In [2]:

```
# Import various libraries

import numpy as np
import astropy
import photutils
import ccdproc
from ccdproc import CCDData, combiner
from astropy import units as u
import astropy.io.fits as fits
from astropy.visualization import SqrtStretch
from astropy.visualization.mpl_normalize import ImageNormalize
import matplotlib.pyplot as plt
from matplotlib.colors import LogNorm
from photutils import centroid_com, centroid_1dg, centroid_2dg
from photutils import CircularAperture
from photutils import aperture_photometry
from photutils import Background2D
from photutils import MedianBackground
from photutils import DAOStarFinder
from photutils import detect_sources, deblend_sources, source_properties
from scipy.ndimage import shift
import gc
from matplotlib.pyplot import *
from numpy import *

from astropy.coordinates import SkyCoord
from astroquery.gaia import Gaia
```

```
Created TAP+ (v1.2.1) - Connection:
        Host: gea.esac.esa.int
        Use HTTPS: True
        Port: 443
        SSL Port: 443
Created TAP+ (v1.2.1) - Connection:
        Host: geadata.esac.esa.int
        Use HTTPS: True
        Port: 443
        SSL Port: 443
```

In [3]:

```
# Here's the star class, which includes a name, coordinates, exposure time, airmass,

class Star:
    def __init__(self):
            self.name = "Default name"                                    # Object name
            self.filename = "NULL                                    "   # Image file r
            self.ra = -99.0                                               # J2000 coords
            self.dec = -99.0
            self.x = -99.0                                                # CCD pixel co
            self.y = -99.0
            self.flux = [-99.0,-99.0,-99.0,-99.0,-99.0]                   # UBVRI Fluxes
            self.mag = [-99.0,-99.0,-99.0,-99.0,-99.0]                    # UBRRI Magnit
            self.gaiamag = [-99.0, -99.0, -99.0]                          # G_BP, G and
            self.exptime = 1.0
            self.airmass = 1.0
```

In [4]:

```
# This reads in the photometry from the Graham (1982) standard star photometry and c

def readgraham(fname):

  # Set the list of standards to a blank list.
    gstandards=[]

    # Open the text file defined by fname, then import the ASCII lines from the file
    with open(fname) as f:
        grahamtext = f.readlines()

    # Step through each line of grahamtext. The current line is thisline and it has
    for imx, thisline in enumerate(grahamtext):

        # Read in coordinates - each variable has specific columns, defined in the F
        rah=float(thisline[15:17])   # Read in RA hours from the ASCII, and covert t
        ram=float(thisline[18:20])   # Read in RA minutes from the ASCII, and covert
        ras=float(thisline[21:25])   # Read in RA seconds from the ASCII, and covert
        decd=float(thisline[27:29])  # Read in Dec degreess from the ASCII, and conv
        decm=float(thisline[30:32])  # Read in Dec arcminutes from the ASCII, and co
        decs=float(thisline[33:35])  # Read in Dec arcseconds from the ASCII, and co

        # Convert the coordinates to decimal degrees.
        ra=15.0*(rah+ram/60.0+ras/3600.0)   # Convert RA to decimal degrees
        dec=decd+decm/60.0+decs/3600.0      # Convert Declination to decimal degrees
        if thisline[26]=='-':               # If the declination is negative, conver
            dec=0-dec

        # Read in the photometry
        tmag=[-99.0,-99.0,-99.0,-99.0,-99.0]        # Temporary UBVRI magnitude list
        tmag[2]=float(thisline[36:41])              # Read V magnitude from thisline
        tmag[1]=float(thisline[50:56])+tmag[2]      # Read in B-V colour from thisli
        if (thisline[47]!=' '):
            tmag[0]=float(thisline[50:56])+tmag[1]  # Read in U-B colour from thisli
        tmag[3]=tmag[2]-float(thisline[57:62])      # Read in V-R colour from thisli
        if (thisline[68]!=' '):
            tmag[4]=tmag[3]-float(thisline[64:69])  # Read in R-I colour from thisli

        # Create temporary tstandard, which has class star, and populate it with dat
        tstandard=Star()                            # Create tstandard with class st
        tstandard.name=thisline[0:12]               # Get the name from thisline
        tstandard.ra=ra                             # Get the RA
        tstandard.dec=dec                           # Get the Declination
        tstandard.mag=tmag                          # Get the UBVRI magnitudes

        # Extend the list gstandards with tstandard
        gstandards.extend([tstandard])

    # The end of the indentation is where there's the end of the for-loop

    # The end of the function, which returns the list of Graham Standards names, coo
    return gstandards
```

In [5]:

```
gstandards=readgraham('Graham1982.txt')
print(len(gstandards))
print(gstandards[0].ra, gstandards[0].dec, gstandards[0].mag)
```

```
102
21.174583333333334 -44.52833333333333 [8.562, 7.4159999999999995, 6.2
7, 5.699999999999999, 5.1899999999999995]
```

In [6]:

```
def getgaia(gstandards):
    for obj in gstandards:
        coord = SkyCoord(ra=obj.ra, dec=obj.dec, unit=(u.degree, u.degree), frame='i
        radius = u.Quantity(0.001, u.deg)
        j = Gaia.cone_search_async(coord, radius)
        r = j.get_results()
        if len(r)>0:
            obj.gaiamag[0]=r[0]['phot_bp_mean_mag']
            obj.gaiamag[1]=r[0]['phot_g_mean_mag']
            obj.gaiamag[2]=r[0]['phot_rp_mean_mag']
    return gstandards
```

In [7]:

```
grgaia=[]
f=open("graham_gaia.csv", "r")
grahamtext = f.readlines()
for line in grahamtext:
    currentline = line.split(",")
    if currentline[0]!="Name":
        tstandard=Star()
        tstandard.name=currentline[0]
        tstandard.ra=float(currentline[1])
        tstandard.dec=float(currentline[2])
        tstandard.mag[0]=float(currentline[3])
        tstandard.mag[1]=float(currentline[4])
        tstandard.mag[2]=float(currentline[5])
        tstandard.mag[3]=float(currentline[6])
        tstandard.mag[4]=float(currentline[7])
        tstandard.gaiamag[0]=float(currentline[8])
        tstandard.gaiamag[1]=float(currentline[9])
        tstandard.gaiamag[2]=float(currentline[10])
        grgaia.extend([tstandard])
f.close()
```

the RA and dec of the reference star are (137.79726, -64.8156)

In [8]:

```
gmag=[-99.0, -99.0, -99.0]
coord = SkyCoord(ra=137.79726, dec=-64.8156, unit=(u.degree, u.degree), frame='icrs'
radius = u.Quantity(0.001, u.deg)
j = Gaia.cone_search_async(coord, radius)
r = j.get_results()
if len(r)>0:
    gmag[0]=r[0]['phot_bp_mean_mag']
    gmag[1]=r[0]['phot_g_mean_mag']
    gmag[2]=r[0]['phot_rp_mean_mag']
    print(gmag)
```

```
INFO: Query finished. [astroquery.utils.tap.core]
[11.795125, 11.157327, 10.412787]
```

these values agree with Aladin, which has corresponding values:

Aladin: g magnitude - 11.154999 bp magnitude - 11.70615 rp magnitude - 10.402192

In [9]:

```
# px and py are x and y axis
px=[]
py=[]
for obj in grgaia:
    if obj.mag[2]>0.0 and obj.mag[2]<20.0 \
    and obj.gaiamag[0]>0.0 and obj.gaiamag[0]<20.0 \
    and obj.gaiamag[1]>0.0 and obj.gaiamag[1]<20.0 \
    and (obj.mag[2]-obj.gaiamag[1])**2<2.25 :
        px.append(obj.gaiamag[0]-obj.gaiamag[1])      # G_BP minus G
        py.append(obj.mag[2]-obj.gaiamag[1])          # V minus G
```

In [10]:

```
# we can find the magnitude for the V, R and I bands using the relations given on mo
# V - G = 0.00858 + 0.22184 (GBP - G) + 0.48446 (GBP - G)2
# R - GRP = 0.01283 + 0.33460 (G - GRP) + 0.40048 (G - GRP)2
# I - GRP = 0.00216 - 0.14069 (G - GRP)
# B - GBP = -0.03308 + 1.41139 (GBP - G)


# V band magnitue
v=0.00858+0.22184*(gmag[0]-gmag[1])+0.48446*(gmag[0]-gmag[1])**2+gmag[1]
print('V band magnitude')
print(v)

# R band magnitude
R=0.01283+0.33460*(gmag[1]-gmag[2])+0.40048*(gmag[1]-gmag[2])**2+gmag[2]
print('R magnitude')
print(R)

# I band magnitude
I=0.00216 - 0.14069*(gmag[1] - gmag[2])+gmag[2]
print('I magnitude')
print(I)

# B magnitude
B=-0.03308+1.41139*(gmag[0]-gmag[1])+gmag[0]
print('B magnitude')
print(B)

Bref=B
Vref=v
Rref=R
Iref=I
```

```
V band magnitude
11.504467751874088
R magnitude
10.896741841092728
I magnitude
10.310198208827972
B magnitude
12.662227163429261
```

Starting with cataloguing V band magnitudes

We start by creating a catalogue of the stars in our image

In [11]:

```
# creating a catalogue for the stars in NGC 2808
# based on lab 3 which used segmentation
scim=CCDData.read("NGC_2808_V_median.fits", unit="adu") # reading the data from the
med=np.median(scim.data) # defining the median of the data
scim.data=scim.data-med #subtracting median
mean, median, std = astropy.stats.sigma_clipped_stats(scim.data, sigma=3.0, maxiters
print('Image stats (mean, median and standard deviation):', mean,median,std) # calcu

segimage=detect_sources(scim.data, 2.00*std, 9, connectivity=4, mask=None) #defining
```

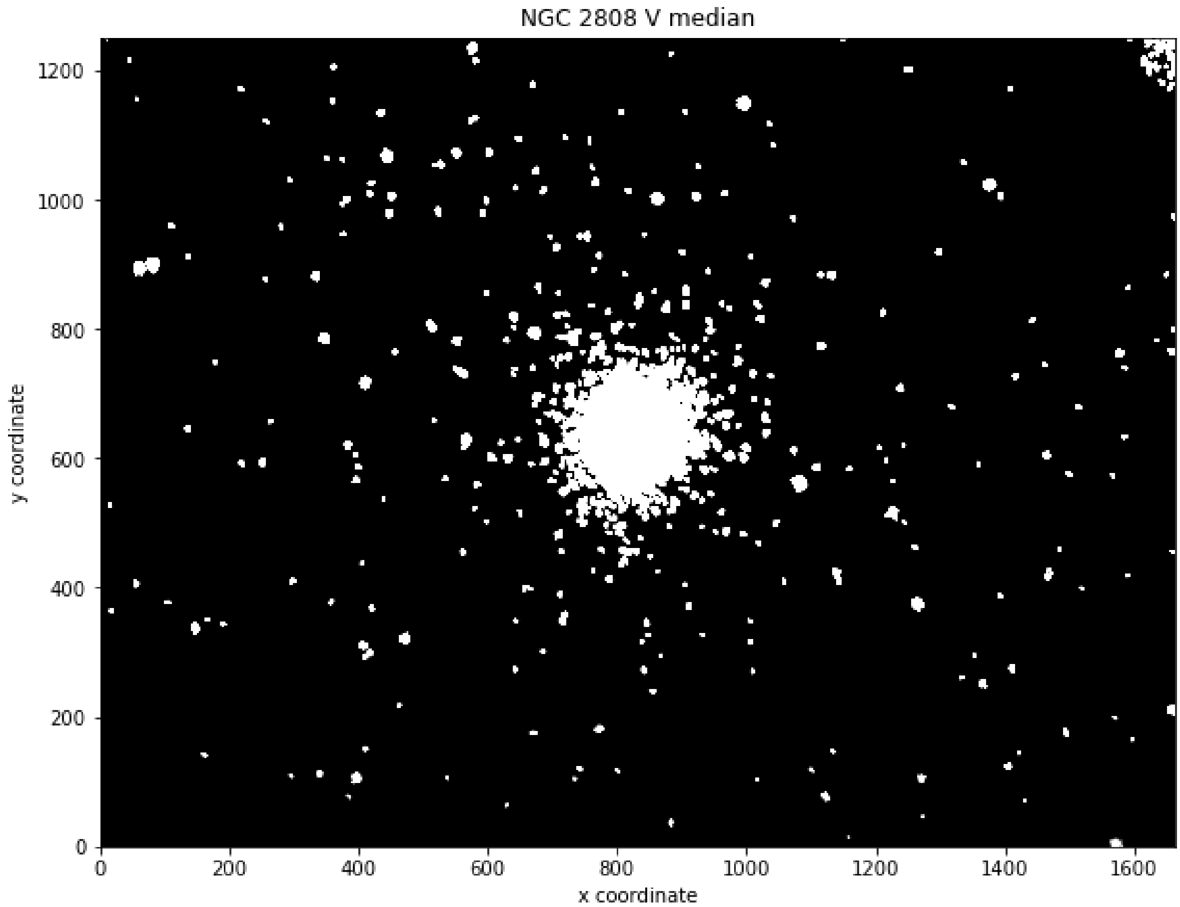
```
INFO: using the unit adu passed to the FITS reader instead of the unit
adu2 in the FITS file. [astropy.nddata.ccddata]
Image stats (mean, median and standard deviation): -0.44846767 -2.0487
838 114.902664
```

In [12]:

```
plt.figure(figsize=(10,10))
plt.imshow(segimage.data, cmap='gray', vmin=0, vmax=1, origin='lower')
plt.title('NGC 2808 V median')
plt.xlabel('x coordinate')
plt.ylabel('y coordinate')
```

Out[12]:

```
Text(0, 0.5, 'y coordinate')
```



The source table gives the x and y pixel coordinates of each star as well as the aperture sum in ADU

In [13]:

```
# Creating a source table
source_table=source_properties(scim.data, segimage, error=None, mask=None, backgrour
print('Source table length:\n', len(source_table))
print('First entry centroid y and x values :') # The dreaded y-x
print(source_table[0].centroid[0].value, source_table[0].centroid[1].value)
```

```
Source table length:
 499
First entry centroid y and x values :
4.905264395724232 1570.1284325241668
```

In [14]:

```
# xy format for centroid positions
positions=[]
for obj in source_table:
    positions.append((obj.centroid[1].value, obj.centroid[0].value))
print(positions[0:10]) # Print example values from the positions list.
apertures = CircularAperture (positions, r=10) #code taken from Lab 3
phot_table_v = aperture_photometry(scim, apertures)
print(phot_table_v)
```

```
[(1570.1284325241668, 4.905264395724232), (1157.246646154549, 14.32187
906690173), (882.3934690141442, 37.028307580722846), (1272.59175439310
37, 46.43042978397951), (628.4191326180189, 64.24233285503477), (1430.
0818798461169, 70.56217347028978), (1121.5945988035332, 77.49679956991
102), (383.9229346192441, 77.23882157911514), (391.81986949428307, 99.
72020358808204), (396.2950152473696, 106.27068736117312)]
 id      xcenter            ycenter          aperture_sum
           pix                pix                adu
--- ------------------ ------------------ ------------------
  1 1570.1284325241668  4.905264395724232  259741.1646383857
  2  1157.246646154549  14.32187906690173  9022.252264400273
  3  882.3934690141442 37.028307580722846 24639.261444878222
  4 1272.5917543931037  46.43042978397951  9292.172108490739
  5  628.4191326180189  64.24233285503477   1695.95701702502
  6 1430.0818798461169  70.56217347028978 12324.733012375618
  7 1121.5945988035332  77.49679956991102  65222.56588430477
  8  383.9229346192441  77.23882157911514  7907.072543245778
  9 391.81986949428307  99.72020358808204 113991.87387931165
 10  396.2950152473696 106.27068736117312  148952.6781575808
...                ...                ...                ...
490 1621.8036557958787 1239.8085320971293  41385.83652364867
491  1641.047850998058 1238.8186443481552  50544.23653838369
492 1654.5824585374073 1240.1571662076772  61367.84168003987
493  1633.151573499017 1241.7780851156915  36850.27943065377
494 1650.7463911120353 1246.8994305717617 53415.454813116616
495  1620.075667437412 1247.2329643355824 36704.487566849624
496 1149.4082775396103 1247.0601457030766 3987.9238766122453
497  1592.895571363799 1247.5330770863534 15024.102207913656
498  9.454108936997049 1248.5908651831508  4889.719683200112
499 1635.7189423051657  1248.981693058348  27386.40037328402
Length = 499 rows
```

In [15]:

```
#### Aperture sum and positions of reference star at (1264,375) in V median image
for idx, obj in enumerate(phot_table_v):
        if obj['xcenter'].value>1200 and obj['xcenter'].value<1300 \
        and obj['ycenter'].value>300 and obj['ycenter'].value<400 :
            refidx=idx
            print(idx, obj['xcenter'].value, obj['ycenter'].value, obj['aperture_sum
```

```
59 1263.05608553992 375.4302838747978 277373.31402309495 adu
```

In [16]:

```
#### Aperture sum and positions fainter star at (1264,375) in V median image
for idx, obj in enumerate(phot_table_v):
        if obj['xcenter'].value>1200 and obj['xcenter'].value<1230 \
        and obj['ycenter'].value>300 and obj['ycenter'].value<400 :
            refidx=idx
            print(idx, obj['xcenter'].value, obj['ycenter'].value, obj['aperture_sum
```

In [17]:

```
# finding the magnitude of the fainter star using the flux and known magnitude of re
f1= 118633.9433912482
f2= phot_table_v[refidx]['aperture_sum'].value # 260577.88624417983
m2= Vref   # roughly 7.02390159404841
m1= m2-2.5*np.log10(f1/f2)
print(m2,f2)
print(m1, f1)
```

```
11.504467751874088 277373.31402309495
12.426607024024381 118633.9433912482
```

In [18]:

```
V_mag=[]
print(m2,f2)
for obj in phot_table_v:
    V_mag.append(Vref-2.5*np.log10(obj['aperture_sum'].value/f2))
print(V_mag)
```

```
11.504467751874088 277373.31402309495
[11.575777481718085, 15.223842025709857, 14.133060225917792, 15.191836
327609044, 17.03859233764865, 14.885185640396084, 13.076134740137954,
15.367090132390778, 12.469944708096115, 12.17950864998501, 13.25976144
0391573, 16.219916746290572, 14.93694877675702, 14.73552331025991, na
n, 13.869048029975446, 14.422315096345791, 15.184052615981491, 14.3875
3002921245, 14.095196196103547, nan, 15.031873956662132, 15.5207943628
44814, 15.072059771022204, 15.96097000370493, 14.071584833961719, 15.1
3121190281211, 13.676089968344385, 17.145805500975875, 13.089583008345
423, 17.805522666894138, 14.045088637851855, 13.993350521567725, 14.60
3753099300045, 14.490086711405747, 15.395893431631231, 13.718243303935
303, 15.372780724323672, 14.366011927342662, 13.672963425896567, 14.87
2983642627126, 17.055076669068413, 13.64921410568924 3, 14.455129001959
95, 13.681438304306763, 12.09875807071633, 14.507794983095714, 15.3220
83480783322, 17.269562363249726, 14.981992821543933, 12.88763392969735
5, 14.792051838990737, 14.110588250427963, 13.092707582086343, 15.4446
15502292544, 14.666586473962258, 15.809024196176097, 16.1505966657572
3, 15.041036802107854, 11.504467751874088, 14.087673006909707, 15.1818
52794847698, 15.444577837618228, 14.59015773302911, 14.42388358056518,
14.204895777849089, 14.60906846989011, 14.608911461256387, 15.17442392
2610723, 13.929884246585868, 14.810132074400606, 16.162550723580882, 1
4.098276617506535, 13.830410570889034, 13.56777015603289, 15.003331695
263773, 14.337709430935725, 14.251567492859829, 14.776999015343746, 1
3.830033888215453, 16.294391982351762, 13.715063435121479, 14.52402490
109042, 12.95254982669998, 14.875974369943542, 13.783204560817586, 14.
558442399377451, 15.328188125015101, 14.013590886375185, 15.5465900722
52354, 14.5964452044336, 15.026239856025306, 14.558068485878165, 14.33
2222393044361, 13.616687291344387, 13.332886236591493, 14.035349333405
065, 14.507376702276572, 14.258551673444977, 15.50413957182716, 13.601
922284754835, 14.206397994188455, 14.96533556700924, 13.56780800253520
3, 14.393067285654842, 13.907071764875674, 13.686457730456492, 15.0237
07971660686, 16.067300032823468, 13.534672524030796, 13.38019627853452
3, 13.827904740504838, 12.32714936199362, 13.331212552384757, 14.95107
3571164295, 13.591378657716058, 13.15669468451166, 14.108530862260412,
13.913436619355993, 12.771585650792384, 13.987809484625553, 13.4992652
7269427, 13.447708869231224, 13.116324592332557, 13.146334835104652, 1
4.836993311133721, 13.355603340383299, 13.248793475042786, 13.18213871
4602265, 13.780469662371752, 17.196075277458622, 13.159887085298438, 1
3.26897546134136, 12.892439376055862, 13.675279015891103, 13.406816399
339016, 13.499233175148204, 13.330125339127079, 9.606611936952042, 13.
41678365314751, 13.4203556104243, 13.92931791714868, 13.51754024738051
5, 18.307423189133885, 13.349794977582, 13.405378354332637, 13.6743355
35737633, 12.788704092882973, 13.901988708631542, 13.700609258502046,
13.15479905708724, 13.507730375859893, 13.184619477779568, 13.98779288
2759928, 12.684456271839707, 13.443609904364097, 12.953542389188565, 1
3.4827850388147, 13.584721297723277, 12.867793945075457, 10.2952442110
77883, 12.843142295275602, 13.806580751638114, 13.175421151649251, 12.
732071835812956, 13.005679739576673, 13.910643920786816, 13.2166403400
78731, 14.615495450113537, 13.945085518028366, 13.930663634067564, 15.
631243312860523, 13.205010037343435, 13.787169860182848, 14.6525402107
5761, nan, 13.538317010431324, 13.00570606062373, 13.035404332026577,
```

```
16.74629822598466, 13.950874960536815, 13.115185114928204, 13.43563969
8348911, 14.086016857253718, 13.37729230848555, 13.623968248890641, 1
3.256997229035019, 13.994253279033924, 15.392735298537639, 14.59787242
5189598, 13.774529535699546, 13.218701127809986, 14.35021993746571, 1
4.635480536269874, 13.157479014007881, 13.697649317197257, 14.05634048
0932162, 13.570324639887415, 13.394655959731502, 13.335614883563343, 1
3.678687153446976, 13.190244903650013, 16.047591276971055, 13.22403198
8213657, 13.40235312124329, 13.653127713875168, 13.092853828253745, 1
3.45219160573594, 13.374370144099695, 13.801466976836098, 15.208664101
910461, 13.06365774263681, 13.375874590061215, 13.920418802888312, 13.
27450409739431, 12.962423807782285, 13.277336950519839, 13.56857075085
722, 12.813677202611027, 14.696212380013225, 13.769840431826319, 12.87
0023879381938, 12.693628512379735, 11.74008834881157, 12.9244234350311
87, 13.027079419719096, 13.049104756918272, 15.676065084304096, 14.346
179593803775, 13.221166378707467, 14.61765984642129, 13.09523278614063
8, 15.056248945951285, 13.143386903229601, 13.11840021188099, 13.95171
3650916444, 13.638972417739218, 13.922027088086944, 13.50980975099280
1, 13.359131982871203, 14.138619944607683, 13.327649724601576, 13.5289
40708519274, 13.410866604108227, 14.751410073487946, 13.13034734108984
8, 14.210753695951567, 15.333507550585423, 14.023963234989132, 16.0942
29651023475, 12.925935470912345, 13.271707033455625, 13.01731726572535
8, 13.494922513589739, 13.363698545783834, 13.958302443489911, 13.7507
31198990927, 16.114060285073744, 15.09326261210206, 13.57467973502963
2, 13.773438914681016, 14.331490534414566, 13.829298576390553, 13.0102
6238985473, 12.825297349548348, 13.823004951603828, 13.04545801865946
2, 14.30189959619701, 13.419819284319686, 13.355424272727381, 13.96803
6564718313, 13.012562657331866, 13.744751019843811, 13.75313642836214
4, 11.678950540063155, 13.958488180307308, 13.115616482707168, 13.2957
18436812633, 12.65751051134244, 13.622528218004883, 13.22009967884786,
13.439977987657027, 13.616384852429976, 13.36920173503714, 13.18740867
8864463, 13.140226232158744, 14.67981797274957, 13.531653638851337, 1
4.273890798501728, 12.975877970976196, 12.97639135276114, 14.333084499
334495, 14.435555897075321, 13.559719230080216, 14.431319654789744, 1
3.424030312910805, 13.814399888541661, 13.255318100177808, 13.95474037
7045814, 13.515486926971848, 12.948181048991284, 14.683482508031277, 1
4.130171776501506, 13.26062450705211, 16.128910263748033, 13.118002063
671838, 13.311670490368419, 14.272709558615524, 13.114268642690055, 1
3.82709536919346, 15.231204687790136, 13.469759045978343, nan, 13.3699
65557927879, 13.494818423701554, 13.694651687918832, 13.92845612352003
2, 13.80302535395408, 14.06214044995636, 13.956458877936461, 14.046187
53489914, 13.675179641399804, 13.353146761354411, 13.61354647452353, 1
3.70573604248451, 14.678869747814037, 12.898395128464903, 13.430266621
994104, 14.189708433834042, 13.621192482599028, 14.841432307002684, 1
4.016101201824174, 13.689581133502193, 13.54981876422965, 13.789079369
311702, 13.787231614794598, 13.747212767070375, 13.361015734037272, 1
3.148322521923644, 13.588745644905085, 13.584863582124985, 14.04266252
428432, 14.331666257804994, 14.57571910159223, 12.462871219344626, 12.
998960107575465, 11.747647649274862, 14.258699893565675, 14.1449420561
95678, 14.327459137150756, 15.83088041344049, 13.271534761596538, 13.4
8039735069842, 11.535521218201005, 14.686764704253314, 13.480216570258
13, 14.236196300750937, 13.752306619647277, 13.507715248050175, 13.263
542982495936, 15.442644090568002, 12.934084863351048, 14.7039723619254
87, 14.94249955519042, 15.107828281309562, 13.482735962793498, 14.1836
56999573028, 14.951674442432118, 13.366345862550641, 13.93549551240692
4, 14.133050464910278, 13.921356173849855, 14.156023600285618, 14.1867
6707966863, 13.97001869505102, 13.701216902176004, 14.943384942204553,
13.650050166212846, 13.172110910307586, 14.11681341779857, 14.56484172
6016382, 13.936141978490134, 14.27405997010504, 14.199730306658985, 1
4.78302378736382, 14.838579349577484, 13.038721970624696, 15.099524762
```

```
96303, 14.325202016066596, 14.848718418109428, 13.472609184719662, 17.
98058566794768, 13.025144871229132, 13.614211947722563, 14.60651085477
9147, 14.845321607235038, 14.644425599855113, 10.582060253309614, 14.7
49554538284432, 9.998536830029277, 14.689375927529472, 14.893542921399
54, 15.766688375211807, 14.634905134207413, 14.314242482419385, 14.212
57433449226, 13.685463178792169, 13.222832139712015, 15.27317134145614
1, 13.617909462532452, 14.941904316248383, 15.413359746486424, 16.6181
79529274165, 14.606288845380117, 15.026787738310288, 15.01171391666458
8, 13.732109079006472, 14.500202691792008, 14.24782097502569, 15.04501
0946981714, 11.359229283616258, 15.062784489947031, 13.92167947162827,
12.980637086139605, 13.367763235650012, 14.28561076073815, 15.10656339
684364, 14.234646307551953, 15.210962586058855, 14.861798318645942, 1
1.869047980524602, 15.204372900384456, 14.610673635942074, 14.23131372
3600197, 15.434638223384816, 14.566155487120238, 13.938296566750836, 1
5.718416579187927, 14.705910816559022, 14.52553672226878, 14.940602898
628807, nan, 11.215617304514117, nan, nan, 13.077905450469071, 13.4889
74052563314, nan, 15.015419823408967, 15.641164210829274, 16.059154451
85194, 15.753539629948293, 13.691516509144979, 18.249504280060922, 13.
340478584959714, 15.199140149271571, 15.262735211094462, 10.9666666760
57102, 15.056919067844685, nan, 15.212138345823394, 14.90554646337252
4, 13.822987397368149, 14.993741964229198, 13.88550122452147, 14.16133
083297133, 13.632511176849064, 13.897283202326575, 14.198514076339853,
13.471633176259246, 14.94228060754756, 13.343410794597322, 13.62586408
0239381, 13.35686810394681, 13.480876741871933, 13.6977733985184, 14.3
42980884490688, 15.15382153527403, 13.438040876119262, 13.623561867265
284, 13.479885893841988, 15.826639135026522, 12.405551693169102, 13.37
1241277605343, 13.46983707781946, 13.49209213912148, 13.34679232557575
5, 13.297292256390229, 13.755723453146508, 13.570000095662731, 13.3529
50338016427, 13.14227731718122, 13.696027476982005, 13.29296211397877
2, 13.700331527382671, 16.110262291243018, 14.670158116973303, 15.8889
19533749643, 14.018292057989669]
```

I band cataloguing

Repeating steps for the other filters (minus source table)

In [19]:

```
scim=CCDData.read("NGC_2808_I_median.fits", unit="adu") # reading the data from the
med=np.median(scim.data) # defining the median of the data
scim.data=scim.data-med #subtracting median
mean, median, std = astropy.stats.sigma_clipped_stats(scim.data, sigma=3.0, maxiters
print('Image stats (mean, median and standard deviation):', mean,median,std) # calcu
```

```
INFO: using the unit adu passed to the FITS reader instead of the unit
adu2 in the FITS file. [astropy.nddata.ccddata]
Image stats (mean, median and standard deviation): -0.037260514 -3.058
9023 164.70932
```

In [20]:

```
# xy format for centroid positions
positions=[]
for obj in source_table:
    positions.append((obj.centroid[1].value, obj.centroid[0].value))
print(positions[0:10]) # Print example values from the positions list.
apertures = CircularAperture (positions, r=10) #code taken from Lab 3
phot_table_i = aperture_photometry(scim, apertures)
print(phot_table_i)

# Aperture sum and positions of reference star at (1265,373) in V median image
for idx, obj in enumerate(phot_table_i):
        if obj['xcenter'].value>1200 and obj['xcenter'].value<1300 \
        and obj['ycenter'].value>300 and obj['ycenter'].value<400:
                print(idx, obj['xcenter'].value, obj['ycenter'].value, obj['aperture
```

```
[(1570.1284325241668, 4.905264395724232), (1157.246646154549, 14.32187
906690173), (882.3934690141442, 37.028307580722846), (1272.59175439310
37, 46.43042978397951), (628.4191326180189, 64.24233285503477), (1430.
0818798461169, 70.56217347028978), (1121.5945988035332, 77.49679956991
102), (383.9229346192441, 77.23882157911514), (391.81986949428307, 99.
72020358808204), (396.2950152473696, 106.27068736117312)]
 id      xcenter             ycenter          aperture_sum
           pix                 pix                 adu
--- ------------------ ------------------ ------------------
  1 1570.1284325241668  4.905264395724232  212682.4140365344
  2  1157.246646154549  14.32187906690173 1247.8469382711837
  3  882.3934690141442 37.028307580722846  20216.16805381552
  4 1272.5917543931037  46.43042978397951  4070.705006016732
  5  628.4191326180189  64.24233285503477  4309.214601035967
  6 1430.0818798461169  70.56217347028978  15980.71146968435
  7 1121.5945988035332  77.49679956991102   70176.8129309905
  8  383.9229346192441  77.23882157911514   14683.5217922284
  9 391.81986949428307  99.72020358808204 151637.89062945382
 10  396.2950152473696 106.27068736117312 196592.21075901028
...                ...                ...                ...
490 1621.8036557958787 1239.8085320971293 7602.2689945434595
491  1641.047850998058 1238.8186443481552 10766.263848229308
492 1654.5824585374073 1240.1571662076772 15069.385816622542
493  1633.151573499017 1241.7780851156915  2851.325211403459
494 1650.7463911120353 1246.8994305717617 15623.545142540432
495  1620.075667437412 1247.2329643355824 10086.040377996067
496 1149.4082775396103 1247.0601457030766  7577.507383774564
497  1592.895571363799 1247.5330770863534 152.38902907742022
498  9.454108936997049 1248.5908651831508 5361.5631869195995
499 1635.7189423051657  1248.981693058348  2646.359526630646
Length = 499 rows
59 1263.05608553992 375.4302838747978 358012.58868146077 adu
```

In [21]:

```
#### Aperture sum and positions fainter star at (1264,375) in V median image
for idx, obj in enumerate(phot_table_v):
        if obj['xcenter'].value>1220 and obj['xcenter'].value<1230 \
        and obj['ycenter'].value>500 and obj['ycenter'].value<520 :
            refidx=idx
            print(idx, obj['xcenter'].value, obj['ycenter'].value, obj['aperture_sum
```

```
112 1225.2625131754075 516.2551530299908 130014.57222438144 adu
```

In [22]:

```
# finding the magnitude of the fainter star using the flux and known magnitude of re
f1= 127139.71057166517
f2= phot_table_i[refidx]['aperture_sum'].value  # 348375.16274801537
m2= Iref # 5.987310474624634
m1= m2-2.5*np.log10(f1/f2)
print(m2, f2)
print(m1, f1)
```

```
10.310198208827972 212155.25998842556
10.866129672089041 127139.71057166517
```

In [23]:

```
I_mag=[]
for i in range(len(phot_table_i)):
    print(phot_table_i['aperture_sum'][i].value)
    I_mag.append(m2-2.5*np.log10(phot_table_i[i]['aperture_sum'].value/f2))
print(I_mag)
print(len(I_mag))
print(I_mag[0])
```

```
212682.4140365344
1247.8469382711837
20216.16805381552
4070.705006016732
4309.214601035967
15980.71146968435
70176.8129309905
14683.5217922284
151637.89062945382
196592.21075901028
40713.75701165882
6009.93607938378
9661.945814777278
18995.402721987608
-10397.307166896564
98509.59811782805
22695.69454139105
4299.369332204415
17526.77465452383
```

R band cataloguing

In [24]:

```
# creating a catalogue for the stars in NGC 2808
# based on lab 3 which used segmentation ASK MICHAEL AND JAMIE
scim=CCDData.read("NGC_2808_R_median.fits", unit="adu") # reading the data from the
med=np.median(scim.data) # defining the median of the data
scim.data=scim.data-med #subtracting median
mean, median, std = astropy.stats.sigma_clipped_stats(scim.data, sigma=3.0, maxiters
print('Image stats (mean, median and standard deviation):', mean,median,std) # calcu
```

```
INFO: using the unit adu passed to the FITS reader instead of the unit
adu2 in the FITS file. [astropy.nddata.ccddata]
Image stats (mean, median and standard deviation): -1.3458027 -2.97073
32 132.12003
```

In [25]:

```
# xy format for centroid positions
positions=[]
for obj in source_table:
    positions.append((obj.centroid[1].value, obj.centroid[0].value))
print(positions[0:10]) # Print example values from the positions list.
apertures = CircularAperture (positions, r=10) #code taken from Lab 3
phot_table_r = aperture_photometry(scim, apertures)
print(phot_table_r)

# Aperture sum and positions of reference star at (1265,373) in V median image
for idx, obj in enumerate(phot_table_r):
        if obj['xcenter'].value>1200 and obj['xcenter'].value<1300 \
        and obj['ycenter'].value>300 and obj['ycenter'].value<400:
                print(idx, obj['xcenter'].value, obj['ycenter'].value, obj['aperture
```

```
[(1570.1284325241668, 4.905264395724232), (1157.246646154549, 14.32187
906690173), (882.3934690141442, 37.028307580722846), (1272.59175439310
37, 46.43042978397951), (628.4191326180189, 64.24233285503477), (1430.
0818798461169, 70.56217347028978), (1121.5945988035332, 77.49679956991
102), (383.9229346192441, 77.23882157911514), (391.81986949428307, 99.
72020358808204), (396.2950152473696, 106.27068736117312)]
 id      xcenter            ycenter          aperture_sum
           pix                pix                 adu
--- ------------------ ------------------ ------------------
  1 1570.1284325241668  4.905264395724232 297272.14518994547
  2  1157.246646154549  14.32187906690173 2592.5698071129245
  3  882.3934690141442 37.028307580722846  29652.64552718414
  4 1272.5917543931037  46.43042978397951 3677.6119910763064
  5  628.4191326180189  64.24233285503477  2652.490122939511
  6 1430.0818798461169  70.56217347028978 11642.583490407946
  7 1121.5945988035332  77.49679956991102  76857.35231900291
  8  383.9229346192441  77.23882157911514 16366.349399223038
  9 391.81986949428307  99.72020358808204 130211.07432452912
 10  396.2950152473696 106.27068736117312 224852.49965123262
...                ...                ...                ...
490 1621.8036557958787 1239.8085320971293 26381.544367960912
491  1641.047850998058 1238.8186443481552  34665.89016872297
492 1654.5824585374073 1240.1571662076772  43217.94298645227
493  1633.151573499017 1241.7780851156915 22280.661821570502
494 1650.7463911120353 1246.8994305717617  37237.00603173868
495  1620.075667437412 1247.2329643355824 26134.101786881507
496 1149.4082775396103 1247.0601457030766  9043.166736959727
497  1592.895571363799 1247.5330770863534 3969.2104738795297
498  9.454108936997049 1248.5908651831508  4155.283372506187
499 1635.7189423051657  1248.981693058348  18632.64204986135
Length = 499 rows
59 1263.05608553992 375.4302838747978 389658.98559589294 adu
```

In [26]:

```
#### Aperture sum and positions of fainter star at (1264,375) in V median image
for idx, obj in enumerate(phot_table_v):
        if obj['xcenter'].value>1220 and obj['xcenter'].value<1230 \
        and obj['ycenter'].value>500 and obj['ycenter'].value<520 :
            refidx=idx
            print(idx, obj['xcenter'].value, obj['ycenter'].value, obj['aperture_sum
```

```
112 1225.2625131754075 516.2551530299908 130014.57222438144 adu
```

In [27]:

```
# finding the magnitude of the fainter star using the flux and known magnitude of re
f1= 130014.57222438144
f2= phot_table_r[refidx]['aperture_sum'].value  # 348375.16274801537
m2= Rref # 5.987310474624634
m1= m2-2.5*np.log10(f1/f2)
print(m2, f2)
print(m1, f1)
```

```
10.896741841092728 211059.0565592117
11.422771743839437 130014.57222438144
```

In [28]:

```
R_mag=[]
for i in range(len(phot_table_r)):
    print(phot_table_r['aperture_sum'][i].value)
    R_mag.append(m2-2.5*np.log10(phot_table_r[i]['aperture_sum'].value/f2))
print(R_mag)
print(len(R_mag))
print(R_mag[0])
```

```
297272.14518994547
2592.5698071129245
29652.64552718414
3677.6119910763064
2652.490122939511
11642.583490407946
76857.35231900291
16366.349399223038
130211.07432452912
224852.49965123262
57689.05619896336
3787.1217362341677
14053.836488472985
19313.25347659336
-9607.680146960898
78046.55275706953
22105.695905287866
6495.215810851161
25039.85918572895
```

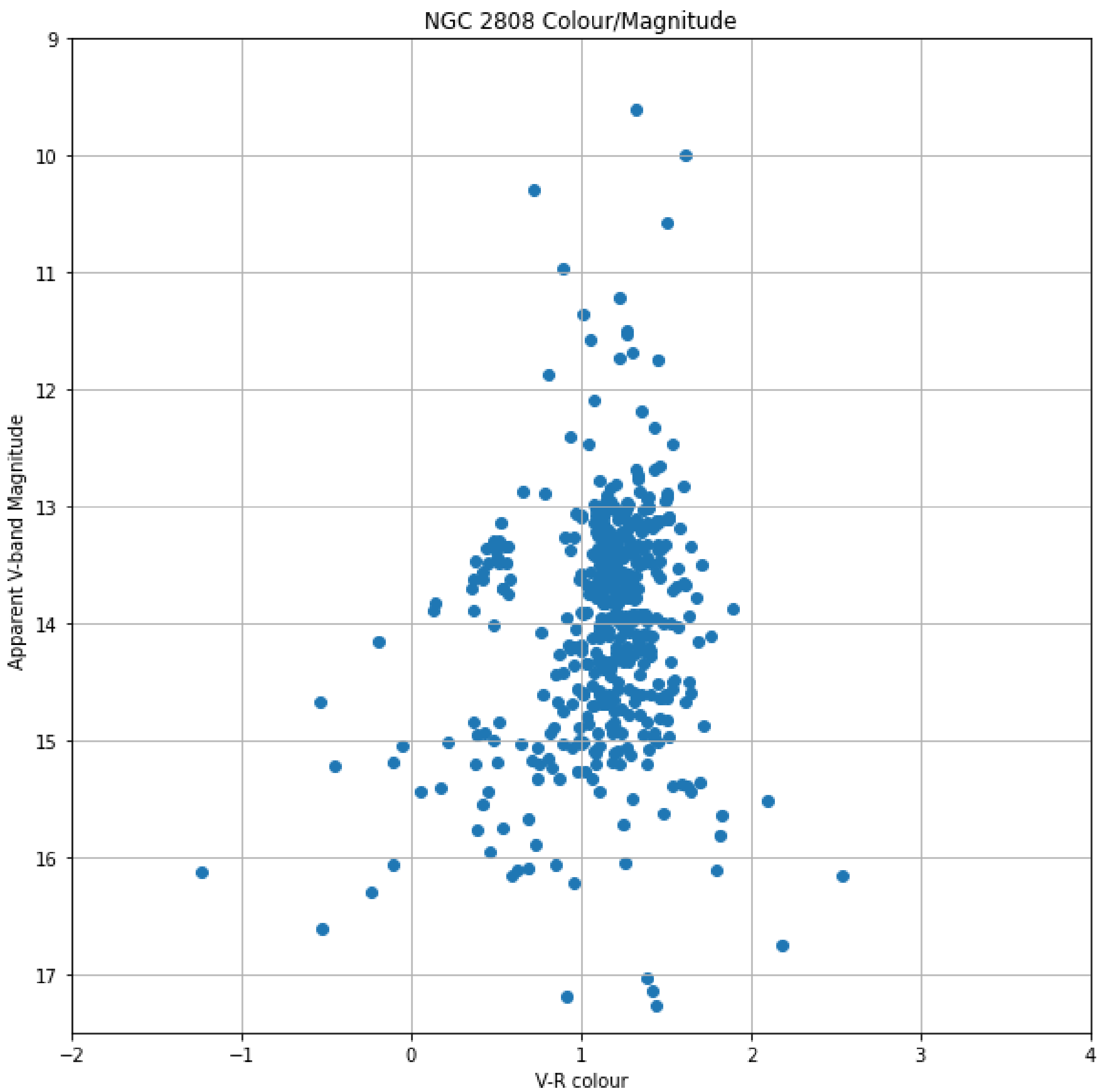
In [29]:

```
print(len(V_mag))
print(len(I_mag))

avg=np.mean(V_mag)
print(avg)
```

```
499
499
nan
```

In [30]:

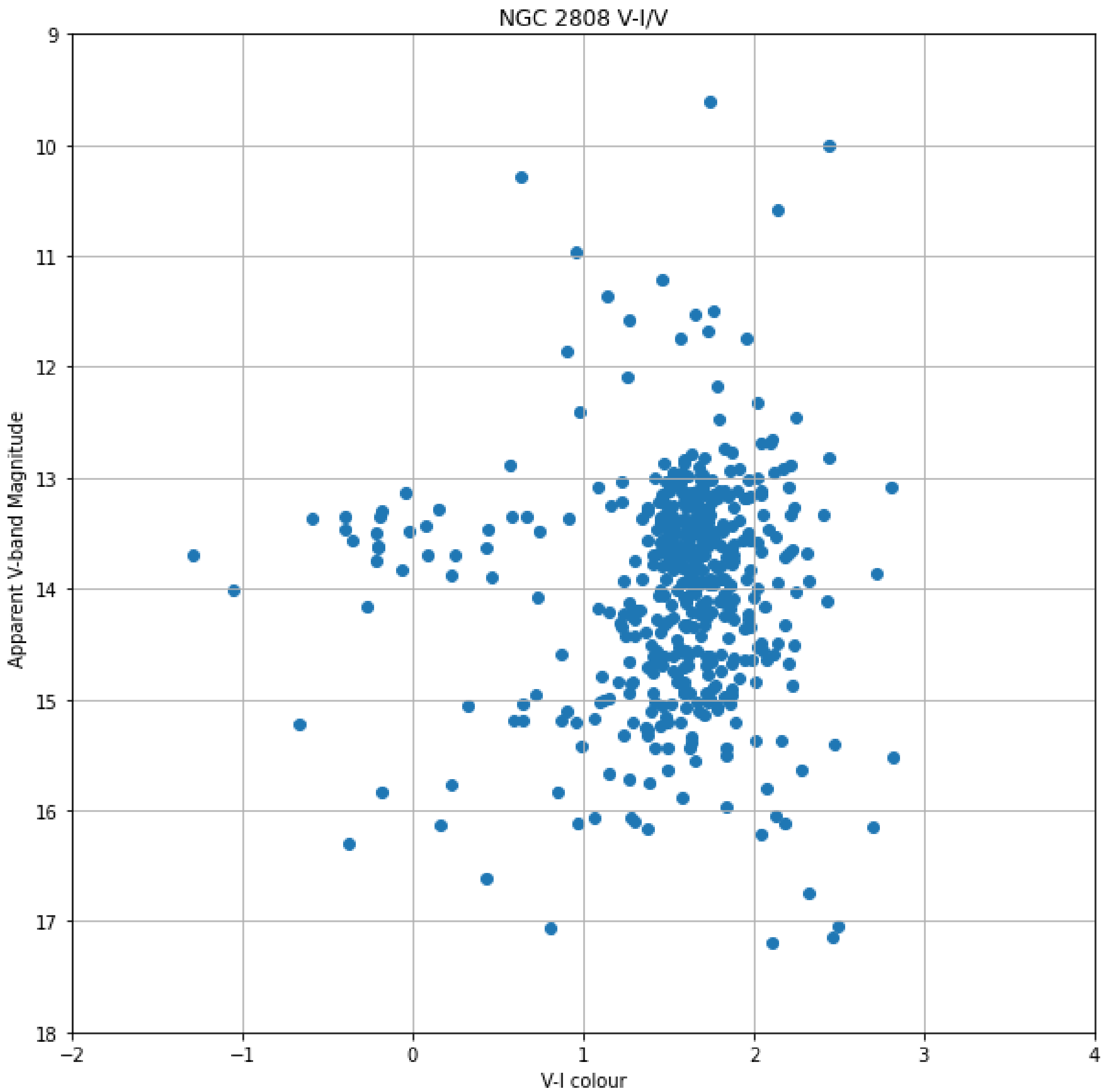
```
# need a V-R against V
# Define plot size
plt.rcParams['figure.figsize'] = [10, 10]
plt.axis([-2.0, 4, 17.5, 9])
# Axis labels and grid
plt.xlabel('V-R colour')
plt.ylabel('Apparent V-band Magnitude ')
plt.title('NGC 2808 Colour/Magnitude')
plt.grid(True)
px=[]
py=[]
for idx, mag in enumerate(V_mag):
    py.append(mag)
    px.append(V_mag[idx]-R_mag[idx])
plt.scatter(px,py)

plt.show()

#plt.savefig("v-rCMD.png")
```




```
<Figure size 720x720 with 0 Axes>
```

In [32]:

```
# need a V-I against V
# Define plot size
plt.rcParams['figure.figsize'] = [10, 10]
plt.axis([-2, 4, 18, 9])
# Axis labels and grid
plt.xlabel('V-I colour')
plt.ylabel('Apparent V-band Magnitude ')
plt.title('NGC 2808 V-I/V')
plt.grid(True)
px=[]
py=[]
for idx, mag in enumerate(V_mag):
    py.append(V_mag[idx])
    px.append(V_mag[idx]-I_mag[idx])
plt.scatter(px,py)

plt.show()
#plt.savefig("viCMD.png")
```




```
<Figure size 720x720 with 0 Axes>
```

In [388]:

```
# Solution

class isochroneclass:
    def __init__(self):
            self.filename = "NULL                            "      # Relevant filen
            self.age = -99.0                                         # Age
            self.m = [-99.0]*1000                                    # Star masses
            self.U = [-99.0]*1000                                    # Star U-band ma
            self.B = [-99.0]*1000                                    # Star B-band ma
            self.V = [-99.0]*1000                                    # Star V-band ma
            self.R = [-99.0]*1000                                    # Star R-band ma
            self.I = [-99.0]*1000                                    # Star I-band ma
```

In [468]:

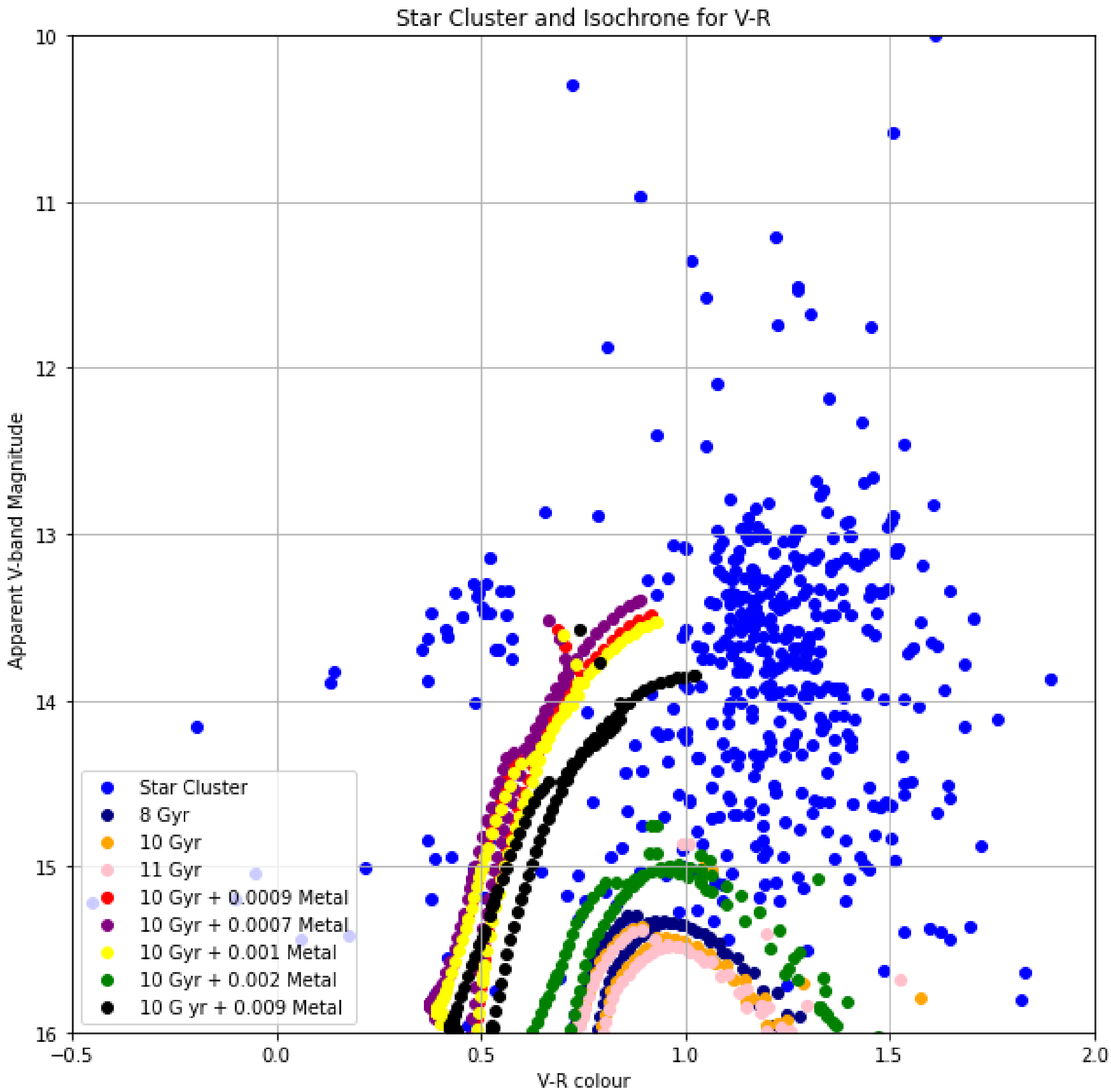
```
ion - 6 marks

e the input file name - this is the output with the defaults http://stev.oapd.inaf.i

['8Gyr.txt', '10Gyr.txt', '11Gyr.txt', '0.0009Metal.txt','0.0007Metal.txt', '10Gyr0.
 '10Gyr0.009Metal.txt', '0.002Metal.txt']
s=[10, 5, 9, 20]
ne=[]


me in fnames:

rone=isochroneclass()
rone.age=age

pen(fname,"r")
es=f.readlines()
=0
 x in lines:
  if x[0]=='#' and x[2]=='Z':
      print(x.split()[28])
 if x[0]!='#' and idx<1000:
     tchrone.m[idx]=float(x.split()[5])
     tchrone.U[idx]=float(x.split()[28])
     tchrone.B[idx]=float(x.split()[29])
     tchrone.V[idx]=float(x.split()[30])
     tchrone.R[idx]=float(x.split()[31])
     tchrone.I[idx]=float(x.split()[32])
     idx=idx+1
lose()
chrone.append(tchrone)
=age*2

Check the first line from the first file is good')
sochrone[0].m[0])
sochrone[0].U[0])
sochrone[0].B[0])
sochrone[0].V[0])
sochrone[0].R[0])
sochrone[0].I[0])
```

```
Check the first line from the first file is good
0.09
22.583
19.965
18.135
15.988
13.743
```

In [486]:

```
# Solution

# Define plot size
plt.rcParams['figure.figsize'] = [10, 10]
plt.axis([-0.5, 2, 16, 10])
# Axis labels and grid
plt.xlabel('V-R colour')
plt.ylabel('Apparent V-band Magnitude ')
plt.title('Star Cluster')
plt.grid(True)
px=[]
py=[]
for idx, mag in enumerate(V_mag):
    py.append(V_mag[idx])
    px.append(V_mag[idx]-R_mag[idx])
plt.scatter(px,py,label='Star Cluster',color='blue')

DM=15

# All of the following lines that are commented out are set up so that we could easi

VR1=np.array(isochrone[0].V)-np.array(isochrone[0].R)
V1=np.array(isochrone[0].V)+DM+VR1
plt.scatter(VR1, V1, label='8 Gyr',color='navy')

VR2=np.array(isochrone[1].V)-np.array(isochrone[1].R)
V2=np.array(isochrone[1].V)+DM+VR2
plt.scatter(VR2, V2, label='10 Gyr',color='orange')

VR3=np.array(isochrone[2].V)-np.array(isochrone[2].R)
V3=np.array(isochrone[2].V)+DM+VR3
plt.scatter(VR3, V3, label='11 Gyr',color='pink')

VR4=np.array(isochrone[3].V)-np.array(isochrone[3].R)
V4=np.array(isochrone[3].V)+DM+VR4
plt.scatter(VR4, V4, label='10 Gyr + 0.0009 Metal',color='red')

VR5=np.array(isochrone[4].V)-np.array(isochrone[4].R)
V5=np.array(isochrone[4].V)+DM+VR5
plt.scatter(VR5, V5, label='10 Gyr + 0.0007 Metal',color='purple')

VR6=np.array(isochrone[5].V)-np.array(isochrone[5].R)
V6=np.array(isochrone[5].V)+DM+VR6
plt.scatter(VR6, V6, label='10 Gyr + 0.001 Metal',color='yellow')

VR7=np.array(isochrone[6].V)-np.array(isochrone[6].R)
V7=np.array(isochrone[6].V)+DM+VR7
plt.scatter(VR7, V7, label='10 Gyr + 0.002 Metal',color='green')

VR8=np.array(isochrone[7].V)-np.array(isochrone[7].R)
V8=np.array(isochrone[7].V)+DM+VR8
plt.scatter(VR8, V8, label='10 G yr + 0.009 Metal',color='black')



# Scatter plot
```

```
# Axis labels and grid
plt.xlabel('B-V colour')
plt.ylabel('Apparent V-band Magnitude ')
plt.title('Star Cluster and Isochrone')
plt.grid(True)
plt.legend(loc='lower left')


# Output file, if wanted
#plt.savefig("V_R_All isochrones.png")

# Axis labels and grid
plt.xlabel('V-R colour')
plt.ylabel('Apparent V-band Magnitude ')
plt.title('Star Cluster and Isochrone for V-R')
plt.grid(True)
plt.legend(loc='lower left')

# Plot to screen
plt.show()


distance=10.0*pow(10,0.2*DM)
print(distance)

print('For comparison, literature values for the Star Cluster distance are about 86
```

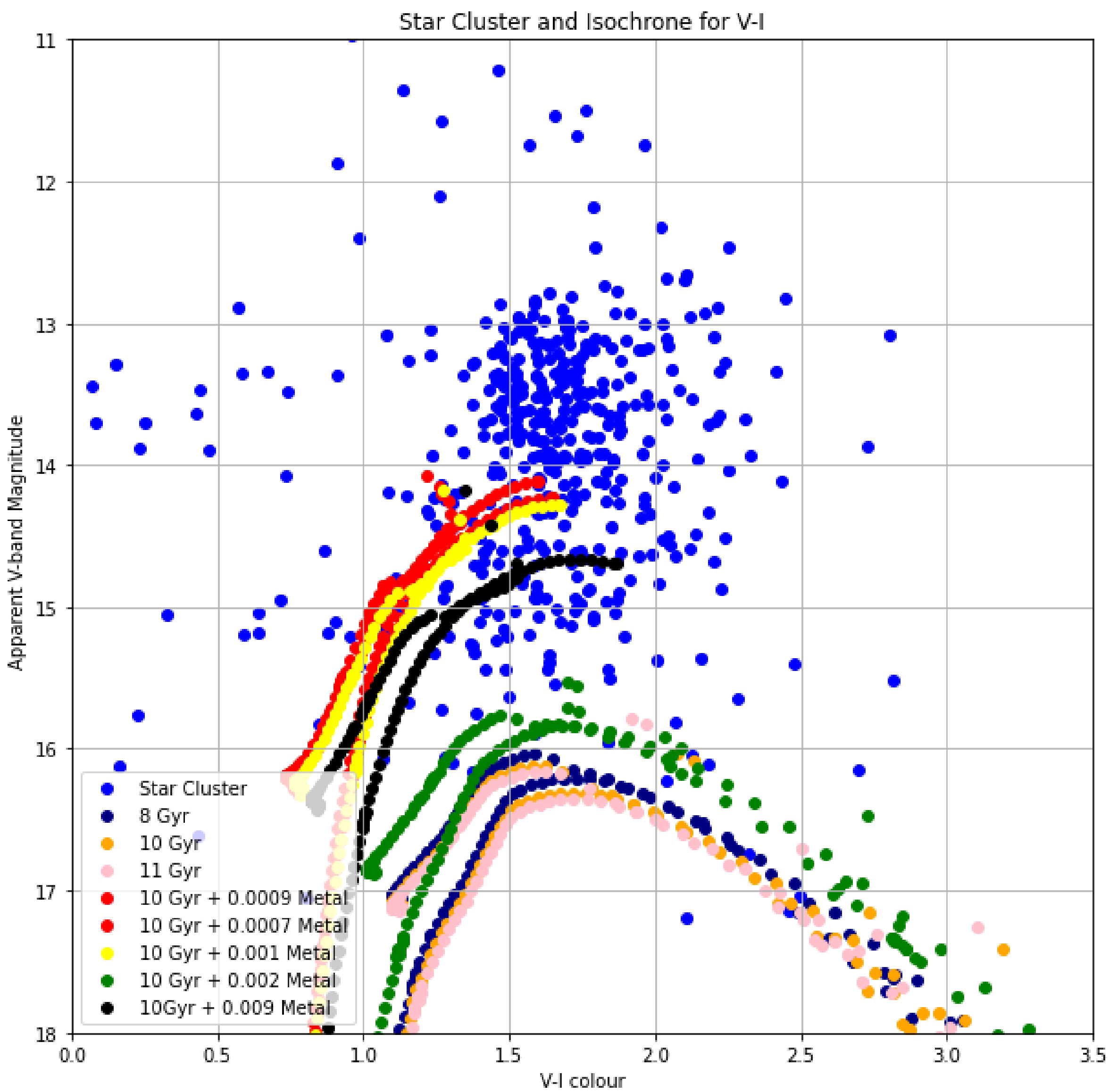



```
10000.0
For comparison, literature values for the Star Cluster distance are ab
out 86 pc
```

In [489]:

```
# Solution

# Define plot size
plt.rcParams['figure.figsize'] = [10, 10]
plt.axis([0,3.5, 18, 11])
# Axis labels and grid
plt.xlabel('V-R colour')
plt.ylabel('Apparent V-band Magnitude ')
plt.title('Star Cluster')
plt.grid(True)
px=[]
py=[]
for idx, mag in enumerate(V_mag):
    py.append(V_mag[idx])
    px.append(V_mag[idx]-I_mag[idx])
plt.scatter(px,py,label='Star Cluster',color='blue')

DM=15

VI1=np.array(isochrone[0].V)-np.array(isochrone[0].I)
V1=np.array(isochrone[0].V)+DM+VI1
plt.scatter(VI1, V1, label='8 Gyr',color='navy')

VI2=np.array(isochrone[1].V)-np.array(isochrone[1].I)
V2=np.array(isochrone[1].V)+DM+VI2
plt.scatter(VI2, V2, label='10 Gyr',color='orange')                          # Scatte


VI3=np.array(isochrone[2].V)-np.array(isochrone[2].I)
V3=np.array(isochrone[2].V)+DM+VI3
plt.scatter(VI3, V3, label='11 Gyr',color='pink')

VI4=np.array(isochrone[3].V)-np.array(isochrone[3].I)
V4=np.array(isochrone[3].V)+DM+VI4
plt.scatter(VI4, V4, label='10 Gyr + 0.0009 Metal',color='red')

VI5=np.array(isochrone[4].V)-np.array(isochrone[4].I)
V5=np.array(isochrone[4].V)+DM+VI5
plt.scatter(VI5, V5, label='10 Gyr + 0.0007 Metal',color='red')

VI6=np.array(isochrone[5].V)-np.array(isochrone[5].I)
V6=np.array(isochrone[5].V)+DM+VI6
plt.scatter(VI6, V6, label='10 Gyr + 0.001 Metal',color='yellow')

VI7=np.array(isochrone[6].V)-np.array(isochrone[6].I)
V7=np.array(isochrone[6].V)+DM+VI7
plt.scatter(VI7, V7, label='10 Gyr + 0.002 Metal',color='green')

VI8=np.array(isochrone[7].V)-np.array(isochrone[7].I)
V8=np.array(isochrone[7].V)+DM+VI8
plt.scatter(VI8, V8, label='10Gyr + 0.009 Metal',color='black')



# Axis labels and grid
plt.xlabel('V-I colour')
plt.ylabel('Apparent V-band Magnitude ')
```

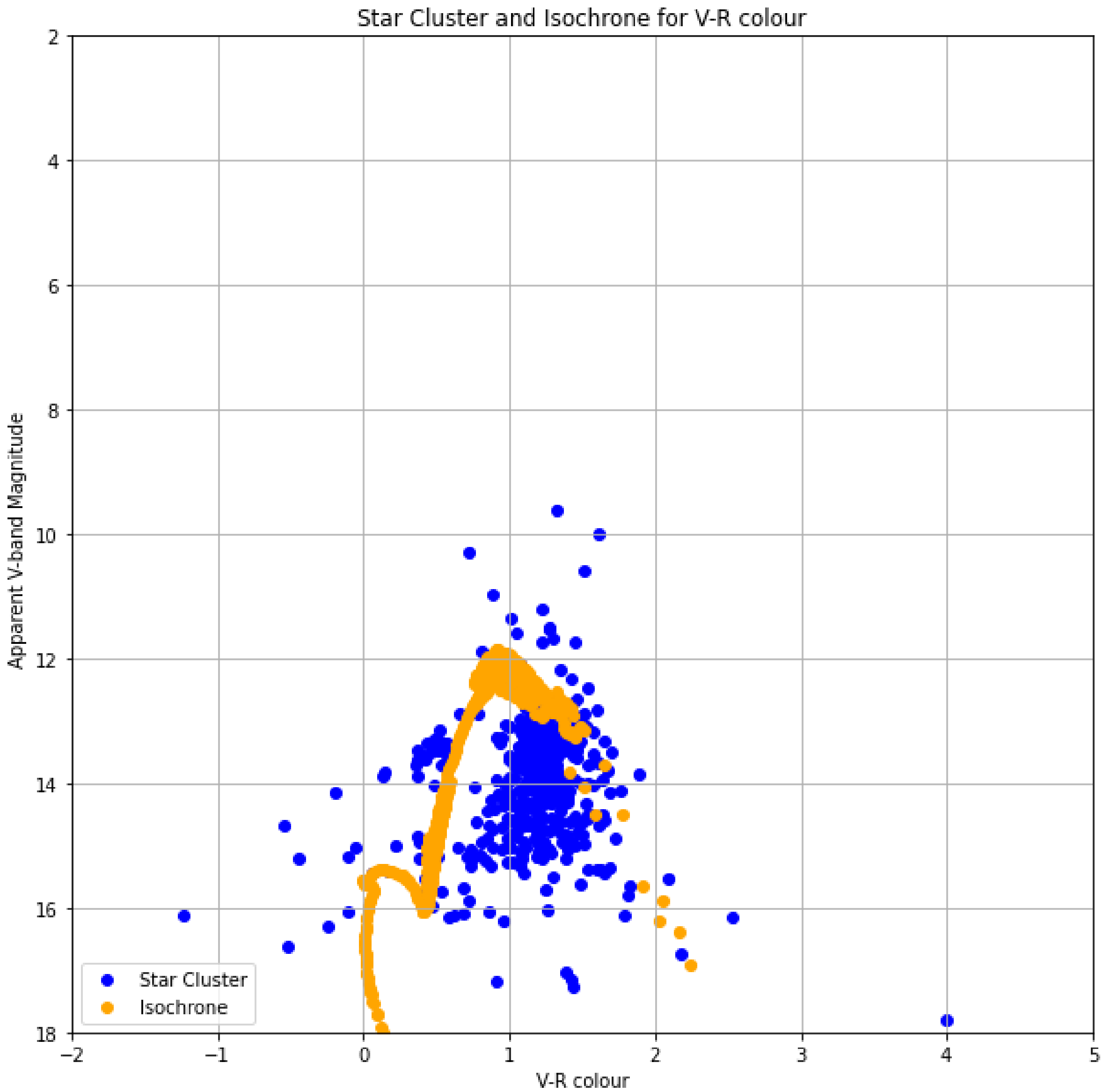
```
plt.title('Star Cluster and Isochrone')
plt.grid(True)
plt.legend(loc='lower left')


# Output file, if wanted
plt.savefig("V-I_All isochrones.png")

# Axis labels and grid
plt.xlabel('V-I colour')
plt.ylabel('Apparent V-band Magnitude ')
plt.title('Star Cluster and Isochrone for V-I')
plt.grid(True)
plt.legend(loc='lower left')

# Plot to screen
plt.show()


distance=10.0*pow(10,0.2*DM)
print(distance)

print('For comparison, literature values for the Star Cluster distance are about 9kp
```



```
10000.0
For comparison, literature values for the Star Cluster distance are ab
out 9kpc
```

In [404]:

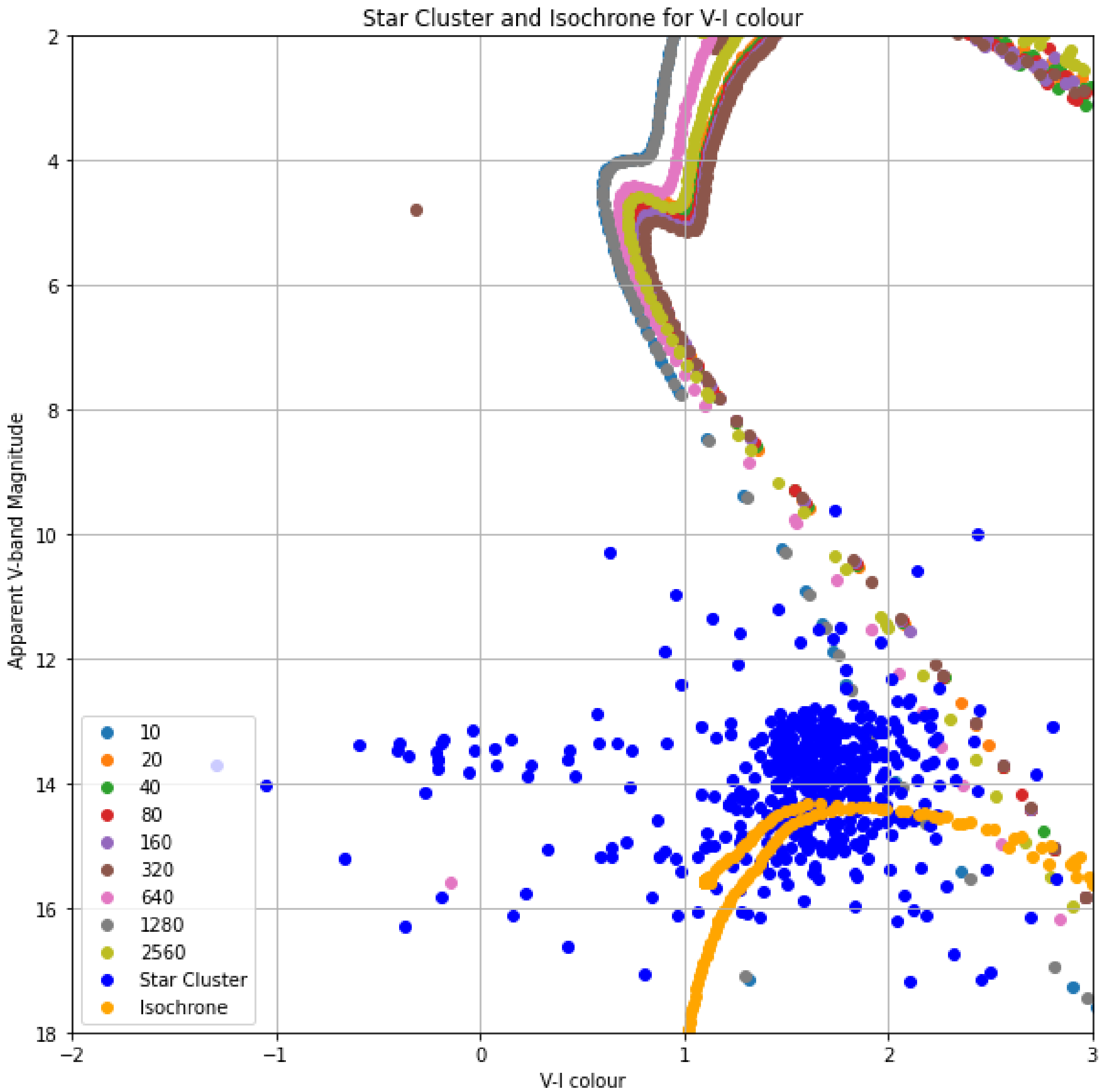
```
# Solution

# Define plot size

plt.axis([-2, 5, 18, 2.0])
# Axis labels and grid
plt.xlabel('V-R colour')
plt.ylabel('Apparent V-band Magnitude ')
plt.title('Star Cluster')
plt.grid(True)
px=[]
py=[]
for idx, mag in enumerate(V_mag):
    py.append(V_mag[idx])
    px.append(V_mag[idx]-R_mag[idx])
plt.scatter(px,py,label='Star Cluster',color='blue')

plt.rcParams['figure.figsize'] = [10, 10]
EBV=0.00
EVR=1.0*EBV       # dust reddening making the V-R colour 0.1 magnitudes redder XXXXX
RV=3.1*EBV      # dust will make the magnitudes appear fainter by 3.1 * E(B-V) - fai
DM =15     # Distance modulus

VR=np.array(isochrone[2].V)-np.array(isochrone[2].R)
V=np.array(isochrone[2].V) +DM
plt.scatter(VR, V, label='Isochrone',color='orange')                      # Scatt

# Axis labels and grid
plt.xlabel('V-R colour')
plt.ylabel('Apparent V-band Magnitude ')
plt.title('Star Cluster and Isochrone for V-R colour')
plt.grid(True)
plt.legend(loc='lower left')


# Output file, if wanted
plt.savefig("V-I_All isochrones.png")

# Axis labels and grid
plt.xlabel('V-R colour')
plt.ylabel('Apparent V-band Magnitude ')
plt.title('Star Cluster and Isochrone for V-R colour')
plt.grid(True)
plt.legend(loc='lower left')

# Plot to screen
plt.show()


distance=10.0*pow(10,DM*0.2)+RV
print(distance)

print('Literature values for the Star Cluster distance are about 9.6 kpc')
```




```
10000.0
Literature values for the Star Cluster distance are about 9.6 kpc
```

In [393]:

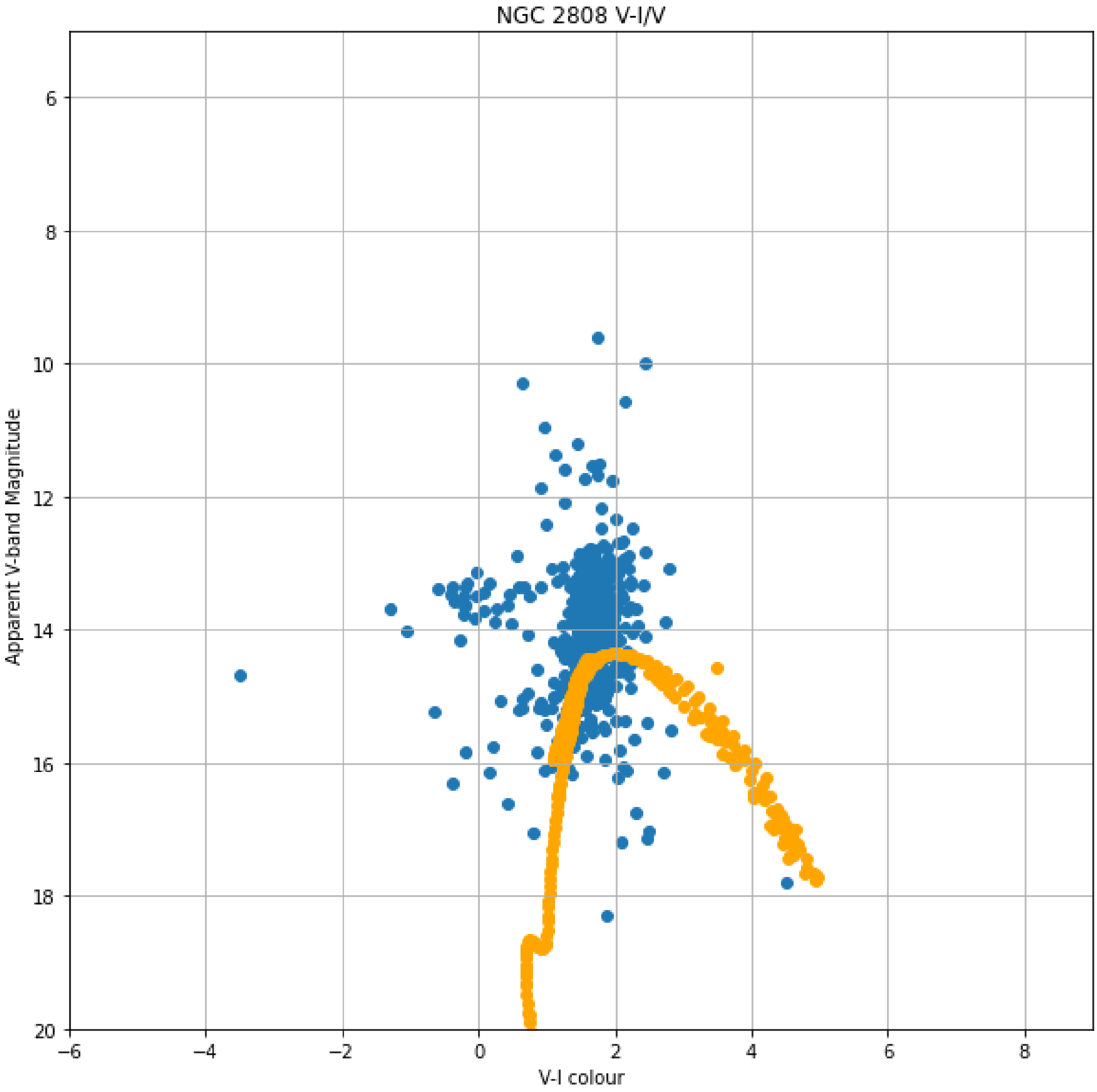
```
# Define plot size
plt.rcParams['figure.figsize'] = [10, 10]
DM=14
EVI=0.05       # dust reddening making the V-R colour 0.1 magnitudes redder
RV=3.1*EVI     # dust will make the magnitudes appear fainter by 3.1 * E(B-V) - fai

for tiso in isochrone:
    VI=np.array(tiso.V)-np.array(tiso.I)+EVI
    V=np.array(tiso.V)+VI
    plt.scatter(VI, V, label=str(tiso.age))                                 # Scatter plot

plt.axis([-2, 3, 18, 2.0])
# Axis labels and grid
plt.xlabel('V-I colour')
plt.ylabel('Apparent V-band Magnitude ')
plt.title('Star Cluster')
plt.legend(loc='lower left')
plt.grid(True)
px=[]
py=[]
for idx, mag in enumerate(V_mag):
    py.append(V_mag[idx])
    px.append(V_mag[idx]-I_mag[idx])
plt.scatter(px,py,label='Star Cluster',color='blue')


VI=np.array(isochrone[2].V)-np.array(isochrone[2].I)
V=np.array(isochrone[2].V)+DM+VR
plt.scatter(VI, V, label='Isochrone',color='orange')                                  # Scatt

# Output file, if wanted
# plt.savefig("test.png")

# Axis labels and grid
plt.xlabel('V-I colour')
plt.ylabel('Apparent V-band Magnitude ')
plt.title('Star Cluster and Isochrone for V-I colour')
plt.grid(True)
plt.legend(loc='lower left')

# Plot to screen
plt.show()


distance=10.0*pow(10,0.2)+RV
print(distance)

print('For comparison, literature values for the Star Cluster distance are about 86
```




```
16.003931924611138
For comparison, literature values for the Star Cluster distance are ab
out 86 pc
```

In [394]:

```
# need a V-I against V
# Define plot size
plt.rcParams['figure.figsize'] = [10, 10]
plt.axis([-6, 9, 20, 5])

# Axis labels and grid
plt.xlabel('V-I colour')
plt.ylabel('Apparent V-band Magnitude ')
plt.title('NGC 2808 V-I/V')
plt.grid(True)
px=[]
py=[]
for idx, mag in enumerate(V_mag):
    py.append(V_mag[idx])
    px.append(V_mag[idx]-I_mag[idx])
plt.scatter(px,py)
DM=15
VI=np.array(isochrone[1].V)-np.array(isochrone[1].I)
V=np.array(isochrone[1].V)+DM
plt.scatter(VI, V, label='Isochrone',color='orange')                          # Scatt

plt.show()

                          # Scatter plot
```

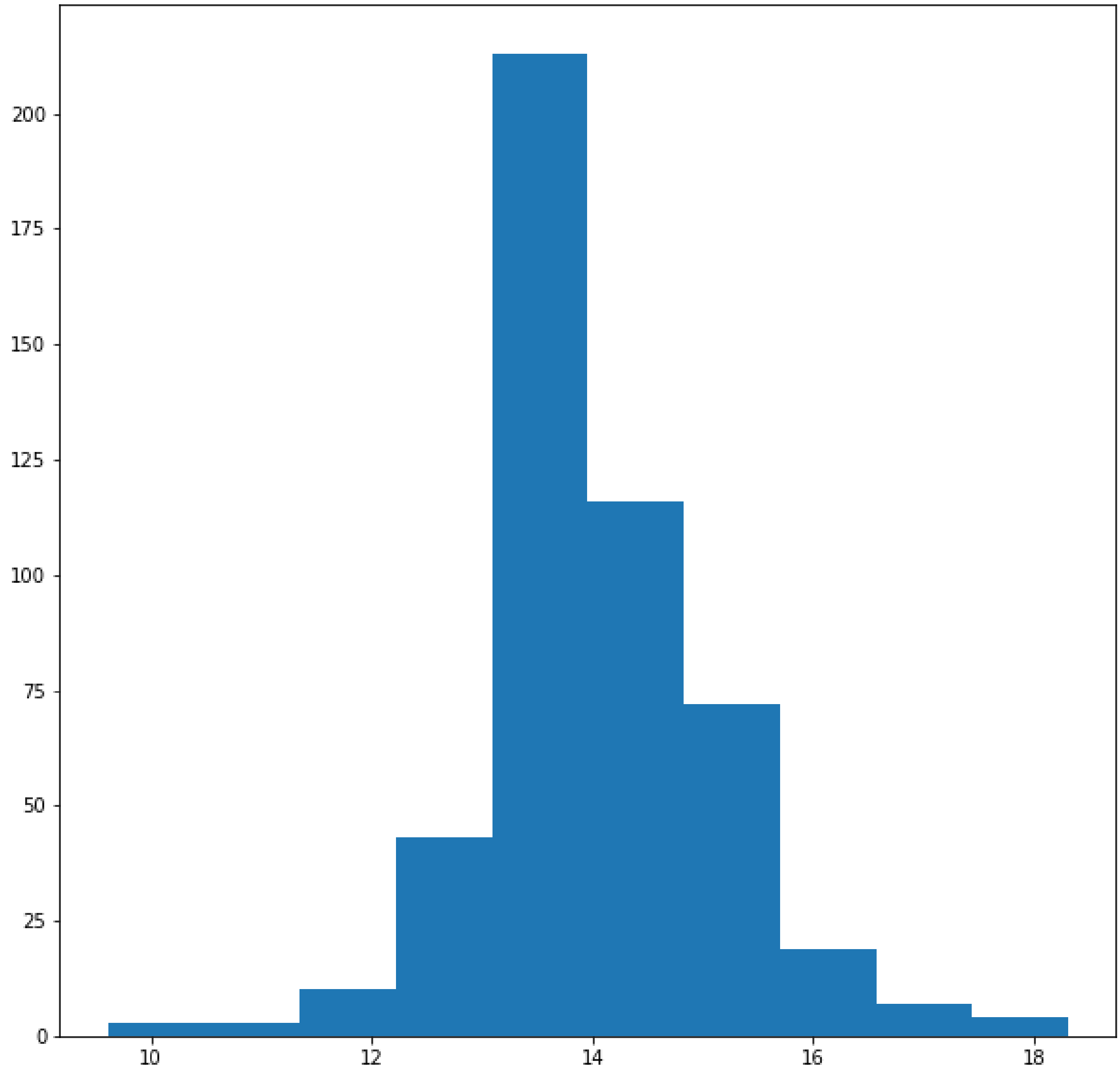

NGC 2808 V-I/V
Apparent V-band Magnitude
V-I colour
6
8
10
12
14
16
18
20
−6
−4
−2
0
2
4
6
8

In [395]:

```
plt.hist(V_mag)
```

Out[395]:

```
(array([  3.,   3.,  10.,  43., 213., 116.,  72.,  19.,   7.,   4.]),
 array([ 9.60661194, 10.47669306, 11.34677419, 12.21685531, 13.0869364
4,
        13.95701756, 14.82709869, 15.69717981, 16.56726094, 17.4373420
6,
        18.30742319]),
 <BarContainer object of 10 artists>)
```



In [396]:

```
for idx in range(len(V_mag)):
    print(V_mag[idx]-I_mag[idx])
```

```
1.2682737162705173
-0.6625873980703041
1.2704746055946998
0.5891776875809835
2.4977549262263636
1.7673431982702255
1.564786124221703
2.157332994477688
1.7951313261761044
1.7865916980923764
1.1572816730702868
2.040258660466735
1.272777552531874
1.8053118557560026
nan
2.7259116800694034
1.6853410724272742
0.6407317834624724
1.3699523182209177
```

In [397]:

```
print(V_mag)
```

```
[11.575777481718085, 15.223842025709857, 14.133060225917792, 15.191836
327609044, 17.03859233764865, 14.885185640396084, 13.076134740137954,
15.367090132390778, 12.469944708096115, 12.17950864998501, 13.25976144
0391573, 16.219916746290572, 14.93694877675702, 14.73552331025991, na
n, 13.869048029975446, 14.422315096345791, 15.184052615981491, 14.3875
3002921245, 14.095196196103547, nan, 15.031873956662132, 15.5207943628
44814, 15.072059771022204, 15.96097000370493, 14.071584833961719, 15.1
3121190281211, 13.676089968344385, 17.145805500975875, 13.089583008345
423, 17.805522666894138, 14.045088637851855, 13.993350521567725, 14.60
3753099300045, 14.490086711405747, 15.395893431631231, 13.718243303935
303, 15.372780724323672, 14.366011927342662, 13.672963425896567, 14.87
2983642627126, 17.055076669068413, 13.649214105689243, 14.455129001959
95, 13.681438304306763, 12.09875807071633, 14.507794983095714, 15.3220
83480783322, 17.269562363249726, 14.981992821543933, 12.88763392969735
5, 14.792051838990737, 14.110588250427963, 13.092707582086343, 15.4446
15502292544, 14.666586473962258, 15.809024196176097, 16.1505966657572
3, 15.041036802107854, 11.504467751874088, 14.087673006909707, 15.1818
52794847698, 15.444577837618228, 14.59015773302911, 14.42388358056518,
14.204895777849089, 14.60906846989011, 14.608911461256387, 15.17442392
2610723, 13.929884246585868, 14.810132074400606, 16.162550723580882, 1
4.098276617506535, 13.830410570889034, 13.56777015603289, 15.003331695
263773, 14.337709430935725, 14.251567492859829, 14.776999015343746, 1
3.830033888215453, 16.294391982351762, 13.715063435121479, 14.52402490
109042, 12.95254982669998, 14.875974369943542, 13.783204560817586, 14.
558442399377451, 15.328188125015101, 14.013590886375185, 15.5465900722
52354, 14.5964452044336, 15.026239856025306, 14.558068485878165, 14.33
2222393044361, 13.616687291344387, 13.332886236591493, 14.035349333405
065, 14.507376702276572, 14.258551673444977, 15.50413957182716, 13.601
922284754835, 14.206397994188455, 14.96533556700924, 13.56780800253520
3, 14.393067285654842, 13.907071764875674, 13.686457730456492, 15.0237
07971660686, 16.067300032823468, 13.534672524030796, 13.38019627853452
3, 13.827904740504838, 12.32714936199362, 13.331212552384757, 14.95107
3571164295, 13.591378657716058, 13.15669468451166, 14.108530862260412,
13.913436619355993, 12.771585650792384, 13.987809484625553, 13.4992652
7269427, 13.447708869231224, 13.116324592332557, 13.146334835104652, 1
4.836993311133721, 13.355603340383299, 13.248793475042786, 13.18213871
4602265, 13.780469662371752, 17.196075277458622, 13.159887085298438, 1
3.26897546134136, 12.892439376055862, 13.675279015891103, 13.406816399
339016, 13.499233175148204, 13.330125339127079, 9.606611936952042, 13.
41678365314751, 13.4203556104243, 13.92931791714868, 13.51754024738051
5, 18.307423189133885, 13.349794977582, 13.405378354332637, 13.6743355
35737633, 12.788704092882973, 13.901988708631542, 13.700609258502046,
13.15479905708724, 13.507730375859893, 13.184619477779568, 13.98779288
2759928, 12.684456271839707, 13.443609904364097, 12.953542389188565, 1
3.4827850388147, 13.584721297723277, 12.867793945075457, 10.2952442110
77883, 12.843142295275602, 13.806580751638114, 13.175421151649251, 12.
732071835812956, 13.005679739576673, 13.910643920786816, 13.2166403400
78731, 14.615495450113537, 13.945085518028366, 13.930663634067564, 15.
631243312860523, 13.205010037343435, 13.787169860182848, 14.6525402107
5761, nan, 13.538317010431324, 13.005706066062373, 13.035404332026577,
16.74629822598466, 13.950874960536815, 13.115185114928204, 13.43563969
8348911, 14.086016857253718, 13.37729230848555, 13.623968248890641, 1
3.256997229035019, 13.994253279033924, 15.392735298537639, 14.59787242
5189598, 13.774529535699546, 13.218701127809986, 14.35021993746571, 1
4.635480536269874, 13.157479014007881, 13.697649317197257, 14.05634048
```

0932162, 13.570324639887415, 13.394655959731502, 13.335614883563343, 1
3.678687153446976, 13.190244903650013, 16.047591276971055, 13.22403198
8213657, 13.40235312124329, 13.653127713875168, 13.092853828253745, 1
3.45219160573594, 13.374370144099695, 13.801466976836098, 15.208664101
910461, 13.06365774263681, 13.375874590061215, 13.920418802888312, 13.
27450409739431, 12.962423807782285, 13.277336950519839, 13.56857075085
722, 12.813677202611027, 14.696212380013225, 13.769840431826319, 12.87
0023879381938, 12.693628512379735, 11.74008834881157, 12.9244234350311
87, 13.027079419719096, 13.049104756918272, 15.676065084304096, 14.346
179593803775, 13.221166378707467, 14.61765984642129, 13.09523278614063
8, 15.056248945951285, 13.143386903229601, 13.1184002118809, 13.95171
3650916444, 13.638972417739218, 13.922027088086944, 13.50980975099280
1, 13.359131982871203, 14.138619944607683, 13.327649724601576, 13.5289
40708519274, 13.410866604108227, 14.751410073487946, 13.13034734108984
8, 14.210753695951567, 15.333507550585423, 14.023963234989132, 16.0942
29651023475, 12.925935470912345, 13.271707033455625, 13.01731726572535
8, 13.494922513589739, 13.363698545783834, 13.958302443489911, 13.7507
31198990927, 16.114060285073744, 15.09326261210206, 13.57467973502963
2, 13.773438914681016, 14.331490534414566, 13.829298576390553, 13.0102
6238985473, 12.825297349548348, 13.823004951603828, 13.04545801865946
2, 14.30189959619701, 13.419819284319686, 13.355424272727381, 13.96803
6564718313, 13.012562657331866, 13.744751019843811, 13.75313642836214
4, 11.678950540063155, 13.958488180307308, 13.115616482707168, 13.2957
18436812633, 12.65751051134244, 13.622528218004883, 13.22009967884786,
13.439977987657027, 13.616384852429976, 13.36920173503714, 13.18740867
8864463, 13.140226232158744, 14.67981797274957, 13.531653638851337, 1
4.273890798501728, 12.975877970976196, 12.97639135276114, 14.333084499
334495, 14.435555897075321, 13.559719230080216, 14.431319654789744, 1
3.424030312910805, 13.814399888541661, 13.255318100177808, 13.95474037
7045814, 13.515486926971848, 12.948181048991284, 14.683482508031277, 1
4.130171776501506, 13.26062450705211, 16.128910263748033, 13.118002063
671838, 13.311670490368419, 14.272709558615524, 13.114268642690055, 1
3.82709536919346, 15.231204687790136, 13.469759045978343, nan, 13.3699
65557927879, 13.494818423701554, 13.694651687918832, 13.92845612352003
2, 13.80302535395408, 14.06214044995636, 13.956458877936461, 14.046187
53489914, 13.675179641399804, 13.353146761354411, 13.61354647452353, 1
3.70573604248451, 14.678869747814037, 12.898395128464903, 13.430266621
994104, 14.189708433834042, 13.621192482599028, 14.841432307002684, 1
4.016101201824174, 13.689581133502193, 13.54981876422965, 13.789079369
311702, 13.787231614794598, 13.747212767070375, 13.361015734037272, 1
3.148322521923644, 13.588745644905085, 13.584863582124985, 14.04266252
428432, 14.331666257804994, 14.57571910159223, 12.462871219344626, 12.
998960107575465, 11.747647649274862, 14.258699893565675, 14.1449420561
95678, 14.327459137150756, 15.83088041344049, 13.271534761596538, 13.4
8039735069842, 11.535521218201005, 14.686764704253314, 13.480216570258
13, 14.236196300750937, 13.752306619647277, 13.507715248050175, 13.263
542982495936, 15.442644090568002, 12.934084863351048, 14.7039723619254
87, 14.94249955519042, 15.107828281309562, 13.482735962793498, 14.1836
56999573028, 14.951674442432118, 13.366345862550641, 13.93549551240692
4, 14.133050464910278, 13.921356173849855, 14.156023600285618, 14.1867
6707966863, 13.97001869505102, 13.701216902176004, 14.943384942204553,
13.650050166212846, 13.172110910307586, 14.11681341779857, 14.56484172
6016382, 13.936141978490134, 14.27405997010504, 14.199730306658985, 1
4.78302378736382, 14.838579349577484, 13.038721970624696, 15.099524762
96303, 14.325202016066596, 14.848718418109428, 13.472609184719662, 17.
98058566794768, 13.025144871229132, 13.614211947722563, 14.60651085477
9147, 14.845321607235038, 14.644425599855113, 10.582060253309614, 14.7
49554538284432, 9.998536830029277, 14.689375927529472, 14.893542921399
54, 15.766688375211807, 14.634905134207413, 14.314242482419385, 14.212

```
57433449226, 13.685463178792169, 13.222832139712015, 15.27317134145614
1, 13.617909462532452, 14.941904316248383, 15.413359746486424, 16.6181
79529274165, 14.606288845380117, 15.026787738310288, 15.01171391666458
8, 13.732109079006472, 14.500202691792008, 14.24782097502569, 15.04501
0946981714, 11.359229283616258, 15.062784489947031, 13.92167947162827,
12.980637086139605, 13.367763235650012, 14.28561076073815, 15.10656339
684364, 14.234646307551953, 15.210962586058855, 14.861798318645942, 1
1.869047980524602, 15.204372900384456, 14.610673635942074, 14.23131372
3600197, 15.434638223384816, 14.566155487120238, 13.938296566750836, 1
5.718416579187927, 14.705910816559022, 14.52553672226878, 14.940602898
628807, nan, 11.215617304514117, nan, nan, 13.077905450469071, 13.4889
74052563314, nan, 15.015419823408967, 15.641164210829274, 16.059154451
85194, 15.753539629948293, 13.691516509144979, 18.249504280060922, 13.
340478584959714, 15.199140149271571, 15.262735211094462, 10.9666666760
57102, 15.056919067844685, nan, 15.212138345823394, 14.90554646337252
4, 13.822987397368149, 14.993741964229198, 13.88550122452147, 14.16133
083297133, 13.632511176849064, 13.897283202326575, 14.198514076339853,
13.471633176259246, 14.94228060754756, 13.343410794597322, 13.62586408
0239381, 13.35686810394681, 13.480876741871933, 13.6977733985184, 14.3
42980884490688, 15.15382153527403, 13.438040876119262, 13.623561867265
284, 13.479885893841988, 15.826639135026522, 12.405551693169102, 13.37
1241277605343, 13.46983707781946, 13.49209213912148, 13.34679232557575
5, 13.297292256390229, 13.755723453146508, 13.570000095662731, 13.3529
50338016427, 13.14227731718122, 13.696027476982005, 13.29296211397877
2, 13.700331527382671, 16.110262291243018, 14.670158116973303, 15.8889
19533749643, 14.018292057989669]
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: