What is time series analysis?
What cycles are contained within the SPECMAP record?
Properties of a cyclical or oscillating system.

Amplitude: the peak value in either the positive or negative direction.

Period: The time occupied in one complete cycle (time domain).

Wavelength: The length occupied in one complete cycle (spatial domain).

Frequency: Number of cycles in unit time (or length in the spacial domain).
**Phase**

The fraction of a whole cycle that has elapsed from a predefined fixed point. Measured as an angle, where a full cycle is $2\pi$. 

![Sine wave diagram with phase markers](image)
Phase

The fraction of a whole cycle that has elapsed from a predefined fixed point. Measured as an angle, where a full cycle is $2\pi$. 
Representation of a sine wave

The disk rotates with a constant angular velocity

The paper moves with a constant velocity.

The pencil moves up and down, drawing a sine wave
We can calculated and plot a sine wave of any given frequency

In Matlab we can make a series of equally space points in time

Variable name --> Start point --> Spacing --> End point --> Transpose

\text{time}=[0:1:800]';

After typing this command and pressing enter you will see that Matlab lists the results of the vector on the screen. Normally we don‘t need to see the result on the screen so we can use the symbol ; for example:

\text{time}=[0:1:800]';
Next we need to calculate a sine wave with a frequency, $f$:

$$signal = \sin(2\pi \times time \times f)$$
Work with $f = 0.01$

*(in MATLAB you can obtain the constant $\pi$ using pi)*

**TYPE:**

\[
f = 0.01 \\
\text{signal} = \sin(2.\pi\cdot\text{time}\cdot f); \\
\text{plot(time,signal)}
\]

It is important to use $\cdot\cdot$ for multiplication rather than simply $\cdot$. In Matlab using only $\cdot$ gives a *matrix* multiplication. We want an *array* multiplication so we must use $\cdot\cdot$. 
What is the amplitude of this signal?

How can we modify the calculation to change the amplitude of a sine wave?
We used:

time = [0:1:800];
f = 0.01;
signal = \sin(2.*\pi.*time.*f);
plot(time,signal)

What should we change if the amplitude is 2.5?
We simply multiply the sine by a constant (in this case 2.5):

time = [0:1:800];
f = 0.01;
signal = sin(2.*pi.*time.*f).*2.5;
plot(time,signal)

So to calculate the sine wave with a given amplitude $A$:

signal = sin(2.*pi.*time.*f).*A;
A sine wave with an amplitude of 2.5
Phase

The fraction of a whole cycle that has elapsed from a predefined fixed point. Measured as an angle, where a full cycle is $2\pi$.

How can we modify the calculation to change the phase of a sine wave?
When the phase was equal to 0 we used:

time = [0:1:800];
f = 0.01;
signal = \sin(2.\pi \cdot \text{time} \cdot f);
plot(time, signal)

What should we change if the phase is $\pi/2$?
We simply add a constant (in this case \(\pi/2\)) to the calculation:

\[
(\text{remember in Matlab you can obtain the constant } \pi \text{ using } \text{pi})
\]

\[
time = [0:1:800];
f = 0.01;
\]

\[
signal = \sin(2.*\text{pi}.*time.*f+(\text{pi.}/2));
\]

\[
\text{plot}(time,signal)
\]

So to calculate the sine wave with a given phase \(\theta\):

\[
signal = \sin(2.*\text{pi}.*time.*f+\text{theta});
\]
A sine wave with a phase-shift of $\pi/2$
What are the frequencies of the 3 orbital cycles (units kyr\(^{-1}\))? 

\[
\text{Eccentricity} = \frac{1}{100} = 0.01 \text{ kyr}^{-1}
\]

\[
\text{Obliquity} = \frac{1}{41} = 0.0244 \text{ kyr}^{-1}
\]

\[
\text{Precession} = \frac{1}{21} = 0.0476 \text{ kyr}^{-1}
\]

Calculate sine waves with the frequencies of the orbital components, add them together and plot the final signal.
> time=[0:1:800]';
> eccen=sin(2.*pi.*time.*(1./100));
> obliq=sin(2.*pi.*time.*(1./41));
> prec=sin(2.*pi.*time.*(1./21));
> final=eccen+obliq+prec;
> plot(time,final)
Now let's consider the opposite situation (*in fact the typical geoscience situation*). Is it possible to calculate the frequency, phase and amplitude of the individual sine waves given only the composite signal?
The Fourier Transform
Jean-Baptiste Joseph Fourier (1768-1830)

The Fourier transform: technique based on decomposing signals into sinusoids.

It is possible to express any single-valued periodic function as a summation of sinusoidal components, of frequencies that are multiples of the frequency of the function.

In normal language, this means that any signal can be split into a collection of sine waves with different frequencies, amplitudes and phases. The Fourier transform gives us this information!
First, let's return to our representation of a sine wave as a rotating disk, remember:

*The disk rotates with a constant angular velocity*

*The paper moves with a constant velocity.*

*The pencil moves up and down, drawing a sine wave*

One complete rotation of the disk will draw a sine wave between 0 and $2\pi$, this pattern is repeated in time between $-\pi$ and $+\pi$
We can represent a cycle using complex numbers

A complex number consists of both a real part and an imaginary part ($i = \sqrt{-1}$).

Representation of a complex number $z = x + iy$ in the complex plane

One full rotation through the complex plane will represent a cycle between 0 and $2\pi$ (just the same as our spinning disc).
The Fourier transform returns a series of complex numbers. Each complex number gives the characteristics (frequency, amplitude, phase) of a sine wave that is included in the total signal.

The complex coefficients, $C_n$, have real and imaginary parts $a_n$ and $ib_n$. For a single term (i.e. a given frequency) of the Fourier sum we can obtain the signal amplitude:

$$|C_n| = \sqrt{a_n^2 + b_n^2}$$

And phase:

$$\cos \phi = \frac{a_n}{\sqrt{a_n^2 + b_n^2}} \quad \sin \phi = \frac{b_n}{\sqrt{a_n^2 + b_n^2}}$$
We will use an Matlab function called \texttt{fft\_plot} to perform a Fourier transform on a signal and display the amplitudes of the sine waves.

First we need to construct the signal which will be analysed:

\begin{verbatim}
>> time = [0:1:200]';
>> f=1./20;
>> data = sin(2.*pi.*time.*f)
>> plot(time,data)
\end{verbatim}

As you can see this signal is made from a single sine wave with a frequency of 0.05 (that is 1/20).
Now we can perform the analysis using `fft_plot`:

```matlab
>> fft_plot(time,data);
```

We have a single peak with an amplitude greater than zero. Looking on the x-axis we find this point has a frequency of 0.05.

The Fourier transform therefore shows us that our input signal contained a single sine wave with a frequency of 0.05.

Normally we work in the "time domain" where we have time as the x-axis of our data. When we work with the Fourier transform we have frequency as the x-axis of the data and we are said to work in the "frequency domain".
The aim of this exercise is for you to investigate how a signal made from a mixture of sine waves is represented by the Fourier transform (in other words how it is represented in the *Frequency Domain*).

Using Matlab calculate 3 different sine waves (i.e. with different amplitudes and frequencies) and add them together to give a composite signal.

1. **Produce a plot which shows your composite signal**

Now, use `fft_plot` to perform and plot the Fourier transform of the composite signal.

2. **Produce a plot which shows the amplitude spectrum of your composite signal.**

So, the next part is the important bit;

3. **Provide an interpretation of the amplitude spectrum that you obtained from your composite signal. Think about the positions of the peaks in the diagram and their relative heights.**

**Exercise 1.2**
Using Matlab calculate 3 different sine waves (i.e. with different amplitudes and frequencies) and add them together to give a composite signal.

1. Produce a plot which shows your composite signal

First I define a time array:

```matlab
>> time=[0:1:800]';
```

Then calculate my 1\textsuperscript{st} sine wave (called \textit{eccen}) with $f=0.01$, phase=$0$ and amplitude=$2.5$

```matlab
>> eccen=sin(2.*pi.*time.*0.01).*2.5;
```

Then calculate my 2\textsuperscript{nd} sine wave (called \textit{obliq}) with $f=0.024$, phase=$\pi/2$ and amplitude=$1$

```matlab
>> obliq=sin(2.*pi.*time.*0.024+(pi./2)).*1.0;
```

Then calculate my 3\textsuperscript{rd} sine wave (called \textit{prec}) with $f=0.048$, phase=$3\pi/2$ and amplitude=$0.5$

```matlab
>> prec=sin(2.*pi.*time.*0.048+(3.*pi./2)).*0.5;
```

Adding together my 3 sine waves will give my final composite signal.

```matlab
>> final=eccen+obliq+prec;
```
Now, use `fft_plot` to perform and plot the Fourier transform of the composite signal.

2. Produce a plot which shows the amplitude spectrum of your composite signal.

```matlab
>> fft_plot(time, final)
```

```matlab
eccen = sin(2.*pi.*time.*0.01).*2.5;
```

```matlab
obliq = sin(2.*pi.*time.*0.024+(pi/2)).*1.0;
```

```matlab
prec = sin(2.*pi.*time.*0.048+(3.*pi/2)).*0.5;
```

What effect does the phase have?
Time series analysis in the Time Domain
Mauna Loa, Hawaii.
Variations in carbon dioxide content in the atmosphere with respect to time.
How can we make a simple estimate of the period of the main cycle in CO$_2$ content?
Event Spacing.

Simply measure the peak-peak (or trough-trough) period between major episode in a time series.

Mean peak spacing = 364 +/- 17 days
Mean trough spacing = 364 +/- 7.5 days
Mean event spacing = 364 +/- 12.5 days
This time series shows the monthly amount of runoff water (measured in inches) from Cave Creek in Kentucky.

Use the event spacing method to estimate the period of each runoff cycle.
In MATLAB we first have to load the data into memory.

To find which variables are in the memory you type:

```matlab
>> load cave_creek
>> whos
```

At the bottom of the screen you will see some text which looks like:

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Bytes</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>month</td>
<td>216x1</td>
<td>1728</td>
<td>double array</td>
</tr>
<tr>
<td>runoff</td>
<td>216x1</td>
<td>1728</td>
<td>double array</td>
</tr>
</tbody>
</table>

Grand total is 432 elements using 3456 bytes

This shows us that the variables `month` and `runoff` have been loaded into the memory, they both contain 216 data values in a column.
We can now make a plot of the data:

```matlab
>> plot(month,runoff)
>> xlabel('month') this adds a label to the x-axis
>> ylabel('runoff (inches) ') this adds a label to the y-axis
```

If you look at the plot window you should now have something which looks like:
We will use the function `ginput` to mark the points on the chart which we think are events:

```matlab
>> [x,y]=ginput
```

You can now use the mouse click to record the positions that you think correspond to events in the runoff data. The variables `x` and `y` now correspond to the coordinates of the positions you clicked on. When you have finished clicking on the events press the `enter` key.

You can mark these points on you chart in the following way:

```matlab
>> hold on
>> plot(x,y,'o')
```

So now we have the timing of the events but how can we find the event spacing?
We just need to find the difference between the values in \( x \), in Matlab this is simple:

\[
>> x\_diff=\text{diff}(x)
\]

Then to find the mean and standard deviation of the event spacing

\[
>> x\_mean=\text{mean}(x\_diff);
\]
\[
>> x\_std=\text{std}(x\_diff);
\]

What is your interpretation of the mean event spacing that you calculated?
Using the minima of the data to mark an "event" (filled circles)

Event spacing = 11.9 ± 1.2 months
**Autocorrelation**

Correlate a data set with itself at a number of different offsets (defined as lags).

The correlation at any given number of lags is called the autocorrelation coefficient, $r_{\tau}$

The autocorrelatogram shows the extent of the correlation at different lags and can be used to explore the cyclicity of the data.

---

**Important**

Data points must be regularly spaced with respect to time.

Linear Trends should be removed from the data set.

There should be at least 50 data points and the lag should no be greater than $n/4$. 
What are “lags”? 

The number of data points by which the two data series are offset!

To demonstrate the idea of lagging data we will attempt to find the period of a sine wave using autocorrelation.

We start with our sine wave and make a copy of it, then we see how closely they are correlated.
Correlation of signals = -0.80012
Sunspots

Areas of “cool” gas linked with local variations in the Sun’s magnetic field.
Sunspot numbers

Sunspot number

Year

1700 1750 1800 1850 1900 1950 2000
Define a linear trend through the data (using least-squares regression)
To detrend the data we subtract the trendline from the original data.
Here we have the original data and an exact copy of the data. They are in perfect alignment, this corresponds to a lag of zero.
Correlation for lags = 0

$r_\tau = 1.00$
Lags=55
Correlation for lags = 55

$r_{\tau} = 0.0449$
Lags=110

110 datapoints
Correlation for lags = 110

\[ r_\tau = 0.4441 \]
Autocorrelogram for the detrended sunspot data

Peak – Peak Correlation

Peak – Trough Correlation
Autocorrelogram for the detrended sunspot data

Hale cycle 11 years
Lets look again at the runoff water data from Cave Creek.

Now we can use autocorrelation to estimate the period of each runoff cycle.
In Matlab we first have to remove any data which might still be in the memory using the *clear* command and then load the Cave Creek data:

```matlab
>> clear all
>> load cave_creek
```

Our first task is to detrend the data. To do this we can use the function `detrend_signal.m` which you downloaded the course web page.

```matlab
>> runoff_d=detrend_signal(month,runoff,1);
```

This will produce the detrended data `runoff_d`, which has had a straight-line trend removed. A figure is produced by the function that shows the original data, the fitted line and the detrended data.

The data is now prepared for the autocorrelation. To do the analysis we use the function `autocorr.m`

```matlab
>> [rt,lags]=autocorr(runoff_d); note this function has two outputs
>> figure make a new figure window
>> plot(lags,rt) our final autocorrelogram
>> xlabel('Lag number')
>> ylabel('r')
```

**Exercise 2.2**
Autocorrelogram for the detrended Cave Creek data
A statistical test to find if the peaks are significant.

Given the autocorrelation coefficient $r$ at lag $t$, where the number of observations is $n$:

$$Z = r\sqrt{n - \tau + 3}$$

A peak in the autocorrelogram is considered to be statistically significant at the 5% level if $Z$ exceeds 1.96.

You can use the function `autocorr_sig.m` to produce an autocorrelogram with the 5% significance level marked on the figure. Only peaks above the 5% line can be considered as statistically reliable;

```matlab
>>[rt,lags] = autocorr_sig(runoff_d);
```
These peaks can be considered as statistically reliable at the 5% level.
Load the SPECMAP file into MATLAB, you will find it contains two variables; age and data (in this case age is in ka and data is normalised oxygen isotope data).

1. Use Matlab to produce a plot of the Specmap record. Add appropriate labels to both the x and the y axes.

2. Detrend the Specmap signal

3. Perform an autocorrelation including significance levels on the detrended series.

4. Make an interpretation of the autocorrelation in terms of known climate variation.

Exercise 2.3
>> load specmap
>> plot(age, data)
>> xlabel('Age (ka)')
>> ylabel('Normalised d18O')
\[ xc = \text{detrend\_signal}(\text{age}, \text{data}, 1) \]
\[ [rt, lags] = \text{autocorr\_sig}(xc); \]
Signal filtering
How can we remove the long-term increase in the CO₂ content, but keep the main cycle?
We can fit a trendline to the time series and extract it from the data:

$$ xc = \text{detrend\_signal}(\text{time}, \text{mauna}, 1) $$

Linear

$$ y = a_1 x + a_0 $$
We can fit a trendline to the time series and extract it from the data.

```matlab
>> xc = detrend_signal(time, mauna, 2)
```

**Quadratic**

\[ y = a_2 x^2 + a_1 x + a_0 \]
We can fit a trendline to the time series and extract it from the data.

\[ y = a_3 x^3 + a_2 x^2 + a_1 x + a_0 \]

\[ \text{Cubic} \]

\[ \text{Original Signal} \]

\[ \text{Detrended Signal} \]
Simple removal of long-term trends – First difference.

The difference between neighbouring points, use: \( d_i = x_{i+1} - x_i \)
Simple removal of long-term trends – First difference.

This is simple to do in Matlab using the `diff` function, you can get information on functions by typing, for example:

```matlab
>> help diff
```

The CO₂ data is in the file `mauna`

```matlab
>> load mauna
>> xd=diff(mauna);
```

This first line simply takes the difference of the CO₂ data, but what does the next line do?

```matlab
>> xt=(diff(time)./2)+time(1:229);
```

```matlab
>> plot(xt,xd)
```

It calculates the time position on which the `xd` data points should be placed, by finding the mid-point between the neighbouring time points.

Exercise 3.2
Removing noise from a signal
The *moving average* or *running mean*

**3-point moving average**

1st average result

Average data points 1, 2 and 3
The **moving average** or **running mean**

3-point moving average

Average data points 2, 3 and 4

2nd average result

Signal

Time

0 2 4 6 8 10
The moving average or running mean

3-point moving average

Signal

Time
The *moving average* or *running mean*
The *weighted* moving average

In the traditional moving average all the points have the same weight (i.e. all the points have the same importance). In some cases it can be useful to make the points have different weights (i.e. different levels of importance).

An example of a weighted 5 point moving average. For each calculation we make the central points more important than the outside points.

If we have 5 points: A, B, C, D, E

Traditional moving average: \((A+B+C+D+E)/5\)

\([1,3,5,3,1]\) weighted moving average:
\[((A*1)+(B*3)+(C*5)+(D*3)+(E*1))/(1+3+5+3+1)\]

\([1,2,2,2,1]\) weighted moving average:
\[((A*1)+(B*2)+(C*2)+(D*2)+(E*1))/(1+2+2+2+1)\]
Normalised porosity data from GeoB4311-02 (Equatorial Atlantic)
Normalised porosity data from GeoB4311-02
5pt moving average
Removing noise from a signal – Frequency domain

![Graph showing time series with high and low noise](image-url)
The first step of the filter is to transfer the data into the frequency domain using the Fourier transform.
Only the frequencies that are of interest are selected (in this case 0.02-0.03 kyr\(^{-1}\)) all the other magnitudes are reduced to zero so they no longer contribute to the spectrum.

How can we take the signal from the frequency domain back into the time domain?
We use the *Inverse Fourier Transform* (IFFT) to convert our modified signal back into the time domain. The new signal only contains the parts of the Specmap record with frequencies in the range 0.02-0.03 kyr\(^{-1}\).
Different types of filter in the frequency-domain

Low-pass filter
removes frequencies above a specified value

High-pass filter
removes frequencies below a specified value

Band-pass filter
removes frequencies outside specified limits
**Low-pass filter:** removes frequencies above a specified value (high frequencies are blocked, low frequencies pass)

Cutoff frequency = $\frac{1}{40}$
**High-pass filter:** removes frequencies below a specified value (low frequencies are blocked, high frequencies pass)

Cutoff frequency = 1/20
**Band-pass filter:** removes frequencies between 2 specified values

Cutoff frequency range = 1/200 to 1/40
 Filtering climate records

Often there are a number of different cycles and sources of noise in climate records. Filtering allows us to remove the noise and study the individual cycles.
Porosity data filtered with a low-pass filter, cutoff $f=0.1$
Filtering in the frequency domain

Use the function `filter_signal` to filter the normalised porosity from core GeoB4311.

Use the subplot command to make plots which compare the filtered porosity record to the SPECMAP stack. The normalised oxygen isotope data for the SPECMAP stack can be found in the MATLAB file `specmap`. 
What effect did we observe with the movie of the spinning bicycle wheel? Why is this effect important in time series analysis?
Consider a 2 m long sediment core which contains a climate cycle with a wavelength of 0.1 m. When we sample the core on the ship we don’t know that this cycle is recorded, at what points should we take samples for measurement?

I choose to take 1 sample every 0.11 m and measure the signal in that sample.
I measure each sample and make a climate curve by connecting the data points.

Here the real signal is *aliased* and I have artificially produced a signal with a wavelength of $> 1$ m
Considering a sinusoid it can be see that the signal is aliased when the sampling interval is greater than 0.5 divided by the signal's frequency.
Considering a sinusoid it can be seen that the signal is aliased when the sampling interval is greater than 0.5 divided by the signals frequency.

To define an oscillation we therefore need at least 2 samples per cycle. This means that the highest frequency information contained in a time series is determined by our sampling interval, with the maximum frequency given by:

\[
\begin{align*}
f_N &= \frac{1}{2\Delta t} \\
\end{align*}
\]

Where \(\Delta t\) is the time interval between samples.

\(f_N\) is called the Nyquist frequency and it tells us the maximum frequency we can obtain from a Fourier Transform.

The Specmap stack has one data point every 2 kyr

\[ \Delta t = 2.0 \text{ kyr} \]

\[ f_N = \frac{1}{2\Delta t} = 0.25 \text{ kyr}^{-1} \]
The resolution of a frequency spectrum depends on the number of data points in the original signal and their separation.

If \( N \) is the number of data points in the time series and \( f_N \) is the Nyquist frequency, then the spacing between the points in the frequency domain is given by:

\[
\Delta f = \frac{2f_N}{N + 1}
\]

For SPECMAP, \( N=392 \), \( f_N = 0.25 \),

\[
\Delta f = \frac{2f_N}{N + 1} = \frac{2 \times 0.25}{392 + 1} = 0.0013
\]
Data variance in the frequency domain

For a simple sinusoid we find that the variance of the signal (the square of the standard deviation) increases with the square of the amplitude divided by 2.

\[
\text{Variance} = \frac{\text{Amplitude}^2}{2}
\]

Variance can be analysed as a function of frequency using a Power Spectrum.
In previous examples we have looked at the absolute magnitude of the Fourier terms. **Parseval’s theorem** shows that if we use the squared magnitude of the Fourier term multiplied by $\Delta f$ they we will get the signal variance at each frequency.

This relationship will only hold for a time series with zero mean (more on this later). Normally the variance information is calculated directly and is shown as a periodogram of variance verses frequency.
Variance $data 1 = 10.125$, Variance $data 2 = 24.5$
Variance $data1 = 10.125$, Variance $data2 = 24.5$
Time information in the Frequency domain

Now we will consider 2 signals that are opposite to each other in the time domain and see how they are different in the frequency domain.

Exercise 4.2
There is no time information in the frequency domain. Therefore flipping the signal will have no effect on the frequency spectrum.
The effect of long-term trends in the Frequency domain

Here again is the CO$_2$ series from Mauna Loa. We used event spacing and autocorrelation to reveal a strong yearly cycle in the data, but what will the periodogram look like?
The periodogram is dominated by the long-term trend in the data. How can we adjust the data to remove this effect?
We can fit a trendline to the time series and extract it from the data:

$$\gg xc = \text{detrend\_signal}(\text{time}, \text{mauna}, 1)$$

Then calculate the periodogram for the detrended data.

**Linear**

$$y = a_1 x + a_0$$

Exercise 4.3b
The effect of the long-term trend has been almost completely removed from the periodogram. It is always important to consider if your data should be detrended.
The effect of data point spacing in the Frequency domain

A fundamental assumption of the Fourier transform is that the data are equally spaced in the time domain.

It is extremely rare however, that we have equally spaced data points in geoscience time series. How can we overcome this problem?

The simplest method is to interpolate the data on to an equally spaced time axis.
We can see that interpolation of the data on to a regularly spaced time axis attenuates the high frequency variation.
To overcome this problem we can use spectral analysis methods that are specifically designed to work with data that is unevenly spaced in the time domain.

Examples of such methods are the Lomb-Scargle technique and the CLEAN algorithm.
White noise can be simulated by drawing random numbers from a normal (Gaussian) distribution.

What will the frequency spectrum of a white noise signal look like?

Signal of random numbers from a normal distribution with mean = 0 and variance = 1

The theoretical spectrum is flat because there should be equal variance at all frequencies.

Exercise 4.6
We find that even as we increase the number of data points in the signal the frequency spectrum doesn't become flat.
To overcome this kind of problem we can use a method called *Welch-overlapped-segment-averaging*. This approach processes the time series in the following way.

- The signal is split into a given number of overlapping segments of equal length.
- The frequency spectrum of each segment is determined.
- All the spectra are averaged to produce the final smoothed frequency spectrum.

This is the method employed in the SPECTRUM software, which is designed to process unequally spaced palaeoclimatic data.

The record is split into 3 overlapping (50%) segments that will be processed individually.
A Hanning window is applied to each segment to smooth the ends of the data.
A frequency spectrum is calculated for each segment, giving a mean spectrum.
The whole process of splitting the signal into overlapping segments, applying the Hanning window and calculation of the mean spectrum can be performed in MATLAB using the function `pwelch`.

```
[Pxx,w] = pwelch(x,window,[],[],Fs)
```

Use the `pwelch` function to calculate consistent autospectra for a 1024 point white noise signal with mean = 0 and variance = 1.

Exercise 4.7
As the number of segments is increased the spectrum becomes closer to the expected theoretical form, but the reduced number of data points in each segment leads to a loss in resolution.
Red noise

The simplest statistical model for a discrete finite red noise series is the first-order autoregressive AR(1) process:

\[ r_n = \rho r_{n-1} + \omega_n \]

\[ n = 1, 2, 3, \ldots, N \]  
Discrete time increments

\[ 0 < \rho < 1 \]  
Lag-one autocorrelation coefficient

\[ \omega_n \]  
Normally distributed random number

For the case of white noise all the data points are independent. For red noise each data point has some "memory" of the points which came before it. What would happen for the case \( \rho = 0 \) ?

\[ \rho = 0 \rightarrow \text{White noise} \]

What do different red noise series look like in the time and frequency domains?

Exercise 4.8
\( \rho = 0.00 \)
\[ \rho = 0.80 \]
\( \rho = 0.99 \)

![Graph showing signal and power over time and frequency](image-url)
For red noise series the spectral density increases as the frequency decreases. Why would we expect to see red noise in nature (e.g. palaeoclimate data)?

It is reasonable to assume that climate at any given time will have been effected by the climate which proceeded it.
The Time-Frequency Plane

• Stationary and nonstationary signals

• Evolutive Spectral Analysis

Werner Heisenberg
(1901 - 1976)
Stationary Signals

A signal is stationary if the frequency, amplitude and phase content do not change with time.
A stationary signal

```matlab
>> time=[0:1:3199]';

>> f1=1./400;
>> f2=1./100;
>> f3=1./41;
>> f4=1./23;

>> signal1=sin(2.*pi.*f1.*time);
>> signal2=sin(2.*pi.*f2.*time);
>> signal3=sin(2.*pi.*f3.*time);
>> signal4=sin(2.*pi.*f4.*time);

>> signal=signal1+signal2+signal3+signal4;
```
A stationary signal
A stationary signal

Exercise 5.1
A nonstationary signal

```matlab
>> time1 = [0:1:799];
>> time2 = [800:1:1599];
>> time3 = [1600:1:2399];
>> time4 = [2400:1:3199];

>> f1 = 1./400;
>> f2 = 1./100;
>> f3 = 1./41;
>> f4 = 1./23

>> signal1 = sin(2*pi*f1*time1);
>> signal2 = sin(2*pi*f2*time2);
>> signal3 = sin(2*pi*f3*time3);
>> signal4 = sin(2*pi*f4*time4);

>> time = [time1; time2; time3; time4];
>> signal = [signal1; signal2; signal3; signal4];
```
A nonstationary signal
A nonstationary signal

The spectra contains a number of extra peaks. These are due to the effects of the nonstationary behaviour.

The spectrum gives us no information about the changing frequency of the signal, instead it is just an average over the whole time period.
Now work with the reverse signal. You can do this easily using the `flipud` function. We are now looking at a signal with a very different history

```python
>>plot(time, flipud(signal))
```
Therefore, even if we reverse the signal the frequency spectrum remains the same. As before the spectrum gives us no information about the changing frequency of the signal, instead it is just an average over the whole time period.

Therefore, even if we reverse the signal the frequency spectrum remains the same.

The frequency spectrum contains no time information.
Most geoscience signals are nonstationary because their frequency content changes with time. How can we analyse these signals and find out when the frequency content changes?

ODP677 Benthic Oxygen Isotope Record
We split the signal into segments that we think contain pseudo-stationary behaviour and we perform spectral analysis on each window.
In this case, our window is 500 kyr long and for each step we move it 100 kyr.
For each segment we perform spectral analysis, giving the behaviour in the frequency domain for a short part of the record.
This procedure is called the **Short-term Fourier transform**, or **Evolutive spectral analysis**. We can use the function `evolpsd` to perform evolutive spectral analysis on our nonstationary signal.

```
evolpsd(time,signal,window,spacing)
```

window = 600, spacing = 50
The time-frequency plane for our non-stationary signal

Each frequency spectrum is placed at a time point corresponding to the middle of the segment.

Exercise 5.3
The time-frequency plane is normally plotted in two-dimensions as a contour map.
Good resolution in time, poor resolution in frequency
500 kyr Window

reduced resolution in time, increased resolution in frequency
800 kyr Window

poor resolution in time, good resolution in frequency
Very poor resolution in time, very good resolution in frequency
Perfect resolution in time, perfect resolution in frequency
Perform a time-frequency investigation of this signal. What features appear when you use the `evolpsd` function?

The signal data is contained in the file `chirp`, you can perform the analysis using the following commands and changing the window size and spacing.

```plaintext
>> load chirp
>> evolpsd(time,signal,200,10);
>> set(gca,'ylim',[0 0.1])
```
Spectrum for the whole record contains no time information. Using a very long window gives poor resolution in time, but what if we use a very short window to improve the time resolution?
The Nyquist frequency tells us the maximum frequency of the Fourier Transform that contains useful information.

\[ f_N = \frac{1}{2\Delta t} \]

\( \Delta f \) gives the resolution in the frequency spectrum. Therefore when the number of points is small we have very poor resolution in the frequency information.
Heisenburg’s Uncertainty Principle

From this picture, we know the exact position of the horse at the time the photograph was taken.

However, this one photograph gives us no information about how quickly the horse is running.
Heisenburg’s Uncertainty Principle

Consider an atom travelling through space.

We cannot determine both the instantaneous speed and position of the atom.

This also applies to our analysis because we cannot determine both the frequency and time of a signal.
Problems with the STFT

Time and frequency localisation varies with chosen window length.

Long window: good frequency localisation, poor time localisation.
Short window: poor frequency localisation, good time localisation.

In the window shown 6 full cycles of the red signal will be analysed by the Fourier transform, but only 3 cycles of the blue signal will be included in the same analysis.

Within a window the STFT is inconsistent in the treatment of different frequency components.
STFT transform in the time-frequency plane

- More
- High
- Cycles
- Frequency
- Less
- Low
- Time
- All spectral components are resolved equally
Make a time-frequency investigation of the ODP677 record, what can you say concerning changes in the climate cycles through time?
STFT: ODP677 Data with 500 kyr window

Input signal

Evolutionary spectral analysis
The long-term cooling trend in the ODP677 data dominates the STFT and must be removed if we want to study the Milankovitch cycles.
STFT: Detrended ODP677 Data with 500 kyr window
STFT: Detrended ODP677 Data with 800 kyr window

Input signal

Evolutionary spectral analysis

Prec
Obliq
Eccen
STFT transform in the time-frequency plane

All spectral components are resolved equally
Adaptation in the time-frequency plane?

The area of the boxes is controlled by the Uncertainty Principle and cannot be reduced. But, if we can reshape the boxes we can make the analysis less biased.

For this scheme, low frequency cycles will be analysed with low resolution in time, but high resolution in frequency.

In addition, the high frequency components will be analysed with low resolution in frequency and high resolution in time.
We start with a sine wave

Infinite span (uniform in time)

Calculate a Gaussian envelope

A Morlet wavelet is produced by multiplying the sine wave and the envelope together.

Finite span (localized in time)
An important property of wavelets is that they can be stretched and compressed. This is termed *scaling* and allows different frequencies to be represented.

Wavelets are scaled for different frequency components, but the same number of cycles is always sampled.
For any given wavelet scale the similarity of the frequency content can be compared to a given time series at any position.

- **Good Similarity**
- **Intermediate Similarity**
- **Poor Similarity**
Because the wavelet can be scaled, we can change the scaling to find the best fit for each of the different parts of the time series.

Poor Similarity

Good Similarity

A longer period wavelet fits the low frequency parts of the time series better, but shows a poor similarity with the high frequency parts.
Wavelets can also provide a measure of the amplitude of a given frequency component of the time series.

High wavelet power will be seen when the wavelet and time series have similar frequencies and the time series has a high amplitude.
**Steps of the Wavelet Transform**

The wavelet transform gives information on the amplitude of periodic signals and the variation of amplitude with time. Therefore, the WT is 2-dimensional and decomposes a signal into time and frequency space simultaneously.

A wavelet of scale, $s$, is placed on the time series at $t = 0$ and their similarity is assessed. The wavelet is then moved on to $t = 1$ and the similarity is assessed again. This process is repeated for all values of $t$.

The scale is then changed and the wavelet is again migrated through the time series. This procedure is repeated for all desired values of $s$.

A Time-Scale (Frequency or period)-Power matrix is formed.
No time information in the frequency domain

Window contains a large number of high frequency cycles but only a small number of low frequency cycles

Window adapts to obtain the optimum time-frequency resolution within the limits of the uncertainty principle.
There are different types of wavelet

Morlet

Mexican Hat

Also known as the DOG (derivative of Gaussian)

Haar

The type of wavelet used to analyse a time series depends on the shape of the expected cycles. The Morlet wavelet is generally used the most and has been shown to be applicable to many geophysical signals.
Wavelet transform of a sine wave.

Notice the edge effects (cone of influence) where the wavelet “falls off the end of the data”
>> load chirp
>> input=[time(:,), signal(:)]  
there are two variables; time and signal
>> wt(input)

Arrays as columns and placed into one matrix.
A nonstationary signal
Theoretical calculation of orbital parameters since 3 Ma.
The combination of these curves gives the ETP curves (eccentricity+tilt-precession)
```matlab
>> load odp677
>> input=[age(:),data(:)]
>> wt(input)
```

there are two variables; *time* and *signal*

Arrays as columns and placed into one matrix.
Perform a wavelet analysis on the sunspot number data (stored in the file: sunspots). Make an interpretation of the wavelet transform in terms of both signal frequency and amplitude.
Principal Component Analysis

Or

Empirical Orthogonal Functions
It is possible to plot projections of >3 dimensional space but it is difficult to interpret.

**Line**

**Square**

**Cube**

**Hyper-Cube**

Can we reduce the number of dimensions required to describe a system?
Consider this $(x, y)$ data set, how many dimensions do we need to fully describe the variability of the data?
Consider this \((x,y)\) data set, how many dimensions do we need to fully describe the variability of the data?

A perfect correlation exists between \(x\) and \(y\). Therefore all the points lie on a single line and we can describe their variation in 1 dimension, i.e. their position on the line.
The idea of Principal Components Analysis (PCA)

• Often in datasets with many variables, a number of the variables will show the same variation because they are controlled by the same process.

• Therefore in multivariate datasets we often have data redundancy. We can take advantage of this redundancy by replacing groups of correlated variables with new uncorrelated variables (the principal components).

• PCA generates a new set of variables based on a linear combination of the original parameters. All the principal components are orthogonal (at right-angles) to each other so there is no redundant information.

• There are as many principal components as original variables, however it is common for the first few principal components to account for a large proportion of the total variance of the original data.
The 1st principal component is a line through the data which has a maximum variance, the 2nd principal component is a line at 90° to the first which has the next highest variance and so on.

Because the principal components are orthogonal, no correlation exists between them. The origin of the components in the multivariate mean of the data.

The principal components are formed by rotating the data axes and shifting the mean.
In this case with three-dimensional data we can use the first 2 principal components to simplify the data into a two-dimensional system and still explain ~92\% of the data variance.

The number of principal components calculated is the same as the number of dimensions of the data set. In many cases a large amount of the variance can be explained by the first 2 or 3 principal components and we can produce a plot which is more easily interpreted.
Using eigenvectors & eigenvalues

The eigenvalues of the matrix correspond to the length of the axes of the ellipse.
The eigenvectors of the matrix correspond to the principal axes of the ellipse.
Data normalisation

In cases where the numbers of one variable are much large than the other variables they can dominate the analysis, for example, in an analysis of humans we could have:

*Height, foot length, finger length and umbilicus radius*

In this case is clear that the absolute values of height are much larger than the other variables therefore they will dominate the analysis.

To avoid this effect the data are *standardized* before the analysis, this means setting the mean of each variable to 0 and the standard deviation to 1.
Oreodont Skull measurements

Braincase width, Cheek tooth length, Bulla length & Bulla depth
The Oreodont measurements are a 4-dimensional data set.
The correlation matrix of Oreodont measurements shows a strong relationship between the different size parameters.
We can calculate the principal components of the 4D data cloud. Plotting the first 2 principal components provides a representation of >95% of the variance.
To understand what each principal component represents, we look at the factor-loadings which tell us what effect each variable has on each principal component:

<table>
<thead>
<tr>
<th>Variable</th>
<th>PC1</th>
<th>PC2</th>
<th>PC3</th>
<th>PC4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Braincase width</td>
<td>0.4971</td>
<td>-0.4879</td>
<td>0.7048</td>
<td>-0.1349</td>
</tr>
<tr>
<td>Cheek tooth length</td>
<td>0.5013</td>
<td>-0.4681</td>
<td>-0.5983</td>
<td>0.4143</td>
</tr>
<tr>
<td>Bulla length</td>
<td>0.5186</td>
<td>0.2901</td>
<td>-0.3073</td>
<td>-0.7433</td>
</tr>
<tr>
<td>Bulla depth</td>
<td>0.4824</td>
<td>0.6773</td>
<td>0.2258</td>
<td>0.5076</td>
</tr>
</tbody>
</table>

All 4 variables have a similar influence on PC1, this suggests PC1 may represent changes in skull size.

For PC2, the size of the Bulla controls the component in a different way to the size of the teeth and braincase. This suggests PC2 is related to the shape of the skull.
The data shows us variation in size (PC1) and shape (PC2). Where the data clusters it suggests that maybe we have skulls from different species of Oreodont.
Now consider the case where the oreodont skulls have been found in a dated stratigraphic sequence. This means the scores can be plotted as a function of (time or depth). In this way we can represent how the data varies along the axis of a given principal component through time.
Analysis of the pattern of sea surface temperature through time.

HadSST2 is a global field of SST on a 5° latitude by 5° longitude grid from 1850 AD.
When considering the SST each location is associated with a time series (1 value per year).

HadSST2 is a global field of SST on a 5° latitude by 5° longitude grid from 1850 AD.
SST record at 29°N, 325°E
Form a two-dimensional data matrix. Each column represents the time series for a given location and each row represents a point in time.
We perform PCA on the data matrix and obtain the loadings and scores for the principal components.
Scores of the first principal component

The loadings give the spatial pattern of the 1st principal component. The scores give the variability of the 1st principal component through time.
Of course, the procedure is not limited to spatial data. The same method could be applied to any multidimensional data set which varies through time.

```
<table>
<thead>
<tr>
<th>Braincase width</th>
<th>Cheek tooth length</th>
<th>Bulla length</th>
<th>Bulla depth</th>
</tr>
</thead>
</table>

Data matrix $X$

$t_1 \downarrow \quad t_2 \downarrow \quad t_3 \downarrow \quad t_n \downarrow$
```
PCA Example: 3 climate time series
First plot the time series

The data are stored in the file `pca_climate.mat`

There are 4 variables, `age` (a common time scale for the data), `ms` (magnetic susceptibility), `gs` (grain size) and `d18O` (oxygen isotopes).

```matlab
>> clear all, close all
Clear the memory and close all figures
>> load pca_climate
Load the data file

>> figure
Create a new figure
>> subplot(3,1,1)
axes in 3 rows & 1 column, activate first set of axes
>> plot(age,zscore(ms))
plot the standardized magnetic susceptibility data
>> ylabel('Normalized Susceptibility')
label the y-axis

>> subplot(3,1,2)
axes in 3 rows & 1 column, activate second set of axes
>> plot(age,zscore(gs))
plot the standardized grain size data
>> ylabel('Normalized Grain size')
label the y-axis

>> subplot(3,1,3)
axes in 3 rows & 1 column, activate second set of axes
>> plot(age,zscore(d18O))
plot the standardized oxygen isotope data
>> ylabel('Normalized d18O')
label the y-axis
>> xlabel('Age [ka]')
label the x-axis
```
Perform the PCA

We’ll first combine the 3 time series into a single matrix and then standardize the column. PCA is performed using the function `princomp`. With the results we can find out what proportion of the variance the 1st PC explains and plot the scores as a function of age.

```matlab
>> X=[ms,gs,d18O]; form a matrix composed of the time series
>> X=zscore(X); standardize the columns of X
>> [loadings,scores,latent]=princomp(X); perform the PCA

>> latent=latent./sum(latent) normalize the PC contributions
>> latent(1) the variance explained by the 1st PC

>> figure
   Generate a new figure
>> plot(age,scores(:,1)) plot the scores of the 1st PC
>> ylabel('PC Score') label the y-axis
>> xlabel('Age [ka]') label the x-axis
```

Using the example above plot the scores of the 2nd PC as a function of age. What proportion of the total variance does the 2nd PC account for?

Exercise 7.1
First Principal Component

PC Score vs Age [ka]
The Hockey Stick curve

Data from thermometers (red) and from tree rings, corals, ice cores and historical records (blue).
Testing the Hockey Stick:
Red noise time series

To start we’ll create a collection of age points between 1400 and 2000 AD and then generate 112 red noise time series with \( \rho = 0.8 \).

```
>> clear all, close all
>> age=[1400:1:2000]'; Create the age points
>> Rn1=AR1n(0.8,numel(age),112); Generate 112 red time series
>> figure New figure
>> plot(age,Rn1) Plot the time series
>> xlabel('Age [Yr]') Label the x axis
>> ylabel('Red Noise input') Label the y axis
```
Testing the Hockey Stick:
Red noise time series

112 timeseries, $\rho = 0.8$
Testing the Hockey Stick:
Red noise time series

Now we’ll calculate the 1st principal component of the data using the classical approach where the standard deviation of each record is set to 1 and the mean is set to 0.

```matlab
>> A=bsxfun(@minus,Rn1,mean(Rn1)); Subtract the mean
>> A=bsxfun(@rdivide,A,std(A)); Divide by the std
>> [coeffA,scoreA] = princomp_hs(A); Perform the PCA on A
>> figure New figure
>> plot(age,zscore(scoreA(:,1)),'b') Plot the 1st component
>> xlabel('Age [Yr]') Label the x axis
>> ylabel('PCA Score') Label the y axis
```
Testing the Hockey Stick:
Red noise time series

![Graph showing normalized PCA score over age (yr)]
Testing the Hockey Stick:
Red noise time series

112 timeseries, $\rho = 0.8$

Reconstruction period
1400-1850

Calibration period
1850-2000
Testing the Hockey Stick: Red noise time series

Using the Mann approach, the standard deviation of each record is set to 1 during the period 1900-2000 AD and the mean is set to 0 during the same period.

```matlab
>> idx=find(age>=1900 & age<=2000); Points in the calibration
>> B=bsxfun(@minus,Rn1,mean(Rn1(idx,:))); Normalized the mean
>> B=bsxfun(@rdivide,B,std(B(idx,:))); Normalized the std
>> [coeffB,scoreB] = princomp_hs(B); Perform the PCA on B
>> hold on Add to the current plot
>> plot(age,zscore(scoreB(:,1)),'r') Plot the scores
>> legend('Classical PCA','Mann PCA',0) Add a legend
```
Testing the Hockey Stick:
Red noise time series

- Classical PCA
- Mann PCA

Age [Yr]

Normalized PCA Score

1400 1500 1600 1700 1800 1900 2000
Correlation between two time series

Pearson's product-moment coefficients of linear correlation.

Effective number of samples

Cross-spectral Analysis

The state-of-the-art
Linear correlation and regression using “least-squares”

**Correlation:** assesses the degree of relationship between two variables.

**Regression:** gives a quantification of the relationship which exists between two variables.
High $r^2$
Strong correlation

Low $r^2$
Weak correlation
A case not involving time

Downstream variation in size for gravels in Arroyo Seco

We appear to have a linear relationship between distance and particle size.

How can we calculate the linear correlation and regression for this data set?
Classical Regression

Linear regression is expressed using a straight line: \( y = mx + c \)

Where \( m \) is the gradient of the line and \( c \) is the intercept.

Residuals represent the difference between a measured point (\( y \)) and its value (\( \hat{y} \)) predicted by the line.

We can find the best-fit line by minimizing the sum of the squared residuals.

In this way the maximum portion of the variability in \( y \) is explained by the regression line.
Classical Regression

\[ S\!S\!_R = \sum_{i=1}^{n} (\hat{y}_i - \bar{y})^2 \]  
*Sum of squares due to regression.*

\[ S\!S\!_D = \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 \]  
*Sum of squares due to deviation.*

\[ S\!S\!_T = \sum_{i=1}^{n} (y_i - \bar{y})^2 \]  
*Total sum of squares.*

\[ R^2 = \frac{S\!S\!_R}{S\!S\!_T} \]  
*Goodness of fit statistic*  
(high when the regression describes a large proportion of the variation in the data).
Using the $r^2$ value, tells us what fraction of the total variance in $x$ and $y$ can be explained by the linear relationship.

$$r^2 = 0.9571^2 = 0.916 \ (91.6\%)$$
Examples of correlation: perfect positive correlation

\[ r = 1, \ r^2 = 1 \]
Examples of correlation: perfect negative correlation

$r = -1, r^2 = 1$
Examples of correlation: no correlation

\[ r = \sim 0, \quad r^2 = \sim 0 \]
The $r^2$ value gives us a limited amount of information, we need to be able to test if a correlation is statistically significant or not.

If there is no relationship between $x$ and $y$ then we would expect the regression line to have a gradient of zero. In other words, we cannot predict $y$ on the basis of $x$.

$$H_0 : m = 0 \text{ (line has no gradient)}$$

If there is a relationship between $x$ and $y$, the regression line has a gradient which is not zero.

$$H_1 : m \neq 0 \text{ (line has gradient)}$$
Significance of a linear fit
dependent on the quality of the fit (given by $r$ & $r^2$) and the number of data points ($n$).

Null hypothesis ($H_0$): $m = 0$
Alternative hypothesis ($H_1$): $m \neq 0$
Degrees of freedom = $n - 2$

Test Statistic

$$t = r \sqrt{\frac{n-2}{1-r^2}}$$

If $t > t_{crit}$ then $m \neq 0$ is significant at the given level.
Performing correlation and regression in Matlab

>> load arroyo
load the grain size data

>> plot(dist,size,'ok')
plot the data

>> r = corrcoef(dist,size)
determine the correlation coefficient

\[
\begin{array}{cc}
1.0000 & 0.9571 \\
0.9571 & 1.0000 \\
\end{array}
\]

The output matrix gives us the correlation of:

\textit{dist vs. dist, dist vs. size, size vs. dist, and size vs. size.}

For dist vs. size we have \( r = 0.9571 \) and \( r^2 = 0.9160 \)
Performing correlation and regression in Matlab

We now need to test if the correlation is significant or not using the test statistic:

\[ t = 0.9571 \sqrt{\frac{23-2}{1-0.9571^2}} \]

\[ t = \text{0.9571.*sqrt((23-2)./(1-0.9571.^2))} \]

\[ t = 15.1367 \]

The function \texttt{tinv} allows us to calculate the critical value (\(t_{\text{crit}}\)) for a significance level of 0.05 and degrees of freedom \(n-2\):

\[ t_{\text{crit}} = \text{tinv(1-0.05./2,23-2)} \]

\[ t_{\text{crit}} = 2.0796 \]

Because \( t > t_{\text{crit}} \), we must reject \(H_0\) and adopt \(H_1\), therefore at the 0.05 significant level there is a linear relationship between \textit{dist} and \textit{size}. 
A key assumption when testing the significance of a correlation coefficient is that the data points are **independent** of each other.

Does such an assumption hold for the correlation of two time series?

Not always, may systems, for example climate show some form of memory that means consecutive data points depend on each other and are not strictly independent.
AR1 process, $\rho = 0.00$
AR1 process, $\rho = 0.80$
AR1 process, $\rho = 0.99$
We can correct the significance test for data that are not independent by calculating an effective sample size.

The test statistic for independent data: \( t = r \sqrt{\frac{n - 2}{1 - r^2}} \)

The test statistic for autocorrelated data: \( t = r \sqrt{\frac{n_{\text{eff}} - 2}{1 - r^2}} \)

Where the effective sample size, \( n_{\text{eff}} \), is: \( n_{\text{eff}} = n \frac{1 - \rho_x \rho_y}{1 + \rho_x \rho_y} \)

\( \rho_x \) and \( \rho_y \) are the first-order autoregressive AR(1) coefficients of the x and y time series.
An example using red noise series

We'll generate two red noise time series and test the significance of their correlation with and without employing the effective sample size.

>> clear all, close all Clear the memory, close all existing figures

>> N=200; Define the length of the time series
>> X=AR1n(0.95,N); 200 point red noise series with \( \rho=0.95 \)
>> Y=AR1n(0.99,N); 200 point red noise series with \( \rho=0.99 \)

>> figure Generate a new figure
>> subplot(2,1,1) 2 x 1 subplot, activate the first plot
>> plot(1:200,X,'k') Plot the X time series
>> ylabel('X Series') label the y-axis

>> subplot(2,1,2) 2 x 1 subplot, activate the second plot
>> plot(1:200,Y,'k') Plot the X time series
>> xlabel('Time') label the x-axis
>> ylabel('Y Series') label the y-axis
>> figure, plot(X,Y,'.'); 
Form an XY plot of the data

>> xlabel('X'), ylabel('Y'); 
Label the axis

>> r = corrcoef(X,Y) 
Calculate the correlation matrix

>> r = abs(r(2,1)); 
Find the absolute value of r
For my data, $r = 0.3357$

Now we’ll calculate the significance of the correlation without taking the autocorrelation into consideration.

```matlab
>> t0 = r.*sqrt((N-2)./(1-r.^2))  % Calculate the t statistic
  t0 = 5.0143

>> t_crit0 = tinv(1-0.05./2,N-2) % Calculate the critical value of t
  t_crit0 = 1.9720
```

Because $t_0 > t_{crit0}$ we can state that the two time series are **significantly correlated** at the $\alpha = 0.05$ level.
For my data, \( r = 0.3357 \)

Now we’ll calculate the significance taking the autocorrelation into consideration (i.e. doing that statistical test correctly).

\[
\text{Effective samples} \quad \text{Neff} = \frac{N \times (1 - 0.95 \times 0.99)}{1 + 0.95 \times 0.99}
\]

\[
\text{Neff} = 6.1324
\]

\[
\text{Calculate the } t \text{ statistic} \quad t_1 = r \times \sqrt{\frac{(\text{Neff} - 2)}{1 - r^2}}
\]

\[
t_1 = 0.7244
\]

\[
\text{Calculate the critical value of } t \quad t_{\text{crit1}} = \text{tinv}(1 - 0.05/2, \text{Neff} - 2)
\]

\[
t_{\text{crit1}} = 2.7417
\]

Because \( t_0 < t_{\text{crit0}} \) we can state that the two time series are not significantly correlated at the \( \alpha = 0.05 \) level.

Exercise 8.1
In this example we generated time series with specified values of \( \rho \), but of course we don’t normally know this for our data.

Instead there are methods we can use to estimate the value of \( \rho \) for a given data set. We will use the function \emph{ar1.m}

\[
>> \text{rho\_hat} = \text{ar1}(X)
\]

\[
\text{rho\_hat} = 0.83
\]

This result is a strange because when we created \( X \), we used a value of \( \rho=0.95 \).

It’s important to realize the \emph{ar1} can only estimate \( \rho \) on the basis of the data presented to it. This means for short time series the estimate of \( \rho \) may not be very good (it may even be necessary to test if \( \rho \) is significantly different to zero).
We’ll repeat the exercise again with a time series with 10,000 data points. You should see that the estimate is much improved.

```matlab
>> X1=AR1n(0.95,10000);
>> rho_hat1 = ar1(X1)
```

```
rho_hat1 = 0.953
```
More problems!

We may also find a spurious correlation between samples if they both show long-term trends (effectively a strong autocorrelation).

Therefore religion causes alcoholism or the population went through a long-term increase.

Does your data need to be detrended before you start your correlation analysis?
A semi-empirical relation is presented that connects global sea-level rise to global mean surface temperature. It is proposed that, for time scales relevant to anthropogenic warming, the rate of sea-level rise is roughly proportional to the magnitude of warming above the temperatures of the pre–Industrial Age. This holds to good approximation for temperature and sea-level changes during the 20th century, with a proportionality constant of 3.4 millimeters/year per °C. When applied to future warming scenarios of the Intergovernmental Panel on Climate Change, this relationship results in a projected sea-level rise in 2100 of 0.5 to 1.4 meters above the 1990 level.
We’ll repeat Rahmstorf’s analysis using the same method and data set as he used in his paper. To demonstrate this we’ll recreate the graphic shown in the previous slide.

All of the MATLAB commands are stored in the script file sealevel_code.m and can be executed simply by giving the command:

```
>> sealevel_code
```

The 3 figures from Rahmstorf’s paper, will be reproduced.
The variables *temp* (x-axis) and *rate* (y-axis) are stored in the file *sealevel_data.mat*

Load the data and find out if the correlation is significant or not.
>> clear all, close all  
Clear the memory, close all figures

>> load sealevel_data  
load the data

>> temp=detrend(temp)  
detrend the temperature rise

>> rate=detrend(rate)  
detrend the rate

>> subplot(2,1,1)  
2 x 1 plots, use the 1\textsuperscript{st} plot

>> plot(temp,'.k-')  
plot the temperature rise

>> ylabel('Rate of Change (cm/yr)')  
label the y-axis

>> subplot(2,1,2)  
2 x 1, use the 2\textsuperscript{nd} plot

>> plot(rate,'.k-')  
plot the rate

>> ylabel('Warming above 1951-1980 mean')  
label the y-axis

>> xlabel('Time')
>> r = corrcoef(temp,rate);  
Calculate the correlation matrix

>> r = abs(r(2,1));  
The absolute correlation coefficient

>> N=length(rate)  
Number of values in the time series

>> t = r.*sqrt((N-2)./(1-r.^2))  
calculate the t statistic

>> t_crit=tinv(1-0.05./2,N-2)  
calculate the critical value of t

We find that $t = 2.55$ and $t_{crit} = 1.98$, therefore the relationship on which the sea level rise model is based is **significant** at the $\alpha=0.05$ level when the autocorrelation is ignored.
>> r = corrcoef(temp,rate);  
Calculate the correlation matrix

>> r = abs(r(2,1));  
The absolute correlation coefficient

>> N=length(rate)  
Number of values in the time series

>> rho_temp=ar1(temp);  
AR1 coefficient of temp

>> rho_rate=ar1(rate);  
AR1 coefficient of rate

>> Neff=N.*(1-rho_temp*rho_rate)./(1+rho_temp*rho_rate)  
effective N

>> t = r.*sqrt((Neff-2)./(1-r.^2))  
calculate the t statistic

>> t_crit=tinv(1-0.05./2,Neff-2)  
calculate the critical value of t

We find that $t = 0.35$ and $t_{crit} = 3.8$, therefore the relationship on which the sea level rise model is based is **not significant** at the $\alpha=0.05$ level.
Cross Spectral Analysis

We can use cross spectral analysis to compare two time series in the frequency domain. This helps to define which cycles exist in both time series.
The peaks above the dashed line show the frequencies which coexist in both time series at the 90% confidence level.
We can perform the same comparison using precession.
Estimating Pearson’s CorrelationCoefficient With Bootstrap Confidence Interval From Serially Dependent Time Series

Manfred Mudelsee

Pearson’s correlation coefficient, $r_{xy}$, is often used when measuring the influence of one time-dependent variable on another in bivariate climate time series. Thereby, positive serial dependence (persistence) and unknown data distributions impose a challenge for obtaining accurate confidence intervals for $r_{xy}$.

http://www.manfredmudelsee.com/
Group sunspot number, Rg, as solar activity indicator, cosmogenic $^{10}\text{Be}$ in Dye 3 ice core and another proxy of solar activity.
B

(1715 to 1870)

$r_{xy} = -0.45 [-0.57; -0.31]$
$r_{xy} = +0.03 \, [-0.15; +0.20]$