Number Density Profiles of Galaxy Groups and Clusters in the SDSS

Jackson M. Norris

Undergraduate Senior Honors Thesis
April 20, 2009

Thesis Advisor: Andreas A. Berlind

We use projected number density profiles of galaxy groups to investigate the relationship between galaxy group structure and dark matter structure within halos. We primarily study a specific model of how galaxies populate dark matter halos by comparing the number density profiles of galaxy groups from the Sloan Digital Sky Survey with number density profiles of galaxy groups taken from dark matter halo simulations. We also investigate the dependence of galaxy locations within galaxy groups on luminosity and color.

1) Introduction

Galaxies tend to cluster on large scales. They form the largest gravitationally bound structures in the universe, which we call galaxy groups and galaxy clusters. Typically, the terms *group* and *cluster* describe two parts of the same spectrum of objects, with *cluster* referring to the largest systems of galaxies and *group* referring to smaller systems of galaxies. For the sake of brevity, this paper will only use the term *group* when referring to any system of three or more galaxies in equilibrium.

Galaxies inhabit large clumps of dark matter called halos. We have become increasingly more knowledgeable about the structure of halos and their constituent dark matter via large cosmological N-body simulations. Although we are highly confident of our understanding of dark matter halo structure, we are ironically much less informed about the structure of galaxy groups, particularly with respect to their interaction with their host halos. There is still much to be determined about the mechanisms that cause galaxy clustering within dark matter halos. We expect galaxy clustering to be correlated with halo structure. This is because we are confident that all galaxies reside within dark matter halos and we know that halos are the biggest contribution to the mass of the system. We wish to know about this interaction in much more detail. If we could definitively determine the location of galaxies within halos, we could gain a much greater understanding of the interplay between galaxies and dark matter.

In most contemporary models, the clustering of galaxies is thought to depend on how the galaxies inhabit dark matter halos. This relation is parameterized by the halo occupation distribution (HOD). In order to constrain the location of galaxies within dark matter halos, we analyze different parameterizations of the HOD. The HOD contains three relations: (Berlind & Weinberg 2002)

1) the probability distribution, $P(N \mid M)$, of N galaxies occupying a halo of mass M;

2) the relative spatial distributions of the galaxies and the dark matter within each halo;

3) the relative velocity distributions of the galaxies and the dark matter within each halo.

We can change the parameters of the HOD to create different prescriptions for how galaxies inhabit dark matter halos. Although all of these relations are important, for this paper we are explicitly concerned with constraining the second relation. After creating different prescriptions, we can then apply the prescriptions to dark matter halos found from cosmological N-body simulations. In this way, each parameterization of the HOD (herein referred to as an "HOD model") gives a different prediction of galaxy locations within halos, and thus a different prediction of how galaxies cluster. We can organize these simulated galaxies into galaxy groups in the same way we organize observed galaxies. We describe this grouping algorithm in more detail in §2.2.

We wish to determine the validity of the predictions HOD models make. The metric we adopt to test these HOD models involves projected number density profiles of galaxy groups. Projected number density profiles show the 2-dimensional (projected) number density of galaxies within groups as a function of radius from the center of each group. We can compare the projected number density profiles of both observed galaxy groups and simulated galaxy groups to help us determine which HOD model most accurately reflects the observational data. Thus, by analyzing projected number density profiles we can constrain the different parameters of the HOD and gain a better understanding of how galaxies inhabit dark matter halos.

We investigate one specific fiducial HOD model. The model we choose to investigate is a fairly common parameterization of the HOD that several other investigators assume in their

research (e.g., Zehavi et al. 2005). It is extremely important to ascertain the validity of this HOD model as an accurate model for how galaxies inhabit dark matter halos since the use of this HOD model is so widespread in the literature.

We also investigate the segregation of galaxies based on color and luminosity. Galaxy segregation is the natural splitting of galaxies into two or more distinct subsets within groups, where each subset has its own unique properties that differentiate it from the other subsets. In the case of concentration, which is a measure of the relative number of galaxies near the center compared to the outer galaxies, color segregation has been found in which red galaxies within the largest groups are more concentrated than blue galaxies (Blanton & Berlind 2007). Similarly, in terms of luminosity, we think segregation exists in which bright galaxies are more concentrated than dim galaxies (Kashikawa et al. 1998; Girardi et al. 2003), although this luminosity segregation has not been studied as extensively as color segregation, nor are we as certain that it exists since some investigators have *not* found luminosity segregation (Pracy et al. 2005). We wish to investigate these segregations in more detail by taking number density profiles of different color and luminosity samples over many groups of differing sizes. We expect the number density profiles for each sample to differ moderately from each other; by studying these differences we can gain a clearer picture of how galaxy location is dependent on color and luminosity within groups.

We compare simulations of galaxy groups with observed galaxy groups. The observational data come from the Sloan Digital Sky Survey (SDSS), and is explained in §2.1. Through the SDSS, we have access to one of the largest surveys of galaxies ever available, which allows us to analyze galaxy groups to an extremely high precision. The total galaxy sample of the SDSS probes a large volume of space with a high degree of accuracy, which means that our statistical errors based on sample size are extremely small. This allows us to look at significantly more groups, both large and small, than was previously possible. For example, rather than looking at color segregation in just the largest groups, we can now also stack thousands of the smallest groups together in order to look at trends across a wide range of group sizes, from groups with 3 galaxies up to groups with hundreds of galaxies.

This is an extremely interesting topic because it uses both the statistical significance of a large sample size as well as a large number of simulated galaxy groups from several cosmological N-body simulations. Additionally, the grouping algorithm used by Berlind et al. (2006) is specifically designed to maximize the probability that grouped galaxies lie within the same halo. All of these details demonstrate that our findings of number density within galaxy groups will accurately tell us something about the deeper association between galaxies and dark matter.

2) Data

2.1) SDSS

The galaxy groups used in this project are found by analyzing data from the SDSS. The SDSS is a major multi-filter imaging and redshift survey based in Apache Point, New Mexico that has catalogued approximately 25% of the sky since its beginnings in 2000. It uses a dedicated 2.5-m wide-angle optical telescope that takes photometric data in five different bands called *u*, *g*, *r*, *i*, and *z*. For the purposes of this project, we are primarily concerned with the g and r band. The imaging survey contains $\sim 10^8$ galaxies down to a survey-limited magnitude of $r \sim 22.5$ galaxies, but the spectroscopic survey with which we are primarily concerned consists of $\sim 10^6$ galaxies down to a magnitude of $r \sim 17.7$ (Schlegel et al. 1998). Targets are assigned to spectroscopic plates using an adaptive tiling algorithm (Blanton et al. 2003), and automated pipelines perform spectroscopic data reduction and redshift determinations for the galaxies.

By using the SDSS, we are able to look at a large number of galaxies with accurate color and luminosity information. Even with the dramatic decrease in the number of galaxies from the imaging survey to the spectroscopic survey, there is still a huge number of galaxies available to us. After taking certain incompletenesses and restrictions into account explained in Berlind et al. (2006), the total galaxy sample we use covers about 8000 square degrees of the sky and contains just under 930,000 galaxies. With this kind of sample size, we can afford to make all of the cuts discussed in §2.2 and §2.3 and still retain enough galaxies within our subsamples to make precise measurements of the data.

2.2) Galaxy Samples and Grouping Procedure

All of the galaxies used in this project are from a volume-limited subsample of the SDSS spectroscopic sample. Volume-limited samples are created by pruning the main sample by creating a radial threshold as well as a luminosity threshold of allowable galaxies. Our volume-limited subsample was created by Berlind et al. (2006) with an r-band magnitude threshold of $M_r < -20$. Volume-limited samples are useful because they eliminate many inherent observational selection effects within the data; as we probe deeper into space, it becomes increasingly more difficult to see less luminous objects simply because of the diminishing intensity of light. In order to combat this systematic error, it is useful to create a maximum radial threshold as well as a minimum luminosity threshold for the sample. The radial threshold corresponds to the maximum distance at which we can observe galaxies of the given luminosity threshold. Thus, the radial threshold establishes a fixed volume, within which we are assured that the subsample of galaxies is homogenous and complete.

Our volume-limited sample is complete to a redshift of $z = 0.1$ and down to an r-band magnitude of $M_r = -20$. It contains 10124 groups, each with a minimum of 3 galaxies. There are 52281 total galaxies within the sample that are included within our groups (note that most galaxies in the universe are not within a group, so the number of galaxies within our groups are a fraction of the total number of galaxies within the SDSS sample).

Of course, it is not enough to simply have a catalog of galaxies if we do not know which galaxies lie within groups. The ultimate goal of grouping galaxies together is to place galaxies that lie within the same dark matter halo within the same group. A grouping algorithm must be used to approximate this since it is impossible to directly observe the host halos. Generally, galaxies are found to be within the same group if they are physically close to one another. Grouping is done by Andreas A. Berlind using the same grouping algorithm detailed in Berlind

et al (2006). This algorithm is a friends-of-friends algorithm, which uses two linking lengths to establish which galaxies lie close enough to one another to be considered part of the same group. One linking length is for the direction perpendicular to the line of sight and the other linking length is for the direction parallel to the line of sight. Any galaxies that are closer to one another than the linking lengths will be considered members of the same group.

It is necessary to use two linking lengths in order to help smooth out distance distortions that arise when using redshift to establish distance. Because of the local velocities of galaxies within the same group, some galaxies will be traveling away from the observer while others will be traveling toward the observer. This smears the observed redshifts of the galaxies away from the mean redshift of their group and introduces a deviation from the Hubble's Law relationship between distance and redshift. This deviation will act to elongate the observed distances of individual galaxies away from the mean distance of the group in the direction of the line of sight, an effect commonly called "Fingers of God." Having two linking lengths allows for the linking length parallel to the line of sight to be longer than the linking length perpendicular to the line of sight. This is explained in more detail in Berlind et al (2006).

2.3) Color and Luminosity Separation

One of the goals of this project is to determine how color and luminosity play a role in the structure of galaxy groups. To accomplish this goal, we divide the galaxies into different color and luminosity samples and then determine number density profiles for each sample. The color samples are determined according to a tilted color division in color-magnitude space described by the equation (Blanton & Berlind 2007)

$$^{0.1}(g-r)_c = 0.80 - 0.03(M_{0.1r} + 20).$$

Galaxies with $^{0.1}(g-r)_c$ values greater than this equation are considered red, while galaxies with values lower than this equation are considered blue. The purpose of using a tilted division rather than a straight division at a particular $^{0.1}(g-r)_c$ value is to more easily distinguish the E-S0 ridgeline from the rest of the sample. This ridgeline can be easily seen in Figure 1 by the overdensity of galaxies on the right side of the graph, where most of the reddest galaxies predominately fall. The green line in the figure indicates the tilted color division.

The luminosity samples are found in a similar manner by separating the galaxies at a straight magnitude division of $M_r = -21.0$, indicated by the red line in Figure 1. Galaxies with M_r values greater than -21.0 are considered "dim" while galaxies with M_r values less than -21.0 are considered "bright."

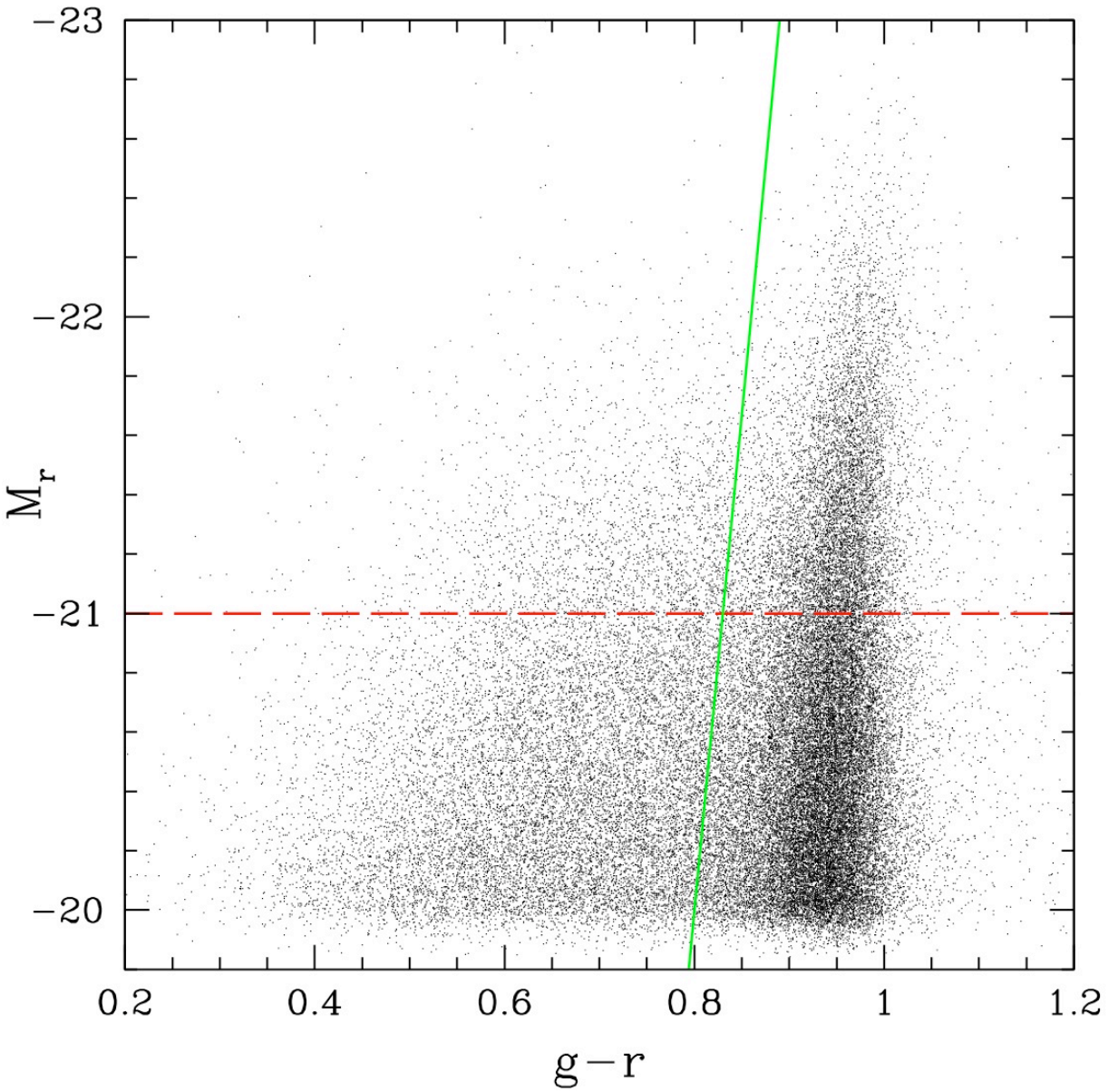

Figure 1: Color-magnitude diagram of galaxies within our sample. Note the E-S0 ridgeline on the right of the graph. The E-S0 ridgeline extends below the graph, but due to the volume-limited nature of the sample, the entire ridgeline cannot be seen. The solid green line and dotted red line correspond to the color and luminosity cuts, respectively. Galaxies above the red line are part of the bright sample, while galaxies below the red line are part of the dim sample. Galaxies to the right of the green line are part of the red sample, while galaxies to the left of the green line are part of the blue sample.

2.4) Mock Galaxies

We are naturally unable to directly observe dark matter halos in the SDSS groups. Since one of the goals of this project is to determine how one HOD model matches observations, it is necessary to have mock galaxy catalogs to which we can compare the SDSS groups. Galaxy

location within dark matter halos is not explicitly known analytically, so these mock galaxy catalogs serve the purpose of a model for comparison. Although analyzing the SDSS data would be interesting by itself, it would not be enough for the purpose of this project to investigate the SDSS data alone, since our goal is to constrain specific aspects of the cosmology of the universe. We explicitly know every aspect of these mock galaxy catalogs, which gives us a good way of isolating the parameters of our HOD model so that we can know exactly to what we are comparing.

The mock galaxy catalogs are created by placing artificial galaxies within halos from cosmological N-body simulations of dark matter created by McBride et al (2009, in preparation). Halos are determined using a friend-of-friends algorithm similar to the one used to find groups; this method looks at all of the N-body "particles" and places those particles that lie within a specific linking length of one another within the same halo. This means that halos are not necessarily neatly spherical objects, but may instead take on many convoluted shapes. Galaxies are placed within these halos according to the prescription established by the HOD model we wish to test. Specifically, the prescription McBride et al. uses places a single central galaxy at the location of the deepest gravitational potential well of each halo. Then, the prescription places satellite galaxies that trace the dark matter throughout the rest of each halo. This means that the probability of a galaxy being placed at a specific location within a halo is directly proportional to the amount of dark matter at that location. Wherever there is a large amount of dark matter, we expect to see a proportional amount of galaxies.

After placing the galaxies within each halo, the artificial galaxies are grouped using the same procedure as the SDSS galaxies. It is important to group both the mock galaxies and the observed galaxies using the same grouping algorithm rather than simply grouping all galaxies within the same halo. This is because we wish to retain any systematic errors that may be caused by the grouping algorithm. For example, the SDSS data has redshift distortions that may erroneously force the grouping algorithm to artificially place a galaxy within the wrong galaxy group. In other words, the grouping algorithm will incorrectly assume that some galaxies are gravitationally bound to others when in fact they are not. This, and other distortions, can cause systematic errors that can be eliminated within the mock galaxy sample since we know precisely where every halo is. Although one might naively assume that it is advantageous to eliminate these errors within the mock galaxies, it is actually extremely important that they are included in the mock galaxy catalogs since the errors reflect what we observe in nature. Therefore, the mock galaxy catalogs contain the same redshift distortions as the SDSS data. We must ensure that the comparison of the SDSS data to the mock galaxy catalogs is a fair comparison and includes the same sources of error for both data.

McBride et al. creates multiple mock galaxy catalogs from different dark matter simulations. Each simulation contains the same statistical similarities in the initial conditions and cosmology the as the actual universe. We use multiple simulations in order to get the full range of possible simulated universes. This increases the statistical significance of our findings.

60 different mock catalogs are used in this study. Each catalog comes from a unique dark matter realization. This means that McBride et al. never applies the same HOD prescription to the same dark matter realization. Because galaxy placement is found using a Monte Carlo method, the exact number of groups and galaxies within each mock galaxy catalog is not constant, but instead deviates moderately from a mean value.

3) Method and Analysis

3.1) Radial Number Density Profiles

For each galaxy catalog, we determine the number density profile. We do this by statistically stacking multiple groups together and counting the number of galaxies that appear within a radial bin across all groups. Each radial bin corresponds to a projected 2-dimensional annulus on the sky that is centered on the centroid of each galaxy group. The number of galaxies that appear within each bin is divided by the area of the corresponding annulus. Thus, the number density of each bin is determined by the equation

$$n_{\perp,\mathrm{bin}} = \frac{N_{\mathrm{bin}}}{\pi\left(r_{outer}^2 - r_{inner}^2\right)} \frac{1}{N_{\mathrm{group}}}$$

where N_{bin} is the number of galaxies found within the annulus, N_{group} is the total number of groups within the sample, and r_{outer} is the outer radius of the annulus, and r_{inner} is the inner radius of the annulus.

We limit the sizes of groups that are stacked together so that we only include galaxies that meet certain minimum and maximum requirements on the number of galaxies contained within each group. This is done to eliminate artificial alterations to the number density profiles that are an artifact of differing group sizes. For example, groups with 20 galaxies are generally larger than groups with only 3 galaxies, so it would be unfair to compare the number density profile of the 20-galaxy sample with the 3-galaxy sample. If we stack groups that have such a disparate number of galaxies, we get an artificial number density spike at small radii and a dip at large radii simply because the smaller groups do not have galaxies at the large physical radii that larger groups have. Taking this into account, we look at three different ranges of group size: groups with 3-5 galaxies, groups with 6-9 galaxies, and groups with 10 or more galaxies.

We stack a large number of groups because there are not enough galaxies to accurately establish a number density profile for any single group. Since we are stacking a large number of groups, we divide by the number of groups in order to see the "average" number density profile of a sample. Additionally, dividing by the number of groups has the added benefit of making it easier to compare the SDSS data with the mock galaxy catalogs since the mock catalogs may contain more groups.

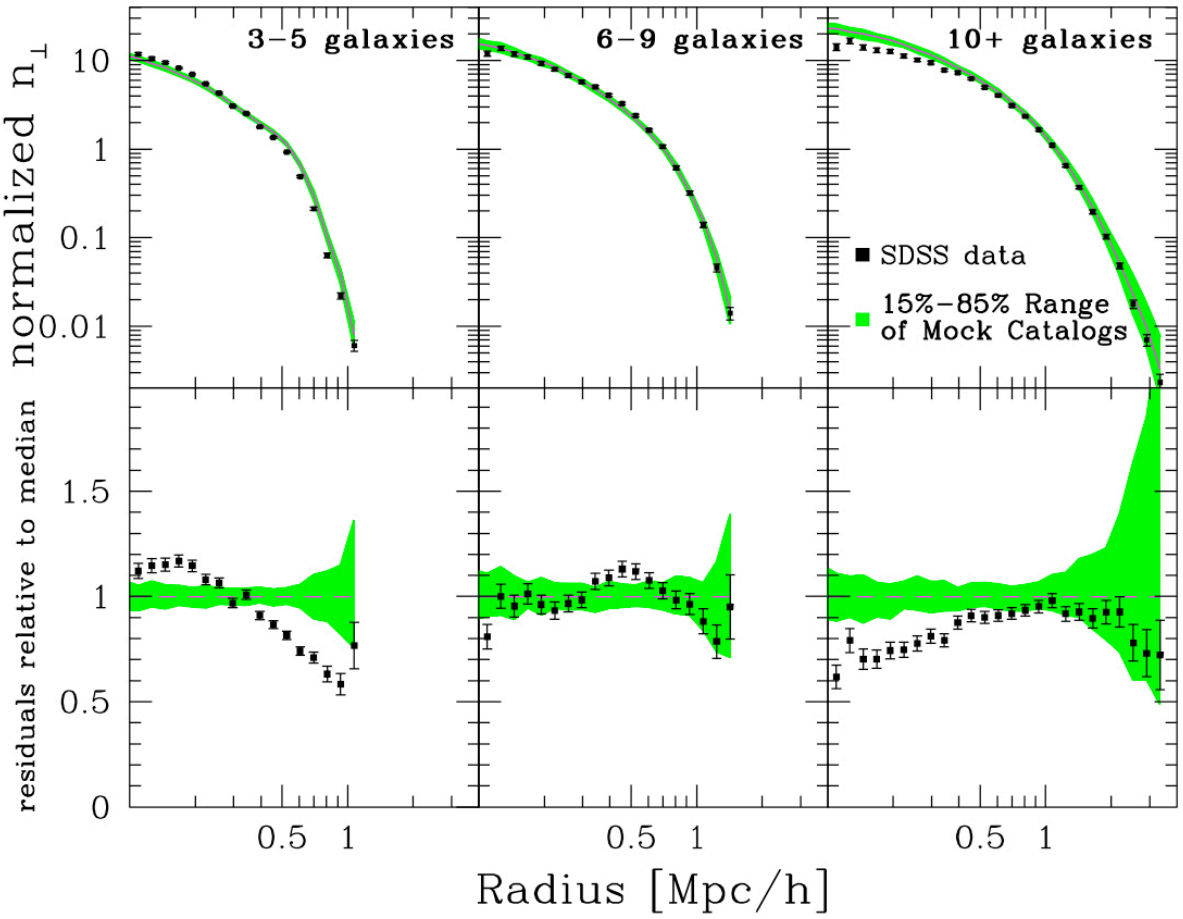

Figure 2: Number density profiles measured for SDSS galaxies (black) and mock galaxies (green). We normalize each profile by the total number of groups in each catalog, in order to facilitate comparison. The number of galaxies within each group, and thus the size of the group, increases from left to right, so that each graph corresponds to a different sample of galaxy sizes. The bottom panels show residuals by dividing the profiles by the median mock profile. The dashed magenta line in the bottom graphs represents the median profile.

Figure 2 shows the average number density profiles of the SDSS data and mock catalogs within our size restrictions. The green band corresponds to the 15-85 percentiles of all the mock galaxy catalogs within each bin. Each graph is labeled with galaxy numbers that correspond to the size of the groups used in constructing the graph (i.e. "3-5 galaxies" indicates that only groups containing 3, 4, or 5 galaxies are included in the stacking algorithm). The error bars on the SDSS data are Poisson errors based on the square root of the number of galaxies found within each bin. In an effort to make the statistical significance more apparent, we also present residual graphs created by dividing all of the profiles by the median mock profile.

As Figure 2 indicates, the SDSS data do not precisely fit the data from the mock catalogs. The figure indicates something unexpected: galaxy concentration within dark matter halos may be dependent on group size. Groups with 3-5 galaxies demonstrate a higher concentration than the mock galaxy catalogs, groups with 6-9 galaxies demonstrate a roughly similar concentration, and groups with 10 or more galaxies have a lower concentration. This is an unexpected result since no one has investigated concentrations of galaxies change as a function of group size. Conceptually, this means that galaxies within small groups probably lie closer to the centers of their host dark matter halos, while galaxies within larger groups probably have a more diffuse placement within their host dark matter halos. If this is correct, then it implies that another parameter may need to be added to the HOD, one that changes the concentration parameter of galaxies depending on how many galaxies are expected to inhabit halo.

It is important to remember that physical three-dimensional number density profiles of galaxy groups may not actually look like our profiles. Most notably, we stack multiple groups together, so if one analyzes any single galaxy group, it is unlikely that a number density profile would be constructed that would look exactly like our profiles. To elucidate this, if we could calculate the number density profiles of individual groups to higher precision, it is reasonable to assume that there would be large degree of scatter across all of the profiles. Also, recall that the number density profiles used in this project are calculated via a projected two-dimensional surface area, so a 3-dimensional analysis may reveal a different shape of the number density profiles altogether. However, we did not perform a 3-dimensional analysis since redshift-space distortions would make it extremely difficult to do so. Additionally, due to selection effects based on luminosity limitations, we may be excluding some galaxies within each group because they are too faint for us to detect. However, any of these selection effects should be included within the mock galaxy catalogs as well, by design.

3.2) Color Segregation

In addition to investigating the composite[1] profiles discussed in §3.1, we also investigate galaxy segregation by color in the profiles of different color samples. As discussed in the introduction, it has been found that red galaxies are more concentrated than blue galaxies, so we should expect a similar difference in concentration when studying number density profiles. To investigate this, we create separate number density profiles for the red and blue galaxy samples described in §2.3. Unlike the composite profiles, however, we compare the two color samples to one another rather than to mock galaxy catalogs.

Rather than normalize by the number of total groups within each sample as done with the composite profiles, we instead normalize the profiles by the total number of galaxies within each color sample. This is because red galaxies are naturally more numerous than blue galaxies at all locations within a group. Normalizing by the total number of groups within the color samples would simply show that the number density of red galaxies is generally greater than the number density of blue galaxies. This has been well established already (Zehavi et al. 2005). What we are instead interested in is the concentration of blue galaxies compared to the concentration of red galaxies. Colloquially, we are interested in comparing the *shapes* of the number density profiles.

[1] For the sake of clarification, we use the term "composite" to refer to the profiles from Figure 2.

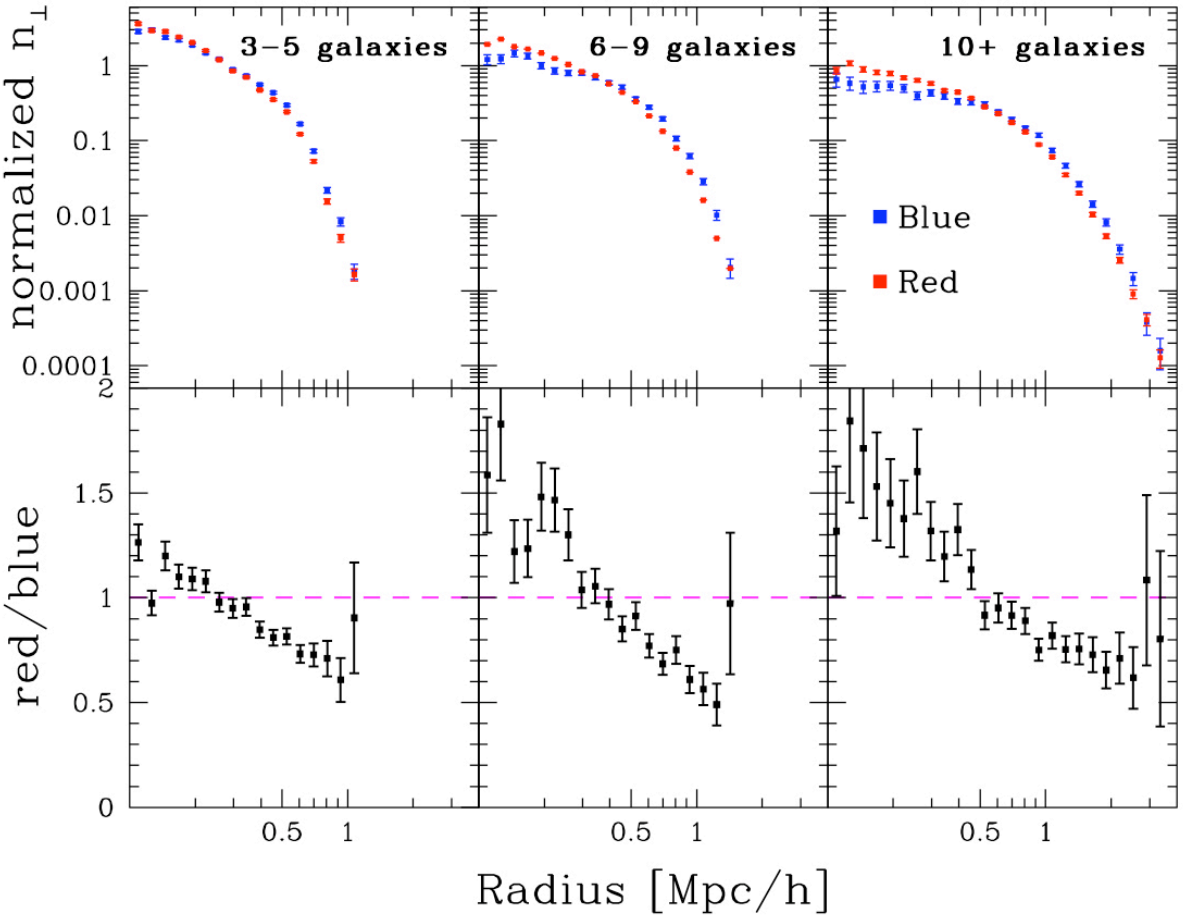

Figure 3: Radial number density profiles measured separately for red and blue galaxies. Since groups contain more red galaxies than blue galaxies, we normalize each profile by the total number of red and blue galaxies, respectively, in order to facilitate their comparison. The top panels show the red and blue profiles while the bottom panels show their ratio. The number of galaxies within each group, and thus the size of the group, increases from left to right, so that each panel corresponds to a different sample of galaxy sizes. The bottom panel shows residuals, and is found by dividing the number density of red galaxies over the number density of blue galaxies, with errors found by combining the number densities associated errors in quadrature. The magenta line in the bottom graphs is placed at $n_{\perp,\mathrm{red}}/n_{\perp,\mathrm{blue}} = 1$ in order to better mark the point when there are equal number densities of both color samples.

As Figure 3 shows, we see the expected color segregation. Note that at small radii, the normalized number density of the red galaxies is greater than the normalized number density of blue galaxies. This difference is much more clearly seen in the bottom panels of the figure, which show the residuals of the red profile divided by the blue profile. Simplifying the calculation of residuals gives

$$\frac{n_{\perp,\text{red}}}{n_{\perp,\text{blue}}} = \frac{N_{bin,red}}{N_{bin,blue}} \frac{N_{galaxies,blue}}{N_{galaxies,red}},$$

where $N_{bin,red}$ is the number of red galaxies within a particular radial bin, $N_{bin,blue}$ is the number of blue galaxies within a particular radial bin, $N_{galaxies,red}$ is the number of all red galaxies within the sample, and $N_{galaxies,blue}$ is the number of all blue galaxies within the sample. The color segregation is more pronounced for larger groups. Note the greater disparity in $n_{\perp,\text{red}}$ and $n_{\perp,\text{blue}}$ within the residuals of the larger galaxy groups. This confirms the initial expectation of color segregation, and also indicates that segregation is stronger in larger groups.

This corresponds with what we expect to find based on the investigation of Yang et al. (2005). What is really interesting about this investigation of color segregation is the fact that we are looking at aggregate profiles of many small groups. As indicated within the introduction, most of the previous work on color segregation looked at single profiles for the largest groups. Our data indicate that color segregation extends to the smallest groups as well.

3.3) Luminosity Segregation

Much like our color analysis in §3.2, we also investigate the dependence of luminosity on number density profiles. As before, our goal is to first see if luminosity segregation exists, and then to see how the concentration of each luminosity sample is different. We create separate number density profiles for each luminosity sample described in §2.3.

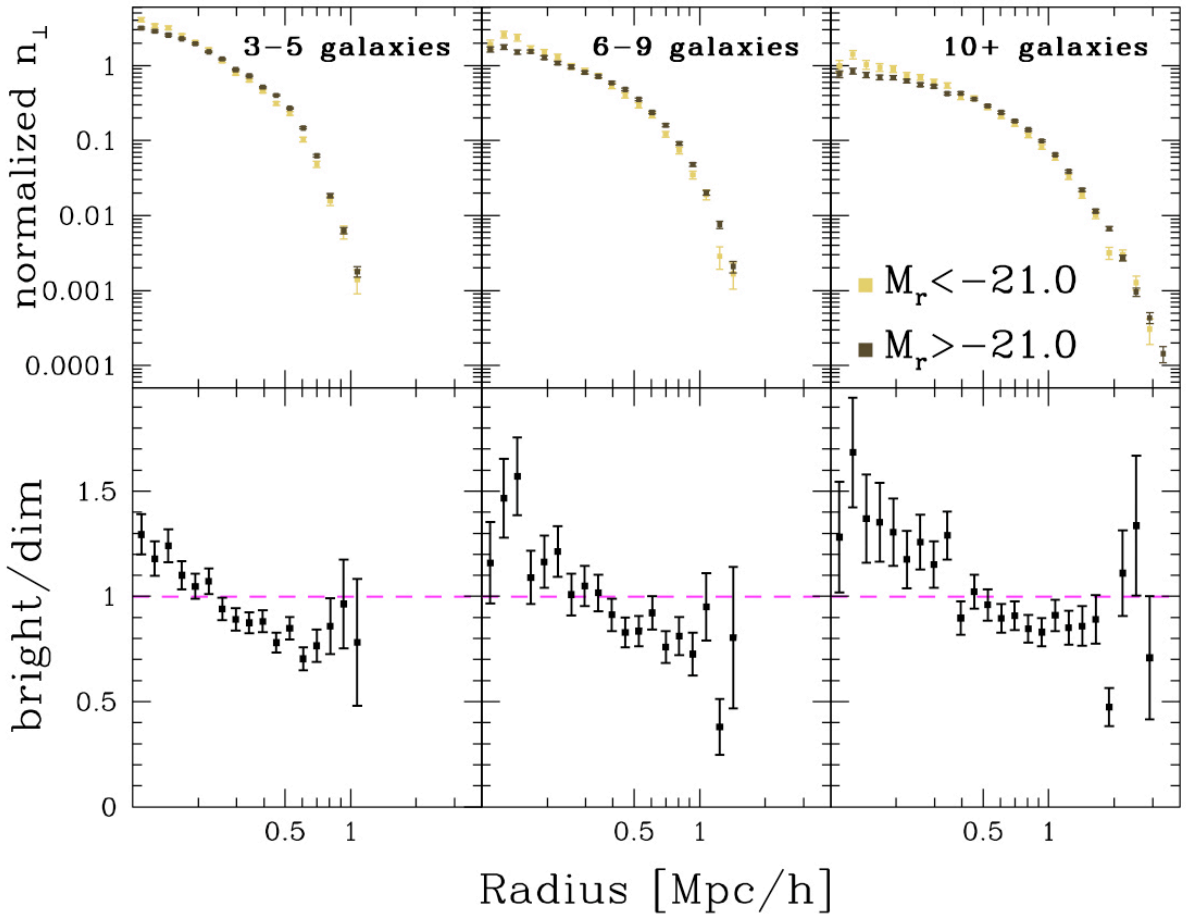

Figure 4: Radial number density profiles measured separately for bright and dim galaxies. As with the red and blue galaxy separation, we normalize the bright and dim number density profiles by the total number of bright and dim galaxies, respectively, in order to facilitate their comparison. The top panels show the profiles while the bottom panels show their ratio. The number of galaxies within each group, and thus the size of the group, increases from left to right, so that each graph corresponds to a different sample of galaxy sizes. The bottom panel shows residuals, and is found by dividing the number density of bright galaxies over the number density of dim galaxies, with errors found by combining the number densities associated errors in quadrature. The magenta line in the bottom graphs is placed at $n_{\perp,\text{bright}}/n_{\perp,\text{dim}} = 1$ in order to better elucidate when there are equal number densities of both luminosity samples.

In Figure 4, we can see clear evidence of luminosity segregation. However, the luminosity segregation is less pronounced than the color segregation, which may be an indicator that color has more dominating effect than luminosity in regards to galaxy locations within groups. Since there is a correlation between red galaxies and the brightest galaxies corresponding to the E-S0 ridgeline, (*see Figure 1*) one might expect this segregation to simply

be a product of the luminosity-color correlation, with the bright galaxies tracing the red galaxies within the profiles. This is corroborated by the fact that the luminosity segregation is greater for the larger groups, just as it is with color segregation. However, this reasoning leaves out other possibilities. It is more likely that both luminosity and color have different roles to play in the location of galaxies within galaxy groups. This is likely not related to galaxy interactions alone, but probably also has relations to dark matter halo structure.

Knowing the interaction between color and luminosity and galaxy location within groups would be extremely interesting and more work should be done on this subject. In addition, there are several ideas that try to determine other theoretical causes for why inner galaxies are brighter than outer galaxies within galaxy groups. For example, Zentner et al. (2005) discusses the possibility that dynamical friction — a force that acts to retard the speeds of small objects within large masses — may cause the heaviest and brightest galaxies to quickly sink to the center of a galaxy group. Unfortunately, none of these theoretical causes are definitive. Future methods for investigating this connection are covered in more depth within §4.

4) Future Work

Although the research completed thus far is promising, there is still more to be done. Future concerns include:

1. Testing other HOD models. As I outlined in §3.1 and §3.2, the SDSS data indicate that groups may be more clustered or less clustered depending on group size. Alternate HOD models may agree more with the data. Additionally, we should look at HOD models that do not necessarily assign a central galaxy, but instead treat every galaxy as a satellite galaxy. By looking at these other HOD models, we can more easily constrain the HOD parameters.
2. So far, we only have one large volume-limited sample from the SDSS data. It would be useful to see how the number density profiles change depending on where one places the lower luminosity cut.
3. We would like to devise a method of easily comparing groups of different sizes to determine if there is an easily understood universal profile. Comparing groups of different sizes will likely involve normalizing the radius of each galaxy within a group. Early attempts of finding a method of radial normalization proved problematic due to statistical artifacts within the number density profiles.
4. Lastly, we wish to investigate whether luminosity or color has a more dominating effect on galaxy location within groups. We have shown that red galaxies are more concentrated than blue galaxies, and we know that bright galaxies are more concentrated than dim galaxies. Moreover, we know that there is a strong correlation between luminosity and color: most luminous galaxies tend to be red. With this in mind, we want to know if galaxy concentration is more dependent on luminosity while color concentration is simply "graphed onto" the luminosity concentration? Is the opposite true? In order to determine this, we are currently in the process of analyzing number density profiles of samples that have both luminosity and color cuts at the same time. For example, instead of comparing the profile of all red galaxies with the profile of all blue galaxies, we compare the profiles of only the luminous red galaxies with that of luminous blue galaxies.

5) Summary

This paper measures projected radial number density profiles of galaxy groups. We investigate composite number density profiles as well as number density profiles of color and luminosity separated samples. We compare number density profiles of galaxy groups from the SDSS with predicted mock catalogs based on an HOD model of how galaxies are placed within dark matter halos. The mock galaxy groups are found by placing galaxies within dark matter halos using a fiducial HOD model that places a single galaxy within each halo, and then places satellite galaxies throughout the halo that trace the dark matter density. Our main results are:

1. For the smallest groups (Figure 1, groups with 3-5 galaxies), number density profiles from the SDSS data indicate that groups have a significantly higher concentration than the mock galaxy catalogs. This indicates that galaxies may be more concentrated within their host dark matter halos if the groups (and thus, halos) are small.
2. For medium-sized groups (Figure 1, groups with 6-9 galaxies), the number density profiles from the SDSS data indicate a general agreement with mock galaxy catalogs.
3. For larger groups (Figure 1, groups with 10 or more galaxies), number density profiles from the SDSS data indicate that groups have a significantly lower concentration than the mock galaxy catalogs. This indicates that galaxies may be less concentrated within their host dark matter halos if the groups are large.
4. Number density profiles from color-separated samples of galaxies reveal that color segregation exists, in which red galaxies are more concentrated than blue galaxies. Furthermore, this segregation is most pronounced for the largest groups.
5. Number density profiles from luminosity separated samples of galaxies reveals that luminosity segregation exists, in which bright galaxies are more concentrated than dim galaxies. Furthermore, this segregation is most pronounced for the largest groups.

References

Berlind, A. A. & Weinberg, D. H., 2002, ApJ, 575, 587
Berlind, A. A., et al., 2006, ApJS, 167, 1
Blanton, M. R. & Berlind, A. A., 2007, ApJ, 664, 791
Blanton, M. R., Lin, H., Lupton, R. H., Maley, F. M., Young, N., Zehavi, I., & Loveday, J., 2003, ApJ, 125, 2276
Girardi, M., Rigoni, E., Mardirossian, F., Mezzetti, M., 2003, A&A, 406, 403
Kashikawa, N. et al., 1998, ApJ, 500, 750
Pracy et al., 2005, MNRAS, 364, 1147
Schlegel, D. J., Finkbeiner, D. P., Davis, M., 1998, ApJ, 500, 525
Yang, X., Mo, H. J., van den Bosch, F. C., Weinmann, S. M., Li, C., Jing, Y. P., 2005, MNRAS, 362, 711
Zehavi, I., Zheng, Z., Weinberg, D. H., Frieman, J., Berlind, A. A. et al., 2005, ApJ, 630, 1
Zentner, A. R., Berlind, A. A., Bullock, J. S., Kravtsov, A. V., & Wechsler, R. H., 2005, ApJ, 624, 505

CODE

The mathematics and statistical analysis used in this project was almost entirely done using the C programming language and was created from scratch by Jackson Norris. All plots were generated using the software package SuperMongo using scripts also created by Norris. We have attached the code in C that is used to generate the number density profiles for all of the samples used in this project.

First, I present the code of functions that I link within the main code. Then I present my main code that calculates number density profiles.

Functions Used in Code:

```
double get_gr_cut(double Mr);
double get_density(double binmass,double lgdr,double lgstartradius,int j);
int get_bin_number(double dr,double radius,double start);
double get_normalradius_BAD_RADIUS(double gx,double gy,double gz,double mx,double
my,double mz,double gredshift);
double get_normalradius(double gx,double gy,double gz,double mx,double my,double
mz,double gredshift,double rmass);
double *sort_rluminosity(int totalgroup,int *rid,double *rluminosity);
double *sort_rmass(int totalgroup,int *rid,double *rmass);
int *sort_rn(int totalgroup,int *rid,int *rn);
int *sort_rid(int totalgroup,int *rid);
int *totalmember_maker(int g);
int *totalgroup_maker(int g);
int kill_test(int N1,int N2,int b);

/***********************************/
double get_gr_cut(double Mr)
{
  double gr;
  gr=.8-.03*(Mr+20);
  return gr;
}

double get_density(double binmass,double lgdr,double lgstartradius,int j)
{
  double pi, density;
  pi=3.14159265358979;
  if(j==0)
    {
      density=binmass/(pi*pow(pow(10,lgstartradius+lgdr),2));
    }
  else
    {
      density=binmass/(pi*pow(pow(10,lgstartradius+lgdr),2)-
pi*pow(pow(10,lgstartradius),2));
    }
return density;
}
```

```
int get_bin_number(double dr,double radius,double start)
{
  //note: bin numbering starts with 0
  int bin_number;

  bin_number=(int)((radius-start)/dr);
  return bin_number;
}

double get_normalradius_BAD_RADIUS(double gx,double gy,double gz,double mx,double
my,double mz,double gredshift)
{
  double pi,c,H0,radius,normalradius,theta;
  c=299792.458; /* km/s */
  pi=3.14159265358979;
  H0=100; /*  (km/s)/Mpc  */
  theta=acos(gx*mx+gy*my+gz*mz);
  if(theta>=pi/2) theta=theta-pi/2;
  radius=(gredshift*c/H0)*tan(theta);
  normalradius=radius/1;
  return (normalradius);
}
double get_normalradius(double gx,double gy,double gz,double mx,double my,double
mz,double gredshift,double rmass)
{
  double pi,c,H0,radius,normalradius,theta;
  int k;
  c=299792.458; /* km/s */
  pi=3.14159265358979;
  H0=100; /*  (km/s)/Mpc  */
  theta=acos(gx*mx+gy*my+gz*mz);
  if(theta>=pi/2) theta=theta-pi/2;
  radius=(gredshift*c/H0)*tan(theta);
  normalradius=radius/rmass;
  if(normalradius == 0)
    {
      for(k=0;k<1e99;k++)
    {
      fprintf(stderr,"ERROR RADIUS%d = 0\n",k);
    }
    }
  return (normalradius);
}

/*************/

double *sort_rluminosity(int totalgroup,int *rid,double *rluminosity)
{
  double *ridsort,*rluminositysort;
  int i,i2;
  ridsort=(double *) calloc(totalgroup,sizeof(double));
  rluminositysort=(double *) calloc(totalgroup,sizeof(double));
```

```
  for(i=0;i<totalgroup;i++)
    {
      ridsort[i]=1e9;
    }
  for(i=0;i<totalgroup;i++)
    {
      for(i2=0;i2<totalgroup;i2++)
     {
       if(i==0)
         {
           if(rid[i2]<ridsort[i])
       {
         ridsort[i]=rid[i2];
         rluminositysort[i]=rluminosity[i2];
       }
         }
       else
         {
           if(rid[i2]<ridsort[i] && rid[i2] > ridsort[i-1])
       {
         ridsort[i]=rid[i2];
         rluminositysort[i]=rluminosity[i2];
       }
         }
     }
    }
  return (rluminositysort);
}

/*****************/

double *sort_rmass(int totalgroup,int *rid,double *rmass)
{
  double *ridsort,*rmasssort;
  int i,i2;
  ridsort=(double *) calloc(totalgroup,sizeof(double));
  rmasssort=(double *) calloc(totalgroup,sizeof(double));
  for(i=0;i<totalgroup;i++)
    {
      ridsort[i]=1e9;
    }
  for(i=0;i<totalgroup;i++)
    {
      for(i2=0;i2<totalgroup;i2++)
     {
       if(i==0)
         {
           if(rid[i2]<ridsort[i])
       {
         ridsort[i]=rid[i2];
         rmasssort[i]=rmass[i2];
       }
         }
```

```
      else
        {
          if(rid[i2]<ridsort[i] && rid[i2] > ridsort[i-1])
      {
        ridsort[i]=rid[i2];
        rmasssort[i]=rmass[i2];
      }
        }
    }
    }
  return (rmasssort);
}

/**************/

int *sort_rn(int totalgroup,int *rid,int *rn)
{
  double *ridsort;
  int *rnsort;
  int i,i2;
  ridsort=(double *) calloc(totalgroup,sizeof(double));
  rnsort=(int *) calloc(totalgroup,sizeof(int));
  for(i=0;i<totalgroup;i++)
    {
      ridsort[i]=1e9;
    }
  for(i=0;i<totalgroup;i++)
    {
      for(i2=0;i2<totalgroup;i2++)
    {
      if(i==0)
        {
          if(rid[i2]<ridsort[i])
      {
        ridsort[i]=rid[i2];
        rnsort[i]=rn[i2];
      }
        }
      else
        {
          if(rid[i2]<ridsort[i] && rid[i2] > ridsort[i-1])
      {
        ridsort[i]=rid[i2];
        rnsort[i]=rn[i2];
      }
        }
    }
    }
  return (rnsort);
}

/******************/
```

```
int *sort_rid(int totalgroup,int *rid)
{
  int *ridsort, i,i2;
  ridsort=(int *) calloc(totalgroup,sizeof(int));
  for(i=0;i<totalgroup;i++)
    {
      ridsort[i]=1e9;
    }
  for(i=0;i<totalgroup;i++)
    {
      for(i2=0;i2<totalgroup;i2++)
     {
       if(i==0)
         {
           if(rid[i2]<ridsort[i])
       {
         ridsort[i]=rid[i2];
       }
         }
       else
         {
           if(rid[i2]<ridsort[i] && rid[i2] > ridsort[i-1])
       {
         ridsort[i]=rid[i2];
       }
         }
     }
    }
  return (ridsort);
}

/***********************/

int *totalmember_maker(int g)
{
  int *totalmember;
  totalmember=(int *) calloc(61,sizeof(int));

  if(g==20)
    {
      totalmember[0]=52281;
      totalmember[1]=44787;
      totalmember[2]=45039;
      totalmember[3]=46981;
      totalmember[4]=46358;
      totalmember[5]=42070;
      totalmember[6]=41732;
      totalmember[7]=43205;
      totalmember[8]=41725;
      totalmember[9]=41666;
      totalmember[10]=46798;
      totalmember[11]=44575;
      totalmember[12]=41564;
```

```
      totalmember[13]=43754;
      totalmember[14]=42021;
      totalmember[15]=43896;
      totalmember[16]=43894;
      totalmember[17]=45232;
      totalmember[18]=44863;
      totalmember[19]=42730;
      totalmember[20]=43858;
      totalmember[21]=40299;
      totalmember[22]=46685;
      totalmember[23]=42071;
      totalmember[24]=43739;
      totalmember[25]=45091;
      totalmember[26]=41667;
      totalmember[27]=44683;
      totalmember[28]=44210;
      totalmember[29]=45861;
      totalmember[30]=44006;
      totalmember[31]=41065;
      totalmember[32]=46657;
      totalmember[33]=49345;
      totalmember[34]=40699;
      totalmember[35]=44091;
      totalmember[36]=42218;
      totalmember[37]=42037;
      totalmember[38]=44565;
      totalmember[39]=45910;
      totalmember[40]=45191;
      totalmember[41]=41984;
      totalmember[42]=46058;
      totalmember[43]=43637;
      totalmember[44]=42172;
      totalmember[45]=44733;
      totalmember[46]=45594;
      totalmember[47]=44564;
      totalmember[48]=44212;
      totalmember[49]=44277;
      totalmember[50]=46605;
      totalmember[51]=47469;
      totalmember[52]=38775;
      totalmember[53]=43028;
      totalmember[54]=46002;
      totalmember[55]=42459;
      totalmember[56]=43908;
      totalmember[57]=46057;
      totalmember[58]=45690;
      totalmember[59]=43210;
      totalmember[60]=42517;
    }
  return (totalmember);
}

/**************/
```

```
int *totalgroup_maker(int g)
{
  int *totalgroup;
  totalgroup=(int *) calloc(61,sizeof(int));
  if(g==20)
    {
      totalgroup[0]=10124;
      totalgroup[1]=8107;
      totalgroup[2]=7938;
      totalgroup[3]=8285;
      totalgroup[4]=8065;
      totalgroup[5]=7583;
      totalgroup[6]=7558;
      totalgroup[7]=7738;
      totalgroup[8]=7670;
      totalgroup[9]=7548;
      totalgroup[10]=8189;
      totalgroup[11]=7990;
      totalgroup[12]=7525;
      totalgroup[13]=7654;
      totalgroup[14]=7464;
      totalgroup[15]=7852;
      totalgroup[16]=7805;
      totalgroup[17]=7948;
      totalgroup[18]=7860;
      totalgroup[19]=7628;
      totalgroup[20]=7842;
      totalgroup[21]=7460;
      totalgroup[22]=8258;
      totalgroup[23]=7595;
      totalgroup[24]=7925;
      totalgroup[25]=8024;
      totalgroup[26]=7592;
      totalgroup[27]=8011;
      totalgroup[28]=7768;
      totalgroup[29]=8196;
      totalgroup[30]=7985;
      totalgroup[31]=7418;
      totalgroup[32]=8158;
      totalgroup[33]=8502;
      totalgroup[34]=7399;
      totalgroup[35]=7802;
      totalgroup[36]=7519;
      totalgroup[37]=7616;
      totalgroup[38]=8034;
      totalgroup[39]=8003;
      totalgroup[40]=8053;
      totalgroup[41]=7539;
      totalgroup[42]=8180;
      totalgroup[43]=7793;
      totalgroup[44]=7665;
      totalgroup[45]=7737;
```

```
      totalgroup[46]=8076;
      totalgroup[47]=7980;
      totalgroup[48]=7704;
      totalgroup[49]=7698;
      totalgroup[50]=8158;
      totalgroup[51]=8184;
      totalgroup[52]=7210;
      totalgroup[53]=7577;
      totalgroup[54]=8072;
      totalgroup[55]=7695;
      totalgroup[56]=7962;
      totalgroup[57]=8176;
      totalgroup[58]=8227;
      totalgroup[59]=7563;
      totalgroup[60]=7666;
    }
  return (totalgroup);
}

/*******************/

int kill_test(int N1,int N2,int b)
{
  int kill_value=0;
  if(N1<0)
    {
      fprintf(stderr,"\nERROR:  MINIMUM GROUP SIZE, %d, IS NEGATIVE\n\n",N1);
      kill_value++;
    }
  if(N2<0)
    {
      fprintf(stderr,"\nERROR:  MAXIMUM GROUP SIZE, %d, IS NEGATIVE\n\n",N2);
      kill_value++;
    }
  if(N2<N1)
    {
      fprintf(stderr,"\nERROR:  MAXIMUM GROUP SIZE IS LESS THEN MINIMUM GROUP SIZE, %d
< %d\n\n",N2,N1);
      kill_value++;
    }
  if(b<=0)
    {
      fprintf(stderr,"\nERROR:  NUMBER OF BINS, %d, IS NOT A POSITIVE INTEGER\n\n",b);
      kill_value++;
    }
  return (kill_value);
}
```

Main Code

```
/*****FILE Mr N1 N2 R1 R2 bins lcut verbose*****/
/*ONLY DOES Mr20 SAMPLE*/
/*takes information from the data set given by "Mr" and produces a radial density
profile, color separation profile, and luminosity separation profile, using only
galaxies between N1 and N2 and between radii of R1 and R2 [in Mpc]*/
/*number of bins given by "bins"*/
/*luminosity cut given by lcut and should be a POSITIVE NUMBER*/
/*for ALL galaxies, N1 = 3 and N2 = 0; for no cap on maximum number of galaxies,N2=0;
for example, N1=8 N2=0 will look at all groups with 8 galaxies or more*/
/*"Mr", "N1", "N2", "bins","bins" must all be integers*/
/*number of total groups and total members must be typed in manually within the code
under the "totalmember" and "totalgroup" arrays*/

#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include "functions_densitizer2.2.c"

#define sqr(x) ((x)*(x))

int main(int argc, char *argv[])
{
  double *gra, *gdec, *gredshift, *gmr, *ggr, sigma, *rperp, gredge, *mra, *mdec,
*mredshift, *mmr, *mgr,mredge, *gx, *gy, *gz, *mx, *my, *mz, *theta, *radius,
*normalradius, maxradius, minradius,lgmaxradius, lgminradius, c, H0,
pi,*density,*densityred,*densityblue,*densitybright,*densitydim,*densityreddim,*densit
ybluedim,*densitybrightred,*densitydimred,*densityredbright,*densitybluebright,*densit
ybrightblue,*densitydimblue,lgdr,lcut,lcutname,dummy;
  int N1,N2,N1name,N2name,*gid, *n, *mid, fibcol, *totalgroup, *totalmember,g, h, i,
j,
threshold,b,*binmass,*binmassred,*binmassblue,*binmassbright,*binmassdim,*binmassreddi
m,*binmassbluedim,*binmassbrightred,*binmassdimred,*binmassredbright,*binmassbluebrigh
t,*binmassbrightblue,*binmassdimblue,totalmassgal,totalmassgalred,totalmassgalblue,tot
almassgalbright,totalmassgaldim,totalmassgalreddim,totalmassgalbluedim,totalmassgalbri
ghtred,totalmassgaldimred,totalmassgalredbright,totalmassgalbluebright,totalmassgalbri
ghtblue,totalmassgaldimblue,totalmassgr,totalmassgrred,totalmassgrblue,totalmassgrbrig
ht,totalmassgrdim,totalmassgrreddim,totalmassgrbluedim,totalmassgrbrightred,totalmassg
rdimred,totalmassgrredbright,totalmassgrbluebright,totalmassgrbrightblue,totalmassgrdi
mblue,  mocknumb, member_start, member_end, verbose,
gals_not_counted_low,gals_not_counted_high;
  FILE *groupdata,
*memberdata,*store,*storered,*storeblue,*storebright,*storedim,*storereddim,*storeblue
dim,*storebrightred,*storedimred,*storeredbright,*storebluebright,*storebrightblue,*st
oredimblue;
  char
groupname[100],membername[100],storename[100],storenamered[100],storenameblue[100],sto
renamebright[100],storenamedim[100],storenamereddim[100],storenamebluedim[100],storena
mebrightred[100],storenamedimred[100],storenameredbright[100],storenamebluebright[100]
,storenamebrightblue[100],storenamedimblue[100],dummychar;

  sscanf(argv[1],"%d",&g);   /*Mr*/
```

```
  sscanf(argv[2],"%d",&N1);
  sscanf(argv[3],"%d",&N2);
  sscanf(argv[4],"%lf",&minradius);
  sscanf(argv[5],"%lf",&maxradius);
  sscanf(argv[6],"%d",&b);   /*number of bins desired*/
  sscanf(argv[7],"%lf",&lcut);  /*luminosity cut*/
  sscanf(argv[8],"%d",&verbose);

  lgminradius=log10(minradius);
  lgmaxradius=log10(maxradius);
  lgdr=(lgmaxradius-lgminradius)/(double)b;  fprintf(stderr,"lgmin=%lg lgmax=%lg
lgdr=%lg\n\n",lgminradius,lgmaxradius,lgdr);
  if(lgdr<=0)
    {
      fprintf(stderr,"\nERROR:  lgdr <= 0: %lf\nABORTING\n\n",lgdr);
      return 0;
    }
  threshold=1;
  lcutname=lcut;
  lcut=lcut*(-1);


  /*ORDER IS IMPORTANT FOR THE N1 AND N2 MANIPULATION*/
  N2name=N2;
  N1name=N1;
  if(N1<3 && N1>=0)  /*no groups with less than 3 members; makes file names remain
more cohesive*/
    {
      N1=3;
      N1name=3;
    }
  if(N2==0)
    {
      N2name=0;
      N2=1e9;   /*sets no cap on galaxies within a group*/
    }
  if(kill_test(N1,N2,b) > 0) return 0;

  if(verbose != 0) fprintf(stderr,"Mr%d; N1=%d; N2=%d; bins=%d;
lcut=%4.1lf\n",g,N1name,N2name,b,lcutname);

  mocknumb=60;
  totalmember=(int *) calloc((mocknumb+1),sizeof(int));
  totalgroup=(int *) calloc((mocknumb+1),sizeof(int));
  totalmember=(int *) calloc((mocknumb+1),sizeof(int));
  totalgroup=(int *) calloc((mocknumb+1),sizeof(int));
  c=299792.458;         /*  km/s  */
  pi=3.14159265358979;
  H0=100;          /*   (km/s)/Mpc   */

  totalgroup = totalgroup_maker(g);
  totalmember = totalmember_maker(g);
```

```
for(h=0;h<=mocknumb;h+=1)
  {
    gra=(double *) calloc(totalgroup[h],sizeof(double));
    gdec=(double *) calloc(totalgroup[h],sizeof(double));
    gredshift=(double *) calloc(totalgroup[h],sizeof(double));
    gmr=(double *) calloc(totalgroup[h],sizeof(double));
    ggr=(double *) calloc(totalgroup[h],sizeof(double));
    rperp=(double *) calloc(totalgroup[h],sizeof(double));    /*rms*/
    gx=(double *) calloc(totalgroup[h],sizeof(double));
    gy=(double *) calloc(totalgroup[h],sizeof(double));
    gz=(double *) calloc(totalgroup[h],sizeof(double));
    mra=(double *) calloc(totalmember[h],sizeof(double));
    mdec=(double *) calloc(totalmember[h],sizeof(double));
    mredshift=(double *) calloc(totalmember[h],sizeof(double));
    mmr=(double *) calloc(totalmember[h],sizeof(double));
    mgr=(double *) calloc(totalmember[h],sizeof(double));
    mx=(double *) calloc(totalmember[h],sizeof(double));
    my=(double *) calloc(totalmember[h],sizeof(double));
    mz=(double *) calloc(totalmember[h],sizeof(double));
    theta=(double *) calloc(totalmember[h],sizeof(double));
    radius=(double *) calloc(totalmember[h],sizeof(double));
    normalradius=(double *) calloc(totalmember[h],sizeof(double));
    gid=(int *) calloc(totalgroup[h],sizeof(int));
    n=(int *) calloc(totalgroup[h],sizeof(int));
    mid=(int *) calloc(totalmember[h],sizeof(int));

    density=(double *) calloc(b,sizeof(double));
    densityred=(double *) calloc(b,sizeof(double));
    densityblue=(double *) calloc(b,sizeof(double));
    densitybright=(double *) calloc(b,sizeof(double));
    densitydim=(double *) calloc(b,sizeof(double));
    densityreddim=(double *) calloc(b,sizeof(double));
    densitybluedim=(double *) calloc(b,sizeof(double));
    densitybrightred=(double *) calloc(b,sizeof(double));
    densitydimred=(double *) calloc(b,sizeof(double));
    densityredbright=(double *) calloc(b,sizeof(double));
    densitybluebright=(double *) calloc(b,sizeof(double));
    densitybrightblue=(double *) calloc(b,sizeof(double));
    densitydimblue=(double *) calloc(b,sizeof(double));

    binmass=(int *) calloc(b,sizeof(int));
    binmassred=(int *) calloc(b,sizeof(int));
    binmassblue=(int *) calloc(b,sizeof(int));
    binmassbright=(int *) calloc(b,sizeof(int));
    binmassdim=(int *) calloc(b,sizeof(int));
    binmassreddim=(int *) calloc(b,sizeof(int));
    binmassbluedim=(int *) calloc(b,sizeof(int));
    binmassbrightred=(int *) calloc(b,sizeof(int));
    binmassdimred=(int *) calloc(b,sizeof(int));
    binmassredbright=(int *) calloc(b,sizeof(int));
    binmassbluebright=(int *) calloc(b,sizeof(int));
    binmassbrightblue=(int *) calloc(b,sizeof(int));
    binmassdimblue=(int *) calloc(b,sizeof(int));
```

```
      /*totalmassgal refers to number of galaxies; totalmassgr refers to number of
groups*/
      totalmassgal=0;
      totalmassgalred=0;
      totalmassgalblue=0;
      totalmassgalbright=0;
      totalmassgaldim=0;
      totalmassgr=0;
      totalmassgrred=0; /**NOTE:  totalmassgrred and blue are not used, since it would
only show that there are more red galaxies than blue galaxies**/
      totalmassgrblue=0;
      totalmassgrdim=0;
      totalmassgrbright=0;

      totalmassgrdimred=0;
      totalmassgrbrightred=0;
      totalmassgrdimblue=0;
      totalmassgrbrightblue=0;
      totalmassgrreddim=0;
      totalmassgrredbright=0;
      totalmassgrbluedim=0;
      totalmassgrbluebright=0;

      totalmassgalreddim=0;
      totalmassgalbluedim=0;
      totalmassgalbrightred=0;
      totalmassgaldimred=0;
      totalmassgalredbright=0;
      totalmassgalbluebright=0;
      totalmassgalbrightblue=0;
      totalmassgaldimblue=0;

      gals_not_counted_low=0;
      gals_not_counted_high=0;
      if(h<10){
    sprintf(groupname,"./mocks/mock0%d_Mr%d_groups.dat",h,g);
    sprintf(membername,"./mocks/mock0%d_Mr%d_members.dat",h,g);
      }
      if(h>=10){
    sprintf(groupname,"./mocks/mock%d_Mr%d_groups.dat",h,g);
    sprintf(membername,"./mocks/mock%d_Mr%d_members.dat",h,g);
      }
      groupdata=fopen(groupname,"r");
      memberdata=fopen(membername,"r");
      if(h==0)
    {
       sprintf(storename,"./Profiles/mocks_Mr%d_profile_%d_to_%d.dat",g,N1name,N2name);
       remove(storename);
       store=fopen(storename,"a");


sprintf(storenamered,"./Profiles/Mr%d_color_slope_profile_red_%d_to_%d.dat",g,N1name,N
```

```
2name);
	remove(storenamered);
	storered=fopen(storenamered,"a");

sprintf(storenameblue,"./Profiles/Mr%d_color_slope_profile_blue_%d_to_%d.dat",g,N1name
,N2name);
	remove(storenameblue);
	storeblue=fopen(storenameblue,"a");


sprintf(storenamebright,"./Profiles/Mr%d_profile_bright%4.1lf_%d_to_%d.dat",g,lcutname
,N1name,N2name);
	remove(storenamebright);
	storebright=fopen(storenamebright,"a");

sprintf(storenamedim,"./Profiles/Mr%d_profile_dim%4.1lf_%d_to_%d.dat",g,lcutname,N1nam
e,N2name);
	remove(storenamedim);
	storedim=fopen(storenamedim,"a");


sprintf(storenamereddim,"./Profiles/Mr%d_profile_red_dim%4.1lf_%d_to_%d.dat",g,lcutnam
e,N1name,N2name);
	remove(storenamereddim);
	storereddim=fopen(storenamereddim,"a");

sprintf(storenamebluedim,"./Profiles/Mr%d_profile_blue_dim%4.1lf_%d_to_%d.dat",g,lcutn
ame,N1name,N2name);
	remove(storenamebluedim);
	storebluedim=fopen(storenamebluedim,"a");


sprintf(storenamebrightred,"./Profiles/Mr%d_profile_bright%4.1lf_red_%d_to_%d.dat",g,l
cutname,N1name,N2name);
	remove(storenamebrightred);
	storebrightred=fopen(storenamebrightred,"a");

sprintf(storenamedimred,"./Profiles/Mr%d_profile_dim%4.1lf_red_%d_to_%d.dat",g,lcutnam
e,N1name,N2name);
	remove(storenamedimred);
	storedimred=fopen(storenamedimred,"a");


sprintf(storenameredbright,"./Profiles/Mr%d_profile_red_bright%4.1lf_%d_to_%d.dat",g,l
cutname,N1name,N2name);
	remove(storenameredbright);
	storeredbright=fopen(storenameredbright,"a");

sprintf(storenamebluebright,"./Profiles/Mr%d_profile_blue_bright%4.1lf_%d_to_%d.dat",g
,lcutname,N1name,N2name);
	remove(storenamebluebright);
	storebluebright=fopen(storenamebluebright,"a");
```

```
sprintf(storenamebrightblue,"./Profiles/Mr%d_profile_bright%4.1lf_blue_%d_to_%d.dat",g
,lcutname,N1name,N2name);
       remove(storenamebrightblue);
       storebrightblue=fopen(storenamebrightblue,"a");

sprintf(storenamedimblue,"./Profiles/Mr%d_profile_dim%4.1lf_blue_%d_to_%d.dat",g,lcutn
ame,N1name,N2name);
       remove(storenamedimblue);
       storedimblue=fopen(storenamedimblue,"a");
     }
      member_start=0; member_end=0;
      for(i=0;i<totalgroup[h];i+=1)
     {
       if(h==0)fscanf(groupdata,"%d %lf %lf %lf %d %lf %lf %lf %lf %lf %lf",
               &gid[i],
               &gra[i],
               &gdec[i],
               &gredshift[i],
               &n[i],&gmr[i],
               &ggr[i],
               &sigma,
               &rperp[i],
               &dummy,
               &gredge
               );
       if(h!=0)fscanf(groupdata,"%c %d %lf %lf %lf %d %lf %lf",
               &dummychar,
               &gid[i],
               &gra[i],
               &gdec[i],
               &gredshift[i],
               &n[i],
               &sigma,
               &gredge
               );
       gx[i]=cos(pi*gra[i]/180)*sin(pi*(90-gdec[i])/180);
       gy[i]=sin(pi*gra[i]/180)*sin(pi*(90-gdec[i])/180);
       gz[i]=cos(pi*(90-gdec[i])/180);
       if(n[i]>=N1 && n[i]<=N2)
         {totalmassgr+=1;}
       member_end+=n[i];
       for(j=member_start;j<member_end;j+=1)
         {
           if(h==0)fscanf(memberdata,"%d %lf %lf %lf %lf %lf %d %lf %lf",
                   &mid[j],
                   &mra[j],
                   &mdec[j],
                   &mredshift[j],
                   &mmr[j],
                   &mgr[j],
                   &fibcol,
                   &dummy,
```

```
                    &mredge
                    );
            if(h!=0)fscanf(memberdata,"%d %lf %lf %lf",
                    &mid[j],
                    &mra[j],
                    &mdec[j],
                    &mredshift[j]
                    );
 /*error check: double checks order*/if(mid[j]!=gid[i]){fprintf(stderr,"\nERROR:
mid=%d, gid=%d; mid!=gid\nABORTING\n\n",mid[j],gid[i]);return 0;}
            mx[j]=cos(pi*mra[j]/180)*sin(pi*(90-mdec[j])/180);
            my[j]=sin(pi*mra[j]/180)*sin(pi*(90-mdec[j])/180);
            mz[j]=cos(pi*(90-mdec[j])/180);
            normalradius[j] =
get_normalradius_BAD_RADIUS(gx[i],gy[i],gz[i],mx[j],my[j],mz[j],gredshift[i]);
            if(n[i]>=N1 && n[i]<=N2)
       {
         totalmassgal+=1;
         if(h==0)
           {
             if(mgr[j]>=get_gr_cut(mmr[j])) totalmassgalred+=1;
             if(mgr[j]<get_gr_cut(mmr[j]))  totalmassgalblue+=1;

             if(mmr[j]>=lcut) totalmassgaldim+=1;
             if(mmr[j]<lcut)  totalmassgalbright+=1;

             if(mgr[j]>=get_gr_cut(mmr[j]) && mmr[j]>=lcut) totalmassgalreddim+=1;
             if(mgr[j]<get_gr_cut(mmr[j]) && mmr[j]<lcut)  totalmassgalbluebright+=1;

             if(mgr[j]>=get_gr_cut(mmr[j]) && mmr[j]<lcut) totalmassgalredbright+=1;
             if(mgr[j]<get_gr_cut(mmr[j]) && mmr[j]>=lcut)  totalmassgalbluedim+=1;

         totalmassgaldimred=totalmassgalreddim;
         totalmassgaldimblue=totalmassgalbluedim;
         totalmassgalbrightred=totalmassgalredbright;
         totalmassgalbrightblue=totalmassgalbluebright;
           }
       }
         }
       member_start+=n[i];
     }

      member_start=0; member_end=0;
      for(i=0;i<totalgroup[h];i+=1)
     {
       member_end+=n[i];
       for(j=member_start;j<member_end;j+=1)
         {
           /*error check: double checks
order*/if(mid[j]!=gid[i]){fprintf(stderr,"\nERROR:  mid=%d, gid=%d;
mid!=gid\nABORTING\n\n",mid[j],gid[i]);return 0;}
           if(n[i]>=N1 && n[i]<=N2)
       {
```

```
        if(get_bin_number(lgdr,log10(normalradius[j]),lgminradius) >=0 &&
get_bin_number(lgdr,log10(normalradius[j]),lgminradius) < b)
           {
             binmass[get_bin_number(lgdr,log10(normalradius[j]),lgminradius)]+=1;
             if(h==0)
              {
                if(mgr[j]>=get_gr_cut(mmr[j]))
                  {

binmassred[get_bin_number(lgdr,log10(normalradius[j]),lgminradius)]+=1;
                   if(mmr[j]>=lcut)
                    {

binmassreddim[get_bin_number(lgdr,log10(normalradius[j]),lgminradius)]+=1;
                    }

                   if(mmr[j]<lcut)
                    {

binmassredbright[get_bin_number(lgdr,log10(normalradius[j]),lgminradius)]+=1;
                    }
                  }
                if(mgr[j]<get_gr_cut(mmr[j]))
                  {

binmassblue[get_bin_number(lgdr,log10(normalradius[j]),lgminradius)]+=1;
                   if(mmr[j]>=lcut)
                    {

binmassbluedim[get_bin_number(lgdr,log10(normalradius[j]),lgminradius)]+=1;
                    }

                   if(mmr[j]<lcut)
                    {

binmassbluebright[get_bin_number(lgdr,log10(normalradius[j]),lgminradius)]+=1;
                    }
                  }

                if(mmr[j]>=lcut)
                  {

binmassdim[get_bin_number(lgdr,log10(normalradius[j]),lgminradius)]+=1;
                   if(mgr[j]>=get_gr_cut(mmr[j]))
                    {

binmassdimred[get_bin_number(lgdr,log10(normalradius[j]),lgminradius)]+=1;
                    }
                   if(mgr[j]<get_gr_cut(mmr[j]))
                    {

binmassdimblue[get_bin_number(lgdr,log10(normalradius[j]),lgminradius)]+=1;
                    }
```

```
                  }

                if(mmr[j]<lcut)
                  {

binmassbright[get_bin_number(lgdr,log10(normalradius[j]),lgminradius)]+=1;
                    if(mgr[j]>=get_gr_cut(mmr[j]))
                     {

binmassbrightred[get_bin_number(lgdr,log10(normalradius[j]),lgminradius)]+=1;
                     }
                    if(mgr[j]<get_gr_cut(mmr[j]))
                     {

binmassbrightblue[get_bin_number(lgdr,log10(normalradius[j]),lgminradius)]+=1;
                     }
                  }
               }
            }
          if(get_bin_number(lgdr,log10(normalradius[j]),lgminradius)<0)
            {
              gals_not_counted_low+=1;
            }
          if(get_bin_number(lgdr,log10(normalradius[j]),lgminradius)>=b)
            {
              gals_not_counted_high+=1;
            }
        }
          }
        member_start+=n[i];
      }
        for(j=0;j<b;j+=1)
      {
        density[j]=get_density(binmass[j],lgdr,(lgminradius+j*lgdr),j);
        if(density[j]>0 && binmass[j]>=threshold) fprintf(store,"%10.7f %12.8e %12.8e
%12.8e %12.8e %12.8e %6d %6d %6d %3d %3d\n",
                                                pow(10,lgminradius+lgdr*(j+0.5)),
                                                density[j],
                                                density[j]/totalmassgal,
                                                density[j]/totalmassgr,

density[j]/(totalmassgal*sqrt(binmass[j])),
                                                density[j]/(totalmassgr*sqrt(binmass[j])),
                                                binmass[j],
                                                totalmassgal,
                                                totalmassgr,
                                                j,
                                                h
                                                );
        if(density[j]<=0 || binmass[j]<threshold) fprintf(store,"%10.7f %12.8e
0               0               0               0 %6d %6d %6d %3d %3d\n",
                                                pow(10,lgminradius+lgdr*(j+0.5)),
                                                density[j],
```

```
                                        binmass[j],
                                        totalmassgal,
                                        totalmassgr,
                                        j,
                                        h);

      if(h==0)
        {
          densityred[j]=get_density(binmassred[j],lgdr,(lgminradius+j*lgdr),j);
          densityblue[j]=get_density(binmassblue[j],lgdr,(lgminradius+j*lgdr),j);

          if(densityred[j]>0 && binmassred[j]>=threshold) fprintf(storered,"%10.7f
%12.8e %12.8e              0 %12.8e              0 %6d %6d %6d %3d %3d\n",
                                            pow(10,lgminradius+lgdr*(j+0.5)),
                                            densityred[j],
                                            densityred[j]/totalmassgalred,

densityred[j]/(totalmassgalred*sqrt(binmassred[j])),
                                            binmassred[j],
                                            totalmassgalred,
                                            totalmassgrred,
                                            j,
                                            h);
          if(densityred[j]<=0 || binmassred[j]<threshold) fprintf(storered,"%10.7f
%12.8e              0              0              0              0 %6d %6d %6d %3d
%3d\n",
                                            pow(10,lgminradius+lgdr*(j+0.5)),
                                            densityred[j],
                                            binmassred[j],
                                            totalmassgalred,
                                            totalmassgrred,
                                            j,
                                            h);
          if(densityblue[j]>0 && binmassblue[j]>=threshold) fprintf(storeblue,"%10.7f
%12.8e %12.8e              0 %12.8e              0 %6d %6d %6d %3d %3d\n",
                                            pow(10,lgminradius+lgdr*(j+0.5)),
                                            densityblue[j],
                                            densityblue[j]/totalmassgalblue,

    densityblue[j]/(totalmassgalblue*sqrt(binmassblue[j])),
                                            binmassblue[j],
                                            totalmassgalblue,
                                            totalmassgrblue,
                                            j,
                                            h);
          if(densityblue[j]<=0 || binmassblue[j]<threshold) fprintf(storeblue,"%10.7f
%12.8e              0              0              0              0 %6d %6d %6d %3d
%3d\n",
                                            pow(10,lgminradius+lgdr*(j+0.5)),
                                            densityblue[j],
                                            binmassblue[j],
                                            totalmassgalblue,
                                            totalmassgrblue,
```

```
                                                    j,
                                                    h);

          densitydim[j]=get_density(binmassdim[j],lgdr,(lgminradius+j*lgdr),j);
          densitybright[j]=get_density(binmassbright[j],lgdr,(lgminradius+j*lgdr),j);

          if(densitydim[j]>0 && binmassdim[j]>=threshold) fprintf(storedim,"%10.7f
%12.8e %12.8e              0 %12.8e              0 %6d %6d %6d %3d %3d\n",
                                                    pow(10,lgminradius+lgdr*(j+0.5)),
                                                    densitydim[j],
                                                    densitydim[j]/totalmassgaldim,

densitydim[j]/(totalmassgaldim*sqrt(binmassdim[j])),
                                                    binmassdim[j],
                                                    totalmassgaldim,
                                                    totalmassgrdim,
                                                    j,
                                                    h);
          if(densitydim[j]<=0 || binmassdim[j]<threshold) fprintf(storedim,"%10.7f
%12.8e              0              0              0              0 %6d %6d %6d %3d
%3d\n",
                                                    pow(10,lgminradius+lgdr*(j+0.5)),
                                                    densitydim[j],
                                                    binmassdim[j],
                                                    totalmassgaldim,
                                                    totalmassgrdim,
                                                    j,
                                                    h);
          if(densitybright[j]>0 && binmassbright[j]>=threshold)
fprintf(storebright,"%10.7f %12.8e %12.8e              0 %12.8e              0 %6d %6d
%6d %3d %3d\n",

pow(10,lgminradius+lgdr*(j+0.5)),
                                                        densitybright[j],

densitybright[j]/totalmassgalbright,

densitybright[j]/(totalmassgalbright*sqrt(binmassbright[j])),
                                                        binmassbright[j],
                                                        totalmassgalbright,
                                                        totalmassgrbright,
                                                        j,
                                                        h);
          if(densitybright[j]<=0 || binmassbright[j]<threshold)
fprintf(storebright,"%10.7f %12.8e              0              0              0
0 %6d %6d %6d %3d %3d\n",

pow(10,lgminradius+lgdr*(j+0.5)),
                                                        densitybright[j],
                                                        binmassbright[j],
                                                        totalmassgalbright,
                                                        totalmassgrbright,
                                                        j,
```

```
                                                        h);

          densitydimred[j]=get_density(binmassdimred[j],lgdr,(lgminradius+j*lgdr),j);

densitydimblue[j]=get_density(binmassdimblue[j],lgdr,(lgminradius+j*lgdr),j);
          if(densitydimred[j]>0 && binmassdimred[j]>=threshold) {
                fprintf(storedimred,"%10.7f %12.8e %12.8e             0 %12.8e
0 %6d %6d %6d %3d %3d\n",
                        pow(10,lgminradius+lgdr*(j+0.5)),
                        densitydimred[j],
                        densitydimred[j]/totalmassgaldimred,
                        densitydimred[j]/(totalmassgaldimred*sqrt(binmassdimred[j])),
                        binmassdimred[j],
                        totalmassgaldimred,
                        totalmassgrdimred,
                        j,
                        h);
          }
          if(densitydimred[j]<=0 || binmassdimred[j]<threshold)
      {
        fprintf(storedimred,"%10.7f %12.8e             0             0
0              0 %6d %6d %6d %3d %3d\n",
              pow(10,lgminradius+lgdr*(j+0.5)),
              densitydimred[j],
              binmassdimred[j],
              totalmassgaldimred,
              totalmassgrdimred,
              j,
              h);
      }
          if(densitydimblue[j]>0 && binmassdimblue[j]>=threshold)
fprintf(storedimblue,"%10.7f %12.8e %12.8e             0 %12.8e             0 %6d
%6d %6d %3d %3d\n",

pow(10,lgminradius+lgdr*(j+0.5)),
                                                        densitydimblue[j],

densitydimblue[j]/totalmassgaldimblue,

densitydimblue[j]/(totalmassgaldimblue*sqrt(binmassdimblue[j])),
                                                        binmassdimblue[j],
                                                        totalmassgaldimblue,
                                                        totalmassgrdimblue,
                                                        j,
                                                        h);
          if(densitydimblue[j]<=0 || binmassdimblue[j]<threshold)
fprintf(storedimblue,"%10.7f %12.8e             0             0             0
0 %6d %6d %6d %3d %3d\n",

pow(10,lgminradius+lgdr*(j+0.5)),
                                                        densitydimblue[j],
                                                        binmassdimblue[j],
                                                        totalmassgaldimblue,
```

```
                                                        totalmassgrdimblue,
                                                        j,
                                                        h);

densitybrightred[j]=get_density(binmassbrightred[j],lgdr,(lgminradius+j*lgdr),j);

densitybrightblue[j]=get_density(binmassbrightblue[j],lgdr,(lgminradius+j*lgdr),j);
          if(densitybrightred[j]>0 && binmassbrightred[j]>=threshold)
fprintf(storebrightred,"%10.7f %12.8e %12.8e              0 %12.8e              0 %6d
%6d %6d %3d %3d\n",

pow(10,lgminradius+lgdr*(j+0.5)),
                                                          densitybrightred[j],

densitybrightred[j]/totalmassgalbrightred,

densitybrightred[j]/(totalmassgalbrightred*sqrt(binmassbrightred[j])),
                                                          binmassbrightred[j],
                                                          totalmassgalbrightred,
                                                          totalmassgrbrightred,
                                                          j,
                                                          h);
          if(densitybrightred[j]<=0 || binmassbrightred[j]<threshold)
fprintf(storebrightred,"%10.7f %12.8e              0              0              0
0 %6d %6d %6d %3d %3d\n",

pow(10,lgminradius+lgdr*(j+0.5)),
                                                          densitybrightred[j],
                                                          binmassbrightred[j],
                                                          totalmassgalbrightred,
                                                          totalmassgrbrightred,
                                                          j,
                                                          h);
          if(densitybrightblue[j]>0 && binmassbrightblue[j]>=threshold)
fprintf(storebrightblue,"%10.7f %12.8e %12.8e              0 %12.8e              0 %6d
%6d %6d %3d %3d\n",

pow(10,lgminradius+lgdr*(j+0.5)),
                                                            densitybrightblue[j],

densitybrightblue[j]/totalmassgalbrightblue,

densitybrightblue[j]/(totalmassgalbrightblue*sqrt(binmassbrightblue[j])),
                                                            binmassbrightblue[j],

totalmassgalbrightblue,
                                                            totalmassgrbrightblue,
                                                            j,
                                                            h);
          if(densitybrightblue[j]<=0 || binmassbrightblue[j]<threshold)
fprintf(storebrightblue,"%10.7f %12.8e              0              0              0
0 %6d %6d %6d %3d %3d\n",
```

```
pow(10,lgminradius+lgdr*(j+0.5)),
                                                densitybrightblue[j],
                                                binmassbrightblue[j],

totalmassgalbrightblue,
                                                totalmassgrbrightblue,
                                                j,
                                                h);

          densityreddim[j]=get_density(binmassreddim[j],lgdr,(lgminradius+j*lgdr),j);

densitybluedim[j]=get_density(binmassbluedim[j],lgdr,(lgminradius+j*lgdr),j);
          if(densityreddim[j]>0 && binmassreddim[j]>=threshold)
fprintf(storereddim,"%10.7f %12.8e %12.8e              0 %12.8e              0 %6d %6d
%6d %3d %3d\n",

pow(10,lgminradius+lgdr*(j+0.5)),
                                                densityreddim[j],

densityreddim[j]/totalmassgalreddim,

densityreddim[j]/(totalmassgalreddim*sqrt(binmassreddim[j])),
                                                binmassreddim[j],
                                                totalmassgalreddim,
                                                totalmassgrreddim,
                                                j,
                                                h);
          if(densityreddim[j]<=0 || binmassreddim[j]<threshold)
fprintf(storereddim,"%10.7f %12.8e              0              0              0
0 %6d %6d %6d %3d %3d\n",

pow(10,lgminradius+lgdr*(j+0.5)),
                                                densityreddim[j],
                                                binmassreddim[j],
                                                totalmassgalreddim,
                                                totalmassgrreddim,
                                                j,
                                                h);
          if(densitybluedim[j]>0 && binmassbluedim[j]>=threshold)
fprintf(storebluedim,"%10.7f %12.8e %12.8e              0 %12.8e              0 %6d
%6d %6d %3d %3d\n",

pow(10,lgminradius+lgdr*(j+0.5)),
                                                  densitybluedim[j],

densitybluedim[j]/totalmassgalbluedim,

densitybluedim[j]/(totalmassgalbluedim*sqrt(binmassbluedim[j])),
                                                  binmassbluedim[j],
                                                  totalmassgalbluedim,
                                                  totalmassgrbluedim,
                                                  j,
                                                  h);
```

```
          if(densitybluedim[j]<=0 || binmassbluedim[j]<threshold)
fprintf(storebluedim,"%10.7f %12.8e              0              0              0
0 %6d %6d %6d %3d %3d\n",

pow(10,lgminradius+lgdr*(j+0.5)),
                                                    densitybluedim[j],
                                                    binmassbluedim[j],
                                                    totalmassgalbluedim,
                                                    totalmassgrbluedim,
                                                    j,
                                                    h);


densityredbright[j]=get_density(binmassredbright[j],lgdr,(lgminradius+j*lgdr),j);

densitybluebright[j]=get_density(binmassbluebright[j],lgdr,(lgminradius+j*lgdr),j);

          if(densityredbright[j]>0 && binmassredbright[j]>=threshold)
fprintf(storeredbright,"%10.7f %12.8e %12.8e              0 %12.8e              0 %6d
%6d %6d %3d %3d\n",

pow(10,lgminradius+lgdr*(j+0.5)),
                                                       densityredbright[j],

densityredbright[j]/totalmassgalredbright,

densityredbright[j]/(totalmassgalredbright*sqrt(binmassredbright[j])),
                                                       binmassredbright[j],
                                                       totalmassgalredbright,
                                                       totalmassgrredbright,
                                                       j,
                                                       h);
          if(densityredbright[j]<=0 || binmassredbright[j]<threshold)
fprintf(storeredbright,"%10.7f %12.8e              0              0              0
0 %6d %6d %6d %3d %3d\n",

pow(10,lgminradius+lgdr*(j+0.5)),
                                                       densityredbright[j],
                                                       binmassredbright[j],
                                                       totalmassgalredbright,
                                                       totalmassgrredbright,
                                                       j,
                                                       h);
          if(densitybluebright[j]>0 && binmassbluebright[j]>=threshold)
fprintf(storebluebright,"%10.7f %12.8e %12.8e              0 %12.8e              0 %6d
%6d %6d %3d %3d\n",

pow(10,lgminradius+lgdr*(j+0.5)),
                                                         densitybluebright[j],

densitybluebright[j]/totalmassgalbluebright,

densitybluebright[j]/(totalmassgalbluebright*sqrt(binmassbluebright[j])),
```

```
                                                                binmassbluebright[j],
totalmassgalbluebright,
                                                                totalmassgrbluebright,
                                                                j,
                                                                h);
          if(densitybluebright[j]<=0 || binmassbluebright[j]<threshold)
fprintf(storebluebright,"%10.7f %12.8e              0              0              0
0 %6d %6d %6d %3d %3d\n",
pow(10,lgminradius+lgdr*(j+0.5)),
                                                                densitybluebright[j],
                                                                binmassbluebright[j],
totalmassgalbluebright,
                                                                totalmassgrbluebright,
                                                                j,
                                                                h);
        }
      if(verbose !=0)
        {
          fprintf(stderr,"       %10.7f %12.8e %12.8e %12.8e %12.8e %12.8e %6d %6d %6d
%3d %3d\n",
            pow(10,lgminradius+lgdr*(j+0.5)),
            density[j],
            density[j]/totalmassgal,
            density[j]/totalmassgr,
            density[j]/(totalmassgal*sqrt(binmass[j])),
            density[j]/(totalmassgr*sqrt(binmass[j])),
            binmass[j],
            totalmassgal,
            totalmassgr,
            j,
            h);
          if(h==0)
       {
          fprintf(stderr,"red    %10.7f %12.8e %12.8e              0 %12.8e
0 %6d %6d %6d %3d %3d\n",
               pow(10,lgminradius+lgdr*(j+0.5)),
               densityred[j],
               densityred[j]/totalmassgalred,
               densityred[j]/(totalmassgalred*sqrt(binmassred[j])),
               binmassred[j],
               totalmassgalred,
               totalmassgrred,
               j,
               h);
          fprintf(stderr,"blue   %10.7f %12.8e %12.8e              0 %12.8e
0 %6d %6d %6d %3d %3d\n",
               pow(10,lgminradius+lgdr*(j+0.5)),
               densityblue[j],
               densityblue[j]/totalmassgalblue,
               densityblue[j]/(totalmassgalblue*sqrt(binmassblue[j])),
```

```
                binmassblue[j],
                totalmassgalblue,
                totalmassgrblue,
                j,
                h);

        fprintf(stderr,"dim    %10.7f %12.8e %12.8e              0 %12.8e
0 %6d %6d %6d %3d %3d\n",
                pow(10,lgminradius+lgdr*(j+0.5)),
                densitydim[j],
                densitydim[j]/totalmassgaldim,
                densitydim[j]/(totalmassgaldim*sqrt(binmassdim[j])),
                binmassdim[j],
                totalmassgaldim,
                totalmassgrdim,
                j,
                h);
        fprintf(stderr,"bright %10.7f %12.8e %12.8e              0 %12.8e
0 %6d %6d %6d %3d %3d\n",
                pow(10,lgminradius+lgdr*(j+0.5)),
                densitybright[j],
                densitybright[j]/totalmassgalbright,
                densitybright[j]/(totalmassgalbright*sqrt(binmassbright[j])),
                binmassbright[j],
                totalmassgalbright,
                totalmassgrbright,
                j,
                h);
      }
        }
    }
      free(gra);
      free(gdec);
      free(gredshift);
      free(gmr);
      free(ggr);
      free(rperp);
      free(gx);
      free(gy);
      free(gz);
      free(mra);
      free(mdec);
      free(mredshift);
      free(mmr);
      free(mgr);
      free(mx);
      free(my);
      free(mz);
      free(theta);
      free(radius);
      free(normalradius);
      free(gid);
      free(n);
```

```
      free(mid);
      free(density);
      free(densityred);
      free(densityblue);
      free(densitydim);
      free(densitybright);
      free(densityreddim);
      free(densitybluedim);
      free(densitydimred);
      free(densitybrightred);
      free(densityredbright);
      free(densitybluebright);
      free(densitydimblue);
      free(densitybrightblue);
      free(binmass);
      free(binmassred);
      free(binmassblue);
      free(binmassdim);
      free(binmassbright);
      free(binmassreddim);
      free(binmassbluedim);
      free(binmassdimred);
      free(binmassbrightred);
      free(binmassredbright);
      free(binmassbluebright);
      free(binmassdimblue);
      free(binmassbrightblue);

      if(verbose !=0)
    {
      fprintf(stderr,"   ~~~~~ Mr%d %d-%d mock%d done: %d galaxies radii too small, %d
galaxies radii too large
~~~~~\n",g,N1name,N2name,h,gals_not_counted_low,gals_not_counted_high);
    }
      else
    {
      fprintf(stderr,"...%d",h);
    }
    }
  fprintf(stderr,"\nMr%d; N1=%d; N2=%d; bins=%d; lcut=%lf\n",g,N1name,N2name,b,lcut);
  fprintf(stderr,"COMPLETE\nFiles stored
at\n%s\n%s\n%s\n%s\n%s\n%s\n%s\n%s\n%s\n%s\n%s\n%s\n",storename,storenamered,store
nameblue,storenamedim,storenamebright,storenamereddim,storenamebluedim,storenamedimred
,storenamebrightred,storenameredbright,storenamebluebright,storenamedimblue,storenameb
rightblue);
  return 0;
}
```

Acknowledgements

I would like to thank some people who have made this research possible or have helped me during my time as an undergraduate physics student. Dr. Andreas Berlind, for always being an immensely helpful advisor. Dr. Brau, Dr. Holley-Bockelmann, and Dr. Weintraub, for agreeing to be on my honors thesis committee. The entire Vanderbilt physics department., for teaching me so much. My undergraduate peers and friends within the physics department, for spending countless nights with me poring over equations. Last, but most certainly not least, my supportive parents, for piquing my scientific curiosity in the first place.