Lecture 2: Probing the power of probability density functions

Recap

Lecture 1 provided a broad overview of the course and key statistical concepts. This lecture will review core concepts in probability. You might have seen some of these concepts in SIOC 221A, so speak up if things seem too repetitious. But we'll quickly try to delve further into the foundatioal concepts before we move on.

Uncertainties and data We started class with an example of fitted parameters from a published paper. The parameters differed by a factor of 2 or so, and we wondered if the differences were statistically significant. That led to some consideration of uncertainties. Here are some ways that data values can be plotted:

- Data points
- Data points with standard error bars (e.g. $\pm 2\sigma$, where σ is one standard deviation). This works well when data are normally distributed. Consider an array "data" dimensioned with the first dimension as "time" and the second dimension as "realization", with statistics to be computed over the ensemble of realizations. To plot in Matlab:

  ```
  errorbar(time,mean(data,2),std(data,0,2))
  ```

  and in Python:

  ```
  import matplotlib.pyplot as plt
  plt.errorbar(time,data.mean(axis=1),yerr=data.std(axis=1))}
  ```

- Data points with box and whiskers presentation indicating 25th percentile, median, 75th percentile, and extrema. In Matlab:

  ```
  boxplot(data',time)
  ```

  and in Python:

  ```
  plt.boxplot(data)
  plt.xticks(np.arange(len(data),time)
  ```

- Probability density functions (pdfs). PDFs have the advantage of showing the full distributoion of values, but they can be hard to plot for each data point. One strategy is the "violin plot", which you can access in Python (matplotlib.pyplot.violinplot), but which does not (yet) seem to be a Matlab default, though there are user submitted packages. Hard to plot, but show full distribution of values.

  ```
  plt.violinplot(data)
  ```

Cumulative distributions and probability density functions

Last time we talked about cdfs and pdfs, and we'll recap a bit here. The **cumulative distribution function**) D is

$$D_x(r) \;=\; \text{Fraction of occurrences with } \;\; x < r, \tag{1}$$

where the fraction of occurrences is a probability. Its **probability density function** (pdf) F is

$$F_x(r) \;=\; \frac{d}{dr} D_x(r) \;\; \text{so that} \;\; F_x(r)\, dr \;=\; \text{Fraction of occurrences with } \; r < x < r+dr. \tag{2}$$

The probability of having a value fall somewhere between $-\infty$ and $+\infty$ is 100% which means:

$$F_x(r) \;\geq\; 0 \tag{3}$$

$$\int_{-\infty}^{\infty} F_x(r)\, dr \;=\; 1 \tag{4}$$

$$D_x(r) \;=\; \int_{-\infty}^{r} F_x(r)\, ds \tag{5}$$

Since the pdf is continuous, it is the limit of a histogram describing the number of occurrences in each of several "bins" of x, normalized so that the area under the curve is 1.

For a single value, or if our sensor always measures the same value, the pdf is a narrow spike corresponding to a fixed value, that is a delta function: $F_x(r) = \delta(r - A)$ and D would be a Heaviside unit step function with its edge at $r = A$. A handy representation of the probability density function is

$$F_x(r) = \langle \delta(r - x) \rangle. \tag{6}$$

To see why this is so, let N be an effectively infinite number of realizations of x and M be the number of realizations with $x < r$. Then, by definition,

$$D_x(r) \;=\; \frac{M}{N} \;=\; \frac{1}{N} \sum_{n=1}^{N} \int_{-\infty}^{r} dy\; \delta(y - x_n) \;=\; \int_{-\infty}^{r} dy \langle \delta(y - x) \rangle \tag{7}$$

from which (6) follows by differentiation with respect to r. The delta function is introduced into the third term to generate 1 when $x_n > r$ and 0 when it is not. The pdf is the complete description of x (taken alone) because it immediately gives the average of any function, say G, of the random variable. This is easy to see using (6):

$$\langle G(x) \rangle = \langle \int dr\, G(r) \delta(r - x) \rangle = \int dr G(r) \langle \delta(r - x) \rangle = \int dr\, G(r)\, F_x(r) \tag{8}$$

Let's think about empirical pdfs. We can start with the distribution of random data. Suppose we use a standard random number generator in Matlab:

```
% generate a histogram of random numbers
hist(rand(10000,1),25)
```

This will produce a histogram of uniform data, with 25 bins. If we want the area under the curve to be 1, we need to normalize by the total number of points and by bin width, or give Matlab instructions to plot as a pdf rather than a raw histogram:

```
% generate a histogram of random numbers
histogram(rand(10000,1),'Normalization','pdf')
```

Geophysical variables are probably usually closer to Gaussian than uniform in distribution. If we want to look at a uniform distribution we can use "randn" instead of "rand":

```
% generate a histogram of random numbers
histogram(randn(10000,1),'Normalization','pdf')
```

This produces random numbers with a Gaussian distribution. Recall that a Gaussian distribution can be written:

$$F(r) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(r-\mu)^2}{2\sigma^2}\right], \tag{9}$$

where μ is the mean, and σ is the standard deviation. As a refresher, we like Gaussian distributions, because they are easy to calculate, and have well defined properties. We know that 68% of measurements will be within $\pm\sigma$ of the mean, and 95% of measurements will be within $\pm 2\sigma$ of the mean.

Of course, as we noted in class, when we look at physical variables, many are decidedly non-Gaussian, and that will lead us to ask how we can characterize non-Gaussian variables.

Generating random numbers with a known pdf

The fact that random number generators most easily produce uniform or Gaussian distributions is of limited utility if you want to produce simulated data that have statistics resembling real-world data that are non-Gaussian. How can we create random data with a specified distribution?

To think about this, let's start by considering a hypothetical situation. Suppose that I have a sock drawer that contains unpaired socks that are either white, blue, or black. I can label the colors 1, 2, or 3 for convenience. We'll also suppose that my socks are equally distributed between the 3 colors. I might want to run Monte Carlo simulations to figure out the probability of randomly extracting two socks of the same color, or the probability of having a blue sock, or I might have an entirely different scenario in mind.

Regardless, we can define a pdf for sock color. If I all of my socks were white, I'd only have one value, and the pdf would be a delta distribution:

$$F_x(r) = \delta(r-1), \tag{10}$$

where I'm assuming 1 to be the color index for white. You should remember that the delta function δ is defined to be 0 unless $r = 1$, with an area under the curve of 1.

$$\int_{-\infty}^{\infty} F_x(r)\, dr = \int_{-\infty}^{\infty} \delta(r-1)\, dr = 1. \tag{11}$$

The corresponding cumulative distribution would be:

$$D_x(r) = H(r-1), \tag{12}$$

where H is the Heaviside step function and is equal to 0 for $r < 1$ and 1 for $r > 1$.

For 3 sock colors, the pdf can be expanded to be:

$$F_x(r) = \frac{1}{3}\left(\delta(r-1) + \delta(r-2) + \delta(r-3)\right), \tag{13}$$

and the cdf will be a step function:

$$D_x(r) = \frac{1}{3}\left(H(r-1) + H(r-2) + H(r-3)\right), \tag{14}$$

Suppose you want to generate a pdf that randomly generates the number 1, 2, or 3 to represent the sock scenario. You might see that you can do that by using a random number generator to obtain a uniform distribution and then assigning a 1, 2, or 3 depending on the random number.

$$F_x(r) = \begin{cases} 1 \text{ if } r \leq 1/3 \\ 2 \text{ if } 1/3 < r \leq 2/3 \\ 3 \text{ if } 2/3 < r \end{cases} \tag{15}$$

Full disclosure: In modern software, we could also use this using a random integer generator. For example, to generate a vector of 10 values between 1 and 3 in Matlab:

```
randi(3,10,1)
```

or in Python:

```
a=np.zeros([10]).astype(int)
for i in range(10):
    a[i]=np.random.randint(1,4)
```

But how do you set general rules for this that apply when we move to problems that are more complicated than sock drawers?

To think about his formally, consider that regardless of distribution, the probably of having a data value between $-\infty$ and $+\infty$ is 1. We just need to map each sliver of the pdf from one distribution to the other.

Given the pdf of a variable x, we can find the pdf of any other variable which is a function of x, say $y = Q(x)$. The number of realizations with $r < x < r + dr$ is the same as the number with y between $Q(r)$ and $Q(r + dr) = Q(r) + \frac{dQ}{dr}dr$. Therefore

$$F_x(r)\ |dr| = F_y[Q(r)]\ |dQ| \tag{16}$$

When the mapping of X to Y is one-to-one this leads to the relation

$$F_x(r)\ |dr| = \frac{dQ}{dr}\ F_y[Q(r)] \cdot |dr| \tag{17}$$

or alternatively

$$F_y(Q(r)) = F_x(r) \cdot \left|\frac{dQ}{dr}\right|^{-1} \tag{18}$$

The absolute value signs take care of cases where Q decreases as x increases. When Q vs. x is not one-to-one, a form similar to (17) results but must include all contributions of dr that map into the same dQ and vice versa. The signs in (17) can be kept straight since F_x and F_y must both be positive.

While this formalism is useful, the details of the math can make the pdf inversion seem unnecessarily complicated. In practice we can think of converting from one pdf to another using the inverse pdf method. Here's the procedure.

1. Given the desired pdf of your output values, find the corresponding cdf:
$$D_x(r) = \int_{-\infty}^{r} F_y(s)\, ds. \tag{19}$$
2. Find an analytic form of the cdf. (Or if the cdf is entirely empirical, define a means to match the cdf to x for any arbitrary value between 0 and 1.
3. Set a random uniform distribution u equal to the cdf of x.
4. Solve for x in terms of u.
5. Now plug uniformly distributed values u into your equation to obtain x.

Let's consider an example:

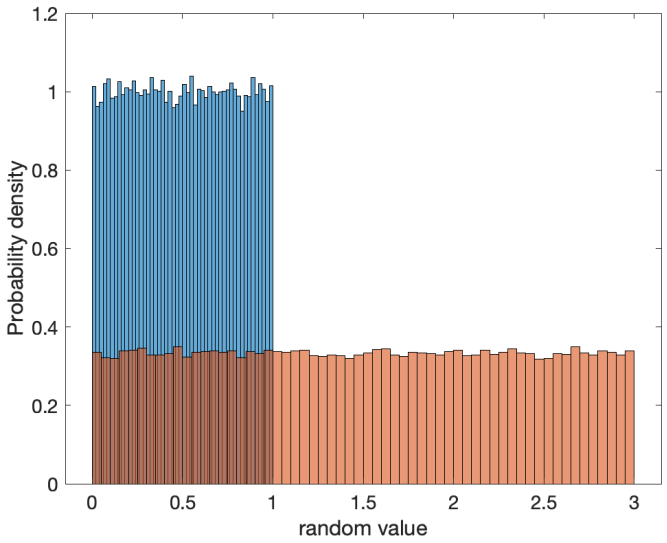
Example 1. *Generate random numbers between 0 and 3.*

To generate random numbers between 0 and 3, we'll first note that our pdf should be
$$F_x(r) = \frac{1}{3} \text{ for } 0 < r \leq 3. \tag{20}$$
The corresponding cdf is
$$D_x = \frac{x}{3} \text{ for } 0 < x \leq 3. \tag{21}$$
We set u equal to the cdf:
$$u = \frac{x}{3} \tag{22}$$
and solve for x:
$$x = 3u. \tag{23}$$
Plugging numbers in, you'll easily see that this will produce a distribution of numbers from 0 to 3.

Here's Matlab code to illustrate this:

```
u=rand(100000,1);
histogram(u,'Normalization','pdf')
hold on
histogram(3*u,'Normalization','pdf')
ylabel('Probability density','FontSize',14)
xlabel('random value','FontSize',14)
h=gca
set(h,'FontSize',14)
```

More to come next time ....


1.2
1
0.8
0.6
0.4
0.2
0
Probability density
0
0.5
1
1.5
2
2.5
3
random value