data analysis with python

Bonnie Zhang
readr feb 2019

plan for the morning

- intro/scope
- some python
- wrangling/plotting/visualising data
- fitting data from a Bayesian viewpoint
- wrap up, Q&A

CV in a slide

- PhB (Sci) Honours in pure maths, ANU 2012
- physics olympiad staff 2009-2016 (deputy director 2014-2016)
- PhD in astrophysics (supernova cosmology), ANU 2018
- software engineer at QuintessenceLabs, 2017-now

scope

- “data analysis with python”
- basic python, data wrangling, plotting
- focus on Bayesian stats
- not much about data science or machine learning

disclaimer

- this will be biased towards what i know reasonably well
- astronomy/astrophysics
- scripts over notebooks (i typically use *ipython --pylab*)
- python2 over python3

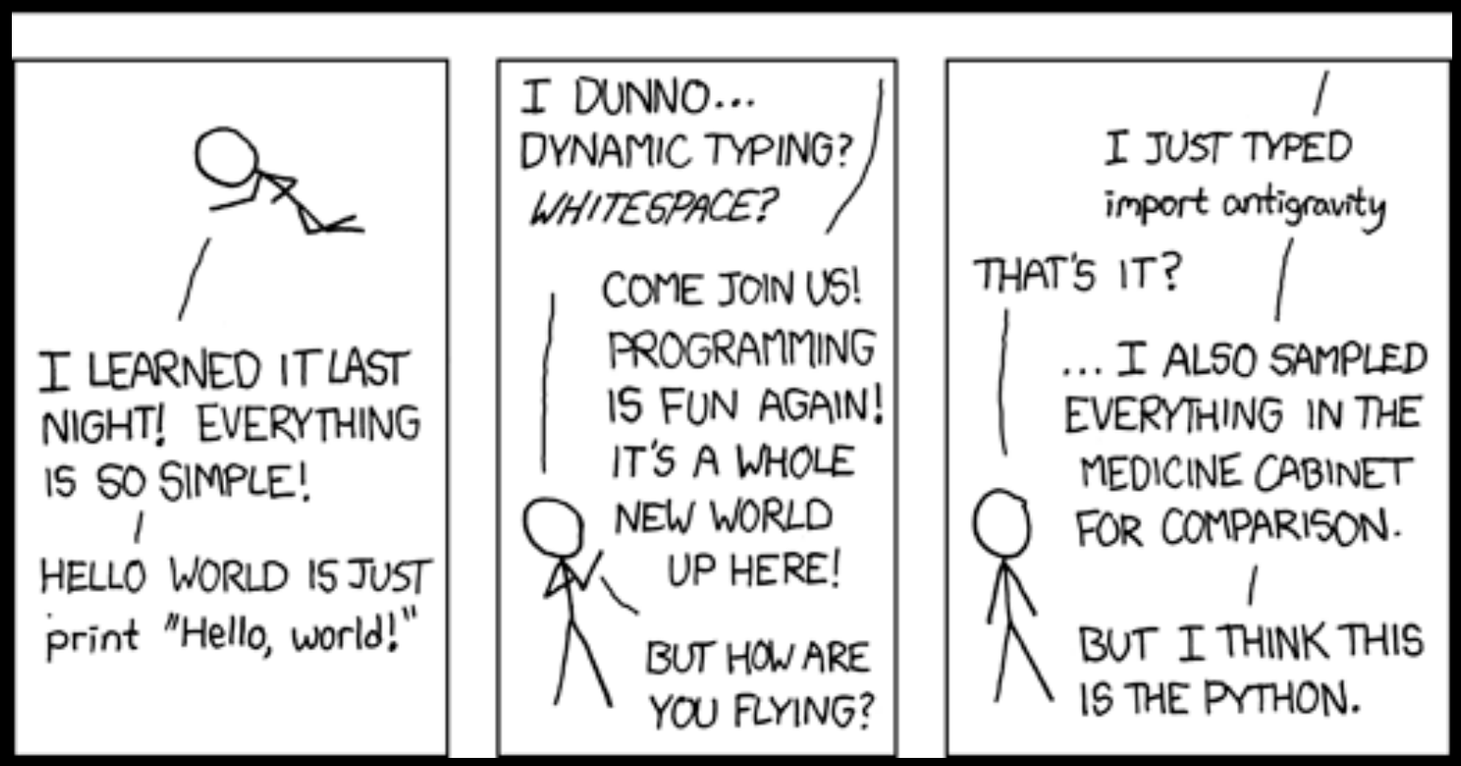
prereqs

- have git repo locally (if not: git clone https://github.com/GroundhogState/readr_2019.git & git pull)
- basic working knowledge of python
- can run scripts in an interpreter or notebook
- willingness to interrupt and ask questions at any point

why python?

- high-level, interpretive
- object-oriented
- easy to learn, little overhead, readable
- excellent suite of scientific libraries
- quick (in the sense of installation, learning, development)


PYTHON!
YOU'RE FLYING!
HOW?


I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"
I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?
I JUST TYPED import antigravity
THAT'S IT?
... I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.

xkcd 353

basic objects in python

- strings, integers, floats, booleans, complex
- lists, dicts, tuples, numpy arrays (multidimensional)
- **functions**: operator that applies rules (define with *def*)
- **class**: general inherited object with specific properties, to be re-used (basis of OOP)

mutability

- **mutable** objects can be changed after creation (e.g. lists, dicts, sets)
- **immutable** objects cannot (e.g. numeric, strings, tuples)

- variables are references to objects:
- id(var) for memory address

```
In [1]: a = [1,2,3]

In [2]: b = a

In [3]: b[0] = -99

In [4]: a
Out[4]: [-99, 2, 3]
```

nomenclature

- **scripts**: .py files with code, that can be run (i like ipython as interactive shell)
- **notebooks**: .ipynb: much the same except with a kernel + browser (and run bit by bit)
- **module**: folder or script to import submodules or functions from(can nest)

operations & other useful bits

- indexing, slices
- extending, appending lists
- define list/array using if:
- ternary function:
- lambda functions:

```
In [1]: import numpy as np

In [2]: np.array([x for x in range(14) if x % 3 == 1])
Out[2]: array([ 1,  4,  7, 10, 13])

In [3]: switch = False

In [4]: 5 if switch else 3
Out[4]: 3

In [5]: switch = True

In [6]: 5 if switch else 3
Out[6]: 5

In [7]: f = lambda x: x**2

In [8]: f(7)
Out[8]: 49
```

numpy-specific

- arithmetic e.g. +-*/, **, log: can apply both to numeric types and numpy arrays
- where, all, any
- zeros, ones
- reshape
- concatenate
- masked arrays

control flow

- if, for, while, break, continue
- error handling
- try, except, continue

misc/gotchas

- importing module in interpreter
- integer division
- cannot starting variable with number
- scope of variables
- __main__ vs parsing arguments
- bools:

```
In [1]: bool(false)
---------------------------------------------------------------
NameError                                 Traceback (mo
<ipython-input-1-3536571ee15f> in <module>()
----> 1 bool(false)

NameError: name 'false' is not defined

In [2]: bool('False')
Out[2]: True
```

debugging & standards

- unit testing
- break down into smallest and/or dumbest example possible
- linters are helpful (e.g. pylint): will save time
- coding standards
- *import this*

ex1_warmup.py: some python tasks to get started

loading data

- python file handling: open, read, readlines
- array-specific: numpy.loadtxt, numpy.genfromtxt, pandas.read_csv
- for astro specific .fits format: astropy.load_fits

why plot your data?

for yourself:

- understand your data e.g. reveal relations, trends
- guide analysis e.g. what does the distribution look like? what is the range?
- assess your analysis

for others:

- to communicate your findings

good first steps plotting

- scatter, try to plot relations, best fit, residuals, time series
- histograms can be useful
- matplotlib: extensive, customisable, well-documented, lots of examples
- https://matplotlib.org/2.0.2/examples/pylab_examples
- some people like gnuplot, mayavi (3D), seaborn (data science)

good vs bad plots

- good rule of thumb: more useful stuff, less useless stuff
- use space and colours with intention

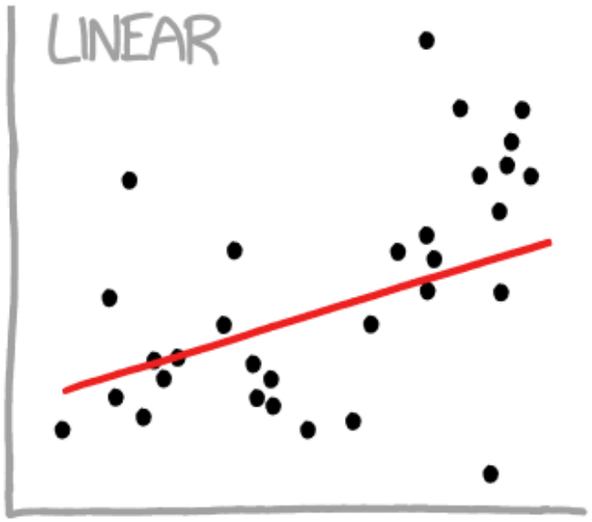
- https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1003833

- http://www.randalolson.com/2014/06/28/how-to-make-beautiful-data-visualizations-in-python-with-matplotlib/

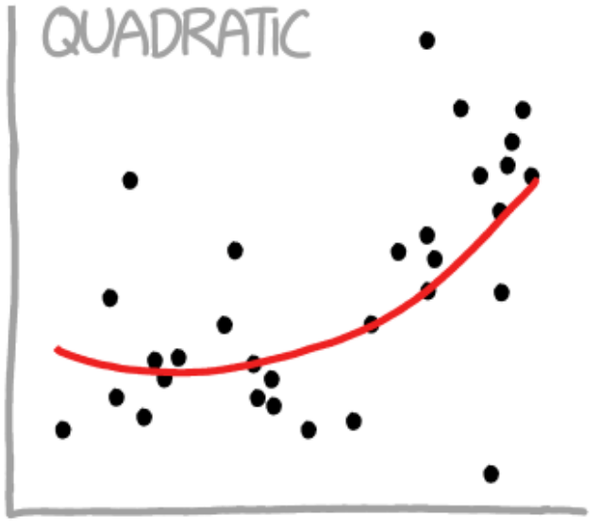
why simulate data?

- quickest way of getting well-understood custom dataset to test your methods on
- for diagnostic purposes
- while waiting for real data
- for validation of your results

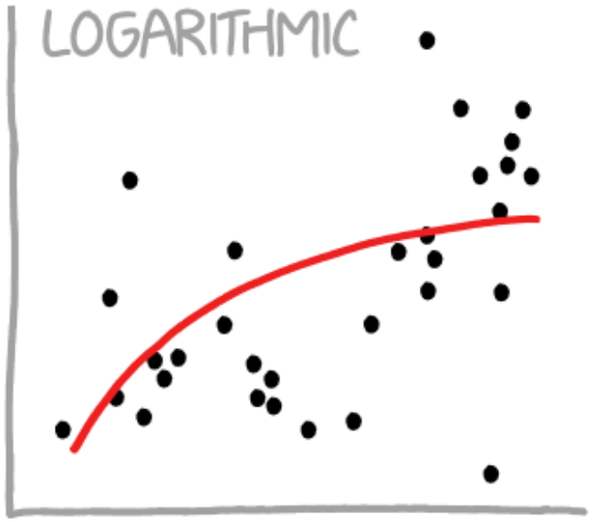
CURVE-FITTING METHODS
AND THE MESSAGES THEY SEND



"HEY, I DID A REGRESSION."

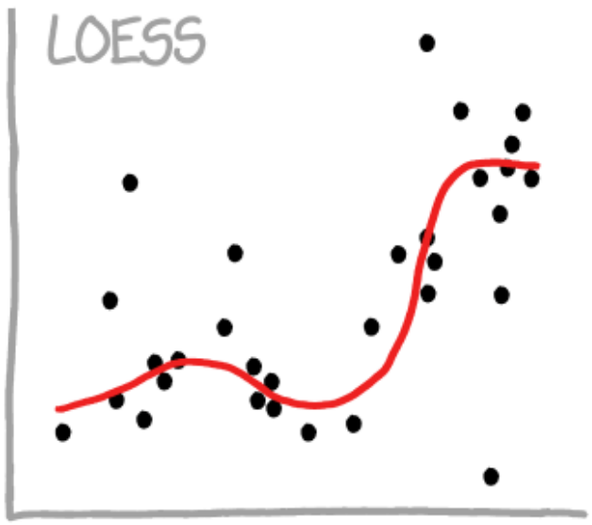


"I WANTED A CURVED LINE, SO I MADE ONE WITH MATH."

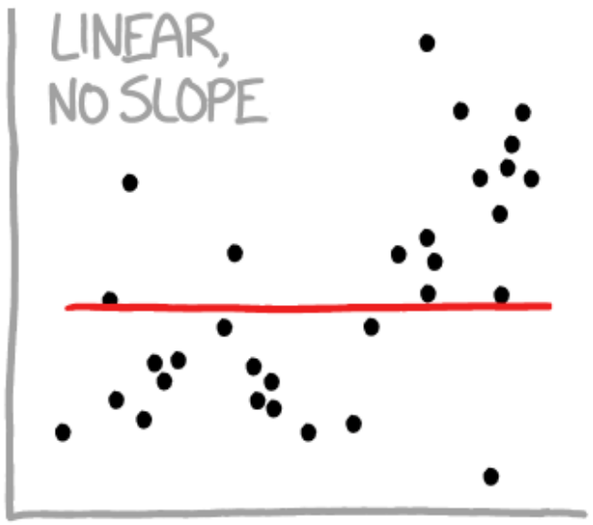


"LOOK, IT'S TAPERING OFF!"

"LOOK, IT'S GROWING UNCONTROLLABLY!"



"I'M SOPHISTICATED, NOT LIKE THOSE BUMBLING POLYNOMIAL PEOPLE."

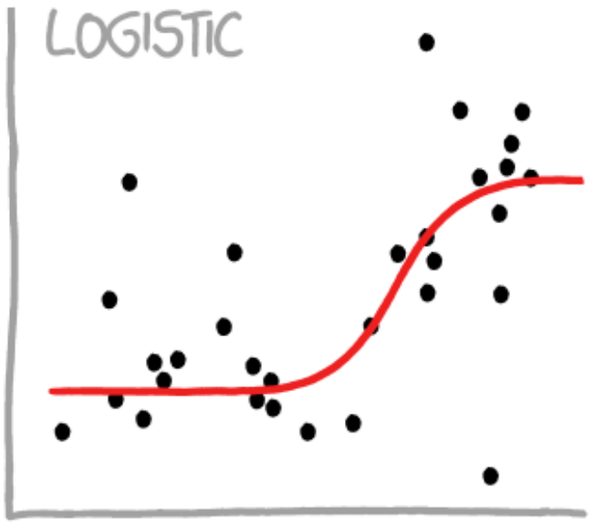


"I'M MAKING A SCATTER PLOT BUT I DON'T WANT TO."

"LOOK, IT'S GROWING UNCONTROLLABLY!"

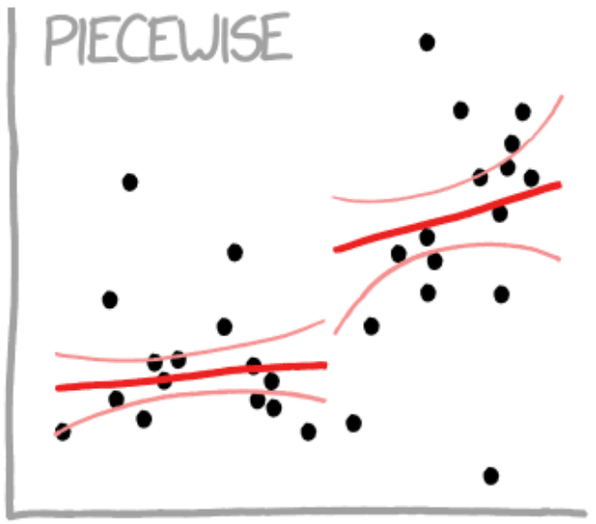
"I'M SOPHISTICATED, NOT LIKE THOSE BUMBLING POLYNOMIAL PEOPLE."

I'M MAKING A SCATTER PLOT BUT I DON'T WANT TO.

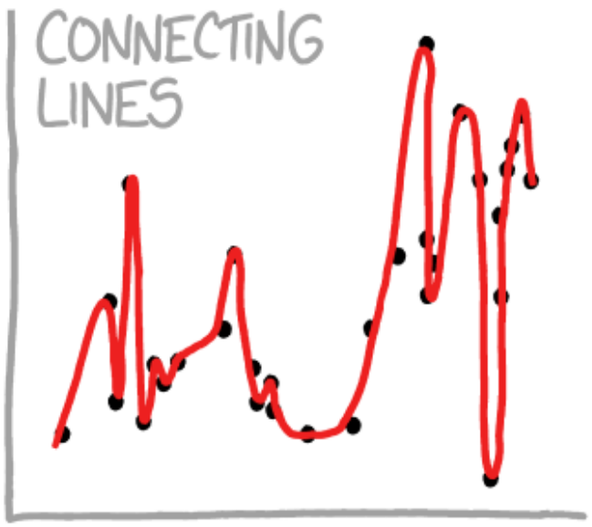


"I NEED TO CONNECT THESE TWO LINES, BUT MY FIRST IDEA DIDN'T HAVE ENOUGH MATH."

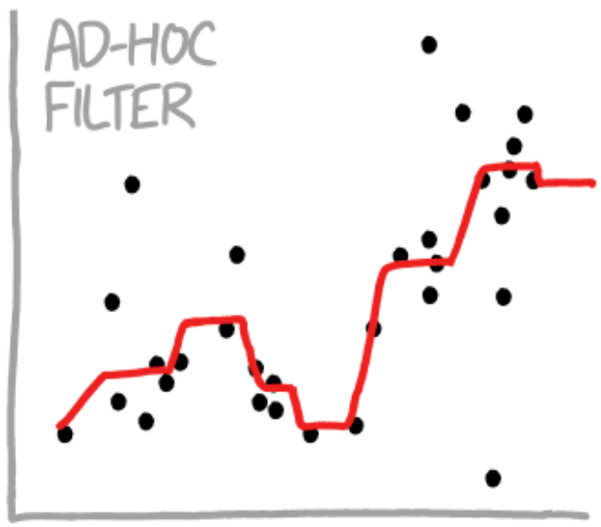
"LISTEN, SCIENCE IS HARD. BUT I'M A SERIOUS PERSON DOING MY BEST."



"I HAVE A THEORY, AND THIS IS THE ONLY DATA I COULD FIND."



"I CLICKED 'SMOOTH LINES' IN EXCEL."



"I HAD AN IDEA FOR HOW TO CLEAN UP THE DATA. WHAT DO YOU THINK?"

"AS YOU CAN SEE, THIS MODEL SMOOTHLY FITS THE- *WAIT NO NO DON'T EXTEND IT AAAAAA!!*"

some tools for curve fitting

- np.polyfit
- scipy.curve_fit
- DIY grid (ex3): teaching example only
- Markov Chain Monte Carlo: later

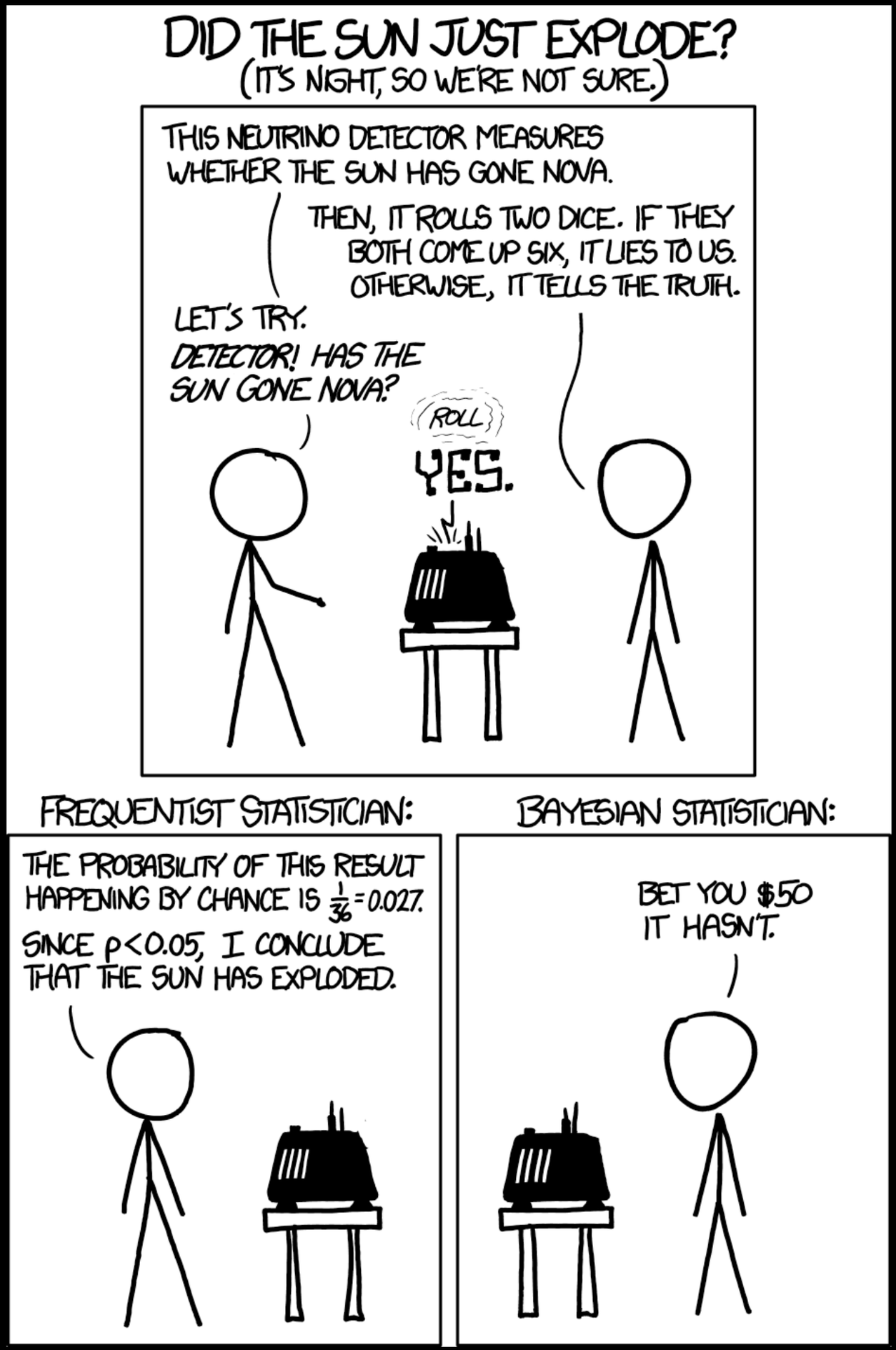
first principles

- simplest example: fitting a line (y=mx+b)
- what might make a (m,b) tuple the ‘best’ fit?
- what metrics are used?
- how would you try to coding this from scratch

ex2_fitting.py: loading/plotting data; parameter fitting using numpy, scipy methods

ex3_medley.py: some simulation; set up for grid-based parameter fitting

these will get you started: we will come back to Bayesian principle



frequentist vs Bayesian

bayesian inference is about:

- updating beliefs when presented with new data
- allowing for incomplete data (immensely useful in cosmology!)
- beliefs as continuous probability distribution function (pdf)

Bayes' theorem

- here, **A** is model (i.e. values for set of parameters **Θ**), **B** is data (**D** in next slide)
- often given as proportionality (evidence is constant normalisation term)
- 'flips' frequentist probability

posterior likelihood prior

$$P(\boldsymbol{\Theta}|\mathcal{D}, H) = \frac{P(\mathcal{D}|\boldsymbol{\Theta}, H)P(\boldsymbol{\Theta}|H)}{P(\mathcal{D}|H)}$$

evidence

actually, there is an implied hypothesis **H** in each of these terms

- **posterior** pdf: what we aim to compute
- **prior** pdf: beliefs before seeing new data **D**
- **evidence**: reflects all observations; constant (i.e. independent of **Θ**)

likelihood

$$\mathcal{L}(\boldsymbol{\Theta}) = P(\mathcal{D}|\boldsymbol{\Theta}, H)$$

- mechanism for updating beliefs
- probability of the model parameters **Θ**, producing the observed data
- often complex/expensive to compute analytically, or else is not tractable

maximal likelihood methods

1. least squares curve fitting

$$\sum_i (y_i - f(X_i, \Theta))^2,$$

2. weight by (inverse) measurement uncertainty: χ^2 minimising

$$\chi^2 = \sum_i \frac{(y_i - f(X_i, \Theta))^2}{\sigma_{y_i}}$$

3. take into account correlated uncertainties: χ^2 minimising with covariance matrix

$$\chi^2 = (y - f(\mathbf{X}, \Theta))\mathbf{C}^{-1}(y - f(\mathbf{X}, \Theta))^T$$

maximal likelihood methods

for normally distributed errors:

$$\mathcal{L}(\boldsymbol{\Theta}) = \exp\left(-\frac{\chi^2(\boldsymbol{\Theta})}{2}\right)$$

$$-\log\mathcal{L} = \frac{1}{2}(\boldsymbol{y} - f(\mathbf{X}, \boldsymbol{\Theta}))\mathbf{C}(\boldsymbol{y} - f(\mathbf{X}, \boldsymbol{\Theta}))^T$$

where the χ^2 function can take any of the forms in the previous slides (even least squares)

covariance matrices

dry definition:

$$C_{ij} = \langle (X_i - \mu_i)(X_j - \mu_j) \rangle$$

if we want to calculate the covariance matrix due to some systematic:

$$\mathbf{C}_{\mathrm{sys}_{ij}} = \sum_k \left(\frac{\partial \eta_i}{\partial k} \right) \left(\frac{\partial \eta_j}{\partial k} \right) (\Delta k)^2$$

understanding errors/uncertainties is important:
more from Geoff tomorrow

Bayesian parameter fitting

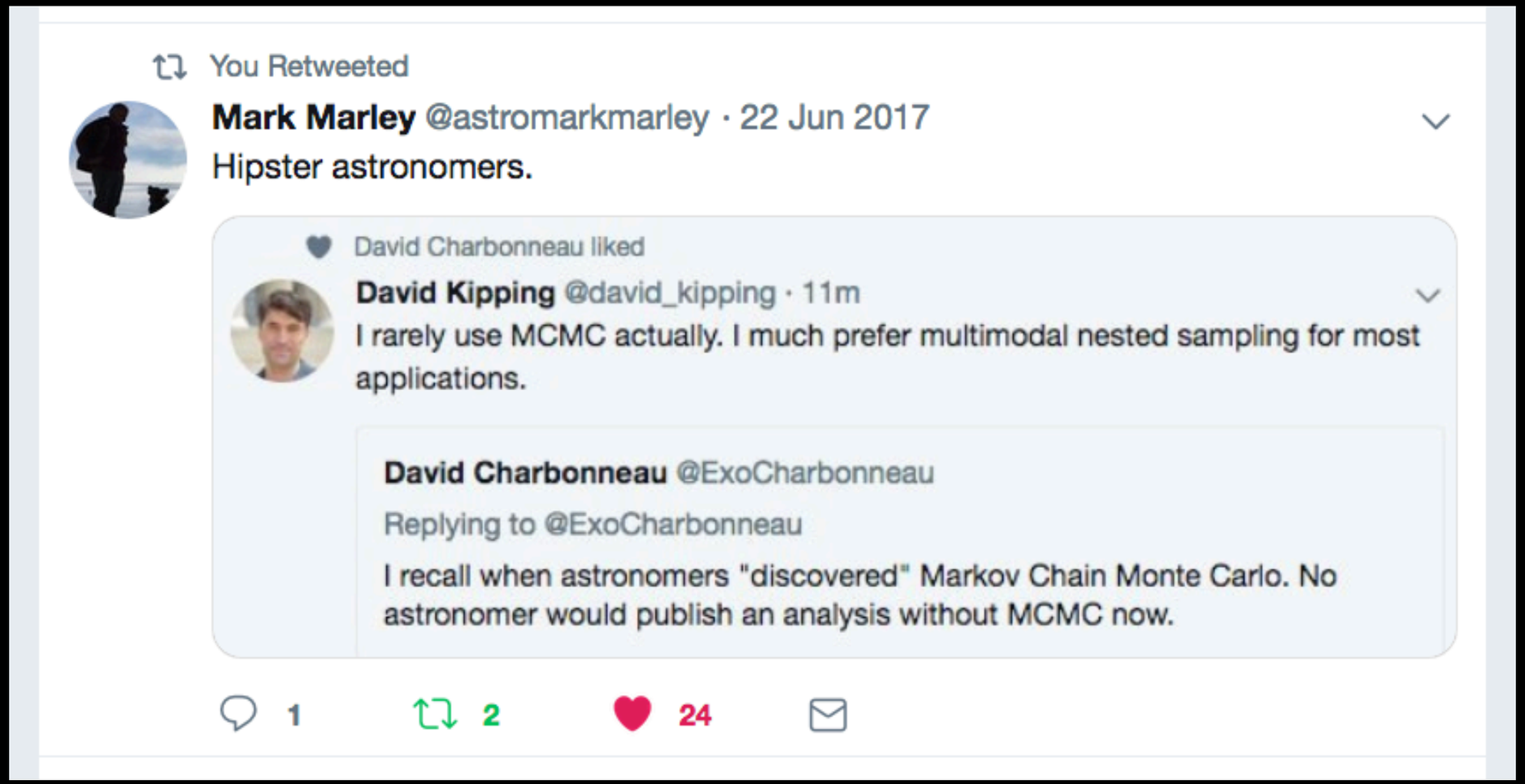
- most of the time the **likelihood** is very difficult or computationally expansive (or both) to evaluate

- you cannot just compute it at every point in your parameter space for some spacing

- sampling based methods: Markov Chain Monte Carlo (MCMC) with Metropolis-Hastings algorithm is the most common

Bayesian parameter fitting

- many more: nested sampling, simulated annealing (variants of MCMC), Approximate Bayesian Computation, Bayesian Hierarchical Modelling, …



MCMC

- Markov Chain: process that only depends on most recent state (has memory of 1)

- Monte Carlo: class of simulation methods

- Markov Chain Monte Carlo (MCMC) describes a class of algorithms which sample the parameter space in a way which faithfully reflects the posterior pdf

- python implementation: *emcee* (http://dfm.io/emcee/current/)

ex3_medley.py: continue

ex4_mcmc_harder.py: example adapted from my science

resources/further reading

- google & stack overflow are your friends
- tutorials, e.g. https://learnpythonthehardway.org
- http://scipy-lectures.org/index.html
- coursera, datacamp, etc.

general advice fwiw

- learn what works for you in terms of learning new skills, reading, writing
- support is available: use it if you can (academic, statistical, psychological, peer)
- other PhD students have been through the same thing - talk to them (although possibly not in their last year/ months when they do not wish to talk to anyone)
- there is life outside of physics

questions?