Lecture 3: Joint probability density functions

Recap

In Lecture 2, we reviewed cumulative distribution functions and probability density functions, and we started looking at how to generate random numbers with a known pdf. This lecture will explore further examples to transform from one pdf to a different pdf, and then examine joint pdfs, covariance and autocovariance.

We finished up with a recipe for converting from one dpdf to another. Here's the procedure.

1. Given the desired pdf of your output values, find the corresponding cdf:

$$D_x(r) = \int_{-\infty}^{r} F_y(s)\, ds. \tag{1}$$

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.

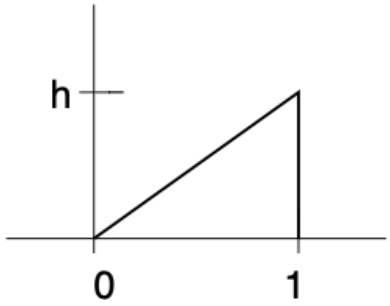
4. Solve for x in terms of u.

5. Now plug uniformly distributed values u into your equation to obtain x.

We looked at one example. Now let's consider a slightly more complicated case.

Example 2. *Generate random numbers that follow a triangle distribution.*

Now we consider a more complicated case, in which randoom numbers follow a triangle distribution of this form:



To create this distribution, first we need to determine the height h of the triangle:

$$F_x(r) = hr \text{ for } 0 < r \leq 1. \tag{2}$$

The integral of this is required to be 1, so we can write.

$$\int_0^1 F_x(r)\, dr = \frac{h}{2} r^2 \Big|_0^1 = \frac{h}{2}, \tag{3}$$

which tells us that $h = 2$. The corresponding cdf is

$$D_x = x^2 \text{ for } 0 < x \leq 1. \tag{4}$$

Again, we set u equal to the cdf:

$$u = x^2 \tag{5}$$

and solve for x:

$$x = \sqrt{u}. \tag{6}$$

We can test whether this worked in Matlab:

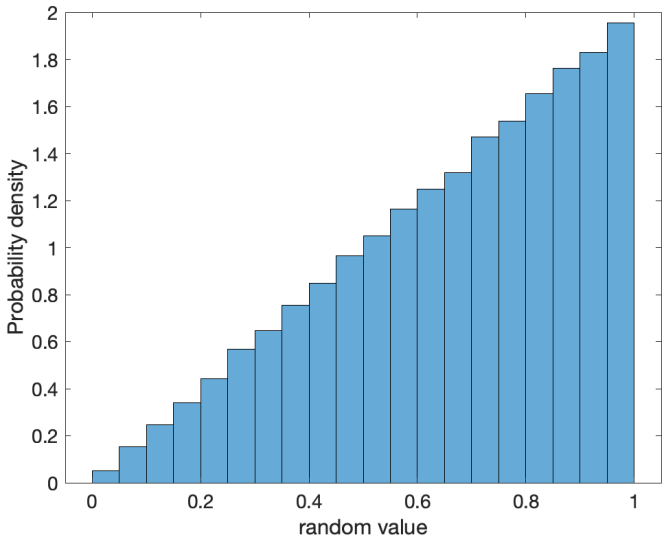
```
% generating random data with a triangle distribution
% pdf(x)=2x from 0 to 1
% cdf(x) = x.^2

y=sqrt(u);
histogram(y,20,'Normalization','pdf');
ylabel('Probability density','FontSize',14)
xlabel('random value','FontSize',14)
h=gca
set(h,'FontSize',14)
```

or in Python

```
N=1000000
u=np.random.random(N)
a=np.sqrt(u)

plt.hist(a,bins=50,density='true')
plt.plot(np.arange(0,1.05,.05),2*np.arange(0,1.05,.05))
```



Example 3. *An arbitrary empirical distribution*

In some cases, you simply want to generate random variables that adhere to a known pdf. For this you won't find an analytic solution, but you will create a mapping from the cdf to the random values (e.g. winds at 55°S). Given a random value between 0 and 1, you can use the empirical cdf (derived from the empirical pdf of the data) as a lookup table to find the corresponding value of your random variable.

Joint probability density functions

Data are not just single values. Most of what we want to know about the ocean involves how one variable is related to another. Examples are how wind stress drives ocean currents, or how vertical fluxes affect primary productivity, or how temperature is linked to salinity. What is the probability of having a high salinity with low temperature? Dynamical equations describe such relationships. If some aspect of the process is random, then we have use for statistics.

A complete description of a pair of random variables x and y is given by the **joint probability density function**:

$$F_{xy}(r,s) \; = \langle \delta(r-x)\delta(s-y) \rangle. \tag{7}$$

The joint pdf has the properties

$$F_{xy}(r,s)\, dr\, ds = \text{ Probability that } r < x \leq r+dr, s < y \leq s+ds. \tag{8}$$

As we noted when we considered the pdf for one variable, the integral of a function multiplied by the pdf gives its mean:

$$\langle g(x,y) \rangle = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} ds\; g(r,s)\; F_{xy}(r,s)\, dr\, ds \tag{9}$$

Since $F_x(r)$ is the pdf of x, without regard for the value of y

$$F_x(r) \; = \; \int_{-\infty}^{\infty} F_{xy}(r,s)\, ds. \tag{10}$$

While we plotted pdfs as line plots in one dimension, joint pdfs make sense as contour plots mapped out in space. Suppose you want to plot a pdf of two independent random variables. We can do that in Matlab with the following code:

```
% Matlab's default, as a 3 dimensional bar plot
histogram2(randn(1000,1),randn(1000,1),'Normalization','pdf')

% or a more conventional 2-d plot
histogram2(randn(1000,1),randn(1000,1),'Normalization','pdf',...
     'Displaystyle','tile')
colorbar
```

and in Python

```
N=1000000
b=np.random.normal(size=[N])
c=np.random.normal(size=[N])

H, xedges, yedges = np.histogram2d(b,c,[30,20],density=True)
plt.pcolormesh(xedges,yedges,H.T)
plt.colorbar()
```

We can plot this for Argo data. In the example in class, I extracted a block of recent Argo profiles from the Argovis website (https://argovis.colorado.edu). That produsces a "json" data file that we can read and plot, as shown in the code below and in Figures and .

```
file='argovis.colorado.edu.json';
fid=fopen(file);
raw=fread(fid,inf); % Reading the contents
str = char(raw'); % Transformation
fclose(fid); % Closing the file
data = jsondecode(str); % Using the jsondecode function
                        % to parse JSON from string

for k=1:length(data)
 if(isfield(data{k}.measurements,'psal'))
  for i=1:length(data{k}.measurements)
   if(~isempty(data{k}.measurements(i).psal))
    psal(i,k)=data{k}.measurements(i).psal;
   else
    psal(i,k)=NaN;
   end
   pres(i,k)=data{k}.measurements(i).pres;
   temp(i,k)=data{k}.measurements(i).temp;
  end
 end
end
xx=find(pres==0 & temp==0 & psal==0);
pres(xx)=NaN; temp(xx)=NaN; psal(xx)=NaN;
xx=find(temp==0 & psal==0);
temp(xx)=NaN; psal(xx)=NaN;

% once we have the data we could plot a conventional T-S diagram
plot(psal,temp,'.')
h1=gca
set(h1,'FontSize',14)
xlabel('salinity','FontSize',14)
ylabel('temperature (^oC)','FontSize',14)

% or we could plot all the points as a joint pdf
histogram2(psal,temp,'Normalization','pdf','Displaystyle','tile');
h1=gca
set(h1,'FontSize',14)
h2=colorbar
h2.Label.String = 'probability density';
set(h2,'FontSize',14)
xlabel('salinity','FontSize',14)
ylabel('temperature (^oC)','FontSize',14)
```

or

```
# courtesy of Donata Giglio, University of Colorado Boulder
from argovisHelpers import helpers as avh
```

```
import datetime, pandas, matplotlib, scipy, numpy
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm

API_ROOT='https://argovis-api.colorado.edu/'
API_KEY=''

# get TS profiles in a region and time period of interest

pac_region = [[-155,-1.5],[-155,-30],[-145,-30],[-145,-1.5],[-155,-1.5]]
argo = {
    'startDate': '2022-12-30T00:00:00Z',
    'endDate': '2023-01-13T00:00:00Z',
    'polygon': pac_region,
    'data': 'temperature,1,salinity,1'
} # querying only profiles that have both temperature and salinity

floats = avh.query('argo', options=argo, apikey=API_KEY, apiroot=API_ROOT)

# let's select only levels that have a valid T and S
def delete_if_none(list1, list2, list3):
    result1 = []
    result2 = []
    result3 = []

    for i in range(len(list1)):
        if list1[i] is not None and list2[i] is not None and list3[i] is not None:
            result1.append(list1[i])
            result2.append(list2[i])
            result3.append(list3[i])

    return result1, result2, result3

def find_variable_index(profile, variable):
    return profile['data_info'][0].index(variable)


filtered_profiles = []
for f in floats:
    filtered_pressure, filtered_temperature, filtered_salinity = delete_if_none(f['
    filtered_profiles.append([filtered_pressure, filtered_temperature, filtered_sal

# only makes sense for profiles that have levels spanning the region of interest

shallow = 10
deep = 1800

# select only profiles that have measurements in the range of interest
profiles = [f for f in filtered_profiles if f[0][0] < shallow and f[0][-1] > deep]
```

```
# dump all the pressure, temperature, and salinity values
# (a more graceful approach would interpolate to a common grid)
pres=[]
psal=[]
temp=[]
for i in range(len(profiles)):
    pres=np.append(pres,list(profiles[i][0]))
    temp=np.append(temp,list(profiles[i][1]))
    psal=np.append(psal,list(profiles[i][2]))

# make a scatter plot
plt.scatter(psal,temp,s=1,c=-pres)
plt.ylabel('Temperature, degC')
plt.xlabel('Salinity, psu')
plt.colorbar()

# make a joint pdf
plt.hist2d(psal,temp,bins=[20,20],density=True,cmin=1.e-100)
plt.ylabel('Temperature, degC')
plt.xlabel('Salinity, psu')
cbar = plt.colorbar()
cbar.set_label('probability density')
```

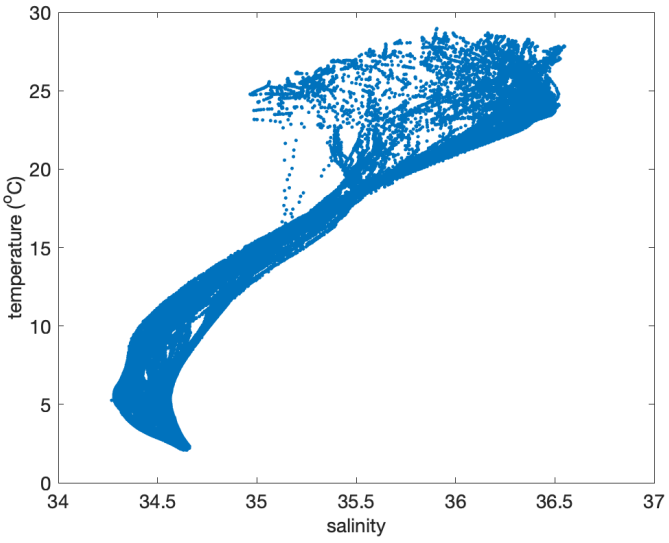


Figure 1: T-S diagram for temperature vs salinity for all Argo profiles in the tropical Pacific from roughly 0 – 30°S and 150 – 130°W, from 30 December 2022 through 13 January 2023.

Conditional probability density function

The **conditional probability density function** is defined as follows:

$$F_x(r\,|s) = \text{probability that } \; r < x \leq r + dr \text{ given that } y = s. \tag{11}$$

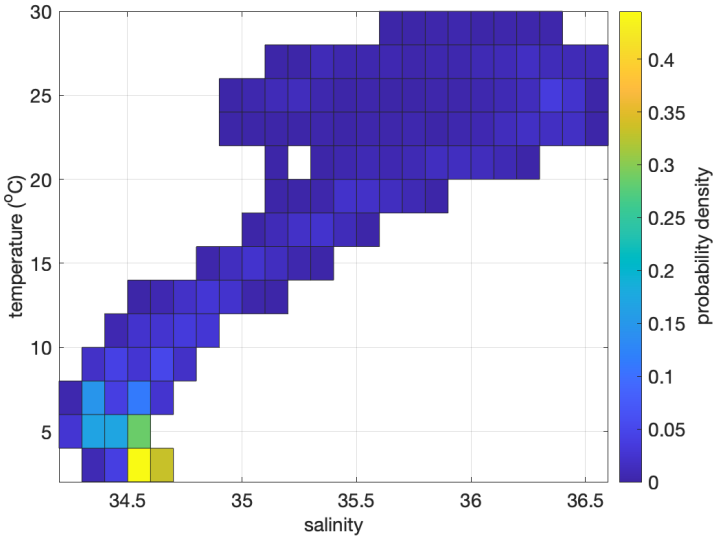


Figure 2: Temperature–salinity data from Figure presented as a joint pdf.

We'd like to be able to write the conditional pdf in terms of the joint pdf. The following may be short of a rigorous mathematical proof, but should help to explain the idea. Suppose we have N realizations of random variables x and y. The number of realizations in a region $r < x \leq r + dr, s < y \leq s + ds$ is:

$$NF_{xy}(r, s)\, dr\, ds. \tag{12}$$

For example, this would be the number of realizations in a bin of Figure 1. The number of realizations in $s < y \leq s + ds$ is

$$NF_y(s)\, ds. \tag{13}$$

For example, this would be the number of realization in a horizontal strip of bins. So the fraction in $r < x \leq r\, dr$ given that $y = s$ is

$$F_x(r\,|s)\, dr = \frac{NF_{xy}(r, s)\, dr\, ds}{NF_y(s)\, ds} \tag{14}$$

Thus the conditional pdf is

$$F_x(r\,|s) = \frac{F_{xy}(r, s)}{F_y(s)} \tag{15}$$

Bayes' Theorem

The formal definition for conditional probability can be written for r in terms of s, or for s in terms of y. We have

$$F_x(r\,|s) = \frac{F_{xy}(r, s)}{F_y(s)} \tag{16}$$

and also

$$F_y(s\,|r) = \frac{F_{xy}(r, s)}{F_x(r)} \tag{17}$$

We can combine these in a number of ways:

$$F_x(r\,|s) = \frac{F_y(s|r)F_x(r)}{F_y(s)} \tag{18}$$

Equivalently:

$$F_x(r\,|s) = \frac{F_y(s|r)F_x(r)}{\int_{-\infty}^{\infty} F_{xy}(r,s)\,dr} = \frac{F_y(s|r)F_x(r)}{\int_{-\infty}^{\infty} F_y(s|r)F_x(r)\,dr} \tag{19}$$

This expression is called Bayes' Theorem and provides a formal framework for considering the probability of an event given prior knowledge.

If the random variables x and y are independent, then $F_x(r|s)$ is independent of s, which implies from (16) that

$$F_{xy}(r,s) = F_x(r)F_y(s). \tag{20}$$

General form of joint Gaussian pdf

To place the formal definitions in context, consider a joint pdf for independent Gaussian variables:

$$F_{xy}(r,s) = \frac{1}{\sqrt{2\pi}\sigma_x}\exp\left(\frac{-r^2}{2\sigma_x^2}\right)\frac{1}{\sqrt{2\pi}\sigma_y}\exp\left(\frac{-s^2}{2\sigma_y^2}\right) \tag{21}$$

$$= \frac{1}{2\pi\sigma_x\sigma_y}\exp\left[\frac{-1}{2}\left(\frac{r^2}{\sigma_x^2}+\frac{s^2}{\sigma_y^2}\right)\right]. \tag{22}$$

If x and y are uncorrelated, the joint pdf is either isotropic (if $\sigma_x = \sigma_y$) or has no tilt.

We can write the joint Gaussian distribution in a general form for a collection of variations $x_1, x_2, ...x_N$, with $\sigma_1 = \sigma_2 = 1$:

$$F_{x_1x_2....}(r_1, r_2...) = (2\pi)^{-N/2}\exp\left[-\frac{1}{2}\sum_{i=1}^{N} r_i^2\right] \tag{23}$$

Of course things change if we have two correlated variables, and in class we looked at the joint pdf that emerges from correlated noise, for example when x is drawn from a Gaussian distribution and $y = x + r$, where r is noise drawn from a Gaussian distribution. We also looked at the correlation of y and z when $z = x + s$, where s is different from r and also drawn from a Gaussian distribution. Both cases result in a tilted joint pdf, providing clear evidence that x and y (or x and z) are correlated.

```
%  define correlated noise
x=randn(100000,1); y=randn(100000,1)+x;
z=randn(100000,1)+x;

% plot joint pdf for x and y
histogram2(x,y,'Normalization','pdf','Displaystyle','tile')

% plot joint pdf for y and z
histogram2(y,z,'Normalization','pdf','Displaystyle','tile')
```

or

```
N=1000000
x=np.random.normal(size=[N])
y=np.random.normal(size=[N])+x
```

```
z=np.random.normal(size=[N])+x

plt.figure(figsize = (10,4))
plt.subplot(1,2,1)
H, xedges, yedges = np.histogram2d(x,y,[30,30],density=True)
plt.pcolormesh(xedges,yedges,H.T)
plt.colorbar()

plt.subplot(1,2,2)
H, xedges, yedges = np.histogram2d(y,z,[30,30],density=True)
plt.pcolormesh(xedges,yedges,H.T)
plt.colorbar()
```

Covariance

Calculating the joint pdf is often more than we can accomplish from real data. The **covariance** is a simple statistic relating variables x and y:

$$C_{xy} = \langle x'y' \rangle, \tag{24}$$

where the primes indicate that these are fluctuations about the mean. The covariance of a variable with itself is the **variance**:

$$C_{yy} = \langle y'y' \rangle. \tag{25}$$

The **correlation** is sort of a normalized covariance:

$$\rho_{xy} = \frac{\langle x'y' \rangle}{\sqrt{\langle x'^2 \rangle \langle y'^2 \rangle}}. \tag{26}$$

How can we interpret the correlation? Let's consider a linear model, where y is a linear function of x. In the following, we assume that variables x and y have zero means, or equivalently that they have had their means removed, so the primes are dropped. A linear relationship between modeled $\hat{y}$ and measured x is

$$\hat{y}' = \alpha x', \tag{27}$$

where α is a constant chosen to make $\hat{y}$ approximate y.

We could also write this in a more general form as a matrix equation to fit lots of coefficients α_j to multiple form of data. In general form, we would write

$$y_i = \sum_{j=1}^{N} A_{ij}\alpha_j, \tag{28}$$

where the elements of A_{ij} represent the jth element of data type i. As a matrix equation we would write where the elements of A_{ij} represent the jth element of data type i. As a matrix equation we would write

$$\mathbf{y} = \mathbf{A}\alpha, \tag{29}$$

where $\mathbf{y}$ is a vector with M elements, α is a vector with N elements, and $\mathbf{A}$ is an $M \times N$ matrix. We'll come back to this case later.

Let's continue with the one variable fit that we're considering now. We choose to minimize the mean-square error (mse):

$$\epsilon = \langle (\hat{y} - y)^2 \rangle = \alpha^2 \langle x^2 \rangle - 2\alpha \langle xy \rangle + \langle y^2 \rangle. \tag{30}$$

Correlation and regression

The best α in the sense that the mse is minimized is found by differentiating with respect to α, setting the result equal to zero, and solving for α. Because $\epsilon \to \infty$ as $\alpha \to \pm\infty$, the result is a minimum.

$$\frac{\partial \epsilon}{\partial \alpha} = 2\alpha \langle x^2 \rangle - 2\langle xy \rangle = 0. \tag{31}$$

Thus:

$$\alpha = \frac{\langle xy \rangle}{\langle x^2 \rangle} \tag{32}$$

The term α is a regression coefficient, and it assumes a fully linear relationship between x and y.

If we plug α into the equation for the mse, we can find the misfit

$$\begin{aligned} \epsilon &= \alpha^2 \langle x^2 \rangle - 2\alpha \langle xy \rangle + \langle y^2 \rangle & (33) \\ &= \frac{\langle xy \rangle^2}{\langle x^2 \rangle} - 2\frac{\langle xy \rangle^2}{\langle x^2 \rangle} + \langle y^2 \rangle & (34) \\ &= \langle y^2 \rangle \left(1 - \frac{\langle xy \rangle^2}{\langle x^2 \rangle \langle y^2 \rangle} \right) & (35) \\ &= \langle y^2 \rangle \left(1 - \rho_{xy}^2 \right) & (36) \end{aligned}$$

Thus the mean-squared error (the mse) is related to the variance of the quantity that we were trying to fit ($\langle y^2 \rangle$) multipled by 1 minus the correlation coefficient squared.

To minimize ϵ k

Sidebar: Moments of the pdf

Here's a quick refresher on the moments of the pdf.

The pdf contains more information than can usually be determined for real processes. Consequently, practical analysis often involves simpler statistical measures. The simplest is, of course, the mean x. Others are concerned with variations about the mean and are most conveniently defined in terms of the **fluctuation**

$$x' = x - \langle x \rangle. \tag{37}$$

A prime on a random variable generally denotes a fluctuation. The **variance** μ and the **standard deviation** σ,

$$\mu_2 = \langle X'^2 \rangle, \qquad \sigma = \mu_2^{1/2}, \tag{38}$$

describe, respectively, the "energy" of the fluctuations and a typical fluctuation size. The variance should not be confused with the mean square x^2. Beyond the mean and variance we might compute any number of higher **moments**

$$\mu_n = \langle x'^n \rangle. \tag{39}$$

As we will later see, as n increases it becomes progressively harder to obtain an accurate estimate of μ_n from a finite set of data. Thus a few lower moments are all that can typically be determined.

As we noted before, given a known pdf of x, we can compute the mean of a function $G(x)$, by integrating the product of $G(x)$ times the pdf. This is useful for estimating moments of the pdf: The first moment of the pdf, the mean, is:

$$\langle x \rangle = \int_{-\infty}^{\infty} r F_x(r)\, dr. \tag{40}$$

The second moment, the variance, is

$$\mu_2 = \langle x'^2 \rangle = \int_{-\infty}^{\infty} (r - \langle x \rangle)^2 F_x(r)\, dr. \tag{41}$$

The third and fourth moments are not terribly interesting by themselves:

$$\mu_3 = \langle x'^3 \rangle = \int_{-\infty}^{\infty} (r - \langle x \rangle)^3 F_x(r)\, dr \tag{42}$$

$$\mu_4 = \langle x'^4 \rangle = \int_{-\infty}^{\infty} (r - \langle x \rangle)^4 F_x(r)\, dr. \tag{43}$$

However, in normalized form, these tell us about the shape of the pdf.

The lopsidedness of the pdf is measured by the skewness:

$$\text{skewness} = \frac{\mu_3}{\mu_2^{3/2}}. \tag{44}$$

The sign of the skewness indicates whether the tails of the pdf are more pronounced on the positive or negative side of the mean.

The "tailedness" of the pdf, that is the relative occurrence of outliers in the distribution, is measured by kurtosis:

$$\text{kurtosis} = \frac{\mu_4}{\mu_2^2}. \tag{45}$$

For a Gaussian distribution, kurtosis = 3, and for a uniform distribution, kurtosis = 1.8.