

YData: Introduction to Data Science



Class 06: Array computations

Overview

Brief review of statistics and visualizations

NumPy arrays

- Creating arrays
- Array computations
- Boolean masking
- If there is time: Percentiles and boxplots




Announcement

Homework 3 has been posted!

It is due on Gradescope on **Sunday September 22nd at 11pm**


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

Alison Bechdel is giving a talk on Wednesday September 18th at 4pm in this classroom

 THE CHUBB FELLOWSHIP • TIMOTHY DWIGHT COLLEGE • YALE UNIVERSITY


ALISON BECHDEL

PROFESSOR IN THE PRACTICE
ENGLISH DEPARTMENT AND FILM & MEDIA STUDIES



*"...and I became a lesbian cartoonist."
Reflections on a Curious Career*

LECTURE AND Q&A
Wednesday, September 18, 2024
4:00-5:30PM
Sheffield-Sterling-Strathcona Hall,
1 Prospect Street, Room 114
Doors open at 3:30 for seating

Admission is free to the Yale Community and the General Public
 Please register here if you plan to attend. No tickets required
Supported by the Department of English and the Program of Film and Media Studies
Please direct questions to:
chubb.fellowship@yale.edu

Assignments and grades

Homework policies

- You can use chatGPT to answer general questions about Python and/or statistical concepts
 - E.g., It is ok to ask chatGPT “What does the np.sum() function in Python do?”
- You are **not** allowed to use it to directly answer homework questions
 - E.g., It is **not** ok to cut and paste a homework questions into chatGPT



Quick review: categorical data

Categorical data

$$\text{Proportion} = \frac{\text{number in category}}{\text{total number}}$$

```
bechdel.count("PASS")/len(bechdel)
```

```
import matplotlib.pyplot as plt
```

```
plt.bar(labels, data)
```

```
plt.pie(data)
```

Quick review: one quantitative variable data

Basics statistics and plots:

`plt.hist(data)`

`statistics.mean(data)`

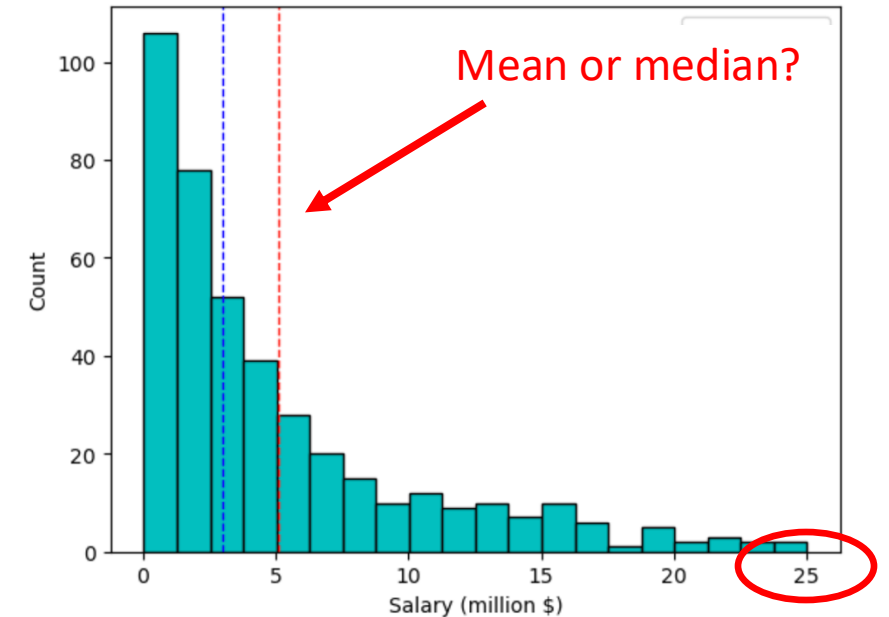
`statistics.median(data)`

$$= \frac{1}{n} \sum_{i=1}^n x_i$$

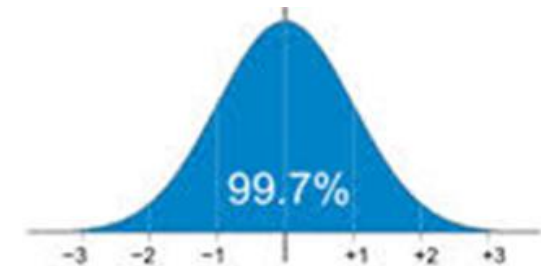
Quantitative data spread

`statistics.stdev(data)`

$$s = \sqrt{\frac{1}{(n-1)} \sum_{i=1}^n (x_i - \bar{x})^2}$$



$$\text{z-score}(x_i) = \frac{x_i - \bar{x}}{s}$$



Outliers

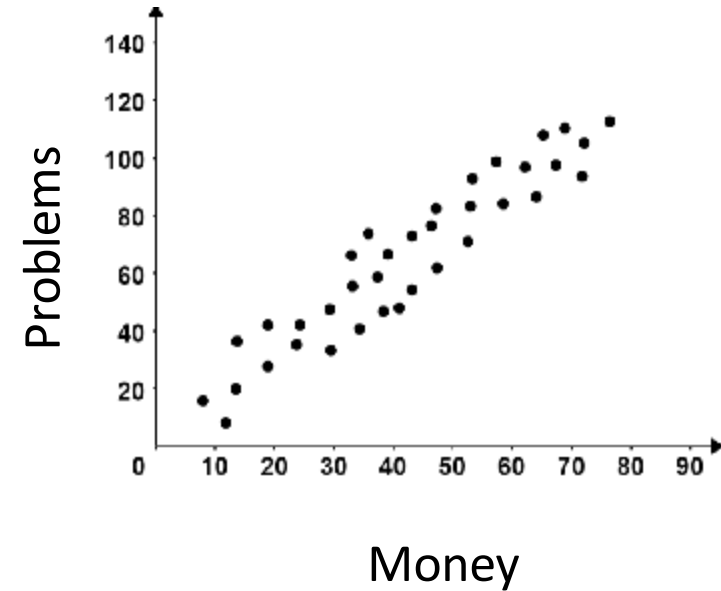
Quick review: two quantitative variables data

Basics statistics and plots:

```
plt.plot(x, y, '.')
```

```
statistics.correlation(x, y)
```

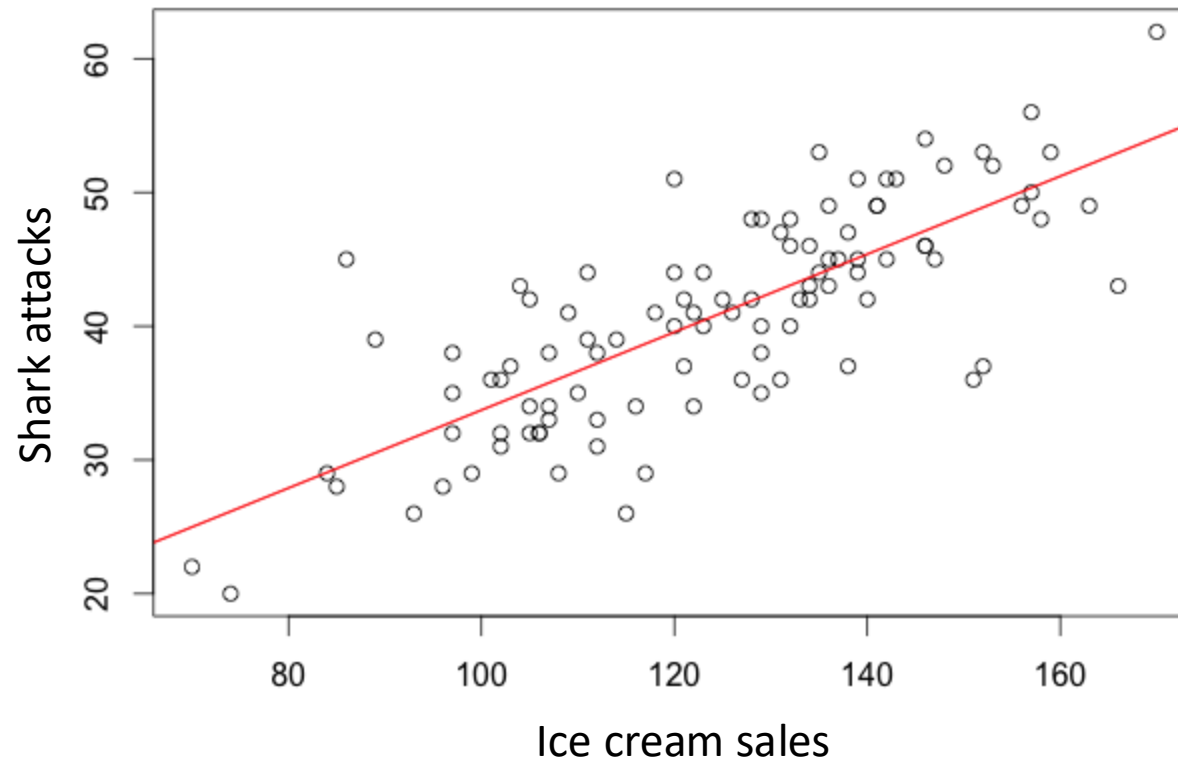
$$r = \frac{1}{(n-1)} \sum_{i=1}^n \left(\frac{x_i - \bar{x}}{s_x} \right) \left(\frac{y_i - \bar{y}}{s_y} \right)$$



- Correlation is always between -1 and 1: $-1 \leq r \leq 1$
- The sign of r indicates the direction of the association
- Values close to ± 1 show strong linear relationships, values close to 0 show no linear relationship
- Correlation is symmetric: $r = \text{cor}(x, y) = \text{cor}(y, x)$

Correlation caution #1

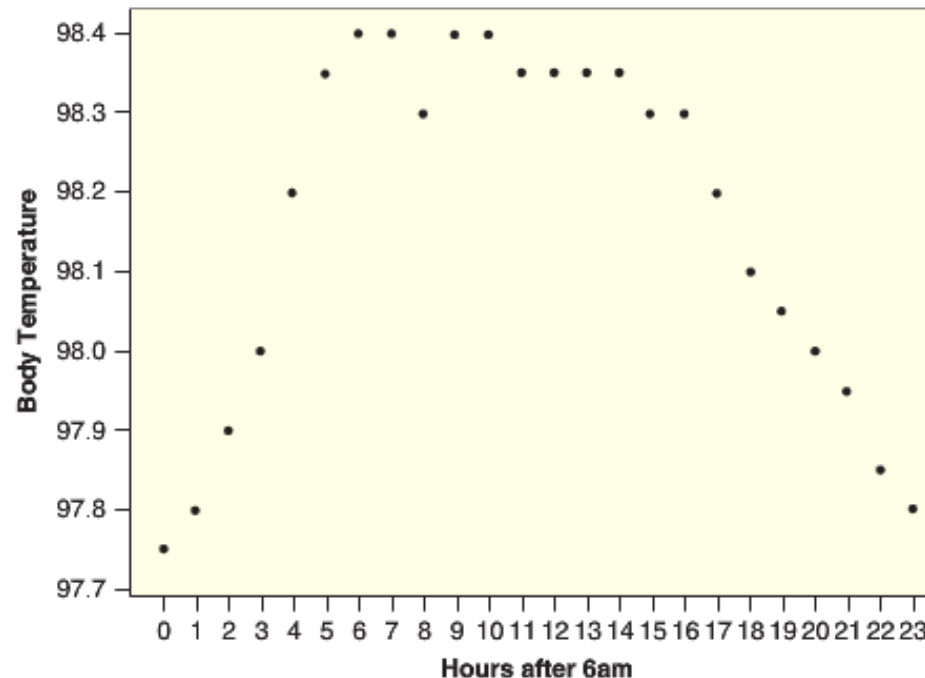
A strong positive or negative correlation does not (necessarily) imply a cause and effect relationship between two variables



Correlation caution #2

A correlation near zero does not (necessarily) mean that two variables are not associated. Correlation only measures the strength of a linear relationship.

Example: Body temperature as a function of time of the day



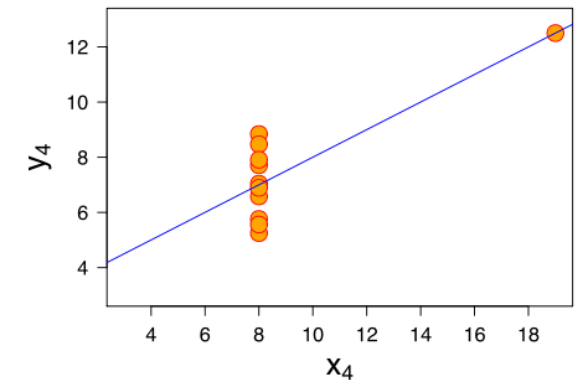
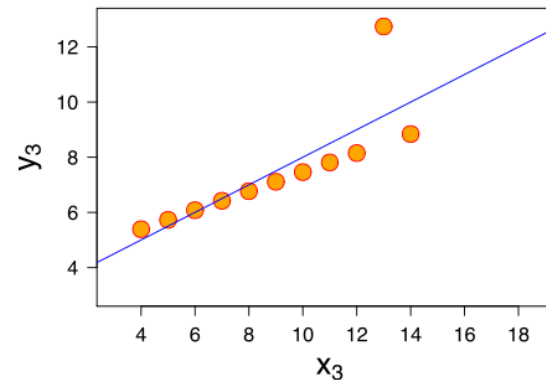
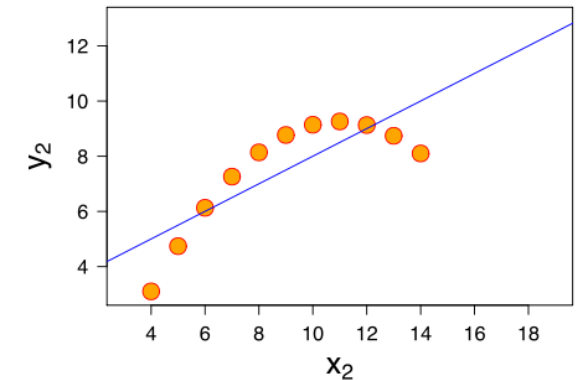
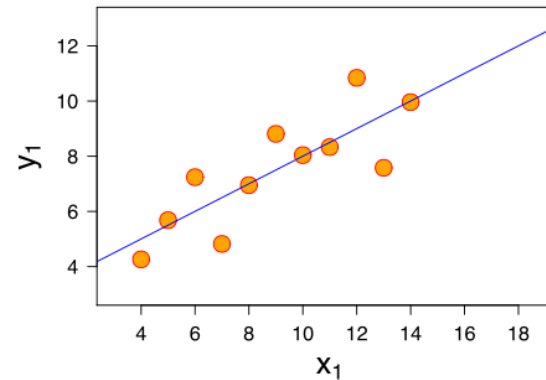
$r = -0.08$

Correlation caution #3

Correlation can be heavily influenced by outliers. Always plot your data!

Example:
“Anscombe’s quartet”

$r = 0.81$ for all plots



Warm-up exercises: NBA salaries

Let's practice analyzing data by explore salaries of NBA players

- (from the 2022-2023 season)



Player Name	Salary	Position	Team	PTS
Stephen Curry	48070014	PG	GSW	29.4
John Wall	47345760	PG	LAC	11.4
Russell Westbrook	47080179	PG	LAL/LAC	15.9
LeBron James	44474988	PF	LAL	28.9
Kevin Durant	44119845	PF	BRK/PHO	29.1
Bradley Beal	43279250	SG	WAS	23.2
Kawhi Leonard	42492492	SF	LAC	23.8
Paul George	42492492	SF	LAC	23.8
Giannis Antetokounmpo	42492492	PF	MIL	31.1
Damian Lillard	42492492	PG	POR	32.2

Array computations

Arrays

Often, we are processing data that is all of the same type

- For example, we might want to do processing on a data set of numbers
 - e.g., if we were analyzing salary data

When we have data that is all of the same type, there are more efficient ways to process data than using a list

- i.e., methods that are faster and take up less memory

In Python, the *NumPy package* offers ways to store and process data that is all of the same type using a data structure called a ***ndarray***

There are also functions that operate on ndarrays that can do computations very efficiently.



ndarrays

We can import the NumPy package using: `import numpy as np`

We can then create an array by passing a list to the `np.array()` function

- `my_array = np.array([1, 2, 3])`

We can get elements of an array using similar syntax as using a list

- `my_array[1]` `# what does this return`

ndarrays have properties that tell us the type and size

- `my_array.dtype` `# get the type of elements stored in the array`
- `my_array.shape` `# get the dimension of the array`
- `my_array.astype('str')` `# convert the numbers to strings`
- `sequential_nums = np.arange(1, 10)` `# creates numbers 1 to 9`

Let's explore this in Jupyter!

NumPy functions on numerical arrays

The NumPy package has a number of functions that operate very efficiently on numerical ndarrays

- `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`

FRED: Federal Reserve Economic Data

[Federal Reserve Economic Data \(FRED\)](#) is a database maintained by the Research division of the Federal Reserve Bank of St. Louis that has more than 816,000 economic time series from various sources.



They cover:

- E.g., Consumer price indexes
- Employment and population
- Gross domestic product
- Producer price indexes
- Etc.

We can read this data into Python using `pandas_datareader`

```
from pandas_datareader.fred import FredReader
```

```
FredReader("GASREGW").read()
```

Let's explore numpy functions using the [price of gas](#) from FRED!

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

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, 4, 6, 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

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

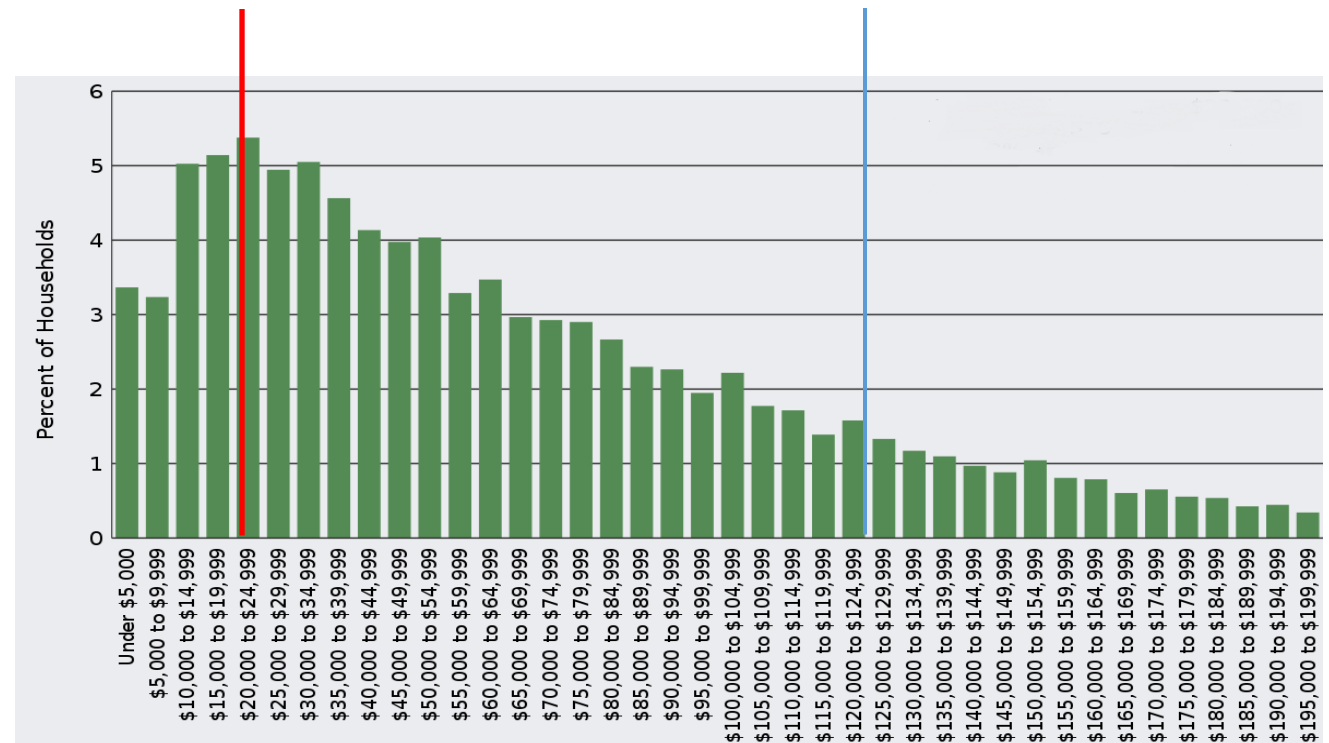
For the US income distribution what are the 20th and 80th percentiles?

We can calculate percentiles using `np.percentile()`

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

20th percentile = \$21,430

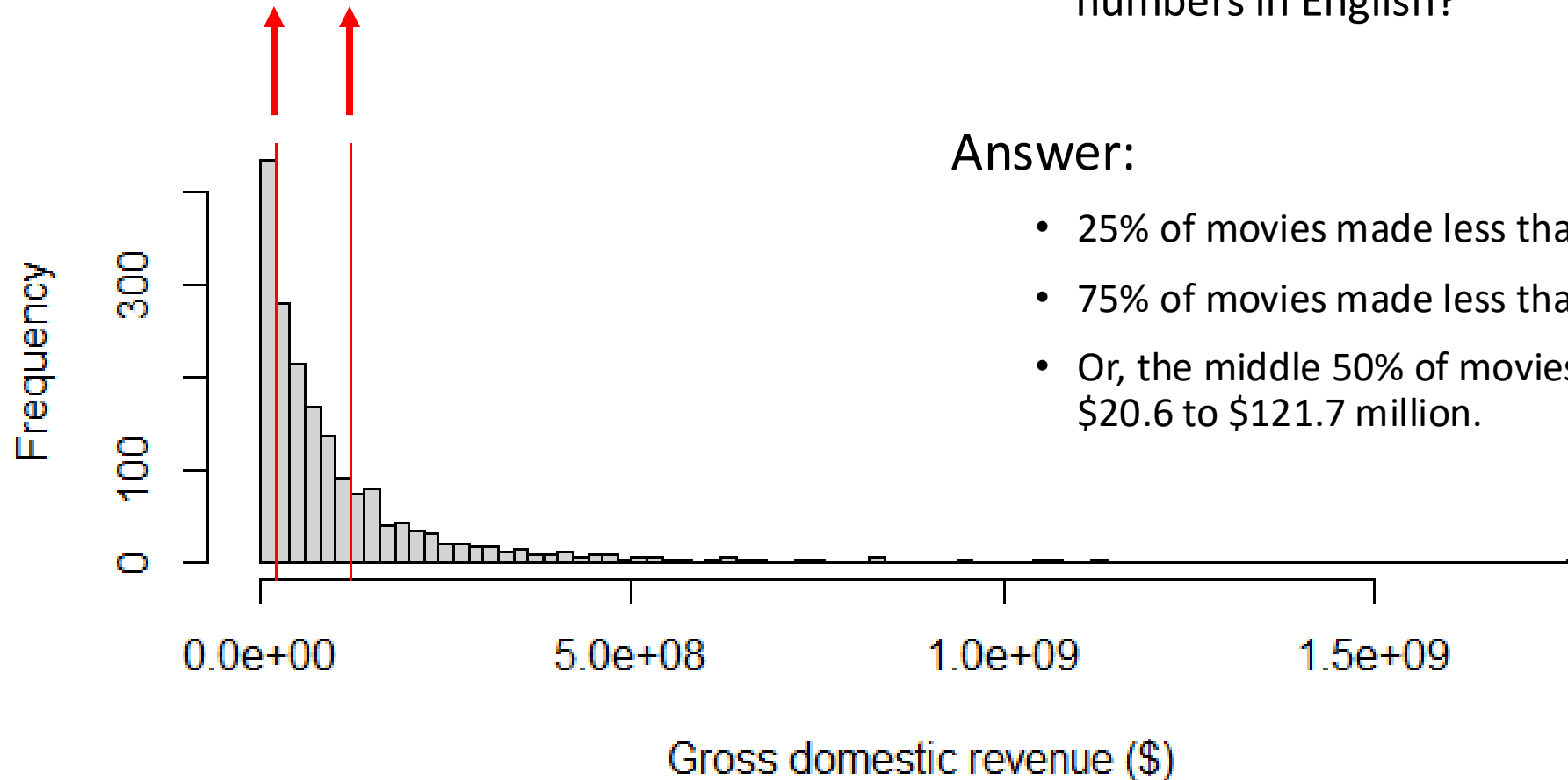
80th percentile = \$112,254



Movie revenue percentiles

25th percentile
= \$20.6 milion

75th percentile
= \$121.7 milion



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: $Q_1 - 1.5 * IQR$

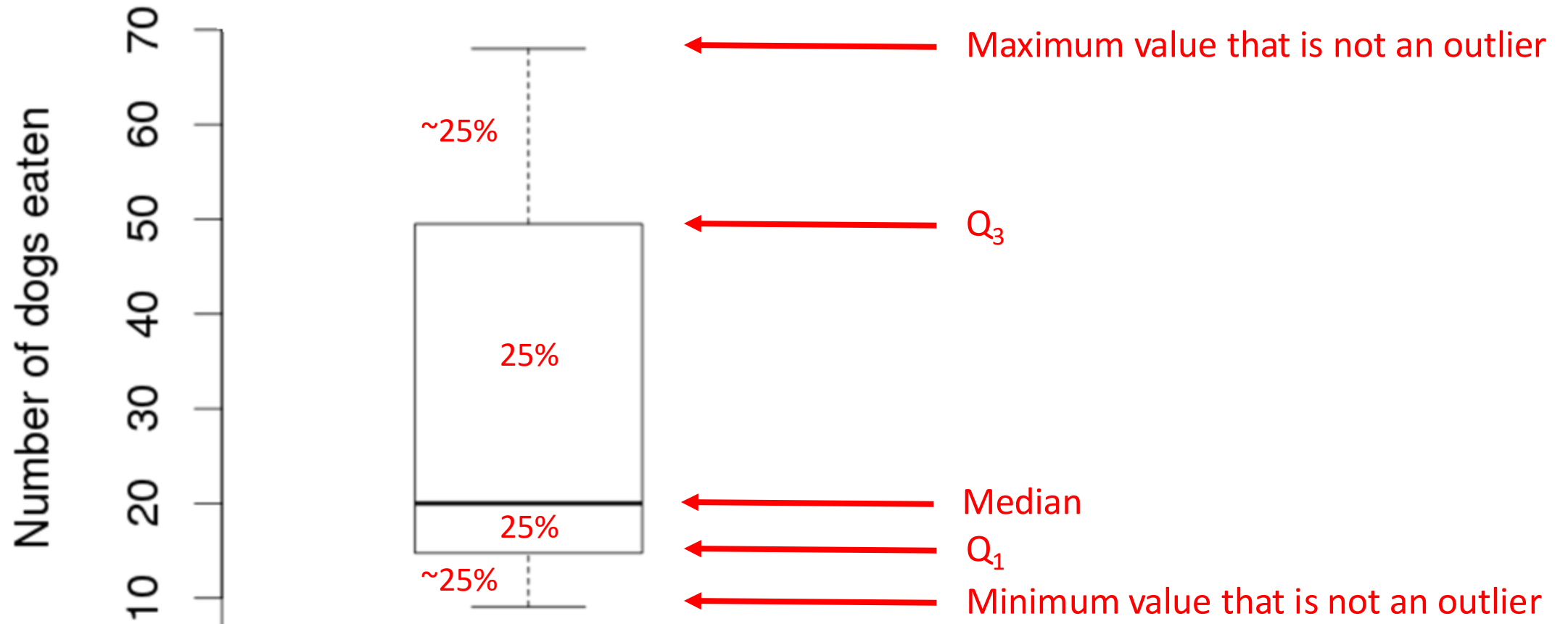
Larger than: $Q_3 + 1.5 * IQR$

Box plots

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

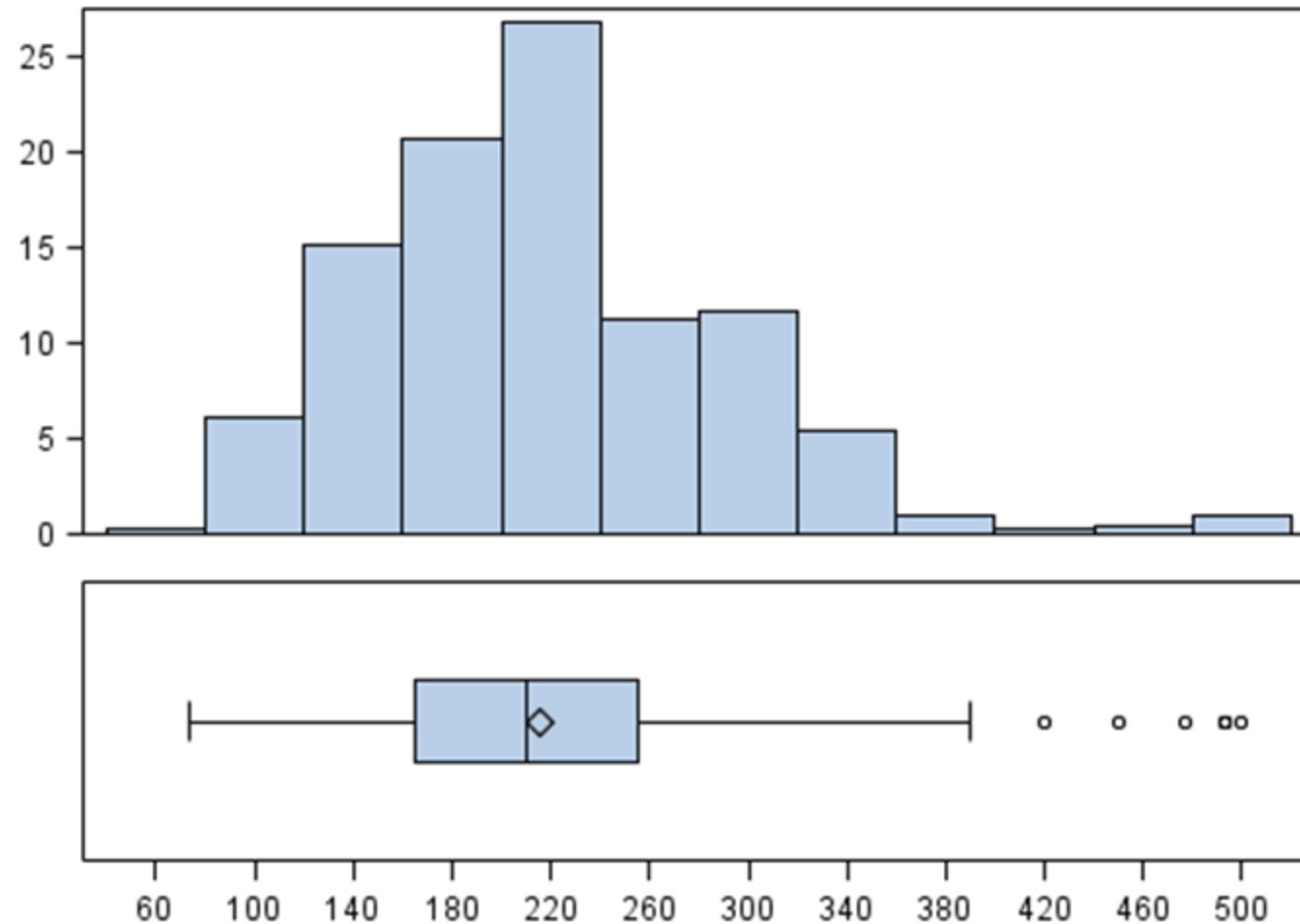
1. Drawing a box from Q_1 to Q_3
2. Dividing the box with a line (or dot) drawn at the median
3. Draw a line from each quartile to the most extreme data value that is not and outlier
4. Draw a dot/asterisk for each outlier data point.

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



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

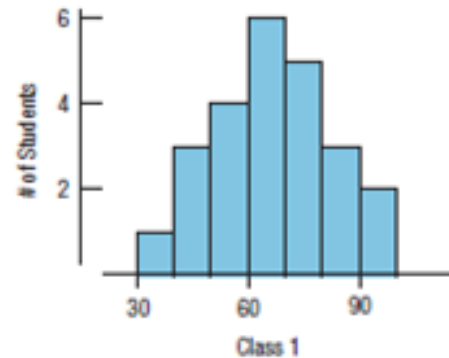
Box plots extract key statistics from histograms



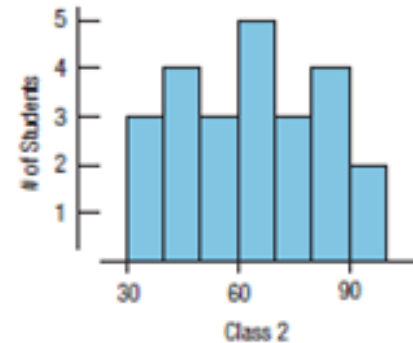
Box plots extract key statistics from histograms

Question: which Box plot goes with which histogram?

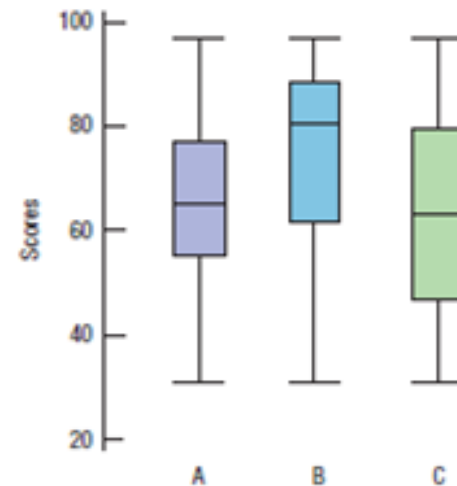
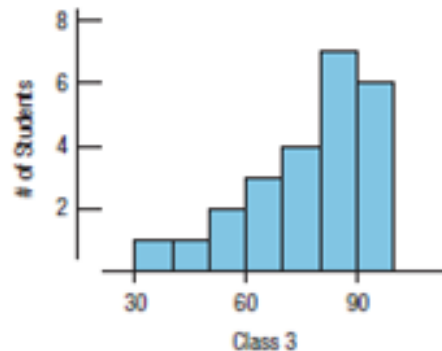
Histogram 1



Histogram 2



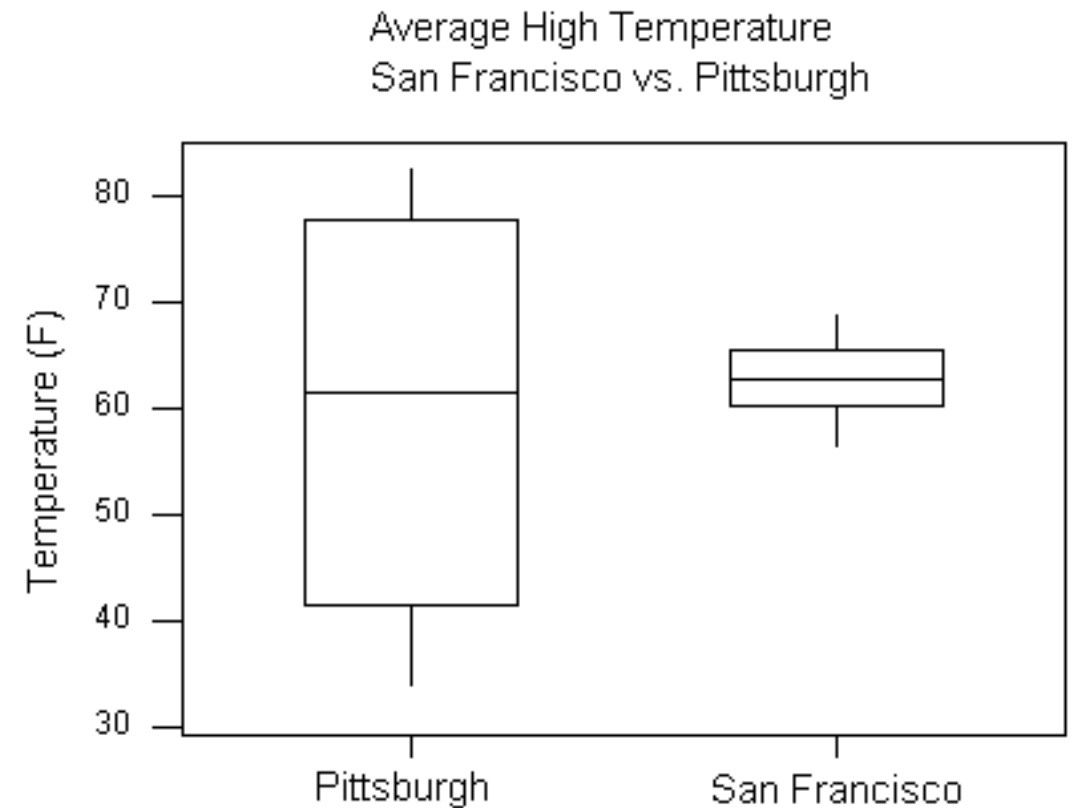
Histogram 3



Comparing quantitative variables across categories

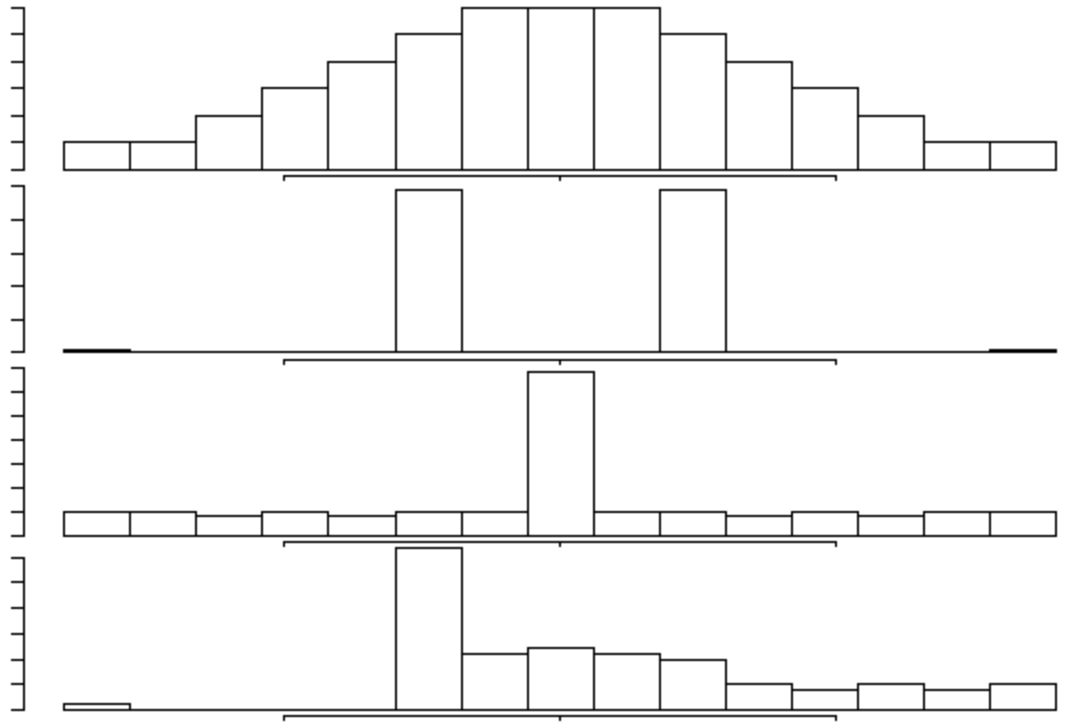
Often one wants to compare quantitative variables across categories

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!