MASS ACCRETION GROUPING ANALYSIS: USING MACHINE LEARNING TO INVESTIGATE DARK MATTER HALOS

By

Nicholas Chason

Dissertation

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

in

Physics

December 17, 2022

Nashville, Tennessee

Approved:

Andreas Berlind, Ph.D.

Kelly Holley-Bockelmann, Ph.D.

David Weintraub, Ph.D.

Stephen Taylor, Ph.D.

Jesse Spencer-Smith, Ph.D.

Dedication

I dedicate this thesis to my family and friends.

And,

Nashville's very own Raising Canes!

May the future graduate students be blessed with its delicious fried chicken.

Acknowledgment

I would like to thank all the people who made it possible for me to attend such a great university in such a grand city, and all the members of my committee (including two former committee members who moved onto other ventures). Especially to Andreas Berlind for always being brilliantly minded, and likewise to Kelly Holley-Bockelmann, for all of her help and contributions.

and

I also owe a special shout out to the ACCRE folks for all their help - even when it seemed that all was lost.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

It is the goal of this paper to evaluate how data science methods can be used to group simulated halos by their mass accretion histories - how a clump of dark matter grows its mass over time. We hope to utilize conventional age parameters, other one-parameter models, and several machine learning methods in order to classify halos into separate groups. We will then evaluate these groups for their spatial clustering to see if they may be related to 'assembly bias' halos, which exhibit different spatial clustering over a parameter other than final mass. We will finally evaluate how these halos split over several halo properties, such as concentration, tidal force, spin and more. We hope to gain an understanding of how halos accrete their mass in simulations, and this accretion's relationship with other halo properties. By the end of the project we hope to gain a deeper understanding of how we can utilize algorithms to aid in the mining of data from large scale simulations of the future.

1.1 Simulating ΛCDM, a Cosmological Model in a Box

Being able to simulate a universe in a box is what makes this project possible in the first place. Without powerful supercomputers and a precise understanding of gravity, we would have to rely only on our observational and theoretical models alone to guide our understanding of the formation of the universes most giant structures. While the last decade has seen an exponential rise in data from conventional observations and even multi-messenger gravitational observations, we still rely heavily on simulations to guide our understanding of the largest scales, mostly guided by dark matter and dark energy which we can not see directly. From observations of the cosmic microwave background (CMB) in the very early universe (around redshift, z=1000), to incredible surveys like the Sloan Digital Sky Survey (SDSS) observing some of the most distant quasars (past z=2.5), our ability to precisely measure the distant universe has increased incredibly. Likewise, the capability of the

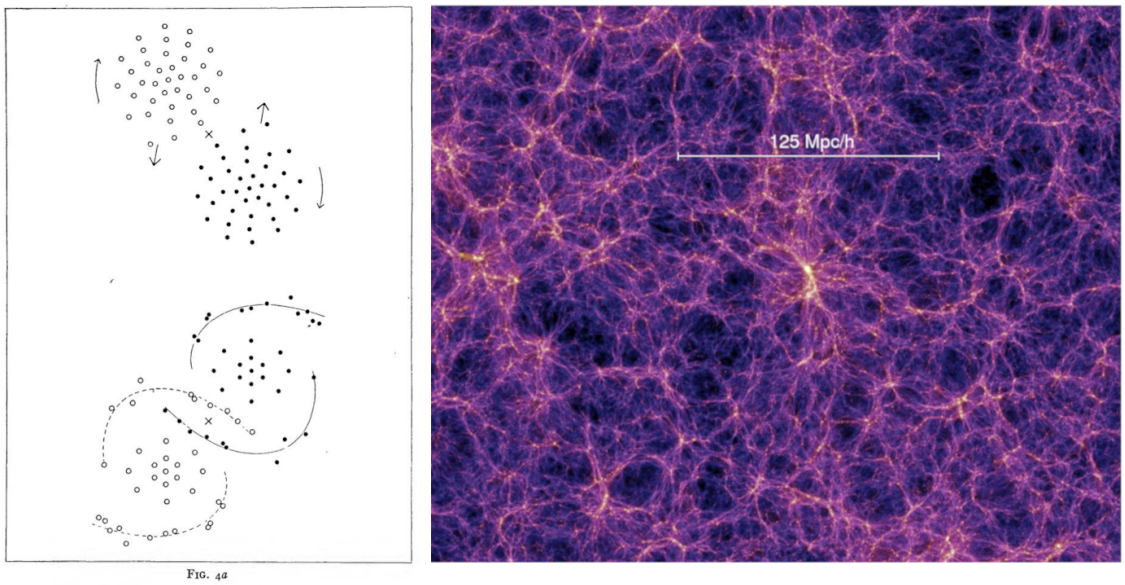

Figure 1.1: We compare the original Holmberg galaxy collision simulation with 37 light-bulbs (left), to the cosmological Millennium Simulation using over 10 billion particles (right) simulating a 500 Mpc/h chunk of the Universe. Highlighted on the figure is a 125 Mpc/h distance, which is nearly equal to the Vishnu boxsize of 130.

simulations we can perform has improved astronomically.

Initially, the first N-body simulation was performed using light bulbs, to approximate the force due to gravity. Rather than calculating the forces on theoretical particles on a computer, a photo-voltaic cell was used to approximate the gravitational force on each respective body or 'particle' from bulbs. This simulation, performed by Holmberg (1941), used only 37 data points in order to simulate the collisions of two galaxies. Holmberg used this simulation to estimate that galaxies that initially collide with sufficient energy to escape to infinity, would lose energy and become bound, resulting in galaxy clusters and groups. This simulation was one of the first to show we can model complex real dynamics by running simulations at lower resolutions. Although estimations from these results did not result in a dense enough cluster of galaxies to match observations, today we can use over a billion simulated particles to perform these calculations, based on Newton's Law of Gravity, on a supercomputer to gain a a deeper understanding of the best cosmological models.

These cosmological models require an understanding of the parameters which govern formation at these large distances, including the matter fraction and dark energy content. According to Aghanim et al. (2020), the Universe is roughly 70% dark energy, 26% dark matter, and only about 4% baryonic matter. From their observations of the CMB and others we know the Universe is consistent with a 6 parameter base-ΛCDM, spatially-flat geometry seeded with Gaussian initial density fluctuations as predicted from inflationary models. Primordial small quantum fluctuations of the energy density are excited during inflation and stretched to large cosmological scales. After they exit the cosmic horizon they are frozen until the horizon reemerges at a matter or radiation-dominated epoch and the fluctuations begin to grow into the structures we observe today. From just a few cosmological parameters, an initial density field, and a lot of computing time, we can generate a simulated Universe in a box over 100 million parsecs on each side. We show a comparison between the original Holmberg data and Millennium, a state of the art cosmological simulation, in Figure 1.1. While the improvement speak for itself, it is important to understand that at its heart, both are just simulating gravity, and yet the simulations are so large, we can not possibly store all the data from each calculation, and instead choose to output the state of the box at regular 'snapshots'. While many simulations will have a few dozen to a hundred snapshots, the Vishnu simulation has nearly 1000 snapshots across its history to precisely study halo evolution.

1.2 Dark Matter Only (DMO) Simulations

Despite knowing about all the vast baryonic physics that occurs at "small" scales, which can have an effect on the dark matter field, we can estimate the physics of halo formation to great precision by knowing gravity alone. We know from Zwicky (1937), and substantial research since, that the baryonic matter alone does not account for the majority of stellar motion we see on the outskirts of galaxies. In fact, there must be an enormous amount of undetected dark matter contributing to a galaxies gravitational potential. This mass is so

great, we can approximate our large cosmological simulations using just the dark matter particles in a co-moving box. We start off with an initial grid (cube) of particles and knowledge of several key cosmological parameters, which allow us to evolve the grid according to some gravitational approximation code, such as the Zel'dovich approximation or 2LPT (second-order Lagrangian Perturbation Theory) until it just begins to become nonlinear. This point is a bit subtle and changes depending on several factors, but running the simulation at early times when things are linear is costly and wasteful. Often simulations will start around a redshift of 50 to 100 after the linear approximation. Then, we allow a gravity solver in the simulation to evolve the particles iteratively, which results in the giant filamentary structure formed from clumps of dark matter. This structure is interesting to us because we know these regions are the hosts to countless galaxies and clusters. The approximations for halo growth, if run forward to today, agree fairly well on extremely large scales with the results of DMO simulations, however, simulating gravity has a significant influence on smaller scales. This means we can approximate linear and second order gravity fairly well up until the point at which non-linearity begins to take over, such as in the formation of clusters and halos.

In these regions, slight over densities of dark matter seeded from the initial approximations, have now grown into dense clumps. We know that the dark matter must be fairly 'cold' allowing it to form into relatively dense halos, and it must be 'dark' limiting its ability to interact via the electromagnetic spectrum. This dark matter must also be noncollisional, meaning that clumps of dark matter can pass right each other. While we still do not know precisely what the dark matter 'particle' is, or what new physics it may bring, we understand its effects enough to simulate via large massive particles. At the smallest scales, to prevent wild, non-physical scattering in these now dense regions, we introduce a force softening effect to tame extremely close encounters. At the edges of the box, we use periodic boundary conditions, such that the edge on one side wraps around to the other side, so that we don't induce spurious finite edge effects in our simulated volume. And

at the largest distances, we approximate the gravitational force, such as with an octree, to reduce an order(N^2) calculation, to a more manageable order(NlogN) considering the vast number of particles (Springel 2005). Even with so many particles we are still simulating absolutely massive objects as just a single point of mass. In Vishnu, these particles are on the order of ten million suns, nearly a small galaxy themselves in mass. Even at this huge scale, we can resolve structures of intricately interacting objects, and when we zoom out to over one-hundred million parsecs, we see the vast cosmic web.

1.3 The Galaxy-Halo Connection

Having a huge set of billions of dark matter positions in a simulated box can result in stunning visualisations of the hierarchical formation of our Universe. It is still a few steps removed from what we observe through our telescopes though. In order to link a DMO output to observations we must place baryonic matter into the dark matter 'halo' clumps which host galaxies. We get these halos from a halo finder code that either looks for spherical over-densities, or connects close halo particles via a linking-length, such as in a friends-of-friends algorithm to form individual halos (Einasto et al. 1984). We will often describe a halo by how over dense it is relative to the background density. These halos are supported by kinetic pressure, becoming roughly virialized, in quasi-equilibrium between their gravitational potential and kinetic components. These halos often interact, either through mergers or close encounters leaving a central dominate halo with smaller subhalos, and sometimes they flyby each other disrupting the other.

While it would be ideal to make the ultimate simulation, which calculates every physical interaction of every particle across the entire Universe, this is not only beyond our current computers, it is so unfeasible with current technology, that it could not be attempted. Similar to how thermodynamics relies on the statistics of particles in order to make simplifying assumptions about the over all state, we have to make simpler physical prescriptions that govern star formation and galaxy physics. In the most modern 'hydro-dynamical' cos-

mological simulations we are simulating the smallest 'galaxies' as single hydro particles among other 'dark halos' without any stars. Processes like black-hole formation have to be solved via a prescription, rather than by individually calculating the extreme physics that forms them in reality. These prescriptions require lots of delicate tuning in order to output realistic galaxies. While hydro-simulations produce brilliant galaxies, and can be done on fairly large scales, they are relatively expensive for their size.

We can bypass these unknown or expensive physics prescriptions entirely by utilizing the galaxy-halo connection with a DMO simulation. Instead of attempting to simulate the physics to create known galaxies from baryonic matter, we can simply simulate the dark matter and input known galaxies themselves. This requires detailed knowledge of how to put galaxies into the halos. Initially, we observe that massive galaxies are typically in massive halos, and small galaxies are in lower mass halos. So the simplest prescription to link galaxies to halos is to link the galaxies to halos by their mass. We call this technique 'sub-halo abundance matching' or SHAM. A second technique, Halo Occupation Distribution or HOD, is to create a model of the number of galaxies that exist in a halo based on whether a halo of a given mass will host a central galaxy or N satellite galaxies (White & Rees 1978; Bond et al. 1996; Holmberg 1941; Kravtsov et al. 2004). Traditionally this distribution just relies on a halo's mass, however, extensions can be made such that the model is 'complex' based on having one or more additional properties effecting the population of halos with galaxies, such as the 'decorated HOD' (Hearin et al. 2016). With this information we can match several of the clustering statistics we use to distinguish between potential cosmological models.

1.4 Testing Simulations

There are many different statistics we use in order to compare observations to predictions from simulations. The simplest statistic we can look at is the number density of objects. A universe with a very low mass percentage relative to dark energy, will have rel-

atively fewer dense objects. Additionally we can look at halo population statistics, or void statistics to compare to observations or theory. Additionally, we can compare the spatial clustering of objects over different distance scales to evaluate the accuracy of our models. The two-point statistics of matter have long been used in cosmology to evaluate the spatial clustering of over densities. We expect that large mass halos are going to be spatially more clustered, or biased in theory based on collapse models (Press & Schechter 1974). Simulations give us a rich test bed to compare this theory to results we see from numerical simulations. The hope is that with the right understanding of how bias can manifest itself, we can use the clustering from expansive galaxy surveys in order to constrain cosmology. Finding the right combination of statistics to utilize has been the subject of extensive work (Zheng 2004; Szewciw et al. 2021).

1.4.1 The Two-Point Correlation Function

One of the most useful methods we have for quantifying a halo's spatial overdensity is with the two-point correlation function (TPCF). This is a calculation of the number of halo-to-halo pairs on a given scale relative to the expected number from a random distribution of points. Intuitively, if looking at a small scale, such as on the order 100 kiloparsecs (kpc), a large value in the TPCF would indicate many pairs of objects clustered near 100 kpc. This is not the same as simply having more objects, as this would would also increase the expected number of pairs at random. The TPCF accounts for this by weighting the counted value by the expected value, which can be computed analytically for a periodic cosmological box. Assuming a given number of objects, a larger value in the TPCF indicates that objects are over clustered relative to random. Evaluating this function over a range of radii gives the full TPCF. Often for large scale halo clustering we are mostly interested in larger scales, around ten million parsecs.

In general the TPCF is broken up into two regimes, the one-halo and two-halo terms. The one-halo term is for close distances, typically less than a Mpc, which represents intr-

acluster distances. The two-halo term is for larger distances between separate halos. The clustering from these two terms combine to roughly a power law across many orders of distance. Central halos will mostly exhibit clustering in the two-halo regime, while subhalos have higher clustering over the one-halo term. Whether one cares about clustering over a particular scale will largely be determined by the scope of the research.

1.4.2 Halo Bias

Authors have looked at spatial bias (Piscionere et al. 2015) and velocity bias (Guo et al. 2014), among others in order to attempt to disentangle the complex relationship between galaxy clustering bias and differences arising from cosmological parameters. A more basic problem arises based on the nature of the dark matter halos themselves and how they cluster. Using the TPCF we can easily detect small scale and large scale clustering bias. Over many scales we observe halo bias, where halos cluster more or less based on their mass. We expect larger mass halos to evaluate to larger values of the TPCF due to the cosmological power spectrum and the formation from over-dense peaks and is easily detected. This halo mass bias is well studied in both theory and simulations. We can look at the virial mass of a halo or something correlated with mass, like V_{max}, to see a clear clustering difference in the highest values and the lowest groups.

1.4.3 Assembly Bias

When we control for halo mass bias, such as by fixing the analysis in narrow bins of mass, we observe a secondary bias effect over several properties like concentration, spin, and age (Salcedo et al. 2018). Meaning, that even after controlling for the spatial clustering correlation with mass, properties of the halo itself correlate with the differential two-point clustering statistics (Sato-Polito et al. 2019). This secondary bias can manifest in many different ways including in galaxies. Initially this difference was observed based on differences in a halo's assembly history (Croton et al. 2007) where they removed the assembly bias signal by shuffling the galaxies at random. The idea was extended to refer

to secondary biases (Gao et al. 2005; Wechsler et al. 2006). This dependence on mass over time or secondary halo assembly characteristics, rather than on final mass is what separates this from normal halo bias. This bias is not simple however, and can vary depending on mass itself. Low mass halos below a characteristic mass scale, that assemble a significant portion of their mass earlier have been shown to cluster more closely, than those which assemble later. However, this trend is less clear in the higher cluster mass regime, where the assembly bias signal is less obvious (Sato-Polito et al. 2019). An understanding of assembly bias is key for reliably using our best HOD models.

1.5 Age and the Mass Assembly History

One of the most common ways of characterizing a halo's mass assembly history, is by looking at its 'age', a measure of how quickly it accreted its mass. In order to eliminate effects due to halo bias we often look in narrow bins of mass, and since most halos start at roughly the same time, their average mass growth over time will be relatively similar. Age tells us rather when a halo built up most of its mass. A halo that built up its mass early in the simulation, to then slowly gain at later times is characterized as 'old'. A halo that instead starts off slowly at low mass, only to experience more rapid growth at later times would be considered 'young'. Often times this is a single parameter.

In order to characterize the accretion history several authors have utilized a formation parameter like 'half-mass' age, the time at which a halo has first reached half of its final day mass. Wechsler et al. (2002) used a one parameter fitting function to analytically define the formation epoch. These fits revealed an interesting tale about formation epochs and correlations. They fit for the formation epoch using both data up to z=1 and data from z=1 up to today at, z=0. They found that both partial fits correlated well on average with the formation epochs obtained from the entire history. Interestingly, they plotted the pre and post z=1 formation epochs against each other, neither were correlated with each other (see: Figure 10 of their paper, shown in Figure 1.2 for reference). The later half of the

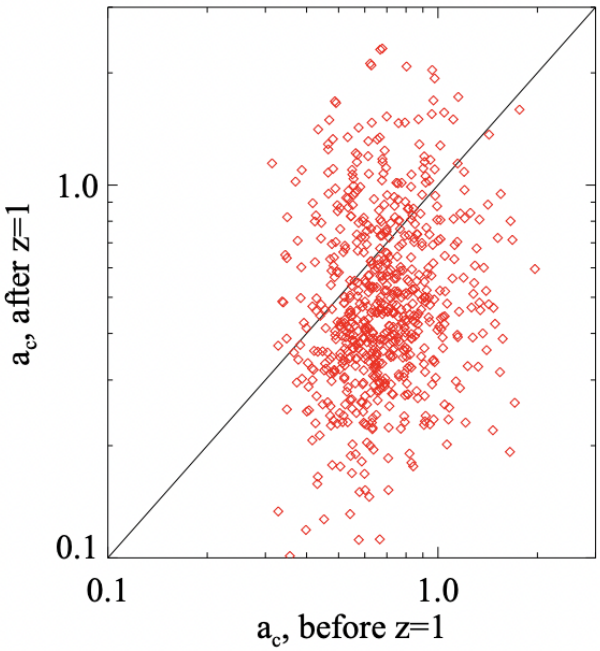

Figure 1.2: Fig 10 from Wecshler's (2002) paper showing the two fit formation epochs for halos greater than ~10^{12} solar masses. The author points out the lack of correlation among the two ages when fit to two different portions of the MAH ('mass growth curves'). Both ages individually correlate with the standard value using the entire history, but they are not clearly correlated with each other.

histories poorly constrain the formation time relative to the earlier history. This is a hint that both parts of a history can provide valuable information about a halo's properties.

In stellar populations, age can have a drastic effect on the look of a galaxy, from red-and-dead galaxies devoid of cold gas needed to form new stars, versus young galaxies bursting with blue star formation. In DMO simulations age does not hold quite the same meaning. The DMO particles themselves do not change their properties nor are they assigned with galaxies or stars throughout. Instead, we observe correlations with age and the halo properties themselves. One of the most drastic correlations for halo age besides the two-point clustering differences is with its concentration, a measure of how centrally concentrated a halo's mass is (Sato-Polito et al. 2019).

Occasionally, instead of using a halo's half-mass age based on final mass, an author may choose to look at the half-peak mass (Salcedo et al. 2018). When looking at a halo's peak mass rather than final mass, the only difference will, by definition, involve halos which have lost mass over their lifetimes. Looking at the peak mass, perhaps we are getting a sense of the mass a halo could have become, had it not lost mass. Typically we would not think about a central halo losing mass. Satellite halos might lose mass as they come in the vicinity of a more massive halo, which could strip away material. When looking at centrals, it is still possible to deal with mass-loss events due to halo finding errors. As halos are not perfect little spheres, but instead are complex arrays of matter overlapping and interacting at finite time steps, it is possible that some central halos could be better defined as satellites, it is also possible that they genuinely lose mass due to energetic merger events and tidal disruption. These distinctions in the age definition can have a dramatic effect on the bias signal. We leave the issue of correcting group finding errors and central mis-classifications for other papers, and assume all halo identifications are correct.

1.6 Additional Halo Properties

When we find halos in a simulation, we are presented with a ton of potential properties we can track. In addition to their mass, a halo finder can output properties relating to physical or historical properties about a halo. Properties can highly correlate with mass, like V_{max}, the maximum circular velocity of a halo in [km/s], or have marginal correlations like the halo spin parameter, a mass (plus additional properties) normalized measure of a halo's total angular momentum. Other properties, like concentration, have very weak (or more complex) correlations with mass. The 'T/U', property measures the absolute value of the kinetic to potential energy ratio of the halo. In many theories of halo formation, we assume halo virialization, a natural stability between 2K and -U, however in simulations this ratio is able to vary. In our sample of halos, T/U generally ranges from just under 0.5 to 0.6, with a small fraction at higher values near 0.8. Some properties are not about the halo itself, but are regarding its assembly history; the 'Number of Progenitors' parameter tracks at each step how many progenitors the host halo had. In most cases, for smaller halos, this parameter is 1, only going above that during merger events. We choose these parameters out of the full list for having potentially meaningful information related to a halo's accretion history, though to get meaningful information from the Number of Progenitors, some additional data pre-processing might be needed. We use the spatial positions of all halos at time z=0 (corresponding to the end of the simulation) for calculating a given population's two-point correlation function (TPCF) described later in the text. There are many additional parameters we did not choose to evaluate including shape parameters and historical information regarding halo satellite population.

We show how these parameters are correlated with each other at 3 different epochs for the primary sample of halos we will be working with in Figure 1.3. These halos are host halos of log(mass) of 11.3 to 11.5. Most correlations become weaker at earlier times, with two notable exceptions being between Concentration and scale radius, Rs, which remains strongly anti-correlated, and V_{max} and the half-mass radius, R0.5, which inverts from a

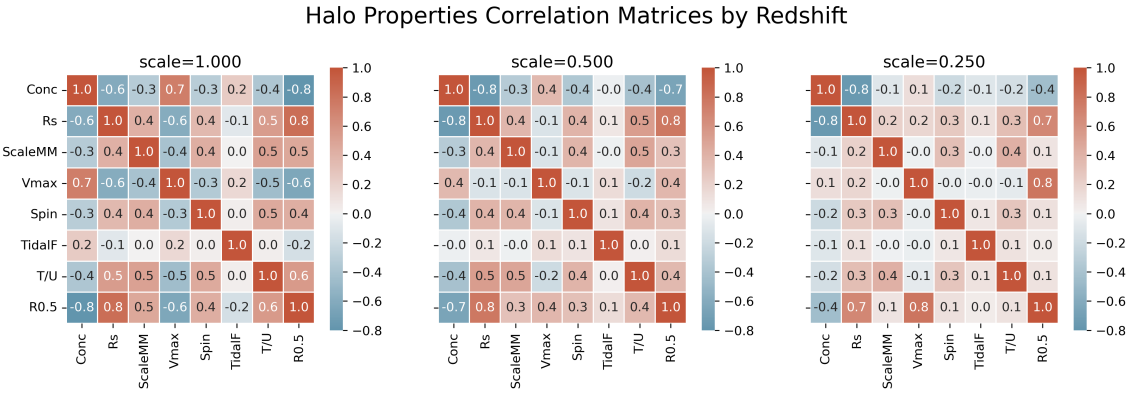

Figure 1.3: We show the (Pearson) correlation matrix among various halo properties at three different scales for our entire sample of halos (final log(mass) = [11.3,11.5]), with the leftmost panel corresponding to the end of the simulation at z=0.

strong correlation at early times, to a strong negative correlation at late times. The relationship between Rs and Concentration is expected from the definition of concentration. The inversion in this population with V_{max} is slightly less intuitive. At late times, objects with larger half radius sizes have lower maximum velocities and higher T/U values. As we are primarily interested in the halo MAH, we were curious about how much these halo property correlations might change if we limited the sample by their MAHs. At each time step we show the same correlations, for only the sample of halos within the 45th and 55th percentile of halo mass, in Figure 1.4. To highlight the differences between the previous figure using our whole sample of halos and this one, limited to the median mass halos, we show the difference for each respective panel in the lower row of the figure. In general, most cells remain unchanged, ±0.1 in Pearson r, despite only using 10% of the halos as in the previous figure. At a scale of 1.0, the left panels, there is very little change, largely due to the tightly constrained mass range already (11.3-11.5). At a scale of 0.5, we see a strong band in the lower row, highlighting the much stronger positive correlation among V_{max} and halo Concentration (and inversely so with Rs), when controlling for the average mass histories. We also see a strong change in the values associated with the V_{max}, R0.5 correlation. When using the entire history we see a fairly strong positive correlation at

Figure 1.4: In the top row, we show the (Pearson) correlation matrix among various halo properties at three different scales for a sample of halos (final log(mass) = [11.3,11.5]) of only those halos which are within ±5% of the mean MAH, leftmost panel corresponding to the end of the simulation, z=0. The bottom row shows the difference between the previous figure (all halos log(mass) = [11.3-11.5]) and the top row of this figure.

early times, however, when controlling for the median mass this correlation turns strongly negative. There is also a slight change in the correlation with halo spin, at around -0.3, and similarly with the 'Scale of last Major Merger' at -0.2. For this analysis we treated each time step independently, however, one could also control for the histories across multiple time-steps together, only accepting those within some percentile of the mean at each time. In the next chapter we explore an alternative way of using the whole halo history to come up with different halo groups and classifications.

1.7 Machine Learning

Cosmological simulations give us so much data that no human researcher could possibly look through it in its entirety. For some big data issues, citizen science can be employed to utilize the ability of the masses to crowd source solutions. For other data, sophisticated filtering pipelines must be constructed to discard the majority of non-useful signals. For some data-sets, we still are not exactly sure what information is relevant and use machine learning to learn about important features. Astronomy is no stranger to such data problems and machine learning has become a powerful tool for reducing mountains of information into more digestible pieces, learning complex relationships among data. At its core, machine learning is the study of mapping inputs to outputs in a data motivated, non-trivial manner. Given a single variate input and a single variate output, machine learning may simply find the optimal linear mapping between x and y; however, it can do much more. Machine learning can additionally embed non-linear or even non-parametric information to better relate the inputs to the outputs. Machine learning isn't just limited to one-dimensional data either. In fact, one of its most useful aspects is its easy adaptability to N-dimensions. Given a vector of inputs of dimensionality, D_1, and length, L, we can use various machine learning algorithms to find an approximately 'optimal' mapping to some set of outputs of dimensionality, D_2, also of length L. The fact that the input and output dimensionality do not have to match means the models are extremely flexible in what type of data they model. A multidimensional input space can easily be partitioned based on uni-variate output space, or vice versa. One example of the former may be learning the relationship between a plethora of customer information and the amount of total money they spend, while an example of the latter may be predicting someones face from an overall GPA score, or predicting various economic indicators based on the current change in stock market performance. The manner in which these inputs are mapped to their outputs and the various ways that prediction errors are handled by each can effect the performance of different learning tasks making it difficult to have a single best architecture. One major

constriction of most models is that the inputs and outputs are matched one-to-one such that one datum input, is mapped to a singular datum output. For some problems, complex feature construction may be required for optimal performance, where the researcher constructs inputs from known data for better performance; other problems (and the corresponding algorithms used with them) allow for the features to be directly fed in without pre-processing or prior feature construction. There is no one gold standard machine learning model for all data problems, so researchers must be mindful of the benefits and shortfalls for various models before relying on them for data analysis.

If working with known inputs and outputs from a training set, we can directly test the performance of each model in training and in a validation set. This type of machine learning is called "Supervised" because there is some truth known to the model during the learning phase. The specifics of the model will depend on the structure of the data-set but regardless of the algorithms used, we can compare the performance of each model through use of a "confusion matrix" which looks at performance of a model versus the known truth values. If evaluating a binomial YES/NO model, the confusion matrix would show four categories: True YES, True NO, False YES, False NO. Objects in the first category were correctly labeled 'YES' by the model, while objects in the second were correctly labeled 'NO'. Objects in the third category were incorrectly labeled 'YES' (meaning they were actually 'NO'), and finally, objects in the last category were incorrectly labeled 'NO'. The performance of any individual machine learning model will depend on how you personally weight each of those four categories. For a science like medical screening, one must weight the impact of missing a positive test versus the impact of incorrectly informing people of a predicted 'positive'. For astronomical object (or feature) detection, one may prefer as high of a 'True YES' as possible if the data-size is fairly manageable, thus minimizing the number of missed objects. Of course there is a natural balance; an algorithm that simply classifies all objects as 'YES' would naturally have a perfect 'True YES' score, at the cost of a terrible 'False YES' score. An algorithm which outputs the same answer every time is

hardly useful, thus, a researcher desiring a high 'True YES' score may set the acceptance boundary such that 99% or more of the true 'YES' objects are correctly categorized. This can generalize to different categories or different boxes in the confusion matrix.

All of these *confusions* are completely avoided if one doesn't know the 'truths'! In this case, a confusion matrix can not be constructed and we rely on *unsupervised learning* to group like vectors. Unsupervised learning can be used to find separable groups in large vector spaces, including images, texts, and recordings. After running an unsupervised algorithm for classification over an input space, each object will be given one (or more) label(s), with similar objects having like label(s). Alternatively, unsupervised learning can be used to map an input space into a lower dimensional vector space. We will utilize both types of unsupervised learning in Chapter 2, along with a discussion of why we can not utilize supervised learning for the purpose of evaluating halo assembly bias.

1.8 Summary

This thesis attempts to leverage the ability of multiple machine learning algorithms and alternative techniques to investigate the contribution of the halo mass accretion history (MAH) to spatial bias at fixed mass. While the primary goal of the research is to better understand halo assembly bias in the context of halo mass growth, we also evaluate how other halo properties may be correlated with data-motivated groups. For each algorithm we try to search over a variety of parameter space while also maintaining a 'manageable' data volume and computational run-time.

A short conclusion is in chapter 4

Chapter 2

MASS ACCRETION GROUPING ANALYSIS

The following work will be submitted to the Monthly Notices of the Royal Astronomical Society Journal and is reprinted below in its entirety

Mass Accretion Grouping Analysis: Using Machine Learning to Investigate Dark Matter Halos

Nicholas Chason[1], Andreas Berlind[1], Christina Davis[1,2], Manodeep Sinha1, 3

[1]Dept. of Physics & Astronomy, Vanderbilt University, Nashville, TN 37235

[2]Belmont University, Nashville, TN 37235

[3]Swinburne University of Technology, Victoria, Australia

2.1 Abstract

We investigate various methods of grouping halo mass accretion histories from a cosmological N-body simulation and the resulting two-point spatial bias of each population. Several authors have used the time at which a halo had accumulated half of its total mass (at z=0) to separate into two (or more) populations - early and late forming halos. As dark matter halos accrete their mass in a noisy manner over most of their history, we explore alternate techniques, using a suite of machine learning methods, to classify halos into 3 groups per algorithm in a data-driven approach. There is great diversity in the groupings of halos by their accretion histories among different methods. We compare the two-point clustering statistics of these groups with the conventional age populations of equal size. We find that most models do not result in groups that are more biased than age; however, a small group of halos, found separately using K-Means and Gaussian Mixture Models, and various mass-loss methods were more biased than equal-sized age-based groups. The

halos in these highly biased populations tended to be older, more concentrated, and were experiencing high tidal forces at z=0. We discuss the implications of data pre-processing and model hyper-parameters on groups obtained via machine learning methods.

2.2 Introduction

When considering halo populations, it is well known both from analytic theory and simulations that dark matter halos spatially cluster differently than the average matter background, a phenomenon coined *halo bias*, more closely mirroring the distribution of galaxies. Furthermore, this clustering depends on mass, with more massive halos exhibiting greater bias (Mo & White 1996; Mo et al. 1996). Since galaxies live within dark matter halos, understanding how halos are biased tracers of the underlying matter field is vital for the future of precision cosmology. An additional spatial clustering bias is found when separating halos by their ages, accounting for the mass-dependent halo bias by looking in bins of fixed halo mass (Gao et al. 2005). This effect, often called assembly bias, could show up in our galaxy observations (Croton et al. 2007) and is therefore important to study.

The concept of 'assembly bias' has been extended more generally to 'secondary bias', to include the dependence of halo clustering on properties beyond age, at fixed mass. The first studies of this bias used halo age to classify halos into early forming and late forming groups, usually based on the time when a halo first accreted fifty percent of its final mass, i.e., its 'half-mass' age (Gao et al. 2005). However, subsequent studies observed strong correlations of halo clustering with a halo's concentration (Wechsler et al. 2006) and other halo properties such as substructure, halo spin, epoch of the last major merger, peak mass, accretion rate and proximity to other massive halos (e.g., Gao & White 2007; Wang et al. 2007; Salcedo et al. 2018).

The physical origin of secondary bias has been studied by several authors. Dalal et al. (2008) found that at the high mass end an age bias is "expected from the statistics of the peaks of Gaussian random fluctuations", while at low mass the signal is largely related to

"halos where accretion has ceased." In their work they compare simulation results to a toy model incorporating halo collapse showing relatively good agreement at the scales where assembly bias is relevant. Mansfield & Kravtsov (2020) studied halo age bias at low masses and found that it is connected to a variety of effects, including the tidal stripping of mass from halos as they pass through larger halos (i.e., 'splashback' halos), and the slowing of mass accretion onto halos due to larger-scale tidal fields. A full understanding of secondary bias that includes halo properties beyond age is more complicated, however, since the dependence of halo bias on some properties, like spin, cannot be simply explained by their correlation with age (Salcedo et al. 2018; Johnson et al. 2019). In addition, single parameters such as age represent a simplistic reduction of a complex merger and mass accretion history. It is this focus on the assembly history of low mass halos and its connection to halo bias that guides our research.

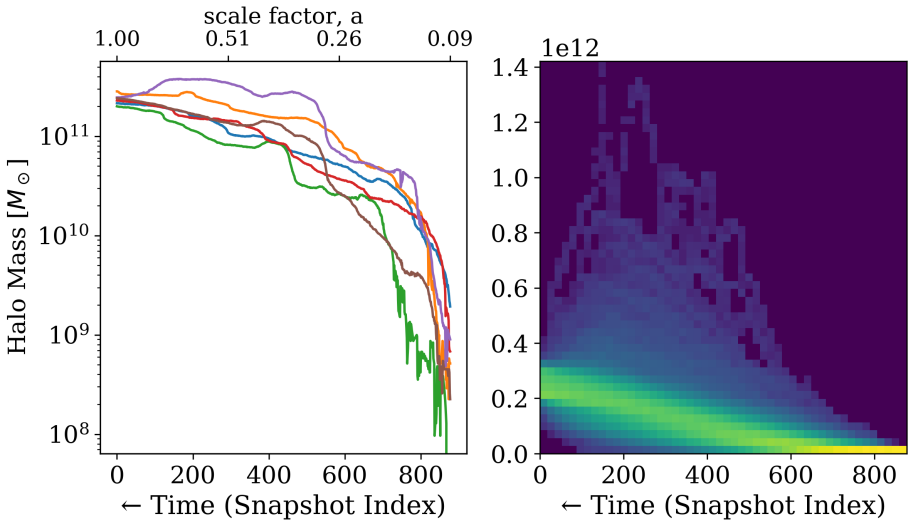

Figure 2.1: Left: A sample of 5 random halo MAHs, with the y-axis showing halo mass on a logarithmic scale, and the x-axis showing the simulation index (bottom axis) and corresponding scale factor, a (top axis). Index 0 corresponds to a=1.0. Right: A 2D histogram of the full MAH data, showing halo mass on a linear scale.

The question we address in this paper is whether using a halo's full history, rather than a single parameter like age, we can uncover a stronger correlation between halo history and halo bias, at fixed final mass. Specifically, we investigate how grouping halos according

to their full assembly history might lead to stronger differences in the spatial clustering of these groups than if we group the halos according to their age. We utilize various machine learning techniques to group the assembly history data into separate classes, attempting to impose minimal assumptions over any specific model. Although halo growth can roughly be described by a one parameter model function and a noise term (Ludlow et al. 2013), with high cadence time series data it becomes apparent that there are several accretion features of individual halos that are not captured by stochastic noise alone. Fakhouri et al. (2010) quantified the large number of mergers that halos of different masses experience. While the features are often associated with the close passage or merger of two or more halos, MAHs can also vary in their smooth accretion rates. These accretion features typically look like sharp hills or valleys imposed on a hypothetical 'mean' history, yet their varied nature makes them difficult to quantify. Some of the features exist only over a few snapshots of time within a simulation, while other features can dominate over the majority of the simulation.

In contrast with the standard halo age model, machine learning algorithms can employ any number of parameters from simple one-parameter linear regressions, to theoretically 'infinitely' parametric (or non-parametric) models. This makes them ideal for exploring a large data set with dense structured time-series data that can not be classified manually. The most common type of machine learning is supervised learning, which uses a training set to learn the relationship between the input features and output targets. While this type of learning has many advantages, it has a major disadvantage for this specific application.

Supervised learning requires knowledge of a specific target value for the training set. This target value can be of any dimensionality, as can the input training data, and the algorithm will 'learn' the relationship between the input data and target output. Since assembly bias is concerned with halo clustering, one target that may seem natural to employ from a cosmological perspective is the two-point correlation function (TPCF), a function describing the over-abundance of spatial clustering among all two-object pairs over a given

distance range. A higher value of the TPCF means that the input object positions, in [X,Y,Z] Cartesian coordinates, form more pairs than a random distribution at a given distance. The TPCF is the main statistic that has been used to describe the clustering of galaxies and halos (e.g., Mo et al. 1996). Unfortunately, supervised targets need to be mapped one-to-one with input values, such that an $N \times M_1$ input list of N objects and M_1 input *features*, is mapped to $N \times M_2$ *targets*, for any M_1 and M_2. The algorithm will then learn the relationship between M_1 and M_2 based on the N data objects available. This mapping is impossible for the TPCF as it is a global property calculated using a population (or sub-population) of 'k' halo positions (which is not known a priori). Meaning, individual halos do not have an individual value (nor vector of values) for the TPCF except in reference to a specific population's measurement. For any single halo, among N total halos, there are $\sum_{k=1}^{N} \binom{N}{k}/N$ possible groups, for any $1 < k < N$, or $\frac{(2^N-1)}{N}$ possibilities. For 1000 halos this is over 10^{298}, and for 10000 halos this is nearly 2×10^{3006} possible halo combinations, making a guess and check approach impossible. Alternatives to this approach using supervised learning and a close proxy of spatial clustering, such as the local density, are not explored in this paper.

In contrast to supervised learning, unsupervised learning classifies objects into groups based on similarity and does not require any knowledge of a target "truth". In our case, unsupervised learning algorithms will place halos into groups with other halos that have similar MAHs. We can then calculate the TPCF of each group of halos to assess its spatial clustering bias. By avoiding focusing on a specific target type, the halo classifications are not inherently biased by our selection of target features and the output groups are a result of MAH data and chosen algorithm only.

Rather than select a single unsupervised technique, we explore several potentially relevant techniques with the goal of capturing time-series or global features that may not have been readily apparent from any one method alone. These algorithms include K-Means, Gaussian Processes, Hierarchical Methods (e.g. dendrogram slicing), Bag-of-Words, and

UMAP, as well as various halo mass-loss descriptions and comparison groups from several different age definitions. As we do not have a 'ground truth', we can not validate the methods; instead, we test the robustness of the results by exploring different methods and by testing several different parameter choices for each method. Since there are so many methods and potential parameter choices for each one, the resulting method outputs are compared via construction of model "archetypes" which combine several methods together to create a new "model" output; this is done mostly as a data reduction technique to make the results from many similar models more interpretable, reducing dozens of individual methods down to just a few archetypes. We also explore some additional models based on observations from our initial model run and discuss how these groups correlate with various halo properties.

The rest of the paper is structured as follows: in Section 2.3, we discuss the data we use, in Section 2.4, we discuss our Grouping Methods and Algorithms, in Section 2.5, we discuss group similarity and creating 'Archetypal' models from our initial methods. In section 2.6 we discuss halo bias and the TPCF, and introduce several additional methods based on the results. In Section 2.7, we show how certain halo properties are distributed based on our groups. Finally, in Section 2.8 we give our summary and conclusions.

2.3 Data

The simulation data used in this paper comes from the `Vishnu` simulation (Sinha), a 130 Mpc dark matter only box with high cadence output across time. The simulation adopted these parameters using a `CAMB` power spectrum: Hubble parameter, h=0.70, the matter fraction, $\Omega_{\rm M}$=0.25, dark energy fraction, Ω_{Λ}=0.75, power spectrum normalization, n=1.0, and the mass density fluctuations at 8 h^{-1} Mpc, σ_8=0.8. Initial positions and velocities were evolved with `2LPT` (Scoccimarro 1998) to z=99, and were run to z=0 using the `GADGET-2` N-body TreeSPH algorithm (Springel 2005) in a ΛCDM cosmology using a minimum particle mass of $3.127\times10^7\ h^{-1}M_{\odot}$ (throughout the rest of the text we simply refer to this as

$M_\odot$ with h^{-1} implied). The particles are grouped into halos using `Rockstar` and halo histories are made using `Consistent Trees` (Behroozi et al. 2013b,a), resulting in 706,060 halos over 20 particles. The outputs of the halo tree files include properties including halo mass, size, rotational velocity, tidal force etc., stored over nearly 1000 time-steps from a scale factor (a) of 0.06 to 1.

While the simulation contains halos ranging from as little as 10^9, to larger than 10^{14} $M_\odot$, we choose a final halo mass range of $10^{11.3}$ to $10^{11.5}$ $M_\odot$. We choose this range so that we have large enough final halos to have well resolved halo growth over most of the simulation, while being small enough to show a strong 'age' assembly bias signal. A halo at the lower end of our mass range, log$M$=11.3, would have over 6300 particles at its final mass. Additionally, we require a narrow mass range, 0.2 dex, for reduced halo bias signal.

When curating the data from the tree files, we first perform a cut on the halo mass range we are interested in. This first cut is to avoid a halo bias signal when looking for a secondary bias signal. We only consider host halos at z=0 and thus cut any halos which are listed as sub-halos of other primaries at the end of the simulation. For the mass range of log$M$: 11.3 – 11.5, we are left with just 11679 unique halos, from over hundreds of thousands of total halos at z=0. Each of these halos histories include all halos above the `Rockstar` size threshold of 20 particles, which have merged together to form the final host halo. At any one time-step there could be several smaller halo mergers with the tracked primary halo or only the primary halo itself. To simplify the data structure, for each primary at z=0, we only consider one halo per history at each time-step; a treatment of the full history may be interesting for future work. Most host halos this mass range exist for around 900 snapshots throughout their history and nearly all for least 830.

We actually explore 3 different ways for selecting which halo to use when multiple smaller halos are present at a single time-step in a history. These are all roughly equivalent at late times, but may have significant variation at early times; nevertheless, which method is used should not significantly impact our results. The simplest way to go through the

halo mass histories from the Vishnu simulation is to take the first entry or primary halo at each snapshot and store the result. The first halo entry from `Consistent Trees` is the 'Main Progenitor' as defined by their merit function (Behroozi et al. 2013b). The main progenitor is initially estimated based on which progenitor halo shares the maximum number of halo particles with the descendant halo. This usually results in the largest halo being selected, however, there are instances where the largest halo at a time-step does not appear first. After the initial guess, the algorithm attempts to refine the guess based on mass ratio and relative velocity to better match the progenitor to descendent (see: Srisawat et al. (2013) for a fuller description of different methods). A second way to build a history, is to track the most massive halo in the tree at any one snapshot, as the "most mass" history. As halos are products of density and collapse, this method has a more physical meaning than somewhat arbitrary labelling conventions and often results in smoother appearing halo histories. Finally, we use the halo IDs from `Consistent Trees` to trace back a history, tracking each halo ID to create a more consistent 1 dimensional history that follows the same 'object' between snapshots. We start at z=0 with the host halo, and iteratively go back to the previous time-step to select the halo that was listed as the predecessor halo to the descendent. If there are multiple halos listed as this parent halo, we select the most massive one, always following the subsequent branch.

These three methods mostly differ at early times in the first half of the simulation, entirely agreeing at later times. The 'most mass' histories are often the smoothest looking, since it is effectively tracking the most massive object in the area of the host halo, which does not oscillate drastically. The first halo and the tracking of the descendants are very similar, primarily with deviations at early times related to the small halo size relative to its interactions. In most cases we are able to ignore this distinction as it does not affect our bulk results; nevertheless, for some individual halos there can be a large difference depending on which tree is used. We use the first method, following the main progenitor, unless otherwise stated. Although not explored in this paper, a fuller description of a halo's mass

history might contain the entire halo tree, with every merger branch included, resulting in vastly different data sizes for quiet versus active histories and much more data overall.

2.3.1 Data Pre-processing

We begin by reading in the raw data as described above. For every halo in our sample, this gives a 1 dimensional vector containing mass at each snapshot at every time our halo existed. We include each snapshot greater than a minimum threshold mass, and impose a floor of $\log M$=7.3, roughly one particle mass, to avoid having large jumps in the data (in log space). Histories which were not yet formed in the simulation will begin with a flat trajectory at the particle floor, as a result each history has the same number of snapshots.

We show a sample of 5 random histories from this sample after initial preprocessing in the left panel of Figure 2.1. The histories begin their lives at relatively low mass, quickly increase their mass by a few orders of magnitude, and finally taper off to their final mass. The approximate scale factor, a, is shown for reference on the top axis. Even with just a few histories it is easy to spot a variety of features across time, which runs from right to left in the plot. None of the histories shown are perfectly smooth, but the features vary by halo from smooth bumps, to large valleys and jagged mass change events. In the right panel of Figure 2.1 we show a 2D-density map of all the data in our sample. Most of the data falls into a narrow range highlighted by the density map, however, faintly one can see a large scatter with relatively few histories above the typical mass per scale factor. This scatter also extends below the average histories though it is not clearly seen due to the linear scaling of the plot.

Once complete the final data size is quite moderate, given 11,679 halos and 870 time steps each, however, it is approaching the size that can start to limit many algorithm's repeatability. For these reasons we do not always use the full 1D histories, instead reducing the size to some fraction of that by using a sub-sampling/interpolation process. For many algorithms the sub-sampling has a minor effect while for others it can drastically alter the

results. We were forced to sub-sample down to 300 data points for some Gaussian process methods in order to get convergence, otherwise sub-sampling was only done in order to test limits in the method.

We must additionally choose which 'scaling' to use for our data for the machine learning input. We often consider the halo masses in terms of log(Halo Mass) in solar mass units to avoid spanning several orders of magnitude in numerical values, however, this decision alters how the data is 'seen' by several algorithms. We distinguish between log and linear inputs when using the mass histories as features, (although it should be noted, they are not always plotted with the same scaling as to how they were found). Log input will more equally weight early and late times (by putting masses at 10^7 on the same order as those at 10^{12}), while a linear scaling will place more emphasis on higher mass values and thus later times. Most of our models use log scaling, though we do use some linear input models.

Next, we can standardize the data using a few different transformations. Although many algorithms can evaluate data that isn't centered on zero or normalized, it can be useful to place all values over roughly the same range when comparing separate quantities or quantities that can span several orders of magnitude to avoid numerical issues simply related to numerical scale. While MAHs are only 1D, only changing with time, this normalization is especially relevant if one doesn't wish to weight values based on the numerical value alone when evaluating dimensions containing different units. Whether the data is initially logged or not, we can remove the global trend over time, or we can normalize the 'power' in each history.

To do this transformation in python we use Sci-Kit Learn's `preprocessing.normalize` feature (Pedregosa et al. 2011a). When dealing with a time series vector, the normalization axis determines over which dimension the data is integrated to determine the normalization factor. We use both normalization axes (0 and 1), as they result in slightly different information. A normalization over the zeroth axis flattens the long term trends of each history showing the local variance between halos at each time. While a normalization over the first

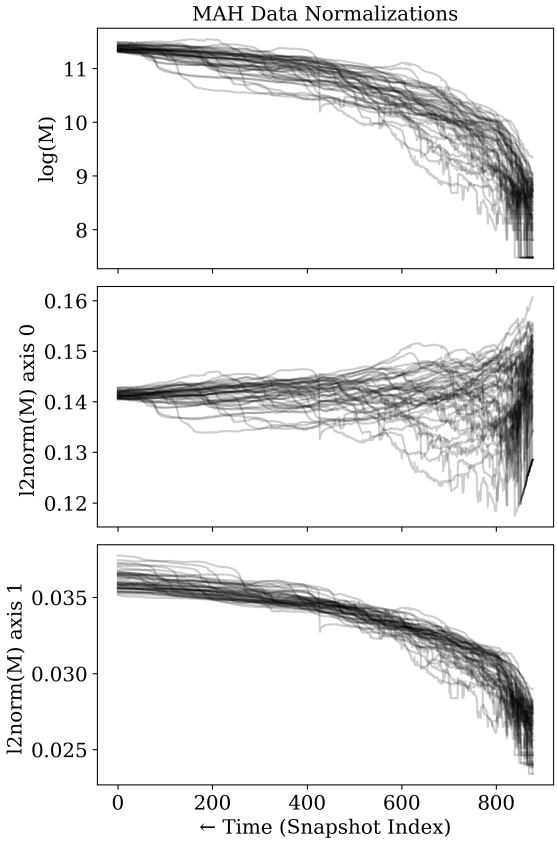

Figure 2.2: A random sample of 25 halos showing 3 different MAH normalizations. Top panel: Raw log$M$ data. Middle panel: Normalized data that removes the overall mean (l2-normalization over axis 0 of the data). Bottom panel: Normalized data that preserves the overall mean trend, while standardizing the variance across time (l2-normalization over axis 1 of the data).

axis maintains the rough shape of each MAH, dividing each history by its 'power', effectively normalizing the variance at each snapshot. This power is computed by the l2-norm, which is based on the euclidean distance of each point from the origin. These different normalizations are shown comparatively in Figure 2.2. Although relative features are preserved in the normalizations between histories, the overall shape and scatter of the data changes dramatically. Whether or not these changes would influence clustering results will depend on the specific machine learning algorithm.

The normalizations discussed above preserve the dimensionality of data, however, one could also use Principle Component Analysis (PCA) in order to transform and/or prune the MAH data (F.R.S. 1901). Using PCA as a dimensionality reduction technique can

minimize the dimensions of a data set while preserving the maximum amount of variance in the data. We use PCA in this manner for one of our K-Means models to reduce the size of of the data from 878 time-steps to just a dozen principle components, preserving ~95% of the data variance in the process. A separate type of data prepossessing will be discussed in the 'Archetype' section 2.5.4.1 when we discuss the use of One-Hot-Encoding to eliminate the effect of having non-ordinal vectors for cluster analysis.

With cyclic data or repeating time-series sequences, sometimes it is more interesting to use 'dynamic time warping' (DTW) procedures to modulate or shift the data before computing the summed squared difference or similar local distance measure (Berndt & Clifford 1994; Meinard 2009). The ability to shift the time input to reduce the difference function comes at a run-time cost of quadratic complexity, although domain restrictions can significantly reduce this space. DTW has been used successfully in signal processing and as a feature for more traditional machine-learning methods for decades.

The technique is very useful for matching segments in data, such as in the area of gene expression, or speech recognition where data is not cleanly synced up in location or time (Aach & Church 2001). For example, if you had two identical sine-waves 180 degrees out of phase, a normal difference would suggest very different curves, while a time warped sine wave, could realign their phase showing a high similarity. For certain halo histories, which have a significant feature displaced from the mean but slightly offset in time, DTW could theoretically shift the features in place, to give a more representative difference function with a comparison history. In practice, it is not clear for the majority of halo pairs that DTW is actually picking out reasonable features to shift to, nor how restricted the time shift domain should be. Each history pair requires a separate DTW calculation over the entire history becoming very computationally costly. For simplicity and due to the non-repetitious appearance and nature of the majority of MAHs, we choose not to utilize dynamic warping in this paper. See Izakian et al. (2015) for an in depth discussion regarding DTW for time-series data classification.

2.4 Grouping Methods

In this paper, we use multiple machine learning methods to group halos together based on the similarity in their MAHs. In order to compare these machine learning models to halo groups obtained by more conventional methods we will use 3 primary types of methods for classifying halo MAHs into separate groups: Age, Mass-Loss, and Machine Learning. The age methods are based on existing definitions of age, adapted to fit this work and extended to utilize more of the halo history. Mass-loss methods were created to try to capture the amount of mass loss experienced by a halo. We believe this may be an interesting representation of the MAH due to the strong bias signal associated with 'arrested development' halos which have ceased growing as quickly, or have even lost mass, due to a proximity to another massive halo. Finally, we explore unsupervised machine learning methods which attempt to find natural groupings using a data driven approach, rather than from a direct parameterization or functional representation.

2.4.1 Methods Pertaining to Age

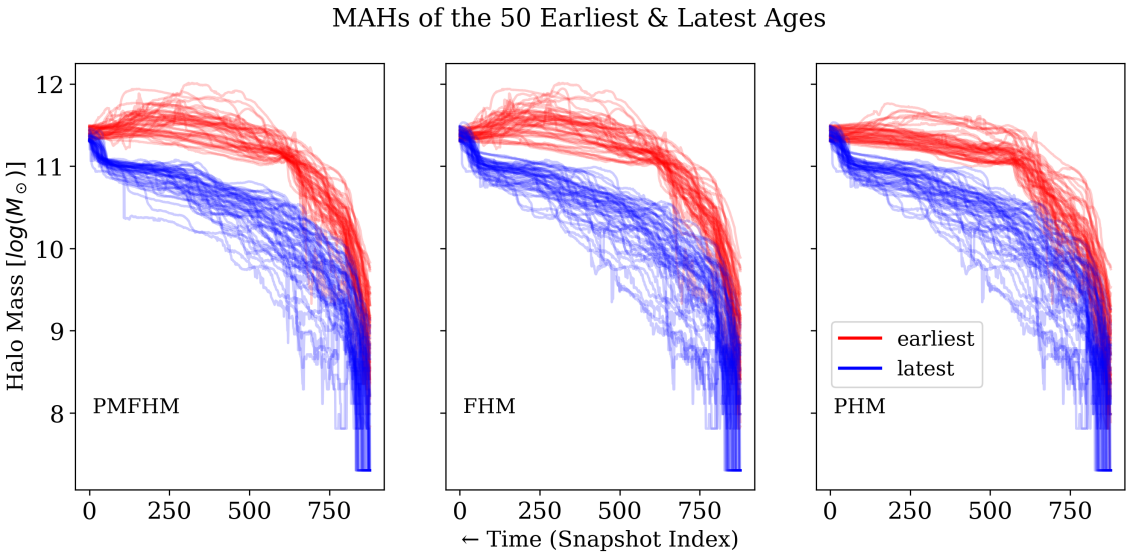

Figure 2.3: The 50 earliest (red, top histories) and latest (blue, bottom histories) halo MAHs using the 3 definitions of Half-Mass age, as labeled in each panel. From left to right: PMFHM (Population Mean Final), FHM (Final), and PHM (Peak) Half-Mass Ages.

One common way to evaluate halo assembly bias is through halo age. The age parameter is often designated the “half-mass" age, meaning it corresponds to the time at which a halo first obtained half of its final mass. Another popular age definition uses half (or some fractions of) the halo’s peak mass, rather than its final mass with qualitatively similar results. A halo that accreted half of its mass at an earlier time, meaning at a larger redshift (or smaller scale factor), is earlier forming or ‘older’, compared to late forming halos or ‘young’ halos which obtained their mass later. Using metrics like the TPCF, older halos have been shown to be more spatially clustered than younger halos (Gao et al. 2005).

Our “Zeroth" model, which will be used as the basis for comparing all additional models, is:

- **Model 0. Population Mean Final Half Mass Age (PMFHM):** Halo groupings are split into 3 equal divisions based on the time at which each halo had first accumulated at least half the ‘final mass’, where final mass refers to the sample’s average mass at z=0. Parameters: $f_{age} = 0.5$. The fractional age value computed in linear space. Ex.: age = t($f_{age} \times 10^{11.4}\ M_{\odot}$), where t(M) is the time at which a halo first reached M mass.

There is one subtlety we consider with the basic age definition regarding which mass is considered. Usually the fractional mass quantity is taken to be some fraction of the halo’s mass at some specified time; however, since we are only considering a narrow bin of final halo mass, we can approximately use a single, population mean final mass definition in place of each individual’s final mass. For the halos in the mass range of $\log M$: 11.3 – 11.5, we set the single final mass value to the average of all the primary halos in that range (roughly 11.39). One benefit of using the PMFHM value, is that the ranks are determined when each halo crosses a single mass threshold, regardless of its ultimate final mass. Additionally, in model 1, our ‘first’ model, we compare the rank ordered halo’s typical, individual final half mass (FHM), and in model 2, the ‘second model’ we use the rank

ordered peak half-mass (PHM) ages to define our groups.

- **Model 1. Final Half Mass Age (FHM):** Groupings are formed based upon the rank order values of the time at which each halo had first accumulated at least half of its final mass value at z=0. Parameters: f_{age}= 0.5
- **Model 2. Peak Half Mass Age (PHM):** Groupings are formed based upon the rank order values of the time at which each halo had first accumulated at least half of its *peak* mass value. Parameters: f_{age} = 0.5

We show an example of these first three methods in Figure 2.3. At first glance the PMFHM and FHM panels look nearly identical, although, we will show that the FHM and PHM methods match better overall, since for most histories peak mass is equal to final mass. Comparing the three panels, focusing on late times with z=0 (corresponding to a simulation index of 0), the FHM and PHM groups are spread out roughly evenly across final masses, while the PMFHM earliest and latest groups have a more clear dependence on final mass. This dependence shows that despite the stochastic nature of MAHs throughout their histories, there is still a slight correlation with a halo's final mass and its half mass, even in our narrow range of final mass.

2.4.1.1 Extending Half-mass formation

When evaluating the standard age definition, which considers a singular fractional mass for each halo, a small but noticeable artifact appears in Figure 2.3. Most easily seen in the top left panel, the oldest halo histories appear to pinch at one point, while the youngest have a late time upturn. This is a result of the histories being fairly noisy and the age definition being fixed at a singular value, considering the first point above the fractional age even if it is a spurious jump. A sharp increase that happens to take a halo over the set threshold will result in an earlier than otherwise expected age, even if this increase was only briefly

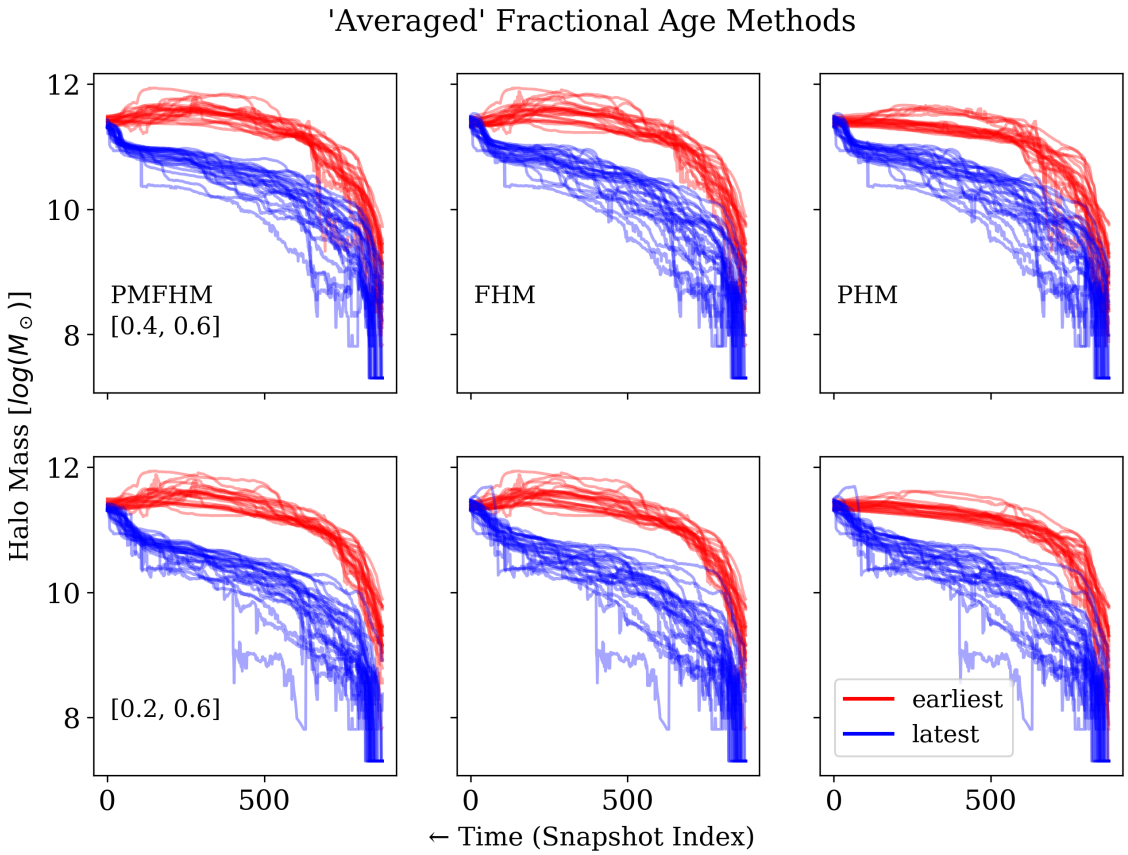

Figure 2.4: Similar to Fig. 2.3, but instead of using 'half' mass age definitions, we assign halo ages by averaging over a range of *fractional* mass age definitions. The top row of panels consider the range of 0.4–0.6, while the bottom row of panels consider the wider range of 0.2–0.6. By taking into account a halo's history over a wider range of times, these age definitions are less sensitive to spurious fluctuations in the history.

reached in a single snapshot before decreasing in mass for several other snapshots. If a halo does not reach half of its mass until very a late time, as we expect from the latest forming halos, then that halo will necessarily experience a period of relatively rapid halo growth at late time, to reach its final mass. In Figure 2.3 we show that each of the three conventional half mass methods based on conventional age definitions show significant halo growth in the latest forming halos, and a noticeable over-dense 'pinch point' in the earliest forming halos at a time index of around 600.

To smooth out these artifacts, one can consider a range of fractional ages. For any one fractional age definition, we can find the age for all of the N halos, which will give each

halo a corresponding age rank 1 to N. To look over a range in fractional ages we choose a minimum fraction and a maximum fraction and find the relative ages for a range of fractions in between them. We subdivide the fractional space into 5 bins, linearly spaced in 'a', and compute the age rank for each. For each halo we average the 5 ranks, and then rank those averages to give an average rank order across the 5 fractional ages.

The top row of Figure 2.4 shows how averaging over a range of 40% to 60% final or peak mass still results in noticeable features around the relevant timescale. For our averaged age models we choose a range from fifth-mass to sixty-percent-mass [0.2, 0.6], to emphasize mid-history growth. For our sample, a sixty-percent-mass upper threshold is roughly the largest mass value that can be explored such that each halo will have a valid age for all our definitions of age. Going too much above a fractional value of 0.6 can result in some halos never reaching such a value when using the single final definition of age (given a primary halo log mass bin of 0.2 dex: [11.3, 11.5]). As an example, if we use the average final mass of roughly 11.4 and then tried to find its 90 percent mass, we get a fractional mass of roughly 11.35. As a result, a halo at the bottom of our final mass bin, that has a final mass of around 11.3, may have never achieved the 90 percent fractional mass value of 11.35. This is one motivation for using each halo's individual final mass (FHM), rather than a group average final mass (PMFHM) in determining the half mass.

Once we have a rank ordered list, we group the halos according to their percentiles in the list; to get three groups, we split on the one-third and two-third percentiles, yielding equal (or nearly equal) group sizes.

- **Model 3. Averaged Population Mean Final Fractional Age:** Groupings are formed based upon the average rank order values for each of the N_{ages} fractional mass ages, at which each halo had accumulated at least 'x_i' fraction of the population mean final mass value, with 'x_i' being linearly spaced between f_{min} and f_{max}, N_{ages} times. Parameters: $N_{ages} = 5$, $f_{max} = 0.6$, $f_{min} = 0.2$

- **Model 4. Averaged Peak Mass Fractional Age:** Groupings are formed based upon the average rank order values for each fractional mass ages, at which each halo had accumulated at least 'x_i' fraction of its *peak* mass value, defined similarly to the previous model. Parameters: $N_{ages} = 5$, $f_{max} = 0.6$, $f_{min} = 0.2$

Several groups have noted that age populations can become more biased when considering a more extreme percentile, and thus more extreme halos. So far all of our groups have been evenly sized, however, several of our later models do not give equal groups. To give some estimation of this effect we use the group sizes from one of our K-Means runs, discussed later in Model 5, to split the ranked age list into unequal group sizes.

- **Model 28 - 32: Age with unequal group sizes:** Models 28 through 32 are identical to Models 0-4 respectively, with the exception that they use group sizes obtained from Model 5: KMeansLOG in order to evaluate the effect of changing group sizes. Parameters: see Models 1-5 respectively.

Note that these groups' 'Model numbers' are higher as they are added after constructing our machine learning models since they are based on K-Means group sizes.

2.4.2 Methods Pertaining to Mass Loss

As we know that large halos undergoing mass loss should be in the vicinity of other large halos, or else their gravitational potential would attract its dark matter towards itself, it stands to reason these halos may be in more populated environments. Dalal et al. (2008) mentions the concept of "arrested development" halos and found that the oldest low-mass halos tended to live in hotter environments, preventing accretion. This idea can be extended to find an 'arrested development' population, where halos have lost mass from their peak (or have begun more slowly accreting (Salcedo et al. 2018)), with clustering perhaps more

similar to the the population of halos which followed a similar MAH up until the mass loss event. In this case the early arrested development history would actually be more similar to halos of a larger final mass. (Wang et al. 2007) suggests that the relationship between halo environment and the reduction in mass is largely produced by the hot tidal fields of nearby more massive halos and the observed clustering bias is entirely due to differences in the initial density peaks. Since more massive halos are biased to be in more dense environments, an arrested development halo may be more likely to live in these environments, potentially inheriting the enhanced clustering or secondary bias.

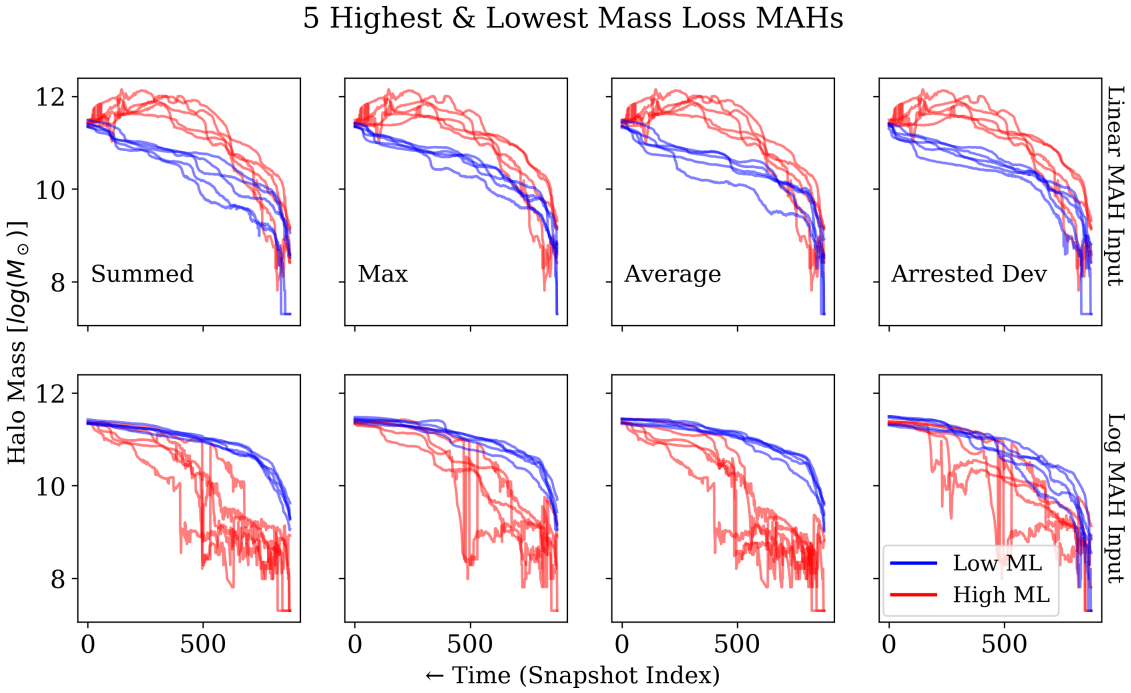

Figure 2.5: A comparison of the most extreme mass loss halos (red) with the least mass loss histories (blue) for our four mass loss models shown in the four columns of panels. In the top row we show the results of running the mass loss methods on the linear mass accretion data M ~[10^8, 10^{12}] $M_\odot$, while in the bottom row we first logged the mass values log$M$ ~[8, 12] $M_\odot$. Both rows show the resulting histories in logarithmic space.

In the spirit of alternative halo classifications, we focus on the concept of MAH mass loss. While we often think of mass loss as a halos where $M_{final} < M_{peak}$, in a time series, halos could have several mass loss events where the mass goes down locally before rising again. If halo interactions throughout the history are significant, a large mass loss

event at early times could be more significant than a small mass loss event at later times. Additionally, just using peak mass - final mass would not cleanly yield a rank ordered list for all halos, as many halos experience no peak-final mass loss as their final mass is their peak. While one could make all halos with a peak-final mass of 0 one group and divide up the other population into two groups to maintain 3 groups, we avoided this method initially, and revisit it later in the paper. We instead choose several different definitions for mass-loss halos which were chosen to be easily defined and easily computed.

The first method is the 'summed mass-loss' where we sum over all adjacent time-steps which have experienced a decrease in mass and rank the halos from greatest to least. The second method is the 'maximum mass-loss' where we store the largest single mass loss event, and rank the halos. The third method is the 'average mass-loss' which takes the sum from the 'sum-mass loss', and divides it by the number of mass loss events for each halo history. Finally, we consider the 'arrested-development' method which is the most involved. In this method we seek to sort halos by their maximum mass-loss event spanning a given 'arrested-development' time scale. The benefit to this method is that it ignores short term transient mass loss events, while searching for events over a wider, fixed time window. While most halos have peak masses that are roughly equal or directly equivalent to their final masses, we defined an initial arrested development population which had a noticeable drop in mass ($\log M_{peak} - \log M_{final} > 0.1$). We estimate a timescale of mass loss, T_{ML}, based on this population, by averaging these histories and finding the time from z=0 to this average peak mass. This time scale gives us a sliding window to search over for mass loss events. We look over each history to find the maximum mass loss value considering this time-scale, and then rank the halos from greatest to least and divide the space into even percentiles. By keeping the number of snapshots fixed for T_{ML} across time, the time in Gyrs will vary depending on where the maximum mass loss event occurs. We find a T_{ML} roughly equal to 150 snapshots or 4 Gyrs.

One must choose whether to evaluate mass in its linear units spanning orders of mag-

nitude, or over its log values spanning roughly 8 to 12. As we will discuss more in depth in the next section, it generally advised to log values that span several orders of magnitude or if the data vector values are composed of different units; the choice can be less obvious when it's a single variable time-series. We initially computed these mass events in logspace, however, we ultimately choose to calculate these models in linear space as extremely early mass scatter was dominating over the second half of the simulation mass loss. We show in Figure 2.5 the most extreme mass loss events found for our different mass loss methods, and note that these most extreme mass loss histories differ wildly between log-space and linear-space input. For each of the four methods we show, more 'classical' mass loss events can be easily observed in the top row of panels, in red, showing the 5 most mass loss halos in contrast with the lowest mass loss halos (blue). In the bottom row, showing the log MAH input, there are still mass loss events, but mostly in the form of drastic jumps in the 8-10 log(mass) range in the first half of the simulation (snapshot indices > 500). Assuming that we wanted to further emphasize high-mass mass loss events, we could transform the linear mass by raising it to some power greater than 1. When raising the linear mass by the power 1.5, we found even more late time mass loss events as expected; however, we did not investigate these input transformations any further and use the untransformed linear mass for each of our mass loss models.

- **Model 24. Summed Mass Loss:** For each MAH, we sum each mass loss event, defined as any event where mass, $M_{t+1} < M_t$, where t+1 is one snapshot closer towards z=0. Halos are then ranked according to their summed mass loss values and are split into 3 equal groups according to their rank.

- **Model 25. Max Mass Loss:** For each MAH, we find the largest single mass loss event, defined as the most negative event where mass, $M_{t+1} < M_t$, where t+1 is one snapshot closer towards z=0. Halos are then ranked according to their maximum mass loss value and are split into 3 equal groups according to their rank.

- **Model 26. Average Mass Loss:** For each MAH, we find every mass loss event, defined as any event where mass, $M_{t+1} < M_t$, where t+1 is one snapshot closer towards z=0. Halos are then ranked according to their average mass loss value and are split into 3 equal groups according to their rank.

- **Model 27. Arrested Development Mass Loss:** For each MAH, we first find a timescale for finding mass loss events. Roughly the timescale the average mass loss halo loses its mass (150 snapshots, or roughly 4 Gyrs). We then look for the largest mass loss event over that timescale, defined as any event where, $M_{t+T_{ML}} < M_t$. Halos are then ranked according to their largest arrested development mass loss value, and are split into 3 groups according to their rank.

2.4.3 Unsupervised Methods

We can employ unsupervised machine learning to aid in clustering our MAH data into separate groups. Clustering is a machine learning tool meant to aid a researcher in "obtaining qualitative and quantitative understanding of large amounts of N-dimensional data". The goal is not to find a "unique, definitive grouping" but rather to provide "reasonably good similarity groups"(MacQueen 1967). In contrast to the single parameter age models or our mass loss definitions, the goal with clustering is to utilize the entire halo history when grouping like halos. While unsupervised learning can be extremely powerful for large data-sets that can not be manually explored, there are many different algorithms and potential implementations, each of which can affect how the data is weighted, with few concrete rules about which to use for any given situation.

2.4.3.1 Parameter Estimation

Unsupervised machine learning methods can be used for classification purposes when a training set is unavailable or when appropriate target labels are not known. Given an input data set composed of X items with Y features, an unsupervised classification method

will group together similar items according to some base algorithm. We wish to use these methods to group together similar halo MAHs based on a variety of parameter choices. We will then use these groups to evaluate the group bias relative to the fiducial groups based on half-mass age.

Nearly all machine learning methods require parameter choices to be made. Some of those choices are directly necessary and others are latent or hidden. Unsupervised methods are unsupervised in that their learning does not require reinforcement through known samples which contain the "truths"; nevertheless, there are parameter choices that must be made. In the case of the standard K-Means algorithm, one must choose how many clusters to initially seed, in addition to a choice (usually random) about where the initial seeds should be placed. Other choices can impact the results in less direct ways - in the t-SNE algorithm one may alter a perplexity and learning rate (parameters which affect the locality of comparisons and how the algorithm explores around minima) which can drastically impact the overall results. Exploring over the full range of parameter space is impossible; however, we explore 5 'base' algorithms with a variety of different hyper-parameter choices for each, detailed in the following sections.

2.4.3.2 Exploring classifications with different K

All classification algorithms we explore require a choice of 'K', the number of separate groups. Each of them can be extended to an arbitrarily high number of groups, with some limitations on group size when using agglomerative hierarchical methods depending on the specific linkage structure. Initially, we designed our code to handle an arbitrarily high number of groups (K) when grouping the MAH data. This was done so we could use the Bayesian Information Criterion (BIC) (Neath & Cavanaugh 2012; Hofmeyr 2020) or similar to evaluate the performance of each of the groupings, to determine the appropriate K to use.

When allowing K>3 groups, we ran a several tests utilizing K-Means and the raw MAH

data. Looking over groups up to K=50, we evaluated the BIC and found that it favored higher number of groups usually larger than 20. Although the BIC is often a good measure, especially for cleanly separated data, in this case 20+ groups was beyond the scope of this analysis (Primarily, we were concerned with the interpretability of comparing more groups which would make ordering the smaller sub groups less meaningful and more challenging.) At lower numbers it did not appear as though there was a natural sub-grouping with fewer clusters as might be found with hierarchical clustering. We will discuss some additional methods for determining 'K' in the following section with regards to K-Means.

For making comparisons between machine learning methods, extending beyond 3 groups becomes challenging, although 4 or 5 group analysis should be feasible. If one were instead only looking for interesting groups within a specific method, or a small subset of methods, higher group numbers may be more useful. If we explored higher K values, the algorithms might ultimately be able to pick out more defining features of the histories, allowing for more interesting subgroups; however, as intra-group variation begins to dominate over the distance between separate group means, any further subdivision is unlikely to yield meaningful populations.

As a result, we choose K=3 to be consistent among all of our methods while significantly reducing the possible space of assignments. This choice has the added benefit of allowing direct comparisons to the age analog of 'early, moderate, and late' forming halos. For the remainder of the paper we always use 3 groups when classifying halo from MAHs.

2.4.3.3 K-Means

A popular choice when first exploring unsupervised learning is K-Means due to its simplicity and ability to quickly subdivide many different data geometries even in high dimensional spaces (Hartigan & Wong 1979). Standard K-Means considers all elements of the data in every iteration and assigns a single label (or classification) to each input object. The resulting assignments partition the space into Voronoi cells, where each data object is

grouped into one of K regions, according to its distance relative to each region's mean.

K-means works by initially seeding K random vectors in the input feature space. The algorithm then assigns each data object, x, to one of the K 'seed vectors', μ_k, minimizing their relative proximity, given by the *inertia*, $I = (x - \mu_k)^2$. With each data object assigned to a group, the vector 'means' of the group objects become the new seed vectors, and the algorithm iterates until convergence. This naturally migrates the mean vectors closer to locations of higher density and divides an input feature space into K regions with hard assignment boundaries. In our case each vector is a MAH, composed of a mass for each time-step. Two histories would be considered close (i.e. have a low interia), if they had similar masses at similar vector indices.

We use the K-Means as a stand alone unsupervised learning algorithm for grouping together similar MAHs, but it can also be used in complement with other techniques to group halos, as discussed in the following section on Bag-of-Words. K-Means can separately be used to group together similar sets of groupings - meaning once we have all of the outputs from each algorithms (including K-Means itself), we can use K-Means to group together similar algorithms - discussed in the Archetype subsection.

While standard K-Means runs quickly on a data set of our size, a faster Mini-Batch K-Means is also available that sub-samples the input data during optimization, which can be used on much larger data sets. Rather than using every data object for each iteration, Mini-Batch will run the optimization process using a random subset of the objects, only assigning the entire sample at the end. Additionally, the K-Means algorithm is not limited to hard clustering; in fuzzy K-Means, the assumption that each input data entry has one output label is relaxed to allow for multiple weighted group assignments per object using soft clustering. Unless otherwise noted, the data considered here uses the standard K-means implementation, using scikit learns "kmeans++" smart seeding algorithm for the initial seed vectors.

We use the Sci-Kit Learn K-Means algorithm (and the SciPy Kmeans2 algorithm)

(Buitinck et al. 2013; Virtanen et al. 2020) which requires a selection of 'K clusters' or groups to initially seed the input space, and a choice for the inertia, for which we use the summed squared error or SSE. All other parameters are set to their defaults unless explicitly stated. As mentioned previously, if a natural selection of K is not known, there are a few different ways one may attempt to determine an appropriate value. Unfortunately all of the methods require running the K-Means algorithm with each sample K, ideally several times, and comparing the results. While visual inspection should be considered first, looking at high-dimensional data can be difficult given 10,000+ objects. There are more systematic ways for validating the number of groups, such as the "elbow" method, the Gap Statistic, the Akaike and the Bayesian Information Criterion (AIC & BIC respectively) - which try to determine an appropriate K given the output distributions. Another method not explored for this data set is Jenks natural breaks optimization that was designed for finding small numbers of groups (<7) based on the SSE deviation of group means versus the population deviation (Jenks 1967).

The elbow method is a simple estimation by eye to determine a 'sharp bend' or kink, when looking at the inertia over a given range of K. Since the inertia is just the intra-cluster sum-squared distance, as the number of clusters goes up the overall squared distance tends to zero. Assuming the line chart looks like an arm, then the 'elbow' on the arm is the value of k that is the best. For data that is not well clustered, the elbow method often fails. There was no definitive elbow feature when evaluating all of the MAH data so we did not attempt to further quantify the 'elbow'.

The Gap Statistic (Tibshirani et al. 2001) compares the evidence for K groups, using the intra-group SSE distances (*inertia*) compared to an 'expectation distance'. The highest Gap Statistic value corresponds to the most likely K. Although the statistic works consistently for well separated groups that are each centrally concentrated, the statistic can 'fail' by yielding K=1, while multiple different feature types are known or desired, or by producing a far higher K number of groups than are expected (especially common in poorly separated

clusters (Dudoit & Fridlyand 2002)). There are at least two options available if finding oneself in such a predicament - the most intuitive is to choose a value of K that seems reasonable for the given task. Alternatively, one can run K-means again on the cluster centers themselves to form a sort of hierarchical two part clustering. As stated previously, we choose the former selecting a value of K=3. Choosing a value of K, that is different than the Gap Statistic's 'suggested' value, implies that the intertia is not minimized over all possible K, however, the K-Means algorithm should still be minimized (at least locally) for a given K.

There are limitations of the gap statistic depending on the nature of the data itself. Yin et al. (2007) reported that the Gap statistic is not able to estimate the number of clusters accurately when the observations in one cluster exceed six-fold the number of observations in others, conditional on cluster separation. As we will see, the input normalization can dramatically effect the cluster sizes found, and thus the gap statistic itself can be very sensitive to choices made regarding the input. Additionally, the curse of dimensionality shows up in evaluations of the Gap statistic, as noted in R.Bellman (1961). They found that among 2D data, the gap statistic could positively identify the number of clusters, however, among their 100D data, the statistic was a strictly increasing function unable to discern known clusters. In Mohajer et al. (2011), they evaluated using a weighted gap static, including computing a null reference distribution which was better able to recover clusters in high dimensional data, however, this was at the cost of identifying the proper number of clusters in more overlapping data.

We found that the Gap Statistic was inconclusive, yielding high numbers of clusters with different values depending on the specific run of K-Means. Two separate implementations of the Gap Statistic yielded K values well into the dozens, which results in some group seeds containing zero assigned halos. This would suggest that the recommended K is too high. Depending on one's goals, a high K can be desirable for finding small local clusters of unique objects.

Similarly, we did not find that AIC nor BIC gave a conclusive answer for the number of groups. The AIC and the BIC are often used interchangeably as both are penalized-likelihood criteria, nevertheless the AIC attempts to find the model that most adequately describes some unknown, high dimensional 'truth', while the BIC assumes that one of the model attempts is the correct model. We find a smooth increase in the BIC evaluated all the way to K=80. For comparison, we also evaluated a random group of points equal in size and structure to our MAH data to confirm the BIC falls off with K given a single group origin. These results suggest there is not a "natural" number of groups for halo MAHs.

Moving forward, when classifying MAHs, we consider K=3 for ease of interpretation. Analysis with just 3 groups can be difficult as which 3 groups are found can dramatically depend on data scaling and normalization, as we will discuss in detail when explaining the UMAP algorithm. There is potential for exploring higher values of K in future work, which could be interesting for finding more specific group types.

We evaluate 6 distinct K-Means models in this paper, varying either the data fed into the model or its normalization. Although we will discuss using K-Means over additional algorithms later in the paper, these 6 methods are all input with MAH information directly, only changing the normalization or how we extract the mass from the halo merger trees. In the first, we run K-Means over the logged MAHs using the main progenitor or primary halo from the merger trees at each time step. In the second K-means model, we first transform the logged MAHs into a reduced 11-dimensional PCA space, preserving 95 percent of the variance while reducing the overall data size significantly to less than 2% its original size. In the third iteration we first normalize the data across its 'zeroth' axis, removing the population mean across time from the data (see Data Normalization, Figure 2.2). Since K-means treats each dimension as though it is linearly independent, linearly scaling the overall normalization would not be expected to significantly alter the groups as the relative distances are preserved. In the fourth iteration we normalize the data across its first axis, keeping the population mean, while standardizing the variance at each time step. Unlike

changing the overall normalization as in the previous model, changing the variance of each time step results in different calculated distances for K-means, potentially changing the resulting groups. The fifth iteration is identical to the first, except that we extract the MAH information from the merger tree using a halos listed descendant ID, rather than simply tracking the primary halo from the tree. These changes impact roughly a third of halo histories, usually at early times, while most histories are unaffected. Finally in the sixth K-Means model, we again use the primary progenitor data, now in the linear space, rather than logged, resulting in greater emphasis at later time where the data values are significantly larger.

- **Model 5. KMeansLOG:** Groupings are found by running K-Means over the logged MAH data. Parameters: K = 3
- **Model 6. KMeansPCA:** Groupings are found by running K-Means over a reduced PCA space preserving 95 percent of the variance in the logged MAH data. Parameters: K = 3, $\text{Variance}_{\text{PCA}}$=0.95
- **Model 7. KMeansNORM0:** Groupings are found by running K-means over an axis 0 normalization of the logged MAH data, subtracting the global mean. Parameters: K = 3 , $\text{Axis}_{\text{norm}}$ = 0
- **Model 8. KMeansNORM1:** Groupings are found by running K-means over an axis 1 normalization of the logged MAH data, preserving the global mean while normalizing the MAH variance at each time step. Parameters: K = 3 , $\text{Axis}_{\text{norm}}$ = 1
- **Model 9. KMeansDESC:** Groupings are found by running K-Means over the logged MAH data, obtained from following a halos specific descendant ID path found in the merger tree files. Parameters: K = 3

- **Model 10. KMeansLINEAR:** Groupings are found by running K-Means over the *linear* MAH data. Parameters: K = 3

2.4.3.4 Bag-of-Words

K-Means considered each time-step 'dimension' as orthogonal to every other time-step and as a result MAH information linking consecutive time-steps is not considered. If we wished to consider information contained in some number of consecutive time-steps, we could use a modified bag-of-words algorithm to consider a collection of small segments of a halo's MAH, preserving local time series information for each history. Bag-of-Words (BOW) is a popular discretization algorithm used extensively in natural language processing for classification when paired with a clustering algorithm like K-Means. BOW has been extended to include image processing with the Bag-of-Visual-Words which has been used in many fields, from genetics to signal processing (DUYGULU 2002; Lin & Li 2009; Wang et al. 2013). The idea is to create a 'dictionary' of the collection of 'words' used in a document and compare the relative frequencies of different types of words which yield different types of 'documents'. If the documents are text documents, the bag of words would contain the unique words and their frequencies. Assuming you had many types of documents, different groups of documents would exhibit different looking 'bags-of-words'. This technique typically removes the spatial information of words and instead treats a document like an unordered collection of words, or phrases.

The technique can easily be adapted to mass accretion data by converting the time-series data into a series of 'letters', which can be linked to form 'words, and grouping the each object by its collection of words. To do this, one must choose a few parameters related to digitizing time-ordered floating point information like time-series. Rather than considering the full extent of floating point parameter space in mass, at each time-step we divide up the masses into letters, corresponding to their relative ranks, similar to the Symbolic Aggregate approximation (SAX) method. For example, if the masses at a given time-step were in the

range of 8.1 to 9.0, with a median of 8.5, we might assign 'A' to the range [8.1-8.4], and 'B' to [8.4-8.7], and 'C' to [8.7,9.0], and this process is repeated for each time-step. Given an alphabet of letters, next one must set a length for 'words'. If the length of words was set at 3, then there would be 27 possible words {AAA, BBB, CCC, ABA, ACA, ABB, ACC, ACB, ABC, AAB, AAC, BAB, BCA, BAA, BCC, BAC, BCB, BBA, BBC, CBC, CAC, CBB, CAA, CBA, CAB, CCA, CCB}. Given a time-series length of 100, and a word length of 3, one can either subdivide it evenly into 3 part chunks (with a remainder), or use a rolling window to create new words. Under the rolling window with 3 letter words, a sequence of AABACA would be listed as AAB, ABA, BAC, ACA, thus repeating letters and increasing word counts. One could additionally set a 'sparsity' to an integer higher than 1, which moves over that many letters when performing the rolling window. We use the standard rolling window set to 1.

With the letters and words defined, each halo will have a collection of words (a 'bag') which no longer contain long-term time series information but instead contain the time-series information quantized into words. A variation on this algorithm imposing a long term trend over the data can actually take into account relative position in time, however for these initial BOW models each time-step chunk is 'alphabetized' independently. Next, comes the actual classification portion when you group together like bags of words using a method like K-Means, considering each unique word an independent dimension.

One way that bag-of-words can capture MAH information is by capturing local variability of mass gain or loss features. Similar histories might be composed of similar local events, regardless of their exact time of occurrence. A calm halo history mostly made of smooth accretion may have several repeat letters, (e.g. AAA, BBB, CCC) depending on whether it is average, or under/over massive. A variable halo history, may contain many words containing different letters (e.g. ACA, BCA, ABC). While any halo could contain any single word in its history, the aggregate of words should tell us something about its overall history whether it is usually increasing, decreasing, calm, variable, or any other in-

formation which can be conveyed in this manner. Depending on the number of letters and letters per word, the complete dictionary of words (made up of every possible letter combination) may be sparsely or densely sampled in the histories. To avoid the complication of dealing with sparse data containing zero counts for many words, we only use the raw counts greater than zero as input to K-Means. Although we did not test this in our models, it may be useful to first normalize, or even log the counts before inputting to K-Means which would place more emphasis on the lower frequency words in a history. Currently, in considering the raw counts, a few more popular words dominate the signal, predominately the words containing repeats of a single letter.

An extremely sparsely populated word list, will result in a high dimensional space with little ability for the classification algorithm to discern between similar words and drastically different words. An extremely densely populated word list will likewise make accurate classification difficult, as information contained in the actual history will be lost in the discretization step, unable to be recovered in classification. Long words, containing many letters, are able to contain more time series information, at the cost of a more sparsely populated dictionary. On the other hand, single letter words, lose all time series information. We sample two different $N_{Letters} = \{3, 5\}$, and two different Word Lengths, $L_{word} = \{4, 8\}$. One must also decide the level of discretization to use in time by setting the number of words per history. If we used a full history, the number of words per history would be determined by the word lengths and the number of snapshots in the history. Each adjacent letter in a word would then represent the time difference between snapshots. We can instead subsample the MAHs to yield a predetermined number of words per history. As the number of words per history decreases, the time covered by each word increases. In order to balance computation and data size, as well as span a reasonable amount of simulation time in each word, we set the Words per MAH, $W = \{25, 50\}$, testing both 25 and 50 words per history. We found in prior tests that even with just 10 words per history, we were able to reliably capture the general structure of the majority of the data. We finally run K-Means

over the bag-of-words space from each of the methods to classify the histories into 3 groups per model.

- **Model 15. BOW1:** Groupings are found by running a Bag-of-words algorithm+Kmeans on logged MAH data. Each time step has 3 'letters', or levels of discretization. Each word is composed of 4 adjacent letters and the history is subdivided into 50 total words. The resulting 'bag' of words is input to Kmeans. Parameters: N_L = 3 , L_W = 4, W = 50 , K = 3

- **Model 16. BOW2:** Groupings are found by running a Bag-of-words algorithm+Kmeans on logged MAH data. This model features longer words, resulting in sparser features in the 'bag'. Parameters: N_L = 3 , L_W = 8, W = 50 , K = 3

- **Model 17. BOW3:** Groupings are found by running a Bag-of-words algorithm+Kmeans on logged MAH data. The same as BOW2 except fewer total words per history. Parameters: N_L = 3 , L_W = 8, W = 25, K = 3

- **Model 18. BOW4:** Groupings are found by running a Bag-of-words algorithm+Kmeans on logged MAH data. This model uses 3 letters per time step to make up words 8 letters in length, with 25 words per history. We use a shorter word length to reduce the sparsity of features. Parameters: N_L = 5 , L_W = 4, W = 25 , K = 3

- **Model 19. BOW5:** Groupings are found by running a Bag-of-words algorithm+Kmeans on logged MAH data. The same as BOW4 except we use more words per history. Parameters: N_L = 5 , L_W 4, W = 50 , K = 3

2.4.3.5 Gaussian Process Mixture Models

While K-Means considers circular or spherical regions (in N-dimensions) around group centroids when minimizing the total the *inertia*, treating all dimensions as independent,

Gaussian Process Mixture Models (GPMMs) (Rasmussen & Williams 2006) additionally allow for the covariance to be modelled between all data dimensions: $I = (x - \mu_k)^2/\sigma$, where 'x' is the data matrix, μ_k is the kth component's mean vector, and σ is a covariance matrix for each component. In 2D, this covariance allows for ellipsoidal probability regions, using a weighted distance metric, compared to K-Means simple SSE. This can easily be extended to any arbitrary number of dimensions including across time-series data (Roberts et al. 2013). For halo histories this means that similar halos have a more complex criteria to be considered 'close', which considers the structure of the data when determining closeness, not just the euclidean distance between two vectors. The general idea behind a Gaussian Process (GP) is to place a prior over the space of all functions, and use Bayesian inference to update the function's belief by observing noisy realisations of the function at a finite number of points. This means that a GP can take in any noisy, finite observations, and fit a function from the space of all possible functions to them using Expectation-Maximization (EM) optimization. The GPMM extends this idea to include multiple GPs. A single GP can be considered an infinite extension to a multivariate Gaussian distribution, while a GPMM is the collection of these GPs. Each GP can be thought of as an infinite series of Gaussian distributions which model a population distribution from finite functions such as MAHs. Unlike typical parametric functions, which require some predefined functional form and relevant parameterization, the GP itself is considered an infinitely parametric model (or 'non-parametric' depending on the definition) which is in practice is dependent on the size and complexity of the input data. GPs can be slow on larger data sets largely due to a matrix inversion operation. Regarding run time, we had no issues fitting GPs on our data set in python, even if they are relatively slow compared to K-Means; nevertheless, we downsampled our first GP model to a vector length of 300 per history to achieve model convergence, which cuts the run time in roughly half. Downsampling to a vector length of 100 cuts the run time to nearly a tenth, but at such a low resolution the model also fails to reach convergence.

Given time-series data, a GP is fully defined by the mean function of the data and a specified covariance matrix. The choice in the covariance kernel, as well as any error specified on the data, can dramatically shape the resulting GP. If there is no error term, the GP is precisely defined at each data point, with the covariance matrix defining the GP at all points away from the data. When considering 'N' GPs together, you get a GPMM, which has N components, analogous to the 'K' groups in K-Means. In fact, many GPMM algorithms begin with an initialization of K-Means to get the initial components. If the number of groups is not known beforehand, a Bayesian GPMM (BGMM) can be used which determines the number of components stocastically. For this work, we set the $N_{Components} = 3$ corresponding to the three groups.

One advantage of a GPMM and BGMM alike is that once fit to the data by minimizing the likelihood given a specified kernel definition, each MAH can be given a 'soft assignment' to each component by computing the likelihood that it came from each. One can easily convert the soft assignment to a hard assignment by simply assigning each halo to the maximum likelihood component. We use the Sci-kit Learn GaussianMixture and BayesianGaussianMixture python packages which have a predict function for assigning data for a specified number of components. Unless otherwise noted we use the default parameters for each. When considering multiple components, one must choose what covariance structure to allow; the GaussianMixture module gives four options: "Full" - each component has its own general covariance matrix, "Tied" - each component shares the same general covariance matrix, "Diag" - each component has its own general diagonal covariance matrix, and "Spherical" - each component has its own single variance. Assuming the data vectors mostly have similar observed covariance structures, there would be little difference between the Tied, and the Full (or Spherical) options. If we instead observe one population of histories having a very different covariance matrix between its data points compared to the other histories, then one might expect better fitting groups to be modelled by using Full (or Spherical) over the Tied option. Fuzzy Clustering K-Means is a special case of diagonal

GMMs when the weights are equal and the covariance matrices are the same.

For the maximum flexibility in our models' ability to fit the data, we use the 'Full' covariance for most of our models. Additionally, we compare using a 'Tied' covariance which forces each component to share a general covariance matrix. Despite the Tied covariance only having to fit one shared matrix, we find it generally takes longer to converge than the Full covariance method. Since we are fixing the components to 3 and all components have significant weight, there is little difference in GMMs versus BGMMs. When using a large number of components, BGMMs give the model the flexibility to set 'unneeded' components to zero, through the "stick-breaking process" guided by the WeightConcentrationPrior (wcp) parameter. The stick-breaking process is an example of a Dirichlet process, an iterative method of probabilistically assigning objects to groups by finding the probability distribution of $N_{Components}$, which themselves are probability distributions - 'a distribution over distributions' as it is called. This process is similar to the 'Chinese Restaurant Table Distribution' (CRT), where each object is assigned to an existing group or placed into its own new group probabilistically based on its similarity to the existing group. Beginning with one group and one data object, the process assigns each subsequent object into its own new group or into one of the existing groups based on the likelihood that it came from each group weighed against the chance of creating a new group. As new groups are created, the likelihood of creating a new group is decreased. A standard GMM would utilize all of the specified components regardless of whether motivated by the data or not. For the BGMM we tried several wcp values to see if there was an appreciable effect on the resulting groups, however, all of the wcp choices we tested resulted in significant weight among the 3 components specified, meaning all components were used. We did not explore higher component groupings in depth for this paper, although higher components appear to linearly decrease in their Dirichlet concentration up to 10 components when evaluated with a 'Full' covariance kernel.

- **Model 11. GMM-Full:** Groupings are found using a GMM where each component has its own general covariance matrix. Parameters: CovarianceType = Full, $N_{Components}$ = 3

- **Model 12. BGMM-Full1:** Groupings are found using a BGMM where each component has its own general covariance matrix. Parameters: CovarianceType = Full , wcp = $1/N_{halos}$, $N_{Components}$ = 3

- **Model 13. BGMM-Full2:** Groupings are found using a BGMM where each component has its own general covariance matrix, the same as the previous group except using a different wcp for halos. Parameters: CovarianceType = Full , wcp = 1E4 , $N_{Components}$ = 3. Note: this model's output is exactly matching the previous model given our input data used. In prior tests we often found little variance in output with the wcp, though not always exactly equivalent. We leave this model in place for testing evaluations with down-sampling.

- **Model 14. BGMM-Tied:** Groupings are found using a BGMM where each component shares a general covariance matrix. Parameters: CovarianceType = Tied, wcp = 1 , $N_{Components}$ = 3

2.4.3.6 Dendrogram methods

Until now all of the methods worked by direct classification, meaning that one run of the method resulted in one label per halo. A dendrogram is a hierarchical grouping graph that results in a fully connected genealogy - either from the bottom up (agglomerative) or the top down (divisive). As a result of this hierarchy, there is initially no single grouping for any given data vector, rather each object is progressively grouped together based on the defined metric. A dendrogram is composed of leafs and nodes, which represent the objects/groups and the connections between groups respectively. The height of the connections gives the relative similarity (or dissimilarity) between the groups, with the top connection

representing the split into two groups. As opposed to a single grouping that has static labels for each halo, a dendrogram is the result of multiple groupings of groupings, which can give class labels based on a defined similarity threshold and distance matrix. At the lowest threshold, connected halos are most similar to each other, with lots of individual groups. As the threshold increases, more and more halo histories are grouped together until all are grouped into one massively connected group.

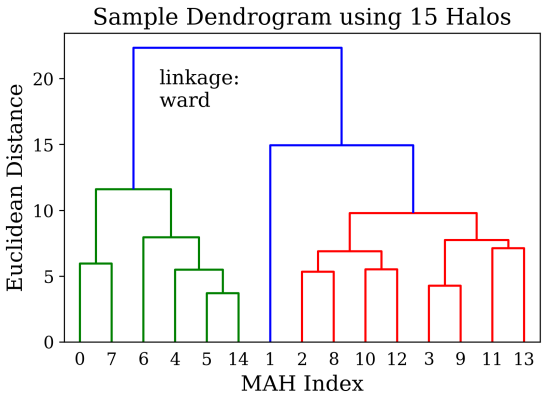

Figure 2.6: A sample dendrogram made from an arbitrary 15 halo histories (labelled 0 to 14) using Euclidean distances computed in log space and the 'ward' linkage function to connect groups. A grouping threshold was set at 12 (roughly half the max distance), to achieve two 'groups' shown in green and red, and a 'singleton' shown as a single blue 'leaf', at MAH Index 1. The closest two histories are shown by the node between MAH Index 5 and 14, while the greatest distance is between the 'green' group, and the node merging together the singleton and 'red' group. In our actual dendrogram methods, using all histories, we ensure 3 groups and no singletons.

At the heart of a dendrogram is the linkage matrix which gives the prescription for reconstructing all of the connections and leafs which build the overall tree. All leaf elements begin in their own groups, and the first connection is made among the two most similar histories. This new connection will be labelled as a new group with the label N+1 (assuming there are N initial leafs). At this point, the linkage type will determine how this newly merged group will be treated and whether individual group member's vectors will be used in subsequent calculations, or if a group mean data vector will be constructed. The process iterates by listing the group each subsequent leaf is joining with a new 'i+1' group label until all groups are connected together into one giant tree. Giving the bigger picture of the

data connections is one of a dendrogram's greatest benefits over something like K-Means, which gives a single grouping of the data per run. Being able to quickly visualize the similarity of various halos at any arbitrary threshold, makes dendrograms powerful tools for when the number of groups is not known beforehand.

Although a dendrogram doesn't give any one specific grouping for a given set of data, in most cases it can be used to determine the dissimilarity threshold at which K unique groups will be cut. This means that for a given dendrogram, one can set a threshold such that K groups are obtained; however, for some linkage methods, this cut can result in many singletons in addition to the main groupings. This situation is rarely desired and we additionally set a minimum group size threshold of 200 halos on certain methods to ensure we never have extremely small groups or lone singleton groups being compared with nearly monolithic groups. In Figure 2.6 for example, we show 15 sample halos and the dendrogram showing their dissimilarity based on the Euclidean distance in the high-dimensional log MAH space. We have arbitrarily chosen a Euclidean Distance of 12 to initially split groups on, such that we have two large groups (green and red) and one singleton (blue). Assuming we wanted 3 groups, with no singleton groups, our algorithm would reassign the singletons to the smallest group then lower the distance threshold until there are no more singletons and we had the desired number of groups. Although it may be more appropriate to ignore the singletons, rather than to reassign them, we did this to preserve the total number of objects in each of our methods. In most cases this step is not needed and limiting minimum group size is not a feature in standard dendrogram slicing methods.

We test 4 different dendrogram models, changing the linkage matrix that each dendrogram is built from. We first start using the 'Ward' linkage matrix, which at each step finds the pair of clusters that leads to the minimum increase in total intra-cluster variance after merging. This means that the very first merge would be between the two MAHs which are the most similar as determined from their squared euclidean distance between them. The next merge would be whichever two halos (or whichever halo and the group formed

from the first merge) result in the next minimum increase in total intra-cluster variance. If all MAHs have been assigned to a group, one then finds which two groups' merger would minimize the increase in intra-cluster variance until all items are grouped.

In addition to the Ward linkage matrix, one can use the 'Average' linkage which instead of minimizing the variance, computes the pairwise distances between all members of each group and merges to minimize the average distances. A 'complete' linkage instead finds the maximum distance in each group and then merges according to the smallest maximum. This method looks at extremes in the data, rather than computing an average or minimizing a variance, making it more sensitive to outliers in the data. Finally, there is the 'weighted' linkage method which uses the arithmetic mean for distances when merging groups. For example, if groups 'i' and 'j' merge together into $(i \cup j)$, and we are computing the distance to group k, the distance, $D((i \cup j), k) = (D_{i,k} + D_{j,k}) / 2$. This method prevents recalculating distances from the raw data when merging groups.

- **Model 20. Dendrogram-Ward:** Groups are found by setting a threshold (to get K=3) on a dendrogram constructed using the 'Ward' linkage matrix. Parameters: LinkageType=Ward, MinimumGroupSize=200

- **Model 21. Dendrogram-Average:** Groups are found by setting a threshold to get K=3, on a dendrogram constructed using the 'Average' linkage matrix. Parameters: LinkageType=Average, MinimumGroupSize=200

- **Model 22. Dendrogram-Complete:** Groups are found by setting a threshold on a dendrogram constructed using the 'Complete' linkage matrix. Parameters: LinkageType=Complete, MinimumGroupSize=200

- **Model 23. Dendrogram-Weighted:** Groups are found by setting a threshold on a dendrogram constructed using the 'Weighted' linkage matrix. Parameters: LinkageType=Weighted, MinimumGroupSize=200

2.4.3.7 UMAP methods

UMAP stands for “Uniform Manifold Approximation and Projection for Dimension Reduction" (McInnes et al. 2018, 2020). It is a non-linear data reduction technique that takes in data in ‘N’ dimensions and reduces the dimensionality, usually to 2. UMAP is touted as an upgrade to the more well known “t-distributed Stochastic Neighbor Embedding" (t-SNE) algorithm (van der Maaten & Hinton 2008). t-SNE looks at pairwise distances in the data among each data object and its closest neighbors, and attempts to map the distances from the initial space indirectly onto a lower dimensional space for easier visualisation. It attempts to optimize the distribution of points in a lower dimensional space by converting data similarities into probabilities and minimizing the KL divergence of the joint probabilities between the low dimensional space and the input space. It does this by prioritizing closer pairs over longer distance pairs and remapping the points onto nonlinear axes in an attempt to group similar objects together while ignoring the relative distances of farther pairs. UMAP makes a few tweaks to the implementation of t-SNE, specifically in using unnormalized probabilities, making it faster and more scalable while attempting to better preserve global structure in the data (Narayan et al. 2020). If using t-SNE on large data sets it is often recommended to first run an alternate form of dimensionality reduction first, like PCA with pruning. Using UMAP, however, we found this to be unnecessary with our data size and did not reduce the data prior to input. Although we do not include t-SNE in our methods list, t-SNE ran 15-20x slower in remapping the MAH data compared to UMAP in tests.

We use UMAP as a data reduction technique in two separate ways. First we employ UMAP on the MAH data itself to reduce the data from a $N_{halo} \times M_{timesteps}$ dimensional data set down to $N_{halo} \times 2$ in a non-linear manner. The points are first initialized by the algorithm in two dimensions using ‘Spectral Clustering’, which is a reduction technique based on graph theory which forms links among neighboring objects and looks for nodes of densely connected points (Ng et al. 2002). Then, gradient optimization is performed over the Spec-

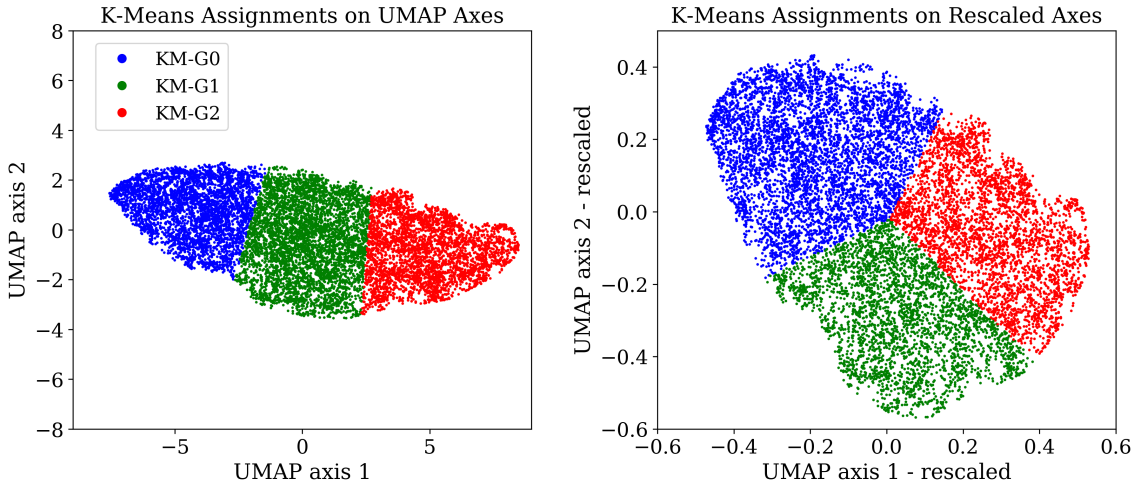

Figure 2.7: We show the effect of axis normalization of the UMAP MAH dimensional reduction space, on the K-Means groups obtained. The colors represent K-Means assignments over the UMAP reduced space. Left panel: Axis 1 & 2 as output from UMAP when input with the MAH data. Right panel: Re-scaled/Normalized dimensions from UMAP before running K-Means, resulting in a more circular distribution and radial boundaries between clusters.

tral Clustering space to find a lower dimensional mapping that attempts to preserve both local and global structure. After reducing the MAH data to just 2 dimensions, we are able to run K-Means just as before. The benefit to this technique, over running just K-Means alone, is that it allows UMAP to embed some additional non-linear information in the two dimensional space which can then be classified with a linear algorithm like K-Means. Although the UMAP MAH data reduction speeds up the K-Means process by over 50x (running in under 0.2s), the UMAP algorithm is significantly slower, thus there is a time penalty to this method over using K-Means alone. In the final column of Table 2.1, we show the run times between the two models, 7 seconds vs 16 seconds for K-MeansLOG & UMAP+K-MMeans (Model 5 & 33). There is one final subtlety when considering the UMAP output for obtaining groups. In Figure 2.7 we show the raw output of UMAP in the left panel, such that the data is oriented over a wide range of values across the UMAP axis 2, and a relatively small range for the axis 1. If we instead were to rescale the UMAP output values, such that the relative positions remain unchanged, the respective groupings from K-Means, shown

in color, can still change drastically. In the left panel, the three K-Means groups divide up linearly across the UMAP axis 2, whereas in the rescaled case the K-Means centroids are divided up radially. We choose not to rescale the UMAP axes for our groups. We would like to note that in our tests we found that the UMAP axis 1 was highly anti-correlated with the t-SNE axis 2, with a Pearson correlation coefficient of -0.98. The ordering of each axis is arbitrary, so an anti-correlation can be considered the same as a strong correlation. Even though their other axes were also fairly anti-correlated, r=-0.76, this was not enough to guarantee a high group match fraction when running K-Means over both distributions. In fact, we found that only roughly 2/3 of the model assignments match between t-SNE and the raw UMAP space, and just over 1/2 of the assignments match using the rescaled UMAP space. Although not shown, visually the t-SNE groupings better reflected the radial pattern we observed in the rescaled UMAP panel (Fig. 2.7 right), however, the actual group assignments had significant differences. This difference indicates that K-Means may not be finding optimal groups for halo bias in the UMAP or t-SNE space, however, alternatives are not explored in this paper.

- **Model 33. UMAP:** Groupings are found using K-Means after first running UMAP dimensionality reduction on the log spaced MAH data into 2 dimensions. Parameters: K = 3, $\text{Dimensions}_{\text{Final}} = 2$, $\text{N}_{\text{Neighbors}} = 15$

Additionally we will use the UMAP algorithm to map each of our 34 model classification outputs into a 2-dimensional space for easy visualization and clustering. This will be described in detail in section 2.5.4, “Group Archetypes".

2.4.4 Complete Method List

We have described several different types of base grouping algorithms to classify different halo mass accretion histories of similar final masses. These methods include the standard age ranked definitions and variants based on age, definitions based on mass-loss,

and unsupervised methods like K-Means, bag-of-words, Gaussian mixture models, hierarchical methods, and UMAP. Now that we have described the methods we run each one on the MAH data, obtaining group assignments (K=3) for each halo. The full list of methods are provided in Table 2.1 along with some of their relevant parameters, their group sizes, and relative approximate run times.

2.5 How similar are groups identified by different methods?

2.5.1 Relabelling for Consistency among Groups

In trying to compare the different algorithms it is natural to wish to know which groups from one algorithm match to other groups from another. Although there is no guarantee that the clusters are selected in similar ways, thus there is not necessarily a natural mapping, we choose to match groups between clusters in the following way: as long as the number of groups is low, it is possible to brute force try all possible matches to find the 'best' match percentage.

First, we take the first half mass grouping (Model 0), sorted from youngest to oldest, and assign them group labels 0, 1,...N_{groups}. For three groups in age, the latest forming objects are group 0, and the earliest are group 2. Each subsequent algorithm's assignment labels are relabelled if necessary so that the most labels from one assignment best match our Zeroth model's group assignments. This is not guaranteed to yield the 'desired' results, such as always ranking the groups by age, and can fail in a few cases, mainly stemming from complications with groups of different sizes. When a model's groups are all roughly equal sizes to another groups, it is possible for all halo assignments to match, however, as the group sizes are more different this fraction tends to go down. At the extreme, a model with all but two of the histories in one group, would only be able to match around 33% with a model with equal group sizes. While we could rank the halos in order of age directly, we instead set our criteria to maximize the match fraction. With this criteria a natural order is obtained for the majority of groups.

It should be noted, relabelling the groups doesn't change any of their relative assignments, only the label used for them. Without performing this relabelling, comparing the groups between certain methods would be meaningless. Since two separate runs of K-Means can have a completely arbitrary initialization for the groups it labels 0 to K, (assuming 3 groups: 0,1,2) even if each run resulted in the same exact divisions among the groups, the match percentage could be zero. If we assume objects in one group labeled with 0 were labelled with a 2 in the other K-Means run with a full mapping like (0,1,2 : 2,0,1), we can see how just the label could result in a near zero match. If instead the mapping were (0,1,2 : 2,1,0) one could still see some agreement among the two methods even with an 'incorrect' mapping. As stated above, we choose the mapping which maximizes the match percentage of objects with our first method, based on equal sized groups in age according to our first model. For some models, particularly those with very unequal group size, this procedure can produce extremely low match percentages. As we were limited to 3 groups we were able to test every possible combination of groupings however the matching problem becomes vastly more difficult as the number of groups increases, requiring different matching methods.

2.5.2 Match Fraction

We show the fraction of similar assignments for each of our different methods in Figure 2.8. This matrix shows the methods arranged in the same order as they were assigned, 0-33, starting with the age based methods at the top left, where the color shows the match fraction between the respective models. In general we see some structure in the matrix roughly corresponding to our base model types. To first order, like models group together, with some notable exceptions. On average we find a match fraction of around 60 percent between models, with fairly low match fractions between age models and other models. We find that our zeroth model (PMFHM) matches the labels in our first model (FHM) 79%, while the first model matches the second model (PHM) at 95.5%. Despite this difference

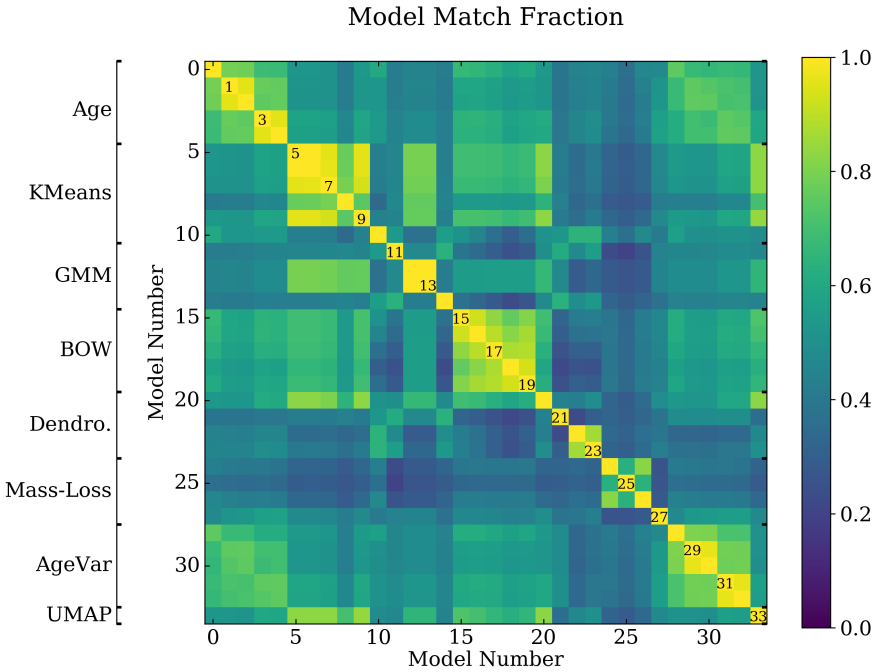

Figure 2.8: A symmetric direct match-percentage matrix showing the matching fraction between all models. The matching fraction between two models is defined as the fraction of all halos that receive the same group classification by both models. The diagonal is set to 1 by definition, and every other model is labelled on the diagonal for convenience. We list the base model types on the left, combining GMMs and BGMMs.

in match percentages, looking at Figure 2.3 and comparing the most extreme half mass halos, the PMFHM and FHM columns appear to be more similar at late time. Looking at the averaged definitions in Figure 2.4, the PMFHM and PHM perhaps look more similar at early times in the top row. The higher match percentage observed with the FHM and PHM makes sense when one considers the vast majority of halos have peak masses equal to or very close to their final masses.

The color-bands in Figure 2.8 appear to line up with several of the different method types, such as between like age based methods and like K-Means based methods. Two exceptions to the very high match percentage among K-Means methods, are Model 8, with data first normalized via axis=1 with a slightly lower relative match percentage to the other

K-Means models, and Model 10, based on linear mass input, which appears different from most other K-Means models. Model 10's match fraction with other models is dissimilar, with an average match of just 46%. However, given its group sizes, the average "best" match possible over all other models is just 61%. Model 10 actually has a sizable fraction with Model 22 and 23, the complete and weighted dendrogram methods. We see that the K-Means methods have a fairly high match fraction around 75% with UMAP, the 'ward' dendrogram (Model 20), and the 'full' covariance GMM methods (Model 11). We observe a high similarly among the GMMs using the 'full' covariance, while the 'tied' covariance BGMM (Model 14) had a lower match fraction with most groups. We find dendrogram based methods appear to have darker color-bands associated with their positions dependent on the linkage type, with 'complete' and 'weighted' linkages (Model 22 & 23) resulting in near 90% agreement. We also notice a match fraction block among three mass loss models (24 - 26), with varying degrees of match among each other, and relatively strong disagreement with other models. The arrested-development model (27), also a mass loss model, is significantly different from the other three, having below 33% agreement

2.5.3 Downsampled Histories

As a result of having such a high cadence output in the Vishnu simulation, we were curious to note how these algorithms would work if running on a sub-sampled halo history. Knowing how consistently our MAHs group using downsampled cadences will help us understand how these results might differ if using a lower resolution simulation. To test the effect of reducing the input data to each method, we sub-sampled each history from down to 800, 400, 150, 75, and 20, time-steps each. We then ran each respective method on this downsampled data, including relabelling for best consistency to our zeroth age model, and we plotted the match percentage with the results from using the full history in Figure 2.9. It was somewhat shocking to see just how differently various methods performed at this test. We find some methods are quite stable, hovering around the 90% agreement level even

downsampling to one-tenth of the original data size, while others, like mass-loss methods, quickly degrade in match fraction with smaller group sizes. It should be noted, that the GMM method (Model 11) was limited to a maximum of 300 points per history in order to reach convergence, thus we wouldn't expect a significant difference at the largest data sizes for those groups. Additionally, the BOW methods (Models 15-19) are restricted based on the number of words parameter, which may artificially bias those results to higher than expected values, as their highest samples are already based on a 'downsampling'. We do not find a degregation in BOW by downsampling until the smallest MAH vector length we tested, 20. This suggests either that BOW is very stable or that we should increase our words per history, or letter time-resolution, to extract additional information from the high resolution history.

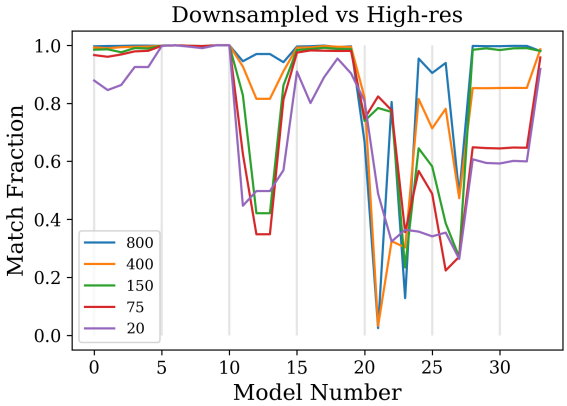

Figure 2.9: The effect of MAH temporal sampling on halo classifications. We show the direct matching fraction between each full Model and 5 down-sampled runs where data was resampled down to (800, 400, 150, 75, and 20) points per history before the model was run. K-Means based models perform incredibly consistently across the variety of input sampling, followed closely by UMAP. Standard age based models and BOW only degrade in consistency at the lowest temporal sampling. BGMMs (Model 12 & 13), Mass-Loss methods (24-27) and Dendrograms recover significantly different groups at lower temporal sampling.

We found that K-Means methods performed nearly perfectly regardless of which history downsample size was tested, matching consistently above 96%. Considering how K-Means works, this is not too surprising, as the method averages information from each time-step to

determine the appropriate class label. Age also performed well, degrading predictably with the size of the downsample. At the lowest resolution, the age groups still matched around 90% with the full resolution run, although the groups noticeably improve until around 150 time-steps. The age methods which were divided using the K-Means group sizes vary more greatly in their match percentages than the evenly divided group age models. For the highest resolutions, the matches are at or above 95% agreement, while the lower resolution run performed around 60%. The variability in performance may arise from the stochastic nature of halos which means the rank order of halos by half mass can be determined largely by short spikes rather than a smooth inherent signal. This variability is not significantly reduced even when averaging over several fractional mass ages.

The Dendrogram groups performs astonishingly poorly at times having less than 10% match. This may seem impossible given that a random grouping would have roughly a 33% match and we match group labels for the best grouping. Assuming two runs of a Dendrogram method both produce very unequal groups, a result of the specific linkage matrix, it is possible that the respective two large groups are not matched together. Assuming one run had groups as follows (assuming 1000 total halos for illustration), with the group label, and size following (group: size) (0:800, 1:100, 2:100), and another run had (0:100, 1:800, 2:100), and each one was relabelled such to maximize the match percentage with age. It is possible the zeroth group in the first run, contains a majority of the oldest halos, so it keeps its label as '0', and likewise the other groups match with the medium age and earliest groups. Likewise, it is possible the second run had the largest over-representation of oldest things in the second group, and thus it might retain its '2' label, with the majority of average things being in the group labeled 1. It is possible the objects labelled 1 were actually more over-represented with earlier halos mapping it to 0. In this hypothetical case, the two largest groups containing the majority of halos, would not be matched to the same label. This means, after relabelling, that two consecutive runs of a model could have significantly below a 33% match. Not all models with disparate group sizes at full resolution

experience a low match percentage at lower resolution. The KMeansLin has very disparate group sizes in the full resolution run, however, it actually experiences extremely stable performance across downsample lengths. Similar to the dendrogram methods, the GMM-full and BGMM-tied (Models 11 and 14) have a very large group, and two disparately sized groups in the full resolution, yet they have fairly high match percentages above 80% for all downsample runs.

2.5.4 UMAP visualization & Archetypes

In addition to using the UMAP as an algorithm for reducing the MAH data (see: 3.3.9.UMAP methods), we also use UMAP to help visualize the collection of model assignments we have generated. A remapping takes the matrix of Model Assignments of size N_{Halos} x M_{Models}, and maps it into the 2D UMAP space. One may be tempted to first use a linear model in order to reduce the Model Assignments space. We attempted to use PCA given the input of Model assignments (in both the ordinal label space and also a One-hot Encoding space), for all 34 models to visualize our model assignment space across its primary and secondary principle components. We found very little separation among the majority of our models in the pruned principle component space, with a dense central cluster and a few models far away from the origin, including our UMAP model 33, an age based method using K-Means group sizes model 32, and in the One-hot encoding space even model 0 was not centrally located. There did not appear to be a strong significance to the PCA remapping in the 2D space with with model type, although some spurious model similarities were noted.

2.5.4.1 One-Hot Encoding

A subtlety one must consider before inputting the model assignments data to be reduced, is the relationship between some classification C_i and some C_j. If two of our classes are labelled with '0', and '2', should their average result in the same as if we had two '1' labels? Although some of our models output ordinal class labels, such as age and mass

loss based methods, GMMs and K-Means do not guarantee ordinality. As a result of the class labels potentially being non-ordinal, we can perform our classification over a 'One-Hot-Encoding' preprocessed space. This converts a vector of scalars C: [C_0, C_1.. C_N] into a matrix [V_0, V_1... V_N] where each V_i is a class basic vector the length of the number of clusters each model contains (3 in our case), containing only a single non-zero value per object vector. Assuming 3 clusters, each V_i is contained in the following set: {[1,0,0], [0,1,0], [0,0,1]} such that if our first classification C_0 = 1, then V_0=[0,1,0]. With this encoding, we prevent the case that a model with C_i=0, and C_j=2 would suggest an average $C_{i \bigoplus j}$ = 1, and instead the combination would simply be a superposition of both class 0 and 2. One hot encoding converts a vector with non-ordinal labels into a vector of vectors containing only binary class labels. This encoding is used specifically as input to the UMAP space, which then outputs a position for each model in a 2D space.

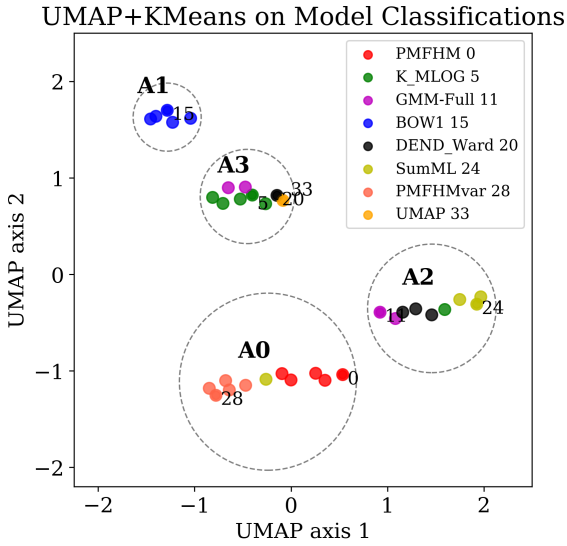

Figure 2.10: The UMAP output of all the various Model assignments for one random seed. We used K-Means (N=4) on the output after standard scaling and label the assignments (A0,A1,A2,A3) above each group centroid. We label a few choice models by their model number. The colors represent the general base type of method associated with each point [Age, Kmeans, Gaussian Processes, Bag-of-Words, Dendrograms, Mass-Loss, Variable Group Size Age, and UMAP], with a sample of models given in the legend.

We show the result of this UMAP dimensionality reduction over the Model Assignment space of our 34 models in Figure 2.10, where each model is represented by a single data

point in the 2D plane. By eye, one can see how the points are not randomly distributed, but instead appear to be lumped together into multiple groups. It is only natural to explore these groups in more depth. While it would be possible to run K-Means, SVMs, or any similar algorithm to group together models by their respective assignment labels in the PCA pruned space, or even in the original space, a UMAP nonlinear remapping based on the one-hot encoding assignments may give a more elucidating result. Once reduced, a clustering algorithm can be used to group like models into archetypal models, we refer to as our “Archetypes". Members of the same Archetype would be expected to group together similar halos to one another.

2.5.5 Defining 4 Archetypes

In the ‘One-Hot Encoding’ UMAP reduced space we see the appearance of some ‘natural’ groupings of many models by their assignments. Since UMAP places similar points in the un-reduced space closer together in the reduced space, we would expect that models with very similar assignments should be placed closely together. In addition to giving us some idea of the similarity between various models, UMAP attempts to create localized clusters as opposed to its spectral clustering initialization which merely maps the data onto an ordered one-to-one space. With localized clusters, K-Means can easily be used to recover the groups in a more automated manner through the use of a criterion like the BIC or the Gap Statistic and confirming by eye. Unlike in the case of running these criterion over our raw MAH data, which results in inconclusive recommendations for the number of K-Means clusters, we find a more robust result when classifying the model outputs themselves and choose 4 total archetypes.

These K-Means groups become the basis of our archetypal models, a way of reducing over 30 models into a more digestible data amount with 4 archetypes. In order to create our representative Archetype models, we select the *mode* assignment for each halo among the individual models contained in each archetype group, shown in Table 2.2. For example,

if in Archetype 0 we find that the majority of included models assign a group label of '2' to some halo index 'i', then that halo will be assigned a group label of '2'. We assign a group label to each halo independently for each Archetype. The mode is a good choice since it is resistant to outliers and is not subject to averaging artifacts (e.g. A halo with one model assignment of 0 and another assignment of 2 should not be given an assignment of 1, despite it being the average), similar to our consideration for using the one-hot encoding for our UMAP reduction. In nearly all cases, there will be a single mode, however, in the chance that a halo happens to be equally assigned two different group labels in the same Archetype, one of the two labels will be assigned randomly.

There is no guarantee that UMAP will map models from similar base methods close together. However, we would only expect methods from the same algorithm to group differently if the hyper parameters we varied among those methods changed the resulting assignments even more than than using a different algorithm entirely. While this may seem unlikely at first, the choices in hyper parameters we make (including preprocessing) can have significant changes in how the method 'sees' the data. In general we find methods from similar algorithms are typically grouped closely together in the UMAP reduced space, although this is not universal - especially for some GMMs and Dendrogram methods. The BOW methods are actually grouped all together, either indicating the stability of the BOW algorithm or that our selected hyper-parameters were too close together to fully explore all the possible groups.

Initially we utilized t-SNE (discussed in the UMAP methods section) for this purpose. UMAP is alleged to better preserve local structure as well as global structure, whereas t-SNE loses information about local structure, in addition to claims about its scalability. We were curious about the performance of UMAP vs t-SNE among data of similar shape and size to our Model Assignment label matrix, based on known mock data. For our tests we created mock data by choosing a few sample model assignments from our real data-set and varying a fixed percentage of the halo assignments randomly. These tests took 5 different

sample Model assignments, and randomized 15% of the assignments, to create similar but not identical 'known' groups. We then performed both a t-SNE and UMAP reduction on the assignment space, and looked at how well our known groups were preserved. We found that t-SNE actually did a better job preserving both local and global structure in these initial tests, but both produced reasonable dimensional reductions. Ultimately we chose to use UMAP for several reasons, it is a newer and faster method, we were already using UMAP for Model 33, and we observed less variance by random seed and initialization then we saw with t-SNE. Despite t-SNE slightly outperforming UMAP in our controlled tests, in general the UMAP clusters were more localized on the Model Assignments, making the K-Means classification step more robust. In general we found that t-SNE and UMAP produced near identical clusters for the majority of random input seeds we tested.

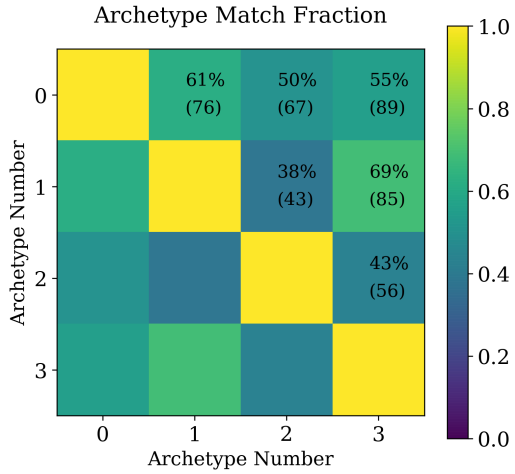

Figure 2.11: The direct matching percentage for each of the four Archetype models, similar to the match matrix for our 34 initial models shown in Fig. 2.8. The matching percentage is labeled in the upper triangular portion, along with the maximum possible match percentage based on relative group sizes, shown underneath in parentheses.

In Figure 2.10, we show the results of UMAP+KMeans on our model classifications. The point distribution in X and Y are solely due to the UMAP output, while the 4 K-Means 'archetype' assignments are labelled above each cluster. We label a sample of individual model numbers below each of the respective models. Due to some some of the points being tightly grouped in the UMAP space, it can be hard to clearly discern each individual

model label in the plot, thus we have listed each of the models by their relative archetype assignment in the 'Archetype Groups' Table 2.2. The Archetypes we found from UMAP follow a few trends by model type, which can be seen visually by the relatively close pairings of similar colors.

We show the match percentages between the various Archetypes across all groups in Figure 2.11. Similar to Figure 2.8, the match fraction / percentage is shown comparing how similar one model's output is to another, or in this case, how similar two archetypes are by comparing their representative assignment labels for each halo history (1 label per history per archetype). In the match percentage figure for Archetypes, we additionally label the match fraction in the upper triangular cells for clarity (the matrix is symmetric), with the diagonal elements defined to 100%. The max possible fraction, given the respective group sizes, are also given under the match percentage, in parentheses. We see that the various archetypes only agree in a range from 38% (between Archetypes 1 & 2) to 69% (between Archetypes 1 & 3). Although 38% is overall a terrible match, it should be noted that given the group sizes the maximum possible match was 43%. For Archetype 0 and Archetype 3, comparing the max possible match with the actual match percentage we see a very low match (55%), compared to a possible 89% based on group size. These match percents are based on each model's assigned label match to our original 'zeroth model' labels, and it is possible that two models would have a better match if allowed to remap their labels optimized between each model i & j.

For each archetype, we show the average history for the 3 groups (in linear space), shown in blue, green, and red for each population in Figure 2.12. The shaded region shows the standard deviation for each respective group. The respective archetype group sizes are given in the legend of each panel. The average histories of the groups agree fairly well, with the notable exception of Archetype 2 group 1 (green). Typically, the group 0 (blue) halo histories have a lower average value at nearly all times, compared with the other groups, while group 2 (red) halos have earlier formation histories and higher average histories at

nearly all times. The group 1 average history falls in-between the average histories for group 0 and 2 except in the case of Archetype 2. Also, notable in Archetype 2 is the slightly larger scatter in the group 1 histories (latest age population for Archetype 2).

Archetype 0. Age + Arrested Development:

The zeroth Archetype largely corresponds to the models based on Age. We see a local division among the first 5 models based on age and the later 5 age models based on equivalent definitions simply using group sizes from the first K-Means based model (labeled "K_MLOG 5" on the plot). As this K-Means model is in a separate group we can see that while the UMAP algorithm was able to split Archetype 0 on its group size differences, the algorithm did not solely use group size information. This archetype has a noticeable bend in the group average halo histories, such that it is clearly based on half mass ages and similar definitions.

Archetype 1. BOW:

The first Archetype is composed of all of the BOW models. The BOW model labels are obtained by running K-Means over the 'bags' of words from the model histories, erasing long scale time information in favor of aggregating local time series information. All BOW members have one large dominate central group with at least half the histories, and two smaller more evenly sized groups. This group size disparity in the individual models is also evident in the Archetypal groups. We find the average histories to be fairly well delineated and smoother than the Zeroth Archetype averages.

Archetype 2. GMM + Dendro + Mass Loss + KMeansLinear:

The second Archetype includes a GMM based on the 'full' covariance, a BGMM using a 'tied' covariance, dendrogram models based on the Average and weighted linkages, and a K-Means model based on linear mass input. The most notable difference in these

models are the existence of a very small group in each of the models. The largest Group 1 size for any of the models included this Archetype is 1162 histories (for the GMM model 11). Group size information alone largely explains the models in this archetype. The group order of later to earlier forming halos by group number is not present here. Instead, we observe that the group 0 halos' average history (blue) is actually slightly more massive than group 1 halos' average history (green) throughout most of the simulation, although they are very similar. We additionally see a fairly large scatter among the group 1 histories, including some halos that appear to be very late forming and some that even appear to lose mass.

Archetype 3. KMeans + GMM + Dendro(Ward) + UMAP:

The third Archetype has a large variety of different models, including the UMAP model run over the individual halo histories themselves. This Archetype includes most of the K-Means models. We find a single Dendrogram based model, using the 'Ward' linkage parameter based on the incremental sum of squares distance. This archetype additionally includes both of the BGMMs using the 'full' covariance. We see a fairly large central group, with a large early forming group and a relatively small late forming group. The average histories of this group all appear to be very smooth with slightly less delineation in between them compared to Archetype 1.

2.5.6 Robustness to UMAP random seeds

We selected UMAP parameters based on a balance that seemed representative of the results in general and visually clear for the plot (i.e. parameters that resulted in points too overlapped were discarded as well those which removed local structure resulting in a "blob" like output.) Once we selected the hyper-parameters for our 'archetype' implementation of UMAP, we wanted to know if the resulting groups were a 'fluke'. To test

this we ran the algorithm over 50 random seeds to check how drastically the results varied by seed. We selected a representative UMAP sample for Figure 2.10 from the collection of random seeds, using an unsupervised method not discussed in this paper, designed to reduce selection bias.

In order to test the robustness of the groups we found from K-Means on the representative UMAP space, we also employed another method to protect against atypical seeds in the UMAP reduction. We sought to check the similarity of how often each model would be grouped together over our 50 random seeds. To test similarity among the models, we either sum and track the pairwise distances in the UMAP space between each model, or we run K-Means on the UMAP space and then sum and track how often each model was paired with every other model, ultimately creating a linkage matrix based on either distance/similarity. Choosing the first method, based on the UMAP distances, has the benefit of not relying on K-Means clusters at all, operating directly on the UMAP 2D spatial points, with a greater distance resulting in a lower similarity. Alternatively, utilizing K-Means to track the model to model similarity (to then be converted into a 'distance') better reflects how we are using the UMAP distribution, for grouping together like objects. If a model-model pair are grouped together they are given a dissimilarity value of 0, and if they are not grouped together they are assigned a 1. The summed distance is computed for each pair of models over all 50 seeds and yields a single dendrogram per linkage matrix (centroid, complete, median, ward etc). Regardless of which method we use, we ultimately find the same conclusion - our UMAP space is fairly robust to changes in random seeds. We show the dendrogram constructed from distances obtained by summing the spatial distance between pairwise points in the UMAP space. We show the single linkage, which computes the distance between clusters based on the closest two pairs, however, we found this result is robust to whatever linkage matrix we use.

2.6 How is Spatial Bias related to Groups

The halo finder code outputs the position of each halo and subhalo in the simulation along with the mass at each time step. We use the positions as input for the TPCF, utilizing `Corrfunc` (Sinha & Garrison 2020), which counts all halo pairs with distances within a specified range, DD, and compares that value to the counts expected from a random distribution of positions, RR. If a distribution of input positions has a large TPCF value at a given distance, the points are spatially overclustered. In general we refer to two regimes when discussing the distance scales, r, for the TPCF - one-halo and two-halo distances. When looking at small scales usually under 1 Mpc, distance pairs are often between subhalos of a host halo, while at larger r the function probes the distances between two halos. As our sample is exclusively dealing with host halos at z=0, we are interested in the two-halo regime. For simplicity we typically only consider a single scale range 3-10 Mpc, to probe the spatial clustering of our samples.

As we are working in a periodic box, we can calculate the counts expected in RR analytically by the square of the number of points divided by the box volume, all multiplied by the volume of the spherical shell, such that RR = $(N_{pts}^2/boxlength^3)(4/3)\pi(R_{max}^3 - R_{min}^3)$. To get errors on our two-point correlation value we used an approximation based on the square root of counts, divided by the expected random counts, Error = $\sqrt{(DD)}$/RR. This error ignore the effects of cosmic variance; however, all of our models are testing data from the same simulation box, so cosmic variance is not as relevant as error in our measurement.

2.6.1 Halo Bias

In order to see if we notice an assembly bias signal by grouping halos according to their full halo histories, we evaluate the halo spatial clustering bias of each of our different groups. Using the TPCF, we evaluate the bias as, $Bias = (\xi_{pop}/\xi_{full})$, (i.e. the subgroup population's clustering over the entire sample's clustering measurement). Our entire sample's fiducial TPCF value from 3-10Mpc is 0.27. Classically, we find that older halos are

more spatially clustered, so it is interesting to see how our groups compare to the populations split on age. Since we are only evaluating over 1 radial distance range (3-10 Mpc) for the TPCF, our bias measurement is likewise evaluated in a singular distance bin.

In Figure 2.14 we compare the TPCF values from each of our groups, for each of our 4 Archetypes, and compare that to the clustering we would get if taking the 'oldest' halo population of an equal size (according to our FHM age definition). We choose to compare to the bias from FHM age rank because we found in general this resulted in the largest bias values compared to other age ranks. If we took each of these group correlation function values and divided them by our total sample's TPCF value, the fiducial value, we would have the bias of each group. If we then divided the bias of one group to the bias of that group's respective age population, we would have the relative bias between the two groups. Since each group has the same fiducial TPCF sample, we can ignore the fiducial value entirely and just compare the relative ratios of each group's TPCF. If we find that a group in one archetype evaluates to a higher relative bias value than the oldest age halos (of a similar given size), that could suggest that our method is finding a relationship between bias and the MAH that is more natural/fundamental than age. If instead we find that our groups have a lower bias than the corresponding age based groups, then age would be shown to be a stronger proxy of clustering than the Archetypes we obtained.

In all cases involving our Archetype groups we find that group 2 halos have the highest bias values compared to the group 0 or 1 halos. We find that the zeroth Archetype based on age models, has a bias roughly equal to that obtained from a similar sized group based on the FHM age rank. We find that Archetypes 1 and 3 have all groups showing a lower bias than the FHM reference group, however, they are competitive with the PMFHM and the PHM ranks. We find that Archetype 2 is atypical, and actually has a higher bias than its corresponding oldest FHM group. We find a relative bias ratio of just under 1.2 for this group, with very small relative Poisson error. Archetype 2's other two groups are also interesting. The green, group 1 bias ratio is extremely low, below 0.1, although due

to a relatively small number of halos, this value has a larger error than other groups. In a comparison of ages, we found the group 1 halos were slightly later forming than the corresponding group 0 objects, which might explain the inversion we see among the blue and green point compared to other groups.

If we instead look model by model, at the bias ratio with equal group sizes in FHM age, as in Figure 2.15, we see a similar picture for most models. This ratio is simply each group's TPCF divided by the TPCF value obtained from the oldest of the FHM halos of equal group size. A value consistent with 1 would mean the group being shown is equally spatially biased compared to the respective oldest age group, while a value of more than one would indicate the group is more biased. As in the case with our Archetype group bias, we find in most cases, the G2 groups are relatively more biased than the G0 or G1 groups; however, we find this is not unanimous for some models, including Model 10 & 27.

One of our K-Means models based on linear MAH input, Model 10, actually shows the G1 group (green dashed, square marker), has a higher bias than its G2 and G0 counterparts and is actually the group with the highest bias ratio, at 1.4 ± 0.2. The fact that this group happens to be assigned G1, is an artifact of our model matching procedure discussed earlier, due to very discrepant group sizes. G1 has a group size of just 118 halos, which compared to the next largest group of the same model, having 5356 halos, is only around 2% in size. The correlation function becomes increasingly noisy with small data sizes. As this model resulted in the highest biased group relative to the FHM Ranked Age Bias value (of equal size), we will explore this model further in a later section. We also find two mass-loss models G2 groups (Model 24 & 26) which exceed the 1.0 threshold. At 1.17 and 1.2 respectively they do not have quite as high of a bias ratio as compared to KMeansLin G1, nonetheless, these points are actually more significant relative to the Poisson error. We find that the KMeansLin group is only 1-2 sigma away, while the two mass-loss groups are well over 5 sigma away from unity.

Additionally we find that several models (11,14,22,23,27) all have some inversions in

the relative order of their bias ratio, such that their G0 groups are closer to 1 than their G1 group. These are models associated with GMMs, Dendrograms, and Mass-Loss definitions. In Figure 2.8 we can see that most of these models have a very low match fraction with our age models, around 40%, which may explain the inversion. This inversion can happen for equal groups but is especially common for extremely disparate group sized models. We only match our groups based on achieving the highest overall match fraction, regardless of if this preserves the 'logical' order in age. We choose this order simply to achieve the highest match fractions possible and the specific group label, whether 0, 1, or 2, is not as important as the respective group populations.

We chose to look at a few of our models more closely to see how each group is distributed in age. Visually looking at the distributions confirms these groups to be noticeably more overlapped than our other models. We show a sample of the full distributions for select models in Figure 2.16. One can see the clear delineations for Models 0 and 1 in the PMFHM and FHM definitions respectively, as a direct result of their definitions, as compared with the the overlapping distributions for the summed mass-loss distributions (Model 24). Most of the mass-loss methods show similar distributions. Models 25 (not shown) and 26, the max and averaged mass-loss models had very similar distributions in age. The Arrested Development (Model 27) distributions more closely resemble those from KMeansLog (Model 5) or UMAP (Model 33) than the other mass-loss models, though still show a large overlap in ages. Although separation in age does not guarantee a large bias, they are strongly correlated values. We can see in the distributions of the KMeansLinear (Model 10), how peaked at early times group 1's ages are, especially in the PMFHM and FHM age definitions. Despite this strong correlation with age, the overlapping distributions in the normalized PDFs show this group isn't simply made of the oldest halos. The BOW method (Model 19) also appears to have a strong correlation with age, with its G2 group being noticeably older than groups with a higher relative bias. Although age is strongly correlated with assembly bias, regardless of which definition we choose, it is clear that it

alone does not fully explain the clustering signal we observe.

2.6.2 Additional Linear-Input Based Models

Due to the promising result from the KMeansLinear (Model 10) group 1's bias measurement and two of our Mass-Loss groups, we decided to evaluate a few additional methods that we did not include in our initial model list. These include: the strongest biased age model (FHM) defined using KMeansLinear group sizes, the original KMeansLinear for comparison, a BGMM, a hybrid Mass-Loss model, a Peak-minus-Final Mass-loss model, a BOW+Kmeans model, and UMAP+Kmeans model, all of which were given MAH input in the linear mass space. We could have included each of these models in our initial model list, however, the addition of too many linear models with the log-scaled models would make for a more difficult interpretation of the output. Including these additional models could complicate the local similarity structure we found from UMAP in order to create the archetypes in the first place. Had we included them all in the initial list, it's possible we may have found evidence for an additional Archetype composed of some of the linear input models, or that the additional models might cluster near their respective base models or perhaps near the mass-loss model assignments which also were given linear inputs to calculate their mass differences. In a test including these models with our initial list, we found less well defined separation among models during the UMAP dimensionality reduction.

One major difference between our current mass-loss implementations and these additional models is that we evenly subdivide the previous mass-loss groups. These additional models all allow for variable group sizes. Had we used a method like K-Means to classify the mass-loss halos from their one-dimensional rank order information the output groups could have been very different. We use a hybrid mass-loss model, which uses K-Means over each of the previous mass-loss definitions combined, albeit using a different input scaling. We started out using the same linear input as with the previous models discussed in this section, however, the resulting K-Means output groups were extremely disparate, with one

group containing just 4 halo histories. This is not enough points to perform a calculation of the TPCF, thus for this one model only we modified the input to be the *square root* of the linear MAH data, discussed. This has the effect of reducing the effect of outliers present in the MAH history containing very high peak masses, and increasing the smallest group size to near 100. We show a sample of halos for each group in Figure 2.17, where each panel corresponds to a different linear input model, and each color to one of its 3 groups/ classes.

It should be noted, the age definition does not change whether the data is given in log-space or linear, as the “half mass" value is taken relative to the actual mass regardless of the numerical representation. This means that for a log mass of 11.3, the half mass value is roughly 11.

We began by manually ordering the group sizes for our zeroth linear model, ‘LM-0’, using the K-Means Linear group sizes, such that the oldest K-Means group’s size would be assigned to the oldest ranked halos. Again, this initial ordering does not really matter, but was done for convenience to make comparisons easier between some of our model groups. Once the group sizes were established in age rank, we found that the assignment labels had a better match with our initial Zeroth Model when reassigned such that the G0, G1, and G2 labelled groups are no longer ranked in age respectively. Instead, we find that G1 corresponds to the earliest forming halos, followed by G2, and finally G0, being the latest forming group for our first four linear models. This is similar to the behavior we saw in our original KMeansLinear model, resulting from disparate group sizes.

Linear Model 0. FHM Age Based: Our zeroth linear model is based on the FHM ages, given group sizes based on the K-Means Linear model groups.

Linear Model 1. K-Means Linear Input: This is the exact same as our initial Model 10. Included here for comparison.

While we specifically required the Age0 linear model to match the KMeansLinear model's group sizes, we did not impose any such restrictions on our next model based on BGMMs. Nevertheless, when we use the linear input data for the BGMM, we find a group containing the exact 118 halos found in the previous KMeansLinear G1 group. Their other two groups share around 80% similarity respectively. The BGMM model used here is identical to Models 12 and 13, except it uses linear input and a wcp inbetween the two, equal to $10/N_{halos}$.

Linear Model 2. Bayesian Gaussian Mixture Model Linear Input: Identical to our initial two GMMs, assuming a 'Full' covariance matrix, and a wcp in between the two, at $10/N_{halos}$.

The next linear model, shown in the fourth panel of Figure 2.17, is a hybrid of the previous mass-loss models which we have already discussed. We also used the square root of the linear masses in the initial mass-loss models to be more in line with the conventional picture of mass loss while still having enough data points in each group to perform a TPCF calculation. For this additional model, 'LM-3', we combine all of the data from the previous mass loss definitions, individually normalize the data vectors, and run K-Means to produce group assignments that capture information from each definition. Essentially, this allows us to combine information of various scales, from unnormalized mass-loss definitions, into a single model. Like with the previous two models, we find one very small group, G1. This group is unlike the previous two in that the histories in it span a large range of ages, similar to what we saw in the average G1 histories from the Archetype 3 output. For this mass-loss hybrid model we do not find the G0 and G2 groups to be as well separated in age as in the linear K-Means classifications.

Linear Model 3. Mass-Loss Hybrid: A hybrid of all of our previous Mass Loss mod-

els. We combine the mass loss information from each, normalize each one using a standard scaler, and use K-Means to find groups. Mass loss values are based on the *square root* of the linear MAH data.

We also include another simple mass-loss model, which is constructed from peak-minus-final mass information. This model does not utilize the history except to find the max mass in order to calculate the peak-minus-final mass, reducing the entire MAH down to one number. The vast majority of halos have negligible values for this quantity.

Linear Model 4. Peak-minus-Final Mass-loss For each halo we compute its peak-mass minus its final mass. The top 118 halos become our peak mass loss group, while any halo with non-zero peak minus final mass is placed in a second group. Finally, all halos with zero peak-minus-final values are placed in a final group.

Additionally, we test a linear BOW + KMeans combination, similar to models 15 through 19. As the BOW groups based on the log space input all had excellent agreement, as seen by their match percentages in Figure 2.8 and their close groupings in the UMAP dimension reduction in Figure 2.10, we do not expect small changes in the hyperparameters to have a significant effect on the output assignments. For example, in additional tests increasing the number of words, W, to 500, we found a 97% agreement match with our 100 word match assignments. As our implementation of the BOW algorithm is essentially a way of discretizing an input space, it is possible that feeding linear mass input values vs log values could make a difference; however, the algorithm uses percentiles from the data to determine the discretization threshold, making the algorithm relatively impervious to monotonic data transformations. Whether we use log space data, linear space data, or even the tenth root of the data, we find over 99.9% agreement between them using identical hyperparameters. As before we choose parameters related to the discretization

process: N_L=5, L_W=4, W=100, corresponding to the letters in alphabet, letters per words, and words per history. Changing only the parameters listed above, we find an average of around 90% agreement with the previous BOW models.

Slightly complicating the explanation, the BOW algorithm also allows for the option to consider the long term data distribution across the data's discretization. This means that instead of each word being made up of only local rank order information, the 'alphabet' is composed from the entire samples MAH mass range across all time. Alternatively, one can flatten the data first by subtracting the mean, and then impose a long term trend on each history such that information about where each chunk is located in time is additionally included. Although this is not directly equivalent to using linear input values vs log input values like with the other models, encoding this long-term information in each chunk may allow for finding different features in different parts of the history, which in the case of mass accretion of large halos means across many orders of magnitude from the beginning of the simulation to the end. Assuming a relatively small scatter in values at each chunk relative to a long term positive trend observed or imposed, at late times words would only be composed of letters towards the end of the available 'alphabet'. Likewise, words at early time would be composed of letters at the beginning of the alphabet. The relative order of the letters in each chunk would still encode information of similarity among halos, allowing the classification algorithm to output group assignments. By encoding a positive long term trend across time, it does not increase the importance of late time mass, but instead now favors using higher threshold letters in late time words. For example, assuming a three letter alphabet [A,B,C] and three letters per history, the average history in the non-trend version would be (B,B,B), while under the trended version it would be (A,B,C). When using this longterm trend information in the BOW discretization, we find only around a 65% match with our other BOW models, even as the groups appear visually to be very similar to the previous cases. For our comparison with the other linear input models, we utilize this long term data distribution option.

Linear Model 5. Bag-Of-Words Modified: We modify our previous BOW implementation by considering the entire data structure when creating the alphabet of letters. This has the effect of preserving some information regarding relative positioning of words. We use N_L=5 letters, and L_W4 letters per word, with W=100 words per history.

Our final linear input model, is also based on K-Means assignments, this time over a UMAP reduced space similar to our initial model 33. We found this model to have very similar group sizes and separation among histories to the BOW linear model, with group sizes (3474, 4558, 3647) for G0,G1, and G2 respectively. In this case one can see more importance towards separation at the end of simulation (0 on the x-axis) compared to previous models. The G0 group shows a strong preference towards late forming histories, and lower final mass halos, however it still samples final masses from across the whole range. In one case, we see a G0 (blue) history actually having a rather large peak mass, over 11.53 $log(M_{sol})$, losing mass to get to our final mass bin value. This odd history appears in each of these linear models' group 0. Other than this apparent mass loss halo, the UMAP+KMeans sample histories, shown in the last panel of 2.17, show a surprising lack of highly peaked histories. Compared to the first four panels of linear models which all show many mass loss halos in green, both the blue and green halos shown for the final model are relatively smooth. It should be emphasized that the mass-loss histories still occur in the UMAP+KMeans and BOW+KMeans groups, however, due to their rarity among their sub-populations, they were not chosen in the random sample.

Linear Model 6. UMAP Linear Input: Identical to our initial UMAP model 33, we instead feed in un-normalized Linear MAHs.

With these additional linear models, it is clear from Figure 2.17 that the algorithm used

to classify histories can drastically alter the nature and sizes of the groups found. When using linear space MAHs we find some models tend to pick out a few very early forming, extreme mass loss histories, with two smoother groups largely separated in age. For the models which compute differences or similarities in linear space, halos which happen to exceed the final mass can do so by a significant amount. Although our highest mass final mass halo is $\log M = 11.5$, the highest mass recorded by any of our histories is $\log M = 12.15$. In log space this difference is just 0.65, while in linear space this corresponds to a magnitude difference of 1.1E12, greater than our final mass bin! This huge difference only exists for a relative few number of histories and thus in an algorithm like K-Means, which seeks to minimize the within-cluster variance, groups together these relatively different histories, while separating the more typical histories into two cleaner groupings with a smaller standard deviations for most time-steps. Comparing the KMLIN and BGMMLIN models to age, we see that half mass age alone does not fully explain the groupings, with the former models finding intense mass loss MAHs.

Despite different fundamental algorithms at play, finding the same halos in a group is unlikely to be an accident, pointing to a more significant separation than if only found by one model alone. The first three additional models are noticeably different to the last two, with the mass loss hybrid in its own class. The BOW algorithm itself does not actually factor in monotonic scaling in the data, and thus does not change simply receiving linear data; however, by allowing the discretization process to have knowledge about the relative rank information across time, we were able to find relatively separated groups in the linear space. The groups found by the BOW model visually look very similar to the UMAP+KMeans Linear groups in linear space and have very similar group sizes. The UMAP algorithm computes distances in the original space, and then tries to preserve the mappings in a lower dimensional space. As these original higher dimensional distances are computed in the linear space the UMAP algorithm has knowledge of the input space's scaling; however, in the 2 dimensional representation extreme halo histories in the full di-

mensional space become more similar, especially given a low value for the *n_neighbors* parameter. If the algorithm does not create well separated clusters, the K-Means algorithm itself will roughly evenly subdivide the lower dimensional space across its longest axis. Tweaking the UMAP n_neighbors parameter could slightly modify these groups, however, our tests showed relative consistency varying the parameter from 10 to 100.

In Figure 2.15 we compared the spatial halo bias of each group in our initial models, with a group made up of oldest age halos of similar group size. Likewise, we compare the bias ratios of our additional linear models and age in Figure 2.18, such that a value over 1 corresponds to a correlation function value for the model group larger than the age group. The seven linear models are listed in order with each dot representing one of their groups. The FHM based 'LM-0' model's group G1, is defined to be at 1.0, being made up of the oldest ranked halos, however, with just over 100 positions there is a fair amount of scatter associated with the bias measurement. Obviously, the highest biased group for the KMLIN and the BGMMLIN model are both equal, with a ratio of around 1.4 times the bias of the respective individual final age group. Despite containing some extreme mass loss halos, the Hybrid Mass Loss G1 group, (LM-3), shows an 'inversion' in relative bias compared to the high bias we saw in the previous models G1 group, with all three MLHybrid groups having low relative bias. This hints that just combining mass-loss events, particularly how we have defined them, across time does not necessarily result in higher spatial clustering. Our LM-4 model, based on peak-minus-final mass-loss has a relative bias ratio above 1, yet is consistent with 1 within the Poisson error. Finally, we see very similar behavior from the BOW and UMAP based linear models, such that neither contain a group more biased than the comparison age groups. While their G2 groups are nearly as biased as age, their G0 groups are less spatially biased than we find with age. These groups are composed of very late forming halos experiencing relatively slow growth until a scale factor of around 0.6, when they rapidly gain mass, occasionally greater than their final mass.

2.6.3 TPCF by Group Size for Various Models, Peak-Final, & MAH Average

In looking at various linear input based models, we find a small number of halos are contributing to a large bias signal, consistent with previous findings looking at age based groups. In addition to typically being earlier forming, we find the vast majority of the histories in the most biased groups appear to be losing mass. We revisit the idea of mass-loss halos based only on peak mass minus final mass. As mentioned when constructing LM-4, the distribution of $sorted(M_{peak} - M_{final})$, is extremely steep at the tail, as most halos have 0 (or near 0) peak-minus-final mass-loss. Of the halos that do experience this type of mass loss, most are at the sub-percent level. A very small fraction of the halos experience significant mass-loss compared to that halo's final mass.

Instead of dividing the peak minus final mass loss halos into a single model, here we vary the size of most mass-loss group, and roughly evenly divide the remaining two groups. We then compute the bias measure for each of the most mass-loss halos, and compare these values to the bias obtained from our most biased group from the KMLIN model and a few other models. In Figure 2.19, we show the correlation function values as a function of the group size (top panel), corresponding to the highest mass-loss histories (red points). Additionally, we show the same thing for FHM age ranked halos (black points, dotted line), varying their group sizes as well. Finally, we introduce the last model, one of the simplest models possible, which we are calling 'MAH Average' (magenta diamonds). We find the average value for each history (one value per history) and rank them; to create a group of size N, we take the highest ranked N objects. In the lower panel of the figure, we show the difference between each Model's TPCF value and the corresponding FHM value, all divided by the averaged error of the two respective groups.

We can see in Figure 2.19 that generally the TPCF values rises with smaller group sizes. While it is possible to find FHM Age groups and Peak-Final groups which are both more spatially clustered than our KMLIN group, in all cases this results in a fewer number of halos in each group. A downturn in the FHM's curve only happens at extremely small

group sizes, while the Peak-Final and MAH Avg. curves appear to keep increasing down to the smallest group size bins. The MAH Average model are nearly as clustered as KMLIN at similar group sizes, and are more significantly clustered at even higher sizes. In nearly all cases below a group size of 1000, the Peak-Final values are more spatially clustered than the similarly sized FHM Age groups, going slightly below at larger group sizes. The MAH Avg. groups are more clustered in the size regime near the K-Means point than FHM or Peak-Final, however, this simple group falls significantly below both near 1000. At large group sizes, roughly equal to one-third of our total sample, the summed mass-loss group (black) is the most significantly over biased, closely followed by the averaged mass-loss group (grey).

2.7 Additional Halo Properties by Grouping

Now that we have several different halo groups from a variety of different algorithms, it may be interesting to look at how these populations differ in halo properties besides their MAHs. In Figure 2.20, we show the distribution of certain halo group properties based on the additional linear models we evaluated. In the top row we show the concentration observing that the high bias groups are also most concentrated. We see very little concentration separation among the group 1 and 2 (green and red) distributions for the last two models based on BOW and UMAP respectively. Although we see a strong correlation with the median of the distributions of concentration and high bias, the G1 KMLIN and the G2 MLPeak groups still show a wide distribution across all concentrations, even more so than we see in the G1 Age1 group. In the second row we show a component of concentration, halo scale radius. The halo scale radius is based on N-body simulations by Navarro, Frenk, & White (1997, NFW) who found that CDM halos can in general be approximated by a two-parameter profile: $\rho(r) = \frac{4\rho_s}{(r/Rs)(1+r/Rs)^2}$. Due to the presence of extreme valued outliers, the Rs distributions are mostly bunched up at low values so we exclude the very high Rs tails in the figure, and focus on the region of interest. The third row shows the scale of last

major merger, where we see a large scatter of values among the most biased groups. The MLhybrid G1 shows halos which preferentially had a very early time last major merger, while showing very little separation in concentration (and radius) and its other two groups. For BOW and UMAP the G0 and G1 distributions nearly overlap for the scale of last major merger, in general having more recent major merger events than their G2 groups. In the fourth row we look at the distribution of halo V_{max}, and find a very high V_{max} among the most biased groups, as expected. The KMLIN and BGMMLIN G1 group distributions are slightly more centrally peaked and separated than the age based group, and they are a different shape than the MLPeak G2 distribution. We find the BOWLIN and UMAPKMLIN distributions are quite similar, characterized by overlapping Gaussian distributions staggered fairly evenly across V_{max}, with the highest bias group, G2, having the largest V_{max}. Among spin, shown in the fifth row, we do not see much separation across our groups with only a very slight preference towards higher spin for our high bias groups. The MLHybrid and MLPeak groups appear to be some of the most overlapped according to their medians, indicating very minimal effect of high variability mass loss on the Spin parameter of a halo. Interestingly, in the sixth row, looking at tidal force, we see a bi-modal distribution with very low near-zero force for some, likely isolated, halos, and then a population with stronger tidal forces. The most biased groups have very few zero-tidal-force halos and instead show preference towards large tidal forces, especially the MLPeak G2 (red) population. The MLHybrid model G1 population shows a fairly strong tidal force, despite exhibiting relatively low bias. We find the BOW and UMAP distributions are nearly overlapping in Tidal Force. We also show the distributions for T/U, a ratio of a halos kinetic energy to its potential, where the most bias halos appear to have slightly lower values of T relative to $|U|$, however there is very little signal in most populations, with the MLHybrid G1 group showing the largest median deviation from its other two groups. Finally, in the last row we show another value for radius, this time the half mass radius, which again, shows a preference towards smaller size for our more biased halos. We do not observe any

notable distinctions among the two radius definitions, Rs nor R0.5, despite covering quite different scales.

2.8 Summary & Conclusions

We began this project considering the halo bias to be a major motivation in pursuing alternative groupings for halo assembly histories from a high time cadence simulation. In this paper we group halo mass accretion histories together, into 3 groups, based on a variety of algorithms and methods. For each of our groups, we compared the (TPCF) bias of the halos spatial positions, with a group of the same size made up of the oldest age halos (our fiducial model). Due to the nature of the TPCF, we had to avoid supervised learning methods in constructing our initial groups. Once we had obtained our initial groups, we evaluated several other groups, including Archetypal groups designed to reduce the number of models to just a handful, and finally, we evaluated several alternative models at the end of the paper. Some conclusions of the work are as follows:

- We find that choices in data pre-processing can have a large effect on which groups are formed, especially whether the input data is logged or not.

- Although we set our group assignment labels based on the highest fraction of matching labels with our zeroth model, half mass age, this somewhat unintuitively, can result in groups that do not necessarily align in order of age.

- Using a variety of different methods results in a large diversity of different groupings, with the average match being only 56% among all halo assignments. We find 4 similar groups of models, we call Archetypes, that are largely based on methods and pre-processing choices.

- We find to first order that the model type matters more than the individual hyper-parameters, with age methods and BOW methods forming tight groups. Dendrogram methods were more dependent on the linkage method for determining similar groups.

Whether the initial mass data is given in log or linear space, which we investigated with Model 10 and mass-loss models, is very important for finding high bias objects.

- Generally, we found the oldest Final Half-Mass (FHM) age group (where the half-mass threshold was set by a halo's individual final mass), was more biased than the oldest groups based on the population mean final half mass (PMFHM) or 'peak' half mass (PHM) ages. As seen in Figure 2.18, our K-Means group, input with the linear mass values (Model 10), and a Bayesian Gaussian Mixture Model, input with linear mass values (Linear Model 2), both produced a group of relatively old halos with the same 118 histories, which were more 1.4 times more biased than a similar sized group based in age, roughly twice the Poisson error. Additionally, we find that two of our mass-loss methods (Models 24 & 26) constructed from the summed mass-loss events and the average mass-loss events, produce groups that are significantly more biased than age, as shown in Figure 2.15.

- At comparable group size, our K-Means Linear G1, is more biased than our other groups, as seen in Figure 2.19, however, Peak-minus-Final mass loss has the highest clustering value at the smallest group sizes. Simply taking the average of each halo's history works surprising well at picking out high biased halos at small group sizes between 100 and 300. SumML and AvgML G2 groups had the most significant bias when weighted by the Poisson error.

- We find our most biased halos tend to exhibit higher concentrations, V_{max}, and tidal forces. Tidal force appears to be slightly more important than former properties. The scale of last major merger is not a good indicator of enhanced bias as seen by the MLHybrid G1 group. We see that high biased groups tend to have smaller Rs and R0.5 values. A high V_{max} alone does not guarantee a large bias value, as seen by the UMAPKMLIN method, which has a fairly large over-representation of high V_{max} objects in its G2(red) group, despite a moderate bias measurement.

At the start of the project, we would have considered it a huge success to find an algorithm or group of algorithms that could cleanly segregate halo into biased and anti-biased groups based on their MAHs, yielding an archetype group that evaluated to a higher halo bias than a similarly sized group created from the oldest halos by half mass age. In our initial run, we found that only groups that used linear input MAH data evaluated to bias higher than our fiducial age groups, and only one of our archetype groups, based on mass-loss contained a positive bias ratio with age. We stumbled upon two algorithms which gave the same 118 halos which were more biased than the respective age based groups, and were more biased than the 118 halos with the largest peak minus final values. While peak-minus-final mass loss certainly is a factor, it does not appear to completely explain the group 1 halos found by KMLIN and BGMMLIN. Surprisingly, a model as simple as using the average MAH value for determining group membership, can produce highly biased groups in some regimes.

Using a large suite of unsupervised algorithms and their hyper-parameters to try to find natural groupings in the data is not without challenges. We found the results using our MAH input are fairly robust to minor changes in which models are included or the exact halo mass range being tested, but downsampling the input can produce dramatically different results for some models, while hardly changing others. For some models that do not automatically rescale the data input, choices in pre-processing can have significant consequences on the output. Occasionally the random seed used for an algorithm will produce sub-optimal results (e.g. the algorithm will not converge) and individual tuning by hand may be required. Due to the drastically different group sizes, matching between models is not straight forward, making comparisons between like groups difficult. We optimized the group order based on match percentage, however, for some projects it may be better to optimize on mean halo age or any other parameter of interest. The loss in flexibility over group sizes makes comparisons with other groups very restrictive. We recommend that one explores over all reasonable options when using unsupervised methods to explore

a data set, especially if the data is not easily separable and metrics like the GAP statistic or BIC are inconclusive. Using unsupervised machine learning can help to automate the process of finding relevant groups for a problem such as calculating the TPCF. Ideally, a much larger grid of models could be set up systematically to explore how similar algorithms or groupings of halos affect group emergent properties such as halo clustering.

This research has made use of NASA's Astrophysics Data System. This research made use of Astropy, a community-developed core Python package for Astronomy (Astropy Collaboration et al. 2018, 2013) This research made use of SciPy (Virtanen et al. 2020) This research made use of Scikit-learn (Pedregosa et al. 2011b) This research made use of NumPy (Harris et al. 2020) This research made use of matplotlib, a Python library for publication quality graphics (Hunter 2007) This work made use of the IPython package (Pérez & Granger 2007) This research made use of pandas (McKinney 2010, 2011).This research made use of itertools (Van Rossum 2020). This research made use of fastcluster (Müllner 2013). This research made use of (McInnes et al. 2020). This research made use of Corrfunc, a 'blazing fast' solver for the correlation function (Sinha & Garrison 2020). This research made use of the Vishnu simulation, provided by Manodeep Sinha. Some of the computation for this work was completed on the Advanced Computing Center for Research and Education (ACCRE) at Vanderbilt University. N. Chason was supported with funding provided by Vanderbilt University.

Table 2.1: List of Initial Methods

The first and second column, i & Method, correspond with the model number and abbreviated name. The third column lists some notable parameters for each method. The fourth gives the group sizes for group G0, G1, and G2, matched to the zeroth model respectively (described in text). Finally, the fifth lists our approximate run time between models classifying the MAH data. notes: parameters are described more fully in the text. It's assumed all methods use K=3 (or equivalent 'K' parameter), resulting in 3 groups whose sizes are listed in column 3 in order from G0 to G2. Run times are hardware dependent and should only be considered relative to each other. Quotations in a column indicate a similar value to the previous row's entry.

i	Method	Parameters	Group Sizes	Approx. Run-Time
0	PMFHM	$Age_{1/2}$=0.5	3893, 3893, 3893	0.4s
1	FHM	$Age_{1/2}$=0.5	3893, 3893, 3893	0.3s
2	PHM	$Age_{1/2peak}$=0.5	3893, 3893, 3893	0.5s
3	Avg_PMFHM	Age_{x}=[0.2,0.6]	3893, 3893, 3893	0.5s
4	Avg_PHM	Age_{xpeak}=[0.2,0.6]	3893, 3893, 3893	3s
5	K_MLOG	(log input)	2365, 5262, 4052	7s
6	K_MPCA	variance=0.95	2373, 5264, 4042	" "
7	K_Mnorm0	$Axis_{Norm}$=0	2426, 5233, 4020	" "
8	K_Mnorm1	$Axis_{Norm}$=1	2223, 5085, 4371	" "
9	K_Mdesc	'desc ID' MAHs	2418 5286 3975	" "
10	K_Mlinear	(non-log input)	6205, 118, 5356	9s
11	GMM-Full	tol=10^{-3}, regco=10^{-6}	3880, 1162, 6637	20s
12	BGMM-Full1	wcp = $\frac{1}{N_{halos}}$	1740, 5133, 4806	25s
13	BGMM-Full2	wcp = 10^{4}	1739, 5161, 4779	" "
14	BGMM-Tied	wcp=1	4059, 677, 6943	30s
15	BOW1	N_L=3, L_W=4, W=50	3386, 5302, 2991	8s
16	BOW2	N_L=3, L_W=8, W=50	2940, 6079, 2660	25s
17	BOW3	N_L=3, L_W=8, W=25	2262, 7221, 2196	20s
18	BOW4	N_L=5, L_W=4, W=25	1938, 7978, 1763	7s
19	BOW5	N_L=5, L_W=4, W=50	2345, 7260, 2074	10s
20	DEND_Ward	Linkage=Ward	3289, 4131, 4259	3m 30s
21	DEND_Avg	Linkage=Average	897 , 171 , 10611	" "
22	DEND_Comp	Linkage=Complete	9680 , 114 , 1885	" "
23	DEND_Weight	Linkage=Weighted	8369 , 402 , 2908	" "
24	SumML	(non-log input)	3893, 3893, 3893	4s
25	MaxML	" "	3893, 3893, 3893	" "
26	AvgML	" "	3893, 3893, 3893	" "
27	ArrDevML	T_{ML}	3893, 3893, 3893	" "
28	PMFHMvar	$Age_{1/2}$=0.5	2365, 4052, 5262	0.3s
29	FHMvar 0.3s	$Age_{1/2}$=0.5	2365, 4052, 5262	0.4s
30	PHMvar	$Age_{1/2peak}$=0.5	2365, 4052, 5262	0.5s
31	Avg_FHMvar	Age_{x}=[0.2,0.6]	2365, 4052, 5262	0.5s
32	Avg_PHMvar	Age_{xpeak}=[0.2,0.6]	2365, 4052, 5262	3s
33	UMAP	$N_{Neighbors}$=15	3676, 4621, 3382	16s

Table 2.2: Archetype Groups

Archetype	Model Numbers
0	0 1 2 3 4 27 28 29 30 31 32
1	15 16 17 18 19
2	10 11 14 21 22 23 24 25 26
3	5 6 7 8 9 12 13 20 33

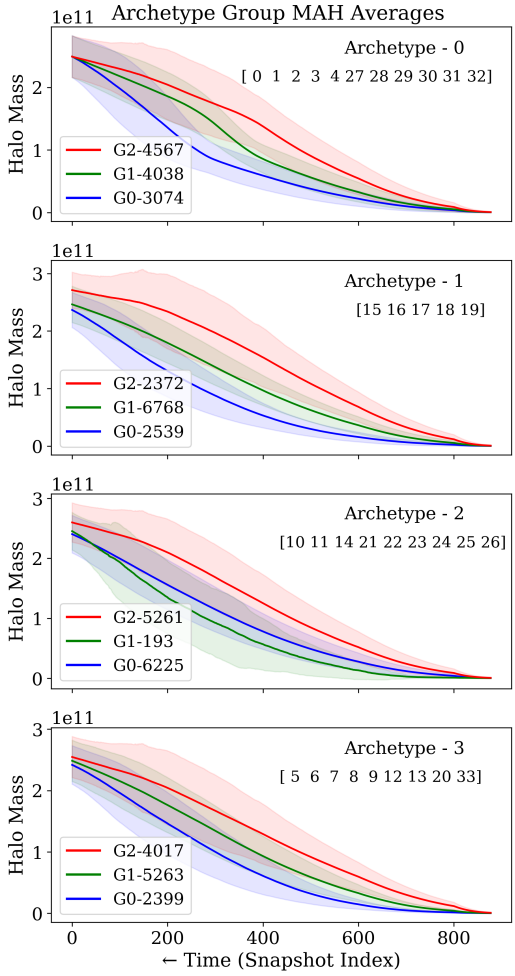

Figure 2.12: Average MAHs for the three groups, as found by the four 'Archetype' models (four panels). The red, green, and blue line in each panel shows the average MAH of halos in the G2, G1, and G0 groups, respectively, while the shaded regions show the standard deviation of halo histories in each of the groups. The three groups roughly correspond to the latest forming, average forming, and earliest forming halos, except for the G1 group in Archetype 2. The group sizes are listed in each panel's legend. In the upper right corner of each panel, we list the Models (0-33) that contribute to each archetype (also listed in Table 2.)

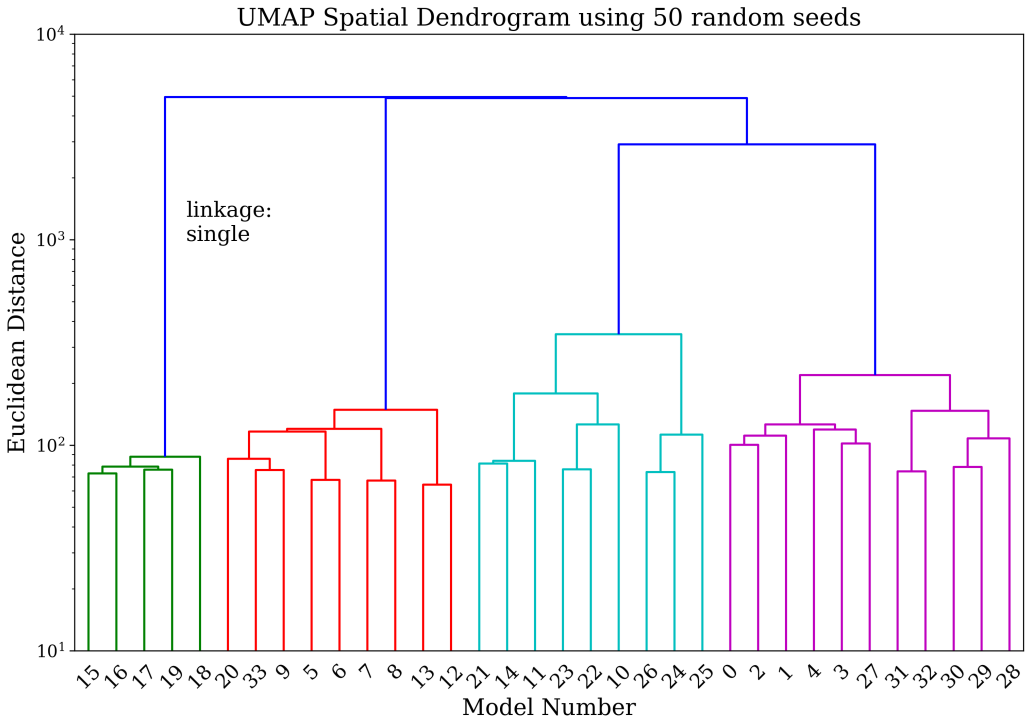

Figure 2.13: Dendrogram showing the hierarchical clustering of the similarity of our initial 34 model assignments as grouped by UMAP, using 50 random seeds to test the robustness of our Archetype groups. We set a distance threshold equal to 1000 to highlight 4 distinct groups, shown in green, red, aqua, and magenta. These groups correspond to Archetypes 1, 3, 2, and 0 respectively from Fig. 2.10. Although we utilize the single linkage in this figure, we find the same groupings regardless of the linkage type. We show the y-axis on a log scale to better highlight the intracluster nodes at small distances.

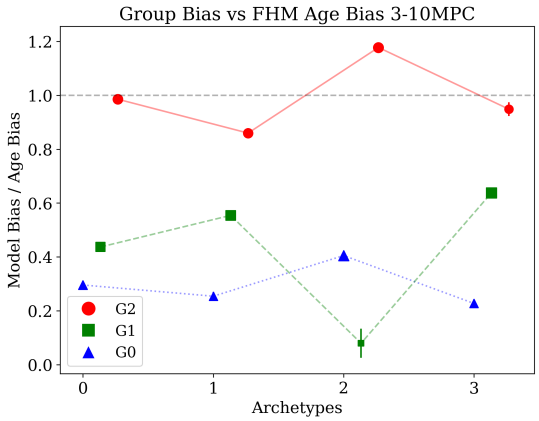

Figure 2.14: Relative spatial clustering bias between each Archetype group and a sample of oldest halos according to age. Specifically, the clustering strength of a given group is calculated as the average value of the TPCF (DD/RR-1) in the range of scales 3-10 Mpc. This is then divided by the clustering strength of an equal size set of oldest halos according to the FHM method (Model 1). A value above 1 (horizontal grey dashed line) thus indicates that the model group is more clustered than the corresponding FHM group. Groups G0, G1, and G2 for each Archetype are represented by a blue triangle, green square, and red circle, respectively. Points also show Poisson errors, which are in most cases too small to be seen. The four Archetypes are shown at positions 0-3 on the x-axis, with small amounts of staggering added in for clarity.

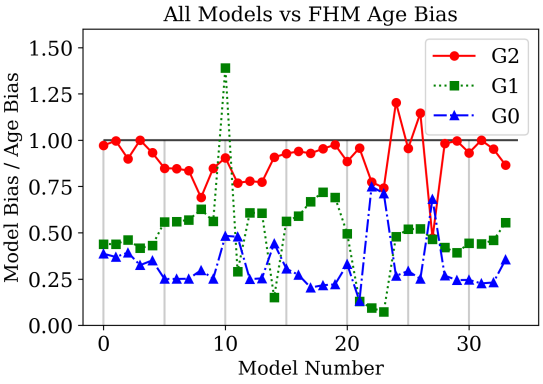

Figure 2.15: Similar to Fig. 2.14, except that we show results for all 34 original Models rather than the four combined Archetypes. Vertical grey lines designate each 5th model to help guide the eye. The G1 group in Model 10 displays the strongest spatial clustering relative to halo age; however, that model's large Poisson error (not shown) makes this clustering ratio less significant than those for Models 24 and 26.

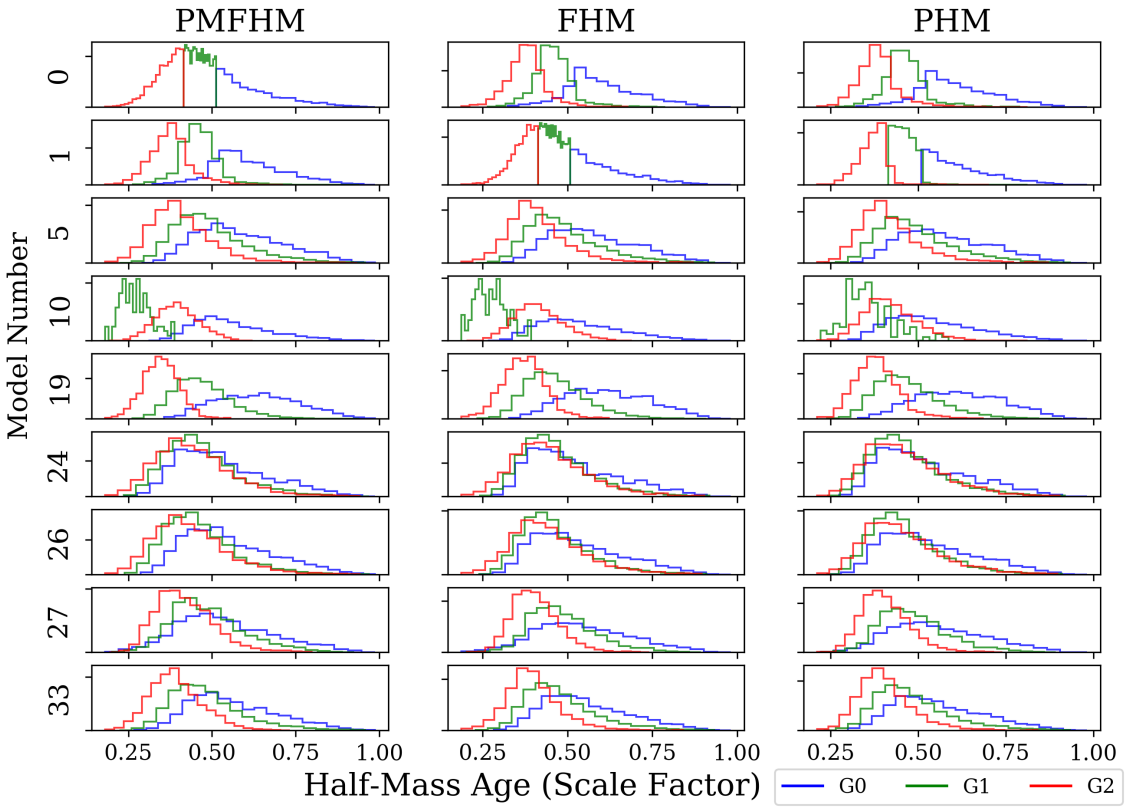

Figure 2.16: Halo age distributions for the three groups identified by each of nine selected models (rows) and for three age definitions (columns). The model numbers on the y-axis correspond to the following respective models: PMFHM Age (0), FHM Age (1), KMeansLog (5), KMeansLinear (10), Summed Mass-Loss (24), Avg. Mass-Loss (26), Arrest. Dev. (27), and UMAP (33). The (blue, green, red) colors in each panel correspond to the (G0, G1, G2) model groups, respectively.

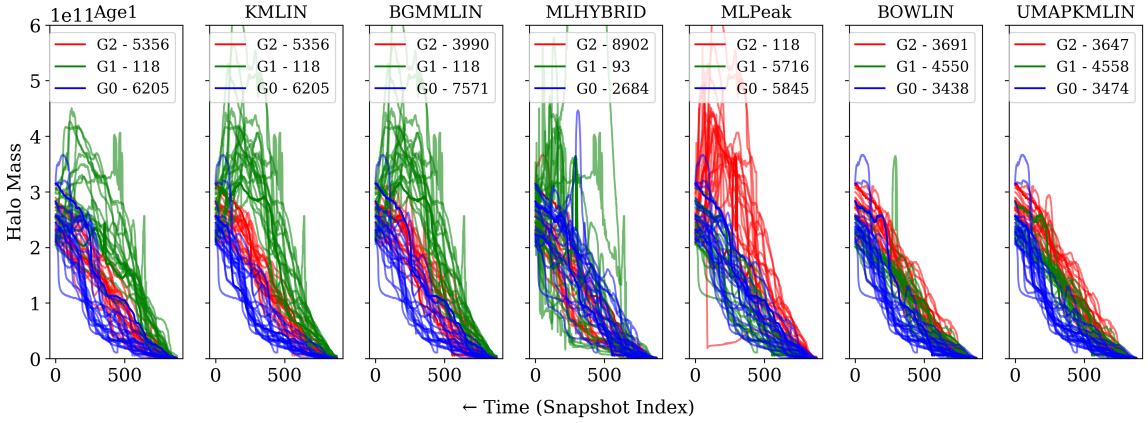

Figure 2.17: A sampling of MAHs drawn from the three identified groups G0, G1, G2 (blue, green, red lines) identified by our seven linear models (panels). Unlike in previous figures, MAHs are shown in linear mass space (y-axis). The Linear Models are described in the text. Group labels and their respective group sizes are shown in each panel.

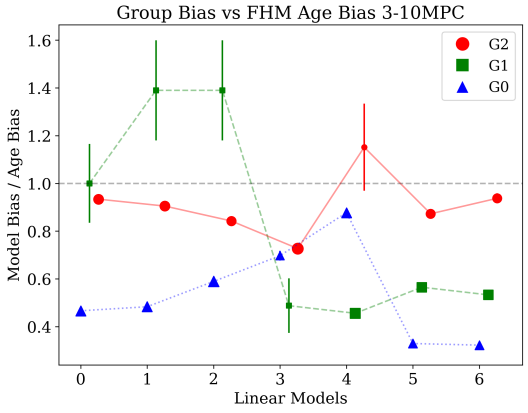

Figure 2.18: Similar to Fig. 2.15, except for our additional linear models: Age1 (0), KMLIN (1), BGMMLIN (2), MLHYBRID (3), Peak-Final (4), BOWLIN (5), UMAPLIN (6). Error bars show Poisson uncertainties in the clustering ratios.

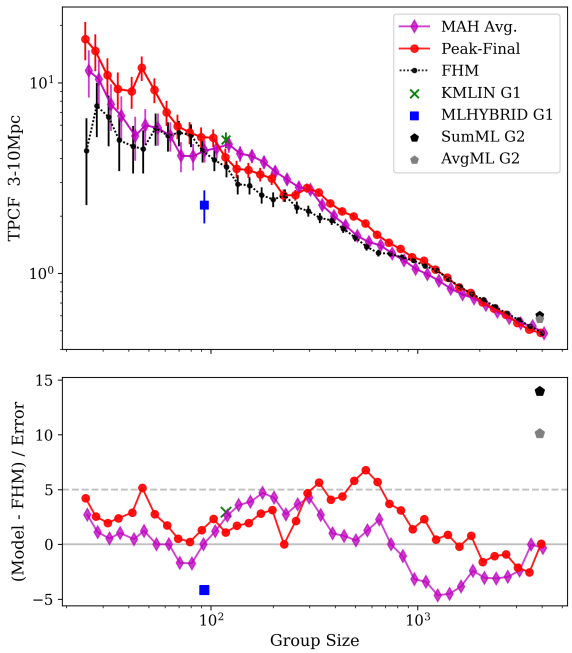

Figure 2.19: Correlation function values over the radial range from 3-10Mpc, by group size. We show the KMLIN most biased group (green 'x'), with a group size of 118, and a range of group sizes for halos based on Peak-minus-Final mass-loss (red curve), and FHM half-mass age (black curve). We also show the small G1 group from the Hybrid mass-loss model (blue square), and the G2 groups from the summed & average mass-loss methods at large group sizes (black and grey pentagon marker respectively). *Bottom panel:* We show the same data as above, referenced to the FHM data and normalized by the propagated error between each model and the FHM model.

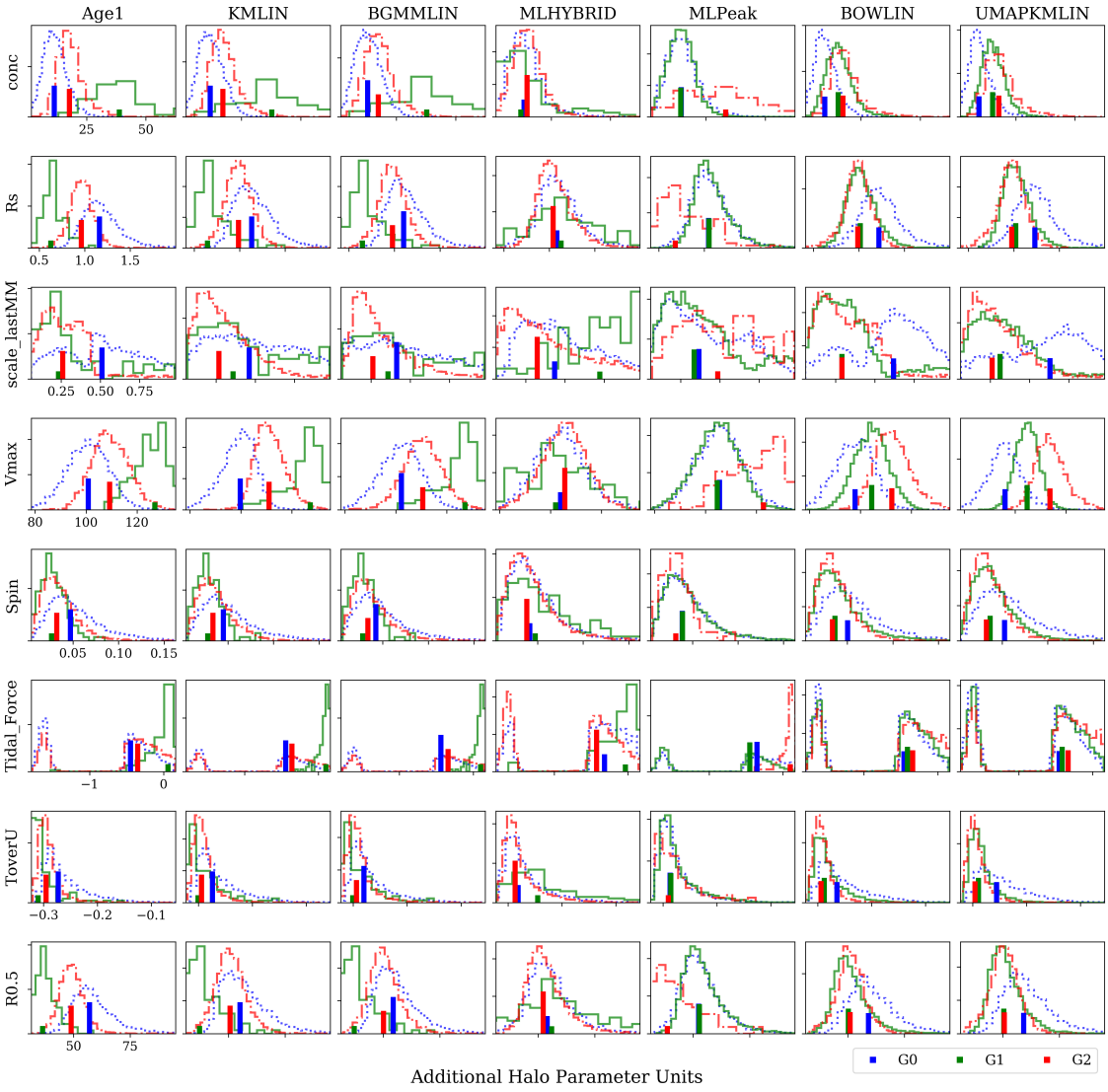

Figure 2.20: Distributions of several halo properties (panel rows) for the three groups identified by each of our seven linear models listed in Section 2.6.2 (panel columns). The x-axis in each row of panels shows a particular halo property. From top to bottom: concentration, scale radius, epoch of last major merger, maximum circular velocity, spin, tidal force due to nearby structures, ratio of kinetic to potential energy, and half mass radius. Groups G0, G1, and G2 are shown with blue, green, and red lines, respectively. All distributions are normalized to the unit area, and have varying bin widths due to variations in group sizes. We show each group's mean with a vertical colored bar whose height is proportional to the group size.

Chapter 3

Mass Accretion Grouping Analysis: Extensions

There were several aspects in Chapter 2 that we were not able to explore in depth. Included below are several additional aspects we considered when creating our model groups or choosing our model parameters & prepossessing. First, we begin with a discussion of Dynamic Time Warping (DTW) and why we chose not to utilize it in the paper despite it being used very successfully to match time-series data in other fields. Second, we show some results from t-SNE and comparisons with UMAP. Next, we show some additional results from our previous clustering models, including a discussion of the "Boosting Factor" which allows for comparisons of group halo property differences among groups of different sizes. We conclude the section with a brief discussion of alternative approaches to unsupervised learning, including supervised learning, random walks, and environmental classification.

3.1 Dynamic Time Warping

Previously, we explained how DTW allows for the remapping of two time-series vectors to be better aligned in time. Whereas a convolution will essentially 'slide' or convolve two vectors across each other, DTW instead allows for multiple time series manipulations to better align the whole curve. A convolution is therefore good for finding a single 'feature' (however complicated), at the point of greatest signal, while DTW is better at matching several features at potentially shifted times. Assuming two sine waves specified over some interval, exactly 180 degrees out of phase, both a convolution and DTW would be able to realign the waves, matching peak to peak, rather than peak to trough. A convolution would fail, however, if instead of a uniform sine wave, we had sine wave broken up at a few random intervals. While the convolution might align the largest peak, or perhaps a few spurious peaks by chance, DTW would allow for a larger degree of freedom across time to match multiple peaks with like peaks. This becomes very useful when looking for specific

signals, distributed around varying lengths of no signal, or when looking for similar signals of different frequencies.

DTW works pairwise, by creating a distance matrix for each two vectors it compares. Given two MAHs, X and Y, DTW will create a distance matrix which computes the distance between every vector instance in X and Y. Assuming there are 1000 time steps per history, this will result in a 1000x1000 matrix of distances for just this one pair of halos. This must be repeated for every halo distance pair we wish to consider. Once we have a distance matrix, the DTW algorithm starts at the origin (bottom corner) of the matrix representing X_0 & Y_0 and progressively works its way up to the end of both histories (top corner) X_{1000} & Y_{1000}, selecting each matrix cell in the DTW path based on the next minimum distance pair, adding each subsequent 'distance' to the total distance between the histories.

It is possible to have multiple points in one data vector be matched to one point in another. The DTW path chooses the next step based on minimizing the distance between each point X_i and each Y_j. If there are no restrictions on the DTW path, a point at the beginning of one history *could* end up being DTW matched with a point at the end of the comparison histories. Not only does this result in some strange comparisons when thinking about the implications for MAHs, it is also very computationally expensive, having to search over the entire 1000x1000 space when finding the optimal DTW path. Instead, a restriction can be placed, so that each point can only be matched within some matching window, specified by some number of time-steps away from the comparison location. The proper choice for this window is project dependent and may be more art than science.

Although we did not utilize DTW in the previous chapter, we do wish to show how it can be used with MAHs. Its important to remember that DTW distances are pairwise, meaning that each halo will have a different 'warping' depending on which halo it is being compared with. This makes matching difficult for many of the algorithms discussed in Chapter 2. DTW could be used for algorithms that pair halo histories with sample 'means' or, potentially, to build certain linkage matrices for dendrograms, where the number of

evaluations for the DTW distance would be minimized. One major issue with using sample 'means' to group items with DTW is that the mean may erase information used initially in minimizing the distance between a given halo pair. Rather than a single population mean, a comparison could be made with a representative sample halo, or generated through the use of a Gaussian Process (GP). We avoid this issue entirely for the rest of this paper by only considering single-pair distances as we do not use DTW for clustering, instead showing how DTW can result in different pairwise distances.

We begin with showing a sample of two down-sampled histories being matched/ linked with DTW, in Figure 3.1. Halo history A and B, are shown in blue and green respectively in the plot, with their DTW map shown in red. These links between A and B, give the warped path of history B with respect to history A. Initially at the origin, which actually corresponds with late times in the halo histories, the MAHs are paired by definition. However, even out to a time index of 10, we see points from B are being paired very near the origin point in A. As we progress to higher snapshot indices, we see this trend persists of higher snapshot index B points being matched to lower snapshot index A points (this corresponds in age with earlier values in B being matched to later values in A). At roughly snapshot 30 we see many A points matching to a single point in B, due to their relatively close proximity in mass value. The DTW mapping ends at (50,50), with the final two points, which are actually rather far apart in mass, but are matched nonetheless as it is the end of the history. Another way to visualize this DTW mapping is with a plot of the path as in Figure 3.2, showing where each history is linked, starting at the the origin and ending at (50,50). In this case, Halo 1, or A, is linked to Halo 2, or B, at the origin, and again at (50,50). Near the end, the paths progress roughly linearly - meaning that each point A_i is matched to each B_j where i roughly equals j. This behavior exists only at the first several final time indices (near 50, corresponding to the early part of the simulation). At the start of the DTW path map, the path is near horizontal, meaning many points in B correspond to a single point in A. Similarly, at several instances for Halo 1, (x-axis), near the origin, 25, and 85, we

observe vertical spikes, where multiple points on B, are being matched to A, just as we observed in the previous figure. The total DTW distance would be the summed squared distance between each A_i and B_j listed in the path. With this method, points do not have to be close in time to achieve a low distance; they must, however, be relatively close in mass in the same given order as the comparison vector. It is hard to gauge simply by looking at the path or the linkages, to know whether a history will be minimized with DTW. The human eye can pick out relatively simple similar vectors without a need for such a method, but as the data size increases, this task becomes more difficult without automation.

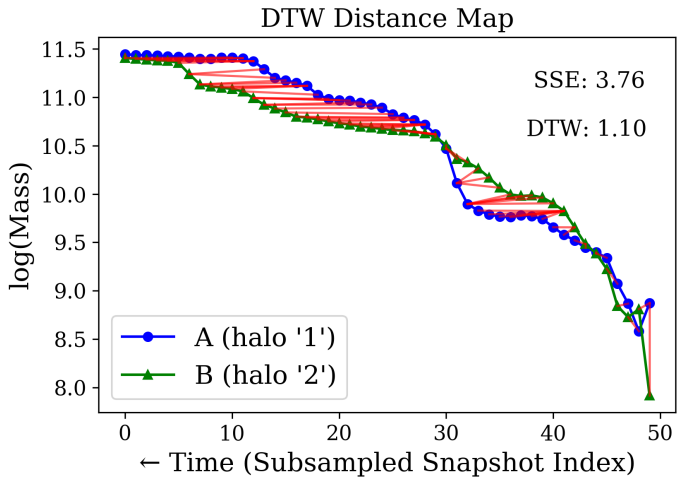

Figure 3.1: A comparison between two random histories (labeled A & B (blue circle, green triangle respectively), and their DTW path map, shown by red links between respective timesteps of A and B. This DTW is performed on downsampled versions of the halo histories containing 50 points per history, rather than the near 1000 points in the full histories.

We show a comparison of the DTW distance, and the standard Euclidean Distance (with no DTW), in Figure 3.3. For each labelled point, we are showing the DTW distance with halo history '0' on the y-axis, and the corresponding euclidean SSE distance on the x-axis. At the origin is the '0' halo, as it perfectly matches in both the DTW distance and SSE distance with itself. In general, we observe a weak correlation between the two distances. A very tight correlation might imply little reason to explore DTW distances, unless there were some points being poorly matched with SSE. A weak correlation does not necessarily

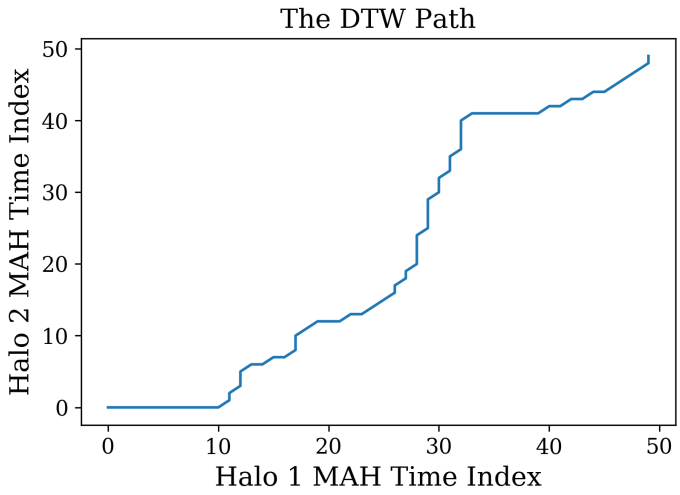

Figure 3.2: We show the DTW path for two down-sampled halos '1' and '2' corresponding to the two halos in Fig. 3.1. The path shows which timesteps from halo 2 are matched to each timestep in halo 1. The SSE distance would be shown by a diagonal one-to-one line.

imply either method is producing poor pairwise distances; however, it does suggest that each method could yield different results if used for classification purposes.

One should note the different scales on the two axes, with the SSE ranging from 0 to 100+, while the DTW distances are from 0 to roughly 12. Points along the x-axis have relatively small DTW distances compared to SSE distances. While there are no histories (other than '0') directly along the y-axis, there are quite a few points near the y-axis, with around 10 SSE. We show a one-to-one line in the figure to illustrate how each of the points has a greater SSE than DTW distance. At the lower left of the plot are points that have a relatively small distance in both, while a point at the top right would have a large distance in both. In Figure 3.4 we show the highlighted histories, all compared with history '0'. First, we focus on history '34', which is high in DTW distance (~12), but relatively low in the SSE distance (~20) compared to the other histories. This history is located in the upper left portion of Figure 3.3. We show the '0' history (black dashed) and the comparison history '34' (red) in the left-most panel of Figure 3.4. The primary difference we see is at early times (800+ on the x-axis) with the comparison history beginning its initial accretion earlier

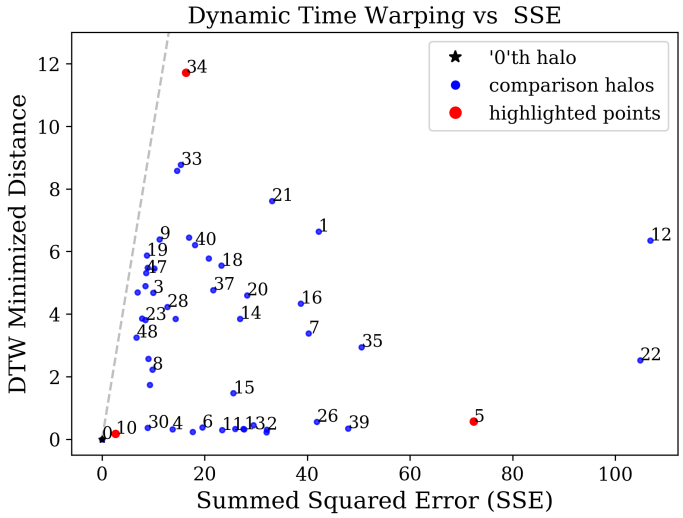

Figure 3.3: A comparison of DTW distances and Euclidean SSE distances for a sample of halos (blue) with halo index '0', the first history in our sample. By definition the '0' point is at the origin (black). Points along the x-axis have relatively small DTW distance. Points 5,10,and 34 are highlighted (red) due to their positions on the figure (lower right, lower left, and upper left respectively.) A one-to-one line is shown in light grey (dashed). Some halo labels are omitted for visual clarity.

than halo '0', starting at a rather large initial mass. At late times (0-200) the comparison history is at lower mass than halo '0'. These histories have quite different shapes, despite staying fairly close throughout most of their histories. Looking at a history from the lower left portion of Figure 3.3, we show the MAH from halo '10', in the middle panel of Figure 3.4. Halo '10' has a small DTW and SSE distance with halo '0' and matches several of its MAH features. At around 200 on the x-axis, both histories have a very similar mass accretion 'bump' followed by a relatively flat finish to their final masses. Both histories begin their trajectories at the minimum possible mass, starting flat for several snapshots, before reaching the 20 particle threshold to start tracking halo growth. Finally, we show halo '5' in the right-most panel of Figure 3.4 with a small DTW distance (below 1) and a large SSE distance (~75). Like the previous figure, MAH '5' starts at the minimum possible mass, however, it stays at the minimum threshold for well over 100 snapshots into the simulation (corresponding to 820 on the x-axis). Even once halo '5' begins to

accrete above the 20 particle threshold, it gains mass relatively slowly compared to halo '0'. Although there are some visible accretion bumps on halo '5' which may be mapped during DTW, the large SSE distance in minimized during DTW primarily by merely 'shifting' the target halo over to better match the comparison halo. These halos have quite different half mass ages and a large SSE distance at nearly every time-step, still, they have a relatively small DTW distance. These three comparison histories help to demonstrate the potential effect of using DTW when comparing two time-series vector objects. A small distance in SSE does not guarantee a relatively small DTW distance, but using both the DTW and SSE distance can give some additional information about the relative vectors. Finding halos with small SSE distances and even smaller DTW distances, like in the case of halo '10', may help to find halo 'twin' pairs/groups with a known comparison history. For small data sets DTW can be an extremely useful took for finding similar time-series vectors; on the contrary, with a large dataset, the pair-wise nature of DTW makes it difficult to utilize for classification purposes.

3.2 t-SNE and UMAP

We have previously given an introduction to both t-SNE and UMAP in Chapter 2, here, we give some additional details regarding our usage of both. As mentioned before, UMAP is an extension to t-SNE, and theoretically has some minor differences, however, the primary practical difference is the run-time; with t-SNE running some 15-20x slower on our MAH dataset. If only reducing the data once, the speed-up associated with UMAP may not be too important. We compare the dimensionality reduction between UMAP and t-SNE below.

If we compare axis to axis between t-SNE and UMAP, we do not see a strong correlation, however, this is an artifact of the nature of semi-arbitrary axes of each algorithm. If we instead compare the second axis of t-SNE with the first axis of UMAP, as in Figure 3.5. One can immediately see a strong negative correlation among the two axes, indicat-

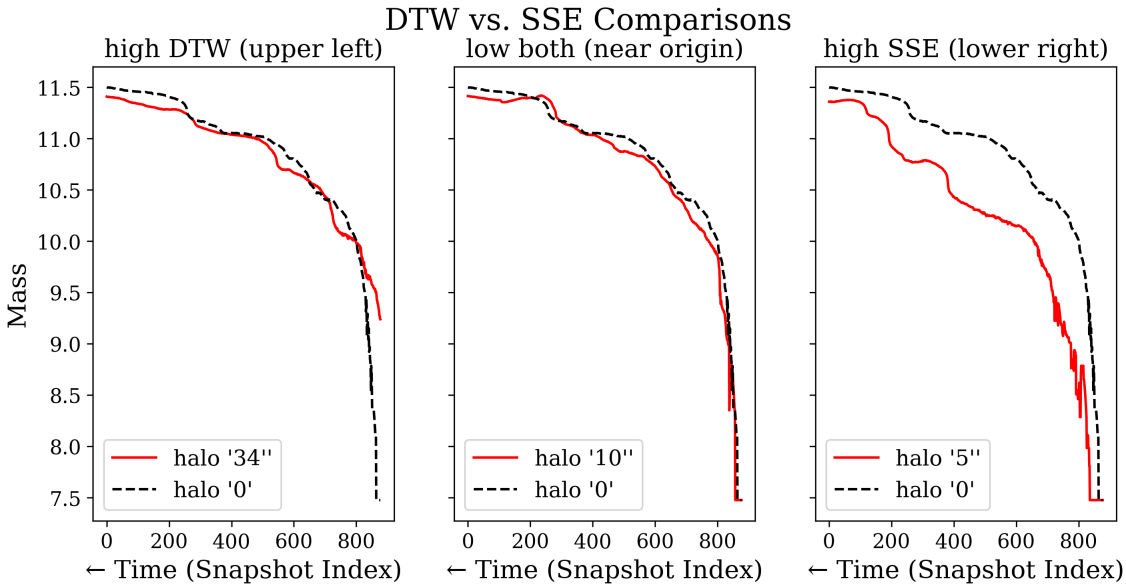

Figure 3.4: We show the three highlighted MAHs from Fig. 3.3 (in red) and halo '0' (dashed black), which was used to calculate their respective DTW and SSE distances. From left to right, we show a history '34', which had high DTW distance and relatively low SSE distance, history '10' with both low DTW and SSE distances, and history '5' with a low DTW and high SSE distance.

ing similar information being encoded into each dimension from the original MAH space. There is no guarantee that two runs of t-SNE or UMAP would have a matching dimension, however, it is an indication that both methods are picking up on similar MAH information. Interestingly, there is a small group of halos at around 8 in the UMAP dimension, which are slightly separated from the main cluster. For the purposes of classification, these halos are still close enough to the primary group that they would likely be grouped together, however, differences like these can be interesting for additional study into atypical halos. At lower UMAP values, the correlation is not as tight, but generally maintains a strong negative correlation, with a Pearson r coefficient of -0.98.

A strong correlation between two axes of t-SNE and UMAP does not guarantee a strong correlation of the other two axes. We show the alternate axes in Figure 3.6; comparing the first t-SNE axis with the second UMAP axis. In this case we still observe a fairly strong negative correlation, with a Pearson r coefficient of -0.76. We observe a main cluster which

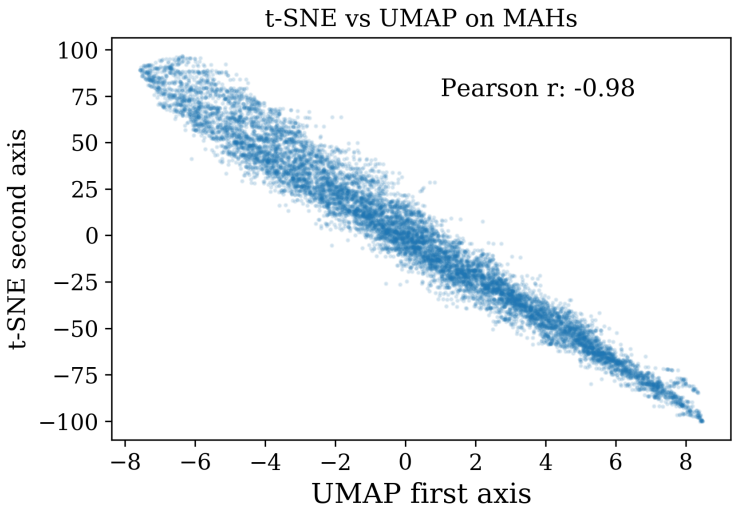

Figure 3.5: We compare the second axis of the t-SNE reduction of the MAH space with the first axis of UMAP reduction on the same data. The Pearson correlation coefficient is given in the upper right.

falls on the negative -1 correlation slope, with several objects scattered off the diagonal towards higher UMAP (and t-SNE) values. The significance of these off diagonal objects is hard to quantify due to the 2 dimensional information encoded in each method.

More importantly for our purposes are the resulting groups that are formed when using a classification algorithm input with each respective reduction. As discussed briefly in Chapter 2, the scaling or normalization of the reduction can skew the classification results. Despite the strong correlations we observe between the relative axes of t-SNE and UMAP, slight changes in scaling or orientation could result in drastically different classification results. As unsupervised learning attempts to use relative differences in the distributions of points to find group boundaries or classifications, even linear, monotonic transformations can alter the types of groups found. We show the results of running K-Means on the UMAP reduced space, in both a raw UMAP output, and a normalized (rescaled) version, in Figures 3.7 and 3.8 respectively. The spatial distribution of points are identical in both figures, showing the t-SNE MAH reduced space. Separately, we grouped the points in the UMAP space, and have shown those groups (G0, G1, G2) with the marker color.

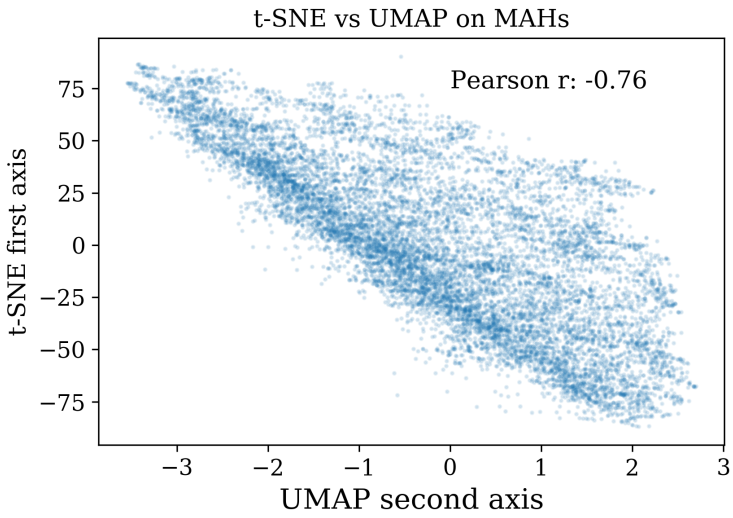

Figure 3.6: Similar to the previous figure, we compare the *first* axis of the t-SNE reduction of the MAH space with the *second* axis of UMAP reduction on the same data. The Pearson correlation coefficient is given in the upper right.

In Figure 3.7, generally the UMAP points divide the t-SNE space quite cleanly in the t-SNE axis 2 direction. There are just a handful of red points which are somewhat deep into the 'green' groups regime, likewise with a couple of blue points in the green etc. If we simply compare the groups from the t-SNE+KMeans and the UMAP+KMeans, using their raw output values, we get a 66.4% max match fraction among assignments. In Figure 3.8, the UMAP points divide the t-SNE space in a more radial direction, with the green group dividing the t-SNE space with a more diagonal division than in the previous plot. This group takes up nearly half the space, with the red and blue UMAP+KMeans group dividing up the lower half, split from the center of the overall population. If we compare the groups from the t-SNE+KMeans and the UMAP+KMeans, this time using rescaled UMAP axes, we get a 53.2% max match fraction among assignments. As discussed previously, neither of these methods is inherently better than the other and a high match fraction merely suggests similar information is being used for classification, not necessarily that it is the 'right' information. For MAHs, these different classifications suggest the importance of trying different normalizations, if possible, when using unsupervised methods, even in

reduced dimensional spaces.

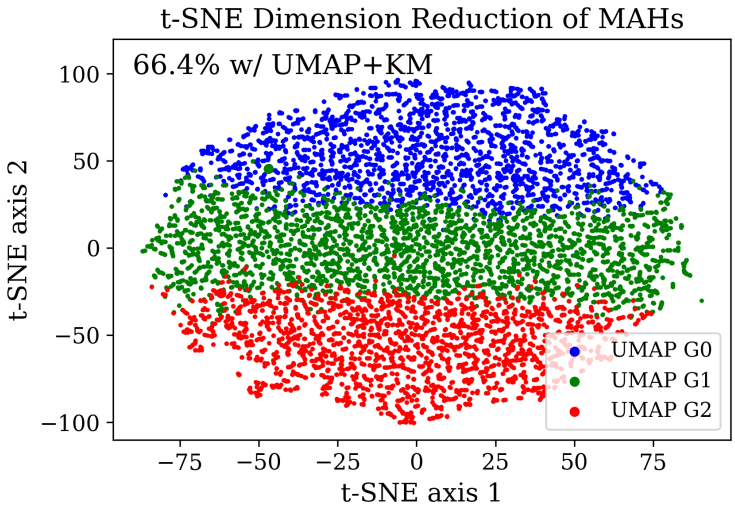

Figure 3.7: We show the MAH data, in a t-SNE reduced space (the spatial position of the points), and group assignments from UMAP+KMeans (run separately on the non-reduced MAH data), in color (G0, G1, G2). The match fraction between the t-SNE groups and the UMAP groups is shown at the top. The UMAP+KMeans groups divide up the t-SNE space primarily on the y-axis.

Previously, we discussed a mock test we made in order to test the ability of t-SNE and UMAP to reduce the dimensionality and cluster our models' outputs of MAH assignments. Rather than using the spatial reduction techniques on the MAH data itself, as we did for Model 33, we discussed how it can be used to 'group' similar grouping algorithms based on their output assignments. For example, multiple runs of an algorithm, just varying the random seed, may be much more similar than multiple runs of an algorithm varying several key hyper-parameters. If the assignments of each run of the algorithms were then combined into a dataframe and input to a data reduction technique, they should group similar runs closer together for easy visualization. To create the mock data-matrix, we started with a choice of 4 (or more) random 'base' models from our full list of models from Chapter 2. We then shuffled (or randomized) some fraction of the points for each model from 1% up to 75%, and then tested t-SNE and UMAP to gauge which better recovered our base models and their derivative groups. It should be noted that a shuffle of x% does not necessarily result in a change in that percentage of values, as it is possible for a shuffle to result in the

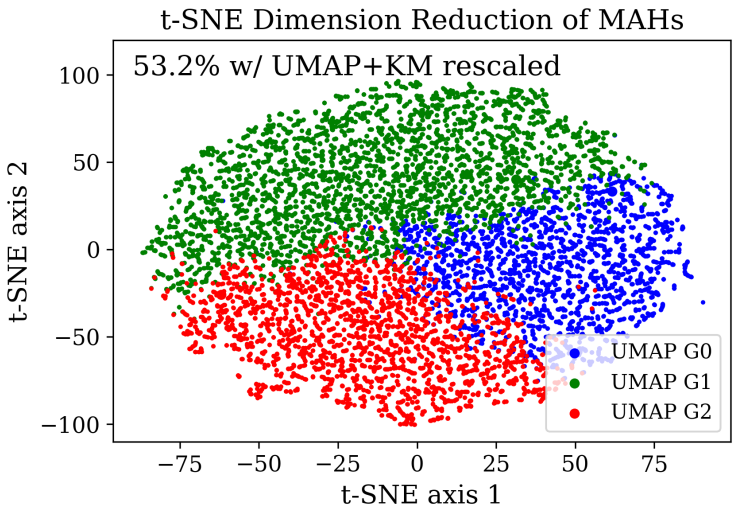

Figure 3.8: Similar to the previous figure, we show the MAH data, in a t-SNE reduced space (the spatial position of the points), and group assignments from UMAP+KMeans (run separately on the non-reduced MAH data), in color (G0, G1, G2); however, in this case, we first normalized the UMAP space before running KMeans. The match fraction between the t-SNE groups and the UMAP groups is shown at the top. The UMAP+KMeans groups divide up the t-SNE space in a more radial manner, dependent on both t-SNE axes' information.

same assignments being "shuffled" with each other. For each shuffle value, we randomized the selection process by seed, 5 times, for a total of 105 total points (5 base models + 20 shuffles each). Shuffling the points over each model vector preserves relative group sizes and ratios, while randomizing allows for the group sizes to change slightly. We did not notice a significant difference regardless of the method used to create the shuffled groups. We show one example of a mock run in Figure 3.9. The five 'base' models are shown in red, observed at the center of each central groups. We did have to play around with the settings for both t-SNE and UMAP in order to get visible separation in our points, as many of our runs had the base model over-plotted with most of its derivative groups. We experimented with several different t-SNE perplexity values to achieve the visuals of the plot, though, the 5 main groups were generally the same regardless. Lower perplexity values resulted in very tight clusters, with a few spurious points typically at the 50% shuffle level, which were not clustered close to a parent group. Higher perplexity values tended to separate the

groups better locally, as well as resulting in higher fidelity groups. We used a perplexity of 25 in our figure which perfectly reproduced out initial groups.

We noticed that t-SNE seemed to better preserve the local structure, meaning the 1% groups tended to be closer (or overplotted), while the greater shuffle fractions were farther from the base. We ultimately used UMAP for our archetype groups in Chapter 2, despite a slight initial preference for t-SNE in the mocks. Due to the flexibility of UMAP to run on much larger models, its performance when paired with KMeans clustering, where small local clustering differences are less important, and the fact that we were using UMAP for our M33 model, we decided to use UMAP, with little consequence on the results. The significance of this type of test is to gain an understanding of how much random variation can exist before a model's group assignments become indistinguishable from the other models' output. Our tests suggest that small variations in group assignments should not have a large effect on our grouping results. This result applies to both t-SNE and UMAP. We find that even mixing up to 70% of the values still allowed for accurate reproduction of our initial mock groups. Shuffling fractions larger than this resulted in distributions that poorly and unreliably reproduced the initial groups. At first it may seem like an astonishing performance from t-SNE (and UMAP) to be able to reproduce the derivative groups from such shuffled assignment vectors. To understand just how scrambled the various base shuffles are, we plot the fraction of matching assignment labels between the base models and their respective shuffled groups, as well as for their their non-group members in Figure 3.10. We begin by focusing on the 5 solid lines (red, blue, green, blue, black) at the top of the plot, which are the match fractions between the base model and each of the shuffled mock arrays. The shuffled mocks are arranged on the x-axis of the figure such that the first point is the base model (0), the next 5 (1-6) correspond to the first 1% shuffle mocks, and the last 5 (16-21) correspond to the most shuffled mocks. By definition, at the top left corner of the plot, the base model perfectly matches itself, and very nearly matches its 1% shuffles. As stated previously, shuffling 1% of the values does not necessary result in a change in 1% of

the values, as a shuffle can result in no change if shuffled with its same assignment value. We find for all 5 base models that the most shuffled mocks still match the base model at 65-75%. The non-solid lines show the match fraction of a base group and its derivative shuffles with a different base group. These lines are to evaluate whether the shuffled mocks are more similar to their parent base model or to other models.

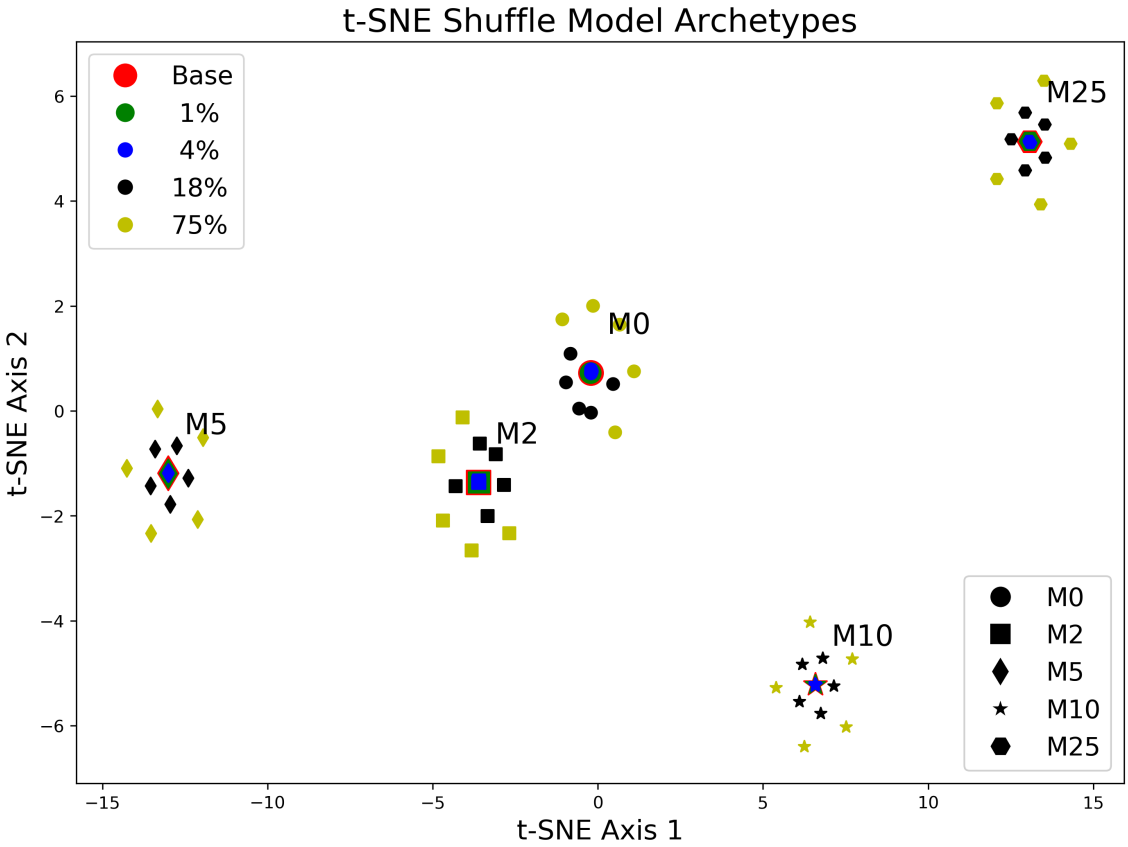

Figure 3.9: A sample plot from our mock tests to determine the ability of t-SNE to preserve similarity among random changes in group assignment. We started off with 5 models (M0, M2, M5, M10, M25; PMFHM, PHM, KMeansLog, KMeansLin, MaxML) from our full model list, and then altered some fraction of the group assignments (given in the legend). The different plot markers correspond to each of the base groups, with the Model Number annotated near each base marker.

In most cases we observe that the non-solid lines are below the solid lines, regardless of the shuffle fraction; meaning, a model and its shuffles are more similar to each other, than they are to other models without shuffling. One notable difference is with the red-dashed line, showing the match fraction between our second base model, M2, and the

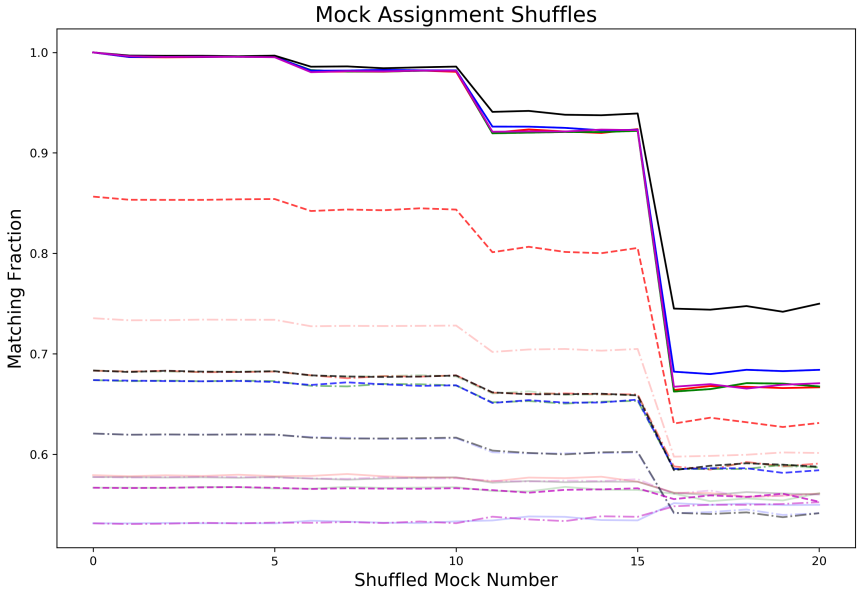

Figure 3.10: We show the fraction of assignments matching between the base group and its derivative shuffles (in solid lines). We additionally show a the fraction of assignments matching among the zeroth base model and two of the other base models (M2 & M5) and the groups and their shuffles (dashed lines). Note: the colors in this figure (red, blue, green black) do not correspond to the colors in the previous figure.

zeroth model, M0 and its shuffles. We find the low shuffle mocks from the M2 model (red-dashed from 0-15), actually agree more with the zeroth model, than the high shuffles from the M0 model (red-solid from 16-21). This indicates a fairly close similarity between base models, as expected, as both are derived from age based methods. t-SNE (and UMAP) are able to properly associate each of the high shuffle groups with their appropriate base model, despite these match fractions intersecting. The other dashed lines show the match fraction of the second base model, and the other models. Likewise, the dashed-dot lines shows the match fraction of the third base model, M5, and the other respective models. Another way of visualizing this information is in matrix form, as shown in Figure 3.11, giving the match fraction between each base model group, and each of the other model. We organize the information as follows: the first row, (y-axis, 0) shows the match fraction of the first base group and all 20 of its shuffles. In the next four rows, we show the match fraction of the first base group and each subsequent base group (and their 20 shuffles) organized into 5 segment

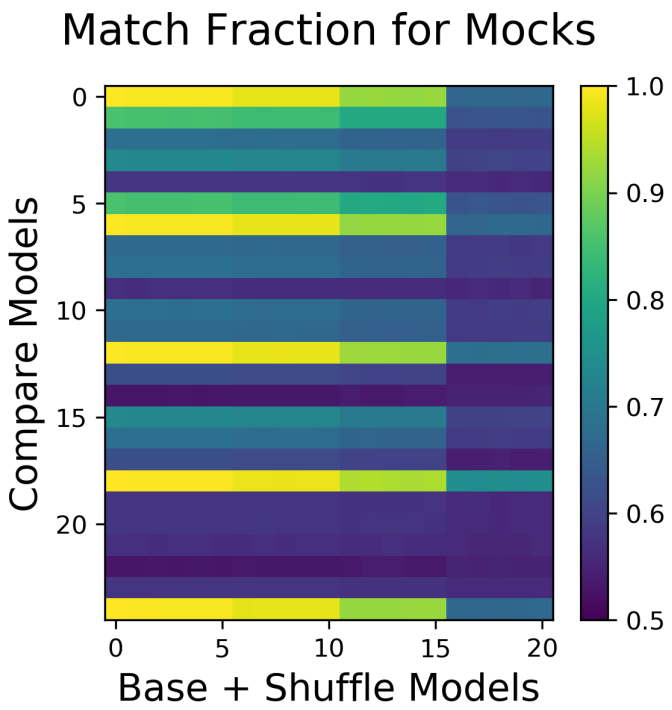

Figure 3.11: We show the fraction of assignments matching between each base group and their derivative shuffles. The first rows shows the match fraction of the first base group and its shuffles. Followed by the match fraction of the first base group with shuffles of the second base group (and each subsequent base group) organized horizontally (x-axis) into 5 shuffle chunks by increasing shuffle fraction. After the first base group has been matched with each base's shuffles. Row 5 shows the match of the 2nd base group with the first base group's shuffles, etc. For example, the bottom-right most rectangular region shows the match fraction of the last base group with it 5 most shuffled vectors.

chunks on the x-axis by increasing shuffle fraction. For example, 5 on the y-axis and 9 on the x-axis gives the match fraction between the second base group and the first base groups second-tier (4%) shuffle mock. They share a match fraction of 85% which, as seen in the last figure, is quite high for a match of an off-base mock. In general the strong yellow bands we observe in the figure correspond to each of the base groups matching with themselves. This is by definition in the zeroth column at each respective base group position (y-axis: 0, 6, 12,18,24), and near 99% by definition for each of the next 5 columns. As expected, as the shuffles get more extreme, the match fractions generally decrease. However, it is not guaranteed; in row 17 (comparing the M5 base group with the M25 shuffled mocks) switching between the 15th and 16th column, we actually see a slight increase in match

fraction, indicating the original base models were rather dissimilar. As expected, we see extremely close agreement among each of the points separated only by random seed, with only marginal differences in match fraction from mock to mock. As we saw before, looking at the match fractions in Chapter 2, we see that the derivative models of the M25 base match poorly with all the other base models at around the 50-60% level. The M25 model generally only matched highly with other mass loss models in the previous chapter and here we are only comparing with age and KMeans.

The results of this previous section led us to feel confident in the ability of either t-SNE or UMAP to create archetypal groups from our full list of models and their assignments. The mock test was not a perfect reproduction of the task at hand, however, it used a comparable number of models and a similar number of base groups as we might expect. We also performed the same test using slightly different number of base groups with similar results. If our models are more mixed in terms of match fraction, it is possible that the algorithm could have difficulty grouping similar models, despite performing perfectly in this mock; however, it appears that t-SNE and UMAP are both powerful algorithms for unsupervised clustering of similar models based their outputs.

3.3 Boosting Factor

Another way we thought to assess the meaningfulness of our various model groups, whether from our initial models, archetypal models, or our later linear models, was by observing the over-representation of additional halo properties in the various halo group population. In Chapter 2, we showed the normalized distributions of various halo properties by group. In order to reduce the vast amount of information contained in the group halo properties, we use the concept we call a 'Boosting Factor', to reduce each distribution into one number: the over or under-representation fraction relative to a random distribution.

We define the Boosting Factor relative to some percentile of extreme values for each property. For example, we may look at the distribution of the top 30 percent values for

each halo property in each of our groups. If we assume that one group had no halos which had values in the top 30% values for whatever property we are looking at, it would have a zero value for the Boosting Factor. If instead we assume that a different group had all the top 30% values in its group, we would then calculate the expected number by random chance (1/3 for 3 equally sized groups), and the Boosting Factor would be the actual number over the 'expected' number. A Boosting Factor of 1 means that the groups population is consistent with a random distribution in terms of that property.

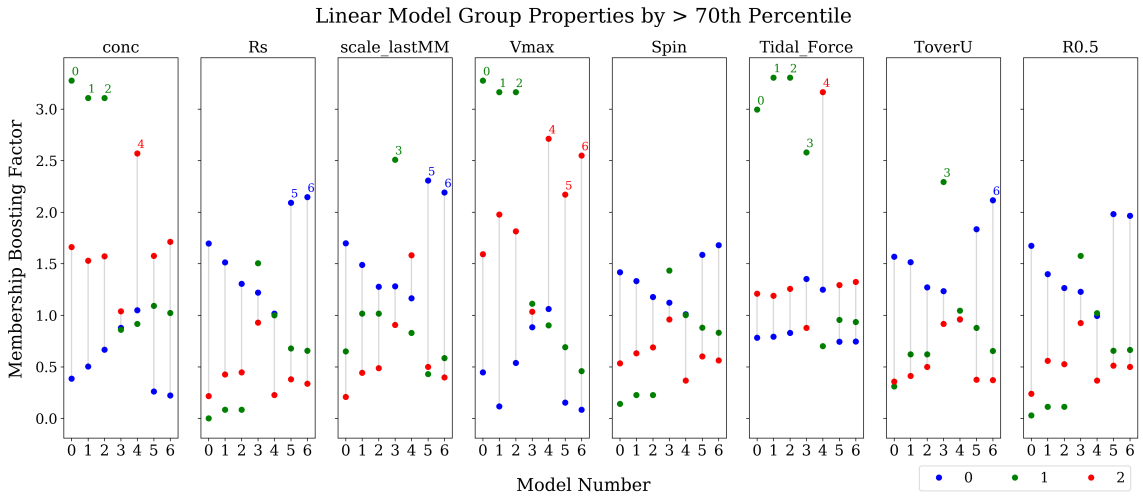

Figure 3.12: The 70th percentile Boosting Factor for each group (0,1,2) for each of our linear models. A Boosting Factor of 1 is consistent with a random distribution for a given property, while larger values indicate over-representation. High values are annotated with the Linear Model number.

We began by investigating the Boosting Factor of each of our Archetype Model groups, for 3 different percentiles. Qualitatively, we observed the same trends regardless of whether we look at the top 50% of objects for computing the Boosting Factor, or if we use 60 or 70%. Selecting a percentile value is a balance between having too few extreme objects versus reducing the signal of 'extreme' objects by including many average objects if using the 50th percentile. The Boosting Factor gives a much easier way of extracting information about which properties may be dominant in a group relative to the amount expected at random. We start by showing the 70th percentile and up Boosting Factor for our Linear Models at the end of Chapter 2. We see the most 'boosted' group for concentration, the

first panel, is actually our Age group 1, closely followed by our KMeans group 1 (and matching BGMM group 1). We see a similar trend in Boosting Factor in the V_{max} panel for our first 3 linear models. Both of these properties are boosted around 300% relative to random, meaning they have roughly 3 times as many highly concentrated, high V_{max} objects than would be expected by chance alone. Likewise, these groups have high tidal force Boosting Factors, however, in this case, the age group is slightly lower than the KMeans (and BGMM) group. Also, we find the group 2, Linear Model 4 (Peak-Final) has a near 300% tidal force boost. Other properties we see an inversion in the boosting factors by groups. For 'Rs' the scale radius, we see the groups which were highly boosted in concentration, V_{max}, and Tidal Force, are very under-boosted here. This is likely a result of the inverse relationship between Rs and concentration. For these properties, which are likely inversely related, we could instead flip the sign for which objects we consider, so instead of considering objects in the top 70th percentile, we could consider objects in the bottom 30. Not all properties have a simple relationship however, and in terms of the scale of last major merger we actually find the groups which were extreme in concentration, V_{max}, and Tidal Force, are actually centrally located around a Boosting Factor of 1. The group 0 and group 2 objects in scale of last major merger more closely reflect those from Rs, showing how sub-populations can exhibit different property correlations than the overall population does. We find the hybrid mass loss model (LM3) is actually the best at boosting extreme scale-of-last-major-merger objects. As this is the scale of last major merger, high values indicate a late time merger, meaning the hybrid mass loss model group 1 is the best of our linear models at finding later merging objects. Finally, we see that our high spatial clustering groups, tend to be under-boosted in terms of spin, and fairly low values in terms of T/U.

We also include the 50th percentile boosting factor for each of our initial models. We show the 50th percentile rather than the 70th percentile simply because it compresses some of the more extreme boosted values, while keeping the same qualitative trends, making

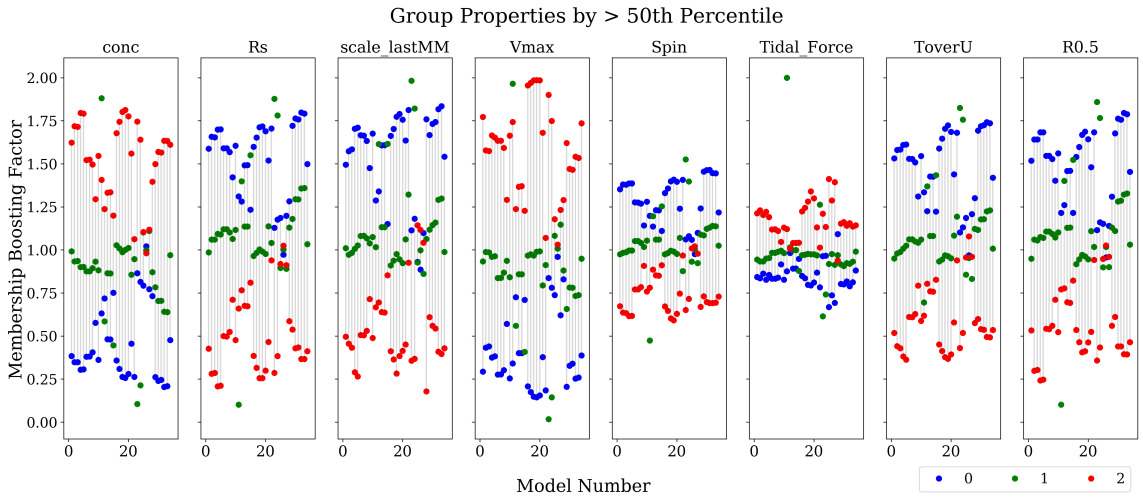

Figure 3.13: For completeness, we show the 50th percentile boosting factor for 8 halo properties, for all of our initial models, similar to the previous figure. Each color represents one of the three group labels, shown in the legend.

the plot more legible. Due to the somewhat natural ordering of certain parts of our full model list, by model type, some patterns arise in the plot. Although it is difficult to see individual models in this figure, it is easy to see how the boosting factor can be used to pick out informative model types. Notably, one can see how strongly boosted in V_{max} the BOW models' (M15-M19) G2 groups are. They are all comparable to the V_{max} we observe from our KMeansLin (M10) G1 group. From the tidal force panel, it is clear just how boosted the KMeansLin G1 group is compared to all the other models. In a property like T/U, however, the G1 group is fairly moderate compared to most of the other highly boosted groups.

The Boosting Factor may be one way of assessing the ability of our MAH groups to split on useful halo properties. We can further refine the current boosting factor definition to be scaled between 0 and 1, measuring the deviation from random chance, whether positively or negatively so. This modification might allow for a more automated way of reducing how successful a certain group is at segregating extreme halo property objects down to one number. We do not include this modification in this paper, though it is fairly trivial to incorporate this change to the model.

3.4 Supervised Learning

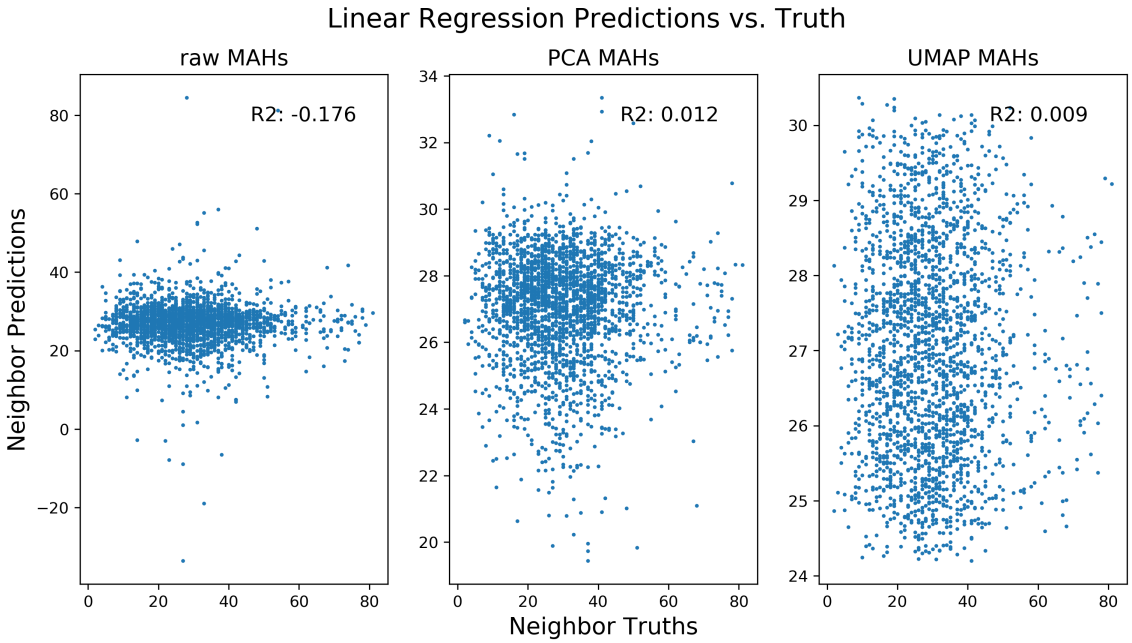

Figure 3.14: We show the test set performance of a a linear regression algorithm learning the relationship between a halos MAH and its count of nearby (closer than 10Mpc) halo neighbors of similar mass. The left panel shows the performance when input with the raw MAHs, the middle panel with 95% PCA input, and the final panel with UMAP input. The R_2 score is shown in each panel.

We made a brief mention of supervised learning in our reasons for pursuing unsupervised learning for the specific problem of finding potential halo groups from MAHs with high spatial clustering. The problem for supervised learning, when using the TPCF for spatial clustering, is that we can not compute the spatial clustering of a halo a priori, as the computation is dependent on the other group objects. If we want to use supervised learning for the problem a one-to-one learning algorithm will not work. One could construct a MCMC-style algorithm that chooses points iteratively based on some acceptance or rejection criteria, or some genetic style algorithm which would turn on or off certain halos without having to exhaustively test each pair. An entirely different approach than using the TPCF data itself, is to use a potential proxy for clustering. Although our TPCF calculation will only include in group pairwise distances, we can estimate a halos TPCF spatial clustering by its number of halos in its near environment. Using this count gives one

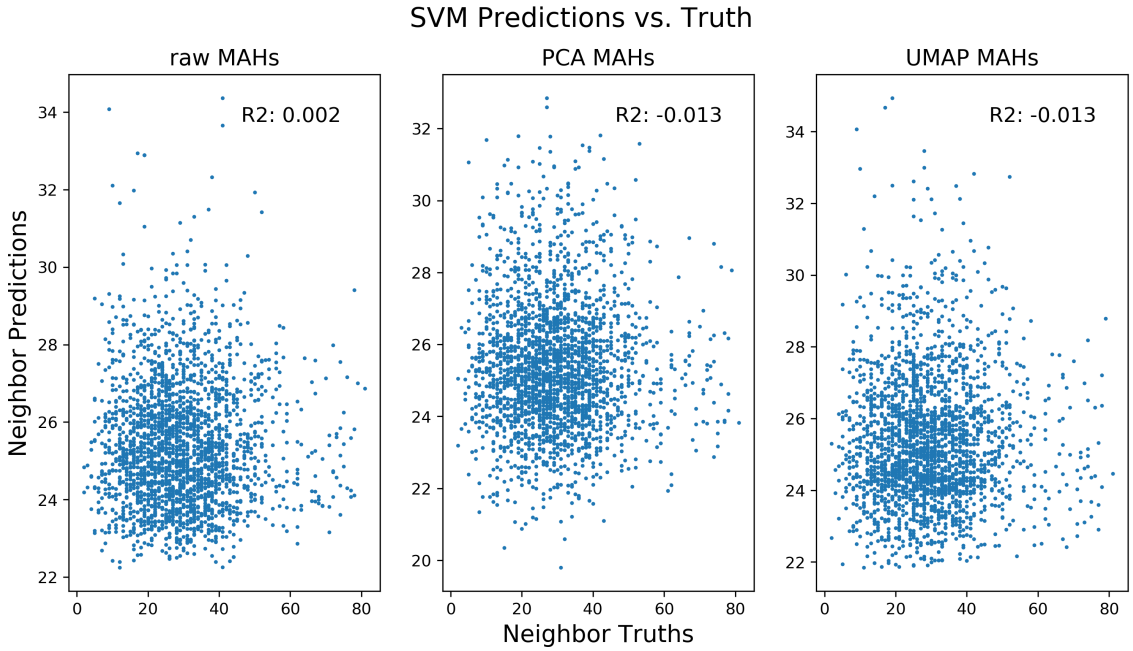

Figure 3.15: We show the test set performance of a SVM learning the relationship between a halos MAH and its count of nearby (closer than 10Mpc) halo neighbors of similar mass. The left panel shows the performance when input with the raw MAHs, the middle panel with 95% PCA input, and the final panel with UMAP input. The R_2 score is shown in each panel.

static number per halo (or multiple if we use different radial distance bins), which allows for supervised learning to take place. This supervised learning would optimize on a slightly different problem than we actually want, but if our environmental proxy is close enough to the real TPCF calculations, it may not matter too much. Using a proxy for clustering opens an entirely new can of worms, and so we will only briefly explore the concept here, without attempting to exhaustively search through the possible options. Just as with unsupervised learning, there are tons of tunable hyperparameters associated with each algorithm. Unless otherwise noted we will stick to the default parameters associated with each supervised algorithm. The general idea will be to count the halo neighbors (of similar mass), within 10Mpc at z=0. This list is the truth set for each algorithm as a proxy for the actual spatial clustering contribution we might expect from this halo being included in a group. The input set will be derived from the MAHs, to try to find a regression mapping from the MAHs to

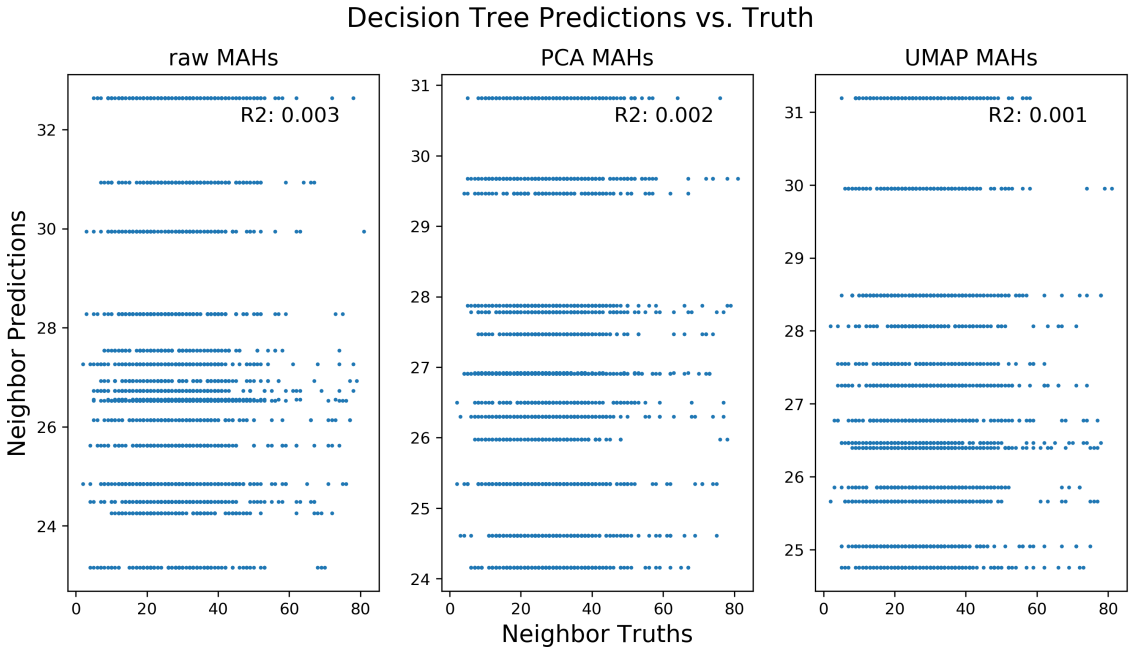

Figure 3.16: We show the test set performance of a a Decision Tree algorithm learning the relationship between a halos MAH and its count of nearby (closer than 10Mpc) halo neighbors of similar mass. The left panel shows the performance when input with the raw MAHs, the middle panel with 95% PCA input, and the final panel with UMAP input. The R_2 score is shown in each panel.

the local neighbor counts. We can use the raw MAHs, PCA reduced MAHs, or even the UMAP reduced MAH space. We evaluate the performance of the regression by testing the correlation of the true neighbor counts with the predicted values from each algorithm. We begin by evaluating the ability of a Linear Regression to find this mapping from the raw MAHs. A multivariable linear regression (MLP) or multiple regression is an extension of the ordinary least squares for a single variate problem. As each halo history has on order of a thousand time-steps, this regression is fitting over many correlated parameters. By first performing PCA on the MAHs, and preserving 95% of the data variance, we can eliminate the vast majority of the data dimensionality, simplifying the MLP problem. Finally, we can use the 2-dimensional UMAP space as input to the regression problem. After creating a training set composed of 80% of our sample and the remaining 20% as a test set, we use *Scipy's LinearRegression* function to fit three models to the local neighbor counts. We use

the default parameters for each algorithm except where noted otherwise. We show all three of these in Figure 3.14 using the Linear regression model. In the left panel we show a linear regression performed on the 978 dimensional raw MAHs, followed by the PCA MAHs containing just 11 dimensions, and the UMAP MAHs with just 2. We would hope for at the very least a strong correlation among the Truth values on the x-axis, and the Predictions on the y-axis. Ideally, we would have a good fit as indicated by a R^2 near 1. Unfortunately, our first panel is actually slightly negative, at -0.176 indicating a worse than random performance on our test sets. The PCA model and UMAP model are roughly consistent with random in terms of their predictive ability. It should be noted that each of the three methods produce very different ranges of data. While the Truth values range from roughly 0 to 80, we find that the predictions from the raw MAHs range from -30 (an impossible value) to up to 80 for a single point, while most remain around 15 to 40. The PCA model had a range from 19 to 34, and the UMAP model from 24 to 31. While all three models produce different results, none of them have predictive power for this task.

We next try SVMs (support vector machines), to attempt the same regression problem in Figure 3.15. Similar to the previous figure, except with more consistent performance regardless of the input type: raw, PCA, or UMAP. Unfortunately, consistent, does not mean good. Each of the SVM models performed consistently bad, performing near random with scores ranging from -0.013 to 0.002. These scores are using the default parameters. We additionally used a linear kernel which resulted in all negative scores. Next, we attempt to use Decision Trees with similar failure in Figure 3.16. For the tree, we set the parameters to require at minimum 250 samples per leaf (and per split), with a shallow max depth of 5. Each split represents a decision boundary. These results are much more discretized, due to the nature of finite splits in decision trees, with little promise. The prediction values for each panel are fairly consistent regardless of the input type, ranging from about 23 to 33. A test with random forests produced results much more similar to SVMs, with more blob like regions, each close to random in terms of their predictive score. In conclusion, it does

not appear that the method of using local neighbor count in combination with supervised learning to find relevant features in MAHs is viable under the given circumstances.

3.4.1 A random walk approach

An entirely different way of approaching the same problem of finding which aspects of MAHs lead to strong TPCF spatial clustering, is to take a random walk in some MAH parameter space, using acceptance or rejection based on the observed clustering to move towards more favorable MAH groups. To reduce the dimensionality of our random walk, we begin with the 2-dimensional UMAP reduced MAH space rather than using the full length MAH vector. Then we randomly choose 30 points in the UMAP space and choose 30 random variables representing the radius of interest around each point. Using `KDTree`, we are able to easily recover the MAHs that are enclosed within each circular region in the UMAP plane. We then calculate the TPCF (in a singular radial bin from 3-10Mpc) for each population enclosed and probabilistically select one of the initial points from the 30, based on the highest spatial clustering value in the (X,Y,Z) simulation space. Similarly, we could simply select the point with the highest overall clustering from the initial random points with little change in effect. Once we have selected an initial point to start, we then randomly assign a derivative vector for each random variable (2 for position, and 1 for radius), and update the original point plus the derivative vector times some learning rate function. For the 2 position coordinates, they can vary over all of real space, positive or negative. While we also technically allow the radius to venture into negative values, given that it uses the same addition of a random derivative value adjusted by the learning rate, we check the feasibility of our values before accepting them. To establish feasibility, we require there to be at least 50 points enclosed, and fewer than 90% of the total points enclosed. These checks are to prevent too few halos from being included that we would expect near 0 counts for the TPCF and to prevent all of the sample from being included. In the case that we are on the lower limit threshold, we iteratively decrease the 2 position scalars (x,y) by some

scaling factor (we used 1%), effectively moving the group center closer to the origin, and similarly increase the radius scalar (r) by the same factor until the conditions are met. In effect this prevents the radius from going negative as a 'negative radius' would include 0 points. If we are instead at the upper limit threshold for points, we reduce the radius by 2% each time until there are fewer than the max threshold of points enclosed; we do not modify the position vectors. The learning rate function, $f(R_L, ...)$ changes how much the random derivative guess (a value between -0.5 and 0.5) modifies the original vector. The learning rate starts with some scalar multiplier R_L (we use 300%) and decreases linearly with each iteration in the chain until 90% of the initial value has been met after M iterations. If the TPCF of the group enclosed by the guess values is higher than the current vector's, we update the chain by *half* of the guessed value, $D \times f(R_L, M)/2$. If instead the TPCF value of the guessed group is less than our current vector's TPCF value, then we reject the point 99% of the time, only accepting it rarely. The chain ends after M iterations; however, it can be restarted again after termination if the chain didn't appear to converge, though this will reset the learning rate back to the original.

To summarize the method:

•Goal: Find the (circular) region in the UMAP space, whose enclosed points maximize the value of the TPCF (in the simulation space.)

•Random Variables: [x, y, r]

•Parameters: R_L=3, f_{Accept}=0.01, M=500 (iterations)

1. Create 30 initial sets of random variables: $[x, y, r]_i$, and probabilistically select the 'best' initial vector, V, based on the highest TPCF result of the enclosed group.

2. Randomly choose derivative values, D, to get a new guess for each random variable, $D \times f(R_L, M) = [\Delta x, \Delta y, \Delta r]$, to create guess $[x', y', r'] = V + [\Delta x, \Delta y, \Delta r]$

3. Let $\Omega = \{p \in UMAP(MAHs) \mid cond(x = x', y = y', r = r')\}$.

   if $|\Omega| \notin (50,\ N_{Total} \times 0.9)$ *then*, iteratively modify x,y, or r as appropriate.

4. **if** TPCF($\Omega(x',y',r')$) > TPCF($\Omega(x,y,z)$): ACCEPT GUESS.

   elif randomVariable $\in$ [0,1] > 0.99: ACCEPT GUESS.

   else REJECT GUESS. continue chain.

5. **if** (Guess Accepted): update $V \leftarrow V + (D \times f(R_{\rm L}))/2$.

6. Go back to Step 2 for each iteration < M. (End after M iterations OR Continue running if chain is not yet stable)

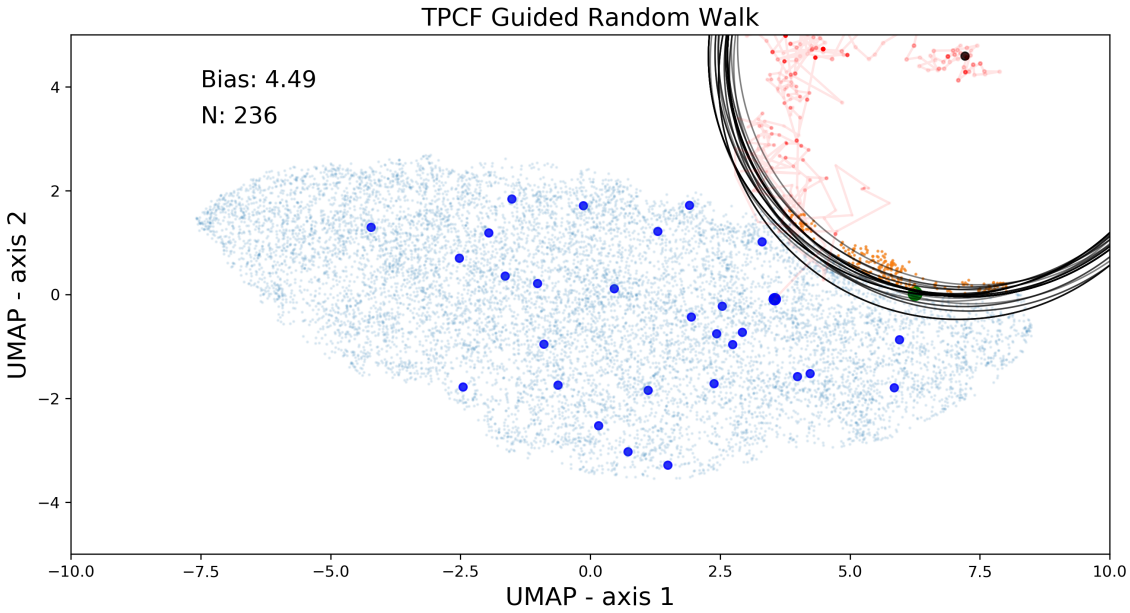

Figure 3.17: We present one random walk chain in x,y, and r, in the UMAP space, guided by step acceptance or rejection based on the relative value of the TPCF (in simulation space) of the points enclosed. The 30 initial guess locations are given by the large blue dots, while the highest TPCF guess is highlighted in green. The actual initial guess that was selected, is shown by a larger blue dot and is where the chain, drawn in red, starts. The guess values of the last few steps are shown by the large black circles, with the points enclosed by the last step of the chain, highlighted in orange. The bias, (with respect to the total population shown) is labeled in the panel, as well as the number of points enclosed.

We find that this method performs surprisingly well given the rather crudeness at each step. While we initially expected to find a mostly circular region of points in the UMAP space that maximized the TPCF, we actually found the algorithm allowed for a different solution. By expanding the x and y points far beyond any of the actual UMAP data values,

and increasing r proportionally, the algorithm was able to walk into a space that roughly linearly divided the UMAP space. We show the results of one chain in Figure 3.17, and highlight the last several steps with the open black circles. Despite all the initial x, y points being interior to the boundary of the UMAP space, the chain center quickly leaves the UMAP points in order to select the points at the upper right of the plot. Once the guess center leaves the UMAP space, it then slowly starts to migrate to the right towards higher UMAP axis-1 values. This behavior was observed in many of the chains we ran, regardless of the specifics of the number of initial points, learning rate, or acceptance fraction. The exact location and size of the circles vary from run to run, possibly indicating the need for a longer chain, or better parameter tuning; however, the same general region is selected time and time again. In the last dozen iterations we observe a general wandering around the same region with fairly consistent radii, at bias ratio of 4.49. It should be noted that between steps the bias value is not consistent and the number of points enclosed jumps around wildly, possibly indicating a need for smaller parameter jumps between steps at later iterations. This bias value is larger than we observe with similar group size in age, but is lower than seen using Peak-Final mass loss of similar group size.

Based on the general success of this chain, we separately adapted the algorithm to be able to handle multiple circles of different sizes. This flexibility would allow for more 'ellipsoidal' regions to be selected, by overlapping circles with one another, or could result in multiple separate regions of the UMAP space being combined into a single group. Typically when using multiple circles we found that either the circles would end up entirely overlapped, or one would end up jetting off to a random 'corner' including the bare minimum points required. While further experimentation with multiple circles or ellipsoidal regions may be interesting for future work, we found that 1 region was enough to recover a biased signal, at a TPCF ratio of 4.49, with 236 points.

We wanted to look at what the random walk selected MAHs looked like when compared to other histories in the original mass vs time space. To evaluate this we show the histories

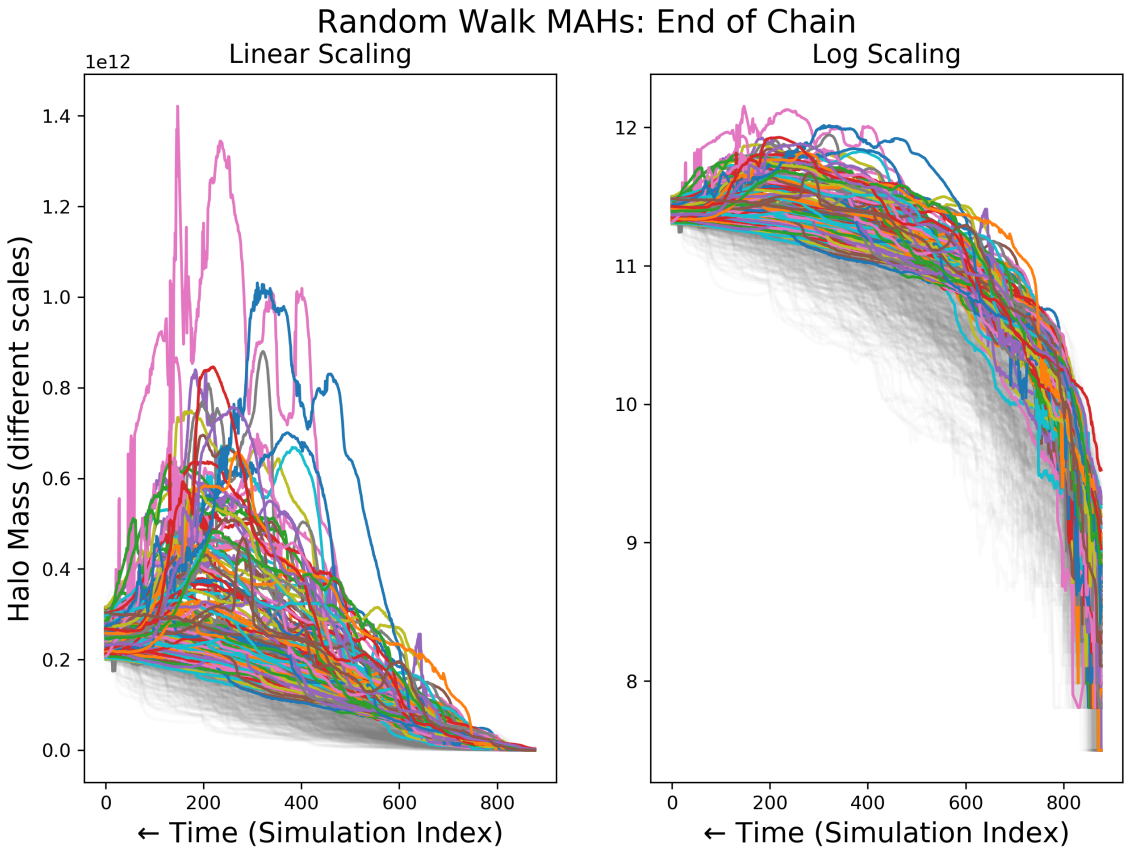

Figure 3.18: We show the results of the final step from the random walk chain from Figure 3.17. Members of the group have their MAHs shown in various colors, while a sample of the histories which were not selected are shown in light grey. Both panels contain the same histories, in either linear or log scaling for the mass axis.

contained by the final step of the random walk in Figure 3.18. From the linear scaling figure it is easy to see the selected points include all of the very high mass valued histories, however, it is less obvious why the more moderate histories were selected. In the log scaling panel, there is a slight indication of the 'knee' or kink in the selected histories around the Time Index of 600, similar to what we observed with Age based models. This knee certainly does not exist for all histories, but the lack of histories in the lower middle of the histories is stark. At very late times and at very early times there appears to be little separation in the histories. In fact, we compute the average separation in Figure 3.19, by subtracting the average history computed from the Selected points, with the average history of the non-selected points, divided by the standard deviation at each time step for both sets.

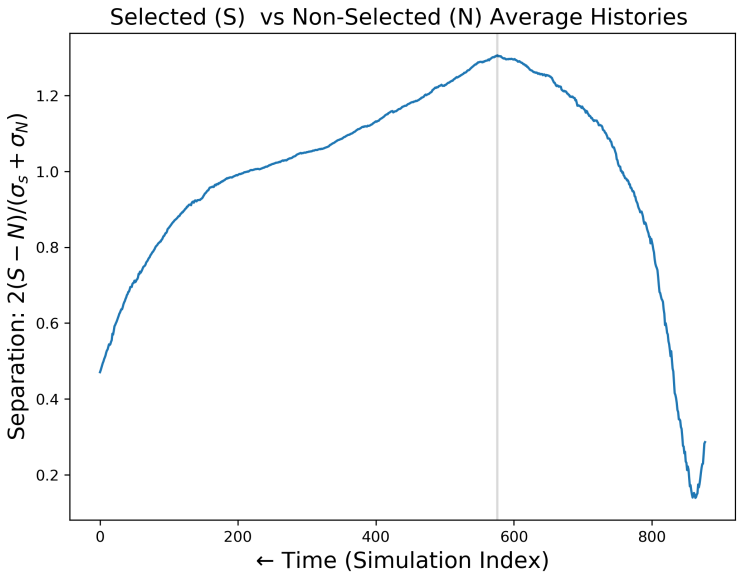

Figure 3.19: We highlight the separation in the random walk selected histories from Figure 3.18 (colored MAHs) & Non-selected histories (grey MAHs). We compute the separation of the average histories, divided by the average standard deviation at each time. We additionally highlight the max separation time with the vertical light grey line.

The separation is always positive, meaning that the selected histories are on average higher than the non-selected histories at all times. As previously stated, this plot confirms that the max separation is at middle times, around the 550 time index, while the minimum separation is at early times. Even at late times, near 0, there is still a positive separation around 0.5 sigma away.

In fact, if we compare the selected histories with various model groups we have used, as in Figure 3.20, we see a very interesting result. In the UMAP space, the difference between the KMeans-Log (2nd row) and FHM age (1st row) becomes far more stark than in the raw MAH space. For most models shown here the blue G0 groups tend to be at low values in the UMAP axis-1 dimension, however, this does not hold for all models. In some cases, like with the Average Mass-Loss model (5th row) and FHM to a lesser extent, the UMAP space is not a good separator of the groups. A model like KMeans-Linear (3rd panel) actually has a fairly clean division in the UMAP space between its G1 group and others. There is a primary cluster of G1 points at the far corner of the UMAP distribution, as well as a

few green points to the left of the primary cluster, very similar to the points selected from our Random Walk algorithm. The G0 and G2 group are fairly well separated with some overlap at the boundary. The BOW5 (4th row) has very good separation for most points in each group, with some significant scatter for various red G2 points, encroaching into the G1 and even the G0 regions when plotted in the UMAP space. Finally, we see the clean divisions among the UMAP panel (last row), as KMeans is able to divide the space into 3 Voronoi cells with straight walled boundaries.

Other models, not shown, that were notable are two of the dendrogram models based on the complete linkage, and the average linkage, which had its G1 points distributed on both sides of the UMAP axis-1, made up primarily of the same histories as in the KMeans-Linear G1 group & some additional points at low UMAP axis-1 values (indicating a potential need for a fourth group division for this model). The complete linkage model in contrast had a large G2 population at high UMAP axis-1 values & a smaller G2 population at lower UMAP axis-1 values. It is completely possible to have a population from a different model naturally split over the UMAP axis, though this is likely a result of how we avoided singleton groups in our dendrogram models. We believe that dendrograms could be used successfully for novel MAH classification, but limitations on the number of groups makes the analysis more complicated than it would otherwise be.

3.5 Halo Environment

The final test we did, was to look at how our MAHs correlated with halo environment. To test this, we first had to classify each region of the simulation into 4 categories, Clusters, Walls, Filaments, and Voids. Although this method does not result in 3 groups like all our other methods, the 4 class environment model makes for a simpler classification scheme. We choose to do this using a fairly crude Hessian method, which evaluates the number of positive dimensions of the Hessian field over a grid to determine which of the four categories each voxel is in. This method yields reasonable looking results across time given

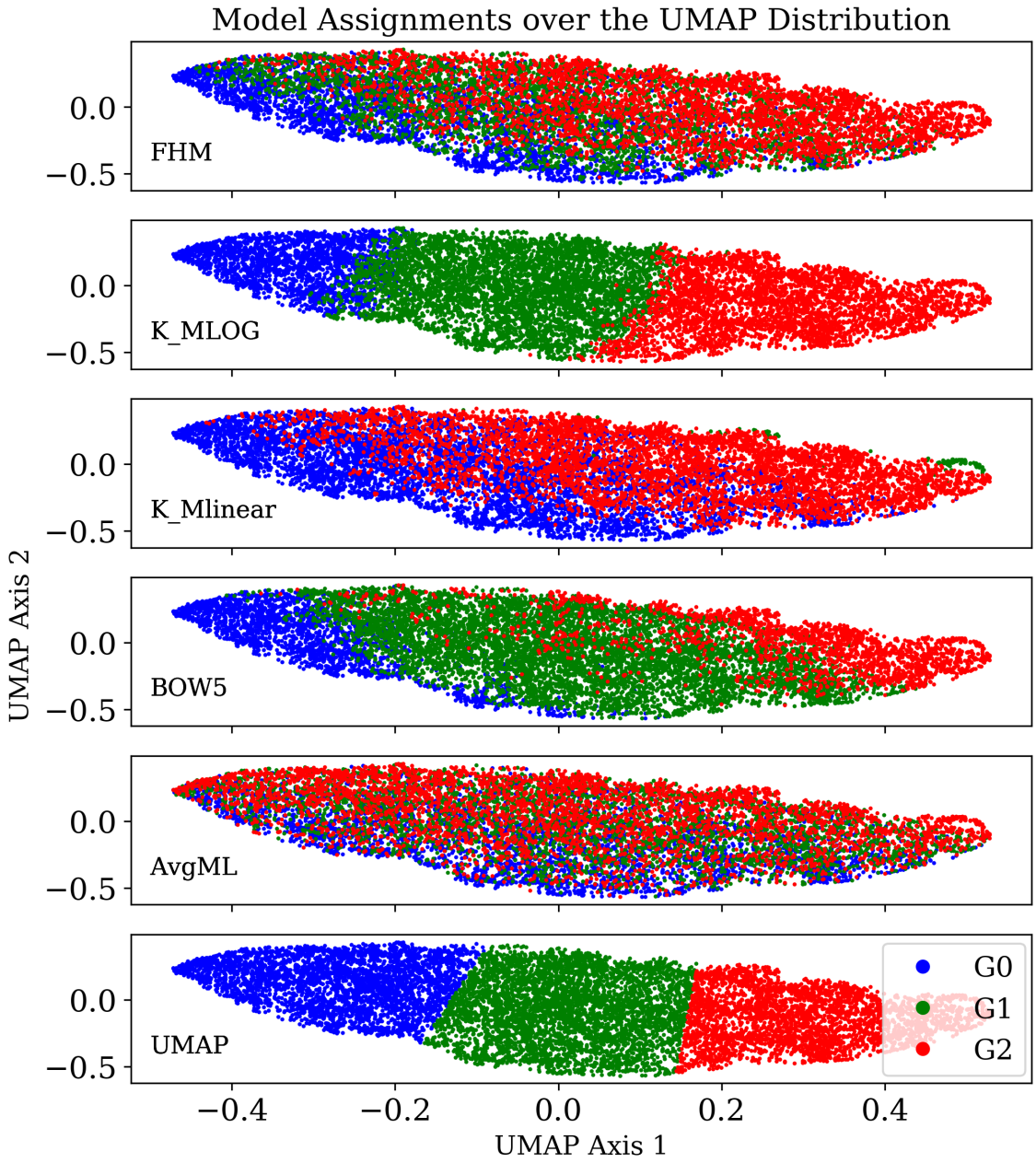

Figure 3.20: Various models' groups (colors) are shown relative to the UMAP(MAHs) dimensionality reduction. We show the Linear KMeans groups in the 3rd row, with its G1 points in the upper right region. The final row shows how KMeans input with the UMAP space cleanly divided up that space.

its simplicity, but it should be noted that these features are relative to this sample only and may not fully agree with other halo environment classifications using the full matter density. We found very few clusters as expected, followed by walls and voids. We found the most filaments, making up nearly half of the voxels we evaluated. Once we had an environmental classification for all voxels in our simulation box, we simply classified each halo according to where it was located in the box. This allowed us to look at the proportion of voxels classified by each method and likewise the proportion of halos classified as each. Despite our assumption that the clusters would have far more halos represented than the voxels would suggest, and similarly fewer halos in the voids, we actually found a roughly equal ratio between voxels and halo classifications, with a slight over-representation of halos in the 'voids'. Given our set of halo classifications we decided to plot the histories of each, and unfortunately did not see any promising results. The 4 classes were intermixed, though with very few cluster halos visible. We found very little correlation with our previous models based on the MAH data, with most Pearson r scores below 0.01 (absolute). While halo environmental classification may be an interesting problem to look at in the context of how it may influence MAHs, it may require a different approach, likely one that looks at a greater sample of histories to get more reliable environmental classifications.

Chapter 4

CONCLUSIONS

Despite being studied for more than 50 years, machine learning is still in its infancy. We have impressive algorithms which are able to work efficiently and understandably and others which are powerful and more obscure. Currently, there is no one-size-fits-all algorithm that will apply to every data set. In order to understand the data output at all, several limitations had to be placed on the data. We only looked in a narrow mass range for each run, while a more capable algorithm would be able to understand how generalizable these results are to other mass regimes. The biggest complication in using mass accretion histories are the extreme variations over the course of a history. Using unsupervised learning on data that spans several orders of magnitude, without knowing precisely how those orders of magnitude effect the importance of the output requires a new way of thinking about the data. There are semi-supervised approaches that could make better informed choices regarding which models are most descriptive, but there is no approach that neatly boils down the information into cleanly well defined categories. We decided to try as many reasonable algorithms that we could, in order to both maximize our chances at finding a meaningful grouping, and to gain a better understanding of how alternative methods might change our clustering results. We found Age based models work surprisingly well given how simple they can be, defining an entire halo history down to one number, based on when you first cross some mass threshold. Most models we tested were not actually able to compete with age in terms of the bias of the resulting groups; however, groups from KMeans, BGMM, and Peak-Final Mass Loss were able to actually outperform the best Age models we tested. Unsupervised learning models were able to group similar halos extremely cleanly, but that did not guarantee simple comparisons between the models. We found Archetypes to be a novel concept to reduce a huge space of potential models, down to a more interpretable

set; however, creating Archetypes also may have had the effect of averaging out some of the potential features were were hoping to find. We found our Archetype 2, including our KMeansLin model, GMM and mass-loss Models, were **more biased** than the equivalent age based group. We ultimately looked at both Archetypes and the original models when making comparisons about histories. We found that the choices of pre-processing can have a huge effect on the resulting groups, especially the mass scaling used for KMeans and whether or not we normalize the dimensions of t-SNE or UMAP. We found that Unsupervised learning was able to reliably give well separated groups of halos without much hands-on parameter tuning. While we ended up limiting the vast majority of our models to 3 groups, unsupervised learning has much more to offer when able to more finely divide up the input space. We ended up using an iterative process, starting with a large list of different models before refining the list and using a more directed set of models which we called our 'Linear Models' due to their focus on linear vs log mass scaling. Once we found a population of halos more biased than the most biased age groups (of similar size), we looked at other halo properties and how they might be correlated with the MAH classifications. Finally, we explored alternative approaches, including a supervised-learning approach using truth values based on a halo's local neighbor density in the simulation box, a random walk approach testing the TPCF of each group in the chain, and an approach using the cosmic web classification as a basis for groups. We found that supervised methods were unable to reliably correlate the local neighbor density with the MAH, and thus we did not pursue the resulting TPCFs of potential groups. The random walk approach worked surprisingly well, and actually selected most of the halos from our KMeansLin high bias group, despite them being tucked away in a far corner of the UMAP space. Extensions to the random walk approach did not seem to increase its ability to find high biased groups, and more work would be needed to reliably place errors on the resulting bias values. More detailed work would be needed before making conclusions about the cosmic web classification and its relationship to halo properties including the MAH. Ultimately, machine learning is a pow-

erful tool when used in conjunction with large datasets that humans can not fully explore; however, it would be a disservice to the data to assume that one algorithm will produce "the" results. Each algorithm is tuned to pick out on certain features and whether those features are relevant to your problem will depend on the specifics of the problem. Often times we do not or can not know the specifics of every aspect which opens the door for more exploratory analysis. Using a suite of algorithms in conjunction can allow for a vast net to be cast, allowing researchers to focus only on the most promising results. While halo age is still an amazing predictor for many halo properties, and is strongly correlated with halo bias, using additional classifications may help to refine which aspects of age or halo growth are most relevant to halo evolution. There is still much work to be done in order to merge the flexibility of machine learning, with interpretability of the results. With every re-normalization or layer of abstraction it is possible to lose some of the meaning behind the results. On the contrary, every layer of simplicity, for fear of delving deeper into the problem, risks oversimplifying a truly complex system. As data sets become more impossible, we may have to rethink our techniques for exploratory data analysis.

“Not all who wander are lost” - Tolkien

REFERENCES

Aach J., Church G. M., 2001, Bioinformatics, 17, 6, 495

Aghanim N., Akrami Y., Ashdown M., et al., 2020, Astronomy & Astrophysics, 641, A6

Astropy Collaboration, Price-Whelan A. M., Sipőcz B. M., et al., 2018, , 156, 123

Astropy Collaboration, Robitaille T. P., Tollerud E. J., et al., 2013, , 558, A33

Behroozi P. S., Wechsler R. H., Wu H.-Y., 2013a, ApJ, 762, 2, 109

Behroozi P. S., Wechsler R. H., Wu H.-Y., Busha M. T., Klypin A. A., Primack J. R., 2013b, ApJ, 763, 1, 18

Berndt D. J., Clifford J., 1994, in KDD Workshop

Bond J. R., Kofman L., Pogosyan D., 1996, Nature, 380, 6575, 603

Buitinck L., Louppe G., Blondel M., et al., 2013, in ECML PKDD Workshop: Languages for Data Mining and Machine Learning, 108–122

Croton D. J., Gao L., White S. D. M., 2007, Monthly Notices of the Royal Astronomical Society, 374, 4, 1303

Dalal N., White M., Bond J. R., Shirokov A., 2008, ApJ, 687, 1, 12

Dudoit S., Fridlyand J., 2002, Genome biology, 3, RESEARCH0036

DUYGULU P., 2002, European Conference on Computer Vision, 2002, IV, 97

Einasto J., Klypin A. A., Saar E., Shandarin S. F., 1984, Monthly Notices of the Royal Astronomical Society, 206, 529

Fakhouri O., Ma C.-P., Boylan-Kolchin M., 2010, Monthly Notices of the Royal Astronomical Society, 406, 4, 2267

F.R.S. K. P., 1901, The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 2, 11, 559

Gao L., Springel V., White S. D. M., 2005, Monthly Notices of the Royal Astronomical Society: Letters, 363, 1, L66–L70

Gao L., White S. D. M., 2007, Monthly Notices of the Royal Astronomical Society, 377, 1, L5

Guo H., Zheng Z., Zehavi I., et al., 2014, Monthly Notices of the Royal Astronomical Society, 446, 1, 578

Harris C. R., Millman K. J., van der Walt S. J., et al., 2020, Nature, 585, 7825, 357

Hartigan J. A., Wong M. A., 1979, JSTOR: Applied Statistics, 28, 1, 100

Hearin A. P., Zentner A. R., van den Bosch F. C., Campbell D., Tollerud E., 2016, Monthly Notices of the Royal Astronomical Society, 460, 3, 2552

Hofmeyr D. P., 2020, Degrees of Freedom and Model Selection for k-means Clustering

Holmberg E., 1941, ApJ, 94, 385

Hunter J. D., 2007, Computing In Science & Engineering, 9, 3, 90

Izakian H., Pedrycz W., Jamal I., 2015, Engineering Applications of Artificial Intelligence, 39

Jenks G. F., 1967, International Yearbook of Cartography, 7, 186

Johnson J. W., Maller A. H., Berlind A. A., Sinha M., Holley-Bockelmann J. K., 2019, Monthly Notices of the Royal Astronomical Society, 486, 1, 1156

Kravtsov A. V., Berlind A. A., Wechsler R. H., et al., 2004, ApJ, 609, 1, 35

Lin J., Li Y., 2009, in SSDBM, 461–477

Ludlow A. D., Navarro J. F., Boylan-Kolchin M., et al., 2013, Monthly Notices of the Royal Astronomical Society, 432, 2, 1103

MacQueen J. B., 1967, in Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability, edited by L. M. L. Cam, J. Neyman, vol. 1, 281–297, University of California Press

Mansfield P., Kravtsov A. V., 2020, Monthly Notices of the Royal Astronomical Society, 493, 4, 4763

McInnes L., Healy J., Melville J., 2020, UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction

McInnes L., Healy J., Saul N., Großberger L., 2018, Journal of Open Source Software, 3, 29, 861

McKinney W., 2010, in Proceedings of the 9th Python in Science Conference, vol. 445, 51–56, Austin, TX

McKinney W., 2011, Python for High Performance and Scientific Computing, 14

Meinard M., 2009, Information retrieval for music and motion, Springer

Mo H. J., Jing Y. P., White S. D. M., 1996, Monthly Notices of the Royal Astronomical Society, 282, 3, 1096–1104

Mo H. J., White S. D. M., 1996, Monthly Notices of the Royal Astronomical Society, 282, 2, 347–361

Mohajer M., Englmeier K.-H., Schmid V. J., 2011, A comparison of Gap statistic definitions with and without logarithm function

Müllner D., 2013, Journal of Statistical Software, 53, 9, 1

Narayan A., Berger B., Cho H., 2020, bioRxiv

Neath A. A., Cavanaugh J. E., 2012, WIREs Comput. Stat., 4, 2, 199–203

Ng A., Jordan M., Weiss Y., 2002, in Advances in Neural Information Processing Systems, edited by T. Dietterich, S. Becker, Z. Ghahramani, vol. 14, MIT Press

Pedregosa F., Varoquaux G., Gramfort A., et al., 2011a, Journal of machine learning research, 12, Oct, 2825

Pedregosa F., Varoquaux G., Gramfort A., et al., 2011b, Journal of Machine Learning Research, 12, 2825

Pérez F., Granger B. E., 2007, Computing in Science and Engineering, 9, 3, 21

Piscionere J. A., Berlind A. A., McBride C. K., Scoccimarro R., 2015, The Spatial Distribution of Satellite Galaxies Within Halos: Measuring the Very Small Scale Angular Clustering of SDSS Galaxies

Press W. H., Schechter P., 1974, ApJ, 187, 425

Rasmussen C. E., Williams C. K. I., 2006, Gaussian processes for machine learning., Adaptive computation and machine learning, MIT Press

R.Bellman, 1961, Adaptive Control Processes: A Guided Tour, Princeton University Press

Roberts S., Osborne M., Ebden M., Reece S., Gibson N., Aigrain S., 2013, Philosophical Transactions of the Royal Society (Part A

Salcedo A. N., Maller A. H., Berlind A. A., et al., 2018, Monthly Notices of the Royal Astronomical Society, 475, 4, 4411–4423

Sato-Polito G., Montero-Dorta A. D., Abramo L. R., Prada F., Klypin A., 2019, Monthly Notices of the Royal Astronomical Society, 487, 2, 1570–1579

Scoccimarro R., 1998, Monthly Notices of the Royal Astronomical Society, 299, 4, 1097–1118

Sinha M., Garrison L., 2020, Monthly Notices of the Royal Astronomical Society, 491, 3022

Springel V., 2005, Monthly Notices of the Royal Astronomical Society, 364, 4, 1105

Srisawat C., Knebe A., Pearce F. R., et al., 2013, Monthly Notices of the Royal Astronomical Society, 436, 1, 150–162

Szewciw A. O., Beltz-Mohrmann G. D., Berlind A. A., Sinha M., 2021, arXiv e-prints, arXiv:2110.03701

Tibshirani R., Walther G., Hastie T., 2001, Journal of the Royal Statistical Society: Series B (Statistical Methodology), 63, 2, 411

van der Maaten L., Hinton G., 2008, Journal of Machine Learning Research, 9, 2579

Van Rossum G., 2020, The Python Library Reference, release 3.8.2, Python Software Foundation

Virtanen P., Gommers R., Oliphant T. E., et al., 2020, Nature Methods, 17, 261

Virtanen P., Gommers R., Oliphant T. E., et al., 2020, Nature Methods, 17, 261

Wang H. Y., Mo H. J., Jing Y. P., 2007, Monthly Notices of the Royal Astronomical Society, 375, 2, 633

Wang J., Liu P., She M. F., Nahavandi S., Kouzani A., 2013, Biomedical Signal Processing and Control, 8, 6, 634

Wechsler R. H., Bullock J. S., Primack J. R., Kravtsov A. V., Dekel A., 2002, ApJ, 568, 1, 52

Wechsler R. H., Zentner A. R., Bullock J. S., Kravtsov A. V., Allgood B., 2006, ApJ, 652, 1, 71

White S. D. M., Rees M. J., 1978, Monthly Notices of the Royal Astronomical Society, 183, 341

Yin Z., Zhou X., Bakal C., et al., 2007, BMC Bioinformatics, 9, 264

Zheng Z., 2004, Constraining galaxy bias and cosmology using galaxy clustering data, Ph.D. thesis, "Ohio State University

Zwicky F., 1937, ApJ, 86, 217