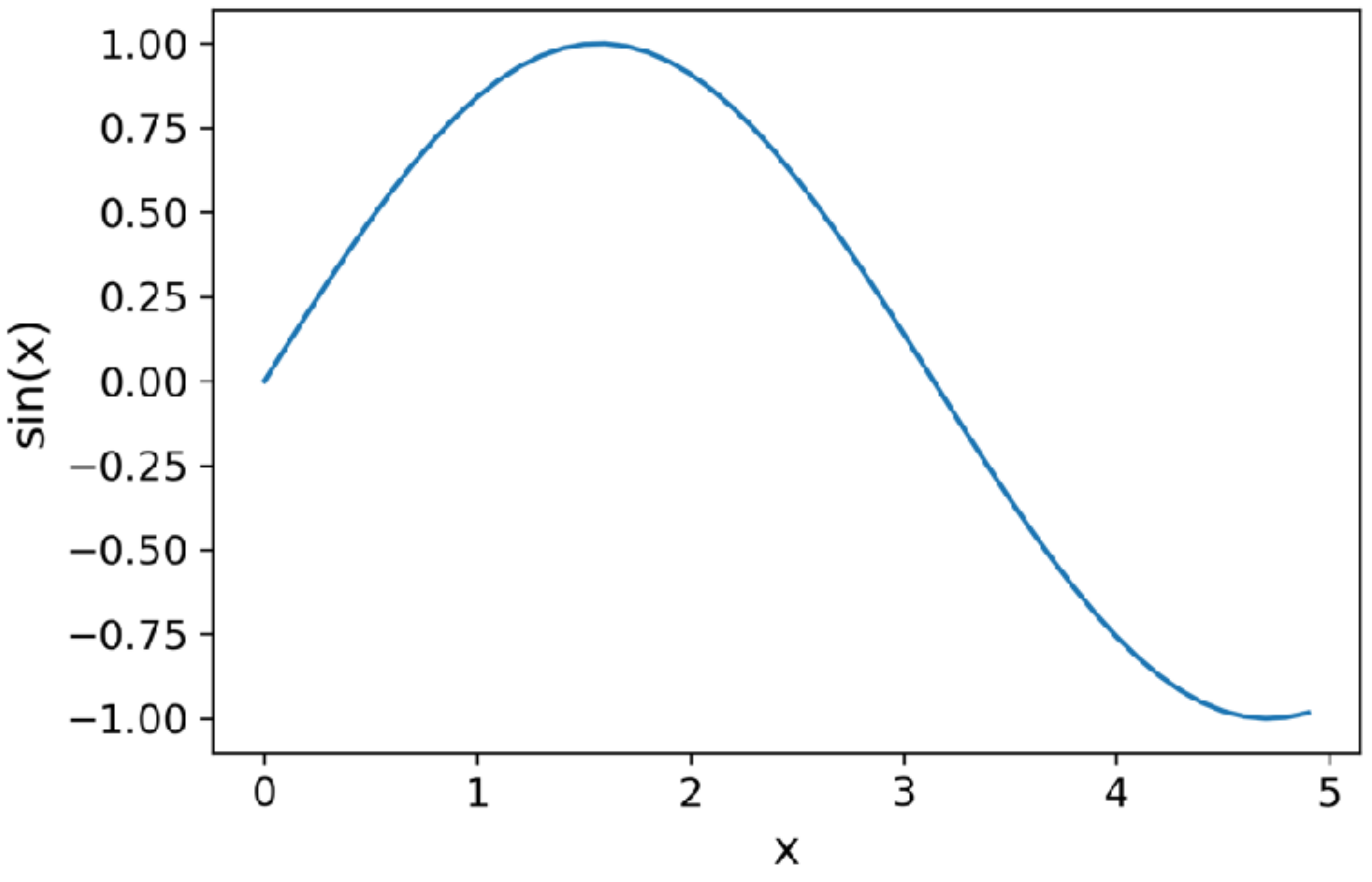
SCIENTIFIC DATA VISUALISATION

with Python

Dany Vohl

Astronomy Data and Computing Services, Swinburne

ADACS ASTRONOMY DATA AND COMPUTING SERVICES



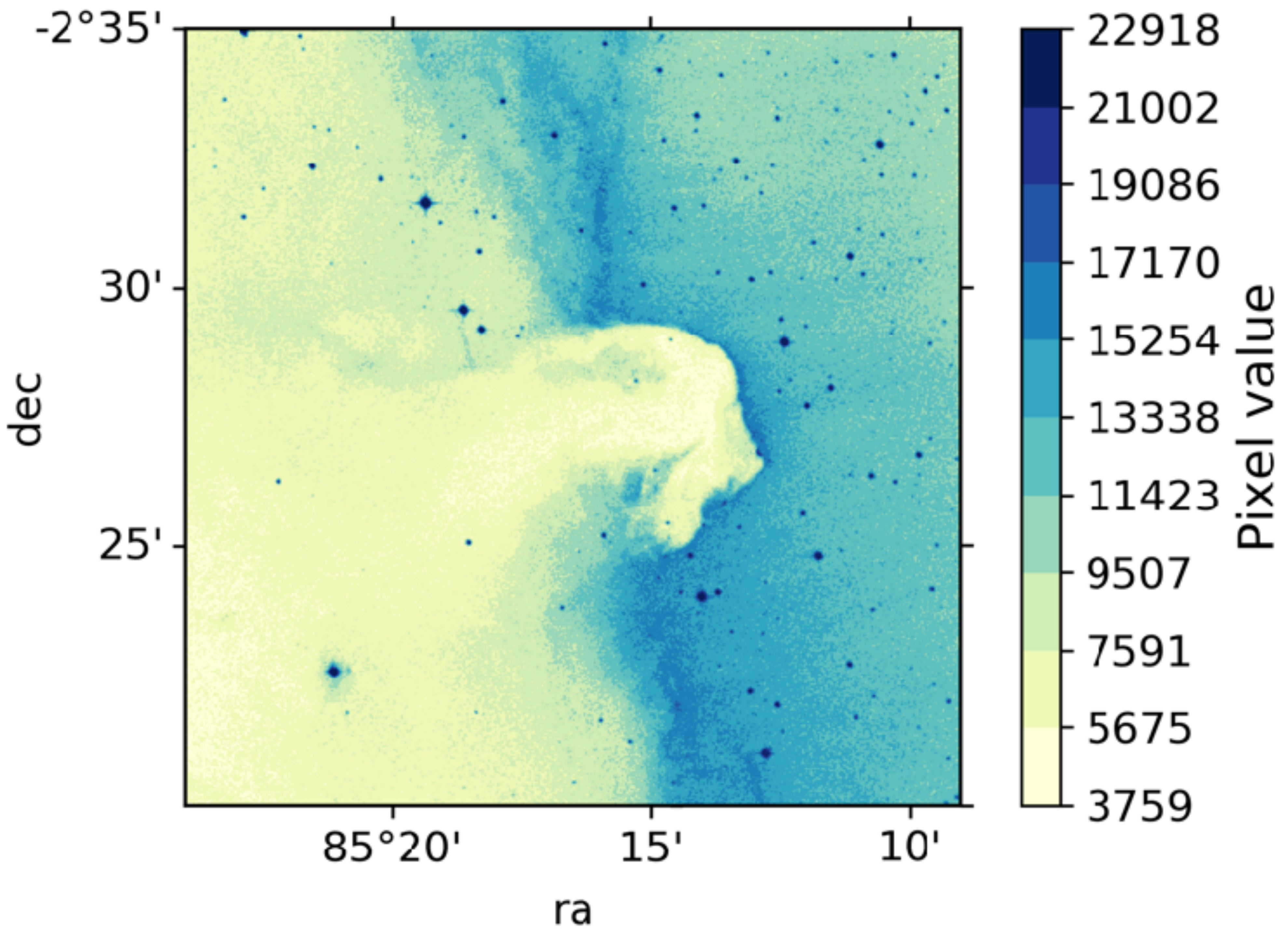
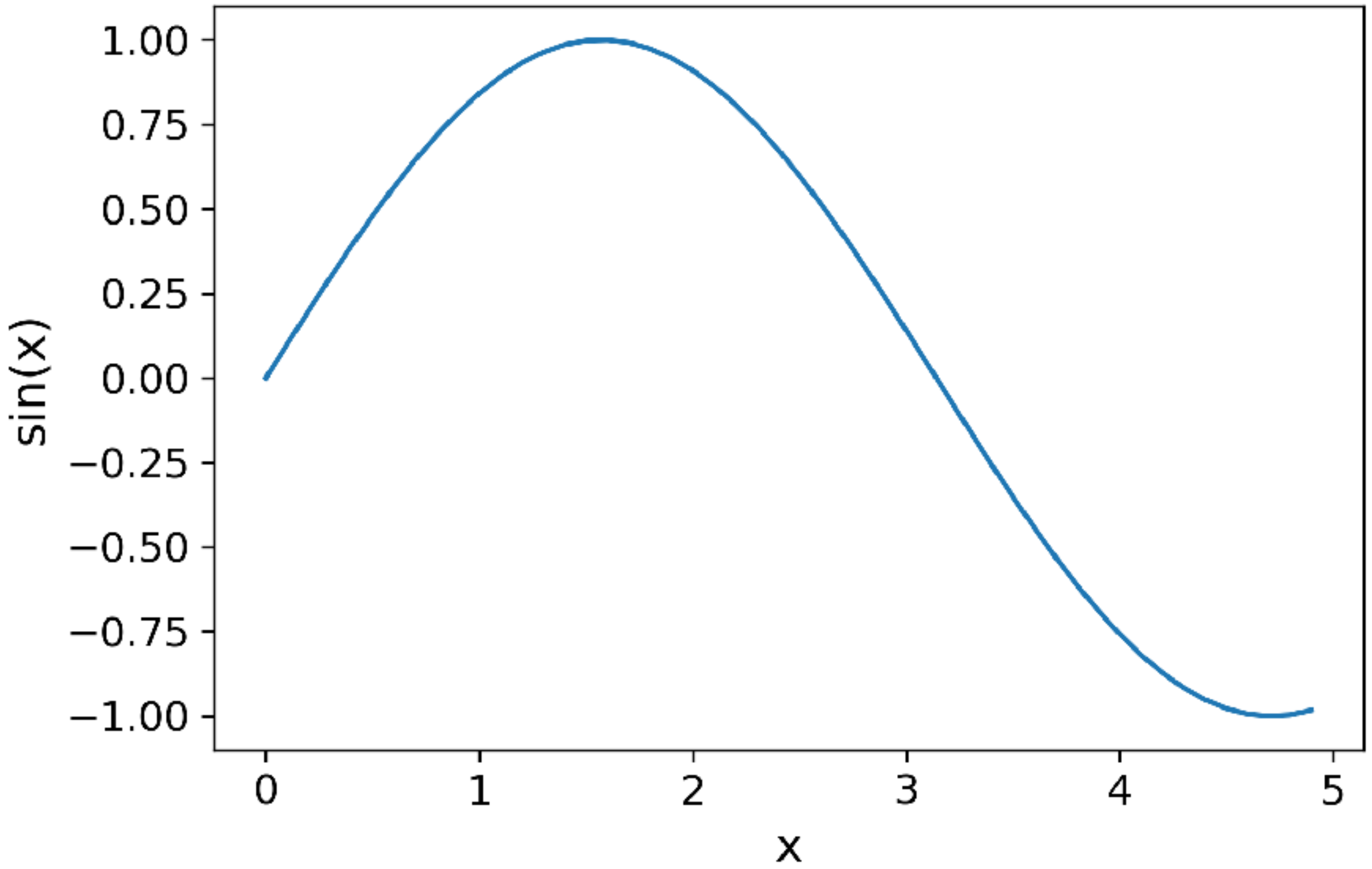
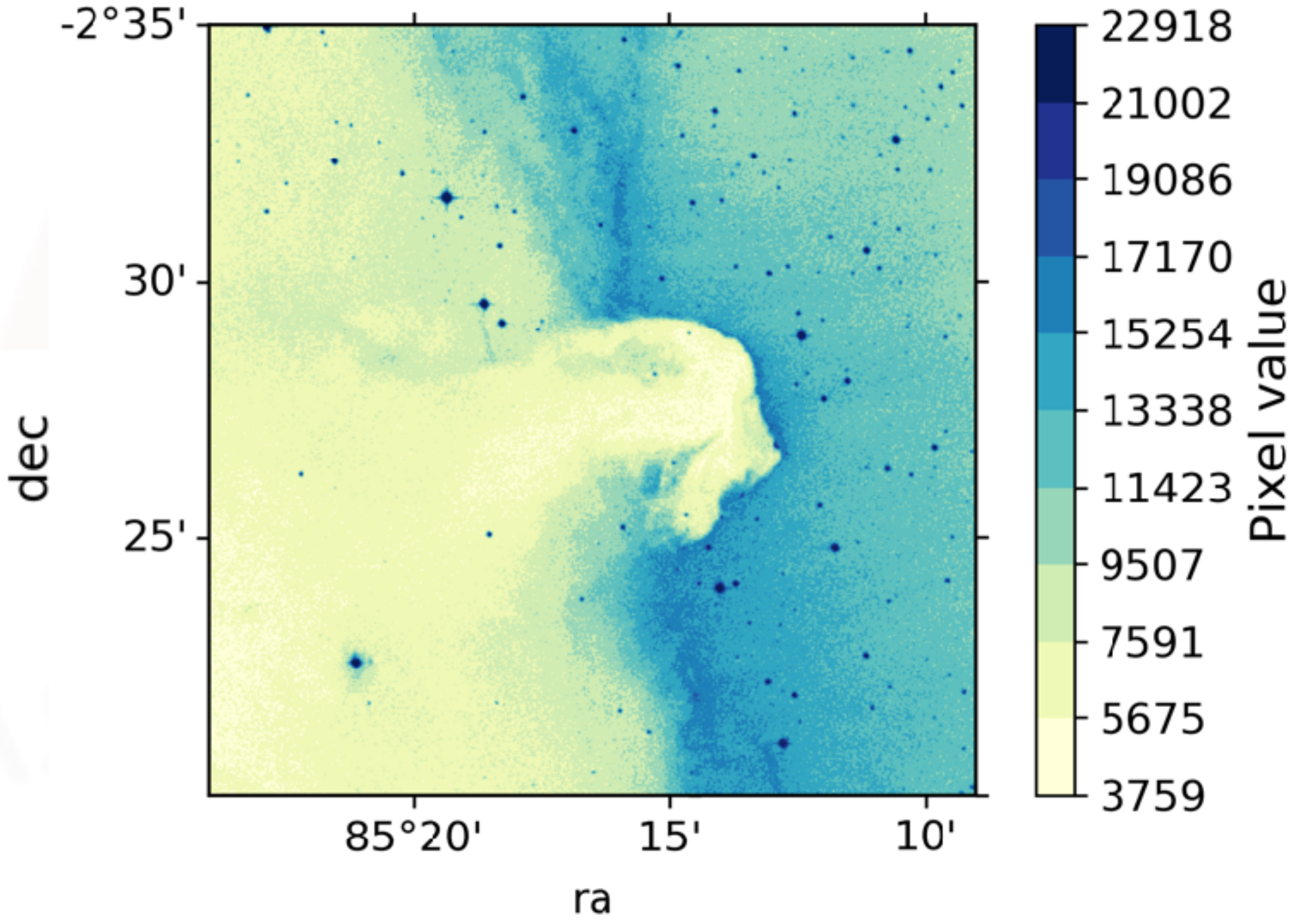


TABLE OF CONTENT

TABLE OF CONTENT

- It is possible to follow along with the code
- All code can be found in a Jupyter Notebook available on ADACS' LMS
  - *http://lms.adacs.org.au/*





INTRODUCTION

Scientific Data Visualisation

FOREWORDS

- Making plots using Python is simple
- Making *nice* and *meaningful* plots requires thinking and work
- **A rule of thumb**
  - ***If your dataset is small*** *(e.g. a few data points)*
    - A table or simply a description is appropriate
  - ***If you have many data points***
    - A visual representation is appropriate
- The type of visualisation depends on:
  - *the type of data and the type of information that you want to carry forward*

VISUALISATION OF QUANTITATIVE INFORMATION

- A good amount of research has been put towards visualising quantitative data
- E.g. see Edward Tufte's book "The Visual Display of Quantitative Information"
  - *Well worth a read!*
- We can inspire ourselves from his work (and others') to lay down a few principles to help us produce good visualisations.

PRINCIPLE 1

The visual elements communicating the data or results, or information about them, must be easily discernible.

PRINCIPLE 1

The visual elements communicating the data or results, or information about them, must be easily discernible.

In short, the **size** and **font** of text should be **readable**,
and **colours** should be **chosen** appropriately **to inform the viewer**.

A good print resolution should also be taken into account.
Python (and matplotlib) includes good mechanisms for this.

PRINCIPLE 1

The visual elements communicating the data or results, or information about them, must be easily discernible.

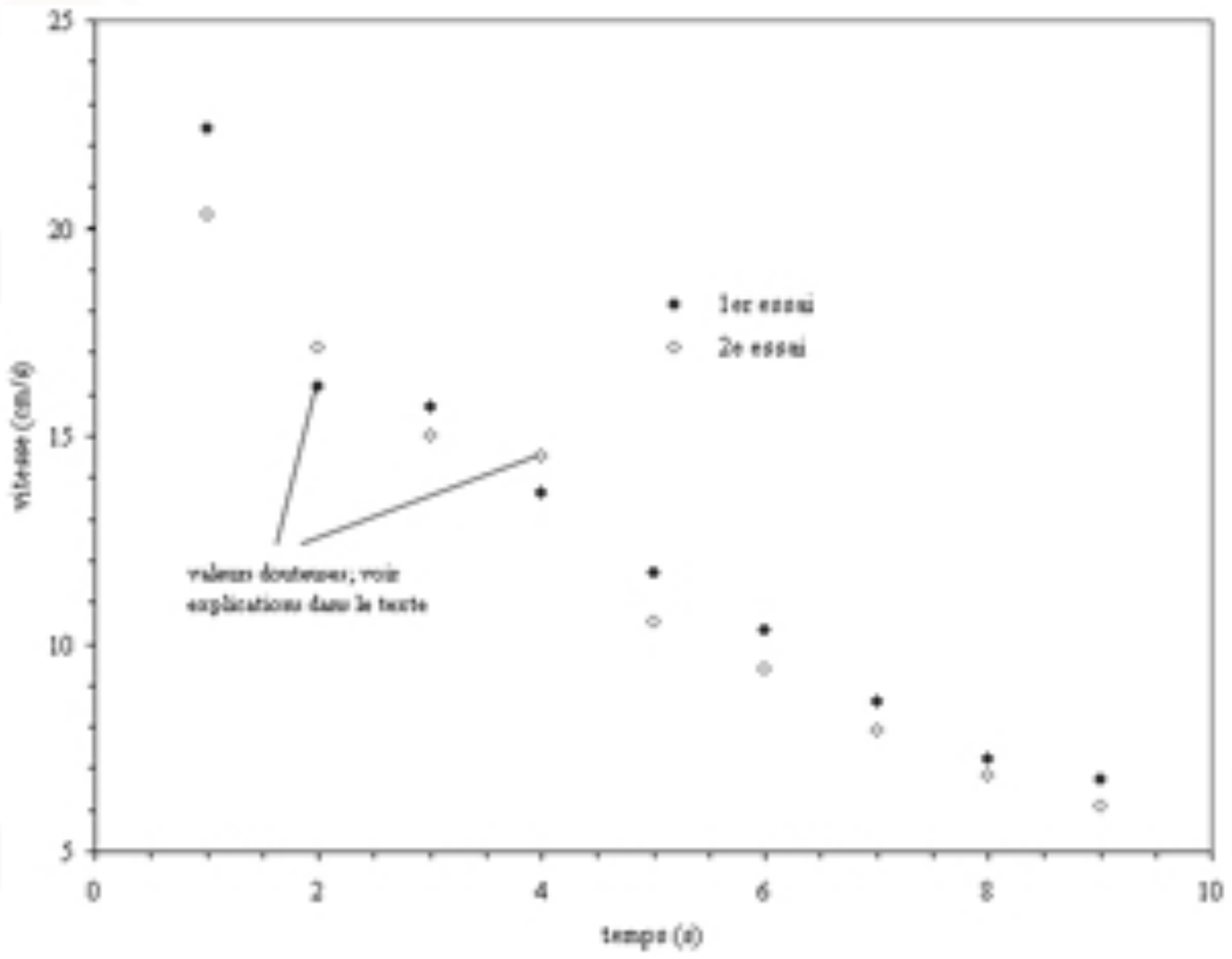


Figure 1. A plot likely produced to be printed elsewhere. When used here, its small text is unreadable.

[Taken from Couture & Francis (2004)].

PRINCIPLE 2

The important aspects of the results must clearly emerge (in fact, those that are deemed important to communicate).

PRINCIPLE 2

The important aspects of the results must clearly emerge (in fact, those that are deemed important to communicate).

For example, in a table,

if you want to show that your results are better than others,
or want to compare results for different parameters,

you should put the best result in **bold** (or those requiring attention), leaving the other values as is.

This will quickly highlight the results of interest.

PRINCIPLE 2

The important aspects of the results must clearly emerge (in fact, those that are deemed important to communicate).

Table 2
Parameters in Part 1 of the JPEG2000 Standard, ordered as encountered in the encoder. The two parameters we investigated are highlighted.

	Parameter
1	Reconstructed image bit depth
2	Tile size
3	Color space
4	Reversible or irreversible transform
5	**Number of wavelet transform levels**
6	Precinct size
7	Code-block size
8	**Coefficient quantization step size**
9	Perceptual weights
10	Block coding parameters: (a) Magnitude refinement coding method (b) MQ code termination method
11	Progression order
12	Number of quality layers
13	Region of interest coding method

Figure 2. The parameters of interest to the discussion are highlighted [taken from Vohl, Fluke & Vernardos (2015)].

PRINCIPLE 2

The important aspects of the results must clearly emerge (in fact, those that are deemed important to communicate).

For graphics,

care should be taken to make sure elements other than those representing data (labels, gridlines, legends, etc.) do not take on an unreasonable importance, in terms of size, number or mere visual presence.

This is sometimes referred to as the **data to ink ratio**.

For example, there are cases where little ink (or colour if not printed) is used for the data, and a lot of ink is used for the axis, the reference grid, the labels, the ancillary elements. **In general, it is better to use the *ink* for the data.**

PRINCIPLE 2

The important aspects of the results must clearly emerge (in fact, those that are deemed important to communicate).

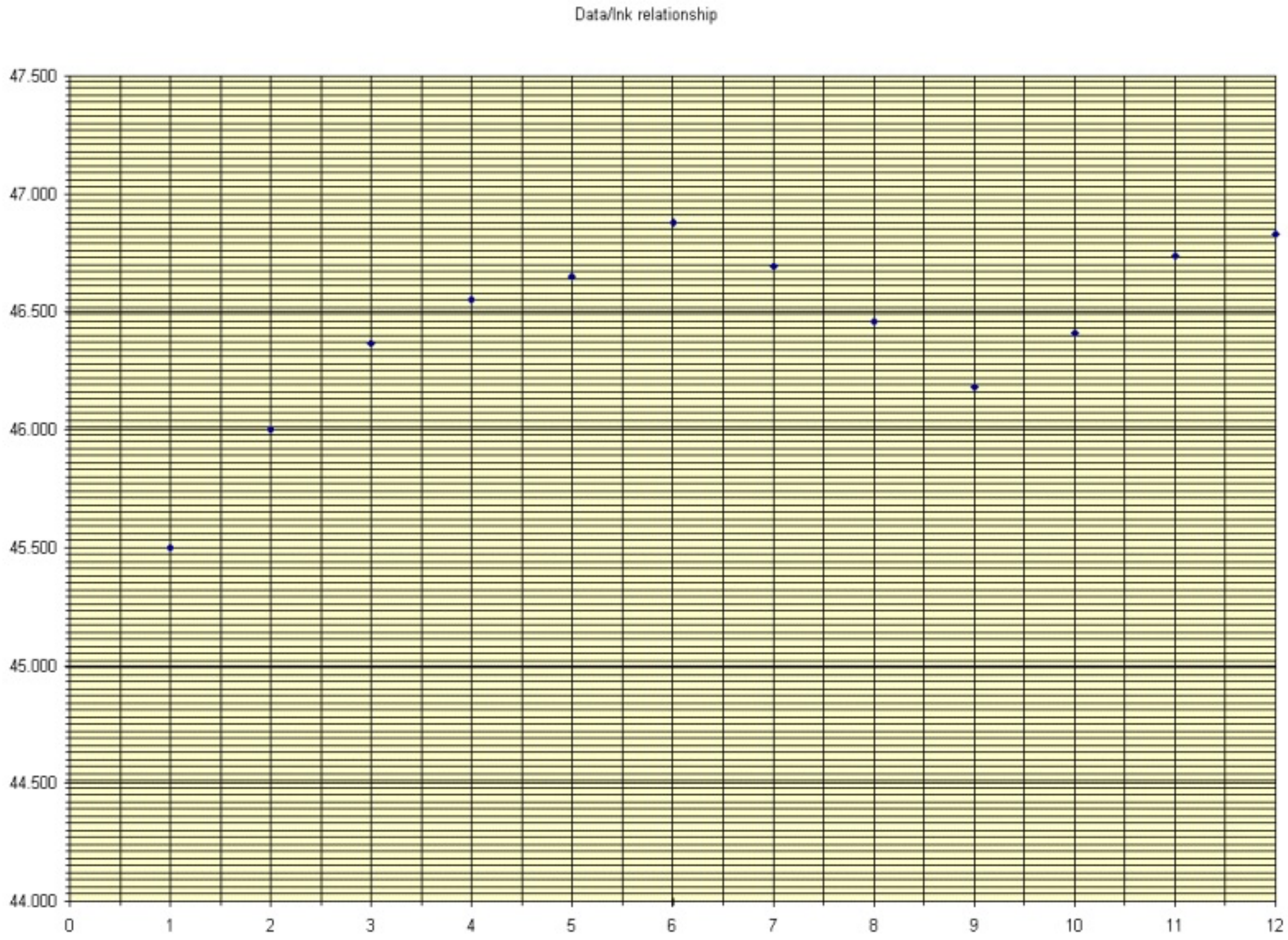


Figure 3. Data to ink ratio: the amount of ink used for the lines and other features of the graph hides the actual data points [taken from Dürsteler (2002)].

PRINCIPLE 3

Simplicity is best.

PRINCIPLE 3

Simplicity is best.

Many software and libraries permit the creation of figures with overly complicated layout.

For example, some software offer the possibility to create 3D histograms, where the depth axis is purely esthetic.

It is advisable to avoid such overly complex practices.
It is best to keep it to the point.

PRINCIPLE 3

Simplicity is best.

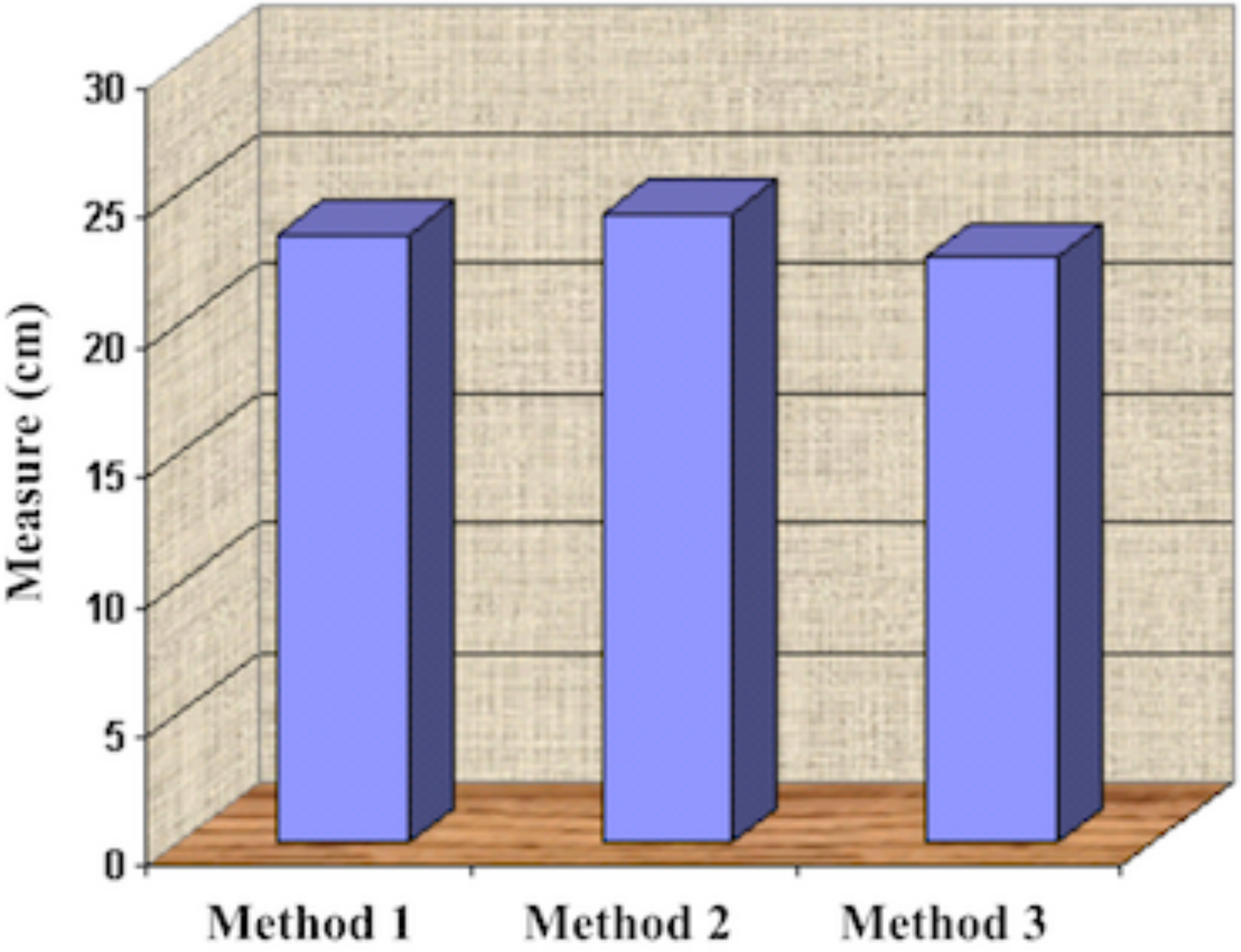


Figure 4. This histogram should be in represented in 2D. The use of 3D does not provide any new insights about the data.

PRINCIPLE 4

Seek the balance between the presentation of the numerical values, and that of the phenomenon or interpretation that these values illustrate or suggest.

PRINCIPLE 4

Seek the balance between the presentation of the numerical values, and that of the phenomenon or interpretation that these values illustrate or suggest.

The most appropriate representation of results is not necessarily the easiest to precisely interpret the data.

Instead, it is **often** the one **that suggests or supports the proposed interpretation**, which highlights the phenomenon in question and shows to what extent the displayed values participate in this phenomenon.

PRINCIPLE 4

Seek the balance between the presentation of the numerical values, and that of the phenomenon or interpretation that these values illustrate or suggest.

It is often the case where experimental data points (**measurements**) are plotted in conjunction with a model (**fit**).

Visually, the eyes will tend to follow the line, which highlights the phenomenon (model), and will let the viewer interpret how the data agrees with the model.

Other statistical measure can provide extra informations about this, too (e.g. error bars).

PRINCIPLE 4

Seek the balance between the presentation of the numerical values, and that of the phenomenon or interpretation that these values illustrate or suggest.

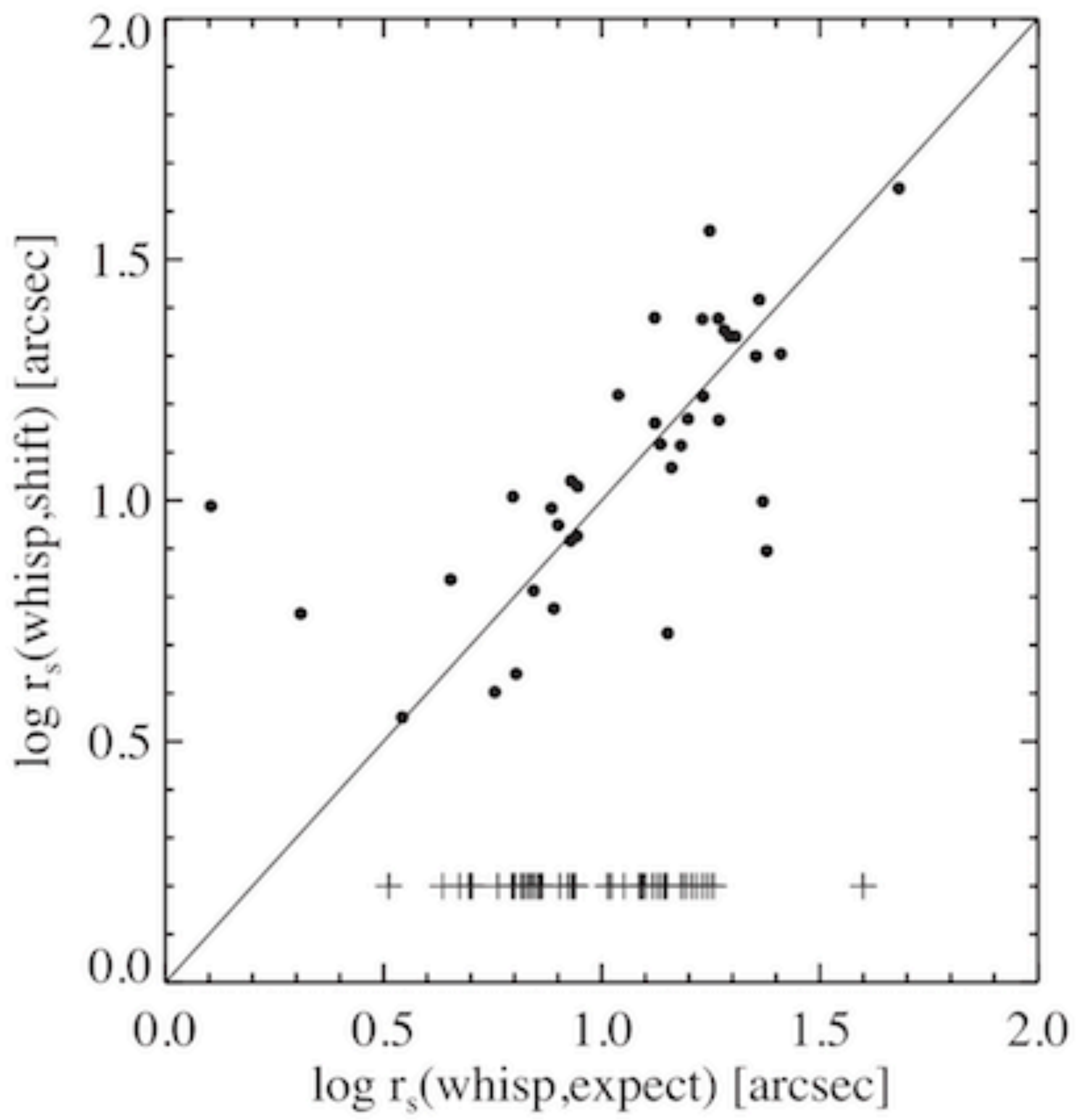
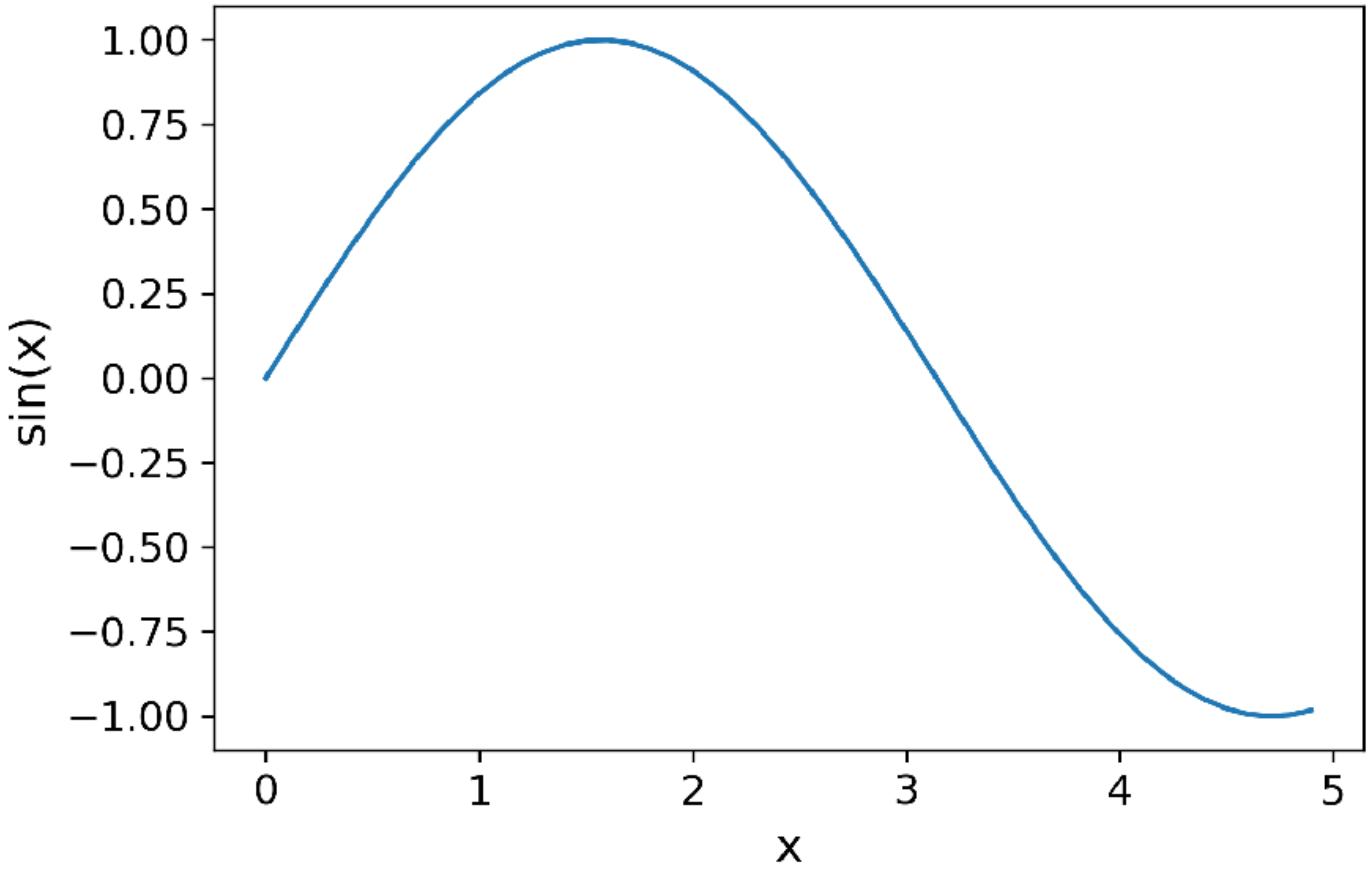
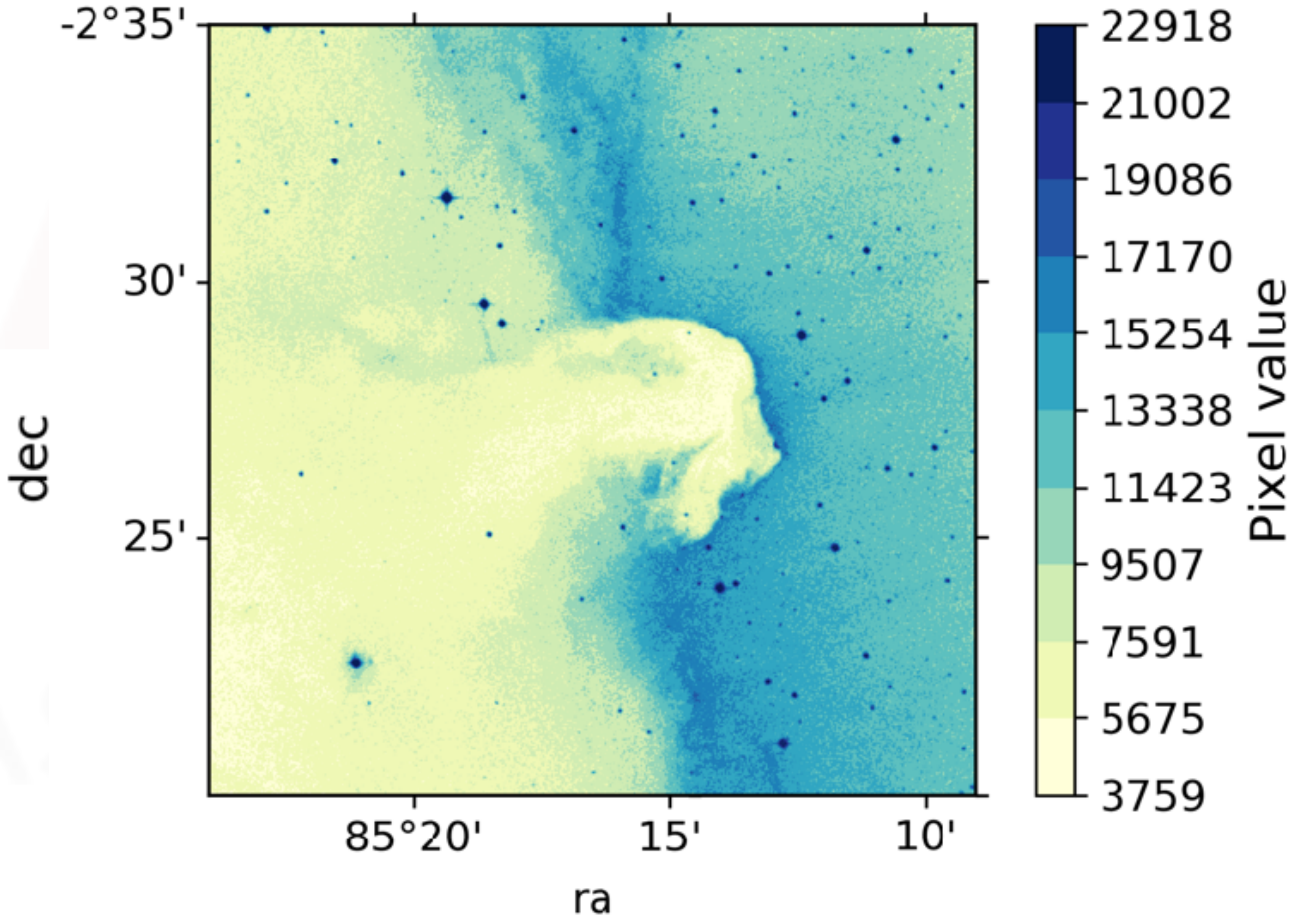


Figure 3. WHISP galaxies are shifted and convolved with the WSRT beam to have similar appearance to the Bluedisk galaxies. r_s(shift) are the scale-length measured from the shifted and convolved images, and r_s(expected) are the true scalelength expected at the redshifts of the Bluedisk galaxies. The crosses show the sizes of the Bluedisk galaxies.

Figure 5. The figure highlights the relation between the model and the observation. To simplify the visual, the text accompanying the figure explains what the symbols represent (avoiding to over-crowd the figure).

[taken from Wang et al. (2015)].





CHOOSING A VISUALISATION

with examples using Python

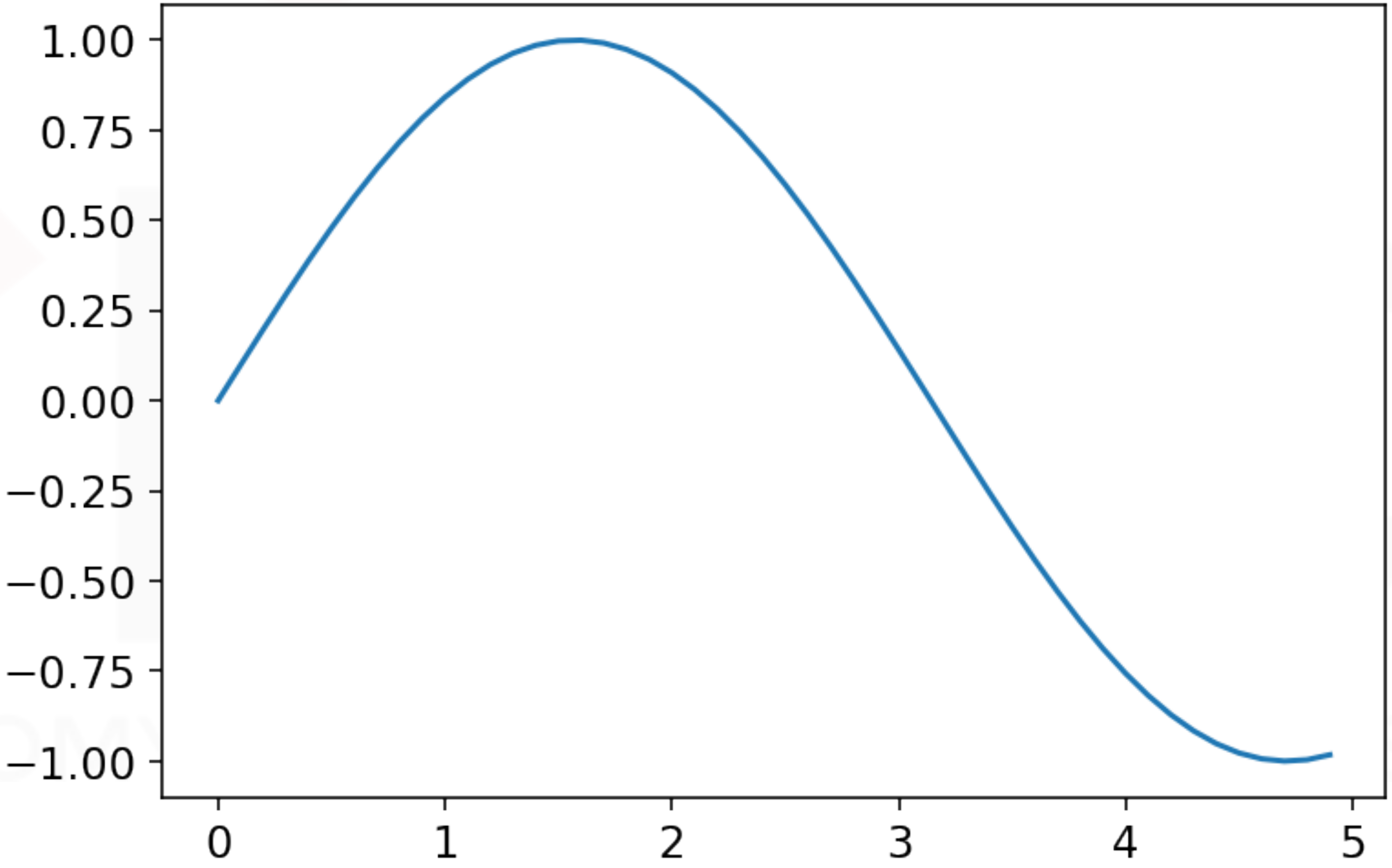
EXAMPLE 1: THE BASICS

```python
# Import basic packages to plot and crunch numbers

import numpy as np

from matplotlib import pyplot as plt
# Set font size for labels and axes ticks
# (http://matplotlib.org/users/customizing.html)
from matplotlib import rc
rc('font', size=12)
rc('axes', titlesize=14)
rc('axes', labelsize=14)
```

EXAMPLE 1: THE BASICS

```
# Example 1.1: Simple plot with generated data

x = np.arange(0, 5, 0.1)
y = np.sin(x)

print ('x =', x)
print ()
print ('y =', y)

plt.plot(x, y)
```

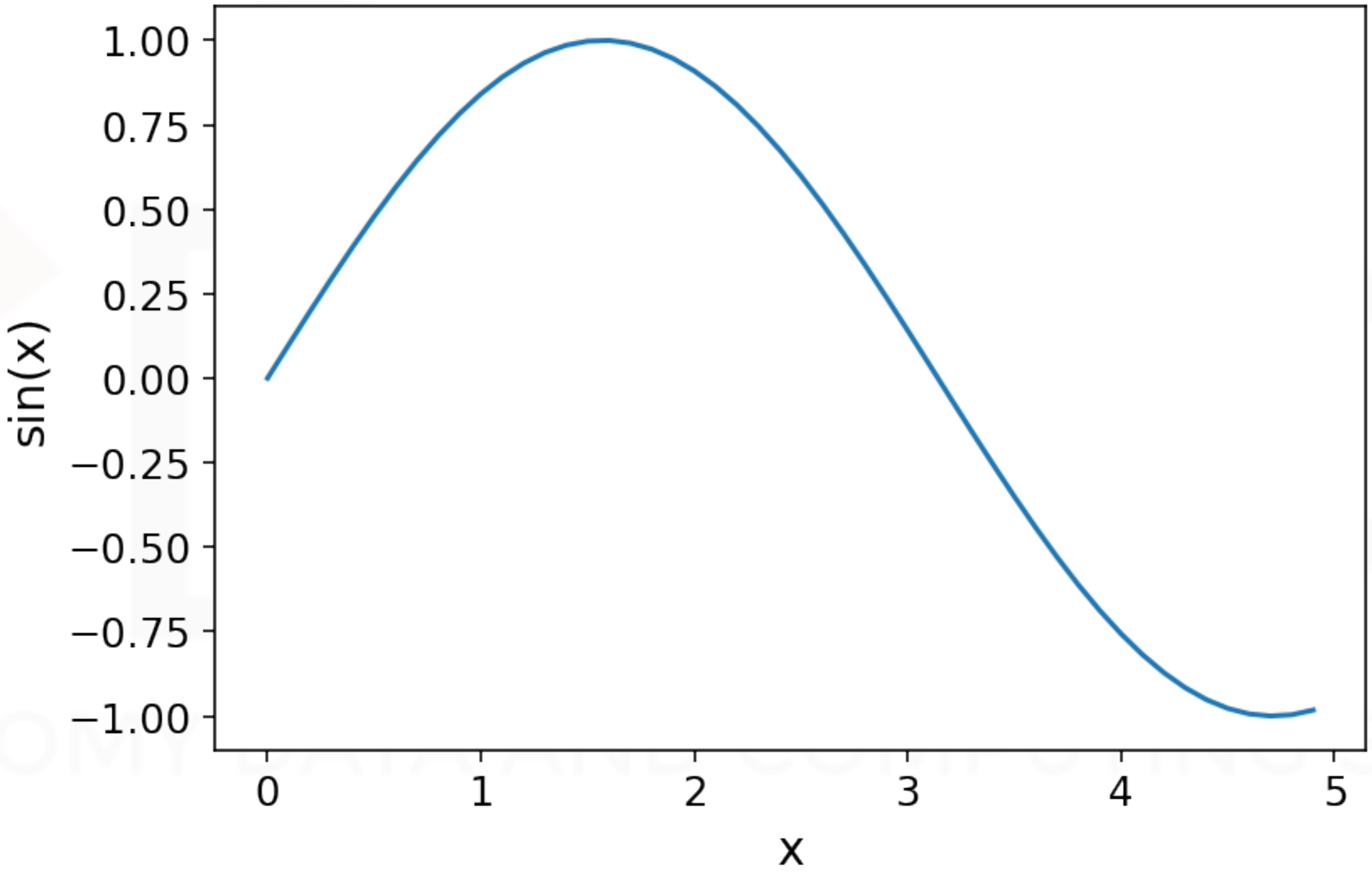
EXAMPLE 1: THE BASICS



EXAMPLE 1: THE BASICS

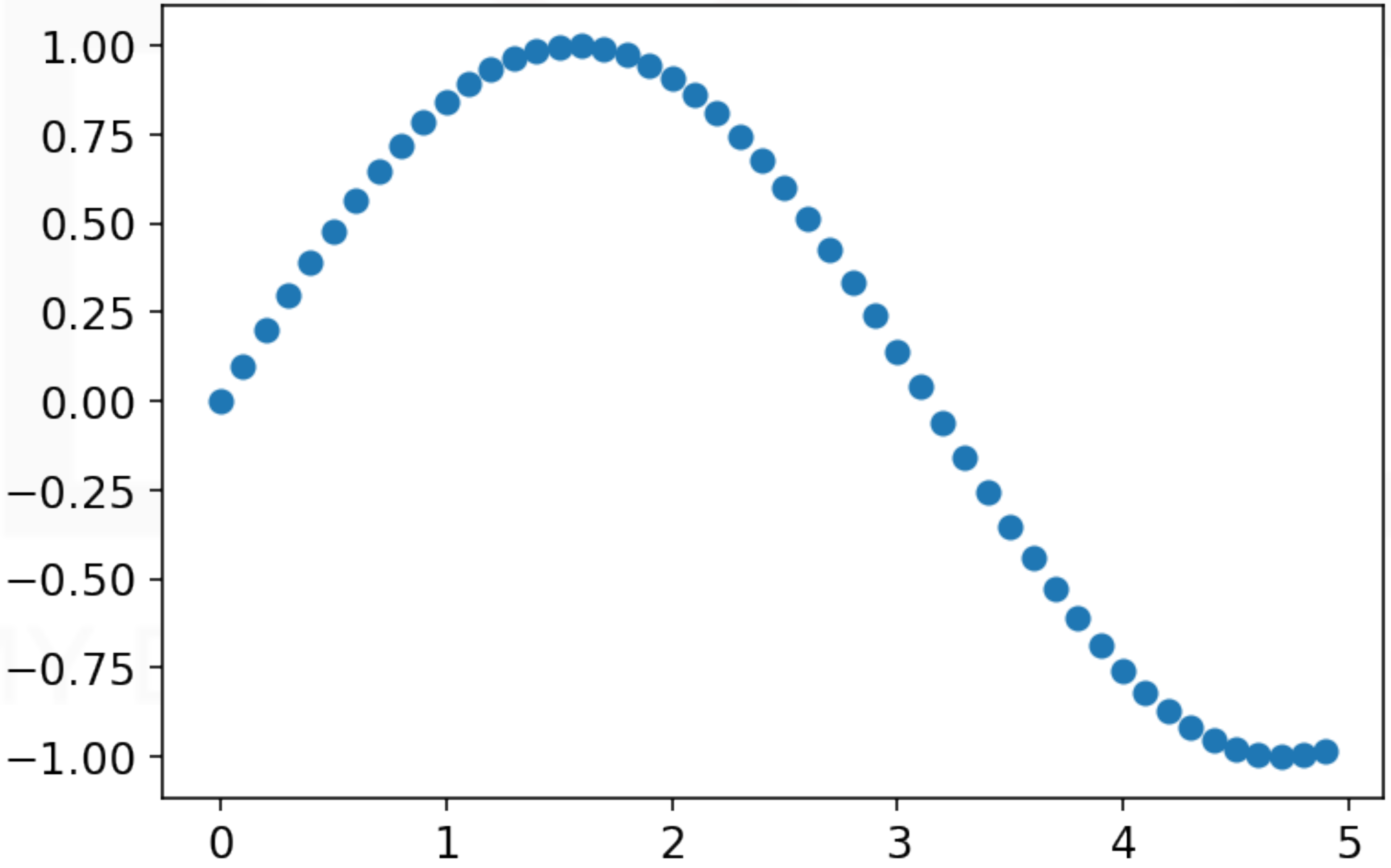
```
# Example 1.2: Axis labels and save the figure to disk

plt.plot(x, y)

plt.xlabel('x')
plt.ylabel('sin(x)')

plt.tight_layout()
plt.savefig('basic.png', dpi=300)
```

EXAMPLE 1: THE BASICS



EXAMPLE 1: THE BASICS

```
# Example 1.3: Plotting the same values with scatter
plt.scatter(x,y)
```



HOW TO CHOOSE THE RIGHT VISUALISATION?

- The type of visualisation depends on the type of variables.
- A variable can be **qualitative** or **quantitative**.
  - *A **quantitative variable** is a variable described by numbers,*
  - *and optionally with a unit and uncertainty.*
  - *It can be **continuous** or **discrete**.*
- We can also describe variables according to the relationship between them.
  - *We distinguish **independent variables** and **dependent variables** (e.g. where the value of one variable is affected by another).*

HOW TO CHOOSE THE RIGHT VISUALISATION?

- The type of visualisation depends on the type of variables.
- A variable can be **qualitative** or **quantitative**.

HOW TO CHOOSE THE RIGHT VISUALISATION?

- The type of visualisation depends on the type of variables.
- A variable can be **qualitative** or **quantitative**.
- **Qualitative variable**
  - *a variable described by characteristics or categories.*
  - *It can be **nominal** or **ordinal**.*

HOW TO CHOOSE THE RIGHT VISUALISATION?

- The type of visualisation depends on the type of variables.
- A variable can be **qualitative** or **quantitative**.
- **Quantitative variable**
  - *a variable described by numbers,*
  - *and optionally with a unit and uncertainty.*
  - *It can be* ***continuous*** *or* ***discrete****.*

HOW TO CHOOSE THE RIGHT VISUALISATION?

- The type of visualisation depends on the type of variables.
- A variable can be **qualitative** or **quantitative**.
- We can also describe variables according to the relationship between them.
  - *We distinguish **independent variables** and **dependent variables** (e.g. where the value of one variable is affected by another).*

HOW TO CHOOSE THE RIGHT VISUALISATION?

- The type of variable at hand will determine the kind of visual we use:
  - *Diagrams*
  - *Plots*
  - *Figures*

HOW TO CHOOSE THE RIGHT VISUALISATION?

- The type of variable at hand will determine the kind of visual we use:
  - ***Diagrams***

> Diagrams are generally used with independent discrete variables that are either qualitative or quantitative.
> Note that a table can also be used in this case, when the number of values to present is small.

  - *Diagrams include **barchart**, **histograms** (a special case of barcharts), and **boxplots**.*

HOW TO CHOOSE THE RIGHT VISUALISATION?

- The type of variable at hand will determine the kind of visual we use:
  - ***Plots***

    > Plots are used when at least one independent variable is quantitative (continuous or discrete). They are often used to show a trend in the data, to compare data with models, or to determine if there is a correlation between two variables. Another common use is time series.

    > To get back to the original question: continuous variables are generally represented as a line, whereas discrete variables are represented by symbols (circles, cross, …).

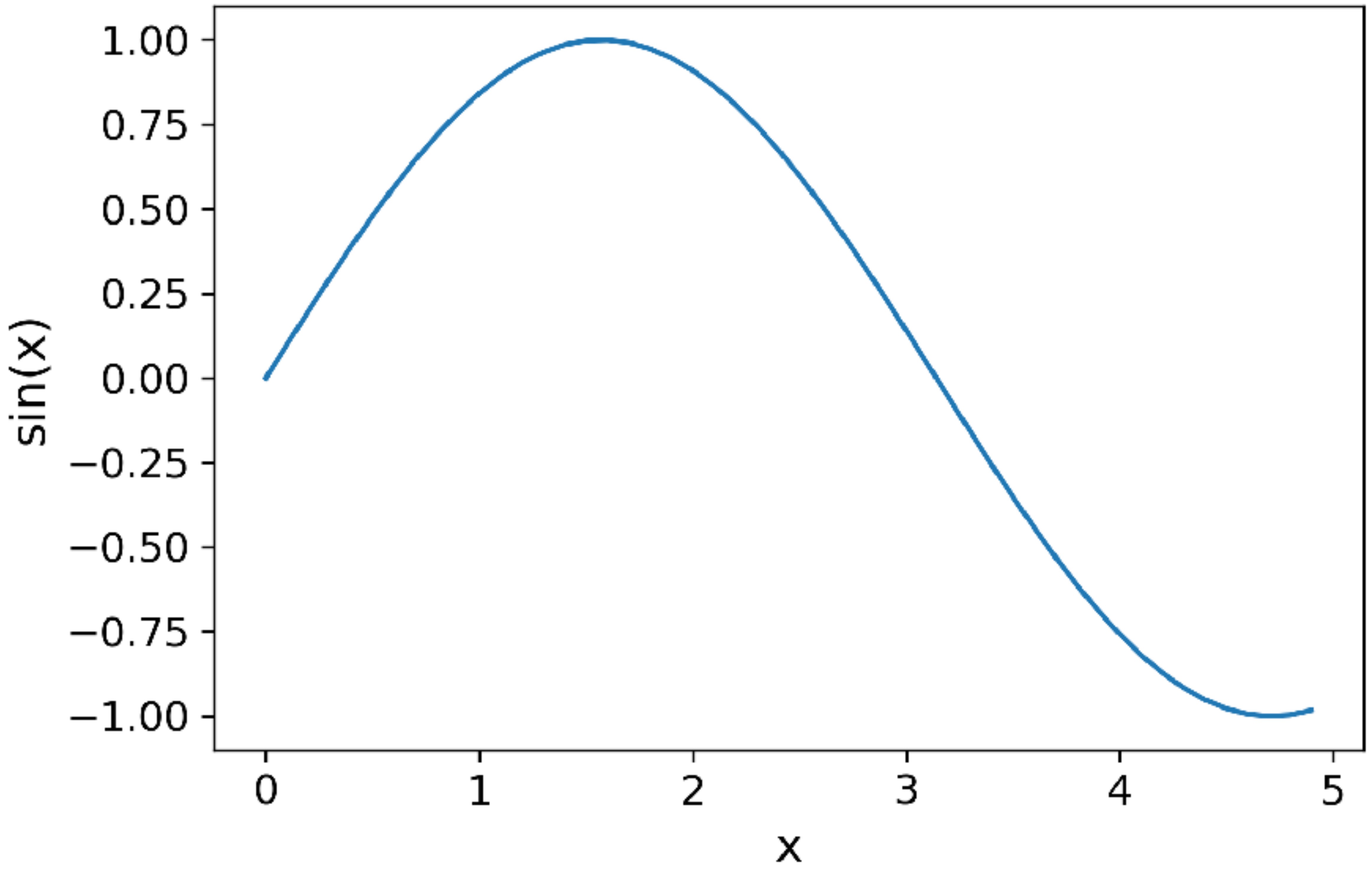
  - *Plots include simple* ***plots****,* ***trend plots****,* ***scatter plots****, etc.*

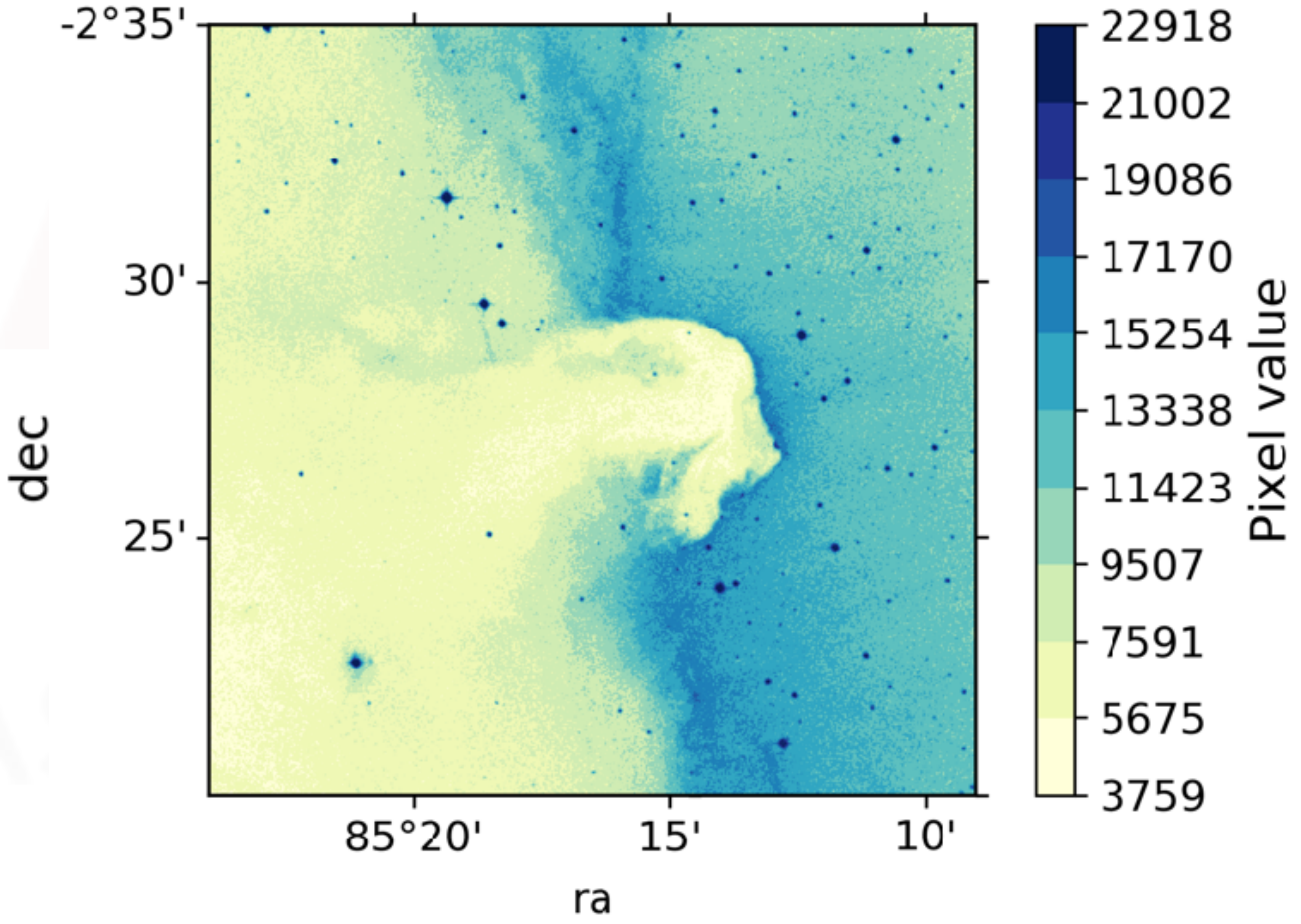
HOW TO CHOOSE THE RIGHT VISUALISATION?

- The type of variable at hand will determine the kind of visual we use:
  - ***Figures***

    > A figure may be used to present data or results when they relate to the appearance, characteristics, condition, or evolution of an object, phenomenon, or process for which a simple description would not suffice. They can also serve to explain all mathematical symbols representing quantities (distance, mass, ...) associated with the situation or the experimental set-up. Finally, they can illustrate the logical or functional links between various elements of a situation or a montage.

  - *Figures include **photos**, **drawing**, **schematic**, etc.*





1. FIGURES

1.1 Image with Matplotlib

1.1 IMAGE

We can use the routine **imshow** from matplotlib to display image data (2D matrix)

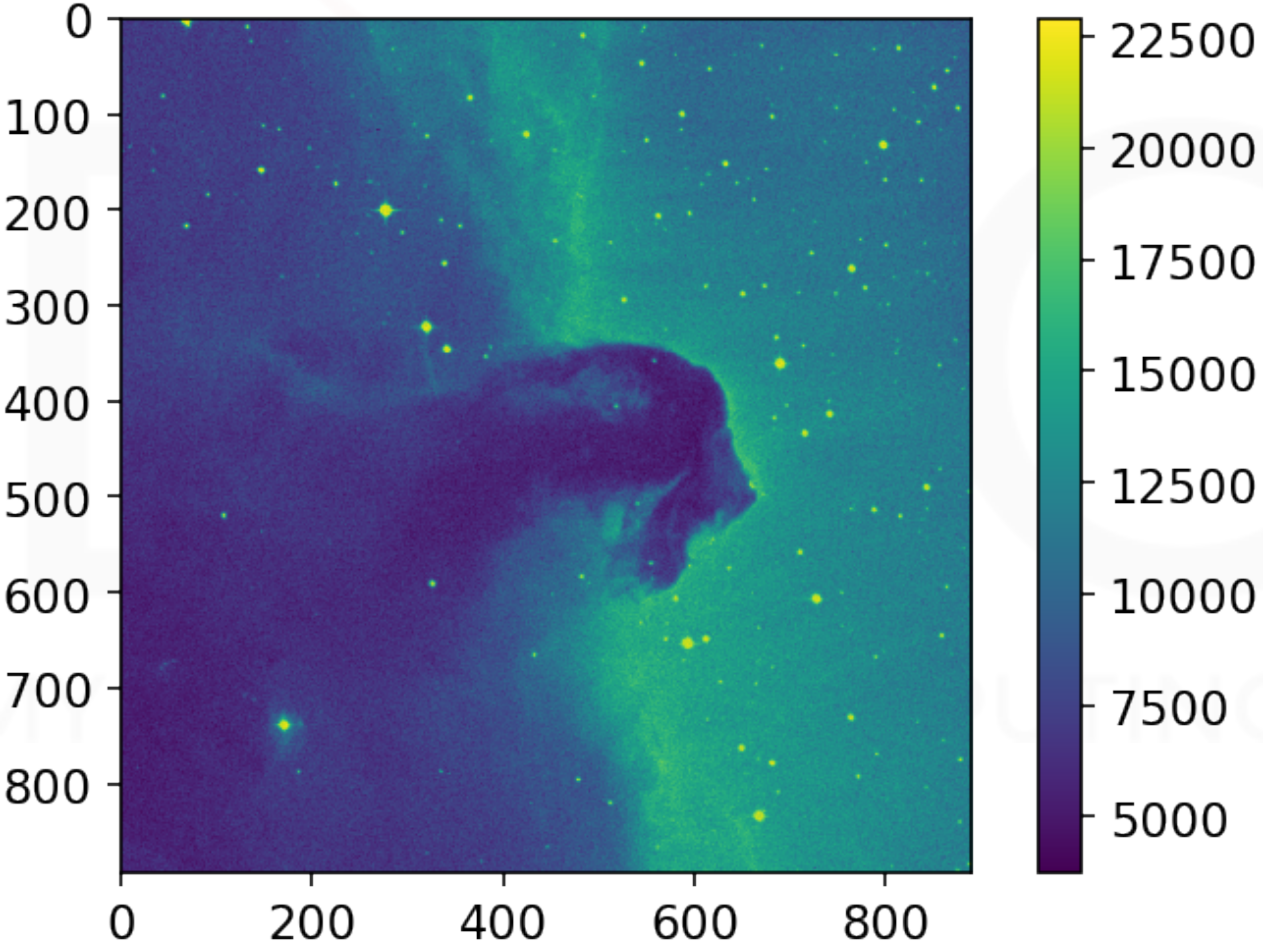
1.1 IMAGE

We can use the routine **imshow** from matplotlib to display image data (2D array)

```
from astropy.io import fits
from astropy.utils.data import download_file

image_file = download_file('http://data.astropy.org/tutorials/FITS-images/HorseHead.fits', cache=True)

image_data = fits.getdata(image_file)
print ("min:", image_data.min(),
       "max:", image_data.max())

plt.imshow(image_data)
plt.colorbar()
plt.show()
```

1.1 IMAGE

We can use the routine **imshow** from matplotlib to display image data (2D matrix)



1.1 IMAGE

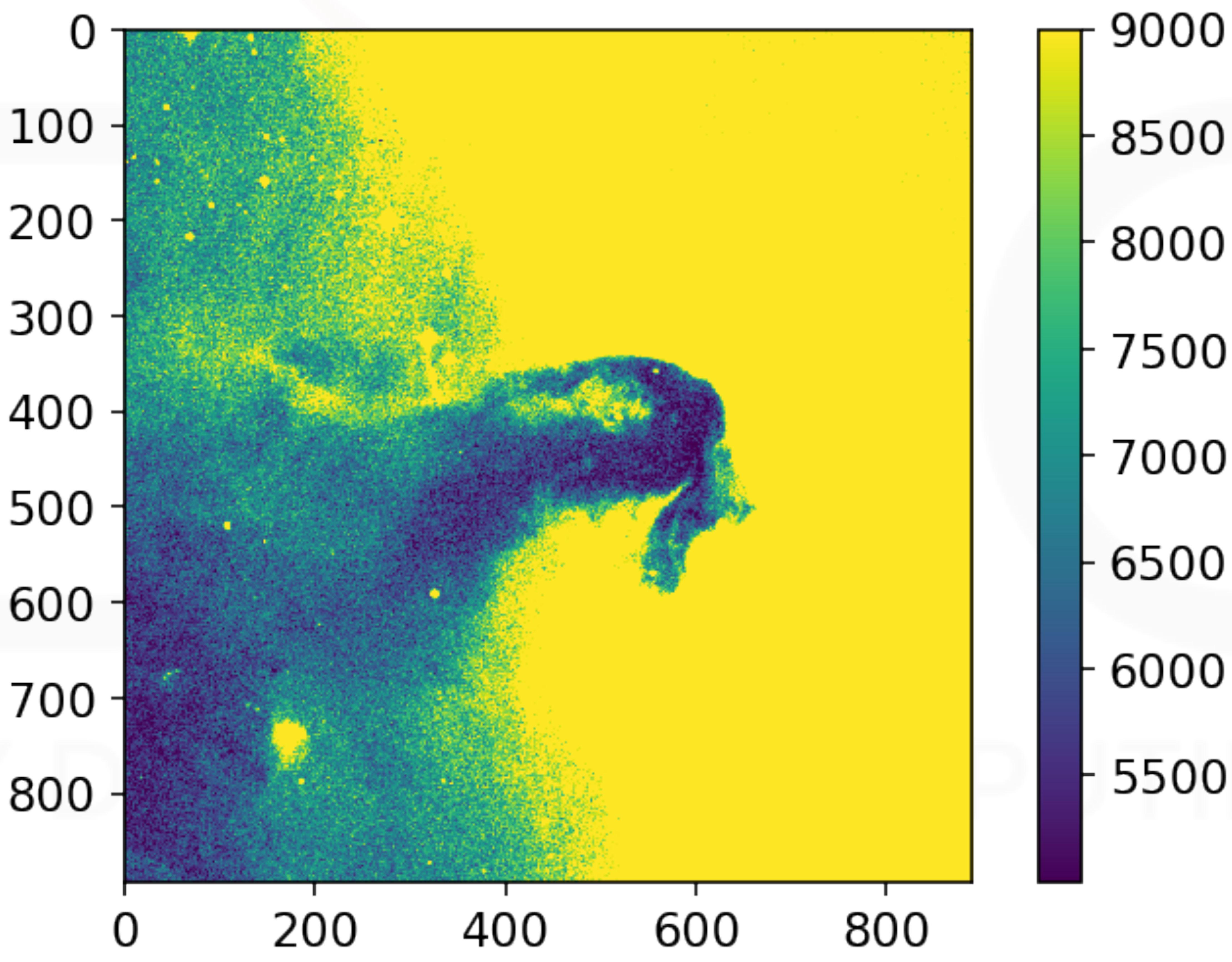
Exercise 1.1 Hide all values smaller than 5000 and greater than 9000.

Exercise 1.1 Hide all values smaller than 5000 and greater than 9000.

```
plt.imshow(image_data, vmin=5001, vmax=9000)
plt.colorbar()
plt.show()
```

1.1 IMAGE

Exercise 1.1 Hide all values smaller than 5000 and greater than 9000.



1.1 IMAGE

Exercise 1.2 Set axes ticks and labels from the fits file header with World Coordinate System.

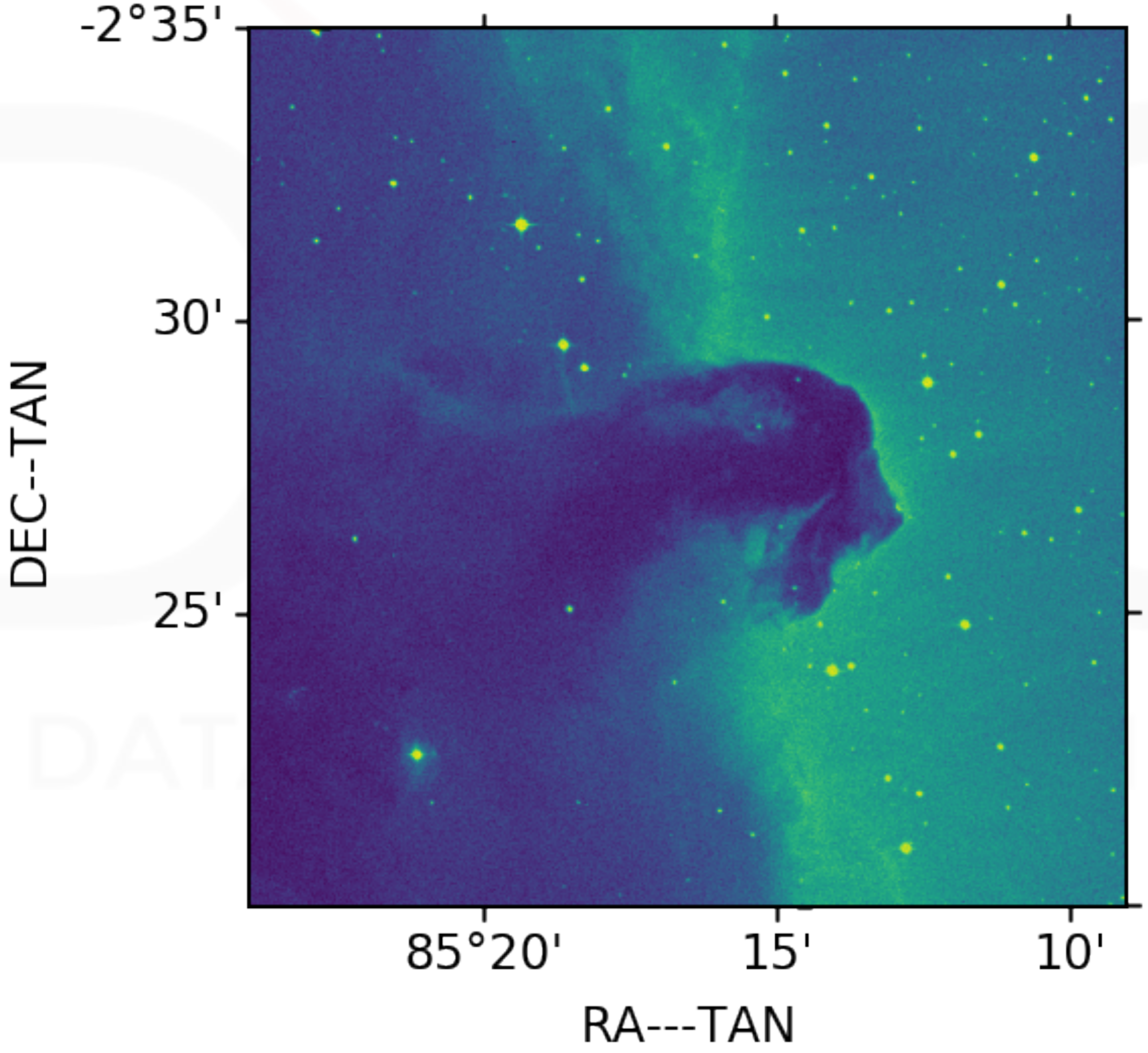
1.1 IMAGE

Exercise 1.2 Set axes ticks and labels from the fits file header with World Coordinate System.

```
from astropy.wcs import WCS

image = fits.open(image_file)
header = image[0].header
data = image[0].data
wcs = WCS(header)

fig = plt.figure()
fig.add_subplot(111, projection=wcs)
plt.imshow(data)
plt.xlabel(header['CTYPE1'])
plt.ylabel(header['CTYPE2'])
plt.show()
```

1.1 IMAGE

Exercise 1.2 Set axes ticks and labels from the fits file header with World Coordinate System.



1.1 IMAGE

Exercise 1.3 Set a discrete colourbar (e.g. 0-5000: blue, 5000-10000: purple,…)

1.1 IMAGE

Exercise 1.3 Set a discrete colourbar (e.g. 0-5000: blue, 5000-10000: purple,…)

```
nbins = 10

# Initialise the plot
fig = plt.figure()
fig.add_subplot(111, projection=wcs)

# define the colormap
cmap = plt.cm.YlGnBu

# define the bins and normalize
min = np.floor(np.min(data))
max = np.floor(np.max(data))
bounds = np.arange(min, max+1, (max-min)/nbins)
norm = colors.BoundaryNorm(bounds, cmap.N)
(…)
```

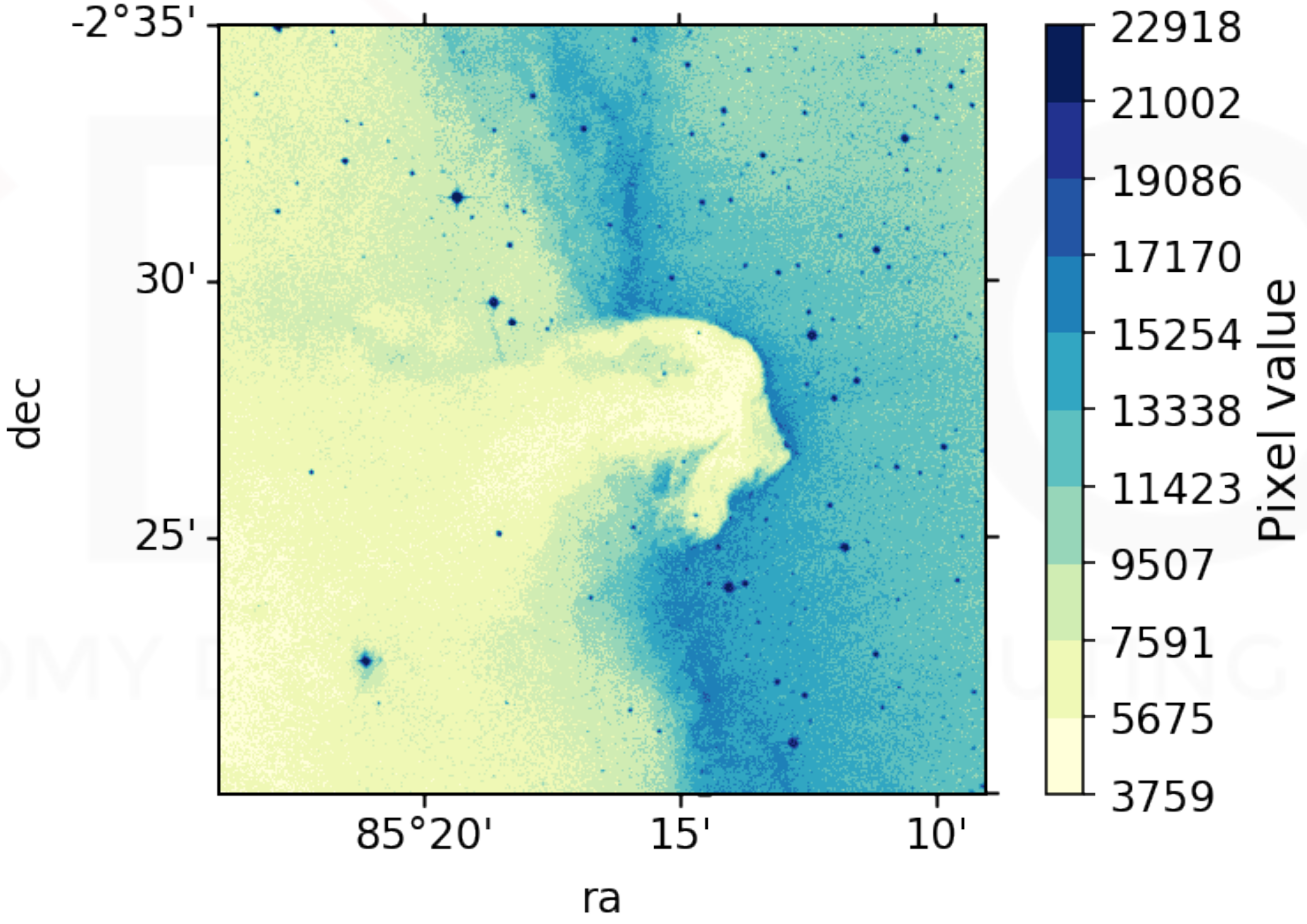
1.1 IMAGE

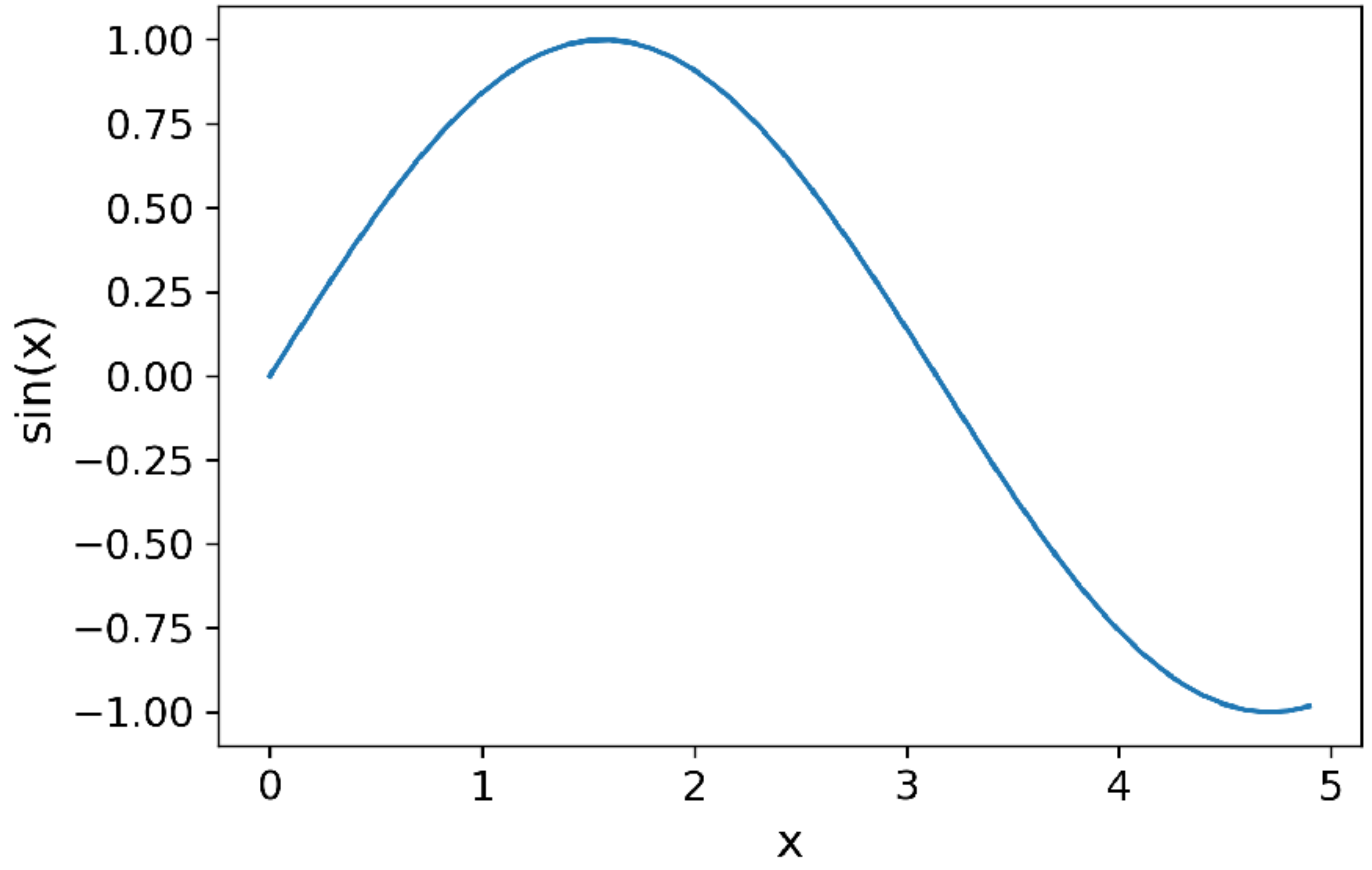
Exercise 1.3 Set a discrete colourbar (e.g. 0-5000: blue, 5000-10000: purple,…)

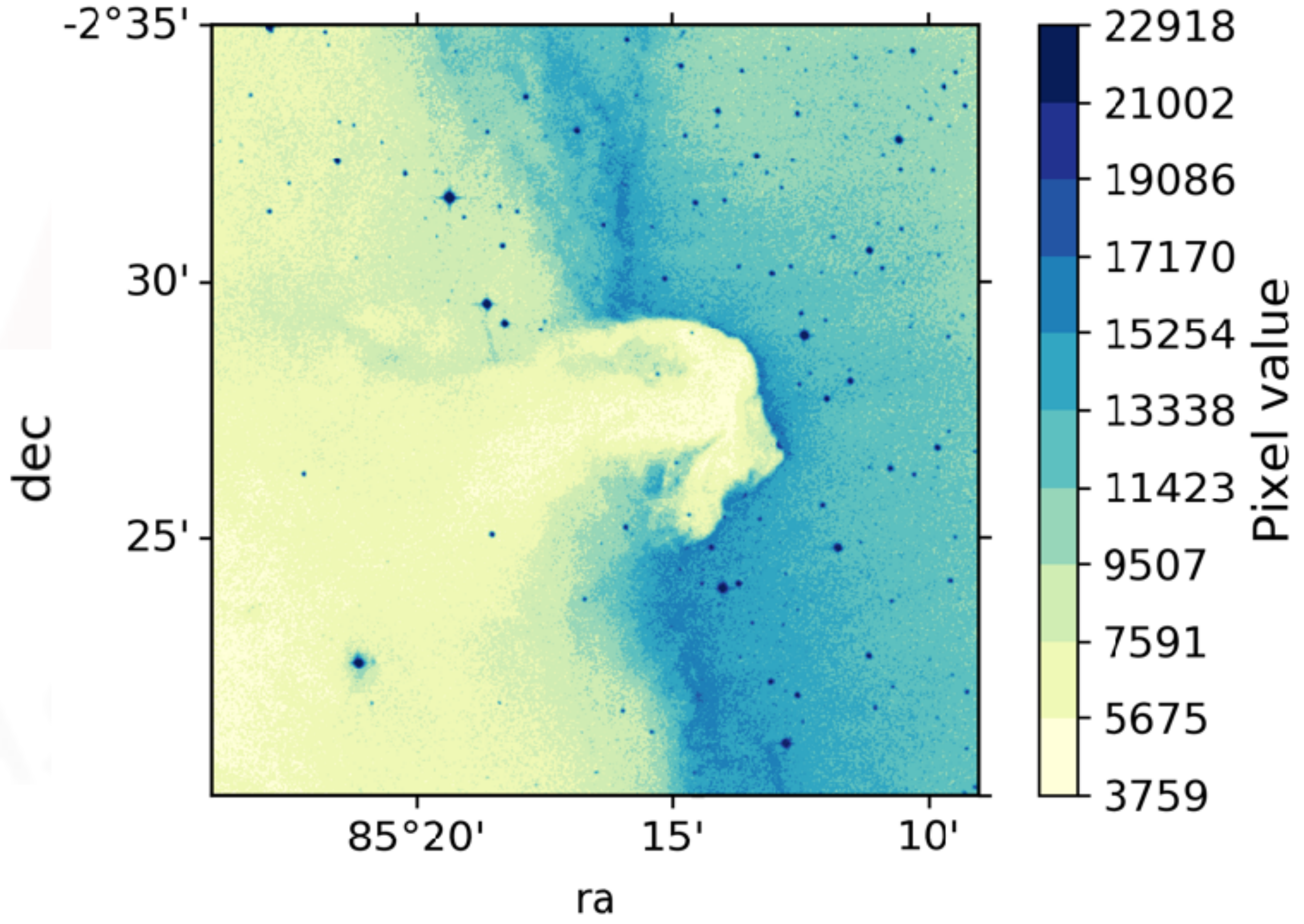
```
(…)
image = plt.imshow(data, cmap=cmap, norm=norm)

plt.xlabel('ra')
plt.ylabel('dec')

cbar = plt.colorbar(image, cmap=cmap,
             ticks=bounds,
             boundaries=bounds)

cbar.set_label('Pixel value', fontsize=14)
```

1.1 IMAGE

Exercise 1.3 Set a discrete colourbar (e.g. 0-5000: blue, 5000-10000: purple,…)







2. DIAGRAMS

2.1 Histogram

2.2 Barchart

2.1 HISTOGRAM

Exercise 2.1 Based on the image, plot the pixel distribution using a histogram.

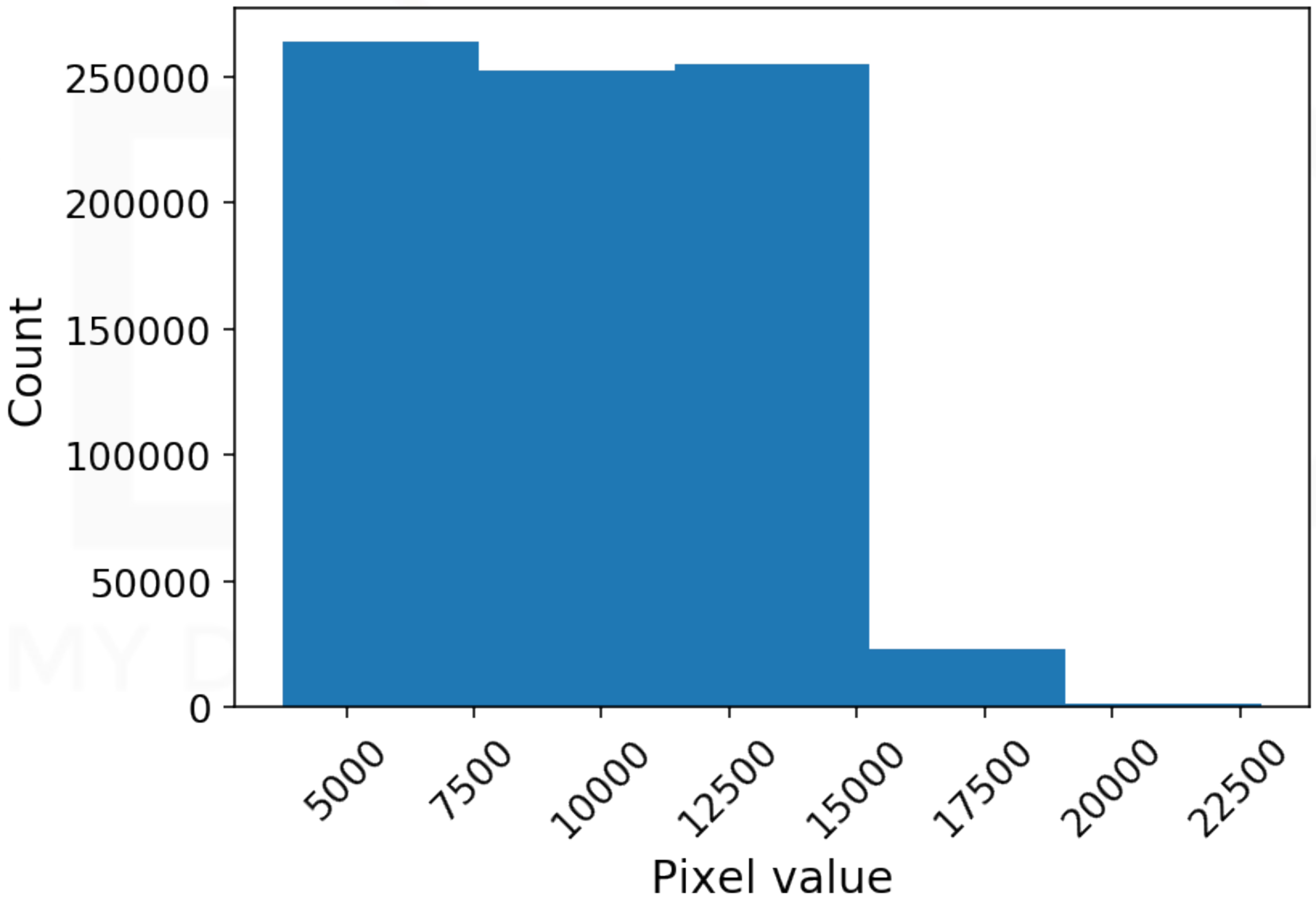
2.1 HISTOGRAM

Exercise 2.1.1 Based on the image, plot the pixel distribution using a histogram.

```
nbins = 5

# matplotlib.pyplot.hist takes a 1D array, or multiple 1D arrays.
# So, let's turn our array into 1D with numpy's flat function.
histogram = plt.hist(image_data.flat, nbins)

plt.xlabel('Pixel value')
plt.ylabel('Count')
plt.xticks(rotation = 45)
plt.show()
```

2.1 HISTOGRAM

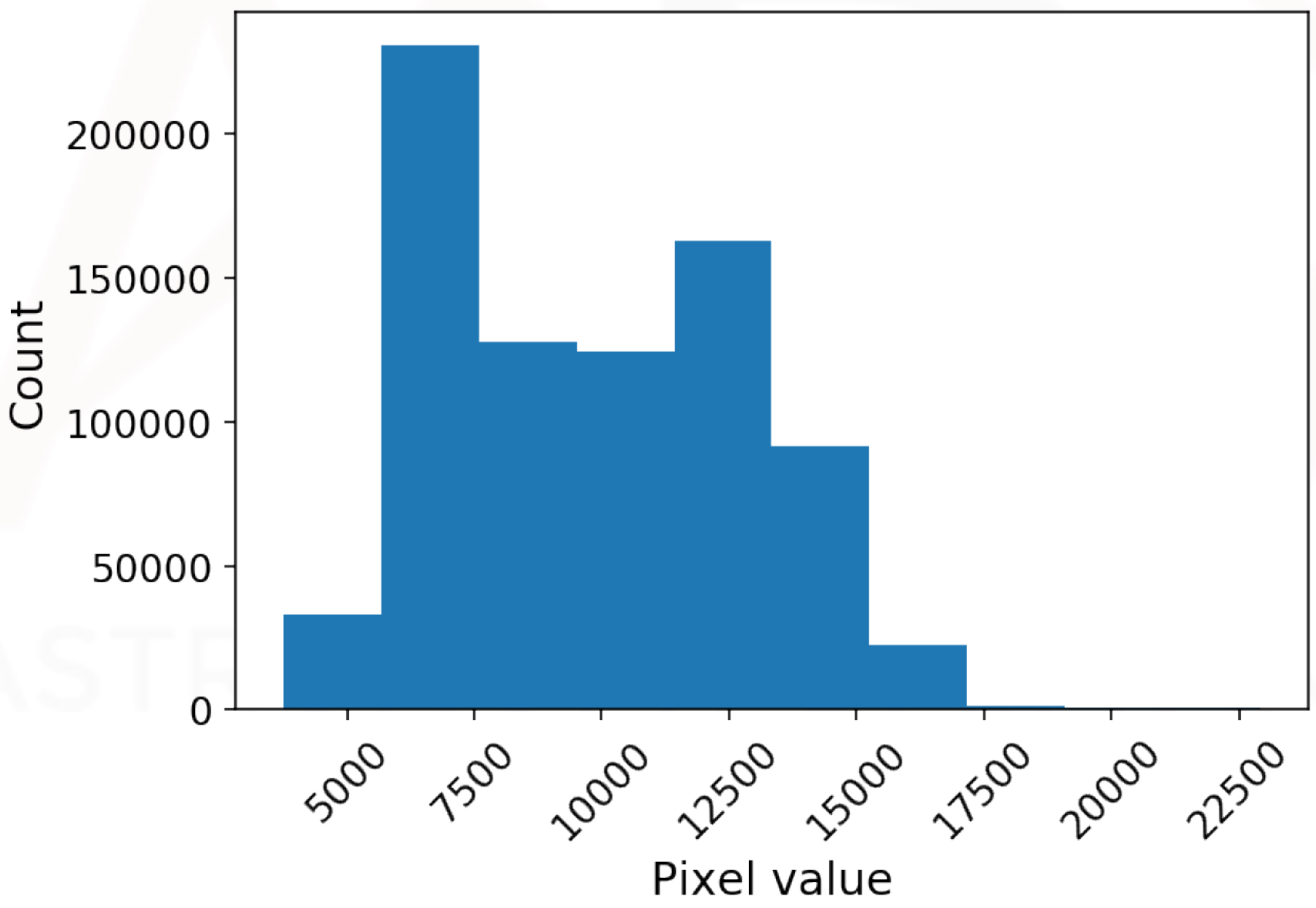
Exercise 2.1.1 Based on the image, plot the pixel distribution using a histogram.



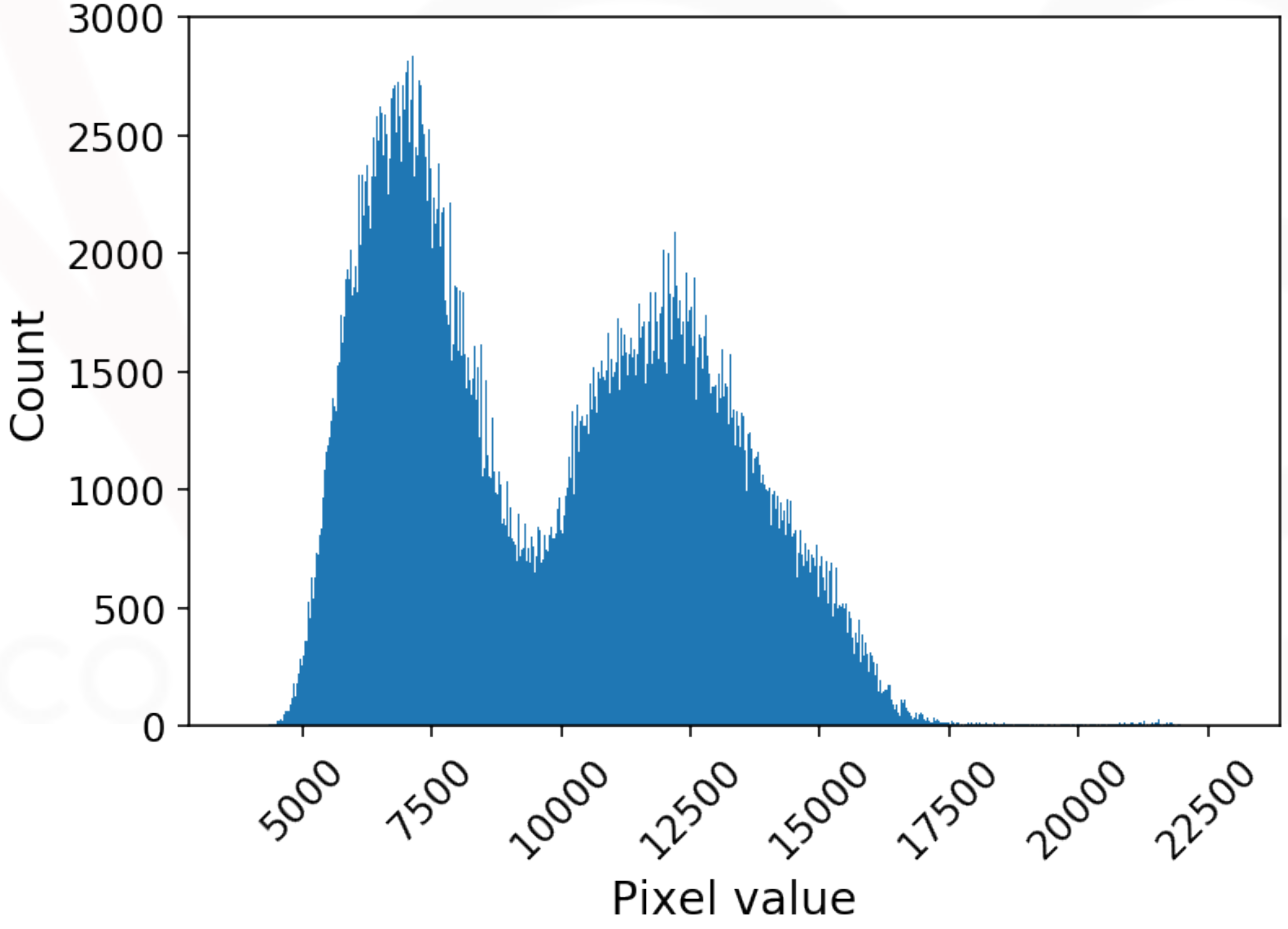
2.1 HISTOGRAM

Exercise 2.1.1 Based on the image, plot the pixel distribution using a histogram.

nbins = 10



nbins = 1000



2.1 HISTOGRAM

Food for thoughts

> How do you select the optimal number of bins for your histogram?

astropy.visualisation has some built-in features to do this, extending matplotlib. You may also be interested in scipy or scikit-learn (and astroML).

e.g. http://docs.astropy.org/en/stable/visualization/histogram.html

2.1 HISTOGRAM

Exercise 2.1.2 Fit this distribution.

2.1 HISTOGRAM

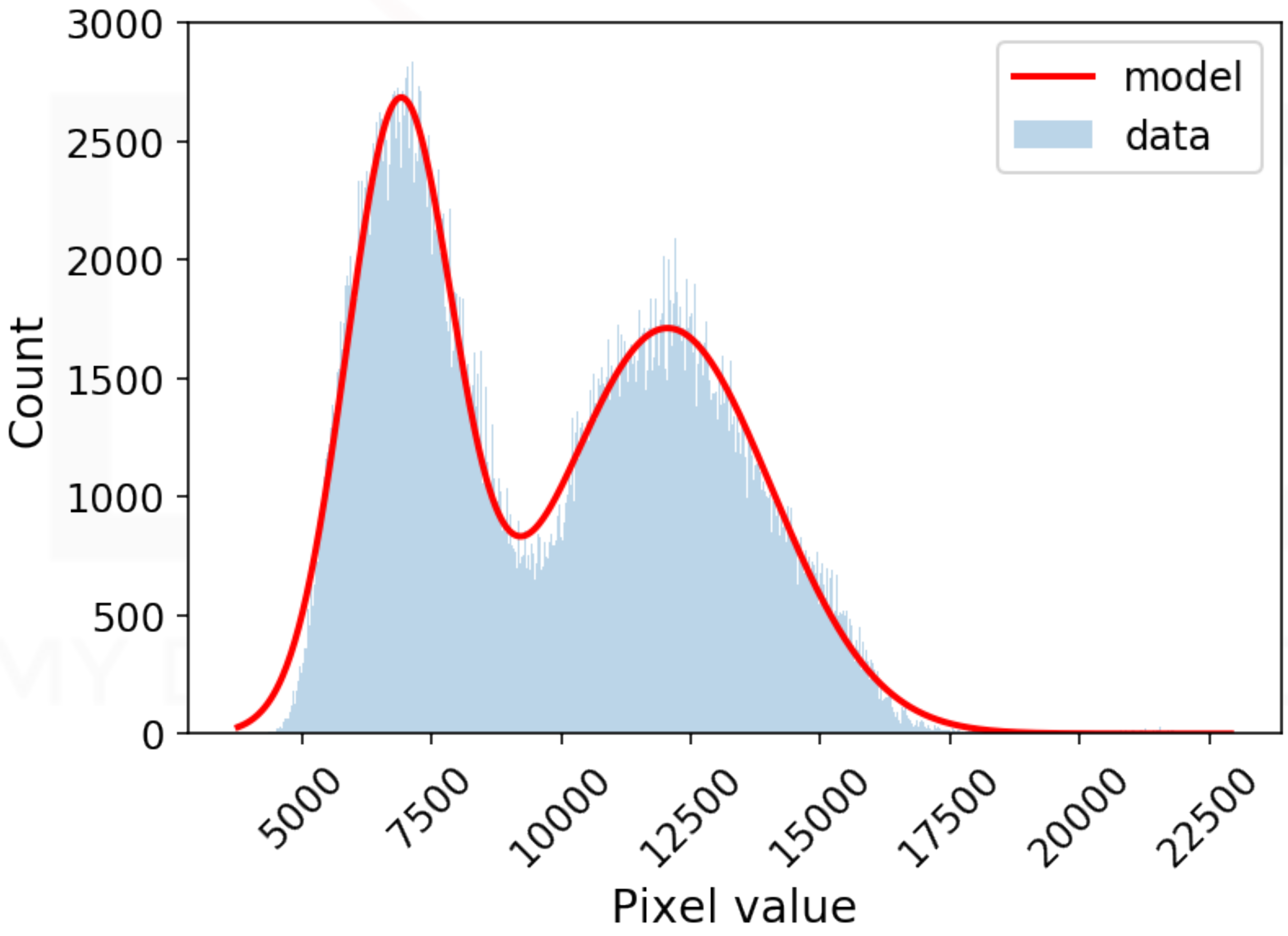
Exercise 2.1.2 Fit this distribution.

```
from scipy.optimize import curve_fit

def gauss(x,mu,sigma,A):
    return A*exp(-(x-mu)**2/2/sigma**2)

def bimodal(x,mu1,sigma1,A1,mu2,sigma2,A2):
    return gauss(x,mu1,sigma1,A1)+gauss(x,mu2,sigma2,A2)

y, x, _ = plt.hist(image_data.flat,1000,alpha=.3,label='data')

x=(x[1:]+x[:-1])/2 # for len(x)==len(y)

(…)
```

Exercise 2.1.2 Fit this distribution.

```
(…)
expected=(6000,1100,2500,12500,1300,2000)
params,cov=curve_fit(bimodal,x,y, expected)
sigma=sqrt(diag(cov))

plt.plot(x,bimodal(x,*params),color='red',lw=2,label='model')
plt.legend()

plt.xlabel('Pixel value')
plt.ylabel('Count')
plt.xticks(rotation = 45)
```

Exercise 2.1.2 Fit this distribution.



2.1 HISTOGRAM

Exercise 2.1.2 Fit this distribution.

```
import pandas as pd
pd.DataFrame(data={'params':params,'sigma':sigma},
             index=bimodal.__code__.co_varnames[1:])
```

	params	sigma
mu1	6880.723609	7.303146
sigma1	1031.798626	7.354773
A1	2621.855243	14.234672
mu2	12050.197185	15.531095
sigma2	2005.387802	17.248589
A2	1711.372081	10.428414

2.2 BARCHART

Exercise 2.2.1 Create a bar chart with data presented below.
Include errorbars, and a legend.

2.2 BARCHART

Exercise 2.2.1 Create a bar chart with data presented below.
Include errorbars, and a legend.

```
# We will have 5 categories
N = 5

# Generate some data
med_means = (20, 35, 30, 35, 27)
med_std = (2, 3, 4, 1, 2)

placebo_means = (25, 32, 34, 20, 25)
placebo_std = (3, 5, 2, 3, 3)
```

2.2 BARCHART

Exercise 2.2.1 Create a bar chart with data presented below.
Include errorbars, and a legend.

```
ind = np.arange(N)  # the x locations for the groups
width = 0.35
fig, ax = plt.subplots()

rects1 = ax.bar(ind, med_means, width,
                color='#1b9e77', yerr=med_std)

rects2 = ax.bar(ind + width, placebo_means, width,
                color='#d95f02', yerr=placebo_std)

(…)
```

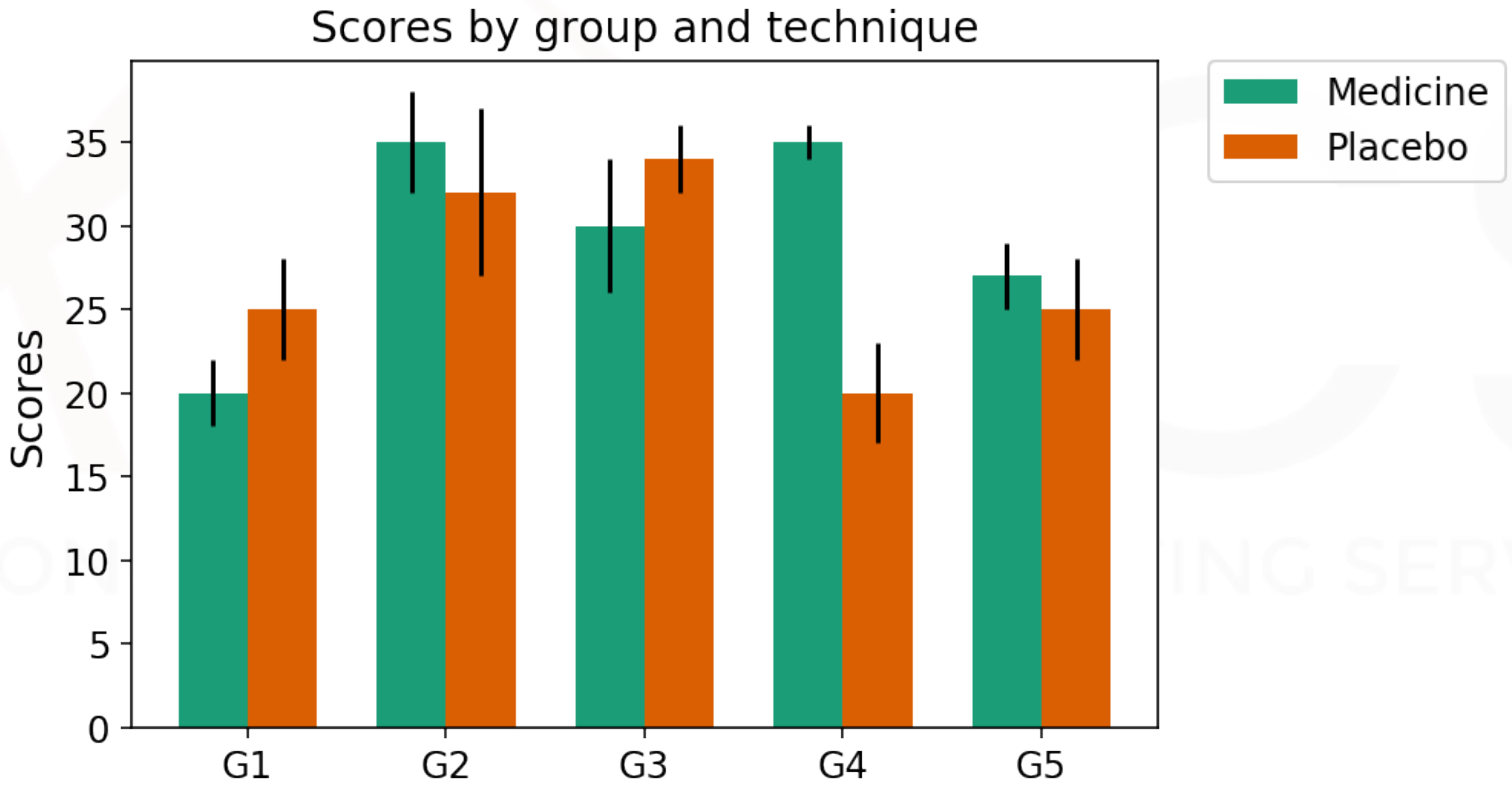
2.2 BARCHART

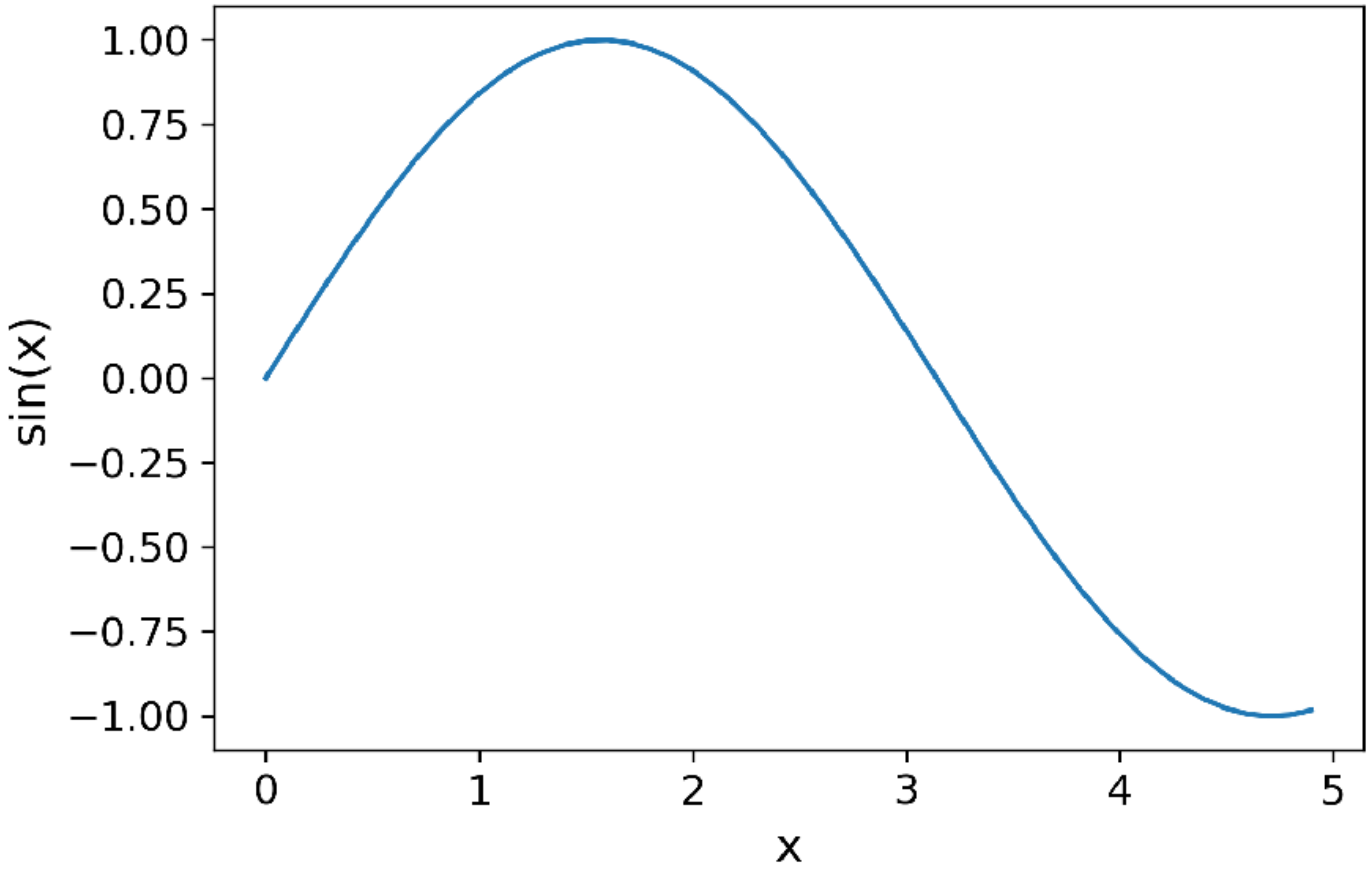
Exercise 2.2.1 Create a bar chart with data presented below. Include errorbars, and a legend.

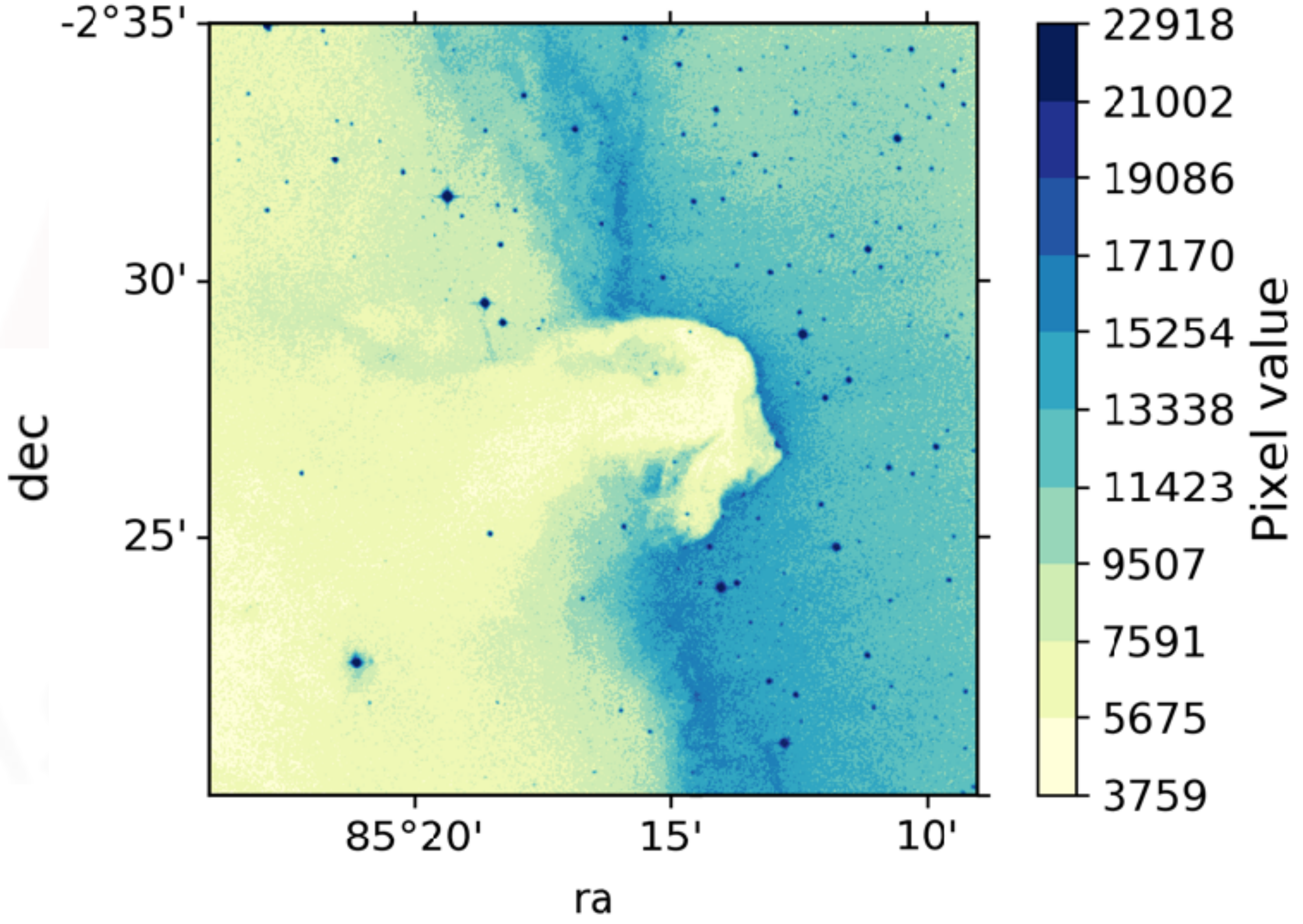
```
(…)

# add some text for labels, title and axes ticks
ax.set_ylabel('Scores')
ax.set_title('Scores by group and technique')
ax.set_xticks(ind + width/2)
ax.set_xticklabels(('G1', 'G2', 'G3', 'G4', 'G5'))

ax.legend((rects1[0], rects2[0]),
          ('Medicine', 'Placebo'),
          bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
```

2.2 BARCHART

Exercise 2.2.1 Create a bar chart with data presented below.
Include errorbars, and a legend.







3. PLOTS

3.1 Box plot
3.2 Plot

3.1 BOX PLOT

Exercise 3.1.1 Create a box plot displaying compression ratios results per technique from the file ratio.csv available in the data directory.

```
import csv
filename = 'data/ratio.csv'
labels = ["Bitpacking","Gzip+","Gzip-","LZ4+","LZ4-",
          "LZMA+","LZMA-","bzip2+","bzip2-","xz+","xz-"]
```

3.1 BOX PLOT

Exercise 3.1.1 Create a box plot displaying compression ratios results per technique from the file ratio.csv available in the data directory.

```
def getColumn(filename, column):
    results = csv.reader(open(filename))
    ii = 0
    row = []
    for result in results:
        if ii == 0:
            # Skip header
            ii += 1
        else:
            row.append(result[column])
    return row
```

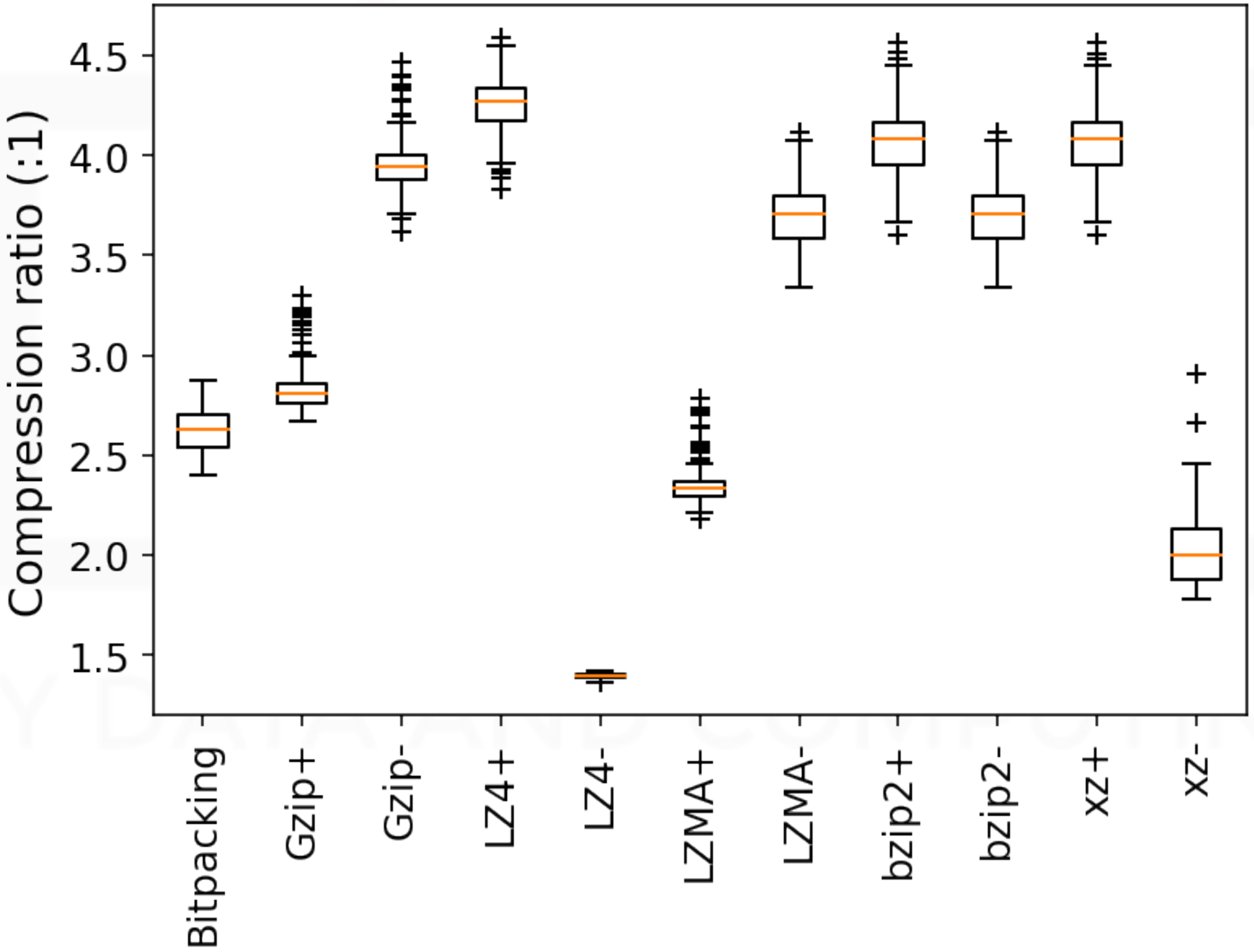
3.1 BOX PLOT

Exercise 3.1.1 Create a box plot displaying compression ratios results per technique from the file ratio.csv available in the data directory.

```
data = []
for i in range(len(headers)):
    data.append(np.asarray(getColumn(filename,i)[1:],
                                       dtype=np.float32))

fig = plt.figure(1)
ax = fig.add_subplot(111)
bp = ax.boxplot(data, sym='+')
ax.set_ylabel('Compression ratio (:1)')
plt.xticks([x+1 for x in range(len(labels))],
           labels, rotation='vertical')
```

3.1 BOX PLOT

Exercise 3.1.1 Create a box plot displaying compression ratios results per per technique from the file ratio.csv available in the data directory.



3.1 BOX PLOT

Exercise 3.1.2 Producing a similar plot with the Pandas package

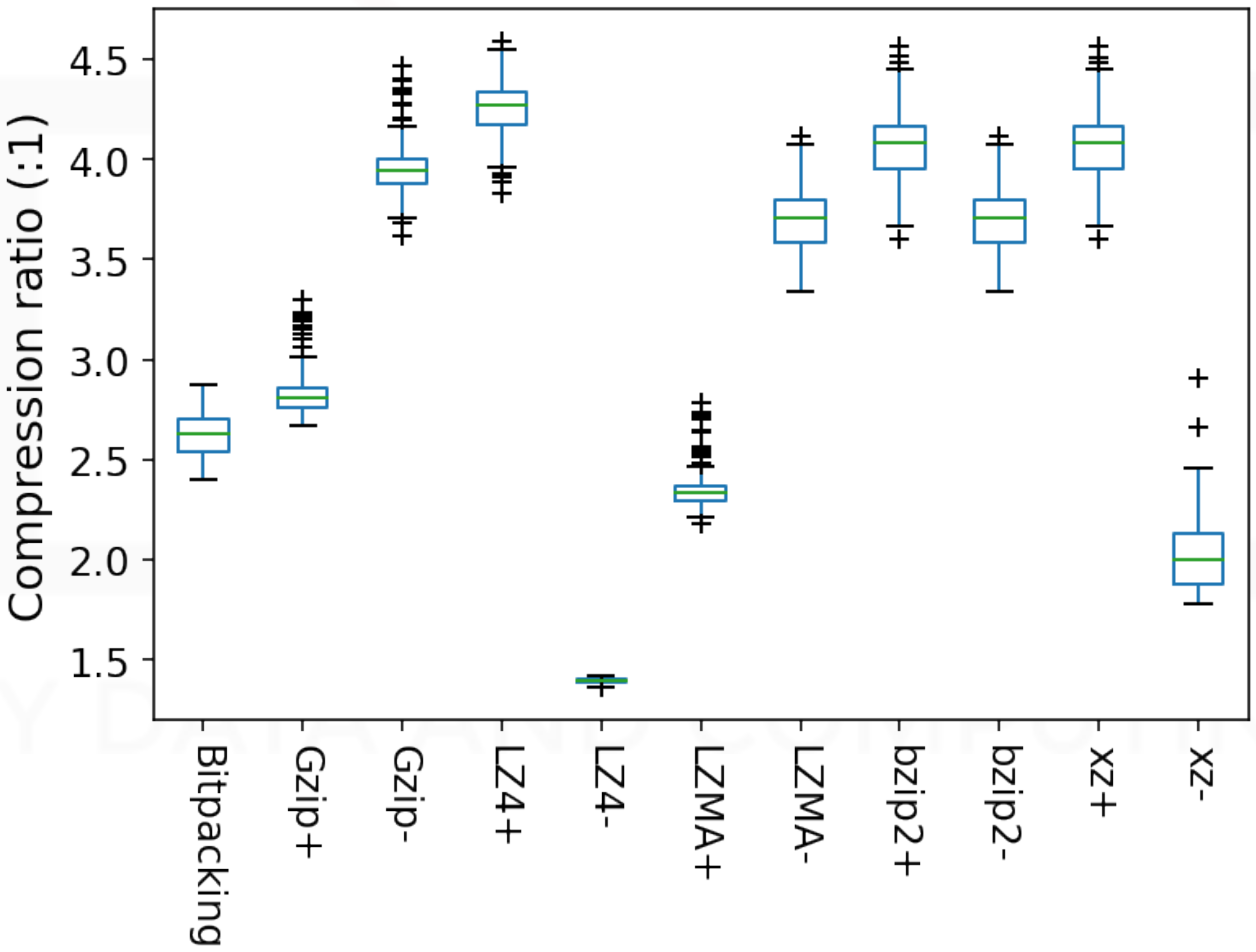
3.1 BOX PLOT

Exercise 3.1.2 Producing a similar plot with the Pandas package

```
from pandas import read_csv

# Load data into Pandas' data frame format (df)
df = read_csv('data/ratio.csv')

ax = df.boxplot(rot=270, sym='+')
ax.set_ylabel('Compression ratio (:1)')
ax.grid(False)
```

3.1 BOX PLOT

Exercise 3.1.2 Producing a similar plot with the Pandas package



3.1 BOX PLOT

Exercise 3.1.3 Make a multi-panel figure (one per experiment)

3.1 BOX PLOT

Exercise 3.1.3 Make a multi-panel figure (one per experiment)

```
# Four axes, returned as a 2-d array
f, axarr = plt.subplots(2, 2, figsize=(10, 8),
                        sharex=True, sharey=True)
# Initialise some counters
x = 0
y = 0

# Fine-tune figure; hide x ticks for top plots
# and y ticks for right plots
plt.setp([a.get_xticklabels() for a in axarr[0, :]],
         visible=False)
plt.setp([a.get_yticklabels() for a in axarr[:, 1]],
         visible=False)
(…)
```

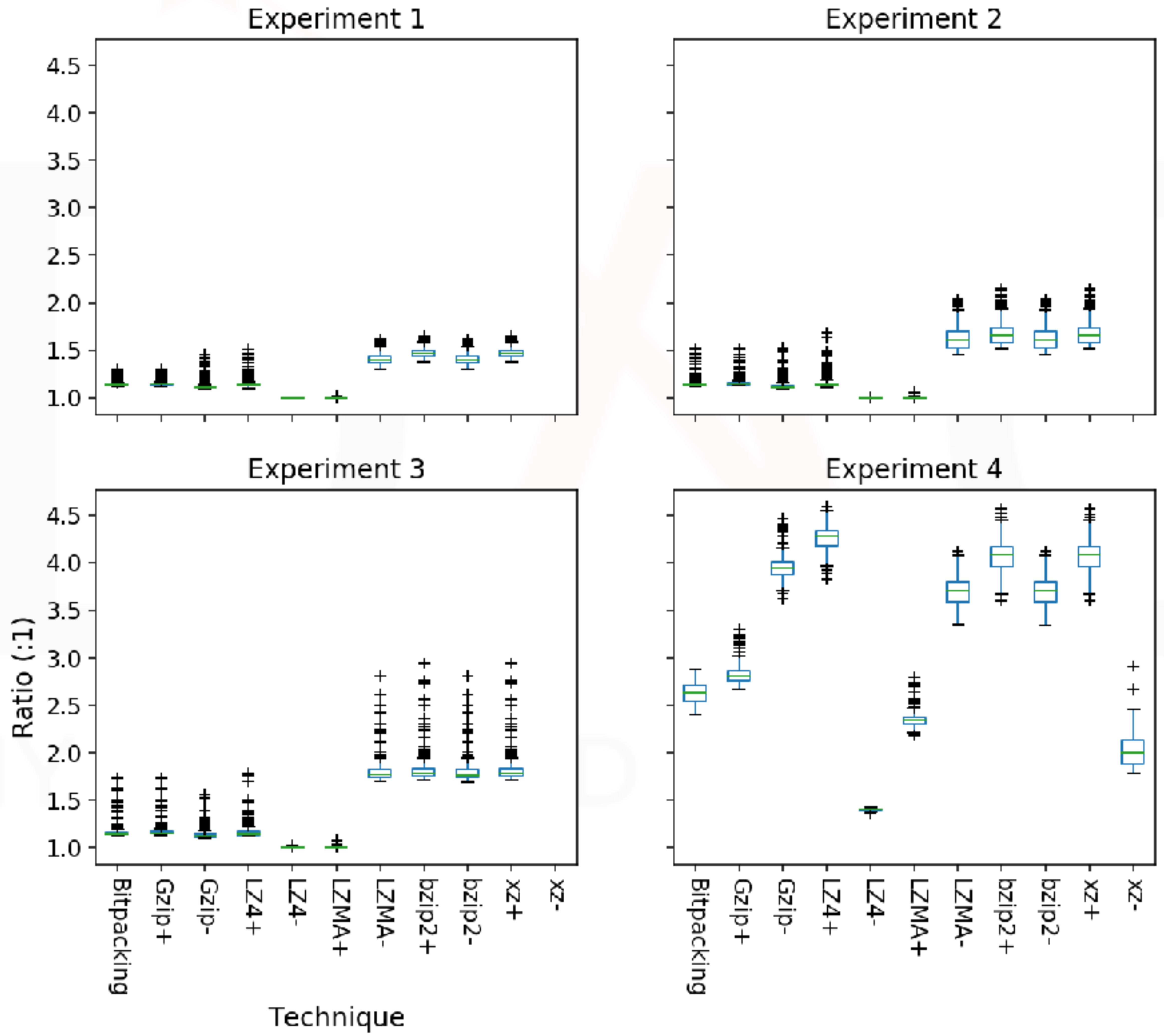
3.1 BOX PLOT

Exercise 3.1.3 Make a multi-panel figure (one per experiment)

```
for i in [1, 2, 3, 4]:
    df = read_csv('data/ratio-exp' + str(i) + '.csv')
    df.boxplot(rot=270, ax=axarr[y, x], sym='+', )
    axarr[y, x].grid(False)
    axarr[y, x].set_title('Experiment ' + str(i))

    x += 1
    if x > 1:
        x = 0
        y += 1

axarr[1, 0].set_ylabel('Ratio (:1)')
axarr[1, 0].set_xlabel('Technique')
```

3.1 BOX PLOT

Exercise 3.1.3 Make a multi-panel figure (one per experiment)



3.2 PLOT

Exercise 3.2.1 Load the 3D fits file containing a spectral cube of NGC 4826 taken from the THINGS survey (NGC_2903_RO_CUBE_THINGS.FITS).

For this exercise, load the data cube into memory using astropy. Then plot the spectrum at the center of the image. Use the units from the file header.

3.2 PLOT

Exercise 3.2.1 Plot the spectrum at the center of the image.

```
cube = fits.open('data/NGC_2903_RO_CUBE_THINGS.FITS')[0]
print (cube.header)

fig, axarr = plt.subplots()

axarr.plot(cube.data[0,
                     :,
                     int(cube.data.shape[2]/2),
                     int(cube.data.shape[3]/2)])

# Get ticks labels to replace them later
fig.canvas.draw()
labels = [item.get_text() for item in axarr.get_xticklabels()]

(…)
```

3.2 PLOT

Exercise 3.2.1 Plot the spectrum at the center of the image.

`CTYPE3`	Type of 3rd axis (e.g. wavelength, velocity, …)
`NAXIS3`	Axis dimension (3rd axis)
`CRVAL3`	Central velocity coordinate
`CRPIX3`	Pixel coordinates of the reference point
`CDELT3`	Increment value
`BUNIT`	Unit of value

3.2 PLOT

Exercise 3.2.1 Plot the spectrum at the center of the image.

xmin = `CRVAL3` – `CRPIX3` * `CDELT3`

xmax = `CRVAL3` + `CDELT3` * (`NAXIS3` – `CRPIX3`)

Exercise 3.2.1 Plot the spectrum at the center of the image.

(…)

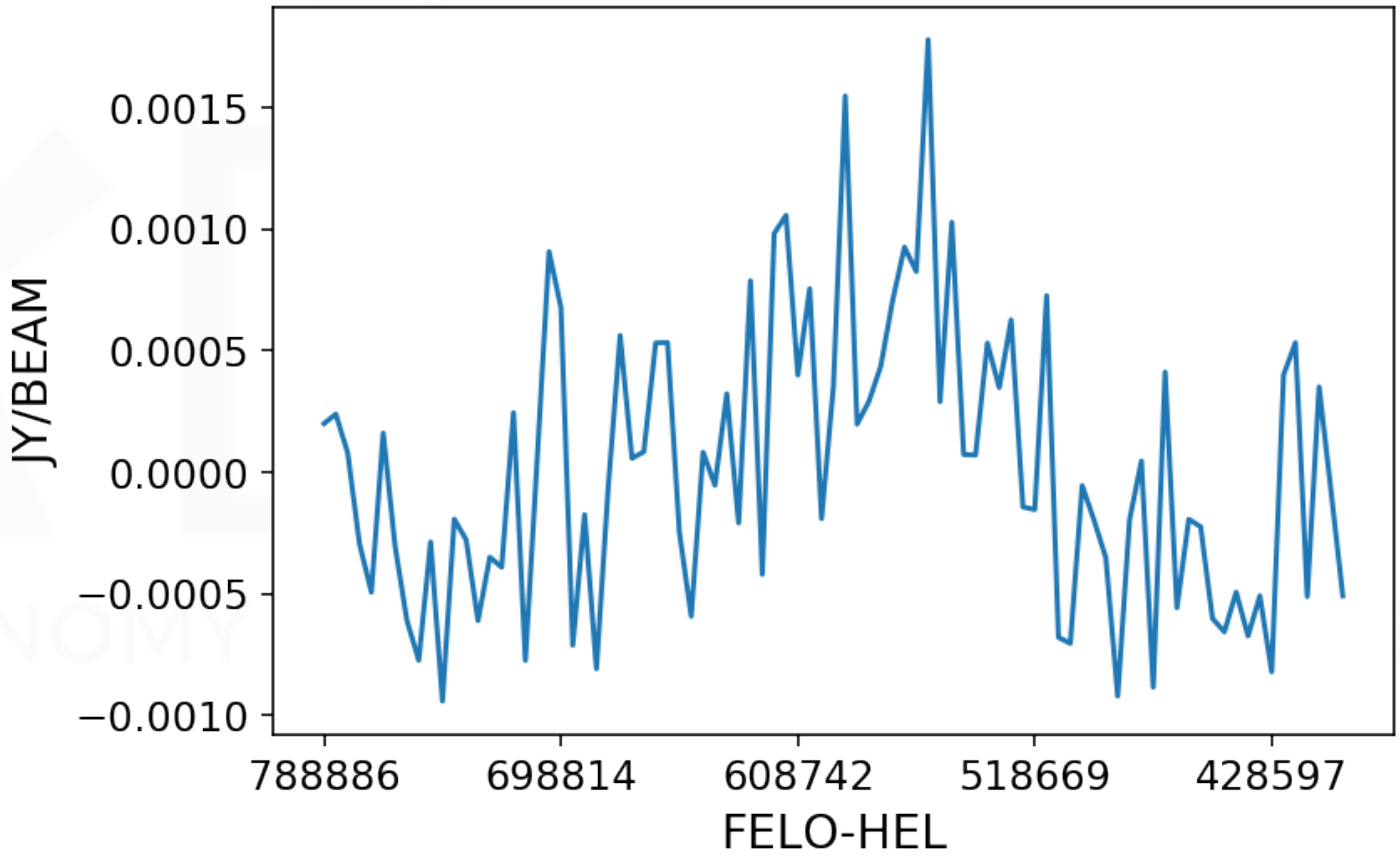
```
header = data.header
x_min = header['CRVAL3'] - header['CRPIX3'] \
                            * header['CDELT3']
x_max = header['CRVAL3'] + header['CDELT3'] \
                            * (header['NAXIS3'] * header['CRPIX3'])
nbins = len(labels)-2

ticks_labels = [int(i)
                for i in np.arange(x_min,
                                   x_max,
                                   (x_max-x_min)/nbins)]
```

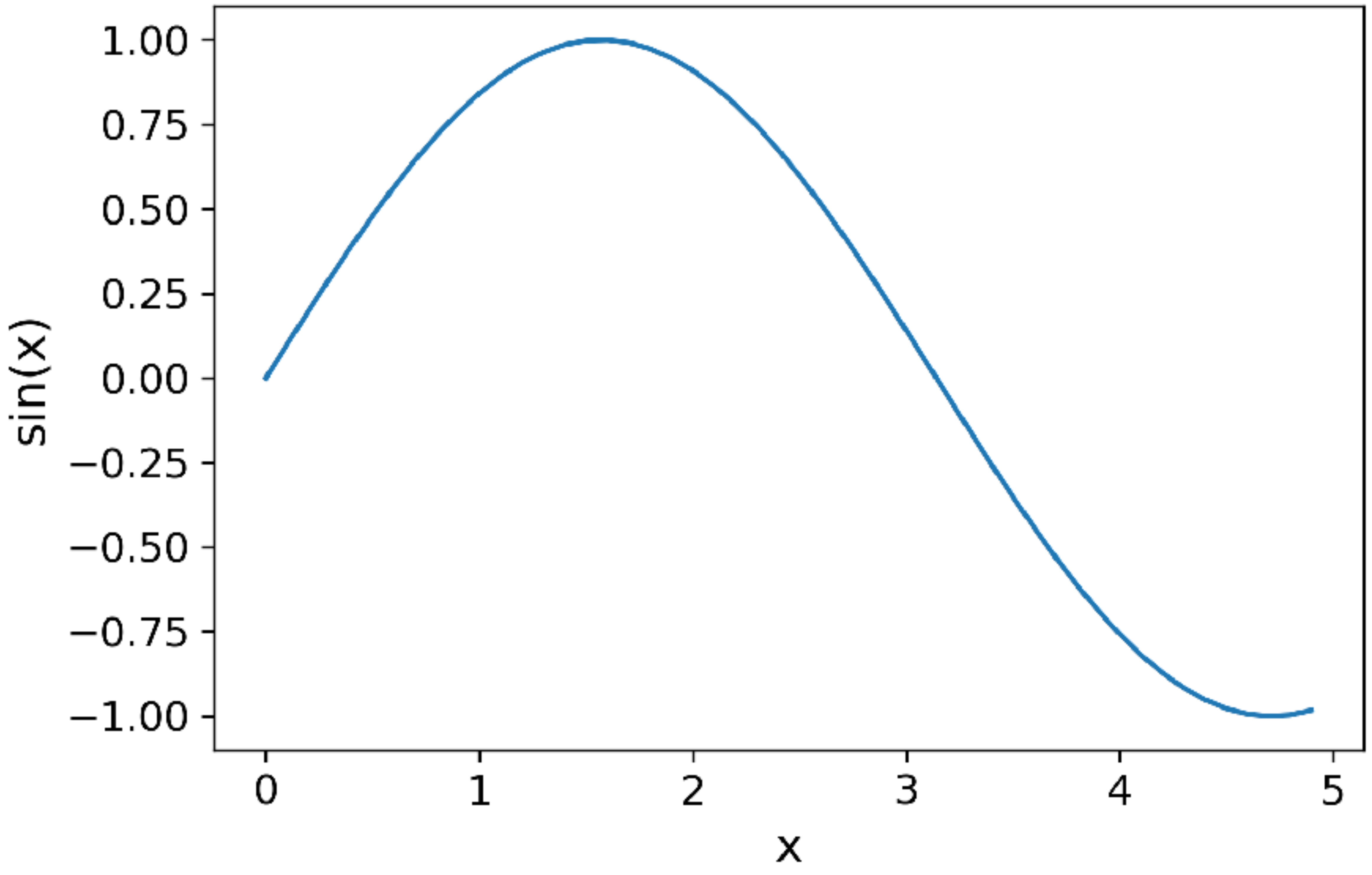
Exercise 3.2.1 Plot the spectrum at the center of the image.

```
(…)

for i in range(len(labels)):
    if i != 0 and i-1 < nbins:
        labels[i] = ticks_labels[i-1]

axarr.set_xticklabels(labels)
axarr.set_xlabel(cube.header['CTYPE3'])
axarr.set_ylabel(cube.header['BUNIT'])
```

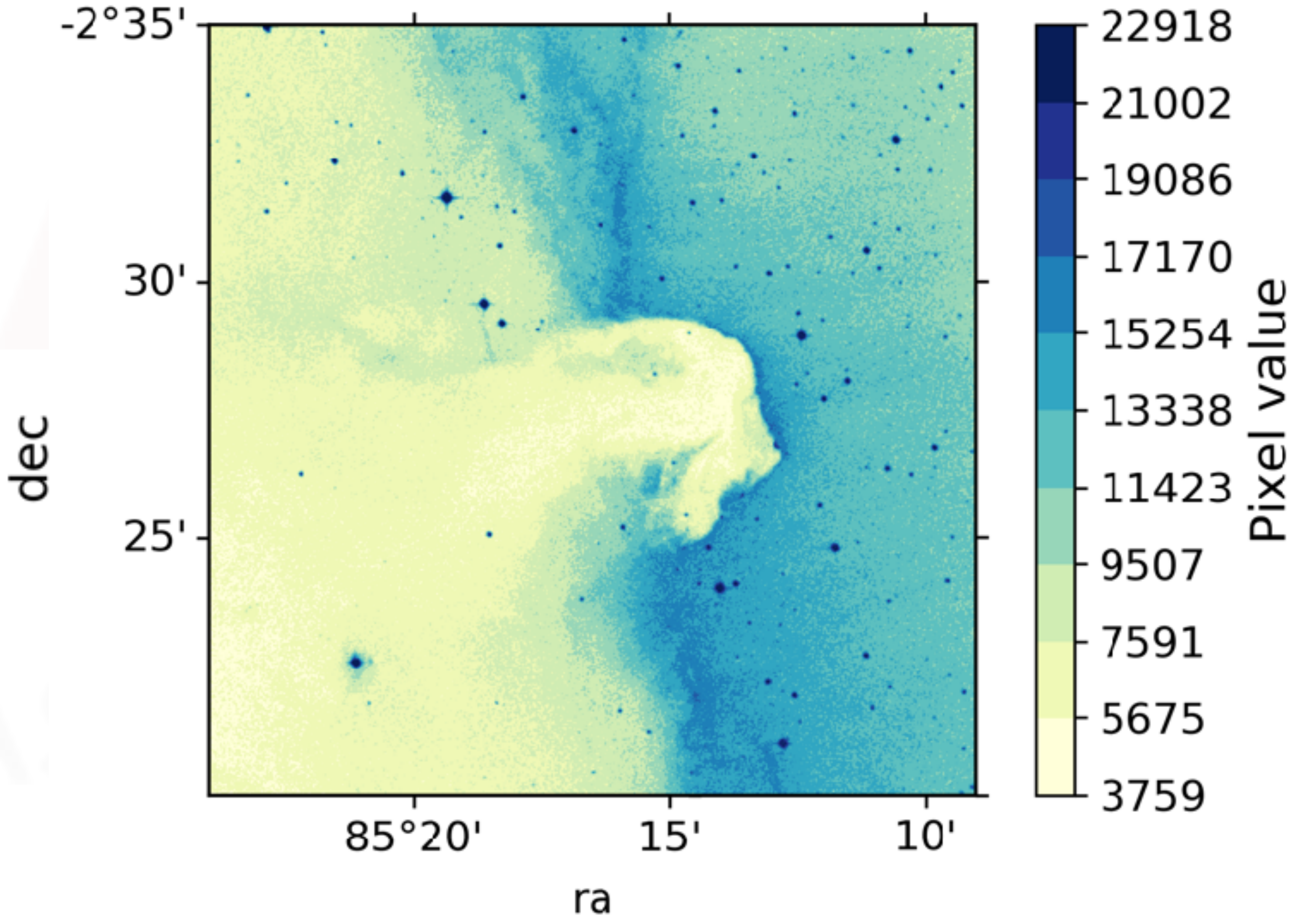
3.2 PLOT

Exercise 3.2.1 Plot the spectrum at the center of the image.





1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
sin(x)
0
1
2
3
4
5
x



-2°35'
30'
25'
dec
85°20'
15'
10'
ra
22918
21002
19086
17170
15254
13338
11423
9507
7591
5675
3759
Pixel value

REFERENCES

REFERENCES

- Dürsteler, J. C. 2002. Rules to make a bad graphic representation. *InfoVis*, 109.
- Gales, L. 2003. "Graphics and Web Design Based on Edward Tufte's Principles."
- Tufte, Edward R. 1985. "The visual display of quantitative information." *Journal for Healthcare Quality* 7.3: 15.
- Couture, M., & Francis, A. 2004. "Introduction aux méthodes de recherche scientifique". *Edition de la Téluq*.
- Vohl, D., Fluke, C. J., & Vernardos, G. 2015. "Data compression in the petascale astronomy era: A GERLUMPH case study." *Astronomy and Computing* 12: 200-211.
- Walter, F., *et al.* 2008, "THINGS: The HI Nearby Galaxy Survey." *The Astronomical Journal*, 136, 2563
- Wang, J., *et al.* 2014. "An observational and theoretical view of the radial distribution of HI gas in galaxies." *Monthly Notices of the Royal Astronomical Society* 441.3: 2159-2172.