additional materials for SET 5

kacper.topolnicki@uj.edu.pl

The `set_5.py` script contains additional materials for the fifth set of exercises. Some materials were taken from:

- https://numpy.org/doc/stable/
- https://docs.scipy.org/doc/scipy/reference/stats.html
- https://stackabuse.com/download-files-with-python/
- https://en.wikipedia.org/wiki/Shapiro%E2%80%93Wilk_test
- https://en.wikipedia.org/wiki/Order_statistic

Obtaining the data

The data used in this tutorial will be downloaded directly from the web. The first thing that we need to do is create a temporary file into which we will download the necessary information. Typically, there is a special directory on your system designated for temporary files. The lifetime of these files is short and usually ends with when the computer is rebooted.

We will be using: [set_5.py line: 42]

```
import tempfile
```

to create a temporary file in a system independent way in [set_5.py line: 46]

```
import os
datafile = os.path.join(tempfile.mkdtemp() , "set_5_data.csv")
print("datafile : " , datafile)
```

The `os.path.join` function is a system agnostic method to join file paths and `tempfile.mkdtemp` returns the path of a newly created directory for temporary files (a subdirectory of the temporary file directory).

Next we will be using: [set_5.py line: 60]

```
import urllib.request
```

to handle the networking and download the file into `datafile`: [set_5.py line: 64]

```
url = "https://gist.githubusercontent.com/nstokoe/7d4717e96c21b8ad04ec91f361b000cb/raw/bf95a
urllib.request.urlretrieve(url , datafile)
```

Parsing the CSV file

This time we will use the `pandas` library to read in the data contained in the downloaded file: [set_5.py line: 85]

```
import pandas
```

There are other ways of doing this and we covered some of those previously. We should start looking at the `pandas` library and parsing CSV files is a good start.

Parsing is done by using `pandas_read_csv`: [set_5.py line: 89]

```
data = pandas.read_csv(datafile)
print("type(data) : " , type(data))
print("data : ")
print(data)
```

and reveals a `DataFrame` object.

We will only be using the last columns of this data that contains information on male height: [set_5.py line: 102]

```
print("data[\"Height\"] : ")
print(data["Height"])
```

Let's get back to the more familiar `numpy` library: [set_5.py line: 113]

```
datanp = data["Height"].to_numpy()
print("datanp.dtype : " , datanp.dtype)
print("datanp.shape : " , datanp.shape)
```

The variable `datanp` will contain our numpy data (notice the lack of `import numpy`).

Drawing the histogram

Let's see what we are dealing with by drawing a histogram: [set_5.py line: 128]

```
import matplotlib.pyplot as plt
plt.title("male height")
plt.xlabel("height")
plt.xlabel("PDF")
plt.hist(datanp , density = True)
plt.show()
```

At first glance, we are dealing with a normal-ish looking distribution. Let's have a better look.

Normal probability plot

The normal probability plot is a way to visually determine if the data was drawn from a given distribution. The idea is simple, for a given probability distribution:

1. take the measurement values $x_1, x_2, \ldots, x_N$
2. sort them and get $x_{(1)}, x_{(2)}, \ldots, x_{(N)}$ where $x_{(1)} \leq x_2 \leq \ldots \leq x_{(N)}$
3. go through the sorted list, for each $x_{(i)}$ estimate the probability p_i that the ordered measurement falls below $x_{(i)}$ (there is a lot of hand waving here, there are different ways to do this)
4. calculate quantile function values $Q(p_1), Q(p_2), \ldots, Q_{p_N}$,
5. plot the points $(Q(p_i), x_{(i)})$ for $i = 1, 2, \ldots, N$

If the measurement points in the data were indeed drawn from the given distribution then we can expect a linear plot.

You can use the `probplot` function from the `stats` library to do all the work. For details on step 3 see the **documentation**. By default no plot is created and only the plot points are calculated. If we pass the `matplotlib.pyplot` library (`plt` is the shorthand we are using for this library) as the optional argument then a plot will be drawn: [set_5.py line: 160]

```
from scipy import stats
npp = stats.probplot(datanp , plot = plt)
plt.show()
```

Shapiro - Wilk test

The null hypothesis is that the data $x_0, x_1, \ldots, x_N$ was drawn from a normally distributed population. The test statistic:

$$W = \frac{\left(\sum_{i=1}^{N} a_i x_{(i)}\right)^2}{\sum_{i=1}^{N} (x_i - \bar{x})^2}$$

where $x_{(1)}, x_{(2)}, \ldots, x_{(N)}$ is an ordered list of mesurments such that $x_{(1)} \leq x_2 \leq \ldots \leq x_{(N)}$ and a_i are coefficients calculated calculated from the **order statistics**. For more details see the entry on **Wikipedia**.

The `scipy` library has an implementation of this test: [set_5.py line: 184]

```
sw = stats.shapiro(datanp)
print("sw : " , sw)
print("sw.statistic : " , sw.statistic)
print("sw.pvalue : " , sw.pvalue)
datanorm = stats.norm.rvs(size = 10000)
swnorm = stats.shapiro(datanorm)
print("swnorm : " , swnorm)
```

```
print("swnorm.statistic : " , swnorm.statistic)
print("swnorm.pvalue : " , swnorm.pvalue)
```

The value of the test statistic and p-value are printed to standard output for the male height data as well as for a numbers sampled directly from a normal distribution.