Week 7

We have introduced Python. Now we need to discuss Excel.

Script

`Excel`

Excel, and spreadsheets in general, are a widely used and understood tool. Its strength is that nearly everyone has it available, and it is fast to do basic tasks with. It's very useful for

- Distributing data
- Performing quick analysis
- Communicating general results rapidly to a broad audience

However, there are major issues with using spreadsheets. One weakness is that it is easy to make mistakes with spreadsheets and particularly with Excel. Amongst the many examples, the Reinhart-Rogoff paper on sovereign debt accidentally excluded five rows from a calculation of an average, which qualitatively changed their results. It has been claimed that political austerity measures were strongly motivated by this paper, which in part relies on a spreadsheet error.

We need to talk about reliable methods for getting data into, and out of, spreadsheets. We also need to talk about data organization principles: how to make it easy for others to re-use your data. Finally, we need to look at the standard Excel mathematical functions for data analysis.

One advantage of Python over Excel is that, as a full-featured programming language, it can be more easily used for long and complex tasks. The disadvantage is that many people may be interested in the results, but not interested in (or have the knowledge of) the programming details required to follow the Python code. To communicate the results whilst still showing those interested the full data, it's useful to *programmatically* take results from Python into a spreadsheet.

For now we will use toy data. As an example,

```python
import numpy as np
x = np.linspace(1, 3)
s = np.sin(x)
```

We now want to create an Excel spreadsheet that contains this data. Working with larger datasets will follow similarly. For this we will use the `openpyxl` library.

```python
import openpyxl
```

We first create an Excel workbook, purely within Python.

```python
wb = openpyxl.Workbook()
```

We will then create a worksheet within the workbook. One already exists, but we will create a new worksheet with a name.

```python
ws = wb.create_sheet("Data")
```

We will then add titles in the first row, indicating what the data is going to be. These will be strings.

```
ws.cell(row=1, column=1, value="x")
ws.cell(row=1, column=2, value="sin(x)")
```

We will now add the actual data. Think carefully about how the loop indexes work here.

```
for i in range(len(x)):
    ws.cell(row=2+i, column=1, value=x[i])
    ws.cell(row=2+i, column=2, value=s[i])
```

Finally, we will save the workbook.

```
wb.save(filename='example.xlsx')
```

We can now open this workbook and check. You should see two sheets within the workbook. The first is blank; the second contains the data from Python.

Exercise

Add more columns to the workbook, plotting different functions such as $\cos^2(x)$ or $\exp(-x)$. Ensure there are suitable headers. How could you use dictionaries to make this more efficient?

Plotting in Excel

In the worksheet, highlight all the data including the header row. From the `Insert` menu, choose `Chart > X Y (Scatter)`. You should see a chart with different coloured lines, and a legend at the bottom corresponding to the header row. However, there are no axis labels and no title.

Select the chart by clicking on it. In the ribbon bar, left edge, select `Add Chart Element` and then `Axis Titles`. Use this to add axis labels.

Try modifying the axes to be logarithmic using `Add Chart Element`, `Axes`, select one axis, and go through the options until it shows the logarithmic options. Once done, convert back to linear axes.

By using `Ctrl` plus clicking (or right-clicking) on the chart, save it as a `png` file and a `pdf` file. We will need this to embed in reports.

Now try modifying the data range so that only the first half of the points are plotted. By `Ctrl` clicking again, choose `Select Data...` and modify the range. Note that the `$` symbols fix the range: this will be important when talking about formulas later.

Finally, some of our data sets now look roughly like straight lines. We want to add a trend line to measure their slope. In the ribbon menu, go to `Chart Design` and select `Add Chart Element, Trendline, Linear`. This should add the best fit line. Can you extract its slope? Can you annotate the plot with the slope?