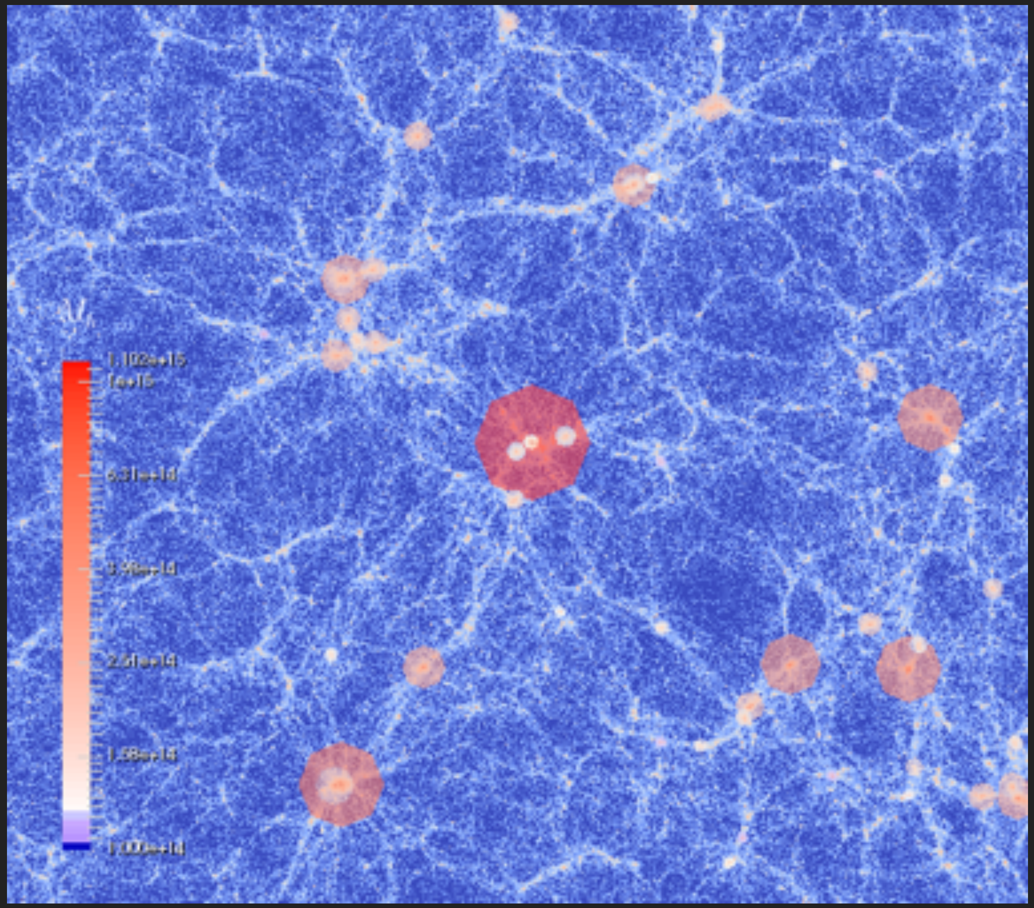
MODELING ISSUES USING N-BODY SIMULATIONS: A CASE FOR SUBARU HSC SURVEY GALAXY-GALAXY LENSING EMULATOR

TAKAHIRO NISHIMICHI (KAVLI IPMU, JST CREST)



- **Kavli IPMU**

Takahiro Nishimichi*
Masahiro Takada
Naoki Yoshida

- **U. Tokyo**

Ken Osato*
Masamune Oguri

- **NAOJ**

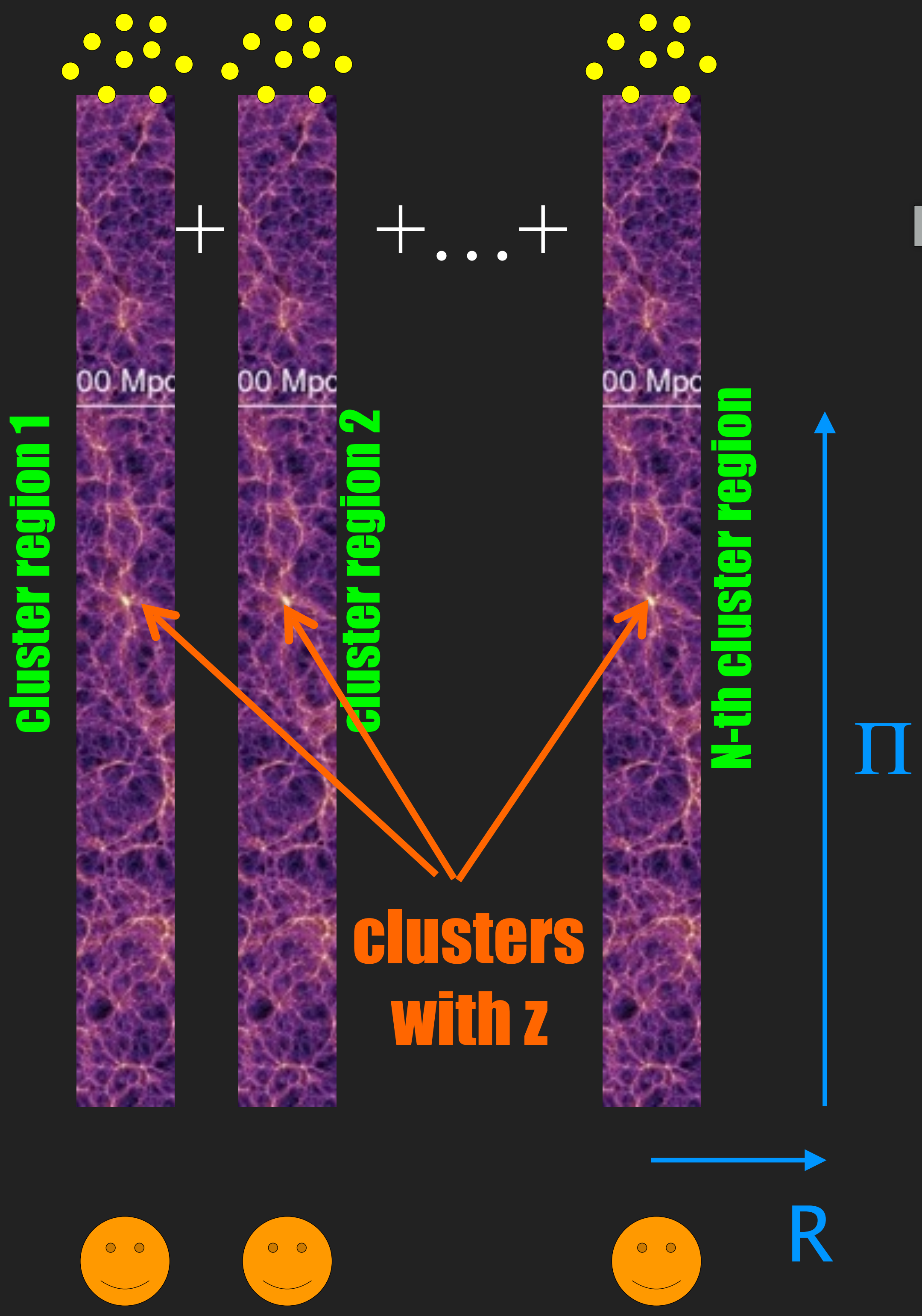
Masato Shirasaki*
Takashi Hamana

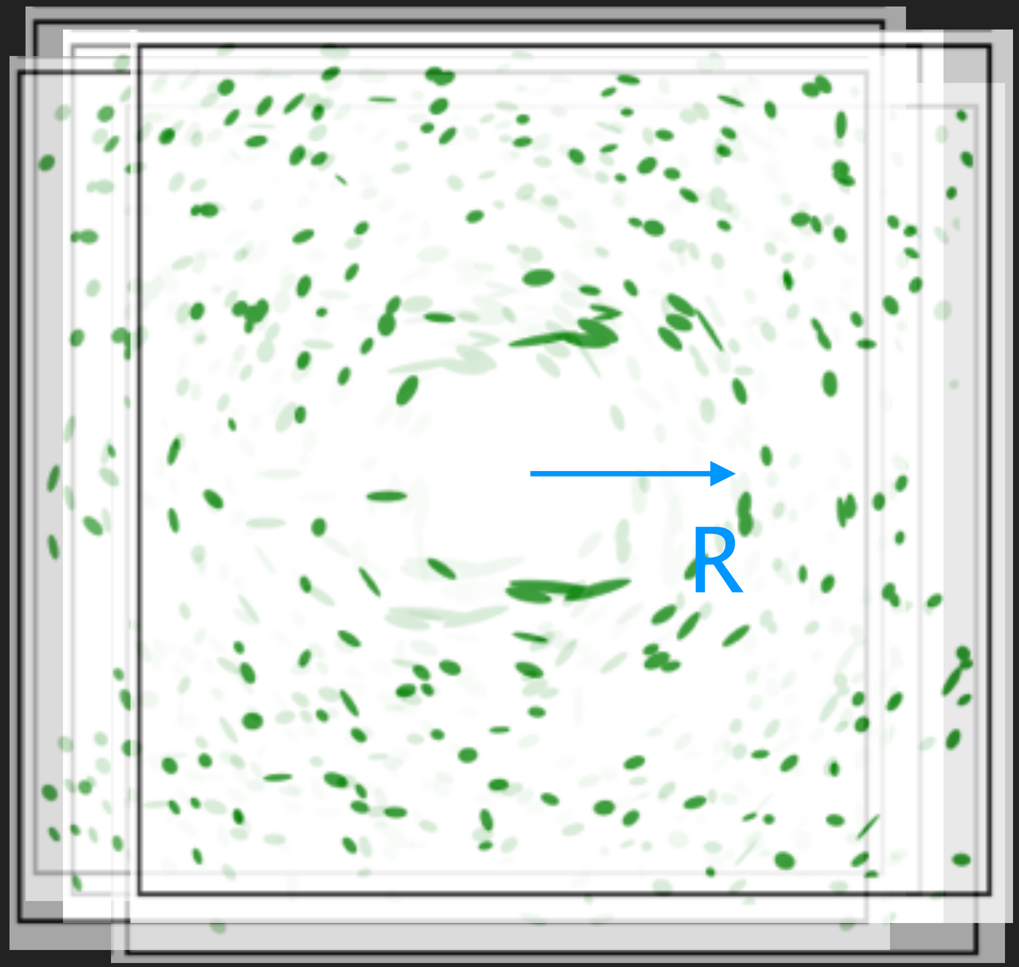
- **Hirosaki U.**

Ryuichi Takahashi*

2016/09/06 KASI; Cos-KASI-ICG-NAOC-YITP Workshop 2016

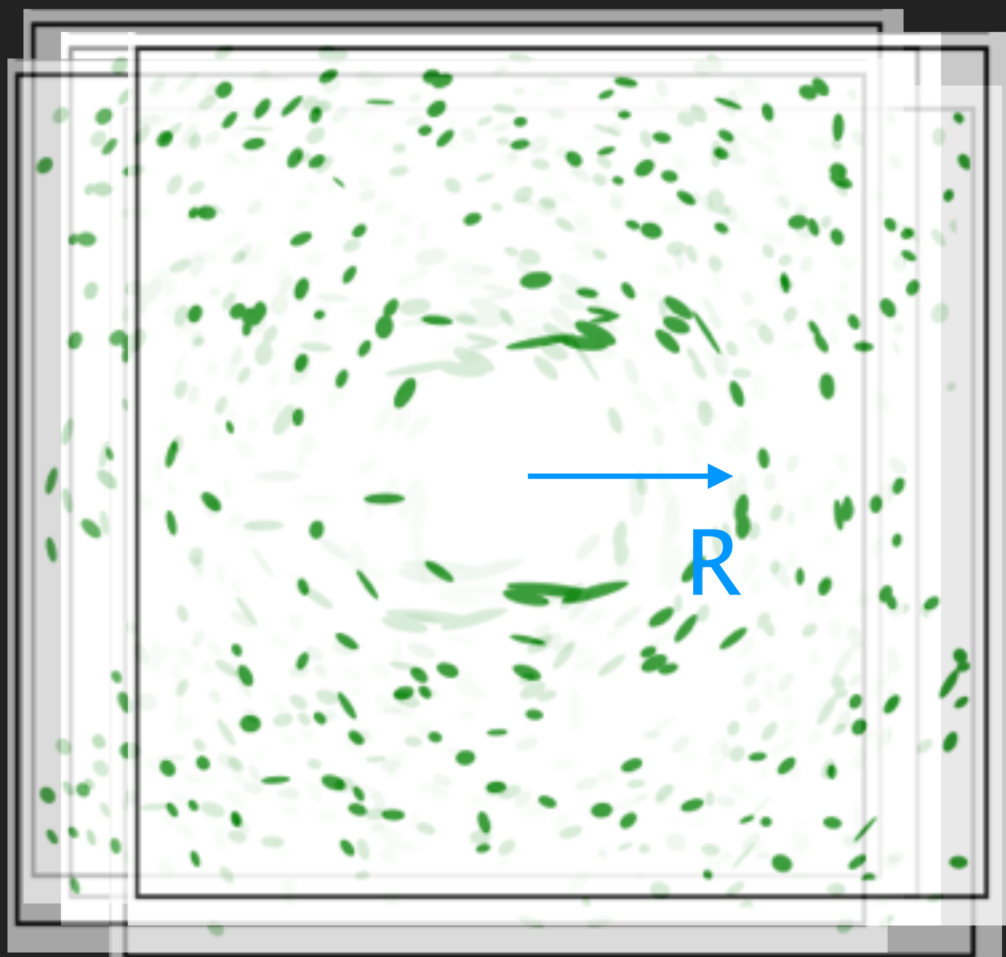
GALAXY(CLUSTER)-GALAXY LENSING

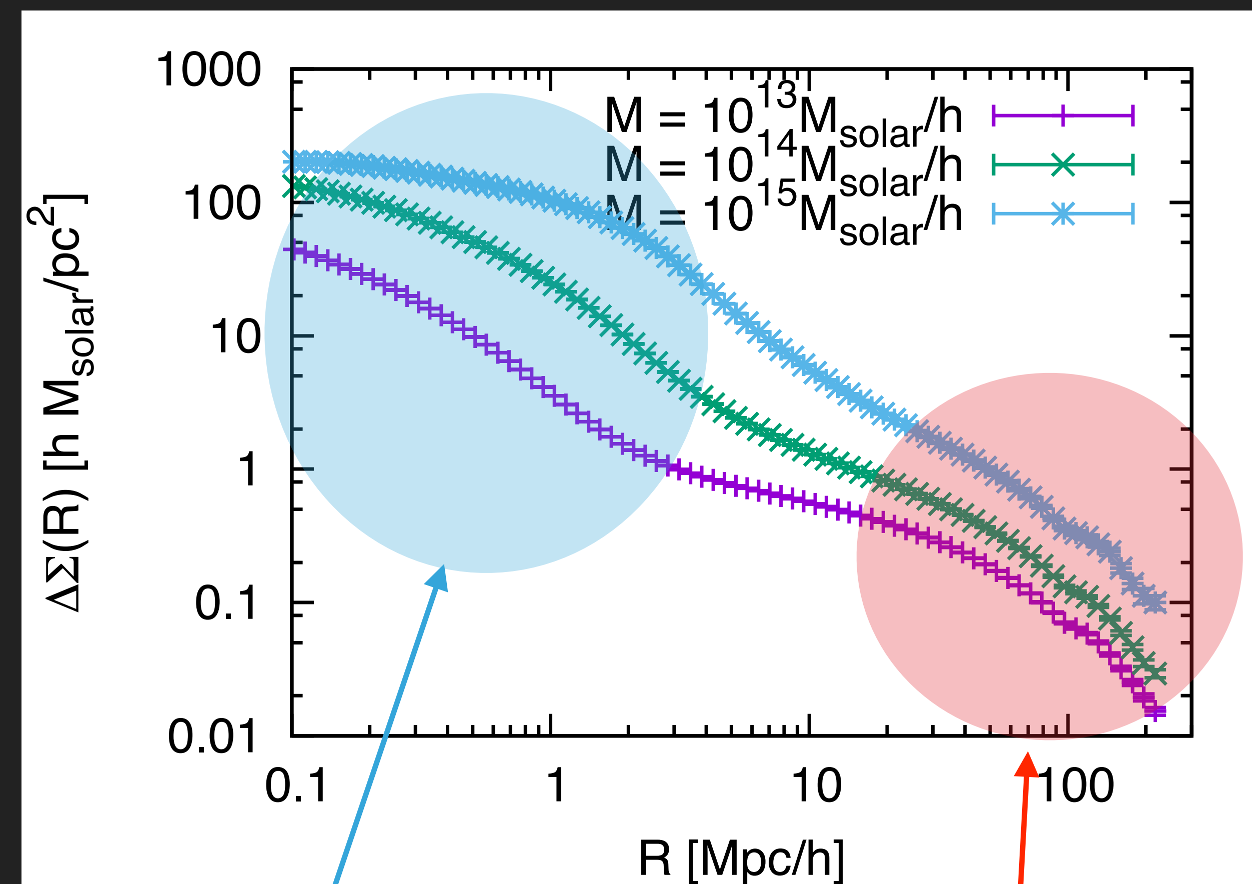




- Lens objects with known z
  - free from projection effect
  - do not have to care too much on the photo-z accuracy
  - can obtain the signal as a function of scale R instead of angle on the sky

G-G LENSING: STACKED WEAK LENSING SIGNAL





1 halo regime

2 halo regime with BAO

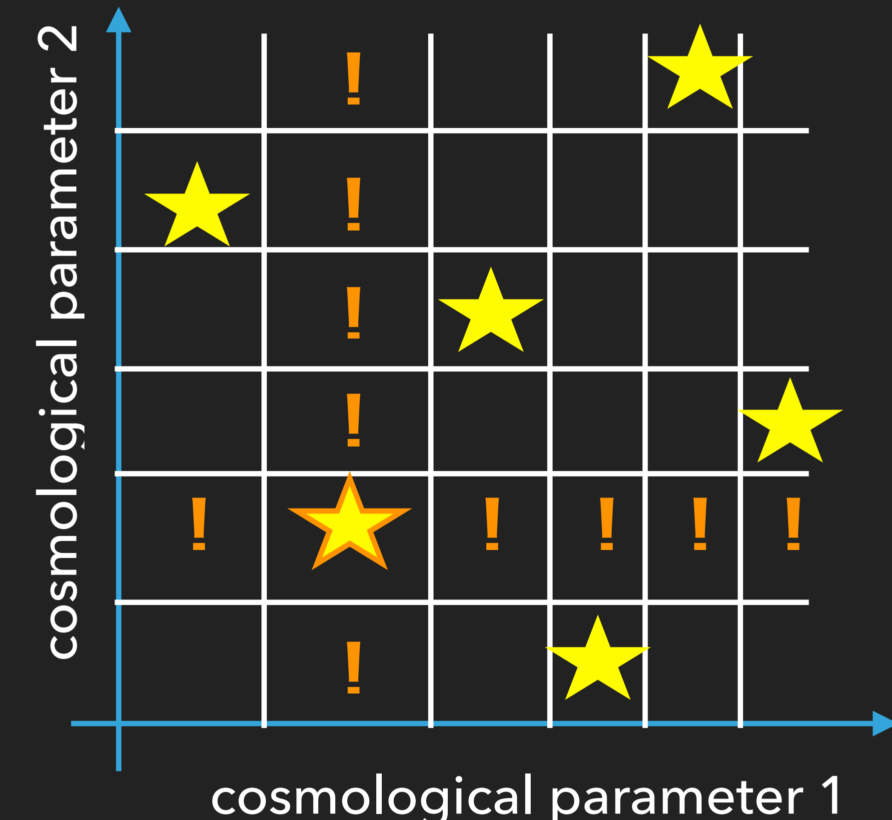
- Wide dynamic range
  - clearly PTs do not apply on small scales
  - analytical bias description still unclear
  - significant volume needed to control large scales
  - cosmological parameter dependence?

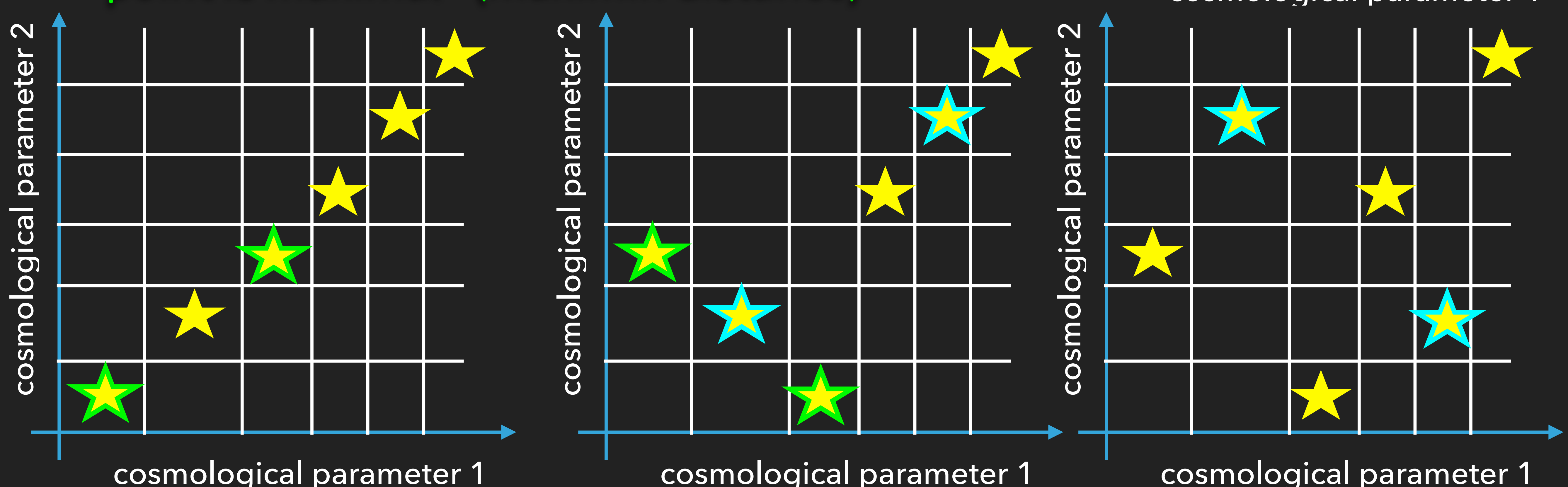
THE GOAL

- ✓ Numerical cosmology: *direct confrontation of obs and sims*
- ✓ Build a *machine* that computes the *halo mass function* and *excess surface mass density* trained by simulation data to predict ***the signal*** in python
  - ✓ covariance is modeled separately
- ✓ Input parameters (6D + α):
  - ✓ cosmological: ω_b, ω_c, Ω_Λ, A_s, n_s, w (flat wCDM) → LH design
  - ✓ + M_h, z (halo mass function)
  - ✓ + M_h, z, R (excess surface mass density)
- ✓ Note that we model *cluster-galaxy lensing signal* first, and thus
  - ✓ ignore subhalos for the moment
  - ✓ parameters to be added: satellite fraction (or HOD params), off-centering, …

EFFICIENT SAMPLING IN MULTI DIMENSIONAL SPACE: LATIN HYPERCUBE

- Each sample is the only one in each axis-aligned hyperplane containing it
  - One can find many realizations of such design (ex. diagonal design)
  - Impose additional condition such as "the sum of the distances to the nearest design point is maximal" (maximin distance)





EFFICIENT SAMPLING IN MULTI DIMENSIONAL SPACE: LATIN HYPERCUBE

fiducial model

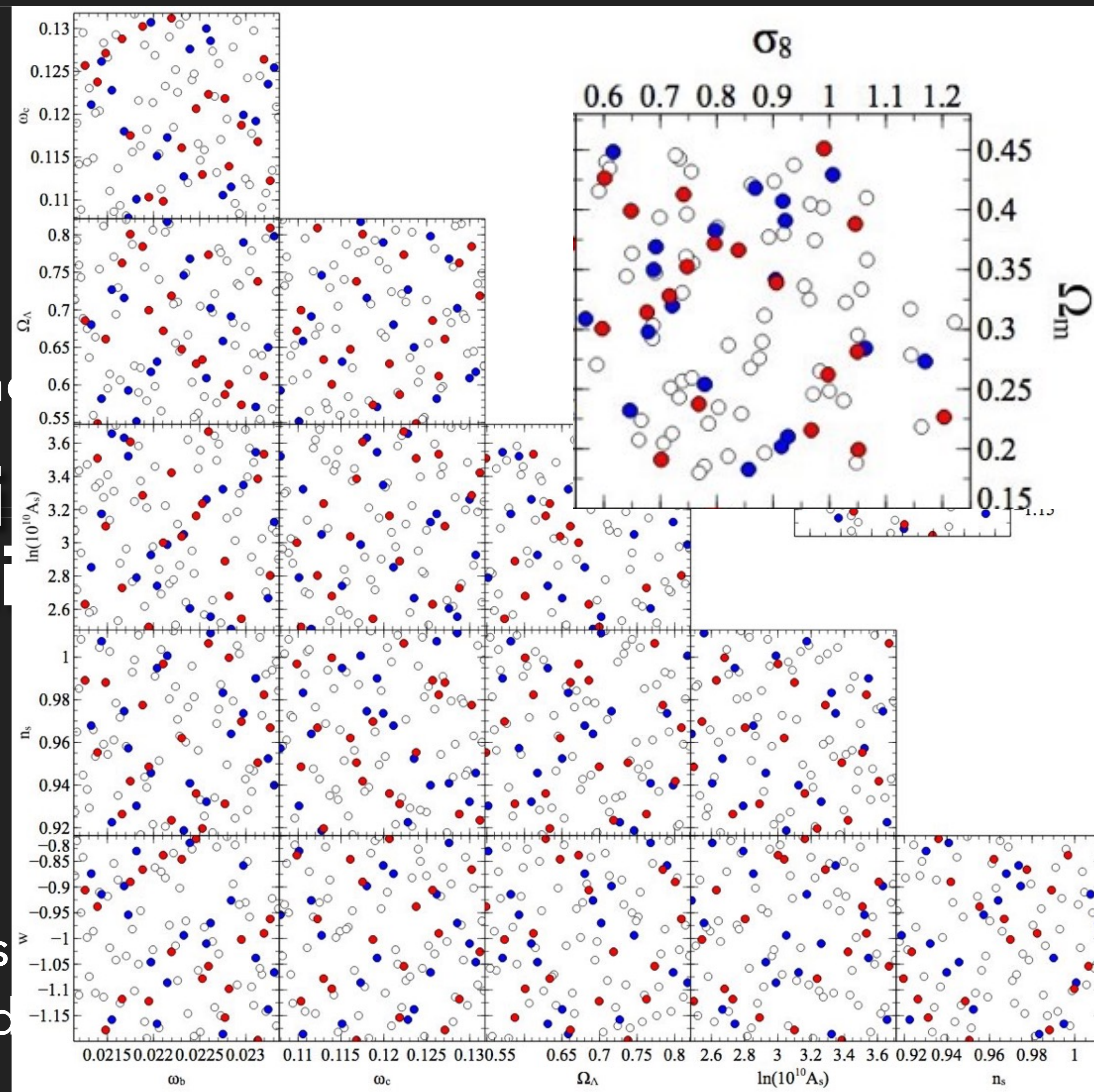
- PLANCK15 flat ΛCDM
- 24 realizations done
- assess statistical error
- check the accuracy of th
  emulator

84 si
avai

varied cosmology

- "sliced" LH design (Ba, Brenneman & Myers '15)
- generate 100 samples eventually
- maxi-min distance LH des
  every 20 models (e.g., red
  points)

e)



SIMULATION PIPELINE

~2TB, 2+2days / 1 run

on analysis servers

prediction

density estimate

final catalog

data analysis

time evolution

mass/profile determination

Gadget2

rockstar

IC generator

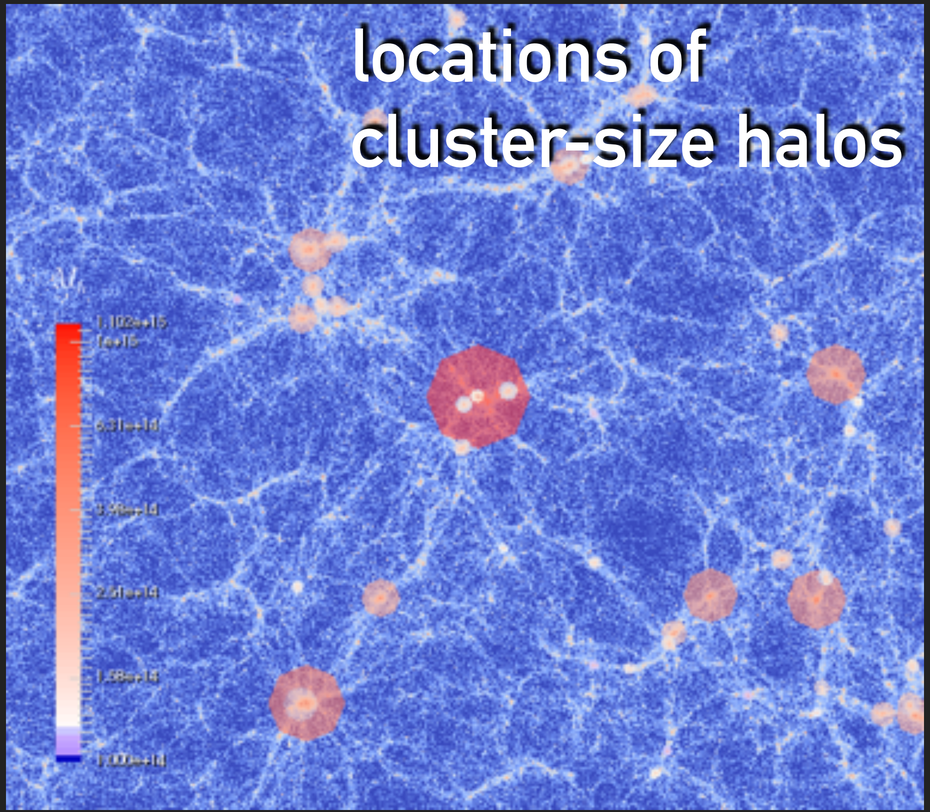
subfind

create "universes"

FOF (on the fly)

object identification

on supercomputer
(XC30 @NAOJ; using 648 CPU cores)



SIMULATION SPEC

- ✓ N of particles: 2048^3
- ✓ box size: $1h^{-1}$Gpc

  resolve a $10^{12}\ h^{-1}M_{solar}$ halo with ~100 particles

- ✓ 2nd-order Lagrangian PT initial condition @ $z_{in}=59$

  (vary slightly for different cosmologies to keep the RMS displacement about 25% of the inter-particle separation)

- ✓ Tree-PM force by L-Gadget2

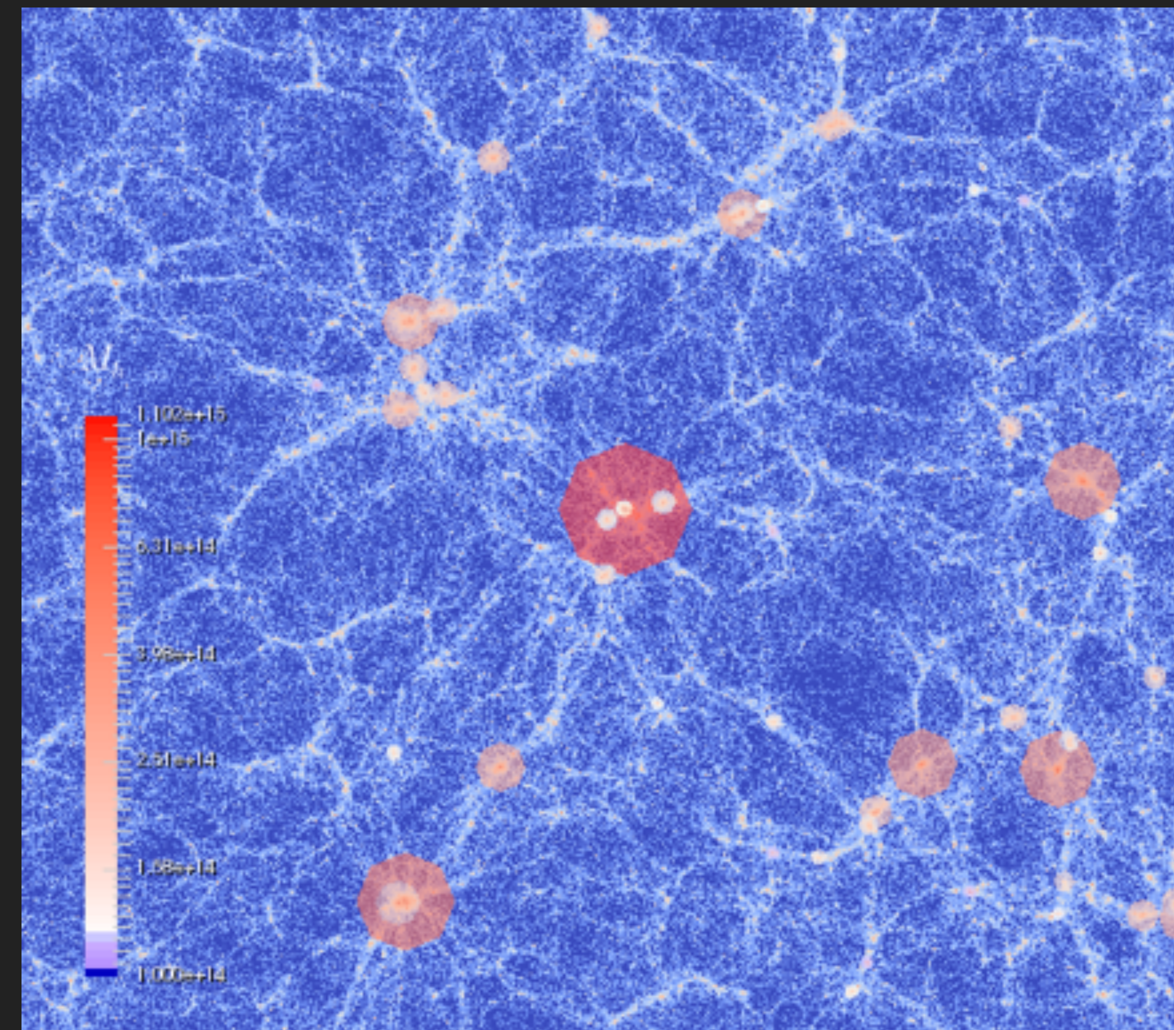
  (w/ 4096^3 PM mesh)

- ✓ 21 outputs in $0 \leqq z \leqq 1.5$

  (equispaced in linear growth factor)

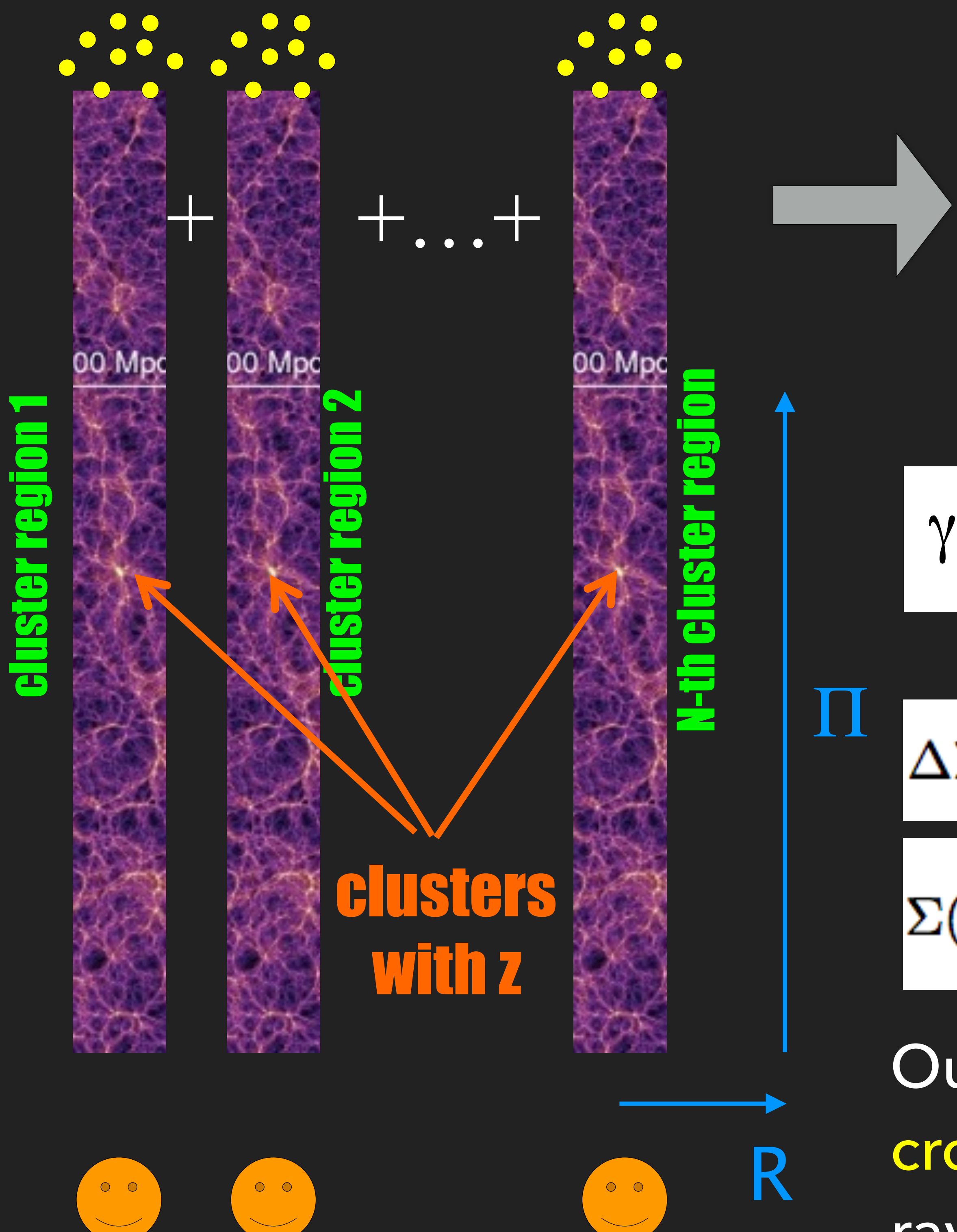
- ✓ Data compression (256GB -> 48GB par snapshot)
  - ✓ positions -> displacement (16 bits par dimension; accuracy ~$1h^{-1}$kpc)
  - ✓ velocity: discard after halo identification
  - ✓ ID: rearrange the order of particles by ID and then discard
  - ✓ already consuming ~200TB in half a year (~observational data)

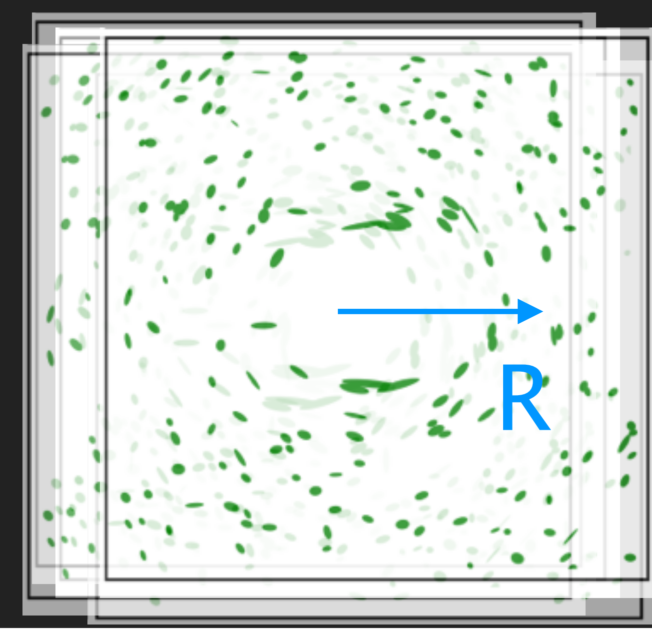
OUR HALO CATALOGS



- ✓ Halo finder
  - ✓ FOF + Subfind
  - ✓ Rockstar (+ merger tree by consistent-trees)
- ✓ Grow sphere centered at
  - ✓ core center (rockstar)
  - ✓ most bound particle (subfind)
- ✓ until the interior density reaches 200ρ_m to determine M_200m
  - ✓ measure and store the density profile at the same time
- ✓ Exclude “satellites” when the center is within r_200m
- ✓ density profile is stored for each halo up to 5Mpc/h
  - ✓ g-g lens signal is obtained immediately once the mass bin, weight and off-centering profile are given

G-G LENSING: STACKED WEAK LENSING SIGNAL





Oguri & Takada '11

$$\gamma_+^{\mathrm{obs}}(\theta_i) = \gamma_+^{\mathrm{cluster}}(\theta_i) + \underline{\gamma_+^{\mathrm{LSS}}(\theta_i)} + \varepsilon_+(\theta_i)$$

projection effect

$$\Delta\Sigma(R) = \gamma_t(R)\Sigma_c = \overline{\Sigma}(<R) - \Sigma(R),$$

$$\Sigma(R) = \overline{\rho} \int \left[1 + \xi_{\mathrm{gm}}\left(\sqrt{R^2 + \Pi^2}\right)\right] d\Pi,$$

Our target statistics is basically the halo-matter cross-correlation function; we do not perform ray-tracing simulations

MEASURING G-G LENSING SIGNAL

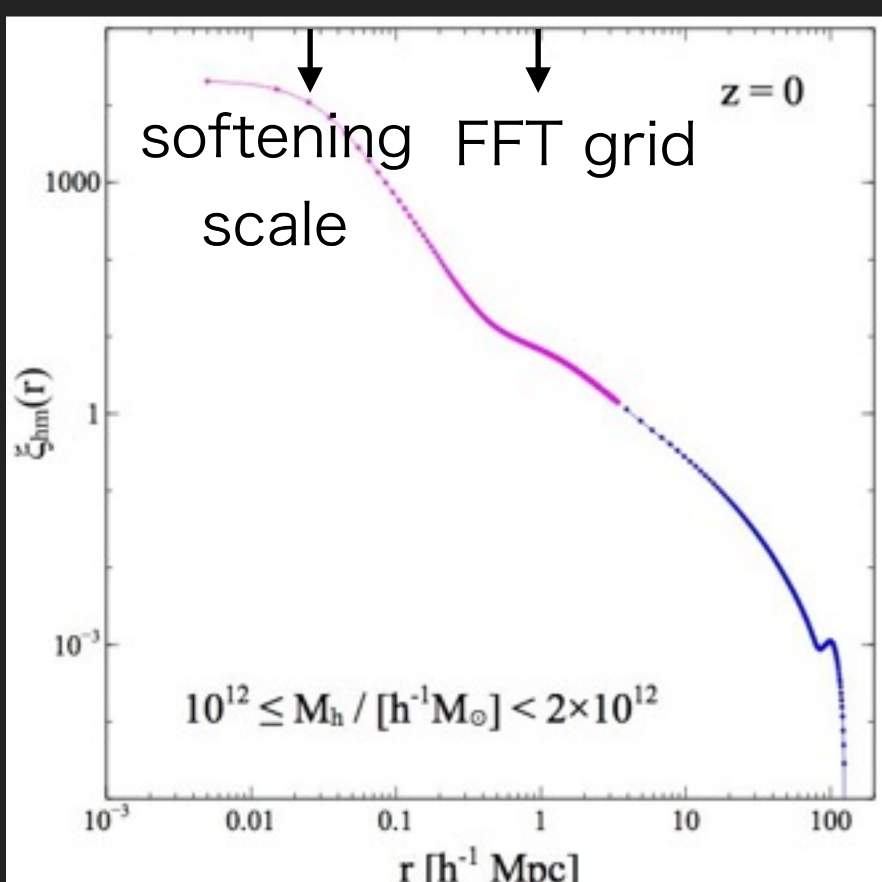
- Work in 3D
- And then projection onto 2D
- Better statistics than working in 3D from the beginning
- Hybrid Fourier-direct scheme

Measure 3D cross spectrum in Fourier space

$P_{\rm hm}(\vec{k})$ (on 1024^3 mesh by FFT)

Inverse FFT to real space and take the spherical average

$$\xi_{\rm hm}(\vec{r}) \xrightarrow[\text{spherical avg.}]{} \xi_{\rm hm}(r)$$



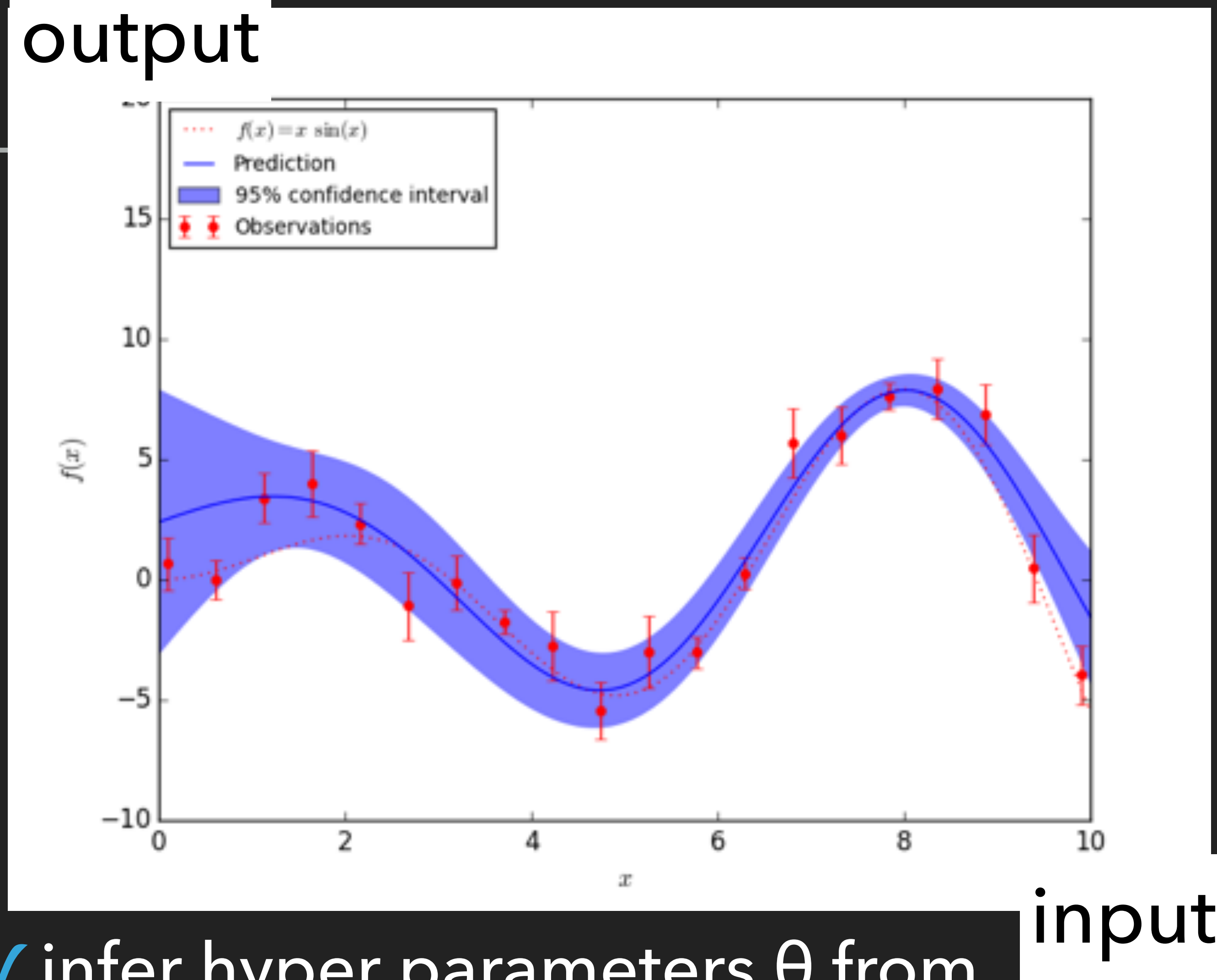
$\xi_{\rm hm}$ on small scale from direct pair count

Finally project onto 2D to have $\Sigma(R)$ and then $\Delta\Sigma(R)$

$$\Sigma(R) = \overline{\rho} \int \left[1 + \xi_{\rm gm}\left(\sqrt{R^2 + \Pi^2}\right)\right] {\rm d}\Pi$$

Easier accuracy control than working in 2D, and quicker than direct pair counting alone

GAUSSIAN PROCESS



- A kind of machine learning that interpolates in function space
  - non-parametric Bayesian inference
  - quick in high dimension input space
- Basic quantities (c.f., normal distribution)
  - mean function (cf. mean)
  - covariance function (cf. variance)
- mean can be anything, set zero usually
- covariance function is characterized by a simple function with several hyper parameters

ex. $$C(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta}) = \theta_1 \exp\left[-\frac{1}{2}\sum_{i=1}^{I}\frac{(x_i - x_i')^2}{r_i^2}\right] + \theta_2.$$

✓ infer hyper parameters θ from training data (x_i, t_i)

✓ Given any input x_{N+1}, infer t_{N+1} from θ and (x_i, t_i)

$$P(t_{N+1} \,|\, \mathbf{t}_N) \propto \exp\left[-\frac{1}{2}\begin{bmatrix}\mathbf{t}_N & t_{N+1}\end{bmatrix}\mathbf{C}_{N+1}^{-1}\begin{bmatrix}\mathbf{t}_N \\ t_{N+1}\end{bmatrix}\right].$$

answer: $$\begin{aligned}\hat{t}_{N+1} &= \mathbf{k}^T\mathbf{C}_N^{-1}\mathbf{t}_N \\ \sigma_{\hat{t}_{N+1}}^2 &= \kappa - \mathbf{k}^T\mathbf{C}_N^{-1}\mathbf{k}.\end{aligned}$$

OUR HMF EMULATOR

- fit by the well-established functional form:

$$f(\sigma) = A[\sigma^{-a} + b] \exp\left[-\frac{c}{\sigma^2}\right]$$

  - sim data is noisy at the high-mass tail
  - resolution issue at low-mass → down weight

- Construct Gaussian Processes for A, a, and c for each z that interpolate in the 6D cosmological parameter space

- Convert mass to σ using classy

- For a given cosmology, z and M dependence is interpolated by spline

OUR HMF EMULATOR: HOW DOES IT WORK

jupyter hmf_demo Last Checkpoint: Last Wednesday at 5:03 PM (autosaved)

File Edit View Insert Cell Kernel Help

Code CellToolbar

In [101]:
```
cparam = np.array([[0.02225,0.1198,0.6844,3.094,0.9645,-1]]) # planck 2015
set_cosmo(cparam)
set_MF()
```

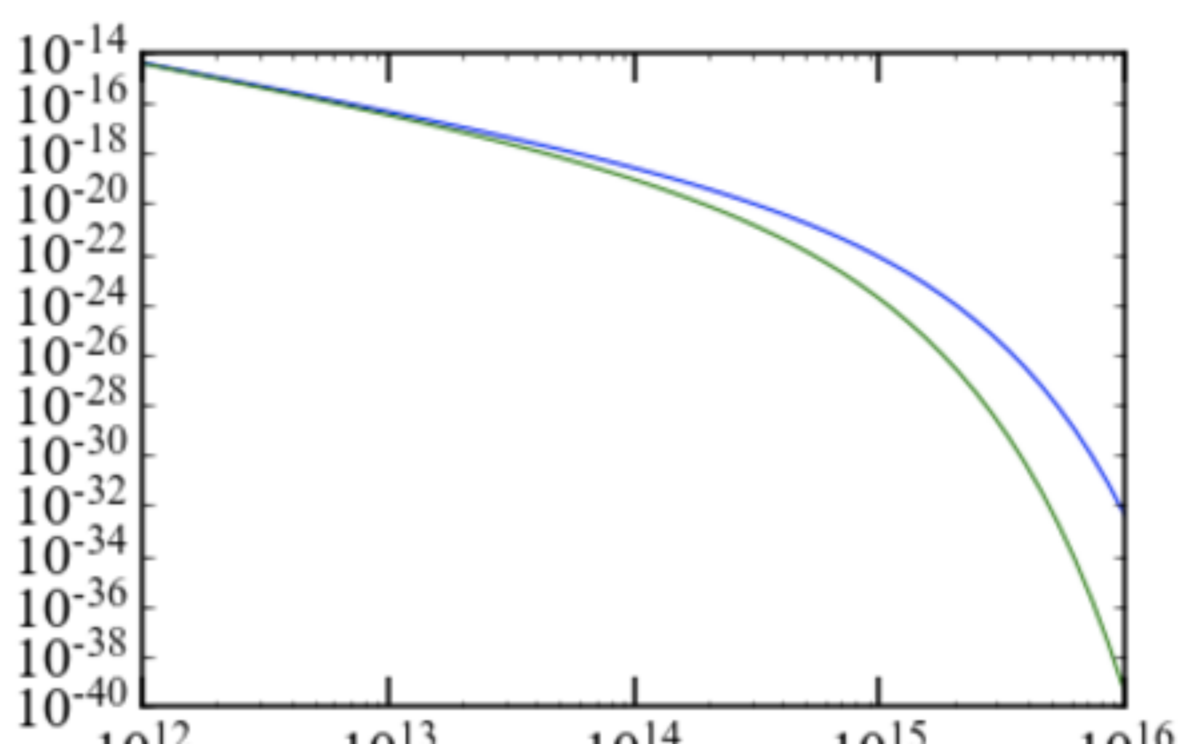
~10s

~25ms

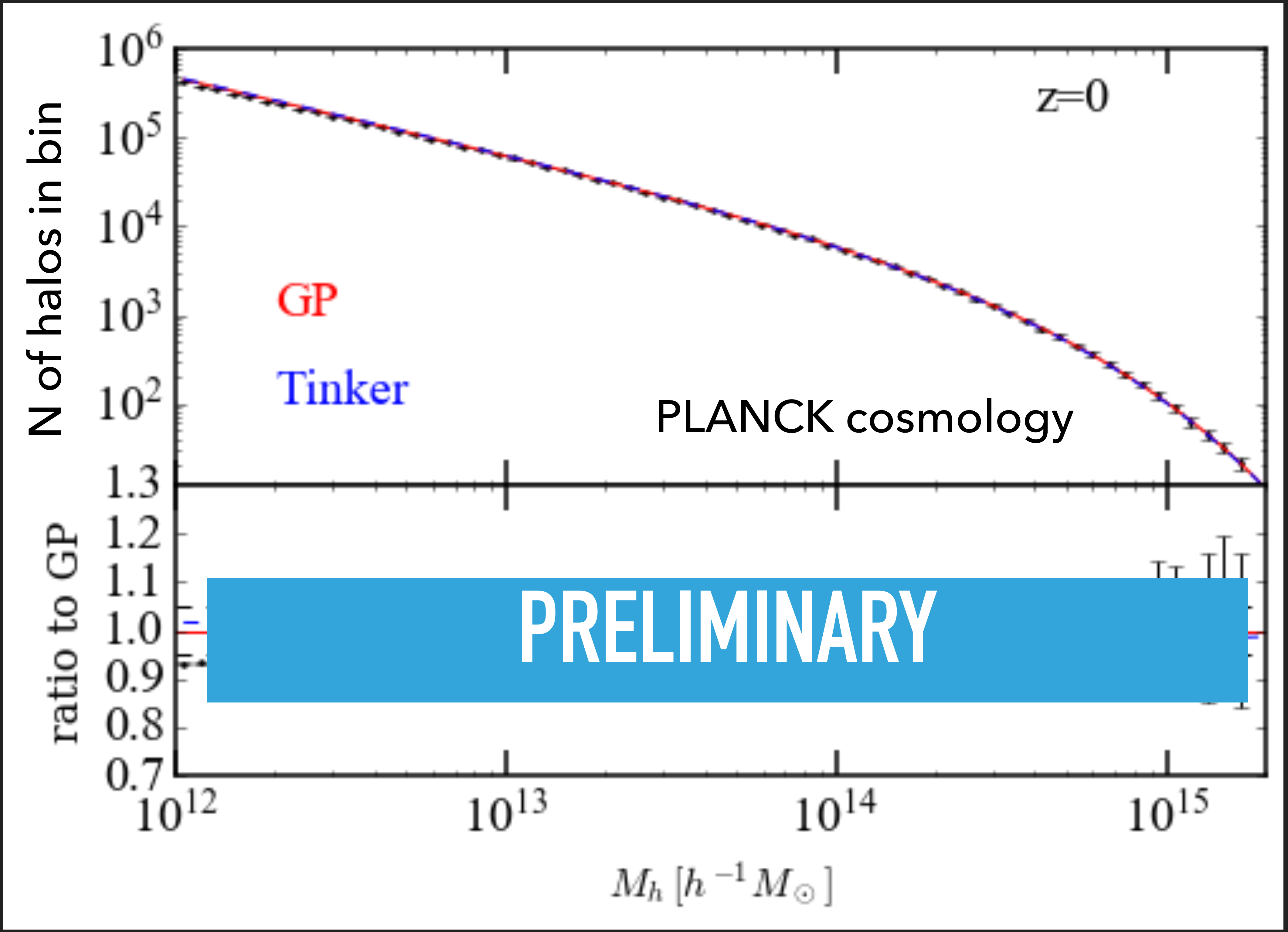
In [109]:
```
logMh=np.linspace(12,16,60)
z = 0.5
plt.loglog(10**logMh,10**f_interp(logMh,z))
z = 1
plt.loglog(10**logMh,10**f_interp(logMh,z))
```

~300μs

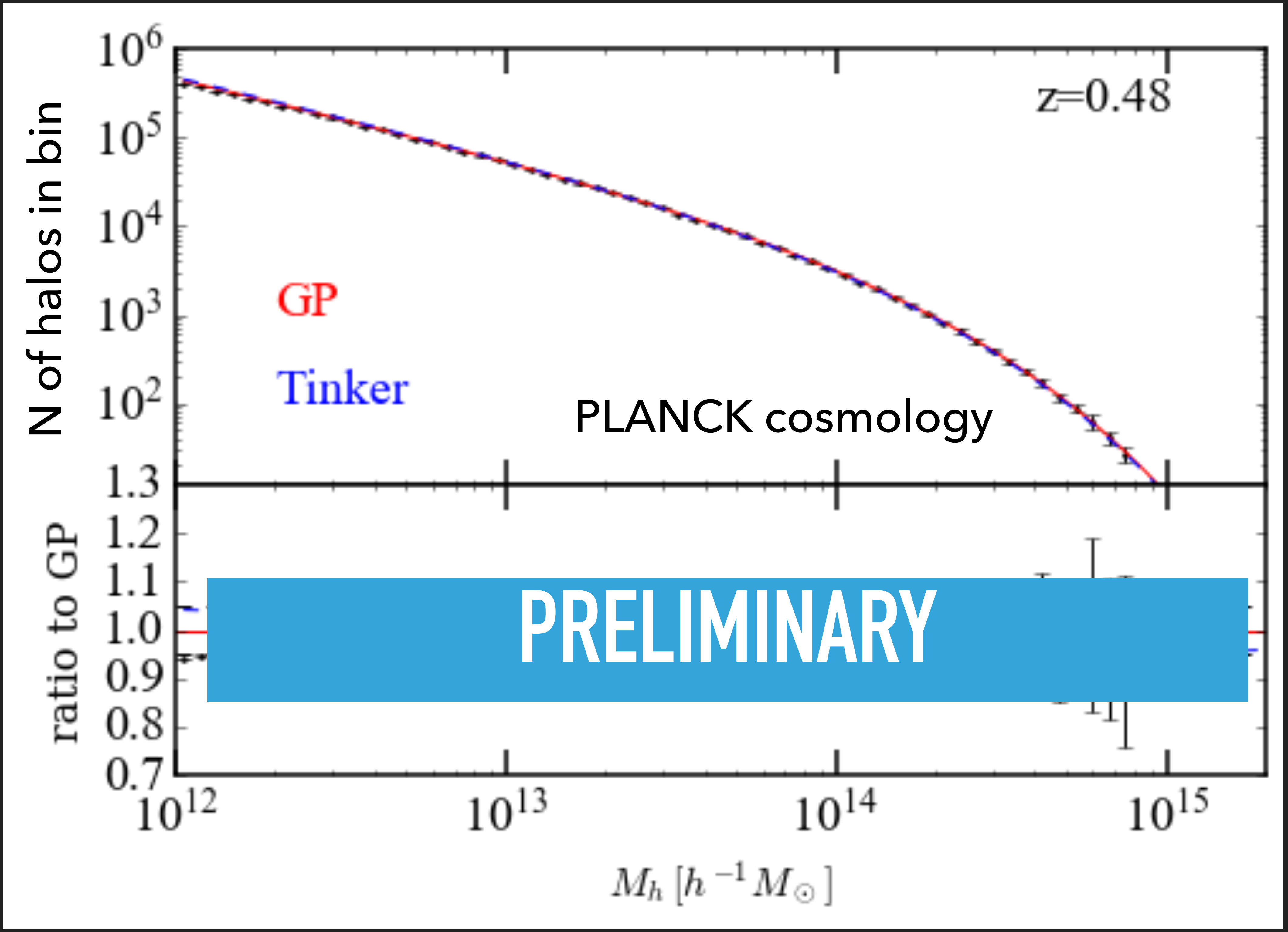
Out[109]: [<matplotlib.lines.Line2D at 0x130c11390>]



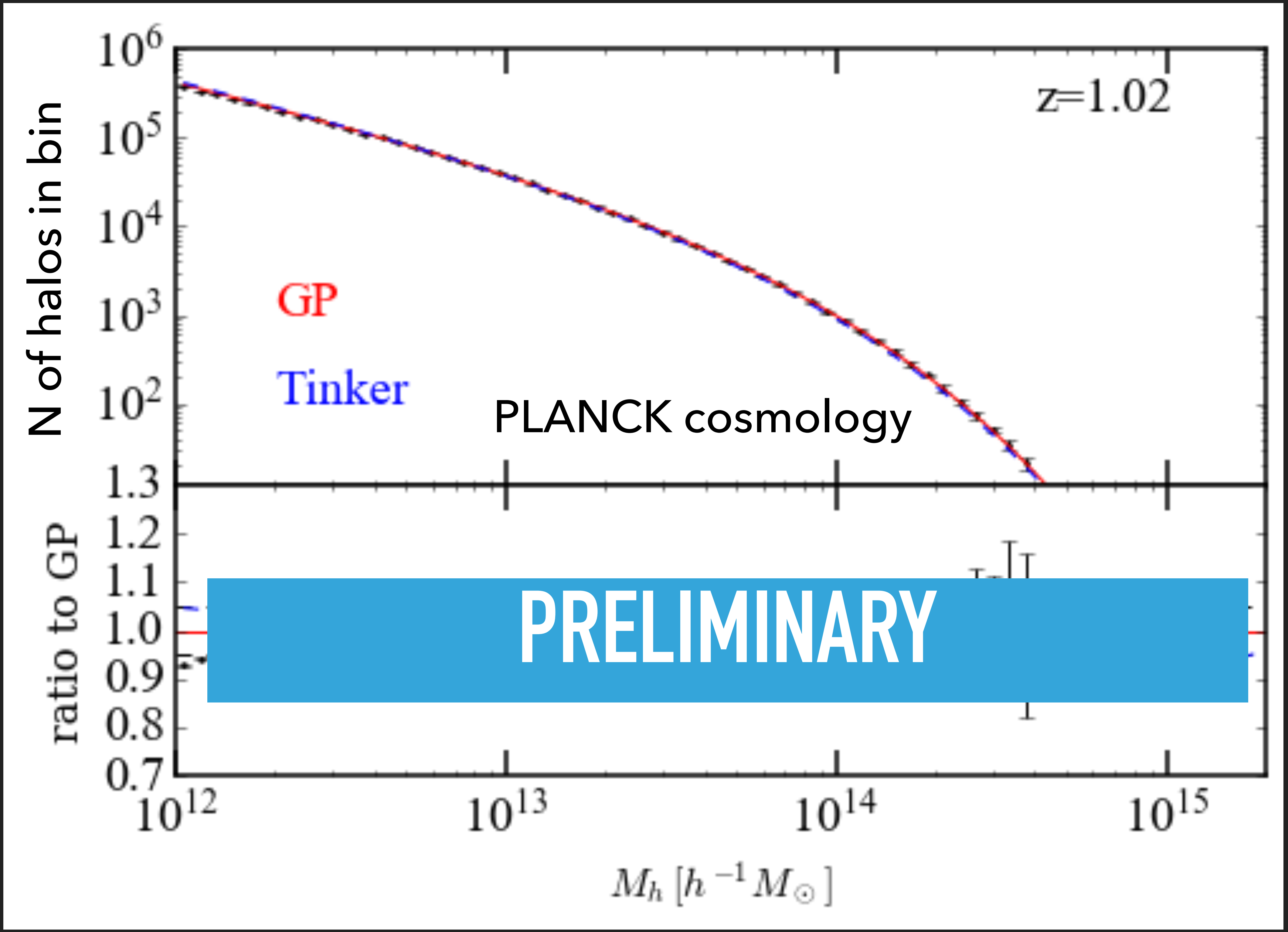
OUR HMF EMULATOR PERFORMANCE



OUR HMF EMULATOR PERFORMANCE



OUR HMF EMULATOR PERFORMANCE



OUR ΔΣ EMULATOR

- construct *mass limit samples* and measure ΔΣ(R)
- at each (z, n_h) decompose ΔΣ(R) by PCA
  - use n_h, instead of M_min to have similar noise level
  - the first 10 PCs are sufficient
- Construct Gaussian Processes for each of the 10 PC coefficients for each (z, n_h) that interpolate in the 6D cosmological parameter space
- Prediction
  - GP evaluates the 10 PCs for every (z, n_h) for a given cosmology
  - Revert them to ΔΣ(R)
  - Finally, (z, n_h, R) dependence is interpolated by 3D spline
- Use HMF GP to convert M_min to n_h

jupyter dsigma_demo Last Checkpoint: 3 minutes ago (autosaved)

File Edit View Insert Cell Kernel Help | Python 2

Code CellToolbar

Read out pre-computed PCA basis function + GP

In [3]:
```
basis = [[joblib.load('basis/S%03d_dens%02d_basis.pkl' % (snap,ndens)) for ndens in range(0,10)] for snap in range(0,21)]
dsig_gp = [[joblib.load('gp/S%03d_dens%02d_gp.pkl' % (snap,ndens)) for ndens in range(0,10)] for snap in range(0,21)]
```

Prepare a table for 3D spline

In [96]:
```
def prep_dsigma_template(params):
    global d_gp
    d_gp = np.zeros((21,10,81))
    for snap in range(0,21):
        for ndens in range(0,10):
            d_gp_pca=dsig_gp[snap][ndens].predict(params)
            d_gp[snap,ndens,] = basis[snap][ndens].inverse_transform(d_gp_pca)[0,]
```

Inputs:

- scale factor
- number density (or halo mass)
- projected distance

In [59]:
```
z=0.25
ascale = 1./(1+z)
Rplot = np.logspace(-1,2.3,100) # in h^{-1}Mpc
Mmin = 1e14 # in h^{-1}M_\odot
```

```
# PLANCK 2015
cparam = np.array([[0.02225,0.1198,0.6844,3.094,0.9645,-1]])
set_cosmo(cparam)
set_redshift(z)
lognh = mh_to_logdens(Mmin)
plt.loglog(Rplot,get_dsigma(ascale, lognh, Rplot),lw=2,color='red')
```

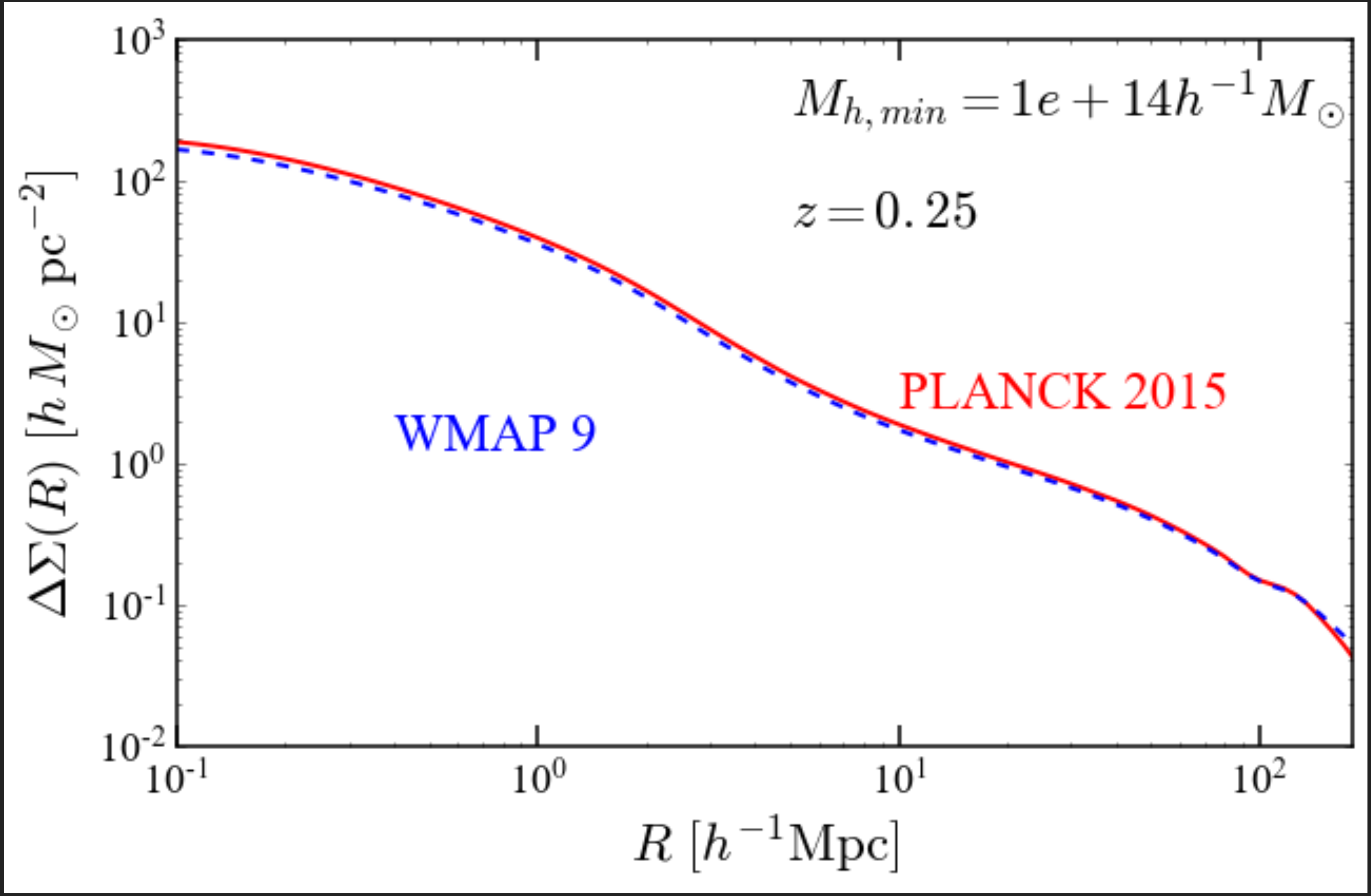
give your cosmological params ~5s

and redshifts ~600ms; HMF GP called inside

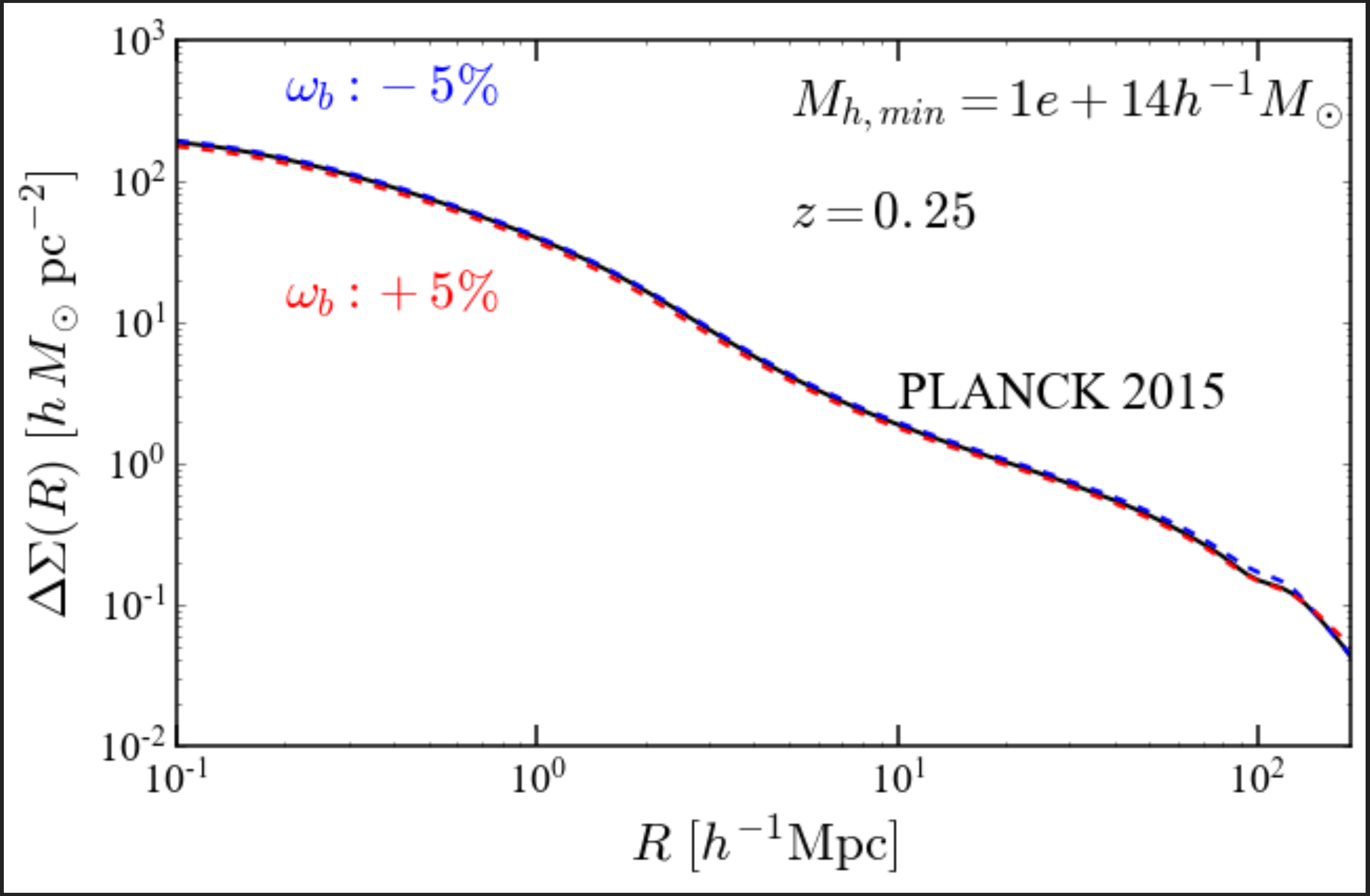
convert M_min to n_h ~50μs

Evaluate !! ~1ms

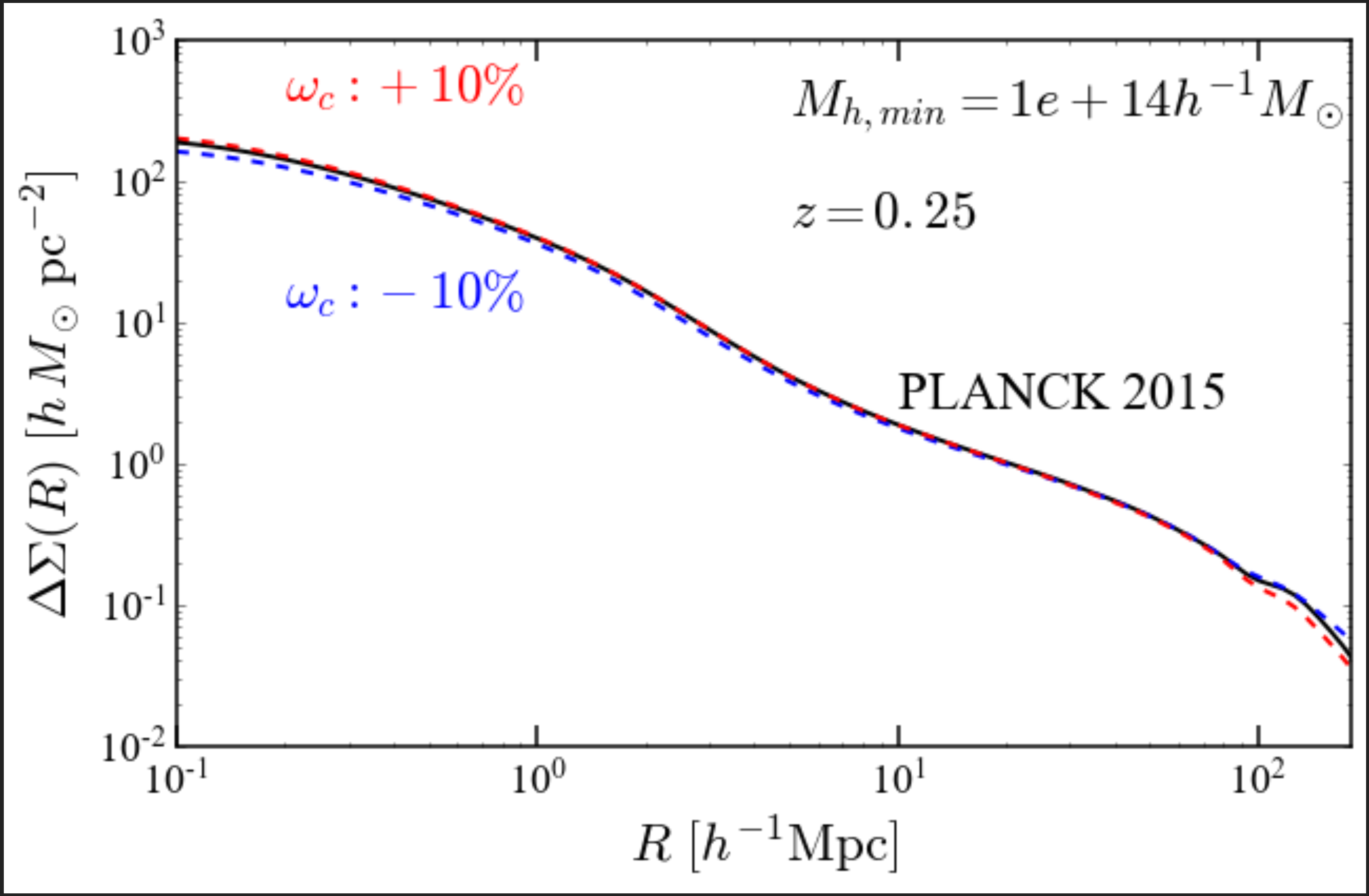
OUR ΔΣ EMULATOR DEMONSTRATIONS



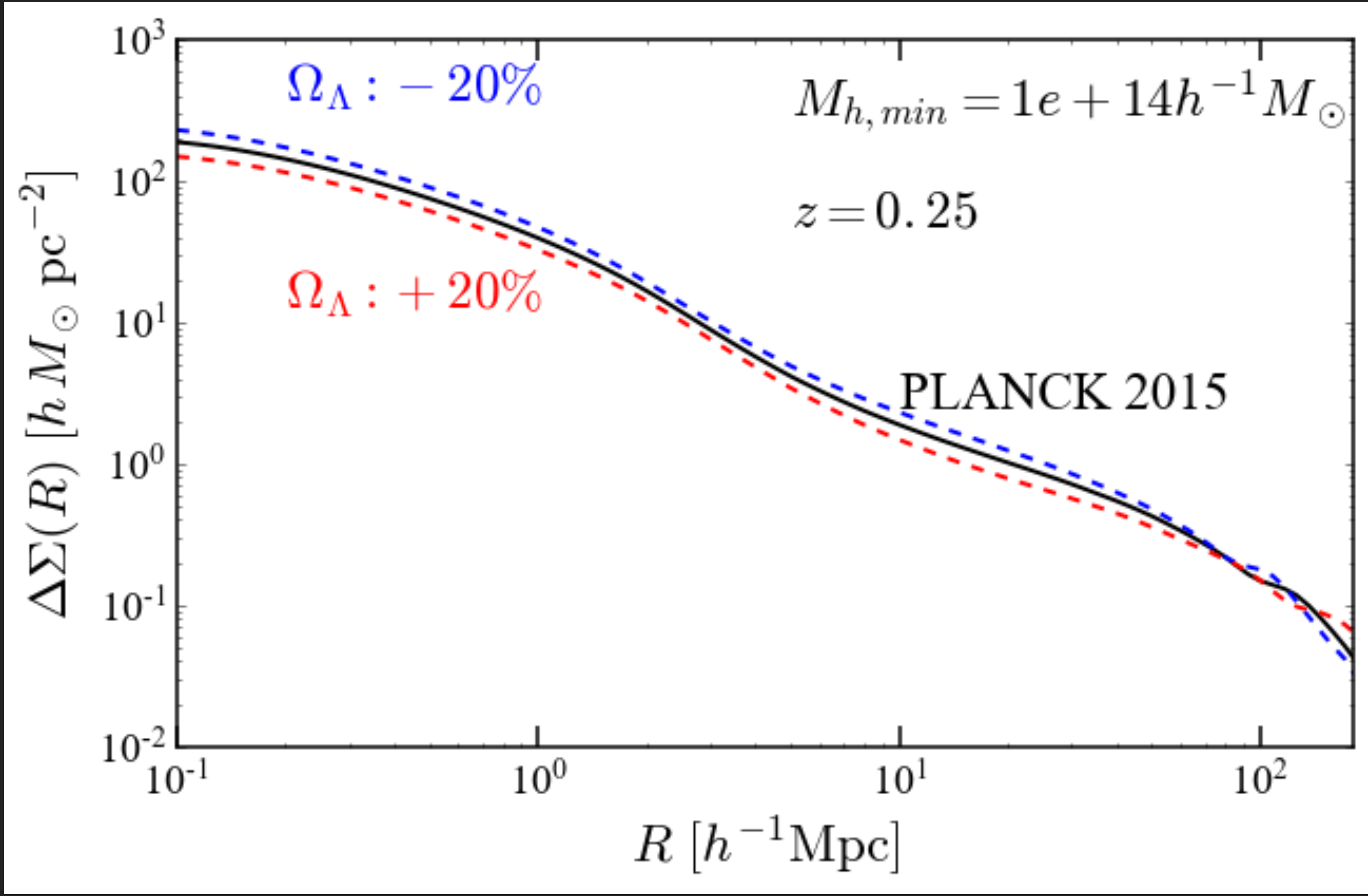
OUR ΔΣ EMULATOR DEMONSTRATIONS



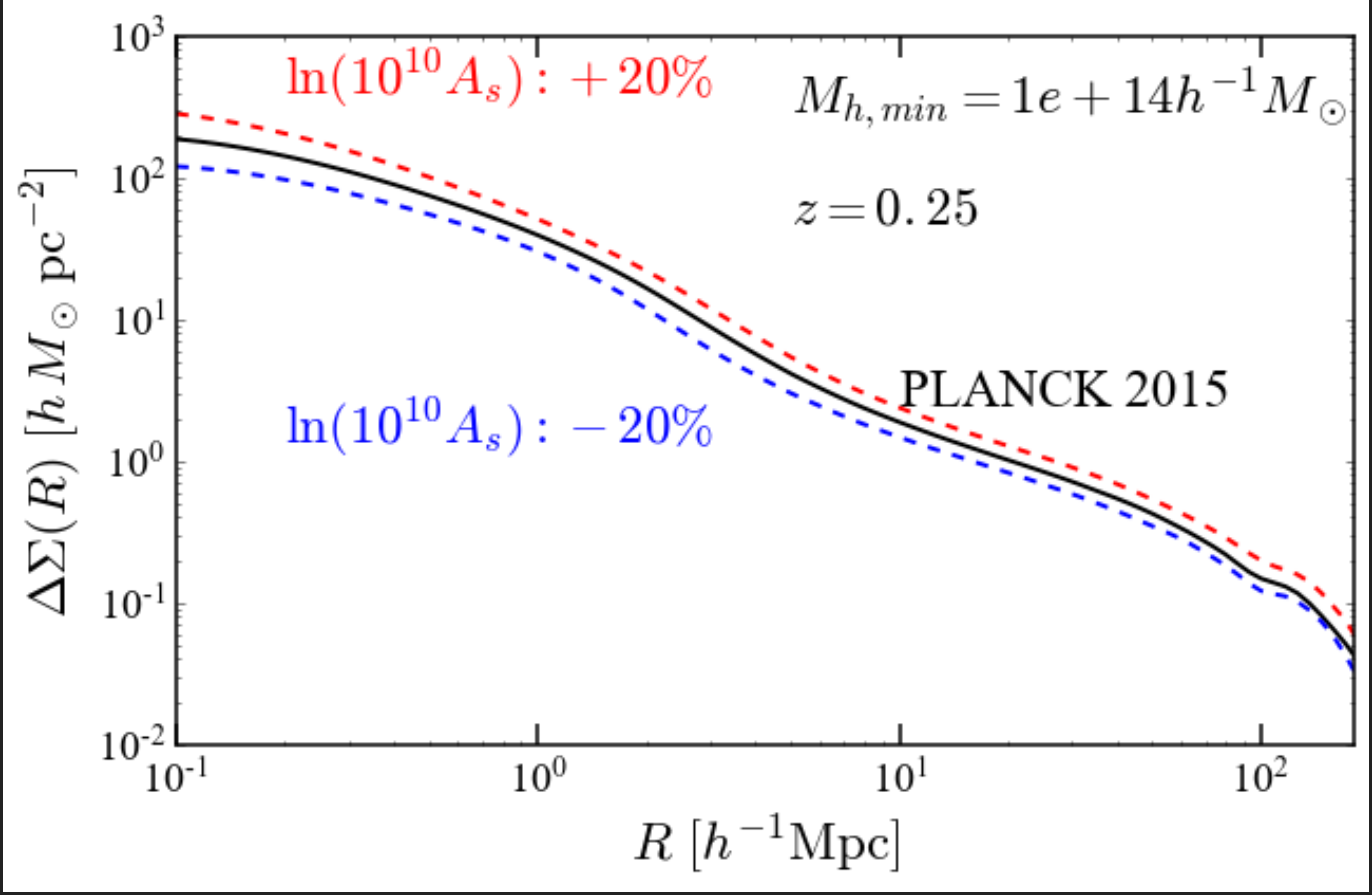
OUR ΔΣ EMULATOR DEMONSTRATIONS



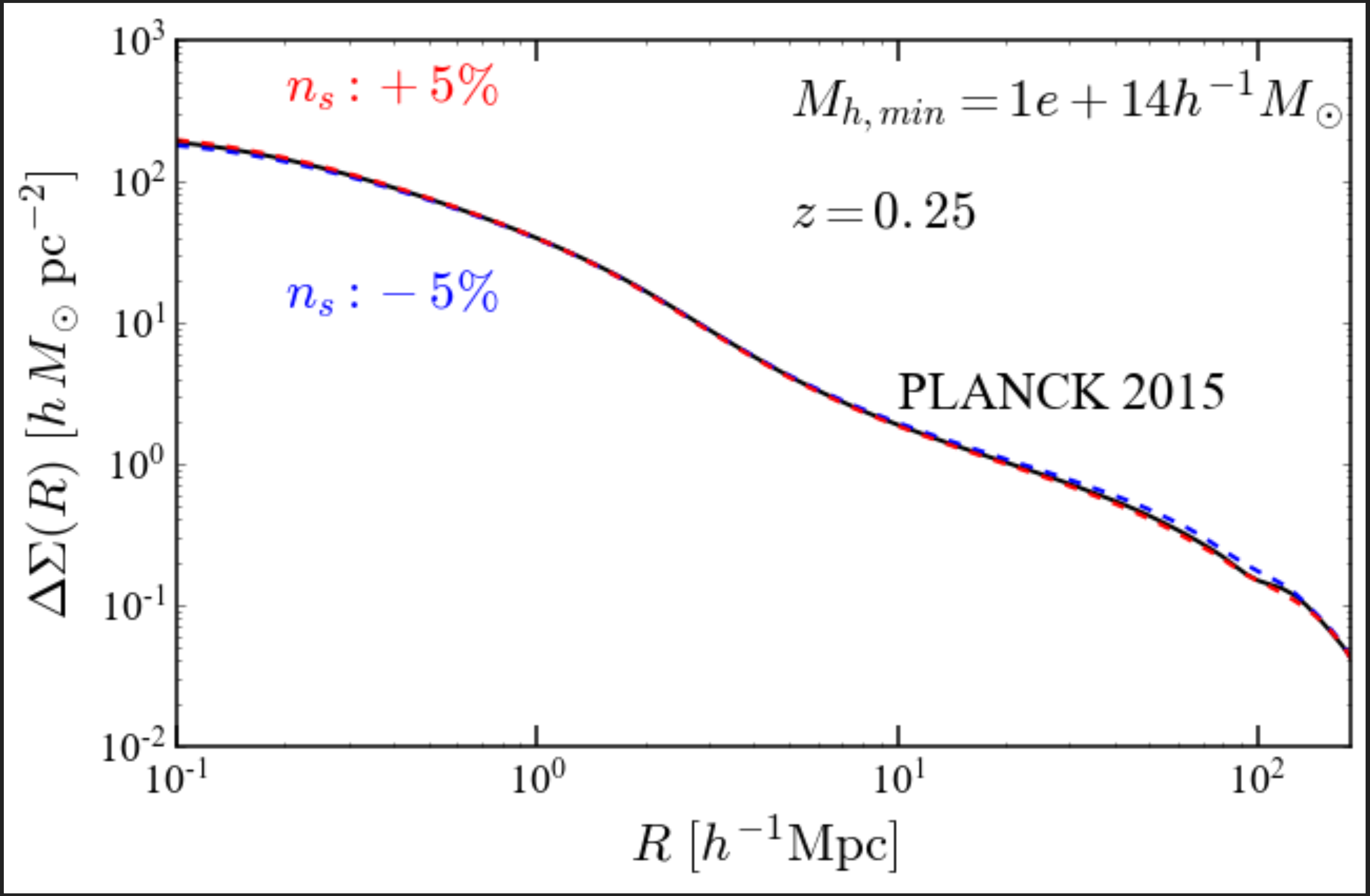
OUR ΔΣ EMULATOR DEMONSTRATIONS



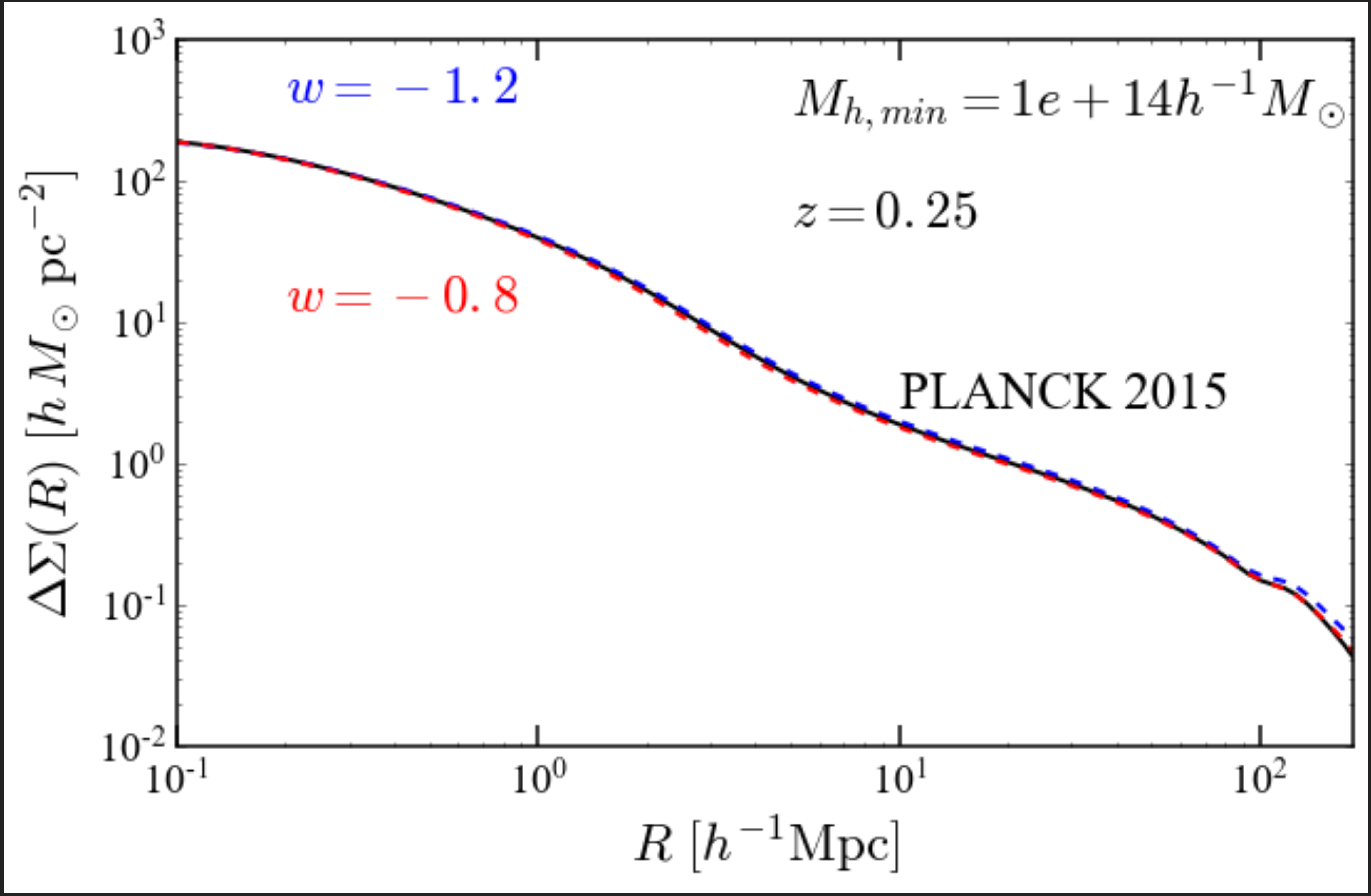
OUR ΔΣ EMULATOR DEMONSTRATIONS



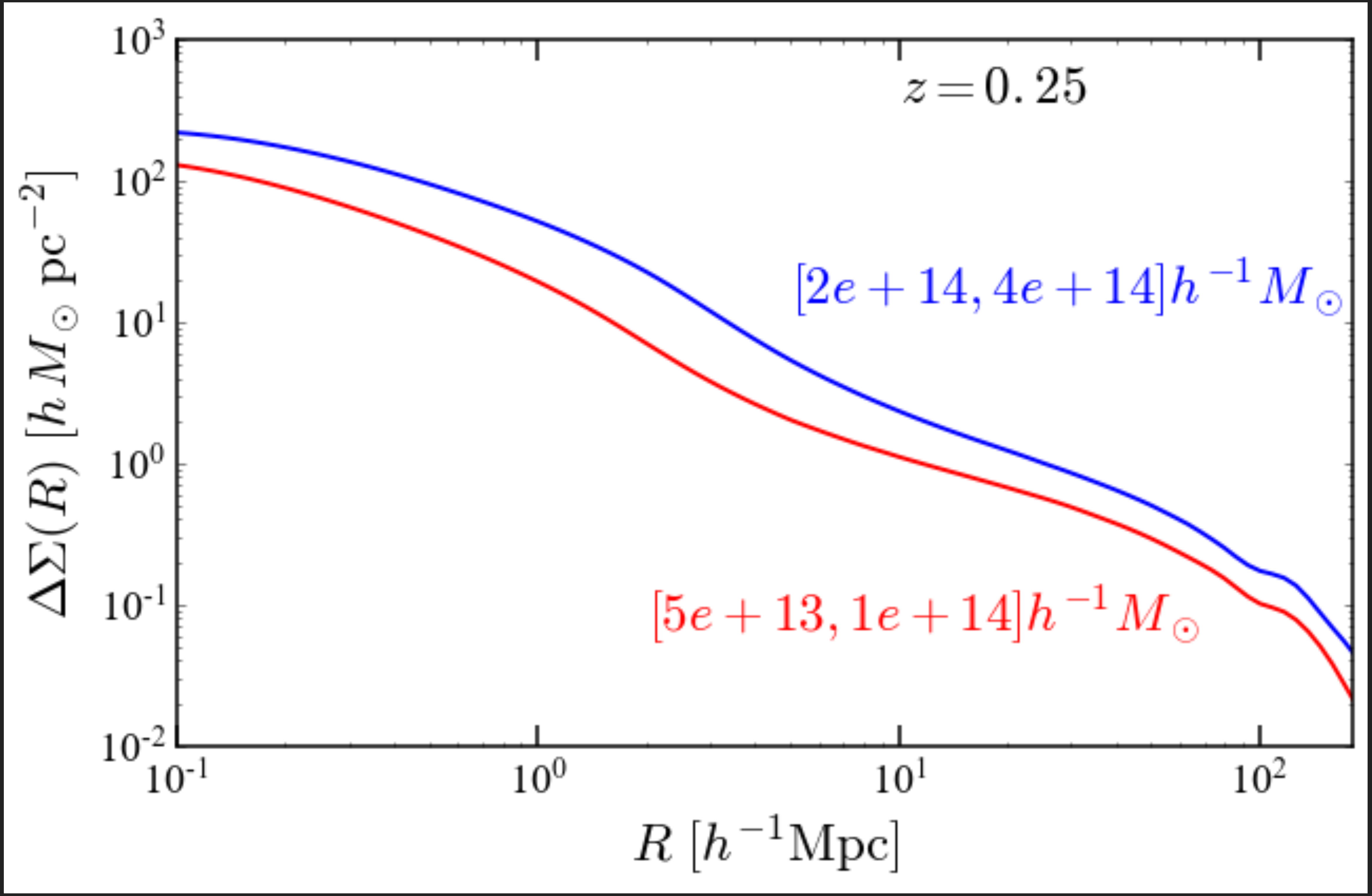
OUR ΔΣ EMULATOR DEMONSTRATIONS



OUR ΔΣ EMULATOR DEMONSTRATIONS

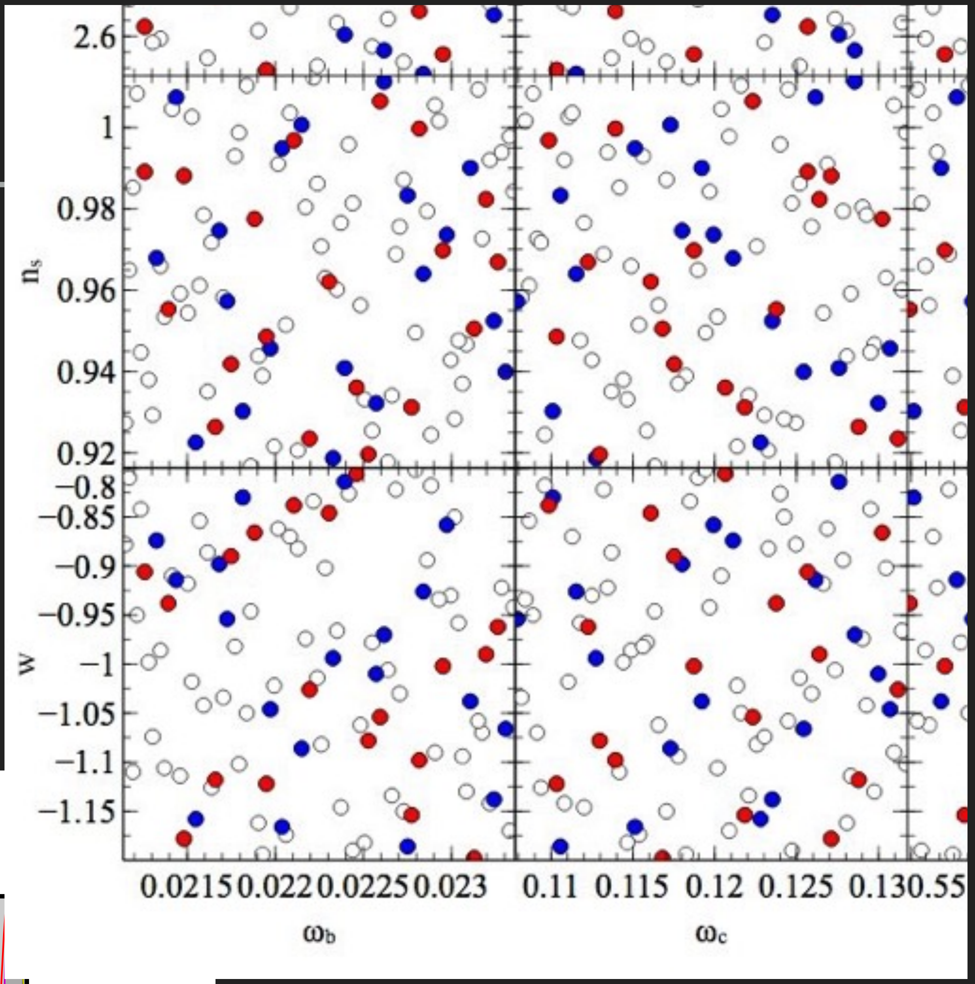


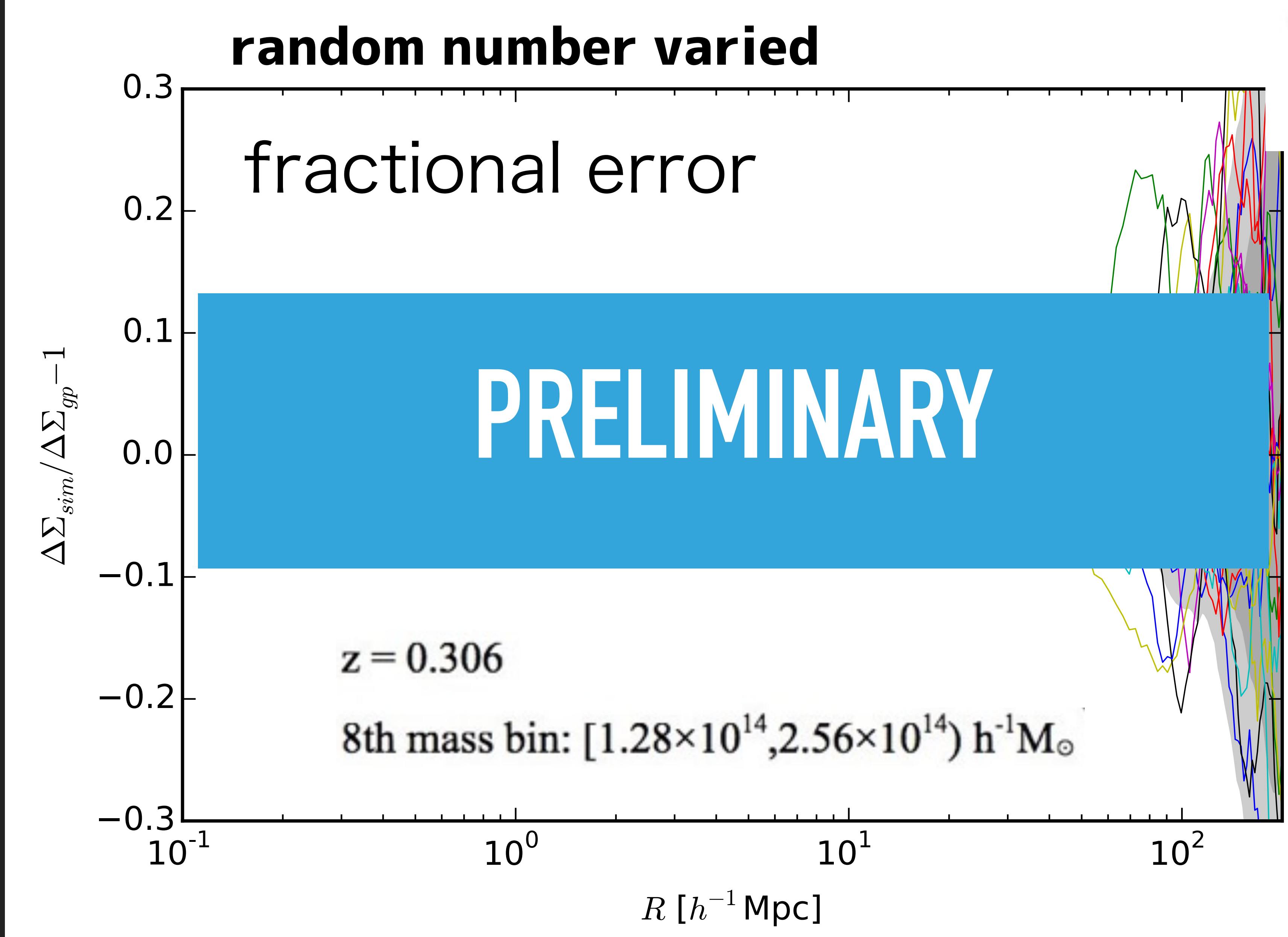
OUR ΔΣ EMULATOR DEMONSTRATIONS



GAUSSIAN PROCESS ACCURACY

training with 20 models (red)
validation with 20 other models (blue)





SUMMARY

- Modeling the halo mass function and galaxy-galaxy lensing signal
  - Latin hypercube design + fitting/GP/spline
  - handy emulator in python almost ready
  - accuracy test undergoing, naively expect 5% accuracy
- To come
  - RSD emulator to combine g-g lensing and 3D clustering
  - further extension under discussion
    - e.g., non-flat, w0-wa cosmologies