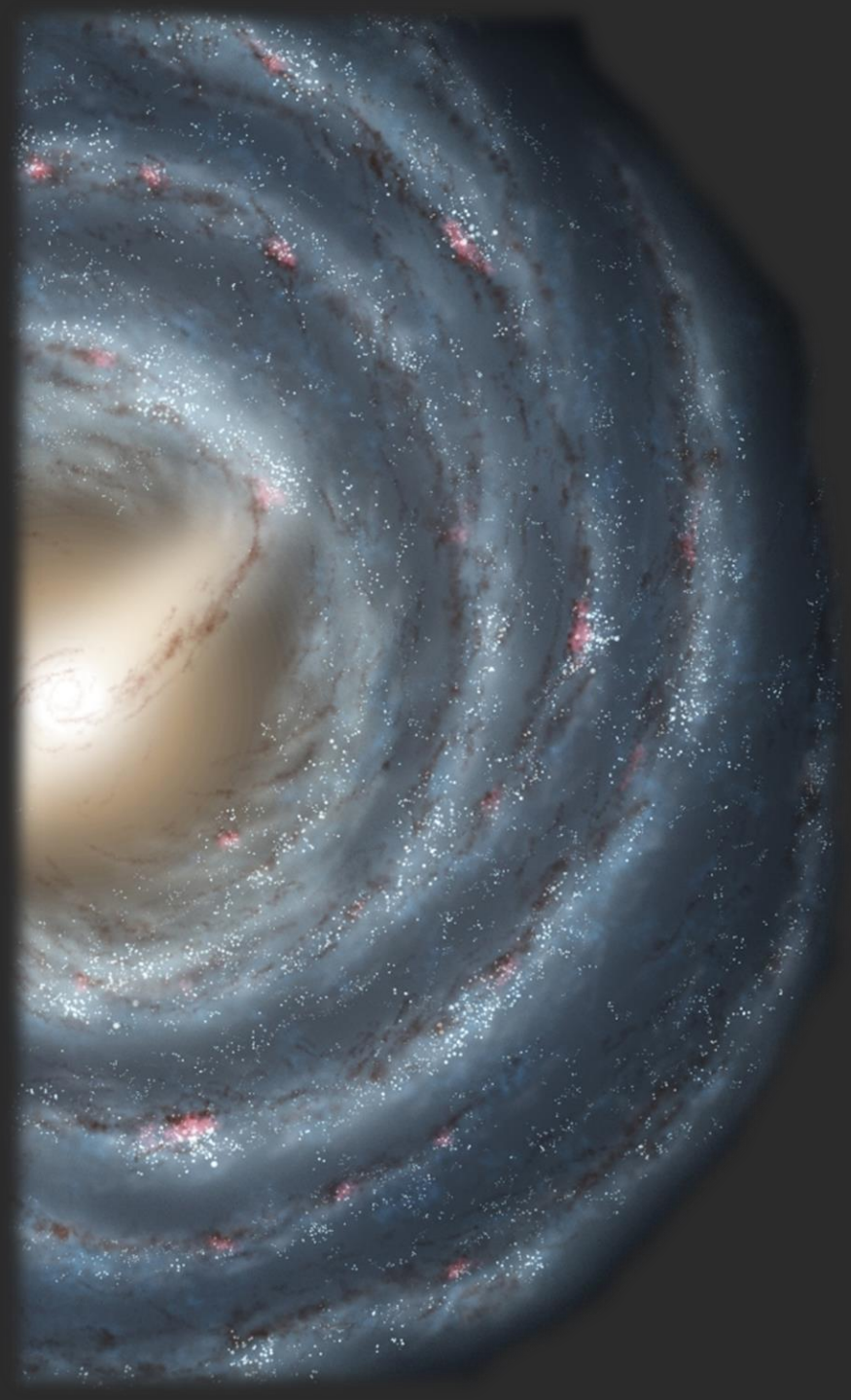


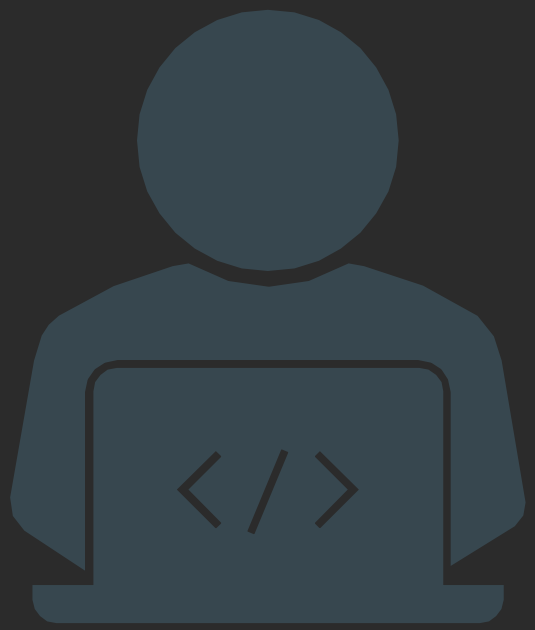
Making a galaxy with Python:

Merits and pitfalls in probabilistic programing

Vital Fernández

PYCONES ALICANTE 2019

So…
what is this about?



~~Programmer~~

~~Developer~~

~~Statistician~~

Lucky bastard for having Python

- To treat astronomical data
- To fit theoretical model
- To publish the results

Talk Outline

Lewis Carrol (1865)

Begin at the beginning (10 min)

- Galaxies: Light from stars and atoms
- Spectra: Celestial histograms
- Atroinformatics: From the telescope to your model

and go on till you come to the end (10 min)

- Frequentist and Bayesian uncertainty
- Monte Carlo samplers
- PyMC3 Probabilistic programing

then stop (10 min)

- Running models
- Printing results
- Diagnosing model convergence

What is a galaxy?

NGC 4414 spiral galaxy

What is a galaxy?

Quick answer:

Bound system of gas, stars and dust

Good answer:

Light

What is light?

Photons we can see

What is darkness?

Photons we can’t see

What is a photon (γ)?

It's an itsy bitsy teenie weenie thingy with a certain energy

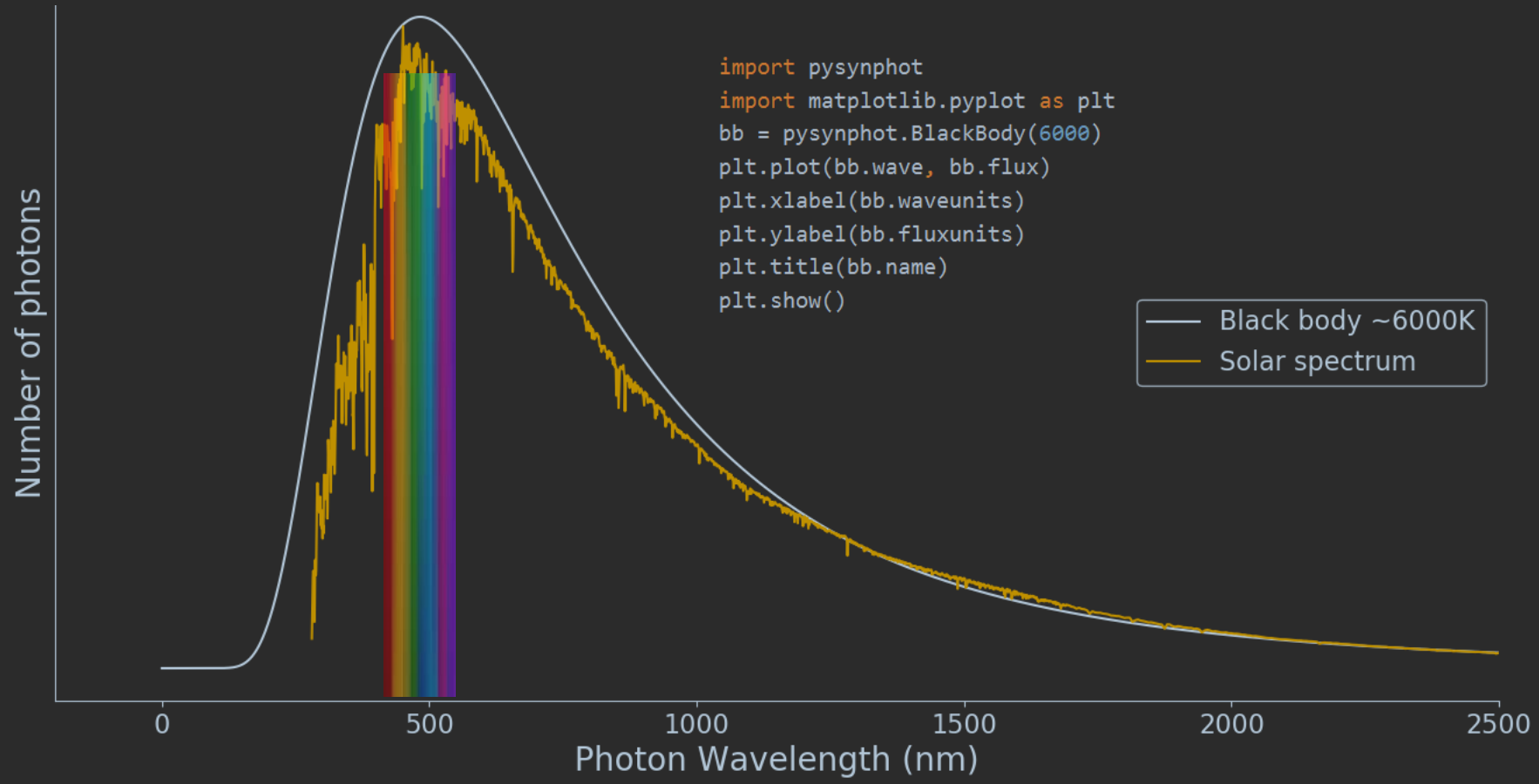
How to describe a bunch?

A Histogram

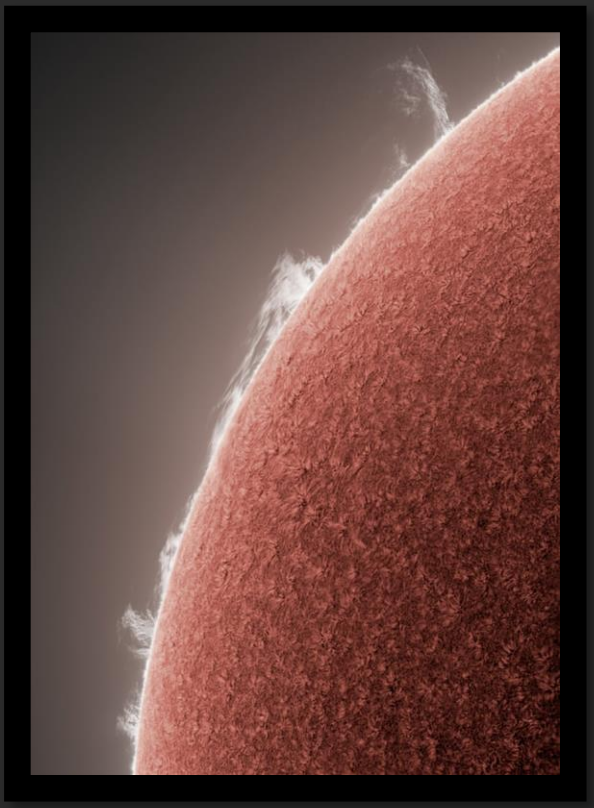
$$E = h\nu = \frac{hc}{\lambda}$$

Photon energy as a function of its frequency (ν) and wavelength (λ), where h is Plancks constant and c is the speed of light.

A spectrum is a celestial histogram which disentangles the photons properties



Sun and black body spectra comparison at ~6000K. Most Earthlings vision has adapted to the atmospheric “window” at peak photons flux



Sun monochrome black and white camera picture courtesy Alan Friedman

How to get spectra?

A) Download it!

Astroquery package in  The Astropy Project

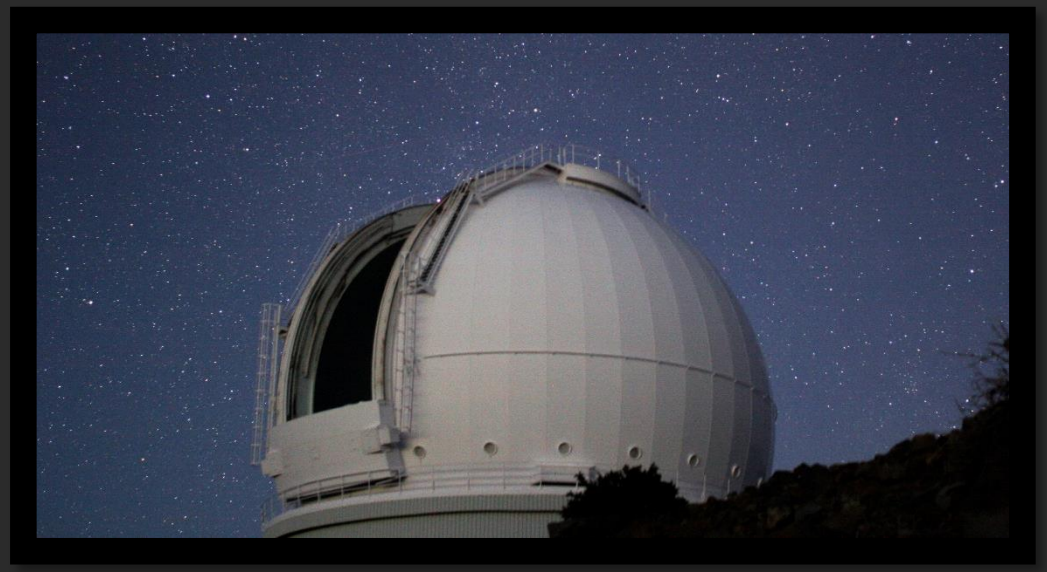
Vamdc Queries (astroquery.vamdc)
VizieR Queries (astroquery.vizier)
VO Simple Cone Search (astroquery.vo_conesearch)
VSA Queries (astroquery.vsa)
xMatch Queries (astroquery.xmatch)
ALFALFA Queries (astroquery.alfalfa)
CosmoSim Queries (astroquery.cosmosim)
Exoplanet Orbit Database (astroquery.exoplanet_orbit_database)
Fermi Queries (astroquery.fermi)
JPL Horizons Queries (astroquery.jplhorizons/astroquery.solarsystem.jpl.horizons)
JPL SBDB Queries (astroquery.jplsbdb/astroquery.solarsystem.jpl.sbdb)
LAMDA Queries (astroquery.lamda)
NASA Exoplanet Archive (astroquery.nasa_exoplanet_archive)
OAC API Queries (astroquery.oac)
OGLE Queries (astroquery.ogle)
Open Exoplanet Catalogue(astroquery.open_exoplanet_catalogue)
SDSS Queries (astroquery.sdss)
Spitzer Heritage Archive (astroquery.sha)
LMA Queries (astroquery.alma)
Atomic Line List (astroquery.atomic)
Besancon Queries (astroquery.besancon)
Cadc (astroquery.cadc)
CDS MOC Service (astroquery.cds)
esa.hubble (astroquery.esa.hubble)
ESASky Queries (astroquery.esasky)
ESO Queries (astroquery.eso)
Gaia TAP+ (astroquery.gaia)
GAMA Queries (astroquery.gama)
HEASARC Queries (astroquery.heasarc)
HITRAN Queries (astroquery.hitran)
IRSA Image Server program interface (IBE) Queries (astroquery.ibe)
IRSA Queries (astroquery.irsa)
IRSA Dust Extinction Service Queries (astroquery.irsa_dust)
JPL Spectroscopy Queries (astroquery.jplspec)
MAGPIS Queries (astroquery.magpis)
MAST Queries (astroquery.mast)
Minor Planet Center Queries (astroquery.solarsystem.MPC)
NASA ADS Queries (astroquery.nasa_ads)
NED Queries (astroquery.ned)
NIST Queries (astroquery.nist)
NRAO Queries (astroquery.nrao)
NVAS Queries (astroquery.nvas)
SIMBAD Queries (astroquery.simbad)
Skyview Queries (astroquery.skyview)
Splatalogue Queries (astroquery.splatalogue)
UKIDSS Queries (astroquery.ukidss)

Finding objects by type and coordinates

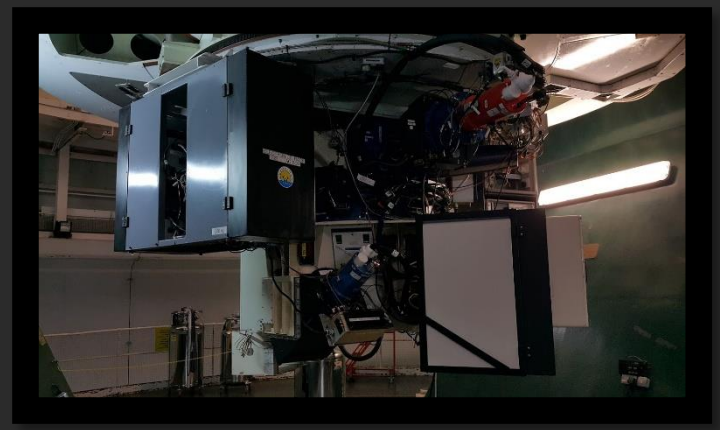
```
from astroquery.vizier import Vizier
from astropy import coordinates
from astropy import units as u

v = Vizier(keywords=['stars:white_dwarf'])

c = coordinates.SkyCoord(0, 0, unit=('deg', 'deg'), frame='icrs')
result = v.query_region(c, radius=2*u.deg)

result[0].pprint()
```

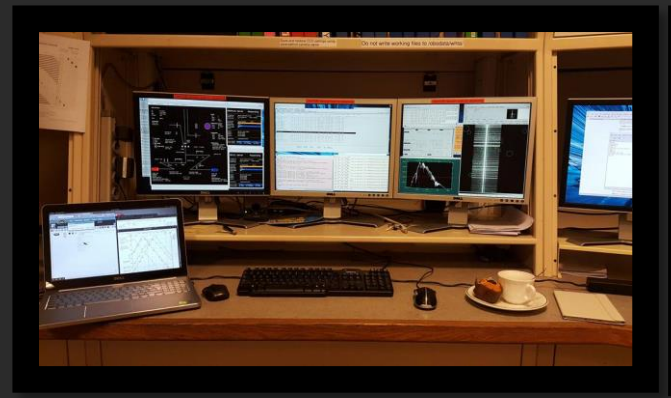
```
    LP     Rem Name  RA1950   DE1950  ...      pm     pmPA  _RA.icrs   _DE.icrs
                    "h:m:s"  "d:m:s"  ... arcsec / yr deg              "d:m:s"
-------- --- ---- -------- -------- ... ----------- ---- ---------- ---------
584-0063          00 03 23 +00 01.8 ...       0.219   93 00 05 56.8 +00 18 41
643-0083          23 50 40 +00 33.4 ...       0.197   93 23 53 13.7 +00 50 15
584-0030          23 54 05 -01 32.3 ...       0.199  193 23 56 38.8 -01 15 26
```

B) Get your own



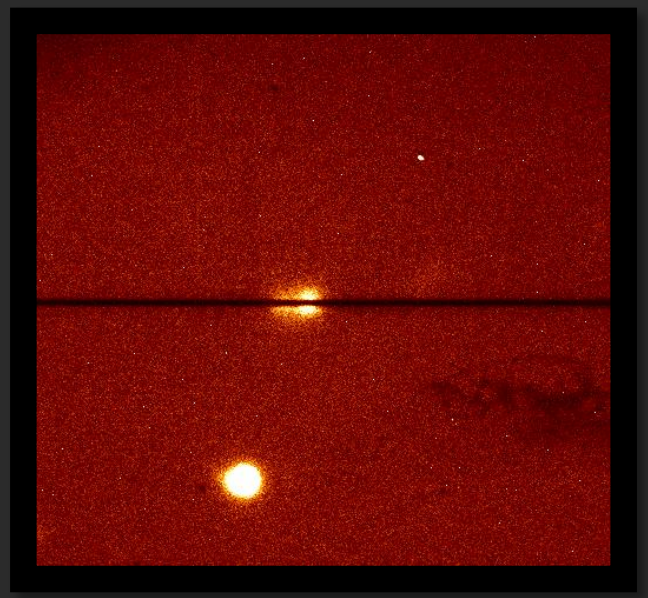
William Herschel Telescope at Roque de los Muchachos, isla La Palma, Canarias



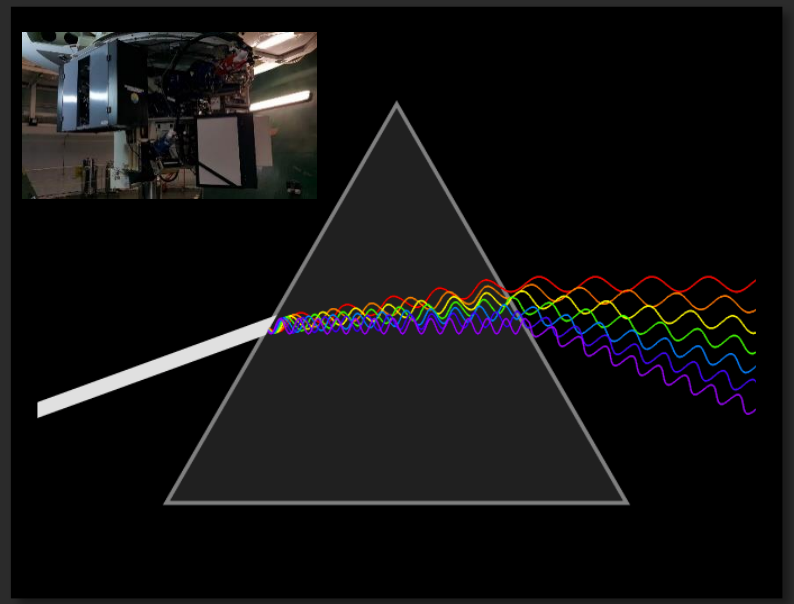


William Herschel Telescope insides

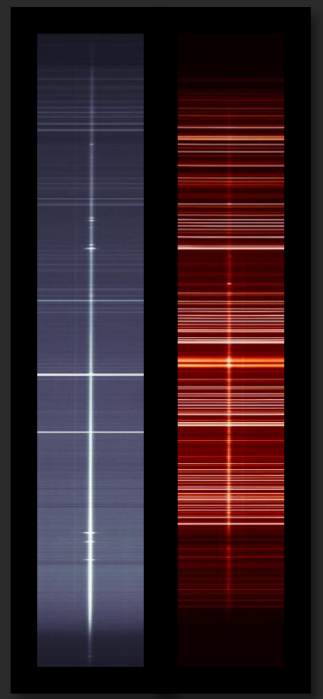
So let's pick a galaxy



William Herschel Telescope instrument ISIS image acquisition of galaxy IZwicky18



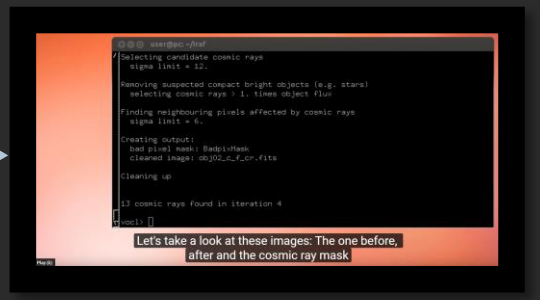
A spectrograph disperses the photons according to their wavelength



Output spectra 2D frames

… but you need to calibrate them

The user who doesn't [see every error as a worthy challenge] will likely be found naked, screaming, in the rain, on a dark night, in their local city park prior to being institutionalized.
- The zen of IRAF (abstract) A. Charles Pullen



Raw telescope data frames

Open libraries with image processing algorithms

IRAF and PyRAF

Python packages

specutils
An astropy package for spectroscopy

ccdproc
An Astropy image reduction package

Scientifically calibrated data frames

"Black box" tools provided by the instrument manufacturer

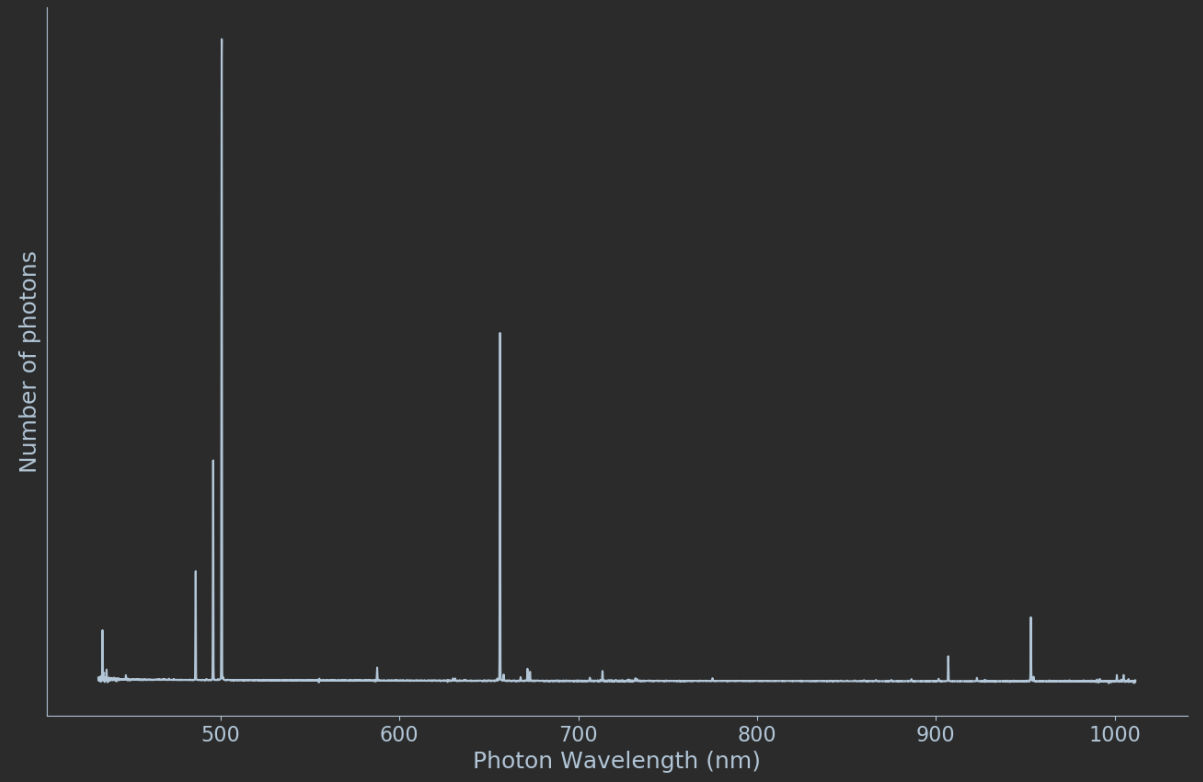
Many in python

- MEGARA at Gran Telescopio de Canarias
- Aztec at Gran Telescopio Milimetrico
- Pipeline at James Web space telescope

So let's take a look at this galaxy photons

IZwicky18 galaxy observation at the Hubble space telescope

Data calibration



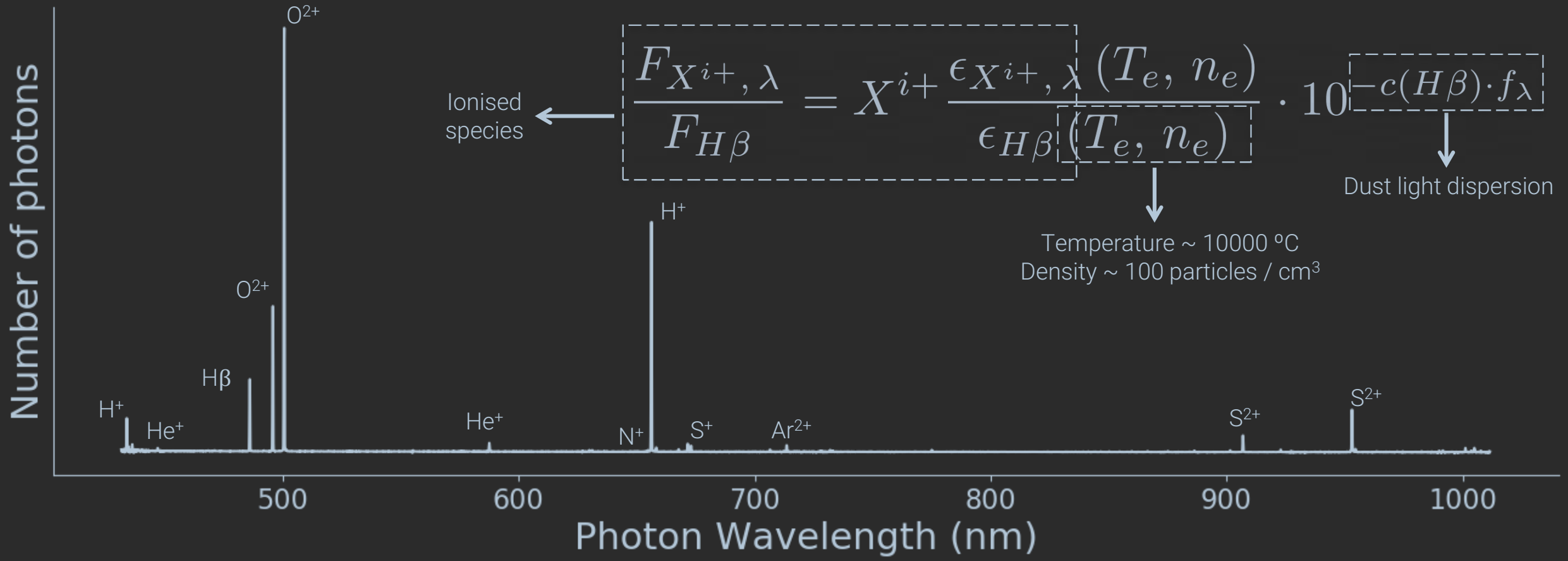
Izwicky18 spectrum from ISIS instrument at William Herschel telescope

Plasma ionisation process

Ionization process video tutorial in the orion Nebula using Cloudy (Ferland et al 2017) (nublado.org)

pyCloudy (Morriset et al 2016) can be used to lunch this program and extract its results efficiently

The photon spikes provide us with the gas chemistry



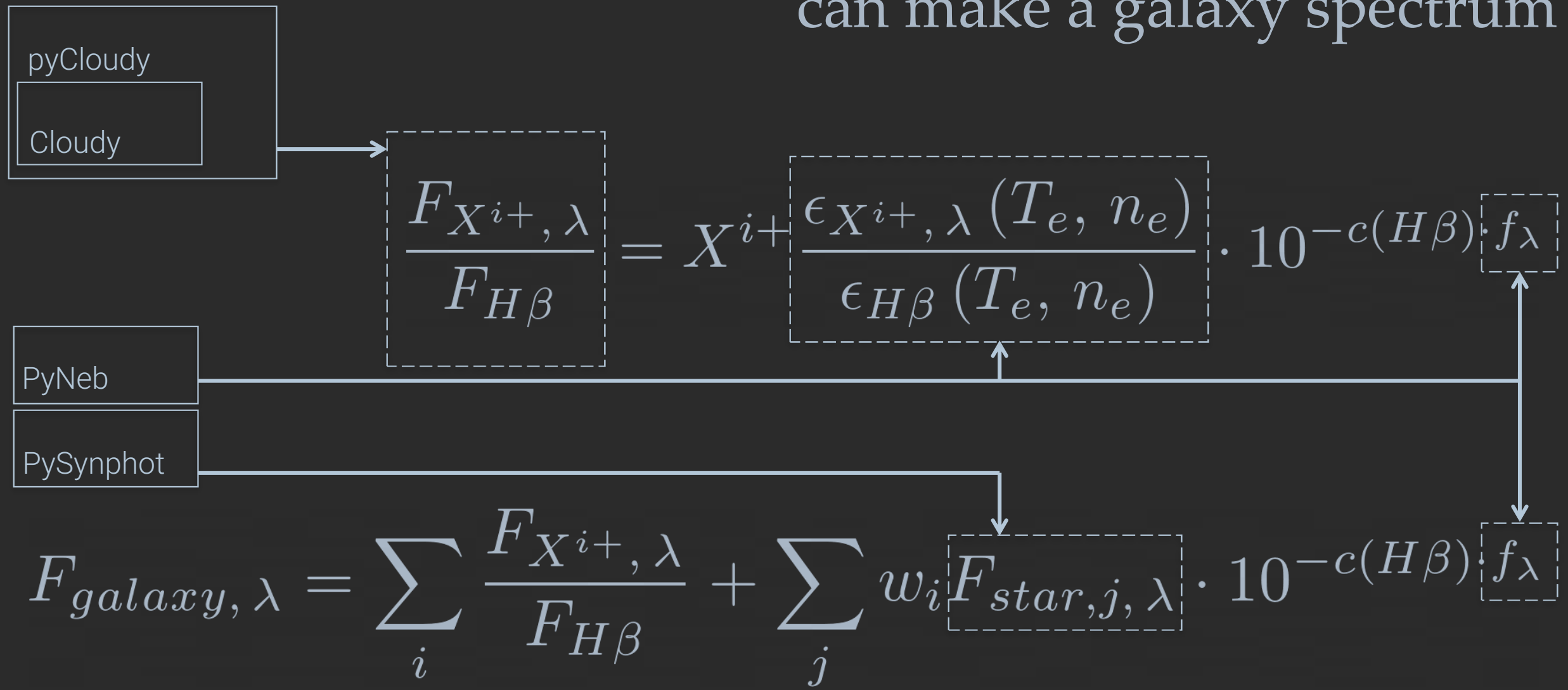
$$\frac{F_{X^{i+},\lambda}}{F_{H\beta}} = X^{i+}\frac{\epsilon_{X^{i+},\lambda}(T_e, n_e)}{\epsilon_{H\beta}(T_e, n_e)} \cdot 10^{-c(H\beta)\cdot f_\lambda}$$

Ionised species

Temperature ~ 10000 ºC
Density ~ 100 particles / cm^3

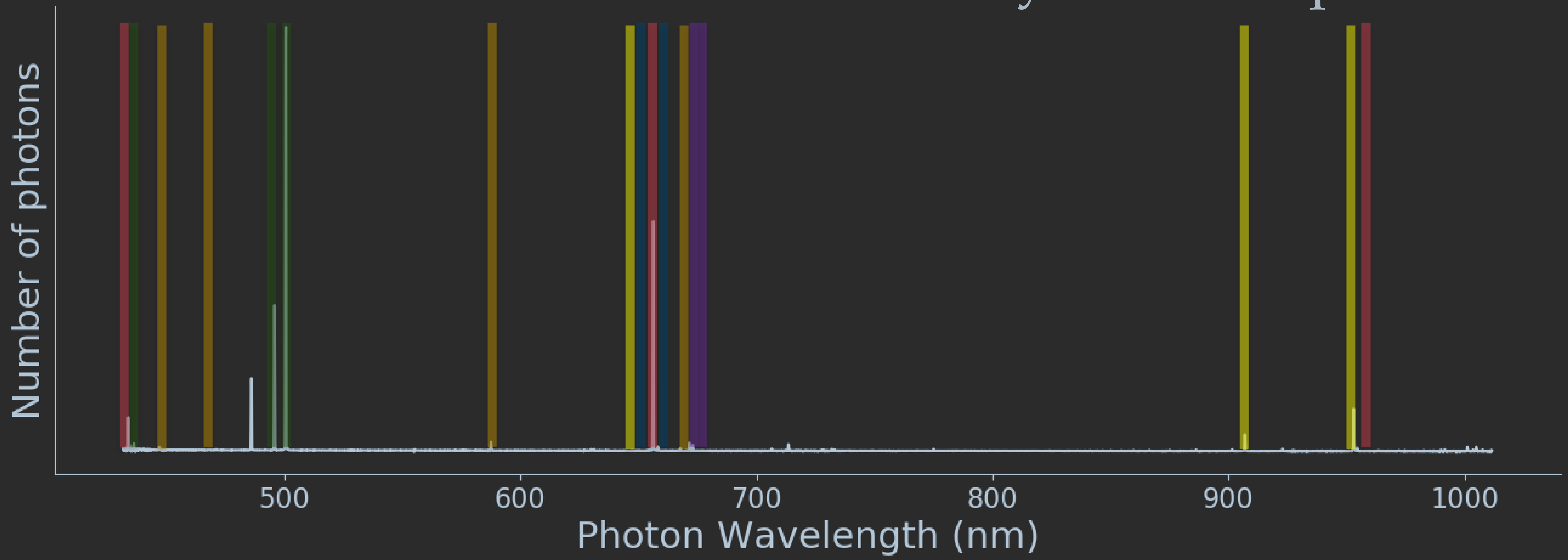
Dust light dispersion

With Python and a few equations you can make a galaxy spectrum



How to measure what a galaxy is made of?

Solve system of equations:



$$\frac{F_\lambda}{F_{H\beta}} = He^{+}\frac{\epsilon_{He^+,\lambda}(T_e,n_e)}{\epsilon_{H\beta}(T_e,n_e)}\cdot 10^{-c(H\beta)\cdot f_\lambda}$$

$$\frac{F_\lambda}{F_{H\beta}} = He^{+}\frac{\epsilon_{He^+,\lambda}(T_e,n_e)}{\epsilon_{H\beta}(T_e,n_e)}\cdot 10^{-c(H\beta)\cdot f_\lambda}$$

$$\frac{F_\lambda}{F_{H\beta}} = He^{+}\frac{\epsilon_{He^+,\lambda}(T_e,n_e)}{\epsilon_{H\beta}(T_e,n_e)}\cdot 10^{-c(H\beta)\cdot f_\lambda}$$

$$\frac{F_\lambda}{F_{H\beta}} = He^{2+}\frac{\epsilon_{He^{2+},\lambda}(T_e,n_e)}{\epsilon_{H\beta}(T_e,n_e)}\cdot 10^{-c(H\beta)\cdot f_\lambda}$$

$$\frac{F_\lambda}{F_{H\beta}} = H^{+}\frac{\epsilon_{H^+,\lambda}(T_e,n_e)}{\epsilon_{H\beta}(T_e,n_e)}\cdot 10^{-c(H\beta)\cdot f_\lambda}$$

$$\frac{F_\lambda}{F_{H\beta}} = H^{+}\frac{\epsilon_{H^+,\lambda}(T_e,n_e)}{\epsilon_{H\beta}(T_e,n_e)}\cdot 10^{-c(H\beta)\cdot f_\lambda}$$

$$\frac{F_\lambda}{F_{H\beta}} = O^{+}\frac{\epsilon_{O^+,\lambda}(T_e,n_e)}{\epsilon_{H\beta}(T_e,n_e)}\cdot 10^{-c(H\beta)\cdot f_\lambda}$$

$$\frac{F_\lambda}{F_{H\beta}} = O^{+}\frac{\epsilon_{O^+,\lambda}(T_e,n_e)}{\epsilon_{H\beta}(T_e,n_e)}\cdot 10^{-c(H\beta)\cdot f_\lambda}$$

$$\frac{F_\lambda}{F_{H\beta}} = O^{2+}\frac{\epsilon_{O^{2+},\lambda}(T_e,n_e)}{\epsilon_{H\beta}(T_e,n_e)}\cdot 10^{-c(H\beta)\cdot f_\lambda}$$

$$\frac{F_\lambda}{F_{H\beta}} = O^{2+}\frac{\epsilon_{O^{2+},\lambda}(T_e,n_e)}{\epsilon_{H\beta}(T_e,n_e)}\cdot 10^{-c(H\beta)\cdot f_\lambda}$$

$$\frac{F_\lambda}{F_{H\beta}} = O^{2+}\frac{\epsilon_{O^{2+},\lambda}(T_e,n_e)}{\epsilon_{H\beta}(T_e,n_e)}\cdot 10^{-c(H\beta)\cdot f_\lambda}$$

$$\frac{F_\lambda}{F_{H\beta}} = S^{+}\frac{\epsilon_{S^+,\lambda}(T_e,n_e)}{\epsilon_{H\beta}(T_e,n_e)}\cdot 10^{-c(H\beta)\cdot f_\lambda}$$

$$\frac{F_\lambda}{F_{H\beta}} = S^{+}\frac{\epsilon_{S^+,\lambda}(T_e,n_e)}{\epsilon_{H\beta}(T_e,n_e)}\cdot 10^{-c(H\beta)\cdot f_\lambda}$$

$$\frac{F_\lambda}{F_{H\beta}} = S^{2+}\frac{\epsilon_{S^+,\lambda}(T_e,n_e)}{\epsilon_{H\beta}(T_e,n_e)}\cdot 10^{-c(H\beta)\cdot f_\lambda}$$

$$\frac{F_\lambda}{F_{H\beta}} = S^{2+}\frac{\epsilon_{S^+,\lambda}(T_e,n_e)}{\epsilon_{H\beta}(T_e,n_e)}\cdot 10^{-c(H\beta)\cdot f_\lambda}$$

$$\frac{F_\lambda}{F_{H\beta}} = S^{+}\frac{\epsilon_{S^+,\lambda}(T_e,n_e)}{\epsilon_{H\beta}(T_e,n_e)}\cdot 10^{-c(H\beta)\cdot f_\lambda}$$

$$\frac{F_\lambda}{F_{H\beta}} = N^{+}\frac{\epsilon_{N^+,\lambda}(T_e,n_e)}{\epsilon_{H\beta}(T_e,n_e)}\cdot 10^{-c(H\beta)\cdot f_\lambda}$$

$$\frac{F_\lambda}{F_{H\beta}} = N^{+}\frac{\epsilon_{N^+,\lambda}(T_e,n_e)}{\epsilon_{H\beta}(T_e,n_e)}\cdot 10^{-c(H\beta)\cdot f_\lambda}$$

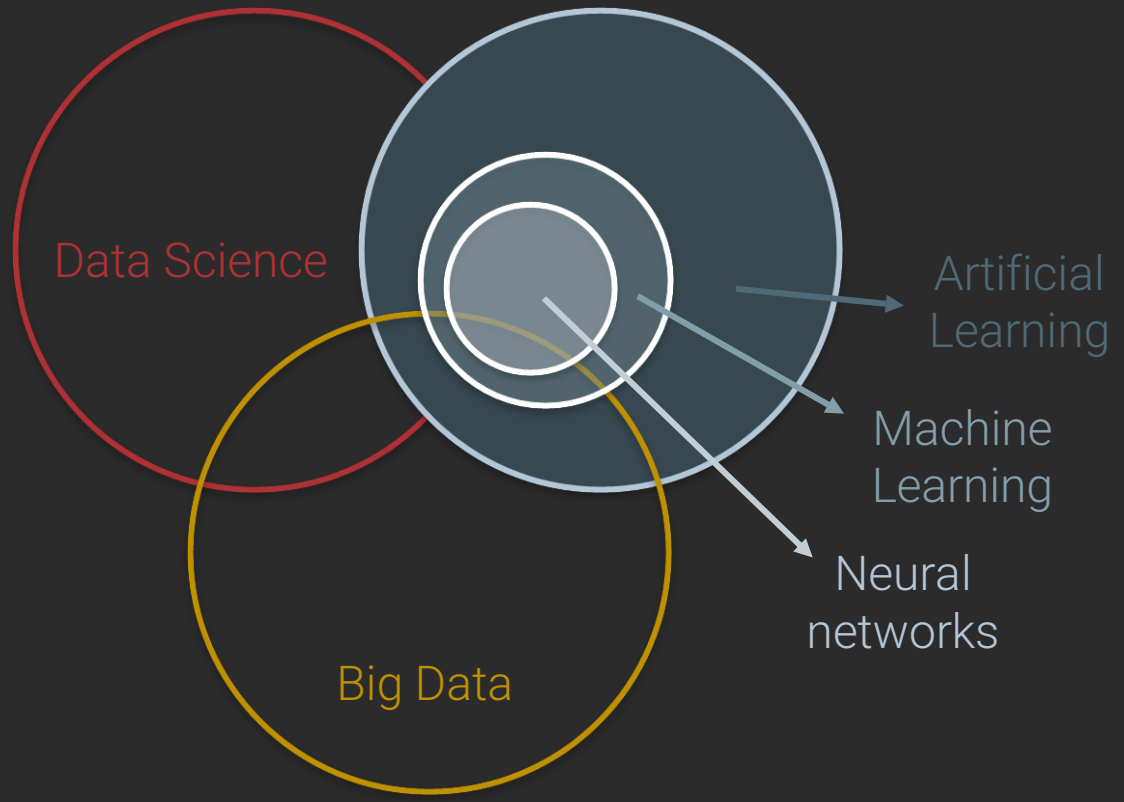
$$\frac{F_\lambda}{F_{H\beta}} = Ar^{2+}\frac{\epsilon_{Ar^{2+},\lambda}(T_e,n_e)}{\epsilon_{H\beta}(T_e,n_e)}\cdot 10^{-c(H\beta)\cdot f_\lambda}$$

$$\frac{F_\lambda}{F_{H\beta}} = Ar^{2+}\frac{\epsilon_{Ar^{2+},\lambda}(T_e,n_e)}{\epsilon_{H\beta}(T_e,n_e)}\cdot 10^{-c(H\beta)\cdot f_\lambda}$$

$$\frac{F_\lambda}{F_{H\beta}} = Ar^{3+}\frac{\epsilon_{Ar^{3+},\lambda}(T_e,n_e)}{\epsilon_{H\beta}(T_e,n_e)}\cdot 10^{-c(H\beta)\cdot f_\lambda}$$

We are still very far from Rosenblatt (1957) prediction:

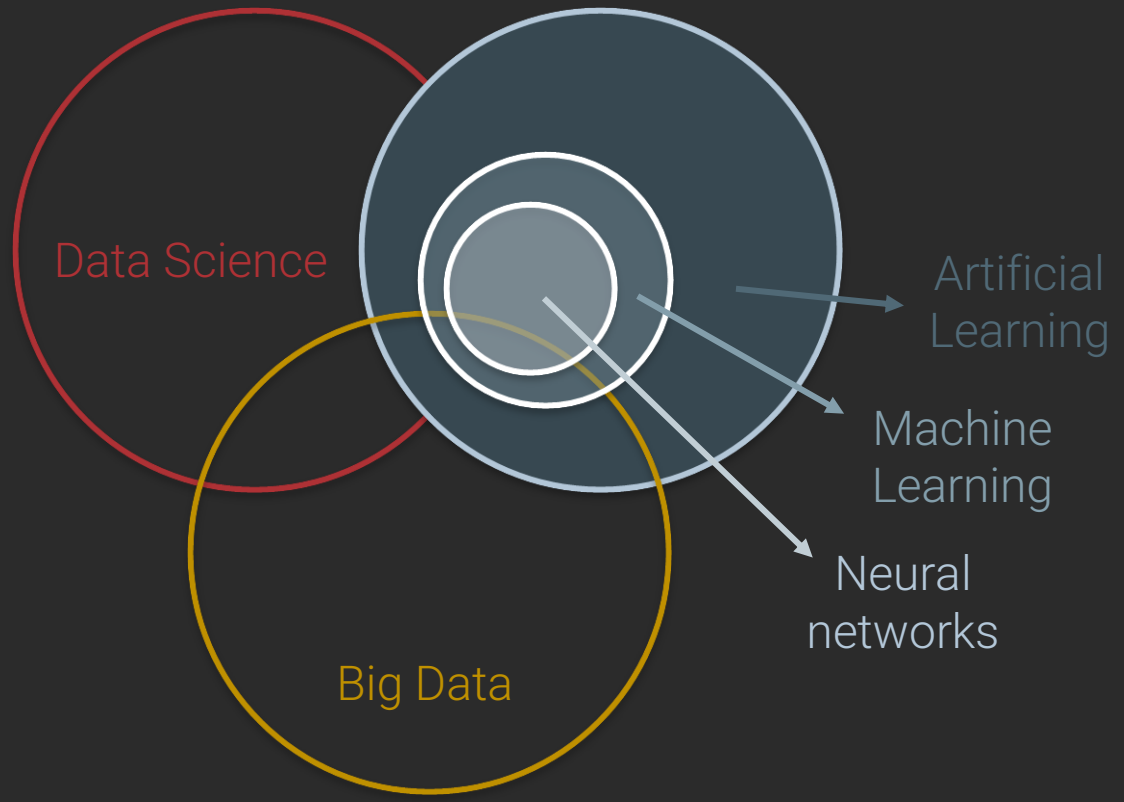
Since the advent of electronic computers [...] an in-creasing amount of attention has been focused on the feasibility of constructing **a device possessing such human-like functions as perception, recognition, concept formation** and the ability to generalize from experience



ESO (European Southern Observatory) Machine Learning workshop 2019
(® Fraunhofer FOKUS)

We are still very far from Rosenblatt (1957) prediction:

a device capable of moving through very complex spaces



ESO (European Southern Observatory) Machine Learning workshop 2019
(® Fraunhofer FOKUS)

Evaluate model against data

Probabilistic programming library PyMC

PyMC2 implementing a
Patil et al (2010)

PyMC3 implementing a
Salvatier et al (2016)

PyMC4 implementing a

Programming language package

MCMC algorithm using a
Markov Chain Monte Carlo

HMC algorithm using a
Hamiltonian Monte Carlo

HMC algorithm using a
Hamiltonian Monte Carlo

Mathematical principle

Adaptive metropolis sampler based on
Haario et al (2001)

NUTs sampler based on
Hoffman et al. (2011)

NUTs sampler based on
Hoffman et al. (2011)

Numerical model

numpy arrays
Oliphant (2006)

tensors
Theano Development Team (2016)

tensors
Tensorflow by Abadi et al (2016)

Memory unit

Probabilistic programming library PyMC

PyMC2 implementing a MCMC algorithm using a Adaptive metropolis sampler based on numpy arrays

- Advantages:Easy to use, great compatibility with external libraries
- Disadvantages: Poor convergence and speed as the number of dimensions increases O(D2)

PyMC3 implementing a HMC algorithm using a NUTs sampler based on tensors

- Advantages: Great convergence and speed for complex models O(D5/4)
- Disadvantages: Hard to code and incompatible with external functions

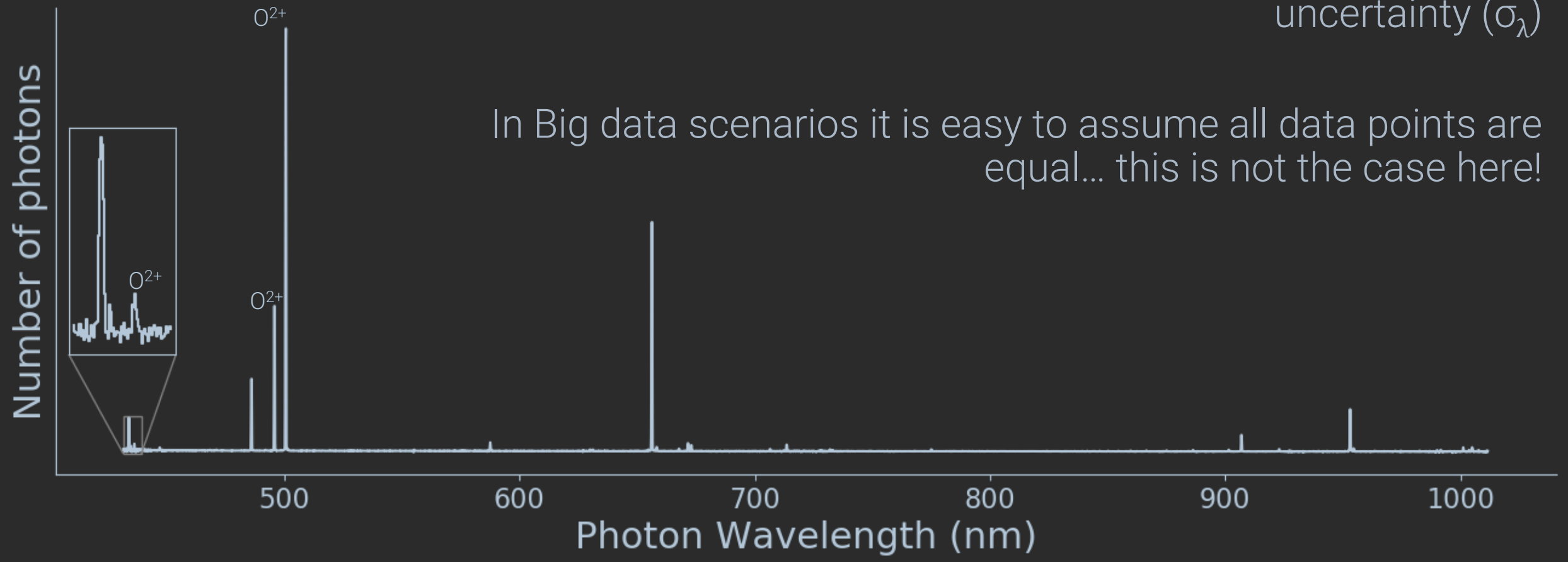
PyMC4 implementing a HMC algorithm using a NUTs sampler based on tensors

- Advantages: Larger Tensorflow community
- Disadvantages: In development

We have two inputs:

Emission line photons flux (F_λ) from every transition and its uncertainty (σ_λ)

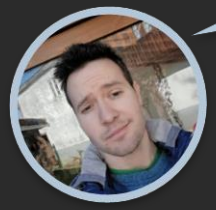
In Big data scenarios it is easy to assume all data points are equal… this is not the case here!



Understanding measurement through its opposite: Uncertainty

In a far away PyCon …

How would you handle data points which such different error magnitude?



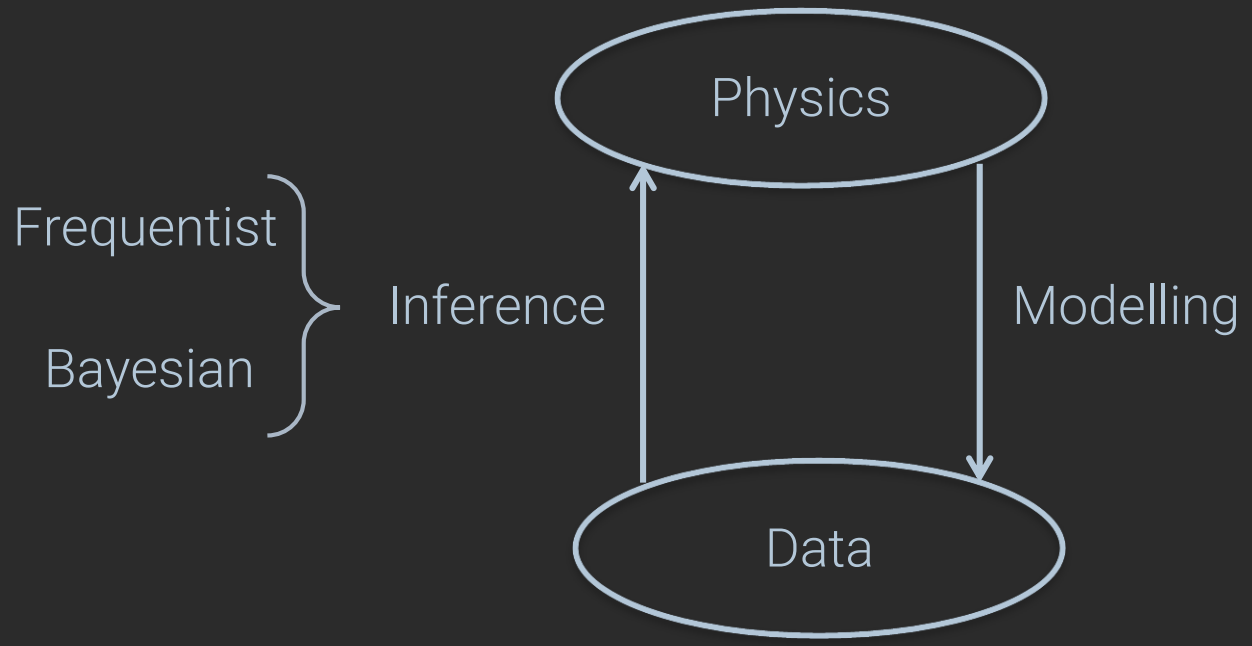
FFFFFFFFFFFFFFFFFFFFFFUUUUUUUUUUU

Data does not have error

Chris Fonnesbeck core
PyMC3 developer

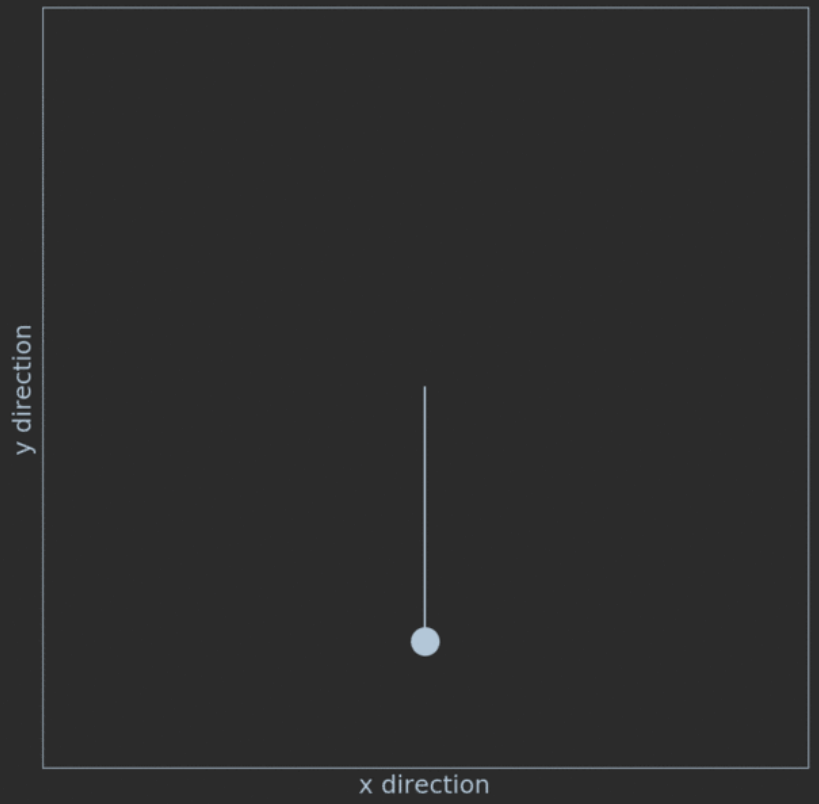
… Yet there is a lot of insight here

Inference purpose



Uncertainty is the main disagreement between both inference paradigms

Inference example: Pendulum period uncertainty



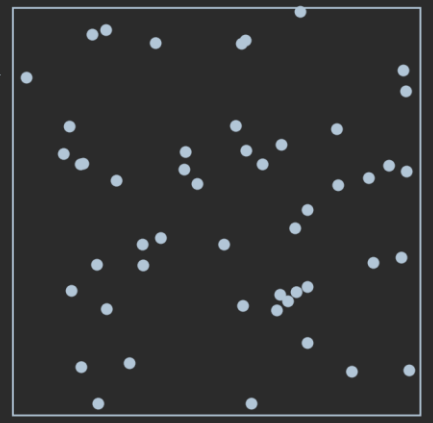
Matplotlib animation tutorial
Jake Vanderplas

- **Frequentist interpretation:** The uncertainty is intrinsic to the measurement. As the sample size increases so does the accuracy and precision in your model parameters.

- **Bayesian interpretation:** The measurement is always true. Any intrinsic randomness is caused by the physical properties, which behave as a distribution. The more you sample this distribution the larger confidence you in the physical properties.

Another example: Electrons temperature in our model

$$\frac{F_{X^{i+},\lambda}}{F_{H\beta}} = X^{i+} \frac{\epsilon_{X^{i+},\lambda}(T_e, n_e)}{\epsilon_{H\beta}(T_e, n_e)} \cdot 10^{-c(H\beta)\cdot f_\lambda}$$



Matplotlib animation tutorial
Jake Vanderplas

It can be proven that the plasma temperature behaves as a gaussian distribution because the electrons speed follows a gaussian Maxwellian distribution

Which inference paradigm to follow?

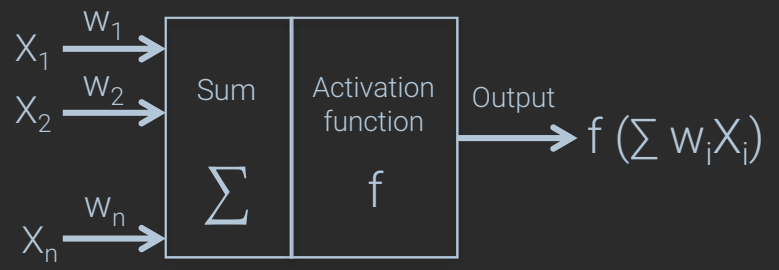
- Both frequentist and Bayesian schemes provide useful tools to fit a model to data
- Moreover, the numerical implementation can be very similar for both principles

In practice, however, all Probabilistic programming libraries are intrinsically Bayesian

- A Bayesian model is easier to implement in high dimensional models

$$P(\theta|y) = \frac{P(y|\theta)\,P(\theta)}{P(y)}$$

- A Bayesian model has a large affinity with neural networks



What does a probabilistic programming library do for you?

$$P(\theta|y) = \frac{P(y|\theta)\,P(\theta)}{P(y)}$$

- Sample: Draw values from probability distributions
- Condition: Use input data to update probabilities
- Infer: Measure something from the data given a model

$$P(\theta|y) = \frac{P(y|\theta)\,P(\theta)}{P(y)}$$

Prior definition

P(θ) represents our knowledge on the model parameters before seeing any data

Un-informative priors are expected since they cover a wide parameter range and have small impact in the sampling

In contrast, formative priors are commonly criticised because they condition the sampling.

$$P(\theta|y) = \frac{P(y|\theta)\,P(\theta)}{P(y)}$$

Pymc3 prior definition

```
import pymc3

with pymc3.Model() as model:

    T_low = pymc3.Normal('Temperature_low', mu=15000.0, sd=5000.0)
    T_high = pymc3.Normal('Temperature_high', mu=15000.0, sd=5000.0)
    n_e = pymc3.Normal('density', mu=75.0, sd=25.0)
    cHbeta = pymc3.Lognormal('cHbeta', mu=0, sd=1)
    O2 = pymc3.Normal('O^+ abundance', mu=5.0, sd=5.0)
    O3 = pymc3.Normal('O^2+ abundance', mu=5.0, sd=5.0)
    S2 = pymc3.Normal('S^+ abundance', mu=5.0, sd=5.0)
    S3 = pymc3.Normal('S^2+ abundance', mu=5.0, sd=5.0)
    Ar3 = pymc3.Normal('Ar^2+ abundance', mu=5.0, sd=5.0)
    Ar4 = pymc3.Normal('Ar^3+ abundance', mu=5.0, sd=5.0)
    N2 = pymc3.Normal('N^+ abundance', mu=5.0, sd=5.0)
    He1 = pymc3.Lognormal('He^1+ abundance', mu=0, sd=1) * 0.01
    He2 = pymc3.Lognormal('He^2+ abundance', mu=0, sd=1) * 0.001
```

Compute the theoretical model from these variables

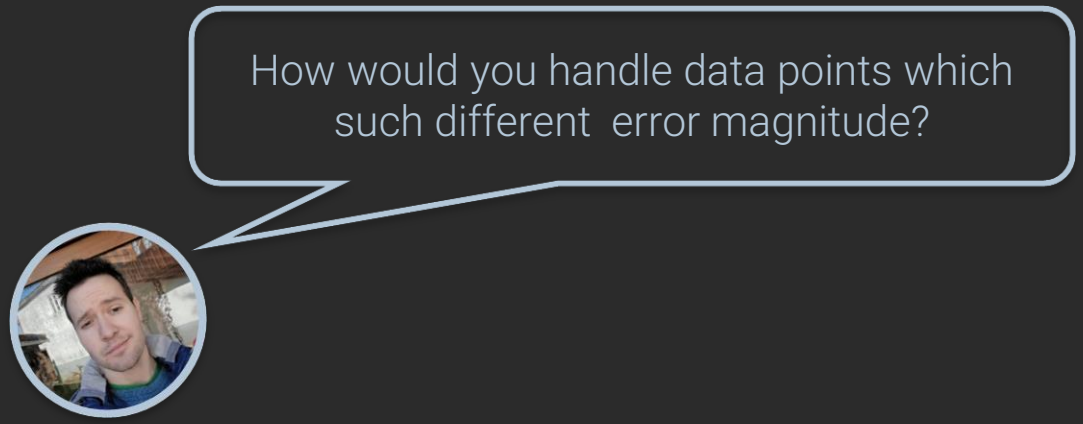
$$\frac{F_{X^{i+},\lambda}}{F_{H\beta}} = X^{i+}\frac{\epsilon_{X^{i+},\lambda}(T_e, n_e)}{\epsilon_{H\beta}(T_e, n_e)} \cdot 10^{-c(H\beta)\cdot f_\lambda}$$

```
import theano.tensor as tt
from theano import function


def emisRatioFunction(temp, den, k1, k2, k3, k4):
    return k1 + k2 / (temp / 10000.0) + k3 * tt.log10(temp / 10000) + tt.log10(1 + k4 * den)


def fluxFunction(abund, emisRatio, flambda, cHbeta):
    return abund * emisRatio * tt.power(10, - cHbeta * flambda)


temp, den, k1, k2, k3, k4 = tt.dscalars('temp', 'den', 'k1', 'k2', 'k3', 'k4')
abund, emisRatio, flambda, cHbeta = tt.dscalars('abund', 'emisRatio', 'flambda', 'cHbeta')

emisRatio_tFunction = function(inputs=[temp, den, k1, k2, k3, k4],
                               outputs=emisRatioFunction(temp, den, k1, k2, k3, k4),
                               on_unused_input='ignore')

flux_tFunction = function(inputs=[abund, emisRatio, flambda, cHbeta],
                          outputs=fluxFunction(abund, emisRatio, flambda, cHbeta),
                          on_unused_input='ignore')
```

$$P(\theta|y) = \frac{P(y|\theta)\,P(\theta)}{P(y)}$$

Likelihood definition

P(y | θ) it provides an evaluation between the theory and the data

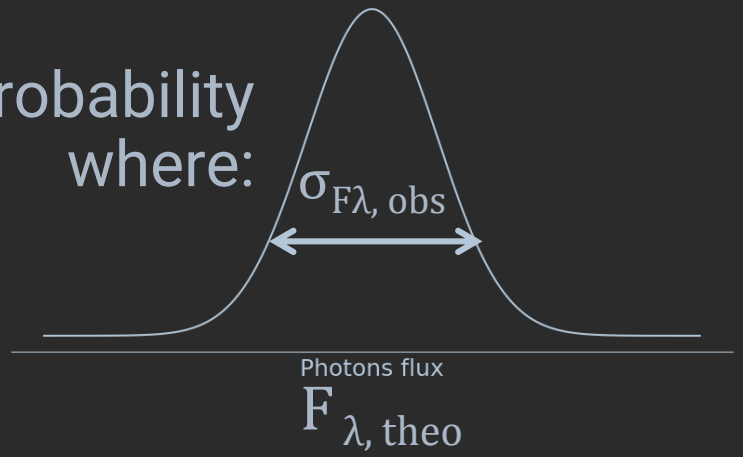


… one way to do it

$$P(\theta|y) = \frac{P(y|\theta)\,P(\theta)}{P(y)}$$

Gaussian likelihood definition to account for data uncertainty

Define a normal likelihood probability where:




```
k1, k2, k3, k4 = coeffs_dict['O3']
emisRatio_i = emisRatio_tFunction(T_high, T_low, k1, k2, k3, k4)
theoFlux = flux_tFunction(O3, emisRatio_i, cHbeta, flamba_i)
likelihood_i = pymc3.Normal('likelihood_i', mu=theoFlux, sd=obsErr[i], observed=obsFlux[i])
```

$$\frac{F_{X^{i+},\lambda}}{F_{H\beta}} = X^{i+}\frac{\epsilon_{X^{i+},\lambda}(T_e,\,n_e)}{\epsilon_{H\beta}(T_e,\,n_e)}\cdot 10^{-c(H\beta)\cdot f_\lambda}$$

$$P(\theta|y) = \frac{P(y|\theta)\,P(\theta)}{P(y)}$$

Posterior definition

This is the result we want to compute. It provides us with the parameter credible region: "Given our observed data, there is a 95% probability that the true value lies in this distribution”

… but we are missing P(y), the evidence. This is the probability the observational has been generated by the theoretical model.

$$P(y) = \int P(y|\theta)\,P(\theta)$$

Posterior sampling

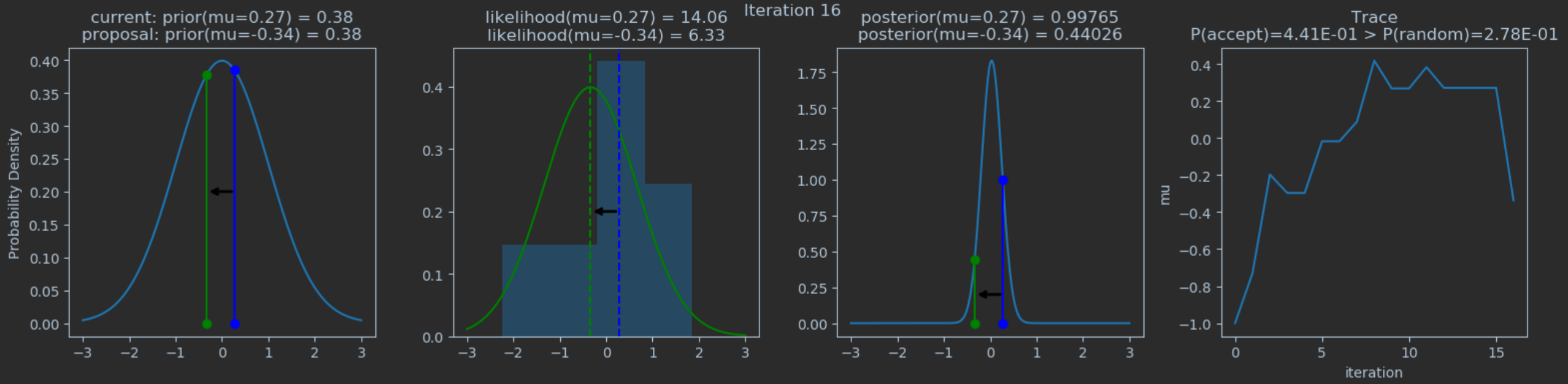
$$\frac{P(\theta|y)_{new}}{P(\theta|y)_{old}} = \frac{P(y|\theta)_{new}P(\theta)_{new}}{P(y|\theta)_{old}P(\theta)_{old}}$$

At each successful iteration, the sampler moves to a new coordinate and the posterior distribution gains a new value

A successful iteration does not always imply a coordinate with a higher probability

Sampler philosophy

The Adaptive metropolis sampler believes that in your path you must fail as many times as you suceed



Thomas Wiecki: MCMC sampling for dummies

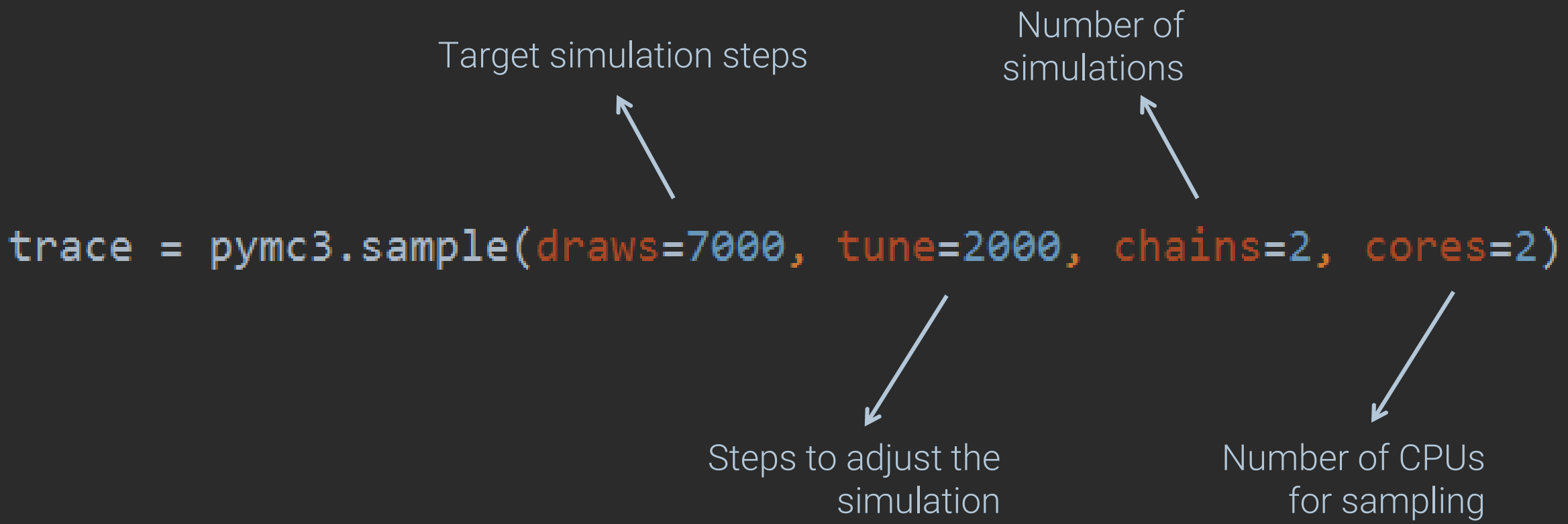
Sampler philosophy

The NUTs (No U Turns) sampler believes that if you have a **path guide**, you don't need to fail so many time.

The NUTS take advantage of the automatic gradient differentiation from the likelihood to reach the target coordinate.

It also avoids over sampling by avoiding U – Turns

Finally you run it!



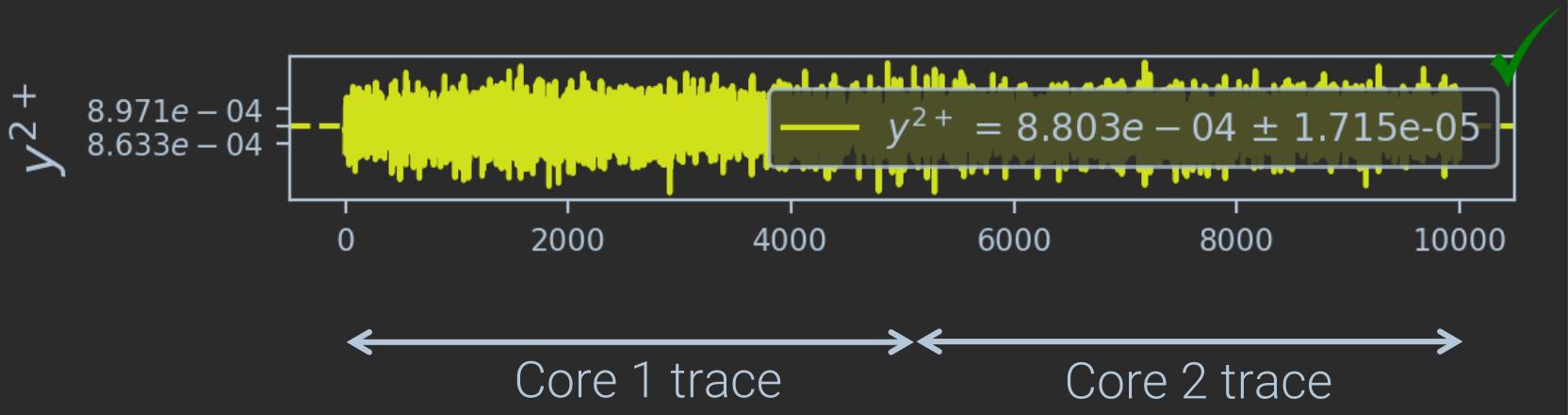
How do we know
the results are correct?

Test fitting accuracy with synthetic data

$$\frac{F_{X^{i+},\lambda}}{F_{H\beta}} = X^{i+}\frac{\epsilon_{X^{i+},\lambda}\left(T_e,\, n_e\right)}{\epsilon_{H\beta}\left(T_e,\, n_e\right)} \cdot 10^{-c(H\beta)\cdot f_\lambda}$$

Parameter	True value
T_{low}	15590
n_e	500
S^{+}	5.48
S^{2+}	6.36
O^{+}	7.80
O^{2+}	8.05
Ar^{2+}	5.72
Ar^{3+}	5.06
N^{+}	5.84
$c(H\beta)$	0.100
T_{high}	16000
y^{+}	0.0850
y^{2+}	0.00088
τ	1.0

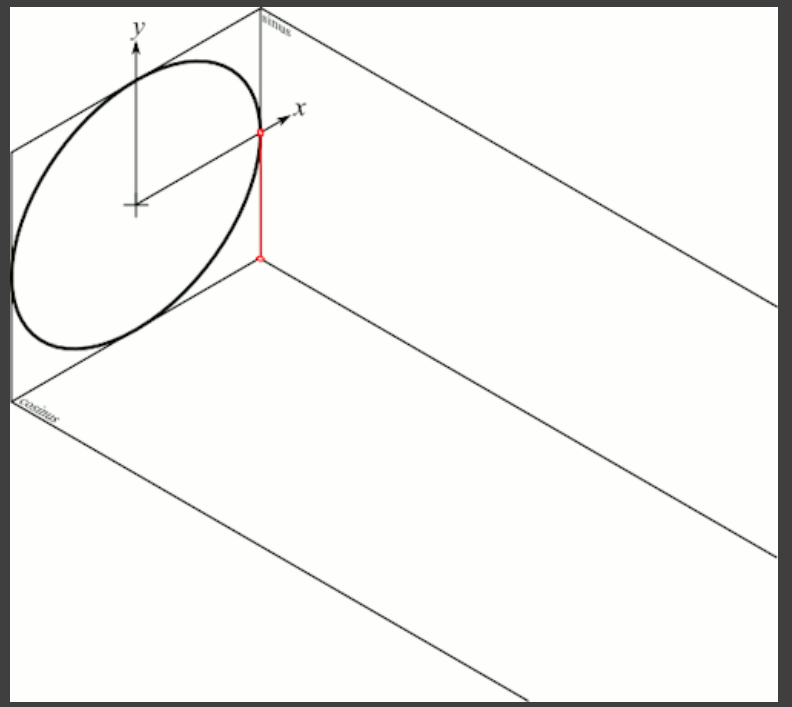
Traces statistics



The traces represent each parameter coordinate during the simulation

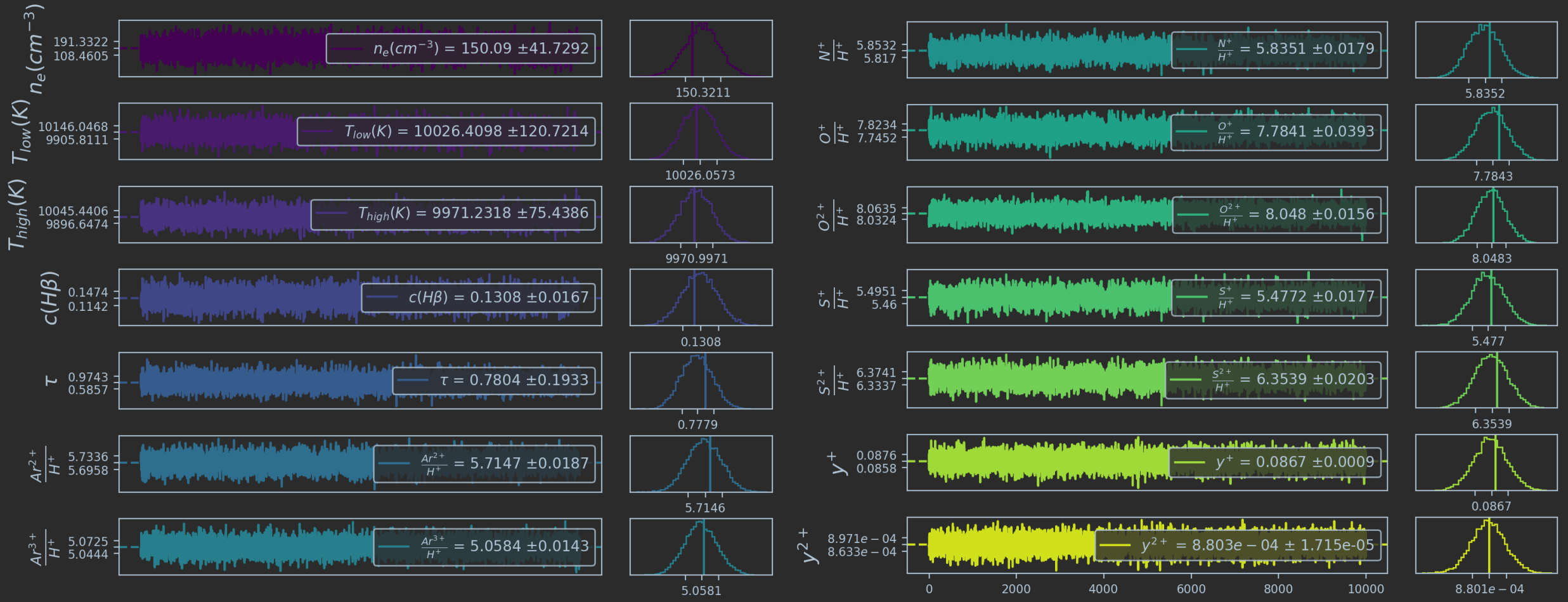
Graphical way to check for divergences in the chain but the probabilistic library should warn you

Several runs are necessary to confirm that the initial conditions are not affecting the results



Visualizing trigonometric waves from a circle.
https://gemmathematics.tumblr.com

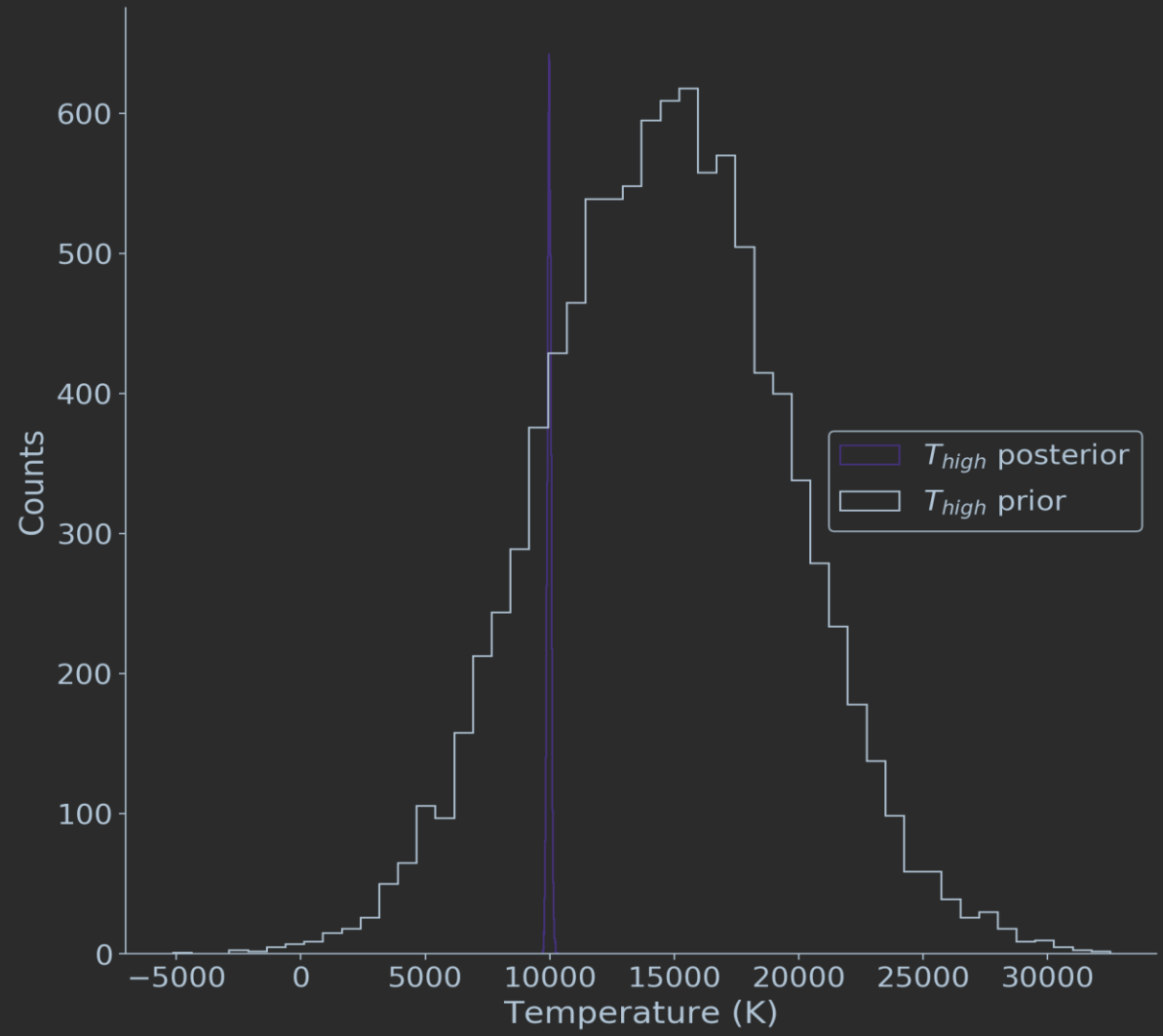
Traces statistics



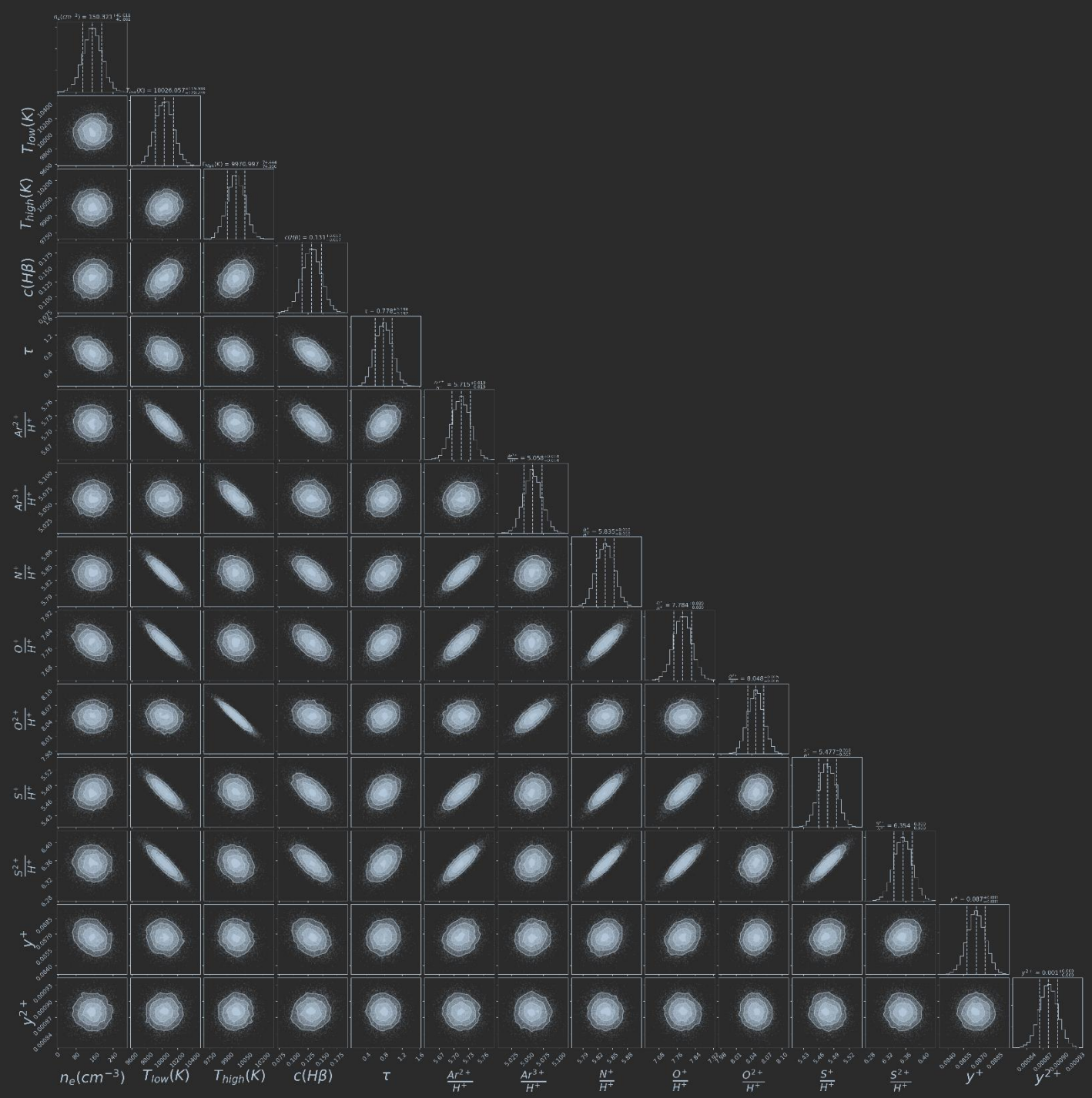
The trace histograms represent the posterior distributions

Traces statistics

Prior – Posterior comparison

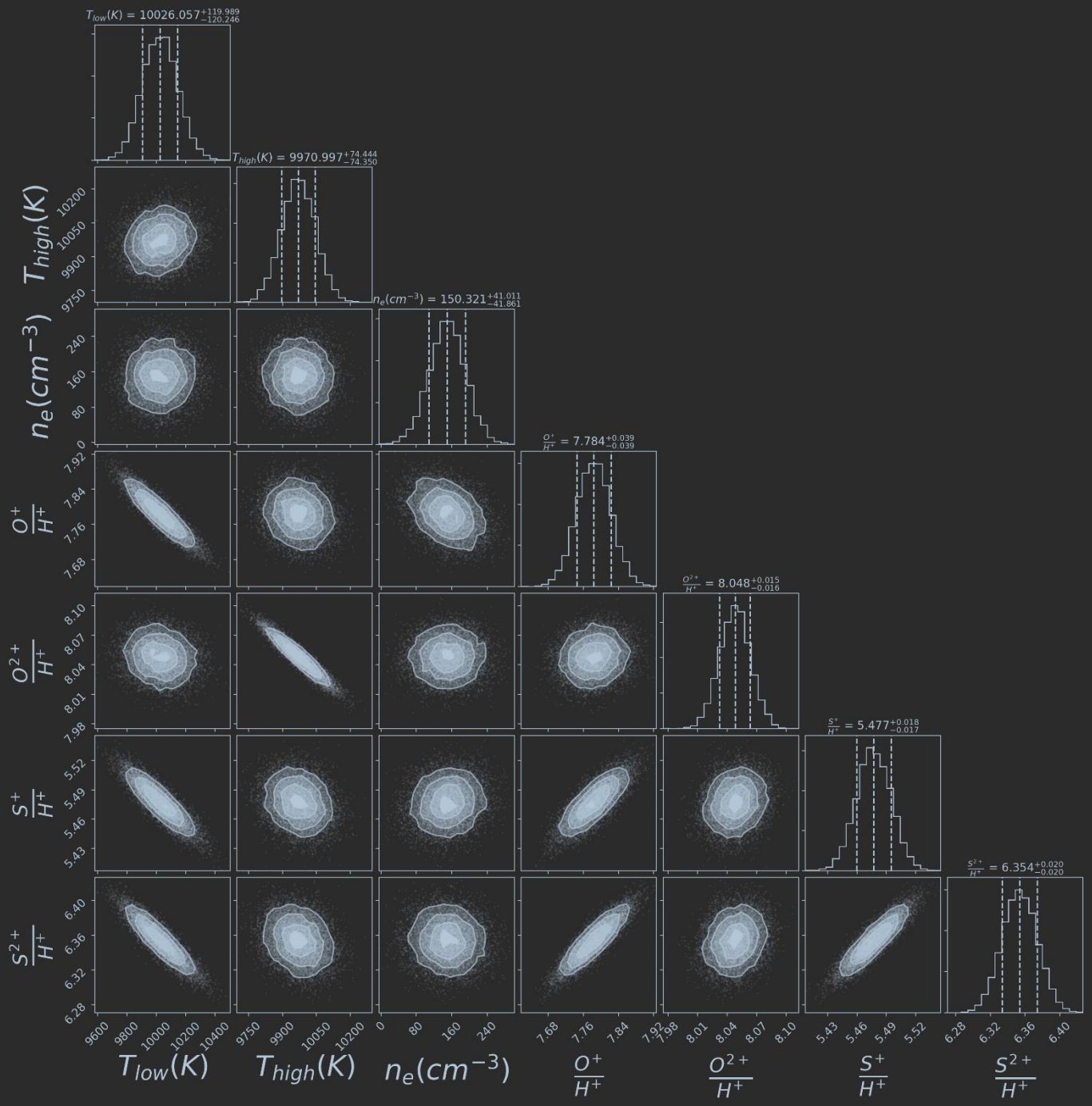


Parameters correlation



Scatter plot matrix from corner.py by Foreman-Mackey

Parameters correlation



Scatter plot matrix from corner.py by Foreman-Mackey

Model reparameterization

$$\underbrace{X^{i+} = \sigma\theta + \mu}_{\frac{F_{X^{i+},\lambda}}{F_{H\beta}} = X^{i+}\frac{\epsilon_{X^{i+},\lambda}\left(T_e,\, n_e\right)}{\epsilon_{H\beta}\left(T_e,\, n_e\right)} \cdot 10^{-c(H\beta)\cdot f_\lambda}}$$

```
O2_prior = pymc3.Normal('O^+ abundance', mu=0.0, sd=1.0)
O2 = 5.0 * O2_prior + 5.0
```

- In this approach, we don’t define the parameter space according to physical dimensions. Instead, we reshape it numerically model to make the parameter domains as even and linear as possible.
- This makes the sampling faster and it can decrease the parameter correlation.

Conclusions

- The Python community is responsible for remarkable break throughs in the astrophysics field
- Neural networks make possible the implementation of models with a size and complexity not seen before
- Probabilistic programing libraries, such as PyMC3, make the fitting of Bayesian models easy. However, it is necessary to be mindful of the results

Some resources

For those interested in Probability theory:

- Michael Betancourt: Probability Theory (For Scientists and Engineers)
https://betanalpha.github.io/assets/case_studies/probability_theory.html

- David S. Rosenberg: Foundations of Machine Learning
https://bloomberg.github.io/foml/#home

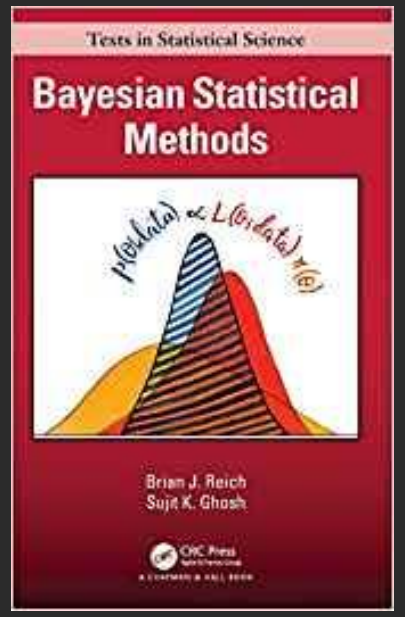
For those interested in PyMC3:

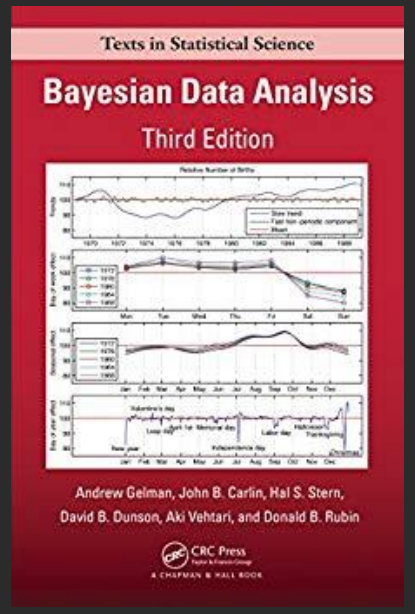
- Luciano Paz Blog
https://lucianopaz.github.io/

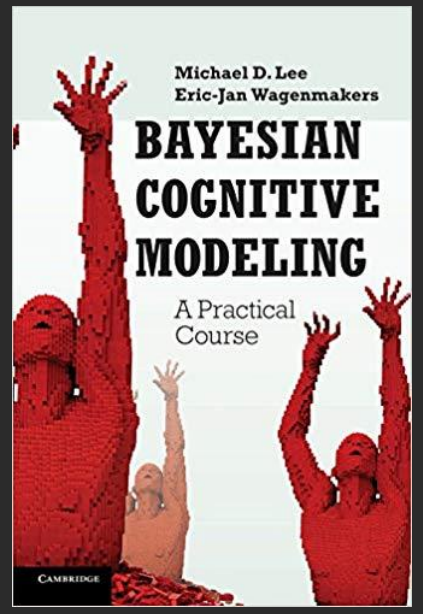
- Dan Foreman-Mackey
https://dfm.io

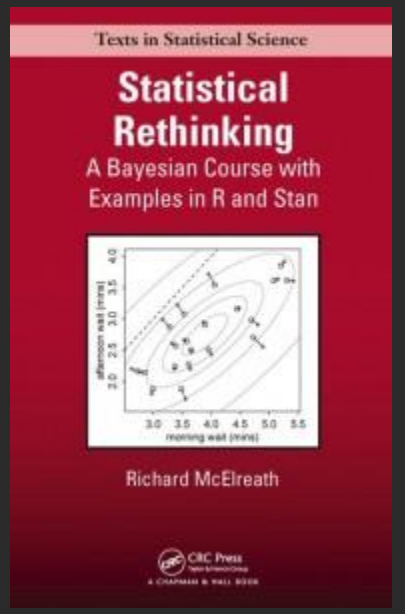
- Thomas Wiecki Blog
https://twiecki.io/

For those interested in Bayesian statistics:









https://github.com/pymc-devs/resources

Physical model

- Peimbert et al: Nebular Spectroscopy: A Guide on H ii Regions and Planetary Nebulae https://doi.org/10.1088/1538-3873/aa72c3
- Pérez Montero: Ionized gaseous nebulae chemical abundance determination using the direct method https://doi.org/10.1088/1538-3873/aa5abb
- Fernández et al (2018): Primordial helium abundance determination using sulfur as metallicity tracer https://doi.org/10.1093/mnras/sty1206
- Fernández et al (2019): A Bayesian direct method implementation to fit emission line spectra: Application to the primordial He abundance determination https://doi.org/10.1093/mnras/stz1433

Thank you very much!

Go NUTs !!!


1.00
0.75
0.50
0.25
0.00
y^2+
0
1000
2000
3000
4000
5000
Steps
1.0
0.5
0.0
−0.5
y^2+
0
1000
2000
3000
4000
5000
Steps