Data analysis in R and Python

Introduction to programming and reproducible research

Ondrej Lexa Vojtěch Janoušek

Institute of Petrology and Structural Geology
Faculty of Science
Albertov 6

lexa@natur.cuni.cz

vojtech.janousek@natur.cuni.cz

2018

What this course is about

This practical course is aimed at senior undergraduate and postgraduate students. It is intended to:

- explain fundamentals of data processing and visualization in geology as well as functioning of computing algorithms in general
- present basics of the R and Python programming languages
- illustrate the usability and versatility of both languages for everyday calculations, as well as for production of publication-quality graphics
- demonstrate examples of using both languages in reproducible research (with certain structural geology and whole-rock geochemistry bias).

An emphasis is on practical examples and exercises.

How to survive and perhaps even enjoy this course

I have been teaching this type of courses for several years now, and one reaction that I get quite often is **fear, panic, and even anger**. A common reaction is: Why do I have to learn all this? How can I do all this programming? And so on...

If you have such questions popping up in your mind, you have to stop and consider a few things before continuing with this course. Education at our department is focused to empirically driven study. It is impossible to get through a master's degree without coming into contact with data. If you are at IPSG, you are automatically committed to an empirically driven education.

Fear, panic and anger...

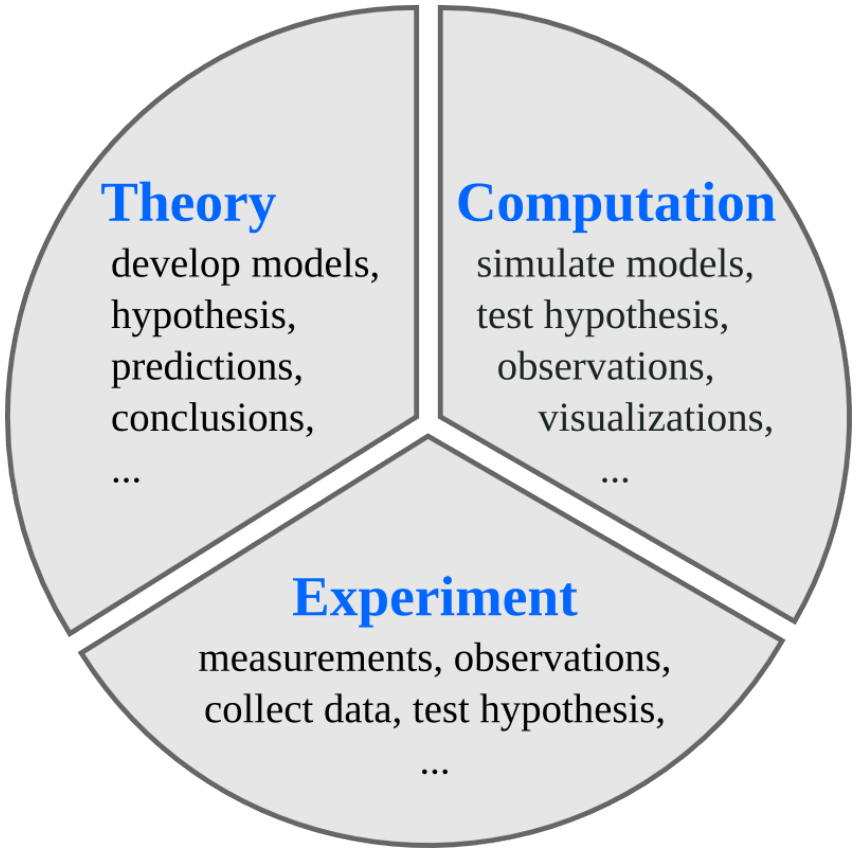
In order to pass this course, you have to understand that you **have to read the lecture notes**, and that **it is not enough** to just passively read these lecture notes. You have to **play with the ideas** by asking yourself questions like "what would happen if..", and then check the answer right there using Python or R. That's the whole point of this approach to teaching data analysis, that you can verify what happens under repeated runs. There is no point in memorizing formulas; focus on developing understanding. The concepts presented here require nothing more than **middle school mathematics**. The ideas are not easy to understand, but simulation is a great way to develop a deeper understanding of the logic of theory.

Outline

The role of computing in science

Science has traditionally been divided into experimental and theoretical disciplines, but during the last several decades computing has emerged as a very important part of science.

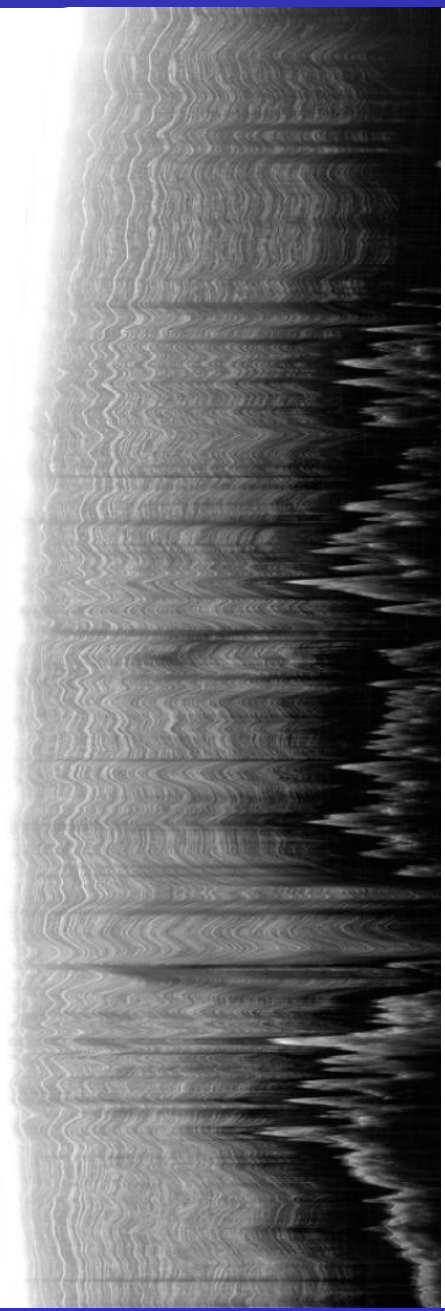
Scientific computing is often closely related to theory, but it also has many characteristics in common with experimental work. It is therefore often viewed as a new third branch of science.



Data Analysis in Earth Sciences

Earth scientists make observations and gather data about natural processes on Earth. They formulate and test hypotheses on the forces that have operated in a certain region to create its structure. They also make predictions about future changes of the planet. All these steps in exploring the system Earth include the acquisition and **analysis of numerical data**.

An earth scientist needs a solid knowledge in statistical and numerical methods to analyze these data, as well as the ability to use **suitable software packages** on a computer.

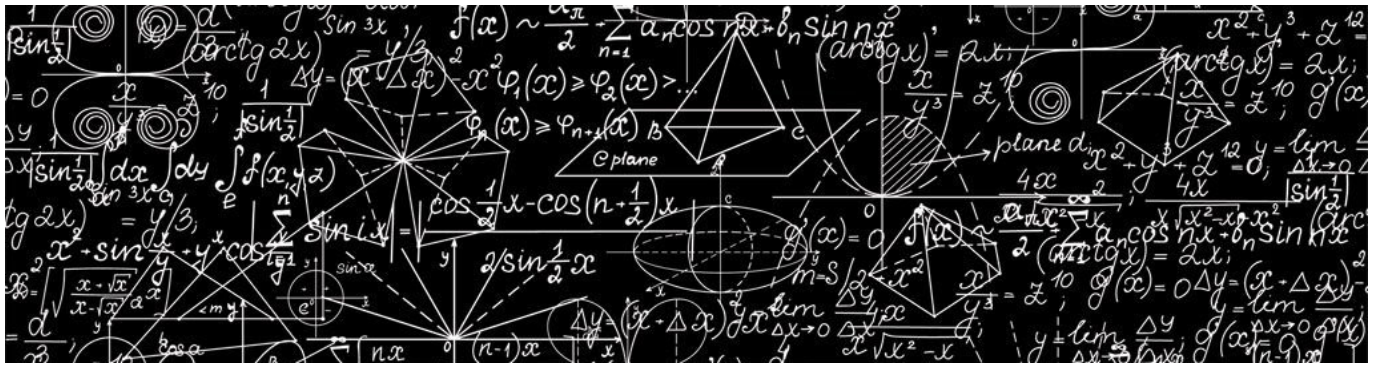


Access to results and methods

In experimental and theoretical sciences there are well established codes of conducts for how **results and methods are published and made available** to other scientists.

In theoretical sciences, derivations, proofs and other results are published in full detail, or made available upon request. Likewise, in experimental sciences, the methods used and the results are published, and all experimental data should be available upon request.

It is considered unscientific to withhold crucial details in a theoretical proof or experimental method, that would hinder other scientists from **replicating and reproducing** the results.



Computational sciences

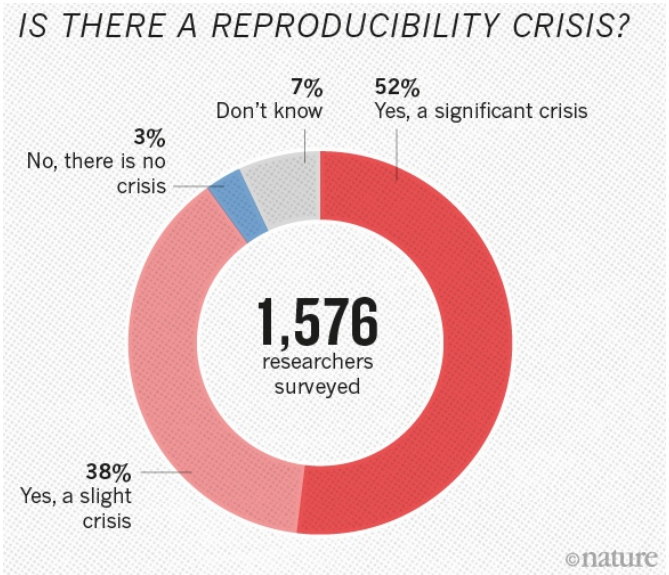
In computational sciences there are not yet any well established guidelines for how source code and generated data should be handled. For example, it is relatively rare that source code used in simulations for published papers are provided to readers, in contrast to the open nature of experimental and theoretical work.

And it is not uncommon that source code for simulation software is withheld and considered a competitive advantage (or unnecessary to publish).

"**Replication is something scientists should be thinking about before they write the paper,**" says Ritu Dhand, the editorial director at Nature.

Reproducibility crisis

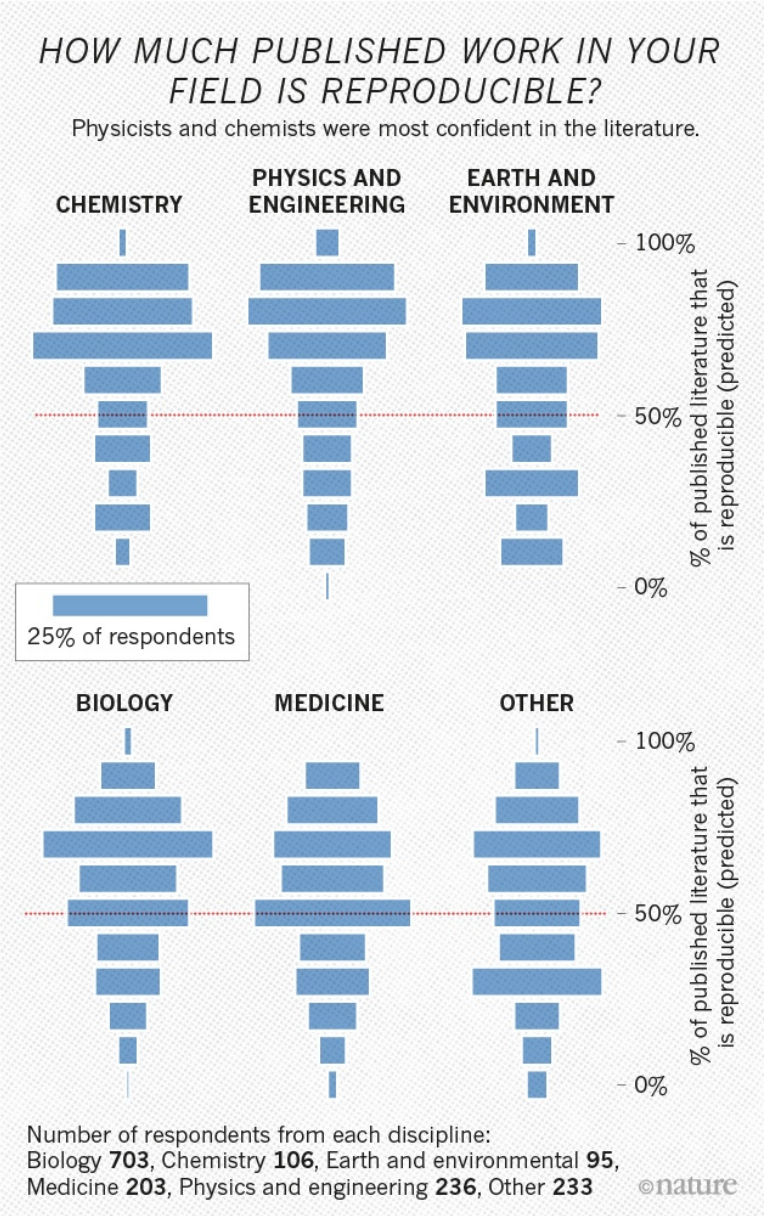
More than 70% of researchers have tried and failed to reproduce another scientist's experiments, and more than half have failed to reproduce their own experiments.
The data reveal sometimes-contradictory attitudes towards reproducibility. Although 52% of those surveyed agree that there is a significant 'crisis' of reproducibility, less than 31% think that failure to reproduce published results means that the result is probably wrong, and most say that they still trust the published literature.



Reproducible Research I.

However, this issue has recently started to attract increasing attention, and a number of editorials in high-profile journals have called for **increased openness in computational sciences**. Some prestigious journals, including Science, have even started to demand of authors to **provide the source code** for simulation software used in publications to readers upon request.

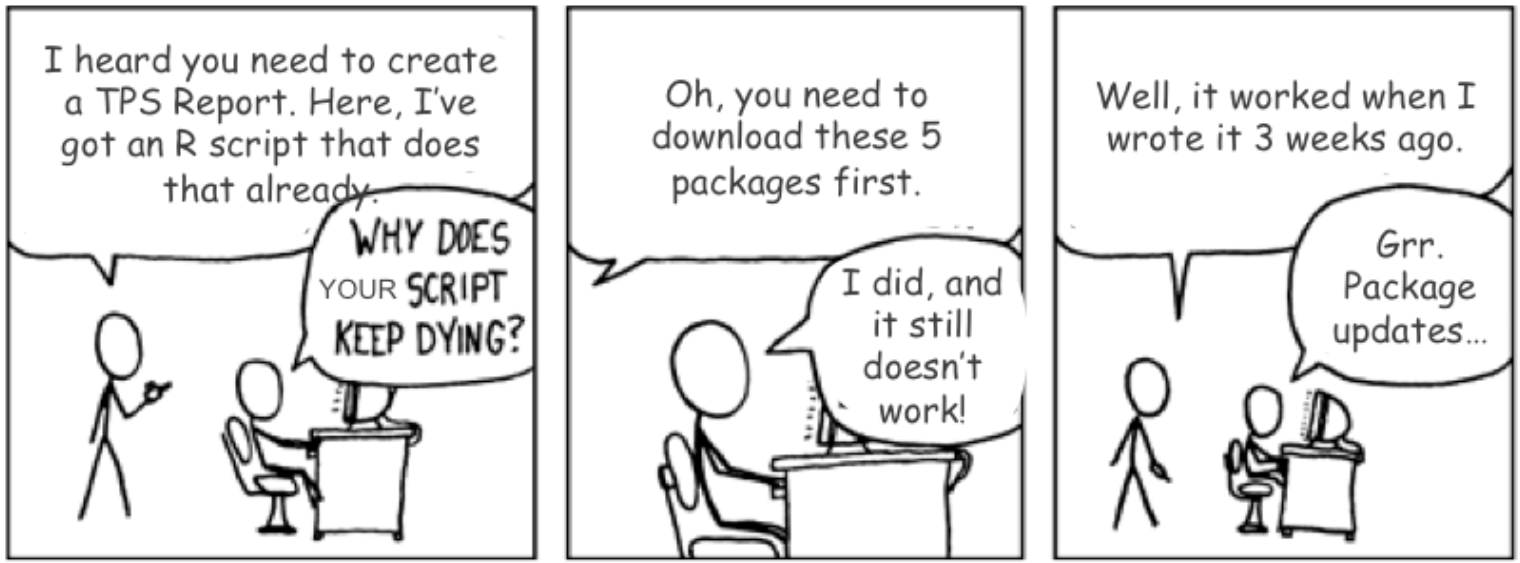
Discussions are also ongoing on how to facilitate distribution of scientific software, for example as supplementary materials to scientific papers.



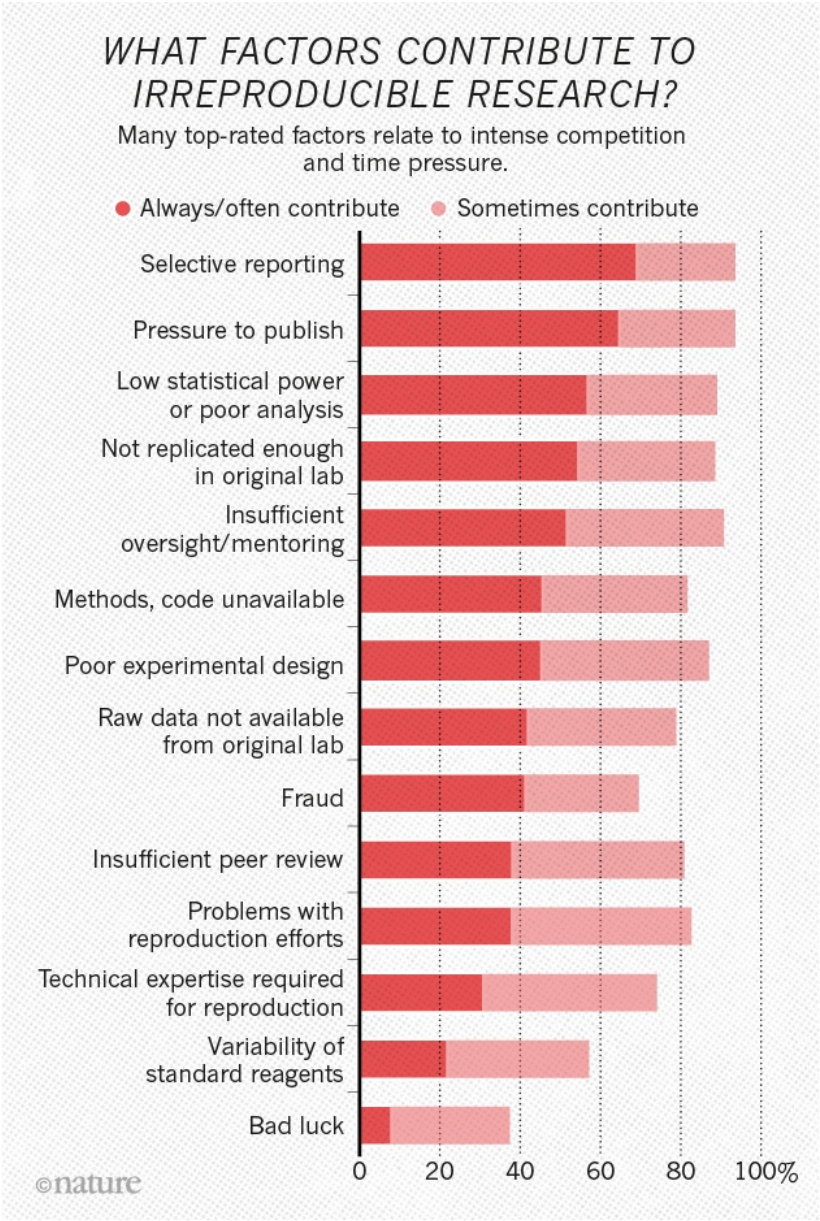
Reproducible Research II.

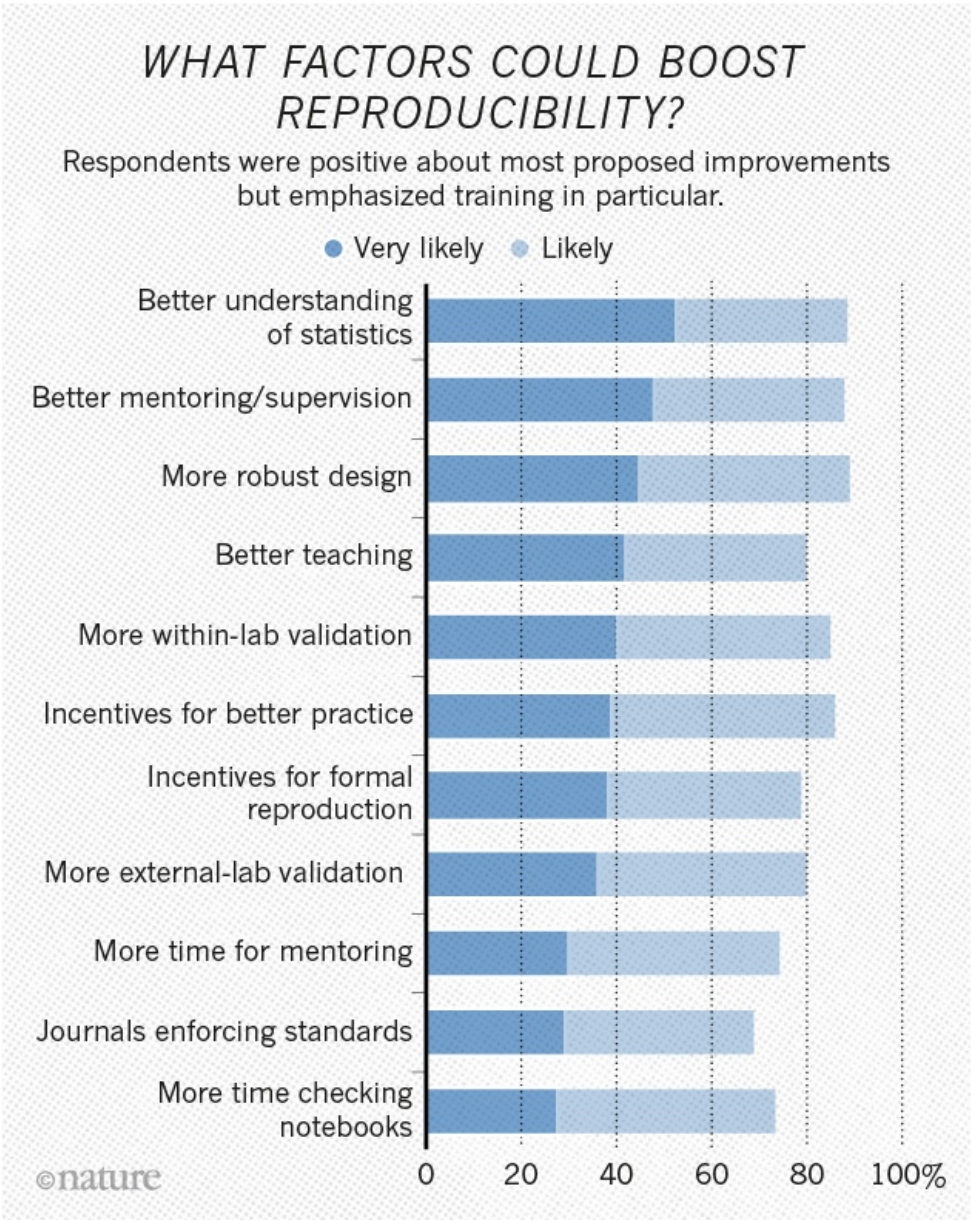
A minimal standard for data analysis and other scientific computations is that they be **reproducible**: that the code and data are assembled in a way so that another group can re-create all of the results (e.g., the figures in a paper).

Our goal is that the students leave the course ready and willing to ensure that all aspects of their computational research (software, data analyses, papers, presentations, posters) are reproducible.



Reproducibility crisis factors





Reproducible Research III.

In this course we will introduces the **Jupyter notebook** project in the context of programming in both **Python and R** for reproducible research and knowledge transfer. After this course a students will have insights into why and how one would use Jupyter notebooks for research, training and/or instruction.

Jupyter notebooks

- have "live" code, which is interactive/modifiable
- allow for easy collaboration on a notebook system
- serve as a scratchpad for both code and notes

Outline

Big Data Analysis in Earth Sciences

The term **Big Data** refers to various forms of large amount of data, which requires specific computing platforms for their analysis. The challenges include:

- capturing
- storage
- searching
- **analysis**
- **visualization**

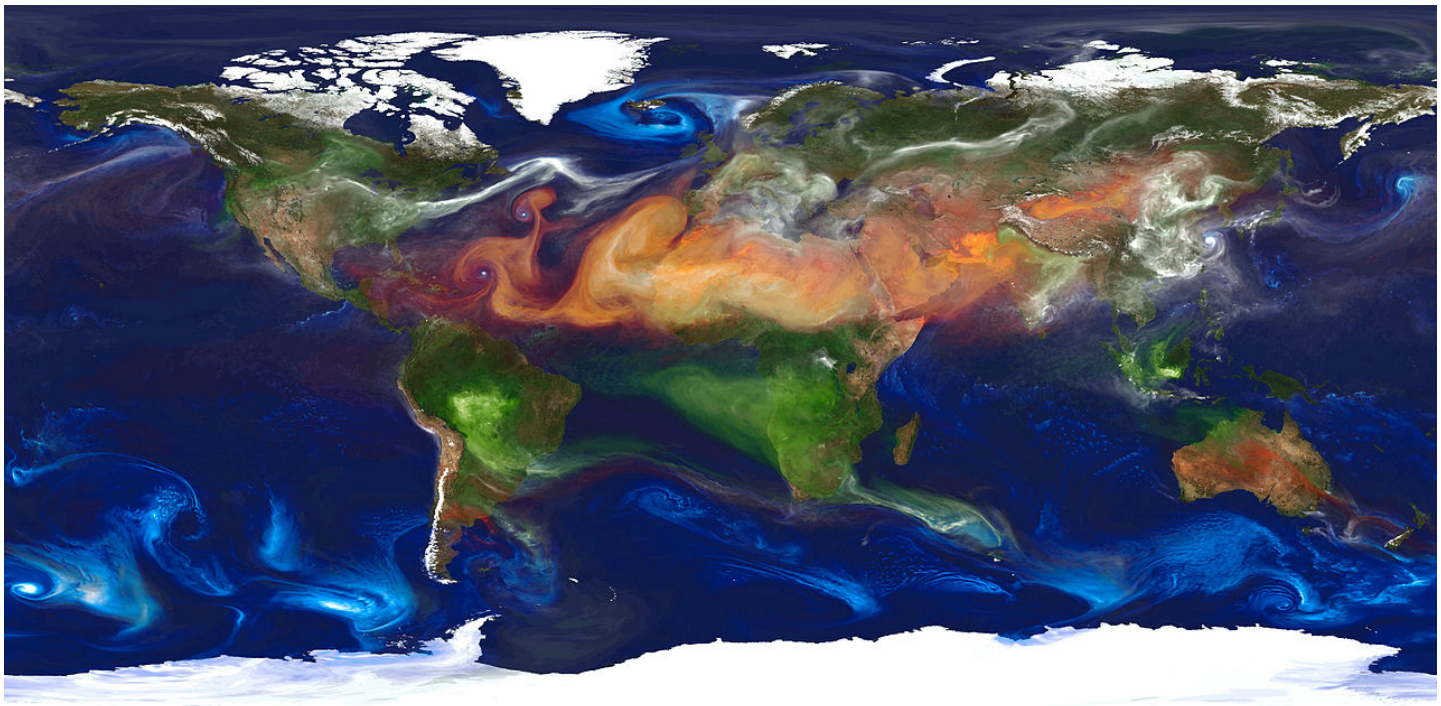
Why Learn To Code?

Earth Scientists Deal with Large Datasets

- Processing and Visualization should be automated

Can make your own tools

- Research is new, so no tools may exist



What Language Should I Learn?

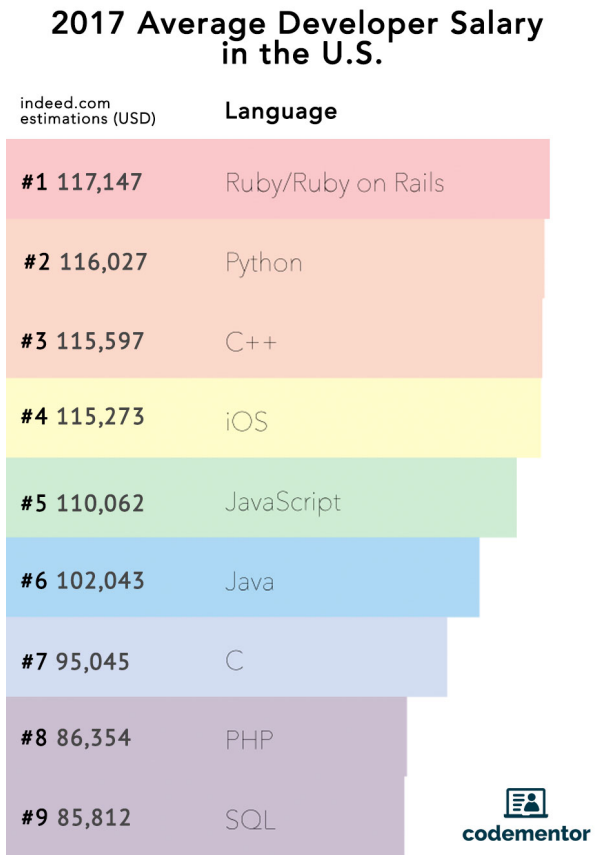
It doesn't really matter

- Once you know one language, you can learn new ones
- Most languages are more similar than different

Commonly used programming languages in Earth sciences

- MATLAB / Octave
- Python
- R
- C / C++
- Fortran
- Java
- Perl
- Mathematica / Maple

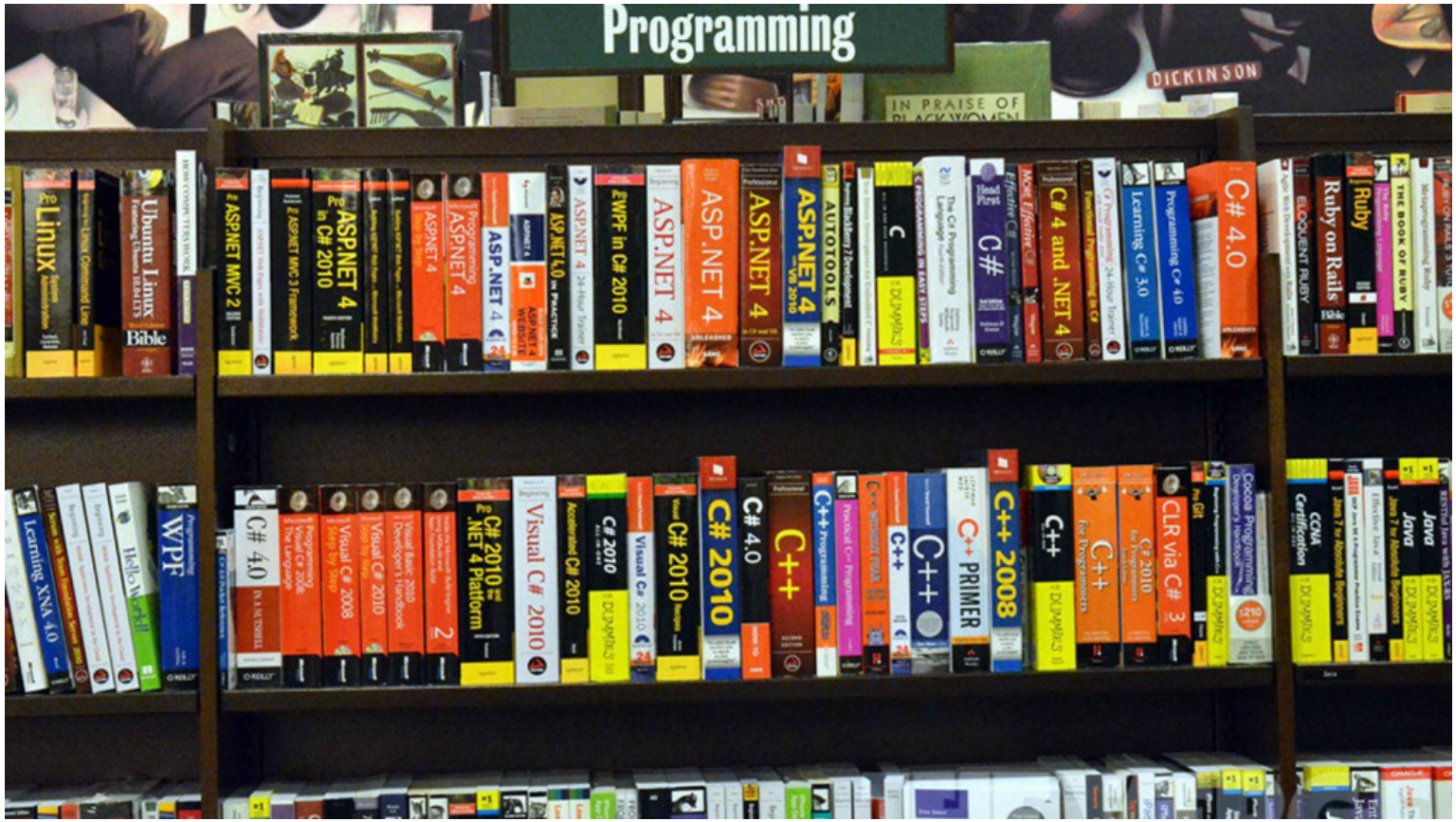
All have strengths and weaknesses



Outline

Why Python?

So, why all the fuss about Python? Perhaps you have heard about Python from a friends, heard a reference to this programming language at a conference, or followed a link from a page on scientific computing, but wonder **what extra benefits** the Python language provides given the suite of powerful computational tools the Earth sciences already has.



Python is beautiful

The chief reason is that it's just a **beautiful** programming language. And it's **versatile**. And it has an **extensive standard library**. And a sheer unimaginably humongous number of third-party libraries and extensions.

The Python Package Index (PyPI) is a repository of software for the Python programming language. There are currently **90052** packages available.

Python is modern

Python is a modern, general-purpose, object-oriented, high-level programming language. It is the programming language of choice for many scientists to a large degree because it offers a great deal of power to analyze and model scientific data with relatively little overhead in terms of learning, installation or development time. Though it has been around for two decades, it exploded into use in the earth sciences after the development community converged upon the **standard scientific packages** needed for earth sciences work.

Python is easy

Python is a robust integration platform for all kinds of earth sciences work, from data analysis to distributed computing, and graphical user interfaces to geographical information systems.

Clean, simple and expressive

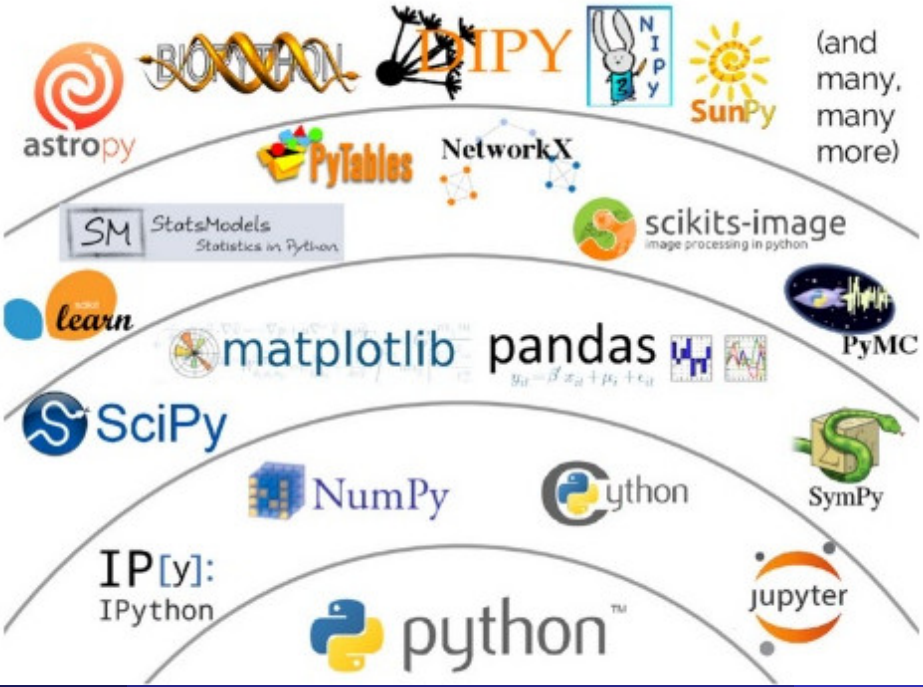
- Easy-to-read minimalistic syntax and intuitive code
- Fewer lines of code, fewer bugs, easier to maintain

It is a language you can pick up in a weekend, and use for the rest of one's life.

Python is open

Finally, **Python's open-source pedigree**, aided by a large user and developer base in sciences, means that your programs can take advantage of the **tens of thousands of Python packages** that exist. These include visualization, numerical libraries, Web services, mobile and desktop graphical user interface programming, and others.

You are not limited to only what one vendor can provide or even what only the scientific community can provide!



Outline

Anaconda Python

Configuration of Python for scientific computing has historically been a huge pain, but recently has gotten a whole lot easier with the advent of two scientific distributions for Python that come with package managers:

- Canopy by Enthought
- Anaconda by Continuum Analytics

I have used Anaconda extensively, and find it a little easier because:

- it stays in a folder where you place it
- doesn't touch the system Python
- doesn't require administrator access to install or modify packages
- it uses a package manager called conda that is extremely easy to use

We suggest to install the Anaconda Python distribution ▸ Link, which provides the Python interpreter itself and all packages we need.

It is available for download for Windows, OS X and Linux operating systems

For Windows and OS X you are given a choice whether to download the graphical installer or the text based installer. If you don't know what the terminal (OS X) or command prompt (Windows) is, then you are better advised to choose the graphical version.
There are two branches of current releases in Python:

- the older-syntax Python 2
- the newer-syntax Python 3

Recently, the Python 3 is mature enough, so you want to install the Python 3 version.

Download the installer ▸ Link, start it, and follow instructions. Accept default values as suggested.