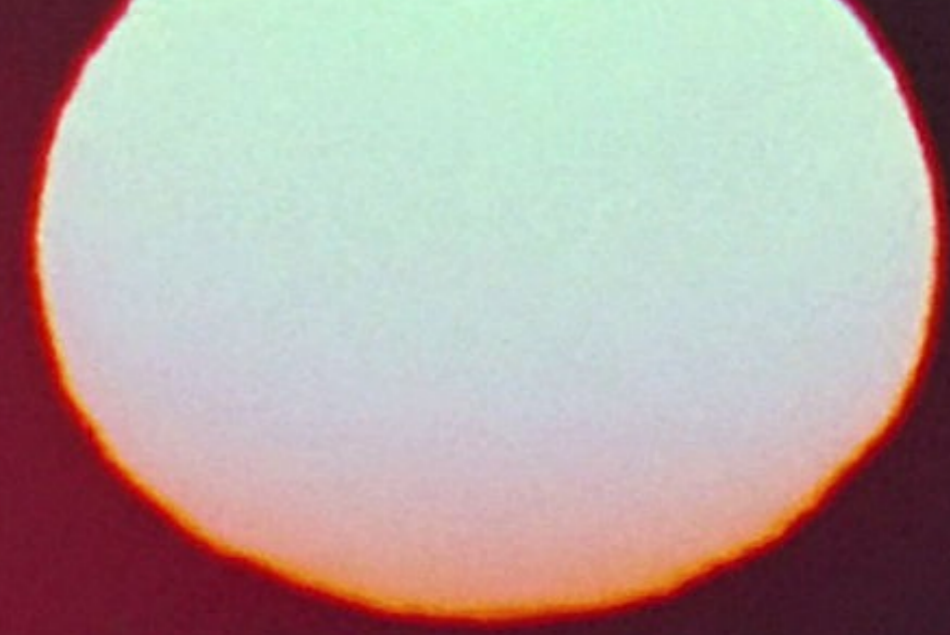


# Binary Populations



**Robert Izzard**  
**University of Surrey**  
(Just south of LHR, west of LGW)



Science & Technology  
Facilities Council



UNIVERSITY OF  
SURREY

# Binaries and Populations II



# Previously

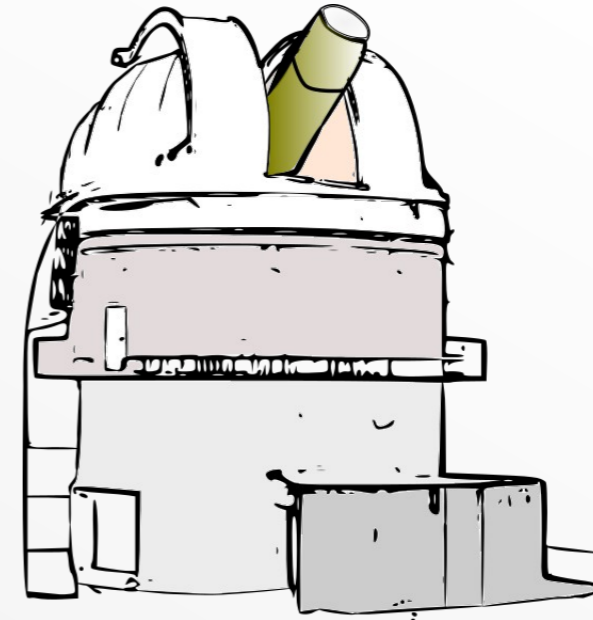
- What is stellar population synthesis?
- Why is it useful?
- The big binary parameter space
- Sampling the parameter space
- Initial distributions in single and binary stars
- Fast synthetic stellar evolution models

# What's coming next

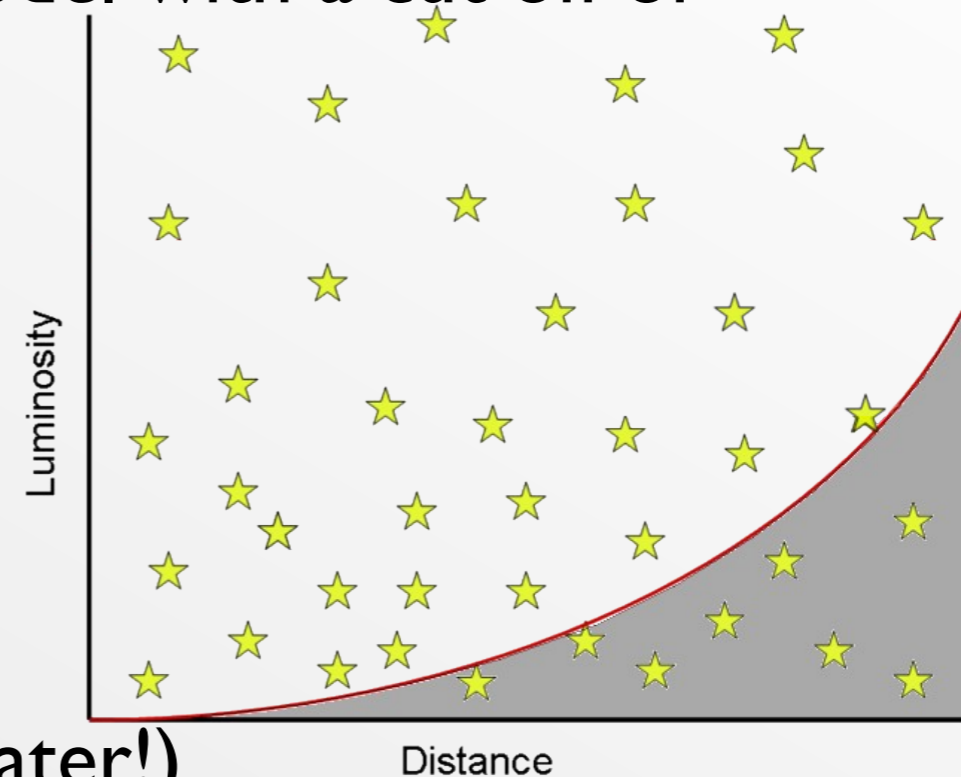
- Observations : the things to compare with
- Selection effects
- How to compare models to obs.
- The power of population synthesis: **binary stars**
- My *binary\_c* code → examples for you!
- Population synthesis case studies

# Observations

- However good our theories, astrophysics is still very much **observationally driven**
- These are usually either surveys e.g.
  - Hipparcos, Gaia
  - SDSS, APOGEE, Galah
  - COROT, Kepler
- Or “random” pointings toward objects of interest (often called “surveys”!)
- Always **SELECTION EFFECTS**



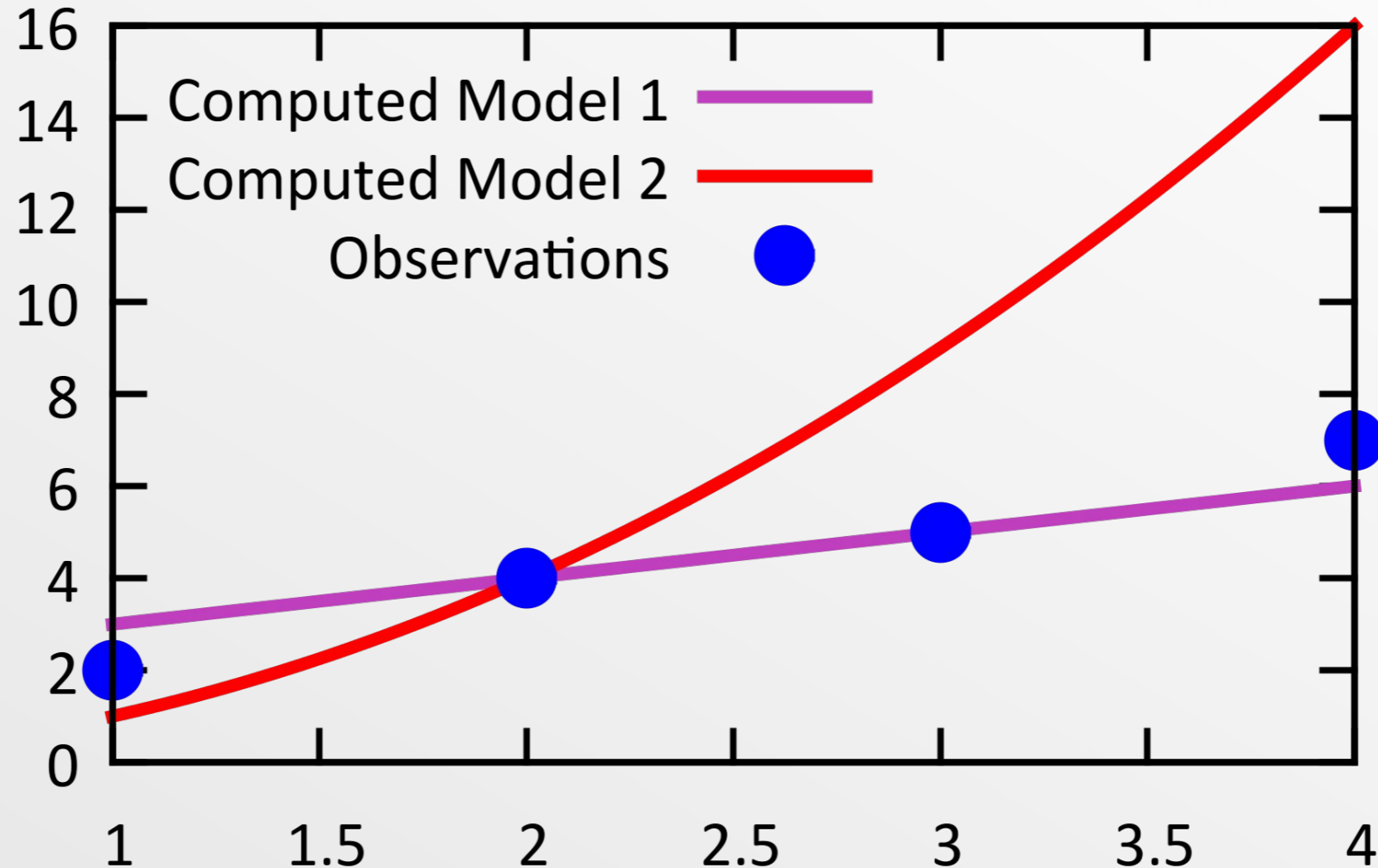
- **Volume** limit is simplest, you need do nothing if you think the sample has enough objects
- **Magnitude** limits you can model with a cut off or completeness function  
(cf. *Malmquist* bias)
- A **complete** survey is rare but possible, e.g. strong barium stars (more on these later!)
- Random choice may be ok if sufficient number of stars to justify the risk. **This may also be all you can get.**



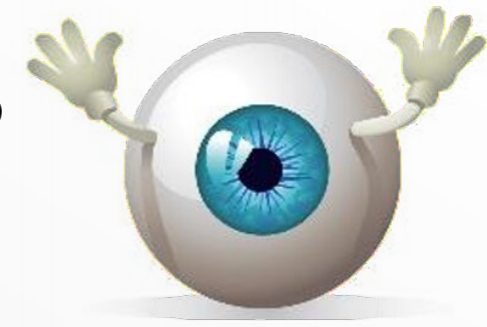
# Models vs observations



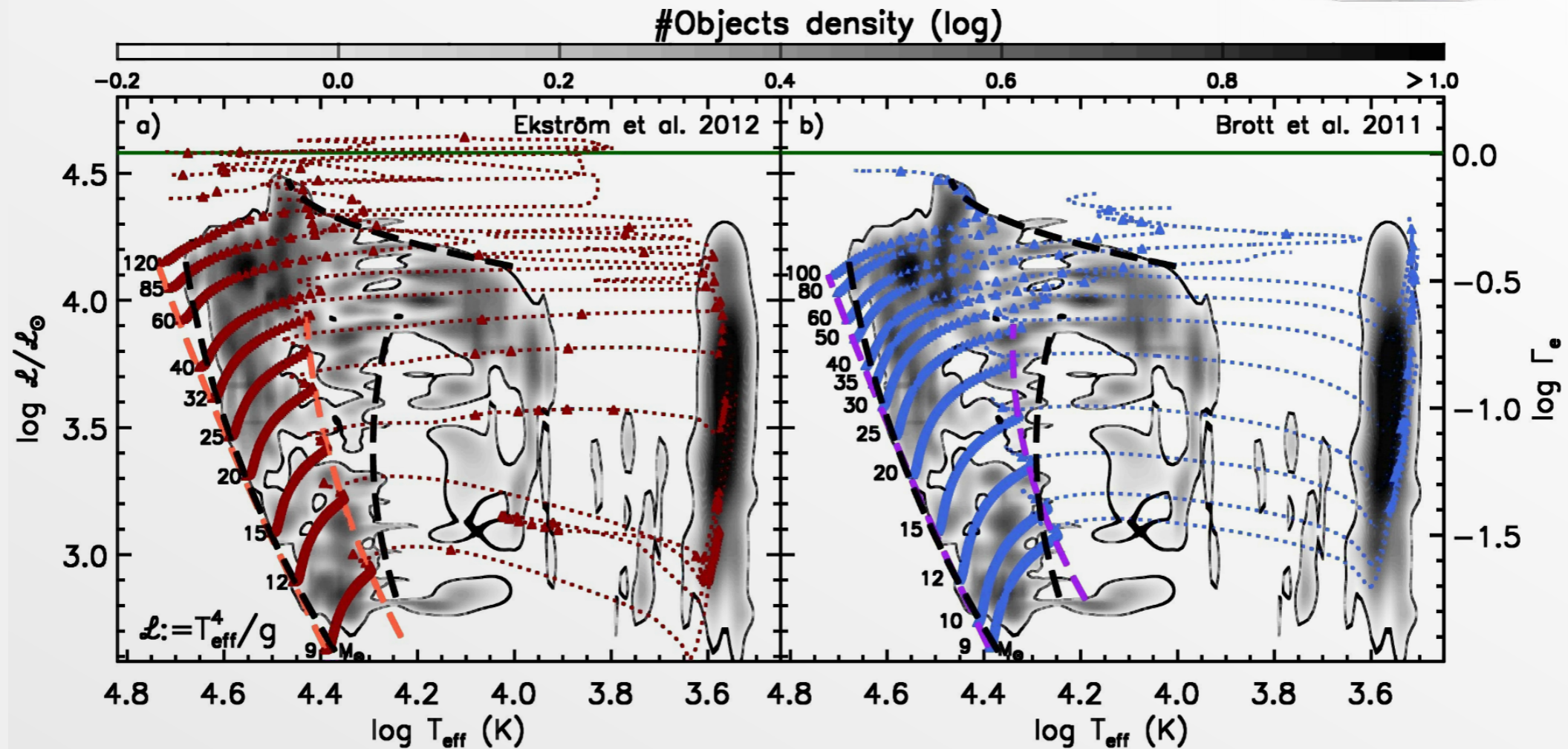
- By eye fitting: which is the best match?



# Models vs observations



- By eye fitting: which is the best match?

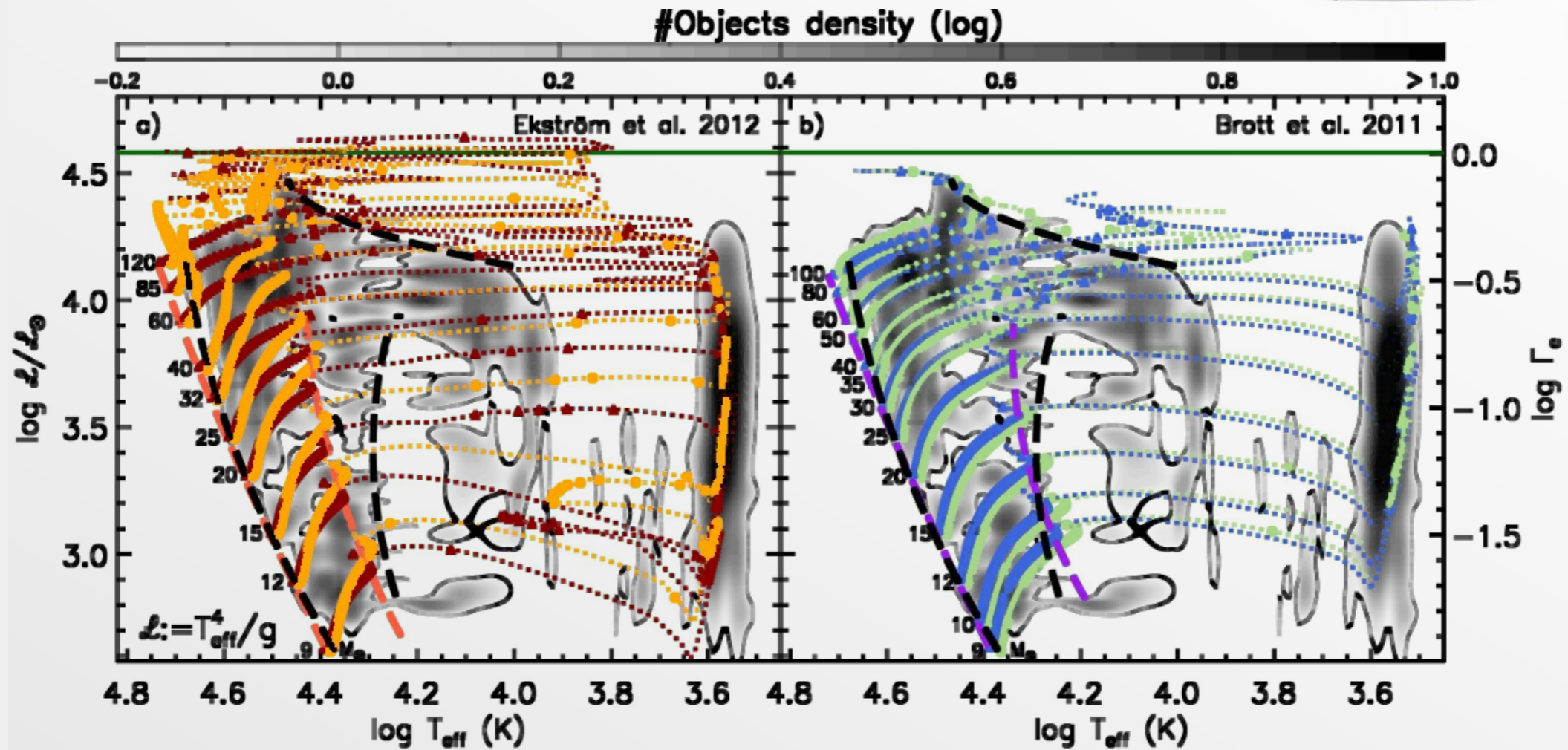




# Models vs observations



- By eye fitting: Rotating models are better?



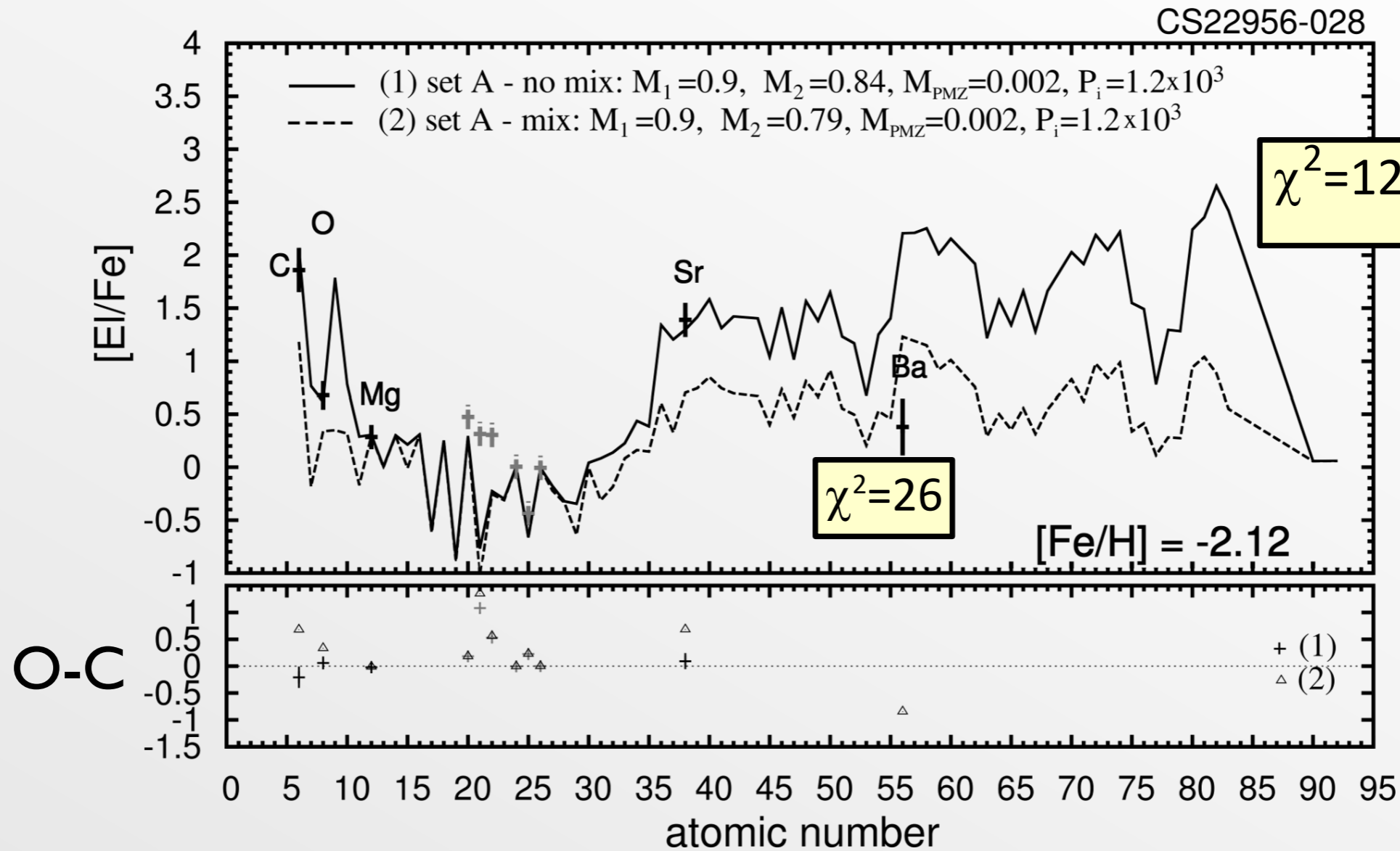
# Models vs observations: $\chi^2$ tests

$$\chi^2_\nu = \frac{1}{\nu} \sum \frac{(O - C)^2}{\sigma^2}$$

Degrees of freedom

$$\nu = N_{\text{obs}} - n_{\text{parameters}} - 1$$

Bad fits have  $\chi^2_\nu \gg 1$

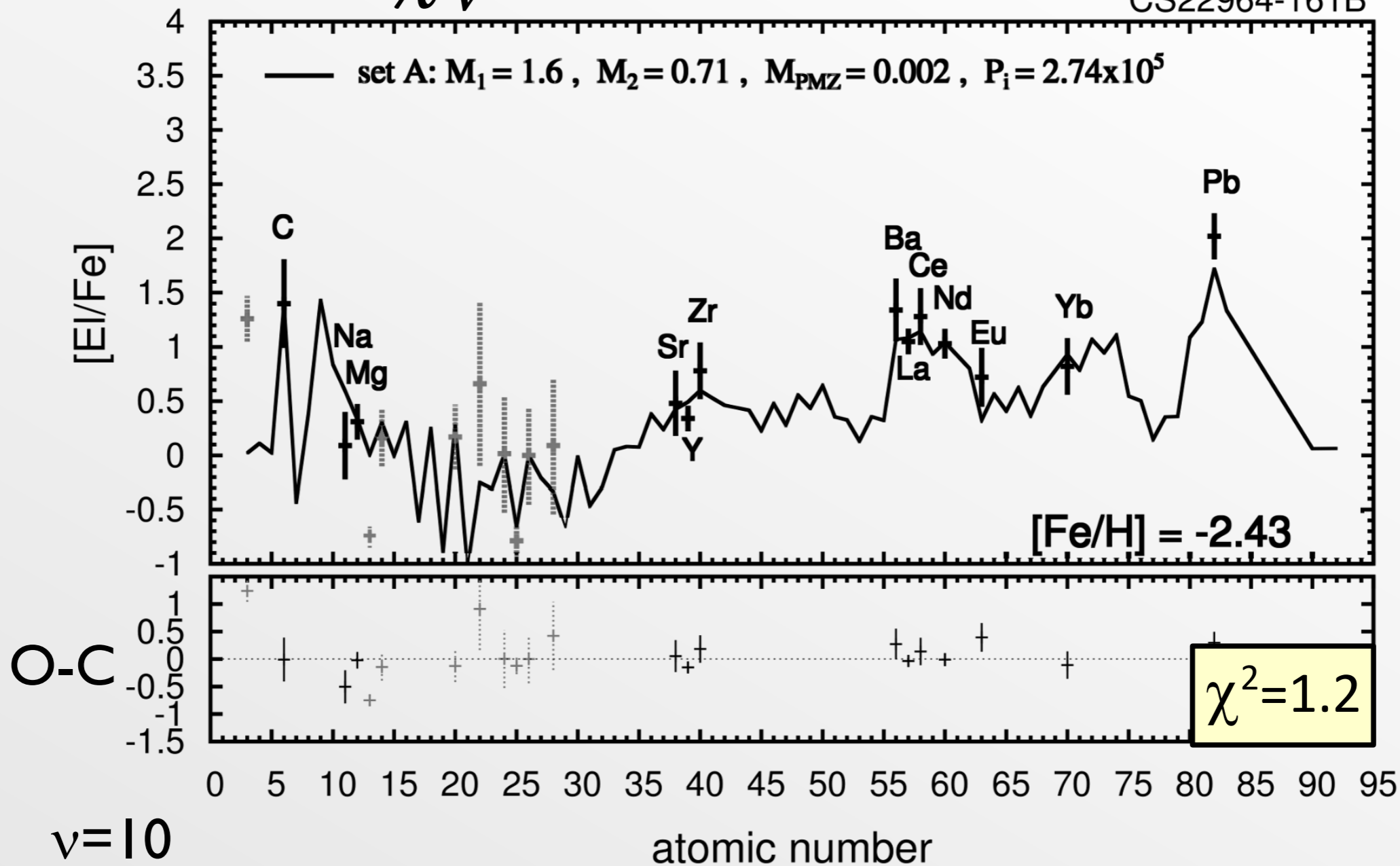


$\nu=2$

**Abate et al. (2014)**

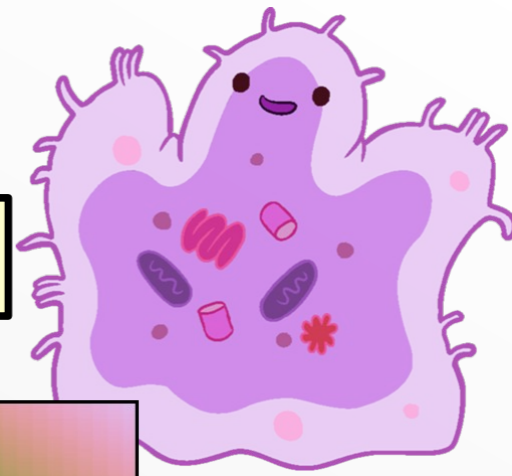
Good fits have  $\chi^2_\nu \sim 1$

CS22964-161B



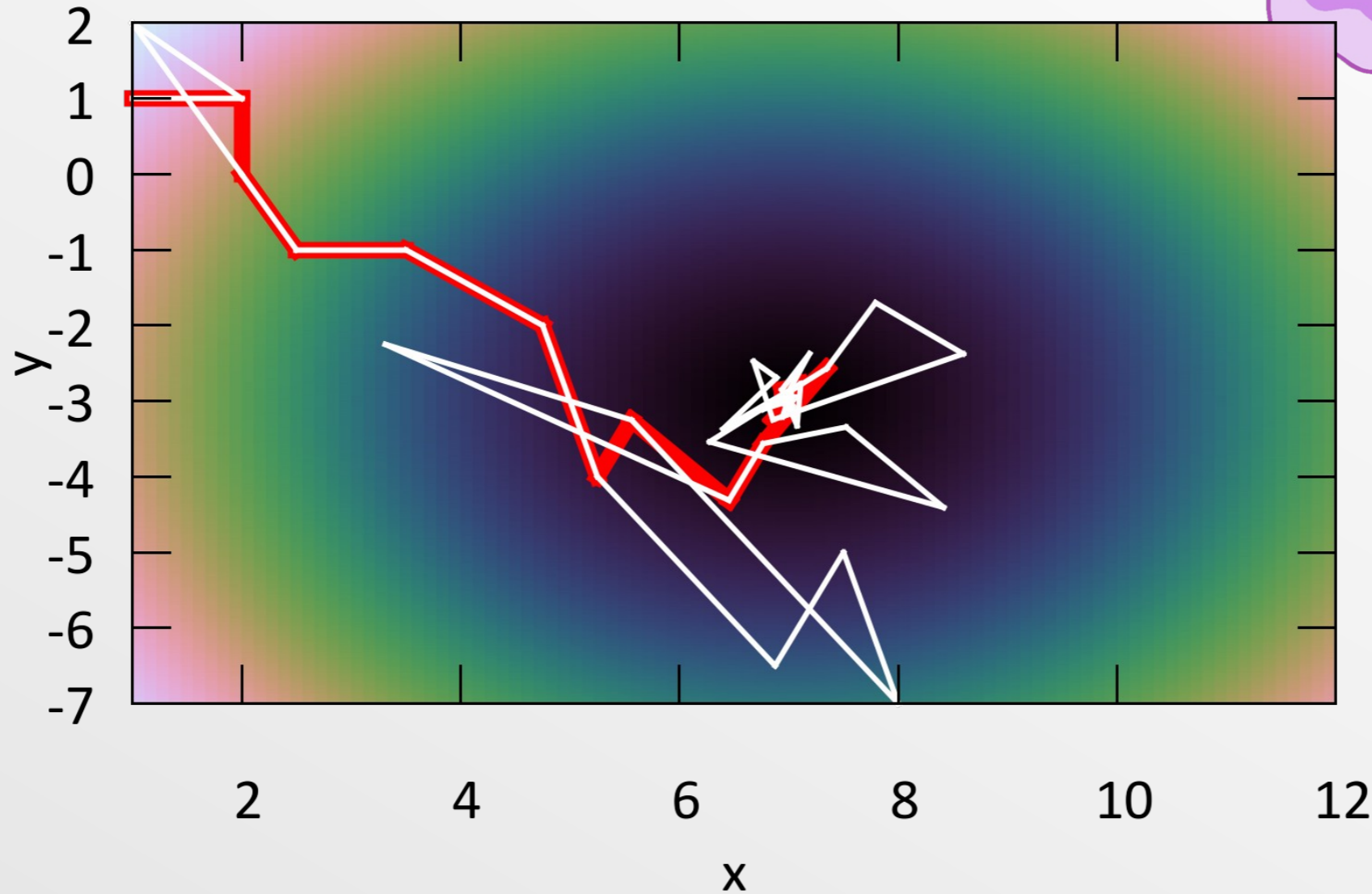
Abate et al. (2014)

# Amoeba/Downhill Simplex



$$(x-7)^2+(y+3)^2$$

**Minimise function O-C**



# Other matching algorithms

- **Genetic**
  - Minimize fitness functions
- **Markov-Chain Monte Carlo**
  - used often e.g. in cosmology
- Etc. → whole branch of **numerical analysis**
- Latest applications to (many) stars:  
**Bayesian methods**



# Bayesian Methods

- Is O-C good enough? **Bayes** says no.

Likelihood of B  
given A

Prior  
Knowledge

$$P(A|B) = \frac{P(B|A) P(A)}{P(B)}$$

Can be used to compare O and C  
and provide a ***distribution*** of the  
(many) best fitting parameters

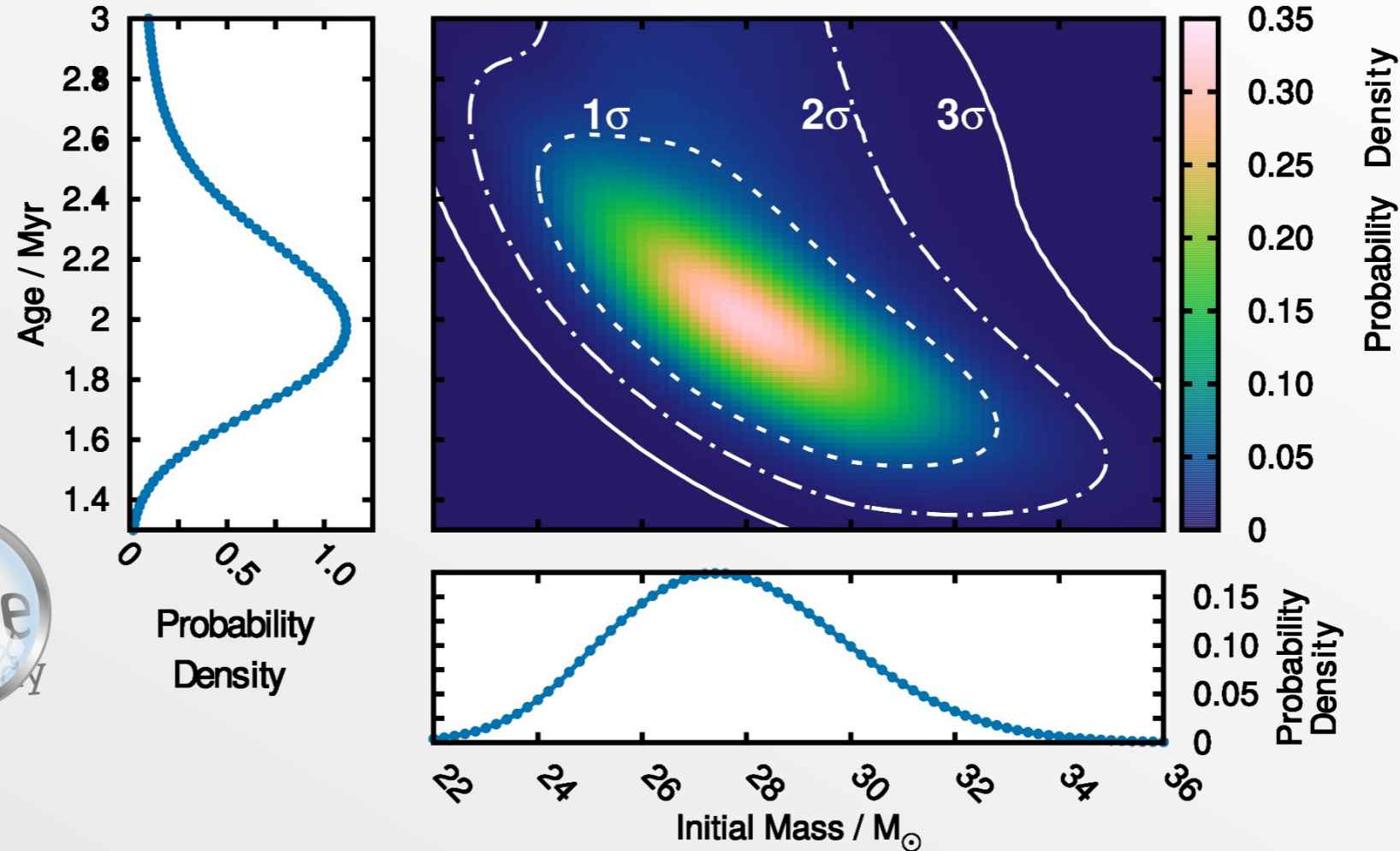


# Case study:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Given  $\log L$ ,  $T_{\text{eff}}$  and  $\log g$ , of star V3903 Sgr A what is  $M$ ?

Torres et al. (2010):  $M = 27.27 \pm 0.55 M_{\odot}$

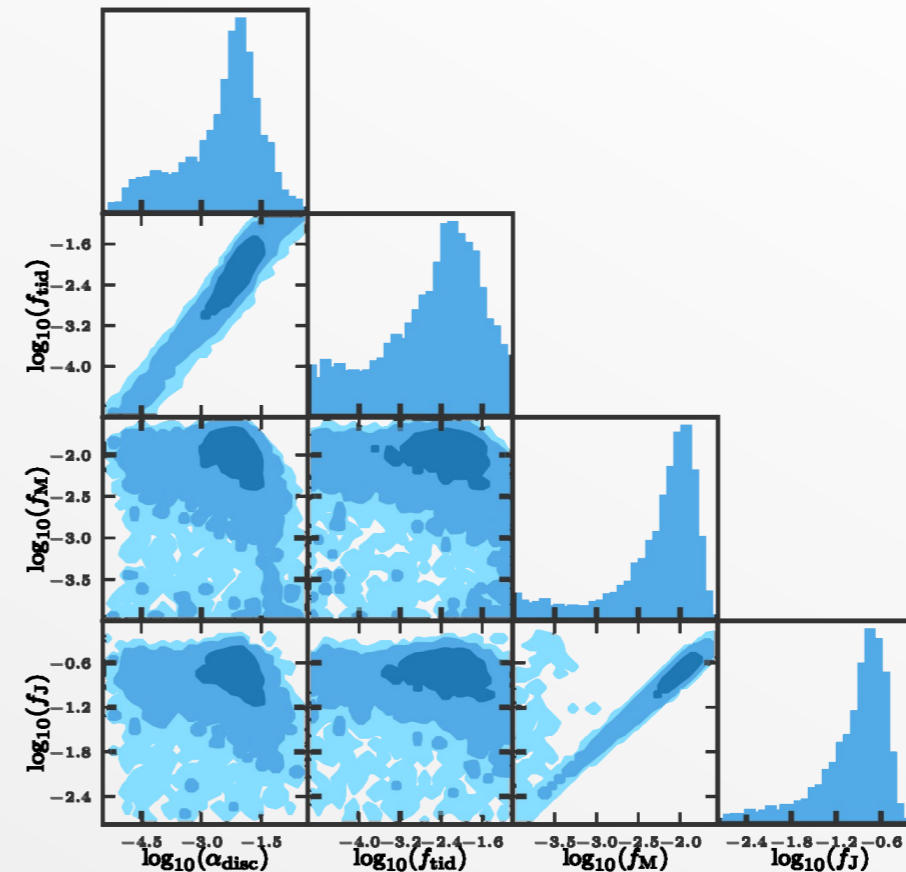


**Best model fit:**  $M = 26.8^{+2.5}_{-2.0} M_{\odot}$  age =  $2.0^{+0.5}_{-0.4}$



# Bayes + (MC)MC is easy these days

- Python packages abound
- *emcee* is popular
  - there are many others!
- better comparison O-C
  - error bars
  - bootstrap
  - constraints



binary\_c-python +  
emcee + pygtc

# Recap: We have discussed

- What a **population** of stars is
- How to **distribute** stars (initial mass function etc.)
- How to make **population statistics**
- How to compare the **population** with **observations**
- But we haven't discussed a powerful aspect of population synthesis...

# Binary stars

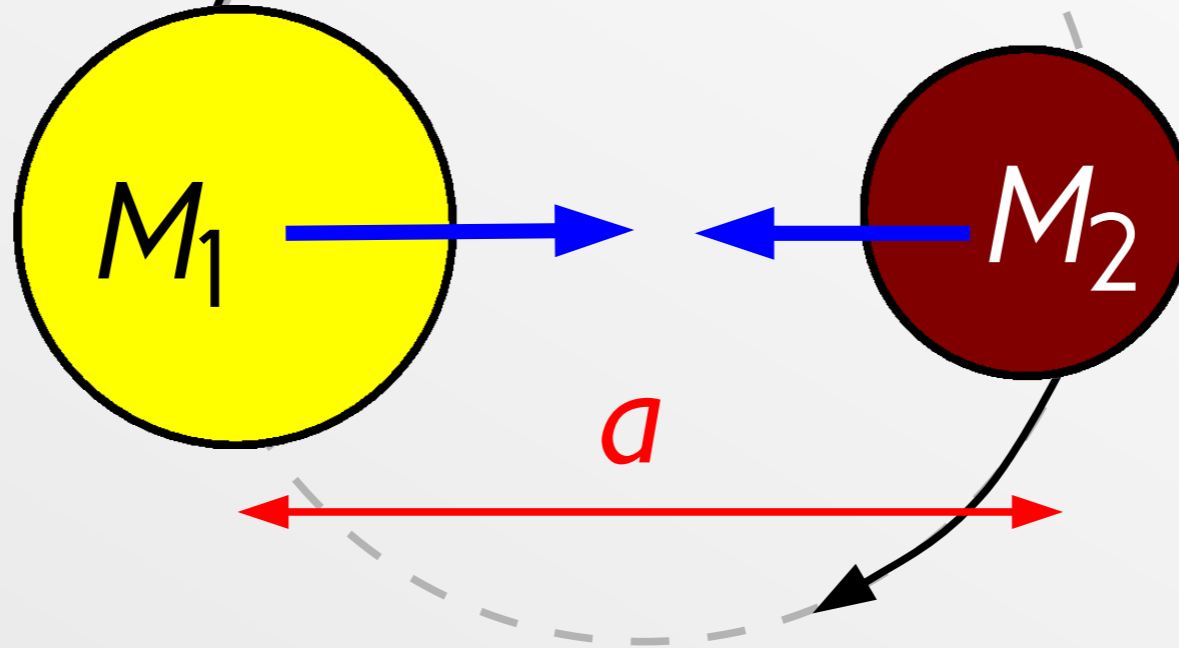
Two stars bound by  
*gravity*

Kepler



$$P^2 \propto a^3$$

1=Primary;  
2=Secondary

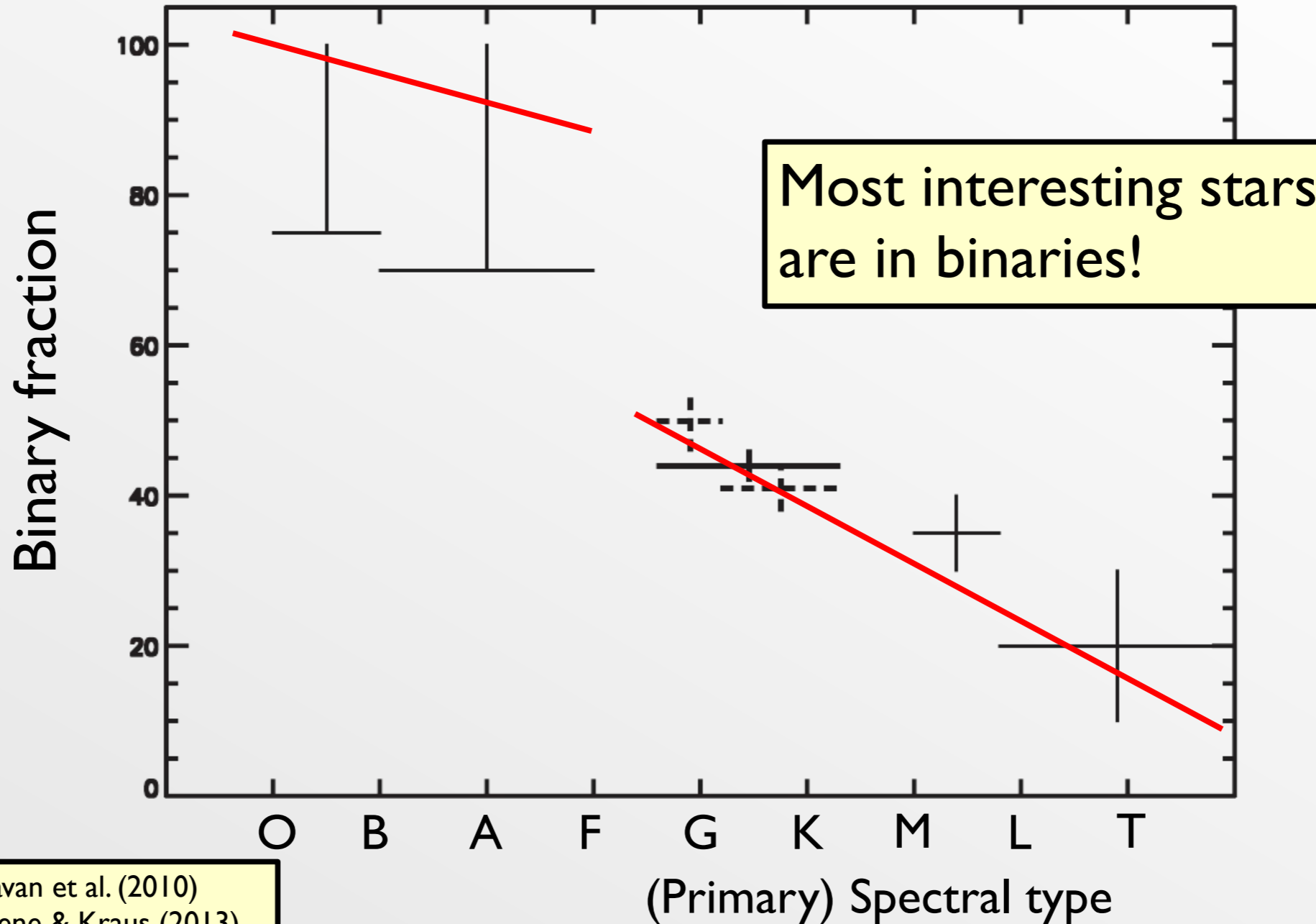


Newton



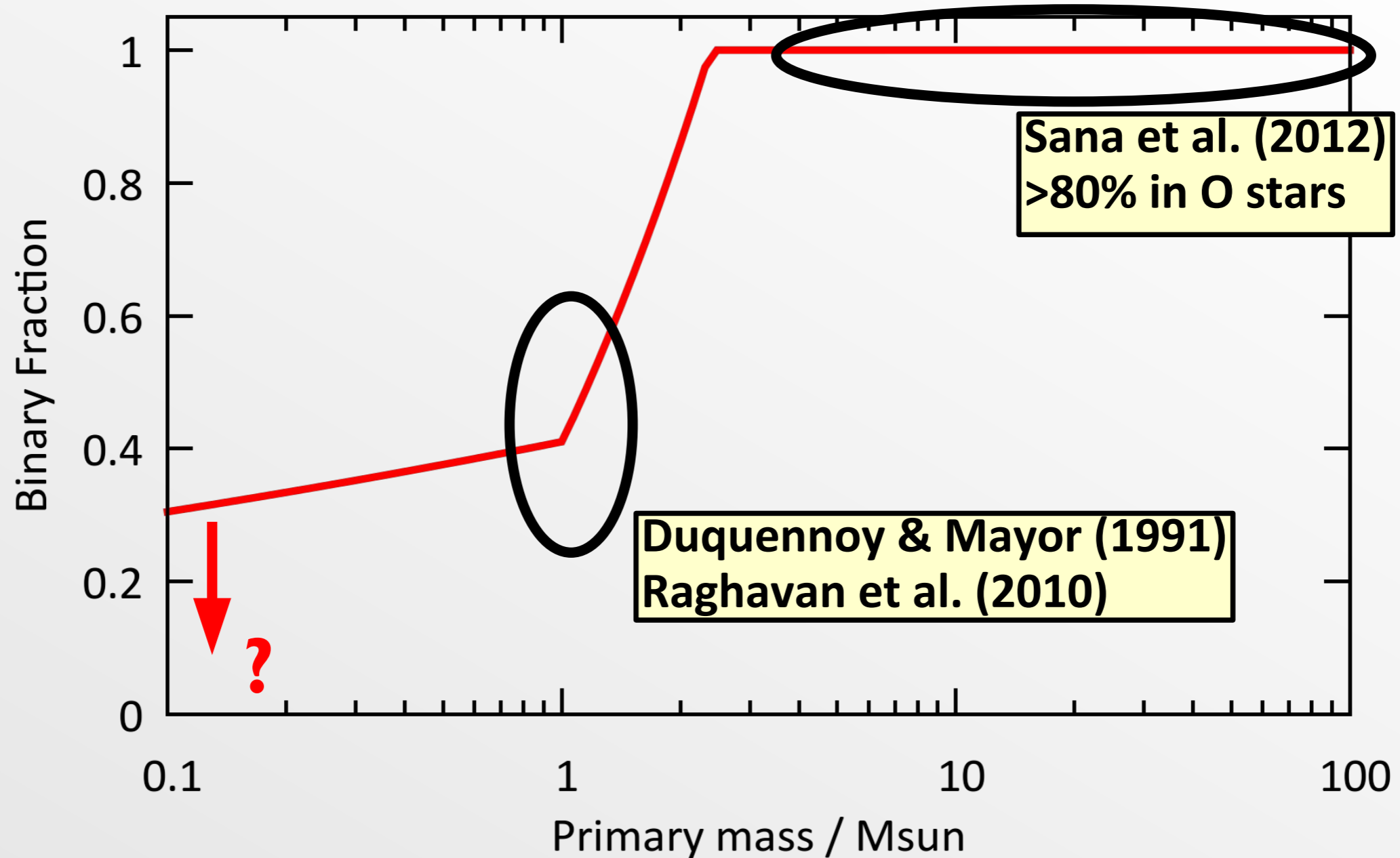
$$F = G \frac{M_1 M_2}{a^2}$$

# Binary fraction



Raghavan et al. (2010)  
Duchene & Kraus (2013)  
Moe and di Stefano (2017)

# Binary fraction

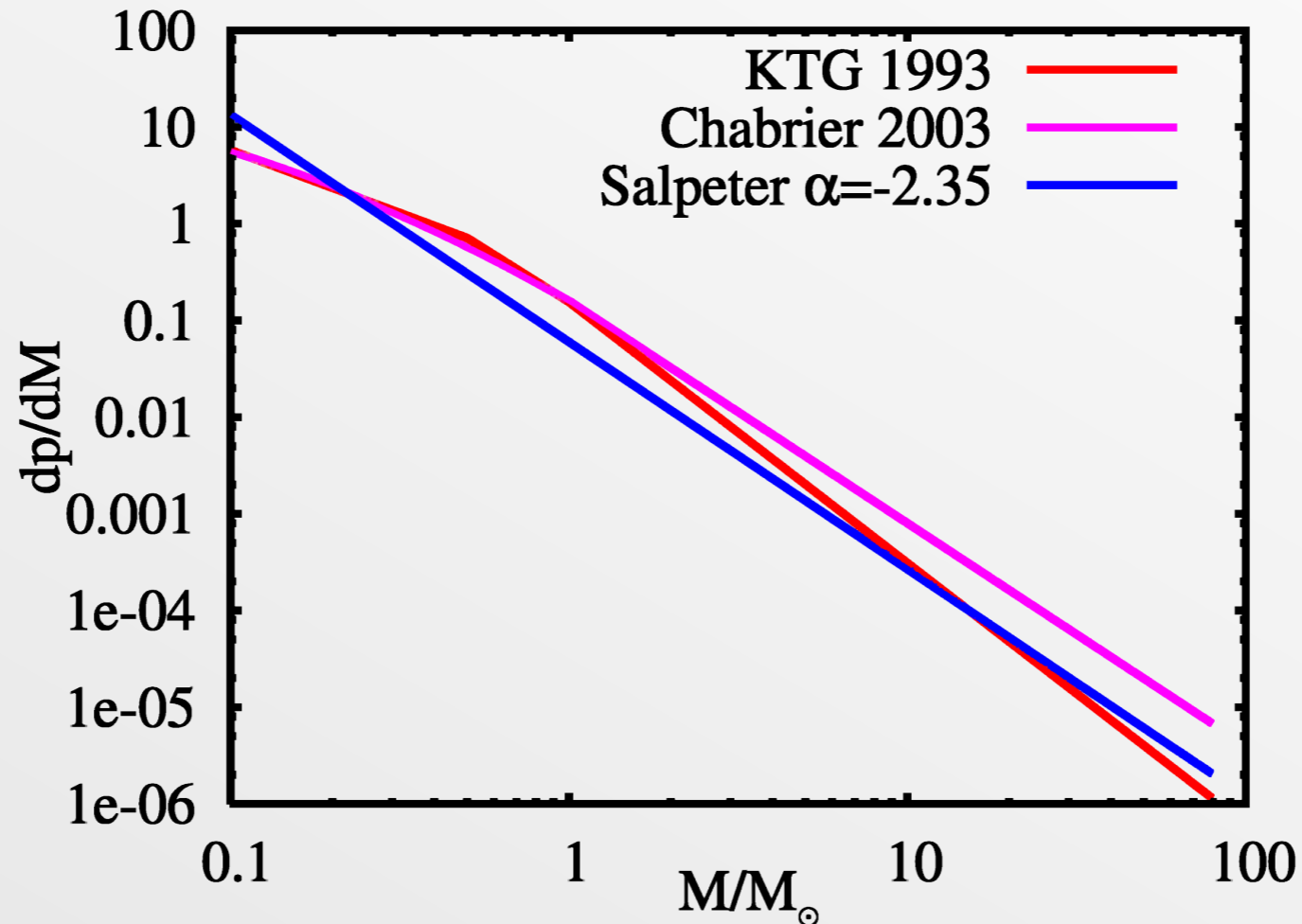


Moe and di Stefano (2017) → combines many surveys and selection effects → initial distributions

# Binary star parameter space

$$P(M_1, M_2, a) = \psi(M_1) \phi(M_2) \chi(a)$$

- Primary mass  $M_1 \sim$  follows single-star IMF  $\psi(M_1)$

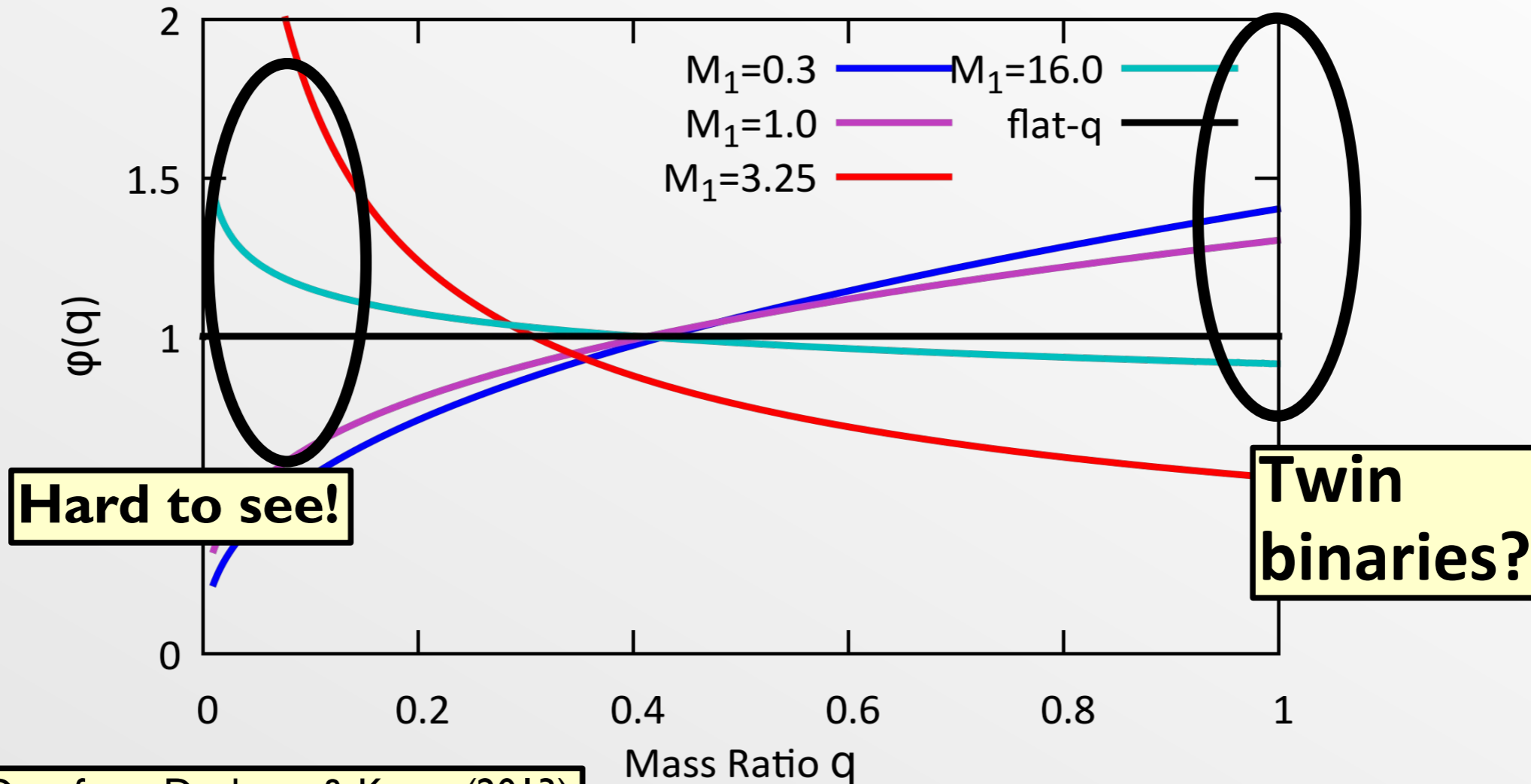


Universal? Probably not ... e.g. *GalIMF* module

# Binary star parameter space

$$P(M_1, M_2, a) = \psi(M_1) \phi(M_2) \chi(a)$$

- Secondary mass  $M_2 = q \times M_1$



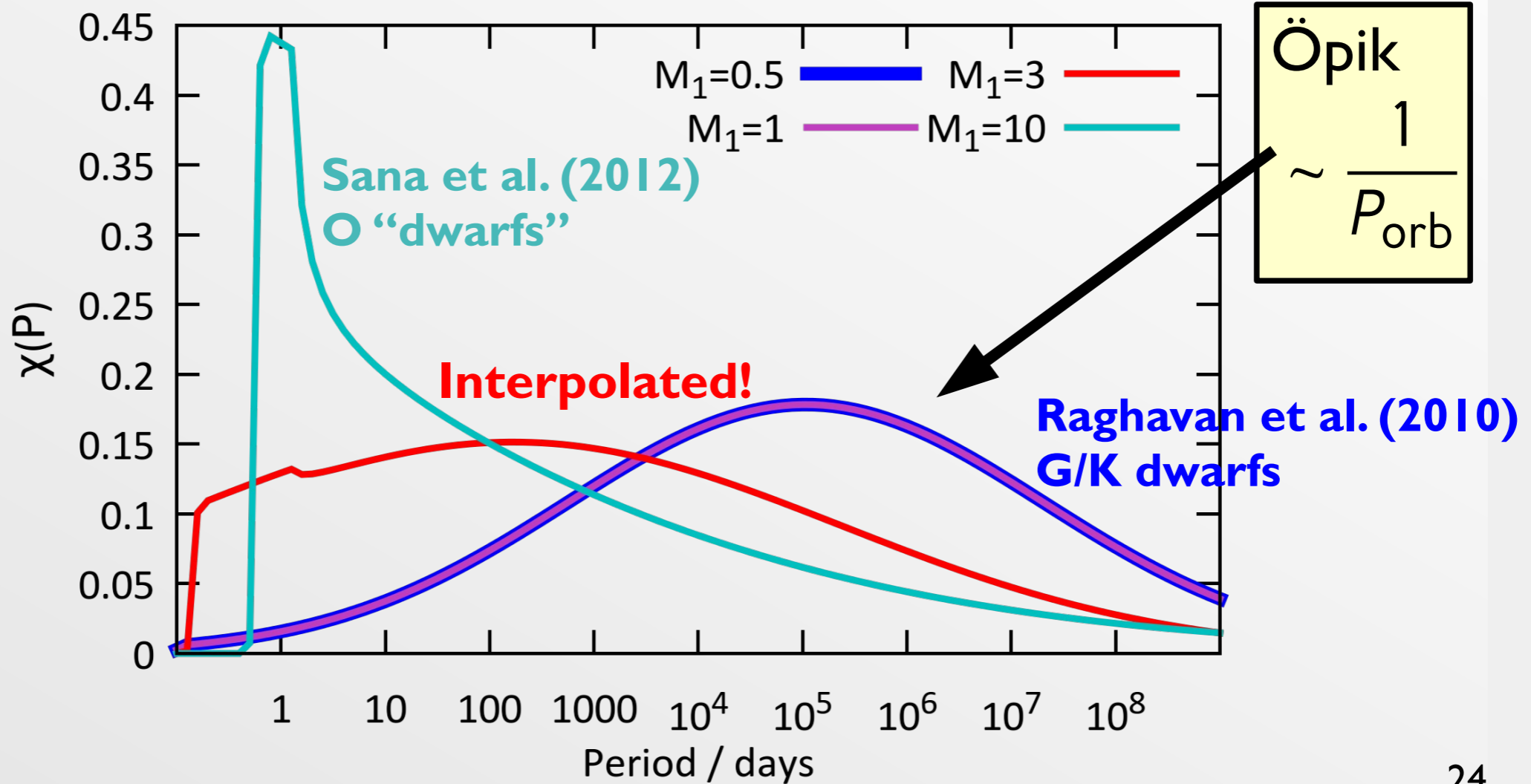
Data from Duchene & Kraus (2013)

# Binary star parameter space

$$P(M_1, M_2, a) = \psi(M_1) \phi(M_2) \chi(a)$$

- Separation/Period

$$P_{\text{orb}}^2 = \frac{4\pi^2}{G(M_1 + M_2)} a^3$$



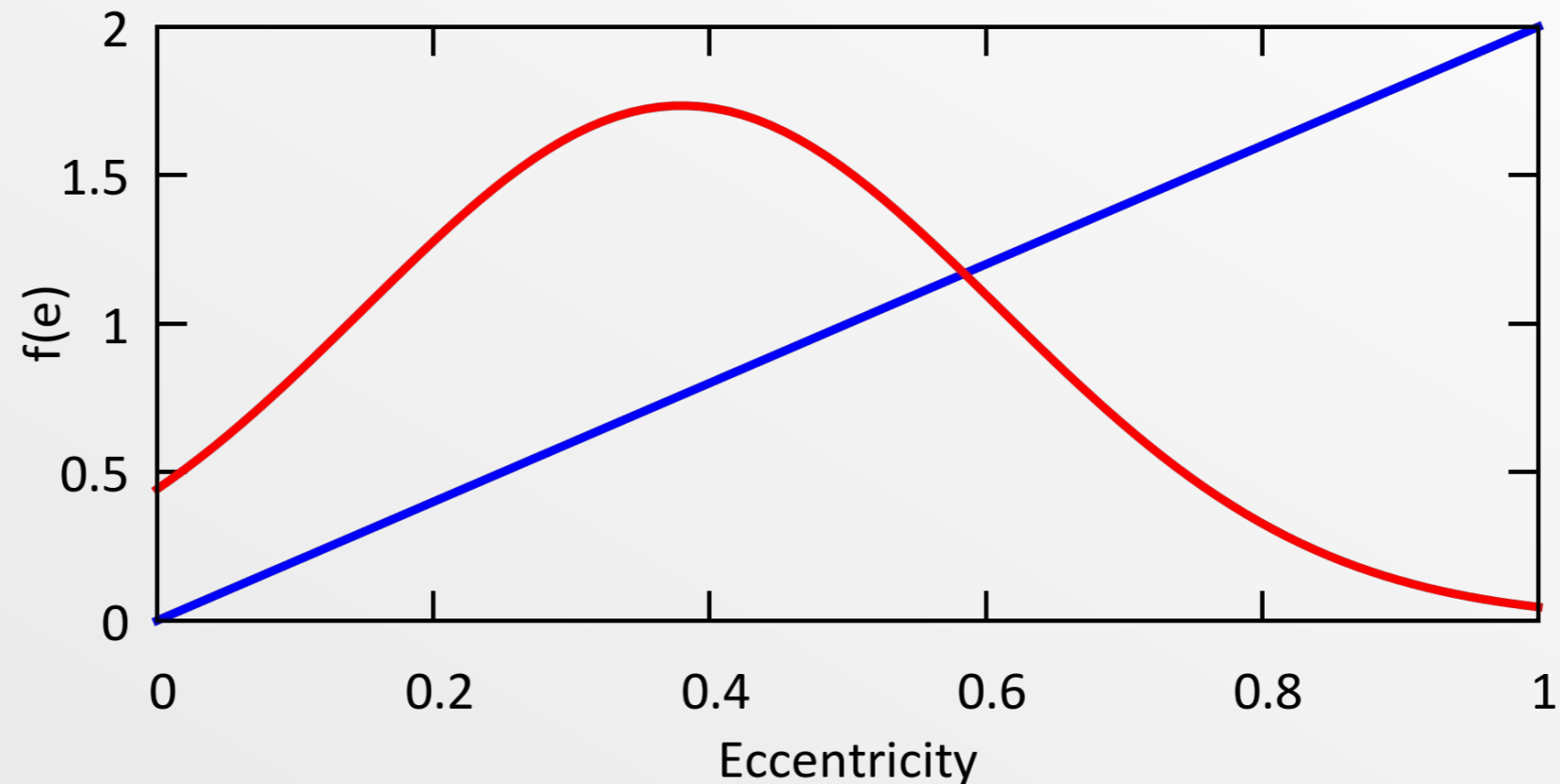


# Binary star parameter space

$$P(M_1, M_2, a, e) = \psi(M_1) \phi(M_2) \chi(a) \times f(e)$$

- Eccentricity

'Thermal'  $f(e)=2e$  — blue line  
Meiborn Mathieu (2005) — red line



**Safe(ish) to assume  $e=0$  ...**

# Binary statistics reminder: not trivial!

Number of stars of interest:

$$N = f_{\text{bin}} \sum_{M_1} \sum_{M_2} \sum_a \sum_e \sum_t S(t) \psi(M_1) \phi(M_2) \chi(a) f(e) \bar{\delta}(t) \delta t + \\ (1 - f_{\text{bin}}) \sum_M \sum_t S(t) \psi(M) \bar{\delta}(t) \delta t$$

Usually: set  $e=0$ ,  $S=1$ , fix  $Z \rightarrow \bar{\delta}(t)$  and calculate ratios

(see previous lecture)

But still a lot of work: *at least* a **quadruple** sum.

Now you see why **fast synthetic codes** are useful!

# Binaries: fundamental properties

Masses:  $M_1$   $M_2$

Energy:  $E = \frac{1}{2} M_1 |\dot{\mathbf{r}}_1|^2 + \frac{1}{2} M_2 |\dot{\mathbf{r}}_2|^2 - \frac{GM_1 M_2}{r}$

Angular momentum:

$$\mathbf{J} = M_1 \mathbf{r}_1 \times \dot{\mathbf{r}}_1 + M_2 \mathbf{r}_2 \times \dot{\mathbf{r}}_2$$



# Tides: synchronization and circularization

- Torque  $\Gamma = -\frac{\Omega - \omega}{\tau_{\text{diss}}} q^2 MR^2 \left(\frac{R}{a}\right)^6$
- Hence lowest energy state:  $\Omega = \omega$  and  $e = 0$

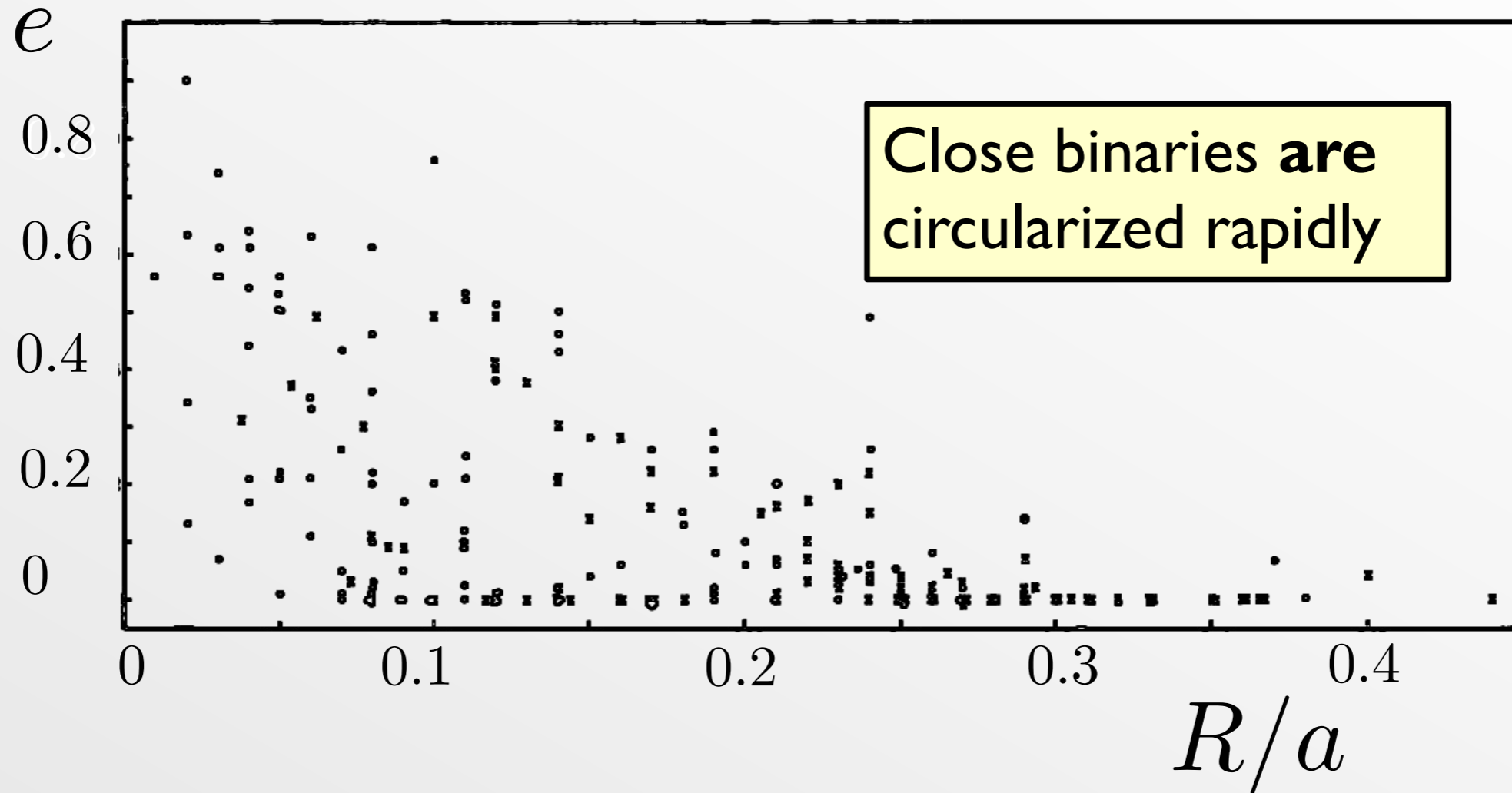
$$\tau_{\text{sync}}^{-1} = \frac{-\Gamma}{I\Omega} = \frac{1}{\tau_{\text{diss}}} \frac{\Omega - \omega}{\Omega} q^2 \frac{MR^2}{I} \left(\frac{R}{a}\right)^6$$

$$\tau_{\text{circ}}^{-1} = \frac{\dot{e}}{e} = \frac{1}{\tau_{\text{diss}}} \left(9 - \frac{11}{2} \frac{\Omega}{\omega}\right) q(1+q) \left(\frac{R}{a}\right)^8$$

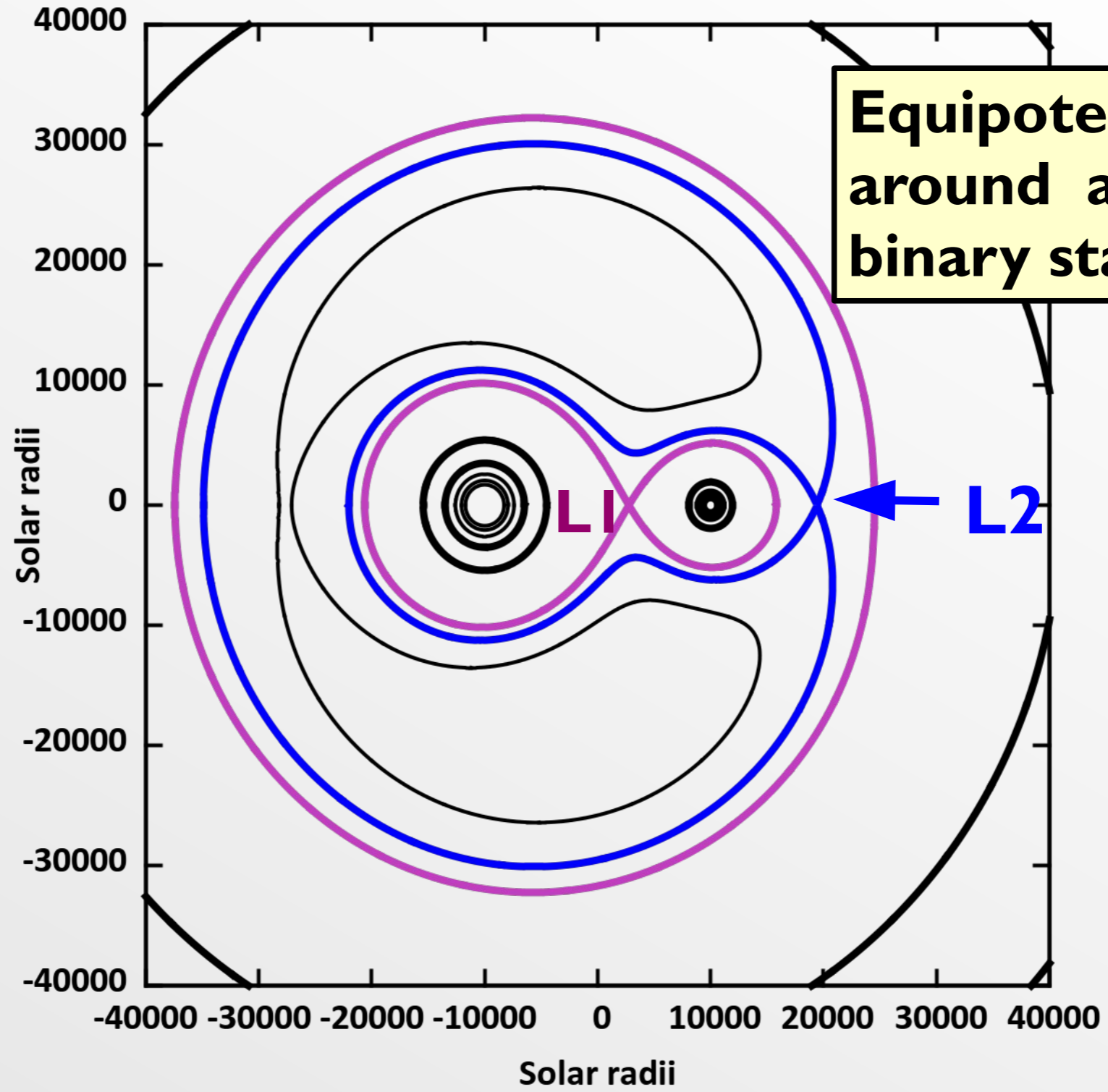
**e.g. Zahn 1972, 1980s**  
**Hut 1981, book of Tassoul etc.**



# $e$ vs $R/a$ (unevolved binaries)



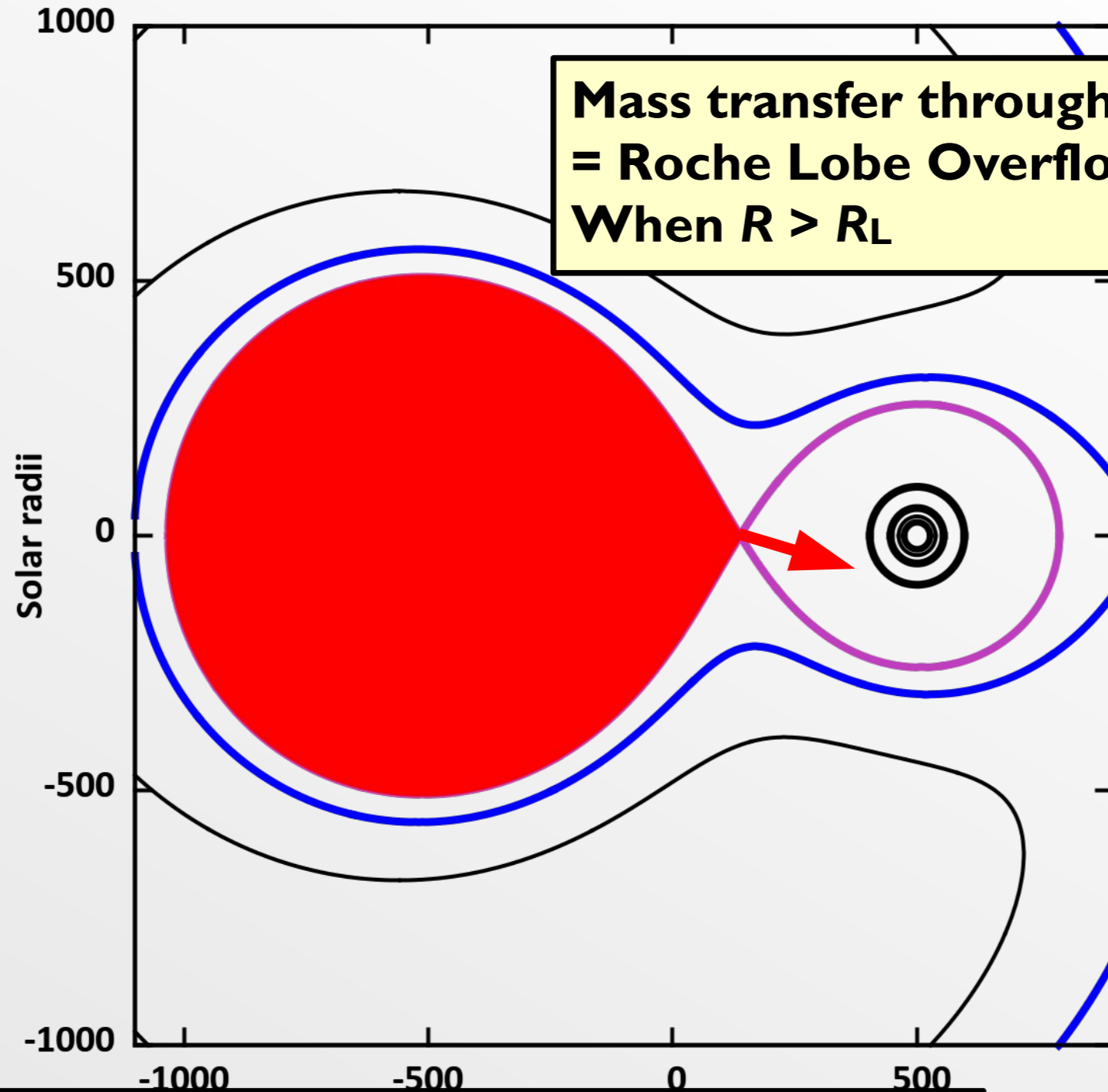
Data from  
Giuricin, G.; Mardirossian, F.; Mezzetti  
Astronomy and Astrophysics 134, 365



**Equipotentials  
around a  
binary star**

**L2**

**L1**

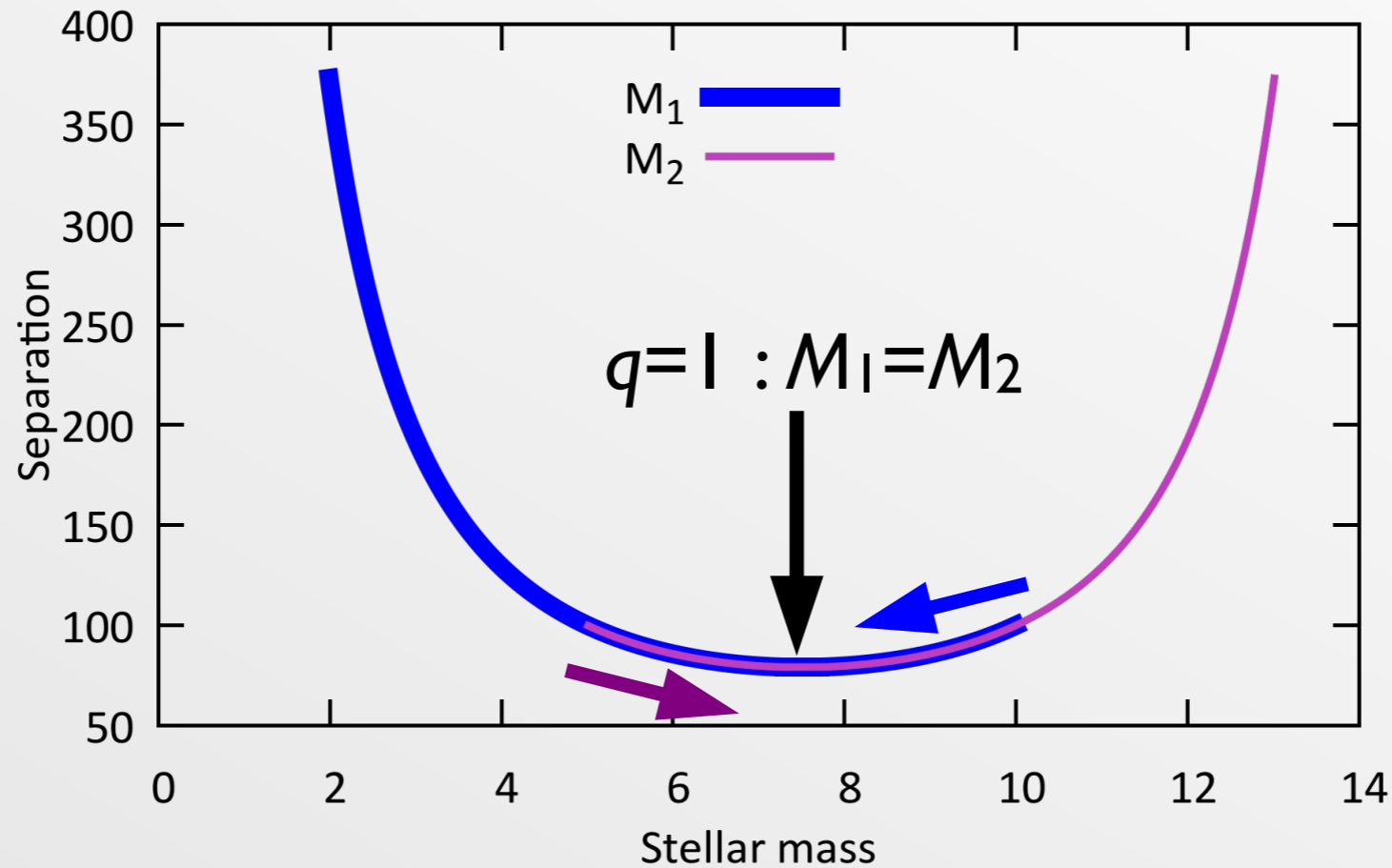


**Mass transfer through  $L_1$   
= Roche Lobe Overflow (RLOF)  
When  $R > R_L$**

**Roche radius  $R_L = a f(q)$  e.g. Eggleton (1983)**

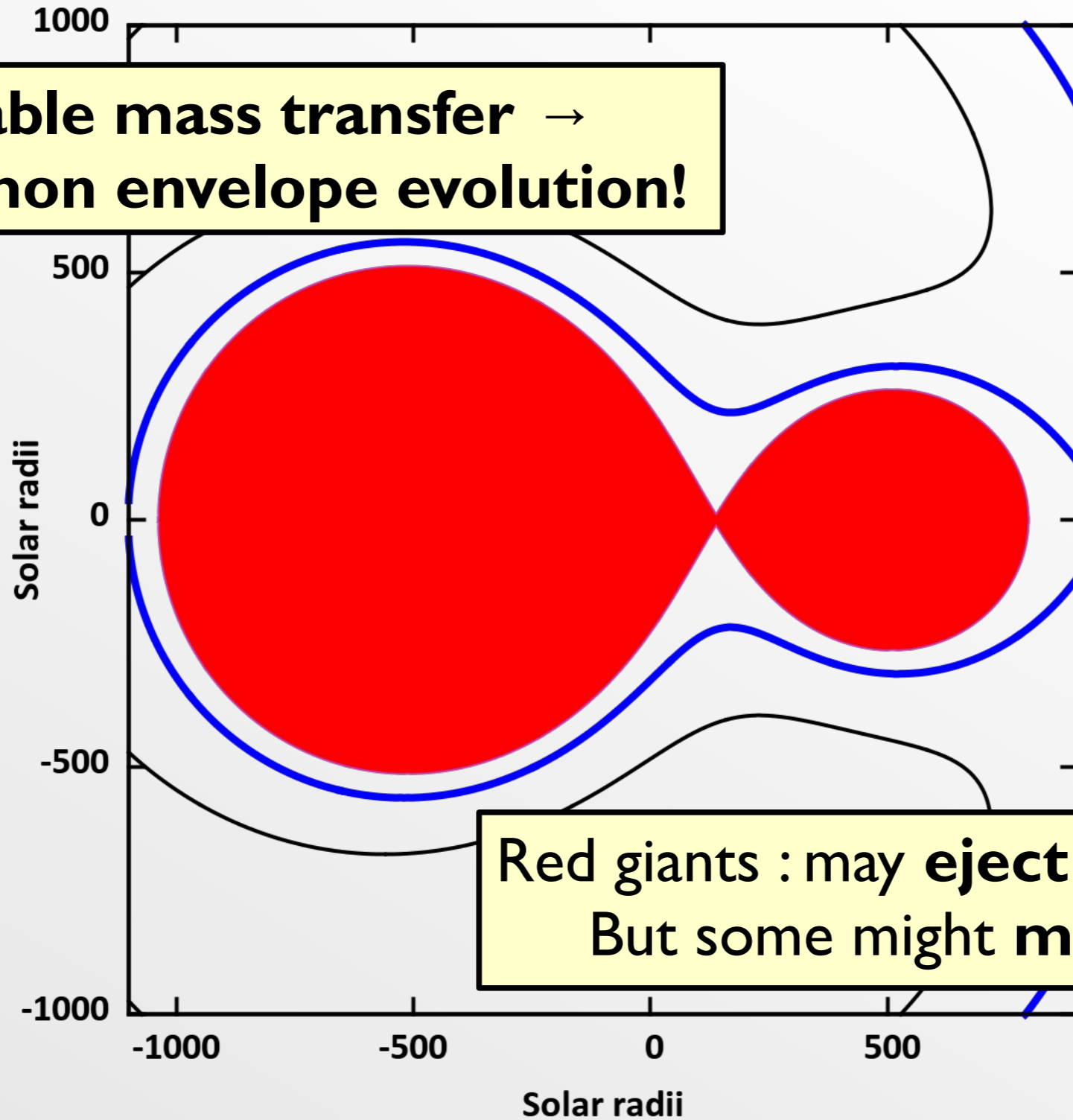
# Conservative mass transfer

$$\left. \begin{aligned} j_1 + j_2 &= 0 & \dot{M}_1 + \dot{M}_2 &= 0 \end{aligned} \right\} (M_1 M_2)^2 a = \text{const.}$$
$$J = \frac{M_1 M_2}{M_1 + M_2} \sqrt{G(M_1 + M_2) a}$$



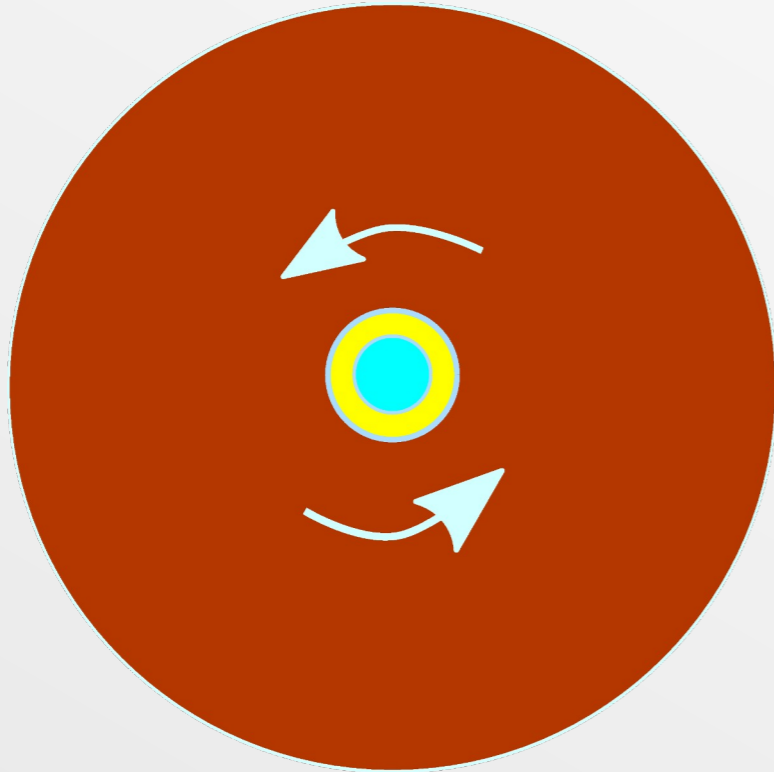


**Unstable mass transfer →  
common envelope evolution!**

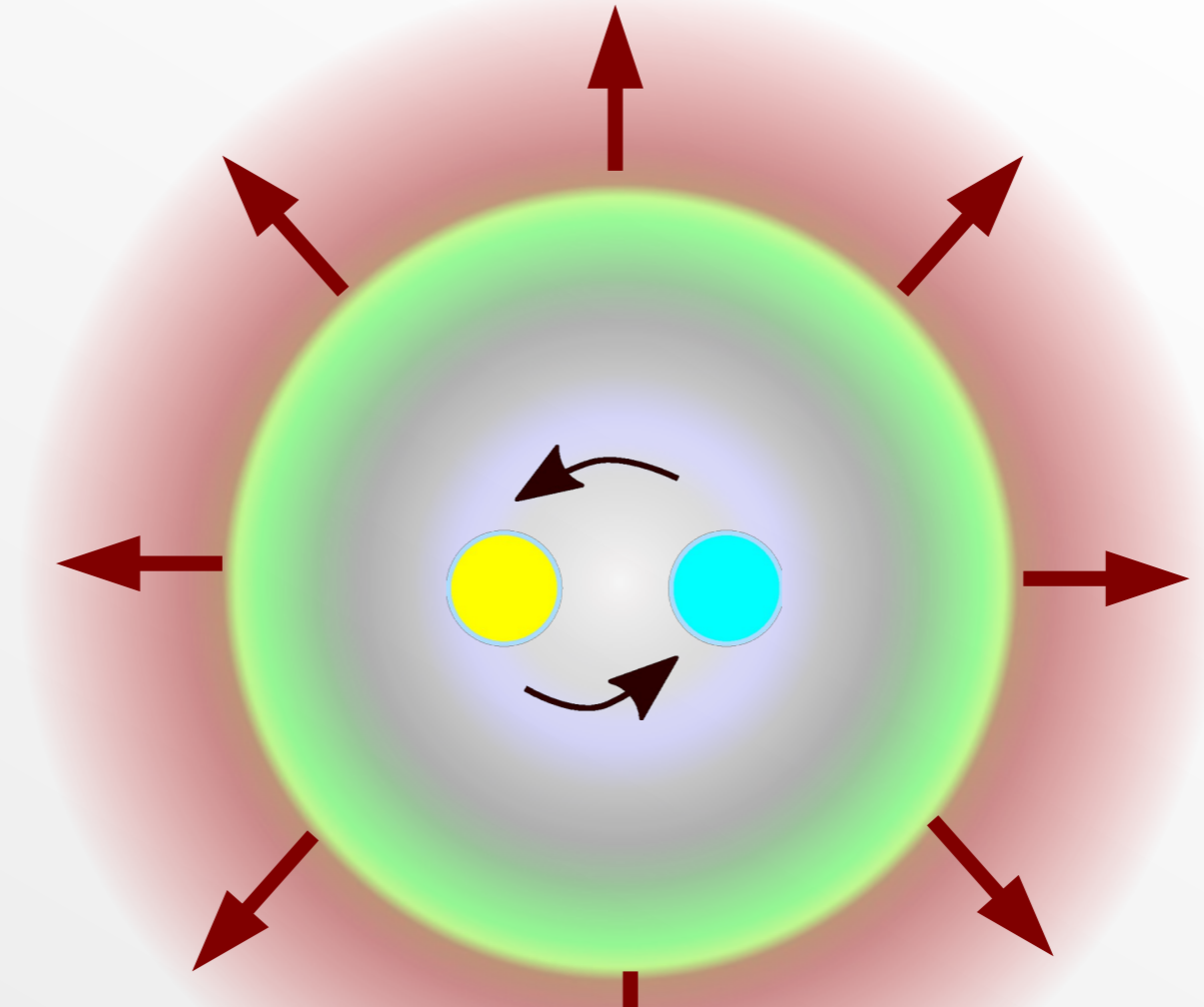


**Red giants : may eject envelope  
But some might merge!**

# Comenv Final Fate ... ?



**Merger:  
Rapidly Spinning  
Giant**



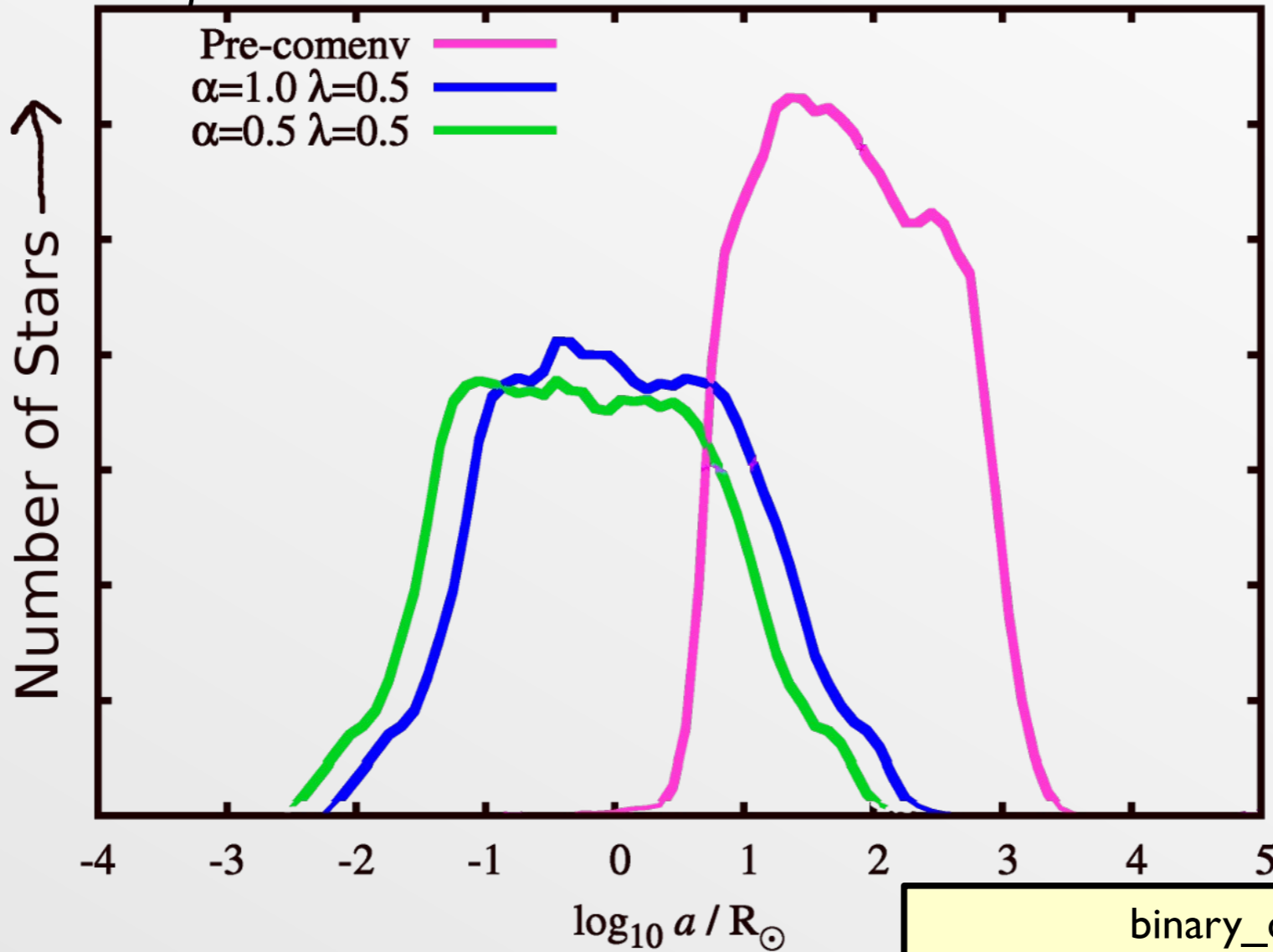
**Ejected (planetary?) nebula  
Close binary at the core with  
white dwarf**

$$\Delta E_{\text{bind}} = \alpha \Delta E_{\text{orb}}$$

$$\frac{a_f}{a_i} = f(M_1, M_{c1}, M_2, \alpha, \lambda)$$

$$\lambda \sim \frac{1}{E_{\text{bind}}}$$

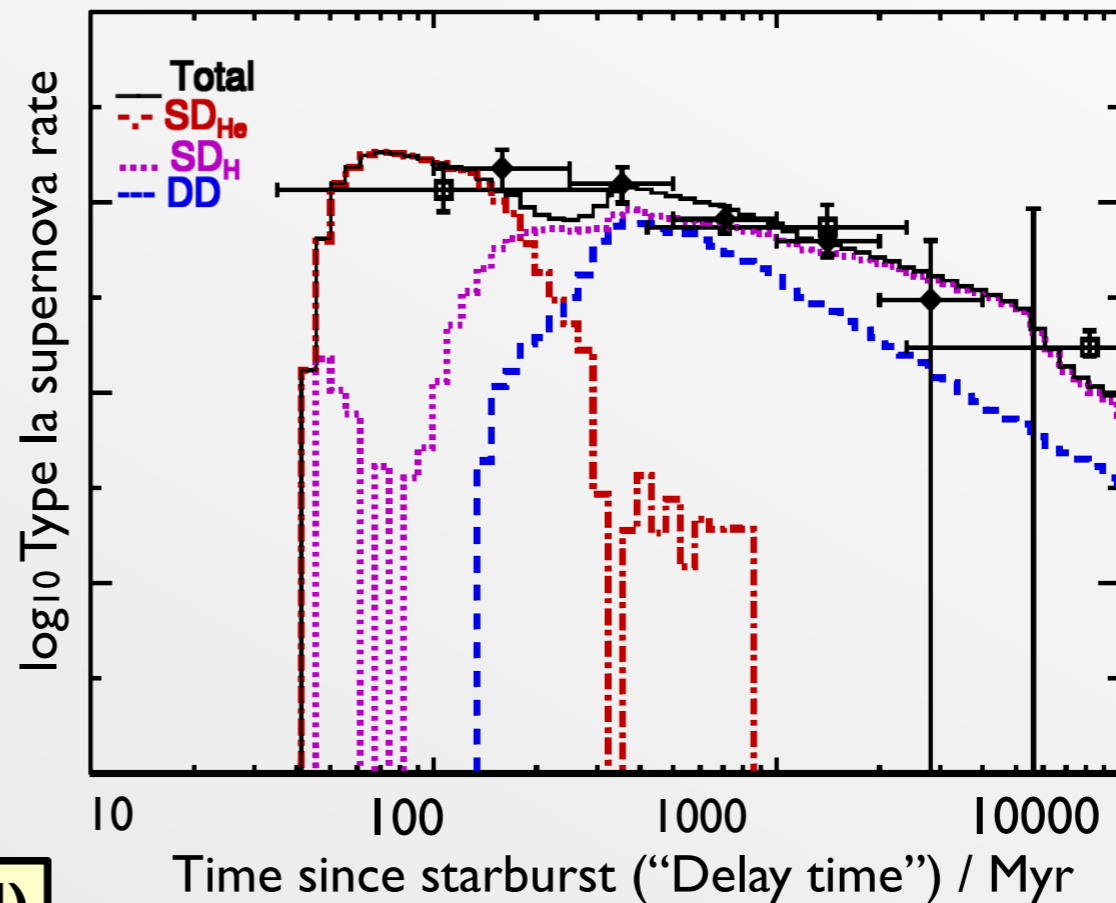
$$\alpha \lesssim 1$$



binary\_c-python  
notebook\_common\_envelope\_evolution

# Close systems I: low/intermediate mass

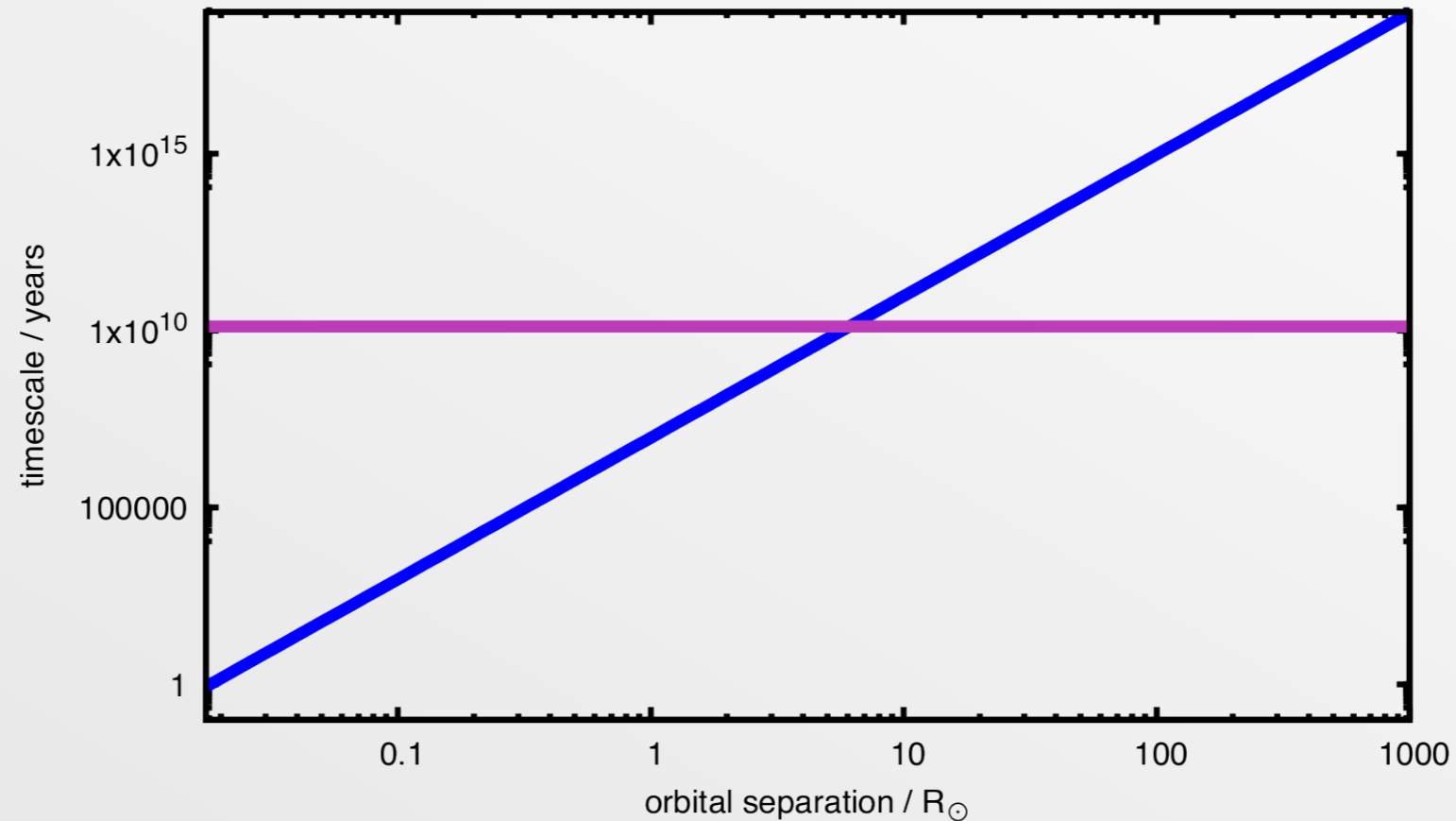
- Donors are *usually* red-giant stars
  - hydrogen shell burning, maybe AGB stars → comenv
- → He/CO/ONeWDs, perhaps sdB/O stars
- accretion from surviving star → novae, type Ia SNe
- COWD? - COWD binaries → SNe Ia



Clayey et al. (2014)

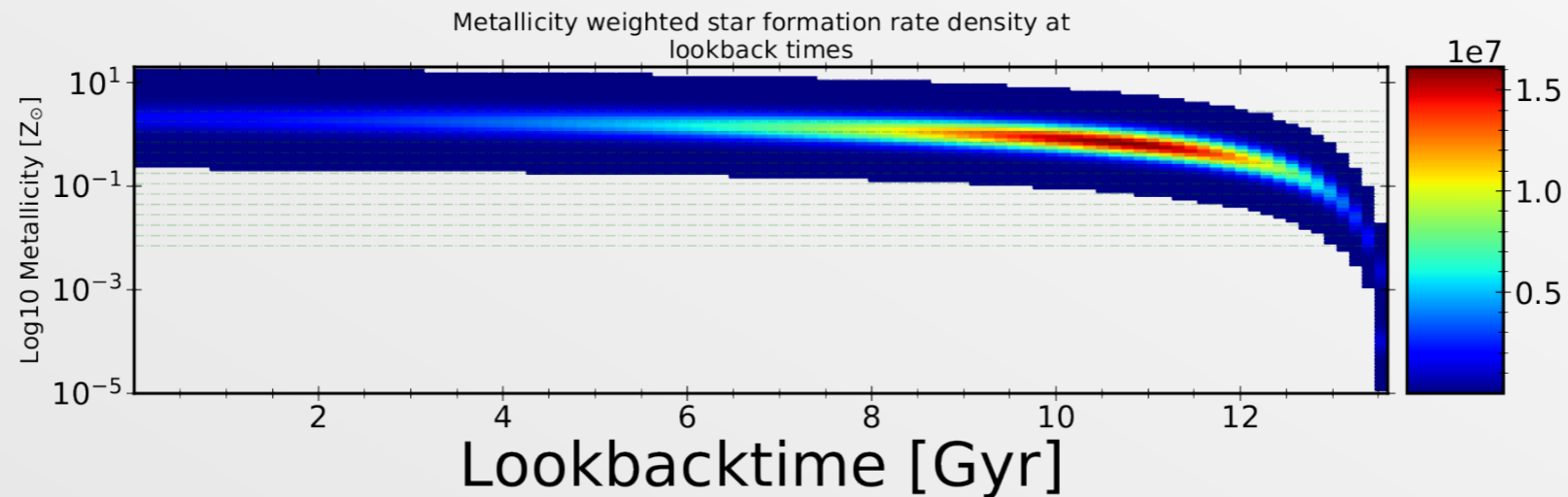
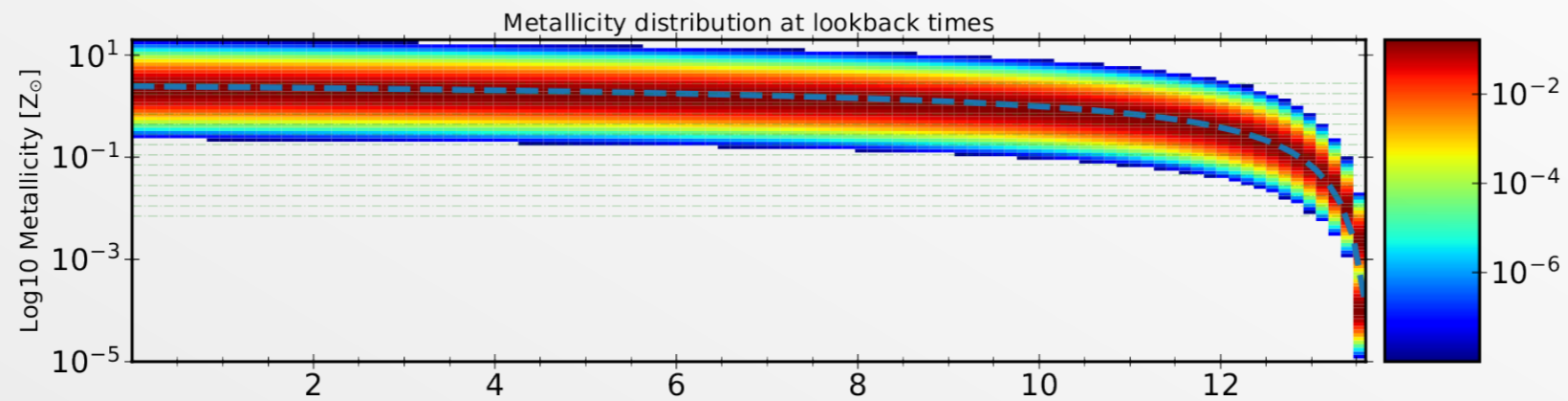
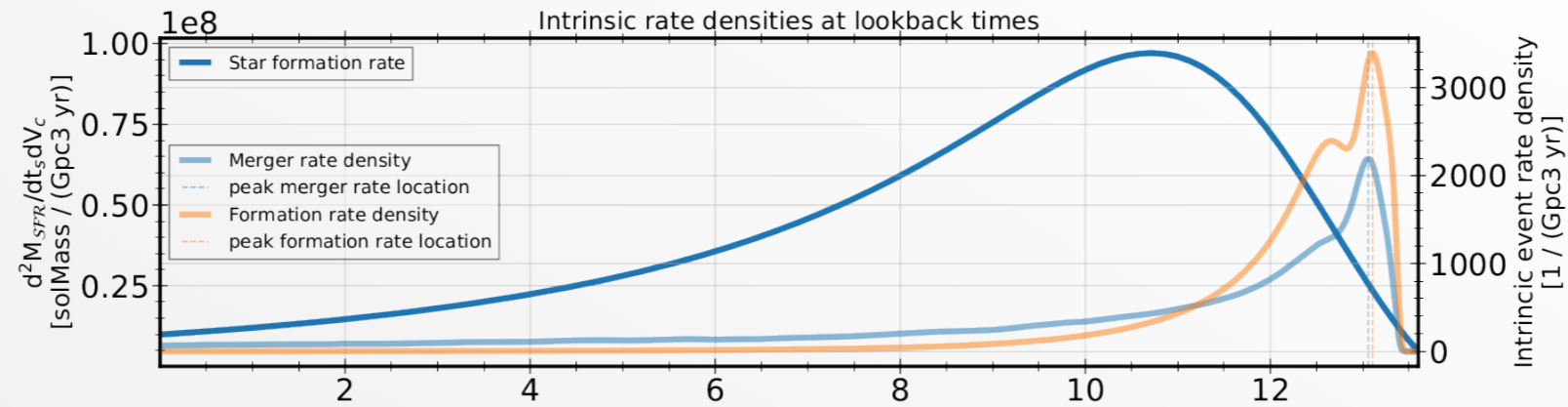
# Close systems II: high mass

- X-ray binaries  $\rightarrow$  NS+NS, NS+BH, BH+BH systems  
 $\rightarrow$  close enough for gravitational radiation
- Peters 1964 timescale



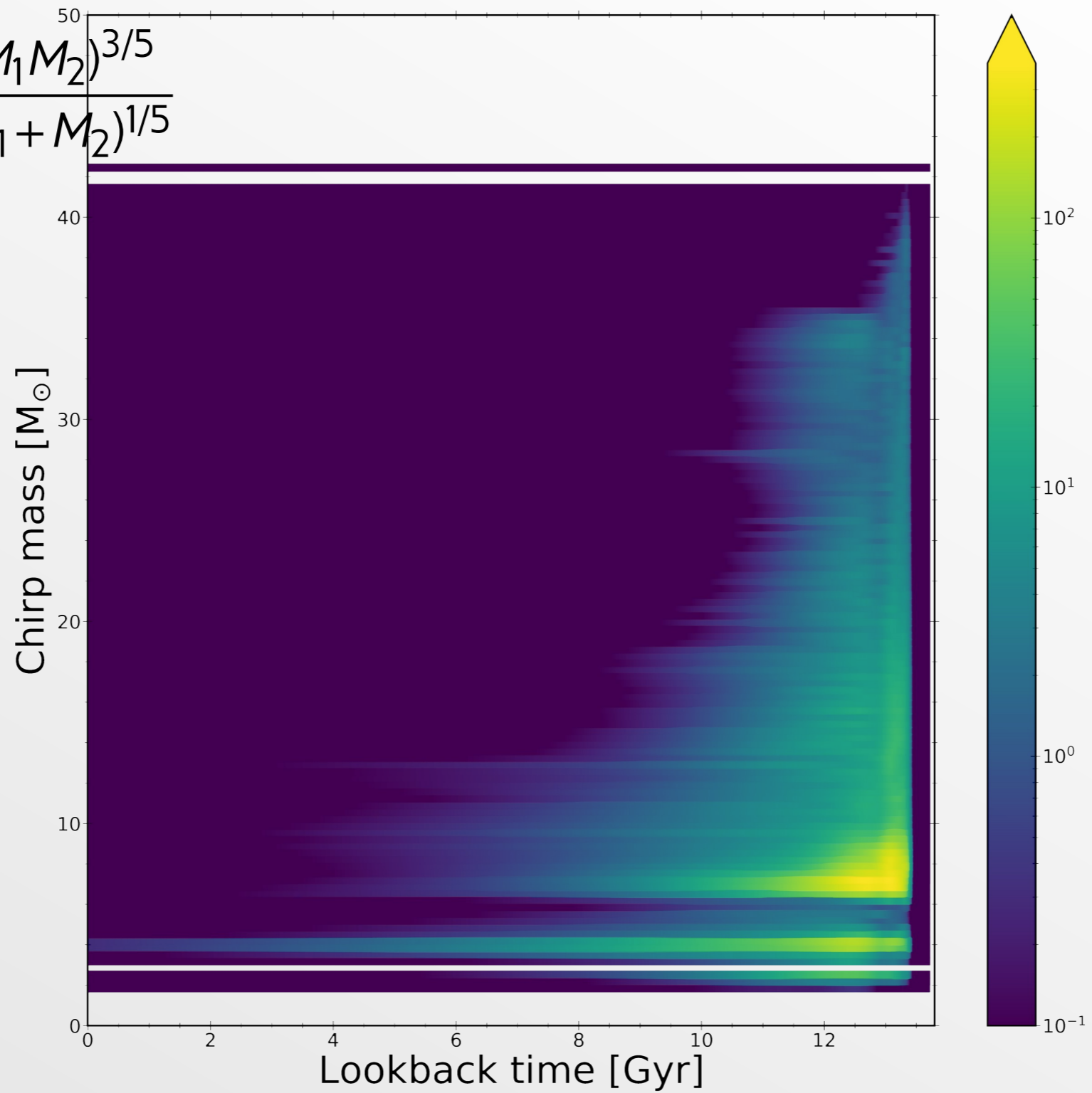
# Close systems II: high mass

- X-ray binaries, NS+NS, NS+BH, BH+BH systems
  - close enough for gravitational radiation
- System needs to survive mass loss *and* SN kicks
  - unlikely, so requires “luck” → many systems



**David Hendriks** models with binary\_c-python

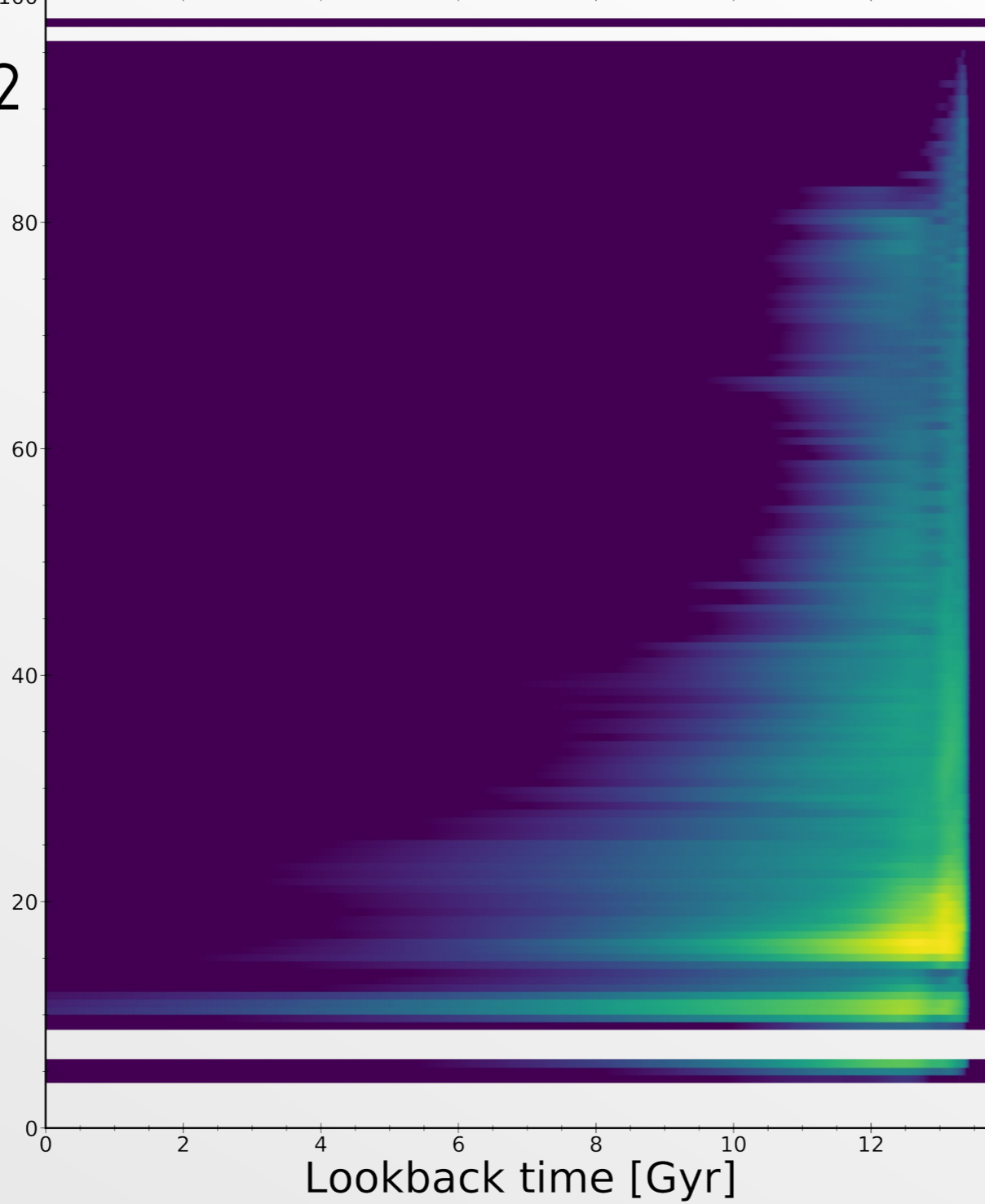
$$\mathcal{M} = \frac{(M_1 M_2)^{3/5}}{(M_1 + M_2)^{1/5}}$$





$M_1 + M_2$

Total mass [ $M_\odot$ ]



Lookback time [Gyr]

$10^2$

$10^1$

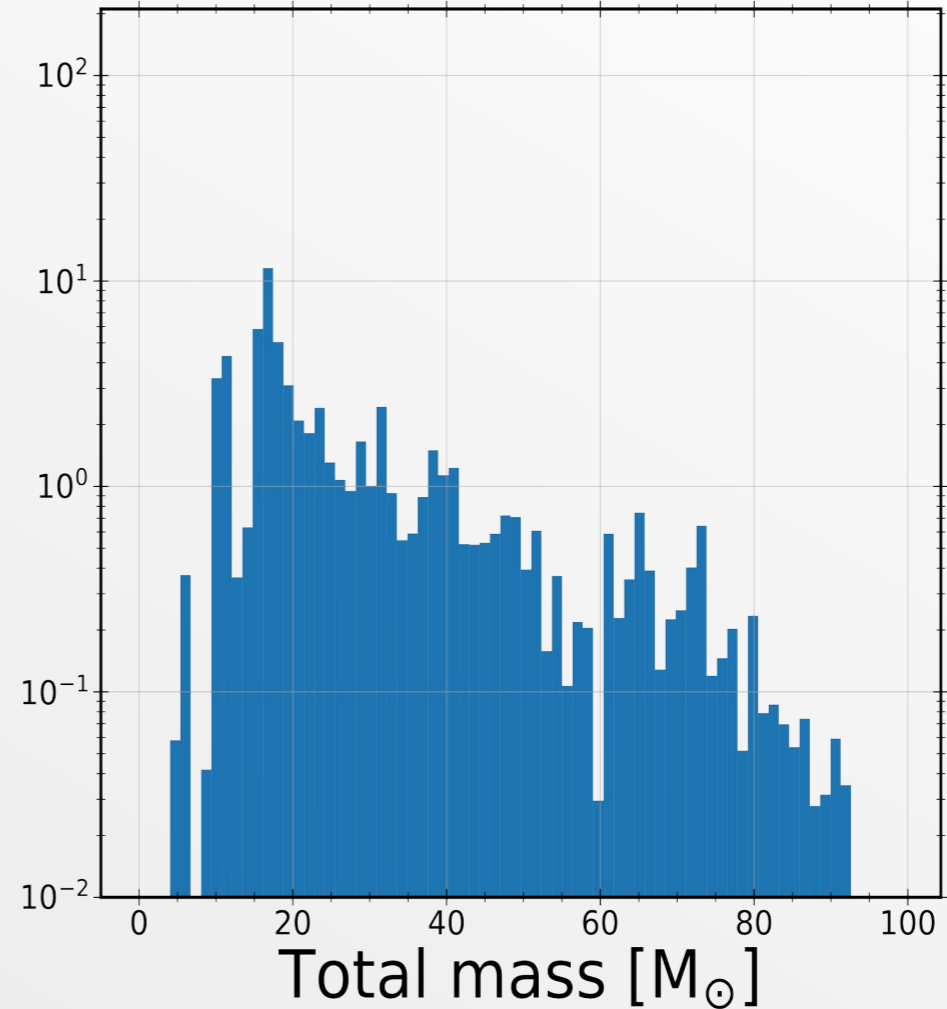
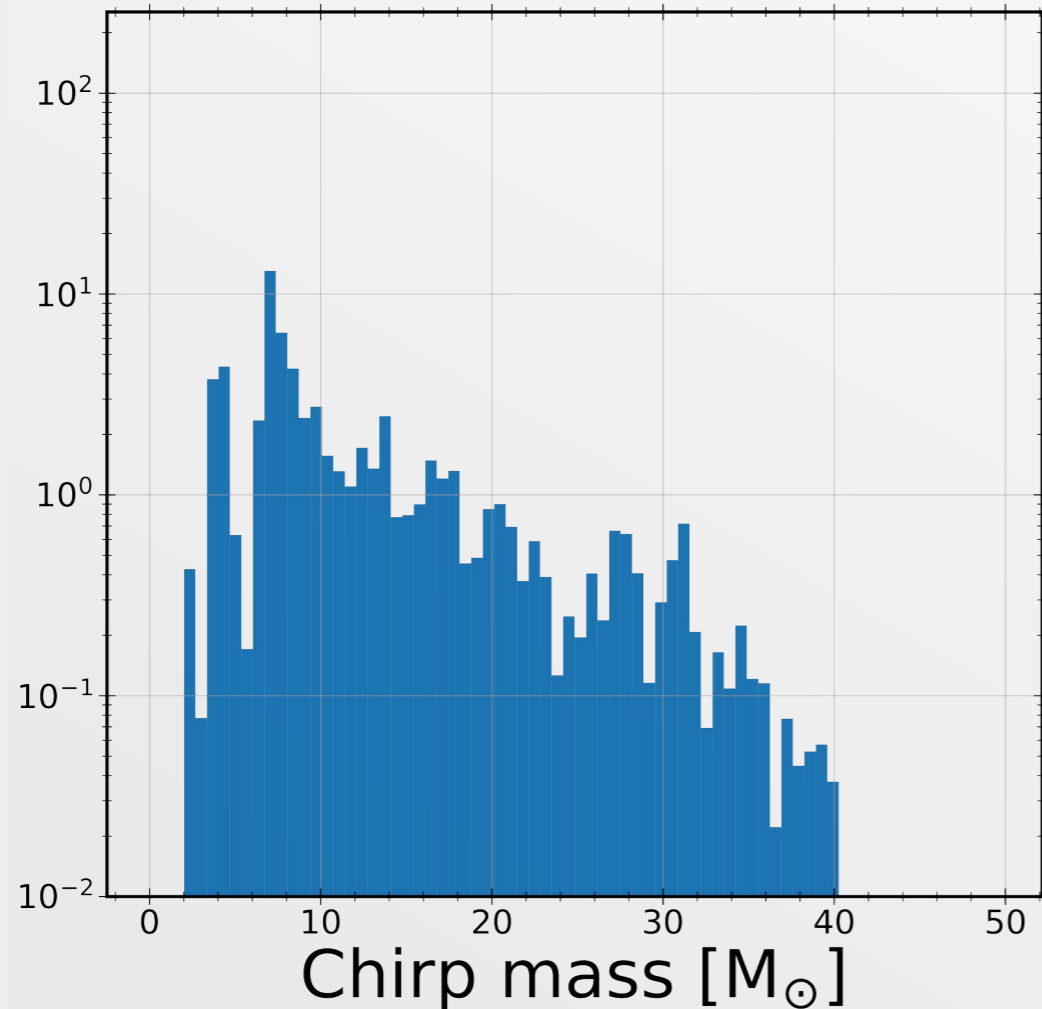
$10^0$

$10^{-1}$

# Predicted distributions of BHBH systems *now*

→ ready to fold in selection effects

→ then compare with observations

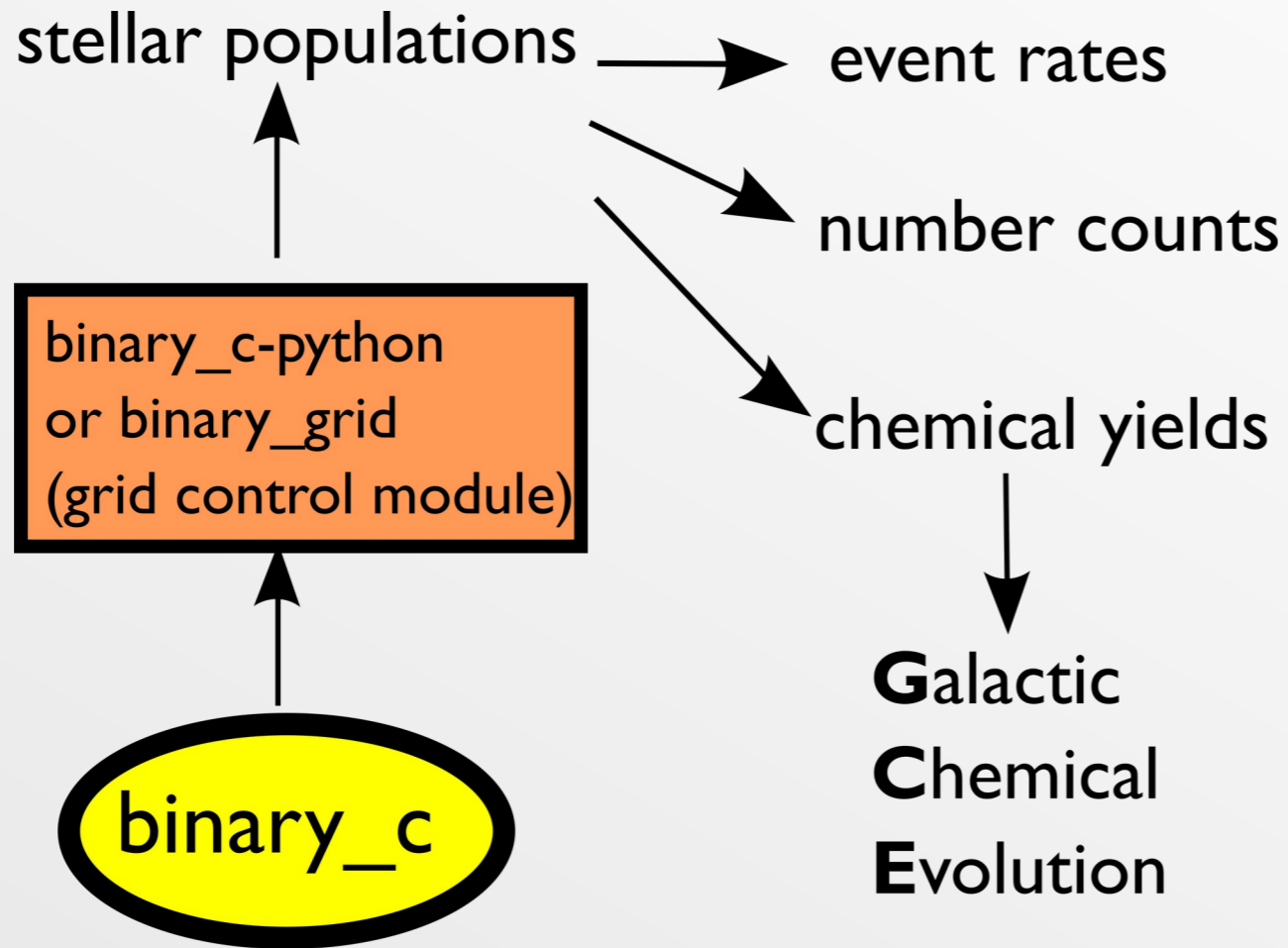


# Other important binary physics

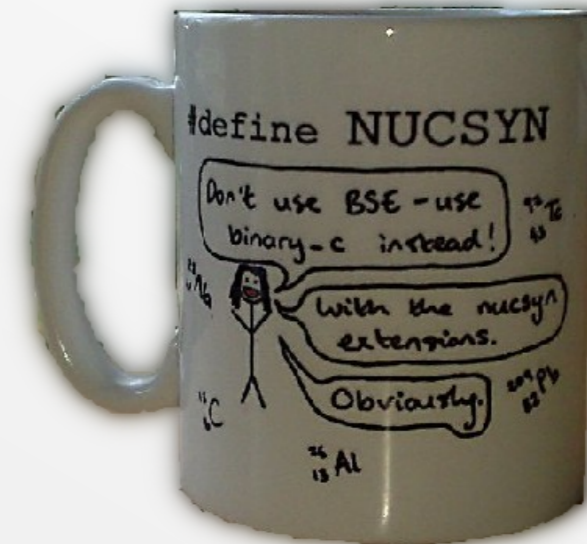
- **Supernova** kicks: e.g. eccentric X-ray binaries
- Exotic stars e.g. **helium stars**, sdB/O stars
- **Gravitational radiation**: spiral in of very close binaries: NS/BH+NS/BH, WD+WD → bang!
- **Rejuvenation**: accretion makes stars younger
- **Wind accretion** in wide binaries (see case studies)
- **Circumbinary discs** → eccentricity pumping, accretion

All this physics is in my binary popsyn code ***binary\_c***  
[http://personal.ph.surrey.ac.uk/~ri0005/binary\\_c.html](http://personal.ph.surrey.ac.uk/~ri0005/binary_c.html)

# *binary\_c* code



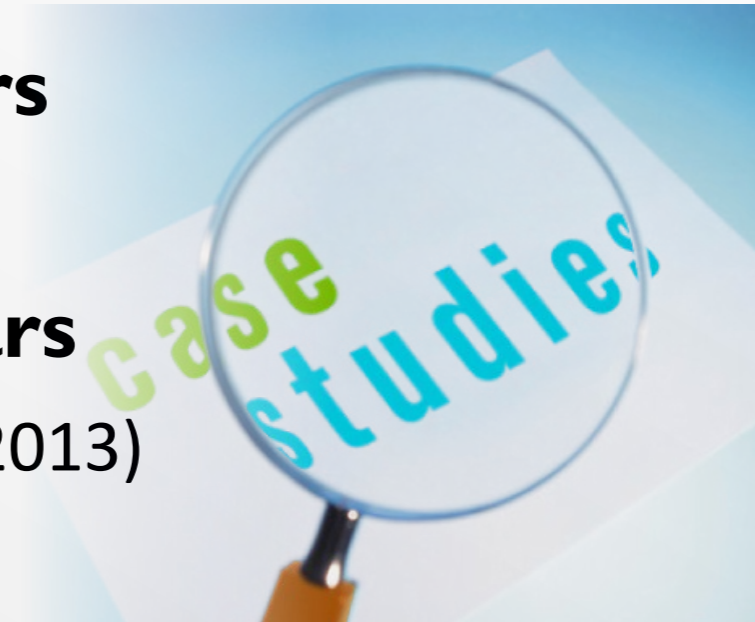
**You'd be a  
mug not to  
use it!**



[http://personal.ph.surrey.ac.uk/~ri0005/binary\\_c.html](http://personal.ph.surrey.ac.uk/~ri0005/binary_c.html)

# Case studies with binary\_c

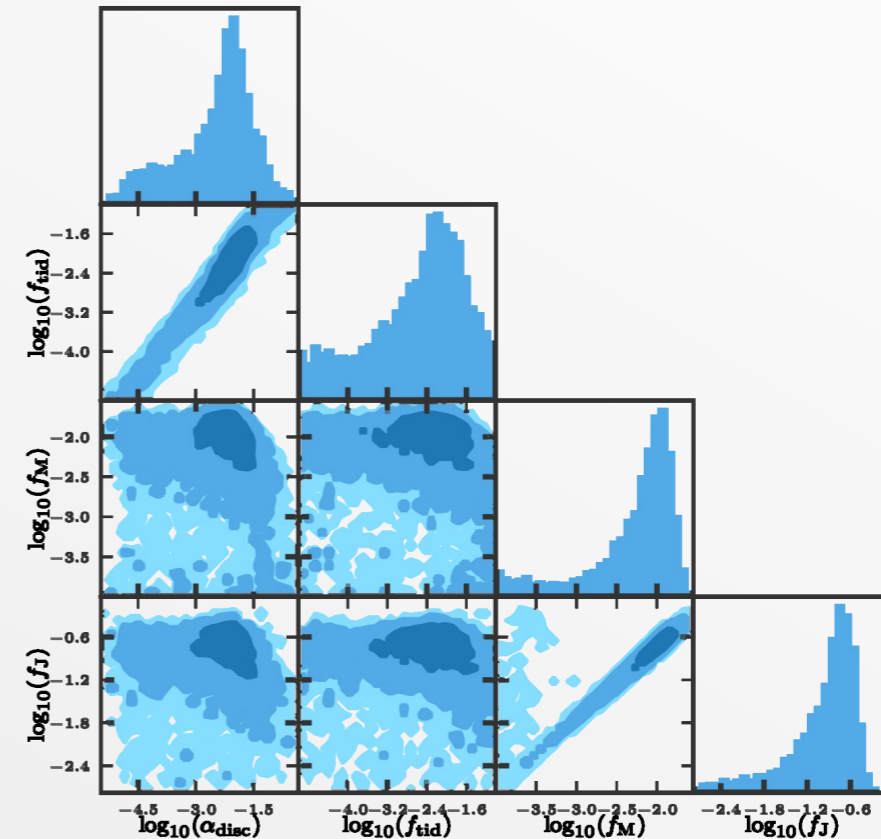
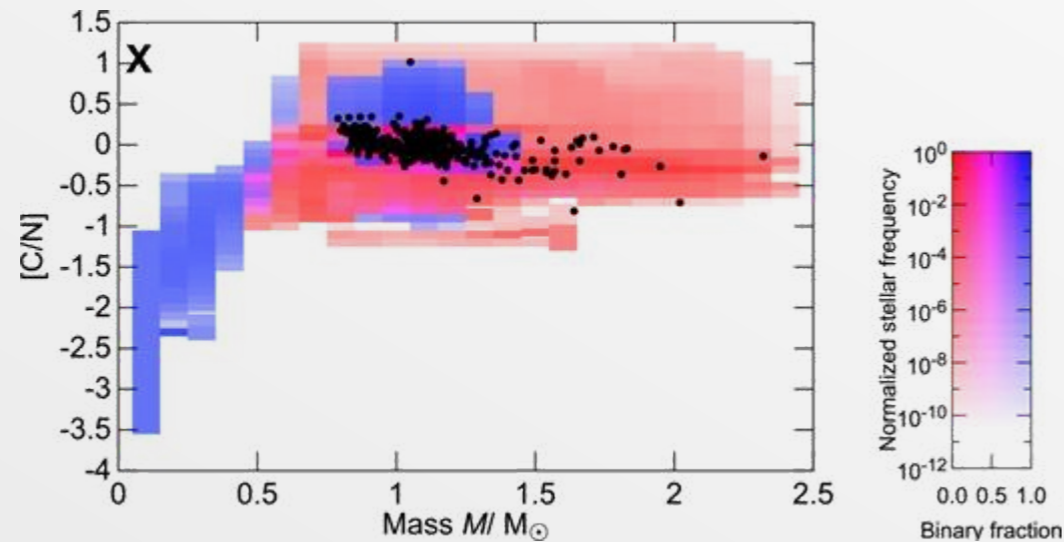
- **Massive stars: spin rates, evolutionary outcomes**  
(Sana et al. 2012, de Mink et al. 2013, 2014)
- **Mass functions with binary stars**  
(Schneider et al. 2014, 2015)
- **Mergers and R-type carbon stars**  
(Izzard et al. 2007, Xiang and Jeffery 2013)
- **Type Ia supernovae**  
(Claeys et al 2013, 2014)
- **Wind accretion and carbon-rich halo stars**  
(Izzard et al 2009, Abate et al. 2013, 2014, 2015)



# Case studies with binary\_c

- **Abundances in merged thick disc stars**

(Izzard et al. 2018)



- **Nova rates in the Milky Way and M3 I**  
(Kemp et al. 2021)
- **Circumbinary discs around post-AGB stars**  
(Izzard and Jermyn 2018, 2021)
- **Tides and the MINT library**  
(Mirouh et al. 2021)

<http://personal.ph.surrey.ac.uk/~ri0005/cgi-bin/binary5.cgi>

binary\_c frontend - Google Chrome

binary\_c frontend

Not secure | personal.ph.surrey.ac.uk/~ri0005/cgi-bin/binary5.cgi

## Binary\_c Online

A frontend to the [binary\\_c](#) code

If you use the results of binary\_c for commercial or published work, please read the [LICENCE](#) file.

Basic Orbit Explosions Binary Wind Nucleosynthesis Display Evolve

Mass 1   $M_{\odot}$

Mass 2   $M_{\odot}$

Metallicity

Stellar type 1

Stellar type 2

Rotational velocity of star 1  km/s (0=Hurley et al 2002, use 0.1 for zero)

Rotational velocity of star 2  km/s (0=Hurley et al 2002, use 0.1 for zero)

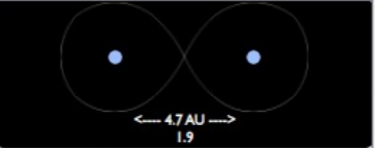
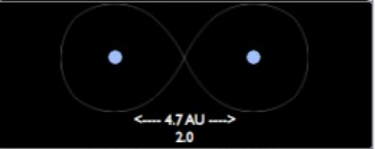
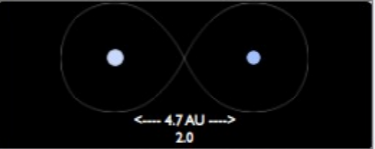
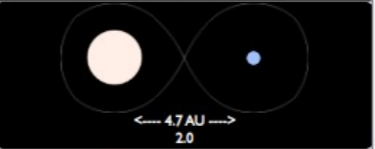

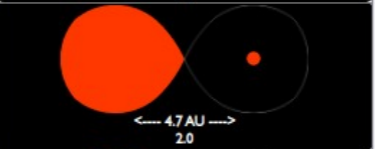
Max. Evolution Time  Myr

# <http://personal.ph.surrey.ac.uk/~ri0005/cgi-bin/binary5.cgi>

binary\_c frontend - Google Chrome

binary\_c\_frontend

Not secure | personal.ph.surrey.ac.uk/~ri0005/cgi-bin/binary5.cgi

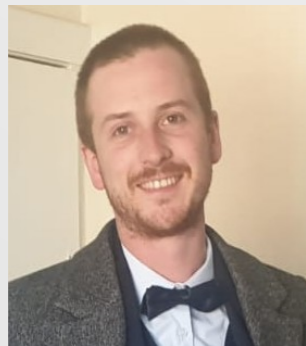
Evolution Time (Myr)	Star 1 mass ( $M_{\odot}$ )	Star 2 mass ( $M_{\odot}$ )	Star 1 type	Star 2 type	Separation ( $R_{\odot}$ )	Period	Eccentricity	Star 1 $R/R_L$	Star 2 $R/R_L$	What's happening?	
0.0000	16.000	12.000	Main Sequence	Main Sequence	1000.5	693.14	0.00	0.013	0.012	Hydrogen ignition	
11.9798	15.620	11.931	Hertzsprung Gap	Main Sequence	1016.8	715.91	0.00	0.030	0.018	Stellar Type Change	
12.0037	15.613	11.931	Core Helium Burning	Main Sequence	1017	716.21	0.00	0.222	0.018	Stellar Type Change	
12.1376	15.508	11.935	Core Helium Burning	Main Sequence	1019.9	720.60	0.00	0.717	0.018	Begin Symbiotic	
12.1492	15.488	11.938	Core Helium Burning	Main Sequence	1020	720.93	0.00	1.000	0.018	Begin Roche Lobe Overflow	
12.1690	15.447	11.944	Core Helium Burning	Main Sequence	1019.7	721.07	0.00	1.631	0.018	Start	



# *binaryc-python* module

- **Total control** over spacing/distributions
- Multi-D parameter space **handled automatically**
- loops over single/binary stars,  $M_1$ ,  $M_2$ ,  $a$ ,  $e$ , whatever!
- **Multi-CPU** or distributed over computer **cluster**

```
population.add_grid_variable(  
    name="M_1",  
    longname="Primary mass",  
    valuerange=[1,10],  
    resolution=100,  
    spacingfunc="const(1,10,100)",  
    precode="M_1=math.exp(lnm1)",  
    probdist="three_part_powerlaw(M_1, 0.1, 0.5, 1.0, 150, -1.3, -2.3, -2.3)*M_1",  
    dphasevol="dlnm1",  
    parameter_name="M_1",  
)  
population.evolve()
```



[https://gitlab.eps.surrey.ac.uk/ri0005/binary\\_c-python](https://gitlab.eps.surrey.ac.uk/ri0005/binary_c-python)

Docs at [https://ri0005.pages.surrey.ac.uk/binary\\_c-python/index.html](https://ri0005.pages.surrey.ac.uk/binary_c-python/index.html)

← **David Hendriks** main binary\_c-python developer 49

# For this school

- <http://personal.ph.surrey.ac.uk/~ri0005/Heidelberg2021/school.html>
- Installation is either FAST with **Virtualbox**
- or SLOW: you build the code **yourself**
- *but* you learn how to do it, vital skills!
  - try both? neither is hard!
- works best on Linux (e.g. Ubuntu) → simple install script
- ... but compiles on OSX and Windows (Linux subsystem)



# binary\_c-python notebooks

- In the examples/ directory

[https://gitlab.eps.surrey.ac.uk/ri0005/binary\\_c-python/-/tree/master/examples](https://gitlab.eps.surrey.ac.uk/ri0005/binary_c-python/-/tree/master/examples)

- With documentation

[https://ri0005.pages.surrey.ac.uk/binary\\_c-python/example\\_notebooks.html](https://ri0005.pages.surrey.ac.uk/binary_c-python/example_notebooks.html)

- Tutorials from David

- Running individual systems with binary\_c-python
- Using custom logging routines with binary\_c-python
- Running populations with binary\_c-python
- Extra features and functionality of binary\_c-python

- Use these to really understand how it works

→ and for vital technical know-how for you!

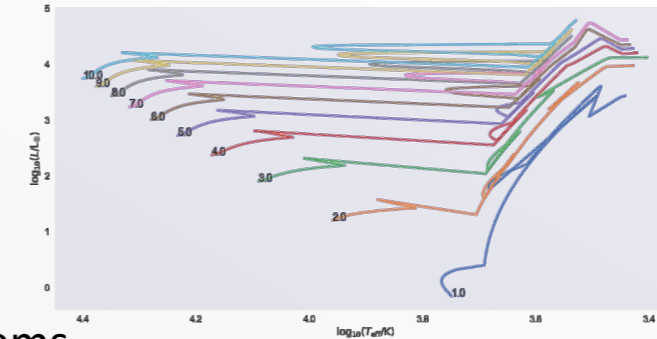
- notebook\_individual\_systems is particularly useful to explore how one system evolves and interact

# binary\_c-python notebooks

Physics examples from me:

- notebook\_HRD

make Hertzsprung-Russell diagrams of single/binary systems

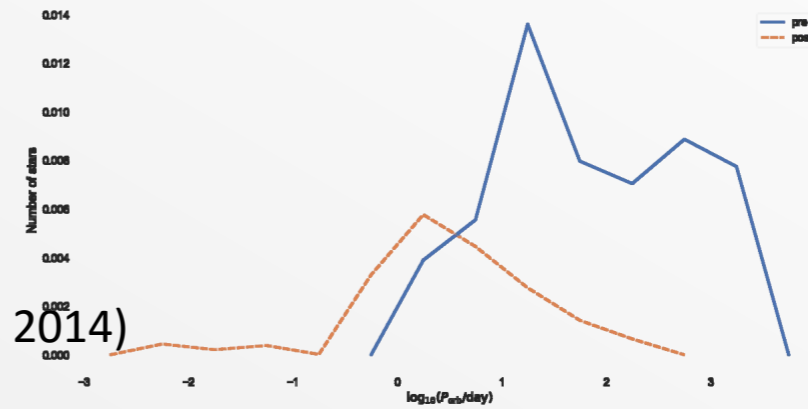


- notebook\_luminosity\_function\_single

single-star luminosity function

- notebook\_luminosity\_function\_binaries

binary-star luminosity functions (cf. Schneider+ 2014)



- notebook\_common\_envelope\_evolution

shows how the orbit changes during common envelope

- notebook\_solar\_system

put some rocks in orbit around the Sun... see what happens!

- notebook\_BHBH

BHBH merging systems - work in progress, please watch for updates

# The future of population synthesis

- Computing speed increase: use detailed models?
- Hybrids may be better: computers have a long way to go to be  $>10^6$  times faster
- Progress in statistics, but how we compare huge parameter spaces and datasets (e.g. Gaia) is still a challenge. Machine learning the way forward?  
Maybe...
- Need smart people!

