

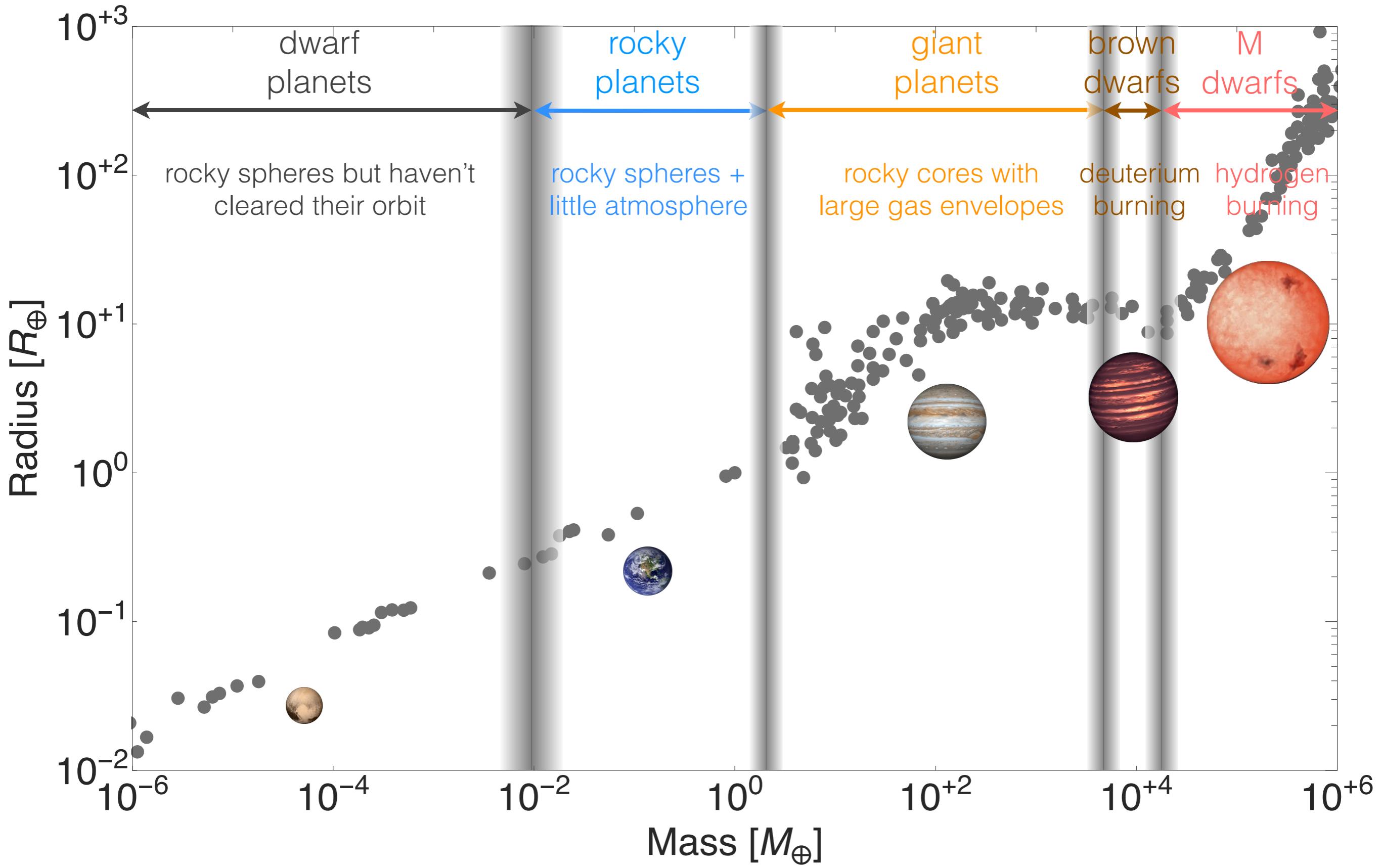


LAB 2

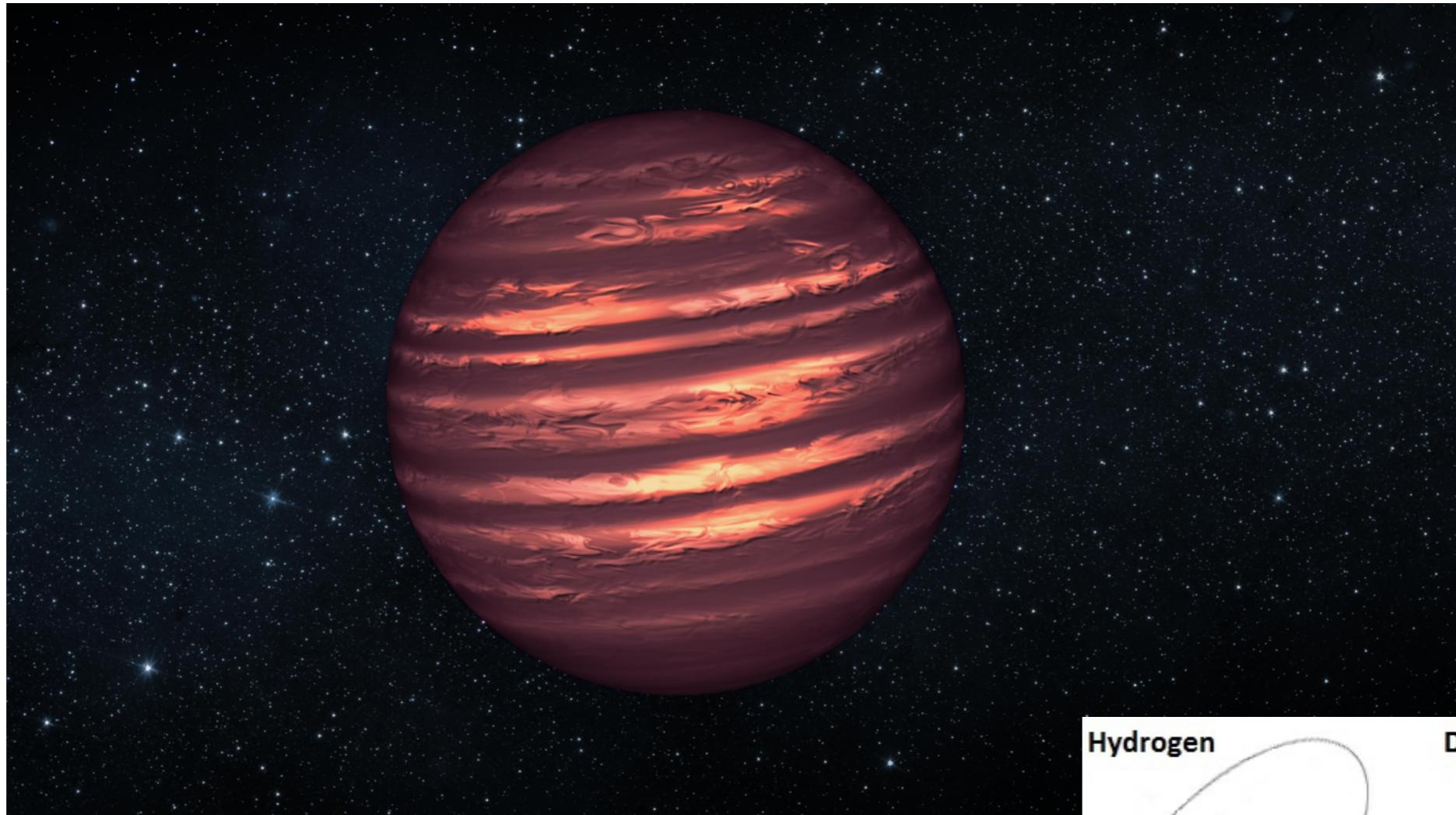
Taking the Temperature of a Cold Brown Dwarf

C4303: Astrostatistics

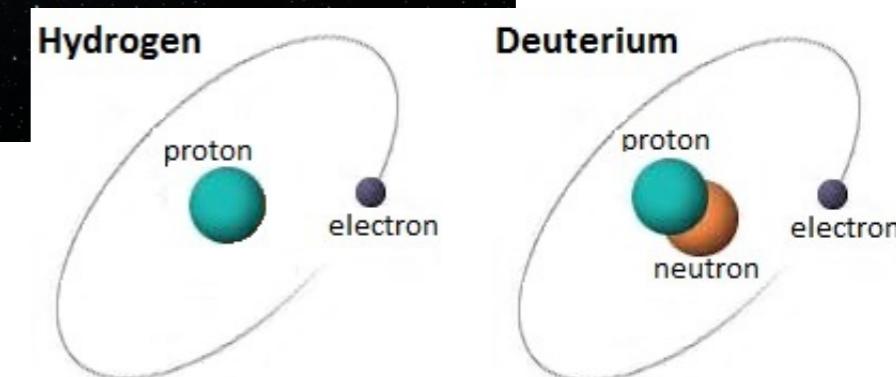
Quick background... Brown Dwarfs



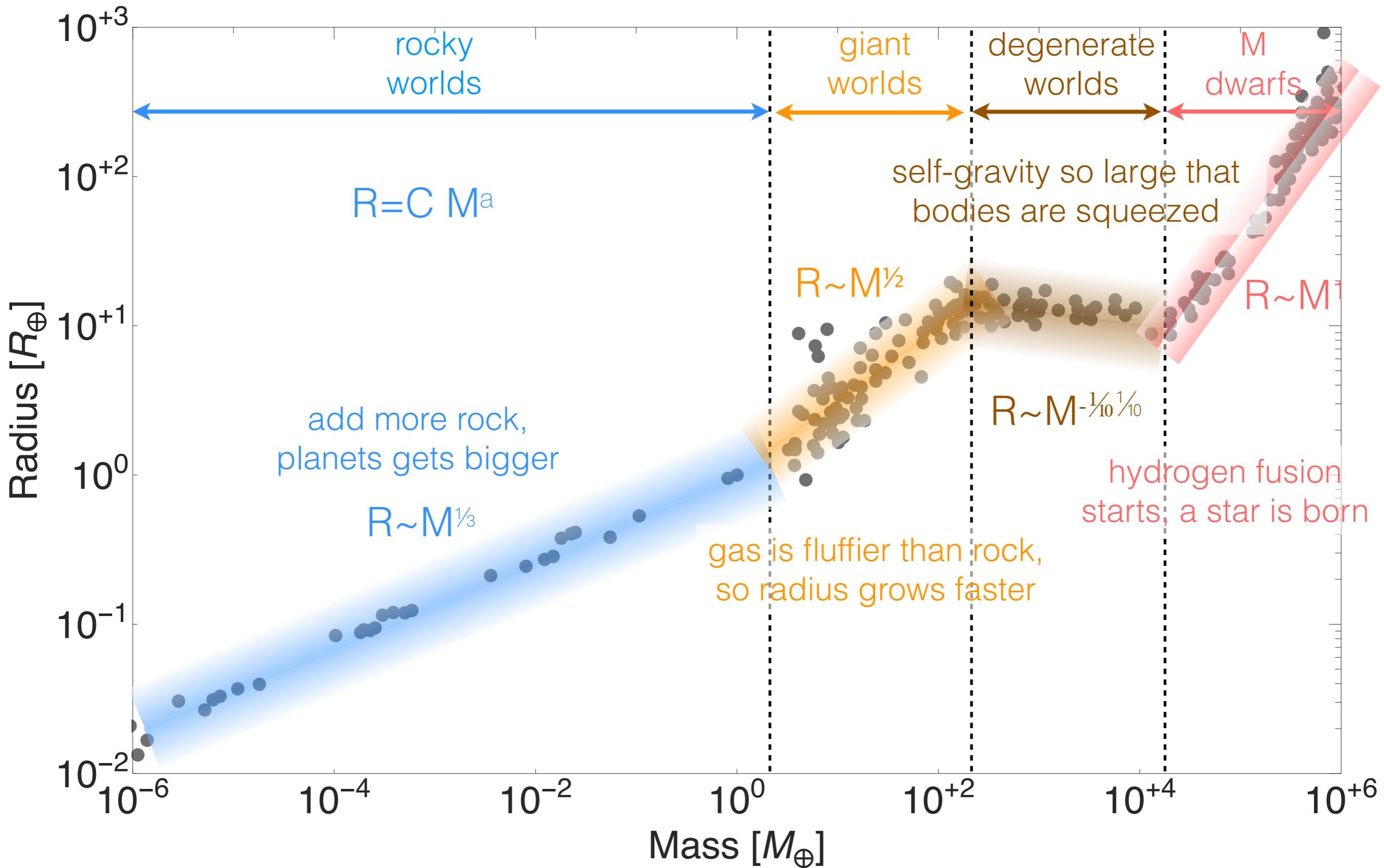
brown dwarfs



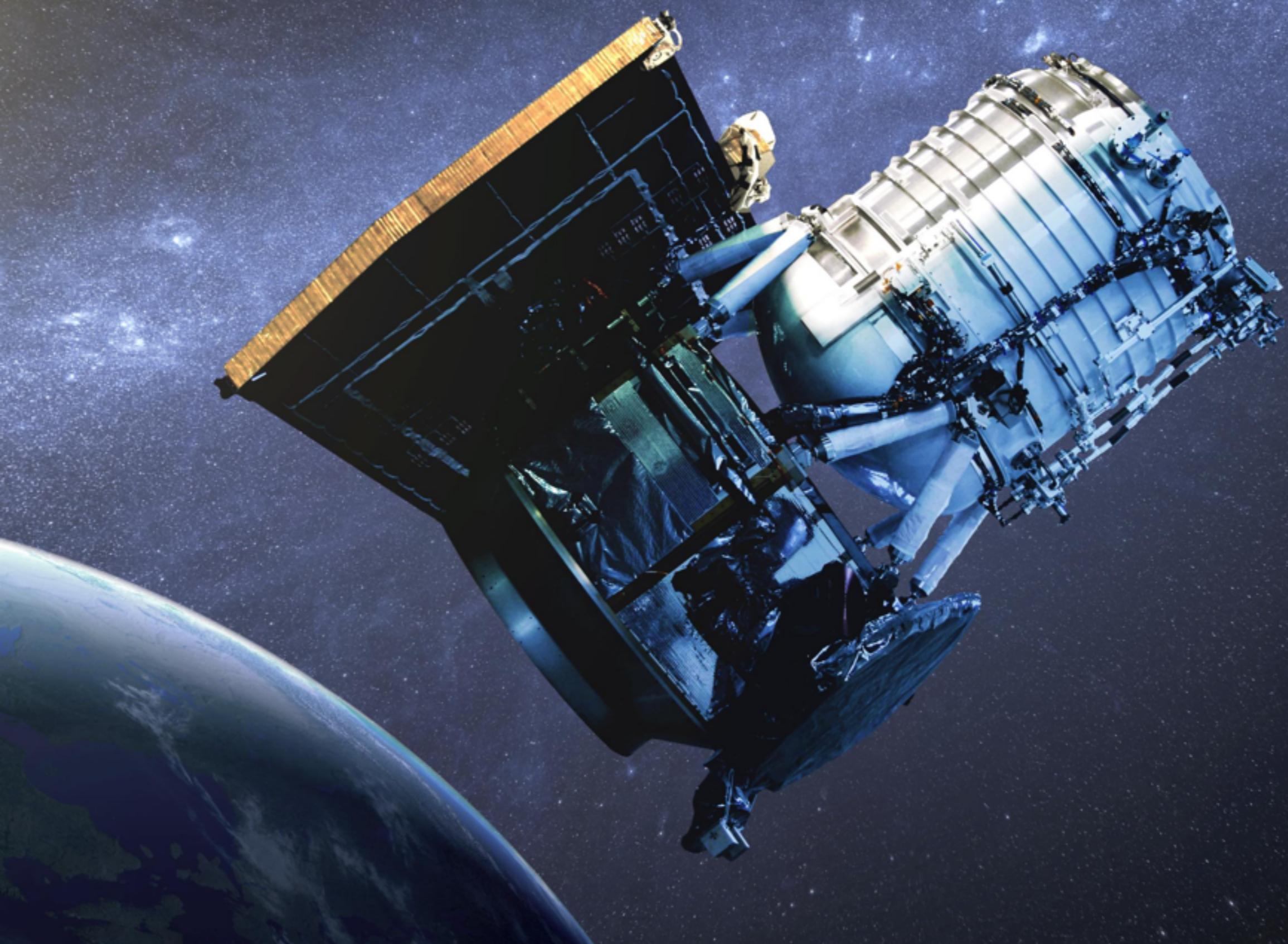
self-gravity so large that bodies are squeezed

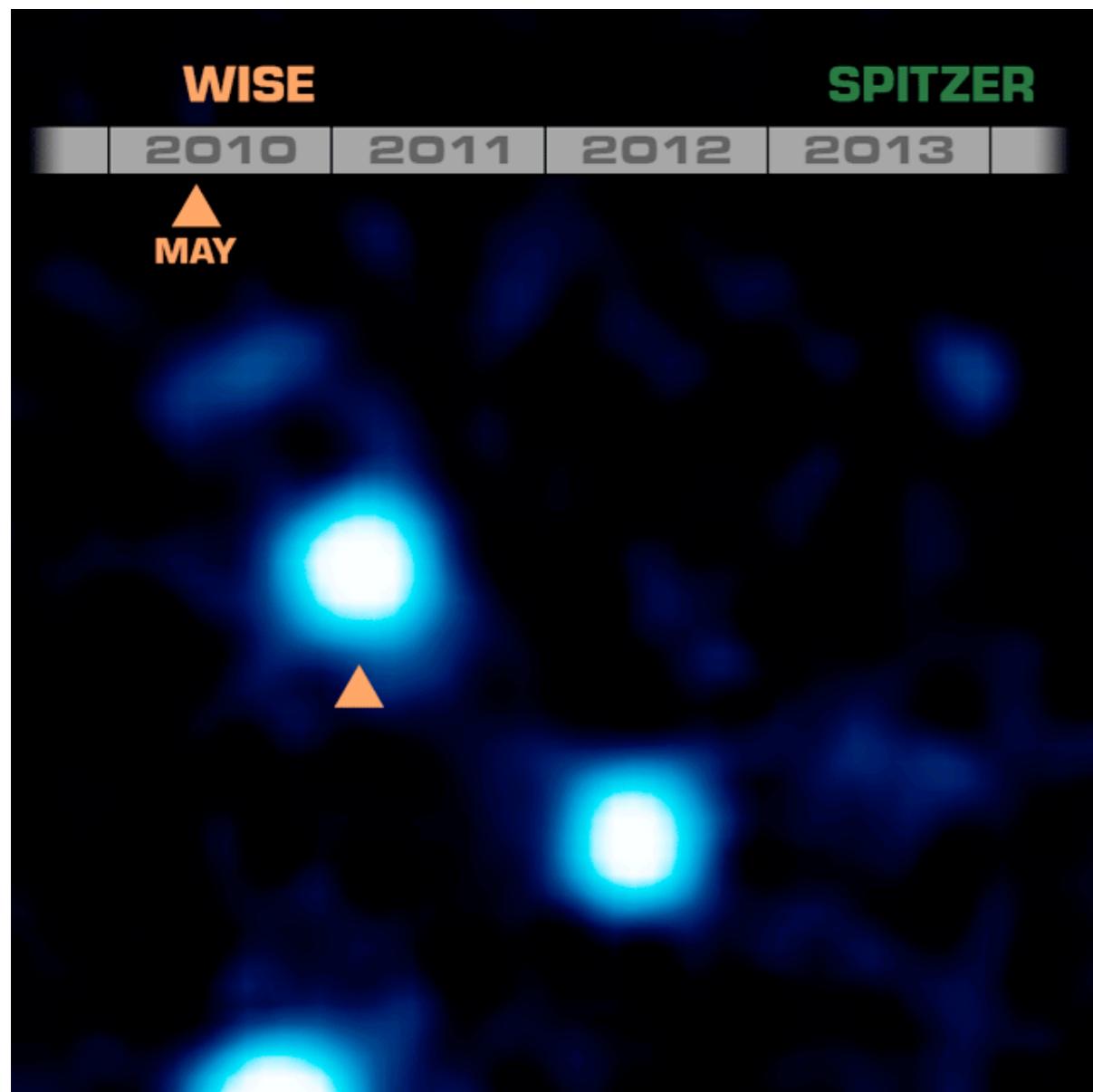


- ▶ deuterium (isotope of hydrogen) is the easiest thing to fuse
- ▶ once $M \geq 13M_{Jup}$, pressures high enough for deuterium fusion
- ▶ but deuterium is rare (10^{-5}) and conditions inefficient and so not body won't start shining until it starts fusing hydrogen-1



NASA Infrared 0.4m telescope, December 2009

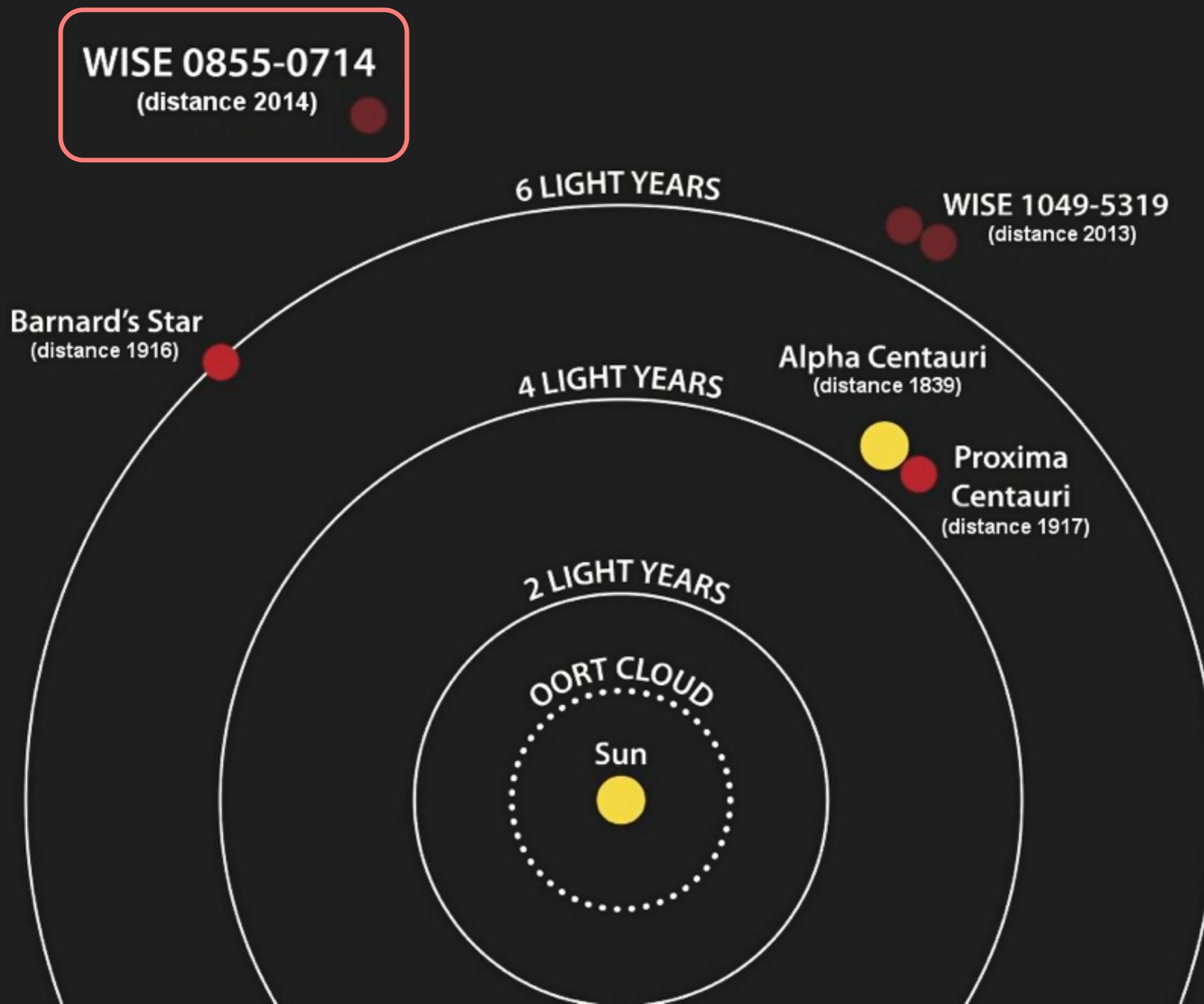


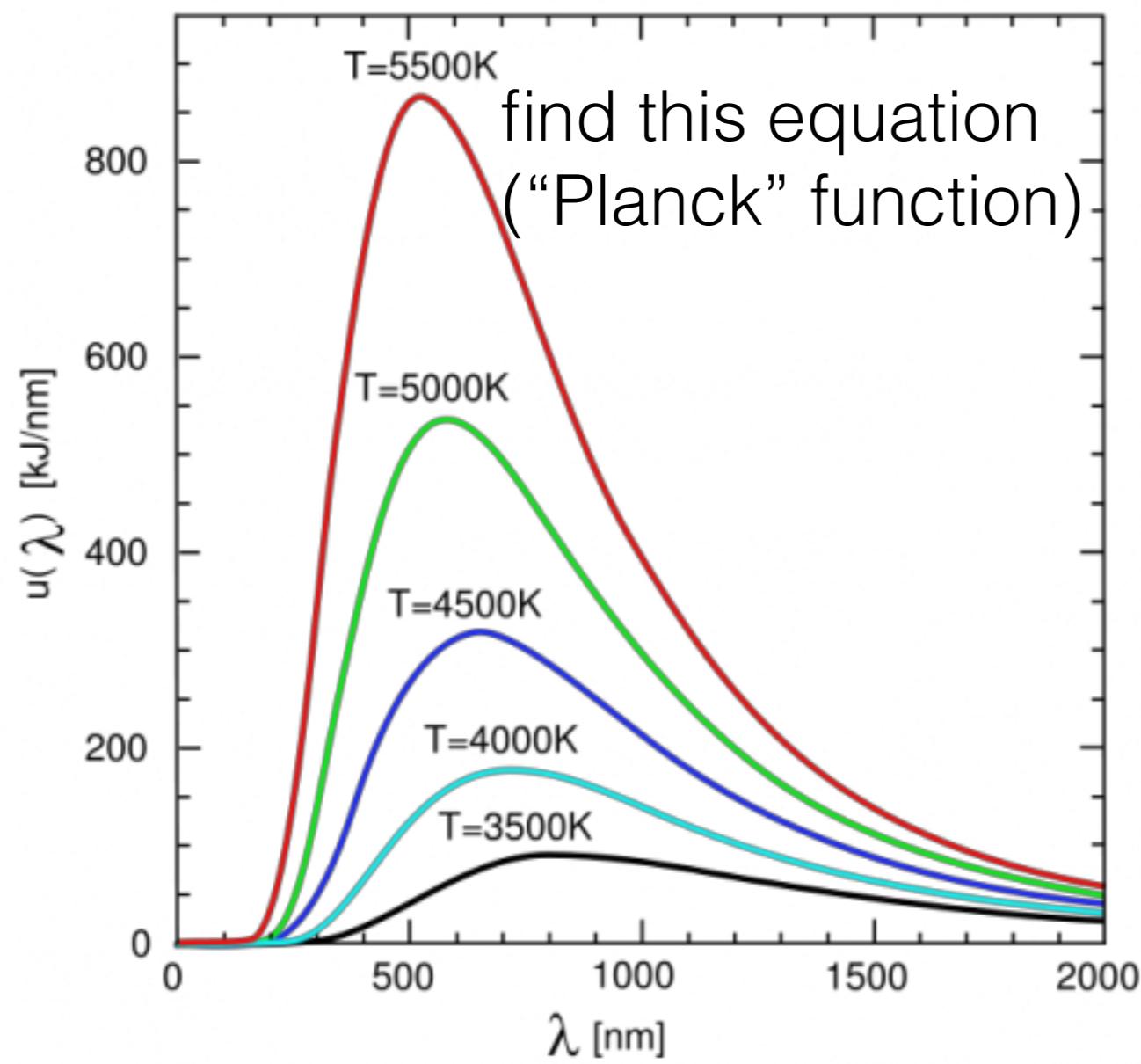


In April 2014, Kevin Luhman announces WISE 0855-0714

Temperature 225 - 260 K (below room temp!)

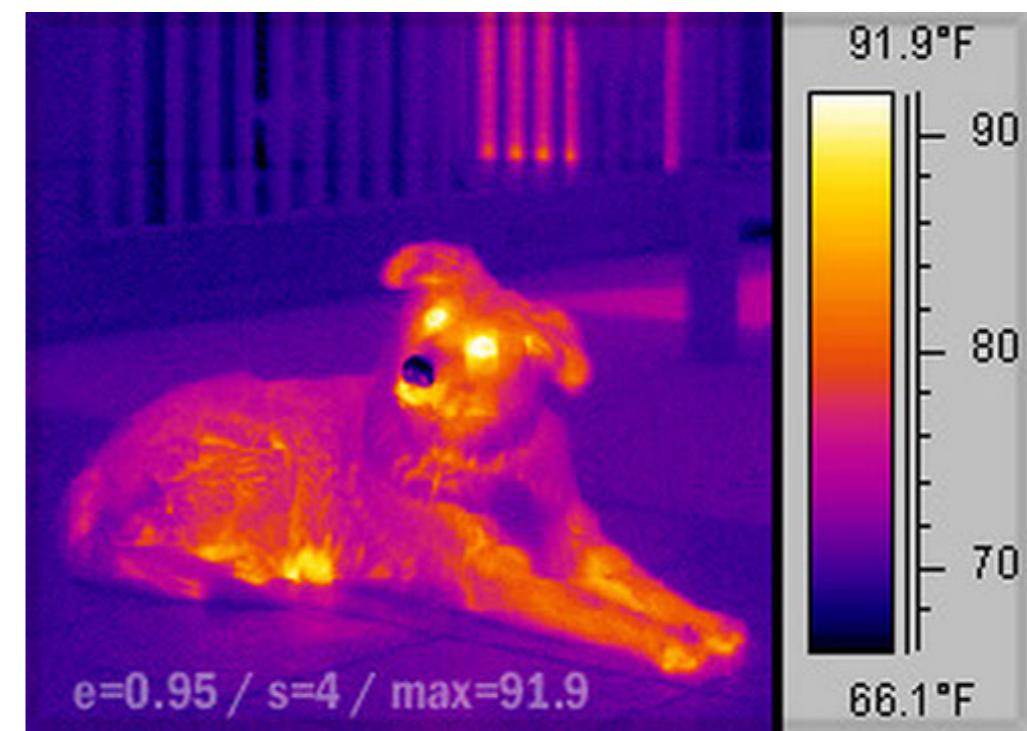
THE SUN'S CLOSEST NEIGHBORS





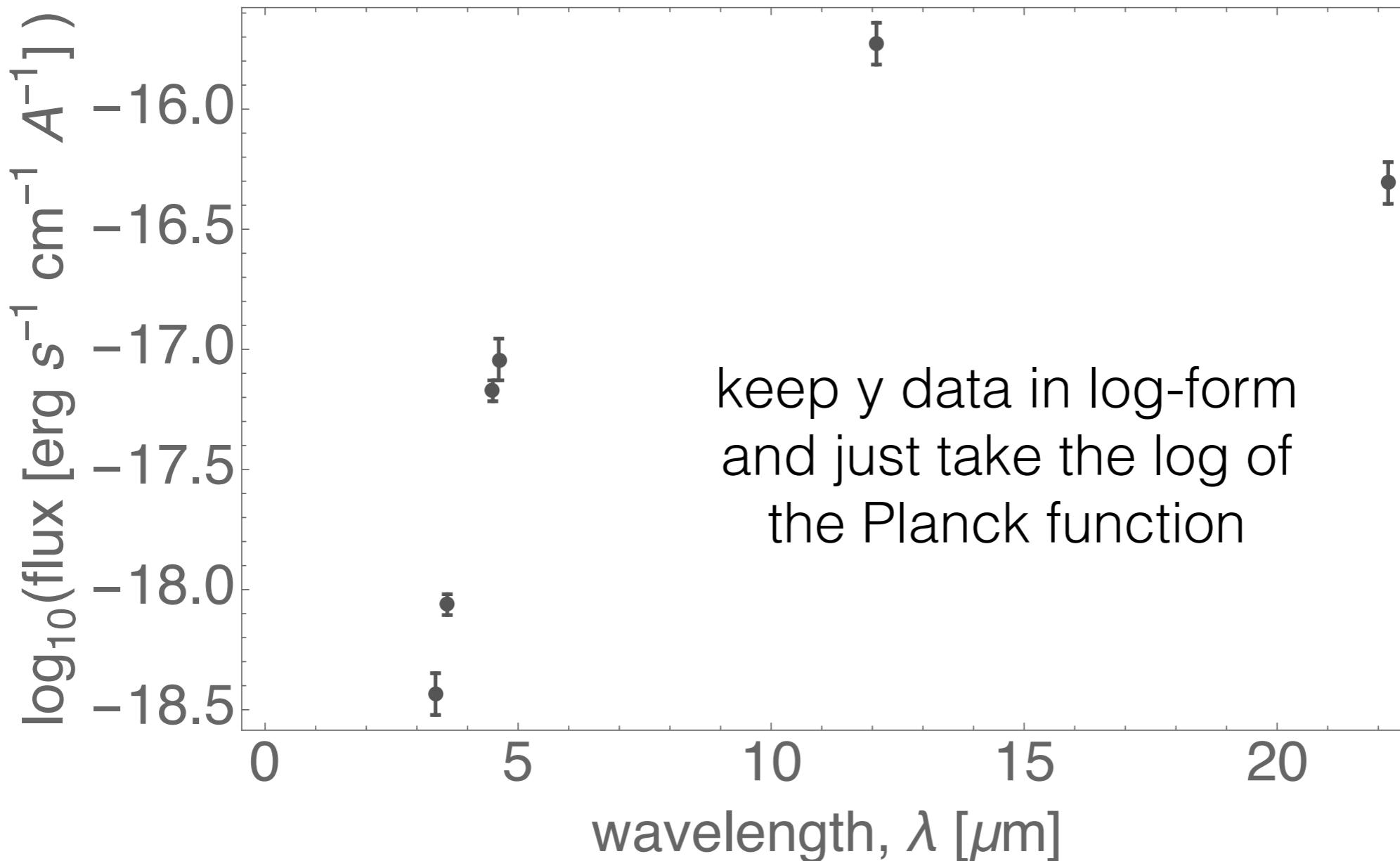
find this equation
("Planck" function)

all objects with $T > 0\text{K}$
emit radiation



ignoring clouds, molecular absorption/emission features,
etc, expect blackbody radiation curve

here's your data of WISE 0855-0714



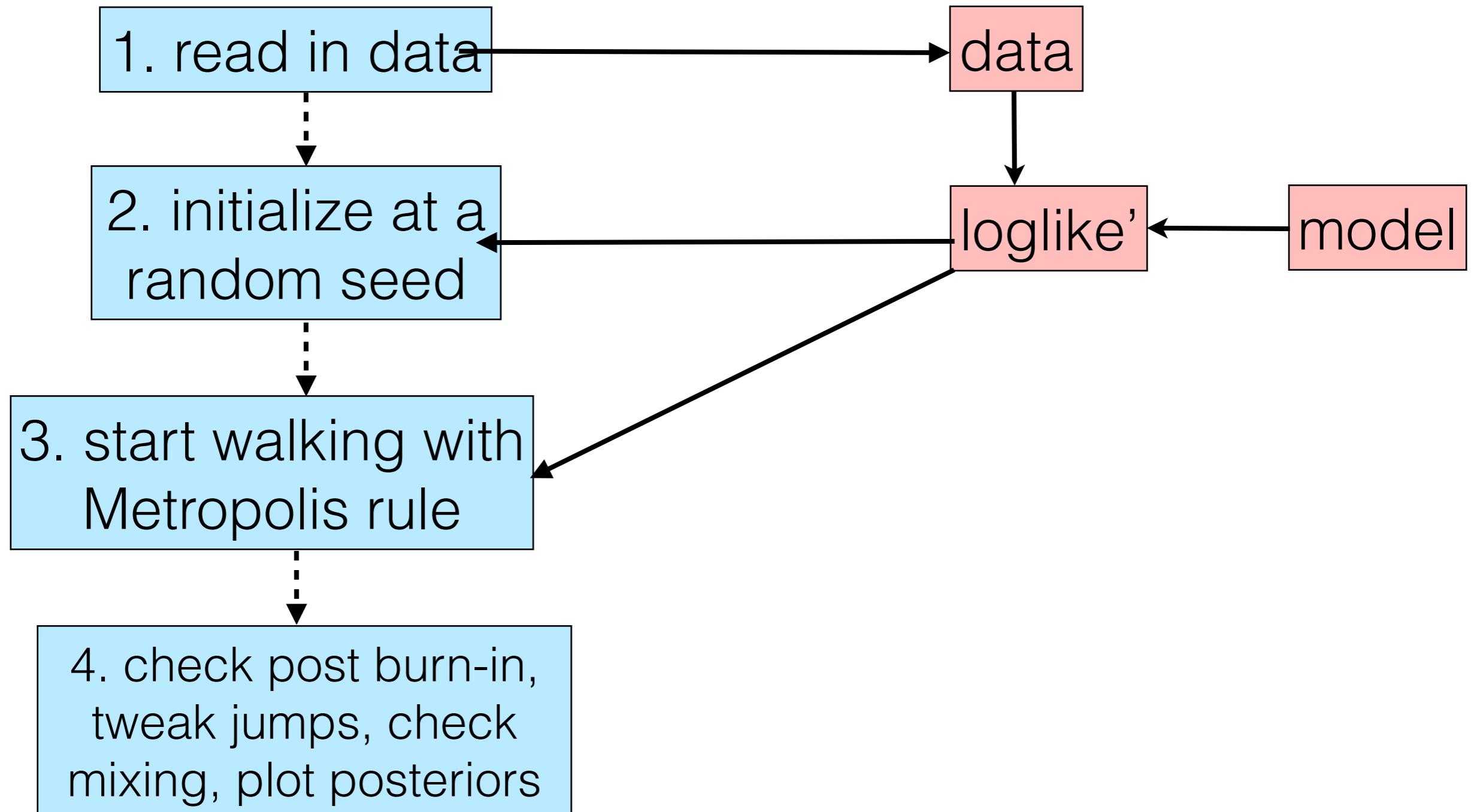
use an MCMC to get posteriors for Temp & multiplicative
factor in front of Planck function

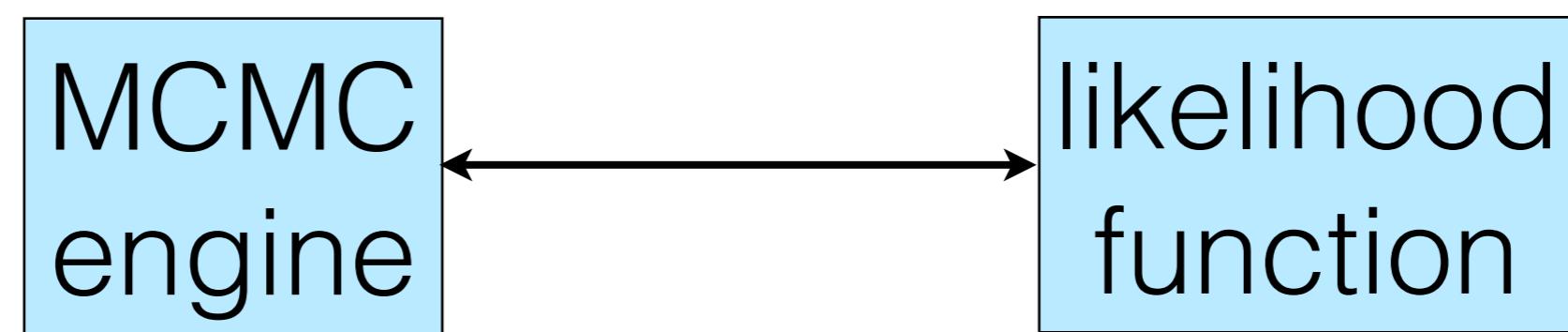
$$f(\lambda, T) = \alpha \frac{2hc^2}{\lambda^5} \frac{1}{\exp(\frac{hc}{\lambda k_B T}) - 1} \longrightarrow \log[f(\lambda, T)] = \log[2h\alpha c^2] + \log \left[\frac{1}{\lambda^5} \frac{1}{\exp(\frac{hc}{\lambda k_B T}) - 1} \right]$$

define likelihood

define priors

define model





```
In [2]: import numpy as np
import matplotlib.pyplot as plt
from astropy.io import ascii
import scipy.constants as con

# Get the data using the astropy ascii
data = ascii.read("../SED.dat", data_start=4)
lam=data[0][:] # Wavelength column
logf=data[1][:] # log10(flux)
errlogf=data[2][:] # Error on log10(flux)

# Shape parameters for priors
```

1. read in data

define priors

```
In [3]: # Set a definition for the model
def model(microns,Teff,logfactor):
    # Compute log flux density
    logflux = logf - 2.5 * logfactor * np.log10(microns / microns[0])
    logflux -= 2.5 * np.log10(1.0 - Teff / 5770)
    logflux -= 2.5 * np.log10((1.0 - Teff / 5770) / (1.0 - Teff))

    return logflux
```

model

```
In [4]: # Set a definition for the loglikelihood, assuming normally distributed data
def log_like(lam,logf,errlogf,theta):
    # Compute log likelihood
    loglike = -0.5 * np.sum((logf - model(lam, theta))**2 / errlogf**2)

    return loglike
```

loglike'

```
In [5]: # Set a definition for the logpriors  
def log_prior(theta,thetashape):
```

```
[REDACTED]
```

```
# Prior for theta[0]: Teff~logU[Teffmin,Teffmax]
```

```
[REDACTED]
```

```
#logprior = np.sum(logpriors)  
return np.sum(logpriors)
```

```
In [6]: # Initialize the MCMC from a random point drawn from the prior
```

```
[REDACTED]
```

```
# Calculate the associated modified loglike
```

2. initialize at a random seed

0])

```
In [7]: # Define the proposal jump size
```

```
# Starting walking
j=0
jmax=10000
while True:

    # Generate a proposal (or jump)
```

```
# Compute Metropolis Rule
```

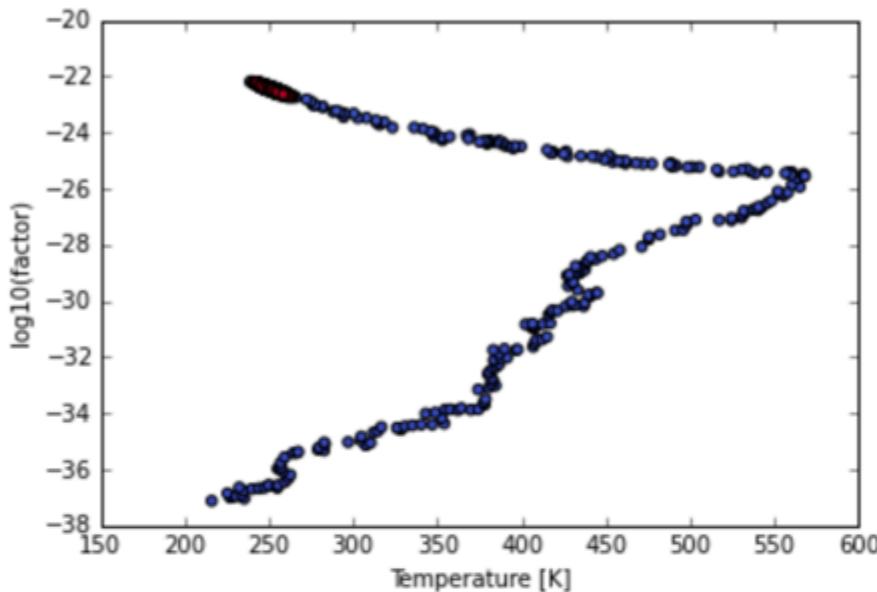
```
if probjump>np.random.uniform(0,1):
    # Accept the jump
    j = j + 1
    loglikechain = np.append(loglikechain,logliketrial)
    thetachain = np.vstack((thetachain,thetatrial))

if j==jmax:
    break
```

3. start walking with
Metropolis rule

```
In [52]: %matplotlib inline
```

```
jlist=np.arange(len(thetachain))
plt.scatter(thetachain[:,0], thetachain[:,1], c=jlist, cmap='coolwarm')
plt.xlabel('Temperature [K]')
plt.ylabel('log10(factor)')
plt.show()
```



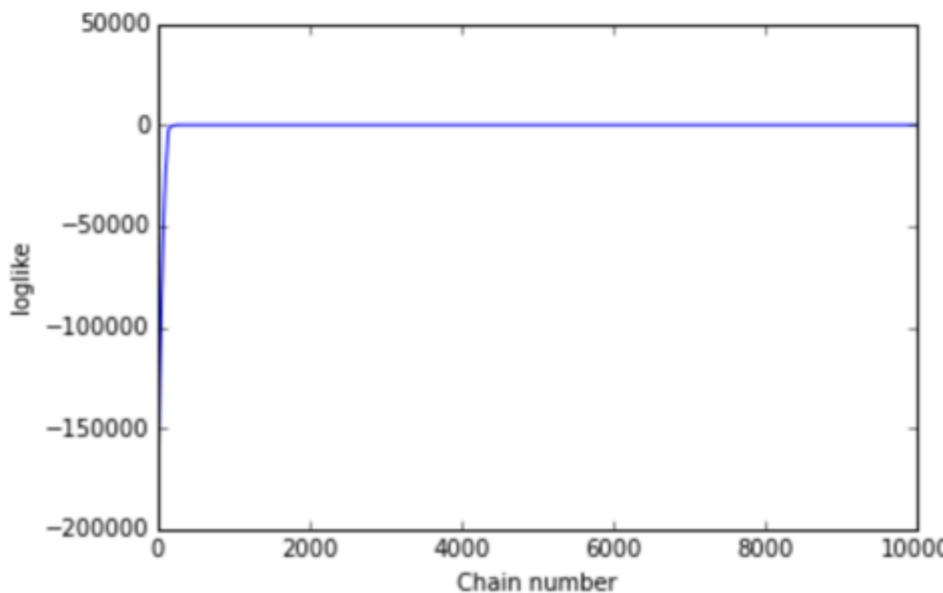
4. check post burn-in,
tweak jumps, check
mixing, plot posteriors

```
In [53]: np.max(loglikechain)
```

```
Out[53]: 8.6442438885663009
```

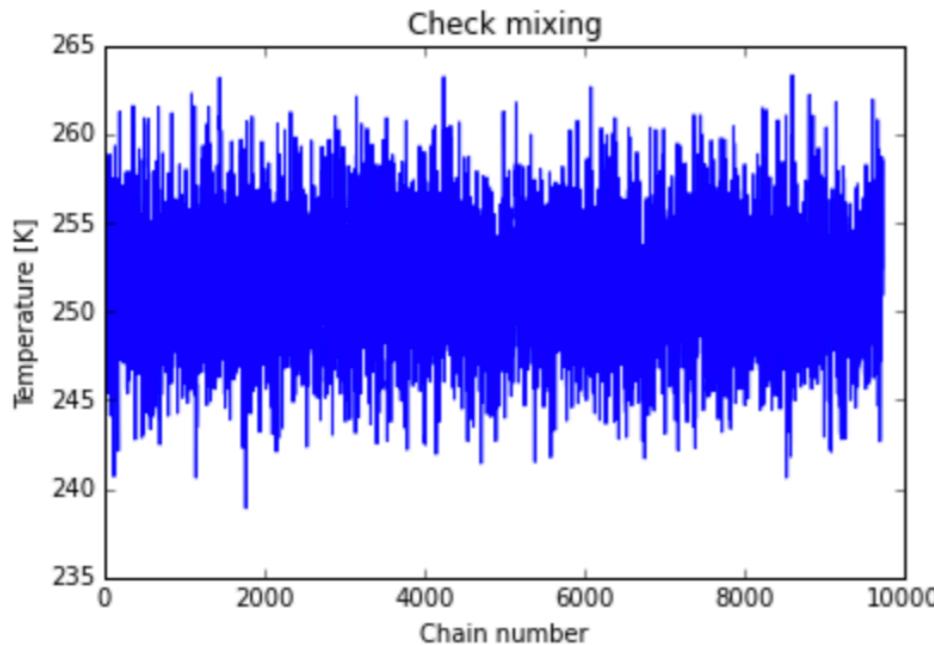
```
In [54]: %matplotlib inline
```

```
plt.plot(loglikechain)
plt.xlabel('Chain number')
plt.ylabel('loglike')
plt.show()
```



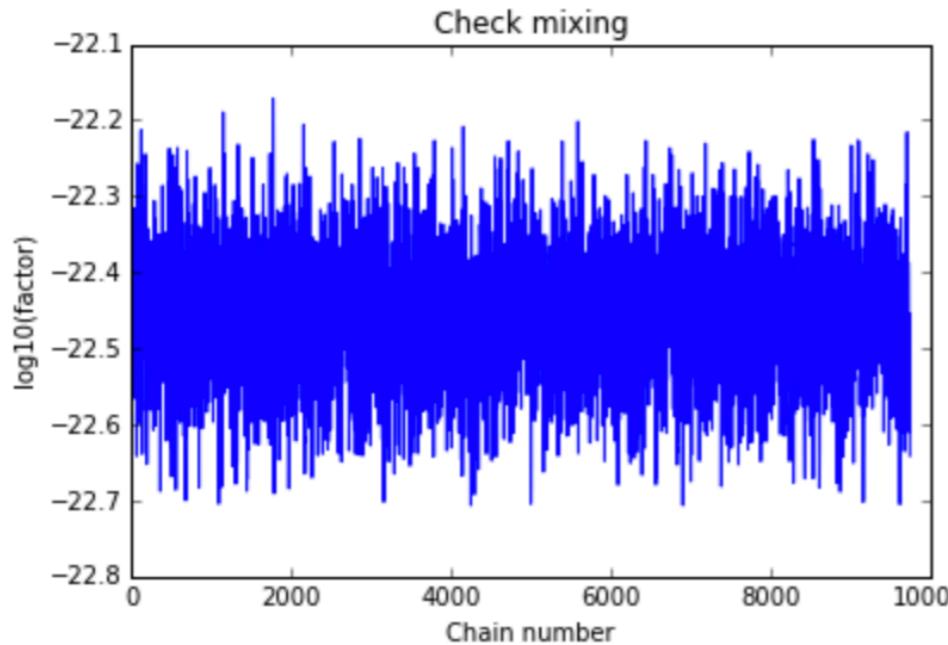
```
In [16]: %matplotlib inline
```

```
plt.plot(thetachain[burnj:,0])
plt.title('Check mixing')
plt.xlabel('Chain number')
plt.ylabel('Temperature [K]')
plt.show()
```



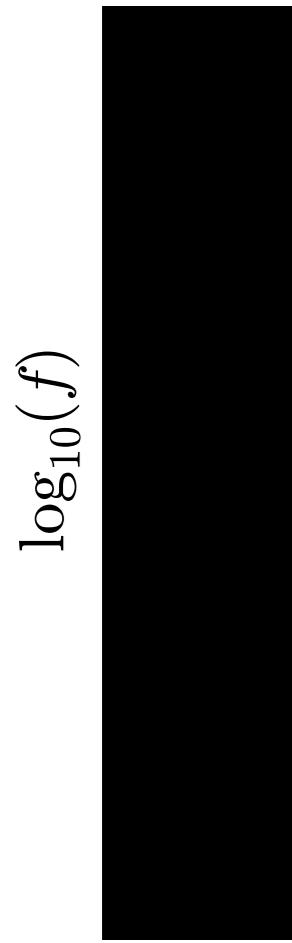
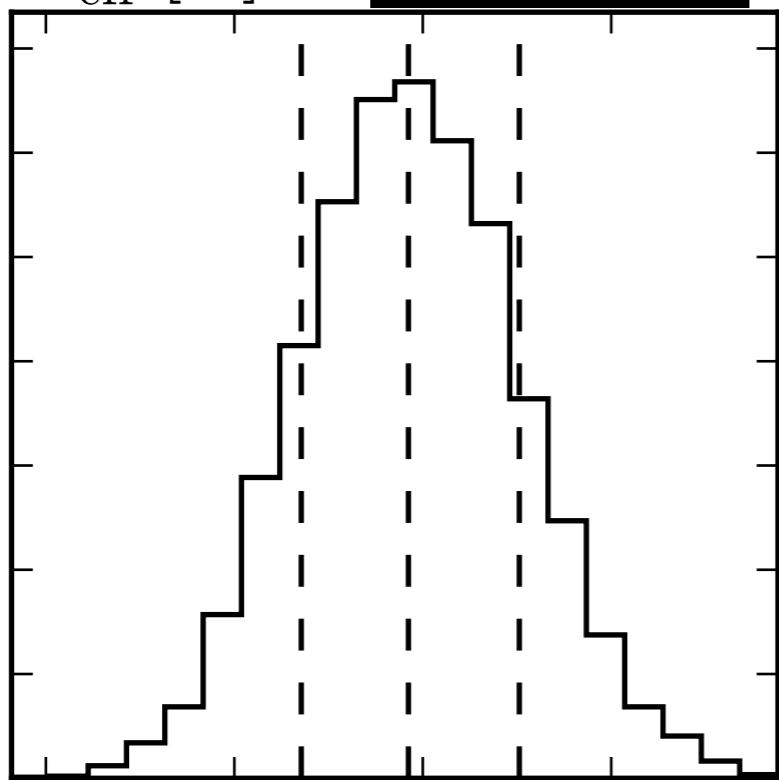
```
In [17]: %matplotlib inline
```

```
plt.plot(thetachain[burnj:,1])
plt.title('Check mixing')
plt.xlabel('Chain number')
plt.ylabel('log10(factor)')
plt.show()
```

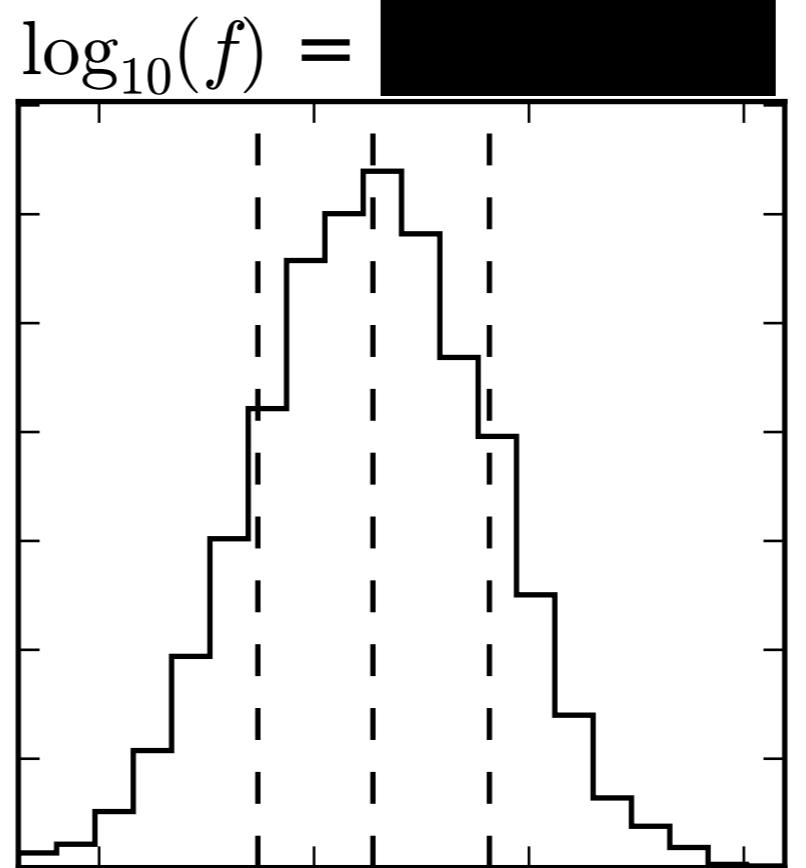


4. check post burn-in,
tweak jumps, check
mixing, plot posteriors

$T_{\text{eff}} \text{ [K]} =$ 

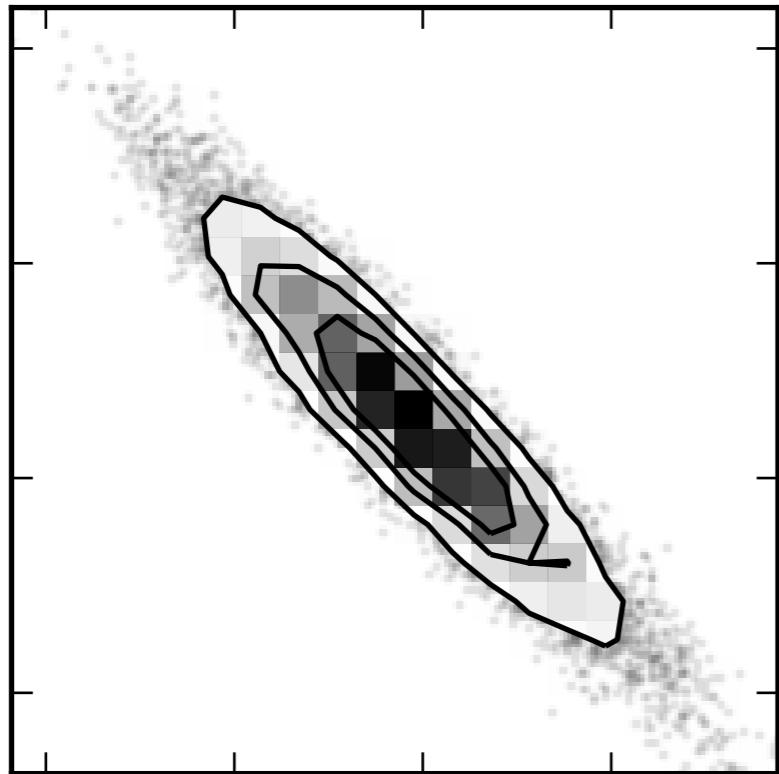


$\log_{10}(f)$



4. check post burn-in,
tweak jumps, check
mixing, plot posteriors

corner.py



$\log_{10}(f)$

YOUR TASK: Use this data set to calculate the temperature of WISE 0855

- 1] Write down your equations for likelihood, (priors) & model
- 2] Import data
- 3] Initialize from a random point and compute loglike'
- 4] Walk from there, using Metropolis rule and Gaussian proposals
- 5] Try just a few steps, tweak jump sizes till things run smoothly, may have to play around with priors if too wide
- 6] Once a good MCMC has run, filter out the burn-in points
- 7] Make some nice plots proving i) burnt-in ii) good mixing and then your posteriors as a histogram
- 8*] Combine multiple chains from independent seeds
- 9^G] Change your T prior (e.g. log-uniform vs uniform vs uniform in T^4)
- 10^G] Add a quadrature extra noise term, with a log-uniform prior
- 11] Write up your slides! Include median +/- 1 sigma quantile final result