

```
/usr/bin/env python
```

```
2 # encoding: utf-8
3 """
4 FieldEllipticals.py
5
6 Created by Sami-Matias Niemi on 2008-07-03.
7 Copyright (c) 2008 __Sami-Matias Niemi__. All rights reserved.
8 """
9
10 #Optimizations:
11 # - maps possible field ellipticals
12 # - maps possible companions for these field ellipticals
13 # - tests if these mapped galaxies actually for a field elliptical system..
14
15 def cubicRealRoot(a, b, c, d):
16     """
17     Calculates the real roots of cubic equation. Note: returns only the
18     _real_ roots!
19     Code as found at http://www.josechu.com/ecuaciones\_polinomicas/
20     """
21     twoonethird = 2.0**(1.0 / 3.0)
22     delta = (-2.0 * b * b * b + 9.0 * a * b * c - 27.0 * a * a * d + ((4.0 *
23         (-b * b + 3.0 * a * c)**3.0)**(1.0/2.0)) + ((-2.0 * b * b * b + 9.0
24         * a * b * c - 27.0 * a * a * d)**2.0))**(1.0 / 3.0)
25     result = (-b / (3.0 * a) - (twoonethird * (-b * b + 3.0 * a * c)) / (3.0
26         * a * delta) + delta / (3.0 * twoonethird * a))
27     return result
28
29 def MillenniumSimulationColumns():
30     """
31     A dictionary that contains columns of the Millennium Simulation Galaxy
32     catalogue data.
33     """
34     MSdata = {
35         'galaxyID' : 0, 'lastProg' : 1, 'descedantID' : 2, 'haloID' : 3, '
36         subHaloID' : 4,
37         'fofID' : 5, 'treeId' : 6, 'firstProg' : 7, 'nextProg' : 8, 'typee'
38         : 9,
39         'snapnum' : 10, 'redshift' : 11, 'centralMvir' : 12, 'phkey' : 13, '
40         x' : 14, 'y' : 15,
41         'z' : 16, 'zIndex' : 17, 'ix' : 18, 'iy' : 19, 'iz' : 20, 'velX' :
42         21, 'velY' : 22,
43         'velZ' : 23, 'np' : 24, 'mvir' : 25, 'rvir' : 26, 'vvir' : 27, 'vmax
44         ' : 28, 'coldGas' : 29,
45         'stellarMass' : 30, 'bulgeMass' : 31, 'hotGas' : 32, 'ejectedMass' :
46         33, 'blackholeMass' : 34,
47         'metalsCG' : 35, 'metalsSM' : 36, 'metalsBM' : 37, 'metalsHG' : 38,
48         'metalsEM' : 39,
49         'sfr' : 40, 'sfrBulge' : 41, 'xrayLum' : 42, 'diskRadius' : 43, '
50         coolingR' : 44,
51         'mag_bc' : 45, 'mag_vc' : 46, 'mag_rc' : 47, 'mag_ic' : 48, 'mag_kc'
52         : 49, 'mag_bB' : 50,
53         'mag_vB' : 51, 'mag_rB' : 52, 'mag_iB' : 53, 'mag_kB' : 54, 'mag_bD'
54         : 55, 'mag_vD' : 56,
55         'mag_rD' : 57, 'mag_iD' : 58, 'mag_kD' : 59, 'massWAge' : 60, '
56         random' : 62}
57     return MSdata
58
59 def MillenniumSimulationFormat():
60     """
```

```

45  Formation file for Millennium Simulation Galaxy catalogue data.
46  """
47  format = {'names': ('galaxyID', 'lastProg', 'descendantID', 'haloID', '
    subHaloID',
48      'fofID', 'treeId', 'firstProg', 'nextProg', 'typee',
49      'snapnum', 'redshift', 'centralMvir', 'phkey', 'x', 'y',
50      'z', 'zIndex', 'ix', 'iy', 'iz', 'velX', 'velY',
51      'velZ', 'np', 'mvir', 'rvir', 'vvir', 'vmax', 'coldGas',
52      'stellarMass', 'bulgeMass', 'hotGas', 'ejectedMass', '
    blackholeMass',
53      'metalsCG', 'metalsSM', 'metalsBM', 'metalsHG', 'metalsEM',
54      'sfr', 'sfrBulge', 'xrayLum', 'diskRadius', 'coolingR',
55      'mag_bc', 'mag_vc', 'mag_rc', 'mag_ic', 'mag_kc', 'mag_bB',
56      'mag_vB', 'mag_rB', 'mag_iB', 'mag_kB', 'mag_bD', 'mag_vD',
57      'mag_rD', 'mag_iD', 'mag_kD', 'massWAge', 'random'),
58      'formats': ('q', 'q', 'q', 'q', 'q', 'q', 'q', 'q', 'q', 'q',
59          'l', 'l', 'f', 'f', 'l', 'f', 'f', 'f', 'q',
60          'l', 'l', 'l', 'f', 'f', 'f', 'l', 'f', 'f',
61          'f', 'f', 'f', 'f', 'f', 'f', 'f', 'f', 'f',
62          'f', 'f', 'f', 'f', 'f', 'f', 'f', 'f', 'f',
63          'f', 'f', 'f', 'f', 'f', 'f', 'f', 'f', 'f',
64          'f', 'f', 'f', 'f', 'f', 'f', 'f', 'f', 'l')}
65  return format
66
67  def tofile(fname, X, fmt='%4.1e', delimiter=' ', header = '#Header'):
68      from numpy import savetxt
69      #from pprint import pprint
70
71      fmtToType = { 'd': int,
72                    'f': float,
73                    'e': float }
74      if type(fmt) == tuple:
75          fh = open(fname, 'w')
76          fh.write(header + '\n')
77          for row in X:
78              #pprint(zip(fmt, row))
79              fh.write(delimiter.join(
80                  [(ft % fmtToType[ft[-1]](value)) for ft, value
81                     in zip(fmt, row)]
82                  ) + '\n')
83              fh.close()
84      else:
85          savetxt(fname, X, fmt, delimiter)
86
87  def basicStats(data):
88      """Calculates basic statistics from a given array"""
89      if (len(data) > 1):
90          import numpy as N
91          median = N.median(data)
92          mean = N.mean(data)
93          std = N.std(data)
94          max = N.max(data)
95          min = N.min(data)
96          var = N.var(data)
97          return mean, median, std, var, max, min
98      else:
99          return (-99,)*6
100
101  def main():

```

```
102     """Main program.
103     """
104     import sys
105     import os.path
106     import ConfigParser
107     from getopt import getopt
108     import numpy
109     import time
110     import numpy.linalg
111
112     starttime = time.time()
113     verbose = False
114
115     log = file("log.out", 'w')
116     progress = file("progress.out", 'w')
117
118     (opts, args) = getopt(sys.argv[1:], 'v')
119     for opt, val in opts:
120         if opt == '-v': verbose = True
121
122     welcome = "\nThis program searches Millennium Simulation Galaxy data for
123             field ellipticals!\n"
124     start = "The run was started at %s \n" % time.asctime(time.localtime())
125     if verbose:
126         print welcome
127         print start
128
129     log.write('This file contains the log of the FieldElliptical.py -program
130             !')
131     log.write(welcome)
132     log.write(start)
133
134     if len(args) < 1:
135         print "Wrong number of commandline arguments! Give me the name of
136             the parameter file!"
137         sys.exit(-1)
138
139     path, fname = os.path.split(args[0])
140
141     #this is for parsing the config file
142     config = ConfigParser.ConfigParser()
143     config.read(fname)
144
145     #parsing the parameters
146     filename = config.get('default_run', 'filename')
147     deltamag1 = config.getfloat('default_run', 'deltamag1')
148     deltamag2 = config.getfloat('default_run', 'deltamag2')
149     distance1 = config.getfloat('default_run', 'distance1')
150     distance2 = config.getfloat('default_run', 'distance2')
151     ellimit = config.getfloat('default_run', 'Ellimit')
152     maglimit = config.getfloat('default_run', 'maglimit')
153     xlow = config.getfloat('default_run', 'xlow')
154     xup = config.getfloat('default_run', 'xup')
155     ylow = config.getfloat('default_run', 'ylow')
156     yup = config.getfloat('default_run', 'yup')
157     zlow = config.getfloat('default_run', 'zlow')
158     zup = config.getfloat('default_run', 'zup')
159     safedist = config.getfloat('default_run', 'safedistance')
160
161     log.write("You selected to use a following cube:\n")
```

```

159 log.write("%6.2f <= x <= %6.2f\n" % (xlow+safedist, xup-safedist))
160 log.write("%6.2f <= y <= %6.2f\n" % (ylow+safedist, yup-safedist))
161 log.write("%6.2f <= z <= %6.2f\n" % (zlow+safedist, zup-safedist))
162
163 #defines MS data column constants as a dictionary
164 MS = MillenniumSimulationColumns()
165
166 #reads the whole file
167 #datafloat = numpy.loadtxt(filename, comments = '#', delimiter=',',
168     skiprows=0)
169 #firstof = data[0,MS['x']]
170 form = MillenniumSimulationFormat()
171 data = numpy.loadtxt(filename, comments = '#', delimiter=' ', skiprows=
172     0, dtype=form)
173 #firstof = data[0][MS['x']]
174
175 galaxies = len(data)
176
177 found = "\nFound %i galaxies from your data file!\n" % galaxies
178 if verbose:
179     print found
180 log.write(found)
181 log.flush()
182
183 #results variables
184 fieldEs = 0; companions = []; results = []; FEllipticals = [];
185 Ellipticals = []
186
187 for line1, galaxy in enumerate(data):
188     if (line1 % 500 == 0):
189         progress.write("%10.6f per cent done...\n" % (float(line1)/float
190             (galaxies)*100.))
191         progress.flush()
192         #print line1
193         #resets temp variables
194         fieldElliptical = False
195         comp1 = []; comp2 = [];
196         #Tests if in safe area
197         if (((galaxy[MS['x']] - xlow) >= safedist) and
198             ((galaxy[MS['y']] - ylow) >= safedist) and
199             ((galaxy[MS['z']] - zlow) >= safedist) and
200             ((xup - galaxy[MS['x']]) >= safedist) and
201             ((yup - galaxy[MS['y']]) >= safedist) and
202             ((zup - galaxy[MS['z']]) >= safedist)):
203             #Just to be sure the bulge magnitude is not 99
204             if (galaxy[MS['mag_bB']] < 30):
205                 bulgemagdiff = galaxy[MS['mag_bB']] - galaxy[MS['mag_bc']]
206                 T = cubicRealRoot(0.0047, -0.054, 0.342, - bulgemagdiff)
207                 - 5.0
208                 galaxy = numpy.void.tolist(galaxy)
209                 galaxy = galaxy + (T,)
210                 if ((T <= ellimit) and (galaxy[MS['mag_bc']] <= maglimit
211                     )):
212                     fieldElliptical = True
213                     for line2, companion in enumerate(data):
214                         if (line1 != line2 and fieldElliptical):
215
216                             #calculates the distance between the objects
217                             coordsGal = numpy.array( (galaxy[MS['x']],

```

```

212         galaxy[MS['y']], galaxy[MS['z']]) )
213     coordsCompanion = numpy.array( (companion[MS
214         ['x']], companion[MS['y']], companion[MS[
215         'z']]) )
216     distance = numpy.linalg.norm( coordsGal -
217         coordsCompanion )
218
219     #calculates the magnititude difference
220     magdif = abs(galaxy[MS['mag_bc']] -
221         companion[MS['mag_bc']])
222
223     #tests if companion fulfils field elliptical
224     criteria
225     #breaks the loop if not
226     if (distance <= distance1):
227         if (magdif <= deltamag1):
228             fieldElliptical = False
229             break
230         if (galaxy[MS['mag_bc']] >= companion[MS
231             ['mag_bc']]):
232             fieldElliptical = False
233             break
234     if (distance <= distance2):
235         if (magdif <= deltamag2):
236             fieldElliptical = False
237             break
238         if (galaxy[MS['mag_bc']] >= companion[MS
239             ['mag_bc']]):
240             fieldElliptical = False
241             break
242
243     #saves the line number of companions and
244     their morphology
245     if (distance <= distance1 and magdif >
246         deltamag1):
247         if (distance <= distance2 and magdif >
248             deltamag2):
249             T2 = 99
250             if (companion[MS['mag_bB']] <= 0 and
251                 companion[MS['mag_bc']] <= 0):
252                 bulgemagdiff2 = companion[MS[
253                     'mag_bB']] - companion[MS[
254                     'mag_bc']]
255                 T2 = cubicRealRoot(0.0047,
256                     -0.054, 0.342, -
257                     bulgemagdiff2) - 5.0
258             else : T2 = 9
259             comp = numpy.void.tolist(companion)
260             comp += (T2,)
261             comp2.append(comp)
262     else:
263         T1 = 99
264         if (companion[MS['mag_bB']] <= 0 and
265             companion[MS['mag_bc']] <= 0):
266             bulgemagdiff1 = companion[MS[
267                 'mag_bB']] - companion[MS[
268                 'mag_bc']]
269             T1 = cubicRealRoot(0.0047,
270                 -0.054, 0.342, -
271                 bulgemagdiff1) - 5.0

```

```

251         else: T1 = 9
252         comp = numpy.void.tolist(companion)
253         comp += (T1,)
254         comp1.append(comp)
255
256         #saves non field ellipticals
257         if (fieldElliptical == False and T <= 0):
258             Ellipticals.append(galaxy)
259
260     #saves output data
261     if (fieldElliptical):
262         fieldEs +=1
263         galaxy += (len(comp1), len(comp2))
264         resultsgal = (0,) + galaxy
265
266         results.append(resultsgal)
267         FEllipticals.append(galaxy)
268
269         for line in comp1:
270             #for line, T in comp1:
271                 #results.append(data[line])
272                 res = (1,) + line + (len(comp1), len(comp2))
273                 results.append(res)
274                 companions.append(line)
275             #print results
276
277         for line in comp2:
278             #for line, T in comp2:
279                 #results.append(data[line])
280                 res = (2,) + line + (len(comp1), len(comp2))
281                 results.append(res)
282                 companions.append(line)
283
284     progress.close()
285
286     #print results
287     #Formats the output
288     outformFE = ('%d', '%d', '%d', '%d', '%d', '%d', '%d', '%d', '%d', '%d', '%d', '%d', '%f',
289                 '%f',
290                 '%d', '%f', '%f', '%f', '%d', '%d', '%d', '%d', '%f', '%f', '%f', '%d', '
291                 '%f',
292                 '%f', '%f', '%f', '%f', '%f', '%f', '%f', '%f', '%f', '%f', '%f', '%f', '
293                 '%f',
294                 '%f', '%f', '%f', '%f', '%f', '%f', '%f', '%f', '%f', '%f', '%f', '%f', '
295                 '%f',
296                 '%f', '%f', '%f', '%f', '%f', '%f', '%f', '%f', '%f', '%f', '%f', '%f', '
297                 '%f',
298                 '%f', '%f', '%f', '%f', '%f', '%f', '%f', '%f', '%f', '%d', '%f', '%d', '
299                 '%d')
300
301     outformmco = ('%d', '%d', '%d', '%d', '%d', '%d', '%d', '%d', '%d', '%d', '%d', '%d', '%f',

```

```

    , '%f ',
301         '%d ', '%f ', '%f ', '%f ', '%d ', '%d ', '%d ', '%d ', '%f ', '%f ', '%f ', '%d ', '
        '%f ',
302         '%f ', '%f ', '%f ', '%f ', '%f ', '%f ', '%f ', '%f ', '%f ', '%f ', '%f ', '%f ', '%f ', '
        '%f ',
303         '%f ', '%f ', '%f ', '%f ', '%f ', '%f ', '%f ', '%f ', '%f ', '%f ', '%f ', '%f ', '%f ', '
        '%f ',
304         '%f ', '%f ', '%f ', '%f ', '%f ', '%f ', '%f ', '%f ', '%f ', '%f ', '%d ', '%f ')
305
306 outformel = ( '%d ', '%d ', '%d ', '%d ', '%d ', '%d ', '%d ', '%d ', '%d ', '%d ', '%d ', '%d ', '%f '
    , '%f ',
307         '%d ', '%f ', '%f ', '%f ', '%d ', '%d ', '%d ', '%d ', '%f ', '%f ', '%f ', '%d ', '
        '%f ',
308         '%f ', '%f ', '%f ', '%f ', '%f ', '%f ', '%f ', '%f ', '%f ', '%f ', '%f ', '%f ', '%f ', '
        '%f ',
309         '%f ', '%f ', '%f ', '%f ', '%f ', '%f ', '%f ', '%f ', '%f ', '%f ', '%f ', '%f ', '%f ', '
        '%f ',
310         '%f ', '%f ', '%f ', '%f ', '%f ', '%f ', '%f ', '%f ', '%f ', '%f ', '%d ', '%f ')
311
312 #prints to files
313 fehed = '#MS columns + T #comp1 #comp2'
314 fched = '#ID + MS columns + T #comp1 #comp2'
315 cohed = '#MS columns + T'
316 elhed = '#MS columns + T'
317 tofile('FieldEllipticals.out', FEllipticals, fmt=outformFE, delimiter='
    ', header=fehed)
318 tofile('FieldEsandCompanions.out', results, fmt=outformre, delimiter='
    ', header=fched)
319 tofile('Companions.out', companions, fmt=outformco, delimiter=' ',
    header=cohed)
320 tofile('Ellipticals.out', Ellipticals, fmt=outformel, delimiter=' ',
    header=elhed)
321
322 #reads files again to different type of array
323 FieldE = numpy.loadtxt('FieldEllipticals.out', comments='#', delimiter
    =' ', skiprows=0)
324 Comps = numpy.loadtxt('Companions.out', comments='#', delimiter=' ',
    skiprows=0)
325 Ell = numpy.loadtxt('Ellipticals.out', comments='#', delimiter=' ',
    skiprows=0)
326
327 #calculates some statistics
328 Compsmass = basicStats(Comps[:,MS['mvir']])
329 CompsMagb = basicStats(Comps[:,MS['mag_bc']])
330 CompsColdGas = basicStats(Comps[:,MS['coldGas']])
331 CompsStellarMass = basicStats(Comps[:,MS['stellarMass']])
332 CompsBHMass = basicStats(Comps[:,MS['blackholeMass']])
333 CompsBulgeMass = basicStats(Comps[:,MS['bulgeMass']])
334 Ellmass = basicStats(Ell[:,MS['mvir']])
335 EllMagb = basicStats(Ell[:,MS['mag_bc']])
336 EllColdGas = basicStats(Ell[:,MS['coldGas']])
337 EllStellarMass = basicStats(Ell[:,MS['stellarMass']])
338 EllBHMass = basicStats(Ell[:,MS['blackholeMass']])
339 EllBulgeMass = basicStats(Ell[:,MS['bulgeMass']])
340 FieldEmass = basicStats(FieldE[:,MS['mvir']])
341 FieldEMagb = basicStats(FieldE[:,MS['mag_bc']])
342 FieldEColdGas = basicStats(FieldE[:,MS['coldGas']])
343 FieldEStellarMass = basicStats(FieldE[:,MS['stellarMass']])
344 FieldEBHMass = basicStats(FieldE[:,MS['blackholeMass']])
345 FieldEBulgeMass = basicStats(FieldE[:,MS['bulgeMass']])

```

```

346
347     #writes statistics to a file
348     fmtt = "%16s"*7 + "\n"
349     fmts = "%16s" + "%16.5f"*6 + "\n"
350     statfile = open('Stats.out', 'w')
351     statfile.write("#This file contains some statistics.\n")
352     statfile.write("#For field ellipticals:\n")
353     statfile.write(fmtt % ("#name", "mean", "median", "std", "var", "max", "
        min"))
354     statfile.write(fmts % (("Mvir",) + FieldEmass))
355     statfile.write(fmts % (("Mag_B",) + FieldEMagb))
356     statfile.write(fmts % (("ColdGas",) + FieldEColdGas))
357     statfile.write(fmts % (("StellarMass",) + FieldEStellarMass))
358     statfile.write(fmts % (("BlackHoleMass",) + FieldEBHMass))
359     statfile.write(fmts % (("BulgeMass",) + FieldEBulgeMass))
360     statfile.write("#For ellipticals (excluding field ellipticals):\n")
361     statfile.write(fmtt % ("#name", "mean", "median", "std", "var", "max", "
        min"))
362     statfile.write(fmts % (("Mvir",) + Ellmass))
363     statfile.write(fmts % (("Mag_B",) + EllMagb))
364     statfile.write(fmts % (("ColdGas",) + EllColdGas))
365     statfile.write(fmts % (("StellarMass",) + EllStellarMass))
366     statfile.write(fmts % (("BlackHoleMass",) + EllBHMass))
367     statfile.write(fmts % (("BulgeMass",) + EllBulgeMass))
368     statfile.write("#For companion galaxies:\n")
369     statfile.write(fmtt % ("#name", "mean", "median", "std", "var", "max", "
        min"))
370     statfile.write(fmts % (("Mvir",) + Compsmass))
371     statfile.write(fmts % (("Mag_B",) + CompsMagb))
372     statfile.write(fmts % (("ColdGas",) + CompsColdGas))
373     statfile.write(fmts % (("StellarMass",) + CompsStellarMass))
374     statfile.write(fmts % (("BlackHoleMass",) + CompsBHMass))
375     statfile.write(fmts % (("BulgeMass",) + CompsBulgeMass))
376     statfile.close()
377
378     #end of loops
379     foundFE= "Found %d Field Ellipticals from your data!\n" % fieldEs
380     if verbose:
381         print foundFE
382     log.write(foundFE)
383
384     stoptime = time.time()
385     stopstr = "Running time of the program was %.2f minutes.\n" % ((stoptime
        -starttime)/60.)
386     succ = "The program terminated successfully!\n"
387     if verbose:
388         print stopstr
389         print succ
390     log.write(stopstr)
391     log.write(succ)
392
393     log.close()
394
395 if __name__ == '__main__':
396     main()

```