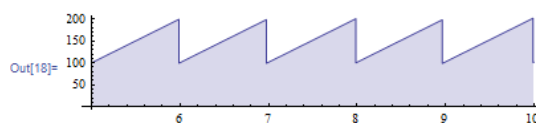


## 2. Distributions of random variables

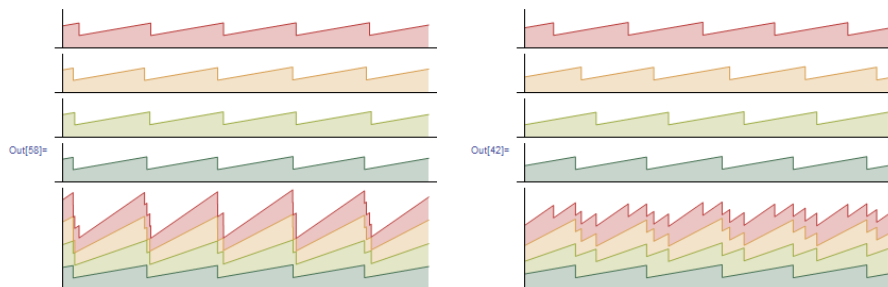
**Goals.** Get practice at generating and reasoning about random variables. Understand what the empirical distribution is, and what it is for. See how limit theorems are used, in the context of Monte Carlo estimation. Form an intuitive understanding of how random samples behave.

### 2.1. Working with random variables

**Application.** For most traffic flows on the Internet, the rate at which the server sends data is controlled by the TCP algorithm. It aims to detect Internet congestion, and it adjusts the data rate to strike a balance between ‘use all available capacity’ and ‘don’t cause overload’. It does this by steadily increasing the sending rate (by 1 packet per second every round trip time) until a packet is dropped, which signifies congestion, whereupon it cuts the sending rate in half. This produces the characteristic “TCP sawtooth”.



Suppose a network operator wants to build in enough capacity to support 1000 users each running at 30 kB/sec. How much capacity is needed? In the worst case the sawteeth might all be aligned, giving a peak rate of 40 MB/sec. (To find this, let  $x_{\text{peak}}$  be the peak rate, note that the trough is  $x_{\text{trough}} = x_{\text{peak}}/2$  because of TCP’s congestion rule. The average is  $(x_{\text{trough}} + x_{\text{peak}})/2$  and this is 30 kB/sec. Solving for  $x_{\text{peak}}$  gives 40 kB/sec.) Intuitively we might guess that perfect alignment is unlikely, and that the troughs on one sawtooth are likely to cancel out the troughs on another. This is called *statistical multiplexing*. How much statistical multiplexing should we expect?



#### RULES FOR EXPECTATION AND VARIANCE

This section of the course is all about numerical random variables. What makes them particularly useful is that they can be summed and averaged, which lets us define their *expectation*. They’re so useful that we often write ‘random variable’ to mean ‘numerical random variable’, and use other wording when it’s not numerical.

$$\mathbb{E}X = \sum_x x\mathbb{P}(X = x) \quad \text{for a discrete random variable,}$$

$$\mathbb{E}X = \int_x xf(x) \quad \text{for a continuous random variable with density } f.$$

For a function of a random variable  $Y = f(X)$ , there are two equivalent ways to compute the expectation:

$$\mathbb{E}f(X) = \sum_y y\mathbb{P}(f(X) = y) = \sum_x f(x)\mathbb{P}(X = x)$$

and similarly for continuous random variables. Now, some handy results about expectation. For any random variable  $X$ , the *variance* and *standard deviation* are

$$\text{Var } X = \mathbb{E}((X - \mathbb{E}X)^2), \quad \text{std. dev}(X) = \sqrt{\text{Var } X}.$$

This equality is called the *law of the unconscious statistician*, because it’s easy to interchange them without even realizing one is doing so.

For all constants  $a$  and  $b$ ,

$$\begin{aligned}\mathbb{E}(aX + b) &= a(\mathbb{E}X) + b \\ \text{Var}(aX + b) &= a^2 \text{Var} X \\ \text{std. dev}(aX + b) &= a^2 \text{std. dev}(X).\end{aligned}$$

For any two random variables  $X$  and  $Y$ ,

$$\mathbb{E}(X + Y) = (\mathbb{E}X) + (\mathbb{E}Y).$$

For any two independent random variables  $X$  and  $Y$ ,

$$\begin{aligned}\mathbb{E}(XY) &= (\mathbb{E}X)(\mathbb{E}Y) \\ \text{Var}(X + Y) &= \text{Var} X + \text{Var} Y \\ \text{std. dev}(X + Y) &= \sqrt{\text{std. dev}(X)^2 + \text{std. dev}(Y)^2}.\end{aligned}$$

The *covariance* of two random variables  $X$  and  $Y$  is

$$\text{Cov}(X, Y) = \mathbb{E}((X - \mathbb{E}X)(Y - \mathbb{E}Y)).$$

The *conditional expectation*  $\mathbb{E}(X | Y = y)$ , for a discrete random variable  $Y$ , is

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

We write  $\mathbb{E}(X | Y)$  to mean “Define  $f(y) = \mathbb{E}(X | Y = y)$ , and return  $f(Y)$ ”. This is a random variable (because it’s a function of  $Y$  which is itself a random variable). When  $Y$  is a continuous random variable,  $\mathbb{P}(Y = y) = 0$  so the conditioning doesn’t make sense—it’s a divide-by-zero error—so we just replace probabilities by densities and sums by integrals.

## CONFIDENCE INTERVALS

Here are two facts about the normal distribution: if  $X$  has a normal distribution then so does  $aX + b$  for any constants  $a$  and  $b$ ; and the interval  $[-1.96, 1.96]$  is a 95% confidence interval for  $\text{Normal}(0, 1)$  i.e.

$$\mathbb{P}(-1.96 \leq \text{Normal}(0, 1) \leq 1.96) \approx 0.95.$$

**Example.** What is a 95% confidence interval for  $\text{Normal}(\mu, \sigma^2)$ , the normal distribution with mean  $\mu$  and variance  $\sigma^2$ ?

Let  $X \sim \text{Normal}(\mu, \sigma^2)$ . Using the rules for expectation and variance,  $\mathbb{E}(X - \mu) = (\mathbb{E}X) - \mu = 0$ , and  $\text{Var}(X - \mu) = \text{Var} X = \sigma^2$ , thus  $\text{Var}((X - \mu)/\sigma) = 1$ . So  $(X - \mu)/\sigma \sim N(0, 1)$ , thus

$$\mathbb{P}(-1.96 \leq \frac{X - \mu}{\sigma} \leq 1.96) \approx 0.95.$$

Rearranging the expression inside the brackets,

$$\mathbb{P}(\mu - 1.96\sigma \leq \text{Normal}(\mu, \sigma^2) \leq \mu + 1.96\sigma) \approx 0.95. \quad (5)$$

\* \* \*

Here’s a general purpose rule of thumb:

*A random variable  $X$  can be approximated by  $\text{Normal}(\mathbb{E}X, \text{Var} X)$ .*

This is so useful and simple that it can’t possibly be true—but what’s surprising is that it’s often nearly true. We’ll see circumstantial evidence for why the approximation is so good in Section 2.3, and in the example sheet you’ll investigate cases where it doesn’t work.

**Example.** I throw a die 100 times and compute the total score. What range of values should I expect?

Let  $X$  be the outcome of a single throw. We can explicitly calculate its mean and variance:

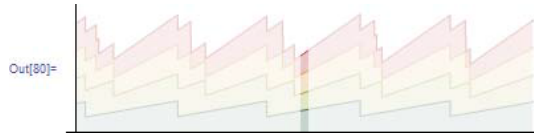
$$\begin{aligned}\mathbb{E}X &= 1 \times 1/6 + 2 \times 1/6 + \cdots + 6 \times 1/6 = 7/2, \\ \text{Var } X &= (1 - 7/2)^2 \times 1/6 + \cdots + (6 - 7/2)^2 \times 1/6 = 35/12.\end{aligned}$$

Let  $Y$  be the sum of 100 independent copies of  $X$ . By the rules for mean and variance of sums of independent random variables,

$$\mathbb{E}Y = 100 \times 7/2, \quad \text{Var } Y = 100 \times 35/12.$$

Using the normal approximation,  $Y \approx \text{Normal}(700/2, 3500/12)$ . Applying the approximation(5), we are 95% confident that  $Y$  lies in the range [316, 384].

**Example (statistical multiplexing).** Returning to the TCP example, consider an arbitrary instant in time. Let  $X_1, \dots, X_n$  be the sending rate of each of  $n = 1000$  flows at this time, and let  $Y = X_1 + \cdots + X_n$  be the total.



We might have caught a flow at any point in its sawtooth, so each  $X_i$  might take any value between the trough and the peak i.e. in the range  $[2x/3, 4x/3]$  where  $x = 30$  kB/sec is the average rate we want to support. Furthermore, because of the shape of the sawtooth, each value in this range is equally likely. The appropriate distribution is thus

$$X_i \sim \text{Uniform}(2x/3, 4x/3)$$

After looking up the formulae for mean and variance on Wikipedia,

$$\begin{aligned}\mathbb{E}X_i &= \frac{2x/3 + 4x/3}{2} = x \\ \text{Var } X_i &= \frac{(4x/3 - 2x/3)^2}{12} = \frac{x^2}{27}.\end{aligned}$$

Using the rule for expectation of sums,

$$\mathbb{E}Y = \mathbb{E}X_1 + \cdots + \mathbb{E}X_n = nx$$

and assuming the  $X_i$  are all independent then

$$\begin{aligned}\text{Var } Y &= \text{Var } X_1 + \cdots + \text{Var } X_n = \frac{nx^2}{27} \\ \text{std. dev}(Y) &= \sqrt{\text{Var } Y} = \sqrt{n} \frac{x}{\sqrt{27}}.\end{aligned}$$

With probability 95%,  $Y$  will lie in the range

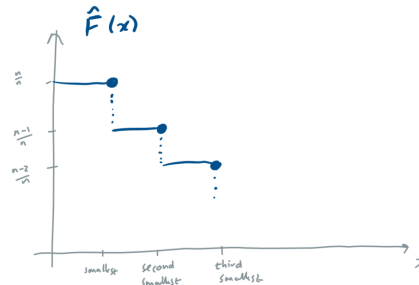
$$[\mathbb{E}Y - 1.96 \text{std. dev}(Y), \mathbb{E}Y + 1.96 \text{std. dev}(Y)]$$

which evaluates to [29.8, 30.2] MB/sec. This is much less than the worst-case value 40 MB/sec.

## 2.2. Generating custom random variables

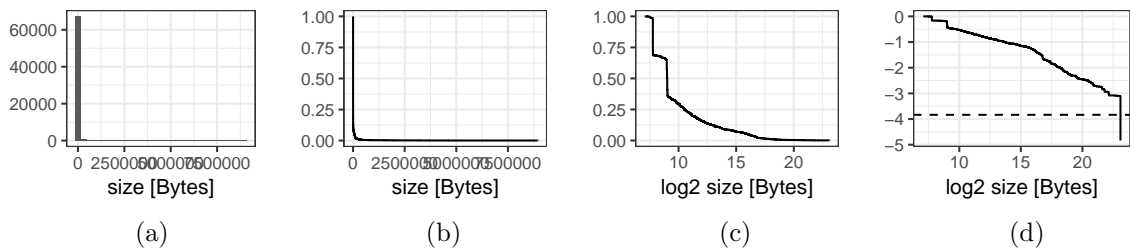
**Application.** I've collected logs from my web server,  $n = 68,506$  records, and I want to program a random number generator that mimics the file sizes I see in these logs. I start by plotting a histogram of file size, shown as (a) below. This is useless, because nearly all sizes are tiny and a handful are gigantic, and the binning of the histogram hides all the detail. A good way to see more detail is to plot what is known as the *empirical distribution*,

$$\hat{F}(x) = \frac{1}{n} (\text{how many items there are } \geq x).$$



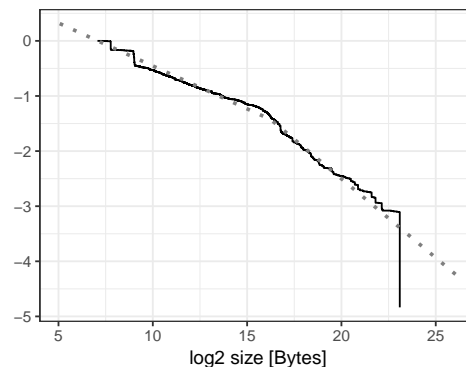
Warning: there is also the empirical cumulative distribution function, which counts the number of datapoints that are  $\leq x_i$ . When you read 'distribution function', you need to work out from the context whether the writer means a cumulative distribution function or a tail distribution function.

which is easy to plot by sorting the data and putting it on the  $x$  axis. The empirical distribution of web server file sizes is shown in (b). It's still not showing very much detail because of the scale, so I'll apply the golden rule of engineering: "if you don't like what you see, take logs". In (c) I've taken logs of the  $x$  axis and in (d) I've also taken logs of the  $y$  axis, and that makes my data look nice and regular. The dotted reference line is at  $\log_{10}(10/n)$  — this way I can see that the precipitous drop at the right hand side of (d) isn't just a single outlier, but it's not much more than 10 datapoints.



How do I know which standard random variable to use, to match this dataset? Or, even better, can I construct a custom random number generator to match? It looks like the data is trying to tell me there are two straight lines (plus a handful of huge files, which I'll ignore for now), i.e. that for some parameters  $\alpha$ ,  $\beta$ ,  $\gamma$  and  $\theta$  which I can fit from the data,

$$\log \hat{F}(x) \approx \alpha - \beta \log x - \gamma \max(\log x - \theta, 0).$$



### THE DISTRIBUTION FUNCTION AND THE INVERSION METHOD

The *distribution function* of a random variable  $X$  is

$$F(x) = \mathbb{P}(X \geq x).$$

Sometimes it's easier to work with the distribution function rather than the density:

**Example.** What is the density function of the continuous random variable  $X$  generated by this code?

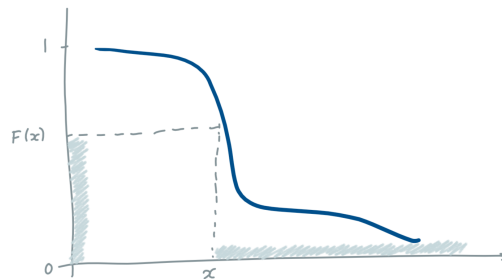
```
1 def rx():
2     U = random.random()
3     return U*U
```

Let's work out its distribution function first.

$$\mathbb{P}(X \geq x) = \mathbb{P}(U^2 \geq x) = \mathbb{P}(U \geq \sqrt{x}).$$

Since  $U$  is a simple uniform random variable on  $[0, 1]$ ,  $\mathbb{P}(U < u) = u$ , and so  $\mathbb{P}(X \geq x) = 1 - \sqrt{x}$ . For continuous random variables, the distribution function and the density function are related to other by integration,  $\mathbb{P}(X \geq x) = \int_x^\infty f(y) dy$ , so  $f(x) = -F'(x)$ . In this case, we end up with  $f(x) = 1/(2\sqrt{x})$ .

There is a universal way to generate a random variable given its distribution function, called the *inversion method*. (1) Generate a simple random variable  $U \sim \text{Uniform}[0, 1]$ . (2) Solve  $F(X) = U$ . (3) That's it,  $X$  has distribution function  $F$ . This plot shows why the method works:



it ensures that for every  $x$  the event  $\{X \geq x\}$  is precisely the event  $\{U \leq F(x)\}$ , which has probability  $F(x)$ . Intuitively, in regions where the density  $f$  is high then the distribution  $F$  will be steep, and so  $X$  is more likely to hit those regions.

The inversion method requires us to solve  $F(X) = U$ , which is easy to do algebraically for simple continuous functions like the two-straight-line fit we found earlier. The method is also correct for discrete random variables, whose step functions are staircases, but here we usually need an algorithmic solution rather than an algebraic one. Section 2.5 and the example sheet look more closely at generating discrete random variables.

\* \* \*

It's worth mentioning some terminology for describing distribution functions. The

<i>first quartile</i>	is a number $x$ such that	$\mathbb{P}(X \leq x) = 25\%$
<i>median</i>	...	$\mathbb{P}(X \leq x) = 50\%$
<i>third quartile</i>	...	$\mathbb{P}(X \leq x) = 75\%$
<i>p-percentile</i>	...	$\mathbb{P}(X \leq x) = p\%$

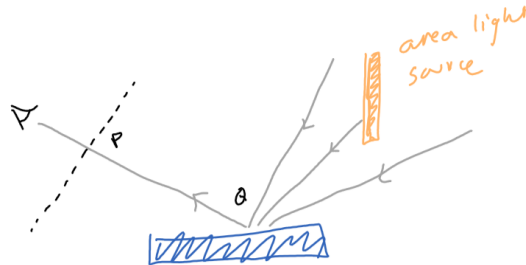
For discrete random variables it may not be possible to get exact percentiles, and there is no convention about rounding.

The range  $[x_1, x_2]$  is called a *95% confidence interval* if  $\mathbb{P}(x_1 \leq X \leq x_2) = 95\%$ . Often we choose a two-sided confidence interval with  $\mathbb{P}(X < x_1) = \mathbb{P}(X > x_2) = 2.5\%$ . In some contexts it may be more useful to report a one-sided confidence interval, either  $[x_1, \infty)$  or  $(-\infty, x_2]$ .

The *cumulative distribution function* is  $\text{CDF}(x) = \mathbb{P}(X \leq x)$ . What we've been calling the distribution is also called the *tail distribution function*, to disambiguate the two.

## 2.3. Limit theorems

**Application.** In computer graphics rendering and shading, we can compute the colour a pixel on the screen by reasoning about light rays. First figure out the surface point  $Q$  that is to be shown at pixel  $P$ , by casting a ray from the camera through  $P$  and finding what surface it hits. Then figure out the colour and shading of  $Q$  by adding up all the light rays that might be illuminating it and reflecting out through  $P$ .



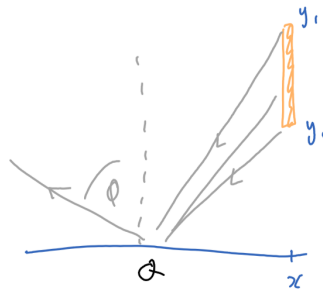
The surface might be perfectly reflective, or perfectly diffuse, or more generally we can model it with a specular lobe function  $\text{BRDF}(\theta, \phi)$ , which measures how much light is emitted at angle  $\phi$  when it comes in at angle  $\theta$ .



When we take into account the intensity of light glancing the surface as a function of angle  $\theta$ , the total light reflected at angle  $\phi$ , from a point light source of intensity  $I$ , is

$$I \cos(\theta) \text{BRDF}(\theta, \phi).$$

If illumination comes from an area light source, then we treat it as though the total intensity  $I$  is smeared across a set of point light sources:



$$\int_{y=y_0}^{y_1} \frac{I}{y_1 - y_0} \cos(\tan^{-1}(y/x)) \text{BRDF}(\tan^{-1}(y/x), \phi) dy.$$

\* \* \*

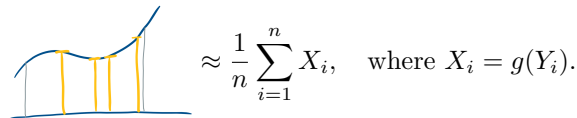
In a more abstract setting, suppose we want to compute

$$\int_{y=a}^b g(y) \frac{1}{b-a} dy.$$

The obvious method is to split the  $y$  range into  $n$  equally sized pieces, and approximate the function by a series of rectangles, e.g. taking the height of the rectangle to be the value of  $g$  at the midpoint.

$$\approx \frac{1}{n} \sum_{i=1}^n x_i, \quad \text{where } x_i = g\left(a + \frac{b-a}{n}(i-1/2)\right).$$

But there’s actually nothing special about sampling  $g$  at grid points. Why not just pick the sampling points at random? In other words, pick  $n$  independent  $\text{Uniform}[a, b]$  random variables  $Y_1, \dots, Y_n$ , and approximate



$$\approx \frac{1}{n} \sum_{i=1}^n X_i, \quad \text{where } X_i = g(Y_i).$$

This is called *Monte Carlo* integration. Is it any good, and if so why?

EXPECTATION

Let’s calculate the expected value of the Monte Carlo approximation.

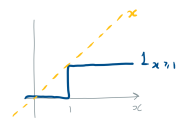
$$\begin{aligned} \mathbb{E}\left(\frac{1}{n} \sum_{i=1}^n X_i\right) &= \frac{1}{n} \sum_{i=1}^n \mathbb{E}X_i \quad \text{by linearity of expectation} \\ &= \frac{1}{n} n \mathbb{E}X_1 \quad \text{since they are IID} \\ &= \mathbb{E}g(Y_1) = \int_{y=a}^b \frac{1}{b-a} g(y) dy \quad \text{by definition of expectation.} \end{aligned}$$

So, the expected value is exactly the integral we want to compute. But this is just punting the question. What does the expected value have to do with anything?

WEAK LAW OF LARGE NUMBERS

Let’s calculate the probability of error. Let  $\mu = \mathbb{E}X_1$  be the integral we want to compute, and let  $\varepsilon > 0$ . Then

$$\begin{aligned} \mathbb{P}\left(\left|\frac{1}{n} \sum X_i - \mu\right| > \varepsilon\right) &= \mathbb{P}\left(\frac{(n^{-1} \sum X_i - \mu)^2}{\varepsilon^2} > 1\right) \quad \text{by simple algebra} \\ &= \mathbb{E}\left(\mathbb{1}\left[\frac{(n^{-1} \sum X_i - \mu)^2}{\varepsilon^2} > 1\right]\right) \quad \text{since } \mathbb{E}\mathbb{1}_A = \mathbb{P}(A) \\ &\leq \mathbb{E}\left(\frac{(n^{-1} \sum X_i - \mu)^2}{\varepsilon^2}\right) \quad \text{since } \mathbb{1}_{x \geq 1} \leq x \\ &= \frac{1}{\varepsilon^2} \text{Var}\left(\frac{1}{n} \sum X_i\right) \quad \text{by linearity of } \mathbb{E} \text{ and definition of Var} \\ &= \frac{1}{n^2 \varepsilon^2} n \text{Var} X_1 \quad \text{by linearity of Var, and independence} \\ &= \frac{1}{n} \frac{\text{Var} X_1}{\varepsilon^2}. \end{aligned}$$



So, the more samples we use the better accuracy we get, and the accuracy depends on the variance of an individual sample.

This result is known as the *weak law of large numbers*. Another way to state it is in terms of the distribution function,

$$\mathbb{P}\left(\frac{1}{n} \sum_{i=1}^n X_i \geq x\right) \rightarrow \mathbb{1}_{x \geq \mu} \quad \text{as } n \rightarrow \infty$$

for all  $x$  where the right hand side (viewed as a function of  $x$ ) is continuous.

It’s useful to know not only that there is convergence, but also how fast convergence happens. We just derived an upper bound on the probability of error. Is it tight?

Why is it called weak? There is also a strong law, which says  $\mathbb{P}(\text{err}_n \rightarrow 0) = 1$ , as opposed to the weak law  $\mathbb{P}(\text{err}_n > \varepsilon) \rightarrow 0$ . The strong law implies the weak law, but is harder to prove.

THE CENTRAL LIMIT THEOREM

In Section 2.1 we saw a general purpose approximation:

$$X_1 \approx \text{Normal}(\mu, \sigma^2) \quad \text{where } \mu = \mathbb{E}X_1 \text{ and } \sigma^2 = \text{Var} X_1.$$

If we apply this approximation to all the  $X_i$ ,

$$\begin{aligned}\sum X_i &\approx \text{Normal}(n\mu, n\sigma^2) \\ \frac{1}{n} \sum X_i &\approx \text{Normal}\left(\mu, \frac{\sigma^2}{n}\right) \\ \frac{1}{n} \sum X_i - \mu &\approx \text{Normal}\left(0, \frac{\sigma^2}{n}\right) \\ \frac{n^{-1} \sum X_i - \mu}{\sqrt{\sigma^2/n}} &\approx \text{Normal}(0, 1).\end{aligned}$$

Another result, the Berry-Esseen theorem, gives a bound on the error in this limit statement.

It is a mathematical theorem that this approximation becomes increasingly accurate (assuming the  $X_i$  are independent and that  $\mu$  and  $\sigma^2$  are finite). The *central limit theorem* makes this precise, as a statement about convergence of the distribution function:

$$\mathbb{P}\left(\sqrt{n}\left(\frac{n^{-1} \sum X_i - \mu}{\sigma}\right) \geq x\right) \rightarrow \mathbb{P}(\text{Normal}(0, 1) \geq x) \quad \text{as } n \rightarrow \infty, \text{ for all } x.$$

I find it more helpful to remember the central limit theorem when it's written as an approximation,

$$\frac{1}{n} \sum_{i=1}^n X_i \approx \text{Normal}\left(\mu, \frac{\sigma^2}{n}\right).$$

We could use this to estimate how many samples we need for the Monte Carlo integration to reach a target level of accuracy, if we knew  $\sigma$ . It tells us that, with probability 95%,  $n^{-1} \sum X_i$  is within  $\pm 1.96\sigma$  of  $\mu$ . We don't know the true  $\sigma$ , but we can just estimate it with Monte Carlo approximation!

$$\sigma^2 = \mathbb{E}(X_1 - \mu)^2 \approx \frac{1}{n} \sum_{i=1}^n (X_i - \mu)^2$$

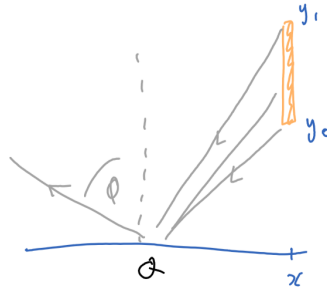
and also plug in the Monte Carlo approximation for  $\mu$ . Our computer graphics code could keep generating more and more samples until it decides the error is small enough as to be imperceptible.

This should make us uneasy! How come it's OK to say "Plug in the Monte Carlo approximations to get a point estimate for  $\sigma$ ", and at the same time "Don't just use the plain Monte Carlo approximation for  $\mu$ , use the central limit theorem to work out how accurate it is"? We will revisit this question in Section 3.



## 2.4. Importance sampling

In the computer graphics problem of Section 2.3, we studied how to use Monte Carlo integration to calculate the illumination at a pixel.



The answer we want is

$$\int_{y=y_0}^{y_1} \frac{1}{y_1 - y_0} g(y) dy$$

where  $g(y)$  is some complicated formula involving the specular characteristics of the surface and the angle of illumination. We approximated it by

$$\frac{1}{n} \sum_{i=1}^n g(Y_i)$$

where  $Y_1, \dots, Y_n$  are independent random variables drawn uniformly from the range  $[y_0, y_1]$ . In words,

$$\begin{array}{l} \text{reflected light} \\ \text{at angle } \phi \end{array} \approx \frac{1}{n} \sum_{i=1}^n \begin{array}{l} \text{light due to a simulated ray coming from} \\ \text{a random point } Y_i \text{ on the light source.} \end{array}$$

We found that the approximation error is of the order of  $\sigma/\sqrt{n}$ , where  $\sigma$  is the standard deviation of the answer from a single light ray. This suggests two questions. First, is there a way to change the simulation to reduce  $\sigma$ ? Second, if it's a highly reflective surface, then there isn't any need to simulate the entire light source, since only a few pieces of it will end up reflected at angle  $\phi$ : how can we change the simulation to achieve this? It turns out that these two questions are asking exactly the same thing.



Let's first try a naive change to the simulation. Physics tells us that the BRDF function is symmetrical. Consider picking a random point  $Y$  on the light source, but not uniformly at random: pick it instead so that angles closer to the center of the specular lobe are more likely. (See Section 2.2 for how to generate a random variable from an arbitrary distribution.) Let the density function be  $f(y)$ , and let  $X = g(Y)$  as before. Then

$$\mathbb{E}X = \int_{y_0}^{y_1} g(y)f(y) dy$$

and if we use Monte Carlo integration we'll get an answer concentrated around  $\mathbb{E}X$  — which is wrong, not the integral we want. But looking at the integral more closely suggests the fix: we should use  $X' = g(Y)/f(Y)$  instead, so that  $f$  cancels out, and we end up with exactly the integral we want. In words,

$$\begin{array}{l} \text{reflected light} \\ \text{at angle } \phi \end{array} \approx \frac{1}{n} \sum_{i=1}^n \left( \begin{array}{l} \text{light due to a simulated ray coming from} \\ \text{a random point } Y_i \text{ drawn from density } f \end{array} / f(Y_i) \right).$$

This is called *importance sampling*. Whatever distribution we choose for the  $Y_i$ , Monte Carlo integration will give us the right answer, and we have a whole design space of sampling distributions to choose from. Our aim is to choose a density function  $f$  to reduce  $\sigma^2 = \text{Var}(g(Y)/f(Y))$ .

\* \* \*

It's surprisingly easy to design the optimal sampling distribution. From the definition of variance,

$$\text{Var} \frac{g(Y)}{f(Y)} = \mathbb{E} \left[ \left( \frac{g(Y)}{f(Y)} - \mu \right)^2 \right] = \mathbb{E} \left[ \left( \frac{g(Y)}{f(Y)} \right)^2 \right] - \mu^2 \quad \text{where} \quad \mu = \mathbb{E} \frac{g(Y)}{f(Y)}.$$

The whole point of importance sampling is that  $\mu$  doesn't depend on how we choose  $f$ . The only term we have left to minimize is

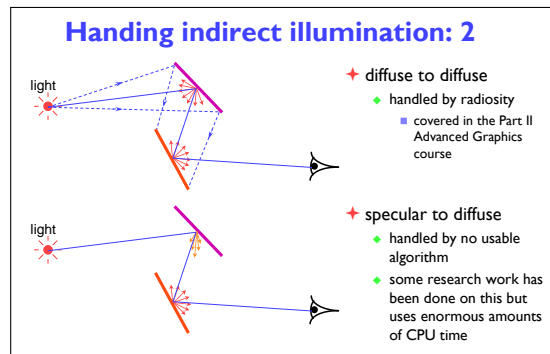
$$\mathbb{E} \left[ \left( \frac{g(Y)}{f(Y)} \right)^2 \right] = \int \left( \frac{g(y)}{f(y)} \right)^2 f(y) dy = \int \frac{g(y)^2}{f(y)} dy$$

and it turns out (using some standard tools from optimization theory) that to minimize this we should pick a density function  $f$  such that  $f(y)$  is proportional to  $g(y)$ .

Unfortunately, if we try to sample a random variable from  $\text{const} \times g(y)$  using the inversion method, we have to integrate  $g(y)$ , which defeats the whole purpose of Monte Carlo integration!

Importance sampling is nonetheless useful as a heuristic. The full distributed ray tracing algorithm, taking account of indirect illumination, is "To work out the shading at a point  $Q$ , sample light rays by following them backwards; pick a random incoming angle at each bounce, and heuristically try to pick an angle in proportion to how much light is expected from that angle." No matter how bad the heuristic, importance sampling will give the right answer for large enough  $n$ . If the heuristic is good, it will give the right answer for small  $n$ .

This is why specular-to-diffuse lighting is tricky: the question "which angle is likely to give most illumination?" can't be answered with only local knowledge at the diffuse surface.<sup>22</sup>



<sup>22</sup>Slide from IA Introduction to Graphics.

## 2.5. The empirical distribution

The empirical distribution function of a dataset  $x_1, \dots, x_n$  is

$$\hat{F}(x) = \frac{1}{n} (\text{how many items there are } \geq x).$$

In Section 2.2 we looked at fitting: we started with a family of distribution functions  $F_\theta(x)$  with some parameter or list of parameters  $\theta$ , we picked specific parameter values  $\hat{\theta}$  so that  $\hat{F}(x) \approx F_{\hat{\theta}}(x)$ , and then we used the inversion method to generate random variables from this fitted distribution. We were implicitly assuming that the empirical distribution function of a random sample should be close to the true distribution function from which the random sample was generated. Now, having learnt about limit theorems in Section 2.3, we can investigate this assumption.

Consider a random sample  $X_1, \dots, X_n$ , independent random variables with common distribution function  $F(x) = \mathbb{P}(X_i \geq x)$ . Let  $\hat{F}_n(x)$  be the empirical distribution function. Then

$$\mathbb{E}\hat{F}_n(x) = \mathbb{E}\left(\frac{\sum_{i=1}^n 1_{X_i \geq x}}{n}\right) = \mathbb{E}1_{X_1 \geq x} = \mathbb{P}(X_1 \geq x) = F(x).$$

By the weak law of large numbers,

$$\mathbb{P}(|\hat{F}_n(x) - F(x)| > \varepsilon) \rightarrow 0 \quad \text{as } n \rightarrow \infty$$

for any  $\varepsilon > 0$ . By the central limit theorem,

$$\hat{F}_n(x) \approx \text{Normal}\left(F(x), \frac{\sigma^2}{n}\right) \quad \text{where } \sigma^2 = \text{Var } 1_{X_1 \geq x} = F(x)(1 - F(x))$$

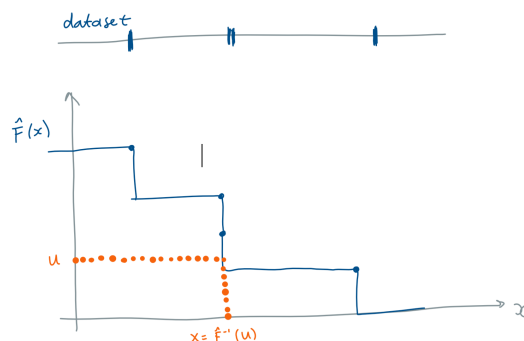
and so a 95% confidence interval for  $\hat{F}_n(x)$  is

$$\mathbb{P}\left(\hat{F}_n(x) \text{ is in the range } F(x) \pm 1.96\sqrt{\frac{F(x)(1 - F(x))}{n}}\right) \approx 0.95.$$

### RESAMPLING

When all we have is a dataset, how do we choose which family of distributions  $F_\theta(x)$  to fit? Sometimes there are sound scientific reasons for choosing a particular family, and our goal is to integrate this background scientific knowledge with the dataset at hand. If there is no background science, then it's daft to use the data to fit a parameterized distribution function as we did in Section 2.2 when a *perfect* fit is staring us in the face, namely the empirical distribution itself! This is a perfectly valid distribution function: it starts at 1, and ends at 0, and is decreasing.

We can sample from the empirical distribution function using the inversion method—and a moment's thought tells us that this is exactly the same as picking a value at random from the dataset, each item in the dataset equally likely. The overall concept, that the best-fitting distribution for a dataset is nothing other than its empirical distribution, is called *resampling*. Conventionally, a superscript \* denotes a random variable drawn in this way from the empirical distribution.



**Example 2.1 (Exact resampling).** Given a dataset  $x_1, \dots, x_n$ , what are  $\mathbb{E}X^*$  and  $\text{Var} X^*$ ? From the definition of expectation,

$$\mathbb{E}X^* = \sum_{x \in \{x_1, \dots, x_n\}} x \mathbb{P}(X^* = x) = \sum_i x_i \frac{1}{n}$$

i.e. the sample mean, often written  $\bar{x}$ . From the definition of variance,

$$\text{Var} X^* = \sum_{x \in \{x_1, \dots, x_n\}} (x - \bar{x})^2 \mathbb{P}(X^* = x) = \sum_i (x_i - \bar{x})^2 \frac{1}{n}$$

i.e. the sample variance.

**Example 2.2 (Monte Carlo resampling).** I collected a dataset  $x_1, \dots, x_n$  and I found the sample mean  $\bar{x}$ . If I repeat the exercise and collect further datasets, how much variability should I expect to see in the sample mean?

The best-fitting distribution is the dataset itself, and the best we can do is assume that subsequent datasets will be drawn from the same distribution. In other words, the next time I collect data, I'll expect to see something like  $\bar{X}_n^*$ , the sample mean of  $n$  random values drawn from the empirical distribution. This is a random variable. It's impractical to do an exact calculations about the distribution of  $\bar{X}_n^*$  as we did in Example 2.1 because there are so many possible values that  $\bar{X}_n^*$  might take—but it's straightforward to use Monte Carlo approximation instead, to find e.g.  $\mathbb{P}(\bar{X}_n^* \geq x)$  or to draw a histogram.

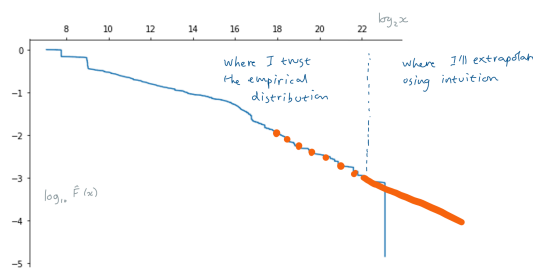
```

1 x = [13, 5, 2, ...] # the dataset
2 def sim_mean():
3     n = len(x)
4     X = random.choices(x, k=n) # k = number of samples to draw
5     return sum(X)/n
6
7 mc_samples = [sim_mean() for i in range(100000)]
8 matplotlib.pyplot.hist(mc_samples)

```

**Example 2.3 (Parametric resampling).** I collected a dataset  $x_1, \dots, x_n$ , and I found that the maximum value was  $m$ . If I repeat the exercise and collect further datasets, how much variability should I expect to see in the maximum? Resampling from the empirical distribution is unable to give an answer  $> m$ , but intuitively I feel that a new dataset might have larger values.

The real question here is: where does my intuition ‘larger values are possible’ come from, and how can I translate it into maths? Perhaps my intuition comes from seeing a straight line on a log-plot of the empirical distribution, as in Section 2.2. If this is so, then I could construct a new semi-parametric distribution function, which starts with the empirical data and switches over at some point to a straight line, whose slope and intercept parameters are fitted from the data. Sampling from this semi-empirical distribution could potentially produce values  $> m$ .



\* \* \*

When should we use parametric models and when should we use the dataset itself, in the form of the empirical distribution? There are no general rules.

- A dataset cannot tell us about values beyond the dataset. This has to come from our background knowledge or intuition. Integrating datasets and background knowledge is an art.
- A parametric distribution saves space: it only needs us to store a handful of parameters, rather than the full dataset. But this is often a premature optimization. For a small dataset of a few tens of thousands of values, on a modern computer, you should spend your time thinking about modeling and not about optimizing storage. For a large dataset, a model with a handful of parameters cannot hope to capture the richness of the data.
- Neural networks are parametric models, one parameter per weight of an edge. A neural network trained for simple image classification might take 140 million parameters, one for each connection. The human brain has roughly  $10^{15}$  connections, and a human lifetime is roughly  $2.5 \times 10^{15}$  microseconds. It seems that making sense of data is more about what you do with it than how you can compress it.
- High-dimensional modeling, i.e. modeling with more parameters than there are samples in the dataset, is an area of active research.

## PARTICLE FILTERS

*Application.* Suppose we’re trying to pinpoint a person’s location, based on noisy GPS readings. First, imagine that the person is not moving, and that we simply want to draw a shape on a map to indicate where the person is likely to be. We could represent our uncertainty about the person’s location by a probability density function, with  $f_0(x)$  our initial belief (spread widely over a map region), and  $f_t(x)$  our belief after  $t$  GPS readings. Using Bayes’ rule,

$$f_t(x) = \frac{\mathbb{P}(r_t | x) f_{t-1}(x)}{\int_y \mathbb{P}(r_t | y) f_{t-1}(y) dy} \quad (6)$$

where  $r_t$  is the reading at time  $t$ . We could implement this for example by storing an array with the value of  $f_t(x)$  at every location  $x$  on a fine enough mesh, and updating the entire array every timestep, replacing the integral by a sum over all points on the mesh.

To take account of motion, we could add dimensions to  $x$  for current velocity and current mode (walking, running, cycling); and we could change (6) to include the system dynamics—i.e. include terms describing how likely a new location is given the old location and velocity, and how likely a new velocity is given the old velocity and mode. The  $f_t(\cdot)$  array would be gigantic.

If more sensors are available, e.g. the gyroscope and compass and accelerometer on a mobile phone, we could change (6) to include terms describing the chance of each of those readings given the present state. This is known as *sensor fusion*.

Instead of storing the density function  $f_t(\cdot)$  as an array of values over mesh, why not approximate it by a random sample drawn from the distribution? We’ve already shown that the empirical distribution converges to the true distribution as sample size increases. We just need an empirical equivalent of (6), i.e. an efficient way to generate a sample from  $f_t(\cdot)$  given a sample from  $f_{t-1}(\cdot)$  and an observation  $r_t$ .

This approach works, and it is called a *particle filter*. Here is an example<sup>23</sup>. It shows a cluster of particles representing the current belief about a person’s position inside a building. The readings are gyroscope and compass and accelerometer. The line shows the path that the person actually followed.

<sup>23</sup>Julian Straub. “Pedestrian Indoor Localization And Tracking Using A Particle Filter Combined with a Learning Accessibility Map”. MA thesis. Technische Universität München, 2010. URL: <http://people.csail.mit.edu/jstraub/pub/Pedestrian-Indoor-Localization-and-Tracking-using-a-Particle-Filter-combined-with-a-learning-Accessibility-Map/>.

