# Python, SQL and the mass function.

**Violeta Gonzalez-Perez**

@violegp

# Observed galaxy stellar mass function

```
violeta:~> wget http://www.astro.ljmu.ac.uk/~ikb/research/data/gsmf-B12.txt
--2016-09-19 18:18:07--  http://www.astro.ljmu.ac.uk/~ikb/research/data/gsmf-B12.txt
Resolving www.astro.ljmu.ac.uk (www.astro.ljmu.ac.uk)... 150.204.240.7
Connecting to www.astro.ljmu.ac.uk (www.astro.ljmu.ac.uk)|150.204.240.7|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 966 [text/plain]
Saving to: 'gsmf-B12.txt.1'

100%[=====================================================================================>] 966

2016-09-19 18:18:07 (188 MB/s) - 'gsmf-B12.txt.1' saved [966/966]

violeta:~> more gsmf-B12.txt
# Galaxy Stellar Mass Function (GSMF) from GAMA data.
# Table 1 of Baldry et al. 2012, MNRAS, 421, 621.
# number density is per dex per 10^3 Mpc^3; assuming H0=70 km/s/Mpc.
# log mass, bin width, number density, error, number in sample.
 6.25 0.50 31.1  21.6    9
 6.75 0.50 18.1   6.6   19
 7.10 0.20 17.9   5.7   18
```

# Starting with python

- A place to start: https://docs.python.org/3/tutorial/
- Jupyter notebooks: http://jupyter.org/
- Plotting with python: http://matplotlib.org/index.html

```
violeta:~> python
Python 2.7.6 (default, Jun 22 2015, 17:58:13)
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> a =3.
>>> b = 2*a
>>> print b
6.0
>>> import numpy as np
>>> x = np.arange(10)
>>> print x
[0 1 2 3 4 5 6 7 8 9]
>>> print x[0],x[1]
0 1
>>> 
```

# A program in python

```
violeta:~/teaching/laPlata16/mf_ex1> emacs loop1.py &
[1] 3260
violeta:~/teaching/laPlata16/mf_ex1> ./loop1.py
./loop1.py: Permission denied.
violeta:~/teaching/laPlata16/mf_ex1> chmod u+x loop1.py
violeta:~/teaching/laPlata16/mf_ex1> ./loop1.py
Lenght of x= 10
3 1
4 2
5 3
6 4
7 5
8 6
9 7
violeta:~/teaching/laPlata16/mf_ex1> 
```

```python
#! /usr/bin/env python

import numpy as np

x = np.arange(10)
print 'Lenght of x=',len(x)

j = 0
for i in range(len(x)):
    if x[i]>2:
        j = j + 1
        print i,j
```

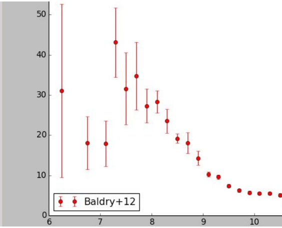
# A program in python

```
violeta:~/teaching/laPlata16/mf_ex1> ls -al loop*py
-rwxrw-r-- 1 violeta violeta 181 Sep 19 18:35 loop1.py
-rw-rw-r-- 1 violeta violeta 157 Sep 19 18:41 loop2.py
violeta:~/teaching/laPlata16/mf_ex1> python loop2.py
Lenght of x= 10
3 1
4 2
5 3
6 4
7 5
8 6
9 7
violeta:~/teaching/laPlata16/mf_ex1>
```

```python
import numpy as np

x = np.arange(10)
print 'Lenght of x=',len(x)

j = 0
for i in range(len(x)):
    if x[i]>2:
        j = j + 1
        print i,j
```

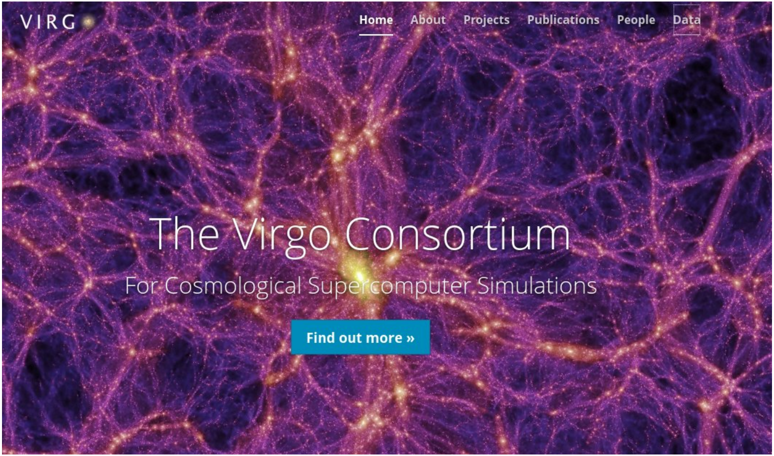
# Reading and plotting in python

```
>>> import numpy as np
>>> ologM, ophi, oe = np.loadtxt('gsmf-B12.txt',usecols=[0,2,3],unpack=True)
>>> print ologM
[  6.25   6.75   7.1    7.3    7.5    7.7    7.9    8.1    8.3    8.5    8.7
   8.9    9.1    9.3    9.5    9.7    9.9   10.1   10.3   10.5   10.7   10.9
  11.1   11.3   11.5   11.7   11.9 ]
>>>
>>> from matplotlib import pyplot as plt
>>> plt.errorbar(ologM, ophi, yerr=oe,\
... color='r',ecolor='r',fmt='o',label='Baldry+12')
<Container object of 3 artists>
>>>
>>> leg = plt.legend(loc=3)
>>> plt.show()
```




**Exercise 1:** Write a program that plots and saves as a pdf the GSMF from Baldry+12, including error bars and in log scales in both axis and units M($M_{\odot} h^{-1}$) and $\Phi(\mathrm{Mpc}^{-3}\mathrm{h}^3/\mathrm{dlogM})$.

# The Millennium simulation




http://www.virgo.dur.ac.uk/

http://virgodb.cosma.dur.ac.uk:8080/Millennium/

- milliMillennium box size $= 62.5$ Mpc$h^{-1}$
- Mass of each dark matter particle $= 8.6 \cdot 10^8 \mathrm{M}_\odot h^{-1}$
- There are different tables with information on the DM only simulations and also on galaxy models used to populate it.

Virgo - Millennium Database

previous up next ToC Single page

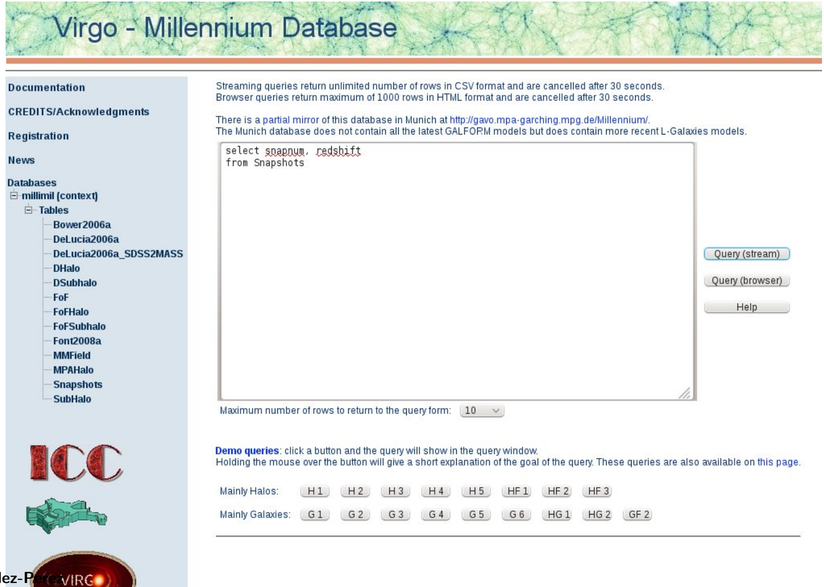
### 3.3.1.1 : Snapshots

This table stores some housekeeping information of the milli-Millennium simulation. In particular, it links redshifts and lookback times to the integer index of the snapshot. Almost all other tables in the millimil database have a snapnum column that corresponds to the one in this table.

| column | type | UCD | unit | description |
|---|---|---|---|---|
| snapnum | integer | | | The order of the snapshot, from 0 to 63 (z=0) |
| z | double | | | The redshift in full precision |
| redshift | real | | | The redshift rounded to two decimal places. |
| lookBackTime | float | | $10^9$ years | The lookback time corresponding to the snapshot |

# A basic Structured Query Language (SQL) query

SQL is a computer language for storing, manipulating and retrieving data stored in relational database.

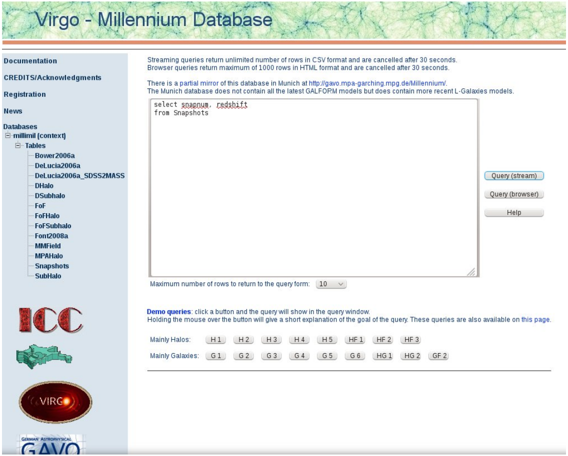


# A basic Structured Query Language (SQL) query

SQL is a computer language for storing, manipulating and retrieving data stored in relational database.

```
#OK
#SQL= select snapnum, redshift
#       from Snapshots
#MAXROWS UNLIMITED
#QUERYTIMEOUT 30 sec
#QUERYTIME 195 millisec
#COLUMN 1 name=snapnum JDBC_TYPE=4 JDBC_TYPENAME=int
#COLUMN 2 name=redshift JDBC_TYPE=3 JDBC_TYPENAME=decimal
snapnum,redshift
0,127.00
1,80.00
2,50.00
3,30.00
4,19.92
5,18.24
6,16.72
7,15.34
8,14.09
9,12.94
10,11.90
11,10.94
12,10.07
13,9.28
```

# A query to get information on the DM haloes



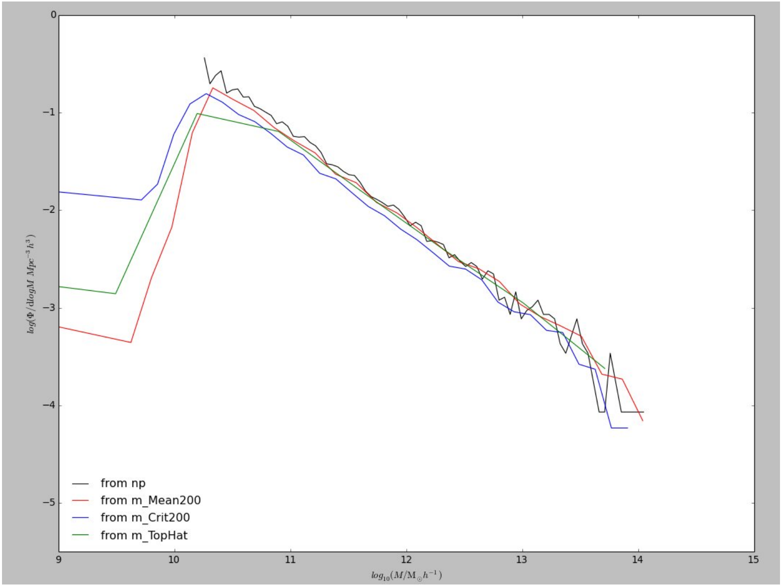

**Exercise 2:** Starting from the 'Demo queries' H1, get all the haloes in the milimillennium including their number of particles and a measure of mass. Save the result into a file.

# A halo mass function (HMF) from your SQL query

**Exercise 3:** Calculate the (HMF) from the milliMillennium in 2 ways. Box size $= 62.5$ Mpc$h^{-1}$, m$_{DM}$ $= 8.6 \cdot 10^{8}$M$_{\odot}h^{-1}$. What happens if you use a different bin size? Make use of np.histogram and of:

```
# Read the SQL query result skipping the header
ff = 'sql_xyz.txt' ; f = open(ff,'r')
data = f.readlines() ; f.close()

# Count number of lines that are not header
nl = 0
for line in data:
    if line[0].isdigit():
        nl = nl + 1
print nl,' read lines'
mass1, mass2 = [np.zeros(shape=(nl)) for i in range(2)]
nl = 0
for line in data:
    if(line[0].isdigit()):
        a = float(line.split(',')[3])
        if (a>0.):
            mass1[nl] = np.log10(a)

        a = float(line.split(',')[4])
        if (a>0.):
            mass2[nl] = np.log10(a)

        nl = nl + 1

print mass1,mass2
```

# The halo mass function: different mass definitions



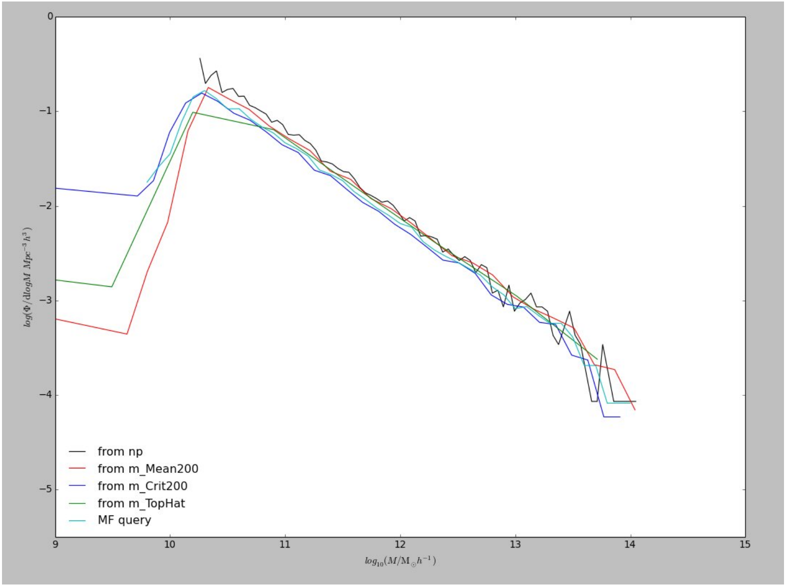

Knebe+15 lists halo mass definitions used in different galaxy models.

# A SQL query for getting directly the HMF

```
select .1*(.5+floor((log10(m_Crit200)+10.)/.1)) as
mass,
log10(count(*)/power(62.5,3.)/.1) as phi
from millimil..MPAHalo
where snapnum= 63 and m_Crit200> 0.
group by .1 * (.5+floor((log10(m_Crit200)+10.)/.1))
order by mass
```

**Exercise 4:** Plot the HMF you obtain from the query above together with the 2 previous ones.

# The halo mass function: two types of queries



**Exercise 5:**

1. Get John Helly's module with useful functions:
   `> wget`
   `http://icc.dur.ac.uk/Eagle/Database/eagleSqlTools.py`
2. Make a simple query. When the URL points at the milli-millennium the username and password are ignored.

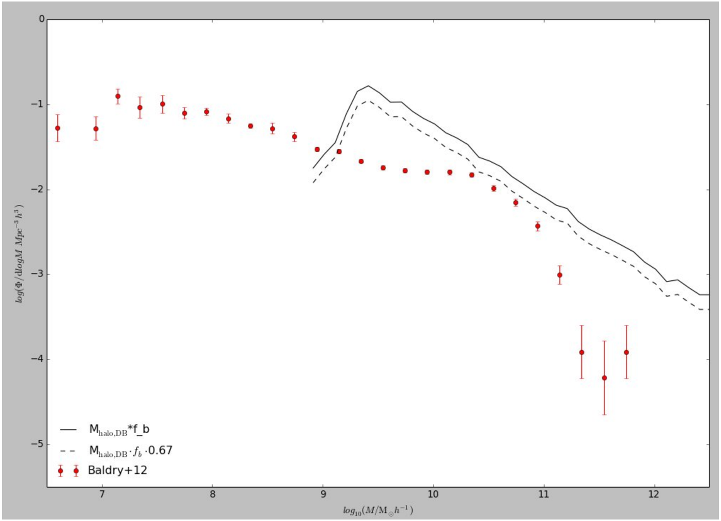
```python
#! /usr/bin/env python

import eagleSqlTools as sql
con = sql.connect("xyz", "abc", url="http://virgodb.dur.ac.uk:8080/Millennium")
data = con.execute_query("select top 10 * from snapshots")
print data
```

3. The result is a numpy record array. Access the columns of the result with expressions like data["snapnum"], data["redshift"] etc. The column names and types are in data.dtype.fields.
4. Now, get the milliMellinium haloes, 'millimil.haloes.txt' with their positions, peculiar velocities, mass, half mass radius and the variables: `haloID`, `firstHaloInFOFgroupId`.

**Exercise 6:** Compare the observed GSMF that you previously downloaded with 2 GSMF derived from the halo mass function, assuming:

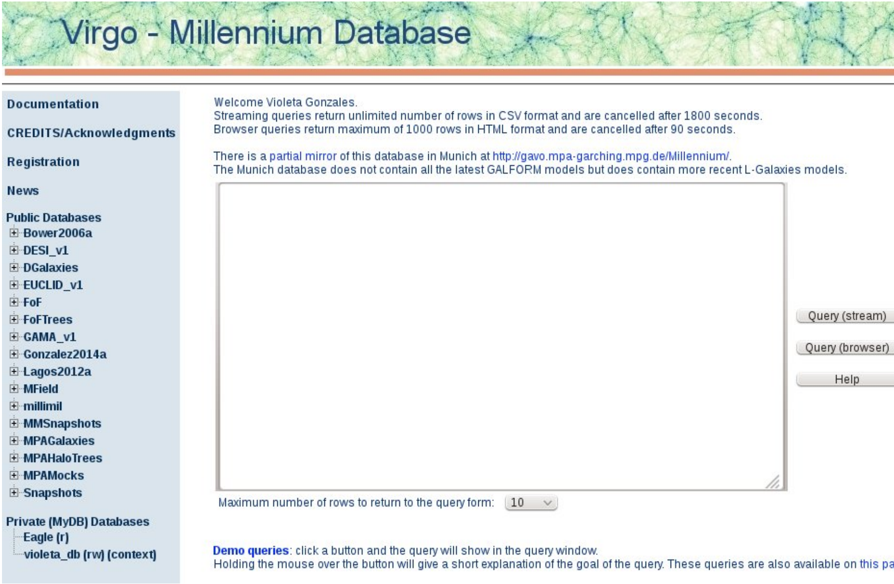
1. That the ratio between halo and stellar mass is the baryonic fraction, $f_b = \Omega_{b,0}/\Omega_{m,0} = 0.04/0.308$, such that: $M_* = M_{\rm halo} \cdot f_b$
2. That the formation of stars and galaxies is inefficient in such a way that: $M_* = \epsilon \cdot M_{\rm halo} \cdot f_b$ (choose $\epsilon$, such that the observed knee of the GSMF is recovered). TIP: Use np.interp().

# The galaxy stellar mass function



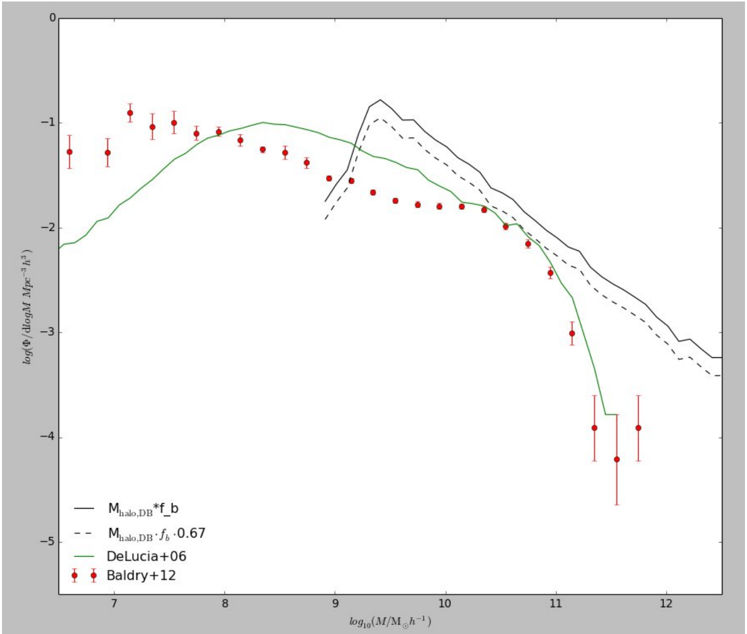

The shapes are very different! We need a better model to connect the luminouse matter to the dark one.

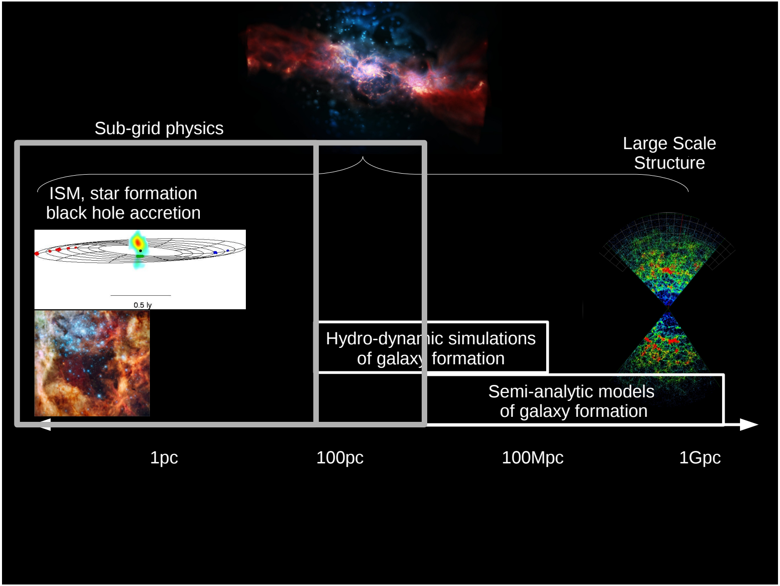
# Populating the Millennium with galaxies



**Exercise 7:** Get the GSMF for the De Lucia et al. 2006 model, which is a comprehensive model of galaxy formation and evolution:
`from millimil..DeLucia2006a`
Save it to a file using `np.savetxt` and plot it together with your previous theoretical GSMF and compared to Baldry et al. 2012 data.

# The GSMF from the milliMillennium





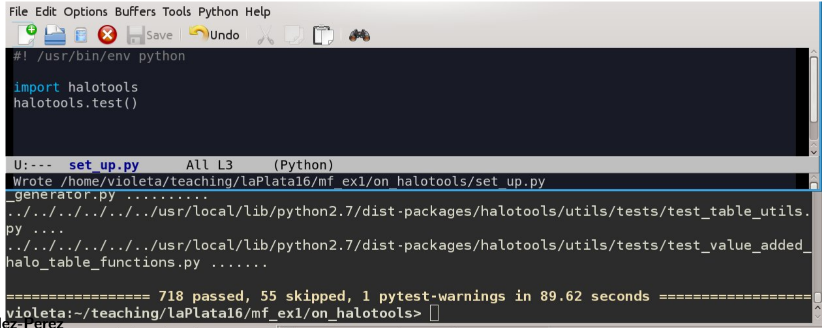

CREDIT: Claudia Lagos

**Exercise 8:** Get halotools,
`https://halotools.readthedocs.io/`
The easiest ways to install `halotools` require either `pip` or `conda`, make sure you have them installed.

If you encounter a problem related to the c compilers, try:

```
> sudo apt-get install python-dev
```

Verify your installation:



**Exercise 9:** Modify the following code such that

- `ids` and `upid` are initialized as 2 integer arrays with the size of the number of haloes downloaded.
- Store `haloID` into the long integer array `ids`.
- If `firstHaloInFOFgroupId`=`haloID` set `upid`$= -1$, and to

```
#! /usr/bin/env python

import numpy as np
from halotools.sim_manager import UserSuppliedHaloCatalog

# Read the SQL query result skipping the header
ff = '../sql_xyz.txt' ; f = open(ff,'r')
data = f.readlines() ; f.close()

# Count number of lines that are not header
nl = 0
for line in data:
    if line[0].isdigit():
        nl = nl + 1
print nl,' haloes'
xm, ym, zm, mass = [np.zeros(shape=(nl)) for i in range(4)]
ids = np.arange(0,nl)

nl = 0
for line in data:
    if(line[0].isdigit()):
        xm[nl] = float(line.split(',')[0])
        ym[nl] = float(line.split(',')[1])
        zm[nl] = float(line.split(',')[2])

        a = float(line.split(',')[5])
        if (a>0.):
            mass[nl] = np.log10(a*0.86) +9.

        nl = nl + 1
```

**Exercise 9:** Pass the arrays you've previously created, fixing the following:

```
halo_catalog = UserSuppliedHaloCatalog(simname='miliMillennium',\
                                       redshift = 0.0,\
                                       Lbox = 62.5,\
                                       particle_mass = 8.6e8,\
                                       halo_x = xm,\
                                       halo_y = ym,\
                                       halo_z = zm,\
                                       halo_id = ids,\
                                       halo_upid = upid)

halos = halo_catalog.halo_table
print(halos.keys())
```

# millimil populated with the Zheng+07 model

In the Zheng+07, the NFWPhaseSpace class from Halotools requires knowledge of halo concentration to assign an intra-halo spatial distribution to the satellites. By default, the concentration of the actual halos in the catalog are used for this purpose. However, we haven't downloaded that attribute so to have satellites distributed according to an NFW profile, we need an analytical model for the concentration-mass relation, such as the one from Dutton & Maccio 2014:

**Exercise 10:** Try

```
model = PrebuiltHodModelFactory('leauthaud11', conc_mass_model='dutton_maccio14')
model.populate_mock(halocat = halo_catalog)
print model.mock.galaxy_table.keys()
```

# The mean HOD

**Exercise 11:** Compare the mean HOD from Halotools with that from De Lucia et al. 2006 model.