List of useful functions

There are four libraries imported for this analysis: `matplotlib`, `numpy`, `pandas`.

- `numpy` is used for performing mathematical operations on arrays
- `matplotlib` allows graphs to be plotted
- `pandas` provides an interface for manipulating datasets

The list below gives some functions that you will find useful, although you can use other functions from the libraries given. Clicking on the heading of each function will take you to the function's main documentation page.

numpy

`numpy.sqrt(n)`

Description	Example

Return the square root of `n`	`> a = numpy.array([1, 4, 9])`
	`> numpy.sqrt(a)`
	`numpy.array([1, 2, 3])`

`numpy.mean(data)`

Description	Example

Return the mean of `data`	`> a = numpy.array([1, 4, 9, 3])`
	`> numpy.mean(a)`
	`4.25`

`numpy.sum(data)`

Description	Example

Sum all elements in `data`	`> a = numpy.array([1, 4, 9])`
	`> numpy.sum(a)`
	`14`

`numpy.minimum(data1, data2)`

Description	Example

Compare two datasets and returns an array containing the element-wise minima	`> a = numpy.array([1, 4, 9])`
	`> b = numpy.array([9, 4, 1])`
	`> numpy.minimum(a, b)`
	`numpy.array([1, 4, 1])`

`numpy.maximum(data1, data2)`

Description	Example

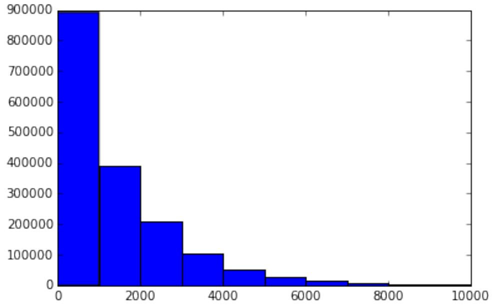
Compare two datasets and returns an array containing the element-wise maxima	`> a = numpy.array([1, 4, 9])`
	`> b = numpy.array([9, 4, 1])`
	`> numpy.maximum(a, b)`
	`numpy.array([9, 4, 9])`

matplotlib

Functions in `matplotlib` are accessed through `pyplot`.

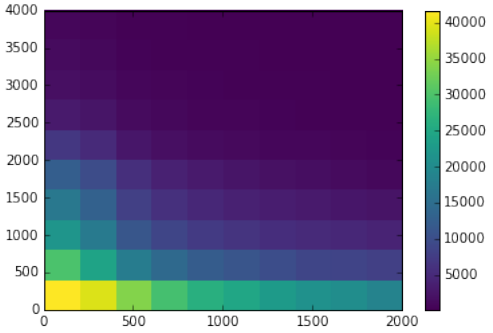
`pyplot.hist(data, n, [x1, x2])`

Description	Example

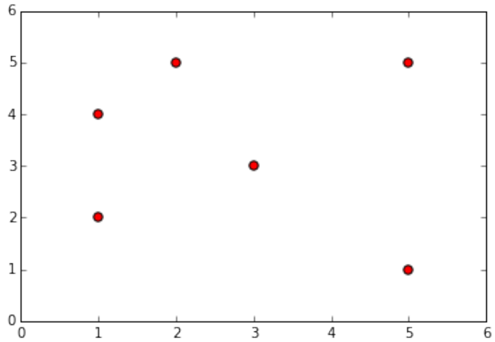
Plots a histogram of given data in `n` equally spaced bins over the range `x1` to `x2`	`> pyplot.hist(`
	`      data.H1_PX, 10, [0, 10000]`
	`  )`
	

`pyplot.hist2d(data1, data2, n, [[x1, x2], [y1, y2]])`

Description	Example

Plot a 2D histogram from two datasets, with n^2 bins equally spaced between `x1` and `x2` in `x` and `y1` and `y2` in `y`	`> pyplot.hist2d(`
	`      data.H1_PX, data.H2_PX,`
	`      10, [[0, 2000], [0, 4000]]`
	`  )`
	`> pyplot.colorbar()`
	

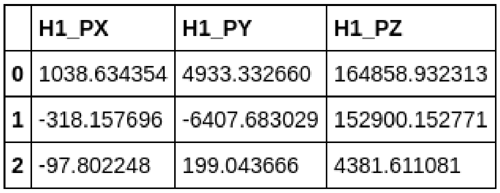
`pyplot.scatter(x, y, size, color)`

<table>
<tr><th>Description</th><th>Example</th></tr>
<tr><td>Plot a scatter plot of <code>x</code> vs <code>y</code> where each point has area <code>size</code> and colour <code>color</code></td><td><code>&gt; pyplot.scatter(
      [1, 2, 5, 1, 3, 5],
      [2, 5, 1, 4, 3, 5],
      40, "red")
)</code>
 </td></tr>
</table>

pandas

The following pandas functions have to be applied to an exisiting `DataFrame` object, see the example analysis for a demonstration.

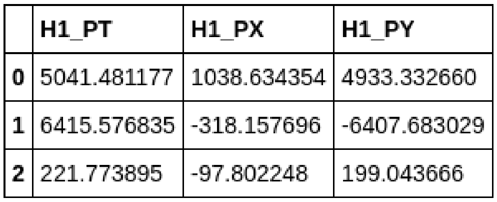
`df.head(n)`

<table>
<tr><th>Description</th><th>Example</th></tr>
<tr><td>Produces a table of the first <code>n</code> rows of data in the structure</td><td><code>&gt; df.head(3)</code>
<table><tr><th></th><th>H1_PX</th><th>H1_PY</th><th>H1_PZ</th></tr><tr><td><b>0</b></td><td>1038.634354</td><td>4933.332660</td><td>164858.932313</td></tr><tr><td><b>1</b></td><td>-318.157696</td><td>-6407.683029</td><td>152900.152771</td></tr><tr><td><b>2</b></td><td>-97.802248</td><td>199.043666</td><td>4381.611081</td></tr></table></td></tr>
</table>

`df.eval(expression)`

<table>
<tr><th>Description</th><th>Example</th></tr>
<tr><td>Evaluate an expression in the context of the DataFrame</td><td>

```
> df['H1_PT'] = data.eval(
      'H1_PT = sqrt(H1_PX**2 +
H1_PY**2)'
  )
df.head(3)
```



	H1_PT	H1_PX	H1_PY
0	5041.481177	1038.634354	4933.332660
1	6415.576835	-318.157696	-6407.683029
2	221.773895	-97.802248	199.043666

</td></tr>
</table>

`df.min()`

<table>
<tr><th>Description</th><th>Example</th></tr>
<tr><td>Return the minimum value for each column in a dataframe</td><td>

```
> df.min()
H1_PX             -122475.373002
H1_PY             -613485.901808
H1_PZ                1420.725768
dtype: float64
```

</td></tr>
</table>

`df.max()`

<table>
<tr><th>Description</th><th>Example</th></tr>
<tr><td>Return the maximum value for each column in a dataframe</td><td>

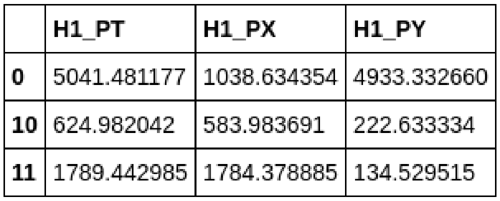
```
> df.max()
H1_PX              2.411849e+05
H1_PY              1.748288e+05
H1_PZ              1.998913e+07
dtype: float64
```

</td></tr>
</table>

`df.query(expression)`

<table>
<tr><th>Description</th><th>Example</th></tr>
<tr><td>Select part of a DataFrame provided <code>expression</code> evaluates to true</td><td>

```
> df_2 = df.query("H1_PX > 0")
df_2.head(3)
```



	H1_PT	H1_PX	H1_PY
0	5041.481177	1038.634354	4933.332660
10	624.982042	583.983691	222.633334
11	1789.442985	1784.378885	134.529515

</td></tr>
</table>