YData: Introduction to Data Science

Class 07: Array computation continued

Overview

Quick review of:

- NumPy arrays
- Numerical computations

More numpy:

- Boolean arrays
- Boolean masking
- Higher dimensional numerical arrays
- Image manipulation

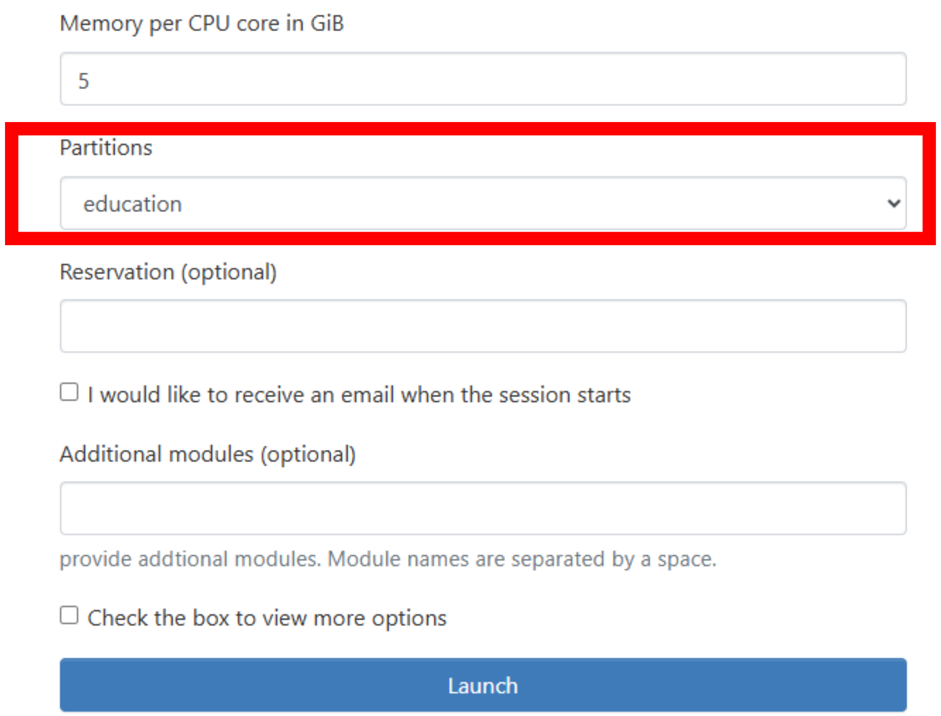
If there is time:

- Tuples
- Dictionaries

Announcement: Homework 3

Homework 3 is due on Gradescope on
Sunday September 21st at 11pm

- **Be sure to mark each question on Gradescope!**



Note:

- On problem 3, if the images are not showing up make sure to close and reopen the notebook
  - If you figure it out, help other people on Ed!

Quick review: ndarrays

The *NumPy package* efficiently stores and processes data that is all of the same type using ***ndarray***

```
import numpy as np

my_array = np.array([1, 2, 3])   # creating an ndarray
my_array[0]                      # accessing the 0th element

my_array.dtype            # get the type of elements
my_array.shape            # get the dimension
my_array.astype('str')    # convert to strings

sequential_nums = np.arange(1, 10)    # creates numbers 1 to 9
```

Quick review: functions on numerical arrays

The NumPy functions:

- np.sum()
- np.max(), np.min()
- np.mean(), np.median()
- np.diff() # takes the difference between elements
- np.cumsum() # cumulative sum

There are also "broadcast" functions that operate on all elements in an array

- my_array = np.array([12, 4, 6, 3, 4, 3, 7, 4])
- my_array * 2

- my_array2 = np.array([10, 9, 2, 8, 9, 3, 8, 5])
- my_array - my_array2

Boolean arrays

Boolean arrays

It is often to compare all values in an ndarray to a particular value

- my_array = np.array([12, 4, 6, 3, 4, 3, 7, 4])
- my_array < 5   # any guesses what this will return
    - array([False, True, False, True, True, True, False, True])

This can be useful for calculating proportions

- True == 1 and False == 0
- Taking the sum of a Boolean array gives the total number of True values
- The number of True 's divided by the length is the proportion
    - Or we can use the np.mean() function

Categorical Variable

PLAYER	POSITION	TEAM	SALARY
str	str	str	f64
"Paul Millsap"	"PF"	"Atlanta Hawks"	18.671659
"Al Horford"	"C"	"Atlanta Hawks"	12.0
"Tiago Splitter...	"C"	"Atlanta Hawks"	9.75625
"Jeff Teague"	"PG"	"Atlanta Hawks"	8.0
"Kyle Korver"	"SG"	"Atlanta Hawks"	5.746479

$$\text{Proportion centers} = \frac{\text{number of centers}}{\text{total number}}$$

Let's explore this in Jupyter!

Boolean masking

Boolean masking

We can also use Boolean arrays to return values in another array

- This is called "Boolean masking”, “Boolean subsetting” or "Boolean indexing"

```
my_array = np.array([12, 1, 6, 2, 3])
boolean_mask = np.array([False, True, False, True, True])

smaller_array = my_array[boolean_mask]
```

This can be useful for calculating statistics on data that meet particular criteria:

- `np.mean(my_array[my_array < 5])    # what does this do?`

Boolean masking

Suppose you wanted to get the average movie revenue for movies that passed the Bechdel test

- domgross_2013: Movie revenue
- bechdel: whether a movie passed the Bechdel test

Can you do it?

Let's explore this in Jupyter!

Percentiles

Percentiles

The **Pth percentile** is the value of a quantitative variable which is greater than P percent of the data

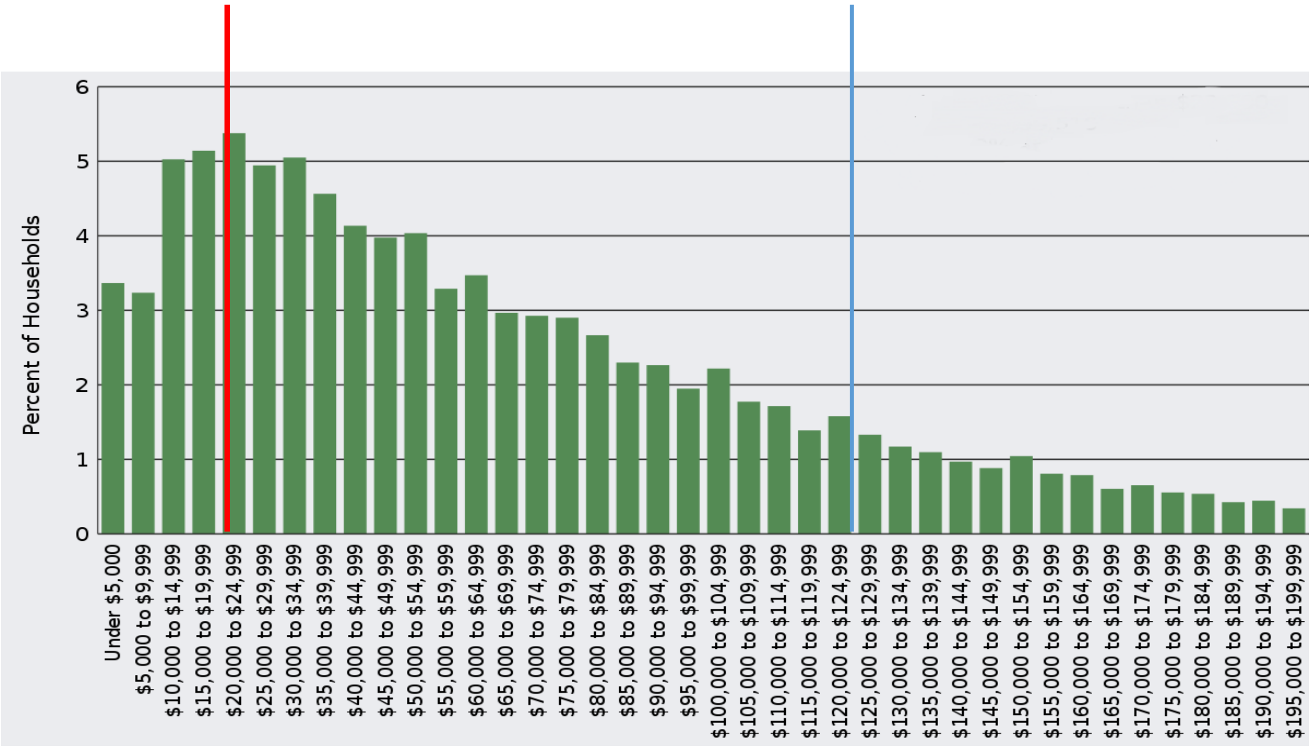
For the US income distribution what are the 20th and 80th percentiles?

We can calculate percentiles using np.percentile()

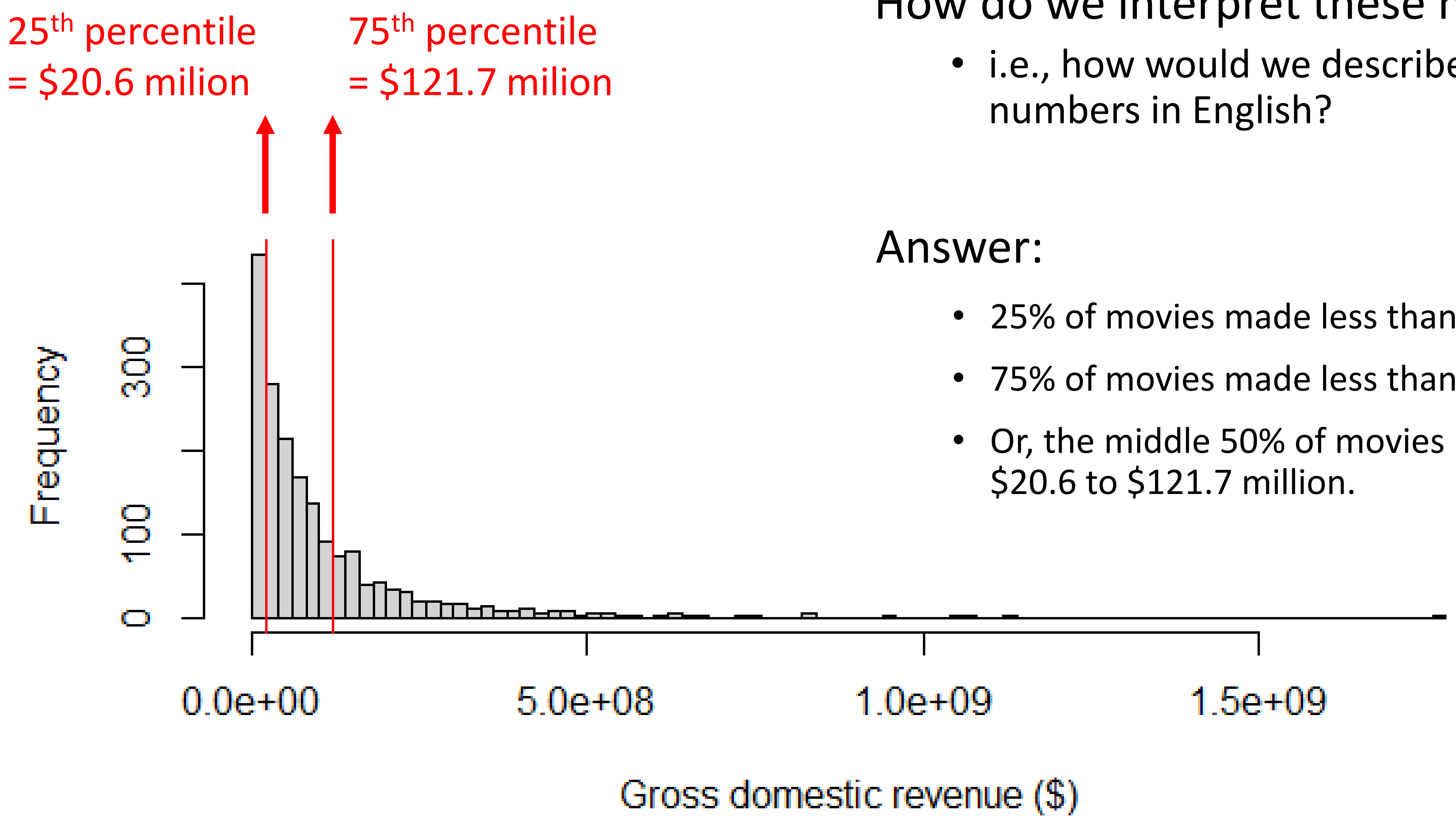
```
np.percentile(data, [20, 80])
```

20th percentile = $21,430

80th percentile = $112, 254



Movie revenue percentiles



How do we interpret these numbers?

- i.e., how would we describe these numbers in English?

Answer:

- 25% of movies made less than $20.6 million
- 75% of movies made less than $121.7 million
- Or, the middle 50% of movies made between $20.6 to $121.7 million.

Five Number Summary

Five Number Summary = (minimum, Q_1, median, Q_3, maximum)

Q_1 = 25th percentile (also called 1st quartile)

Q_3 = 75th percentile (also called 3rd quartile)

Roughly divides the data into fourths

Range and Interquartile Range

Range = maximum – minimum

Interquartile range (IQR) = $Q_3 - Q_1$

Let's calculate these statistics on Bechdel movie revenue data!

Box plots and outliers

Detecting of outliers

As a rule of thumb, we call a data value an **outlier** if it is:

Smaller than: $Q_1 - 1.5 * IQR$

Larger than: $Q_3 + 1.5 * IQR$

Box plots

A **box plot** is a graphical display of the five-number summary and consists of:

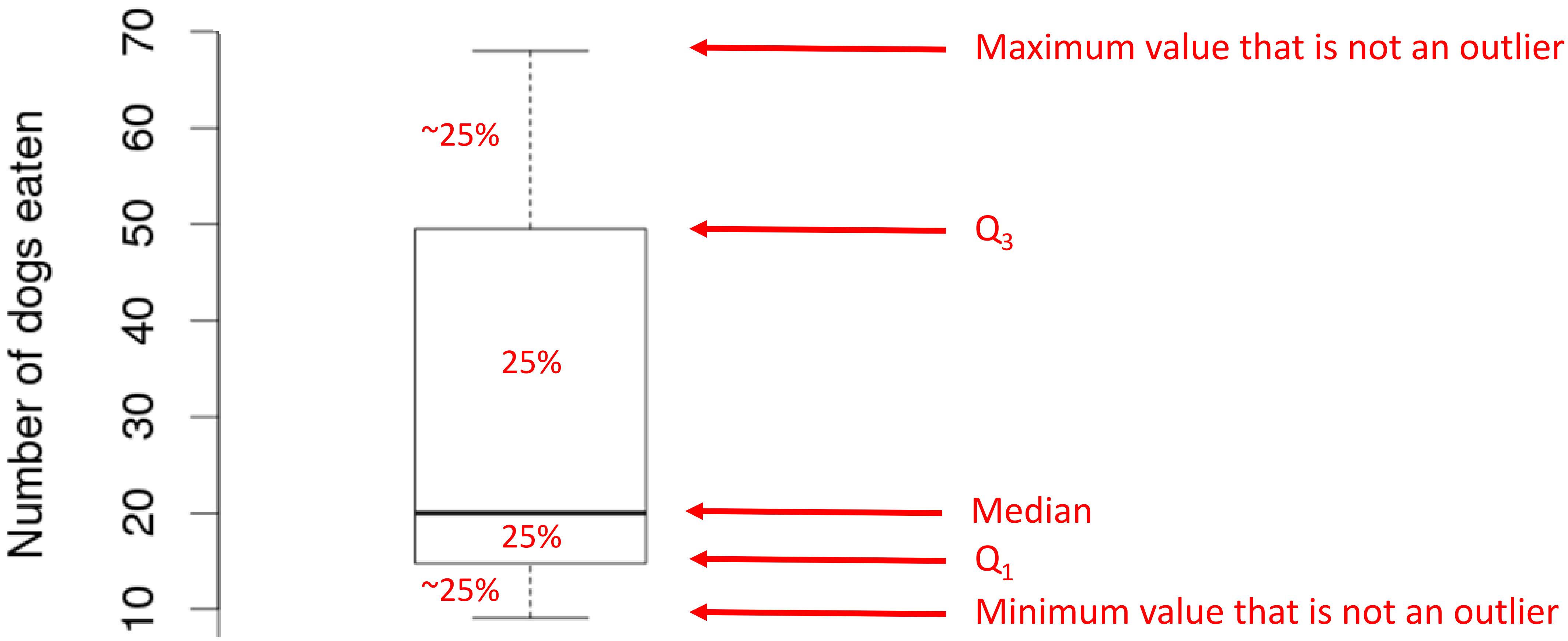
1. Drawing a box from Q_1 to Q_3

2. Dividing the box with a line (or dot) drawn at the median

3. Draw a line from each quartile to the most extreme data value that is not and outlier

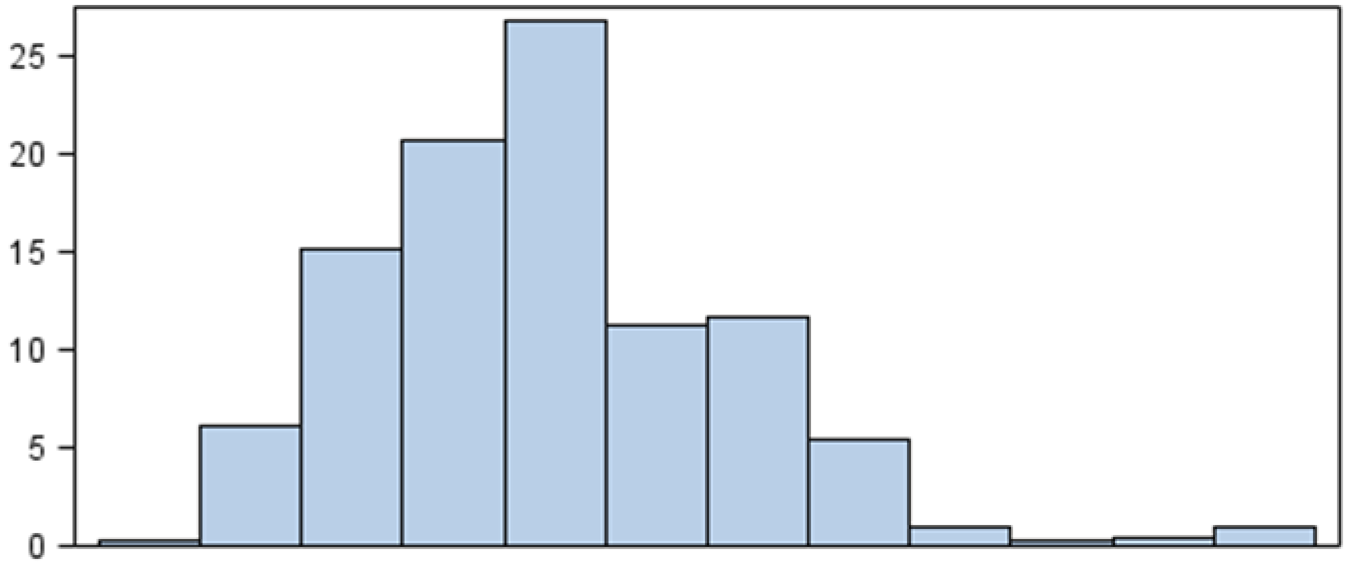
4. Draw a dot/asterisk for each outlier data point.

Box plot of the number of hot dogs eaten by the men's contest winners 1980 to 2010



Matplotlib: `plt.boxplot(data, labels)`

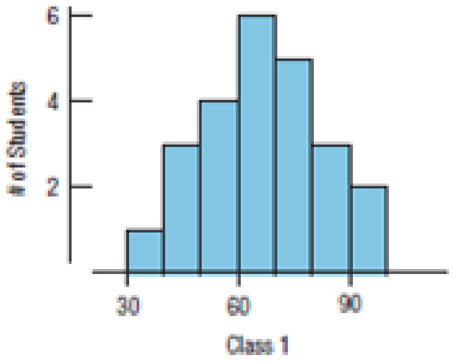
Box plots extract key statistics from histograms



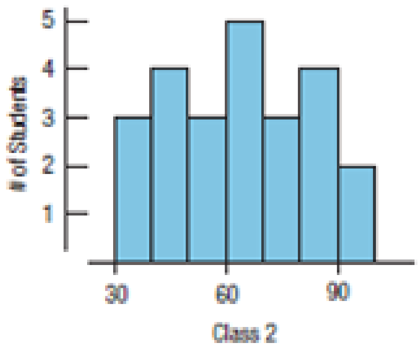
Box plots extract key statistics from histograms

Question: which Box plot goes with which histogram?

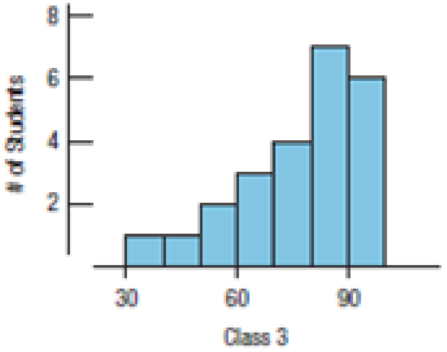
Histogram 1

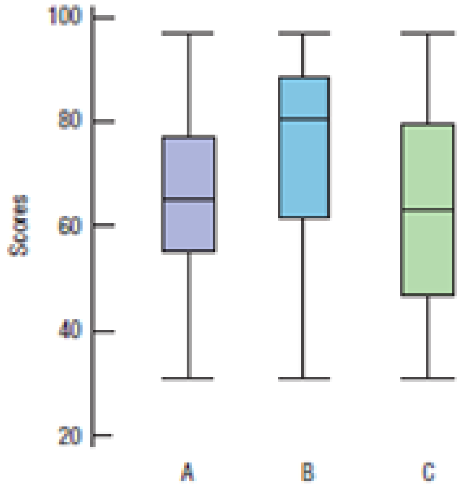


Histogram 2



Histogram 3

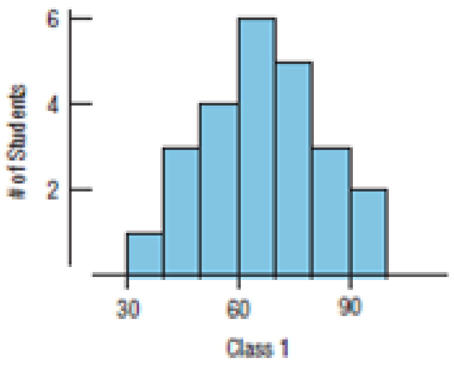




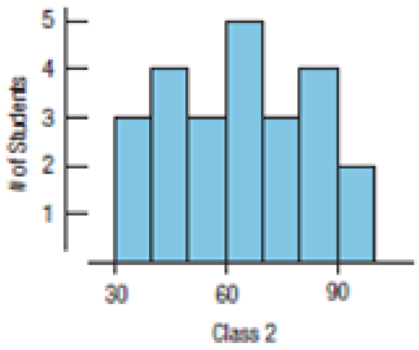
Box plots extract key statistics from histograms

Question: which Box plot goes with which histogram?

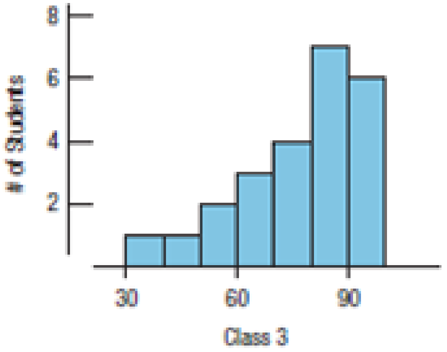
A

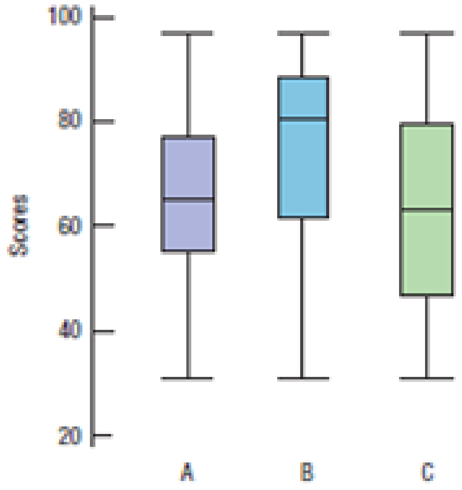


C



B

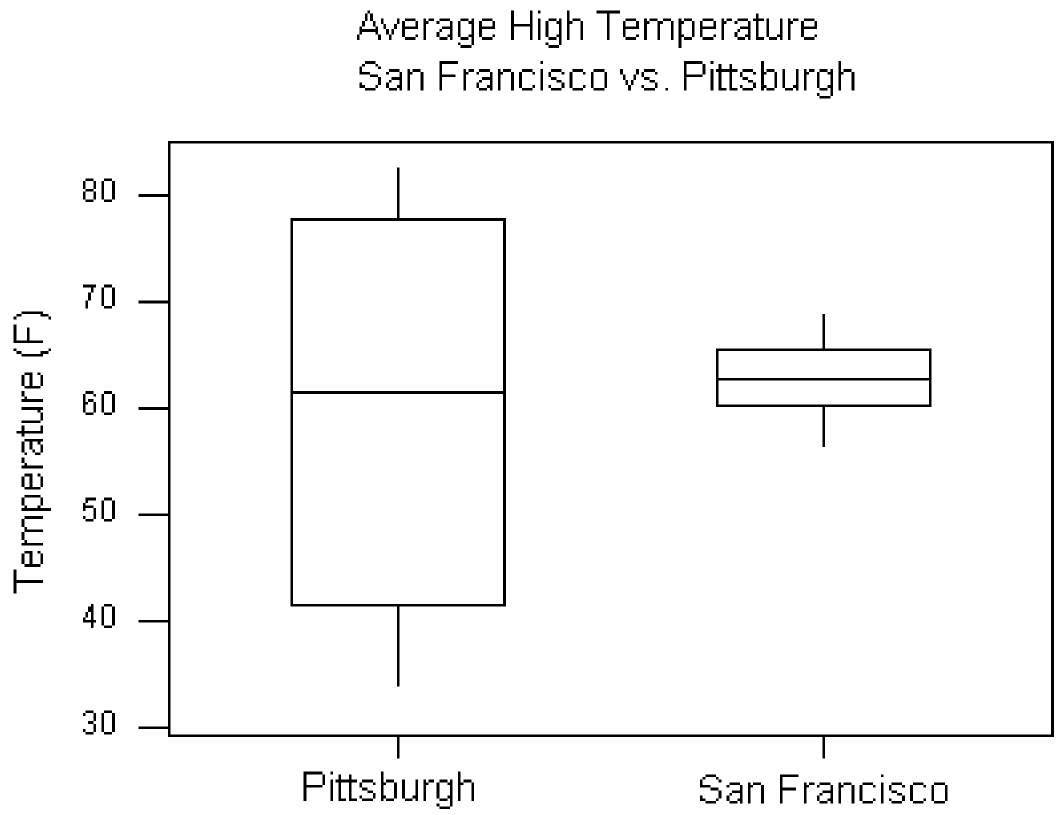




Comparing quantitative variables across categories

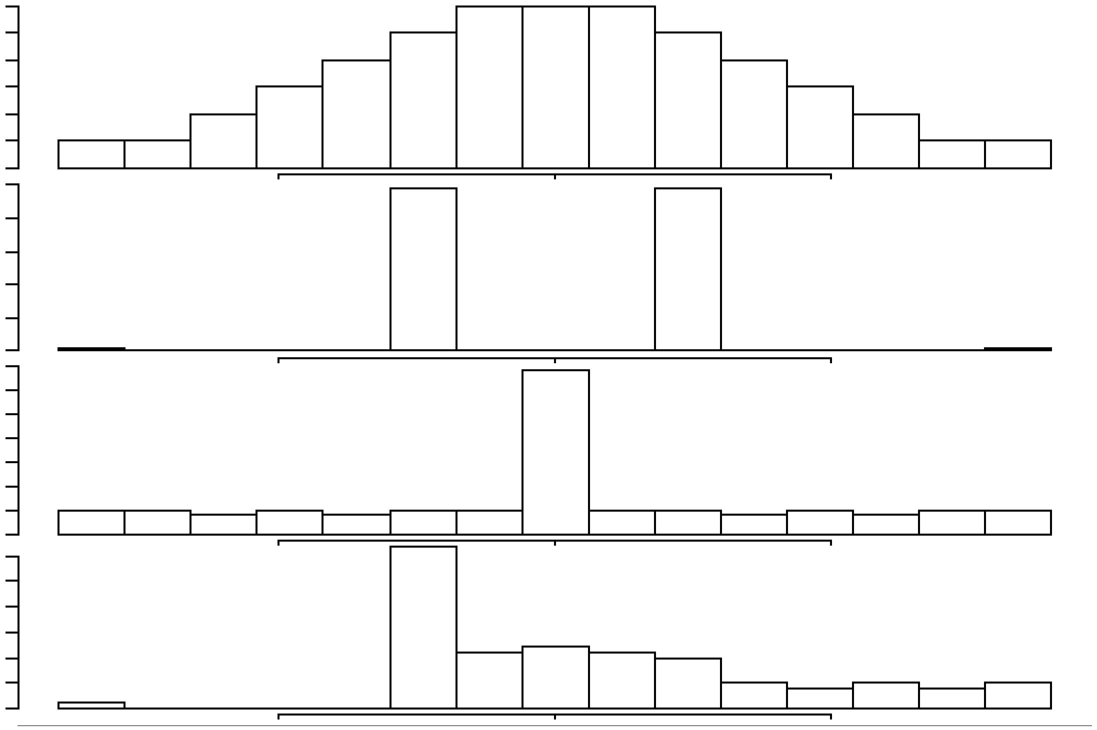
Often one wants to compare quantitative variables across categories

Side-by-Side graphs are a way to visually compare quantitative variables across different categories




```
plt.boxplot([data1, data2],   labels = ["name 1", "name 2"])
```

Box plots don’t capture everything



Do you think the box plots for these distributions look similar?

Let's explore side-by side boxplots on the Bechdel data to try to see if movies that pass the Bechdel test make a larger profit!

Higer dimensional arrays

Higher dimensional arrays

We can make higher dimensional arrays

- (matrices and tensors)

  my_matrix = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

  my_matrix

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

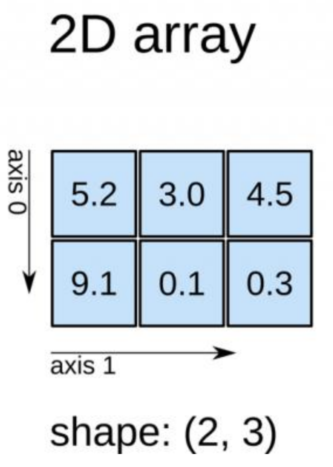
We can slice higher dimensional array

- my_matrix[0:2, 0:2]

We can apply operations to rows, columns, etc.

- np.sum(my_matrix, axis = 0)   # sum the values down rows

Let's explore this in Jupyter!



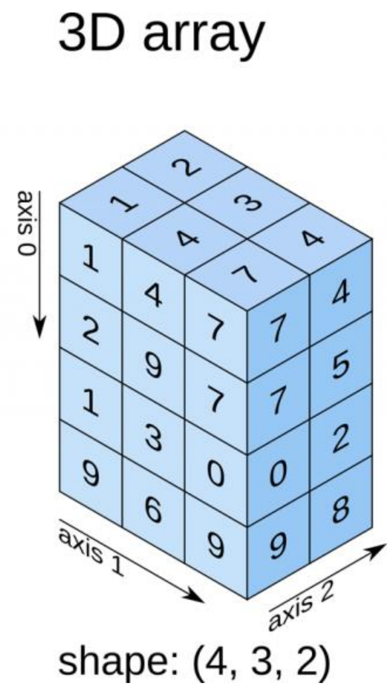


Image processing

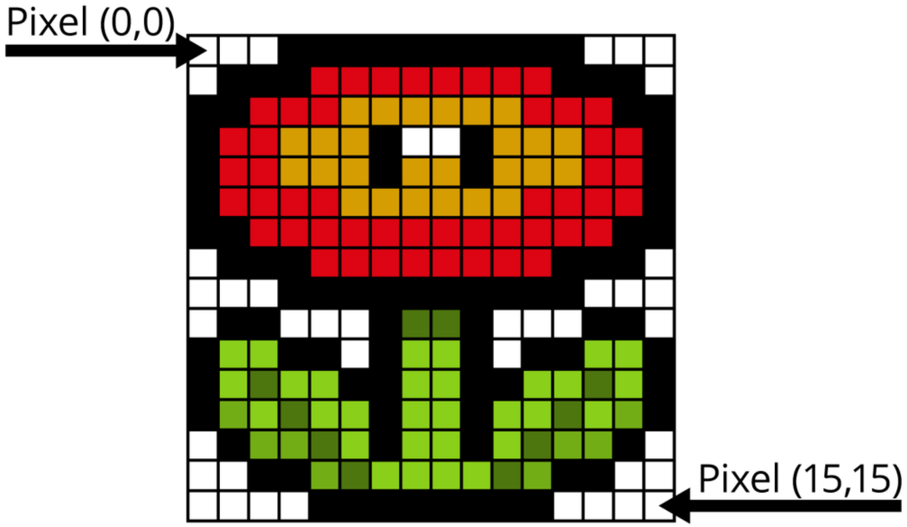
Image processing

We can use higher dimensional numpy arrays to store and manipulate images

Digital images are made up of pixels

Each pixel consists of a red (R), green (G), and Blue (B) color channel

- i.e., we have an "RGB image"
- See the [RGB color picker]



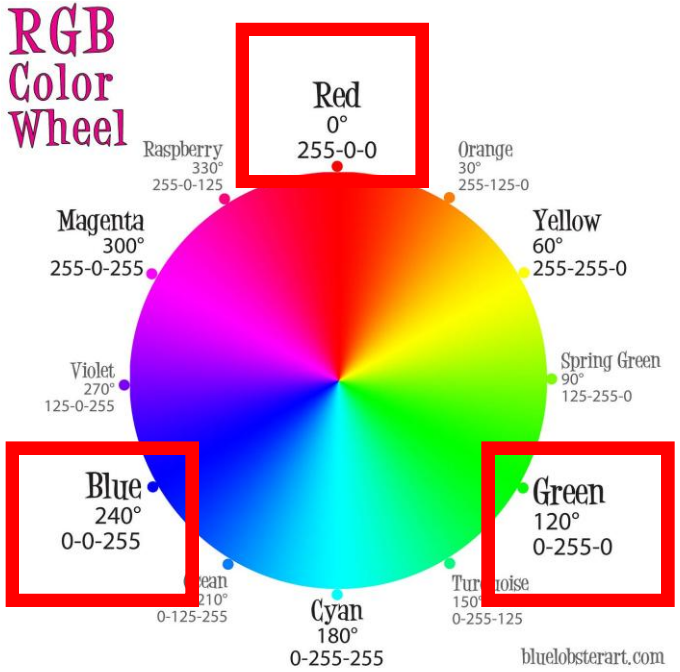
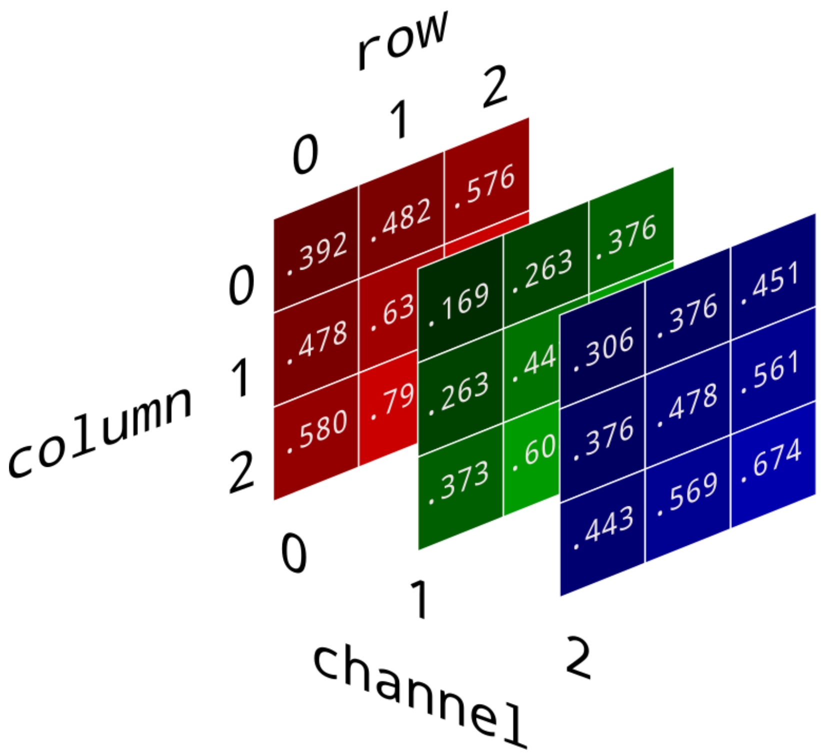
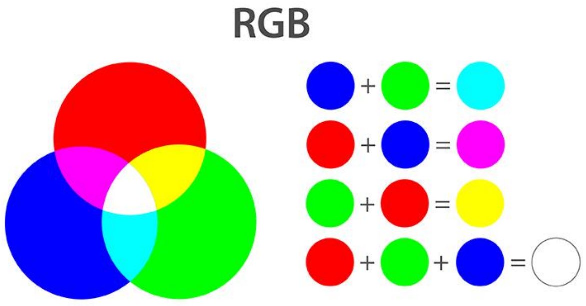


Image processing

We can use 3-dimemsional numerical arrays to store digital RGB images

We can use masking and other array operations to process images





Let's explore this in Jupyter!

Tuples

Tuples

Tuples are like lists but they are immutable; i.e., once they are created we can't change the values in a tuple.

We can create a tuple using:

- my_tuple = (10, 20, 30)

Like lists, we can access elements of tuples using square brackets

- my_tuple[1]

We can't change values in tuples:

- my_tuple[1] = 50   # Error!!!

Tuples

We can assign values in tuples into regular names using “tuple unpacking”

```
my_tuple = (10, 20, 30)

val1, val2, val3 = my_tuple

val3
```

Let's explore this in Jupyter!

Dictionaries

Dictionaries

Dictionaries allow you to look up ***values*** based on a ***key***

- i.e., you supply a “key” and the dictionary returns the stored value

We can create dictionaries using the syntax:

- my_dict = { 'key1': 5, 'key2': 20}

We can retrieve dictionary values by supplying a key using square brackets []

- my_dict['key2']

Let's explore this in Jupyter!