**Download Chapter notebook (ipynb)**

- How is time series data visualised?
- What information is provided by the Fourier spectrum?
- When is the term “harmonics” used?

- Use offsets to display multiple time series
- Describing EEG data plots
- Interpreting Fourier spectra of time series

## Time Series

### Import Time Series

In this lesson, we will use the **Pandas** package to import time series data. The Pandas package for data science is included in the Anaconda distribution of Python. It is also included in our virtual environment.

To use a function contained in Pandas, it first needs to be imported. Our time series data is in `.csv` format, and we therefore need to read this data in from a csv file. In order to do this, we will import the function `read_csv`. This function will create a Pandas dataframe.

Note that the location of the data file is specified within quotes by the relative path to the subfolder `data` followed by the file name. Use the JupyterLab file browser to check that this `data` subfolder exists within the same folder as this notebook, and that it contains the `EEG_background.txt` file.

### Plotting NumPy series

As an example, let us import a time series data. This represent human electroencephalogram (EEG) as recorded during normal background activity.

```
from pandas import read_csv
```

```
# Please check the path to the file on your machine!

df1 = read_csv("data/EEG_background.txt", delim_whitespace=True)

df1.head()
```

```
        FP1       FP2        F3        F4  ...       EO2       EM1       EM2
                  PHO
```

```
0  -7.4546  22.8428   6.28159  15.6212  ...  13.7021  12.9109  13.7034
    9.37573
1 -11.1060  21.4828   6.89088  15.0562  ...  13.7942  13.0194  13.7628
    9.44731
2 -14.4000  20.0907   7.94856  14.1624  ...  13.8982  13.1116  13.8239
    9.51796
3 -17.2380  18.7206   9.36857  13.0093  ...  14.0155  13.1927  13.8914
    9.58770
4 -19.5540  17.4084  11.06040  11.6674  ...  14.1399  13.2692  13.9652
    9.65654

[5 rows x 28 columns]
```

The specification of `delim_whitespace` is needed because data are not comme-separated but space-separated!

The `head()` function gives you an overview of the resulting dataframe.

If you want to know how many rows and columns there are, use `shape`.

```
df1.shape
```

```
(2373, 28)
```

We can then convert the dataframe into a Numpy array.

```
data_1 = df1.to_numpy()

data_1.shape
```

```
(2373, 28)
```

#### Time Series Plot

If we directly plot the Numpy array, the result will be a mess, as each data recording is plotted on top of another, and it becomes very difficult to visualise.

```
from matplotlib.pyplot import subplots, show

fig, ax = subplots()

ax.plot(data_1);
```

Below is a block of code that improves the plot. We are defining a **Python function**. To activate it, execute the cell.

```
def plot_series(data, sr):
    '''
```

```
    Time series plot of multiple time series
    Data are normalised to mean=0 and var=1

    data: nxm numpy array. Rows are time points, columns are channels
    sr: sampling rate, same time units as period
    '''
    from numpy import flip
    from matplotlib.pyplot import subplots
    from numpy import arange, linspace, zeros


    samples = data.shape[0]
    sensors = data.shape[1]

    period = samples / sr

    time = linspace(0, period, samples)

    offset = 5 # for mean=0 and var=1 normalised data

    # Calculate means and standard deviations of all columns
    means = data.mean(axis=0)
    stds = data.std(axis=0)

    # Plot each series with an offset of 2 times the standard
       deviations
    fig, ax = subplots(figsize=(7, 8))

    ax.plot(time, (data - means)/stds + offset*arange(sensors-1,-1,-1))
       ;

    ax.plot(time, zeros((samples, sensors)) + offset*arange(sensors
       -1,-1,-1),'--',color='gray');

    ax.set(xlabel='Time')
    ax.set_yticks(offset*arange(sensors))
    ax.set_yticklabels(flip(arange(sensors)+1))
```

**Python Function**

Please execute the above function definition before proceeding. The function code takes data and creates a plot of all columns as time series, one above the other, using a constant offset. When you execute the function code nothing happens. Similar to the import, running a function code will only activate it and make it available for later use.

The function is now activated. To use it, you simply need to call it by the name that we assigned to it, in the cell above: `plot_series()`.

Any keyword arguments have to be provided in parantheses, in the correct order. The first argument is an array containing the data. The second argument is the sampling rate. The sampling rate of the imported EEG is 256.

```
sr = 256

plot_series(data_1, sr);

# To only see the first 20 channels, use:
# plot_series(data_1[:, :20], sr);

show()
```

Observations: In this display of a non-pathological (background) EEG, you should be able to observe the following:

- There are irregular oscillations of all recorded brain potentials.
- Oscillations recorded at different locations above the brain differ.
- Oscillations are not stable, but modulated over time.
- There are different frequency components in each trace.

**Exercise**

Try to import the data in file `data/EEG_absence.txt` into a new dataframe and convert the to a Numpy array called `data_2`. Then display `data_2` as a time series using the same function.

```
df2 = read_csv("data/EEG_absence.txt", delim_whitespace=True)

df2_np = df2.to_numpy()

data_2 = df2_np[:, :20]

data_2.shape
```

```
(1721, 20)
```

```
plot_series(data_2, sr)

show()
```

This is a recording of the human EEG during an epileptic seizure. Here are some observations regarding this pathological recording:

- There are regular oscillations.

- Oscillations recorded at different locations are not identical but similar or at least related in shape.
- Despite some modulation, oscillations are fairly stable over time.
- There are repetitive motifs composed of two major components throughout the recording, a sharp spike and a slow wave.

### Fourier Spectrum

The Fourier spectrum decomposes the time series into a sum of sine waves. The spectrum gives the coefficients of each of the sine wave components. The coefficients are directly related to the amplitudes needed to optimally fit the sum of all sine waves to recreate the original data.

However, the assumption behind the Fourier transform is that the data are provided as in infinitely long stationary time series. These assumptions are not fulfilled as the data are finite and stationarity of a biological system can typically not be guaranteed. Thus, interpretation needs to be cautious.

We import the Fourier transform function for real data `rfft` from scipy.fft. We can use it to transform all columns of a Nympy array at the same time.

We then find out how many rows there are in the data and calculate the (normalised) amplitudes and the corresponding frequencies. The latter uses `rfftfreq` from scipy.fft.

```
from scipy.fft import rfft, rfftfreq

data_2_fft = rfft(data_2, axis=0)

rows  = data_2.shape[0]

# amplitude
amplitudes = (2.0 / rows)*abs(data_2_fft)

# frequencies
freqs = rfftfreq(rows, 1 / sr)
```

```
amplitudes.shape
```

```
(861, 20)
```

To plot the results, we pick a single time series and display its Fourier spectrum. Note how the frequency range is controlled using `set_xlim()`.

```
from matplotlib.pyplot import subplots

chan = 0
```

```
4
5  fig, ax = subplots()
6
7  ax.plot(freqs, amplitudes[:, chan]);
8  ax.set_xlim(0, 20);
9  ax.set_xlabel('Frequency (Hz)')
10 ax.set_ylabel('Amplitude');
11 # ax.set_yscale('log')
```

We can see that in channel with index 0, the main amplitude contributions lie between 2 and 3 Hz, the so-called fundamental frequency. There are also some further local maxima. These are interpreted as "harmonics", integer multiples of the fundamental frequency.

Clinically, this means that the signal recorded during this epileptic seizure is dominated by an approximately 3Hz rhythm where each cycle lasts about a third of a second. The presence of harmonics points to the nonlinear nature of the rhythm.

Here is a cell with code to display the spectrum of all columns in your array:

```
1  cols = data_2.shape[1]
2
3  fig, axes = subplots(figsize=(6, 15), nrows=cols, sharex=False)
4
5  names = df2.columns
6
7  for index, ax in enumerate(axes.flat):
8
9      axes[index].plot(freqs, amplitudes[:, index])
10     axes[index].set_xlim(0, 12)
11     axes[index].set(ylabel=f'{names[index]}')
12
13 axes[index].set(xlabel='Frequency (Hz)');
14
15 fig.tight_layout()
16
17 show()
```

We find that most channels display the fundamental between 2.5 and 3 Hz. Clinically, this is then referred to as a Generalised Seizure, meaning it can be recorded over a large area of cortex.

**Keypoints**

- `plot_series` is a self-made/user-defined Python function created to display multiple time-series plots.
- The Fourier spectrum decomposes a time series into a sum of sine waves to find the dominant frequencies.

- `rfft` is a SciPy function to calculate the Fourier transform of all columns of a (real-valued) Nympy array.