

Frequentism and Bayesianism: A Python-driven Primer

Jake VanderPlas*†



Abstract—This paper presents a brief, semi-technical comparison of the essential features of the frequentist and Bayesian approaches to statistical inference, with several illustrative examples implemented in Python. The differences between frequentism and Bayesianism fundamentally stem from differing definitions of probability, a philosophical divide which leads to distinct approaches to the solution of statistical problems as well as contrasting ways of asking and answering questions about unknown parameters. After an example-driven discussion of these differences, we briefly compare several leading Python statistical packages which implement frequentist inference using classical methods and Bayesian inference using Markov Chain Monte Carlo.¹

Index Terms—statistics, frequentism, Bayesian inference

Introduction

One of the first things a scientist in a data-intensive field hears about statistics is that there are two different approaches: frequentism and Bayesianism. Despite their importance, many researchers never have opportunity to learn the distinctions between them and the different practical approaches that result.

This paper seeks to synthesize the philosophical and pragmatic aspects of this debate, so that scientists who use these approaches might be better prepared to understand the tools available to them. Along the way we will explore the fundamental philosophical disagreement between frequentism and Bayesianism, explore the practical aspects of how this disagreement affects data analysis, and discuss the ways that these practices may affect the interpretation of scientific results.

This paper is written for scientists who have picked up some statistical knowledge along the way, but who may not fully appreciate the philosophical differences between frequentist and Bayesian approaches and the effect these differences have on both the computation and interpretation of statistical results. Because this passing statistics knowledge generally leans toward frequentist principles, this paper will go into more depth on the details of Bayesian rather than frequentist approaches. Still, it is not meant to be a full introduction to either class of methods. In particular, concepts such as the likelihood are assumed rather than derived, and many

advanced Bayesian and frequentist diagnostic tests are left out in favor of illustrating the most fundamental aspects of the approaches. For a more complete treatment, see, e.g. [Wasserman2004] or [Gelman2004].

The Disagreement: The Definition of Probability

Fundamentally, the disagreement between frequentists and Bayesians concerns the definition of probability.

For frequentists, probability only has meaning in terms of **a limiting case of repeated measurements**. That is, if an astronomer measures the photon flux F from a given non-variable star, then measures it again, then again, and so on, each time the result will be slightly different due to the statistical error of the measuring device. In the limit of many measurements, the *frequency* of any given value indicates the probability of measuring that value. For frequentists, **probabilities are fundamentally related to frequencies of events**. This means, for example, that in a strict frequentist view, it is meaningless to talk about the probability of the *true* flux of the star: the true flux is, by definition, a single fixed value, and to talk about an extended frequency distribution for a fixed value is nonsense.

For Bayesians, the concept of probability is extended to cover **degrees of certainty about statements**. A Bayesian might claim to know the flux F of a star with some probability $P(F)$: that probability can certainly be estimated from frequencies in the limit of a large number of repeated experiments, but this is not fundamental. The probability is a statement of the researcher's knowledge of what the true flux is. For Bayesians, **probabilities are fundamentally related to their own knowledge about an event**. This means, for example, that in a Bayesian view, we can meaningfully talk about the probability that the *true* flux of a star lies in a given range. That probability codifies our knowledge of the value based on prior information and available data.

The surprising thing is that this arguably subtle difference in philosophy can lead, in practice, to vastly different approaches to the statistical analysis of data. Below we will explore a few examples chosen to illustrate the differences in approach, along with associated Python code to demonstrate the practical aspects of the frequentist and Bayesian approaches.

A Simple Example: Photon Flux Measurements

First we will compare the frequentist and Bayesian approaches to the solution of an extremely simple problem. Imagine that

* Corresponding author: jakevdp@cs.washington.edu

† eScience Institute, University of Washington

Copyright © 2014 Jake VanderPlas. This is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

1. This paper draws heavily from content originally published in a series of posts on the author's blog, [Pythonic Perambulations](#) [VanderPlas2014].

we point a telescope to the sky, and observe the light coming from a single star. For simplicity, we will assume that the star's true photon flux is constant with time, i.e. that it has a fixed value F ; we will also ignore effects like sky background systematic errors. We will assume that a series of N measurements are performed, where the i^{th} measurement reports the observed flux F_i and error e_i .² The question is, given this set of measurements $D = \{F_i, e_i\}$, what is our best estimate of the true flux F ?

First we will use Python to generate some toy data to demonstrate the two approaches to the problem. We will draw 50 samples F_i with a mean of 1000 (in arbitrary units) and a (known) error e_i :

```
>>> np.random.seed(2) # for reproducibility
>>> e = np.random.normal(30, 3, 50)
>>> F = np.random.normal(1000, e)
```

In this toy example we already know the true flux F , but the question is this: given our measurements and errors, what is our best point estimate of the true flux? Let's look at a frequentist and a Bayesian approach to solving this.

Frequentist Approach to Flux Measurement

We will start with the classical frequentist maximum likelihood approach. Given a single observation $D_i = (F_i, e_i)$, we can compute the probability distribution of the measurement given the true flux F given our assumption of Gaussian errors:

$$P(D_i|F) = (2\pi e_i^2)^{-1/2} \exp\left(\frac{-(F_i - F)^2}{2e_i^2}\right).$$

This should be read "the probability of D_i given F equals ...". You should recognize this as a normal distribution with mean F and standard deviation e_i . We construct the *likelihood* by computing the product of the probabilities for each data point:

$$\mathcal{L}(D|F) = \prod_{i=1}^N P(D_i|F)$$

Here $D = \{D_i\}$ represents the entire set of measurements. For reasons of both analytic simplicity and numerical accuracy, it is often more convenient to instead consider the log-likelihood; combining the previous two equations gives

$$\log \mathcal{L}(D|F) = -\frac{1}{2} \sum_{i=1}^N \left[\log(2\pi e_i^2) + \frac{(F_i - F)^2}{e_i^2} \right].$$

We would like to determine the value of F which maximizes the likelihood. For this simple problem, the maximization can be computed analytically (e.g. by setting $d \log \mathcal{L} / dF|_{\hat{F}} = 0$), which results in the following point estimate of F :

$$\hat{F} = \frac{\sum w_i F_i}{\sum w_i}; \quad w_i = 1/e_i^2$$

The result is a simple weighted mean of the observed values. Notice that in the case of equal errors e_i , the weights cancel and \hat{F} is simply the mean of the observed data.

² We will make the reasonable assumption of normally-distributed measurement errors. In a Frequentist perspective, e_i is the standard deviation of the results of the single measurement event in the limit of (imaginary) repetitions of *that event*. In the Bayesian perspective, e_i describes the probability distribution which quantifies our knowledge of F given the measured value F_i .

We can go further and ask what the uncertainty of our estimate is. One way this can be accomplished in the frequentist approach is to construct a Gaussian approximation to the peak likelihood; in this simple case the fit can be solved analytically to give:

$$\sigma_{\hat{F}} = \left(\sum_{i=1}^N w_i \right)^{-1/2}$$

This result can be evaluated this in Python as follows:

```
>>> w = 1. / e ** 2
>>> F_hat = np.sum(w * F) / np.sum(w)
>>> sigma_F = w.sum() ** -0.5
```

For our particular data, the result is $\hat{F} = 999 \pm 4$.

Bayesian Approach to Flux Measurement

The Bayesian approach, as you might expect, begins and ends with probabilities. The fundamental result of interest is our knowledge of the parameters in question, codified by the probability $P(F|D)$. To compute this result, we next apply Bayes' theorem, a fundamental law of probability:

$$P(F|D) = \frac{P(D|F) P(F)}{P(D)}$$

Though Bayes' theorem is where Bayesians get their name, it is important to note that it is not this theorem itself that is controversial, but the Bayesian *interpretation of probability* implied by the term $P(F|D)$. While the above formulation makes sense given the Bayesian view of probability, the setup is fundamentally contrary to the frequentist philosophy, which says that probabilities have no meaning for fixed model parameters like F . In the Bayesian conception of probability, however, this poses no problem.

Let's take a look at each of the terms in this expression:

- $P(F|D)$: The **posterior**, which is the probability of the model parameters given the data.
- $P(D|F)$: The **likelihood**, which is proportional to the $\mathcal{L}(D|F)$ used in the frequentist approach.
- $P(F)$: The **model prior**, which encodes what we knew about the model before considering the data D .
- $P(D)$: The **model evidence**, which in practice amounts to simply a normalization term.

If we set the prior $P(F) \propto 1$ (a *flat prior*), we find

$$P(F|D) \propto \mathcal{L}(D|F).$$

That is, with a flat prior on F , the Bayesian posterior is maximized at precisely the same value as the frequentist result! So despite the philosophical differences, we see that the Bayesian and frequentist point estimates are equivalent for this simple problem.

You might notice that we glossed over one important piece here: the prior, $P(F)$, which we assumed to be flat.³ The prior allows inclusion of other information into the computation, which becomes very useful in cases where multiple measurement strategies are being combined to constrain a single model (as is the case in, e.g. cosmological parameter estimation).

The necessity to specify a prior, however, is one of the more controversial pieces of Bayesian analysis.

A frequentist will point out that the prior is problematic when no true prior information is available. Though it might seem straightforward to use an **uninformative prior** like the flat prior mentioned above, there are some surprising subtleties involved.⁴ It turns out that in many situations, a truly uninformative prior cannot exist! Frequentists point out that the subjective choice of a prior which necessarily biases the result should have no place in scientific data analysis.

A Bayesian would counter that frequentism doesn't solve this problem, but simply skirts the question. Frequentism can often be viewed as simply a special case of the Bayesian approach for some (implicit) choice of the prior: a Bayesian would say that it's better to make this implicit choice explicit, even if the choice might include some subjectivity. Furthermore, as we will see below, the question frequentism answers is not always the question the researcher wants to ask.

Where The Results Diverge

In the simple example above, the frequentist and Bayesian approaches give basically the same result. In light of this, arguments over the use of a prior and the philosophy of probability may seem frivolous. However, while it is easy to show that the two approaches are often equivalent for simple problems, it is also true that they can diverge greatly in other situations. In practice, this divergence most often makes itself most clear in two different ways:

1. The handling of nuisance parameters: i.e. parameters which affect the final result, but are not otherwise of interest.
2. The different handling of uncertainty: for example, the subtle (and often overlooked) difference between frequentist confidence intervals and Bayesian credible regions.

We will discuss examples of these below.

Nuisance Parameters: Bayes' Billiards Game

We will start by discussing the first point: nuisance parameters. A nuisance parameter is any quantity whose value is not directly relevant to the goal of an analysis, but is nevertheless required to determine the result which is of interest. For example, we might have a situation similar to the flux measurement above, but in which the errors e_i are unknown. One potential approach is to treat these errors as nuisance parameters.

Here we consider an example of nuisance parameters borrowed from [Eddy2004] that, in one form or another, dates all the way back to the posthumously-published 1763 paper written by Thomas Bayes himself [Bayes1763]. The setting is

3. A flat prior is an example of an improper prior: that is, it cannot be normalized. In practice, we can remedy this by imposing some bounds on possible values: say, $0 < F < F_{tot}$, the total flux of all sources in the sky. As this normalization term also appears in the denominator of Bayes' theorem, it does not affect the posterior.

4. The flat prior in this case can be motivated by maximum entropy; see, e.g. [Jeffreys1946]. Still, the use of uninformative priors like this often raises eyebrows among frequentists: there are good arguments that even "uninformative" priors can add information; see e.g. [Evans2002].

a gambling game in which Alice and Bob bet on the outcome of a process they can't directly observe.

Alice and Bob enter a room. Behind a curtain there is a billiard table, which they cannot see. Their friend Carol rolls a ball down the table, and marks where it lands. Once this mark is in place, Carol begins rolling new balls down the table. If the ball lands to the left of the mark, Alice gets a point; if it lands to the right of the mark, Bob gets a point. We can assume that Carol's rolls are unbiased: that is, the balls have an equal chance of ending up anywhere on the table. The first person to reach six points wins the game.

Here the location of the mark (determined by the first roll) can be considered a nuisance parameter: it is unknown and not of immediate interest, but it clearly must be accounted for when predicting the outcome of subsequent rolls. If this first roll settles far to the right, then subsequent rolls will favor Alice. If it settles far to the left, Bob will be favored instead.

Given this setup, we seek to answer this question: *In a particular game, after eight rolls, Alice has five points and Bob has three points. What is the probability that Bob will get six points and win the game?*

Intuitively, we realize that because Alice received five of the eight points, the marker placement likely favors her. Given that she has three opportunities to get a sixth point before Bob can win, she seems to have clinched it. But quantitatively speaking, what is the probability that Bob will persist to win?

A Naïve Frequentist Approach

Someone following a classical frequentist approach might reason as follows:

To determine the result, we need to estimate the location of the marker. We will quantify this marker placement as a probability p that any given roll lands in Alice's favor. Because five balls out of eight fell on Alice's side of the marker, we compute the maximum likelihood estimate of p , given by:

$$\hat{p} = 5/8,$$

a result follows in a straightforward manner from the binomial likelihood. Assuming this maximum likelihood probability, we can compute the probability that Bob will win, which requires him to get a point in each of the next three rolls. This is given by:

$$P(B) = (1 - \hat{p})^3$$

Thus, we find that the probability of Bob winning is 0.053, or odds against Bob winning of 18 to 1.

A Bayesian Approach

A Bayesian approach to this problem involves *marginalizing* (i.e. integrating) over the unknown p so that, assuming the prior is accurate, our result is agnostic to its actual value. In this vein, we will consider the following quantities:

- B = Bob Wins
- D = observed data, i.e. $D = (n_A, n_B) = (5, 3)$
- p = unknown probability that a ball lands on Alice's side during the current game

We want to compute $P(B|D)$; that is, the probability that Bob wins given the observation that Alice currently has five

points to Bob's three. A Bayesian would recognize that this expression is a *marginal probability* which can be computed by integrating over the joint distribution $P(B, p|D)$:

$$P(B|D) \equiv \int_{-\infty}^{\infty} P(B, p|D) dp$$

This identity follows from the definition of conditional probability, and the law of total probability: that is, it is a fundamental consequence of probability axioms and will always be true. Even a frequentist would recognize this; they would simply disagree with the interpretation of $P(p)$ as being a measure of uncertainty of knowledge of the parameter p .

To compute this result, we will manipulate the above expression for $P(B|D)$ until we can express it in terms of other quantities that we can compute.

We start by applying the definition of conditional probability to expand the term $P(B, p|D)$:

$$P(B|D) = \int P(B|p, D)P(p|D)dp$$

Next we use Bayes' rule to rewrite $P(p|D)$:

$$P(B|D) = \int P(B|p, D) \frac{P(D|p)P(p)}{P(D)} dp$$

Finally, using the same probability identity we started with, we can expand $P(D)$ in the denominator to find:

$$P(B|D) = \frac{\int P(B|p, D)P(D|p)P(p)dp}{\int P(D|p)P(p)dp}$$

Now the desired probability is expressed in terms of three quantities that we can compute:

- $P(B|p, D)$: This term is proportional to the frequentist likelihood we used above. In words: given a marker placement p and Alice's 5 wins to Bob's 3, what is the probability that Bob will go on to six wins? Bob needs three wins in a row, i.e. $P(B|p, D) = (1 - p)^3$.
- $P(D|p)$: this is another easy-to-compute term. In words: given a probability p , what is the likelihood of exactly 5 positive outcomes out of eight trials? The answer comes from the Binomial distribution: $P(D|p) \propto p^5(1 - p)^3$
- $P(p)$: this is our prior on the probability p . By the problem definition, we can assume that p is evenly drawn between 0 and 1. That is, $P(p) \propto 1$ for $0 \leq p \leq 1$.

Putting this all together and simplifying gives

$$P(B|D) = \frac{\int_0^1 (1-p)^6 p^5 dp}{\int_0^1 (1-p)^3 p^5 dp}$$

These integrals are instances of the beta function, so we can quickly evaluate the result using `scipy`:

```
>>> from scipy.special import beta
>>> P_B_D = beta(6+1, 5+1) / beta(3+1, 5+1)
```

This gives $P(B|D) = 0.091$, or odds of 10 to 1 against Bob winning.

Discussion

The Bayesian approach gives odds of 10 to 1 against Bob, while the naïve frequentist approach gives odds of 18 to 1 against Bob. So which one is correct?

For a simple problem like this, we can answer this question empirically by simulating a large number of games and count the fraction of suitable games which Bob goes on to win. This can be coded in a couple dozen lines of Python (see part II of [VanderPlas2014]). The result of such a simulation confirms the Bayesian result: 10 to 1 against Bob winning.

So what is the takeaway: is frequentism wrong? Not necessarily: in this case, the incorrect result is more a matter of the approach being "naïve" than it being "frequentist". The approach above does not consider how p may vary. There exist frequentist methods that can address this by, e.g. applying a transformation and conditioning of the data to isolate dependence on p , or by performing a Bayesian-like integral over the sampling distribution of the frequentist estimator \hat{p} .

Another potential frequentist response is that the question itself is posed in a way that does not lend itself to the classical, frequentist approach. A frequentist might instead hope to give the answer in terms of null tests or confidence intervals: that is, they might devise a procedure to construct limits which would provably bound the correct answer in $100 \times (1 - \alpha)$ percent of similar trials, for some value of α – say, 0.05. We will discuss the meaning of such confidence intervals below.

There is one clear common point of these two frequentist responses: both require some degree of effort and/or special expertise in classical methods; perhaps a suitable frequentist approach would be immediately obvious to an expert statistician, but is not particularly obvious to a statistical layperson. In this sense, it could be argued that for a problem such as this (i.e. with a well-motivated prior), Bayesianism provides a more natural framework for handling nuisance parameters: by simple algebraic manipulation of a few well-known axioms of probability interpreted in a Bayesian sense, we straightforwardly arrive at the correct answer without need for other special statistical expertise.

Confidence vs. Credibility: Jaynes' Truncated Exponential

A second major consequence of the philosophical difference between frequentism and Bayesianism is in the handling of uncertainty, exemplified by the standard tools of each method: frequentist confidence intervals (CIs) and Bayesian credible regions (CRs). Despite their apparent similarity, the two approaches are fundamentally different. Both are statements of probability, but the probability refers to different aspects of the computed bounds. For example, when constructing a standard 95% bound about a parameter θ :

- A Bayesian would say: "Given our observed data, there is a 95% probability that the true value of θ lies within the credible region".
- A frequentist would say: "If this experiment is repeated many times, in 95% of these cases the computed confidence interval will contain the true θ ."⁵

Notice the subtle difference: the Bayesian makes a statement of probability about the *parameter value* given a *fixed credible region*. The frequentist makes a statement of probability about the *confidence interval itself* given a *fixed parameter value*. This distinction follows straightforwardly from the definition of probability discussed above: the Bayesian probability is a statement of degree of knowledge about a parameter; the frequentist probability is a statement of long-term limiting frequency of quantities (such as the CI) derived from the data.

This difference must necessarily affect our interpretation of results. For example, it is common in scientific literature to see it claimed that it is 95% certain that an unknown parameter lies within a given 95% CI, but this is not the case! This is erroneously applying the Bayesian interpretation to a frequentist construction. This frequentist oversight can perhaps be forgiven, as under most circumstances (such as the simple flux measurement example above), the Bayesian CR and frequentist CI will more-or-less overlap. But, as we will see below, this overlap cannot always be assumed, especially in the case of non-Gaussian distributions constrained by few data points. As a result, this common misinterpretation of the frequentist CI can lead to dangerously erroneous conclusions.

To demonstrate a situation in which the frequentist confidence interval and the Bayesian credibility region do not overlap, let us turn to an example given by E.T. Jaynes, a 20th century physicist who wrote extensively on statistical inference. In his words, consider a device that

“...will operate without failure for a time θ because of a protective chemical inhibitor injected into it; but at time θ the supply of the chemical is exhausted, and failures then commence, following the exponential failure law. It is not feasible to observe the depletion of this inhibitor directly; one can observe only the resulting failures. From data on actual failure times, estimate the time θ of guaranteed safe operation...” [Jaynes1976]

Essentially, we have data D drawn from the model:

$$P(x|\theta) = \begin{cases} \exp(\theta - x) & , \quad x > \theta \\ 0 & , \quad x < \theta \end{cases}$$

where $p(x|\theta)$ gives the probability of failure at time x , given an inhibitor which lasts for a time θ . We observe some failure times, say $D = \{10, 12, 15\}$, and ask for 95% uncertainty bounds on the value of θ .

First, let's think about what common-sense would tell us. Given the model, an event can only happen after a time θ . Turning this around tells us that the upper-bound for θ must be $\min(D)$. So, for our particular data, we would immediately write $\theta \leq 10$. With this in mind, let's explore how a frequentist and a Bayesian approach compare to this observation.

Truncated Exponential: A Frequentist Approach

In the frequentist paradigm, we'd like to compute a confidence interval on the value of θ . We might start by observing that

5. [Wasserman2004], however, notes on p. 92 that we need not consider repetitions of the same experiment; it's sufficient to consider repetitions of any correctly-performed frequentist procedure.

the population mean is given by

$$E(x) = \int_0^\infty xp(x)dx = \theta + 1.$$

So, using the sample mean as the point estimate of $E(x)$, we have an unbiased estimator for θ given by

$$\hat{\theta} = \frac{1}{N} \sum_{i=1}^N x_i - 1.$$

In the large- N limit, the central limit theorem tells us that the sampling distribution is normal with standard deviation given by the standard error of the mean: $\sigma_{\hat{\theta}}^2 = 1/N$, and we can write the 95% (i.e. 2σ) confidence interval as

$$CI_{\text{large } N} = \left(\hat{\theta} - 2N^{-1/2}, \hat{\theta} + 2N^{-1/2} \right)$$

For our particular observed data, this gives a confidence interval around our unbiased estimator of $CI(\theta) = (10.2, 12.5)$, entirely above our common-sense bound of $\theta < 10$! We might hope that this discrepancy is due to our use of the large- N approximation with a paltry $N = 3$ samples. A more careful treatment of the problem (See [Jaynes1976] or part III of [VanderPlas2014]) gives the exact confidence interval (10.2, 12.2): the 95% confidence interval entirely excludes the sensible bound $\theta < 10$!

Truncated Exponential: A Bayesian Approach

A Bayesian approach to the problem starts with Bayes' rule:

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)}.$$

We use the likelihood given by

$$P(D|\theta) \propto \prod_{i=1}^N P(x_i|\theta)$$

and, in the absence of other information, use an uninformative flat prior on θ to find

$$P(\theta|D) \propto \begin{cases} N \exp[N(\theta - \min(D))] & , \quad \theta < \min(D) \\ 0 & , \quad \theta > \min(D) \end{cases}$$

where $\min(D)$ is the smallest value in the data D , which enters because of the truncation of $P(x_i|\theta)$. Because $P(\theta|D)$ increases exponentially up to the cutoff, the shortest 95% credibility interval (θ_1, θ_2) will be given by $\theta_2 = \min(D)$, and θ_1 given by the solution to the equation

$$\int_{\theta_1}^{\theta_2} P(\theta|D)d\theta = f$$

which has the solution

$$\theta_1 = \theta_2 + \frac{1}{N} \ln \left[1 - f(1 - e^{-N\theta_2}) \right].$$

For our particular data, the Bayesian credible region is

$$CR(\theta) = (9.0, 10.0)$$

which agrees with our common-sense bound.

Discussion

Why do the frequentist CI and Bayesian CR give such different results? The reason goes back to the definitions of the CI and CR, and to the fact that *the two approaches are answering different questions*. The Bayesian CR answers a question about the value of θ itself (the probability that the parameter is in the fixed CR), while the frequentist CI answers a question about the procedure used to construct the CI (the probability that any potential CI will contain the fixed parameter).

Using Monte Carlo simulations, it is possible to confirm that both the above results correctly answer their respective questions (see [VanderPlas2014], III). In particular, 95% of frequentist CIs constructed using data drawn from this model in fact contain the true θ . Our particular data are simply among the unhappy 5% which the confidence interval misses. But this makes clear the danger of misapplying the Bayesian interpretation to a CI: this particular CI is not 95% likely to contain the true value of θ ; it is in fact 0% likely!

This shows that when using frequentist methods on fixed data, we must carefully keep in mind what question frequentism is answering. Frequentism does not seek a *probabilistic statement about a fixed interval* as the Bayesian approach does; it instead seeks probabilistic statements about an *ensemble of constructed intervals*, with the particular computed interval just a single draw from among them. Despite this, it is common to see a 95% confidence interval interpreted in the Bayesian sense: as a fixed interval that the parameter is expected to be found in 95% of the time. As seen clearly here, this interpretation is flawed, and should be carefully avoided.

Though we used a correct unbiased frequentist estimator above, it should be emphasized that the unbiased estimator is not always optimal for any given problem: especially one with small N and/or censored models; see, e.g. [Hardy2003]. Other frequentist estimators are available: for example, if the (biased) maximum likelihood estimator were used here instead, the confidence interval would be very similar to the Bayesian credible region derived above. Regardless of the choice of frequentist estimator, however, the correct interpretation of the CI is the same: it gives probabilities concerning the *recipe for constructing limits*, not for the *parameter values given the observed data*. For sensible parameter constraints from a single dataset, Bayesianism may be preferred, especially if the difficulties of uninformative priors can be avoided through the use of true prior information.

Bayesianism in Practice: Markov Chain Monte Carlo

Though Bayesianism has some nice features in theory, in practice it can be extremely computationally intensive: while simple problems like those examined above lend themselves to relatively easy analytic integration, real-life Bayesian computations often require numerical integration of high-dimensional parameter spaces.

A turning-point in practical Bayesian computation was the development and application of sampling methods such as Markov Chain Monte Carlo (MCMC). MCMC is a class of algorithms which can efficiently characterize even high-dimensional posterior distributions through drawing of randomized samples such that the points are distributed according

to the posterior. A detailed discussion of MCMC is well beyond the scope of this paper; an excellent introduction can be found in [Gelman2004]. Below, we will propose a straightforward model and compare a standard frequentist approach with three MCMC implementations available in Python.

Application: A Simple Linear Model

As an example of a more realistic data-driven analysis, let's consider a simple three-parameter linear model which fits a straight-line to data with unknown errors. The parameters will be the y -intercept α , the slope β , and the (unknown) normal scatter σ about the line.

For data $D = \{x_i, y_i\}$, the model is

$$\hat{y}(x_i|\alpha, \beta) = \alpha + \beta x_i,$$

and the likelihood is the product of the Gaussian distribution for each point:

$$\mathcal{L}(D|\alpha, \beta, \sigma) = (2\pi\sigma^2)^{-N/2} \prod_{i=1}^N \exp\left[-\frac{[y_i - \hat{y}(x_i|\alpha, \beta)]^2}{2\sigma^2}\right].$$

We will evaluate this model on the following data set:

```
import numpy as np
np.random.seed(42) # for repeatability
theta_true = (25, 0.5)
xdata = 100 * np.random.random(20)
ydata = theta_true[0] + theta_true[1] * xdata
ydata = np.random.normal(ydata, 10) # add error
```

Below we will consider a frequentist solution to this problem computed with the statsmodels package⁶, as well as a Bayesian solution computed with several MCMC implementations in Python: emcee⁷, PyMC⁸, and PyStan⁹. A full discussion of the strengths and weaknesses of the various MCMC algorithms used by the packages is out of scope for this paper, as is a full discussion of performance benchmarks for the packages. Rather, the purpose of this section is to show side-by-side examples of the Python APIs of the packages. First, though, we will consider a frequentist solution.

Frequentist Solution

A frequentist solution can be found by computing the maximum likelihood point estimate. For standard linear problems such as this, the result can be computed using efficient linear algebra. If we define the *parameter vector*, $\theta = [\alpha \ \beta]^T$; the *response vector*, $Y = [y_1 \ y_2 \ y_3 \ \dots \ y_N]^T$; and the *design matrix*,

$$X = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ x_1 & x_2 & x_3 & \dots & x_N \end{bmatrix}^T,$$

it can be shown that the maximum likelihood solution is

$$\hat{\theta} = (X^T X)^{-1} (X^T Y).$$

6. statsmodels: Statistics in Python <http://statsmodels.sourceforge.net/>

7. emcee: The MCMC Hammer <http://dan.iel.fm/emcee/>

8. PyMC: Bayesian Inference in Python <http://pymc-devs.github.io/pymc/>

9. PyStan: The Python Interface to Stan <https://pystan.readthedocs.org/>

The confidence interval around this value is an ellipse in parameter space defined by the following matrix:

$$\Sigma_{\hat{\theta}} \equiv \begin{bmatrix} \sigma_{\alpha}^2 & \sigma_{\alpha\beta} \\ \sigma_{\alpha\beta} & \sigma_{\beta}^2 \end{bmatrix} = \sigma^2(M^T M)^{-1}.$$

Here σ is our unknown error term; it can be estimated based on the variance of the residuals about the fit. The off-diagonal elements of $\Sigma_{\hat{\theta}}$ are the correlated uncertainty between the estimates. In code, the computation looks like this:

```
>>> X = np.vstack([np.ones_like(xdata), xdata]).T
>>> theta_hat = np.linalg.solve(np.dot(X.T, X),
...                             np.dot(X.T, ydata))
>>> y_hat = np.dot(X, theta_hat)
>>> sigma_hat = np.std(ydata - y_hat)
>>> Sigma = sigma_hat ** 2 * \
...         np.linalg.inv(np.dot(X.T, X))
```

The 1σ and 2σ results are shown by the black ellipses in Figure 1.

In practice, the frequentist approach often relies on many more statistical diagnostics beyond the maximum likelihood and confidence interval. These can be computed quickly using convenience routines built-in to the `statsmodels` package [Seabold2010]. For this problem, it can be used as follows:

```
>>> import statsmodels.api as sm # version 0.5
>>> X = sm.add_constant(xdata)
>>> result = sm.OLS(ydata, X).fit()
>>> sigma_hat = result.params
>>> Sigma = result.cov_params()
>>> print(result.summary2())
```

```
=====
Model:                OLS      AIC:                147.773
Dependent Variable:   y          BIC:                149.765
No. Observations:    20          Log-Likelihood:     -71.887
Df Model:             1          F-statistic:        41.97
Df Residuals:        18          Prob (F-statistic): 4.3e-06
R-squared:            0.70        Scale:              86.157
Adj. R-squared:      0.68

-----+-----+-----+-----+-----+-----+
          Coef.  Std.Err.  t      P>|t|  [0.025  0.975]
-----+-----+-----+-----+-----+
const    24.6361  3.7871   6.5053  0.0000  16.6797  32.592
x1        0.4483  0.0692   6.4782  0.0000   0.3029   0.593
-----+-----+-----+-----+-----+
Omnibus:            1.996      Durbin-Watson:      2.75
Prob(Omnibus):      0.369      Jarque-Bera (JB):   1.63
Skew:               0.651      Prob(JB):           0.44
Kurtosis:           2.486      Condition No.:      100
=====
```

The summary output includes many advanced statistics which we don't have space to fully discuss here. For a trained practitioner these diagnostics are very useful for evaluating and comparing fits, especially for more complicated models; see [Wasserman2004] and the `statsmodels` project documentation for more details.

Bayesian Solution: Overview

The Bayesian result is encapsulated in the posterior, which is proportional to the product of the likelihood and the prior; in this case we must be aware that a flat prior is not uninformative. Because of the nature of the slope, a flat prior leads to a much higher probability for steeper slopes. One

might imagine addressing this by transforming variables, e.g. using a flat prior on the angle the line makes with the x -axis rather than the slope. It turns out that the appropriate change of variables can be determined much more rigorously by following arguments first developed by [Jeffreys1946].

Our model is given by $y = \alpha + \beta x$ with probability element $P(\alpha, \beta)d\alpha d\beta$. By symmetry, we could just as well have written $x = \alpha' + \beta'y$ with probability element $Q(\alpha', \beta')d\alpha' d\beta'$. It then follows that $(\alpha', \beta') = (-\beta^{-1}\alpha, \beta^{-1})$. Computing the determinant of the Jacobian of this transformation, we can then show that $Q(\alpha', \beta') = \beta^3 P(\alpha, \beta)$. The symmetry of the problem requires equivalence of P and Q , or $\beta^3 P(\alpha, \beta) = P(-\beta^{-1}\alpha, \beta^{-1})$, which is a functional equation satisfied by

$$P(\alpha, \beta) \propto (1 + \beta^2)^{-3/2}.$$

This turns out to be equivalent to choosing flat priors on the alternate variables $(\theta, \alpha_{\perp}) = (\tan^{-1} \beta, \alpha \cos \theta)$.

Through similar arguments based on the invariance of σ under a change of units, we can show that

$$P(\sigma) \propto 1/\sigma,$$

which is most commonly known as the *Jeffreys Prior* for scale factors after [Jeffreys1946], and is equivalent to flat prior on $\log \sigma$. Putting these together, we find the following uninformative prior for our linear regression problem:

$$P(\alpha, \beta, \sigma) \propto \frac{1}{\sigma} (1 + \beta^2)^{-3/2}.$$

With this prior and the above likelihood, we are prepared to numerically evaluate the posterior via MCMC.

Solution with emcee

The `emcee` package [ForemanMackey2013] is a lightweight pure-Python package which implements Affine Invariant Ensemble MCMC [Goodman2010], a sophisticated version of MCMC sampling. To use `emcee`, all that is required is to define a Python function representing the logarithm of the posterior. For clarity, we will factor this definition into two functions, the log-prior and the log-likelihood:

```
import emcee # version 2.0

def log_prior(theta):
    alpha, beta, sigma = theta
    if sigma < 0:
        return -np.inf # log(0)
    else:
        return (-1.5 * np.log(1 + beta**2)
                - np.log(sigma))

def log_like(theta, x, y):
    alpha, beta, sigma = theta
    y_model = alpha + beta * x
    return -0.5 * np.sum(np.log(2*np.pi*sigma**2) +
                        (y-y_model)**2 / sigma**2)

def log_posterior(theta, x, y):
    return log_prior(theta) + log_like(theta, x, y)
```

Next we set up the computation. `emcee` combines multiple interacting “walkers”, each of which results in its own Markov chain. We will also specify a burn-in period, to allow the chains to stabilize prior to drawing our final traces:

```
ndim = 3 # number of parameters in the model
nwalkers = 50 # number of MCMC walkers
nburn = 1000 # "burn-in" to stabilize chains
nsteps = 2000 # number of MCMC steps to take
starting_guesses = np.random.rand(nwalkers, ndim)
```

Now we call the sampler and extract the trace:

```
sampler = emcee.EnsembleSampler(nwalkers, ndim,
                               log_posterior,
                               args=[xdata, ydata])
sampler.run_mcmc(starting_guesses, nsteps)

# chain is of shape (nwalkers, nsteps, ndim):
# discard burn-in points and reshape:
trace = sampler.chain[:, nburn:, :].T
trace = trace.reshape(-1, ndim).T
```

The result is shown by the blue curve in Figure 1.

Solution with PyMC

The PyMC package [Patil2010] is an MCMC implementation written in Python and Fortran. It makes use of the classic Metropolis-Hastings MCMC sampler [Gelman2004], and includes many built-in features, such as support for efficient sampling of common prior distributions. Because of this, it requires more specialized boilerplate than does emcee, but the result is a very powerful tool for flexible Bayesian inference.

The example below uses PyMC version 2.3; as of this writing, there exists an early release of version 3.0, which is a complete rewrite of the package with a more streamlined API and more efficient computational backend. To use PyMC, we first we define all the variables using its classes and decorators:

```
import pymc # version 2.3

alpha = pymc.Uniform('alpha', -100, 100)

@pymc.stochastic(observed=False)
def beta(value=0):
    return -1.5 * np.log(1 + value**2)

@pymc.stochastic(observed=False)
def sigma(value=1):
    return -np.log(abs(value))

# Define the form of the model and likelihood
@pymc.deterministic
def y_model(x=xdata, alpha=alpha, beta=beta):
    return alpha + beta * x

y = pymc.Normal('y', mu=y_model, tau=1./sigma**2,
               observed=True, value=ydata)

# package the full model in a dictionary
model = dict(alpha=alpha, beta=beta, sigma=sigma,
             y_model=y_model, y=y)
```

Next we run the chain and extract the trace:

```
S = pymc.MCMC(model)
S.sample(iter=100000, burn=50000)
trace = [S.trace('alpha')[:,], S.trace('beta')[:,],
        S.trace('sigma')[:,]]
```

The result is shown by the red curve in Figure 1.

Solution with PyStan

PyStan is the official Python interface to Stan, a probabilistic programming language implemented in C++ and making

use of a Hamiltonian MCMC using a No U-Turn Sampler [Hoffman2014]. The Stan language is specifically designed for the expression of probabilistic models; PyStan lets Stan models specified in the form of Python strings be parsed, compiled, and executed by the Stan library. Because of this, PyStan is the least “Pythonic” of the three frameworks:

```
import pystan # version 2.2

model_code = """
data {
  int<lower=0> N; // number of points
  real x[N]; // x values
  real y[N]; // y values
}
parameters {
  real alpha_perp;
  real<lower=-pi()/2, upper=pi()/2> theta;
  real log_sigma;
}
transformed parameters {
  real alpha;
  real beta;
  real sigma;
  real ymodel[N];
  alpha <- alpha_perp / cos(theta);
  beta <- sin(theta);
  sigma <- exp(log_sigma);
  for (j in 1:N)
    ymodel[j] <- alpha + beta * x[j];
}
model {
  y ~ normal(ymodel, sigma);
}
"""

# perform the fit & extract traces
data = {'N': len(xdata), 'x': xdata, 'y': ydata}
fit = pystan.stan(model_code=model_code, data=data,
                 iter=25000, chains=4)
tr = fit.extract()
trace = [tr['alpha'], tr['beta'], tr['sigma']]
```

The result is shown by the green curve in Figure 1.

Comparison

The 1σ and 2σ posterior credible regions computed with these three packages are shown beside the corresponding frequentist confidence intervals in Figure 1. The frequentist result gives slightly tighter bounds; this is primarily due to the confidence interval being computed assuming a single maximum likelihood estimate of the unknown scatter, σ (this is analogous to the use of the single point estimate for the nuisance parameter p in the billiard game, above). This interpretation can be confirmed by plotting the Bayesian posterior conditioned on the maximum likelihood estimate $\hat{\sigma}$; this gives a credible region much closer to the frequentist confidence interval.

The similarity of the three MCMC results belie the differences in algorithms used to compute them: by default, PyMC uses a Metropolis-Hastings sampler, PyStan uses a No U-Turn Sampler (NUTS), while emcee uses an affine-invariant ensemble sampler. These approaches are known to have differing performance characteristics depending on the features of the posterior being explored. As expected for the near-Gaussian posterior used here, the three approaches give very similar results.

A main apparent difference between the packages is the

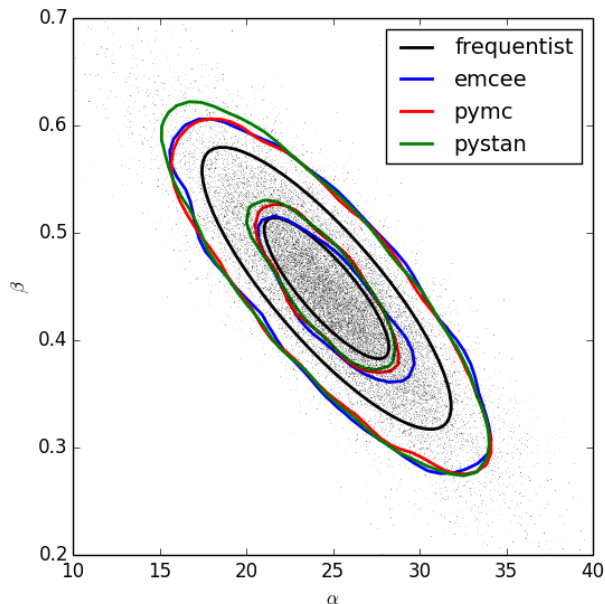


Fig. 1: Comparison of model fits using frequentist maximum likelihood, and Bayesian MCMC using three Python packages: *emcee*, *PyMC*, and *PyStan*.

Python interface. *Emcee* is perhaps the simplest, while *PyMC* requires more package-specific boilerplate code. *PyStan* is the most complicated, as the model specification requires directly writing a string of Stan code.

Conclusion

This paper has offered a brief philosophical and practical glimpse at the differences between frequentist and Bayesian approaches to statistical analysis. These differences have their root in differing conceptions of probability: frequentists define probability as related to *frequencies of repeated events*, while Bayesians define probability as a *measure of uncertainty*. In practice, this means that frequentists generally quantify the properties of *data-derived quantities* in light of *fixed model parameters*, while Bayesians generally quantify the properties of *unknown models parameters* in light of *observed data*. This philosophical distinction often makes little difference in simple problems, but becomes important within more sophisticated analysis.

We first considered the case of nuisance parameters, and showed that Bayesianism offers more natural machinery to deal with nuisance parameters through *marginalization*. Of course, this marginalization depends on having an accurate prior probability for the parameter being marginalized.

Next we considered the difference in the handling of uncertainty, comparing frequentist confidence intervals with Bayesian credible regions. We showed that when attempting to find a single, fixed interval bounding the true value of a parameter, the Bayesian solution answers the question that researchers most often ask. The frequentist solution can be informative; we just must be careful to correctly interpret the frequentist confidence interval.

Finally, we combined these ideas and showed several examples of the use of frequentism and Bayesianism on a more realistic linear regression problem, using several mature packages available in the Python language ecosystem. Together, these packages offer a set of tools for statistical analysis in both the frequentist and Bayesian frameworks.

So which approach is best? That is somewhat a matter of personal ideology, but also depends on the nature of the problem at hand. Frequentist approaches are often easily computed and are well-suited to truly repeatable processes and measurements, but can hit snags with small sets of data and models which depart strongly from Gaussian. Frequentist tools for these situations do exist, but often require subtle considerations and specialized expertise. Bayesian approaches require specification of a potentially subjective prior, and often involve intensive computation via MCMC. However, they are often conceptually more straightforward, and pose results in a way that is much closer to the questions a scientist wishes to answer: i.e. how do *these particular data* constrain the unknowns in a certain model? When used with correct understanding of their application, both sets of statistical tools can be used to effectively interpret of a wide variety of scientific and technical results.

REFERENCES

- [Bayes1763] T. Bayes. *An essay towards solving a problem in the doctrine of chances*. Philosophical Transactions of the Royal Society of London 53(0):370-418, 1763
- [Eddy2004] S.R. Eddy. *What is Bayesian statistics?*. Nature Biotechnology 22:1177-1178, 2004
- [Evans2002] S.N. Evans & P.B. Stark. *Inverse Problems as Statistics*. Mathematics Statistics Library, 609, 2002.
- [ForemanMackey2013] D. Foreman-Mackey, D.W. Hogg, D. Lang, J. Goodman. *emcee: the MCMC Hammer*. PASP 125(925):306-312, 2014
- [Gelman2004] A. Gelman, J.B. Carlin, H.S. Stern, and D.B. Rubin. *Bayesian Data Analysis, Second Edition*. Chapman and Hall/CRC, Boca Raton, FL, 2004.
- [Goodman2010] J. Goodman & J. Weare. *Ensemble Samplers with Affine Invariance*. Comm. in Applied Mathematics and Computational Science 5(1):65-80, 2010.
- [Hardy2003] M. Hardy. *An illuminating counterexample*. Am. Math. Monthly 110:234-238, 2003.
- [Hoffman2014] M.C. Hoffman & A. Gelman. *The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo*. JMLR, submitted, 2014.
- [Jaynes1976] E.T. Jaynes. *Confidence Intervals vs Bayesian Intervals (1976)* Papers on Probability, Statistics and Statistical Physics Synthese Library 158:149, 1989
- [Jeffreys1946] H. Jeffreys *An Invariant Form for the Prior Probability in Estimation Problems*. Proc. of the Royal Society of London. Series A 186(1007): 453, 1946
- [Patil2010] A. Patil, D. Huard, C.J. Fonnesbeck. *PyMC: Bayesian Stochastic Modelling in Python* Journal of Statistical Software, 35(4):1-81, 2010.
- [Seabold2010] J.S. Seabold and J. Perktold. *Statsmodels: Econometric and Statistical Modeling with Python* Proceedings of the 9th Python in Science Conference, 2010
- [VanderPlas2014] J. VanderPlas. *Frequentism and Bayesianism*. Four-part series (I, II, III, IV) on *Pythonic Perambulations* <http://jakevdp.github.io/>, 2014.
- [Wasserman2004] L. Wasserman. *All of statistics: a concise course in statistical inference*. Springer, 2004.