Manuscript Details

Manuscript number	ASCOM_2017_95
Title	correlcalc: A 'Generic' Recipe for Calculation of Two-point Correlation Function
Article type	Full Length Article

Abstract

This article provides a method for quick computation of galaxy two-point correlation function(2pCF) from redshift surveys using python. One of the salient features of this approach is that it can be used for calculating galaxy clustering for any arbitrary geometry (or Cosmology) model. Being efficient enough to run fast on a low-spec desktop computer, this `recipe' can be used for quick validation of alternative models and for pedagogical purposes.

Keywords	Two-point correlation function; BallTree; Galaxy Clustering ; redshift surveys ; 2pCF for alternative models
Manuscript category	General computational techniques used for astronomy
Corresponding Author	Yeluripati Rohin
Corresponding Author's Institution	Department of Physics & Astrophysics, University of Delhi
Order of Authors	Yeluripati Rohin

Submission Files Included in this PDF

File Name [File Type]

System Generated Manuscript.pdf [Manuscript File]

Submission Files Not Included in this PDF

File Name [File Type]

astrocompv1.zip [LaTeX Source File]

To view all the submission files, including those not included in the PDF, click on the manuscript title on your EVISE Homepage, then click 'Download zip file'.

correlcalc: A 'Generic' Recipe for Calculation of Two-point Correlation Function

Yeluripati Rohin[1]

Department of Physics & Astrophysics, University of Delhi, Delhi

Abstract

This article provides a method for quick computation of galaxy two-point correlation function(2pCF) from redshift surveys using `python`. One of the salient features of this approach is that it can be used for calculating galaxy clustering for any arbitrary geometry (or Cosmology) model. Being efficient enough to run fast on a low-spec desktop computer, this 'recipe' can be used for quick validation of alternative models and for pedagogical purposes.

Keywords: Two-point correlation function, `BallTree`, Galaxy Clustering, redshift surveys, 2pCF for alternative models

1. Introduction

The two-point correlation function (2pCF from here on) is one of the vital statistics that can be obtained from the galaxy redshift surveys. 2pCF provides information about the galaxy clustering in redshift space and in turn provides crucial observables such as Baryon Acoustic Oscillation peak position (BAO peak)[1], structure growth rate[2] and test of geometry of the Universe through Alcock-Paczynski test[3].

The brute-force calculation of 2pCF is $O(N^2)$ process[4] as one needs to calculate pair-wise distances of data points and bin them. They get more computationally intensive as histogram construction time gets added to the time

---

taken to compute distances as per the assumed Cosmology model[5]. This demands code optimization for multiple-CPUs, clusters and GPUs to decrease the computational time[4][6][7]. A. Moore *et. al.* [8] was one of the first to propose application of nearest neighborhood methods (specifically KDTree) in Cosmology for 'fast' calculation of N-point statistics. All the modern codes for 2pCF use nearest neighbor methods in combination with CPU/GPU computational optimizations.

Many tools and software packages based on these faster implementations to calculate 2pCF exist[9][10][11][12] and the need for a new software may be questioned. Two primary reasons for this can be explained as follows. First, all of the implementations using nearest neighborhood methods limit themselves to Euclidean distance metric[10][11][12][13]. While these can be used for a multitude of flat cosmologies, those studying alternative models dealing with non-Euclidean spaces have no generic codes available to obtain correlation statistics. As a result, they resort to using the model-independent methods/parameters to constrain their models[14][15]. Second, most of the existing codes are designed with box geometry in mind meant for evaluating the N-body simulations and not for complicated survey geometries. A beginner looking to extract 2pCF from redshift surveys like SDSS/BOSS, has no easy way to start as it needs multiple steps ranging from random catalog creation for survey geometry, defining distance metrics to final measurement of 2pCF.

`correlcalc` aims to fill this gap with an easily accessible generic python code for a beginner with step-by-step explanation of the procedure to extract 2pCF statistics from a large scale catalog. The goal of this work is two-fold. First, to present a generic 'recipe' to compute 2pCF for any arbitrary cosmology model of any geometry. Second, to compute galaxy 2pCF quickly and reliably with minimal computational resources (on a typical laptop/desktop).

Organization of the content in this paper is as follows. After listing a few basic formulae needed for computation of distances (2.2) *etc.* Section 2 introduces nearest neighborhood methods and their popular implementations in `python`(2.4). The section on algorithm(3) sequentially covers the important as-

pects of the recipe *viz.* data preparation(3.1), random catalog creation(3.2), data visualization(3.3), defining custom distance metrics(3.4), computing data pairs into bins to finally calculate 2pCF(3.5) and anisotropic 2pCF(3.6). After the discussion of algorithm, the article concludes with results(4) obtained using this recipe for DR7 VAGC[16] catalog with brief discussions. `python` implementation of the algorithm is available in the form of python package hosted on github[17] (also on ASCL and PyPI).

2. Theory

The two-point correlation function (ξ) is defined[2] by the joint probability of finding an object in both of the volume elements δV_1 and δV_2 at separation χ_{12}. It measures the excess probability of finding a couple of galaxies separated by spatial distance χ_{12} (or angular distance θ) in comparison to the galaxies separated by the same distance or angle in a Poisson random distribution.

$$\delta P = n^2 \delta V_1 \delta V_2 [1 + \xi(\chi_{12})] \tag{1}$$

Here n is number density of the objects.

2.1. Calculation of 2pCF

To empirically calculate 2pCF we can use any of the common estimators listed in Vargas *et. al.* [18] such as the generalized Landy-Szalay estimator [19] given below in equation 2.

$$\xi(s; z) = \frac{DD - 2DR + RR}{RR} \tag{2}$$

Here DD, RR, DR are the normalized values of number of data–data ($dd(s)$), random–random ($rr(s)$) and data–random($dr(s)$) pairs in the specific co-moving distance radius.

$$DD = \frac{dd(s)}{N_d(N_d - 1)/2} \quad RR = \frac{rr(s)}{N_r(N_r - 1)/2} \quad DR = \frac{dr(s)}{N_d.N_r} \tag{3}$$

Here, N_d and N_r are no. of galaxies (data points) and no. of points in the random catalog respectively.

2.2. Calculation of distance between two galaxies

Galaxy surveys cannot directly measure the distances of individual galaxies. Instead, they measure redshifts of the objects. Assuming isotropic and homogeneous expansion of the Universe and a cosmology model a priori (fiducial model), we convert these redshifts(z) to co-moving distances ($\chi(z)$) using the following equation for standard ΛCDM model. For other models like $wCDM$, co-moving distance may be similarly defined.

$$\chi(z) = \frac{c}{H_0} \int_0^z \frac{dz'}{E(z')} \tag{4}$$

where

$$E(z) \equiv \sqrt{\Omega_{\mathrm{M}} \left(1 + z\right)^3 + \Omega_k \left(1 + z\right)^2 + \Omega_\Lambda} \tag{5}$$

Distance between any two points with given redshift (z) and angular position (RA and DEC) can be found using simple triangle law for flat geometry. For an arbitrary geometry, distance between $P_1(z_1, RA_1, DEC_1)$ and $P_2(z_2, RA_2, DEC_2)$) can be written as[20]

$$\chi_{12} = \left[S^2(z_1) + S^2(z_2) - 2C(z_1)C(z_2)S(z_1)S(z_2)\cos\theta - \Omega_k S^2(z_1)S^2(z_2)\left(1 + \cos^2\theta\right) \right]^{1/2} \tag{6}$$

Where $\chi(z)$ is co-moving distance at z and Ω_k is curvature defined as $\Omega_k = \Omega_M + \Omega_\Lambda - 1$

$$
\begin{aligned}
S(z) &\equiv S_K[\chi(z)] \\
&= \begin{cases}
(-\Omega_k)^{-1/2}\sinh\left[(-\Omega_k)^{1/2}\chi(z)\right], & (\Omega_k < 0), \\
\chi(z), & (\Omega_k = 0), \\
\Omega_k^{-1/2}\sin\left[\Omega_k^{1/2}\chi(z)\right], & (\Omega_k > 0),
\end{cases} \\[6pt]
C(z) &\equiv \frac{dS_K}{d\chi}(z) = \begin{cases}
\cosh\left[(-\Omega_k)^{1/2}\chi(z)\right], & (\Omega_k < 0), \\
1, & (\Omega_k = 0), \\
\cos\left[\Omega_k^{1/2}\chi(z)\right], & (\Omega_k > 0).
\end{cases}
\end{aligned}
\tag{7} \tag{8}
$$

4

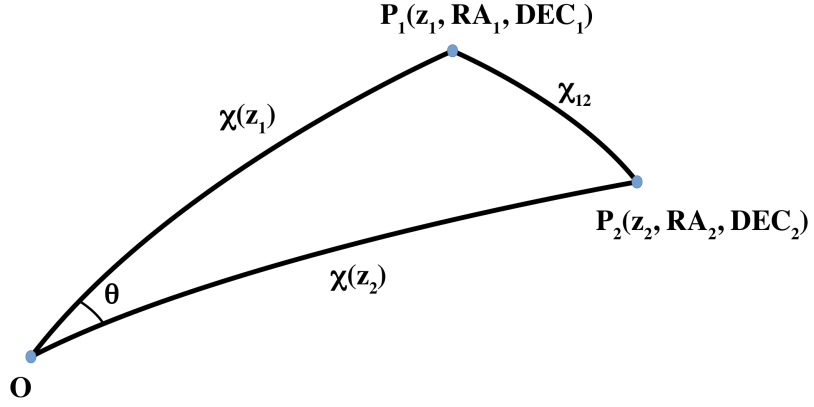Figure 1: Geometry of the relative position of the observer and the two points. The space is not necessarily Euclidean.

2.3. Calculation of LOS distance

The line-of-sight distance between two galaxies separated by $\Delta z = z_1 - z_2$ is dependent on the Hubble parameter at the average redshift (z) as

$$s_\| = \frac{c\Delta z}{H(z)}$$

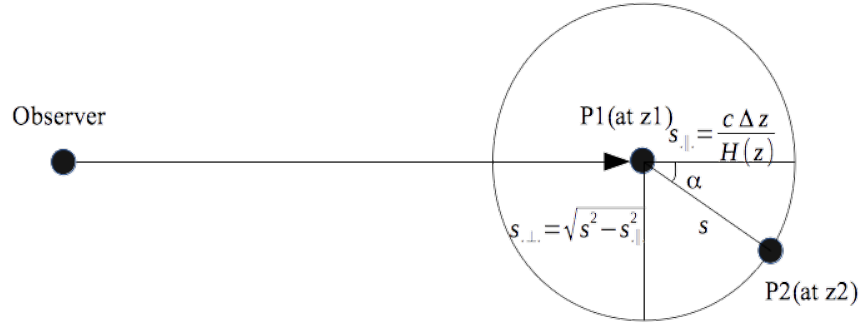Figure 2: Distance measures for 3D 2pCF

5

Perpendicular to the line-of-sight distance can be calculated using Pythagoras theorem assuming locally flat geometry or using angular diameter distance (D_A). We use the former and write $(s_\perp)$ as

$$s_\perp = \sqrt{s^2 - s_\parallel^2}$$

where s is the total distance between the two galaxies calculated using eqn. 6.

Angle between the two galaxies from the LOS axis (α) can be written as

$$\alpha = \cos^{-1}(\mu)$$

where $\mu = s_\parallel/s$. Two-point correlation as a function of distance(s) and μ $(\xi(s, \mu))$ is very useful in calculation of redshift distortions (β parameter) as it can be expanded as multipoles of μ.

2.4. Nearest Neighbor Methods

Formally, the nearest-neighbor (NN) search problem is defined[21] as follows: given a set S of points in a space M and a query point $q \in M$, find the closest point in S to q.

Nearest neighbor search (NNS), is the optimization problem of finding the point in a given set that is closest to a given point. Closeness is typically expressed in terms of a dissimilarity function: the less similar the objects, the larger the function values. Most commonly, M is a metric space and dissimilarity is expressed as a distance metric. Between two points, a distance metric must be symmetric and should satisfy the triangle inequality. Any arbitrary metric one likes to define must follow these two conditions. Often, M is taken to be the d-dimensional vector space where dissimilarity is measured using the Euclidean distance, Manhattan distance or other distance metric[22].

Brute-force method is to simply calculate distances between all the pairs and find the closest points by binning them based on the 'closeness'. This process is inefficient for large no. of data points as the computational time scales as N^2 where N is the no. of data points. One can, however, deploy some machine

6

learning to analyze the data/node structure in terms of a given metric and efficiently find nearest neighbors based on the 'learning'.

Several space-partitioning methods have been developed for solving the NNS problem. The simplest example is the KDTree, which iteratively bisects the search space into two regions containing half of the points of the parent region. Queries are performed via traversal of the tree from the root to a leaf by evaluating the query point at each split. Depending on the distance specified in the query, neighboring branches that might contain hits may also need to be evaluated. For constant dimension query time, average complexity is $O(\log N)$[23].

Most codes that calculate 2pCF (for e.g. CUTE[11]) use KDTree to find the nearest neighbors which leads to very fast results on even ordinary computers. However, KDTree implementations inherently work in Euclidean space. As our target is to find a generic recipe irrespective of the geometry, we use `BallTree` for application in Non-Euclidean spaces. BallTree algorithm is similar to KDTree but instead of calculating the median values and assigning space-zones iteratively, BallTree divides space into leaves with varying radius. Example in figure **??** depicts how a simple Ball tree partition of a two-dimensional parameter space is created step-by-step.

Once an indexing structure is created, one can find the number of points near any random node (within a specified distance) by simply counting the leaves of the tree. Based on the 'leaf-size' of the tree one might need to calculate individual distances by brute-force method near the edges of the tree. More details on the implementation of these algorithms can be found at scikit-learn (sklearn) documentation.

3. Algorithm

First of all, we take galaxy/quasar data from a redshift survey. In this paper, we take SDSS DR7 VAGC from Kazin *et. al.*[16] to demonstrate the procedure. Typical data from a redshift survey contains list of galaxies with their observed redshift (z), Right ascension (RA) and declination angles (DEC) that provide
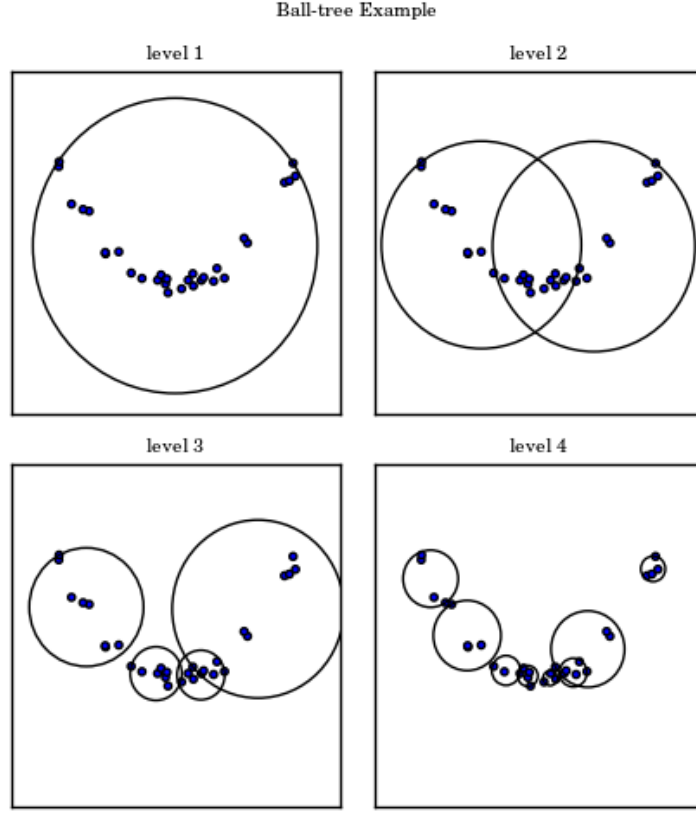
7

Figure 3: `BallTree` creation steps (figure taken from [13])

the position of each galaxy. Often, a value added catalog limits the magnitude and chooses a specific type of galaxy such as Luminous Red Galaxies - LRGs in DR72 VAGC. There are more parameters in a typical catalog such as magnitude,

140 survey completeness etc. In this paper, we shall only make use of the redshift, angular positions and radial weights of each galaxy from the catalog. Survey geometry and angular distribution are taken in mangle polygon file formats (`.ply`) provided by the respective survey data release. Value added catalogs such as the ones taken from (non-official) SDSS DR72 [24] also provide random

145 catalogs. Thye are typically much larger than the galaxy catalog ($15 - 30 \times$).

Hence, the computation time depends mainly on the size of the random catalog. The procedure provided in this paper can certainly be run on these random catalogs. However, as we will see later (3.2), to reduce the computational effort we can create a smaller random catalog without much loss of accuracy in the final 2pCF result. Figure **??** depicts flowchart of the recipe. Implementation of the algorithm with block-by-block explanation and code is available in the form of a `python` package `correlcalc` on github[17].

3.1. Data Preparation

After installing relevant packages, we need to obtain data from a galaxy catalog. In this work, we are demonstrating with the DR72 VAGC taken from Kazin *et. al.* [16]. One can also create a large scale structure galaxy catalog by following instructions given in the SDSS LSS catalog creation tutorial[25]. After obtaining the catalog with redshift(z) and angular co-ordinates, we calculate the co-moving distance as per the model of our choice (fiducial cosmology). Next the data is loaded into the RAM as $3 \times N_d$ matrix $[s, rar, decr]$ where s is co-moving distance, rar and $decr$ are right ascension and declination in radians. While these calculations can be done as part of the distance metric to be defined in section 3.4, it is faster to calculate them first and keep distance calculation minimal. It is also wise to save the data in $3 \times N_d$ matrix format in an ascii file or in pickled data form for easy access.

3.2. Random Catalog Preparation

If we use the random catalog given by the survey, we follow the same steps to load random catalog into RAM in form of $3 \times N_r$ matrix with $[s, rar, decr]$ values. However, it is computationally intensive to calculate correlation in Landy-Szalay method (and other methods which need random-random correlation) for high no. of random points. So, we can create a smaller random catalog which can give reasonably accurate results in comparison to the standard random catalog. (Only exception is probably Davis & Peebles method[18] that only needs DD and DR, in which case this recipe will yield even quicker results.)

9

175   All the major surveys provide the survey geometry in form of angular masks in `fits` and/or `ply` formats. We use mangle polygons [26] [27] given by the galaxy surveys to create random points that fall within the survey geometry.

We use `pymangle` – a python wrapper to a faster `C/C++` code to manipulate mangle angular masks[28]. It can create a random catalog with a specified no. of
180   data points that follow the angular masks provided by the survey. Mangle gives random points in the angular domain only providing RA and DEC. We will also need redshift values to be assigned to these random points. These redshifts need to follow the survey radial distribution to create a matching random catalog. If we are creating a random catalog that is of the same size as the data catalog
185   then it is easier to re-use the redshifts of the data and shuffle them to assign to the random catalog points. This ensures the radial distribution of the random points to be the same as galaxy catalog. However this can result in reduced accuracy due to suppressed radial modes. In stead it is better to create a catalog that has same distribution as the input data catalog[16].

190   If we are to create a random catalog of different size (recommended to use at least $\geq 2\times$ of the data catalog to reduce shot noise.) we can plot the histogram or Kernel density estimator(KDE) of the data redshift distribution and create a random catalog that mimics this distribution. As KDE is a better choice than using a histogram, we create a random distribution points for redshifts
195   using the same. `pymangle` though faster, is not parallelized to generate random angular points quick enough for high number of data points. (It can take more than 2 hours to create a random catalog of $2\times$ size for DR7, this process is a bottle-neck). Once the catalog is created and stored in an ascii file it can be reused for different realizations and mock catalogs. To minimize the variance of
200   clustering measurements for an inhomogeneous sample, we assign radial weights to each random point created. While calculating the pair counts, we assign to each data point a radial weight of $1/(1 + n(z)P_w)$[29], where $n(z)$ is the radial selection function and $P_w \sim 10^4 h^{-3} Mpc^3$[16]

10

3.3. Healpix Visualization

It is often useful to have a visual representation of the galaxy catalog distribution. HealPix[30] (and the wrapper healpy) are extremely useful to visualize the galaxy surveys. After creating the random catalog it is good to cross-check if they all indeed follow the supplied geometry. Using Healpix, we plot the random catalog containing RA and DEC data with points created as per the mask as shown in figure 5.

Depending on the NSIDE parameter, each pixel covers some angular area in which galaxies of relevant RA, DECs fall. Pixels are assigned values of total weights added of all the galaxies. Iterating this process to calculate pixel data for all the galaxies (or random points) in the survey area creates a pixelized map of the point distribution.

3.4. Defining distance metric

Implementations of nearest neighbor methods in popular packages like `sklearn` [22] offer many distance metrics. Among them, only Euclidean metric is useful for application in the study of Large Scale Structures. Among the nearest neighbor implementations, only `BallTree` implementation of scikit-learn contains a provision to define 'custom' metric in stead of the default metrics. This metric should not be confused with FRW metric of cosmology. Here metric definition is for calculation of distances between two points given their co-ordinates (in this case co-moving distance and angular positions).

As described in section 2, we use the formulae given in Matsubara[20] to define custom metrics for arbitrary geometry. It is worth noting that calculation of co-moving distances done in section 3.1 is already dependent on the cosmology of our choice. As math functions are inherently faster in `C` than in `python`, we use `Cython`(a `C` implementation of `python`) to improve the speed of computation. We know that angle between two points on a sphere (points need not be at the same distance since we are only calculating angle between them. It can be projected onto a surface - celestial sphere) is

$$\cos\theta = \sin(DEC1)\sin(DEC2) + \cos(DEC1)\cos(DEC2)\cos(RA1 - RA2) \quad (9)$$

11

It is advisable to use square of the distance instead of the square root to reduce the additional computational effort that is needed to calculate `sqrt` function. In the later steps, binning can be done based on distance squares. For closed and open Universe cases, distance units become important as the values inside sin and sinh need to be dimensionless. We use c/H_0 as the default units for distance measurement where c is velocity of light and H_0 is Hubble parameter at the present epoch. Also, one needs to take into account that the formulae given in section 2.2(from [20]) are in natural units ($c = 1$). So in order to get dimensionless numbers inside sin or sinh we need to multiply the co-moving distance with an appropriate factor assuming the current Hubble expansion rate. After compiling the `Cython` code into a `python` module, we can import the same in the main code and use it as a custom metric in the BallTree creation.

3.5. Creating `BallTrees` and calculating 2pCF

We now create a BallTree using the custom metric for data catalog and random catalog separately. In creating the BallTree we can find the optimal leaf size to improve computation time/efficiency[23]. Optimizing `leaf_size` parameter gives best trade-off between node query time and brute-force distance computation. For more details on benchmarking and optimization parameters nearest neighbor algorithms check the article in reference [23].

After creating a BallTree object, we can use `query_radius` method to find the number of data points within a given radius. There is also a faster method- `two_point_correlation` (though not parallel) which calculates auto correlation function of the data within given bins. We use this for calculation of DD,RR and DR (RD) by subtracting the no. of pairs in the smaller bins progressively. We do this using `numpy.diff` method. To calculate data-random correlation (DR), we need to find the random points in the data-BallTree or vice-versa. This gives us DR. As the nearest neighbor methods are approximate methods, while counting the pairs in a given radius some of the nearest neighbors may not be counted while few others may get included. This can result in slight deviation

12

(especially at large scales) in DR and RD values as they use different trees for calculation of cross-correlation. Hence, we normalize this effect by taking an average of DR and RD to use it in the 2pCF formula.

The above mentioned approach assigns a pair to the relevant bin by adding "1" every time a pair is found within the specified distance. However, in surveys such as SDSS we often need to add relevant weights instead of adding 1's. These weights have to be systematically added to the bins to obtain accurate 2pCF. For more details on different types of weights used in the preparation of LSS catalog please refer to the "Weights" section of SDSS tutorial [25].

To achieve this weighted sum, we can either patch the existing `two_point_correlation` method in `sklearn` or define a method that can do it with similar efficiency. Here, we follow the latter approach and create methods to find autocorrelation and cross-correlation of a distribution with previously defined metrics. Just as in the case of brute-force approach, one can calculate all the pairwise distances to bin them to obtain DD, RR and DR. But instead of calculating distances for all the pairs, we narrow down the search of pairs to the maximum distance needed by the provided bins. For e.g. if we like to calculate 2pCF up to 200Mpc distance, we first need to find the pairs that are in 200Mpc radius and calculate distances only for the pairs that fall in this radius. We repeat this process for each data/random point removing them one by one after the neighbor distances are calculated. This ensures we do not count the same pairs multiple times.

Implementing this strategy is straightforward once a `BallTree` is created for data and random points. To calculate the autocorrelation (DD and RR), we first create a `BallTree` of the data and iterate through all the points in that dataset to find the nearest neighbors within the given radius (max. value of the bins provided). This narrows down the no. of pairs to a smaller manageable number. We can then calculate distances between the chosen point and its neighbors with the specified custom metric using `scipy.spatial.distance.cdist` and bin the distances as per the weights of those neighbors (using `numpy.histogram`'s weights option). This process when iterated over all the points gives us autocorrelation. This iterative loop can be parallelized to achieve speed of computation.

13

To calculate cross-correlations (RD for e.g.) we iterate over random catalog points in the data `BallTree` to find nearest neighbors within the radius. We then calculate distances between the nearest neighbors of random catalog points in the data point catalog to bin them with (data) weights to calculate the cross-correlation.

Having obtained DD, RR and DR, we can find any of the common estimators as described in the theory section. For quick computation using Landy-Szalay estimator on low-spec machines, we can create a smaller (typically $2\times$) size random catalog3.2. For quick results, we can also use Davis - Peebles estimator as it only needs DD and DR. It is easy to compute as it avoids calculation of RR for random catalogs of large size. Although it is not a very accurate estimator[18] at scales above 100-125Mpc (for say BOSS geometry), it is good enough for pedagogical purposes as it can compute 2pCF very fast and yet show basic features such as power-law of 2pCF at small scales and BAO peak. In situations where we need more accurate results we use the Landy-Szalay estimator [19] given in equation 2.

3.6. Calculating anisotropic 2pCF

Anisotropic(3D) 2pCF is more useful as it can separate evolution effects in line-of-sight and angular diameter distance in the perpendicular direction. To calculate anistropic 2pCF, we need to find distances using two metrics - typically parallel to the line-of-sight(LOS) - $s_\parallel$ and perpendicular to the LOS - $s_\perp$. We can also use co-moving distance (s) and cosine of the angle from LOS ($\mu = \cos(\alpha)$). Definition of these parameters is shown in figure ??.

Alternatively, there is Δz and $z\Delta\theta$ parametrization for model-independent calculation of anisotropic 2pCF[14] which can be extended to all general cases by using suitable metrics. This method can also be used to calculate other realizations for small Δz. We compute anisotropic 2pCF in all the above realizations to demonstrate the algorithm.

To efficiently calculate anistropic 2pCF, we follow similar approach to the calculation of 2pCF. Since we have two metrics instead of one, we narrow down

14

the the search of pairs using one 'filter' metric (typically the radius) and cal-
culate pairwise distances using both the metrics. For e.g. if one metric cal-
culates Δz and another $z\Delta\theta$, we first select all the pairs that fall in radius of
$\sqrt{(z\Delta\theta)^2 + (\Delta z)^2}$ and then calculate distances using both the metrics. This
is similar to the approach taken by Lopez [14] and it helps in reducing the
computational effort. Algorithm to calculate anisotropic 2pCF (also called 3D
two-point correlation) is described in figure **??**.

4. Results and Discussions

Following the recipe in section 3, we obtain two-point correlations for stan-
dard model - flat ΛCDM with $\Omega_M = 0.3$ and $\Omega_\Lambda = 0.7$ as fiducial model for
SDSS DR7 VAGC catalog and summarized in one plot(1). Demonstrating the
recipe's capability to work for alternative models, we also obtain 2pCF for open
geometry (Milne) and $R_h = ct$ to compare with the standard model (fig.1a). To
reduce the computational time, a smaller random catalog of $2\times$ size is created.
It is remarkable to note that it doesn't cause any significant loss of accuracy in
comparison to the default random catalog ($\approx 16\times$) (fig.1b). Hence, this recipe
and the associated package can be extremely useful to quickly validate alter-
native models and can be used for pedagogical purposes. Computation time
using $2\times$ random catalog was about an hour to obtain the final results using LS
estimator[1]. We also tested the impact of adding weights instead of doing '+1'
in counting pairs. As also confirmed in Kazin *et.al.*[16], there is no significant
difference observed in using weights in pair counting as can be seen in fig.1c.
For comparison, a plot of with DR3-LRG catalog (original BAO peak detection
data) is also presented. We can see the BAO peak clearly around $100h^{-1}Mpc$
in all the plots.

We calculated and plotted 3 different realizations of the 3D two-point corre-
lation function. Fig.1d is model-independent plot of 2pCF $\xi(\Delta z, z\Delta\theta)$. $\xi(s_\parallel, s_\perp)$

---

[1]Run on i7-2.4GHz quad-core with 16GB RAM

is plotted as a function of line-of-sight distance $(s_\parallel)$ and perpendicular to the LOS $(s_\perp)$ in fig.1e. In fig.1f, we plot $\xi(s,\mu)$ as a function of distance(s) and μ. These match the results obtained by Kazin *et.al.* [31] as expected.

While the algorithm presented here is usable and extend-able to numerous cosmology models of different geometries, it can be further optimized to achieve better speed. We plan to implement a full Cython code in future revisions of the code. Study of correlation statistics in alternative models using `correlcalc` will also be explored in the future work(s).

5. Acknowledgments

16

References

References

[1] D. J. Eisenstein, I. Zehavi, D. W. Hogg, R. Scoccimarro, M. R. Blanton, R. C. Nichol, R. Scranton, H.-J. Seo, M. Tegmark, Z. Zheng, et al., Detection of the baryon acoustic peak in the large-scale correlation function of sdss luminous red galaxies, The Astrophysical Journal 633 (2) (2005) 560.

[2] P. J. E. Peebles, The large-scale structure of the universe, Princeton university press, 1980.

[3] X.-D. Li, C. Park, C. G. Sabiu, H. Park, D. H. Weinberg, D. P. Schneider, J. Kim, S. E. Hong, Cosmological constraints from the redshift dependence of the Alcock–Paczynski effect: Application to the SDSS-III BOSS DR12 galaxies, The Astrophysical Journal 832 (2) (2016) 103.

[4] R. Ponce, M. Cárdenas-Montes, J. J. Rodriguez-Vazquez, E. Sánchez, I. Sevilla, Application of gpus for the calculation of two point correlation functions in cosmology, arXiv preprint arXiv:1204.6630.

[5] M. Cárdenas-Montes, M. Á. Vega-Rodríguez, I. Sevilla, R. Ponce, J. J. Rodríguez-Vázquez, E. S. Álvaro, Concurrent cpu-gpu code optimization: The two-point angular correlation function as case study, in: Conference of the Spanish Association for Artificial Intelligence, Springer, 2013, pp. 209–218.

[6] D. Bard, M. Bellis, M. T. Allen, H. Yepremyan, J. M. Kratochvil, Cosmological calculations on the gpu, Astronomy and Computing 1 (2013) 17–22.

[7] M. Cárdenas-Montes, M. A. Vega-Rodrıguez, J. Westerholm, R. Ponce, E. Yurtesen, I. Sevilla, E. S. Alvaro, J. J. Rodrıguez-Vázquez, M. Aspnas, V. Timonen, et al., Calculation of two-point angular correlation function: implementations on many-core and multicore processors, in: Ibergrid Conference Editorial Universitat Politècnica de València, 2013, pp. 203–214.

17

[8] A. W. Moore, A. J. Connolly, C. Genovese, A. Gray, L. Grone, N. Kanidoris II, R. C. Nichol, J. Schneider, A. S. Szalay, I. Szapudi, et al., Fast algorithms and efficient statistics: N-point correlation functions, in: Mining the Sky, Springer, 2001, pp. 71–82.

[9] B. Reid, cosmoxi2d: Two-point galaxy correlation function calculation, Astrophysics Source Code Library.

[10] A. Hearin, D. Campbell, E. Tollerud, P. Behroozi, B. Diemer, N. J. Goldbaum, E. Jennings, A. Leauthaud, Y.-Y. Mao, S. More, et al., High-precision forward modeling of large-scale structure: An open-source approach with halotools, arXiv preprint arXiv:1606.04106.

[11] D. Alonso, Cute - correlation utilities and two-point estimation, `https://github.com/damonge/CUTE`, [Online; accessed 09-Sept-2017] (2011).

[12] M. Sinha, L. Garrison, Corrfunc: Blazing fast correlation functions on the cpu, Astrophysics Source Code Library.

[13] J. Vanderplas, A. Connolly, Ž. Ivezić, A. Gray, Introduction to astroml: Machine learning for astrophysics, in: Conference on Intelligent Data Understanding (CIDU), 2012, pp. 47 –54. `doi:10.1109/CIDU.2012.6382200`.

[14] M. Lopez-Corredoira, Alcock-paczyński cosmological test, The Astrophysical Journal 781 (2) (2014) 96.

[15] F. Melia, M. López-Corredoira, Alcock–paczyński test with model-independent bao data, International Journal of Modern Physics D 26 (06) (2017) 1750055.

[16] E. A. Kazin, M. R. Blanton, R. Scoccimarro, C. K. McBride, A. A. Berlind, N. A. Bahcall, J. Brinkmann, P. Czarapata, J. A. Frieman, S. M. Kent, et al., The baryonic acoustic feature and large-scale clustering in the sloan digital sky survey luminous red galaxy sample, The Astrophysical Journal 710 (2) (2010) 1444.

18

[17] Y. Rohin, correlcalc - python package to calculate 2pCF., `https://github.com/rohinkumar/correlcalc`, [Online; accessed 04-Oct-2017] (2017).

[18] M. Vargas-Magaña, J. E. Bautista, J.-C. Hamilton, É. Aubourg, A. Labatie, J.-M. Le Goff, S. Escoffier, M. Manera, C. McBride, D. Schneider, et al., An optimized correlation function estimator for galaxy surveys, Astronomy & Astrophysics 554 (2013) A131.

[19] S. D. Landy, A. S. Szalay, Bias and variance of angular correlation functions, The Astrophysical Journal 412 (1993) 64–71.

[20] T. Matsubara, Correlation function in deep redshift space as a cosmological probe, The Astrophysical Journal 615 (2) (2004) 573.

[21] D. E. Knuth, The art of computer programming, vol. 3: Sorting and searching," 1973, This book describes the binary search on 409–426.

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[23] J. Vanderplas, Benchmarking nearest neighbor searches in python, `https://jakevdp.github.io/blog/2013/04/29/benchmarking-nearest-neighbor-searches-in-python/`, [Online; accessed 09-Sept-2017] (2013).

[24] E. A. Kazin, (the non-official) sloan-digital sky survey luminous red galaxy sample, `http://cosmo.nyu.edu/~eak306/SDSS-LRG.html`, [Online; accessed 21-Aug-2017] (2010).

[25] SDSS, Creating a large scale structure galaxy catalog, `http://www.sdss3.org/dr9/tutorials/lss_galaxy.php`, [Online; accessed 09-Sept-2017] (2011).

[26] A. Hamilton, M. Tegmark, A scheme to deal accurately and efficiently with complex angular masks in galaxy surveys, Monthly Notices of the Royal Astronomical Society 349 (1) (2004) 115–128.

[27] M. E. Swanson, M. Tegmark, A. J. Hamilton, J. C. Hill, Methods for rapidly processing angular masks of next-generation galaxy surveys, Monthly Notices of the Royal Astronomical Society 387 (4) (2008) 1391–1402.

[28] E. Sheldon, M. White, pymangle 0.9.0 - simple python code to read and work with mangle masks., `https://pypi.python.org/pypi/pymangle/`, [Online; accessed 21-Aug-2017] (2016).

[29] H. A. Feldman, N. Kaiser, J. A. Peacock, Power spectrum analysis of three-dimensional redshift surveys, arXiv preprint astro-ph/9304022.

[30] K. M. Górski, E. Hivon, A. J. Banday, B. D. Wandelt, F. K. Hansen, M. Reinecke, M. Bartelmann, HEALPix: A Framework for High-Resolution Discretization and Fast Analysis of Data Distributed on the Sphere, ApJ 622 (2005) 759–771. `arXiv:astro-ph/0409513`, `doi:10.1086/427976`.

[31] E. A. Kazin, M. R. Blanton, R. Scoccimarro, C. K. McBride, A. A. Berlind, Regarding the line-of-sight baryonic acoustic feature in the sloan digital sky survey and baryon oscillation spectroscopic survey luminous red galaxy samples, The Astrophysical Journal 719 (2) (2010) 1032.
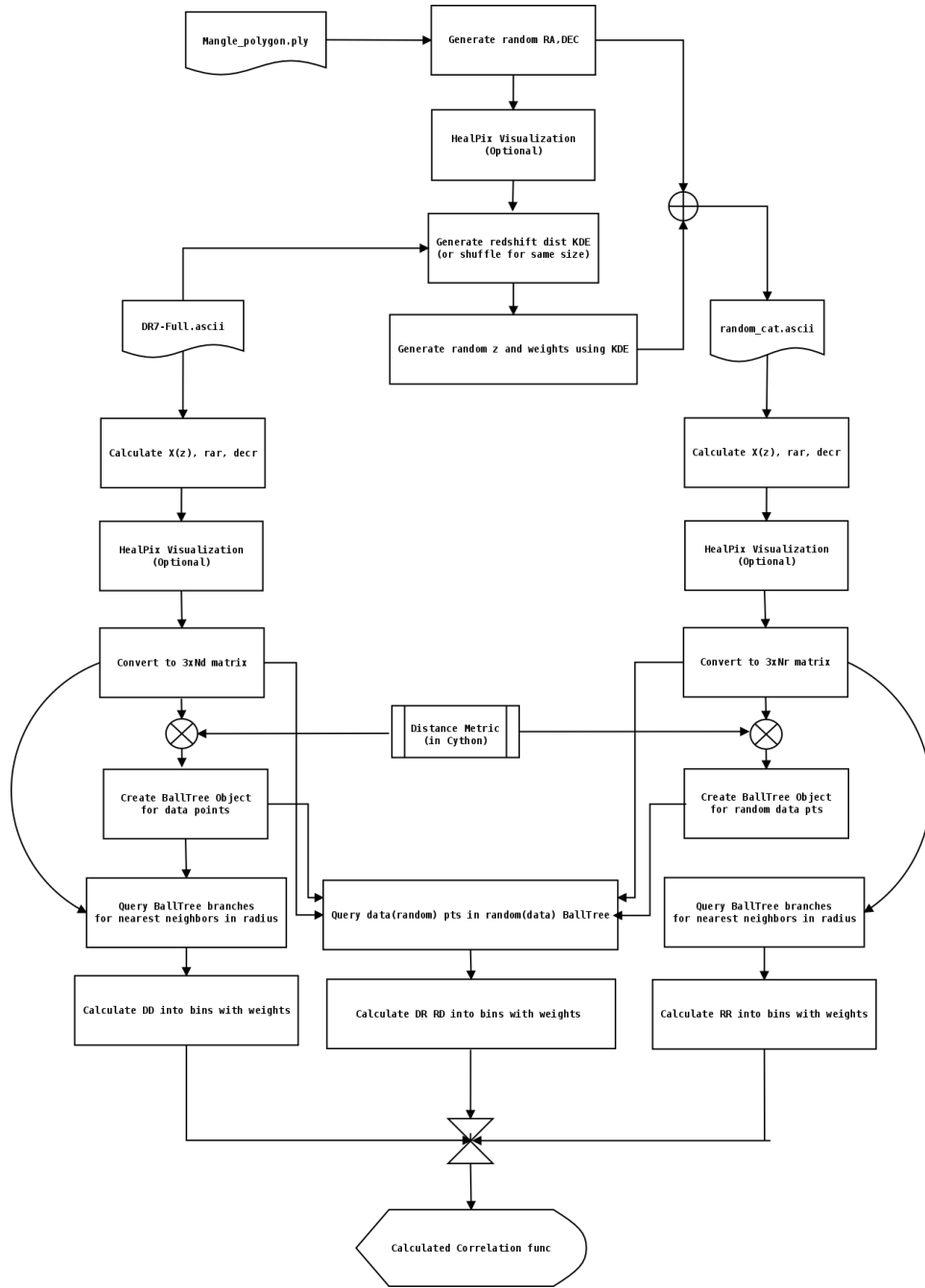
Figure 4: Flow chart of the 'generic' 2pCF computation recipe
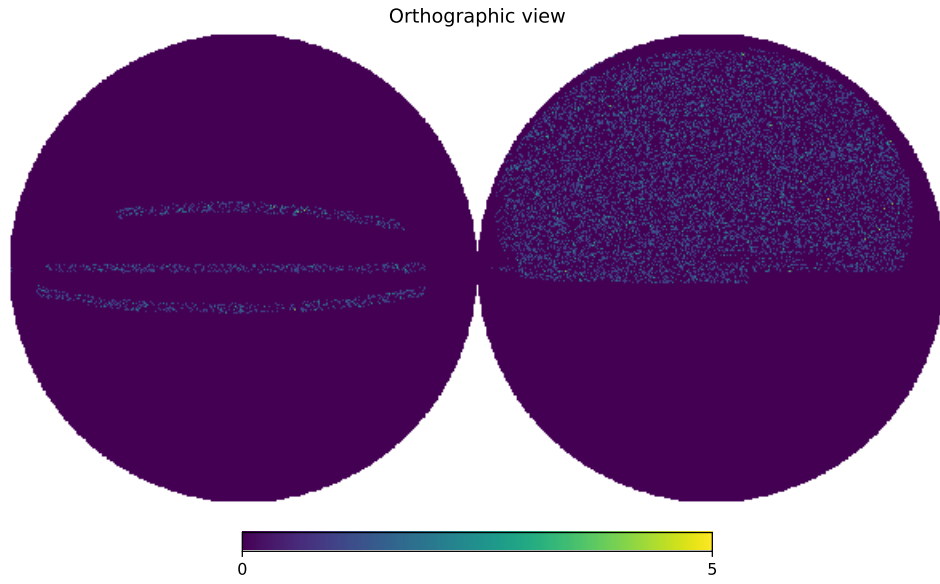
Figure 5: Generated random catalog with DR72 mask - `orthview`
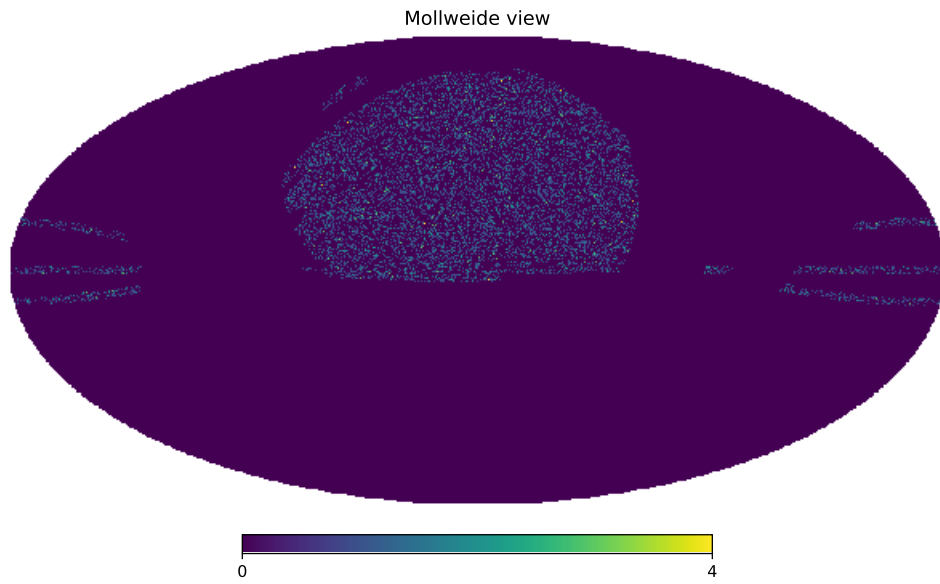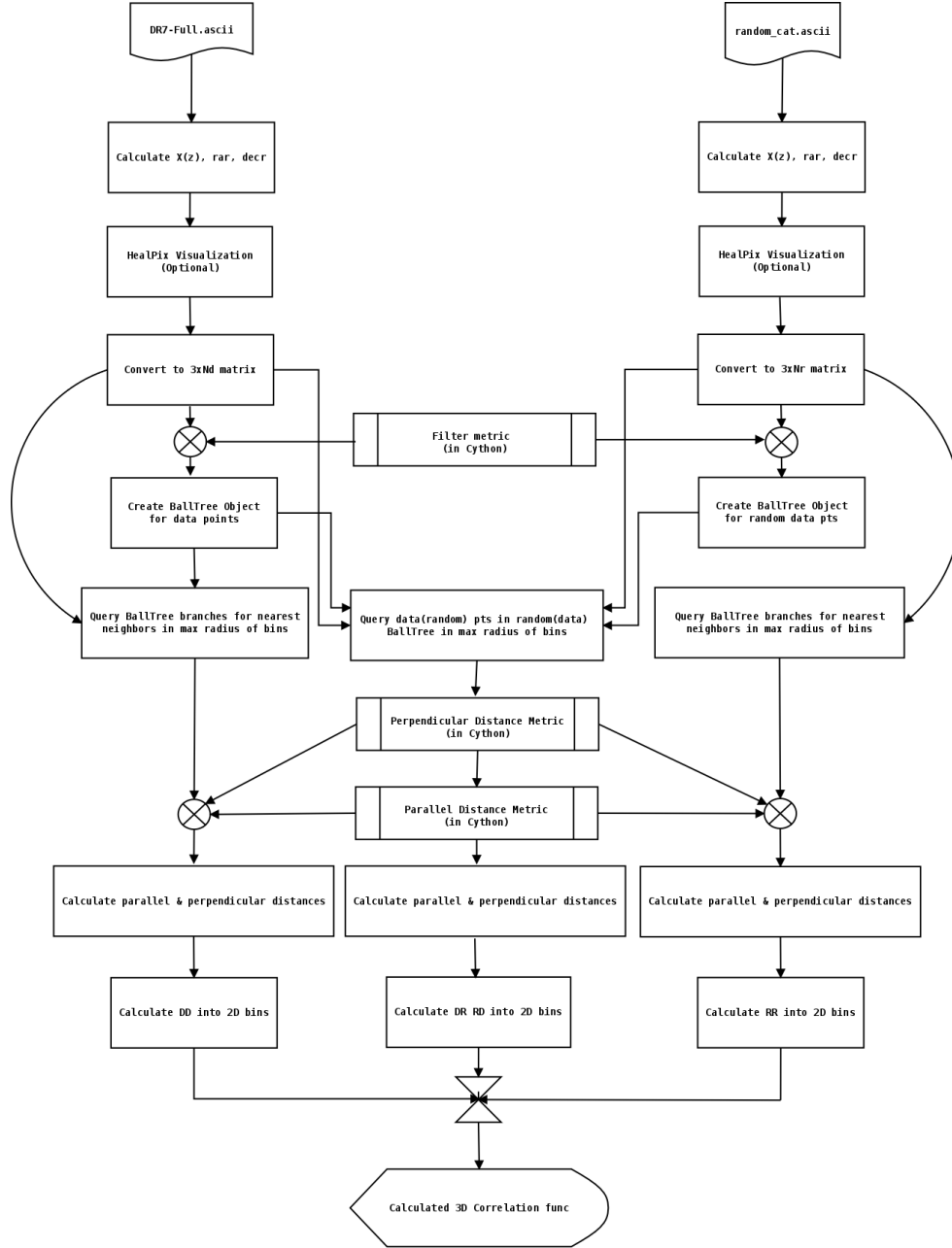

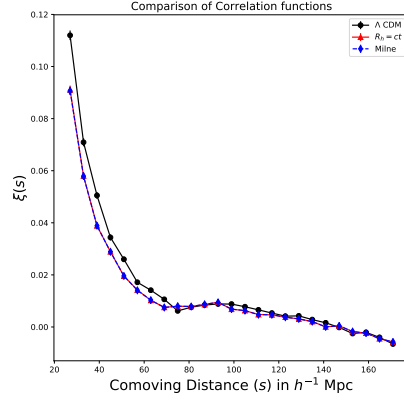
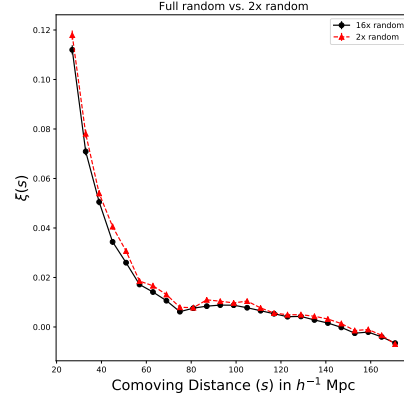Figure 6: Generated random catalog with DR72 mask - `mollview`

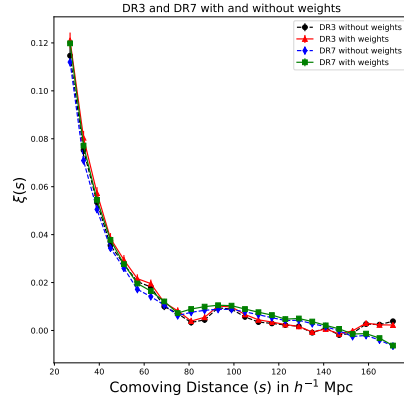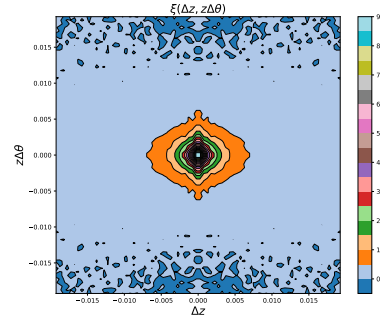Figure 7: Flow chart of the 3D 2pCF computation recipe
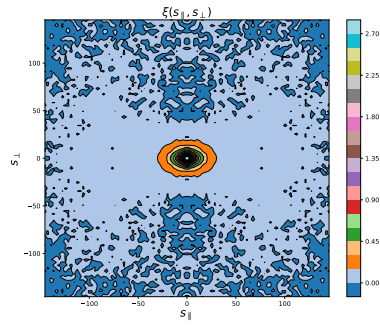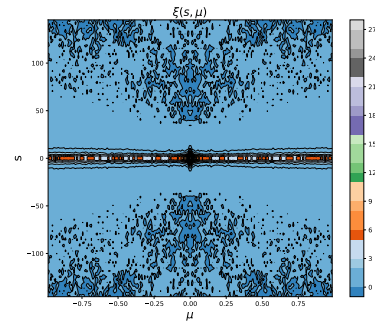
(a) Figure 1a

(b) Figure 1b

(c) Figure 1c

(d) Figure 1d

(e) Figure 1e

(f) Figure 1f

Table 1: Two-point correlation functions obtained using `correlcalc`