Tully_Fisher

November 13, 2019

0.1 Tully-Fisher Relation Homework

Author: Tatsuo Schaufus ### Task 1: Use the given data tables to calibrate a Tully-Fisher relationship for the known galaxies

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        from numpy.polynomial.polynomial import polyfit
        %matplotlib inline
```

Since there was a lot of data to be held, I decided to add it into a `dat` file which I could simply call in for use. Values for inclination angle, internal extinction and absolute magnitude were given 0s as a placeholder.

```
In [2]: # creating an array to hold all the given data for calibration
        dtype = []
        dtype.append( ('galaxy_name', 'U200') )
        dtype.append( ('Inclination_angle', float) )
        dtype.append( ('Distance_Mpc', float) )
        dtype.append( ('Apparent_magnitude_B', float) )
        dtype.append( ('Galactic_extinction', float) )
        dtype.append( ('Major_axis', float) )
        dtype.append( ('Minor_axis', float) )
        dtype.append( ('HI_width', float) )
        dtype.append( ('Galactic_extinction_internal', float) )
        dtype.append( ('Absolute_Magnitude_B', float))

        galaxies = np.genfromtxt('/home/taterz/galaxies.dat', dtype=dtype)
```

The next step was for me to determine those placeholder's real values, which were done with the estimations given from class. For the inclination angle, I used the formula $i = arccos(\frac{b}{a})$. Since `numpy` deals in radians, I manually changed the value to degrees for storage in the `Inclination_angle` flag. I also used the formula given in the assignment to determine the extinction that is internal to each galaxy and stored that in the `Galactic_extinction_internal` flag.

```
In [3]: # using the data given to find inclination, internal extinction, and absolute magnitud
```

```
galaxies['Inclination_angle'] = np.arccos(galaxies["Minor_axis"]/galaxies['Major_axis'

galaxies['Galactic_extinction_internal'] = 0.28*(galaxies["Major_axis"]/galaxies['Mino

extinct=galaxies['Apparent_magnitude_B']-galaxies['Galactic_extinction']-galaxies['Gal
galaxies['Absolute_Magnitude_B'] = -(5*np.log10(galaxies['Distance_Mpc']*100000)) + ex
```

This next cell was just to confirm what all my values were, and allow myself to check to see if they are actually making sense. I had values of absolute magnitude in the -40s which I knew were wrong, and it took far too long to realize I was using `np.log` (base e) and not `np.log10` (base 10).

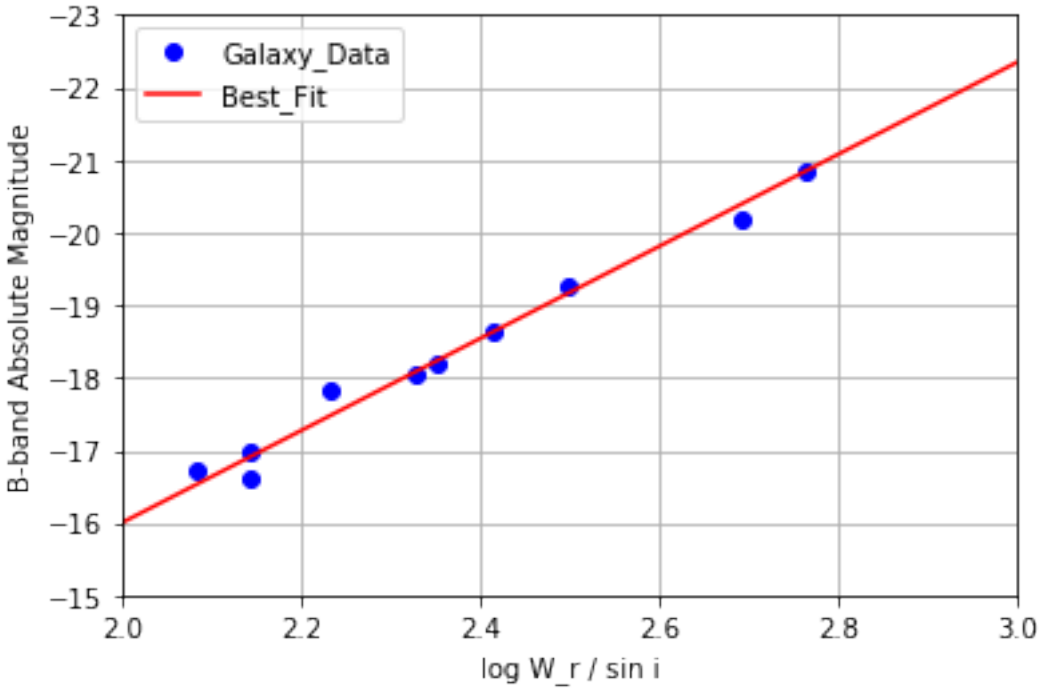
```
In [4]: galaxies
```

```
Out[4]: array([('M31', 68.56273681, 0.71 ,  4.33, 0.44, 197. , 72. , 540., 0.48611111, -20.852
               ('M33', 50.31590462, 0.817,  6.19, 0.12,  83. , 53. , 200., 0.15849057, -18.649
               ('M81', 66.42182152, 3.25 ,  7.85, 0.07,  35. , 14. , 450., 0.42      , -20.199
               ('NGC2403', 58.85261008, 3.25 ,  8.8 , 0.24,  29. , 15. , 270., 0.26133333, -19
               ('NGC4236', 69.74775326, 3.25 , 10.05, 0.02,  26. ,  9. , 200., 0.52888889, -18
               ('IC2574', 60.        , 3.25 , 10.91, 0.04,  16. ,  8. , 120., 0.28      , -16.9
               ('NGC5204', 60.        , 3.25 , 11.41, 0.19,  10. ,  5. , 120., 0.28      , -16
               ('NGC5585', 49.06727543, 7.24 , 11.25, 0.  ,   8.7,  5.7, 170., 0.14736842, -18
               ('NGC5204', 49.45839813, 7.24 , 11.62, 0.  ,   8. ,  5.2, 130., 0.15076923, -17
               ('HoIV', 65.45646012, 7.24 , 12.95, 0.  ,   6.5,  2.7, 110., 0.39407407, -16.741
              dtype=[('galaxy_name', '<U200'), ('Inclination_angle', '<f8'), ('Distance_Mpc',
```

With everything looking good, the next step was to determine the corrected rotation speed. After which I made some single letter variables to make it easier for me to use the `polyfit` function. What I found was that these best fit algorithms were not accurately finding the lines of best fit, so I had to manually adjust it. Luckily this wasn't a long process. Again, since `numpy` works with radians I had to manually change the values back in radians from degrees. It's a bit of a redundant process, but I did not use the `units` import so this was my own fault.

```
In [5]: W_r = galaxies['HI_width']/np.sin(galaxies['Inclination_angle']*np.pi/180)
        x = np.log10(W_r)
        y = galaxies['Absolute_Magnitude_B']

        # using polyfit to find best line
        m, b = polyfit(x, y, 1)

        # polyfit could use some manual adjustments
        b = b + 2.8
        m = m - 2.5

        # plotting the data, with the galaxy data as a blue circle and the best fit as a red l
        plt.plot(x, y, 'bo', label='Galaxy_Data')
        plt.plot(np.arange(10), m*np.arange(10) + b, 'r-', label='Best_Fit')

        plt.xlabel('log W_r / sin i')
        plt.ylabel('B-band Absolute Magnitude')
```

```
plt.gca().invert_yaxis()
plt.legend(loc='best')
plt.xlim(2,3)
plt.ylim(-15,-23)
plt.grid()
plt.show()
```




```
In [6]: print("The a and b values for the Tulley-Fisher relation are %.3f" % m, "and %.3f" % b
```

```
The a and b values for the Tulley-Fisher relation are -6.339 and -3.331
```

Here I plotted the values and had an extra cell output the values for the slope and intercept of the best fitting line. The data came out to a very distinct line, so making adjustments was very easy.

0.1.1 Task 2: Calculate the distance to the galaxies NGC3893, NGC3953, and NGC3992. Estimate uncertainty. Determine the Hubble constant.

With the values for the slope and intercepts calculated, I could now use this to find the absolute magnitudes elsewhere. My first step was the exact same, I set up a new array to hold the values for the knowns and unknowns for the three galaxies. Unlike before, I did not make a `dat` file. Instead I created an empty array that I would fill with values later.

```
In [7]: # creating a new array to hold the values for the estimated galaxies
        dtype2 = []
        dtype2.append( ('galaxy_name', 'U200') )
        dtype2.append( ('Distance_Mpc', float) )
        dtype2.append( ('Apparent_magnitude_B', float) )
        dtype2.append( ('HI_width', float) )
        dtype2.append( ('Absolute_Magnitude_B', float))
        dtype2.append( ('Recessional_Velocity', float))
        dtype2.append( ('Galactic_extinction', float))
        dtype2.append( ('Inclination_angle', float))

        galaxies_est = np.empty(3, dtype=dtype2)
```

In order to estimate the width of the 21 cm line, I decided to measure from the point at which the gap starts to make a distinct shift upwards and downwards. I have never worked with spectrographic data before, so I am just assuming that is how it's done. Internal extinction was calculated using NED's handy galactic coordinate and extinction calculator. Coordinate data was obtained from SIMBAD.

Data for the inclination angles were obtained from the following two papers:

- https://aas.aanda.org/articles/aas/pdf/1998/12/ds7136.pdf
- https://www.aanda.org/articles/aa/pdf/2005/04/aah4175.pdf

Or in a shorter format:

- https://tinyurl.com/yzwzs7a4
- https://tinyurl.com/ydqmgrhs

```
In [8]: # using the new array to hold the values for the estimations
        galaxies_est[0] = 'NGC3893', 0, 10.6, 280, 0, 1030, 0.077, 42
        galaxies_est[1] = 'NGC3953', 0, 10.8, 410, 0, 1040, 0.109, 59.9
        galaxies_est[2] = 'NGC3992', 0, 10.7, 500, 0, 1130, 0.106, 58.5
```

Given my extremely long variable names, I used an intermediary variable called `dist_mod` to hold the values for the distance modulus. This was calculated using the Tully-Fisher relation found previously. I also converted my distance from parsec to mega parsec manually, as I did not include the `units` import.

Since I calibrated the values for the Tully-Fisher relation previously, I use those values to estimate the B-band absolute magnitude of the 3 galaxies we need to check.

```
In [9]: # finding the values for the absolute magnitude using the Tully Fisher calibration
        galaxies_est['Absolute_Magnitude_B'] = m*np.log10(galaxies_est['HI_width']/\
                        np.sin(galaxies_est['Inclination_angle']*np.pi/180)) + b

        # accounting for extinction

        dist_mod=galaxies_est['Apparent_magnitude_B']-galaxies_est['Absolute_Magnitude_B']-gala

        galaxies_est['Distance_Mpc'] = 10**(dist_mod/5 + 1) / 1000000
```

The next line is just as before, to confirm whether or not the values make any amount of sense. Given that everything is falling within what the Schechter Function suggests the range of galaxy luminosities, I figured it was good.

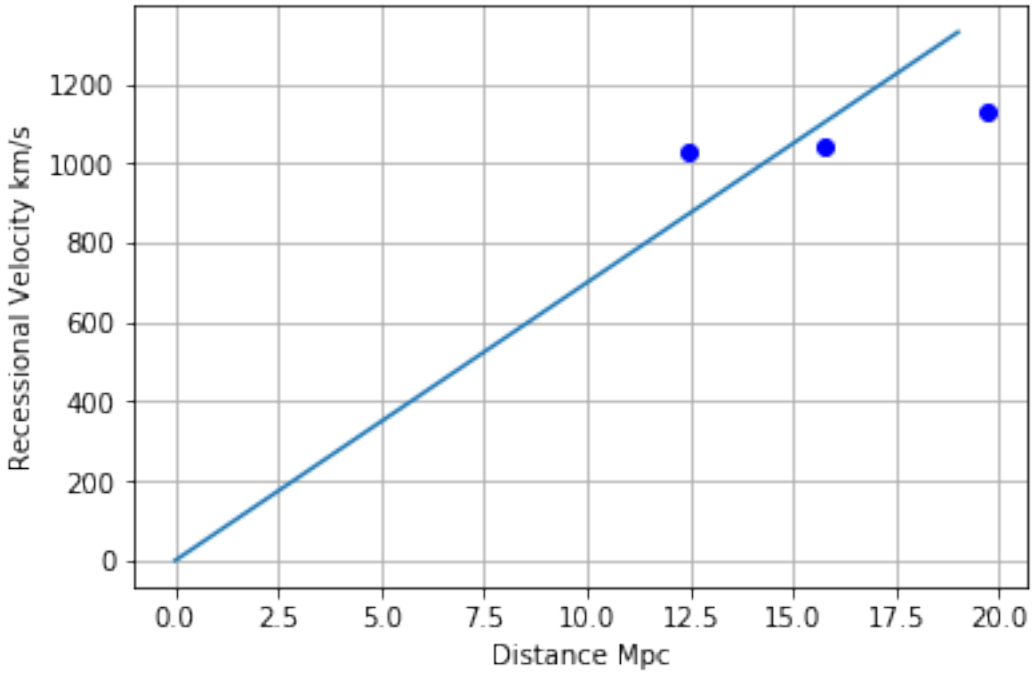
```
In [10]: galaxies_est
```

```
Out[10]: array([('NGC3893', 12.43731744, 10.6, 280., -19.95063359, 1030., 0.077, 42. ),
                ('NGC3953', 15.7343742 , 10.8, 410., -20.29324737, 1040., 0.109, 59.9),
                ('NGC3992', 19.71266735, 10.7, 500., -20.87972697, 1130., 0.106, 58.5)],
               dtype=[('galaxy_name', '<U200'), ('Distance_Mpc', '<f8'), ('Apparent_magnitude_
```

The next step was to plot my values and determine the Hubble constant. Given three points of data, I knew the line could not have a y-intercept greater than 0 otherwise it would make no sense. I once again tried to use `polyfit` to no avail, and adjusted the slope to a reasonable 70. This was to be one of two estimations I made to find the slope.

```
In [11]: plt.plot(galaxies_est['Distance_Mpc'], galaxies_est['Recessional_Velocity'], 'bo')

         m2, b2 = polyfit(galaxies_est['Distance_Mpc'], galaxies_est['Recessional_Velocity'],
         plt.plot(np.arange(20), np.arange(20)*70)

         plt.xlabel('Distance Mpc')
         plt.ylabel('Recessional Velocity km/s')

         #plt.xlim(0,20)
         #plt.ylim(0,2000)
         plt.grid()
         print("Hubble constant estimation (by eye): 70 km/s/Mpc.")
```

```
Hubble constant estimation (by eye): 70 km/s/Mpc.
```




```
In [12]: rise_over_run = galaxies_est['Recessional_Velocity'] / galaxies_est['Distance_Mpc']
         estimation = (rise_over_run[0] + rise_over_run[1] + rise_over_run[2]) / 3
         print('Hubble estimation (by crude math): %.2f' % estimation)
```

```
Hubble estimation (by crude math): 68.75
```

A more crude method to determine this value assumes each point carries equal weight. I then simply took the slope at each point and took an average of the three added together. This value came out to roughly 68.75 $\frac{km}{sMpc}$, which is near the 70 $\frac{km}{sMpc}$ I used by eye.

Given that the current value of the Hubble constant is slightly less than 74 $\frac{km}{sMpc}$, I would argue my value comes within a reasonable range. However, my algorithms to find the best fitting lines were not functioning as well as I would have hoped, so there is a bit of estimation error in those values. When I did not account for inclination angles, my estimates were similar to what Tully and Fisher reported in their paper of 80 $\frac{km}{sMpc}$. What I found from this is that my coefficients for the Tully-Fisher equation also drastically alter my final values for the Hubble Constant.

The major sources of uncertainty would be in the measurements for the HI velocity width, as that was largely a guess on my part. Another slightly less potent source of extinction would be the measurement for the apparent magnitude. The uncertainty in the distance also could drastically affect the measurement for the magnitudes. The inclination angle is also a potentially massive source of error, since the division by the `sin` of this angle will alter the value of the rotational velocity. Altering the values for the width varied my final results greatly.