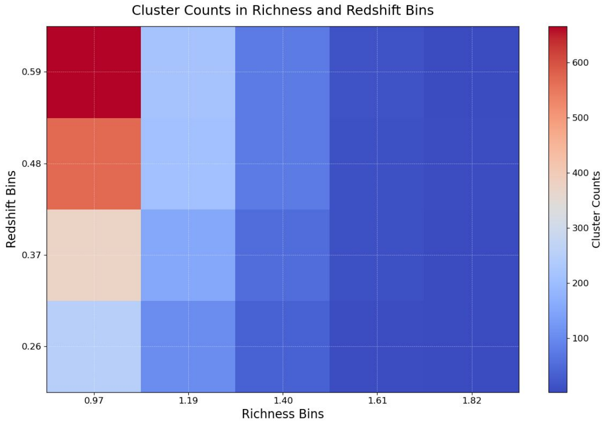
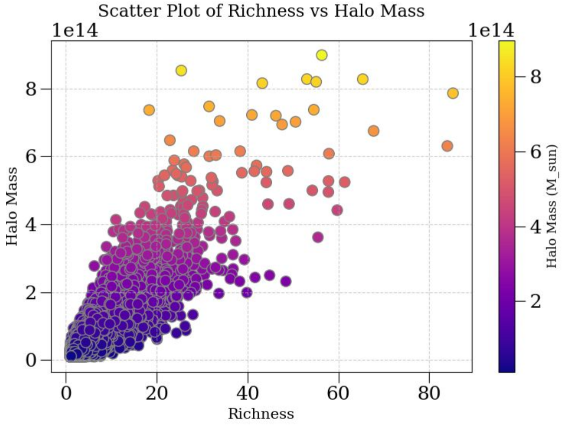
Crow: Cluster Reconstruction of Observables Workbench (Eduardo Barroso, Michel Aguena, Caio Oliveira, Henrique Letieri, and others)

- Cluster Prediction Module
    - This is theoretical prediction code to be used in the DESC-Cluster Pipeline
- It has extra functions but it also calls other DESC-tools such as:
    - CLMM → Shear Computation
    - PYCCL → Halo mass function and Cosmology objects
- The code was previous inside Firecrown
- Now the code is completely independent of Firecrown and it can be used there (check PR #581)
- The code is more flexible, easier to use and to develop and has faster(100x)/efficient options!

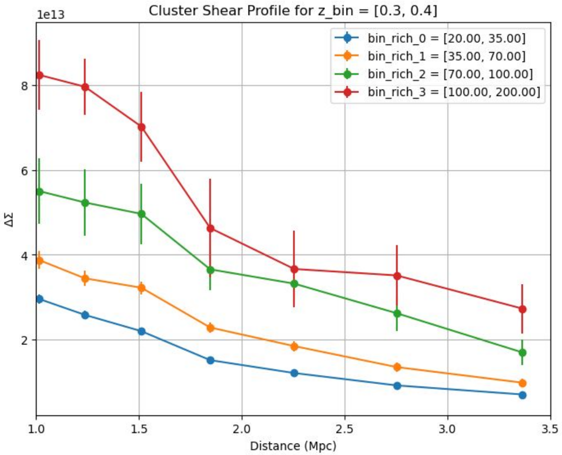
Quick cluster recap: We do a binned analysis of clusters to reduced noise, systematics impact and computation time. For each richness **X** redshift bin:

Cluster Number Counts





Cluster Lensing Profile (gt or ΔΣ)



We cannot measure the cluster mass and there is a scatter between the proxy and the mass

Also, the cluster catalogs may not be perfect: miss detection (incomplete) or false detection (impure)

Crow

We want to compute mean cluster observables for a binned analysis. The observables that we can measure with crow are:

Cluster Counts

$$\mathrm{N}_{ij} = \Omega \int_{z_i}^{z_{i+1}} dz \int_{\lambda_j}^{\lambda_{j+1}} d\lambda \int_{m_{\min}}^{m_{\max}} dm \underbrace{\frac{dn(m,z)}{dm}}_{\text{hmf}} \frac{d^2V(z)}{dz d\Omega} \underbrace{\frac{c(m,z)}{p(\lambda,z)}}_{\text{Selection Function}} \underbrace{P(\lambda \mid m, z)}_{\text{Mass-richness}},$$

Cluster Delta Sigma Profile

$$\Delta\Sigma_{ij} = \Omega \int_{z_i}^{z_{i+1}} dz \int_{\lambda_j}^{\lambda_{j+1}} d\lambda \int_{m_{\min}}^{m_{\max}} dm \frac{dn(m,z)}{dm} \frac{d^2V(z)}{dz d\Omega} \frac{c(m,z)}{p(\lambda,z)} P(\lambda \mid m, z) \underbrace{\Delta\Sigma}_{\text{Excess Density Surface Mass}},$$

Cluster Shear Profile

$$g_{t\,ij} = \Omega \int_{z_i}^{z_{i+1}} dz \int_{\lambda_j}^{\lambda_{j+1}} d\lambda \int_{m_{\min}}^{m_{\max}} dm \frac{dn(m,z)}{dm} \frac{d^2V(z)}{dz d\Omega} \frac{c(m,z)}{p(\lambda,z)} P(\lambda \mid m, z) \underbrace{g_t}_{\text{Reduced Tangential Shear}},$$

Cluster Mean Mass

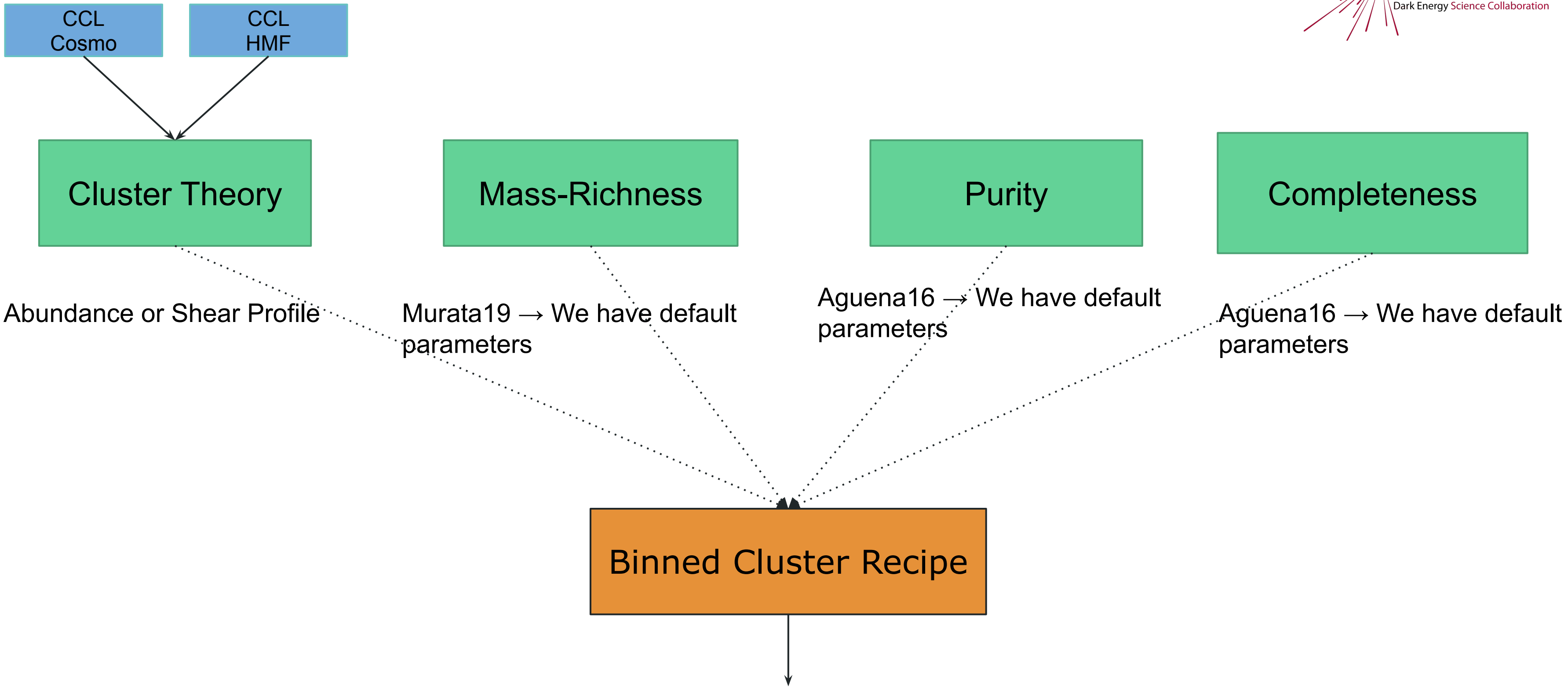
$$logM_{ij} = \Omega \int_{z_i}^{z_{i+1}} dz \int_{\lambda_j}^{\lambda_{j+1}} d\lambda \int_{m_{\min}}^{m_{\max}} dm \frac{dn(m,z)}{dm} \frac{d^2V(z)}{dz d\Omega} \frac{c(m,z)}{p(\lambda,z)} P(\lambda \mid m, z) logM,$$

Cluster Mean Redshift

$$z_{ij} = \Omega \int_{z_i}^{z_{i+1}} dz \int_{\lambda_j}^{\lambda_{j+1}} d\lambda \int_{m_{\min}}^{m_{\max}} dm \frac{dn(m,z)}{dm} \frac{d^2V(z)}{dz d\Omega} \frac{c(m,z)}{p(\lambda,z)} P(\lambda \mid m, z) z,$$

ij represent the redshift and richness bin

Crow: How to get the theoretical predictions?

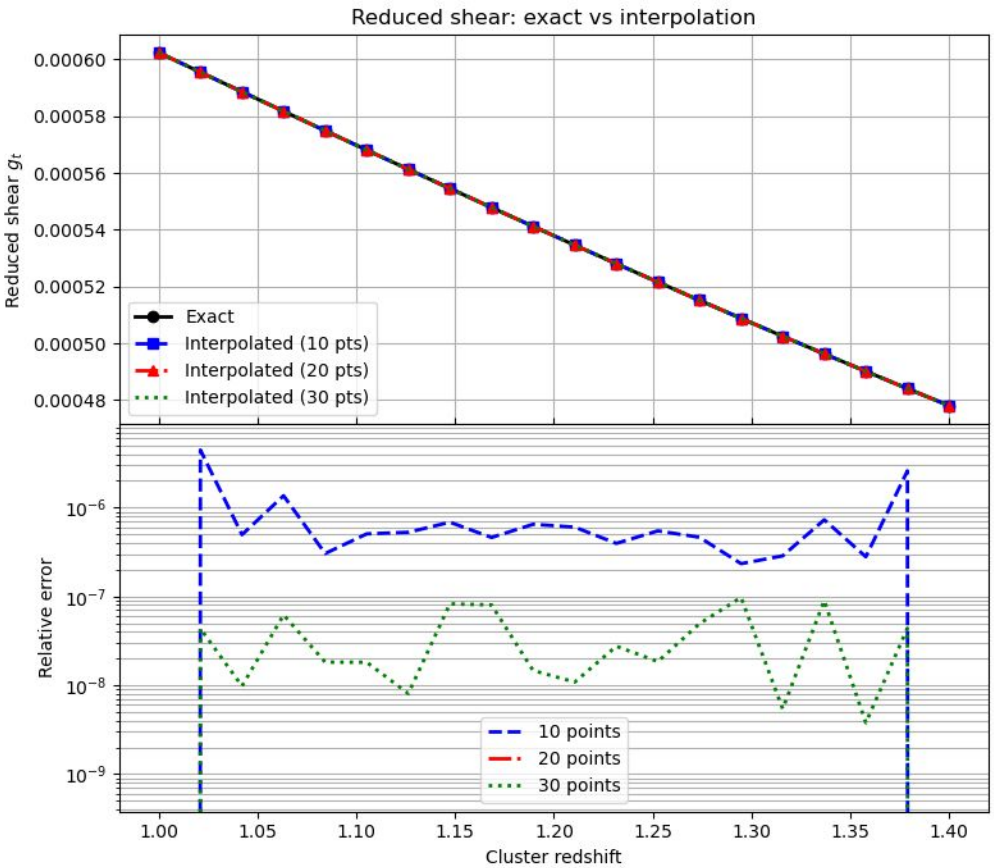


Compute shear (ΔΣ or gt) or counts (include mean mass and redshift)

Crow: New implementations in the stand-alone module

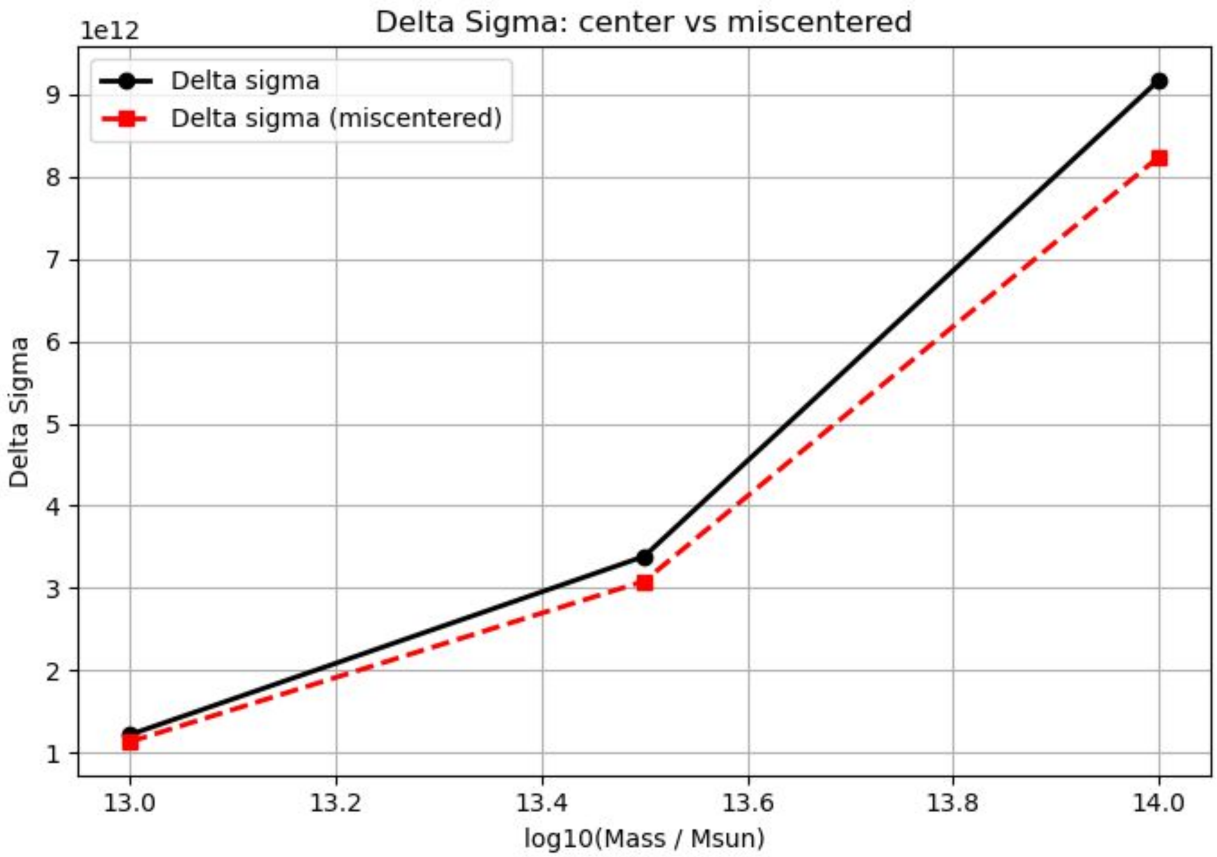
- Shear Computation
  - Reduced shear prediction
    - Lens efficiency with interpolation or exact (integrated functions)
  - Miscentering computation
    - Default distribution or given by the user
  - Vectorized functionality, two-halo term and boost-factor
- Cluster Recipe
  - Unified recipe for all analysis
  - Second recipe version: Now we can either use integration or a grid computation for the observables that is <u>much faster</u>!
- Theoretical prediction can now easily be called by the user
  - Previous implementation in Firecrown required sampler parameters to be set

ClusterShearProfile: Individual evaluation (not binned!)



Interpolated lens efficiency version is 10x faster (0.11s vs 0.01s)

ClusterShearProfile: Individual evaluation (not binned!)

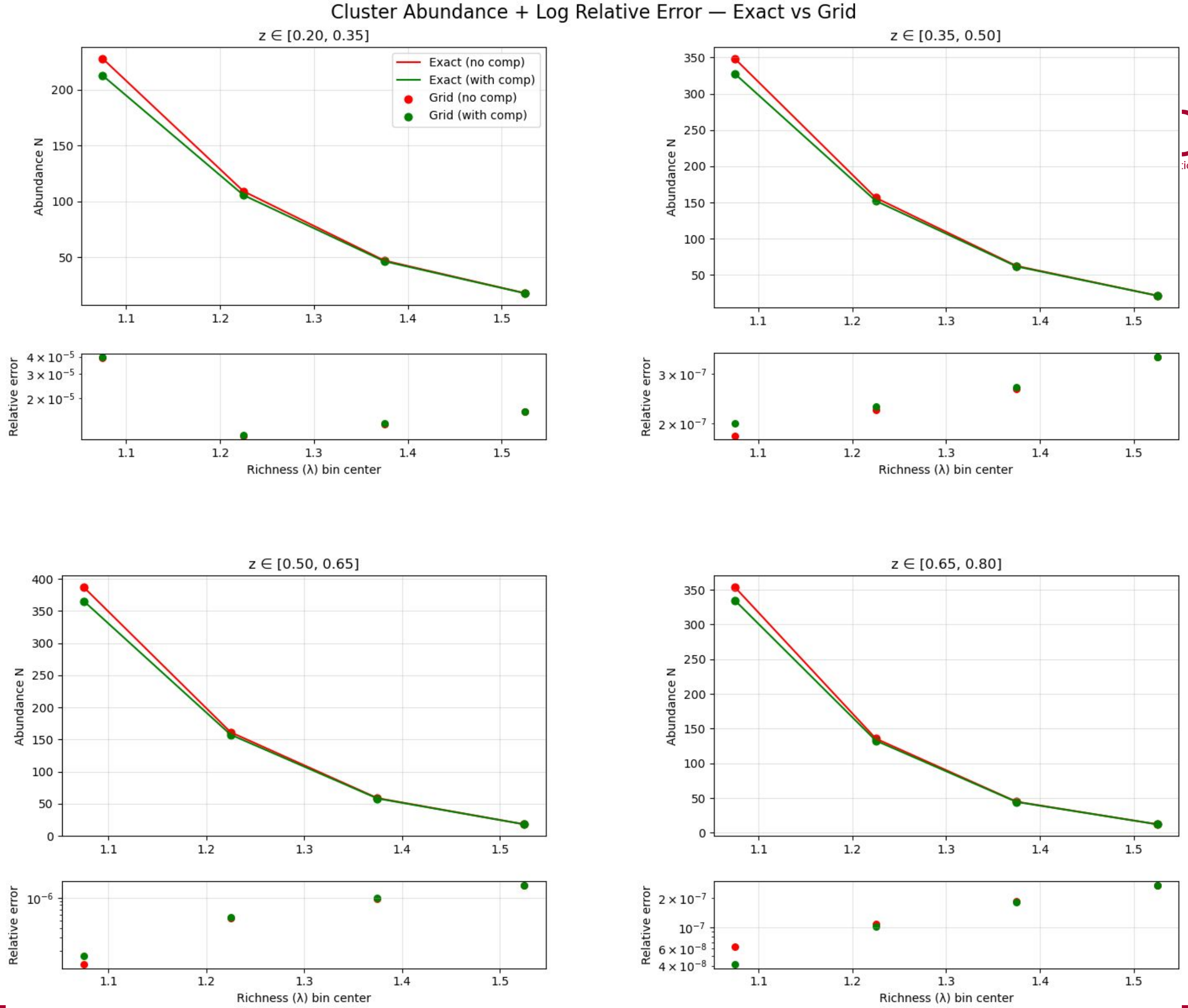


Same can be computed for reduced shear

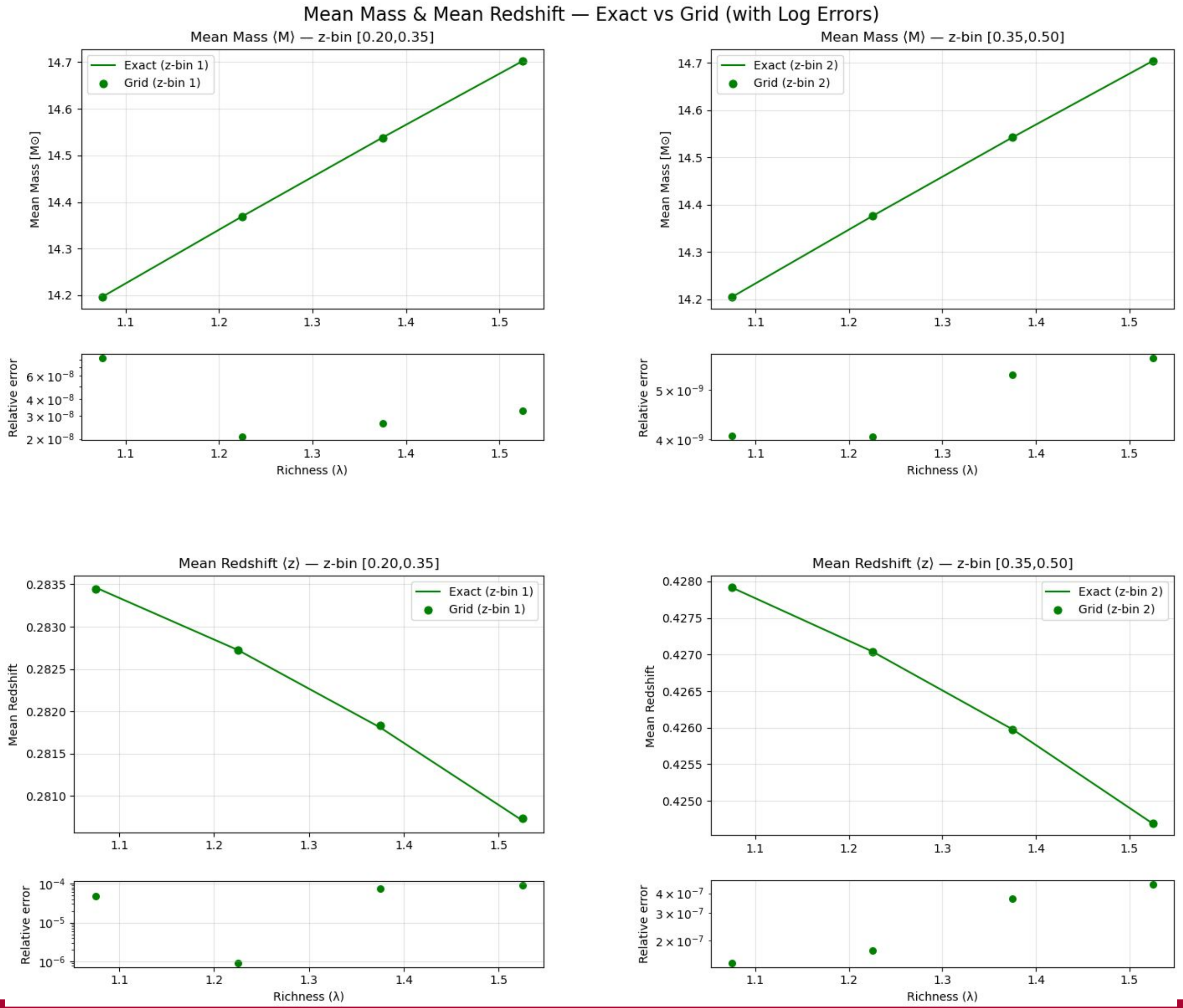
Miscentering is not implemented on the binned analysis yet!

Cluster Recipes: Counts

Both codes are very efficient and take around 0.2s

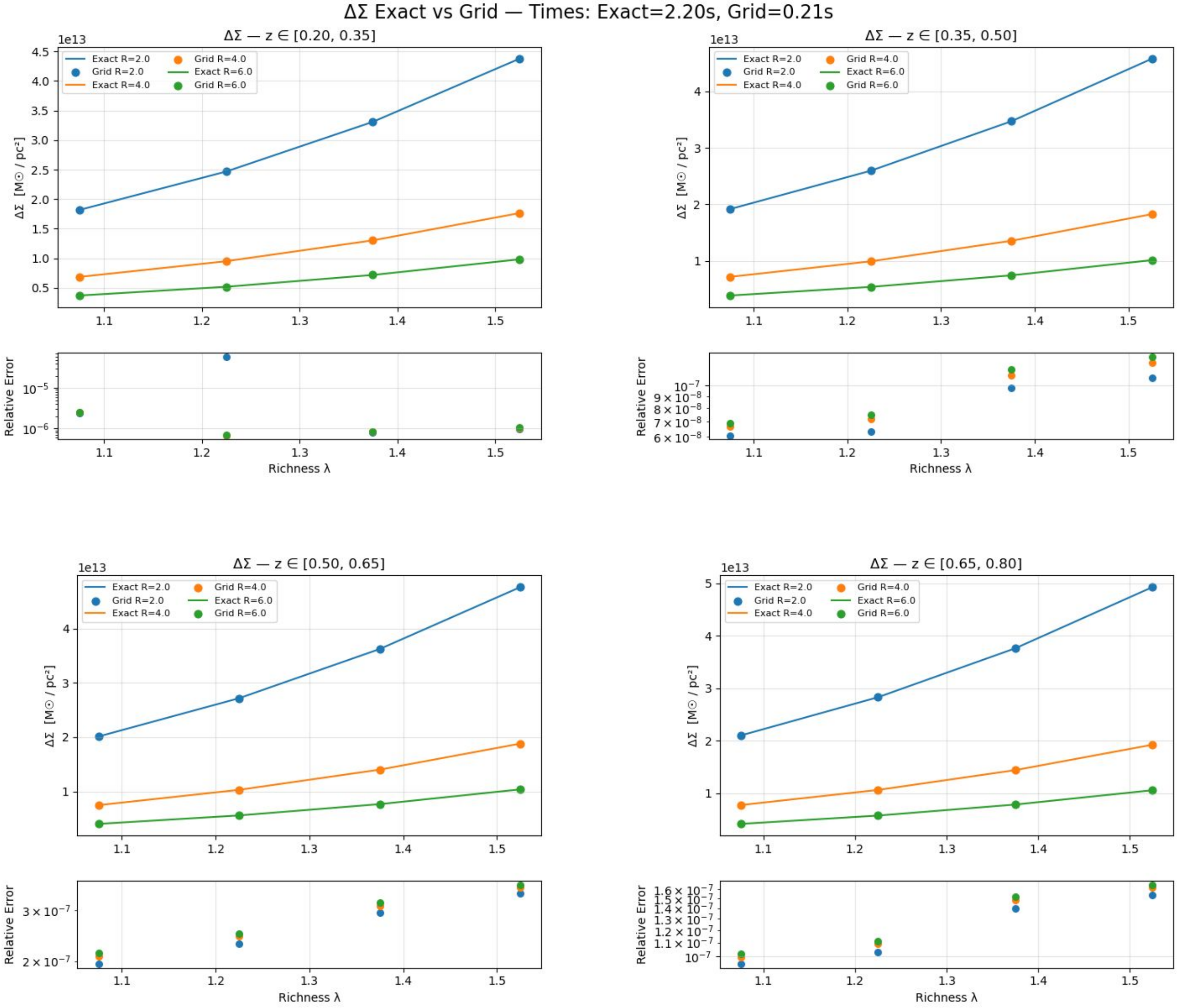


Cluster Recipes

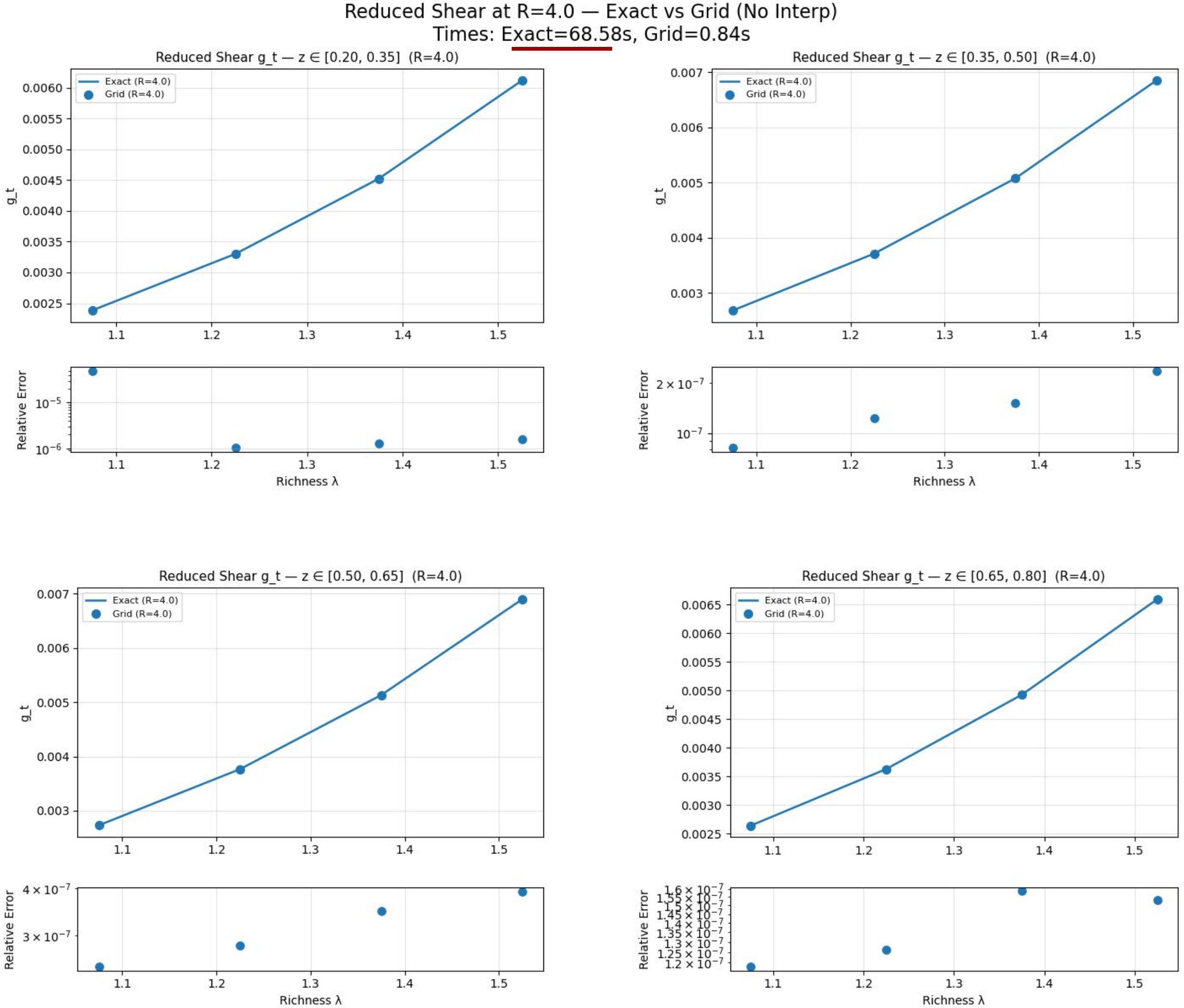


Cluster Recipe: Delta Sigma

Here we show for only 3 radius, but for 10, integral code takes up to 20s while grid only 0.27s



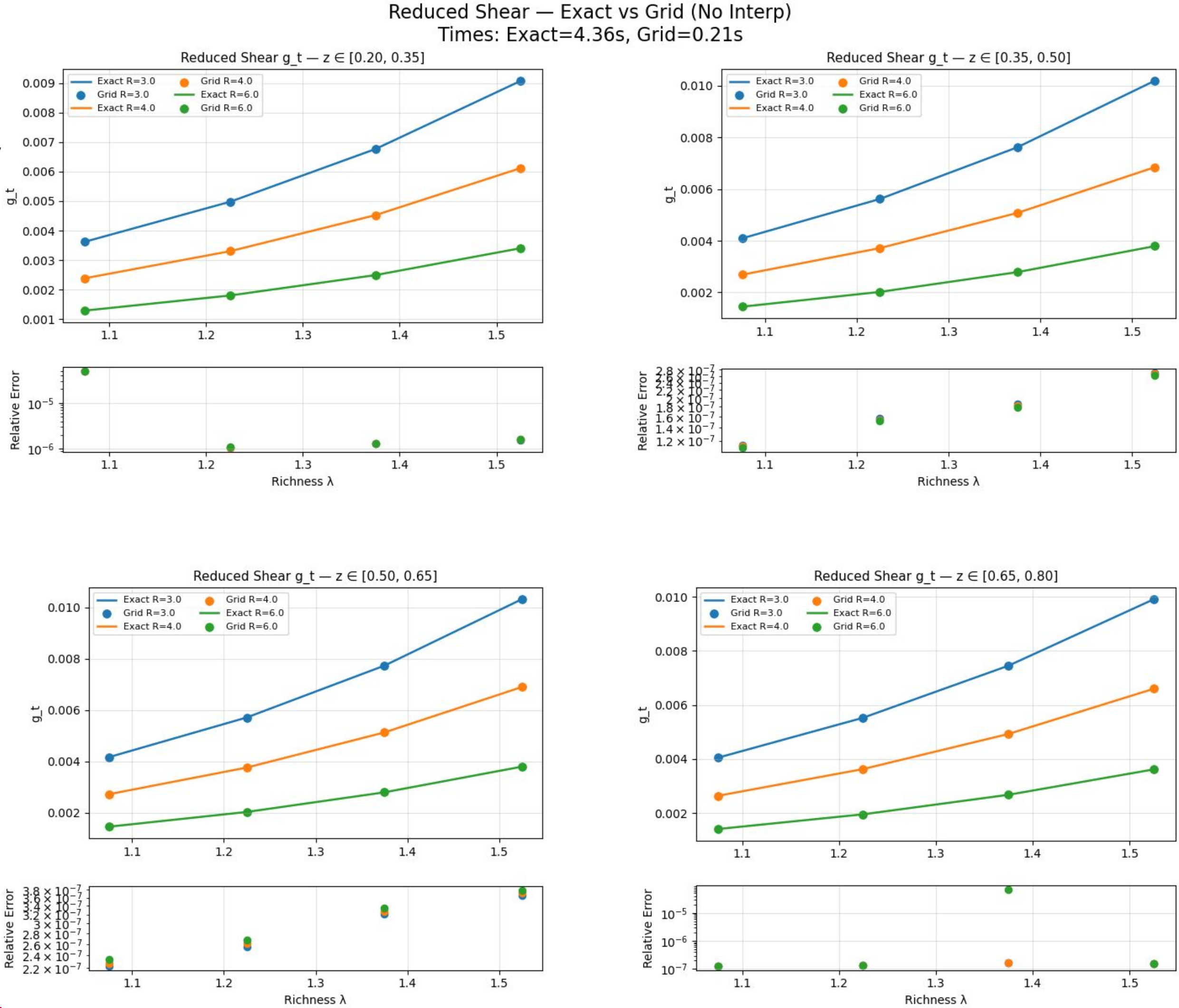
Cluster Recipe: Reduced shear



The time is the reason why we have the interpolation version for lens efficiency!!

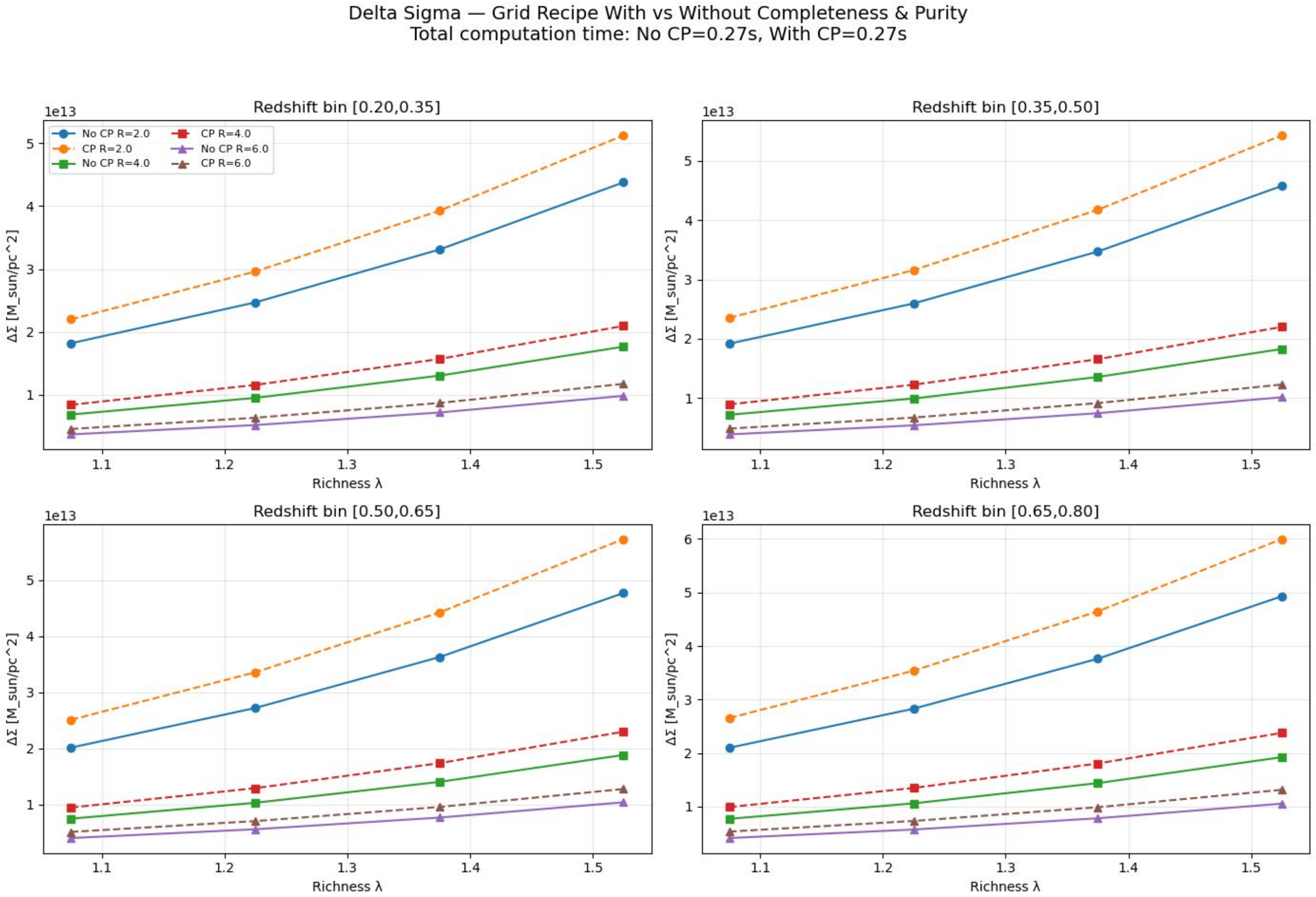
Cluster Recipe:
Reduced shear
with interpolated lens efficiency

Integral →time improvement of ~15x
grid →time improvement of ~4x



Cluster Recipe: Grid examples with selection function

We are changing the exact recipe to work with purity



Conclusion

- Crow is now an independent cluster prediction module that can be used with other DESC-tools such as Firecrown
- It is pip and conda installable (package name desc-crow)
- The code has exact computations and now also tabulated ones with great time efficiency and maintaining good precision (100x faster, less than 0.001% error)
- We are working on documenting the code and creating user friendly examples (PR#63)
- If you have suggestions, feedback, questions, contact us/ and or submit an issue on the repo!
    - Prospections:
        - Add miscentering to the grid
        - Fix purity in the exact recipe
        - Add unfixed concentration option to grid