

COMP1730/COMP6730

Programming for Scientists

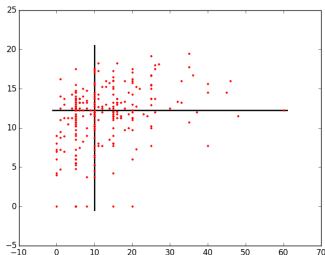
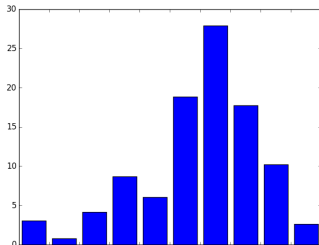
Data Science

Lecture outline

- * Analysing data
- * Visualisation
- * Data science modules

Data analysis

- * Reading data files
- * Representing tables
- * Working with data:
selecting, visualising,
counting
- * Interpretation



Reading data files

- * Many data file formats (e.g., excel, csv, json, binary).
- * Use a python module that helps with reading the file format:

```
import csv
with open("filename.csv") as csvfile:
    reader = csv.reader(csvfile)
    data = [ row for row in reader ]
```

- * More about (reading and writing) files later in the course.

Representing tables

- * Lists are 1-dimensional, but a list can contain values of any type, including lists.
- * A table can be stored as a list of lists, by row, for example:

```
data[i] # i:th row
```

```
data[i][j] # j:th column of i:th row
```

- * Indexing (and slicing) are *operators*
- * Indexing (and slicing) associate to the left:

```
data[i][j] == (data[i])[j].
```



- ★ A *list comprehension* creates a list by evaluating an expression for each value in an iterable collection (e.g., a sequence).

```
first_col = [ row[0] for row in data ]  
last_two_cols = [ row[-2:]  
                  for row in data ]
```

- ★ Can also have a filtering condition:

```
sel_rows = [ row for row in data  
             if row[0] > 1 ]
```



- * `sorted(seq)` returns a list with values in `seq` sorted in default order (`<`).
 - We can sort the rows in a table.
 - Reminder: comparison of sequences is lexicographic.
- * `sorted(seq, key=fun)` sorts value `x` by `fun(x)`.

```
def new_order(row):  
    return -row[-1] # decreasing  
                    # on last col  
  
sd = sorted(data, key=new_order)
```



Descriptive statistics

- * `min(seq)`;
- * `max(seq)`;
- * `mean(sum(seq) / len(seq))`;
- * variance.
- * No built-in function for median.

```
def median(seq):  
    return sorted(seq)[len(seq) // 2]
```


Visualisation

- * The purpose of visualisation is to see or show information – not to draw pretty pictures!
- * Different kinds of plots show different things:
 - histogram, pie-chart or cumulative distribution
 - scatterplot
 - line and area plot
- * Use one that best makes the point!

Using matplotlib

```
import matplotlib.pyplot as plot  
  
plot.hist([first_col, last_col])  
plot.legend(["column A", "column D"])  
plot.show()
```

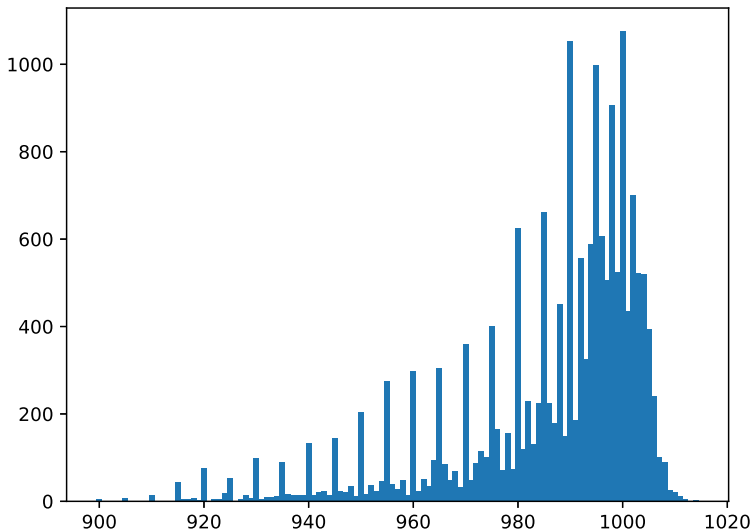
```
plot.plot(first_col, last_col)  
plot.xlabel("column A")  
plot.ylabel("column D")  
plot.show()
```

* **Documentation:** matplotlib.org

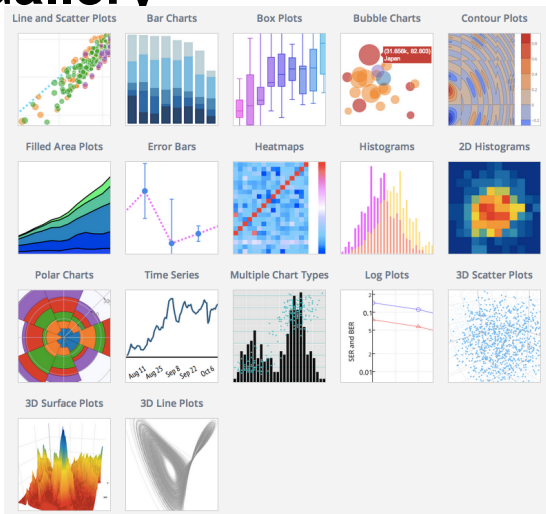


Interpretation

- * Understand what the data represents.
- * Statistical significance.
- * Over-fitting.
- * Correlation is not causation.



Gallery



Source: <https://plot.ly/javascript/basic-charts/>

Visualisation Tips

- * Use a chart that is appropriate for your data.
- * Format your chart appropriately, labels, title, axis, scale, etc., from within the code.
- * Make sure your colour scheme works well for printed reports (including black and white).
- * Be consistent with your colours and styles across figures in the same report.

Animation, Interfaces and Videos

- * You can produce animations in `matplotlib`.
- * Think of animation as drawing several individual graphics, one after another.
- * You can also use `matplotlib` to create interactible graphical user interfaces, with buttons and other controls.
- * If you have proper codecs installed, you can turn your animation into videos.
- * There are good tutorials available if you are interested in exploring these topics further (we don't go over them in this course).

Data Science Modules

NumPy and SciPy

- * The NumPy and SciPy libraries are not part of the python standard library, but often considered essential for scientific / engineering applications.
- * The NumPy and SciPy libraries provide
 - an n -dimensional array data type (`ndarray`);
 - fast math operations on arrays/matrices;
 - linear algebra, Fourier transform, random number generation, signal processing, optimisation, and statistics functions;
 - plotting (via `matplotlib`).
- * Documentation: `numpy.org` and `scipy.org`.

NumPy Arrays

- * `numpy.ndarray` is sequence type, and can also represent n -dimensional arrays.
 - `len(A)` is the size of the first dimension.
 - Indexing an n -d array returns an $(n - 1)$ -d array.
 - `A.shape` is a sequence of the size in each dimension.
- * All values in an array must be of the same type.
- * Element-wise operators, functions on arrays.
- * Read/write functions for some file formats.

Generalised indexing

- * If A is a 2-d array,
 - $A[i, j]$ is element at i, j (like $A[i][j]$).
 - $A[i, :]$ is row i (same as $A[i]$).
 - $A[:, j]$ is column j .
 - $:$ can be *start:end*.
- * If L is an array of `bool` of the same size as A , $A[L]$ returns an array with the elements of A where L is `True` (does not preserve shape).
- * If I is an array of integers, $A[I]$ returns an array with the elements of A at indices I (does not preserve shape).

Pandas

- * Library for (tabular) data analysis.
 - Special types for 1-d (`Series`) and 2-d (`DataFrame`) data.
 - General indexing, selection, alignment, grouping, aggregation.
- * Documentation: `pandas.pydata.org`
- * *Beware*: Pandas data types do not behave as you expect.