Why you never should and certainly never have to use spreadsheets

Clara Kofler (originally written for the RV newspaper *Distribution*)

For almost no experimentalist a week goes by without data analysis. This is no one's favorite job but it has to be done, and it has to be done well; otherwise, what is the point of recording the data at all?

There are obviously many means to that end - all roads lead to Rome. However, some roads will be more painful than others, some roads will lead to more then 20% of papers featuring gene names containing wrong data [1] and some roads will lead to international humiliation of the Austrian government [2].

To ensure you do not have to go down these roads (and also because it is just more convenient in every way), here a short tutorial on using python to do your data analysis.

Setting up your system:

The easiest way to do data analysis with python is by using Jupyter, a web interface for interactive computing. We will use conda to set this up. If you already have a running conda and jupyterlab, please skip the following steps.

Installing Miniconda

If you already have Miniconda or another version of conda, please skip this paragraph.

To install Miniconda, follow the steps for your operating system on their website: `https://docs.anaconda.com/miniconda/miniconda-install/`

After doing this, Windows users should find an Anaconda Prompt application on their system, opening up a command line window. Linux and Mac OS users should see a new (base) prefix in their terminal after restarting it.

Installing Jupyterlab in a new environment

Please follow the following commands in your conda prompt/terminal to set up a new virtual environment and install the needed packages:

```
    $ conda create -n data-analysis python
    $ conda activate data-analysis
```

This should create a new environment named *data-analysis* and activate it. You should see the (base) change to (data-analysis) as you change environment. You can deactivate the environment with `conda deactivate`, and activate it again any time.
Then install the packages we need in this environment:

```
    $ conda install -c conda-forge jupyterlab
    $ conda install -c anaconda git
    $ conda install -c conda-forge pandas
    $ conda install scipy matplotlib tabulate
```

Image by ChatGPT

Useful bash commands

ls	list contents
cd <dir>	move to <dir>
mkdir <dir>	create new folder
rm <file>	delete <file>
mv <thg> <plce>	move things
cp <thg> <plce>	copy things
pwd	where am I?

Cloning example data and notebook

I have created a starting notebook and shared it on github, please clone it to a folder of your choice using:

```
    $ git clone https://github.com/kofcla/data-analysis-tutorial.git
```

Now open jupyter by typing

```
    $ jupyter lab
```

(you might need to copy the link into your browser).
There, open **data-analysis-tutorial.jpynb**.

References

1. Lewis, D. Autocorrect errors in Excel still creating genomics headache. *Nature.* ISSN: 1476-4687. `http://dx.doi.org/10.1038/d41586-021-02211-4` (Aug. 2021).
2. *Babler ist nach Auszählungspanne statt Doskozil neuer SPÖ-Chef* `https://www.derstandard.at/story/3000000173335/babler-ist-nun-doch-chef-der-spoe`. 2023.

The basics of data visualization and analysis

How to use this page: The bread and butter of learning programming is to try out things and play around with it. We cannot do that for you unfortunately. But here we want to help you get started.

The example notebook `data-analysis-example.ipynb` had examples of creating plots and doing data analysis tasks similar to what you encounter in lab courses. We provide explanations and small tasks for you. Please note that we show one way to do it, but it is by far not the only way. So, have a look through the notebook, do the tasks, play around with the code!

The rest of this page contains examples similar to the notebook and suggestions for the packages to use for certain tasks. This should give you a quick overview of what we cover in the notebook.

Loading data

To load your data we advise you to use `numpy`. It is specialized for dealing with arrays, so perfect for handling numerical data. It also has a lot of math functionalities that will prove useful. If you want to look at your data in the form of a table you should use `pandas`. It has the so-called DataFrames that are very convenient to work with.

```
#basic example of numpy data loading
data = np.loadtxt('my_filename.csv', delimiter=',')

#reading as csv file into pandas dataframe
df = pandas.read_csv('my_filename.csv')
```

Creating simple plots

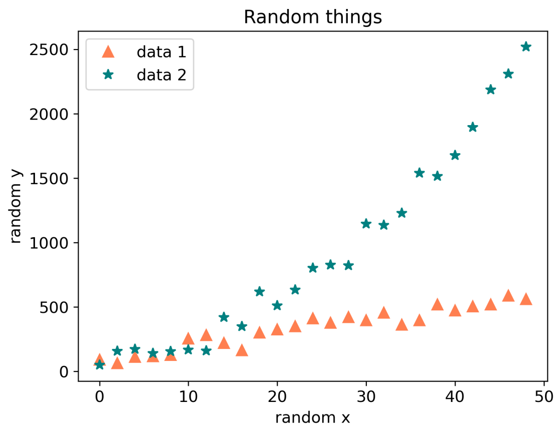
There are a million ways to plot something in python, we decided on using `matplotlib`. This package provides a massive amount of data visualization possibilities. Sometimes it is a bit unintuitive but it gets the job done.

```
#basic plotting

plt.plot(x1, y, marker='^', ls='', color='coral',
    label='data 1')
plt.plot(x1, z, marker='*', ls='', color='teal',
    label='data 2')

plt.xlabel('random x')
plt.ylabel('random y')
plt.title('Random things')
plt.legend()

plt.show()
```



Fitting data

For data fitting, we recommend `scipy`. This package contains many analysis tools, including an optimization module that offers different functions for data fits. Here the important thing is to provide proper starting values, otherwise, your fits will not converge. For curve fitting `scipy.optimize.curve_fit` works well.
Pro tip: If you need to use scientific constants you can import them from `scipy.constants`.

Image processing

For image processing you can use the `scipy.ndimage`. Also `numpy` offers certain helpful functionalities. Depending on your specific task there might already be some more specified python package. You can install it in your environment like we did on the previous page.

Exporting LaTex tables

The package `tabulate` is amazing for creating simple tables. With the command `tabulate(table, headers, tablefmt="latex")` you can create a table to copy into your LaTex document.

Tips for nice figures:

1. Before you start, think about the story of the figure. What do you want to show? Choose the data accordingly.
2. Try to manipulate the raw data as little as possible. If you have to, make sure to document it properly and communicate it transparently.
3. Think about what type of plot best presents your argument.
4. Take a pen and sketch the figure before you start plotting.
5. You want to avoid rescaling. So choose your font and object sizes wisely.
6. Make sure to save the images as vector graphics (SVG or PDF) and with a high enough DPI value (e.g. 300).
7. Check if the saved figure looks like you want it to.
8. If you need to make larger plots containing many sub figures consider assembling them in a vector graphics program like *Inkscape*.