Main

July 13, 2018

```
In [1]: import halotools
        import numpy as np
        import matplotlib.pyplot as plt
        from scipy.optimize import curve_fit
        %matplotlib inline
```

1 1) Import data

```
In [2]: from halotools import sim_manager
        from halotools.sim_manager import CachedHaloCatalog
        halocat = CachedHaloCatalog(simname = 'bolplanck', halo_finder = 'rockstar'
```

We begin with the analysis of a subsample of the Bolshoi simulation with Planck cosmology as of 2013 (h=0.7, Ω_Λ=0.69289, Ω_m = 0.30711, Ω_b = 0.048, n_s = 0.96, σ_8 = 0.82). The simulation started at z_{ini} = 80 and followed 2048^3 particles of mass $M_{dm} = 1.55 \times 10^8\ M_{sun}/h$ in their evolution in a volume of boxsize L=250 Mpc/h.

```
In [3]: #Catalog of halos itself
        halos = halocat.halo_table
        #To see what properties are available
        print(halos.keys())

['halo_vmax_firstacc', 'halo_dmvir_dt_tdyn', 'halo_macc', 'halo_scale_factor', 'hal
```

2 2) Organize the data into samples

```
In [4]: from halotools.utils import group_member_generator
        #Group halos by halo host id
        halos.sort('halo_hostid')
        grouping_key = 'halo_hostid'
        requested_columns = []
        group_gen = group_member_generator(halos, grouping_key, requested_columns)

        #calculates the number of subalos per host
        nsub = np.zeros(len(halos))
        for first, last, member_props in group_gen:
```

```
            nsub[first:last] = last - first - 1
        #Adds a new column to the ith-row of the table which gives the number of su
        halos['num_subhalos'] = nsub
```

```
In [5]: NumberHosthalos=0
        NumberSubhalos=0
        for i in range(0,len(halos)):
            if halos["halo_pid"][i]==-1:
                NumberHosthalos= NumberHosthalos+1
            else:
                NumberSubhalos = NumberSubhalos+1

        print("Number of halos in total, unfilterd bolplank catalog", len(halos))
        print("Number of hosthalos, unfilterd bolplank catalog", NumberHosthalos)
        print("Number of subhalos, unfilterd bolplank catalog", NumberSubhalos)
```

```
Number of halos in total, unfilterd bolplank catalog 1444262
Number of hosthalos, unfilterd bolplank catalog 1078614
Number of subhalos, unfilterd bolplank catalog 365648
```

```
In [9]: #Attribute Subhalos with the Positions of their respective host in order to
        #center of mass frame:
        halos.add_column(halos["halo_x"], name="Host_x")
        halos.add_column(halos["halo_y"], name="Host_y")
        halos.add_column(halos["halo_z"], name="Host_z")
        halos.add_column(halos["halo_vx"], name="Host_vx")
        halos.add_column(halos["halo_vy"], name="Host_vy")
        halos.add_column(halos["halo_vz"], name="Host_vz")
```

The catalog is filtered by the following criteria. We generate two samples :

Sample 1 consists of all host halos with $M_{vir} < 10^{15} M_{sun}$ and with $M_{vir}/M_{Dm,partcl} > 100$ together with the subhalos with $M_{vir}/M_{Dm,partcl} > 100$, $M_{vir}/M_{Vir,Host} > 10^{-3}$ and with $M_{Vir,Host} < 10^{15} M_{sun}$.

Sample 2 is constrained under the same criteria as Sample 1, but one of the conditions on the subhalos, namely $M_{vir}/M_{Vir,Host} > 10^{-3}$, is removed.

```
In [6]: ##We consider two main samples:
        ##Sample 1: Consider all halos consisting of more than 100 dm particles and
        ##Sample 2: Take into account all halos and still exclude hosts with less t
        #The mask for the hosts
        M_dm_particle = 1.55e8
        host_maskSample1 = ((halos['halo_upid'] == -1) & (halos["halo_mvir"]<1e15)
        hostsSample1 = halos[host_maskSample1]


        #mask for the subhalos

        sub_maskSample1 = ((halos["halo_upid"]>0)&((halos['halo_mvir']/halos["halo_
```

```
subsSample1 = halos[sub_maskSample1]
sub_maskSample2 = ((halos["halo_upid"]>0)&(halos["halo_mvir_host_halo"]< 1e
subsSample2 = halos[sub_maskSample2]

print("Number of hosthalos in Sample 1 = ", len(hostsSample1))
print("The mean concentration of host halos in sample 1 is  ", "r$ \bar{c}$
print("Number of subhalos in Sample 1 = ", len(subsSample1))
print("Number of subhalos in Sample 2 = ", len(subsSample2))
```

```
C:\Users\ThimoPreis\Anaconda3\lib\site-packages\ipykernel\__main__.py:12: RuntimeWa


Number of hosthalos in Sample 1 =  1071702
The mean concentration of host halos in sample 1 is   r$ ar{c}$ =  16.0799
Number of subhalos in Sample 1 =  224818
Number of subhalos in Sample 2 =  289020
```

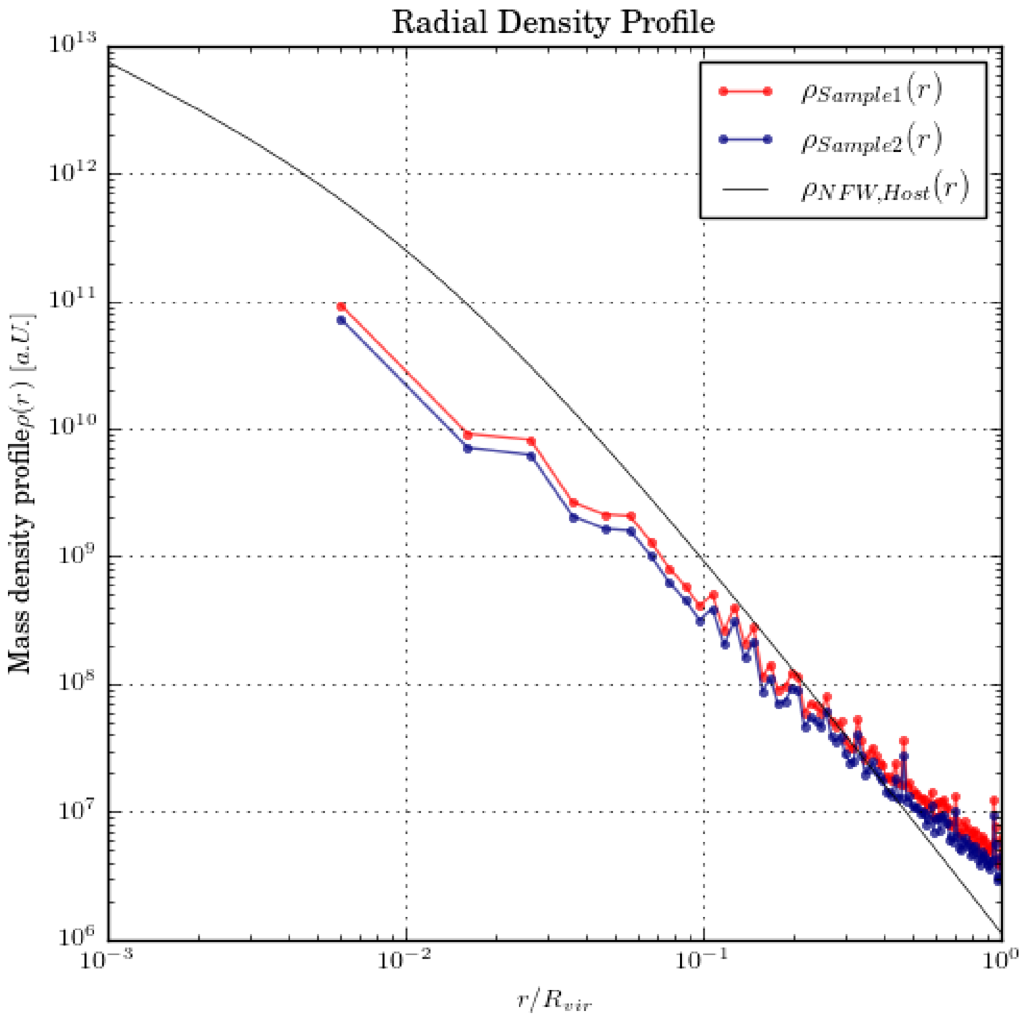
In [ ]:

3 3) Radial density profile

In [7]:
```
from halotools.mock_observables import return_xyz_formatted_array

host_pos_sample1 = return_xyz_formatted_array(hostsSample1['halo_x'], hosts

sub_pos_sample1 = return_xyz_formatted_array(subsSample1['halo_x'], subsSam
sub_pos_sample2 = return_xyz_formatted_array(subsSample2['halo_x'], subsSam

hx1 = hostsSample1["halo_x"]
hy1 = hostsSample1["halo_y"]
hz1 = hostsSample1["halo_z"]
sx1 = subsSample1["halo_x"]
sy1 = subsSample1["halo_y"]
sz1 = subsSample1["halo_z"]
sx2 = subsSample2["halo_x"]
sy2 = subsSample2["halo_y"]
sz2 = subsSample2["halo_z"]

hvx1 = hostsSample1["halo_vx"]
hvy1 = hostsSample1["halo_vy"]
hvz1 = hostsSample1["halo_vz"]
svx1 = subsSample1["halo_vx"]
svy1 = subsSample1["halo_vy"]
svz1 = subsSample1["halo_vz"]
svx2 = subsSample2["halo_vx"]
svy2 = subsSample2["halo_vy"]
svz2 = subsSample2["halo_vz"]
```

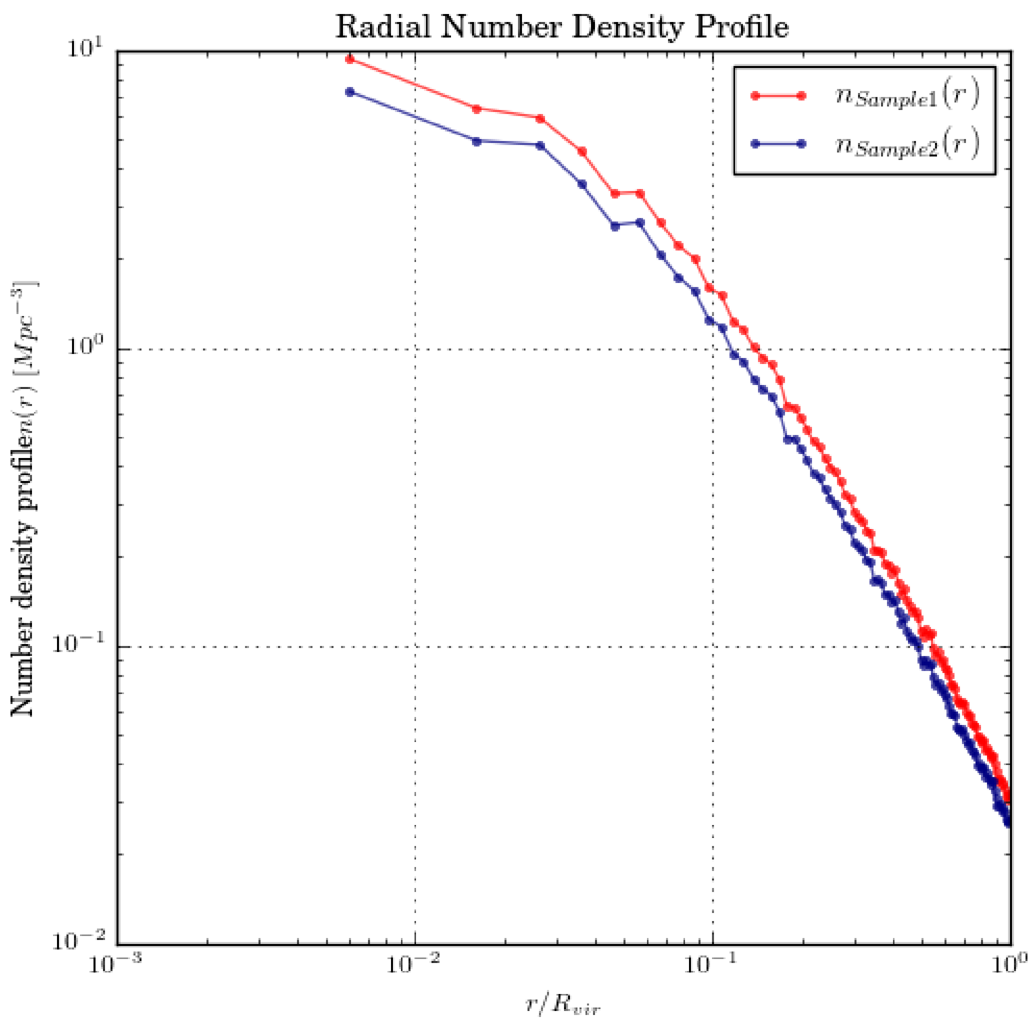
```
In [9]: from halotools.mock_observables import radial_profile_3d

        rbins_normalized = np.linspace(0.001, 1, 100)
        rbins_midpoints = (rbins_normalized[:-1] + rbins_normalized[1:])/2

        Density_profileSample1,CountsSample1 = radial_profile_3d(host_pos_sample1,
                                          rbins_normalized = rbins_normalized,
                                          normalize_rbins_by = hostsSample1['halo_rvir'],
                                          period=halocat.Lbox, return_counts = True)

        Density_profileSample2,CountsSample2 = radial_profile_3d(host_pos_sample1,
                                          rbins_normalized = rbins_normalized,
                                          normalize_rbins_by = hostsSample1['halo_rvir'],
                                          period=halocat.Lbox,return_counts = True)

In [10]: shell_volumes = 4/3*np.pi*(rbins_normalized[1:]**3-rbins_normalized[:-1]**
         MeanNumberDensitySample1 = (CountsSample1/float(len(subsSample1)))/shell_v
         MeanNumberDensitySample2 = (CountsSample2/float(len(subsSample2)))/shell_v

In [19]: #Model NFW profil of the hosts
         #from halotools import empirical_models
         #from halotools.empirical_models import NFWProfile

         #nfw = NFWProfile(conc_mass_model='direct_from_halo_catalog', mdef='vir',c
         #model_conc = nfw.conc_NFWmodel(table=hostsSample1)

         #Write NFW profile function yourself:
         from halotools.empirical_models import density_threshold
         from astropy import cosmology
         from astropy.cosmology import LambdaCDM
         cosmo = LambdaCDM(H0=70, Om0=0.3, Ode0=0.7)
         m_dm=1.55e8

         def NFW(r):
             return density_threshold(cosmo,0,mdef="vir")*0.3/3*np.mean(hostsSample


         rhocrit = cosmo.critical_density0
         #Another definition, completely equivalent to NFW, not neccessary to use i
         c=np.mean(hostsSample1["halo_nfw_conc"])

         def nfwProfile(r):
             return (200*rhocrit*c)/3/(np.log(1+c-c/(1+c))) /(r/np.mean(hostsSample




         #r_values_hosts=np.linspace(0.001,1,len(hostsSample1))
         #rho_nfw = nfw.dimensionless_mass_density(scaled_radius =r_values_hosts,co
```

In [25]:
```
plt.rc('text', usetex=True)
plt.rc('font', family='serif')

r_values =np.linspace(0.001,1,1000)

# Create a new figure
fig = plt.figure(figsize=(6,6), dpi=80)
plot_profile = fig.add_subplot(111)
plot_profile.set_xscale('log')
plot_profile.set_yscale('log')
plot_profile.grid(True)

#plot_ps.set_xlim(1.0e-5, 1.0e5)
#plot_profile.set_ylim(0.001, 1e3)

plot_profile.set_xlabel(r' $ r/R_{vir} $')
plot_profile.set_ylabel(r'Mass density profile$\rho (r) \;[a.U.]$')

plot_profile.plot(rbins_midpoints, Density_profileSample1/shell_volumes/fl
plot_profile.plot(rbins_midpoints, Density_profileSample2/shell_volumes/fl
#plot_profile.plot(r_values_hosts, rho_nfw, color="black", marker=".",labe
plot_profile.plot(r_values, NFW(r_values)/4e3, color="black",lw=0.5,label=



plot_profile.legend(prop={'size':12}, loc="best")
plt.title("Radial Density Profile")
# Save figure using 80 dots per inch
plt.savefig("RadialDensityProfile.pdf",dpi=80)
```




```
In [26]: # Create a new figure
         fig = plt.figure(figsize=(6,6), dpi=80)
         plot_profile = fig.add_subplot(111)
         plot_profile.set_xscale('log')
         plot_profile.set_yscale('log')
         plot_profile.grid(True)

         #plot_ps.set_xlim(1.0e-5, 1.0e5)
         #plot_profile.set_ylim(0.001, 1e3)

         plot_profile.set_xlabel(r' $ r/R_{vir} $')
         plot_profile.set_ylabel(r'Number density profile$n(r) \;[Mpc^{-3}]$')

         plot_profile.plot(rbins_midpoints, MeanNumberDensitySample1, color="red",
         plot_profile.plot(rbins_midpoints, MeanNumberDensitySample2, color="navy",
```

```
plot_profile.legend(prop={'size':12}, loc="best")
plt.title("Radial Number Density Profile")
# Save figure using 80 dots per inch
plt.savefig("RadialNumberDensityProfile.pdf",dpi=80)
```



4 4)Velocity profiles

```
In [ ]: #Replace every position and velocity of the subhalos by the position and ve
        #frame of their respective host halo:



        for i in range(0, len(hostsSample1)):
```

```
for j in range(0, len(subsSample1)):

    if hostsSample1["halo_id"][i]==subsSample1["halo_hostid"][j]:

        sx1[j] = sx1[j]-hx1[i]
        sy1[j] = sy1[j]-hy1[i]
        sz1[j] = sz1[j]-hz1[i]
        svx1[j] = svx1[j]-hvx1[i]
        svy1[j] = svy1[j]-hvy1[i]
        svz1[j] = svz1[j]-hvz1[i]

    else:

        sx1[j] = 10000
        sy1[j] = 10000
        sz1[j] = 10000
        svx1[j] = 10000
        svy1[j] = 10000
        svz1[j] = 10000

for m in range(0, len(subsSample2)):

    if hostsSample1["halo_id"][i]==subsSample2["halo_hostid"][m]:

        sx2[m] = sx2[m]-hx2[i]
        sy2[m] = sy2[m]-hy2[i]
        sz2[m] = sz2[m]-hz2[i]
        svx2[m] = svx2[m]-hvx2[i]
        svy2[m] = svy2[m]-hvy2[i]
        svz2[m] = svz2[m]-hvz2[i]

    else:

        sx1[m] = 10000
        sy1[m] = 10000
        sz1[m] = 10000
        svx1[m] = 10000
        svy1[m] = 10000
        svz1[m] = 10000
```

5 Put every component to zero, when subhalo does not have a corresponding host halo

for i in range(0,len(subsSample1)): if indexReturn(subsSample1[“halo_hostid”][i])==“NOTFOUND”: sx1[i] = 0 sy1[i] = 0 sz1[i] = 0 svx1[i] = 0 svy1[i] = 0 svz1[i] = 0

```
else:

    sx1[i] = sx1[i]- hx1[indexReturn(subsSample1["halo_upid"][i])]
    sy1[i] = sy1[i]- hy1[indexReturn(subsSample1["halo_upid"][i])]
    sz1[i] = sz1[i]- hz1[indexReturn(subsSample1["halo_upid"][i])]
    svx1[i] = svx1[i]- hvx1[indexReturn(subsSample1["halo_upid"][i])]
    svy1[i] = svy1[i]- hvy1[indexReturn(subsSample1["halo_upid"][i])]
    svz1[i] = svz1[i]- hvz1[indexReturn(subsSample1["halo_upid"][i])]
```

for k in range(0,len(subsSample2)):

```
if indexReturn(subsSample1["halo_hostid"][i])=="NOTFOUND":
    sx2[k] = 0
    sy2[k] = 0
    sz2[k] = 0
    svx2[k] = 0
    svy2[k] = 0
    svz2[k] = 0
else:
    sx2[k] = sx2[k]- hx1[indexReturn(subsSample2["halo_upid"][k])]
    sy2[k] = sy2[k]- hy1[indexReturn(subsSample2["halo_upid"][k])]
    sz2[k] = sz2[k]- hz1[indexReturn(subsSample2["halo_upid"][k])]
    svx2[k] = svx2[k]- hvx1[indexReturn(subsSample2["halo_upid"][k])]
    svy2[k] = svy2[k]- hvy1[indexReturn(subsSample2["halo_upid"][k])]
    svz2[k] = svz2[k]- hvz1[indexReturn(subsSample2["halo_upid"][k])]
```

```
In [12]: #Go into spherical coordinates
         r_s1 = np.sqrt(sx1**2+sy1**2+sz1**2)
         r_s2 = np.sqrt(sx2**2+sy2**2+sz2**2)

         vr_s1 = (sx1*svx1+sy1*svy1+sz1*svz1)/r_s1
         vr_s2 = (sx2*svx2+sy2*svy2+sz2*svz2)/r_s2

         vphi_s1 = (svx1*sy1-sx1*svy1)/(sx1**2+sy1**2)
         vphi_s2 = (svx2*sy2-sx2*svy2)/(sx2**2+sy2**2)

         vth_s1 = (sz1*(sx1*svx1+sy1*svy1)-svz1*(sx1**2+sy1**2))/np.sqrt(sx1**2+sy1
         vth_s2 = (sz2*(sx2*svx2+sy2*svy2)-svz2*(sx2**2+sy2**2))/np.sqrt(sx2**2+sy2
```

```
In [14]: #Average the radial velocity profiles
```

```
from halotools.mock_observables import radial_profile_3d

rbins_normalized = np.linspace(0.001, 1, 100)
rbins_midpoints = (rbins_normalized[:-1] + rbins_normalized[1:])/2

aver_vr_s1 = radial_profile_3d(host_pos_sample1, sub_pos_sample1, vr_s1,
                          rbins_normalized = rbins_normalized,
                          normalize_rbins_by = hostsSample1['halo_rvir'],
                          period=halocat.Lbox)

aver_vr_s2 = radial_profile_3d(host_pos_sample1, sub_pos_sample2, vr_s2,
                          rbins_normalized = rbins_normalized,
                          normalize_rbins_by = hostsSample1['halo_rvir'],
                          period=halocat.Lbox)

aver_vphi_s1 = radial_profile_3d(host_pos_sample1, sub_pos_sample1, vphi_s
                          rbins_normalized = rbins_normalized,
                          normalize_rbins_by = hostsSample1['halo_rvir'],
                          period=halocat.Lbox)

aver_vphi_s2 = radial_profile_3d(host_pos_sample1, sub_pos_sample2, vphi_s
                          rbins_normalized = rbins_normalized,
                          normalize_rbins_by = hostsSample1['halo_rvir'],
                          period=halocat.Lbox)


aver_vth_s1 = radial_profile_3d(host_pos_sample1, sub_pos_sample1, vth_s1,
                          rbins_normalized = rbins_normalized,
                          normalize_rbins_by = hostsSample1['halo_rvir'],
                          period=halocat.Lbox)

aver_vth_s2 = radial_profile_3d(host_pos_sample1, sub_pos_sample2, vth_s2,
                          rbins_normalized = rbins_normalized,
                          normalize_rbins_by = hostsSample1['halo_rvir'],
                          period=halocat.Lbox)
```

In [24]:
```
#Normalize by virial velocity
np.max(sx2)
```

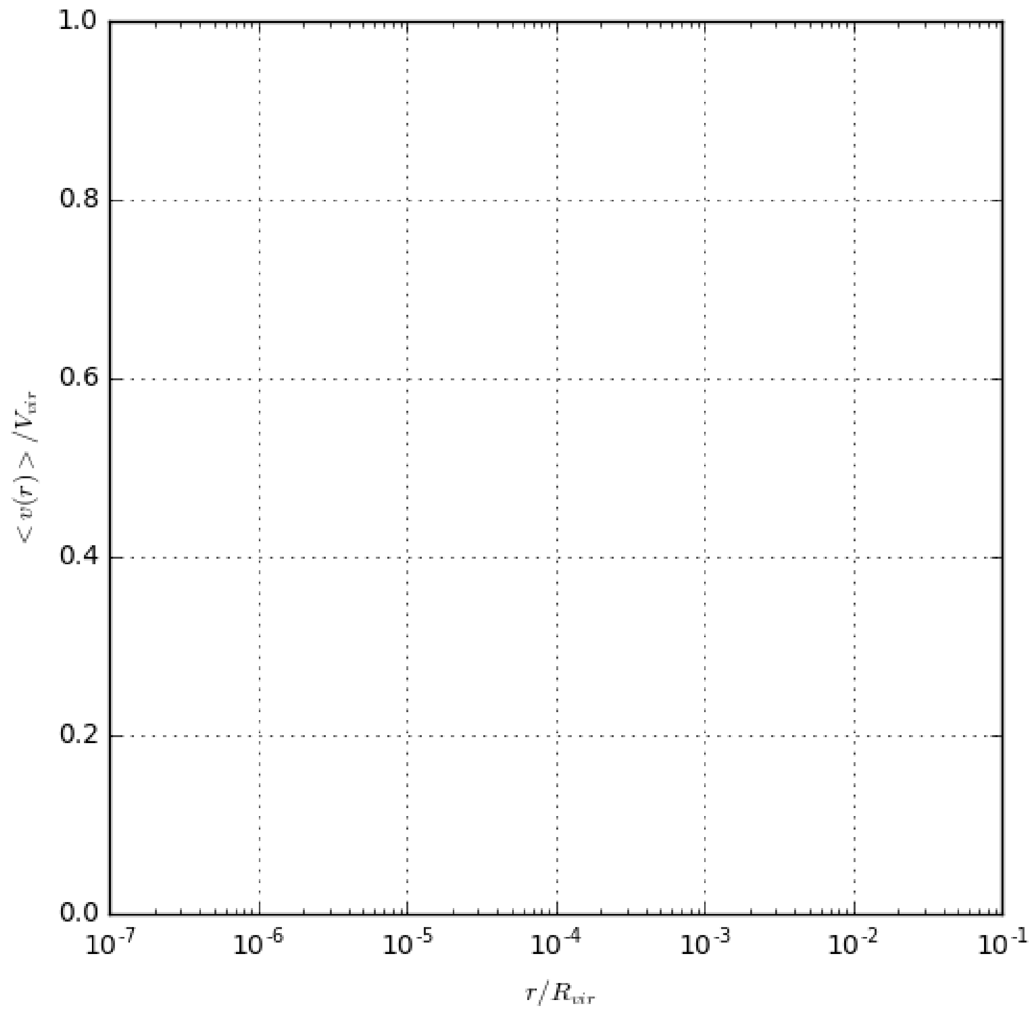
Out[24]: 0.0

In [16]:
```
# Create a new figure
fig = plt.figure(figsize=(6,6), dpi=80)
plot_profile = fig.add_subplot(111)
plot_profile.set_xscale('log')
#plot_profile.set_yscale('log')
plot_profile.grid(True)
```

```
#plot_ps.set_xlim(1.0e-5, 1.0e5)
#plot_profile.set_ylim(0.001, 1e3)

plot_profile.set_xlabel(r' $ r/R_{vir} $')
plot_profile.set_ylabel(r'$<v(r)>/V_{vir}$')

plot_profile.plot(rbins_midpoints, aver_vr_s1, color="red", marker=".",lab
plot_profile.plot(rbins_midpoints, aver_vphi_s1, color="navy", marker=".",
#plot_profile.plot(r_values_hosts, rho_nfw, color="black", marker=".",labe
plot_profile.plot(rbins_midpoints, aver_vth_s1, color="black",lw=0.5,label



plot_profile.legend(prop={'size':12}, loc="best")
plt.title("Radial Average Velocity Profile Sample 1")
# Save figure using 80 dots per inch
plt.savefig("RadialVelocityProfilesSample1.pdf",dpi=80)
```

```
---------------------------------------------------------------------------

ValueError                                Traceback (most recent call last)

<ipython-input-16-e09fe14ad7b0> in <module>()
     12 plot_profile.set_ylabel(r'$<v(r)>/V_{vir}$')
     13
---> 14 plot_profile.plot(rbins_midpoints, aver_vr_s1, color="red", marker=".",
     15 plot_profile.plot(rbins_midpoints, aver_vphi_s1, color="navy", marker="
     16 #plot_profile.plot(r_values_hosts, rho_nfw, color="black", marker=".",l


C:\Users\ThimoPreis\Anaconda3\lib\site-packages\matplotlib\__init__.py in i
   1816                     warnings.warn(msg % (label_namer, func.__name__),
   1817                                   RuntimeWarning, stacklevel=2)
-> 1818             return func(ax, *args, **kwargs)
   1819         pre_doc = inner.__doc__
   1820         if pre_doc is None:


C:\Users\ThimoPreis\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py in
   1384             lines.append(line)
   1385
-> 1386         self.autoscale_view(scalex=scalex, scaley=scaley)
   1387         return lines
   1388


C:\Users\ThimoPreis\Anaconda3\lib\site-packages\matplotlib\axes\_base.py in
```

```
   2166                 x1 += delta
   2167             if not _tight:
-> 2168                 x0, x1 = xlocator.view_limits(x0, x1)
   2169             self.set_xbound(x0, x1)
   2170


    C:\Users\ThimoPreis\Anaconda3\lib\site-packages\matplotlib\ticker.py in vie
   1615         if minpos <= 0 or not np.isfinite(minpos):
   1616             raise ValueError(
-> 1617                 "Data has no positive values, and therefore can not be
   1618                 "log-scaled.")
   1619


    ValueError: Data has no positive values, and therefore can not be log-scale
```



```
In [ ]: # Create a new figure
        fig = plt.figure(figsize=(6,6), dpi=80)
        plot_profile = fig.add_subplot(111)
        plot_profile.set_xscale('log')
        plot_profile.set_yscale('log')
        plot_profile.grid(True)

        #plot_ps.set_xlim(1.0e-5, 1.0e5)
        #plot_profile.set_ylim(0.001, 1e3)

        plot_profile.set_xlabel(r' $ r/R_{vir} $')
        plot_profile.set_ylabel(r'$<v(r)>/V_{vir}$')

        plot_profile.plot(rbins_midpoints, aver_vr_s1, color="red", marker=".",labe
        plot_profile.plot(rbins_midpoints, aver_vphi_s1, color="navy", marker=".",l
        #plot_profile.plot(r_values_hosts, rho_nfw, color="black", marker=".",label
        plot_profile.plot(rbins_midpoints, aver_vth_s1, color="black",lw=0.5,label=



        plot_profile.legend(prop={'size':12}, loc="best")
        plt.title("Radial Average Velocity Profile Sample 2")
        # Save figure using 80 dots per inch
        plt.savefig("RadialVelocityProfilesSample2.pdf",dpi=80)

In [ ]:

In [52]:

2821926828
1071307
243.125
243.142
0.0174408


In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:
```