$ and ∞. The parameter μ is the mean of the distribution and σ is the standard deviation. Values of x closer to μ are more probable that those far from μ.

The uniform distribution is also quite common

$$x \sim \mathcal{U}(a,b) \quad \text{if} \quad \text{pdf}(x|a,b) = \frac{1}{a-b} \tag{19}$$

For the uniform distribution, all values between a and b are equally probable. The probability that $x < a$ is 0 and the same for $x > b$. These values of x are *impossible*.

Another distribution which might come in handy is the reciprocal distribution (it is not shown in Fig. 1):

$$x \sim \mathcal{R}(a,b) \quad \text{if} \quad \text{pdf}(x|a,b) = \frac{1}{x\,[\log(b) - \log(a)]} \tag{20}$$

where a and b are the lower and upper bounds of the support (note that log is the natural logarithm, sometimes denoted as ln).

Let us introduce briefly one more distribution, the t-distribution:

$$x \sim t(\nu,\mu,\sigma) \quad \text{if} \quad \text{pdf}(x|\nu,\mu,\sigma) = \frac{\Gamma(\frac{\nu+1}{2})}{\Gamma(\frac{\nu}{2})\sqrt{\pi\nu}\,\sigma}\left[1 + \frac{1}{\nu}\left(\frac{x-\mu}{\sigma}\right)^2\right]^{-\frac{\nu+1}{2}} \tag{21}$$

with the degrees of freedom ν, the mean μ and the scale σ as parameters. In Eq. (21), Γ is the gamma function.

This distribution looks complicated but I mention it here for one reason: note how in Fig. 1, the normal and t distributions look very similar, they are both bell-shaped. The distinguishing feature is that the t-distribution has *heavier tails*, that is, values of x far from the mean are more probable if x is t distributed than if x if normally distributed. How heavy the tails are depends on the parameter ν. For example, imagine that

$$x_1 \sim \mathcal{N}(0,1) \tag{22}$$

$$x_2 \sim t(2,0,1) \tag{23}$$

and you obtain 100 000 random samples from these two distributions. Then you calculate the minimum value of all the samples for x_1 and all the samples for x_2. Will these values be very different?

```
from scipy.stats import norm, t

print min( t(1., loc=0, scale=1).rvs(100000) )
print min( norm(loc=0, scale=1).rvs(100000) )
```

They are completely different! By orders of magnitude!
The t-distribution allows for more extreme values. This will come in handy when we deal with outliers.

Other distributions have different pdfs and different parameters. Wikipedia usually has a lot of information about the most common ones, just search for the names you see in Fig. 1.

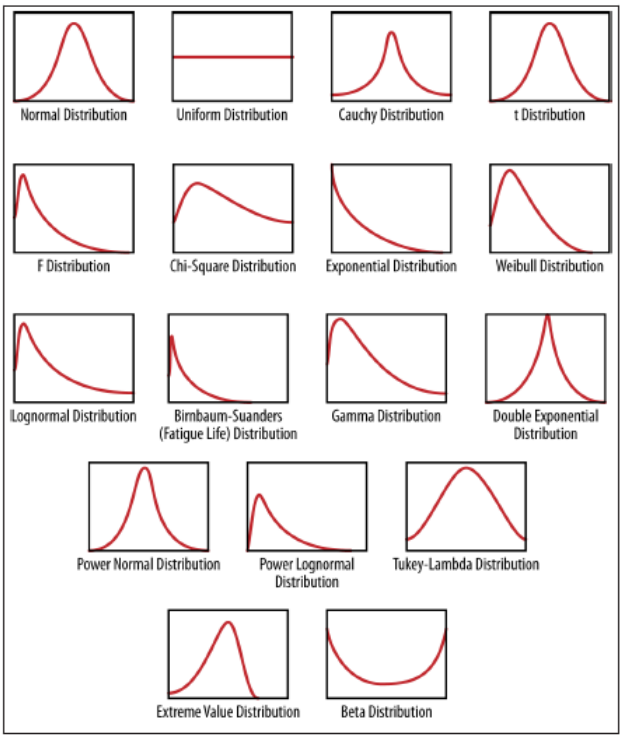

Figure 1: Continuous probability distributions. Adapted from here.

The key concept about priors is to use (some of) these probability distributions to describe our beliefs about the parameters in our models.

- if any value of the parameter seems equally likely but you have an idea of its order of magnitude, try a uniform distribution in a sensible range.
- if one value is somewhat more likely and, again, you know the order of magnitude, try a normal distribution centred in that value.
- if any value is equally likely within a few orders of magnitude, use a log-normal or an exponential distribution. [check!!!]
- if the parameter takes values between 0 and 1, the beta distribution is defined in this interval and can take a wide range of forms, depending on its parameters.

Every time you come up with a prior, you are using your expert knowledge about the problem. Don't underestimate this knowledge by trying to set an "uninformative" prior at all costs. If you are not sure, try a few different priors and see if the results make sense and are consistent. If you

reach different conclusions by changing the priors, mention this when you write your paper.

The only way in which you can go completely wrong and change the results dramatically is by setting the prior to 0. No amount of data can change a prior which is zero.

think about it

If the prior distribution is 0, the posterior distribution is 0, independent of the likelihood. This reflects a very strong assumption.

2.1.1 An example

Let's try to write a `Python` code to calculate the prior in a simple example.

First, we state some of the information that is included in $\mathcal{I}$. The radial-velocity signal induced in a star by an orbiting planet depends

- on the orbital period, P
- on the semi-amplitude, K
- on the eccentricity, e
- on the argument of periastron, ω
- on the phase of periastron, χ

and we know both the units and the appropriate ranges for most of these parameters. For some parameters, like the orbital period and semi-amplitude, we may not be completely sure about the allowed ranges, but we can still make some informed guesses based on our knowledge of physics and astronomy.

What about the mathematical forms of the priors. Well, we can start by dividing the parameters into two classes: location and scale parameters. The semi-amplitude is clearly a scale parameter as it does not matter if we measure it in $m\,s^{-1}$, $km\,s^{-1}$, or any other velocity unit. If we want our prior to be *scale-invariant*, then an appropriate distribution to use is the reciprocal distribution of Eq. (20). The same can be said for the orbital period, which can be measured in days, seconds, months, etc.

The argument of periastron and phase of periastron are angles that place the planet's orbit in the three-dimensional space. Angles can be measured with respect to an arbitrary direction, so these are location parameters. We don't care if our parameter is defined as ω or as $\omega' = \omega + \pi$. An appropriate prior here is the uniform prior.

It's harder to justify that the eccentricity is also a location parameter. But in any case we know that, for an elliptical orbit, it should take values

between 0 and 1. If we don't know much more (though sometimes we do, see [6]) we can go with a uniform prior for e as well.

So, in summary:

orbital period	$p(P\|\mathcal{I}) \sim \mathcal{J}(1, 1000)$ days
semi-amplitude	$p(K\|\mathcal{I}) \sim \mathcal{J}(0.1, 100)$ ms^{-1}
eccentricity	$p(e\|\mathcal{I}) \sim \mathcal{U}(0, 1)$
argument of periastron	$p(\omega\|\mathcal{I}) \sim \mathcal{U}(0, 2\pi)$
phase of periastron	$p(\chi\|\mathcal{I}) \sim \mathcal{U}(0, 2\pi)$

and we can further assume that these parameters are independent, so their joint prior distribution is the product of the individual priors

$$p(P, K, e, \omega, \chi|\mathcal{I}) = p(P|\mathcal{I})\, p(K|\mathcal{I})\, p(e|\mathcal{I})\, p(\omega|\mathcal{I})\, p(\chi|\mathcal{I}) \tag{24}$$

Let's write this in `Python`, using the distributions from `Scipy`. We want a function `prior(P,K,e,w,X)` that returns the value of the prior pdf, given values of the parameters.

```
from scipy.stats import *
from numpy import product

def prior(P,K,e,w,X):
  prior_P = reciprocal(a=1,b=1000).pdf(P)
  prior_K = reciprocal(a=0.1,b=100).pdf(K)
  prior_e = uniform().pdf(e)
  prior_w = uniform(scale=2*pi).pdf(w)
  prior_X = uniform(scale=2*pi).pdf(X)

  return product([prior_P, prior_K, prior_e, prior_w, prior_X])
```

This will work in most cases, but we are in danger of underflowing: multiplying small numbers can lead to *very* small numbers. So it might be better to work with logarithms:

```
from scipy.stats import *
from numpy import sum

def log_prior(P,K,e,w,X):
  log_prior_P = reciprocal(a=1,b=1000).logpdf(P)
  log_prior_K = reciprocal(a=0.1,b=100).logpdf(K)
```

```
    log_prior_e = uniform().logpdf(e)
    log_prior_w = uniform(scale=2*pi).logpdf(w)
    log_prior_X = uniform(scale=2*pi).logpdf(X)

    return sum([prior_P, prior_K, prior_e, prior_w, prior_X])
```

Note: the functions defined above are probably not the most efficient way to calculate the prior. But they are certainly very readable! If "future-you" will understand the code "present-you" wrote, that's a good thing!

2.2 The likelihood

The likelihood is a prior. I will repeat so you know this is not a typo: the likelihood is a prior. The term $p(D|\theta, I)$ represents your beliefs on what the data will be like, given parameters θ and information I. Therefore, the likelihood encodes prior information. Following [7], the likelihood is *not*

- the process that generated the data
- the pdf that your data kind of looks like when you plot it in a histogram

It is, instead, what we usually call *the model*. And the model is nothing else than a set of assumptions about the data and how they relate to the parameters. **The likelihood provides a (*the*) connection from θ to D.**

But now comes the part which, conceptually, is harder to understand. When we write $p(D|\theta, I)$, this is a distribution for the data, given some values for the parameters θ. A "point" taken randomly from $p(D|\theta, I)$, corresponds to a random dataset. But the likelihood is sometimes seen instead as a function of the parameters, given the observed dataset. In this form, the symbol $\mathcal{L}(\theta)$ is often used.

- $p(D|\theta, I)$ - distribution for datasets, given value of θ
- $\mathcal{L}(\theta)$ - function of θ

This used to confuse me. If it confuses you too, maybe an example will help. Consider Fig. 2, which tries to illustrate the likelihood.

We are considering a model of the form $f(x) = A_1 + A_2\,x + A_3\,x^2$ and a Gaussian likelihood. The model prediction for a specific value of the parameters ($A_1 = 0.5; A_2 = 0.8; A_3 = -0.06$) is shown in blue. The data we have actually observed are 5 pairs of values (x_i, y_i), marked by the red crosses. On the z-axis we depict the Gaussian likelihood distributions. These are Gaussian functions centered at each $f(x_i)$. We assumed that the width of these Gaussians is the same for all, and equal to σ.

You can see that, for the assumed choice of model parameters, the probability of any d_i is proportional to the height of the Gaussian curve directly above that data point, as shown by the solid lines. The probability of the complete dataset, $\{d_i\}_{i=1}^5$, is the product of the individual probabilities.

If our data comes with error-bars, that is we have (x_i, y_i, σ_i), and we want to incorporate that in the likelihood, then things are more like in Fig. 3. Now each Gaussian has a different width and the data is depicted with an errorbar on the y direction.

Hopefully these figures shed some light on the "distribution for data" interpretation of the likelihood. But why is it a function of the parameters?

Consider now Fig. 4, which shows what happens when the parameters A_1, A_2, A_3 change. The left panel is as before. In the right panel, note how the probabilities of each data point are now smaller. So their product will also be smaller, and therefore the likelihood of the full dataset is smaller. **The first set of parameters has a higher likelihood than the second set of parameters**.

2.2.1 An example

We now want to write a `Python` code to calculate the likelihood in the same exoplanet example. As before, we start by stating some of the information that is included in $\mathcal{I}$. A typical radial-velocity dataset contains N observations (t_i, v_i, σ_i) corresponding to the radial-velocity v_i and its associated uncertainty σ_i observed at time t_i.

We also know that the radial-velocity signal as a function of time t, produced by an orbiting planet is given by a Keplerian function

$$\mathrm{kep}(t, P, K, e, \omega, \chi)$$

This is our *model*. We will assume that the likelihood is Gaussian

$$p(v_i|t_i, \sigma_i, P, K, e, \omega, \chi, \mathcal{I}) \sim \mathcal{N}(\mathrm{kep}(t_i, P, K, e, \omega, \chi), \sigma_i) \tag{25}$$

and that the observations are independent.

Let's write this in `Python`, using the distributions from `Scipy`. We want a function `log_likelihood(t,v,sigma,P,K,e,w,X)` that returns the value of the logarithm of the likelihood, given the data and values of the parameters.

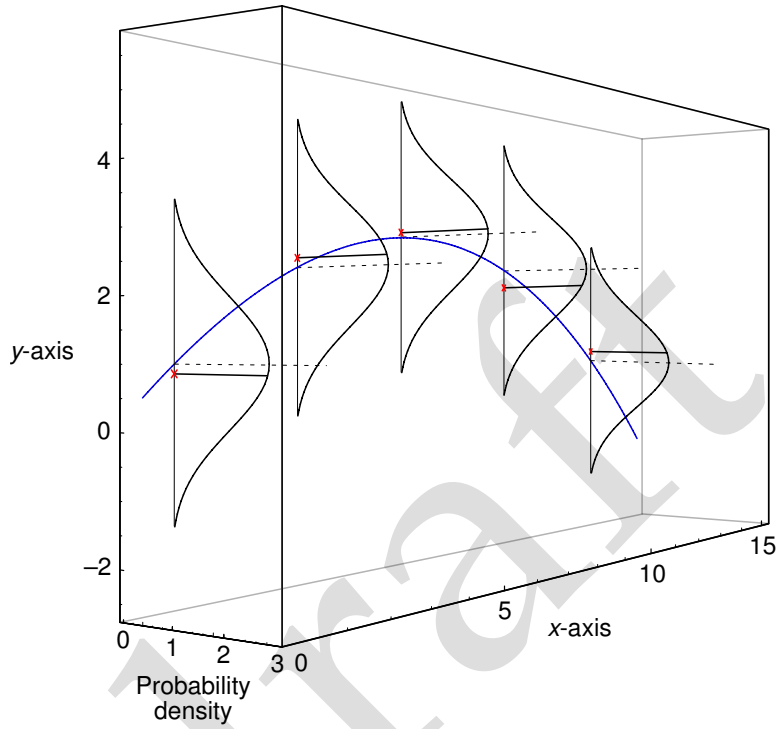

Figure 2: Illustration of the calculation of the likelihood function for a model of the form $f(x) = A_1 + A_2\,x + A_3\,x^2$. The smooth blue curve is the model prediction for a specific choice of the parameters. The predicted values of $f(x_i)$ for each choice of the independent variable x_i are marked by a dashed line. The actual measured value of d_i (represented by a red cross) is located at the same value of x_i but above or below $f(x_i)$ as a result of the uncertainty σ. At the location of each $f(x_i)$ value we have constructed a Gaussian probability density function, with probability plotted in the z-coordinate. For the assumed choice of model parameters, the probability of any d_i is proportional to the height of the Gaussian curve directly above that data point, which is shown by the solid lines. Adapted from [8].

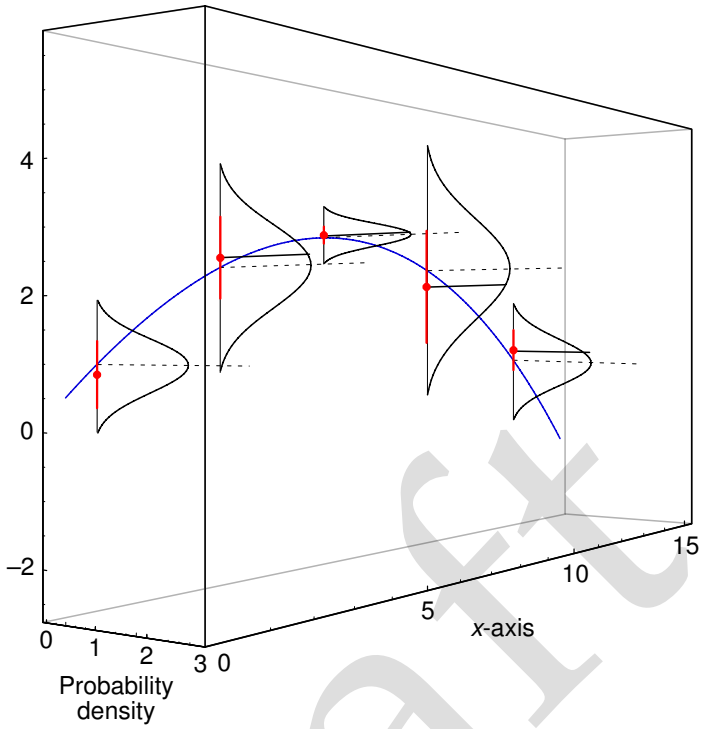

Figure 3: Same as Fig. 2 but for data with error-bars.

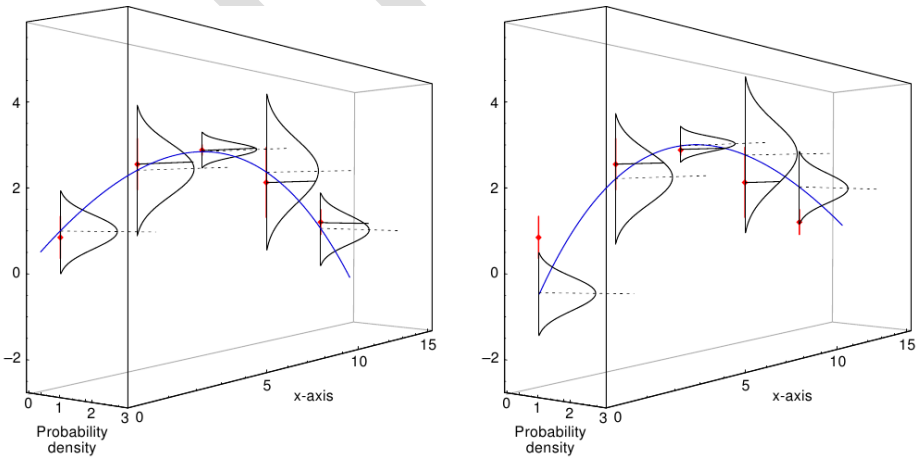

Figure 4: Same as Fig. 3, but with two different values for the parameters A_1, A_2, A_3. The data is the same as before but the model prediction and therefore the means of the Gaussian distributions, have changed.

```python
from scipy.stats import norm
from numpy import sum

def likelihood(t,v,sigma,P,K,e,w,X):
  # t, v and sigma are arrays
  # P, K, e, w, X are floats

  # the function kep calculates the Keplerian signal
  model_rv = kep(t, P, K, e, w, X)

  loglike = sum([norm(loc=rv, scale=s).logpdf(obs_v)
                 for rv, s, obs_v in zip(model_rv, sigma, v)])

  return loglike
```

3 MCMC

Markov chain Monte Carlo (MCMC) is the workhorse of Bayesian statistics. But how does it work? And what does it actually do? We will write the code for our own MCMC in the practical classes. If that is enough for you to *understand* it, skip the next section. But if you want to know *why* it works, carry on reading.

3.1 A basic MCMC implementation

We will only talk about the Metropolis-Hastings algorithm to do MCMC. It is probably not the simplest, but it is definitely the most often used.
The following is adapted slightly from Wikipedia:

> Let $f(x)$ be a function that is proportional to the desired probability distribution $P(x)$ (a.k.a. the target distribution).
>
> **Initialization:**
> Choose an arbitrary point x_0 to be the first sample, and choose an arbitrary probability density $Q(x|y)$ which suggests a candidate for the next sample value x, given the previous sample value y. A usual choice is to let $Q(x|y)$ be a Gaussian distribution centered at y. The function Q is referred to as the proposal density or jumping distribution.
>
> **For each iteration t:**
>
> - Generate a candidate x' for the next sample by picking from the distribution $Q(x'|x_t)$.
> - Calculate the acceptance ratio $\alpha = f(x')/f(x_t)$. Because f is proportional to P, we have that $\alpha = f(x')/f(x_t) = P(x')/P(x_t)$.
> - If $\alpha \geq 1$, then the candidate is more likely than x_t: automatically accept the candidate by setting $x_{t+1} = x'$.
>   Otherwise, accept the candidate with probability α;
>   if the candidate is rejected, set $x_{t+1} = x_t$, instead.

Let's see this in `Python` code

```
1 import numpy as np
2 from scipy.stats import norm
3
4 niter = 500 # number of iterations
5
```

```
# posterior distribution (the distribution we want to sample from)
p = lambda x: norm.pdf(x)

x = [0.5] # starting point

for step in range(niter):
    d = np.random.randn() # propose a step

    # calculate the ratio of posterior probabilities
    alpha = p(x[-1]+d) / p(x[-1])

    if alpha > 1.:
        x.append(x[-1]+d)      # accept the new step
    else:
        u = np.random.uniform()
        if ratio > u:
            x.append(x[-1]+d)  # accept the new step
        else:
            x.append(x[-1])    # reject the new step (stay where we are)
```

References

1. Andrew Gelman. *Bayesian data analysis*. Chapman & Hall/CRC Texts in Statistical Science. CRC Press, 3rd edition, 2014.

2. Thomas J. Loredo. The Return of the Prodigal: Bayesian Inference in Astrophysics. Valencia, 1994.

3. E. T. Jaynes and G. Larry Bretthorst. *Probability theory: the logic of science*. Cambridge University Press, Cambridge, UK ; New York, NY, 2003.

4. David W. Hogg. Data analysis recipes: Probability calculus for inference. *ArXiv e-prints*, 1205:4446, May 2012.

5. Rene Andrae, Tim Schulze-Hartung, and Peter Melchior. Dos and don'ts of reduced chi-squared. *ArXiv10123754 Astro-Ph Physicsphysics Stat*, December 2010.

6. David M. Kipping. Parametrizing the exoplanet eccentricity distribution with the Beta distribution. *Mon. Not. R. Astron. Soc. Lett.*, 434(1):L51–L55, 2013.

7. B. J. Brewer. The prior isn't the only prior, 2013.

8. P. C. Gregory. *Bayesian logical data analysis for the physical sciences*. Cambridge University Press, 2010.