Accepted for publication in PASP.

CfunBASE: A Cosmological Functions Library for Astronomical Databases

M. Taghizadeh-Popp [1]

ABSTRACT

CfunBASE is a customizable C# cosmological functions library in the .NET framework. Its primary use is in CasJobs/SkyServer, where the functions are ported into a MS-SQL Server database hosting the SDSS science archive, and can be executed through simple T-SQL commands. This gives cloud-computing users the possibility of calculating cosmological distances, volumes and times as a function of redshift, as well as their respective inverse functions. Also included are basic data exploratory analysis tools, such as binning, N-dimensional weighted histograms and quantile and cumulative distribution aggregate functions. Advanced astronomical examples are presented and discussed, such as the T-SQL implementation and fast computation of luminosity functions, color-magnitude diagrams and the friends-of-friends galaxy cluster identification algorithm.

Subject headings: Tutorial — Data Analysis and Techniques — Astronomical Techniques

1. Introduction

Science has evolved today to include the essential use of computer science and applied math and statistics. After starting with plain observation, it was complemented with a theoretical background and, lately, computer simulations have been used to reproduce complex phenomena out of the reach of theory.

Nowadays, we experience the emergence of a fourth paradigm: data-intensive or e-science (see articles in Hey et al. (2009)), which engulfs the previous three by providing ways of

1. Collecting a huge amount of observational data provided by technological advances ("data deluge"), either from networks of cheap massified sensors, high output state-of-the-art instruments, computer simulations, or other mechanisms with output reaching now Petabytes/year (Szalay et al. 2001).

[1]Department of Physics and Astronomy, Johns Hopkins University. 3400 North Charles Street, Baltimore, MD 21218, USA. e-mail: mtaghiza@pha.jhu.edu

2. Preprocessing, archiving and preserving the data in servers, with emphasis in many low energy consumption hard drives with fast I/O and parallel CPUs or GPUs, supported by an efficient database management system (DBMS).

3. Visualizing and analyzing the complex high-dimensional data space with statistical datamining tools right where the data is stored physically, having the small sized aggregated results made available to the client for further scientific use.

In fact, the data deluge makes infeasible to either move, download or analyze all the data in personal computers or Beowulf type clusters (Bell et al. 2009). This forces to "bring the computation to the data", having it stored in clusters of server optimized for a balance between parallel computing power and data access, transfer and analysis for many on-demand users, rather than only performance-centered computational power for a small or centralized amount of users (Szalay et al. 2009a,b). The implementation of data warehouses brings the possibility of cloud computing (Armbrust et al. 2009), by means of which any client can connect to its own small database or environment in servers (i.e. "the cloud"), where services are offered, data can be stored, queried or shared and provided software can be run.

Current examples, in and outside science, can be found in the study of consumer habits, track of global commercial transactions, Geospatial Information Systems and web mapping (e.g Google Earth), genomics (e.g. Genome Project, Roehm & Blakeley (2009)), online social networks (e.g. Facebook), web search engines (e.g. Google, Yahoo), experimental particle physics (e.g. Large Hadron Collider), data/mass surveillance (Kolda et al. 2005; Popp & Poindexter 2006), etc. In astronomy, the spreading of high resolution CCDs and high quality optics telescopes has fostered an avalanche of data. A pioneering example is the Sloan Digital Sky Survey (SDSS) (York et al. 2000), that with a 120Mpix CCD camera and 2.5m telescope has scanned 25% of the sky, generating ∼30TB of data. A successful example of cloud computing is SkyServer (Szalay et al. 2001; Singh et al. 2006), which provides public access to SDSS data archive by means of the web based service CasJobs [1] running on MS-SQL Server. Data storage and analysis in next generation surveys will be much more demanding, as Pan-STARRS [2], LSST (Ivezic et al. 2008; Abell et al. 2009) and others will be providing time series data at a rate of 30TB/night of raw data.

In this paper we present *CfunBASE* [3], a .NET framework library written in C# that contains functions relevant in cosmology and extragalactic astronomy, made ready to be uploaded into a MS-SQL 2005/2008 server database and executed *in situ* using T-SQL queries. Several previous cosmology libraries have been included in web services e.g. Wright (2006) and Kitching et al. (2009), but until now there is a lack of a library written in a fast low level programming language

[1] `http://cas.sdss.org/dr7/en/`

[2] `http://pan-starrs.ifa.hawaii.edu/public/`

[3] Downloadable from `http://www.skyserver.org/cfunbase`

with database integration in order to handle the data deluge.

CfunBASE's primary intended use is in CasJobs [4], where it provides measurements of cosmological distances, volumes and times as a function of redshift, with which astronomers or the general public can do sample selection, identification of galaxy clusters and filtering of extragalactic objects inside "the cloud", without having to download superfluous data into their computers. Current and future sky surveys, such as LAMOST (Sun et al. 2006; Wang et al. 2009), Pan-STARRS and LSST will provide redshift measurements obtained either from spectroscopy ($\sim 10^6$-10^7) or multiband photometry ($\sim 10^9$-10^{10}), on which *CfunBASE* might be of great utility. The library also includes basic data exploratory analysis and statistical tools, such as quantile aggregate functions, basic binning and creation of N-dimensional weighted histograms, helpful for visualizing the high dimensional data space. Section 2 of this paper reviews the theory of the universe dynamics under general relativity and lists basic cosmological functions, Section 3 discusses details on the implementation of the library and Section 4 shows *CfunBASE*-related advanced examples of astronomical interest, which are shown written as T-SQL scripts in Appendix B.

2. Theoretical Background

Quantities of basic interest in cosmology are measurements of the distances, volumes and times involved in dynamics of an expanding universe. The theoretical framework is widely known, but is included here for completeness. Some papers and books on these matters include Hogg (2000), Hogg et al. (2002), Liske (2000), Peacock (1999), Peebles (1993), Martinez & Saar (2002), Bonometto et al. (2002), Dodelson (2003), Weinberg (2008) and others.

Under General relativity, Einstein's field equation (Peacock 1999; Bonometto et al. 2002; Dodelson 2003) relates information about the content of the universe (energy-momentum tensor $T^{\mu\nu}$) with its geometry (Einstein tensor $G^{\mu\nu}$).

$$G^{\mu\nu} + \Lambda g^{\mu\nu} = -\frac{8\pi G}{c^4} T^{\mu\nu}. \tag{1}$$

where Λ is Einstein's famous cosmological constant. Here, $g^{\mu\nu}$ is the Robertson-Walker metric tensor, obtained from considerations of an isotropic and homogeneous universe (Weinberg 2008), under which the space-time line element (dependent on the curvature k) is given by (Bonometto et al. 2002; Martinez & Saar 2002)

$$ds^2 = c^2 dt^2 - R^2(t)[dr^2 + S_k^2(r)(d\theta^2 + \sin^2(\theta) d\phi^2)], \tag{2}$$

$$S_k(r) = \begin{cases} \sinh(r) & \text{if } k = -1, \text{ open universe} \\ r & \text{if } k = 0, \text{ flat universe} \\ \sin(r) & \text{if } k = 1, \text{ closed universe.} \end{cases} \tag{3}$$

[4]A preliminary version of *CfunBASE* is included as part of the DR7 database, while the full version will be provided in the next data release.

The coordinates r, θ and ϕ behave as dimensionless angles, being called "comoving" as they are defined on a 3-dimensional hypersurface that expands according to $R(t)$, which is the scale factor or curvature radius of the universe. The $k = 1$, $k = -1$ and $k = 0$ universes have the geometry of a 3-sphere, 3-hyperboloid and 3-plane respectively. Manipulations on Eqs. (1) and (2) lead to the Friedman equations (Bonometto et al. 2002; Martinez & Saar 2002):

$$H^2 \equiv \frac{\dot{R}^2}{R^2} = \frac{\dot{a}^2}{a^2} = \frac{8\pi G}{3}\rho + \frac{\Lambda c^2}{3} - \frac{kc^2}{R^2}, \tag{4}$$

$$\frac{\ddot{a}}{a} = -\frac{4\pi G}{3}(\rho + \frac{3p}{c^2}) + \frac{\Lambda c^2}{3}, \tag{5}$$

$$\dot{\rho} = -3\frac{\dot{a}}{a}(\rho + \frac{p}{c^2}). \tag{6}$$

Equation (4) defines the Hubble parameter $H(t)$, which determines the expansion rate of the universe, measured today to be $H_0 \equiv H(t = 0) \equiv h_0 \times 100$ Km/s Mpc^{-1}, where $h_0 = 0.705$ according to Komatsu et al. (2009). Here,

$$a(t) \equiv \frac{R(t)}{R_0} = \frac{1}{1+z} \tag{7}$$

is the normalized scale factor ($a_0 = 1$). The redshift $z \in [0, \infty]$ ($z(t = 0) = 0$) is used by astronomers as a proxy for time t, since it can be obtained directly by measuring the Doppler shift (with respect to a rest frame) of the observed wavelength λ in known spectral features of the spectrum of a galaxy that moves away from us, together with the universe expansion (or "Hubble flow"):

$$z \equiv \frac{\lambda_{\text{obs}}}{\lambda_{\text{rest}}} - 1. \tag{8}$$

In the Friedman equations, Λ, the pressure p and the density ρ give information about the content of the universe. The density is defined as $\rho = \rho_M + \rho_R$ (matter plus radiation), whereas the fiducial dark energy density can be written as $\rho_\Lambda = \Lambda c^2/(8\pi G)$. Using Eq. (4), we obtain that at any time there exists a total critical density $\rho_{Tc} = 3H^2(t)/(8\pi G)$ for which the universe is flat ($k = 0$). It amounts nowadays to 5.6×10^{-27}kg/m^3. We can define the normalized densities

$$\Omega_i = \rho_i/\rho_{Tc} \qquad (i \in \{M, R, \Lambda\}), \tag{9}$$

$$\Omega_T = \sum \Omega_i, \tag{10}$$

$$\Omega_k = -kc^2/(HR)^2, \tag{11}$$

with which Eq. (4) results in

$$\Omega_M + \Omega_R + \Omega_\Lambda + \Omega_k = \Omega_T + \Omega_k = 1. \tag{12}$$

We can solve for $H(t)$ by using Eq. (6) and a suitable equation of state (EOS) $p_i = w_i \rho_i c^2$ for each one of the universe's components. That leads us to

$$\rho_i(a) = \rho_{i,0} a^{-3(1+\bar{w}_i(a))} = \Omega_{i,0}\frac{3H_0^2}{8\pi G} a^{-3(1+\bar{w}_i(a))}, \tag{13}$$

$$\bar{w}_i(a) = \frac{1}{\ln a}\int_0^{\ln a} d\ln(a')w_i(a'). \tag{14}$$

Note that $w_M = 0$ and $w_R = 1/3$. For dark energy, the most basic non-evolving EOS is defined by $w_\Lambda = w_{\Lambda,0} \equiv w_0 = -1$. Summing up, the dynamics is determined by

$$\begin{aligned} H(z) &= H_0 E(z), & (15)\\ E^2(z) &= \Omega_{M,0}(1+z)^3 + \Omega_{R,0}(1+z)^2 + \\ &\quad \Omega_{\Lambda,0}(1+z)^{-3(1+\bar{w}_\Lambda)} + \Omega_{k,0}(1+z)^2. & (16)\end{aligned}$$

Knowledge of today's values of the cosmological parameters Ω_M, Ω_R, Ω_Λ, Ω_k, ω_0 and h_0 allows to determine the dynamics of the expansion and the cosmological functions concerning distances and times. The estimated values measured by using WMAP (5yr) combined with measurements from Type Ia supernovae and Baryon Acoustic Oscillations (Komatsu et al. 2009) are:

$$\begin{aligned}(\Omega_{M,0}, \Omega_{R,0}, \Omega_{\Lambda,0}, h_0, w_0) = \\ (0.2739, 10^{-4}, 0.726, 0.705, -1),\end{aligned} \tag{17}$$

which have been rounded to support the agreed observed fact that $\Omega_{T,0} = 1$ and $\Omega_{k,0} = 0$ (flat universe). Here we have considered the measured effective number of neutrinos equal to $N_{\rm eff} = 4.4$ and $\Omega_R = \Omega_\gamma(1 + 0.2271 N_{\rm eff})$, where $\Omega\gamma = 4.968 \times 10^{-5}$ according to Komatsu et al. (2009).

Distance and time scales are defined by

$$D_H(z) = cH^{-1}(z), \tag{18}$$

$$T_H(z) = H^{-1}(z), \tag{19}$$

called Hubble distance and time respectively, with current values of $D_{H,0} = 4252.38$ Mpc and $T_{H,0} = 13.87$ Gyr. Straight from Eqs. (11) and (12), we can obtain the curvature radius of the universe, given by

$$R(z) = D_H\sqrt{\frac{k}{\Omega_T - 1}} = \frac{D_{H,0}}{1+z}\sqrt{\frac{k}{\Omega_{T,0} - 1}}, \tag{20}$$

which is non-defined for $\Omega_{T,0} = 1$ (flat universe). As a photon always travels along null geodesics in space-time ($ds^2 = 0$), we have from Eq. (2) that $dr = cdt/R(t)$ when the light travels along the radial direction (i.e. line of sight). Also, from Eqs. 4 and 7 we obtain $dt = dz[(1+z)H]^{-1}$. Now, by including Eq. (15) we can obtain expressions for the intervals of time ΔT and comoving distance ΔD_c (line of sight) comprehended in the redshift interval $z \in [z_1, z_2]$:

$$\Delta D_C(z_1, z_2) = R_0 \Delta r(z_1, z_2) = D_{H,0}\int_{z_1}^{z_2} \frac{dz}{E(z)}, \tag{21}$$

$$\Delta T(z_1, z_2) = T_{H,0} \int_{z_1}^{z_2} \frac{dz}{(1+z)E(z)}, \tag{22}$$

$$D_C(z) \equiv \Delta D_C(0, z), \tag{23}$$

$$T_{LB}(z) \equiv \Delta T(0, z), \tag{24}$$

$$T_{AU}(z) \equiv \Delta T(z, \infty). \tag{25}$$

Here we have defined the line of sight comoving distance $D_C(z)$, look back time $T_{LB}(z)$ and age of the universe $T_{AU}(z)$. For the current values of Ω_i, the line of sight comoving distance has a finite maximum at infinite redshift, as photons meeting a particle at $z = 0$ have a limited time to travel since the beginning of the universe ($T_{AU}(z = 0) = 13.71$ Gyr). In fact, $D_{hor}(z = 0) \equiv D_c(z = \infty) = 1.428 \times 10^4$Mpc is the current value of the so called particle horizon, outside of which the events are not connected causally with the particle at redshift $z = 0$.

On the other hand, from Eq. 2, the transverse physical size ΔL of an object is connected to its angular diameter $\Delta\theta$ by means of $\Delta L = R(z)S_k(\Delta r(0, z))\Delta\theta$. Hence, the angular diameter distance $D_A(z)$ is given by

$$D_A(z) = \frac{\Delta L}{\Delta\theta} = \frac{D_{H,0}}{(1+z)|\Omega_{k,0}|^{\frac{1}{2}}} S_k\left(\frac{D_C|\Omega_{k,0}|^{\frac{1}{2}}}{D_{H,0}}\right). \tag{26}$$

From Eq. 26 another distance scale, the transverse comoving distance, can be defined:

$$D_M(z) = (1+z)D_A(z). \tag{27}$$

In a more general way, we can calculate the comoving distance between events at different redshifts and angular separation θ in the sky. Straight from Eq. 2, we have that the line element $dL^2 = R_0^2(dr^2 + S_k^2(r)d\theta^2)$ is defined on the surface of a 3-sphere, 3-hyperboloid or 3-plane. Therefore, the comoving separation distance between points 1 and 2 is

$$\Delta L = R_0 r_{12}, \tag{28}$$

where by using the law of cosines in spherical ($k = 1$) and hyperbolic (k=-1) geometry we have respectively (Peacock 1999; Liske 2000)

$$\cos r_{12} = \cos r_1 \cos r_2 + \sin r_1 \sin r_2 \cos\theta, \tag{29}$$

$$\cosh r_{12} = \cosh r_1 \cosh r_2 - \sinh r_1 \sinh r_2 \cos\theta. \tag{30}$$

The angles $r_1 = \Delta r(0, z_1)$ and $r_2 = \Delta r(0, z_2)$ are related to the line of sight comoving distance by means of Eq. 21. In the case of flat geometry ($k = 0$), we have

$$\Delta L^2 = D_C^2(z_1) + D_C^2(z_2) - 2D_C(z_1)D_C(z_2)\cos\theta. \tag{31}$$

The differential of volume is $dV = 4\pi R^2(t) S_k^2(r) dr$, which under integration leads to (Martinez & Saar 2002)

$$V_C(z) = 2\pi R_0^3[r - \sin(r)\cos(r)] \quad (k = 1), \tag{32}$$

$$V_C(z) = 2\pi R_0^3[\sinh(r)\cosh(r) - r] \quad (k = -1), \tag{33}$$

$$V_C(z) = \tfrac{4}{3}\pi D_C^3(z) \quad (k = 0), \tag{34}$$

where $r = \Delta r(0, z)$.

With respect to magnitudes, the distance to the light source has to be redefined in order to adjust for changes due to the expansion universe. In fact, the total luminosity per frequency at a redshift z is measured as a total flux density per frequency $S = L/(4\pi D_L^2(z))$ received at $z = 0$, where the luminosity distance $D_L(z)$ takes into account the changes in the frequency of the light, the arrival rate and energy and the change in the spectral bandwidth (Peacock 1999). This leads to

$$D_L(z) = D_M(1 + z) = D_A(1 + z)^2. \tag{35}$$

Hence, we can define the distance modulus DM (Hogg 2000) as

$$DM \equiv m - M = 5\log[D_L(z)/1\text{Mpc}] + 25, \tag{36}$$

where M is the absolute magnitude, m is the apparent magnitude (extinction and k-corrected) (Hogg et al. 2002). Usually, DM is used a proxy for D_L.

3. Database Integration and Functions Implementation

The library is written in $C\#$ under the .NET framework. In this environment, any .NET programming language is compiled into a lower common intermediate language (CLI) and then executed by the .NET run time. Starting 2005, MS-SQL Server incorporates this Common Language Runtime (CLR), allowing the user to create his own code written in any .NET language, then upload the dll or assembly into a database and execute the ported code as simple T-SQL commands. This includes, amongst others, user defined functions (UDFs), table valued functions (UDTVFs), stored procedures (UDSPs) and aggregate functions (UDAFs), which can handle calculations of higher complexity and faster than what can be done with a T-SQL routine. The SQL scripts for uploading the code into a database are given in two kinds, depending on the version of MS-SQL Server (2005 or 2008). All ported code and examples can be found in Appendixes A.1, A.2, A.3, B.1 and B.2.

3.1. Cosmological Functions

The class containing the cosmological functions can be first instantiated with customizable cosmological and auxiliary numerical parameters, allowing the execution of functions either with

high precision or high speed. The SQL-CLR functions $F(z)$ that comprehend all distances, times and volume as a function of redshift (Eqs. (18), (19), (23)-(28), (31)-(36)) are implemented in two versions, either with fixed (from Eq. 17) or free cosmological parameters (names starting with `"fCosmf"` and `"fCosmo"` respectively). The user can modify the code as necessary or adding new features, such as a time evolving state equation for dark matter. Figure 1 shows the normalized functions $F(z)$ up to redshift $z = 2.5$, obtained from executing the `fCosmfQuantities` UDTVF. This function returns the cosmological functions evaluated at a grid of redshifts. At redshift z_i, part of the integrals $I(0, z_i) = I(0, z_{i-1}) + I(z_{i-1}, z_i)$ in Eqs. (21) and (22) have already been calculated in the previous redshift intervals. As a result, the quantities at each redshift can be streamed (using C#'s "yield return") with no extra overhead and grid memory allocation.

In order to calculate the aforementioned integrals, numerical integration is implemented through the Romberg method, which evaluates (without extra overhead) a progressively finer grid for the trapezium method within the interval of integration (Press et al. 1992). Extrapolation of the integral's value is performed to the case of zero grid separation by means of Neville extrapolation (Press et al. 1992). The use of an open interval trapezium method is necessary when integrating in the interval $z \in [0, \infty[$, as well as the change of variable $z \to z^{-1}$. The fractional precision of the integrals is by default set to $\Delta \leq 10^{-9}$.

For obtaining the inverses $Z(F) \equiv F^{-1}(F(z))$ of the cosmological quantities, i.e. the redshift at particular values of distances, volume or times, we use a root finding method that includes Newton-Raphson method (Press et al. 1992), Neville interpolation and simple bracketing. The first iteration starts by using the Newton-Raphson algorithm, with a seed defined as the middle point in the interval of length L_1 that brackets a root (or redshift). If the new calculated redshift lies inside the interval, a new smaller interval is selected and a new iteration is made. If the redshift lies outside the bracket, we partition the original bracket in 4 intervals and use Neville interpolation to compute a new root. A smaller root-containing bracket is defined as plus minus the error in the redshift that results from Neville's method. If the error interval is bigger than the initial bracket, we find a new smaller bracket using the 5 redshifts values where the function was already evaluated when performing Newton-Raphson algorithm and Neville interpolation. Then a new iteration is started. Once this method has bracketed a root, it will never lose the solution, and stops when either the root or the function at the root have reached a fractional precision of $\Delta \leq 10^{-9}$.

The convergence rate of the root finding algorithm is defined by the exponent n in the relation $L_i = K \times L_{i+1}^n$, where K is a constant and i indexes the iterations (Press et al. 1992). In the best case, the algorithm has quadratic convergence ($n = 2$) when Newton-Raphson dominates, and in the worst case it's linear ($n = 1$) when Neville's method ($K \gg 1$) or simple bracketing ($K = 4$) dominate. Figure 2 shows the actual fractional precision $\Delta = |Z(F) - z|/z$ reached by the root finding method. The probability distribution $P(\log \Delta)$ in Figure 3 shows a significant accumulation at $\Delta \sim 10^{-16} - 10^{-14}$ and in $\Delta = 0$ as well, both far from the imposed $\Delta = 10^{-9}$ cutoff, which suggests that Newton-Raphson dominates at least in the last iteration.

3.2. Binning Functions and Procedures

Simple data exploratory analysis tools include the function `fMathBin`, which returns the bin centers where the data points fall in, given a one dimensional user defined grid. The function `fMathGrid` returns a column table with a user defined grid, either in linear or logarithmic scale, and can be used together with `fMathBin` and the `group by` clause in order to construct histograms or any other aggregate related statistic (see Appendix B.1) . The bin and grid values are rounded to a fractional precision of $\Delta = 10^{-14}$. This eliminates numerical error created when constructing the grid, allowing clean `join` clauses between tables on values generated by these functions. Fractional precision rounding can be implemented with `fMathRound`, which rounds a value to a particular number of significant digits. Midpoint rounding of type `"ToEven"` is applied to reduce bias.

The advanced UDSP `spMathHistogramNDim` returns a table with an N-dimensional histogram, given data points distributed in N columns returned by a user defined query. In order to avoid SQL-injection attacks, the query is filtered out using the `fMathSafeSqlCommand` UDF (see Section 3.4). The normalized-to-one probability distribution is also returned, which considers optional multiplicative weighting of the data points. This UDSP can be of use when computing a color-magnitude diagram under the V_{MAX} method, as shown in Section 4.1 and Appendix B.1.

3.3. Statistical Aggregate Functions

Basic statistical aggregates implemented are the quantile function and its inverse, the cumulative distribution function (CDF). In MS-SQL Server 2005, the total serialization in a CLR UDAF is limited to 8000 Bytes, which cripples the capabilities of a CLR aggregate that stores and sorts a large amount of values (Coles 2008). However, the existence of ranking functions such as `ROW_NUMBER()` allow the creation of UDFs that implement these statistical aggregates. In *CfunBASE*, the function `fMathCDF` consists of a SQL script that uses `ROW_NUMBER()` and returns the (linearly interpolated) CDF, given a set of values returned as a column by a user defined query. On the other hand, its inverse `fMathQuantile` gets the (linearly interpolated) quantile function. SQL-injection attacks are prevented by using `fMathSafeSqlCommand` (see Section 3.4).

In MS-SQL Server 2008, the absence of a 8000 bytes serialization limit, together with multiple-input-capable aggregates, allow the creation of simpler-syntax CLR UDAFs such as `aMathQuantile` and `aMathCDF`. A drawback is the fact that serialization methods (i.e. `read()` and `write()`) have to be implemented in the CLR code, instead of using the SQL native serialization. This makes this aggregates slower than the UDFs in a factor 2-5 (tested for 10^6 records).

3.4. Security Related Functions

Submitted queries have the potential to affect the server negatively. In fact, SQL injection attacks exploit the security breach in systems where the user's query is not filtered out of harmful commands. CasJobs sets particular user privileges and checks any query against a list of commands, keywords or characters known to be potentially dangerous by using the `spExecuteSafeSQL` UDSP, on which *CfunBASE*'s `fMathSafeSqlCommand` is based. The latter returns a safe command, that can be passed to any function or procedure that requires executing it (note that UDSPs can not be executed inside UDFs or UDTVFs). The maximum number of rows returned is tentatively set to be 10^7.

4. Advanced Examples

The cosmological functions bring up the view of a 3-dimensional universe and its evolution through time, in which galaxies can be studied according to their close environment, intrinsic luminosity, colors, spectral quantities, etc. Here we show basic features of interest that can be easily implemented, including luminosity functions and the friends-of friends algorithm for galaxy groups identification. The testing of *CfunBASE* and these advanced examples was performed on a GrayWulf cluster (Szalay et al. 2009a; Simmhan et al. 2009), designed for fulfilling the specific needs of data intensive computing. It holds an SDSS DR7 database managed by MS-SQL Server 2008, running under Windows HPC server 2008.

4.1. Luminosity Function

The luminosity function (LF) $\Phi(M)$, i.e the galaxy number density per absolute magnitude M, can be easily estimated using the $V_{\rm Max}$ method (Schmidt 1968). Basically, it creates a weighted histogram of the different values of M. The weights $w_i = 1/V_{{\rm Max},i}$ are defined by the maximum volume $V_{{\rm Max},i}$ inside which the ith galaxy can be observed, given the apparent magnitude and redshift cuts of the survey and the fractional sky covering area F_A of the footprint. The weighting takes care of the fact that the density of the less luminous galaxies becomes very low at high redshifts in flux limited surveys, phenomena known as Malmquist bias. A limitation is that it assumes that the galaxies are evenly distributed in the sky, which is not the case. CasJobs offers access the SDSS DR7 Main Galaxy Sample (MGS) (Strauss et al. 2002; Abazajian et al. 2009), which is flux limited by an r-band petrosian apparent magnitude cut of $m_r \lesssim 17.77$, and whose selection function (or redshift distribution) peaks at $z \simeq 0.1$. We can construct a smaller sample defined by the intervals $z \in [z_1 = 0.04, z_2 = 0.12]$ and $m_r \in [m_1 = 13.5, m_2 = 17.77]$. If the ith galaxy of apparent magnitude m at a luminosity distance D_L were to have limiting apparent magnitudes $m_{1,2}$, we obtain from Eq. 36 that it should be moved to a limiting luminosity distance

$D_{L;m_{1,2}}$ given by

$$\begin{aligned} D_{L;m_{1,2}} &\equiv D_L(z_{\rm lim}; m_{1,2}) \\ &= D_L \times 10^{(m_{1,2}-k(z_{\rm lim})-m+k)/5}. \end{aligned} \tag{37}$$

Hence, the maximum volume is defined by the biggest interval of D_L inside which a galaxy can appear in the survey:

$$\begin{aligned} V_{{\rm Max},i} &= [V(\min(D_L(z_2), D_{L;m_2})) \\ &- V(\max(D_L(z_1), D_{L;m_1}))] \times F_A. \end{aligned} \tag{38}$$

Note that Eq. 37 defines $z_{\rm lim}$ in an implicit way. In order to solve for it iteratively, the redshift dependent k-correction $k(z_{\rm lim})$ has to be known. This would need additional information related to the galaxy's spectral type, such as a decomposition into a linear combination of spectral templates. As a rough approximation, we can use $k \simeq k(z_{\rm lim})$. The galaxies whose m is bright enough for having $V_{\rm Max}$ equal to the survey's volume $V_{\rm S} = [V(z_2) - V(z_1)] \times F_A$ are considered a volume limited sample, and can be used for further studies without caring about the $V_{\rm Max}$ weighting. The LF $\Phi(M)$ and its Poisson error $\Delta\Phi(M)$ are estimated at the center of absolute magnitude bins of width ΔM, having the form

$$\Phi(M) = \frac{1}{\Delta M} \sum_{i={\rm galaxies}} \frac{c_i}{V_{{\rm Max},i}}, \tag{39}$$

$$\Delta\Phi(M) = \frac{1}{\Delta M} \sqrt{\sum_{i={\rm galaxies}} \left[\frac{c_i}{V_{{\rm Max},i}}\right]^2}, \tag{40}$$

where the sum is performed on the galaxies located in the corresponding bins. The factor c_i takes into account possible incompleteness in the sample. In this case, we choose $c_i = 1$ for simplicity.

Appendix B.1 shows the T-SQL implementation of the LF based on the previous magnitude and redshift cuts and other constraints, sampling $\simeq 270000$ MGS galaxies from DR7, where the area of the spectroscopic survey amounts to 7932.125 deg^2 ($\simeq 19\%$ of the sky). The result can be seen in Fig. 4, where both the LF and the color magnitude diagram (CMD) are shown. The LF obtained by the $V_{\rm Max}$ method uses the function `fMathBin` to aggregate the $1/V_{\rm Max}$ weights in the absolute magnitude M_r bins, measured in petrosian magnitudes. The color $C_{ur} \equiv u_{model} - r_{model} \equiv u - r$ uses model magnitudes, which proves to be a better separator for the blue and red galaxy clumps in the CMD (Baldry et al. 2004). The CMD is basically a weighted 2-dimensional histogram, where the weights are also $1/V_{\rm Max}$, and can be obtained by means of executing `spMathHistogramNDim`.

4.2. Friends-of-Friends Galaxy Cluster Identification Algorithm

The knowledge of the spatial distribution of galaxies is a relevant issue. Measurements of the 3-D correlation function have shown that galaxies, as a collective group, distribute in space according

closely to a power law, where red and blue galaxies present different clustering (e.g. Budavari et al. (2003)). At a more detailed level, various methods have been used for identifying members of galaxy clusters and groups, allowing studies of galaxy properties depending on surrounding density and environment. One widespread method is the friends-of-friends algorithm (FoF) (Huchra & Geller 1982), which is shown implemented in Appendix B.2. The algorithm starts with a galaxy labeled as the first group member, and includes as its group members all the immediate neighbors closer than predetermined linking comoving distances, both perpendicular ($D_{\mathrm{Link},\perp}$) and parallel ($D_{\mathrm{Link},\parallel}$) to the line of sight. The process is repeated recursively on new neighbors of the neighbors, until no more new neighboring galaxies are found. Then a new galaxy is picked and the whole process is again repeated. Following Berlind et al. (2006), for a flat universe, $z \lesssim 0.1$ and a volume limited sample, we define the maximum values for the comoving distances $D_\perp$ and $D_\parallel$ between 2 objects separated at an angular separation θ:

$$\begin{aligned} D_\perp &= (D_c(z_1) + D_c(z_2)) \sin(\theta/2) \\ &\leq D_{\mathrm{Link},\perp} \equiv b_\perp \bar{n}^{-1/3} \end{aligned} \tag{41}$$

$$D_\parallel = |D_c(z_1) - D_c(z_2)| \leq D_{\mathrm{Link},\parallel} \equiv b_\parallel \bar{n}^{-1/3} \tag{42}$$

where $\bar{n}$ is the average count density of galaxies. The values $b_\perp$ and $b_\parallel$ are chosen to find a balance between including field galaxies as group members, and finding just the cluster's core members. Also, the galaxy peculiar velocities around clusters have to be taken into account, as they create distortion in redshift space (fingers of God, pancakes of God, etc.). Since we use a volume limited sample, the linking distances are not weighted by the selection function, but remain constant across different redshift values. A simple solution is to take a cylindrical searching volume around a galaxy, with parameters $b_\perp = 0.14$ and $b_\parallel = 0.75$ (Berlind et al. 2006). In our sample we have $\bar{n} = 0.001283$ Mpc^{-3}, which leads to $D_{\mathrm{Link},\perp} = 6.902$ Mpc and $D_{\mathrm{Link},\parallel} = 1.288$ Mpc.

Finding the immediate neighbors of N galaxies is greatly eased by clustered indexing of the spatial coordinates. Brute force spatial searches are expensive ($O(N)$), hence the importance of identifying and storing the table rows on disk in a hierarchical schema according to the spatial distribution of objects, thus decreasing searching times. The script in Appendix B.2 stores the group labels of galaxies in the `Groups` table, which is filled up by running the FoF algorithm on the table `NeighborsLSS`. The latter contains the neighbors of all galaxies based on the criteria shown in Eqs. 42 and 41, being created out from spatial searches based on the *ra*,*dec* and redshift values given in table `DR7smallLSS`. Note that although the neighbors of each galaxy can also be retrieved "on the fly" in each step of the FoF algorithm from internal nested queries to `DR7smallLSS`, this performs however slower than accessing this information right from an indexed `Neighbors` table. This method might be nonetheless of practical use when storing all the neighbors into a table becomes expensive, in the case of having too many objects ($> 10^9$) in `DR7smallLSS`, as is expected in next generation all sky surveys.

A simple custom spatial indexing schema for table `DR7smallLSS` is to create a 3-dimensional grid in space, whose cell side length is the longest distance between $D_{\mathrm{Link},\perp}$ and $D_{\mathrm{Link},\parallel}$. Using

the function `fMathBin`, we can index each galaxy according to the 3-dimensional grid inside which they fall. This leads us to 3 grid cell center values (`Gx,Gy,Gz`) for each galaxy, that together with the `SpecObjID` can define a primary key or unique clustered index, i.e, an spatial index on table rows stored sequentially on disk according to the following nested order of unique values: (`Gx,Gy,Gz,SpecObjID`). This greatly reduces searching time ($O(\log N)$) when using the `BETWEEN` clause in searches around the x, y, z comoving coordinates of galaxies. A further improvement can be the use of an Octree data structure, where a cube in space is divided recursively into 8 identical cubes. A spatial index can be constructed by hashing together successive node IDs from the different node levels, up to the level whose side length is the longest between $D_{\mathrm{Link},\perp}$ and $D_{\mathrm{Link},\parallel}$. R-trees (Guttman 1984) could be implemented as well. It might be the case that the original table with 3-D coordinates of the galaxies becomes too big, or has already a primary key. In that case, we can define a unique non clustered index in the previously binned spatial coordinates, or, if the table is static (as usual in astronomical databases), we can even create a view of the table that includes this index. In this case, however, the spatial searches might run a bit slower.

Another spatial searching possibility is to use native spherical indexing in MS-SQL server 2008, which is implemented by means of the `Geography` type and associated to the unique clustered index of the table. This type is defined based on a constant latitude/constant longitude tessellation of the sphere. It takes as input the ra, dec coordinates and a spatial reference system identifier (SRID) definition of the earth [5], allowing the calculation of physical distances between points on the earth's surface, but running however much slower than the previous grid based method. By default, the database carries different SRID of the Earth's shape and size, which implement only ellipsoidal geometry (i.e. a non constant radius). This leads to inconsistencies when calculating the separation angles between galaxies at different positions in the sky. The user can therefore use this type only if spherical geometry based SRIDs are uploaded.

More sophisticated methods include different sphere tessellation schemas based on nested tree data structures that define the cells, such as HTM (Szalay et al. 2005), HEALPix (Górski et al. 2005), IGLOO (Crittenden 2000). A clustered index composed by the nested cell IDs of the galaxies is enough for spatial searches, since it's trivial to compute the neighboring cell's IDs based on the cell ID where a galaxy falls in. Currently, CasJobs includes an HTM library with functions that greatly facilitate spatial searches.

The spatial galaxy distribution shown in Figure 5 is obtained from running the FoF algorithm on the volume limited sample specified in Section 4.1. Only galaxies in the northern galactic cap of the DR7 footprint are shown. The groups of 10 or more galaxy members are highlighted by drawing 7 Mpc semitransparent spheres around each galaxy. Finger of God like structures can thus be easily identified.

[5] See definitions by the European Petroleum Survey Group standard

5. Conclusion

As soon as new generation surveys release their data products, astronomers might be forced to work inside databases, and even to write their own customized code and upload it into "the cloud". In extragalactic astronomy, the trend of mapping the large scale structure with precision multiband photometric redshifts will bring $N \geq 10^{10}$ redshift measurements, which require fast specific database integrated software, implementing algorithms not slower than $O(N \log N)$ (Szalay et al. 2002). *CfunBASE* might fill some gaps, by being customized for either precision or speed, being expanded as new functions are required, and being ported into other DBMSs, such as MySQL, Oracle, PostgreSQL or SciDB if necessary. In the future, a need for an expanded Structured Query Language for scientific computing might arise, where data types and mathematical expressions of higher complexity (such as linear algebra, calculus, statistical functions) are integrated as a natural part of the language. If the market is big enough, this would be the replacement of IDL, R, Python and other high level languages widely used in the scientific community, but that need the entire dataset to be first stored in the available RAM/virtual memory or might run slowly when dealing with big datasets.

The author is grateful to Mark Neyrinck for reviewing the paper, and to Alex Szalay, Tamas Budavári, Ching-Wa Yip, Sebastien Heinis and Miguel Aragon-Calvo for useful advice. The Gray-Wulf cluster was financially supported by the Gordon and Betty Moore Foundation, Microsoft Research and the Pan-STARRS project.

A. List of CLR Functions

A.1. List of Cosmological Functions.

The cosmological functions shown here are the ones with fixed cosmological parameters drawn from Eq. 17. The names of the ones with free parameters start with "`fCosmo`" instead of "`fCosmf`", and have the extra input parameters added at the end of the function declaration:

", @OmegaM float=0.2739, @OmegaL float=0.726, @OmegaR float=1e-4, @w_0 float=-1, @h_0 float=0.705)"
Note that a float type in T-SQL corresponds to double precision in C#.

1. `fCosmfDl`(@z float)

   Returns the luminosity distance [Mpc] at a given redshift.

2. `fCosmfDc`(@z float)

   Returns the line of sight comoving distance [Mpc] at a given redshift.

3. `fCosmfDcInterval`(@Zmin float, @Zmax float)

   Returns the line of sight comoving distance [Mpc] comprehended in the redshift interval [@Zmin,@Zmax].

4. `fCosmfDa`(@z float)

   Returns the angular diameter distance [Mpc] at a given redshift.

5. `fCosmfDm`(@z float)

   Returns the transverse comoving distance [Mpc] at a given redshift.

6. `fCosmfComovingVolume`(@z float)

   Returns the comoving volume [Mpc^3] between here and a given redshift.

7. `fCosmfAbsMag`(@m float, @z float)

   Returns the absolute magnitude of a galaxy at a particular redshift.

8. `fCosmfDistanceModulus`(@z float)

   Returns the distance modulus at a particular redshift.

9. `fCosmfQuantities`(@zMin float, @zMax float, @NumBin int)

   Returns a table with the midpoints values of a grid of redshifts, together with their corresponding values of the cosmological distances,comoving volume and time intervals.

10. `fCosmfLookBackTime`(@z float)

    Returns the time interval [Gyr] between the present time and a particular redshift.

11. `fCosmfAgeOfUniverse`(@z float)

    Returns the time interval [Gyr] between a particular redshift and the beginning of the universe.

12. `fCosmfTimeInterval`(@zMin float, @zMax float)

    Returns the time interval [Gyr] between redshifts zMin and zMax.

13. `fCosmfHubbleDistance`(@z float)

    Returns the Hubble Distance [Mpc] at a particular redshift.

14. `fCosmfHubbleTime`(@z float)

    Returns the Hubble Time [Gyr] at a particular redshift.

15. `fCosmfComovDist2Objects`(@Redshift1 float, @Redshift2 float, @AngularSeparation float)

    Returns the comoving distance [Mpc] between 2 objects at different redshifts and locations in the sphere.

16. `fMathAngSepXYZ`(@x1 float, @y1 float, @z1 float, @x2 float, @y2 float, @z2 float)

    Returns the angular separation (in radians) between 2 points in Cartesian coordinates.

17. `fMathAngSepRADEC`(@Ra1 float, @Dec1 float, @Ra2 float, @Dec2 float)

    Returns the angular separation (in radians) between 2 points in Equatorial coordinates.

18. `fCosmfZfromDl`(@LuminosityDistance float)

    Returns the redshift at a given luminosity distance [Mpc].

19. `fCosmfZfromDa`(@AngularDiamDist float)

    Returns a row with the first and second solution for the redshifts at a given angular diameter distance [Mpc].

20. `fCosmfZfromDm`(@ComovDistTransverse float)

    Returns the redshift for a given transverse comoving distance [Mpc].

21. `fCosmfZfromDc`(@ComovDistLineOfSight float)

    Returns the redshift at a given line of sight comoving distance [Mpc].

22. `fCosmfZfromAgeOfUniverse`(@AgeOfUniverse float)

    Returns the redshift at a given age of the universe [Gyr].

23. `fCosmfZfromLookBackTime`(@LookBackTime float)

    Returns the redshift at a given look back time [Gyr].

24. `fCosmfZfromComovVolume`(@ComovVolume float)

    Returns the redshift at a given comoving volume [Mpc^3].

25. `fCosmfComovVolumeFromDl`(@LumDistance float)

    Returns the comoving volume at a given luminosity distance [Mpc].

26. `fCosmfZfromTh`(@HubbleTime float)

    Returns the redshift at a given Hubble time [Gyr].

27. `fCosmfZfromDh`(@HubbleDistance float)

    Returns the redshift at a given Hubble distance [Mpc].

A.2. List of Basic Data Exploratory Analysis and Statistical Tools.

1. `fMathGrid`(@x1 float, @x2 float, @NumBin int, @IsLinearScale bit, @IsMidPoints bit)

   Returns a column containing the tick marks (either interval boundaries or midpoints) that define a grid of @NumBin bins in the interval [@x1,@x2]. The scale is either linear or logarithmic.

2. `fMathBin`(@x float, @x1 float, @x2 float , @NumBin int, @HasOpenUpperBound bit, @IsLinearScale bit)

   Returns the bin's center where @x falls in, given a grid of @NumBin bins in the interval [@x1,@x2]. The scale is either linear or logarithmic. Each bin can have open (or closed) upper (or lower) bounds.

3. `fMathRound`(@x float, @SigPlace int)

   Rounds a value up to the first @SigDigits significant digits.

4. `spMathHistogramNDim`(@Query nvarchar(512), @Dimension int, @String_X1 nvarchar(128), @String_X2 nvarchar(128), @String_NumCell nvarchar(128), @HasOpenUpperBound bit)

   Returns an N-dimensional histogram (with optional weighting) of N-dimensional data points returned by a user defined select statement.

5. `aMathCDF`(@AggregatedValues float, @Value float)

   Aggregate function that returns the cumulative distribution function (CDF) at a @value, given a column of values @AggregatedValues. If the value is not in the column of values, linear interpolation is performed.

6. `aMathQuantile`(@AggregatedValues float, @CDFvalue float)

   Aggregate function that returns the inverse of the cumulative distribution function (CDF), also called Quantile Function; evaluated at a value of the CDF @CDFvalue in the interval [0,1]. The aggregate works on a column of values @AggregatedValues, on which linear interpolation is performed.

7. `fMathCDF`(@Query nvarchar(256), @Value float)

   Returns the cumulative distribution function (CDF) at a value @Value, given a column of values returned by a user defined query @Query. If the value is not in the column of values, linear interpolation is performed.

8. `fMathQuantile`(@Query nvarchar(256), @CDFvalue float)

   Returns the inverse of the cumulative distribution function (CDF), also called Quantile Function; evaluated at a value @CDFvalue of the CDF in the interval [0,1]. The functions works on a column of values returned by a user defined query @Query. Linear interpolation between the column values is performed.

A.3. List of Utility Functions

1. fMathSafeSqlCommand(@cmd VARCHAR(8000), @limit INT = 1000)

   Parses and checks the command @cmd against SQL-injection. Returns a string with a safe version of the command, enforced to return a maximum of @limit rows. Based on spExecuteSQL from
   http://casjobs.sdss.org/dr7/en/help/browser/browser.asp.

2. fMathReplace(@oldstr VARCHAR(8000), @pattern VARCHAR(1000), @replacement VARCHAR(1000))

   Case-insensitive string replacement. Identical to fReplace from http://casjobs.sdss.org/dr7/en/help/browser/browser.asp.

3. fMathIsNumbers(@string varchar(8000), @start int, @stop int)

   Checks that the substring is a valid number. Identical to fIsNumbers from http://casjobs.sdss.org/dr7/en/help/browser/browser.asp.

B. Advanced Examples

Refer to http://www.skyserver.org/cfunbase/ for the T-SQL code that implements these advanced examples.

B.1. Galaxy Luminosity Function and Color Magnitude Diagram (CMD)

This T-SQL script creates a small MGS and computes its luminosity function and color magnitude diagram. Refer to Section 4.1 for details.

```
--Calculating the spectroscopic survey area:
SELECT SUM(area) FROM BestDR7..Region
WHERE type='SECTOR' -- AreaSurvey = 7932.12522550822 = 7932.125 SqDeg

--Setting up the redshift, luminosity distance and apparent magnitude cuts:
DECLARE @z_1 float, @z_2 float, @Dlum_1 float, @Dlum_2 float, @m1im_1 float, @mlim_2 float,
        @FracVolume float, @AreASurvey float, @Vsurvey float
SET @z_1 = 0.04;
SET @z_2 = 0.12;
SET @Dlum_1=dbo.fCosmfDl(@z_1); print @Dlum_1
SET @Dlum_2=dbo.fCosmfDl(@z_2); print @Dlum_2
SET @m1im_1 = 13.50
```

```
SET @mlim_2 = 17.77
-- The fraction of the entire sky volume occupied by the survey footprint:
SET @FracVolume = 0.1922801 -- @FracVolume = AreaSurvey / ( 4*PI()*power((180/PI()),2) )

-- Creating small table with galaxy sample:
CREATE TABLE DR7small(
 SpecObjID bigint primary key not null, RA float not null, DEC float not null,
 z float not null, m_r float not null, Vmax float not null, Color_u_r float not null
)

INSERT DR7small
SELECT
 specobjid, ra, dec, z, petromag_r-extinction_r as m_r,
 (dbo.fCosmfComovVolumeFromDl(
   CASE
   WHEN @Dlum_2 < dbo.fcosmfDl(z)*POWER(10.0,(@mlim_2-(petromag_r-extinction_r))/5.0)
     THEN
       @Dlum_2
     ELSE
       dbo.fcosmfDl(z)*POWER(10.0,(@mlim_2-(petromag_r-extinction_r))/5.0)
   END)-
 dbo.fCosmfComovVolumeFromDl(
   CASE
   WHEN @Dlum_1 > dbo.fcosmfDl(z)*POWER(10.0,(@m1im_1-(petromag_r-extinction_r))/5.0)
     THEN
       @Dlum_1
     ELSE
       dbo.fcosmfDl(z)*POWER(10.0,(@m1im_1-(petromag_r-extinction_r))/5.0)
   END) )*@FracVolume AS Vmax,
 (modelMag_u-extinction_u)-(modelMag_r-extinction_r) AS Color_u_r
FROM BestDR7.dbo.specphoto
WHERE
z between @z_1 and @z_2
and primtarget&(64|128|256)!=0 -- This chooses the MGS galaxies
and petromag_r-extinction_r between @m1im_1 and @mlim_2
--add your custom photometric and spectroscopic constraints here.

-- Getting the Luminosity function and its error.
-- Absolute magnitude range is [-25,-15], with number of bins = 100
-- (bin size DeltaM = 0.1 magnitudes)
SELECT dbo.fMathBin(v.AbsMag_r,-25, -15, 100 ,1, 1) AS AbsMag,
sum(1/v.Vmax)/0.1 AS Phi,
sqrt(sum( 1/(v.Vmax*v.Vmax) ) )/0.1 AS PhiError,
count(*) AS counts
FROM ( SELECT dbo.fCosmfAbsMag(m_r,z) AS AbsMag_r, Vmax FROM DR7small) AS v
GROUP BY dbo.fMathBin(v.AbsMag_r,-25, -15, 100 ,1, 1)
```

```
ORDER BY dbo.fMathBin(v.AbsMag_r,-25, -15, 100 ,1, 1)

-- This gets the same luminosity function as a probability distribution, i.e.,
-- the area under it is unity.
EXECUTE spMathHistogramNDim 'SELECT dbo.fCosmfAbsMag(m_r,z), 1.0/(Vmax) FROM DR7small'
,1, '-25', '-15', '100' ,1

-- This gets the color magnitude diagram as a probability distribution, i.e.,
-- the area under it is unity. The color range is [0,5] with number of bins = 100.
EXECUTE spMathHistogramNDim 'SELECT dbo.fCosmfAbsMag(m_r,z),Color_u_r, 1.0/Vmax FROM DR7small'
,2, '-25,0', '-15,5', '100,100' ,1
```

B.2. Galaxy Groups Finding: The Friends-of-Friends Algorithm

This T-SQL script creates the NeighborsLSS table with the neighbors of the galaxies, and executes the FoF algorithm, storing the galaxies labeled by group ID in table Groups. Refer to Section 4.2 for details.

```
--This table stores spatial information:
CREATE TABLE DR7smallLSS(
  SpecObjID bigint not null, RA float not null, DEC float not null, z float not null,
  Gx float not null, Gy float not null, Gz float not null
--If native MS-SQL server 2008 geospatial indexing is also wanted, add the following line:
--,GeographyCol geography not null
)
-- This creates a unique clustered index or primary key with spatial information:
ALTER TABLE DR7smallLSS ADD PRIMARY KEY (Gx,Gy,Gz,SpecObjID);

INSERT DR7smallLSS
SELECT
  specobjid, ra, dec , z,
-- The following gets the midpoints of the 3-dimensional grid cells. Note that
-- the cell side length is rounded to 7Mpc, having 80 cell divisions per dimension.
  dbo.fMathBin(dbo.fCosmfDc(z)*SIN(radians(90.0-dec))*COS(radians(RA)),0,560,80,1,1),
  dbo.fMathBin(dbo.fCosmfDc(z)*SIN(radians(90.0-dec))*SIN(radians(RA)),0,560,80,1,1),
  dbo.fMathBin(dbo.fCosmfDc(z)*COS(radians(90.0-dec)),0,560,80,1,1)
--If native SQL 2008 server geospatial indexing is also wanted, add the following line:
--,GEOGRAPHY::Point(dec,ra,4326)
-- Note that 4326 is the SRID number of a custom Earth shape definition.
FROM DR7small
--Add the following line if a volume limited sample is wanted
--WHERE (dbo.fCosmfComovingVolume(0.12)-dbo.fCosmfComovingVolume(0.04))*0.1922801 <= Vmax
```

```
ORDER BY
dbo.fMathBin(dbo.fCosmfDc(z)*SIN(radians(90.0-dec))*COS(radians(RA)),0,560,80,1,1),
dbo.fMathBin(dbo.fCosmfDc(z)*SIN(radians(90.0-dec))*SIN(radians(RA)),0,560,80,1,1),
dbo.fMathBin(dbo.fCosmfDc(z)*COS(radians(90.0-dec)),0,560,80,1,1), specObjID


-- This creates the Neighbors table:
CREATE TABLE NeighborsLSS(
SpecObjID bigint not null, NeighborSpecObjID bigint not null
)
CREATE unique clustered index SpecObjIDNeighborSpecObjID ON
  NeighborsLSS(SpecObjID,NeighborSpecObjID)

INSERT NeighborsLSS
SELECT h1.Specobjid, h2.SpecObjID
FROM DR7smallLSS as h1, DR7smallLSS as h2
WHERE
-- The value 7.1 Mpc intead of 7 Mpc is chosen to avoid eventual problems with rounding:
h2.Gx between h1.Gx - 7.1 and h1.Gx + 7.1 and
h2.Gy between h1.Gy - 7.1 and h1.Gy + 7.1 and
h2.Gz between h1.Gz - 7.1 and h1.Gz + 7.1 and
-- If native SQL 2008 server geospatial indexing is also wanted,
-- erase the previous 3 lines and add the following line:
-- h1.GeographyRaDec.STDistance(h2.GeographyRaDec) <= (6360000.0)*0.00794
-- Note that 6360000 is a custom Earth radius in meters, and 0.00794 radians is the
-- angular size distance of 1.288 Mpc at redshift 0.04
and
-- The following defines the cylindrical volume around the galaxies:
(dbo.fCosmfDc(h1.z) + dbo.fCosmfDc(h2.z)) *
 SIN(dbo.fMathAngSepRADEC(h1.ra,h1.dec,h2.ra,h2.dec)/2.0) <= 1.288 and -- Transverse
ABS(dbo.fCosmfDc(h1.z)-dbo.fCosmfDc(h2.z)) <= 6.902 and -- Line of sight
h1.Specobjid != h2.SpecObjID
GROUP BY h1.Specobjid,h2.SpecObjID
ORDER BY h1.Specobjid,h2.SpecObjID
GO



IF OBJECT_ID('Groups') IS NOT NULL DROP TABLE Groups
IF EXISTS (SELECT name FROM tempdb..sysobjects WHERE name like '#temp%')
  DROP TABLE #temp
IF EXISTS (SELECT name FROM tempdb..sysobjects WHERE name like '#Remaining%')
  DROP TABLE #Remaining
IF EXISTS (SELECT name FROM tempdb..sysobjects WHERE name like '#PreviousNeighbors%')
  DROP TABLE #PreviousNeighbors
IF EXISTS (SELECT name FROM tempdb..sysobjects WHERE name like '#Neighbors%')
```

```
  DROP TABLE #Neighbors
IF EXISTS (SELECT name FROM tempdb..sysobjects WHERE name like '#NewNeighbors%')
  DROP TABLE #NewNeighbors

CREATE TABLE Groups(
SpecObjID bigint not null, GroupID int not null
)

-- This finds and labels into groups the galaxies that have no neighbors:
SELECT SpecObjID,IDENTITY(int,1,1) AS GroupID INTO #temp FROM DR7smallLSS
WHERE SpecObjID NOT IN (SELECT SpecObjID FROM NeighborsLSS)
ORDER BY SpecObjID
INSERT Groups SELECT * FROM #temp
DROP TABLE #temp
GO

-- #Remaining has the galaxies that have not been yet labeled:
CREATE TABLE #Remaining(SpecObjID bigint primary key not null)
INSERT #Remaining SELECT specobjid FROM DR7smallLSS
  WHERE SpecObjID NOT IN (SELECT SpecObjID FROM Groups) ORDER BY SpecObjID

-- #Neighbors stores the neighboring galaxies of the ones in table #PreviousNeighbors, and
-- #NewNeighbors stores the neighboring galaxies of the ones in table #Neighbors that
-- are not in table #PreviousNeighbors
CREATE TABLE #PreviousNeighbors(SpecObjID bigint not null)
CREATE TABLE #Neighbors(SpecObjID bigint not null)
CREATE TABLE #NewNeighbors(SpecObjID bigint not null)

--SET NOCOUNT ON

-- Starting the FoF algorithm:

DECLARE @SpecObjID bigint
DECLARE @i int, @counts int

-- @i stores the group label:
SELECT @i=COUNT(*)+1 FROM Groups

-- Getting the first galaxy:
SELECT TOP 1 @SpecObjID= specobjid FROM #Remaining
INSERT #PreviousNeighbors SELECT @SpecObjID
-- Finding the neighbors:
INSERT #Neighbors SELECT NeighborSpecObjID FROM NeighborsLSS WHERE SpecObjID = @SpecObjID
-- labeling the group:
INSERT Groups SELECT @SpecObjID,@i
--Updating #Remaining:
```

```
DELETE FROM #Remaining WHERE SpecObjID=@SpecObjID

--Starting the loop that finds new neighbors:
WHILE 1=1
BEGIN
  TRUNCATE TABLE #NewNeighbors
-- Getting the new neighbors of the neighbors:
  INSERT #NewNeighbors
  SELECT n1.NeighborSpecObjID
  FROM NeighborsLSS as n1 JOIN #Neighbors n2 on n1.SpecObjID=n2.SpecObjID
  WHERE n1.NeighborSpecObjID NOT IN
    (SELECT SpecObjID FROM #PreviousNeighbors UNION SELECT SpecObjID FROM #Neighbors)
  GROUP BY n1.NeighborSpecObjID

  SELECT @counts = COUNT(*) FROM #NewNeighbors
  IF (@counts>=1)-- If there are new neighbors, label current the group members and
                 -- set the new neighbors as the current neighbors:
    BEGIN
      INSERT Groups
      SELECT SpecObjID, @i FROM #Neighbors
      DELETE FROM #Remaining WHERE SpecObjID in (SELECT SpecObjID FROM #Neighbors)
      TRUNCATE TABLE #PreviousNeighbors
      INSERT #PreviousNeighbors SELECT SpecObjID FROM #Neighbors
      TRUNCATE TABLE #Neighbors
      INSERT #Neighbors SELECT SpecObjID FROM #NewNeighbors
    END
  ELSE -- If there are no new neighbors, store the group label and start with new galaxy:
    BEGIN
      INSERT Groups
      SELECT SpecObjID, @i FROM #Neighbors
      DELETE FROM #Remaining WHERE SpecObjID in (SELECT SpecObjID FROM #Neighbors)
      SET @i=@i+1
      SELECT TOP 1 @SpecObjID = specobjid FROM #Remaining
-- If all galaxies have been labeled, stop:
      IF NOT EXISTS (select top 1 specobjid from #Remaining)
        BREAK -- this exits the while loop
      TRUNCATE TABLE #PreviousNeighbors
      INSERT #PreviousNeighbors SELECT @SpecObjID
      TRUNCATE TABLE #Neighbors
      INSERT #Neighbors
        SELECT NeighborSpecObjID FROM NeighborsLSS WHERE SpecObjID = @SpecObjID
      INSERT Groups SELECT @SpecObjID,@i
      DELETE FROM #Remaining WHERE SpecObjID=@SpecObjID
    END
END
```

REFERENCES

Abell, P. et al. 2009, *LSST Science Book, Version 2.0*, arXiv:0912.0201v1

Abazajian, K. N. et al. 2009, ApJS, 182, 543A

Armbrust, M. et al. 2009, *Above the Clouds: A Berkeley View of Cloud Computing*, Berkeley Technical Report No. UCB/EECS-2009-28

Baldry, I. K. et al. 2004, ApJ, 600, 681

Bonometto, S. et al. 2002, *Modern Cosmology*, Institute of Physics Publishing, Bristol & Philadelphia

Bell, G., Hey, T. and Szalay, A.S. 2009, Science 323,5919,1297-1298

Berlind, A. et al. 2006, ApJS, 167, 1

Blanton M.R. et al. 2001, AJ, 121, 2358

Budavari, T. et al. 2003, ApJ, 595, 59

Coles, M. 2009, *Pro T-SQL 2008 Programmers Guide*, Apress, Berkely, CA, USA

Crittenden, R.G. 2000, ApL&C, 37, 377C

Dodelson, S. 2003, *Modern Cosmology*, Academic Press, Amsterdam, Netherlands

Górski, K. M. et al. 2005, ApJ, 622, 759

Guttman, A. 1984, ACM SIGMOD International Conference on Management of Data, pp. 47-57

Hey, T., Tansley, S., Tolle, K. 2009, *The Fourth Paradigm: Data Intensive Scientific Discovery*, Microsoft Research, Redmond, Washington

Hogg, D. 2000, astro-ph/9905116v4

Hogg, D. et al. 2002, astro-ph/0210394v1

Huchra, J.P., & Geller, M.J. 1982, ApJ, 257, 423

Ivezic, Z. et al. 2008, arXiv:0805.2366v1

Kitching, T.D. et al. 2009, arXiv:0901.3143

Kolda, T. et al. 2005, LLNL Report UCRL-TR-208926

Komatsu, E. et al. 2009, ApJS, 180, 330

Liske, J. 2000, MNRAS, 319, 557

Martinez, V. J. and Saar, E., 2002, *Statistics of the Galaxy Distribution*, Chapman and Hall/CRC Press, Boca Raton

Peacock, J. A. 1999, *Cosmological Physics*, Cambridge University Press, Cambridge

Peebles, P. J. E. 1993, *Principles of Physical Cosmology*, Princeton University Press, Princeton, New Jersey

Popp, R. and Poindexter, J. 2006, IEEE Security & Privacy, Published by The IEEE Computer Society, 1540-7993/06, p.18-27

Press, W. et al. 1992, *Numerical Recipes in C: The Art of Scientific Computing*, 2nd Edition, Cambridge University Press, Cambridge

Roehm, U. and Blakeley, J. 2009, arXiv:0909.1764

Schmidt, M. 1968, ApJ, 151, 393

Simmhan, J. et al. 2009, *GrayWulf: Scalable Software Architecture for Data Intensive Computing*, 42nd Hawaii International Conference on System Sciences

Singh, V. et al. 2006, Microsoft Research Tech. Rep. MSR TR-2006-190

Sun, L. et al. 2006, Chin. J. Astron. Astrophys, 6, 2, 155164

Szalay, A.S et al. 2001, Microsoft Research Tech. Rep. MSR-TR-2001-104

Szalay, A.S et al. 2002, Microsoft Research Tech. Rep. MSR-TR-2002-84

Szalay, A.S et al. 2005, Microsoft Research Tech. Rep. MSR-TR-2005-123

Szalay, A.S et al. 2009, *GrayWulf: Scalable Clustered Architecture for Data Intensive Computing*, 42nd Hawaii International Conference on System Sciences

Szalay, A.S et al. 2009, *Low-Power Amdahl-Balanced Blades for Data Intensive Computing*, SOSP Workshop on Power Aware Computing and Systems (HotPower '09).

Strauss, M.A. et al. 2002, AJ, 124, 1810

Wang, X. et al. 2009, MNRAS 394, 1775

Weinberg, S. 2008, *Cosmology*, Oxford University Press, Oxford

Wright, N. 2006, PASP, 118, 1711

York, D.G. et al. 2000, ApJ, 120, 1579

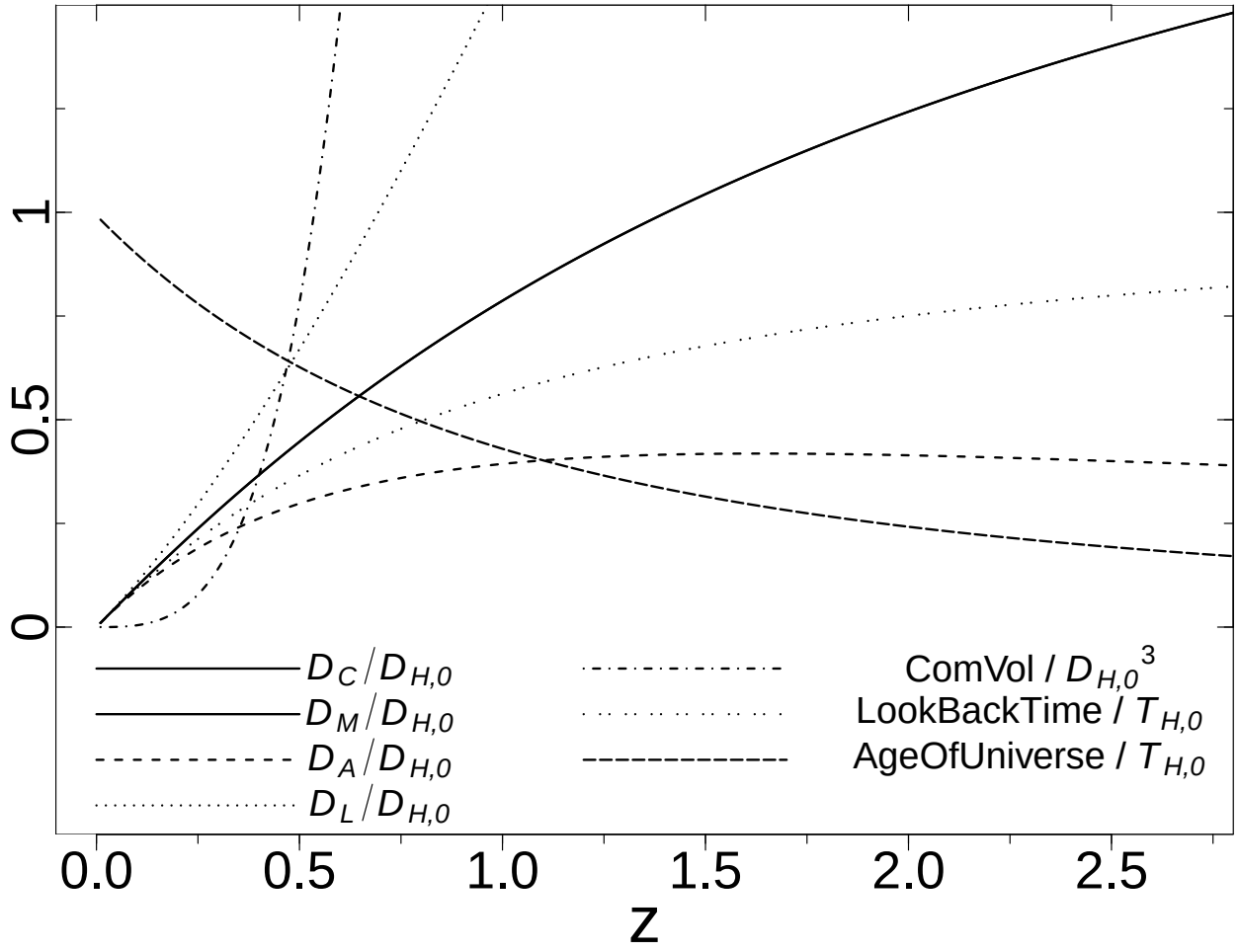

Fig. 1.— Normalized cosmological functions for the best cosmological parameters shown in Eq. 17. Note that this corresponds to a flat universe, in which $D_C(z) = D_M(z)$. Note that, for $z \lesssim 0.1$, the straight line $zD_{H,0}$ is a good approximation for all distances, while $zT_{H,0}$ fits well the look back time in the same regime.

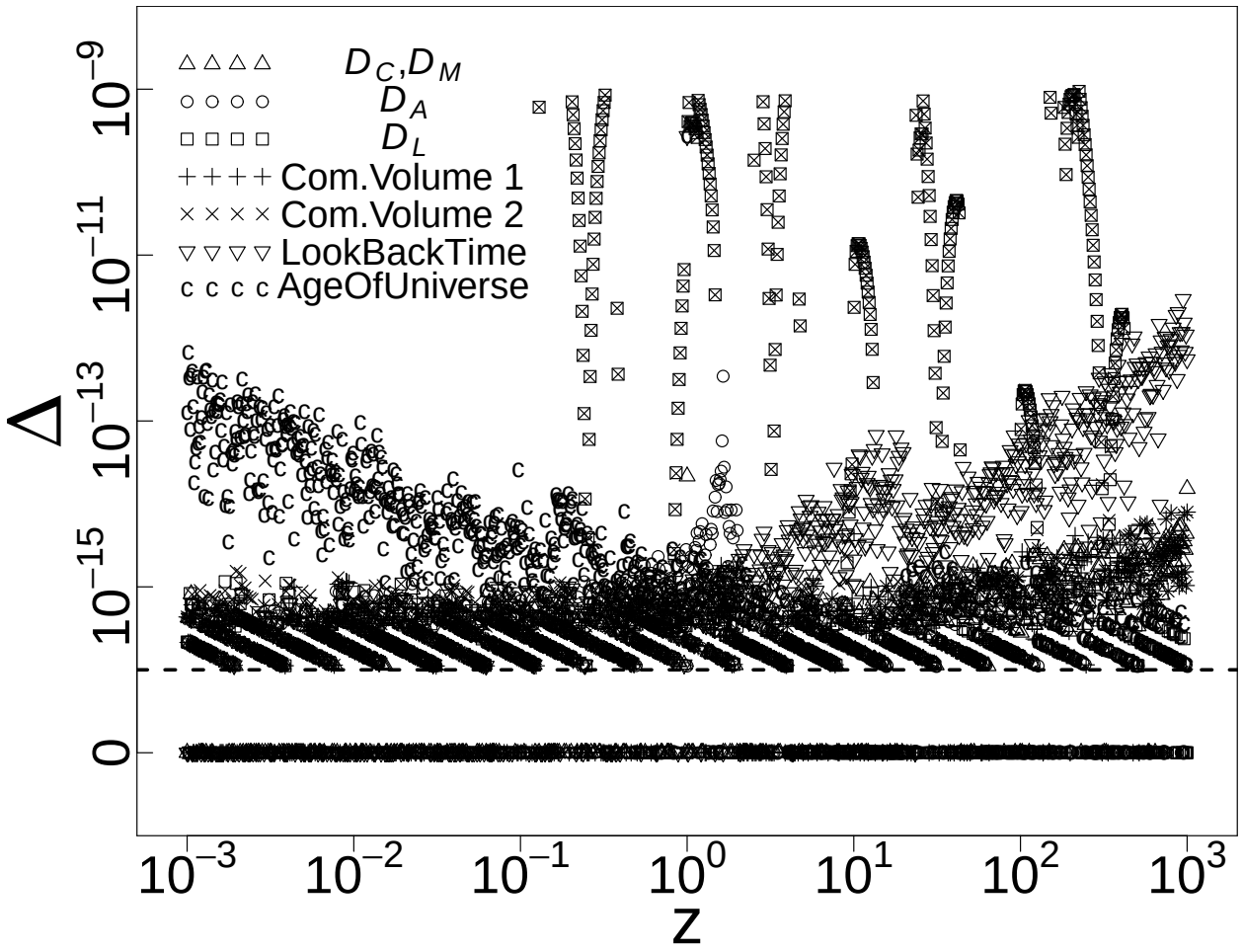

Fig. 2.— Actual fractional precision $\Delta = |Z(F) - z|/z$ of $Z(F)$ with respect to the original redshift z that the value of $F(z)$ is derived from. Note the cutoff at $\Delta \leq 10^{-9}$, imposed in the root finding method. The horizontal dashed line denotes $\Delta = 10^{-16}$, which is the precision for the double data type. The values at $\Delta = 0$ are plotted off the logarithmic scale. The inverse function Com.Volume 1 refers to redshift as a functions of comoving volume, whereas Com.Volume 2 is redshift as a function of D_L composed with D_L as a function of redshift.

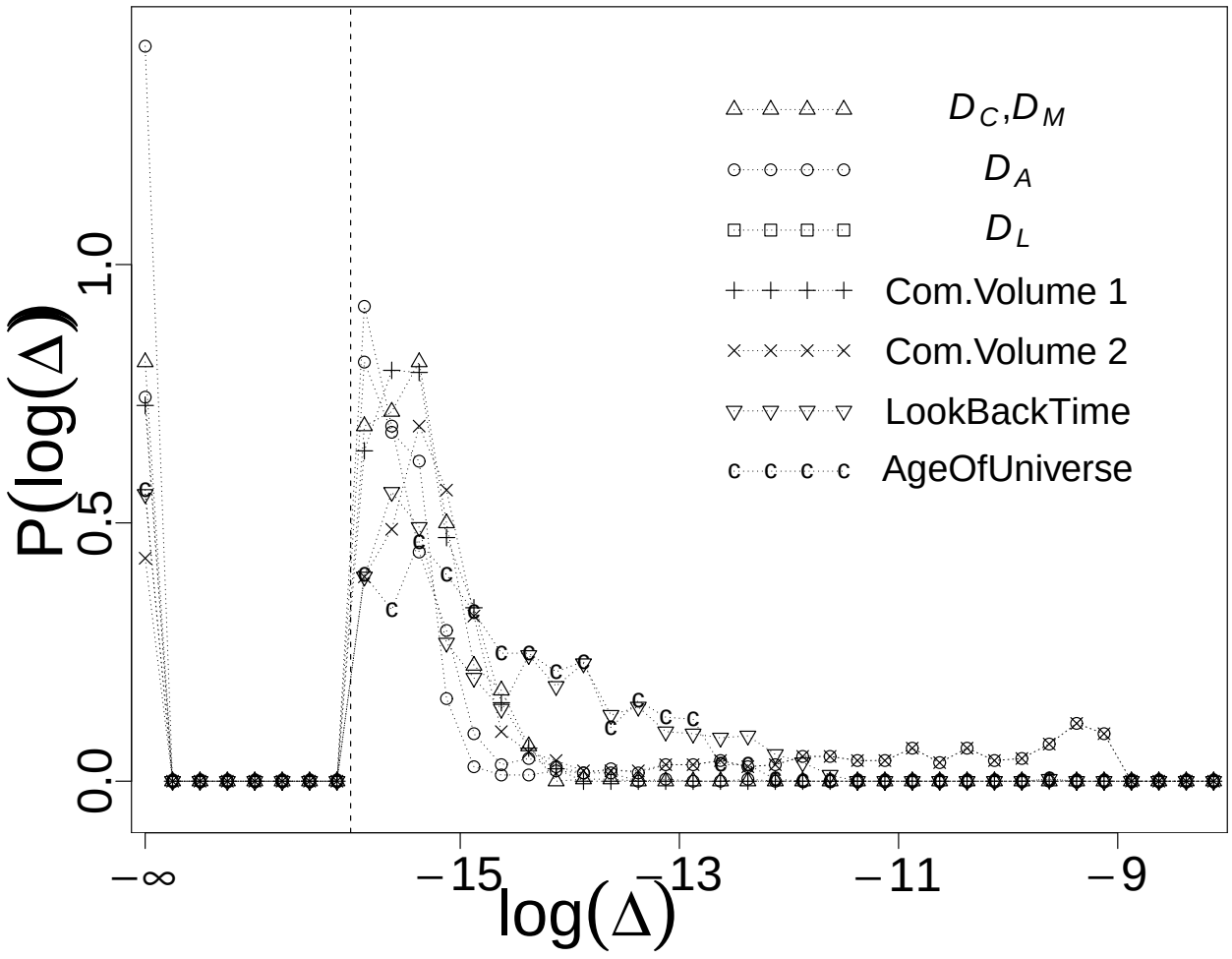

Fig. 3.— Probability distribution of $\log \Delta$, derived from Figure 2. The vertical dashed line denotes $\Delta = 10^{-16}$. The values at $\log \Delta = -\infty$ are plotted off the logarithmic scale.

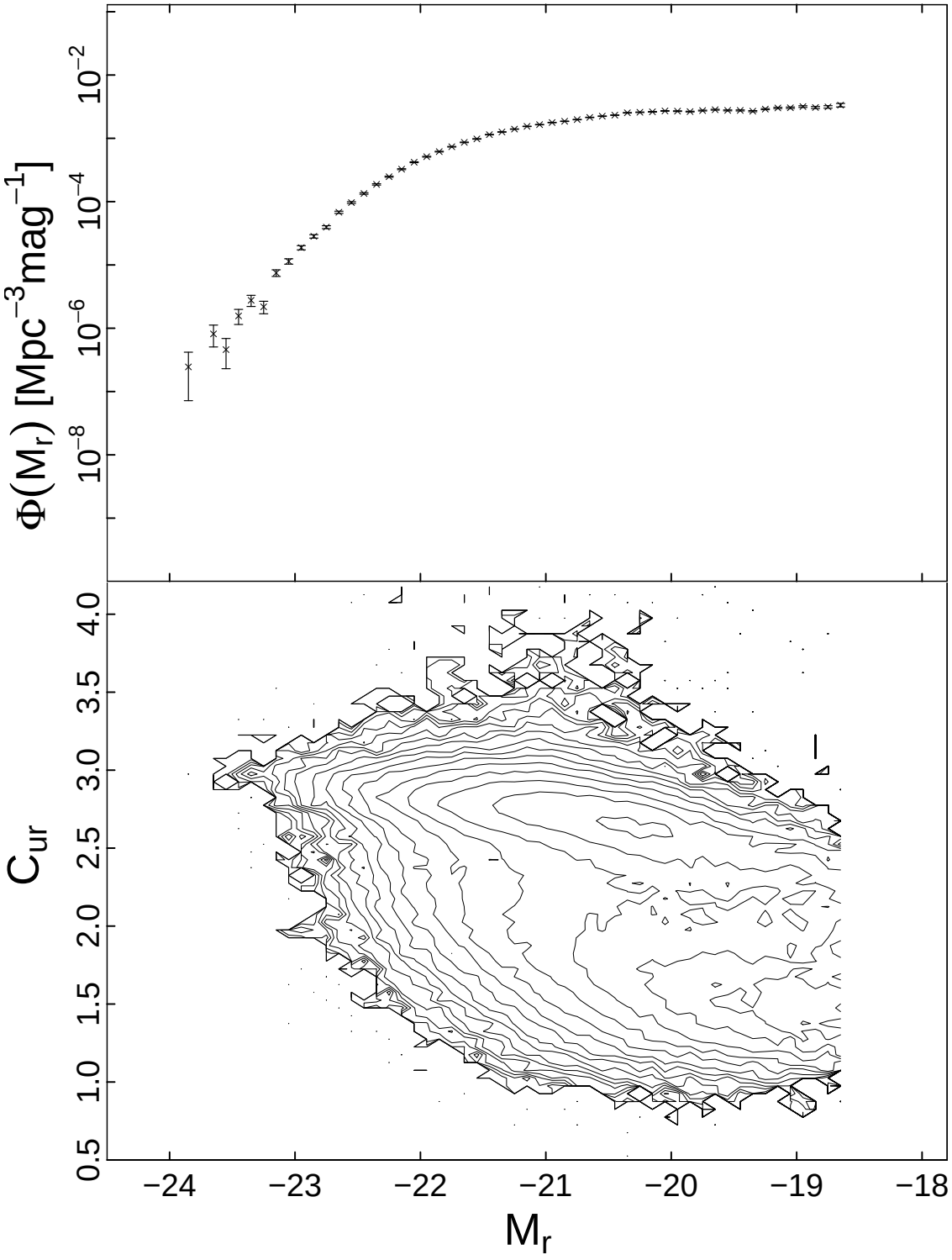

Fig. 4.— Luminosity Function (top) and color magnitude diagram (bottom) of the sample discussed in Section 4.1. The bin sizes used are $\Delta M_r = 0.1$ and $\Delta C_{ur} = 0.05$.

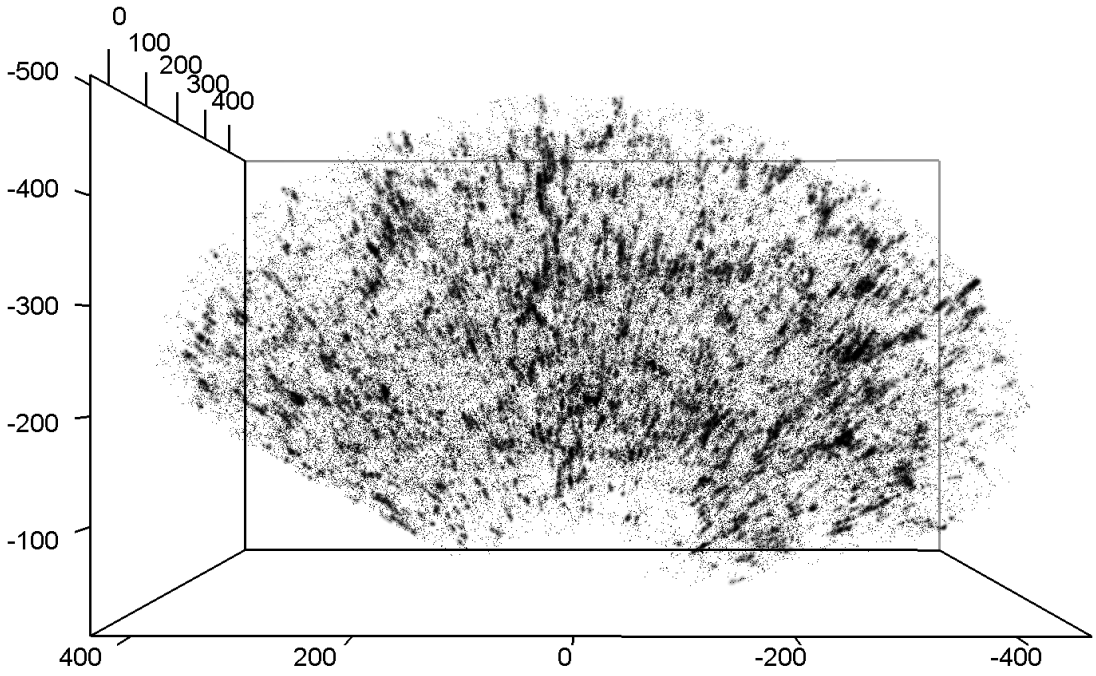

Fig. 5.— Large Scale structure and galaxy groups (derived from the Friends-of-Friends algorithm) in the DR7 northern galactic cap. The axes show comoving distance in Mpc. Details about the volume limited galaxy sample are presented in Sec. 4.1. The galaxies are shown as points, while semitransparent spheres (7 Mpc. radius) are drawn on top of galaxies belonging to groups of more or equal than 10 members.