# Analytics Tool & Visualisation Exercises

September 7, 2017

Before you start:
- create a suitable directory for this exercise
- download the zipped material from `http://www.physik.uzh.ch/~python/python/lecture7/`

## Exercise 1: Matrix Calculation

**Targeted libraries:** `numpy`
You can find in the file `matrix.txt` a $100 \times 100$ matrix. Read it in and use SciPy to calculate the determinant, the eigenvalues and -vectors. Since it is a positive definite symmetric matrix you can also try the dedicated functions for such matrices, *i.e.* `eigh` and `eigvalsh`, in particular in terms of CPU timing. You can also test matrix addition and multiplication using the matrix and its inverse and some other matrix operations you are interested in as well as the Cholesky and QR and SVD (Singular value decomposition) decomposition. Use `%timeit` to monitor the performance.

## Exercise 2: Zebra Tracking

**Targeted libraries:** `pandas`,`numpy`, `matplotlib`, `datetime`
This exericise uses the same data as exercise 4 from the session on data structures.
You find the position data of several zebras (Plain zebras, Equus quagga burchelli) in northern Botswana (Source: movebank.org) in the file `ZebraBotswana.txt`.
The data consists of the date and time of the measurement in unix format (*i.e.* seconds since 1970-1-1), the longitude and latitude of the measured position (in degrees) and the number of the corresponding Zebra.

Read in the data and plot for each zebra its path.

Create two plots showing for each zebra the measured longitude and latitude, respectively, as a function of time. What can you learn about the migration pattern?

## Exercise 3: Financial Analysis – Visualisation & Monte-Carlo Simulation

**Targeted libraries:** `pandas`,`numpy`, `matplotlib`, `datetime`, `scipy`
This exericise uses the same data as exercise 5 from the session on data structures.
The file `FinanceData.csv` contains as time series the adjusted closing prices of the six highest-weighted stocks (IBM, Goldman Sachs, 3M, Boeing, Chevron and United Technologies; VISA is not included due to its initial public offering only in 2008) in the Dow Jones Industrial (DJI) index plus the index itself. The first row is the date in unix format.

Read in the data and plot the different stocks and the DJI index.

Calculate the daily increase or decrease of their value in percent and plot the 30-day rolling mean and standard deviation.

Plot the distribution of the daily changes. *Tip:* Use the `histogram` function from NumPy and – to plot – the `bar` function from Matplotlib.

Use the rolling mean and standard deviation determined above to sample with the `scipy.stats` package Gaussian distributed changes to simulate the movement of the DJI. Does it look different compared to using the mean and standard deviation considered over the full time range of the data?

## Exercise 4: How Cold is the Universe?

You can also try out some data analysis you have done with other tools (*e.g.* MATLAB, R) and try to figure out if it is possible in Python with the introduced libraries. If you have a lack of data, the file `Cobe.txt` contains data from the COBE satellite (more info about COBE at `http://lambda.gsfc.nasa.gov/product/cobe/`). It shows the spectrum of the cosmic microwave background. The first row gives the frequency (actually the inverse of the wavelength in 1/cm), the second row the spectrum in MJy/sr (MJy: Mega-Jansky, 1 Jy= $10^{-26}$ W/Hz·m$^2$ ; sr: Steradian), so it is a measure of the spectral flux per solid angle. The third row shows the uncertainty on the spectrum in kJy/sr.

Use the `scipy.optimize.leastsq` to perform a least-square fit of the data. The function that should describes the data is the Planck law $f(x) = A_0 \cdot x^3/(\exp(1.439x/T) - 1)$ where $x$ is the frequency in 1/cm. $A_0$ and $T$ are the fit parameters, where $A_0$ is the amplitude and $T$ the temperature of the universe. The factor 1.439 K·cm comes from $h \cdot c/k_B$ in the chosen unit frame. So you can determine from the fit how colde the universe is.

## Exercise 5: Visualisation

This exercise focusses on advanced visualisation with matplotlib. This exercise's data is pseudo-data coming from astrophysical observations and is used to determine the density parameters of energy and matter in our universe. A density parameter is the density fraction of a certain kind of energy and matter with respect to the so-called critical density. The sum of the density parameters, called the total density $\Omega_{\text{tot}}$, allows to make statements about the shape of the universe. If the total density is larger than 1, the universe is closed, it is open if $\Omega_{\text{tot}} < 1$ and it is flat if this quantity is equal to 1. We consider only the density parameter of dark energy, $\Omega_\Lambda$, and the parameter of the remaining matter and energy (*i.e.* usual matter and energy and dark matter), $\Omega_m$. The available data comes from measurements in association with the red-shift of super-novae (SNe Type Ia), the cosmic microwave background (CMB) and the baryonic acoustic oscillation (BAO). The files `SNe_LH.txt`, `CMB_LH.txt`, and `BAO_LH.txt` contain the measured likelihoods in the $\Omega_m$-$\Omega_\Lambda$ plane where the first column shows the $\Omega_m$ values and the first row the $\Omega_\Lambda$ values. The entry at $[0, 0]$ can be discarded.

Display the 1-, 2- and 3$\sigma$-confidence regions (68.3%, 95.4% and 99.7%) of each likelihood as well as of the combined likelihood (*i.e.* the product of the likelihoods since the measurements are independent). What are the most probable values of $\Omega_m$ and $\Omega_\Lambda$? Thus, what is the geometry of the universe?
*Tip:* Use `contourf` or `contour` to plot and `LinearSegmentedColormap` from `matplotlib.colors` to specify the colormaps. The likelihood is proportional to the probability.
*Remark:* Since the data is only approximated with respect to the actual data, the obtained result may also differ.

## Exercise 6: Maximum-Likelihood Fitting

**Targeted libraries:** `scipy` and `matplotlib`
The goal of this exercise is to develop an algorithm to perform maximum-likelihood fitting of data with Python.77 The file `DJI_daily.txt` contains the daily returns of the Dow Jones Index since the start of 2000.

Perform a maximum-likelihood fit to this data. It's up to you to perform an normal or abinned maximum-likelihood fit. You can start by using a Gaussian distribution (`scipy.stats.gauss`) and later use other suitable distributions like the Student-t distribution (`scipy.stats.t`).

Plot the distribution of data using histograms or errobars and the fitted distribution.

What happens when you restrict the data to a shorter period of time *e.g.* 200 days? The data is chronologicaly ordered.

*Tip:* For numerical reasons using the negative log-likelihood function

$$_{log}\mathcal{L} = -\sum_i \log(f(x_i|p))$$

and perform a minimisation with respect to the parameters $p$ is a way to go. $f$ is the distribution and $x_i$ are the observations.
The ordering of parameters in the distributions from the `scipy.stats` package are not necessarily intuitve. So please consult the documentation.

## Exercise 7: Multidimensional Interpolation & Terrain Models

**Targeted libraries:** `scipy` and `matplotlib`
In this exercise we want to apply an easy way offered by `scipy` to interpolate multidimensional data. The file `zrh_terrain.txt` contain about 8'000 rows of sampled terrain data in the area of Zurich. The rows are of the form latitude,longitude, and elevation (in metres).

Use the `griddata` function in the `scipy.interpolate` library to create a highly granular terrain model.
*Tip:* Study the doc string of the function in particular what the interpolation domain can be.

Visualise the resulting model. Try different plot types (heatmap, different 3D plots) offered by `matplotlib` or other visualisation packages.

Test the different interpolation methods in `griddata` and see how the models differ due to the different methods.

*Optional:* Create a very simple waterfall-type hydrology model to leverage the terrain model to simulate flooding

## Project 1: NYC Taxi

**Targeted libraries:** `csv`, `requests`, `datetime`, `bokeh`
New York's TLC (Taxi and Limousin Commission) publishes since 2010 data about every single trip of the yellow cabs – since their initialisation in 2013 also of the green boro taxis. The data is available as CSV files (one per month and vehicle category) from `http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml`. The data contain various information (time and date, length of the trip, pick-up/drop-off locations, payment method, tip). Analyse for example the taxi rides from and to the Bronx Zoo (Entrance coordinates: 40.843° N / 73.877° W) according to the time of the day, the weekdays or seasonal effects. Investigate for example differences in the tipping behaviour between the boroughs or as a function of the time of day and the weekday or as seasonal effects.
*Hints:* Download only a sample set to test your code and run the code over the whole data set online. If you need geographical information about coordinates (*e.g.* a gazeetter), you can use the `geopy` module.

## Project 2: SNB API

**Targeted libraries:** `pandas`, `requests`, `datetime`, Visualisation
The Swiss National Bank (SNB) offers a large amount of economical and financial information. The corresponding data sheets are downloadable from `https://data.snb.ch/en` in CSV or Excel format. Unfortunately there is no API allowing to load the data directly from a Python script. Try to build such an API allowing to retrieve the data specified by the corresponding id code (*cf.* snbnomu for the

number of banknotes and coins in circulation) and provide them as a timeseries. You can also build with object-oriented functionalities an SNB data specific container allowing the storing of metadata. On top of that try to use the API to build a data inspector allowing to get a first glance at the data.
*Hints:* You can find details about how the data can be retrieved as CSV at `https://data.snb.ch/en/help`. The CUBE id is the mentioned id code. In addition You can find details about how the data can be retrieved as CSV.

## Project 3: News Headline API

**Targeted libraries:** `pandas`, `requests`, `BeautifulSoup`, Visualisation & Databases
The Reuters News archive is accessible for each day via `http://www.reuters.com/resources/archive/us/YYYYMMDD.html` where `YYYYMMDD` represents the corresponding date (*e.g* `http://www.reuters.com/resources/archive/us/20170701.html`).

Use the `requests` functionalities to download the archive page of the corresponding day and use `BeautifulSoup` to extract the title and further informations (link, time).

Finally load the information into a database and run the algorithm over the range of several days.

Use the retrieved data to *e.g.* how the frequency articles about Donald Trump or Hilary Clinton have developed since beginning of 2015.