PYU33C01, S. Hutzler, 2024

Assignment 2: Least Square Fits for Modelling Population Growth

The aim of this exercise is a numerical analysis of the growth of human population since 1750. Is its variation exponential?

The two files *population-gapminder.dat* and *population-census-bureau.dat*, available on Blackboard, contain data for the total world population (in millions) for the periods 1750-1940 (Gapminder) and 1950-2016 (US Census Bureau), respectively. (For further information on such data see `https://en.wikipedia.org/wiki/Estimates_of_historical_world_population`).

- Plot both sets of data into the same graph using a Python script and label the graph appropriately (axes labels, title, key).

To probe for exponential growth it is best to plot the *natural logarithm* of the data as a function of time. (One can also use a logarithmic scale for the population axis, but you should *not* do this here, in order to simplify the interpretation of the least-square fits that follow.)

- Plot the logarithm of population as a function of time, again for both data sets, in one figure.

If the population rises exponentially, i.e. according to

$$n(t) = n_0 \exp[\lambda(t - t_0)], \tag{1}$$

with $n(t_0) = n_0$, time offset t_0, and growth constant λ, then $\ln n(t)$ varies linearly with time,

$$\ln n(t) = \ln n_0 + \lambda(t - t_0). \tag{2}$$

Setting

$$a = \ln n_0 - \lambda t_0; \quad b = \lambda; \tag{3}$$

we can identify the functional form of $\ln n(t)$ as that of a straight line, i.e.

$$\ln n(t) = a + bt \tag{4}$$

We are now ready to analyse our population data set by performing a *linear regression analysis*, i.e. fitting a straight line to $\ln n(t)$ as a function of time t.

- Study your $\ln n(t)$ graph and determine two time ranges over which the data is reasonably well described by a straight line.
- Write Python code to fit straight lines for these ranges (the starting point of such a range being t_0) and report the values for their slopes b and offsets a. (See below for comments on fitting in Python.)
- From your values for a and b compute and write down the values for the fit parameters λ and n_0 of Eqn(1). Note that we determine n_0 from the fit and not directly from the data, so as not to give too much relevance to the actual value of $n(t_0)$.
- Using the values that you determined for λ and n_0 show Eqn(1) as (two) lines in your first graph.
- How do you assess the *current* evolution of population, based on your results?
- Finally, specify whether you used any AI software, e.g. ChatGBT, in any parts of your code writing. You are not allowed to use AI for the writing of the report.

Performing least square fits using Python

- You can use a scipy library to perform a non-linear least square fit of your data. You require the following commands:

```
import scipy.optimize as optimization
import numpy
```

To load the data into your code you can use

```
year, population = np.loadtxt("DATA.dat", skiprows=1, unpack=True)
```

where

```
skiprows=1
```

skips the first row of the input data and

```
unpack=True
```

returns a pair of arrays (here called: year, population), one for each column.

You can choose a selection of the data that you want to fit by defining for example

```
firstLine = np.arange(0, 10)
```

which selects the data for the year recored in rows 0 to 10 of your file, when used in the form

```
 year[firstLine]
```

or

```
 population[firstLine]
```

The fitting function (containing two or more fit parameters) is defined by

```
def func(x, a,b):
...
     return ...
```

Fitting of the selected data is done via

```
fitparameter=optimization.curve_fit(func, year[firstLine],
cases[firstLine])[0]
```

Placing [0] at the end specifies that only the values of the fitparameters are stored (and not also the covariance matrix of the fit).

The fit parameters returned by the fit are accessed via

```
par1=fitparameter[0]
par2=fitparameter[1]
```

(For more info on fitting simply google "Least-squares fitting in Python" or check out `https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.curve_fit.html` for a very detailed description of the fit package.)

- Submit a report (as pdf) **via Blackboard** which details your findings, including relevant figures. In addition submit your Python code as a *.py* file which should also print your name and student number onto the screen when executed.

submission deadline: Wednesday, October 9, 2024