[1] Draft version May 9, 2019

Corresponding author: Felipe Leonardo Gómez-Cortés
fl.gomez10@uniandes.edu.co

Typeset using LaTeX **preprint** style in AASTeX62

A Large Scale Structure Void Identifier for Galaxy Surveys Based on the β-Skeleton Graph Method

FELIPE LEONARDO GÓMEZ-CORTÉS,[1] JAIME E. FORERO-ROMERO,[1] AND XIAO-DONG LI[2]

[1] *Physics Department, Universidad de Los Andes*
[2] *Shcool of Physics, Korean Institute for Advanced Study*

ABSTRACT

This goes at the end

1. THE ALGORITHM

The main goal of this project is to identify voids in the large scale structure of the universe, using the nobel method in astrophysics: the β-Skeleton graph. In order to develop the code, a toy model catalog structure is used, then a second catalog of random points populates the same volume, the graph is calculated and then the voids are identified as the second catalog points without connections with the first catalog points.

1.1. *Random Points in Empty Regions*

We start with a sample of N points randomly placed inside a cubic box of lenght L. (Figure 1. In the middle of the box there is a spherical void, i.e. an spherical region without points. This set of points is called "the observed catalog" (**OC**). This toy model will represent -in a gross aproximation- the LSS with galaxies and voids.

Then another set of N random points populates the whole box, without restrictions. This set is called "the random catalog" (**RC**). It will populate even the void region. It's only necesary to know the volume of the OC and the number of points N to create the RC.

The Li's Skeleton library can operate over a single set of points. This set is called "the full catalog" (**FC**). This set of points is created using the `vstack` python function. The first N elements are the RC points, the last N elements are the OC points, then the FC has 2N elements.

The 1-Skeleton graph is calculated over the FC. Runing the code over a FC of $\sim 10^4$ points in a Core i5 (2nd gen.) Linux machine, it takes 192 seconds to complete the calculus. The result is stored as the Full Catalog Beta Skeleton Graph (**fcBSk**).

The fcBSk is chopped to the half, we are interested about the RC points that are not connected with the OC points. Then a droplist is created as the subset of connections where the second point belongs to the OC, i.e., its index is greater than N.

Using the python `set` class, its easy to find the complement of the droplist in the fsBSk. After droping the RC points connected with the OC, it remains the points inside the void. Those points are labelled as **true void points**.

There are some points that belong to the void but they have connections in the β-Skeleton with some observational points. They are identified as frontier points.

The algorithm sucessfully recognized the spherical void, and also identified some other underdense regions. (Fig. 2).

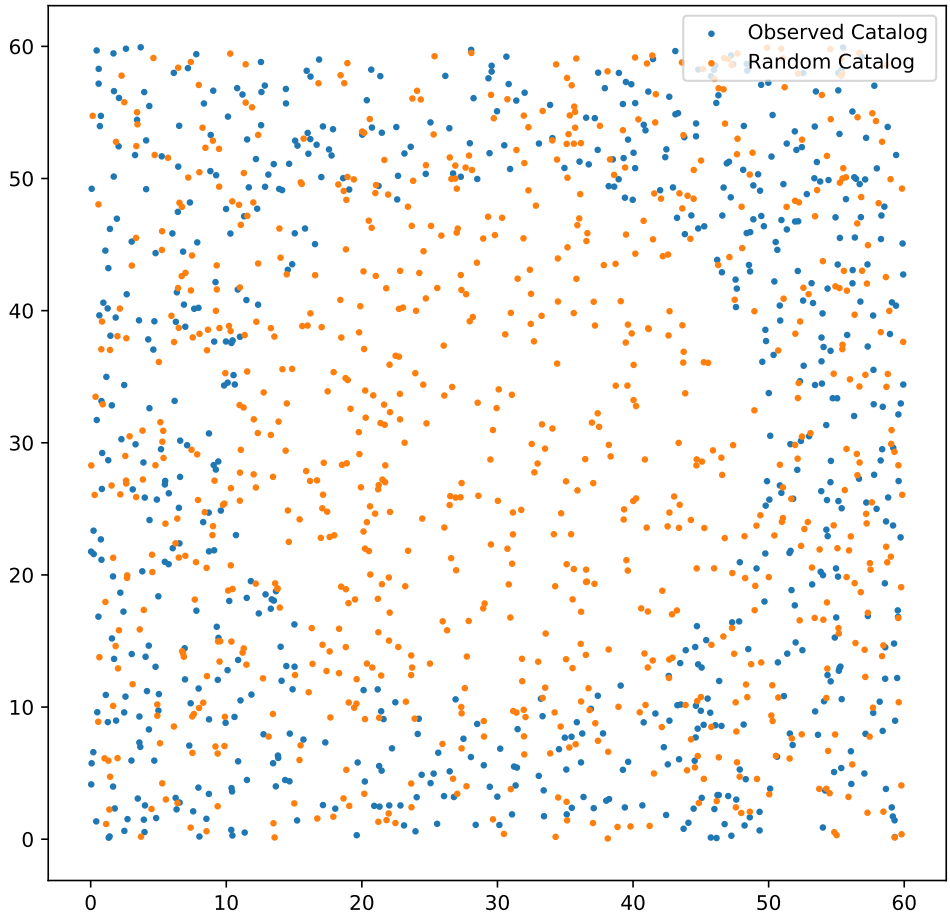


Figure 1. The Algorithm uses two sets of points: The Observed Catalog (LSS like in blue with a spherical void in the middle) and the random Catalog (orange). This is a slice of the Toy Model Catalog to test the Algorithm; a cubic box with an random uniform distribution of points over the box volume with a spherical void at the center.

A second test was designed with four spherical voids in different locations, showing how well defined where the two bigger voids, but there is a lower resolution limit related to the void size and the lenght of the Random Points Catalog. (Fig. 3).

1.2. *Tagging Voids*

Once true void points are identified, the β-Skeleton Graph is used again to identify their neighbours. The set of True Void Points connected between them, together with the frontier points is tagged as a void.

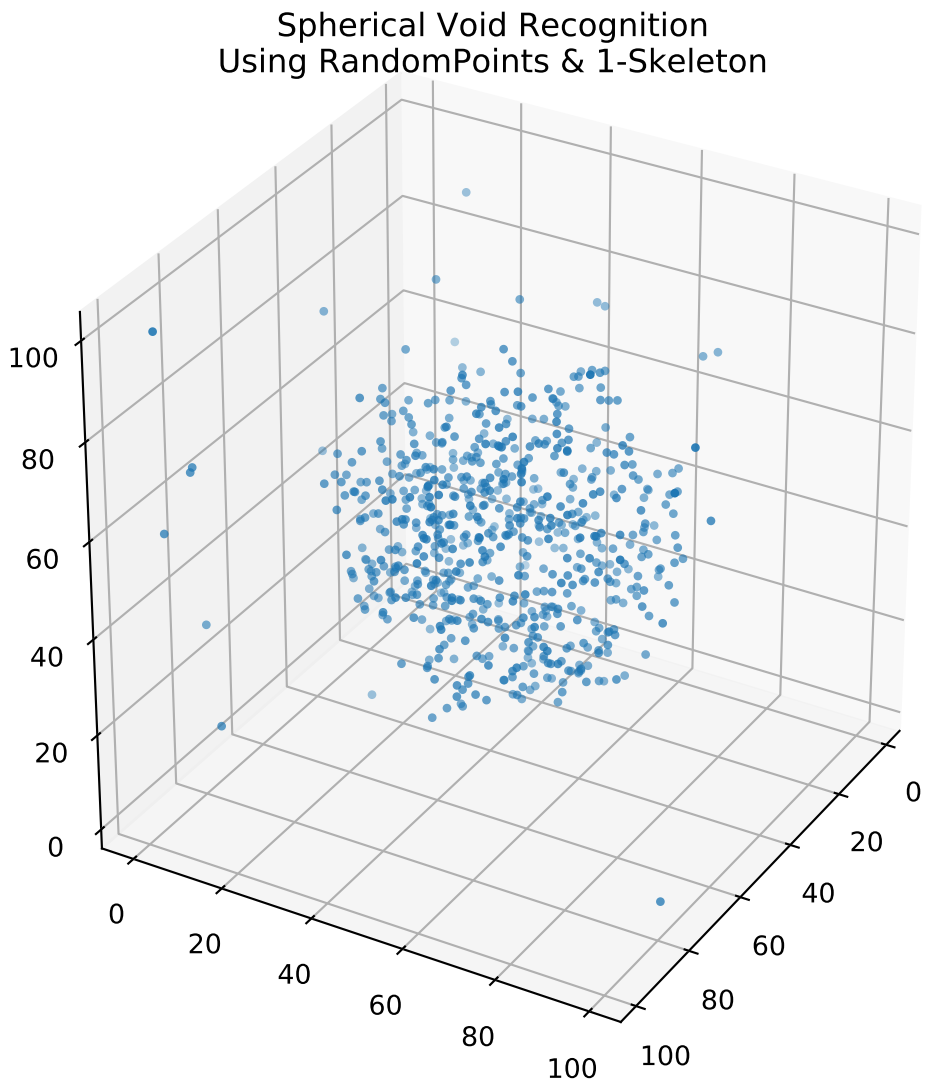


Figure 2. Points identified as "Void Points". They are points from the Random Catalog connected by the 1-Skeleton only to other Random Catalog points. N=5000 points, BoxLength = 100 units and void radius R=37 units.

From the Full Catalog Beta-Skeleton Graph (**fcBSk**), that is just a $M \times 2$ dimentional array of indexes, we can extract the connections of void points. Each void point must be conected solely with other points from the RC. They may be other void points or frontier points.

Frontier points are poins from the RC than are connected to RC points as well to OC points. Frontier points where rejected on early stages of the algorithm, but now they are taken into account. Frontier points are over the "Void Surface". A similar surface was the aiming of the previous algorithm (comparing two different β-Skeletons).

We did not achieve the stage of exploring voids with sparse galaxies in the previous algorithm, but now, a void with sparse galaxies is easily identified as a "plum pudding", where the void is the pudding with frontier points enclosing the "plums" (sparse galaxies),

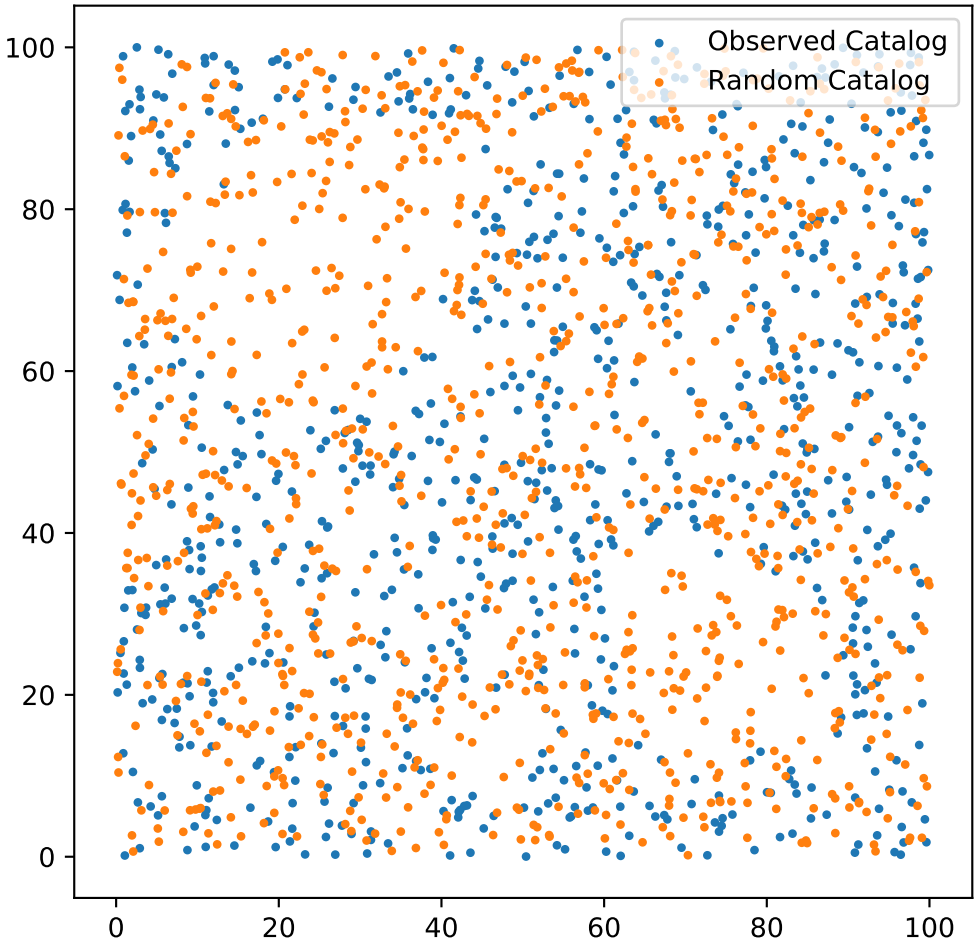


Figure 3. Slice at z=70 of an structure with four spherical voids of radii 5, 10, 5 and 20 units. N=5000, BoxLength = 100, the voids radius where selected as R= 5, 10, 15 and 20.

The algorithm starts looking for the connections of the first Void Point in the Random Catalog. A list including the first void point and their neighbohrs is created. This list is going to be the first voids tagged. Then the second point is checked, if it appears in the list the the second point and its neighbohrs are added to the previous list, they are identified as part of the first void. If not, then a new list is created with the second point and their neighbohrs, giving place to the second void. When the n-th is reached, a search is performed over the existent lists of voids. The n-th point may belong to a previous lists of voids, or may create its own list for a m-th void ($m \leq n$).

The master list of catalogs at this moment is incomplete, all the Void Points in the list are counted and checked if they belong to another list. But frontier points are not checked. The master lists contains voids sharing frontiers. Is necesary to run again a recount, looking for repetitions between lists, thinking each list as a set of points, and looking for intersection between sets. (Python offers

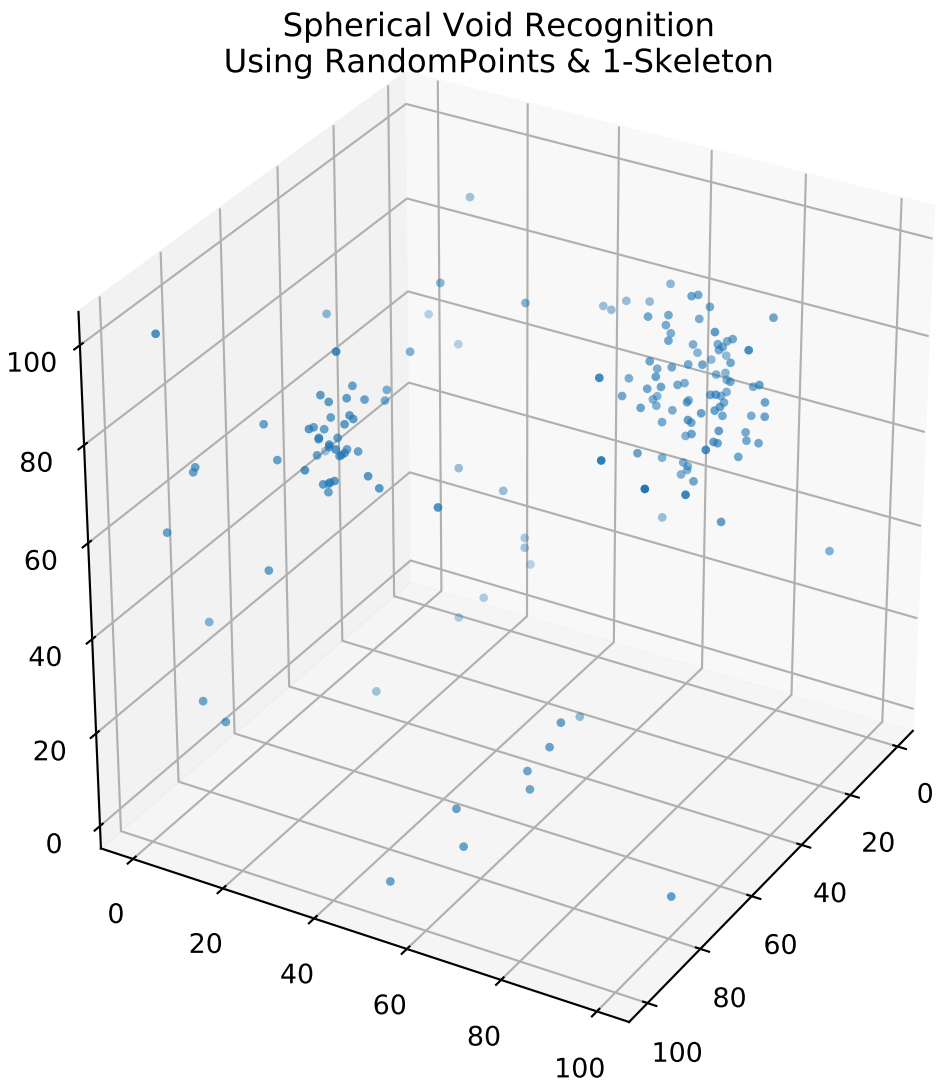


Figure 4. Points identified as "Void Points". This tests aims to check the resolution of the algorithm. Some scattered micro-voids appear. The "biggest" voids are recognized (R=15,20), but small structures doesn't appear (R=5,10). N=5000, BoxLength = 100, the voids radius where selected as R= 5, 10, 15 and 20.

this implementation using sets by invoking the class `set`). When a intersection between voids is detected, the two voids are merged into a new one. Finally, the voids are tagged (by the ID of the first void particle).

This algorithm was able to detect irregular voids made of intersecting ellipsoids. This test was performed over a artificial structure, a toy model catalog.

2. FINDING VOIDS IN SIMULATIONS

2.1. *Abacus-Cosmos Simulation*

From a snapshot of the Abacus-Cosmos simulation with fiducial cosmology, a set of halos is selected, within a sphere of radius $R = 100$Mpc/h and a M_{cut} given to get $\sim 10^4$ halos.

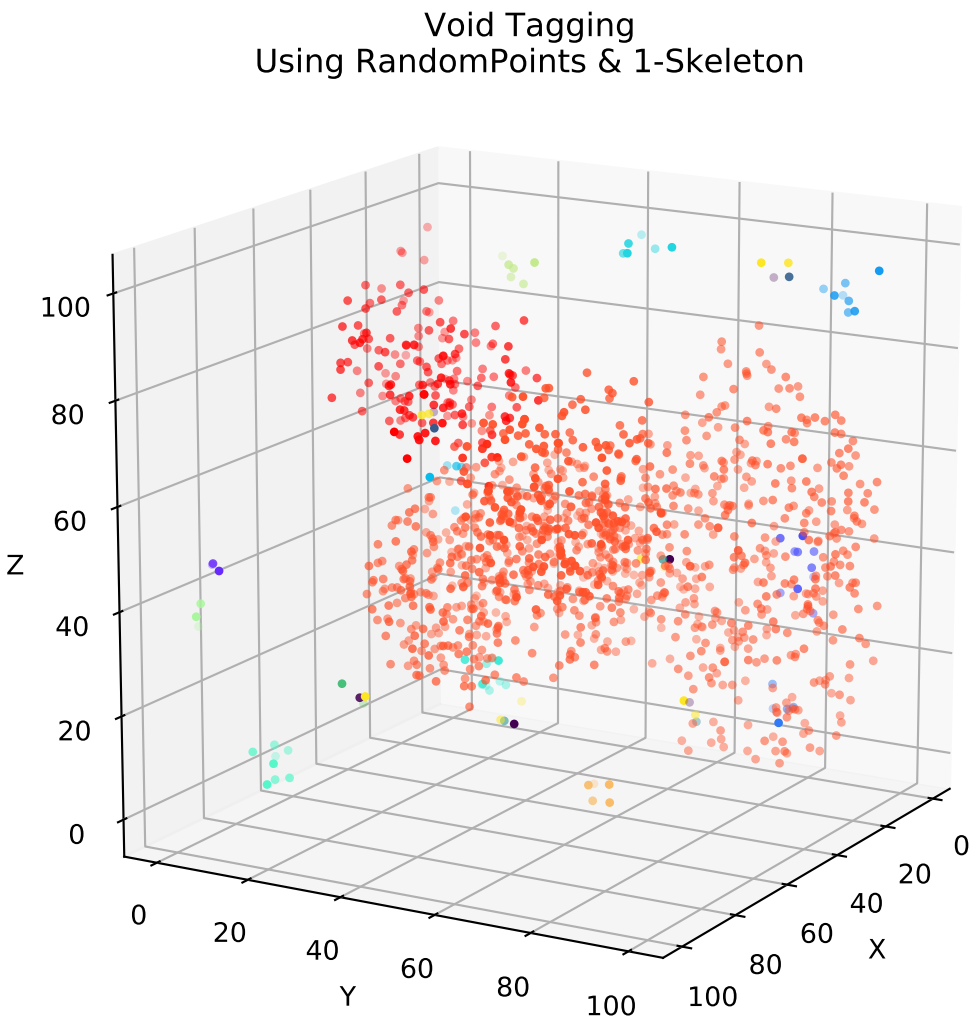
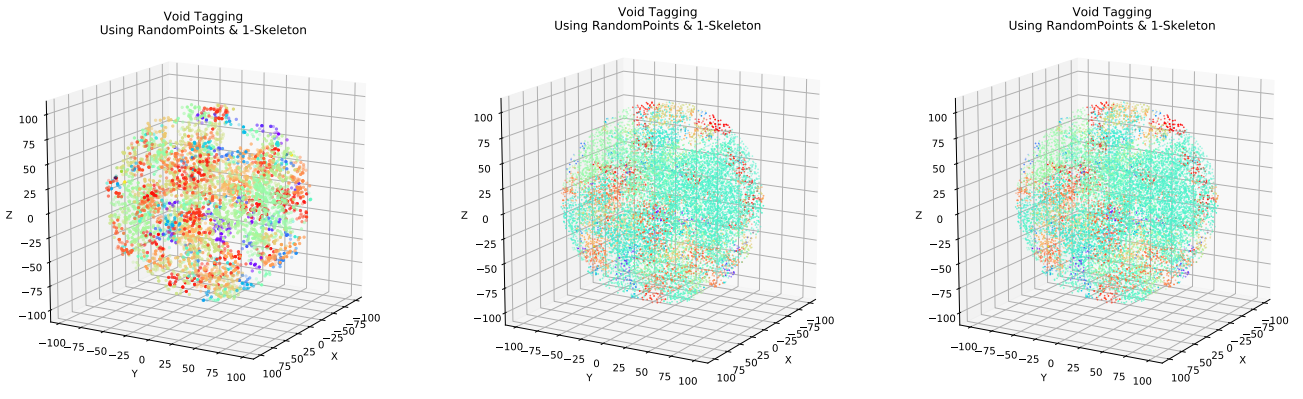


Figure 5. Tagging Voids. This algorithm can indentify irregular structures, a step ahead the previous algorithm. The irregular voids are made of overlapping ellipsoidal voids, with semiaxes proportion 1:0.7:0.5 as the literature suggets. N=5000, BoxLength = 100, the voids major semiaxes where chosen as R= 40, 30, 30 and 40.

The Random Catalog is generated, taking care of having a uniform space distribution. A common error is to generate the radius using a uniform distribution from 0 to 1 and scaling by a factor R, but must be taken instead the cubic root of the randon number between 0 and 1, then multiplied by R. Another error is generating the angles again, taking a random number between zero and one, then scaling by 2π. This error can be avoided by generating a randon number between 0 and 1 as the cosine of θ, then calculating the angle as the arc-cosine. The unitary vector can be calculated using a gaussian distribution for x, y and z (centered at zero with the same deviation for them three). The norm is calculated as usually, by taking the square root of the sum of the coordinates, then dividing each coordinate by the norm of the vector.



(a) (b) (c)

Figure 6. Tagging Abacus-Cosmos Voids. This is the first test over "real" data, not a toy model. The LSS shows highly dense packed halos, where voids have almost none of them. The idea is to increase the ratio of Random Points over Catalog points to study (in later weeks) how void detection can be affected. (a) Ratio 1:1, (b) ratio 2:1 and (c) ratio 3:1. The catalog is a spherical set of points, with radius 100Mpc/h, N halos = 9981, N random points = 10000 (a), 2N (b) and 3N (c).

The algorithm ran over the "real" dataset using the rations 1:1, 2:1 and 3:1 for the number of random points vs. catalog points.

A kind of percolation phenomenon is apreciated at figure 6. Having a similar number of Random Points and Observational Points (a) conducts to sucessfull void identification, while a higher ration 2:1 (b) and 3:1 (c)

2.2. *Running over 64 cuts*

2.3. *Running over Corner Plots*

3. FINDING VOIDS IN GALAXY REDSHINF SURVEYS

3.1. *SDSS Planck 2015 data*

The SDSS has ~ 57000 galaxies in the truncated cone section from RA 0 to 50 deg, and DEC from -40 deg to $+40$ deg. Using Planck-2015 cosmology, reshift was converted into the range of distance from 0 to 300 Mpc. (figure 7).

The void size density function is calculated as the probability to find a void of a given radius in to the total volume. There is a discrepancy between our density function and Platen (2008) results, maybe they did not normalize by the volume. Figure 8.

The void ellipticity is calculated as $\epsilon = 1 - c/a$, with a the biggest semi-axe, and c the smallest semi-axe. They two graphs seem to have similar values. The probability density is not normalized by the volume. Figure 9.

The last comparison was made using the two axis ratios to find how prolate, oblate or close to spheroid. Our results are pretty close to Platen results (2008).

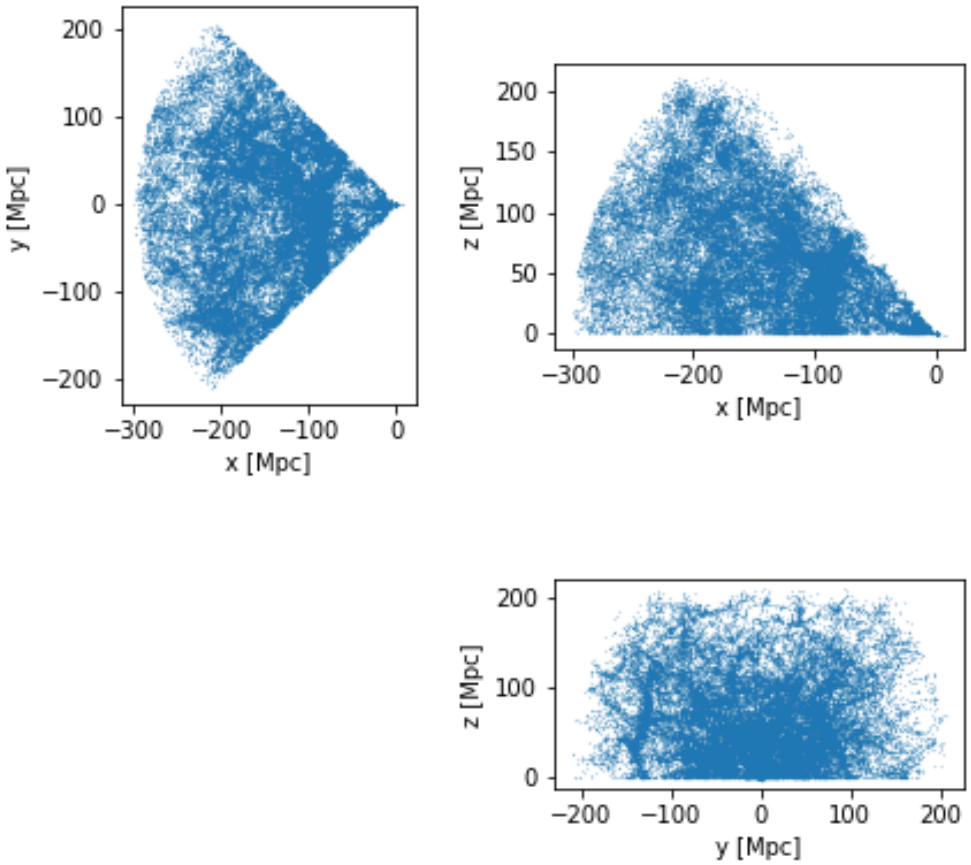


Figure 7. Dataset from SDSS. Comoving distances are in Mpc, calculated from the redshift measurements using the module "cosmology" from the library "astrpy" with Planck15 cosmological parameters.

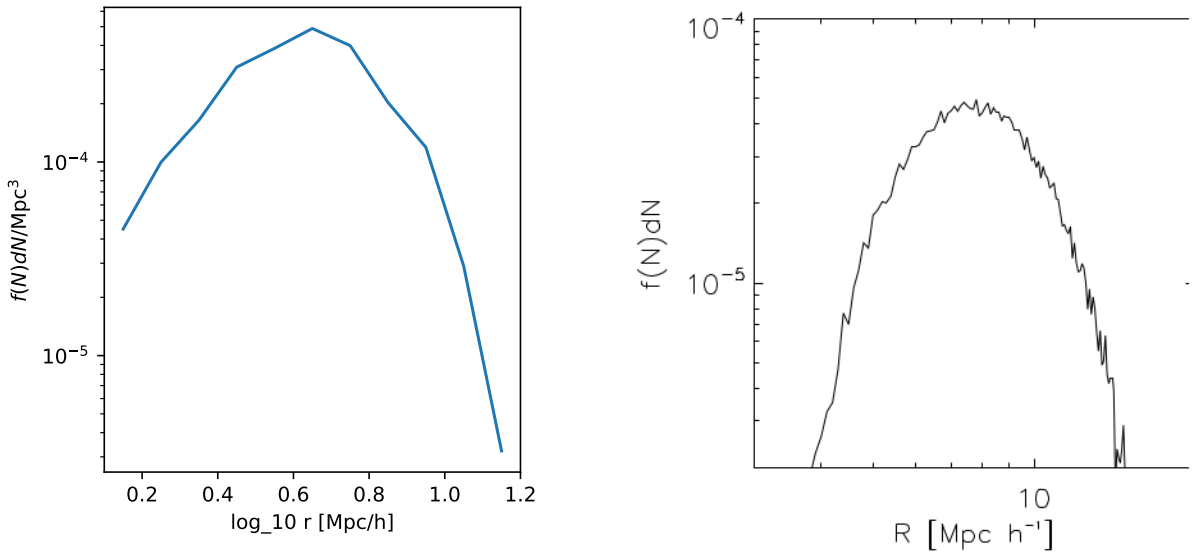


Figure 8. Void size density function. Left, our analysis. Right, Platen et al (2008).

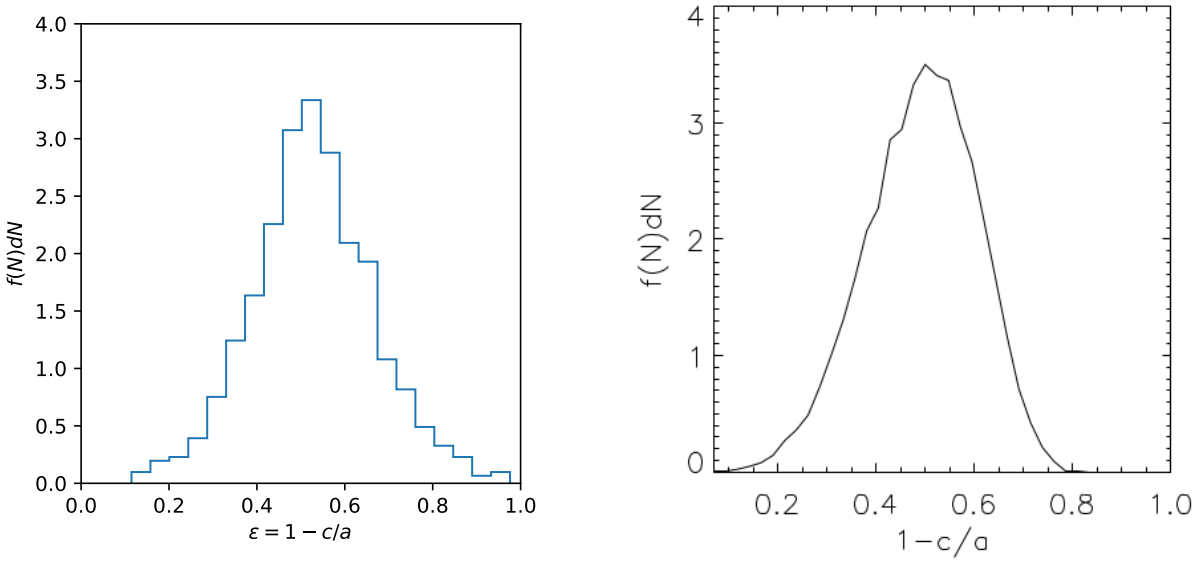


Figure 9. Void Ellipticity. Left, analysis of voids found with our algorithm. Rigth, Platen et al (2008).

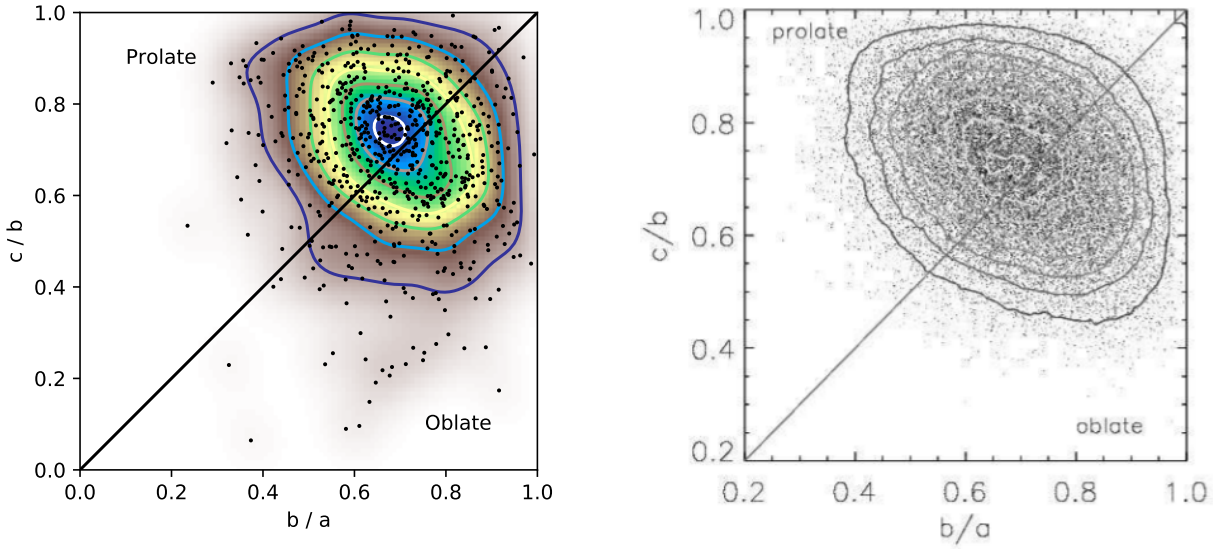


Figure 10. Scatter diagram of two axis ratios. In the approximation of the voids as ellipsoidal forms, the semi-axes are labeled, from shorter to larger, as a, b and c. A perfect sphericall void should be placed at the rigth top corner of the diagram.