Lecture 4:

Recap

Key concepts from last time focused on the standard error of the mean and its links to error propagation, the chi-squared and Rayleigh distributions, and methods for telling whether one pdf is statistically like another (the Komogorov-Smirnov test and the chi-squared test).

You'll recall that the Rayleigh distribution represents the square root of two independent Gaussian components, $y = \sqrt{x_1^2 + x_2^2}$.

$$p(y) = \frac{y}{\sigma^2} \exp\left[-\frac{y^2}{2\sigma^2}\right]. \tag{1}$$

And the χ^2 distribution represents the sum of n squared variables:

$$\chi_n^2 = z_1^2 + z_2^2 + z_3^2 + ... + z_n^2. \tag{2}$$

The slides show examples of these.

Are two pdfs different?

So now let's return to the heart of our problem. How do we tell if two pdfs differ? We've already noted that two data sets can look wildly different but still have the same mean and variance, so clearly we need something more than just the mean and variance. We can go back to our Gaussian overlaid on empirical pdf and eyeball the difference to say that they're close enough, or not plausibly similar. We can evaluate whether the mean and standard deviation differ. All of this is good, but it doesn't exploit the full range of information in the pdf. We need a metric to measure how different two distributions are.

Here are a couple of strategies. One notion is to ask about the largest separation between 2 pdfs. We compute two cdfs—in this case one empirical and one theoretical, but we can also do this with two empirical cdfs. We find the maximum separation between the distributions, the Komogorov-Smirnov statistic:

$$D_n = \sup_n |C_n(x) - C(x)| \tag{3}$$

and then we can predict the probability that a data set with n elements should differ from the ideal distribution by D_n. Matlab has a "kstest" function (or "kstest2") that sorts through the parameters for this. However, we have to be careful with this, because usually our data are correlated, and we don't have as many degrees of freedom as we think. The easiest solution is to decimate the data set so that the number of elements reflects the number of degrees of freedom.

A second strategy is to bin the data and ask whether the number of data in the bin is consistent with what we'd expect, using a χ^2 statistics. In this case for comparisons with a theoretical pdf,

$$\chi^2 = \sum_i \frac{(N_i - n_i)^2}{n_i}, \tag{4}$$

where N_i is the observed number of events in bin i, and n_i is the theoretical or expected number of events in bin i. For comparisons between two distributions,

$$\chi^2 = \sum_i \frac{(N_i - M_i)^2}{N_i + M_i}, \tag{5}$$

where N_i and M_i are each observed numbers of events for bin i. The values of χ^2 are evaluated using the χ^2 probability function $Q(\chi^2|\nu)$, which is an incomplete gamma function, where ν is the number of bins (or the number of bins minus one, depending on normalization). In Matlab this is

```
gammainc(chi_squared/2,nu/2)
```

Fitting a function to data: least-squares fitting

Now, we've laid a lot of ground work. Let's think about our time series. If we look at SST records, for example, how can we determine whether temperatures are increasing or decreasing over time. Let's suppose we're looking for a linear trend. Then

$$T = T_o + bt, \tag{6}$$

where T represents our measured temperature data, T_o is a constant (unknown), t is time, and b is the time rate of change. We have lots of observations, so we really should represent this using vectors (which we'll indicate with bold face):

$$\mathbf{T} = T_o + b\mathbf{t}. \tag{7}$$

We'll want to find the best estimates of the scalars T_o and b to match our data. Formally, provided that we have more than two measurements, this is an over-determined system. Of course, we're talking about real data, so we should acknowledge that we have noise, and our equations won't be perfect fits. We could write:

$$\mathbf{T} = T_o + b\mathbf{t} + \mathbf{n}, \tag{8}$$

where $\mathbf{n}$ represents noise and is unknown. Now the system is formally underdetermined. But we won't lose hope. We'll just move forward under the assumption that the noise is small.

Let's write this in a more general form as a matrix equation:

$$\mathbf{Ax} + \mathbf{n} = \mathbf{y}, \tag{9}$$

where

$$\mathbf{A} = \begin{bmatrix} 1 & t_1 \\ 1 & t_2 \\ 1 & t_3 \\ \vdots & \vdots \\ 1 & t_N \end{bmatrix}, \tag{10}$$

making $\mathbf{A}$ an $N \times 2$ matrix. And $\mathbf{y}$ is an N-element column vector containing, for example, our temperature data:

$$\mathbf{y} = \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ \vdots \\ T_N \end{bmatrix}. \tag{11}$$

Then $\mathbf{x}$ is the vector of unknown coefficients, in this case with 2 elements (e.g. $x_1 = T_o$ and $x_2 = b$).

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{12}$$

How can we find the best solution to this equation to minimize the misfit between the data $\mathbf{y}$ and the model $\mathbf{Ax}$? The misfit could be positive or negative, and absolute values aren't mathematically tractable, so let's start by squaring the misfit.

$$\epsilon = (\mathbf{Ax} - \mathbf{y})^T(\mathbf{Ax} - \mathbf{y}) = \mathbf{x}^T\mathbf{A}^T\mathbf{Ax} - 2\mathbf{x}^T\mathbf{A}^T\mathbf{y} + \mathbf{y}^T\mathbf{y}. \tag{13}$$

Then we can minimize the squared misfit. The natural route to minimization comes by taking the derivative, and then setting the results equal to zero. Our unknown is $\mathbf{x}$, so we'll minimize in terms of that:

$$\frac{\partial \epsilon}{\partial \mathbf{x}} = 2\mathbf{A}^T\mathbf{A}\mathbf{x} - 2\mathbf{A}^T\mathbf{y} = 0, \tag{14}$$

and this implies that

$$\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{y}, \tag{15}$$

so

$$\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{y}. \tag{16}$$

In Matlab, you'd code this as follows:

```
% assuming time matches the SST standard with days counting from
% January 1, 1800
% and data is the monthly sea surface temperature
plot(time+datenum(1800,1,1),data,'LineWidth',2); hold on
A=[ones(length(time),1) time(:)];
x=inv(A'*A)*A'*data;
plot(time+datenum(1800,1,1),A*x,'r','LineWidth',2)
```

As we've noted, if we want to find a trend, we define $\mathbf{A}$ to have a column of ones (to identify the mean) and a column containing the time, to identify the rate of change. We can make our model $\mathbf{A}$ progressively more complicated by adding additional columns. What do we do if we want to find the annual cycle? Before I give you any answers, take a moment to think about this.

We could use:

$$\mathbf{A} = \begin{bmatrix} 1 & \cos(t_r) & \sin(t_r) \\ \vdots & \vdots & \vdots \end{bmatrix}, \tag{17}$$

where time t is measured in days, and $t_r = 2\pi t/365.25$, is the time in radians. We need the sine and cosine because we don't know the phase of our annual cycle exactly. You might imagine that we could fit for the phase (e.g. $\sin(t_r + \phi)$), and we could, but that would be a non-linear fitting process, and the power of least-squares fitting won't work if we try that—we'd quickly be plunged into the murky world of non-linear fitting procedures, which is messy, unreliable, and not necessary in this case.

Here's an example

```
% continuing from previous example....
A2=[ones(length(time),1) time(:) sin(2*pi*time/365.25) ...
    cos(2*pi*time/365.25)];
x2=inv(A2'*A2)*A2'*data;
plot(time+datenum(1800,1,1),A2*x2,'m','LineWidth',2)
```

Orthogonality and Least-Squares Fits

Let's think about one important detail of our fitting procedure. What would happen if we wanted to fit $T = x_1 + x_2 t_r + x_3 t_r$, that is to fit two constants to the same variable? In this case, clearly x_2 and x_3 are completely indistinguishable. What happens when we try to use our redundant functions in our matrix $\mathbf{A}$?

$$\mathbf{A} = \begin{bmatrix} 1 & t_1 & t_1 \\ 1 & t_2 & t_2 \\ \vdots & \vdots & \vdots \\ 1 & t_N & t_N \end{bmatrix}. \tag{18}$$

Then we find:

$$\mathbf{A}^T\mathbf{A} = \begin{bmatrix} N & \sum t_i & \sum t_i \\ \sum t_i & \sum t_i^2 & \sum t_i^2 \\ \sum t_i & \sum t_i^2 & \sum t_i^2 \end{bmatrix}. \tag{19}$$

The second and third rows of $\mathbf{A}^T\mathbf{A}$ are identical, which tells us that the third row is adding no additional information to the system. As a result, the matrix $\mathbf{A}^T\mathbf{A}$ is singular, and we won't be able to find an inverse for it. (In Matlab, when you try to do the inversion, you'll see a message, "Warning: Matrix is singular to working precision.")

You probably wouldn't try to fit coefficients to two identical functions, but you might do something that was fairly similar. For example, $T = x_2 t_r + x_3 \sin(t_r)$ poses a similar problem when t_r is near zero. In this case, the rows of $\mathbf{A}^T\mathbf{A}$ might not be identical, but they might be nearly the same so that Matlab would give you an error message.

Similarly, you'll have trouble if you try: $T = x_1 + x_2 t_r + x_3(1 + t_r)$.

None of this was an issue when we used sine and cosine, because they are orthogonal, so they contain no redundant information.

Building complexity: multiple oscillatory signals

Now let's think about the pressure record from the Scripps pier, because we know that had a lot of sinusoidal variability. What might we include in that fit? We might hypothesize that data collected on the pier could be influenced by an annual cycle (1 cycle/365.25 days), a diurnal cycle (1 cycle/24 hours), and a tidal cycle (1 cycle/0.5175 days = 1 cycle/(12 hours + 25.2 minutes)). How would we fit for all of these components?

You might wonder if you'll bias your results by fitting for all of the sinusoidal variability all at once. Usually, the answer is no. Assuming that your time series is long enough, sinusoidal frequencies are orthogonal to each other: there is no correlation between $\sin(2\pi t/T)$ and $\sin(2\pi 2t/T)$, just as there is no correlation between the sine and cosine components.

So we can take this to the maximum limit. Suppose we just want to fit sines and cosines to our data. How many frequencies can we fit? If I'm going to do this, then I'll need to make sure that each row of my matrix equation is linearly independent, which means that I'll want to make sure that each column of $\mathbf{A}$ is orthogonal, so I can't choose frequencies that are too closely spaced.

Least-Squares Fitting Sines and Cosines

Least-squares fitting is particularly tidy when the functions that we use for our fit, the columns of our matrix $\mathbf{A}$, are completely orthogonal, because then the fit to one function has no impact on the fit to the other functions.

Consider the special case where the columns of $\mathbf{A}$ are made up of sines and cosines, so

$$A = \begin{bmatrix} 1 & \cos(\omega t) & \sin(\omega t) & \cos(2\omega t) & \sin(2\omega t) & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \end{bmatrix}, \tag{20}$$

where $\omega = 2\pi/T$ and T is the total duration of the data record. The dot product of any two columns i and j of $\mathbf{A}$ is zero if $i \neq j$. If I have data at N evenly spaced time increments, $t_1, t_2, ...t_N$, then this orthogonality property holds for all frequencies from ω through $N\omega/2$. Since I have a sine and cosine at each frequency (up to frequency $N\omega/2$ where sine might be zero at all points in time), this means that I can define a total of N independent orthogonal columns in $\mathbf{A}$.

On the other hand, if I define a column of $\mathbf{A}$ to have a frequency $\omega/2$, it won't be orthogonal to my other functions over the range of this data. For example, between 0 and T, $\sin(\omega/2)$ varies from 0 to 1 to 0 and is always positive, meaning that it will be positively correlated with a constant. In fact, sines and cosines with frequencies that are ω multiplied by integers rangings from 0 to $N/2$ make a complete set that spans all space, and there are no additional N-element vectors that I can add to $\mathbf{A}$ that would also be orthogonal to all other columns of $\mathbf{A}$.

The orthogonality of the columns of $\mathbf{A}$ is really important. It means that my solution for x_1 is completely independent of my solution for x_2. Here are some results for a set of 128 random numbers, b.

$$\hat{b} = -0.0629 - 0.0620\cos(\omega t) - 0.1339\sin(\omega t) \tag{21}$$

$$\hat{b} = -0.0629 - 0.0960\cos(2\omega t) + 0.1117\sin(2\omega t) \tag{22}$$

$$\hat{b} = -0.0629 - 0.0620\cos(\omega t) - 0.1339\sin(\omega t) - 0.0960\cos(2\omega t) + 0.1117\sin(2\omega t), \tag{23}$$

where $\hat{b}$ is our fitted approximation to b. You can see that the amplitudes of $\cos(\omega t)$ and $\cos(2\omega t)$ are the same regardless of whether $\mathbf{A}$ contains 3 columns or 5 columns.

If we take a time series of N elements, then the lowest frequency that we can resolve is 1 cycle per N elements, so $\cos(2\pi i/N)$, where our counter i runs from 1 to N (or from 0 to $N-1$). We can find two coefficients for this: one for the $\cos$ component and one for the $\sin$ component.

Actually, maybe a better way to think about this is that the lowest frequency we can resolve is $\cos(0i/N) = 1$, which is a constant and represents the mean. Since $\sin(0) = 0$, there is only a cosine component for frequency 0.

At any rate, after considering 1 cycle per N points, the next frequency we can resolve that will actually be fully orthogonal is 2 cycles per N points. We can keep counter upward: 3 cycles per N points, 4 cycles per N points, and so forth. All of these are guaranteed to be orthogonal over our domain of N points.

What is the maximum number of cycles that we can resolve in N points? One possibility would be that the maximum is N cycles per N points. That would require a full sinusoidal oscillation squeezed between data element 1 and data element 2. But if you think about it, we wouldn't expect to have enough information to determine the amplitude of a sine wave that had to squeeze itself between consecutive observations. Moreover if N cycles per N points were the maximum, this would mean that we'd be solving for $2N$ coefficients with only N data points. Clearly that would require more information than we have available.

So the maximum must be less than N. There are two ways to think about this. One way is to think about the information content of our data. At the most basic level, if I have N elements in my data matrix, then at best, I can hope to have N independent equations, so I can fit N coefficients as an exactly determined problem. So I can solve for frequencies from 0 cycles per N points to $N/2$ cycles per N points, with 2 coefficients for all but the end members of my record. The other way to think of this is to observe that you'll need at least 2 data points per cycle to determine even a minimum amount of information about the sine or cosine amplitudes of your data. So the highest frequency that you can possibly detect of $N/2$ cycles per N data points. This is the *Nyquist frequency* and it is one of the most central concepts in time series analysis.

And the strategy of least-squares fitting all possible frequencies that can be resolved represents the *discrete Fourier transform*. It's a slow and inefficient Fourier transform, but it is the essence of this class and it will be the building block for everything we do in the remainder of the quarter.