Automation workflows at Data Lab / Modeling of AGN torus properties with ML techniques

Keenan Fiedler

Automation workflows at Data Lab

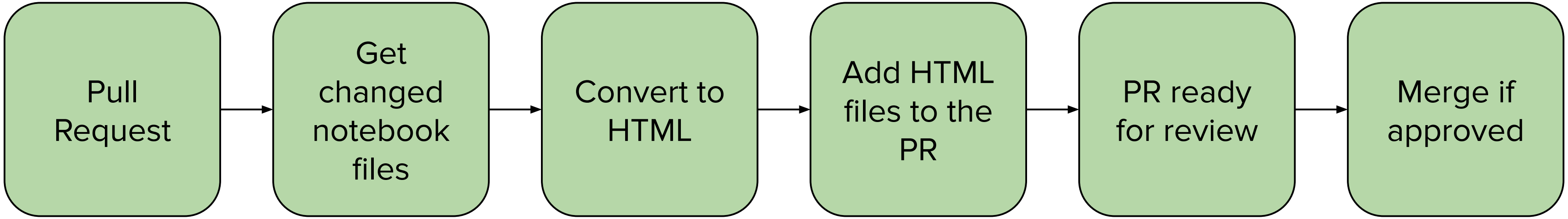
Technical Project

Github Actions allow for automation of tasks

- Github is a service for storage, collaboration, and creation of code and files
- Code can run on a github repository after various git actions are taken
  - Pull requests - requests to add a new change to the main repository, affecting everyone
- Uses yaml and bash scripting
  - Yaml is a programming language used for configuration files and messaging. Github uses it for running steps and setting up environments in actions.
- Useful for error checking, unit testing, and many more functions
- Runs on a virtual machine on Github servers
- Many open-source Github Actions have been created by other users

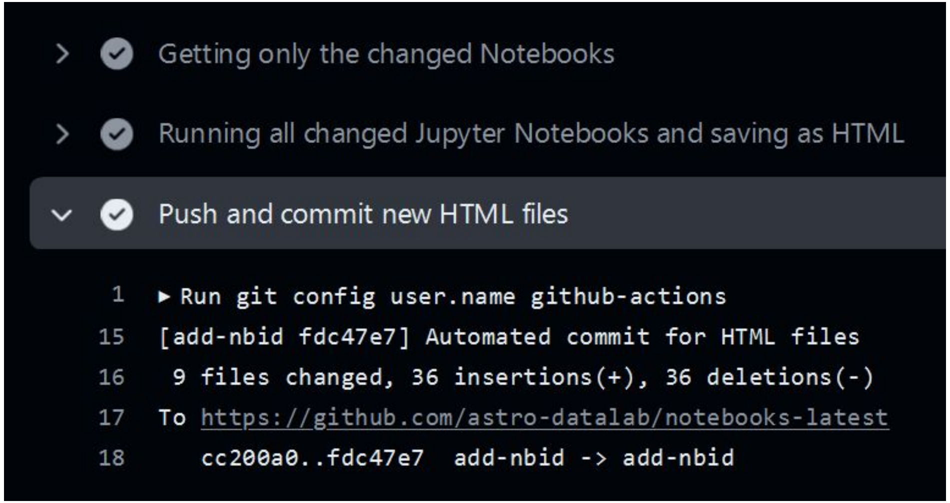
Automating conversion of notebooks to HTML

- Data Lab has a Github repository for science example Jupyter Notebooks
  - Each notebook has an HTML copy for easier preview without loading a Jupyter environment
- Problem: HTML generated by hand after changes finalized
- nbconvert package allows conversion of Jupyter Notebooks
- Open-source github actions can get all changed files in a PR



HTML conversion is fully automated for PRs

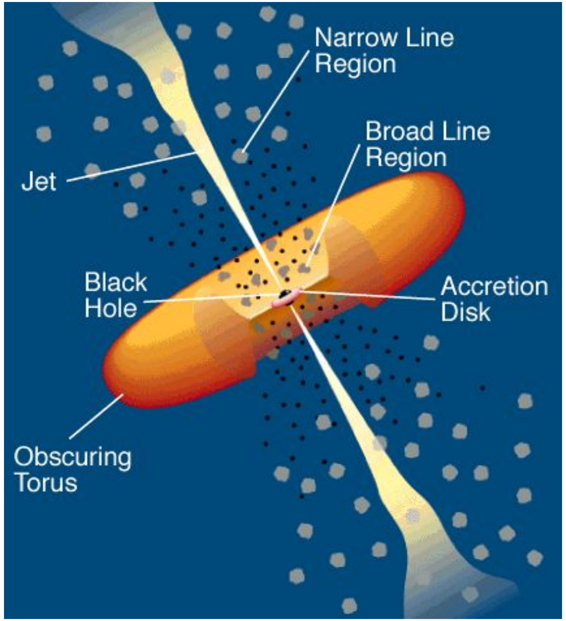
- Successfully working and in production for the past few months
- Takes only 1-2 minutes to run on about 10 notebook files
- Massive time save and ease of adding new notebooks by contributors



Modeling of AGN torus properties with ML techniques

Science Project

AGN should have a torus of dust in clumps

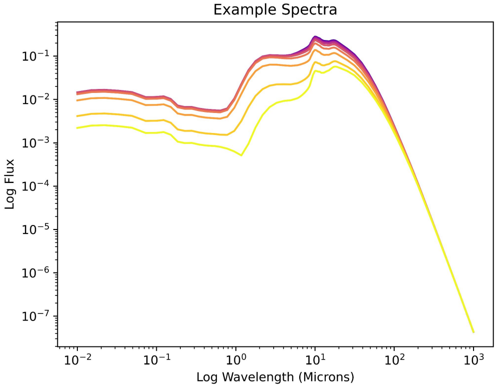


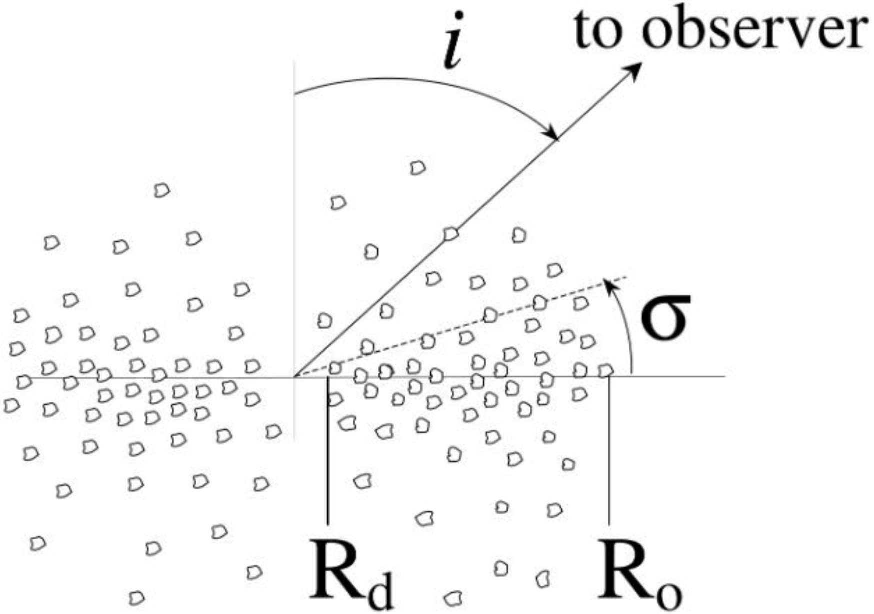
AGN unification
(from Urry & Padovani 1995)

- AGN don't have a stellar (black-body) spectrum
- They show an IR peak ➔ dust can do that
- They come in two flavors: type-2 with narrow emission lines & type-1 with both broad and narrow lines ➔ orientation effect
- Broad lines observed in type-2s as well, but in polarized light ➔ scattered towards us
- Smooth dusty torus can't sustain the vertical height required to explain type-1/type2 number statistics ➔ dust in clumps can
- Line-of-sight variation of obscuring gas column observed in X-rays ➔ variations consistent with orbiting clumps

CLUMPY model explains these features

- Six parameters as input
  - τ_V - single cloud optical depth
  - N_0 - clouds in equatorial plane
  - σ - angular torus width
  - $Y = R_o/R_d$ - torus thickness
  - r^{-q} - radial cloud distribution
  - i - observer viewing angle
- Outputs SED





Nenkova+2008b

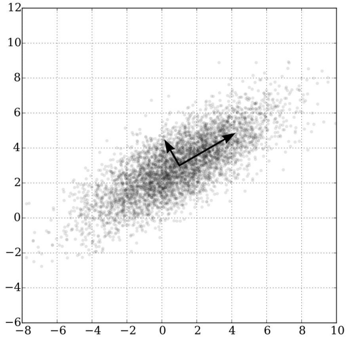
Models are slow to build and expensive to store

- Existing set of ~1.2 million models
  - Covers wide parameter space
  - ~1.2 GB of disk usage
  - 0.5 GB for storing SED only
  - Cumbersome to interpolate
  - From: www.clumpy.org, precomputed due to slow build (radiative transfer)
- Goals for new set
  - Same parameter space coverage
  - Smaller storage footprint
  - Fast to load and generate new models within parameter space
  - Accurate as possible to original CLUMPY models with same input parameters

Two methods: PCA and Machine Learning

Principal Component Analysis

- Reduce dimensionality of models and find rotation of coordinate system such that projection onto new set of axes minimizes total variance; first PCA explains most variance, etc.
- Can recreate original CLUMPY SEDs with just a few PCAs (and with a small loss of accuracy)
- Simple to implement with SciPy

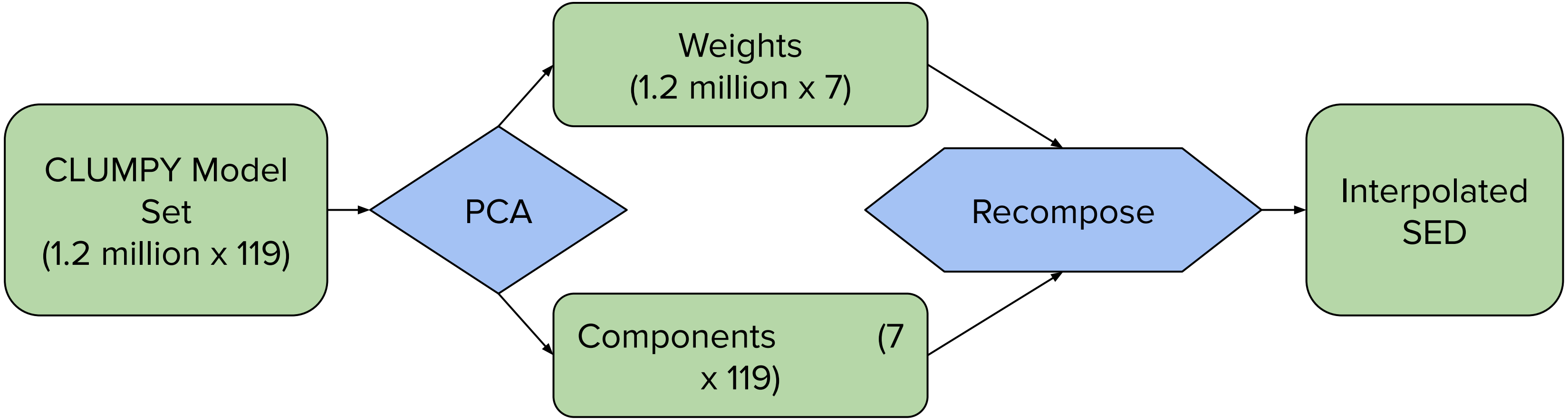


Machine Learning - Autoencoder

- Create a neural network of weights - decoder half of autoencoder
- Train it on the original CLUMPY model set
- Full set of models reduced to a small set of weights
- Load ML model and input any parameters to reproduce an SED with some error
- Simple to implement with Tensorflow and Keras

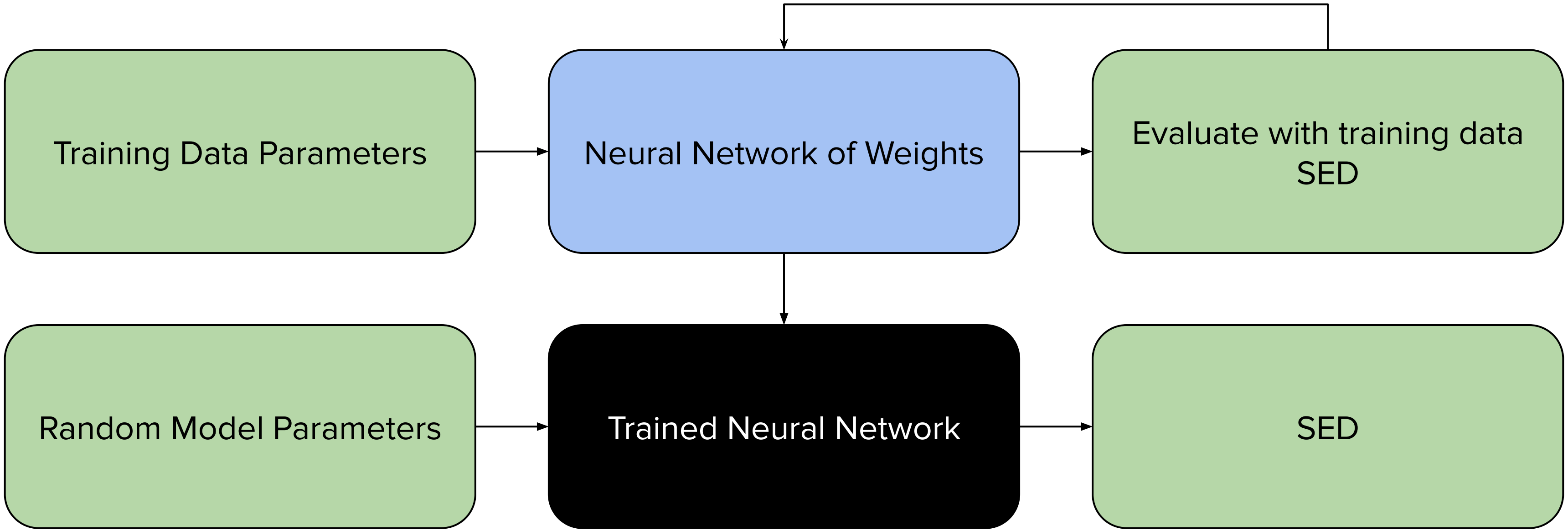
First Method: decompose original models with PCA

- Decompose originals into weights and components
- Multiply matrices to recompose original dataset
- Interpolate over the 6-dimensional parameter space to generate new SED



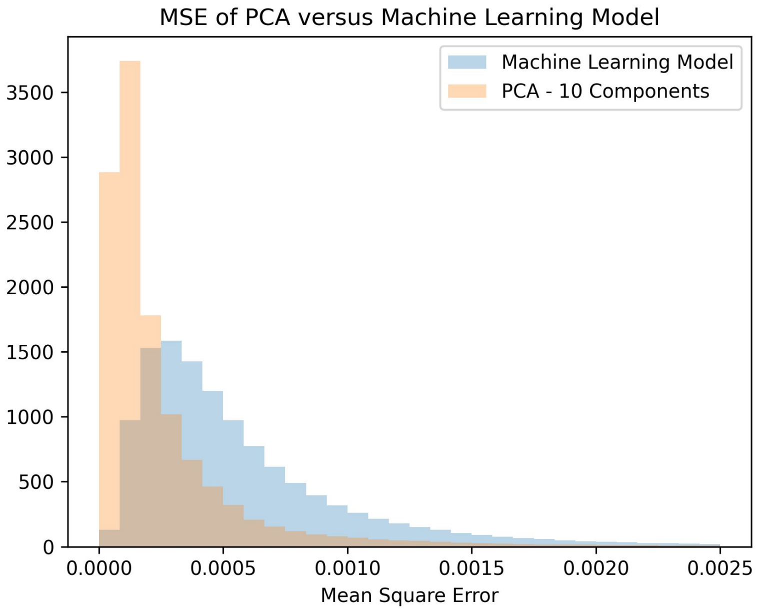
Second Method: uncover latent model from machine learning

- Train with CLUMPY models to cover full parameters space
- Then generate random parameters and get an SED out



ML is faster, smaller, and more accurate than PCA

- Faster to reconstruct, slower to train
  - No need to set up n-D interpolation
  - No need to reload full set of models from decomposed file
  - Can generate 1 million SED in ~16 seconds
- Lightweight disk size
  - ML: 340 KB, ~1275x compression
  - PCA: 49 MB, ~9x compression
- Slightly lower accuracy
  - At higher number of components but a smaller file and faster reconstruction (trade-offs to make)

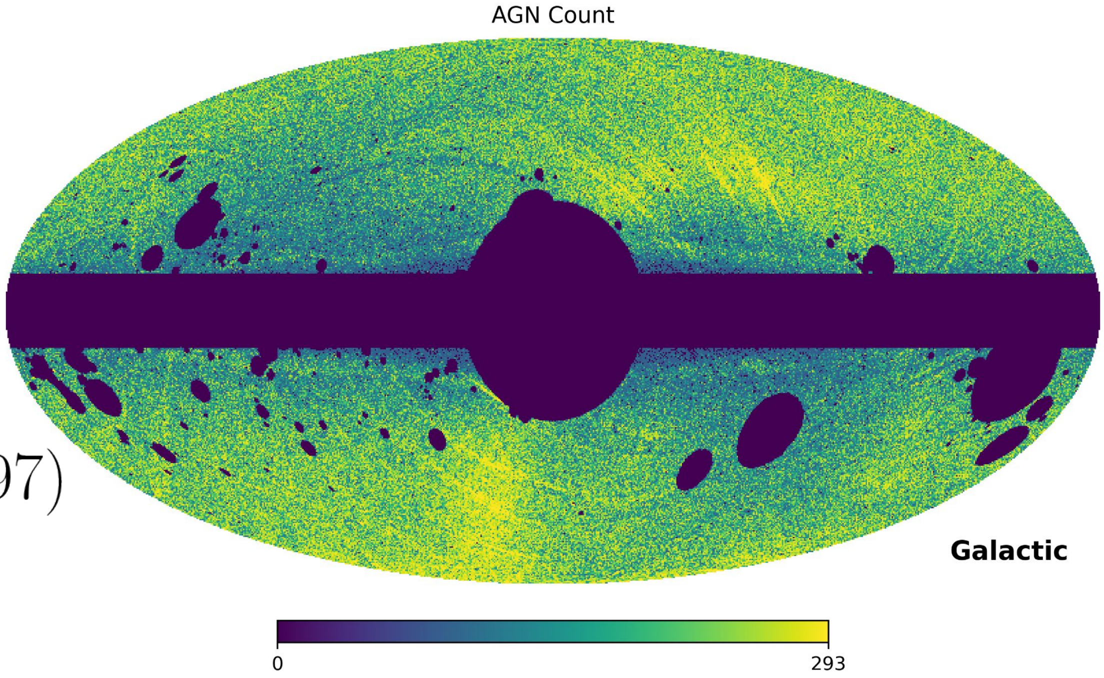


Select clean AGN sample from WISE catalog

- Cut out galactic plane and galactic center
- Interfering objects
  - Nearby galaxies
  - Nebulae
  - Bright stars
- Cut in WISE colors

$$W1 - W2 > \alpha e^{\beta (W2-\gamma)^2}$$

$$(\alpha, \beta, \gamma) = (0.662, 0.232, 13.97)$$



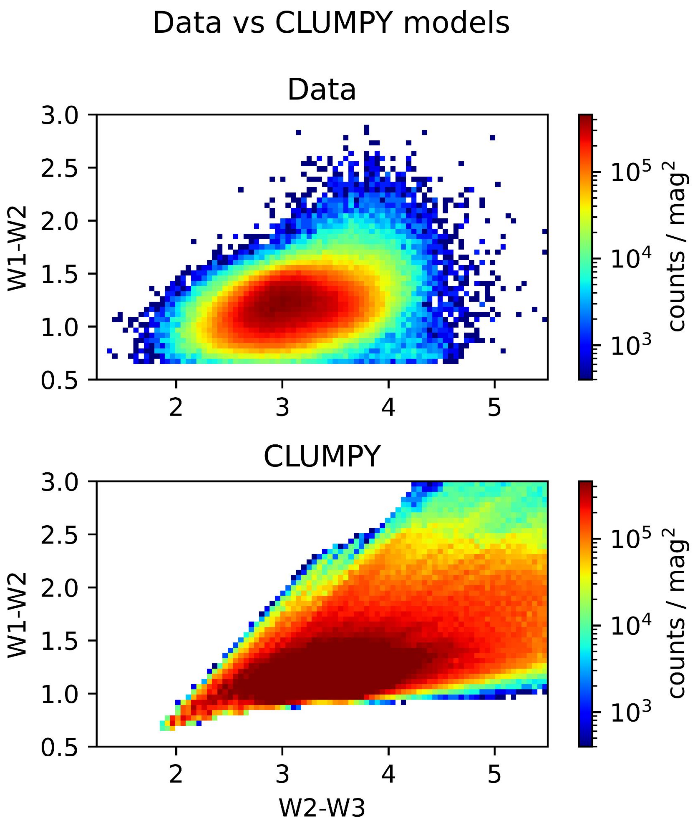
All selection criteria from Assef et. al. 2018

Confirmed AGN by crossmatch with SDSS and DESI EDR

- Crossmatched our data with SDSS and DESI EDR spectra
  - Used Data Lab crossmatch tool and 1.5 arcsecond radius
- Took only objects classified in AGN categories e.g. QSO
- Require accurate redshift for future reddening in spectra of models
  - cut objects with high redshift error and high/low redshift
- Final catalog is ~250,000 objects

Find model weights with regression

- Sampling of model parameters uniform. True distribution in nature is unknown.
- Goal: find distribution of model parameters consistent with observations.
- Generate many model color tracks (here 1M)
    - Compute colors as function of viewing angle
    - Requires 100 million SED - 100 viewing angles/model
- Perform regression on color tracks: find weights for each model such that the linear combination explains the color-color distribution of WISE AGN

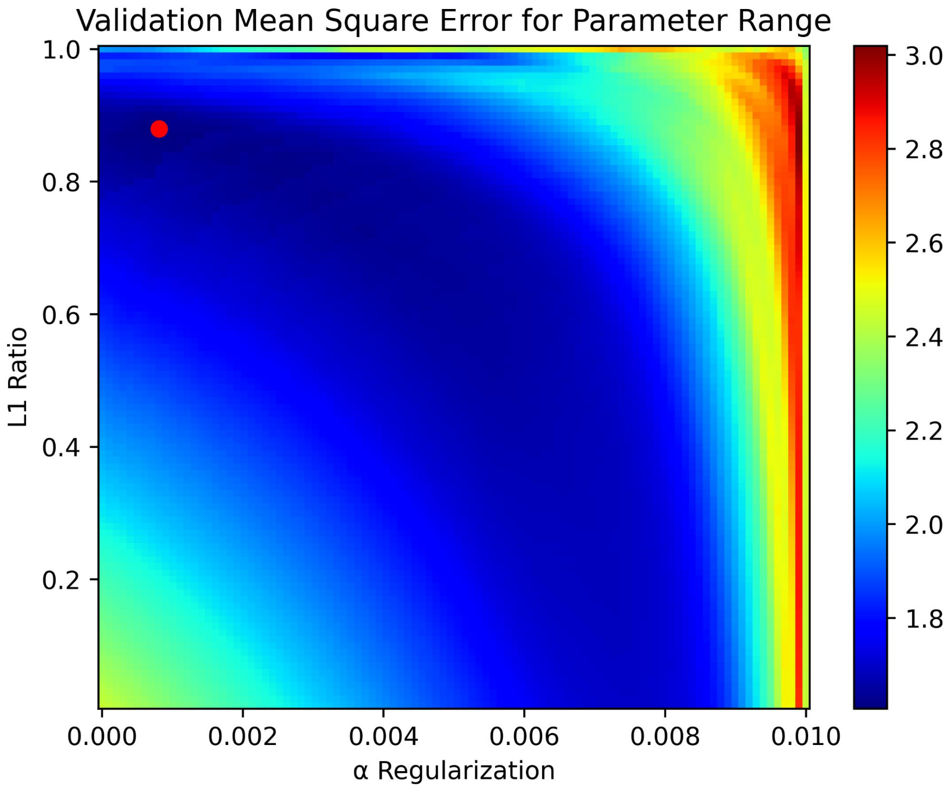


Regression and regularization

- Need to find a combination of models to recreate observed data (CC histogram)
    - Regression finds weights for each model such that the linear combination minimizes the residuals
- Regularization of weights
    - Regularization penalizes too large individual weights
    - Is also used to enforce non-negative weights, since a model can not contribute negatively (unphysical)
    - L1 regularization (Lasso) - adds the sum of the absolute value of parameters to MSE
    - L2 regularization (Ridge) - adds the sum of the square of parameters to MSE
    - Both penalize outliers by encouraging smaller parameter values and reduce variance
- Multiplication of L1 and L2 by constant value α modifies this penalization
    - $\alpha < 1$ - less penalization than baseline
    - $\alpha > 1$ - more penalization than baseline

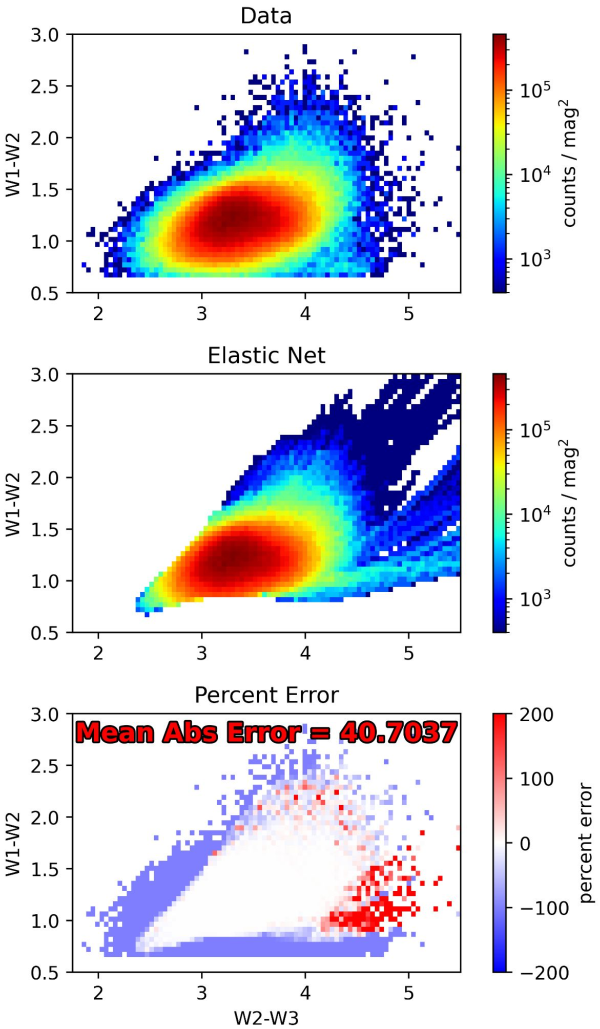
Cross Validation of Regularization and Normalization Ratio

- ElasticNet uses a combination of the L1 and L2 normalizations
- Cross validating by running a matrix of α and L1 ratio
- Ran on ~1 million random SED
- Found best α and L1 ratio for our case empirically
- L1 ratio is the percent of L1 normalization
    - L1 ratio of 1 is 100% L1 normalization
    - L1 ratio of 0 is 100% L2 normalization



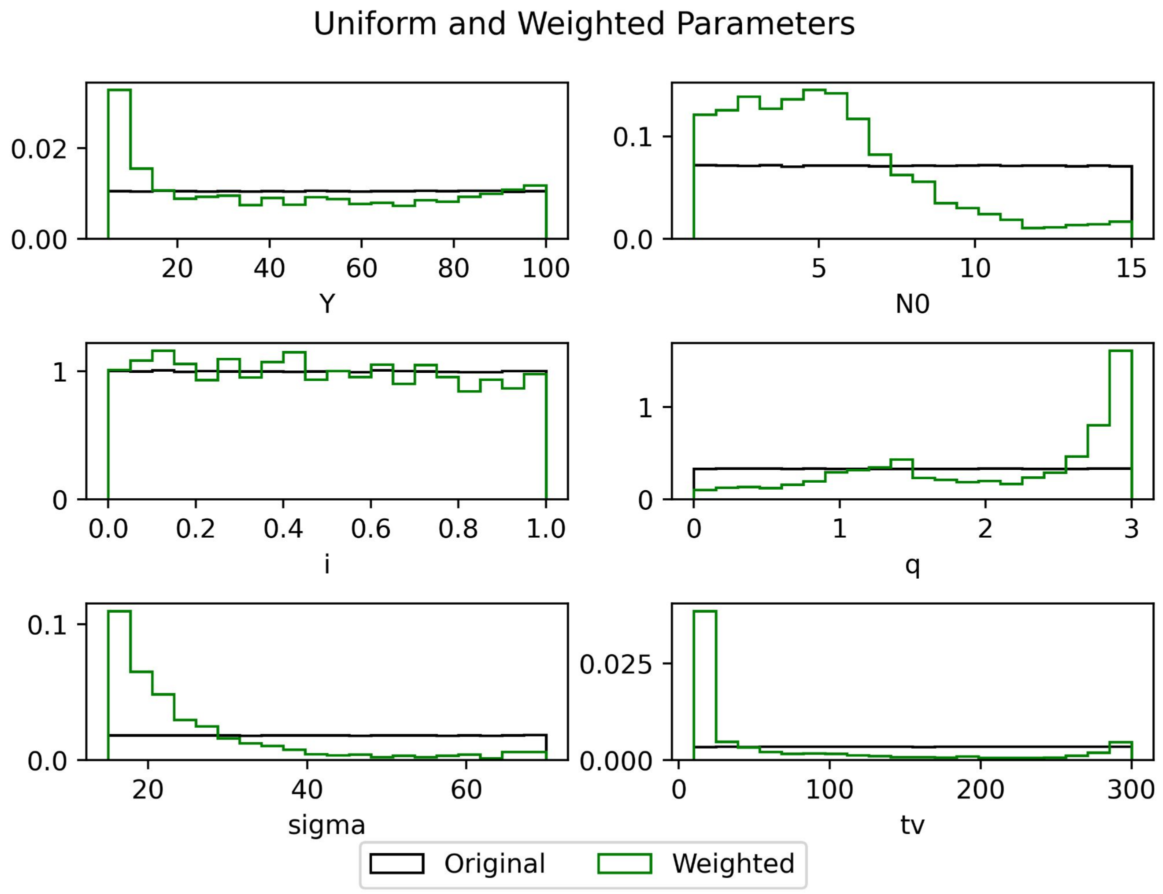
Models reproduce observed AGN counts

- The Elastic Net produces a close approximation of the data
- Residuals next to none in areas where models cover
- Models are not yet reddened ➔ need to account for drop-outs
- Also, models have a known deficiency in 1-2 micron flux, making them a bit too red ➔ next steps



Parameter Distribution

- Regression weights change distribution of parameters from uniform input sampling
- Can determine what parameter range of CLUMPY models is more physical for real AGN
- Initial results suggest: small torus size Y, small to intermediate number of clouds N0, steep radial distribution of clouds, torus width consistent with type1/2 number statistics, relatively small cloud optical depth.



Questions?