Trombini_Quentin_Week11

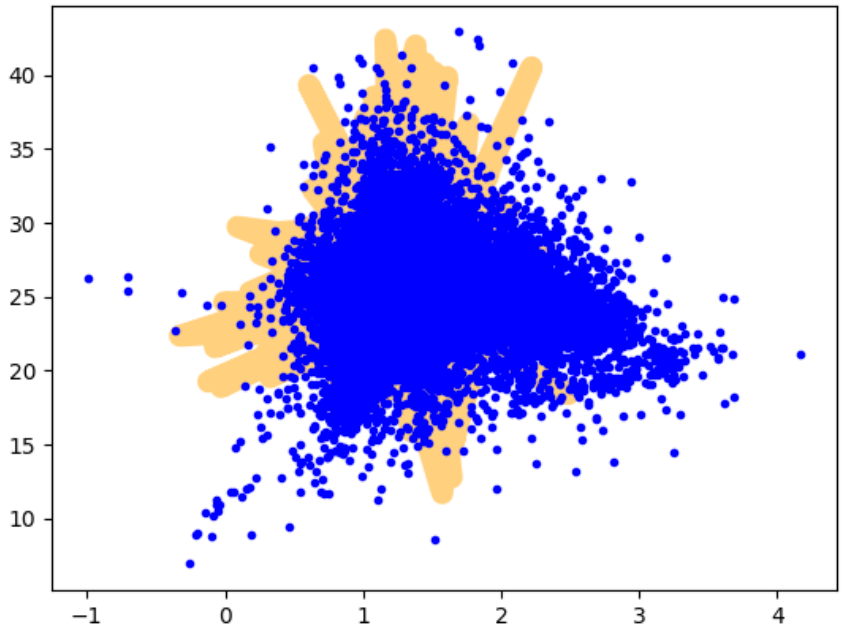
S11_01

S11_01_a

```
import os
import ssl
import numpy as np
import astropy.units as units
import astroquery.gaia as gaia
import matplotlib.figure as figure
import astroquery.simbad as simbad
import astropy.coordinates as coord
import astropy.io.votable as votable
import matplotlib.backends.backend_agg as agg

if os.path.isfile("./S11/t10.vot.gz") == False:
    ssl._create_default_https_context = ssl._create_unverified_context

    gaia.Gaia.MAIN_GAIA_TABLE ="gaiadr3.gaia_source"
    gaia.Gaia.ROW_LIMIT = -1

    target ="Trumpler 10"
    file_output ="./S11/t10.vot.gz"
    radius_arcmin = 30.0
    radius_deg = radius_arcmin / 60.0

    u_deg = units.deg
    u_ha = units.hourangle
    u_arcmin = units.arcmin

    result_simbad = simbad.Simbad.query_object(target)

    obj_ra = result_simbad["RA"][0]
    obj_dec = result_simbad["DEC"][0]

    coord = coord.SkyCoord(obj_ra, obj_dec, frame ="icrs", unit =(u_ha, u_deg))

    ra_deg = coord.ra.deg
    dec_deg = coord.dec.deg

    table = f"gaiadr3.gaia_source"
    field = f"*"
    point = f"POINT({ra_deg:8.4f} ,{dec_deg:8.4f})"
    circle = f"CIRCLE(ra, dec ,{radius_deg})"
    query = f"SELECT{field} from {table} WHERE 1= CONTAINS({point} ,{circle});"



    job = gaia.Gaia.launch_job_async(query, dump_to_file = True, output_format ="votable", output_file = file_output)
    result = job.get_results()



table = votable.parse_single_table("./S11/t10.vot.gz").to_table()

data_parallax = np.array(table ["parallax"])
data_g = np.array(table ["phot_g_mean_mag"])
```

```
data_br = np.array(table["bp_rp"])
colour = np.array(table["pseudocolour"])

distance = np.array([])

for parallax in data_parallax:
    if np.isnan(parallax) or parallax<=0:
        distance = np.append(distance,-1.0)
    else:
        distance = np.append(distance,1000/parallax)


data_g_abs = data_g + 5.0 * np.log10(distance/1000)+5

list_ms_colour = np.array([])
list_ms_absmag = np.array([])

for items,color_br in zip(data_g_abs,colour):
    try :
        colour_br = color_br
    except :
        colour_br = 999.999


    if ((colour_br < 100.0) and (items < 100.0)):
        list_ms_colour = np.append(list_ms_colour,colour_br)
        list_ms_absmag = np.append(list_ms_absmag, items)

data_ms_colour = np.array(list_ms_colour)
data_ms_absmag = np.array(list_ms_absmag)

fig = figure.Figure()
canvas = agg.FigureCanvasAgg(fig)
ax = fig.add_subplot(111)

ax.plot(data_br, data_g_abs, linestyle="None", marker ="o", markersize =3, color ="blue", zorder =0.2)
ax.plot(data_ms_colour, data_ms_absmag, linestyle="-", linewidth = 10, color ="orange", alpha =0.5, zorder =0.1)

# saving file
fig.savefig("./S11_01")
```

S11_01_b



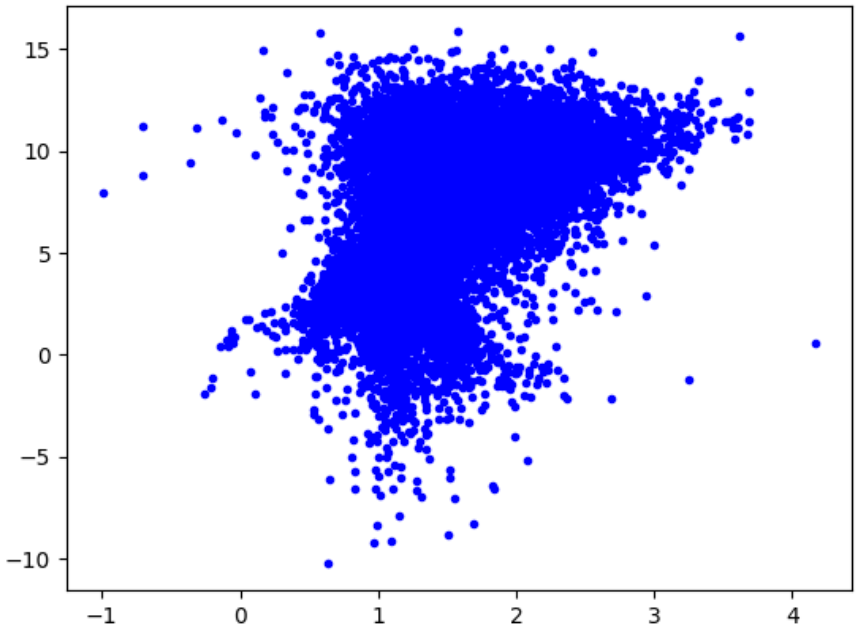
S11_02

S11_02_a

```
import os
import ssl
import numpy as np
import astropy.units as units
import astroquery.gaia as gaia
import matplotlib.figure as figure
import astroquery.simbad as simbad
import astropy.coordinates as coord
import astropy.io.votable as votable
import matplotlib.backends.backend_agg as agg

if os.path.isfile("./S11/ngc2232.vot.gz") == False:
    ssl._create_default_https_context = ssl._create_unverified_context

    gaia.Gaia.MAIN_GAIA_TABLE ="gaiadr3.gaia_source"
    gaia.Gaia.ROW_LIMIT = -1

    target ="NGC 2232"
    file_output ="./S11/ngc2232.vot.gz"
    radius_arcmin = 30.0
    radius_deg = radius_arcmin / 60.0

    u_deg = units.deg
    u_ha = units.hourangle
    u_arcmin = units.arcmin
```

```
    result_simbad = simbad.Simbad.query_object(target)

    obj_ra = result_simbad["RA"][0]
    obj_dec = result_simbad["DEC"][0]

    coord = coord.SkyCoord(obj_ra, obj_dec, frame ="icrs", unit =(u_ha, u_deg))

    ra_deg = coord.ra.deg
    dec_deg = coord.dec.deg

    table = f"gaiadr3.gaia_source"
    field = f"*"
    point = f"POINT({ra_deg:8.4f} ,{dec_deg:8.4f})"
    circle = f"CIRCLE(ra, dec ,{radius_deg})"
    query = f"SELECT{field} from {table} WHERE 1= CONTAINS({point} ,{circle});"



    job = gaia.Gaia.launch_job_async(query, dump_to_file = True, output_format ="votable", output_file = file_output)
    result = job.get_results()



table = votable.parse_single_table("./S11/ngc2232.vot.gz").to_table()

data_parallax = np.array(table ["parallax"])
data_g = np.array(table ["phot_g_mean_mag"])
data_br = np.array(table["bp_rp"])

distance = np.array([])

for parallax in data_parallax:
    if np.isnan(parallax) or parallax<=0:
        distance = np.append(distance,-1.0)
    else:
        distance = np.append(distance,1000/parallax)

data_g_abs = data_g + 5.0 * np.log10(distance/1000)+5

fig = figure.Figure()
canvas = agg.FigureCanvasAgg(fig)
ax = fig.add_subplot(111)

ax.plot(data_br, data_g_abs, linestyle="None", marker ="o", markersize =3, color ="blue", zorder =0.2)

# saving file
fig.savefig("./S11/S11_02")
```

S11_02_b



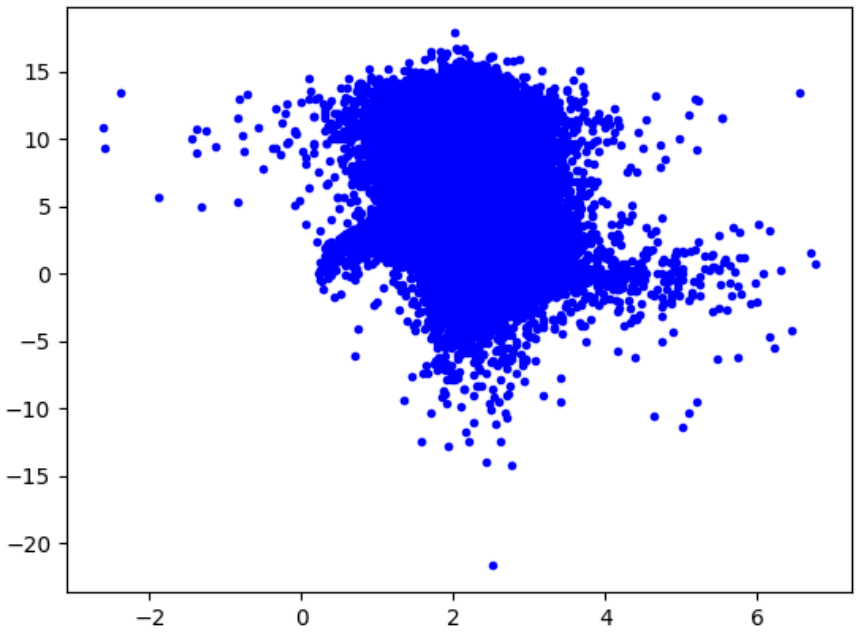
S11_03

S11_03_a

```
import os
import ssl
import numpy as np
import astropy.units as units
import astroquery.gaia as gaia
import matplotlib.figure as figure
import astroquery.simbad as simbad
import astropy.coordinates as coord
import astropy.io.votable as votable
import matplotlib.backends.backend_agg as agg

if os.path.isfile("./S11/ngc5662.vot.gz") == False:
    ssl._create_default_https_context = ssl._create_unverified_context

    gaia.Gaia.MAIN_GAIA_TABLE ="gaiadr3.gaia_source"
    gaia.Gaia.ROW_LIMIT = -1

    target ="NGC 5662"
    file_output ="./S11/ngc5662.vot.gz"
    radius_arcmin = 30.0
    radius_deg = radius_arcmin / 60.0

    u_deg = units.deg
    u_ha = units.hourangle
    u_arcmin = units.arcmin
```

```
    result_simbad = simbad.Simbad.query_object(target)

    obj_ra = result_simbad["RA"][0]
    obj_dec = result_simbad["DEC"][0]

    coord = coord.SkyCoord(obj_ra, obj_dec, frame ="icrs", unit =(u_ha, u_deg))

    ra_deg = coord.ra.deg
    dec_deg = coord.dec.deg

    table = f"gaiadr3.gaia_source"
    field = f"*"
    point = f"POINT({ra_deg:8.4f} ,{dec_deg:8.4f})"
    circle = f"CIRCLE(ra, dec ,{radius_deg})"
    query = f"SELECT{field} from {table} WHERE 1= CONTAINS({point} ,{circle});"



    job = gaia.Gaia.launch_job_async(query, dump_to_file = True, output_format ="votable", output_file = file_output)
    result = job.get_results()



table = votable.parse_single_table("./S11/ngc5662.vot.gz").to_table()

data_parallax = np.array(table ["parallax"])
data_g = np.array(table ["phot_g_mean_mag"])
data_br = np.array(table["bp_rp"])

distance = np.array([])

for parallax in data_parallax:
    if np.isnan(parallax) or parallax<=0:
        distance = np.append(distance,-1.0)
    else:
        distance = np.append(distance,1000/parallax)

data_g_abs = data_g + 5.0 * np.log10(distance/1000)+5

fig = figure.Figure()
canvas = agg.FigureCanvasAgg(fig)
ax = fig.add_subplot(111)

ax.plot(data_br, data_g_abs, linestyle="None", marker ="o", markersize =3, color ="blue", zorder =0.2)

# saving file
fig.savefig("./S11/S11_03")
```

S11_03_b



S11_01/2/3_c

The code is basically the same for the three exercise, the only only changing thing is the name of the cluster I'm getting. I could have done only one code using argparse but to show the slight difference in the three part of this report I let everything like it is.

Before further explication I apologize to present false result in the report but I don't understand which data I should use for each exercise and which discrimination criterion I should use(I really don't have time these two week to read the book you send us).

Back to the code first of all I will import all the needed library and create alisases for the one with long name.

once this is done I will use os.path.isfile to check if I already created the VOtable else I will make a query to the simbad db using simbad.Simbad.Query(<name of the target> to get the VOtable Then i will make an SQL Query to create the table from the resumt and make a file of it

Once I get the file I will use votable.parse_single_table() to get a table object ou of the file that I will be able to read later using numpy (np.array(table[columnname]))

After that there are some mathematical operation and some data sepration: the goals is to rank the cluster's members from the farthest to the closest and to remove the data that is too far away.

I think my error is coming from this part because I don't really know how to compute that

The final part is the ploting part: using matplotlib.pyplot I create a canvas and I use the plot() function to draw all the point in the colour andstyle that suit me