Calculating Scaling Relations of DES Galaxy Clusters using Multiwavelength Data

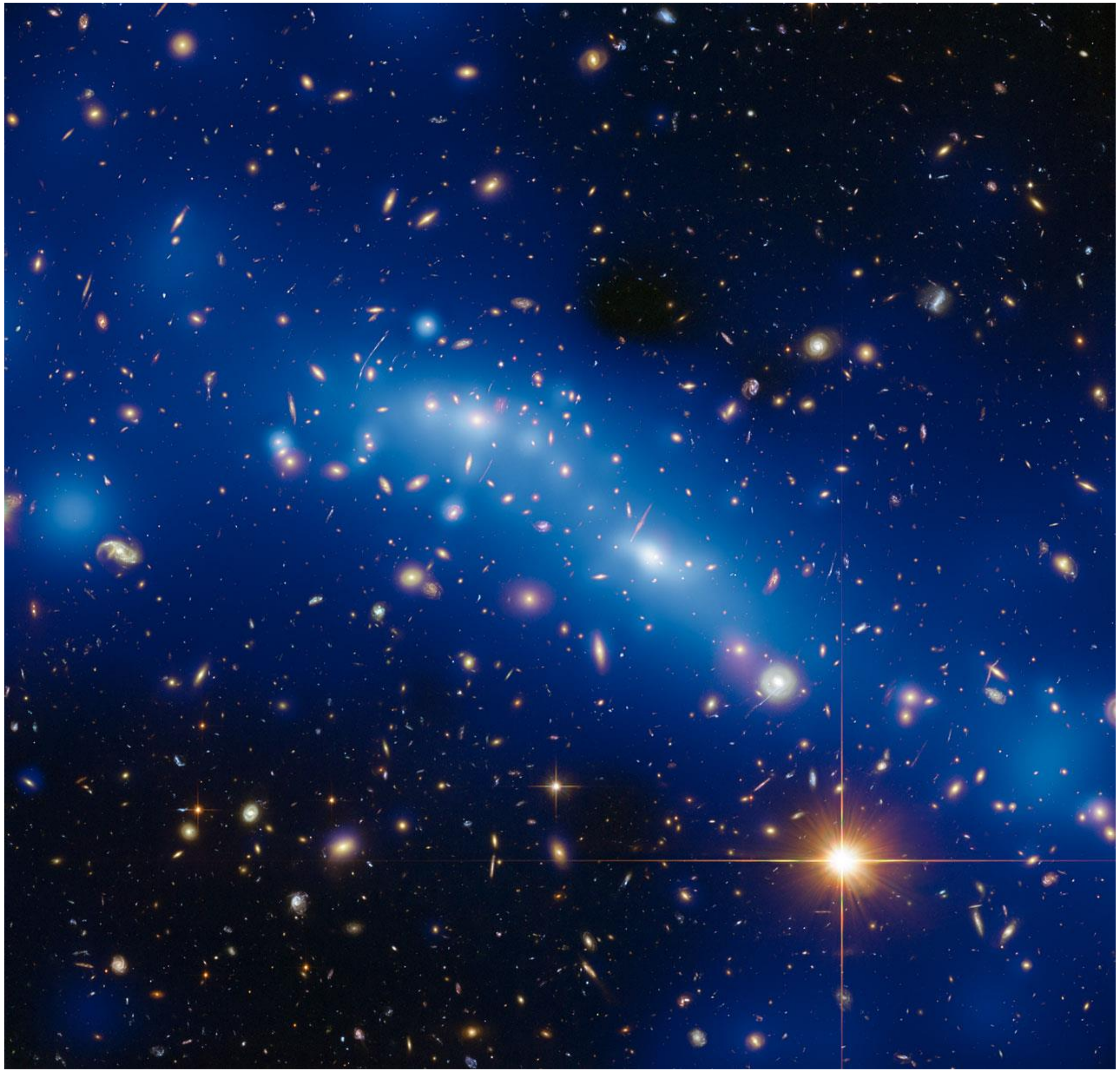
Brandon Coy-Naperville North High School

Mentors: Marcelle Soares-Santos and Huan lin

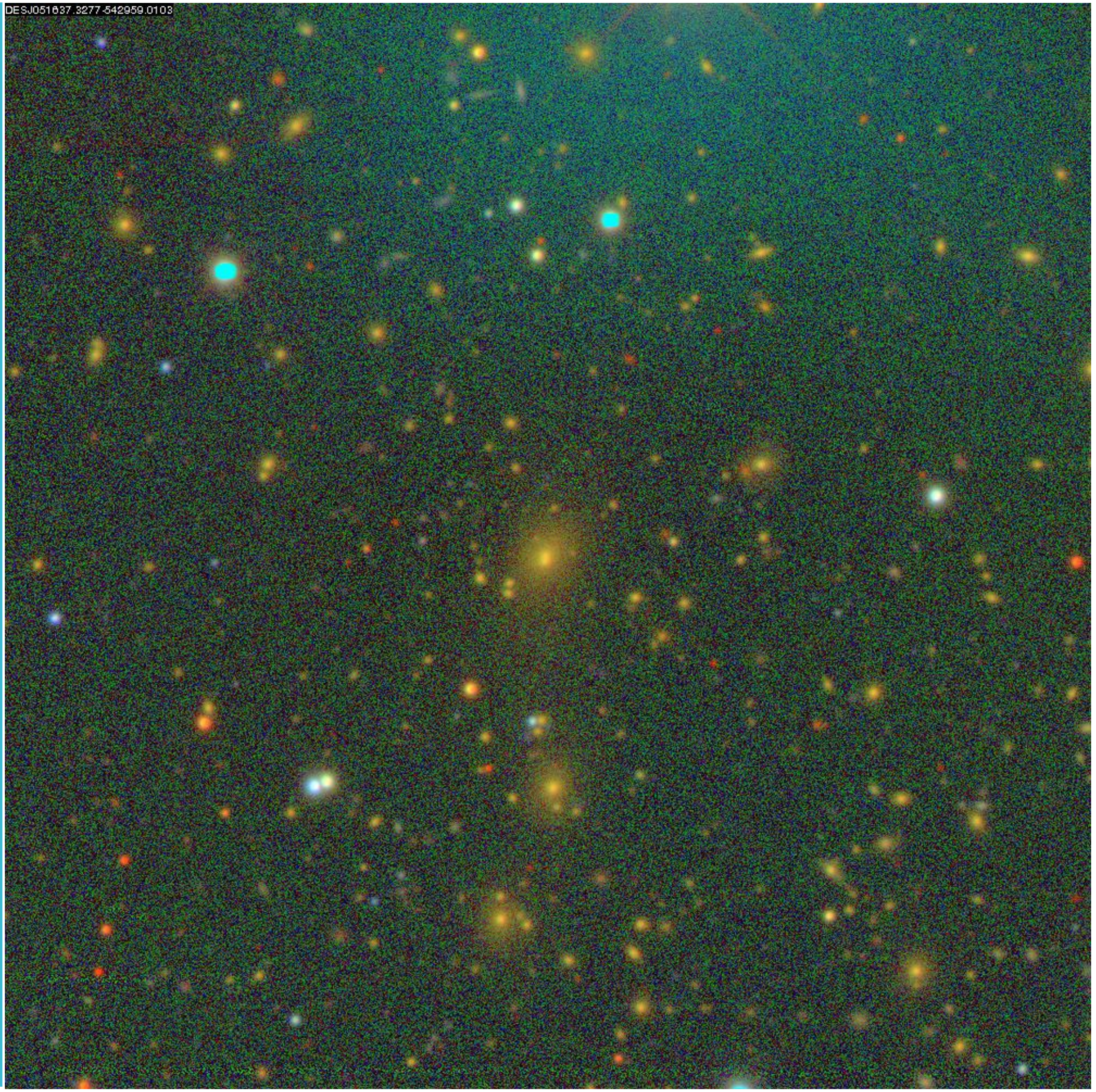
8/1/14

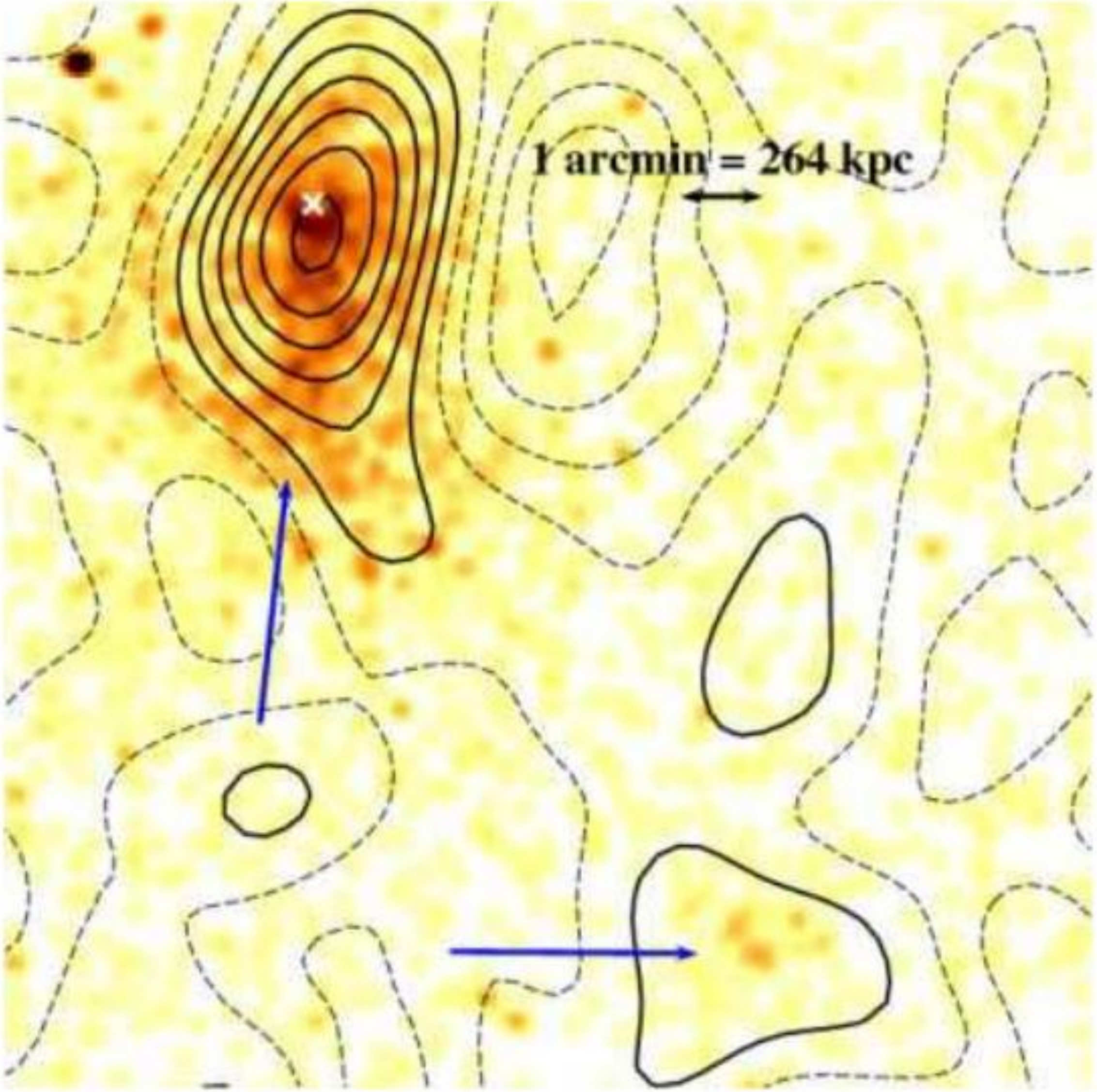
Purpose

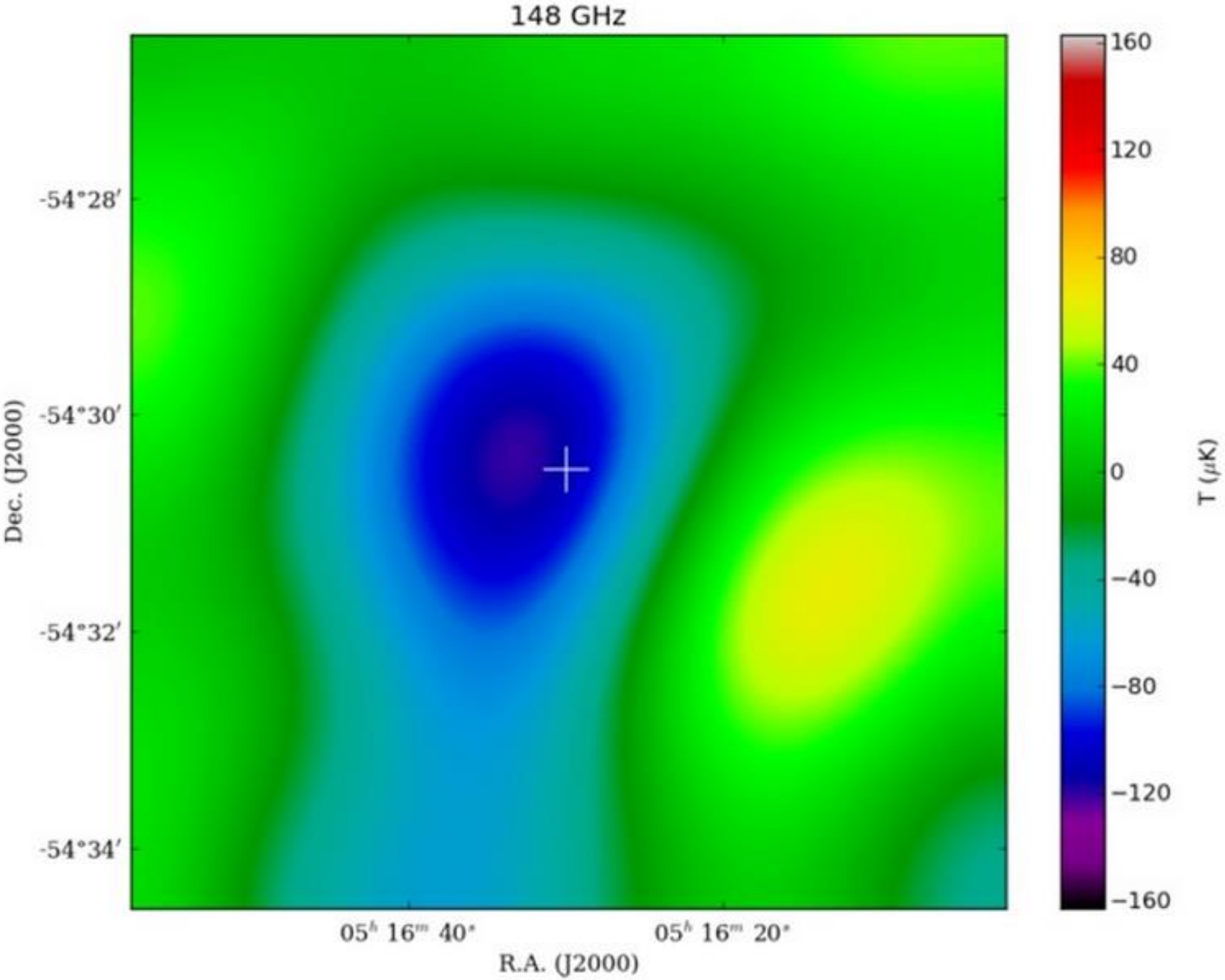
Validate galaxy clusters and data found by DES using the VT method and explain discrepancies found in mass and redshift



Hubble-Galaxy Cluster MACS J0416

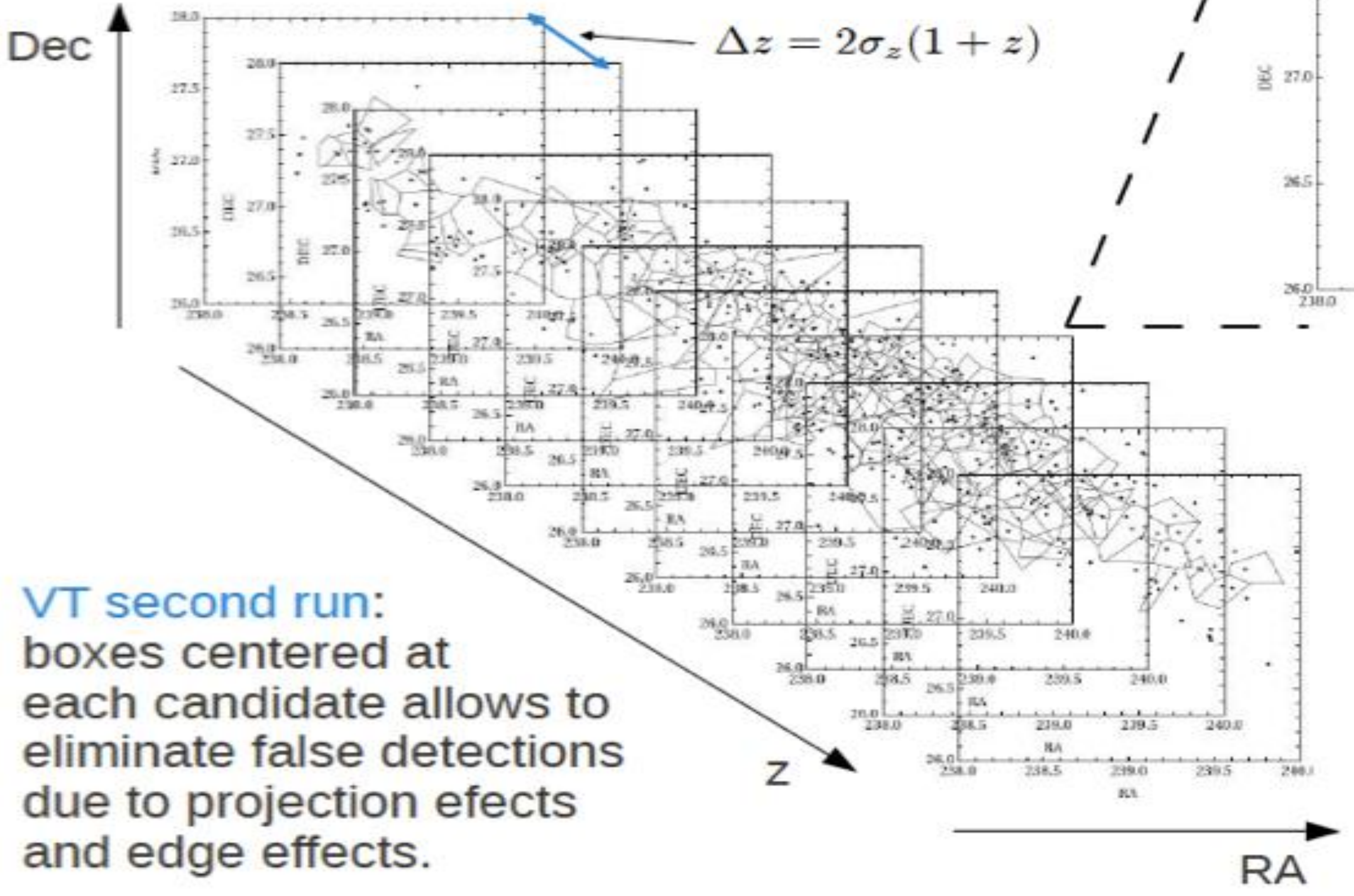

DESJ051637.3277-542959.0103

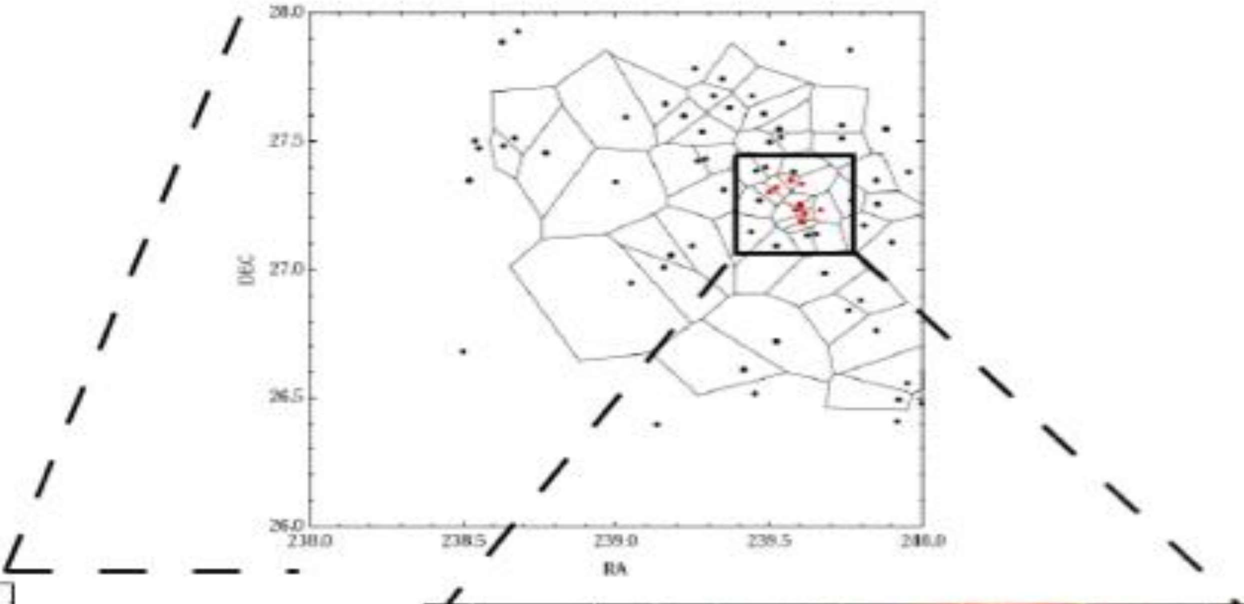

1 arcmin = 264 kpc


148 GHz
-54°28′
-54°30′
-54°32′
-54°34′
Dec. (J2000)
05h 16m 40s
05h 16m 20s
R.A. (J2000)
160
120
80
40
0
−40
−80
−120
−160
T (μK)

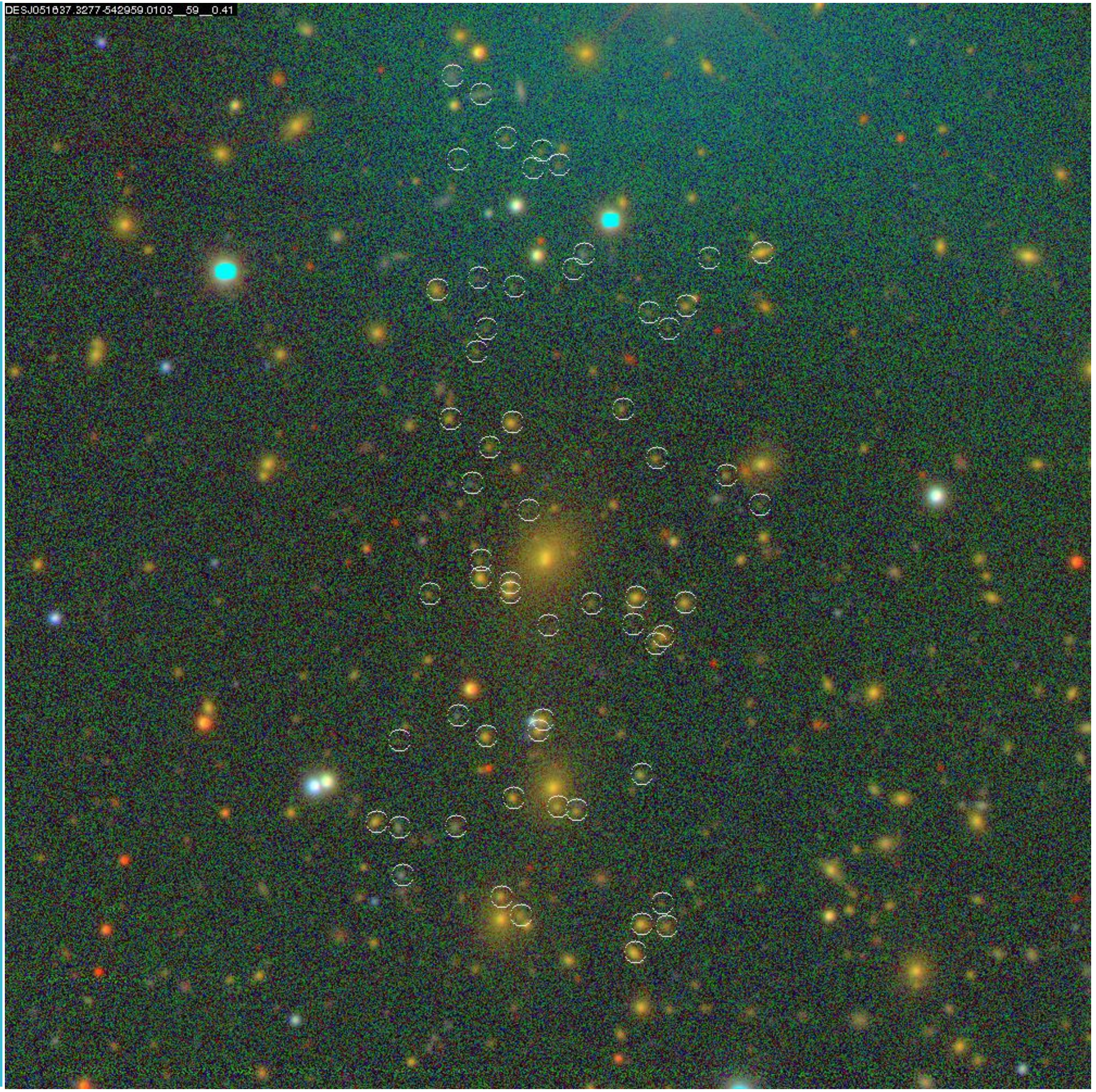
VT cluster finder in 2+1D

VT first run: cluster candidates detected in photo-z shells





VT second run: boxes centered at each candidate allows to eliminate false detections due to projection efects and edge effects.


DESJ051637.3277-542959.0103__59__0.41

Background-DES

DARK ENERGY SURVEY

- Sky survey-began collecting data late last year
- Cosmology-focused
- 5000 square degree survey planned-1500 so far
- Optical wavelengths (grizY bands): from ~400nm to ~1μm

Background-Galaxy clusters

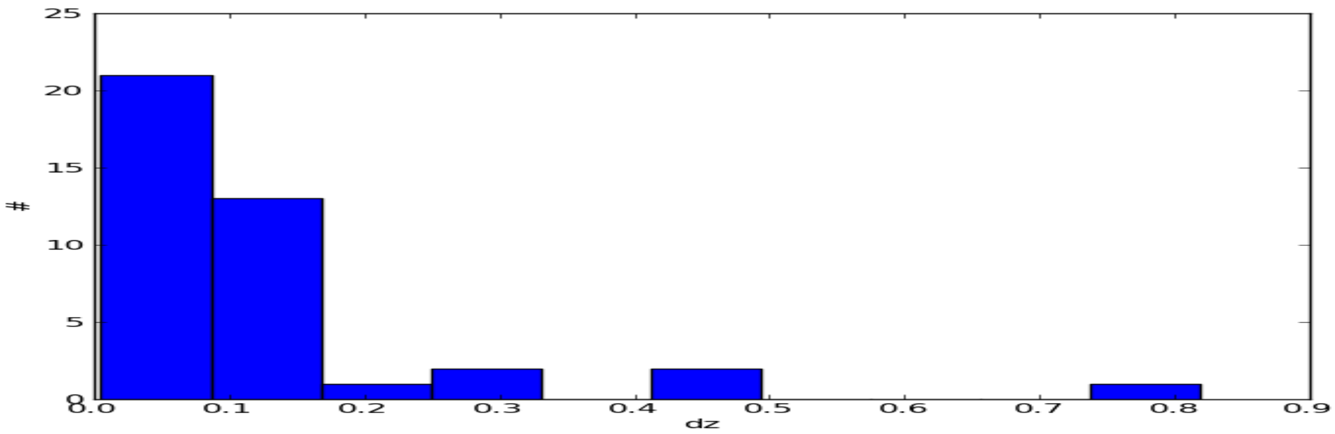
- Galaxy clusters are the largest structures in the universe
- Clusters are formed and held together by gravity
- Composed primarily of dark matter, gas, and galaxies
- Massive; usually 10^{13}-10^{15} solar masses

Catalogs Used

Data was taken from the following:

- Stripe 82 SDSS coadd VT Cluster Catalog
- DES 'gold' VT Cluster Catalog
- SDSS Max BCG Public Catalog
- XMM and MCXC X-ray Cluster Catalogs
- Hasselfield et al. (2013) (ACT-SZ)
- Song et al. (2012) (SPT-SZ)
- Ruel et al. (2013) (SPT-SZ)
- SZ in MaxVis, MainSPT, and SpecZ catalogs

Matching of Clusters



- Clusters were first matched by right ascension and declination using the Hierarchical Triangular Mesh method (a python module), with the cutoff match radius set at 1.5'
- Redshift differences-given redshifts in most catalogues tended to be given with an error of ~.03-.09, the cutoff of redshift differences was set to .2
- This was done by creating a 'box' around the to-be matched cluster of z, RA, and dec
- Nvt was also limited to be greater than 9 as smaller clusters tend to give less accurate results

Plotting Mass Versus Mass

- Estimated masses of each cluster were calculated using variables supplied by their respective catalogues (including y, luminosity, velocity dispersion, et cetera), or by the mass already estimated by the catalogue
- Equations for each method are included in their respective slides
- Masses for the VT clusters were obtained with the Weak Lensing methods devised by Matt Wiesner and Huan Lin
- For the 'gold' catalog, the following was used $M_{200c} = 1.44A(\frac{N_{VT}}{20})^B$
- Masses were plotted using matplotlib-a python matlab simulator

```python
import pyfits
from matplotlib.pyplot import *
import esutil
import math
from pylab import *
h=esutil.htm.HTM(10)
data1=pyfits.getdata("xray_in_westSPT.fits")
data4=pyfits.getdata("des_sval_gold_vt_1.1_clustercat_nvtmin_5.fit")
data2=pyfits.getdata("xray_in_maxvis.fits")
data3=pyfits.getdata("xray_in_mainSPT.fits")
ra4=data4.field('ra')
m1=[];m2=[];m3=[];m41=[];m42=[];m43=[]
dec4=data4.field('dec')
ra1=data1.field('ra');ra2=data2.field('ra');ra3=data3.field('ra')
dec1=data1.field('dec');dec2=data2.field('dec');dec3=data3.field('dec')
l1=data1.field('L500');l2=data2.field('L500');l3=data3.field('L500')
z1=data1.field('z');z2=data2.field('z');z3=data3.field('z');z4=data4.field('z')
m1c=[];m2c=[];m3c=[]
merr1=[]
merr2=[]
n4=data4.field('nvt')
m4c1=[];m4c2=[];m4c3=[]

def func(i):
 ma1=[];ma2=[]
 ma1,ma2,d=h.match(ra1[i],dec1[i],ra4,dec4,90./3600,maxmatch=-1)
 k=0
 while k<len(ma1):
  if math.fabs(z1[ma1[k]]-z4[ma2[k]])>.2 or n4[ma2[k]]<10:
   k+=1
  else:
   return i,ma2[k]
 return

for i in range(len(ra1)):
 if func(i)!=None:
  m1.append(i)
  m41.append(func(i)[1])


def func(i):
 ma1=[];ma2=[]
 ma1,ma2,d=h.match(ra2[i],dec2[i],ra4,dec4,90./3600,maxmatch=-1)
 k=0
 while k<len(ma1):
  if math.fabs(z2[ma1[k]]-z4[ma2[k]])>.2 or n4[ma2[k]]<10:
   k+=1
  else:
   return i,ma2[k]
 return
for i in range(len(ra2)):
 if func(i)!=None:
  m2.append(i)
  m42.append(func(i)[1])


def func(i):
 ma1=[];ma2=[]
 ma1,ma2,d=h.match(ra3[i],dec3[i],ra4,dec4,90./3600,maxmatch=-1)
 k=0
 while k<len(ma1):
  if math.fabs(z3[ma1[k]]-z4[ma2[k]])>.2 or n4[ma2[k]]<10:
   k+=1
  else:
   return i,ma2[k]
 return

for i in range(len(ra3)):
 if func(i)!=None:
  m3.append(i)
  m43.append(func(i)[1])



for j in range(len(m1)):
 om=.26*math.pow(1+z1[m1[j]],3)/(math.pow(1+z1[m1[j]],3)+.74)
 f=(.26*math.pow(1+z1[m1[j]],3)+.74)*math.pow((18*math.pi*math.pi+82*(om-1)-39*(om-1)*(om-1))/(18*math.pi*math.pi+82*(.26-1)-32*(.26-1)*(.26-1)),.5)
 m1c.append(math.pow(l1[m1[j]]/math.pow(f,7./3),.75))


for k in range(len(m2)):
 om=.26*math.pow(1+z2[m2[k]],3)/(math.pow(1+z2[m2[k]],3)+.74)
 f=(.26*math.pow(1+z2[m2[k]],3)+.74)*math.pow((18*math.pi*math.pi+82*(om-1)-39*(om-1)*(om-1))/(18*math.pi*math.pi+82*(.26-1)-32*(.26-1)*(.26-1)),.5)
 m2c.append(math.pow(l2[m2[k]]/math.pow(f,7./3),.75))


for l in range(len(m3)):
 om=.26*math.pow(1+z3[m3[l]],3)/(math.pow(1+z3[m3[l]],3)+.74)
 f=math.pow((.26*math.pow(1+z3[m3[l]],3)+.74),.5)*math.pow((18*math.pi*math.pi+82*(om-1)-39*(om-1)*(om-1))/(18*math.pi*math.pi+82*(.26-1)-32*(.26-1)*(.26-1)),.5)
 m3c.append(math.pow(l3[m3[l]]/math.pow(f,7./3),.75))


for i in range(len(m1)):
  if z4[m41[i]]<=.4:
   m=1.44*.338*math.pow(n4[m41[i]]/20.,.76)
   m4c1.append(m)
   if n4[m41[i]]>=20:
    merr1.append(m-1.44*.333*math.pow(n4[m41[i]]/20.,.66))
    merr2.append(1.44*.343*math.pow(n4[m41[i]]/20.,.86)-m)
   else:
    merr1.append(m-1.44*.333*math.pow(n4[m41[i]]/20.,.86))
    merr2.append(1.44*.343*math.pow(n4[m41[i]]/20.,.66)-m)
  elif z4[m41[i]]<=.7:
   m=1.44*.425*math.pow(n4[m41[i]]/20.,.82)
   m4c1.append(m)
   if n4[m41[i]]>=20:
    merr1.append(m-1.44*.422*math.pow(n4[m41[i]]/20.,.76))
    merr2.append(1.44*.428*math.pow(n4[m41[i]]/20.,.88)-m)
   else:
    merr1.append(m-1.44*.22*math.pow(n4[m41[i]]/20.,.88))
    merr2.append(1.4*.453*math.pow(n4[m41[i]]/20.,.76)-m)
  else:
   m=1.44*.445*math.pow(n4[m41[i]]/20.,.84)
   m4c1.append(m)
   if n4[m41[i]]>=20:
    merr1.append(m-1.44*.437*math.pow(n4[m41[i]]/20.,.64))
    merr2.append(1.44*.453*math.pow(n4[m41[i]]/20.,1.04)-m)
   else:
    merr1.append(m-1.44*.437*math.pow(n4[m41[i]]/20.,1.04))
    merr2.append(1.44*.453*math.pow(n4[m41[i]]/20.,.64)-m)


for i in range(len(m2)):
  if z4[m42[i]]<=.4:
   m=1.44*.338*math.pow(n4[m42[i]]/20.,.76)
   m4c2.append(m)
   if n4[m42[i]]>=20:
    merr1.append(m-1.44*.333*math.pow(n4[m42[i]]/20.,.66))
    merr2.append(1.44*.343*math.pow(n4[m42[i]]/20.,.86)-m)
   else:
    merr1.append(m-1.44*.333*math.pow(n4[m42[i]]/20.,.86))
    merr2.append(1.44*.343*math.pow(n4[m42[i]]/20.,.66)-m)
  elif z4[m42[i]]<=.7:
   m=1.44*.425*math.pow(n4[m42[i]]/20.,.82)
   m4c2.append(m)
   if n4[m42[i]]>=20:
    merr1.append(m-1.44*.422*math.pow(n4[m42[i]]/20.,.76))
    merr2.append(1.44*.428*math.pow(n4[m42[i]]/20.,.88)-m)
   else:
    merr1.append(m-1.44*.22*math.pow(n4[m42[i]]/20.,.88))
    merr2.append(1.4*.453*math.pow(n4[m42[i]]/20.,.76)-m)
  else:
   m=1.44*.445*math.pow(n4[m42[i]]/20.,.84)
   m4c2.append(m)
   if n4[m42[i]]>=20:
    merr1.append(m-1.44*.437*math.pow(n4[m42[i]]/20.,.64))
    merr2.append(1.44*.453*math.pow(n4[m42[i]]/20.,1.04)-m)
   else:
    merr1.append(m-1.44*.437*math.pow(n4[m42[i]]/20.,1.04))
    merr2.append(1.44*.453*math.pow(n4[m42[i]]/20.,.64)-m)


for i in range(len(m3)):
  if z4[m43[i]]<=.4:
   m=1.44*.338*math.pow(n4[m43[i]]/20.,.76)
   m4c3.append(m)
   if n4[m43[i]]>=20:
    merr1.append(m-1.44*.333*math.pow(n4[m43[i]]/20.,.66))
    merr2.append(1.44*.343*math.pow(n4[m43[i]]/20.,.86)-m)
   else:
    merr1.append(m-1.44*.333*math.pow(n4[m43[i]]/20.,.86))
    merr2.append(1.44*.343*math.pow(n4[m43[i]]/20.,.66)-m)
  elif z4[m43[i]]<=.7:
   m=1.44*.425*math.pow(n4[m43[i]]/20.,.82)
   m4c3.append(m)
   if n4[m43[i]]>=20:
    merr1.append(m-1.44*.422*math.pow(n4[m43[i]]/20.,.76))
    merr2.append(1.44*.428*math.pow(n4[m43[i]]/20.,.88)-m)
   else:
    merr1.append(m-1.44*.22*math.pow(n4[m43[i]]/20.,.88))
    merr2.append(1.4*.453*math.pow(n4[m43[i]]/20.,.76)-m)
  else:
   m=1.44*.445*math.pow(n4[m43[i]]/20.,.84)
   m4c3.append(m)
   if n4[m43[i]]>=20:
    merr1.append(m-1.44*.437*math.pow(n4[m43[i]]/20.,.64))
    merr2.append(1.44*.453*math.pow(n4[m43[i]]/20.,1.04)-m)
   else:
    merr1.append(m-1.44*.437*math.pow(n4[m43[i]]/20.,1.04))
    merr2.append(1.44*.453*math.pow(n4[m43[i]]/20.,.64)-m)

m4f=[]
mf=[]
for item in m4c1:
  m4f.append(item)


for item in m4c2:
  m4f.append(item)


for item in m4c3:
  m4f.append(item)
for item in m1c:
  mf.append(item)


for item in m2c:
  mf.append(item)


for item in m3c:
  mf.append(item)


errorbar(m4f,mf,xerr=[merr1,merr2],fmt=None)
carpe,diem=polyfit(m4f,mf,1)
#semilogx();semilogy()
ylim(0,1)
xlim(0,3)
plot([0,3],[diem,3*carpe+diem])
scatter(m4c1,m1c)
scatter(m4c2,m2c)
scatter(m4c3,m3c)
print carpe
print diem
id=data4.field('id')
xlabel("M200 (10^14 Solar Masses)")
ylabel("(L/(Fz^7/3))^(3/4)")
savefig('xraymatch.png')
f=open('table.txt','w')
for i in range(len(m41)):
 f.write(str(id[m41[i]])+'\t'+str(ra4[m41[i]])+'\t'+str(dec4[m41[i]])+'\t'+str(n4[m41[i]])+'\t'+str(z4[m41[i]])+'\t'+str(m4c1[i])+'\t'+str(m1c[i])+'\t-\t'+str(l1[m1[i]])+'\t-\t-\t'+

for i in range(len(m42)):
 f.write(str(id[m42[i]])+'\t'+str(ra4[m42[i]])+'\t'+str(dec4[m42[i]])+'\t'+str(n4[m42[i]])+'\t'+str(z4[m42[i]])+'\t'+str(m4c2[i])+'\t'+str(m2c[i])+'\t-\t'+str(l2[m2[i]])+'\t-\t-\t'+

for i in range(len(m43)):
 f.write(str(id[m43[i]])+'\t'+str(ra4[m43[i]])+'\t'+str(dec4[m43[i]])+'\t'+str(n4[m43[i]])+'\t'+str(z4[m43[i]])+'\t'+str(m4c3[i])+'\t'+str(m3c[i])+'\t-\t'+str(l3[m3[i]])+'\t-\t-\t'+

f.close()



total=0
for i in range(len(mf)):
 total+=mf[i]/m4f[i]

print total/len(mf)
total=[]
for i in range(len(mf)):
 total+=[mf[i]/m4f[i],]
total.sort()
print total[int((len(mf)-1)/2)]
```

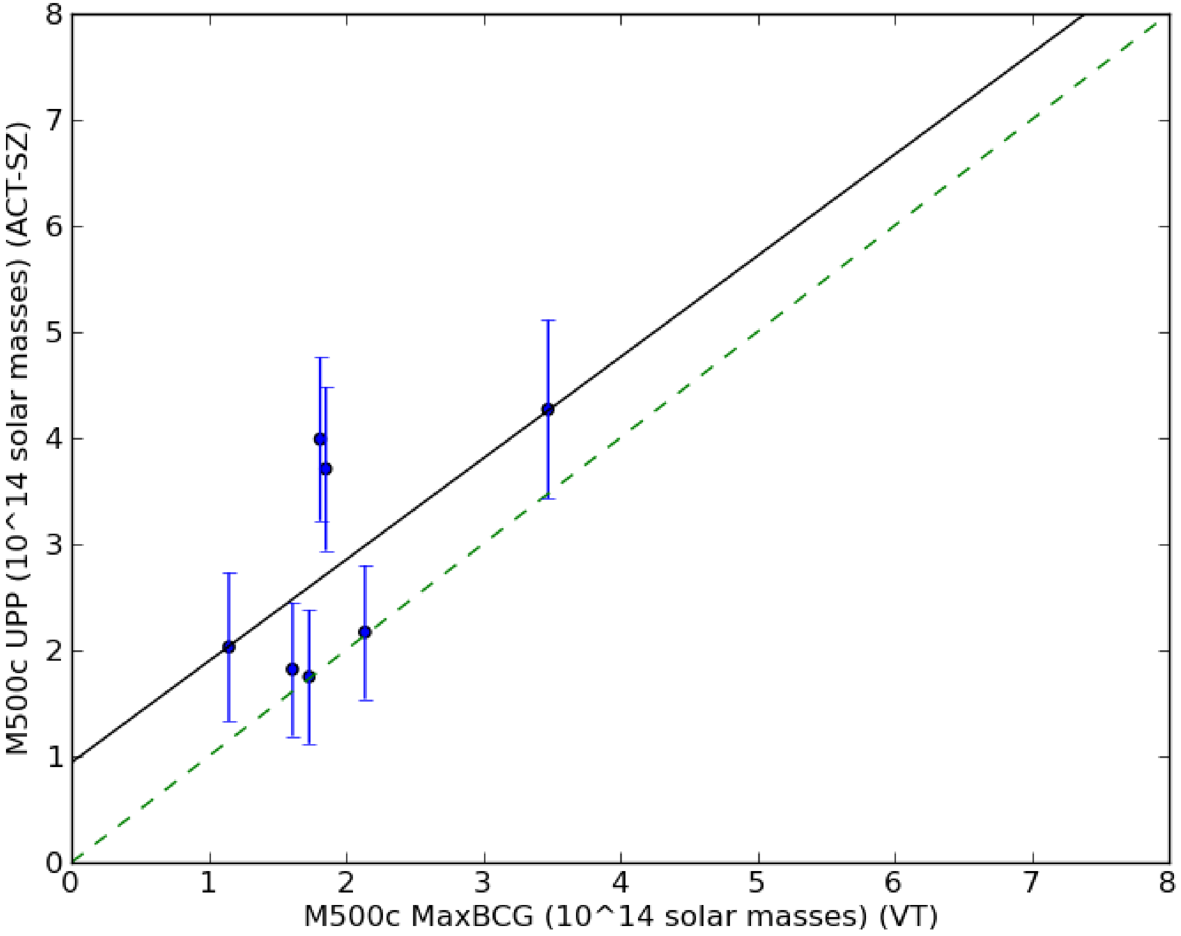
maxBCG vs ACT-SZ

Initial check

UPP

Average $\frac{M}{M_{VT}}$=1.48

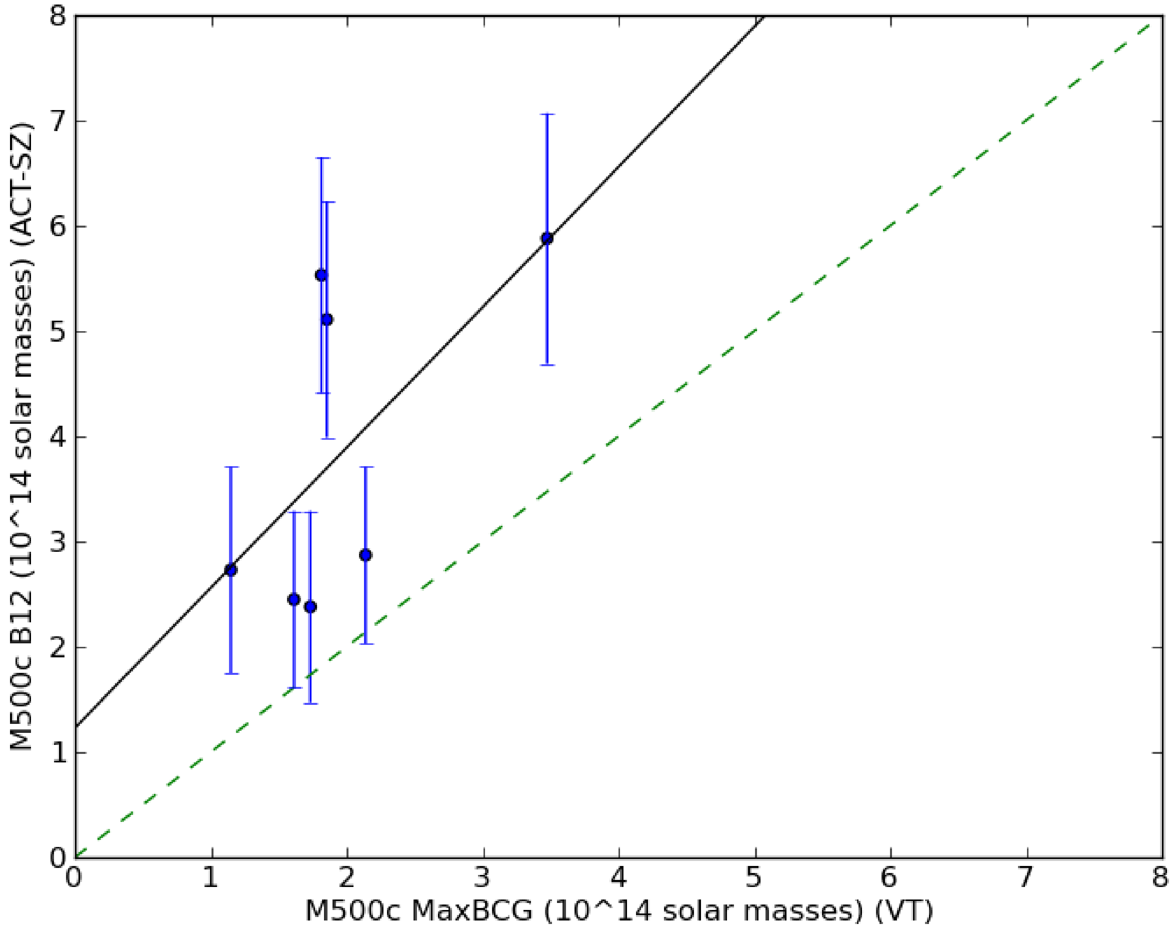
Median $\frac{M}{M_{VT}}$=1.23



b12

Average $\frac{M}{M_{VT}}$=2.02

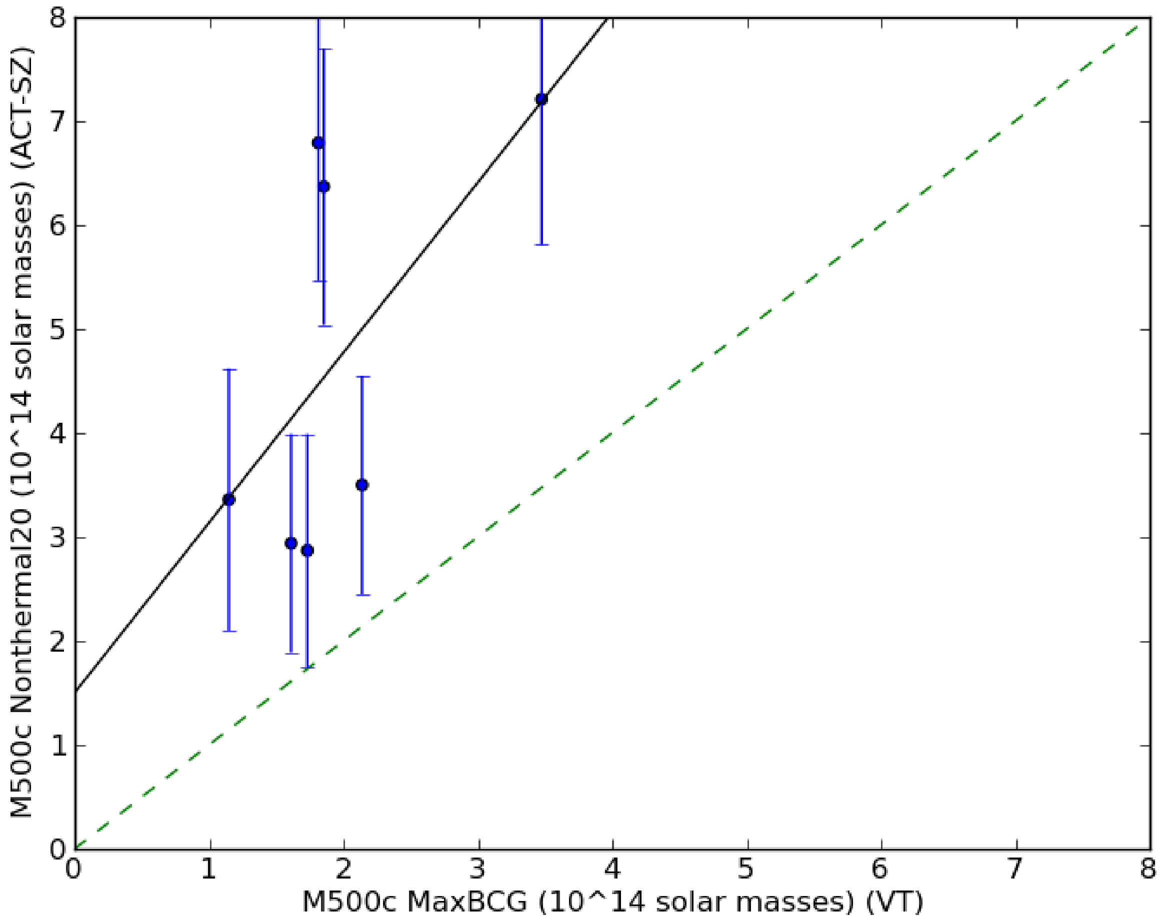
Median $\frac{M}{M_{VT}}$=1.69



Nonthermal20

Average $\frac{M}{M_{VT}}$=2.47

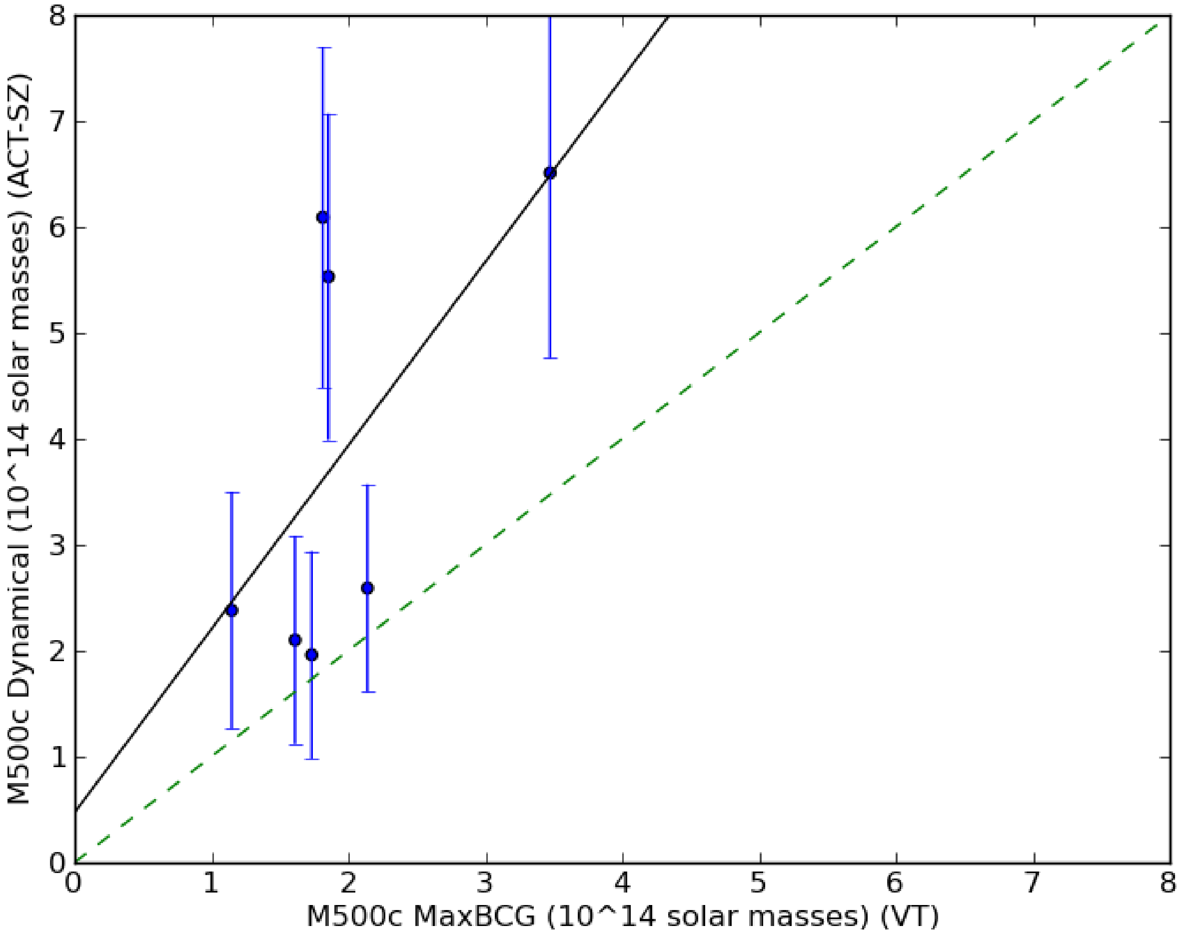
Median $\frac{M}{M_{VT}}$=2.08



Dynamical

Average $\frac{M}{M_{VT}}$=1.99

Median $\frac{M}{M_{VT}}$=1.87

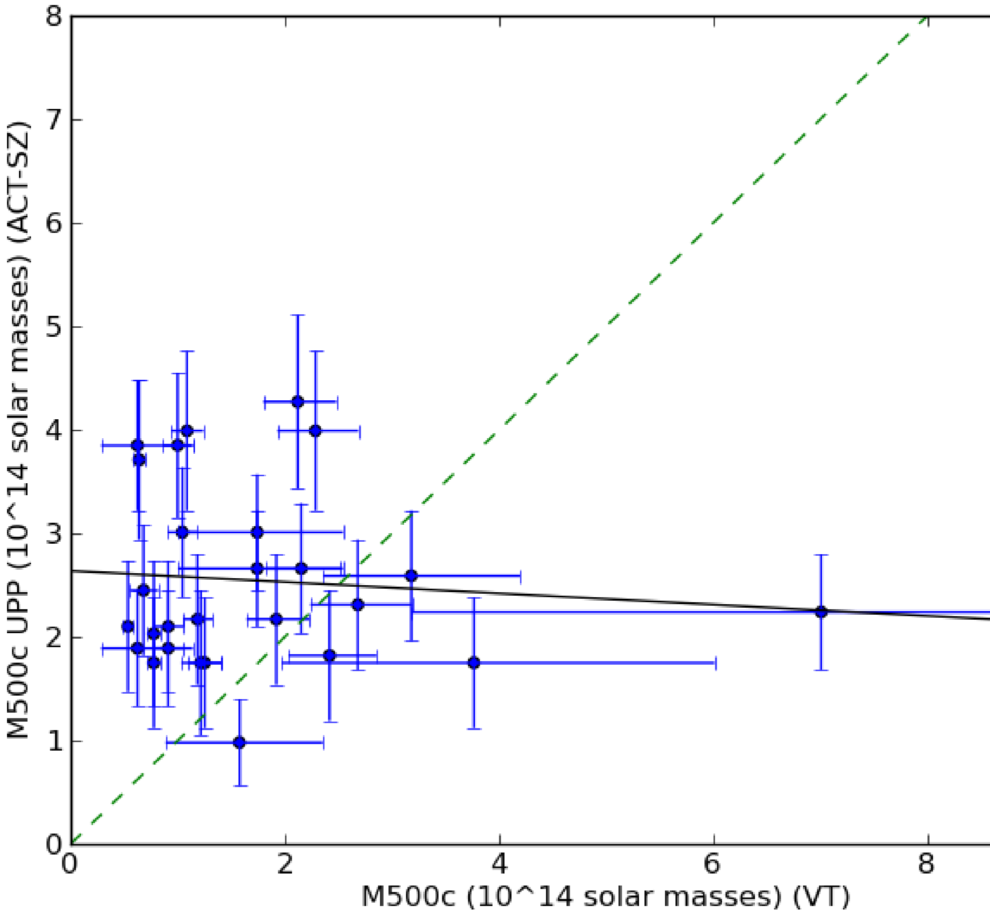


Mass UPP

Average $\frac{M}{M_{VT}}$ =2.23

Median=1.83

UPP Mass is based on the y Parameter (For Stripe 82 clusters, $M_{200c} = A(\frac{N_{VT}}{20})^B$ was used)

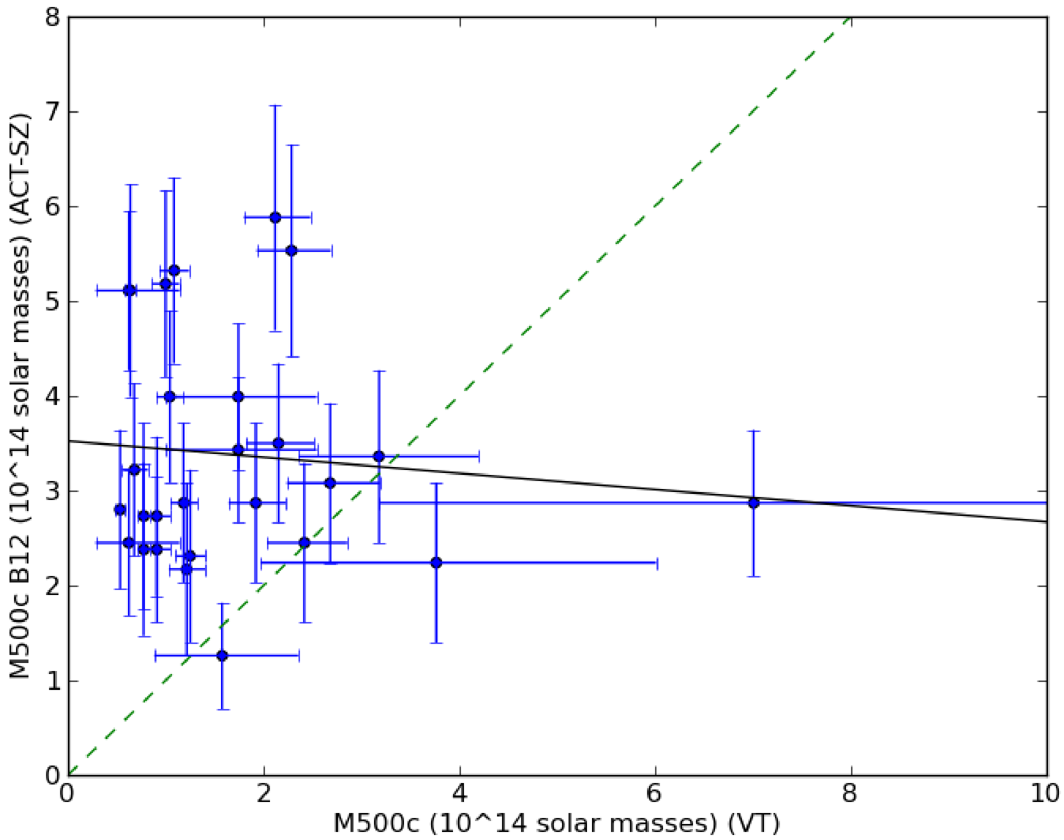


Mass B12

Average $\frac{M}{M_{VT}}$ =2.96

Median=2.43

Based on the B12 model



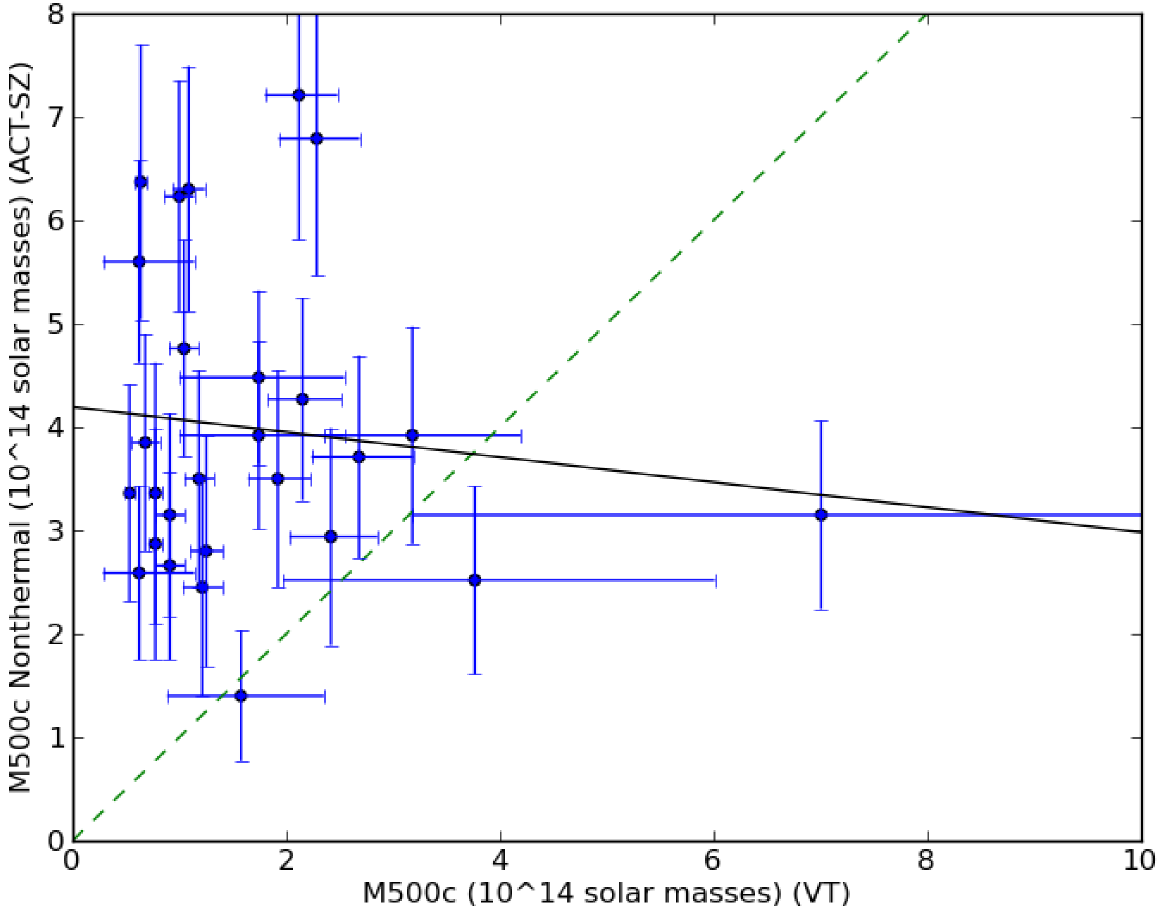
Nonthermal20 mass

Average $\frac{M}{M_{VT}}$ =3.49

Median=2.96

Based on the

Nonthermal20 model

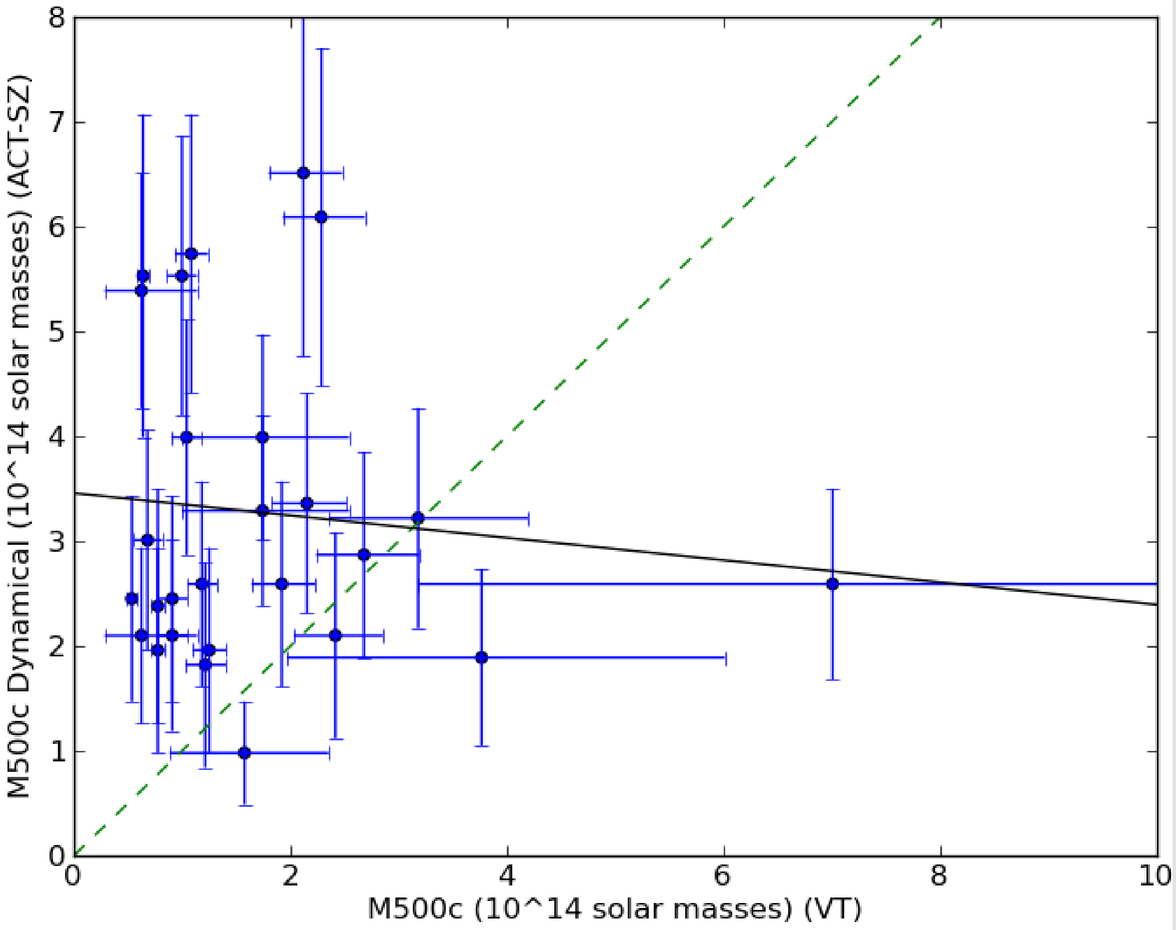


Dynamical mass

Average $\frac{M}{M_{VT}}$=2.87

Median=2.31

Dynamical Mass is based off of velocity dispersion and is detailed in Sifon et al. 2011



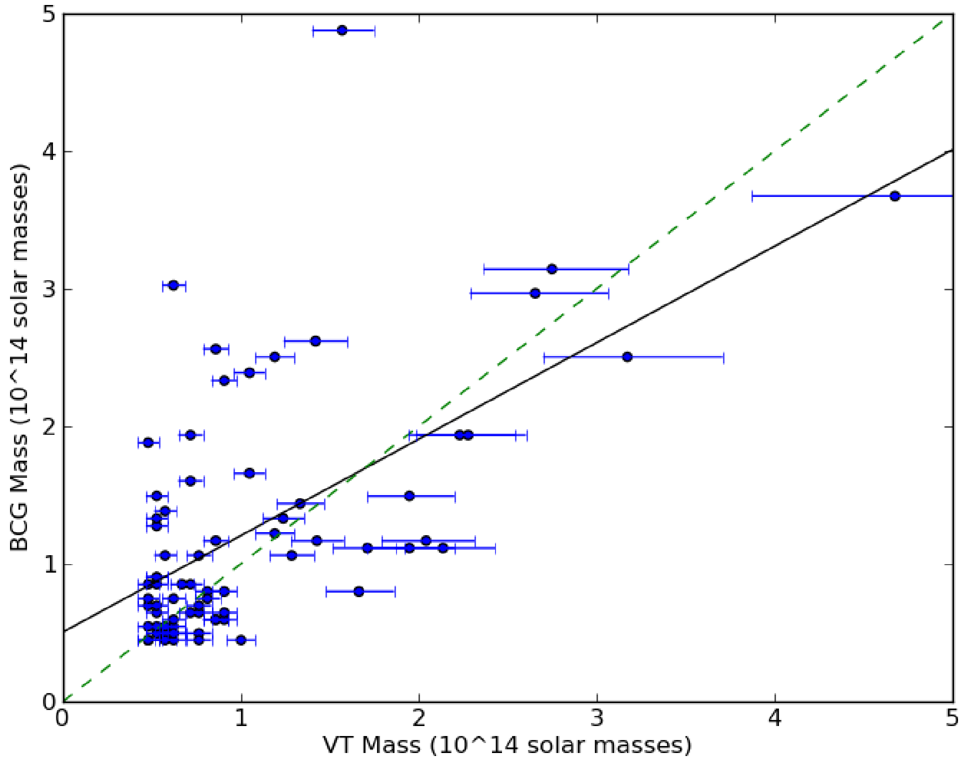
MAXBCG

- $M_{200c} = M_{200|20}(\frac{N_{200}}{20})^{\alpha}$

Equation 9 of Simet et al. 2012

Average $\frac{M}{M_{VT}}$ =1.34

Median=.98



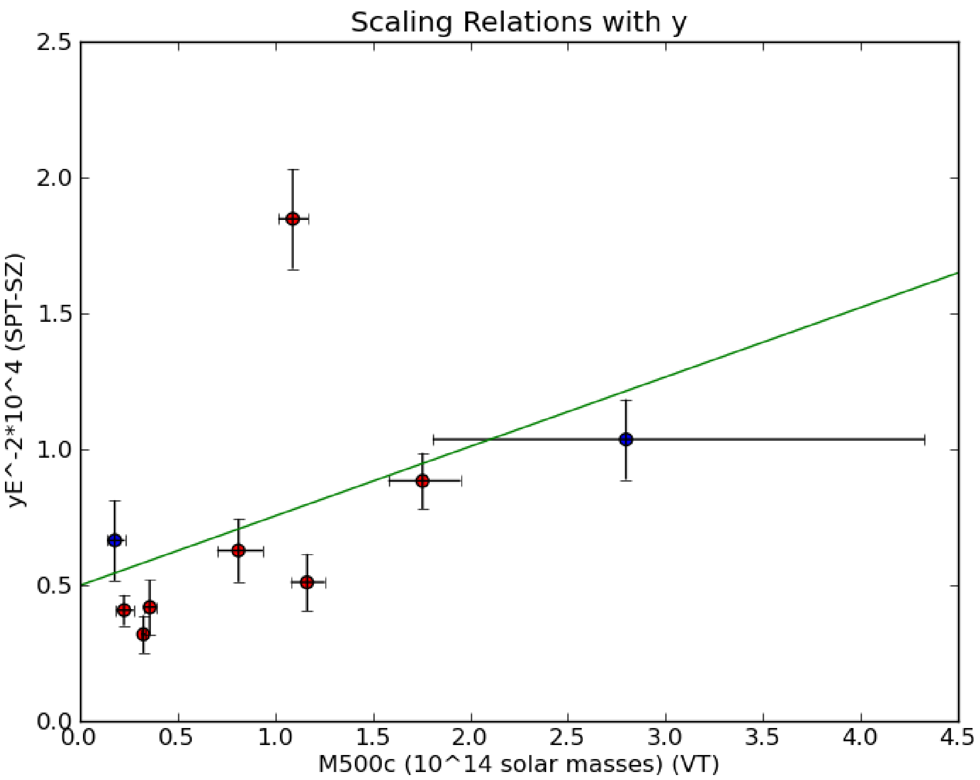
Y (SPT/ACT-SZ)

$yE_z^{-2} \alpha M_{500c}$

Eq. A4 of Marriage et al. 2011

Slope=.2556

Y-intercept=.4989



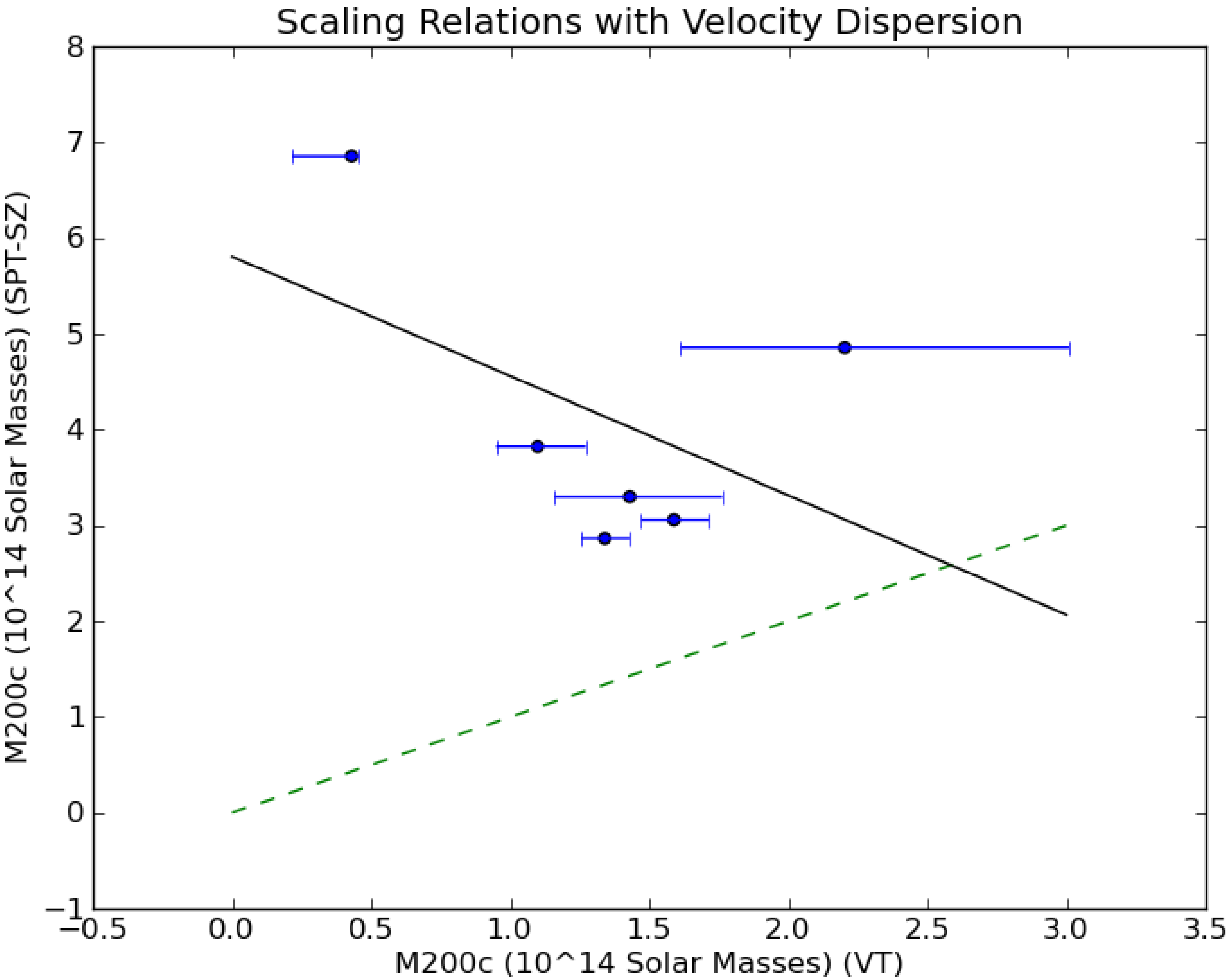
Velocity Dispersion (SPT-SZ)

$$\frac{M_{200c}}{10^{14} M_{\odot}} = \frac{10}{h(z)} \left(\frac{\sigma_{DM}}{\sigma_{15}}\right)^{\frac{1}{\alpha}}$$

Eq. 2 of Buckley-Geer et al. 2011

(Originally Evrard et al.)

Average $\frac{M}{M_{VT}}$=4.67, median=2.20

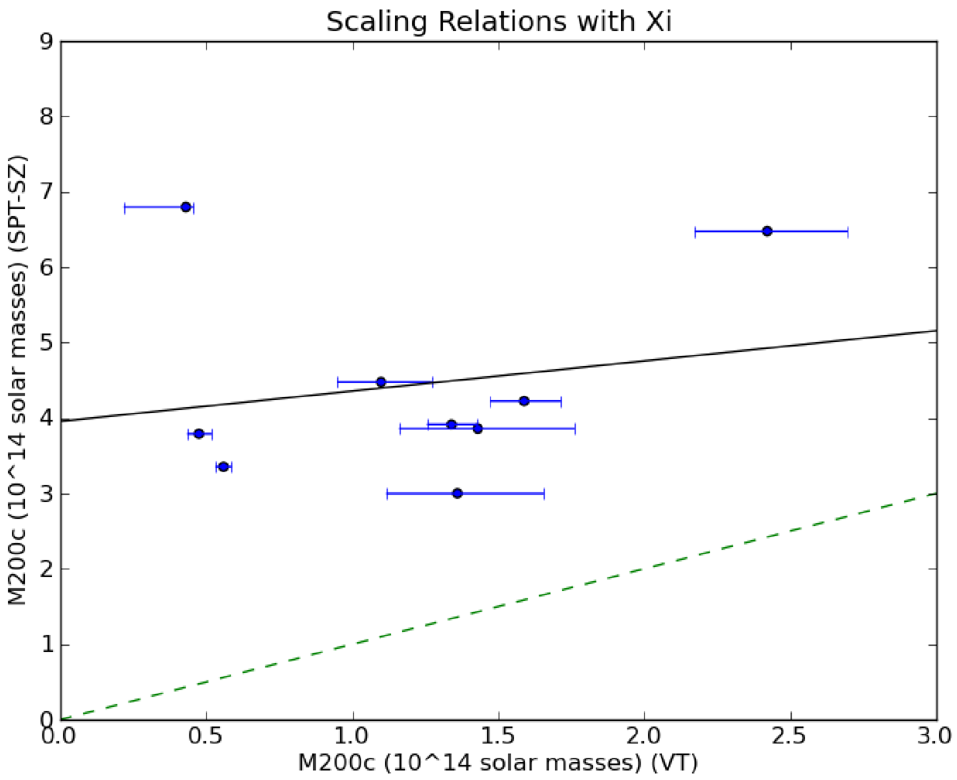


ξ (SPT-SZ)

$$\sqrt{<\xi>^2-3} = A\left(\frac{M}{5*10^{14}M_{\odot}h^{-1}}\right)^B\left(\frac{1+z}{1.6}\right)^C$$

Equation 1 of Vanderlinde et al 2010

Average $\frac{M}{M_{VT}}$ =5.22, median=2.92



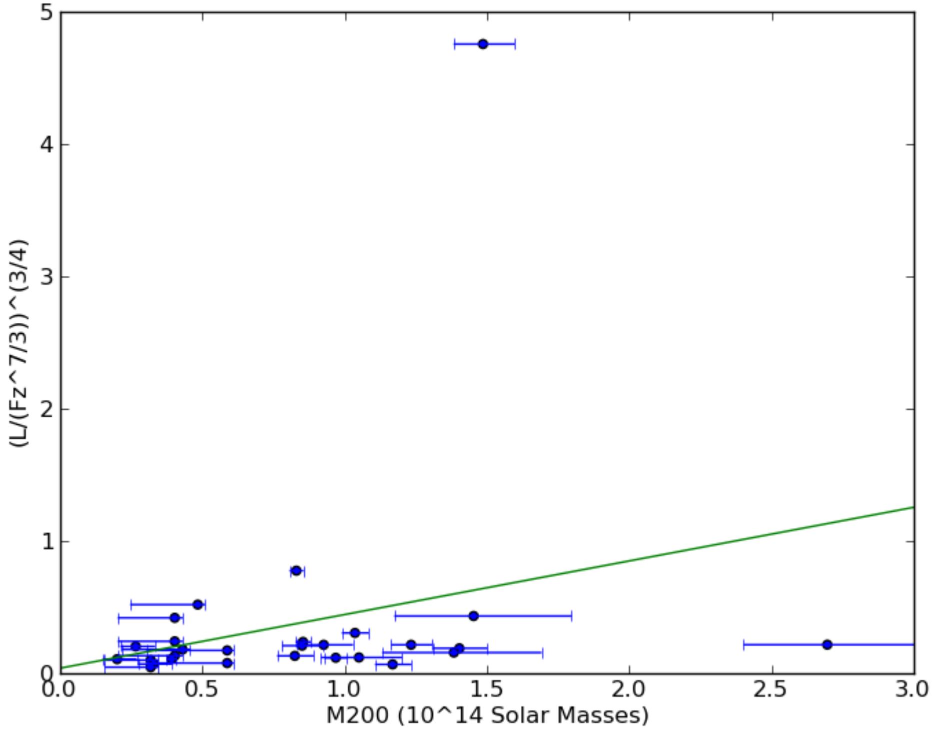
Luminosity (XMM& MCXC)

- $L_X \propto F_Z^{\frac{7}{3}} M_{total}^{\frac{4}{3}}$

Equation 16 of Giodini et al 2013

Slope=0.405

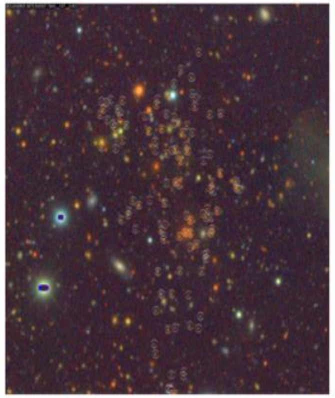
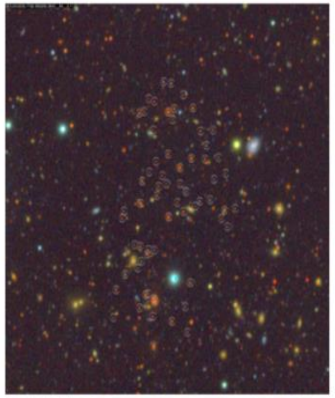
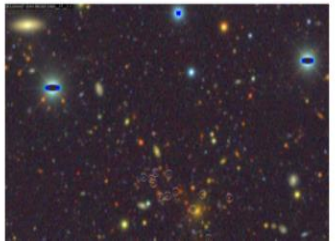
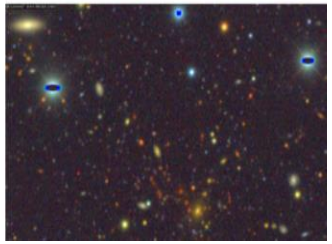
Y-Intercept=0.039



Website

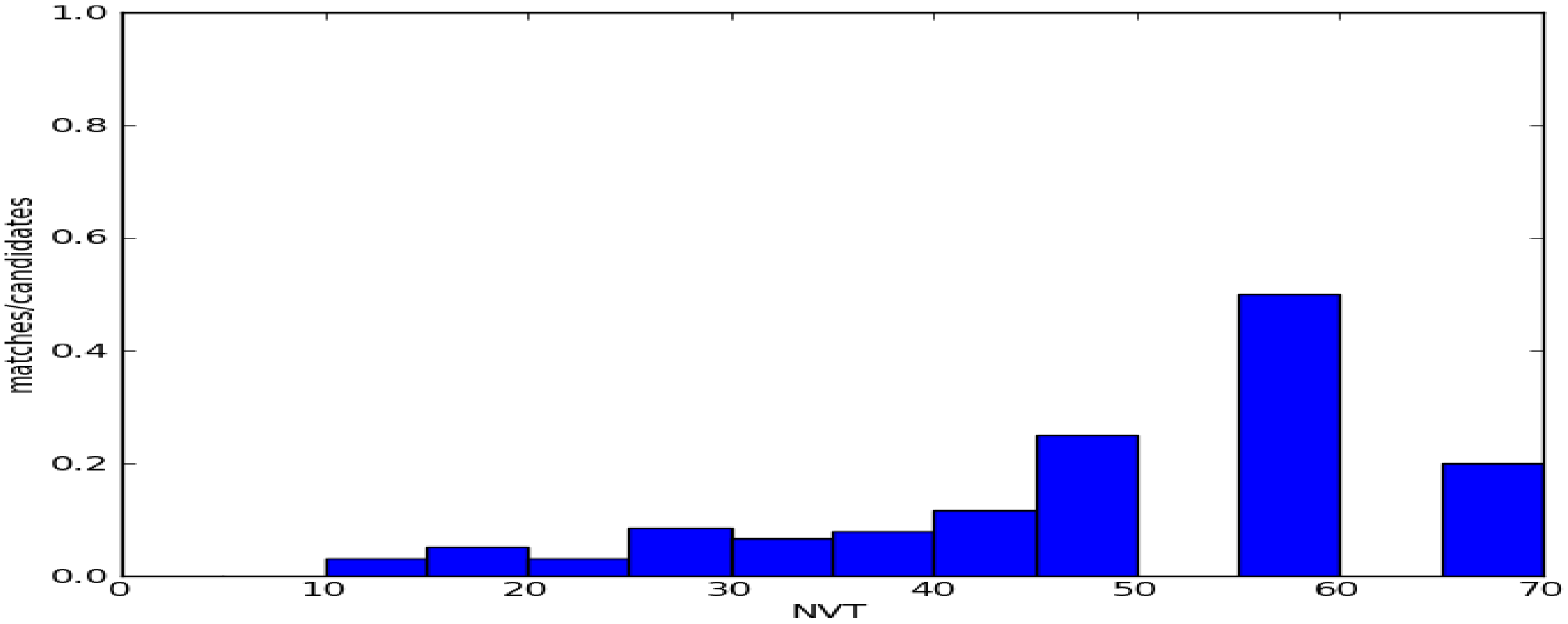
Data (including redshift, RA, dec, mass, etc.) was compiled into an html file including pictures of clusters

Site makes for quick and easy access of findings

VT ID	RA	Dec	Nvt	Redshift(VT)	M200c (VT)	M200c	ξ	Lx (10^44 ergs/s)	y (10^-4)	Velocity Dispersion (km/s)	Redshift	DES Image (with bubbles)	DES Image (w/o bubbles)
10665500502642	66.5245731935	-54.9188317817	107	0.63	2.4210	6.4763	8.86	-	-	-	0.63		
10685700502512	68.249080204	-56.4998233347	64	0.7	1.5885	4.2240,3.0584	5.35	-	-	817	0.65,0.691915		
10715700900442	71.1133536745	-56.0570249659	17	1.1	0.5590	3.3538	5.3	-	-	-	0.98		

Reasoning for discrepancy

- Many VT clusters did not find matches of other established catalog, indicating they may be fake or unreal clusters
- Equations across separate papers are not consistent



Conclusions

- In every case, the mass predicted by weak lensing was less than the mass predicted by the respective relationship for each catalogue, by a semi-consistent factor of ~2-3
- Most plots show a weak correlation of masses, suggesting a problem with VT mass calibration

Acknowledgements

- Marcelle,Huan
- Chris,George,Ian
- Quarknet Interns