



NETWORK SCIENCE OF ONLINE INTERACTIONS

Chapter I exercises +
Scientific Computing

Joao Neto

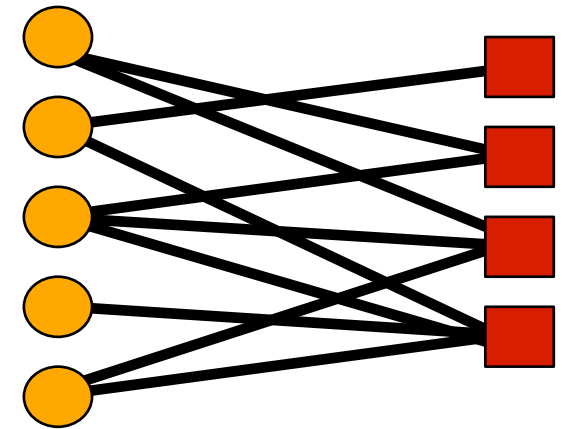
26/Apr/2023

BOOK EXERCISES – CHAPTER I

- Exercises: 1.13, 1.24, 1.25

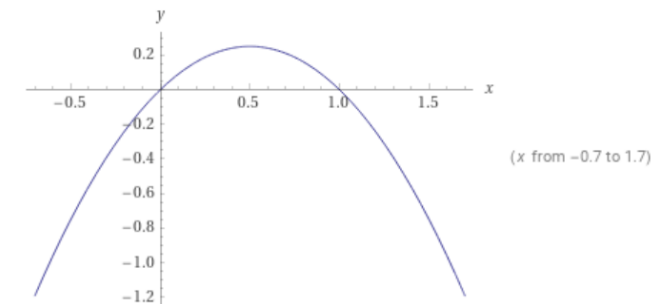
1.13 Consider a bipartite network of N nodes, N_1 nodes of type 1 and N_2 nodes of type 2 (so that $N_1 + N_2 = N$). What is the maximum number of links in this network?

- $L_{max} = N_1 N_2 = N_1 (N - N_1)$
- $\frac{d}{dN_1} L_{max} = N - 2N_1 = 0 \quad \therefore N_1 = N/2$
- $L_{max} = N(N - N/2)/2 \quad \therefore L_{max} = N^2/4$



plot $x(1-x)$ $x = 0$ to ∞

Plots



BOOK EXERCISES – CHAPTER I

1.24 We have seen that Facebook's network is incredibly sparse. Assume it has approximately 1 billion users, each with 1000 friends on average.

- Suppose Facebook releases its annual report and it shows that while the number of users in the network has stayed the same, the average number of friends per user has increased. Would this imply that the network density increased, decreased, or stayed the same?
- Suppose instead that both the number of users and the average number of friends per user doubled. Would this imply the network density increased, decreased, or stayed the same?

→ Increased

→ Decreased, because
 $d \sim O(L, 1/N^2)$

- $d = \frac{2L}{N(N-1)}, \langle k \rangle = d(N-1) = 2L/N$
- $N' = 2N, \langle k \rangle' = 2\langle k \rangle \therefore L' = N'\langle k \rangle'/2 = 2N\langle k \rangle$
- $d' = \frac{2L'}{N'(N'-1)}$

BOOK EXERCISES – CHAPTER I

1.25 Netflix keeps data on customer preferences using a big bipartite network connecting users to titles they have watched and/or rated. Netflix’s movie library contains approximately 100,000 titles if you count streaming and DVD-by-mail. In the fourth quarter of 2013, Netflix reported having about 33 million users. Assume the average user’s degree in this network is 1000. Approximately how many links are in this network? Would you consider this network sparse or dense? Explain.

- $N_t = 10^5, N_u = 3.3 \times 10^7, \langle k \rangle = 10^3$

- $L = N_u \langle k \rangle = 3.3 \times 10^{10}$

- $d = \frac{L}{L_{max}} = \frac{N_u \langle k \rangle}{N_u N_t} = 10^{-2}$

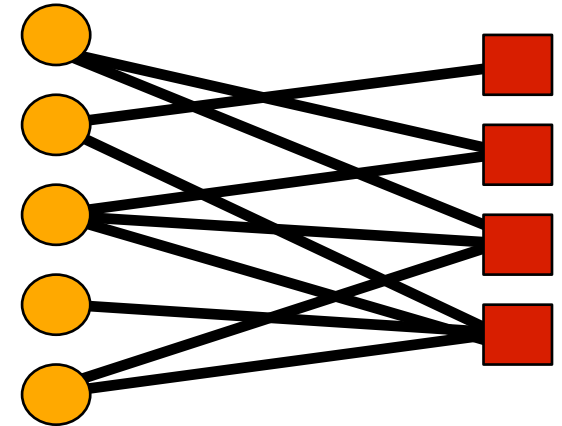


Table 1.1 Basic statistics of network examples. Network types can be (D)irected and/or (W)eighted. When there is no label the network is undirected and unweighted. For directed networks, we provide the average in-degree (which coincides with the average out-degree).

Network	Type	Nodes (N)	Links (L)	Density (d)	Average degree ($\langle k \rangle$)
Facebook Northwestern Univ.		10,567	488,337	0.009	92.4
IMDB movies and stars		563,443	921,160	0.000006	3.3
IMDB co-stars	W	252,999	1,015,187	0.00003	8.0
Twitter US politics	DW	18,470	48,365	0.0001	2.6
Enron Email	DW	87,273	321,918	0.00004	3.7
Wikipedia math	D	15,220	194,103	0.0008	12.8
Internet routers		190,914	607,610	0.00003	6.4
US air transportation		546	2,781	0.02	10.2
World air transportation		3,179	18,617	0.004	11.7
Yeast protein interactions		1,870	2,277	0.001	2.4
C. elegans brain	DW	297	2,345	0.03	7.9
Everglades ecological food web	DW	69	916	0.2	13.3

BOOK EXERCISES – CHAPTER I

- Questions?

A person wearing a black hoodie is standing in the center of the frame. They are positioned on a floor that appears to be a glowing purple grid, receding into the distance. The background is a dark, starry space with some nebulae. The overall aesthetic is futuristic and digital.

HACKERMAN

SCIENTIFIC COMPUTING

LETS TALK ABOUT CODING

- Scientific computing
 - We are (usually) not professional programmers
 - Our code is (usually) not for distribution
 - We are (always) under time constraints
 - Yet we (really, really) want our code to be correct

- How do you prevent that?
 - You don't =(
 - ... but better coding practices help mitigate risks =)

- Example horror story:

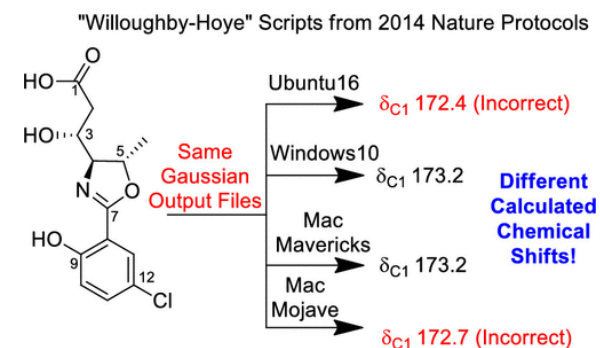
A guide to small-molecule structure assignment through computation of (^1H and ^{13}C) NMR chemical shifts

Patrick H Willoughby, Matthew J Jansma & Thomas R Hoye 

Nature Protocols **9**, 643–660(2014) | [Cite this article](#)

7132 Accesses | **185** Citations | **74** Altmetric | [Metrics](#)

- Widely-distributed code to study chemical spectra
- The bug: devs assumed Python's **glob** function sorted results
- The fallout: likely many papers affected, with no real way to know



(Neupane et al, 2019)

LETS TALK ABOUT CODING

■ Part I: Scientific computing

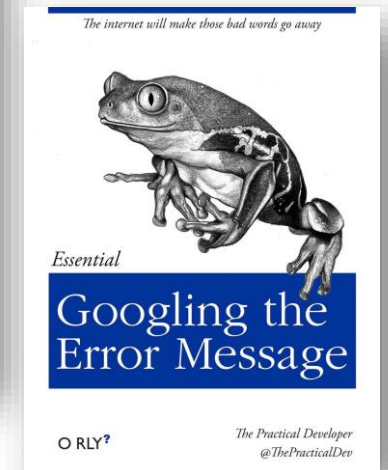
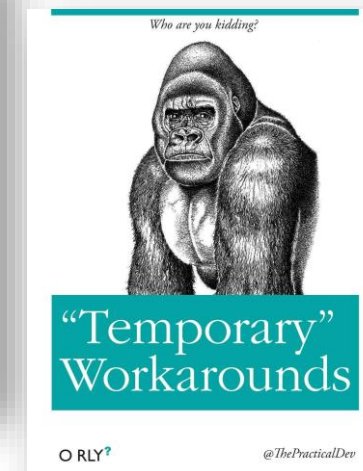
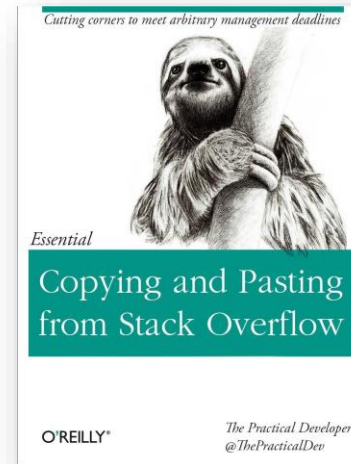
PERSPECTIVE

Good enough practices in scientific computing

Greg Wilson¹*, **Jennifer Bryan**², **Karen Cranston**³, **Justin Kitzes**⁴, **Lex Nederbragt**⁵, **Tracy K. Teal**⁶

1 Software Carpentry Foundation, Austin, Texas, United States of America, **2** RStudio and Department of Statistics, University of British Columbia, Vancouver, British Columbia, Canada, **3** Department of Biology, Duke University, Durham, North Carolina, United States of America, **4** Energy and Resources Group, University of California, Berkeley, Berkeley, California, United States of America, **5** Centre for Ecological and Evolutionary Synthesis, University of Oslo, Oslo, Norway, **6** Data Carpentry, Davis, California, United States of America

■ Part II: data exploration workflow (in Python)



SCIENTIFIC COMPUTING

- <https://software-carpentry.org/>
 - Initiative to teach good coding to researchers
- 6 categories:
 - Data management
 - Software
 - Collaboration
 - Project organization
 - Keeping track of changes
 - Manuscripts

OPEN ACCESS Freely available online



Community Page

Best Practices for Scientific Computing

Greg Wilson^{1*}, D. A. Aruliah², C. Titus Brown³, Neil P. Chue Hong⁴, Matt Davis⁵, Richard T. Guy⁶, Steven H. D. Haddock⁷, Kathryn D. Huff⁸, Ian M. Mitchell⁹, Mark D. Plumbley¹⁰, Ben Waugh¹¹, Ethan P. White¹², Paul Wilson¹³

1 Mozilla Foundation, Toronto, Ontario, Canada, **2** University of Ontario Institute of Technology, Oshawa, Ontario, Canada, **3** Michigan State University, East Lansing, Michigan, United States of America, **4** Software Sustainability Institute, Edinburgh, United Kingdom, **5** Space Telescope Science Institute, Baltimore, Maryland, United States of America, **6** University of Toronto, Toronto, Ontario, Canada, **7** Monterey Bay Aquarium Research Institute, Moss Landing, California, United States of America, **8** University of California Berkeley, Berkeley, California, United States of America, **9** University of British Columbia, Vancouver, British Columbia, Canada, **10** Queen Mary University of London, London, United Kingdom, **11** University College London, London, United Kingdom, **12** Utah State University, Logan, Utah, United States of America, **13** University of Wisconsin, Madison, Wisconsin, United States of America



PERSPECTIVE

Good enough practices in scientific computing

Greg Wilson^{1☯*}, Jennifer Bryan^{2☯}, Karen Cranston^{3☯}, Justin Kitzes^{4☯}, Lex Nederbragt^{5☯}, Tracy K. Teal^{6☯}

1 Software Carpentry Foundation, Austin, Texas, United States of America, **2** RStudio and Department of Statistics, University of British Columbia, Vancouver, British Columbia, Canada, **3** Department of Biology, Duke University, Durham, North Carolina, United States of America, **4** Energy and Resources Group, University of California, Berkeley, Berkeley, California, United States of America, **5** Centre for Ecological and Evolutionary Synthesis, University of Oslo, Oslo, Norway, **6** Data Carpentry, Davis, California, United States of America

SCIENTIFIC COMPUTING

■ Data management

- Save the raw data. ←
- Ensure that raw data are backed up in more than one location.
- Create the data you wish to see in the world. ←
- Create analysis-friendly data. ←
- Record all the steps used to process data. ←
- Anticipate the need to use multiple tables, and use a unique identifier for every record.
- Submit data to a reputable DOI-issuing repository so that others can access and cite it.

■ Software


- Place a brief explanatory comment at the start of every program. ←
- Decompose programs into functions. ←
- Be ruthless about eliminating duplication. ←
- Always search for well-maintained software libraries that do what you need.
- Test libraries before relying on them.
- Give functions and variables meaningful names.
- Make dependencies and requirements explicit.
- Do not comment and uncomment sections of code to control a program's behavior.
- Provide a simple example or test data set. ←
- Submit code to a reputable DOI-issuing repository

SCIENTIFIC COMPUTING


■ Collaboration

- Create an overview of your project.
- Create a shared "to-do" list for the project.
- Decide on communication strategies.
- Make the license explicit.
- Make the project citable.


■ Project organization

- Put each project in its own directory, which is named after the project.
- Put text documents associated with the project in the *doc* directory.
- Put raw data and metadata in a data directory and files generated during cleanup and analysis in a results directory. 
- Put project source code in the *src* directory.
- Put external scripts or compiled programs in the *bin* directory

■ Keeping track of changes

- Back up (almost) everything created by a human being as soon as it is created.
- Keep changes small.
- Share changes frequently.
- Create, maintain, and use a checklist for saving and sharing changes to the project.
- Store each project in a folder that is mirrored off the researcher's working machine.
- Add a file called *CHANGELOG.txt* to the project's docs subfolder.
- Copy the entire project whenever a significant change has been made.
- Use a version control system 

■ Manuscripts

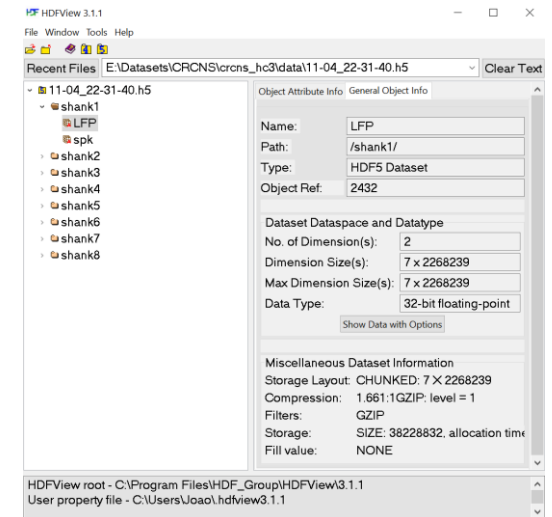
- Write manuscripts using online tools with rich formatting, change tracking, and reference management. 
- Write the manuscript in a plain text format that permits version control.

SCIENTIFIC COMPUTING

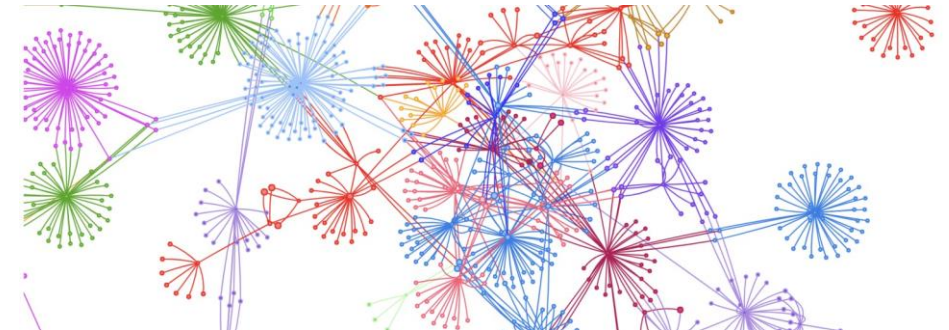
- Save the raw data.
 - Storage is cheap, time is expensive (*we have TBs of space*)
- Create the data you wish to see in the world.
 - Making data human and machine-readable makes it shareable
 - *We already receive dirty data: it becomes useless if the person receiving it doesn't clean it up*
 - *Save all attributes (parameters, etc) you possibly can: you never know what will be useful later*
- Different data types, different formats
 - Tabular data: CSV
 - Structured data: HDF5
 - Graph data: JSON

epoch	train_loss	valid_loss	accuracy	time
0	0.871673	0.690569	0.560000	00:02
1	0.609895	1.017932	0.660000	00:02
2	0.474530	1.093525	0.780000	00:02
3	0.418120	1.386144	0.780000	00:02
4	0.330656	0.690509	0.860000	00:02
5	0.244123	0.844521	0.760000	00:02
6	0.178164	0.834935	0.800000	00:02
7	0.131585	0.842775	0.800000	00:02
8	0.098164	0.836589	0.800000	00:02
9	0.074047	0.848848	0.800000	00:02

tabular data



structured data



Graph data

SCIENTIFIC COMPUTING

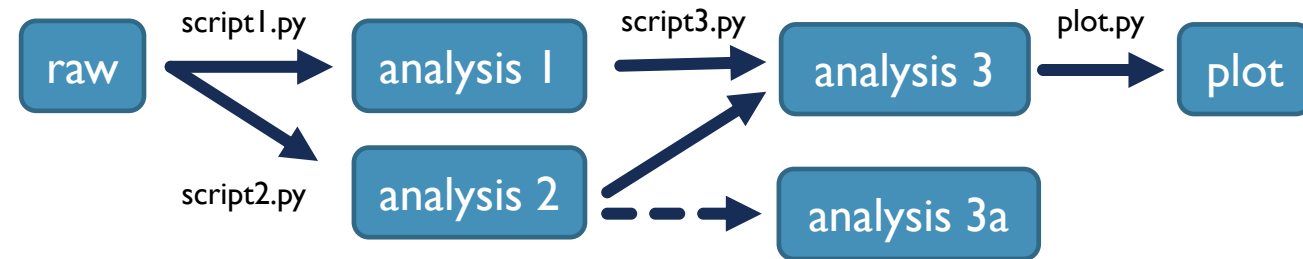
- Create analysis-friendly data.
 - “tidy data” (for tabular data):
 - Each column is a variable
 - Each row is an observation
 - *Timeseries with millions of points: attributes should be tidy*

site	1999	2000
Whitehorse	745	2666
Yellowknife	37737	80488
Inuvik	212258	213766



site	year	cases
Whitehorse	1999	745
Whitehorse	2000	2666
Yellowknife	1999	37737
Yellowknife	2000	80488
Inuvik	1999	212258
Inuvik	2000	213766

- Record all the steps used to process data.
 - Write scripts for every stage of data processing
 - *Useful for data exploration*
 - *In particular: save the data points you plot!*



SCIENTIFIC COMPUTING

- Place a brief explanatory comment at the start of every program.
 - Add an example of usage
 - Include reasonable parameters
 - *Python: using a standard docstring format (numpy or Google) makes it parseable*
- Decompose programs into functions.
 - Makes things modular and reusable
 - Makes things understandable (especially to future you)
- Be ruthless about eliminating duplication
 - Reuse functions (no boilerplate code)
 - Always search for well-maintained code first (but test it!)
 - *Very important to stop spaghettification of your code*

```
def delay_cases(
):
    """
    Convolves the input by a lognormal distribution, in order to model a delay:

    * We have a kernel (a distribution) of delays, one realization of this kernel is
      applied to each pymc3 sample.

    * The kernel has a median delay D and a width that correspond to this one
      sample. Doing the ensemble average over all samples and the respective
      kernels, we get two distributions: one of the median delay D and one of the
      width.

    * The (normal) distribution of the median of D is specified using
      'pr_mean_of_median' and 'pr_sigma_of_median'.

    * The (lognormal) distribution of the width of the kernel of D is specified
      using 'pr_median_of_width' and 'pr_sigma_of_width'. If
      'pr_sigma_of_width' is None, the width is fixed (skipping the second
      distribution).

    Parameters
    -----
    cases : :class:`~theano.tensor.TensorVariable`
        The input, typically the number of newly infected cases from the output of
        :func:`SIR` or :func:`SEIR`.

    name_delay : str
        The name under which the delay is saved in the trace, suffixes and prefixes
        are added depending on which variable is saved.
        Default : "delay"
    """
```

delay_cases

Definition: delay_cases(cases, name_delay="delay", name_cases=None, name_width="delay-width", pr_mean_of_median=10, pr_sigma_of_median=0.2, pr_median_of_width=0.3, pr_sigma_of_width=None, model=None, len_input_arr=None, len_output_arr=None, diff_input_output=None)

Type: Function of covid19_inference.model.delay module

Convolves the input by a lognormal distribution, in order to model a delay:

- We have a kernel (a distribution) of delays, one realization of this kernel is applied to each pymc3 sample.
- The kernel has a median delay D and a width that correspond to this one sample. Doing the ensemble average over all samples and the respective kernels, we get two distributions: one of the median delay D and one of the width.
- The (normal) distribution of the median of D is specified using *pr_mean_of_median* and *pr_sigma_of_median*.
- The (lognormal) distribution of the width of the kernel of D is specified using *pr_median_of_width* and *pr_sigma_of_width*. If *pr_sigma_of_width* is None, the width is fixed (skipping the second distribution).

Parameters

cases: TensorVariable

The input, typically the number of newly infected cases from the output of SIR() or SEIR().

name_delay: str

The name under which the delay is saved in the trace, suffixes and prefixes are added depending on which variable is saved. Default : "delay"

SCIENTIFIC COMPUTING

- Provide a simple example or test data set
 - For others and for future you
 - Check if it gives the correct result (*unit testing*)
 - *Notebooks (e.g. jupyter) particularly useful*
- Put raw data and metadata in a data directory and files generated during cleanup and analysis in a results directory.
- Write manuscripts using online tools with rich formatting, change tracking, and reference management.
- **Use a version control system (git!)**
 - *Extremely useful to keep track of changes and work in different things in parallel, even by yourself*
 - *Necessary for collaboration*
 - *Continuous Integration (CI): commit little, commit often*

```
[1]: !pip install git+https://github.com/Prieemann-Group/covid19_inference.git
```

Hierarchical Bayesian Model for all German states (Bundeslaender).

Runtime ~ 1h

This file creates the model with three changing points of the paper
<https://science.sciencemag.org/content/early/2020/05/14/science.abb9789>.

Importing modules

```
[2]: import datetime
import sys
import pandas as pd
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import scipy.stats
import theano
import theano.tensor as tt
import pymc3 as pm
```

```
[3]: try:
    import covid19_inference as cov19
except ModuleNotFoundError:
    sys.path.append("../..")
    import covid19_inference as cov19
```

Creating the model

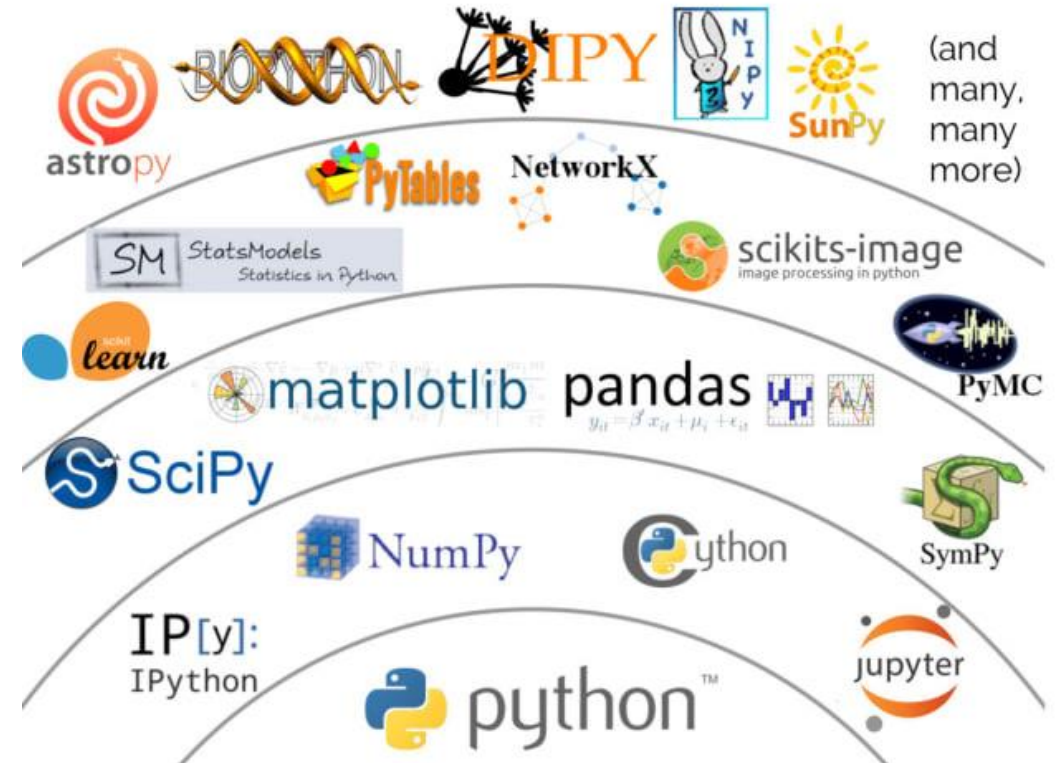
This cell downloads data and builds the pymc3 model for the Bayesian inference. More information can be found in the [documentation](#).

```
[4]: # Dates to obtain the data from
date_begin_data = datetime.datetime(2020, 3, 10)
date_end_data = datetime.datetime(2020, 4, 19)
```

```
[5]: # Downloads 2-D array of new cases (for each state) from the Robert Koch Institute
rki = cov19.data_retrieval.RKI(True)
df_bundeslaender = rki.filter_all_bundesland(date_begin_data, date_end_data)
new_cases_obs = np.diff(np.array(df_bundeslaender), axis=0)[: , :]
```

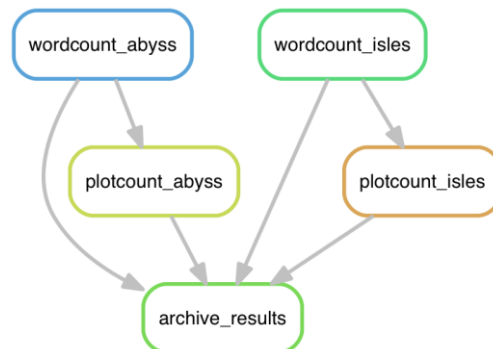
DATA EXPLORATION WORKFLOW

- There is a lot of languages
 - Python, Julia, C++, R, MATLAB, etc
- There is a lot of IDE's/editors
 - Atom, VS Code, PyCharm, Spyder, Sublime, Jupyter, etc
- There is **a lot** of packages
 - pypi lists 236k projects, CRAN lists 15k
- n developers \rightarrow more than n workflows

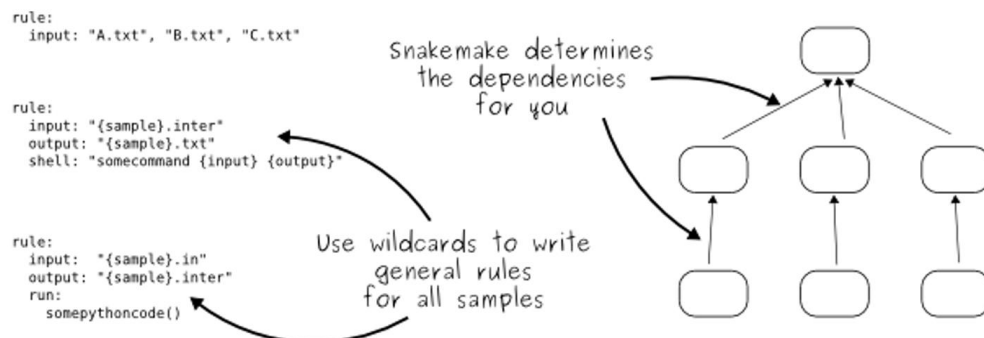


DATA EXPLORATION WORKFLOW

- Well-defined analysis pipeline
 - Applying some established method



- snakemake**: does everything for you
 - Alternative: **pypet**

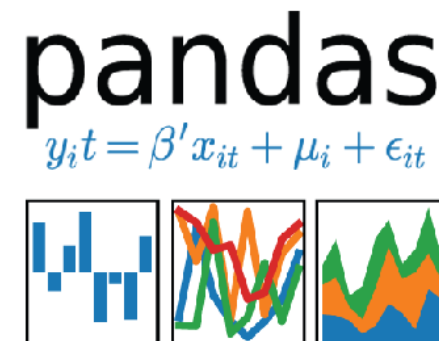
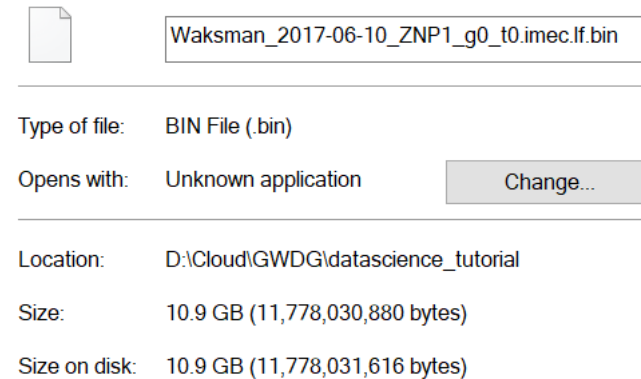


- Data exploration workflow
 - Manual effort: a human* must stop at each step and decide on the next
 - We want flexible tools that help do that
 - We also want then to be robust and well-documented
- The workflow we will show here:



DATA EXPLORATION WORKFLOW

- Data: CSV/HDF5
 - Comma Separated Values (CSV) : anyone can open tables
 - Hierarchical Data Format 5 (HDF5)
 - can have structured data
 - can save attributes
 - can manipulate data storage topology
 - Ex: Neuropixels file (384 channels × timeseries)
 - File saved by columns, want to access by row
 - Need to read the entire file 384 times
- Manipulation: **pandas**
 - Primary data type: DataFrame (table)
 - “Excel for python”: manipulates tabular data



	BandName	WavelengthMax	WavelengthMin
0	CoastalAerosol	450	430
1	Blue	510	450
2	Green	590	530
3	Red	670	640
4	NearInfrared	880	850
5	ShortWaveInfrared_1	1650	1570
6	ShortWaveInfrared_2	2290	2110
7	Cirrus	1380	1360

DATA EXPLORATION WORKFLOW

■ Plotting: seaborn/matplotlib

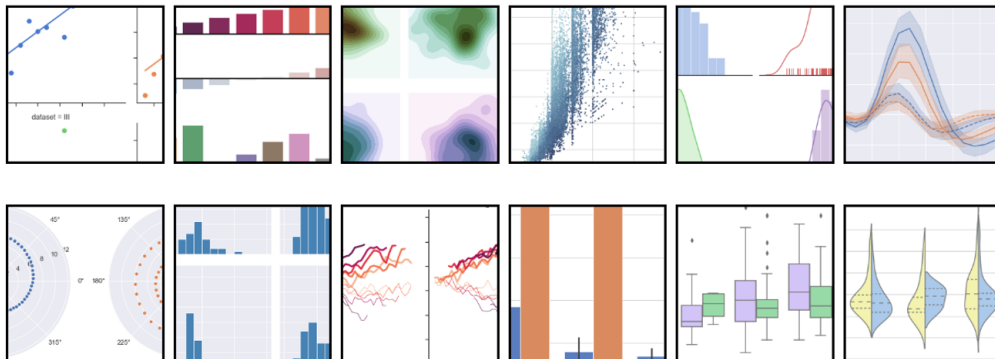
■ Matplotlib:

- Love it or hate it, you will use it

■ Seaborn:

- High-level interface for matplotlib
- can extremely easily make very pretty plots
- Integrates with pandas
- Worse than using matplotlib directly for custom stuff

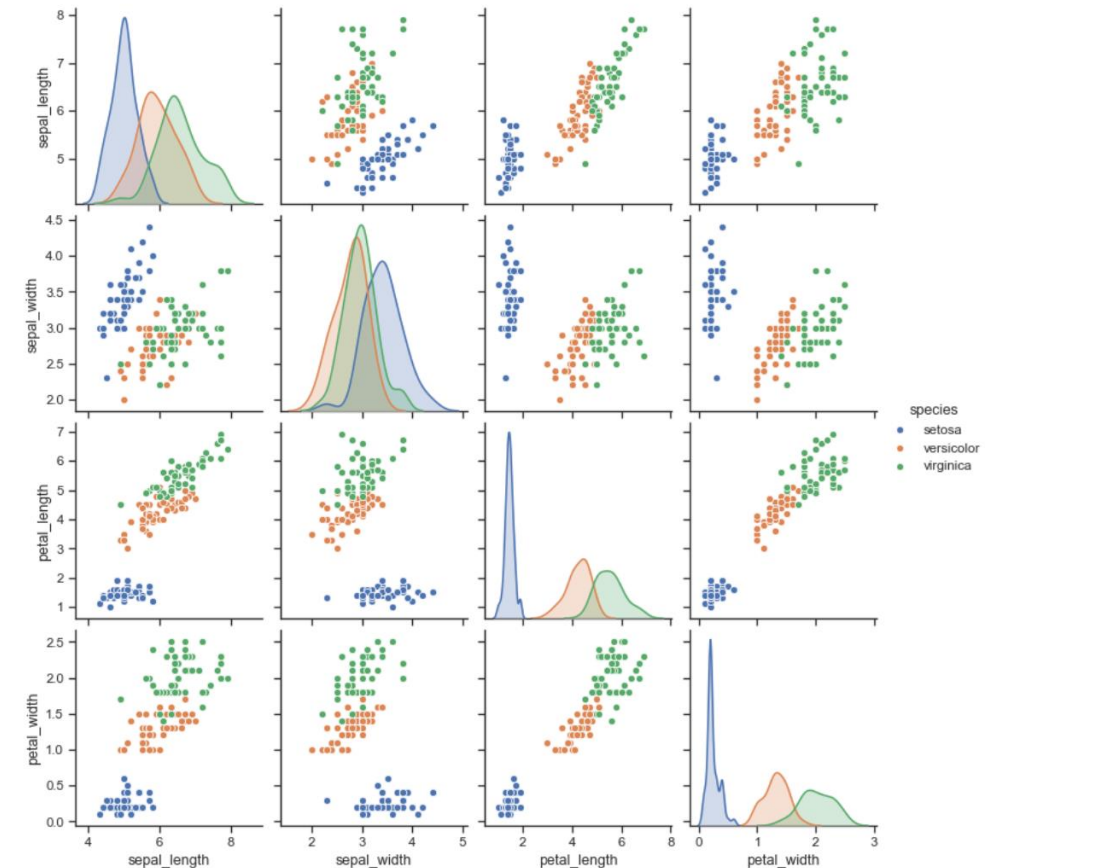
Example gallery



```
import seaborn as sns
sns.set(style="ticks")

df = sns.load_dataset("iris")
sns.pairplot(df, hue="species")
```

Scatterplot Matrix



TAKE HOME MESSAGES

- Use git
- Use git
- Investing time into making your code maintainable pays off in the long-term, even for private code

"We should forget about small efficiencies, say about 97% of the time: premature optimization is the root of all evil. Yet we should not pass up our opportunities in that critical 3%." – Donald Knuth

