EASC2410 Lecture 21

Introduction to Time Series Analysis 1: Data types, Spectrum and Periodogram

Dr. Binzheng Zhang
Department of Earth Sciences

Review of Lecture 20:

- 3-D line plots
- 3-D surface plots
- Vector and streamline plots
- Mapping data on maps

In Lecture 21, you will learn:

- learn basic concepts of data types
  - Section data
  - Time series data
  - Panel data
- the time domain versus frequency domain
- spectrum and power spectrum (periodogram)
- how to generate a periodogram using Python

What is a time series

Definition

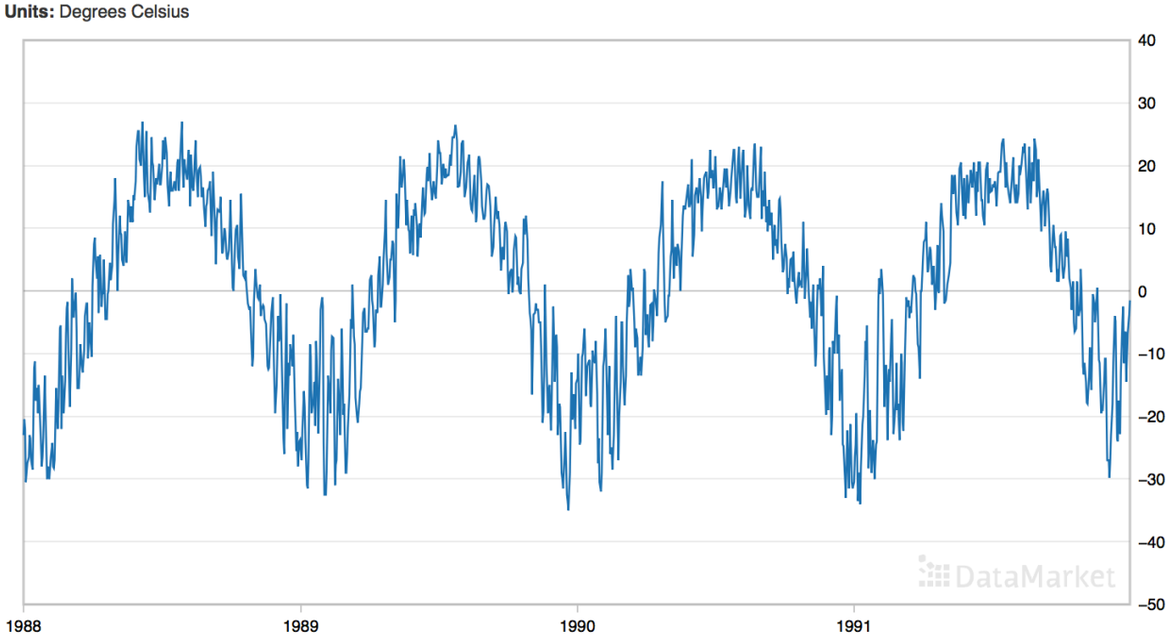
- The data is a time series in the form of a sequence of quantitative observations about a system or process and made at successive points in time.
- Usually time series data is equally spaced in time (if not, then resamplings are needed)

Examples

- gross domestic product versus year
- sales volumes versus seasons
- stock prices versus day
- weather attributes versus hours

Mean daily temperature, Fisher River near Dallas, Jan 01, 1988 to Dec 31, 1991

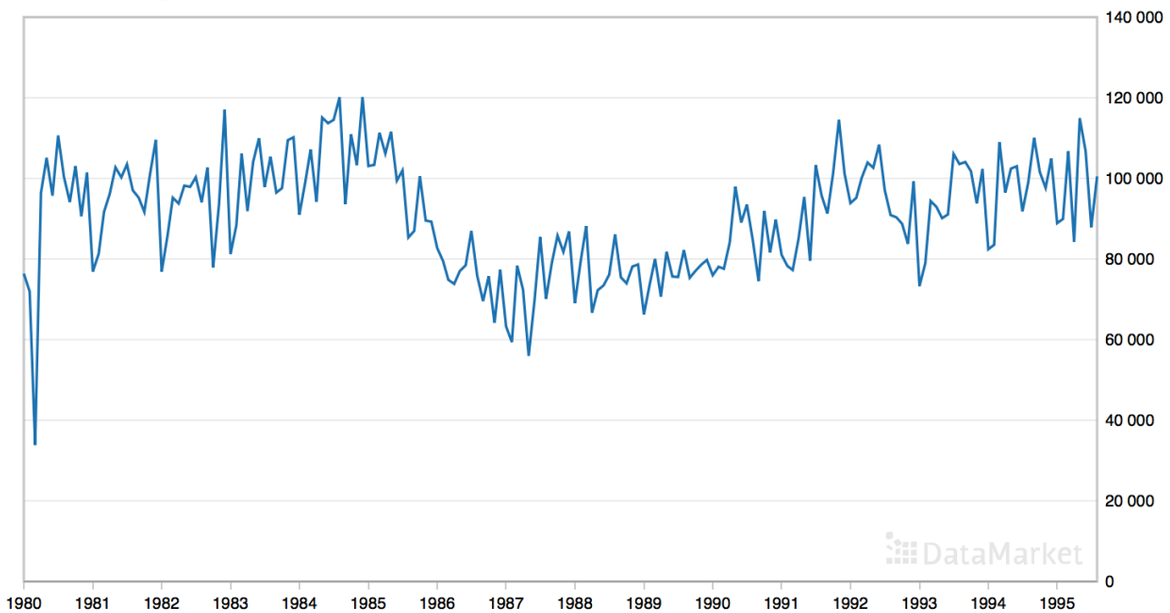
Units: Degrees Celsius



Source: Time Series Data Library (citing: Hipel and McLeod (1994))

Monthly total number of pigs slaughtered in Victoria. Jan 1980 – August 1995

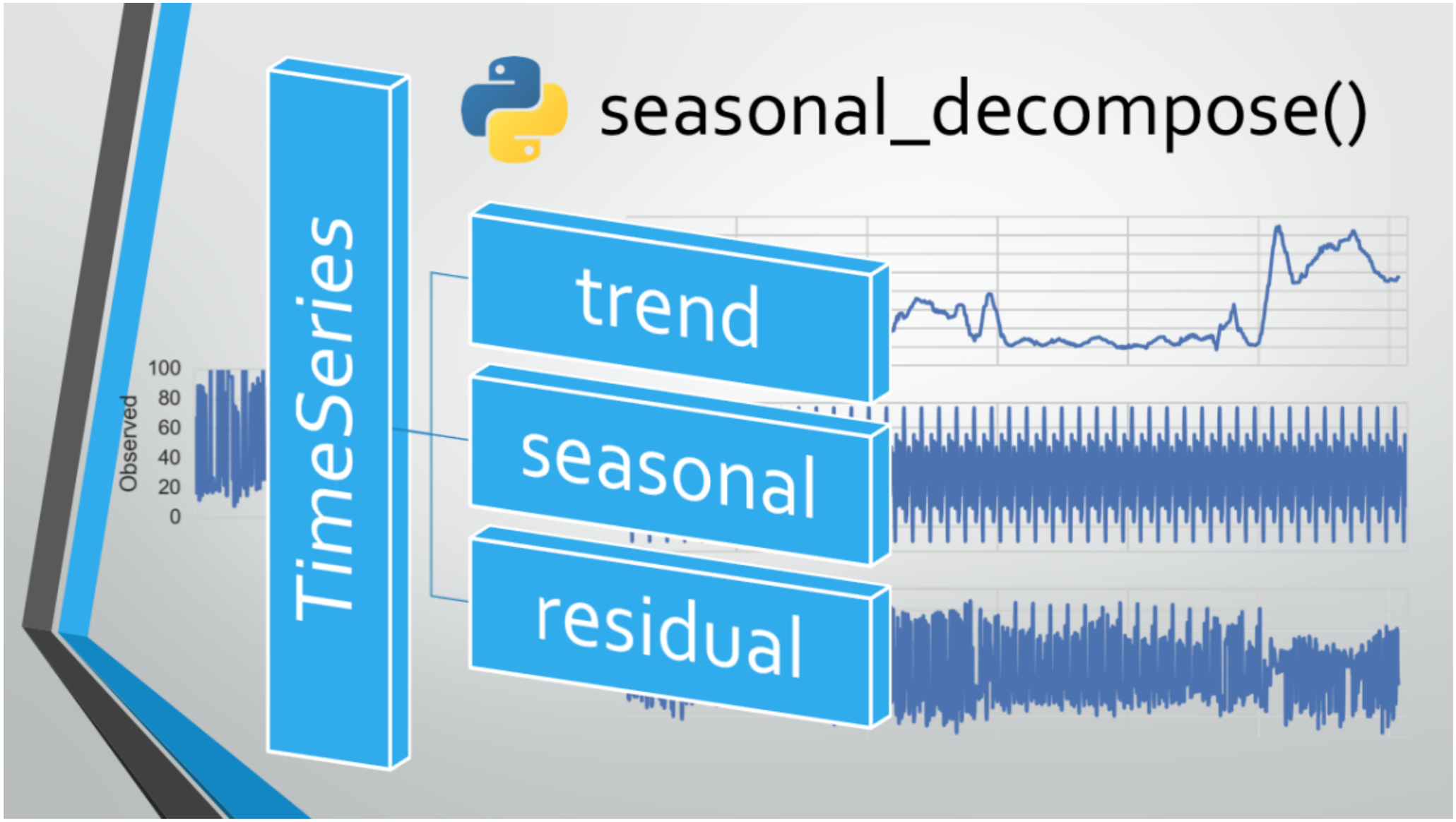
Units: Number of pigs



Source: Time Series Data Library (citing: Australian Bureau of Statistics)

Why is a time series analysis useful?

Time series analysis applies different statistical methods to explore and model the **internal structures of the time series data** such as periodicity, trends, seasonal fluctuations, cyclical behavior, and random/irregular/unexpected changes.



Types of data (1-D series)

Data scientists come across many different types of data in their analytics projects. Most data commonly found in academic and industrial projects can be broadly classified into the following categories:

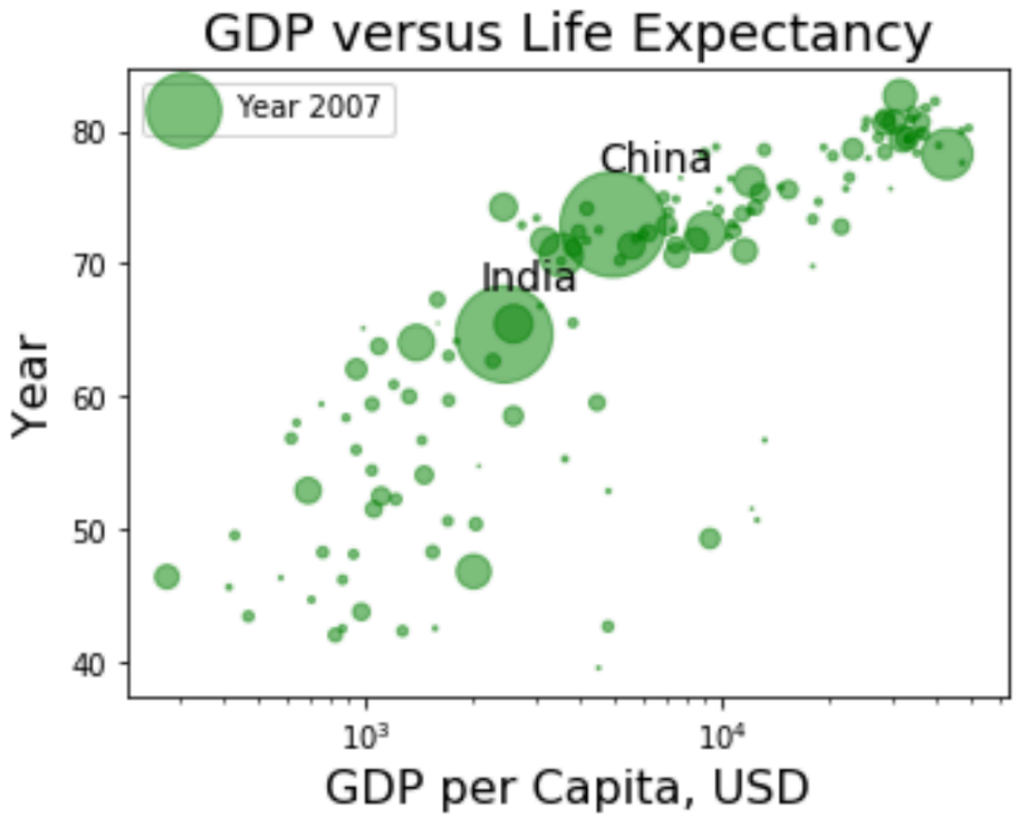
- Cross-sectional data
- Time series data
- Panel data

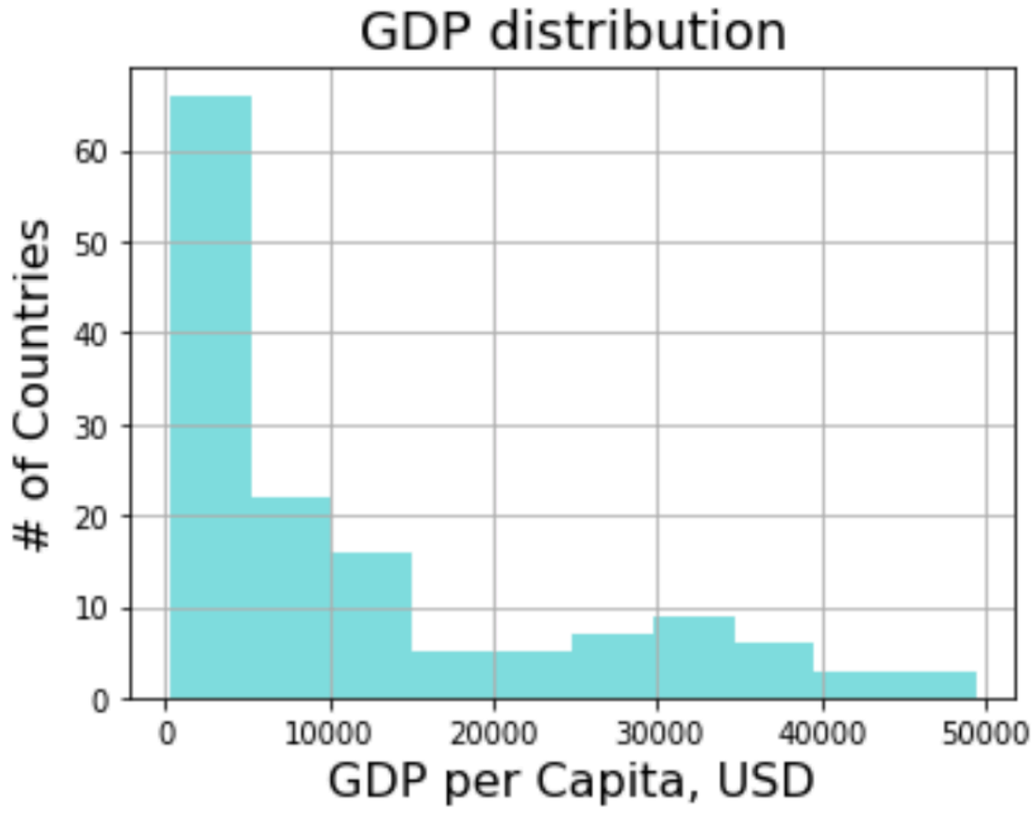
Understanding what type of data is needed to solve a problem and what type of data can be obtained from available sources is important for formulating the problem and choosing the right methodology for analysis.

Data types: Cross-Sectional Data

- Cross-sectional data or cross-section of a population is obtained by taking observations from multiple individuals **at the same point in time**.
- Cross-sectional data can comprise of observations taken at different points in time, however, in such cases time itself **does not** play any significant role in the analysis.

Let's take a look at the “gdp_versus_life” example we've seen a lot in previous lectures:

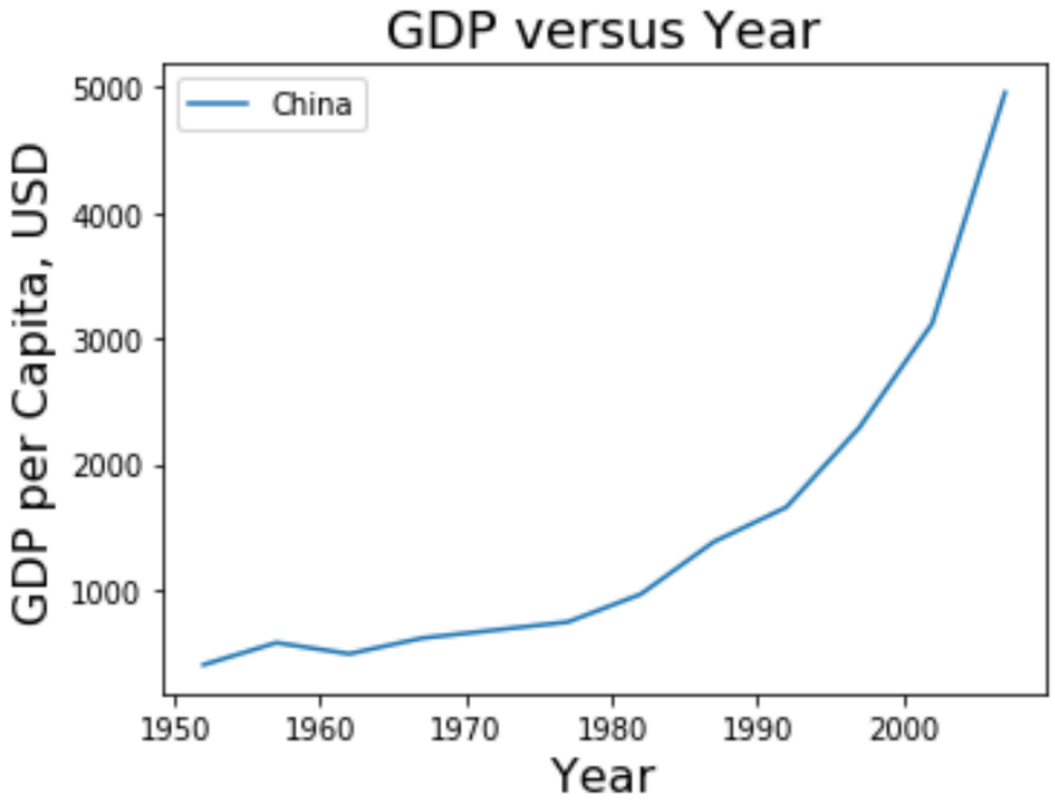


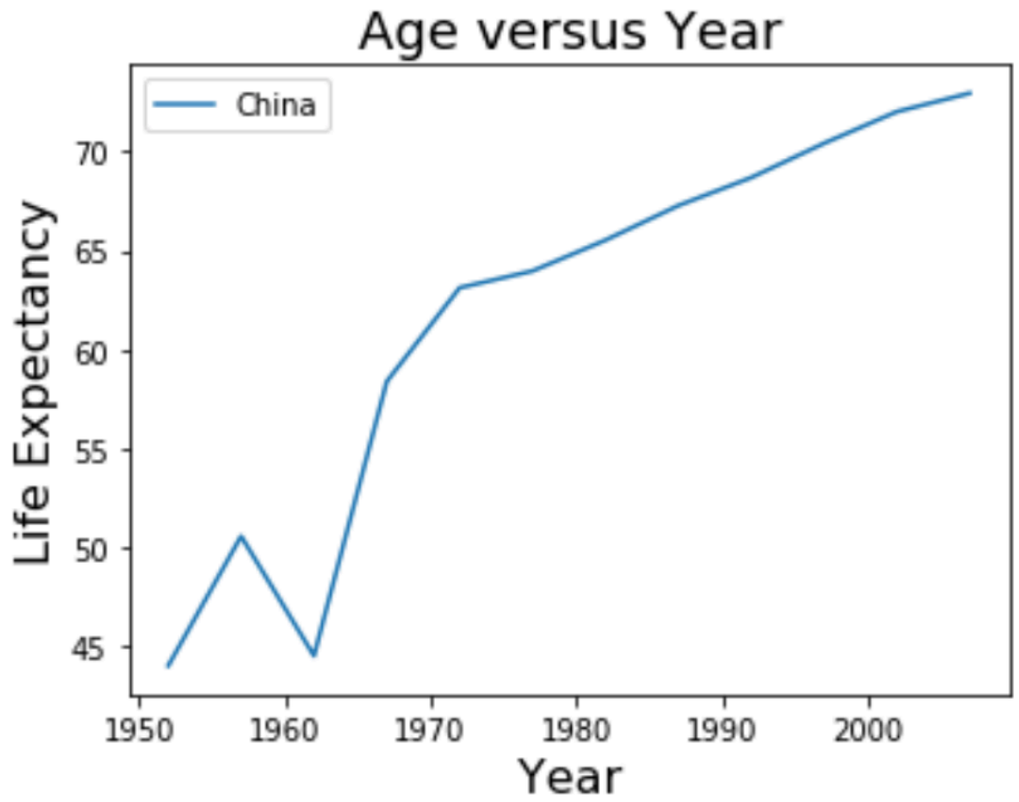


In the above example, we are only showing all the GDP, life expectancy, population etc. data for **year 2007**, and there's **NO** comparisons between different years, which means time does not play an important role in interpreting the data (yet). So they are cross-sectional data.

Data types: Time-Series Data

The example of cross-sectional data discussed earlier is from the year 2007 only. However, instead if we consider the GDP data from only one country, for example China, and plot it as a function of time:

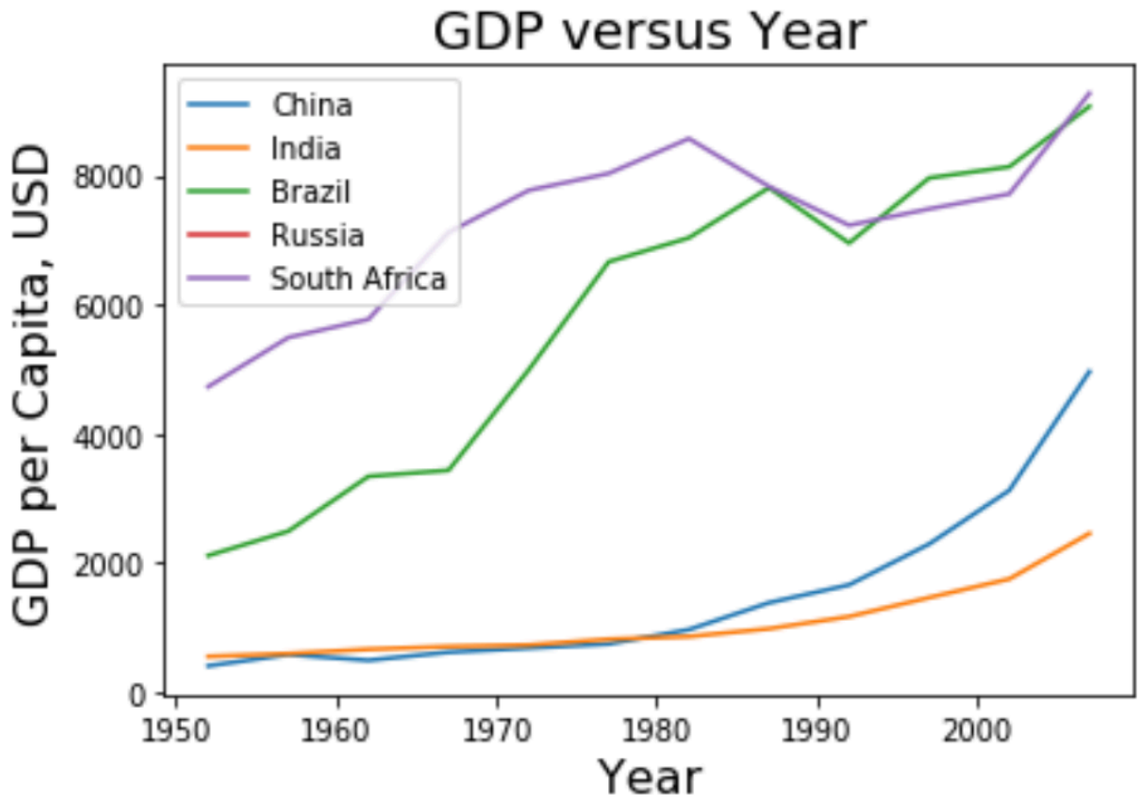


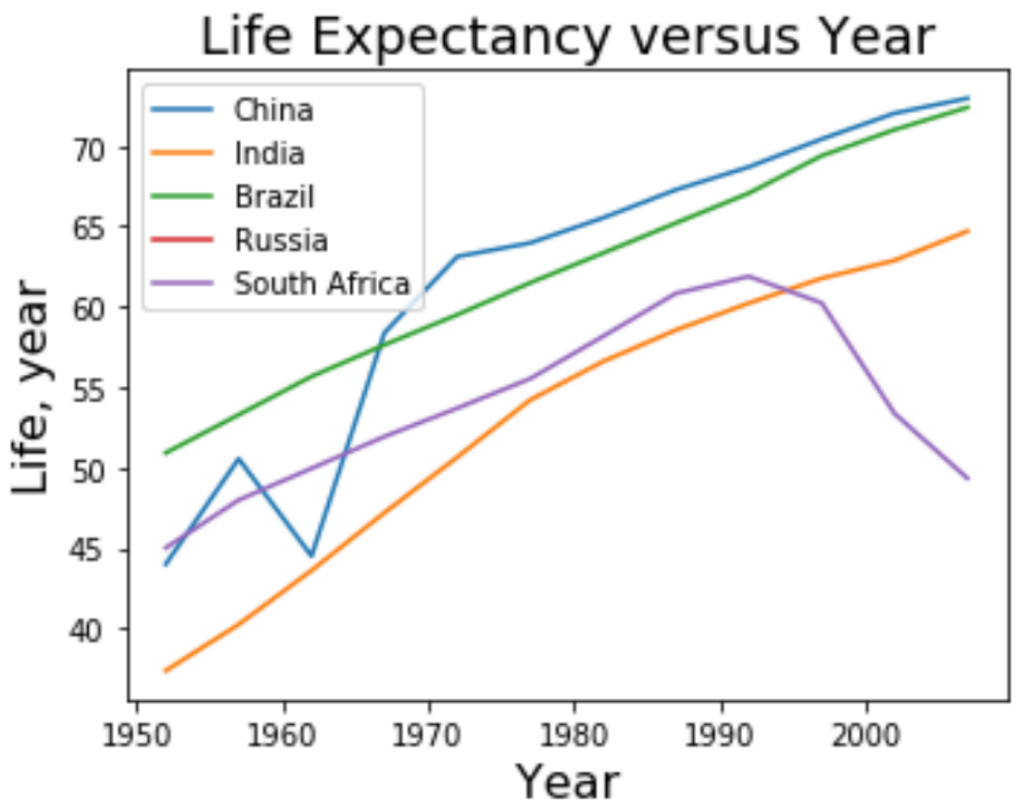


You can see the variations of the data points as functions of time (over the past 60 years). These are examples of time series in the gdp_life data set you have already been working with for quite a few times.

Data types: Panel Data

So far, we have seen data taken from multiple individuals but at one point in time (cross-sectional) or taken from an individual entity but over multiple points in time (time series). However, if we observe *multiple entities over multiple points in time* we get a **panel data** also known as **longitudinal** data. Extending our earlier example about the military expenditure, let us now consider the "BRICS" countries over the same period of 1952-2007. The resulting data will be a panel dataset. The figure given below illustrates the panel data in this scenario.





You can see that panel data shows both variations in time and between different countries - they have one more dimension compared to either the cross-sectional data or the time-series data.

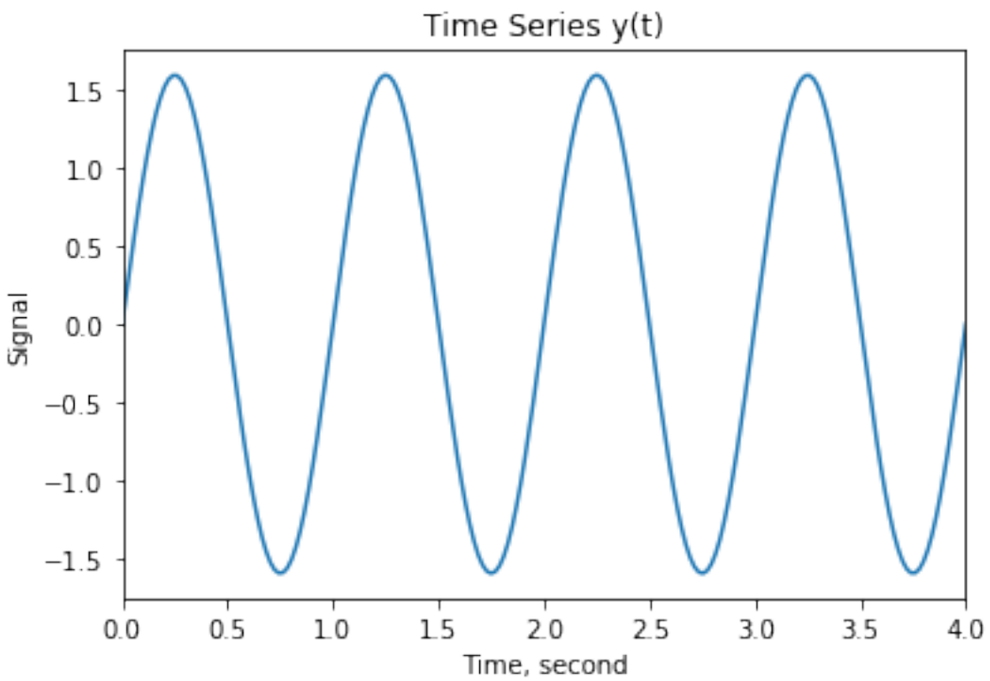
Properties of time serial data: Time Domain

In time series data, the variation of a particular variable is respect to time (in years, months, days, hours,minutes, seconds, etc.). In Signal processing, we call these data points are defined in the **time domain**, which means that in a line plot, the x-axis is for time.

For example, let's use Python to generate the following signal which is a simple sine wave in the time domain:

$$y(t) = 1.6\sin(2\pi t)$$

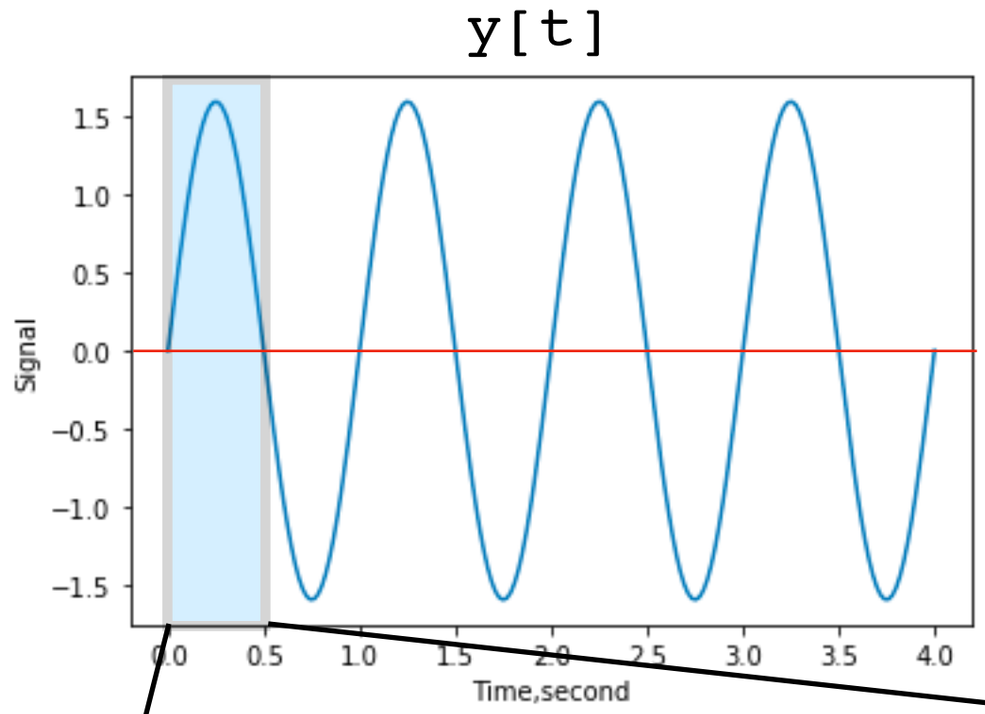
```
t = np.linspace(0,4,200)
y = 1.6*np.sin(2*np.pi*t)
plt.plot(t,y)
plt.xlabel('Time, second')
plt.ylabel('Signal')
plt.xlim([0,4])
plt.title('Time Series y(t)')
```



y(t) is a **time series**, in physics and engineering, y(t) is often called a **signal**. As its name stands, y(t) is a time-domain signal, which means the data y varies as a function of time t.

The processing of y(t) for engineering applications, is known as signal processing, which is basically everyday life! (beyond the scope of this course)

Properties of time serial data: Time Domain

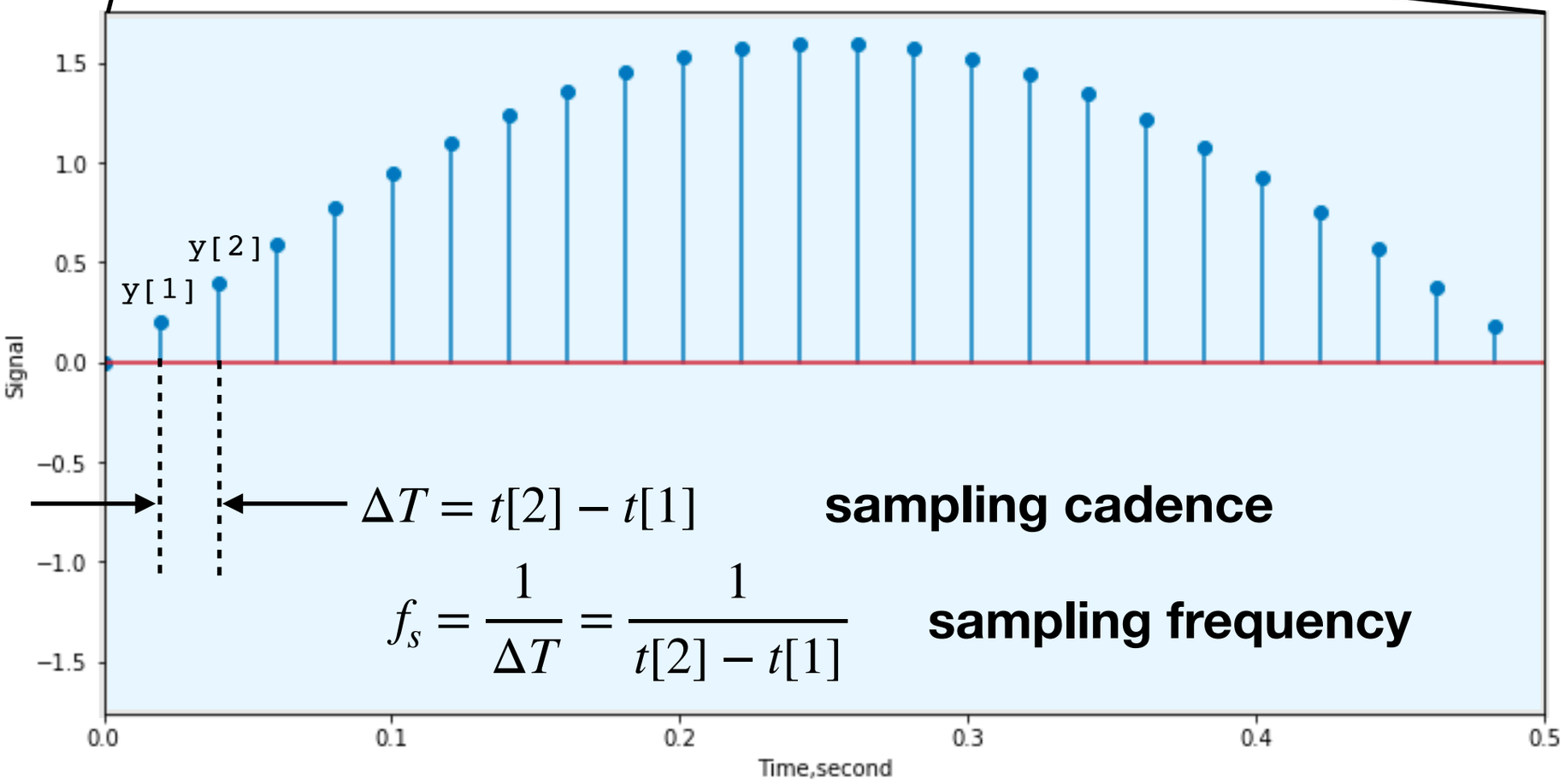


$$y(t) = A\sin(\omega t) = A\sin(2\pi f t)$$

magnitude

Angular frequency

frequency



$$\Delta T = t[2] - t[1]$$ **sampling cadence**

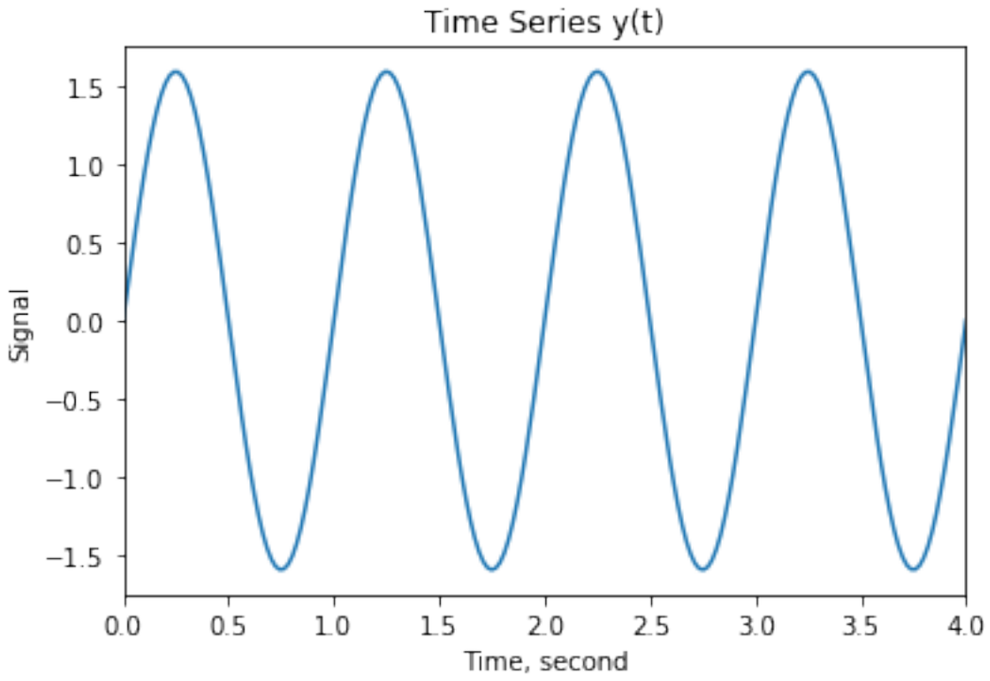
$$f_s = \frac{1}{\Delta T} = \frac{1}{t[2] - t[1]}$$ **sampling frequency**

Properties of time serial data: Frequency Domain

Let's use take a close look at the time domain data (signal)

$$y(t) = 1.6 \sin(2\pi t)$$

- it shows sinusoid variations with respect to time
- it has a magnitude of 1.6
- it has a periodicity of 1.0 second

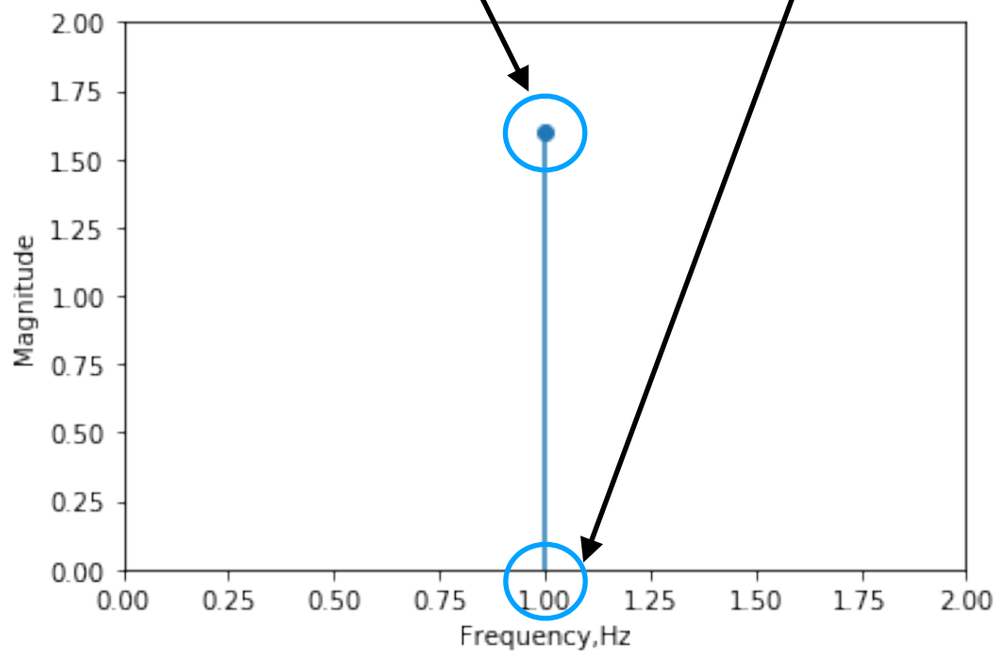


Time Domain

magnitude

Frequency (angular)

Now let's make another simple (almost trivial) plot showing the relationship between the wave frequency (x-axis) and magnitude (y-axis):



Frequency Domain

Properties of time serial data: Time Domain

Such a frequency-magnitude plot is called a "spectrum". Let's take a look at another example of signals with multiple frequencies:

$$y(t) = 0.5 + \Sigma_{n=1}^{n=6} \frac{1}{n} \cos(2\pi n t)$$

Which is a constant 0.5 with 8 frequency components from 1 to 1/6. More specifically, the signal y is written as:

$$y(t) = 0.5 + 1\cos(2\pi t) + \frac{1}{2}\cos(4\pi t) + \frac{1}{3}\cos(6\pi t) + \frac{1}{4}\cos(8\pi t) + \frac{1}{5}\cos(10\pi t) + \frac{1}{6}\cos(12\pi t)$$

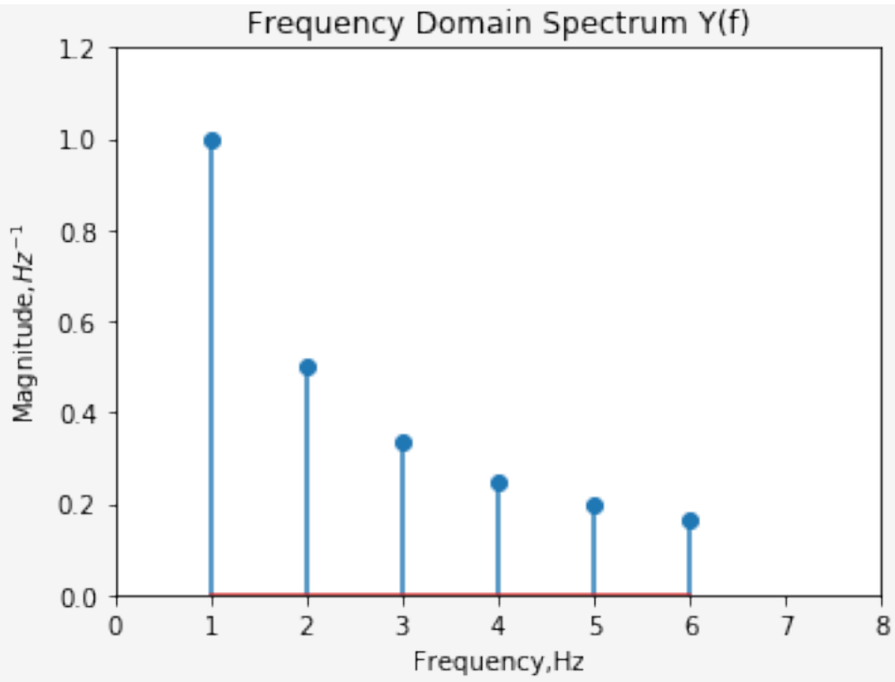
```
n = np.array(range(1,7))

freq = n
magn = 1/n

t = np.linspace(0,4,200)

y = magn[0]*np.sin(2*np.pi*freq[0]*t)+ \
    magn[1]*np.sin(2*np.pi*freq[1]*t)+ \
    magn[2]*np.sin(2*np.pi*freq[2]*t)+ \
    magn[3]*np.sin(2*np.pi*freq[3]*t)+ \
    magn[4]*np.sin(2*np.pi*freq[4]*t)+ \
    magn[5]*np.sin(2*np.pi*freq[5]*t)+0.5
```

```
n = [1,2,3,4,5,6]
```



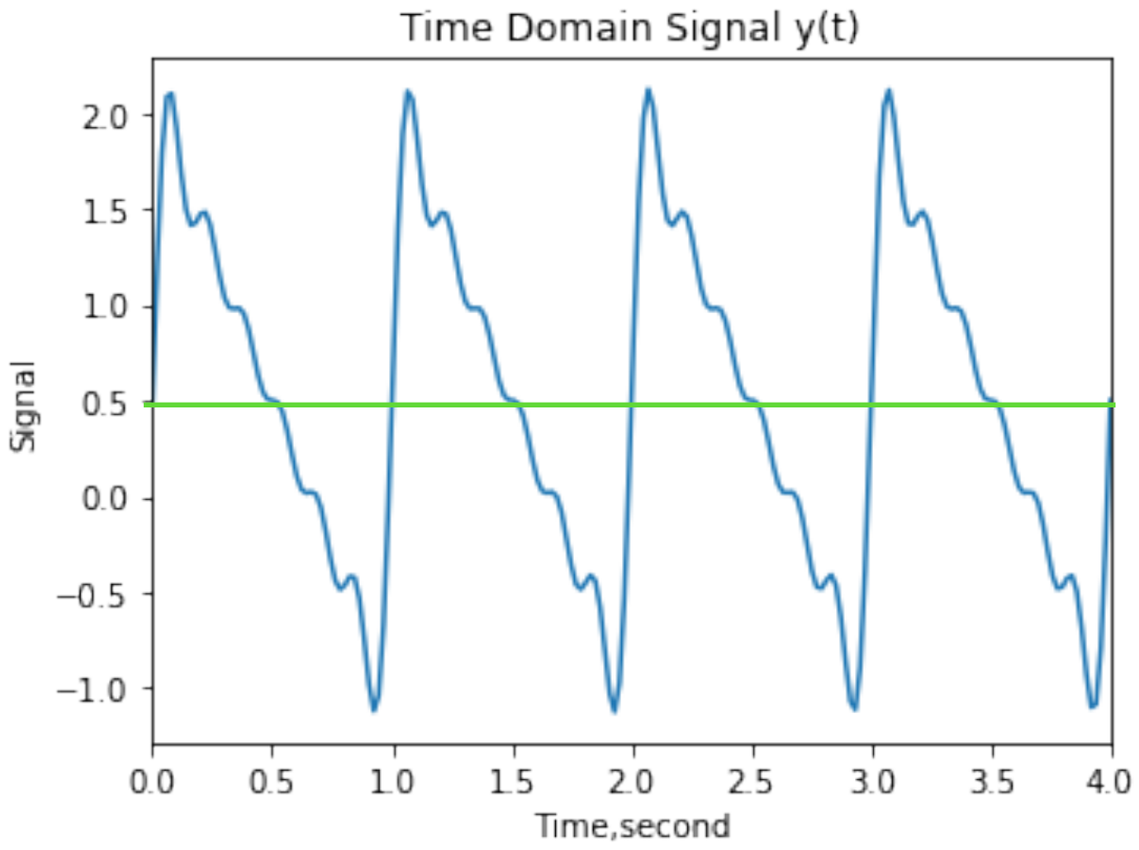
Properties of time serial data: Frequency Domain - Spectrum

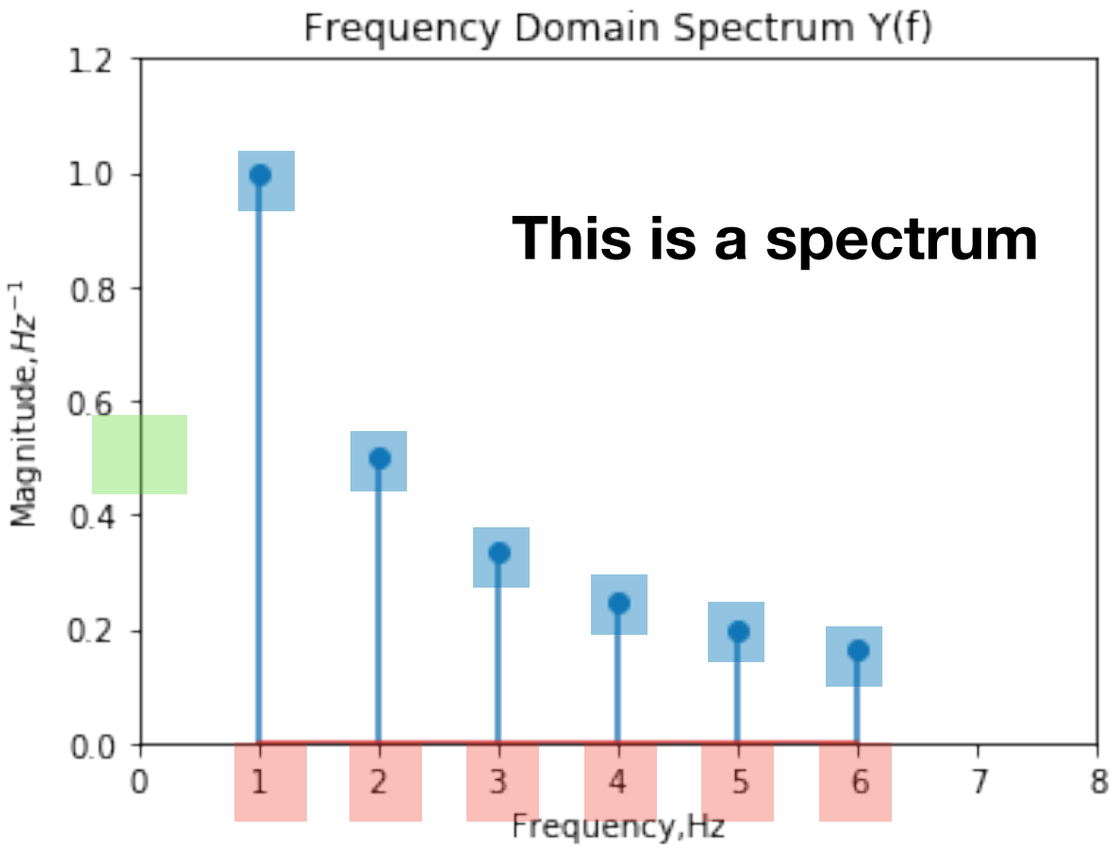
Such a frequency-magnitude plot is called a "spectrum". Let's take a look at another example of signals with multiple frequencies:

$$y(t) = 0.5 + \Sigma_{n=1}^{n=6} \frac{1}{n} \cos(2\pi n t)$$

Which is a constant 0.5 with 8 frequency components from 1 to 1/6. More specifically, the signal y is written as:

$$y(t) = 0.5 + 1\cos(2\pi t) + \frac{1}{2}\cos(4\pi t) + \frac{1}{3}\cos(6\pi t) + \frac{1}{4}\cos(8\pi t) + \frac{1}{5}\cos(10\pi t) + \frac{1}{6}\cos(12\pi t)$$





Properties of time serial data: Frequency Domain - Power Spectrum

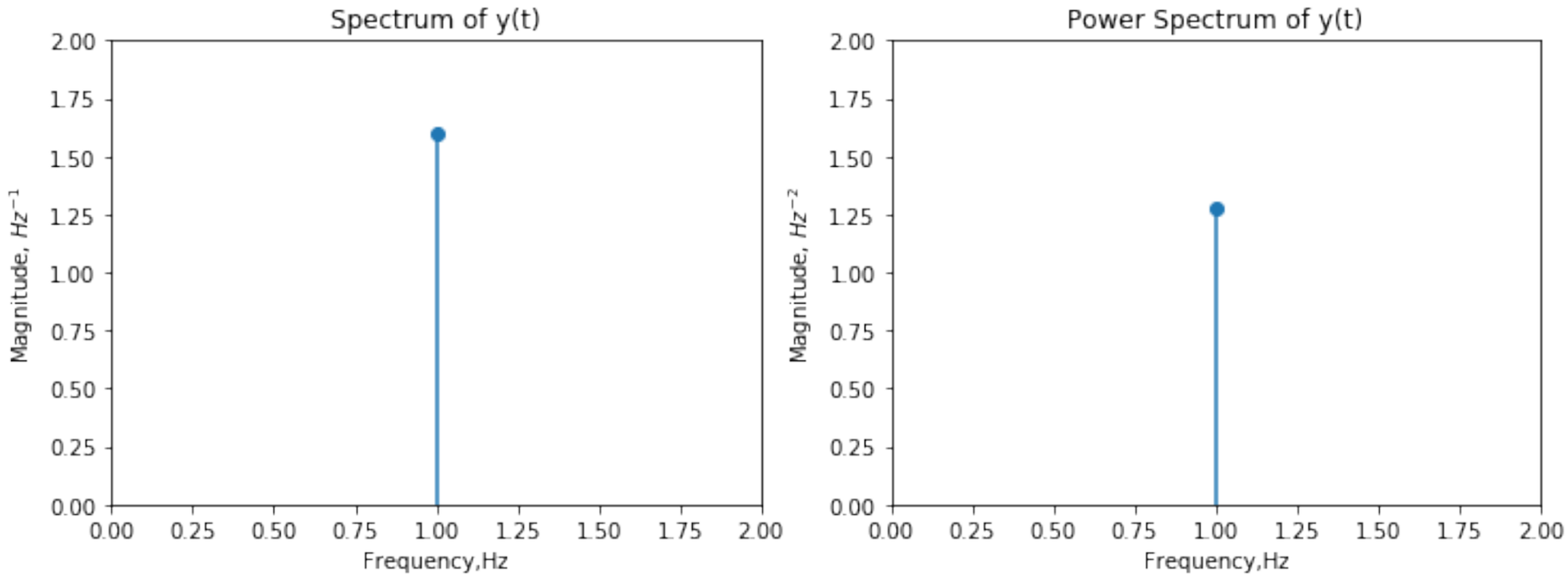
In time series data analysis, the frequency information of a time series is very important for understanding the dynamic nature of the data, which is also useful for physical interpretation, decision making and predictions. Besides the frequency spectrum, a more useful description for the frequency information of a time series is the **Power Spectrum**, which is the power in each frequency component of the time series.

Power Spectrum is relatively straightforward to understand in the frequency domain. For example let's re-visit the simple sine signal

$$y(t) = 1.6\sin(2\pi t)$$

We know that y(t) has one frequency component with a magnitude of 1.6, imagine that y(t) is the electric voltage measured at home, then the power associated with voltage signal y(t) is simply

$$P = \frac{1}{2}y_{mag}^2 = \frac{1}{2}\cdot 1.6^2 = 1.28$$

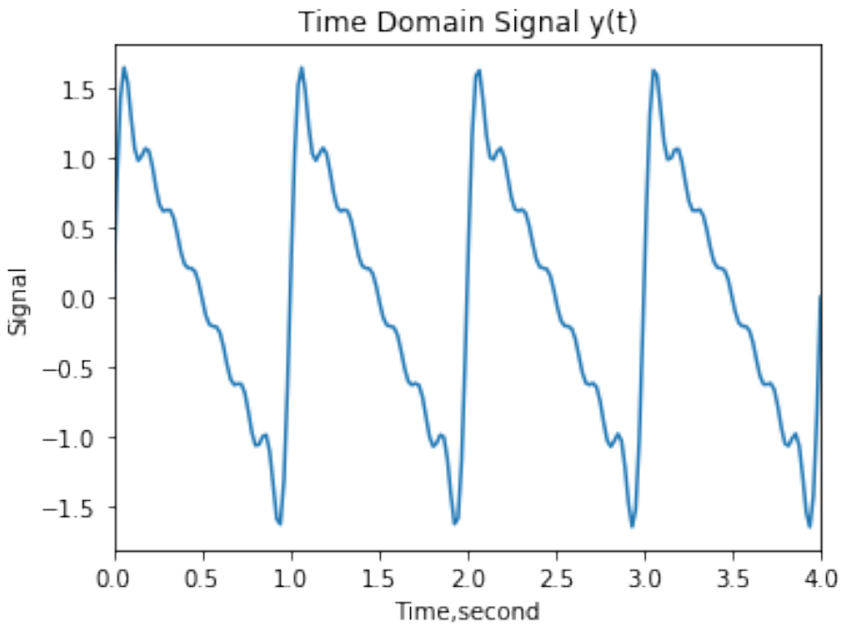


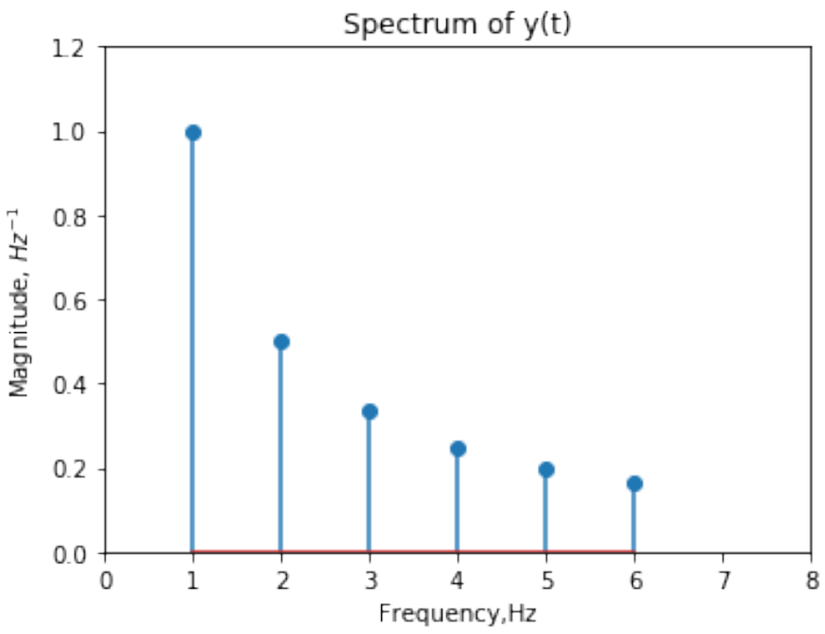
Properties of time serial data: Frequency Domain - Power Spectrum

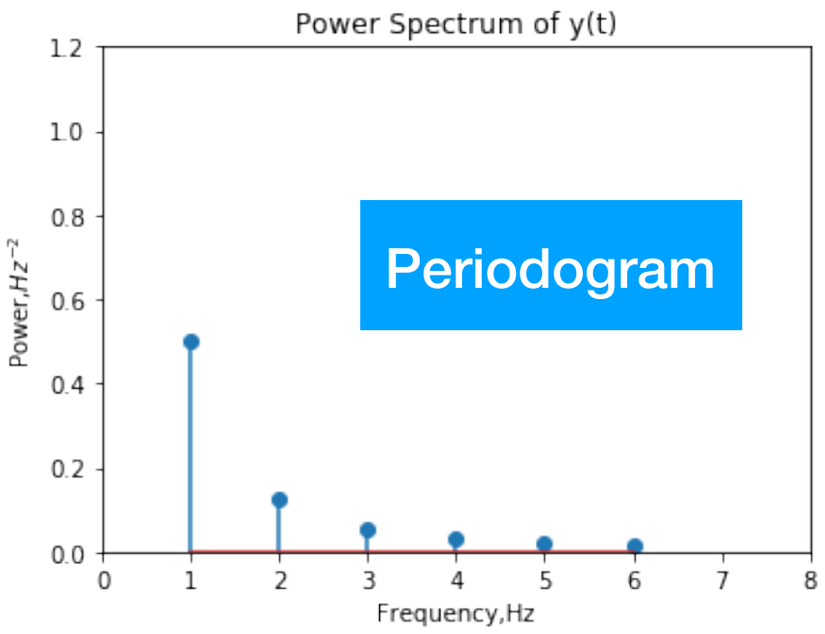
Spectrum: the magnigude of each wave component as a function of the corresponding frequency is called a **spectrum**, it gives you a sense of the relative magnitudes between different frequency components within a singal.

Power Spectrum: the power (or energy density) of each wave component as a function of the corresponding frequency is called a **power spectrum**, it gives you a sense of the power of each freqeuency component with in a signal, which is also known as a **periodogram**

$$y(t) = 1\cos(2\pi t) + \frac{1}{2}\cos(4\pi t) + \frac{1}{3}\cos(6\pi t) + \frac{1}{4}\cos(8\pi t) + \frac{1}{5}\cos(10\pi t) + \frac{1}{6}\cos(12\pi t)$$







Periodograms in Python

We use the **signal** module from **scipy** to make periodograms in Python

```
from scipy import signal # import the scipy.signal module

freq = n
magn = 1/n
```

`n = [1,2,3,4,5,6]`

$$y(t) = 1\cos(2\pi t) + \frac{1}{2}\cos(4\pi t) + \frac{1}{3}\cos(6\pi t) + \frac{1}{4}\cos(8\pi t) + \frac{1}{5}\cos(10\pi t) + \frac{1}{6}\cos(12\pi t)$$

example time series

```
t = np.linspace(0,4,200) # time domain, from 0 to 4 seconds, 200 data points

y = magn[0]*np.sin(2*np.pi*freq[0]*t)+ \
    magn[1]*np.sin(2*np.pi*freq[1]*t)+ \
    magn[2]*np.sin(2*np.pi*freq[2]*t)+ \
    magn[3]*np.sin(2*np.pi*freq[3]*t)+ \
    magn[4]*np.sin(2*np.pi*freq[4]*t)+ \
    magn[5]*np.sin(2*np.pi*freq[5]*t)

fs=50 # sampling frequency, why it's 50?

prec_freqs,prec_power=signal.periodogram(y,fs) # compute power spectrum using scipy
```

Syntax for the signal.periodogram( ) function:

scipy.signal.periodogram(*x*, *fs=1.0*, *window=None*, *nfft=None*, *detrend='constant'*, *return_onesided=True*, *scaling='density'*, *axis=-1***)**

- x is the time series for computing a periodogram
- fs is the sampling frequency, default as 1.0 (which means the data points are collected every 1 time unit)

Documentation for the signal.periodogram( ) function:

https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.signal.periodogram.html

Periodograms in Python

Note on sampling frequency fs:

$$f_s = \frac{1}{\Delta t}$$

Here Δt is the time difference between two consecutive data points in a time series.

```
from scipy import signal # import the scipy.signal module

freq = n
magn = 1/n

t = np.linspace(0,4,200) # time domain, from 0 to 4 seconds, 200 data points

y = magn[0]*np.sin(2*np.pi*freq[0]*t)+ \
    magn[1]*np.sin(2*np.pi*freq[1]*t)+ \
    magn[2]*np.sin(2*np.pi*freq[2]*t)+ \
    magn[3]*np.sin(2*np.pi*freq[3]*t)+ \
    magn[4]*np.sin(2*np.pi*freq[4]*t)+ \
    magn[5]*np.sin(2*np.pi*freq[5]*t)

fs=50 # sampling frequency, why it's 50?

prec_freqs,prec_power=signal.periodogram(y,fs) # compute power spectrum using scipy
```

time spacing info

```
fs = 1.0/(t[1]-t[0])
print('The sampling frequency is',fs,' Hz')
```

```
The sampling frequency is 50.0  Hz
```

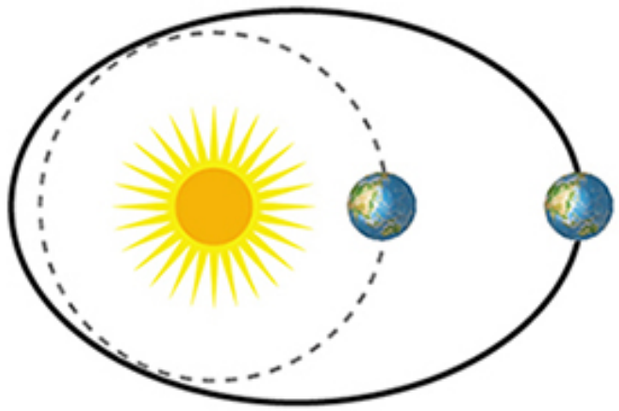
Here the np.linspace(0,4,200) function tells Python to generate a numpy array between 0 and 4 seconds, with 200 data points, which means that the data points are separated by Δt = (4-0)/200 = 0.02 second. Thus,

$$f_s = \frac{1}{\Delta t} = \frac{1}{0.02} = 50 \quad \text{(Hz)}$$

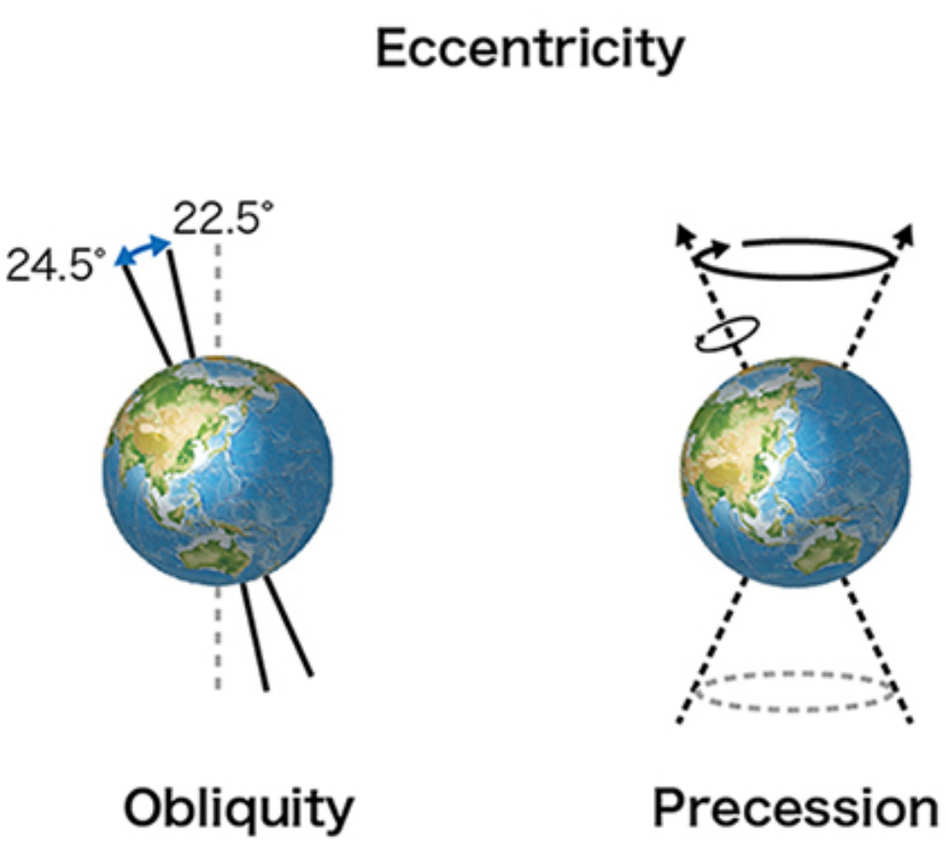
Question: what if t is not generated by a numpy array, how do you get fs?

Periodograms in Earth Sciences: Earth Orbit

The orbital cycles can be calculated by knowing the exact configuration of the planets. Milankovitch famously took the first stab at it from his prison cell during WWI. Nowadays it is calculated with fancy computers. The key parameters are *eccentricity* (or ovalness of the orbit around the sun), the *obliquity* (tilt) of the spin axis and the *precession* of the spin axis.



The Earth's orbital parameters of ellipticity, obliquity and precession vary in predictable ways. One commonly used model for variations in them over the last few hundred million years was published by Laskar et al. (2004; http://dx.doi.org/10.1051/0004-6361:20041335).



Let's take a look for the behavior of the last few million years using the data file from the Laskar et al. (2004) paper.

```
# Read in the datafile into a Pandas DataFrame
cycles=pd.read_csv('Datasets/INSOLN.LA2004.BTL.100.csv')
print (cycles.columns)
```

```
Index(['Age (ka)', 'Eccentricity', 'Obliquity', 'Precession'], dtype='object')
```

[*Figure from* http://www.jamstec.go.jp/e/about/press_release/20141027/; *see also* http://www.sciencecourseware.org/eec/GlobalWarming/Tutorials/Milankovitch/].

Periodograms in Earth Sciences: Earth Orbit

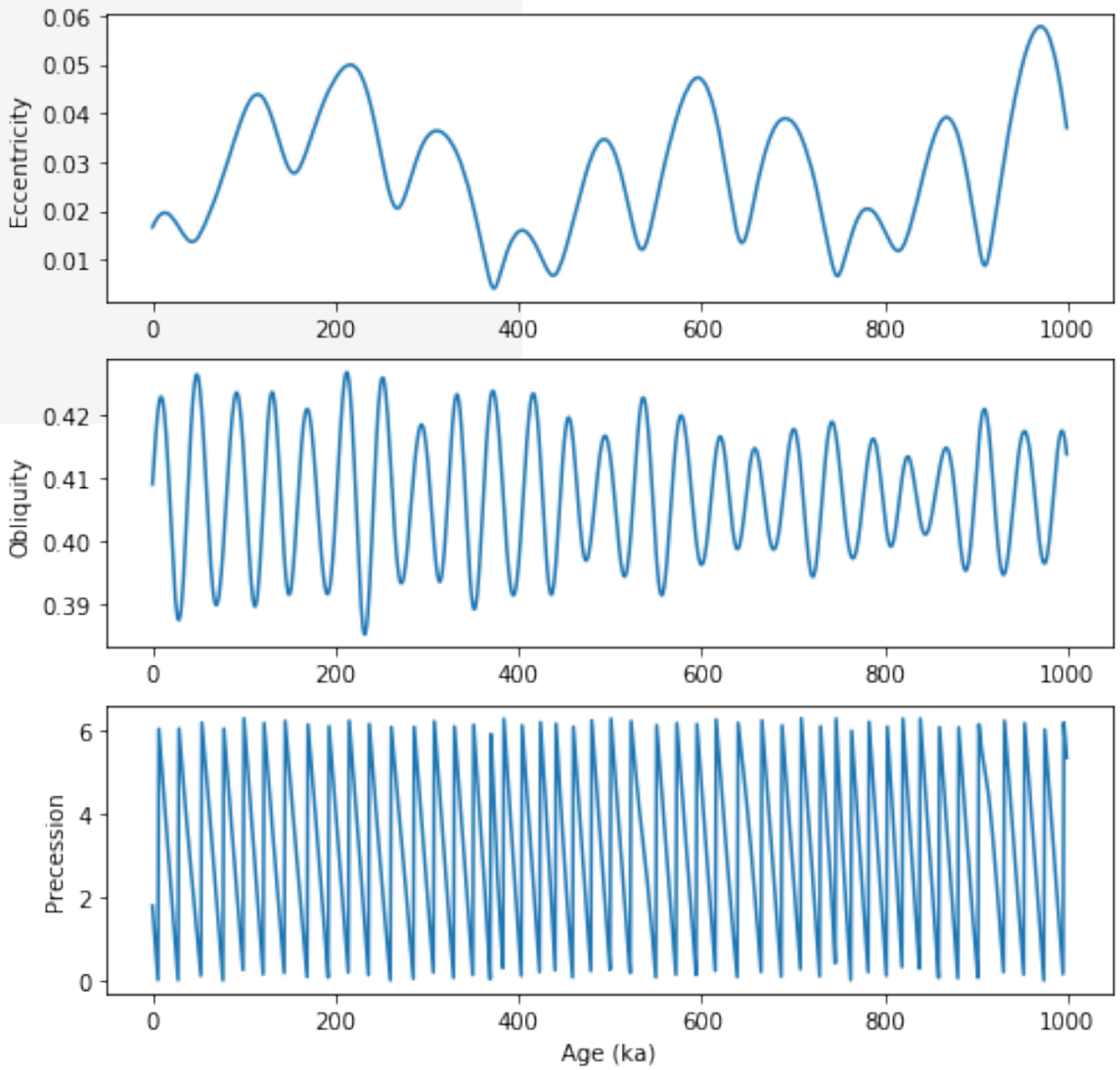
First, we filter the data for the last million years and then we plot it as a time series.

```
cycles_1Ma=cycles[cycles['Age (ka)']<1000] # only look at last 1000 ka (1 Million).

# set up the plot as three rows
# start with Eccentricity
fig=plt.figure(1,(8,8)) # make a nice big plot
fig.add_subplot(311) # notice how you do not need the commas!
plt.plot(cycles_1Ma['Age (ka)'],cycles_1Ma['Eccentricity'])
plt.ylabel('Eccentricity')

# add obliquity
fig.add_subplot(312)
plt.plot(cycles_1Ma['Age (ka)'],cycles_1Ma['Obliquity'])
plt.ylabel('Obliquity')

# add precession
fig.add_subplot(313)
plt.plot(cycles_1Ma['Age (ka)'],cycles_1Ma['Precession'])
plt.ylabel('Precession')
plt.xlabel('Age (ka)');
```



- Eccentricity, Obliquity and Precession are all time series data
- Precession varies at faster time scales compared to Obliquity and Eccentricity
- How do we use Python to get the periodicities in the three physical processes?

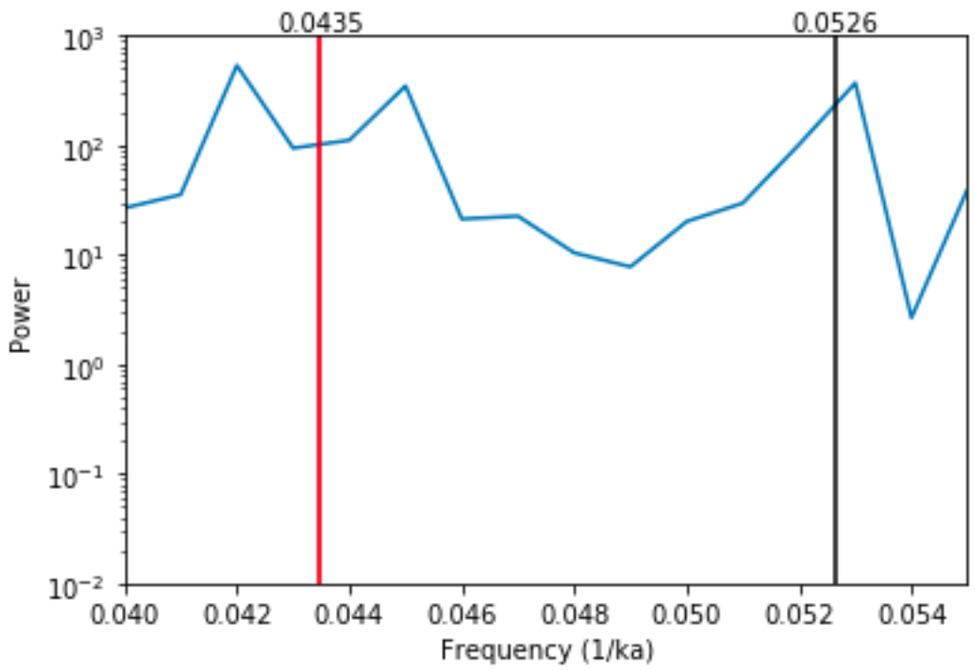
Periodograms in Earth Sciences: Earth Orbit

```
# make an array out of the desired data series
prec=np.array(cycles_1Ma['Precession'])
time = np.array(cycles_1Ma['Age (ka)'])
fs = 1/(time[1]-time[0])

# and calculate the frequencies
prec_freqs,prec_power=signal.periodogram(prec,fs)
# plot on a linear x, log y plot (using semilogy( ))
plt.semilogy(prec_freqs,prec_power)
plt.ylim(.01,1000) # truncate the Y axis
plt.xlim(.001,.06) # truncate the X axis
# put on the precessional frequencies
plt.axvline(x=1./23.,color='red')  # use a vertical line
plt.axvline(x=1./19.,color='black')
plt.xlabel('Frequency (1/ka)') # label the axes
plt.ylabel('Power')
plt.text(1./23.,1000,'0.0435',ha='center',va='bottom')
plt.text(1./19.,1000,'0.0526',ha='center',va='bottom');
plt.xlim(.04,.055) # truncate the X axis
plt.show()
```

call the periodogram() function to get outputs

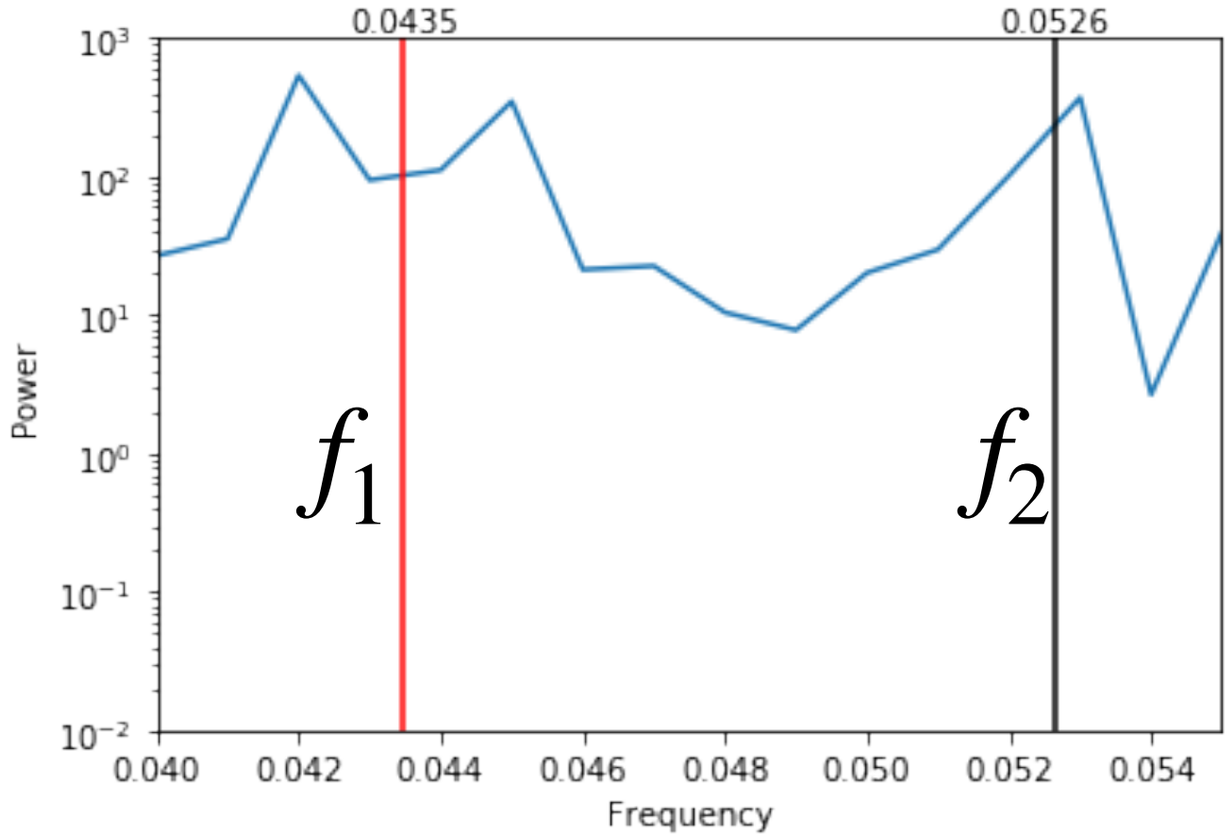
- Step 1: convert pandas column to Numpy array
- Step 2: compute sampling frequency fs
- Step 3: use the signal.periodogram( ) function
- Step 4: plot power spectrum



It is a little rough but you can clearly see the two peaks near 23 and 19 kyr. We could smooth out the diagram by exploring *windowing* options, but for now, let's just use what the default options are.

How to get the 23 ayr and 19 kyr estimations?

Periodograms in Earth Sciences: Earth Orbit



Back to the data unites:

$$T = \frac{1}{f_1}$$

Peak Frequency from periodogram

We found two peaks in the periodogram: f1 = 0.0425 (1/ka) and f2 = 0.0526 (1/ka).
The normalized periodicity will be

$$T_1 = \frac{1}{f_1\ (ka^{-1})} = 1/0.0425 \approx 23.5\ \ ka$$

Now let’s make some periodograms