Data Analysis – Analyzing and Visualizing

15-110 – Monday 11/18

Please sit next to someone today!

Announcements

- Check6-1 was due Monday
  - How did it go?
- Updates on Check6-2
  - See Piazza post
- Prof OH
  - OH cancelled today
  - Go to Prof. Taylor's office for tomorrow and Wednesday times

Learning Goals

- Perform **basic analyses** on data, including calculating **statistics** and **probabilities**, to answer simple questions

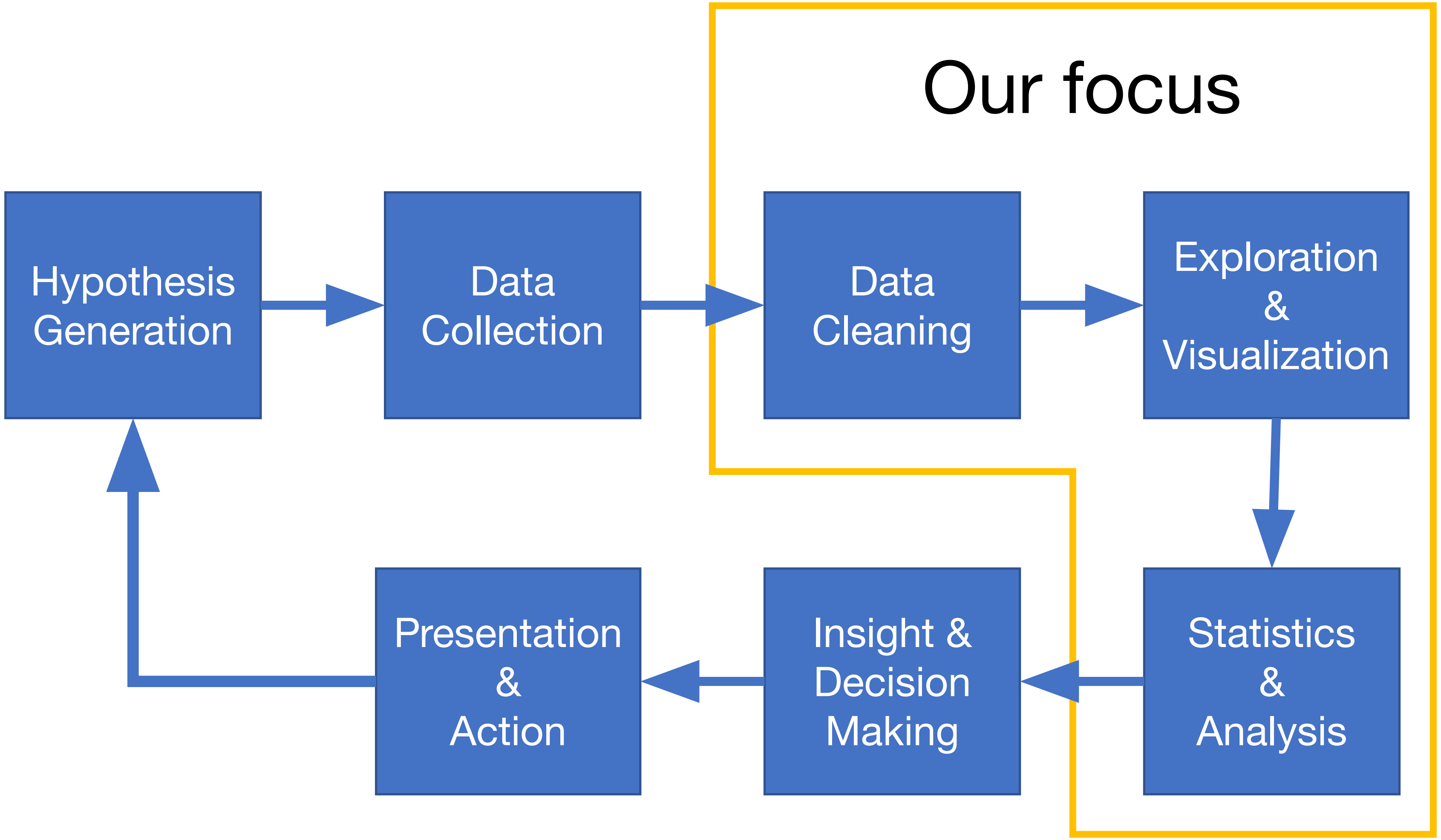
- Choose an appropriate **visualization** to create based on the number of **dimensions** and **data types**

- Create simple **matplotlib visualizations** that show the state of a dataset

Last week we discussed data processing and cleaning: processing data and putting it into formats and structures we want.

We'll focus mainly on three steps:

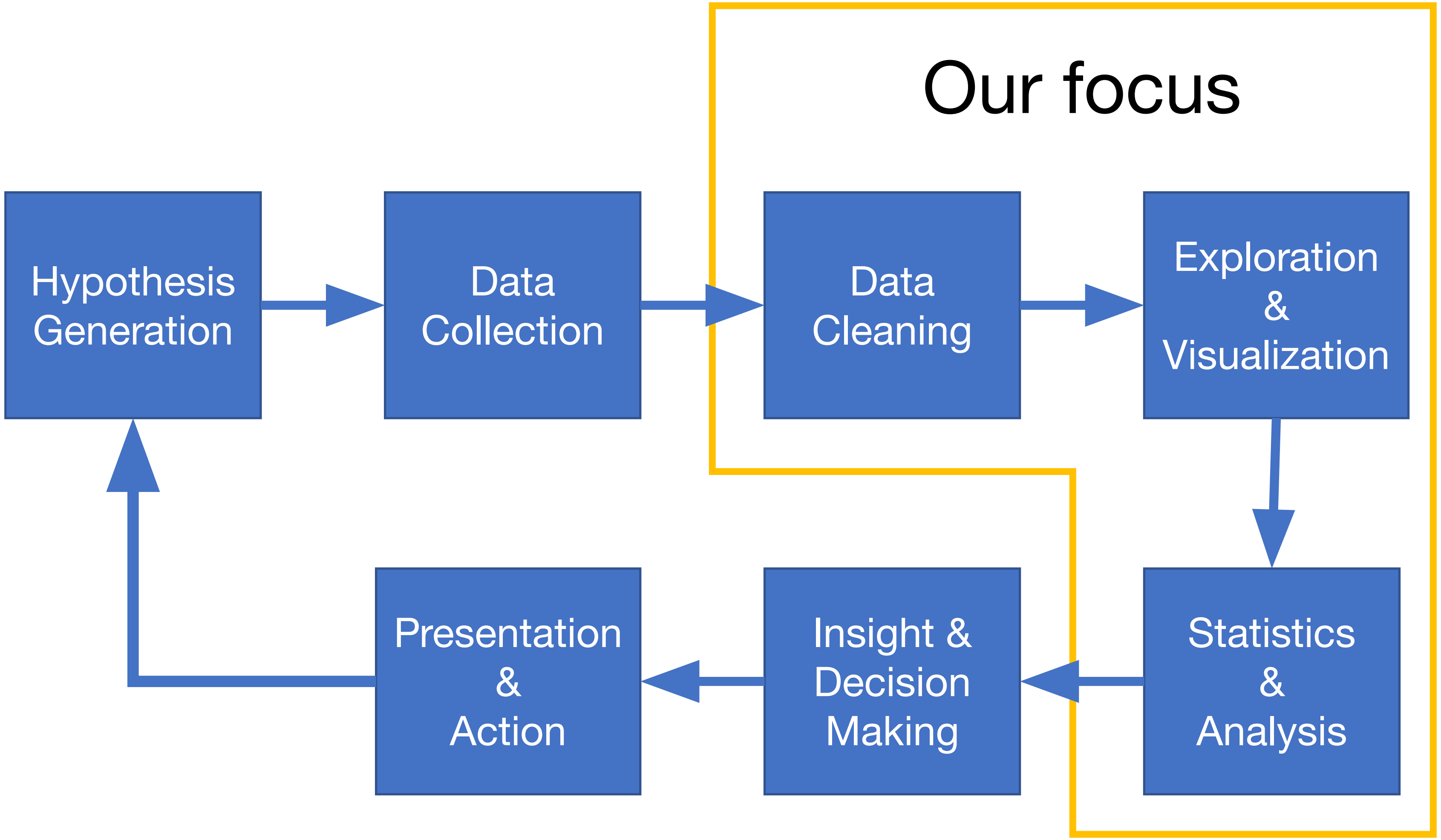
1. **Data Cleaning**
2. Exploration & Visualization
3. Statistics & Analysis



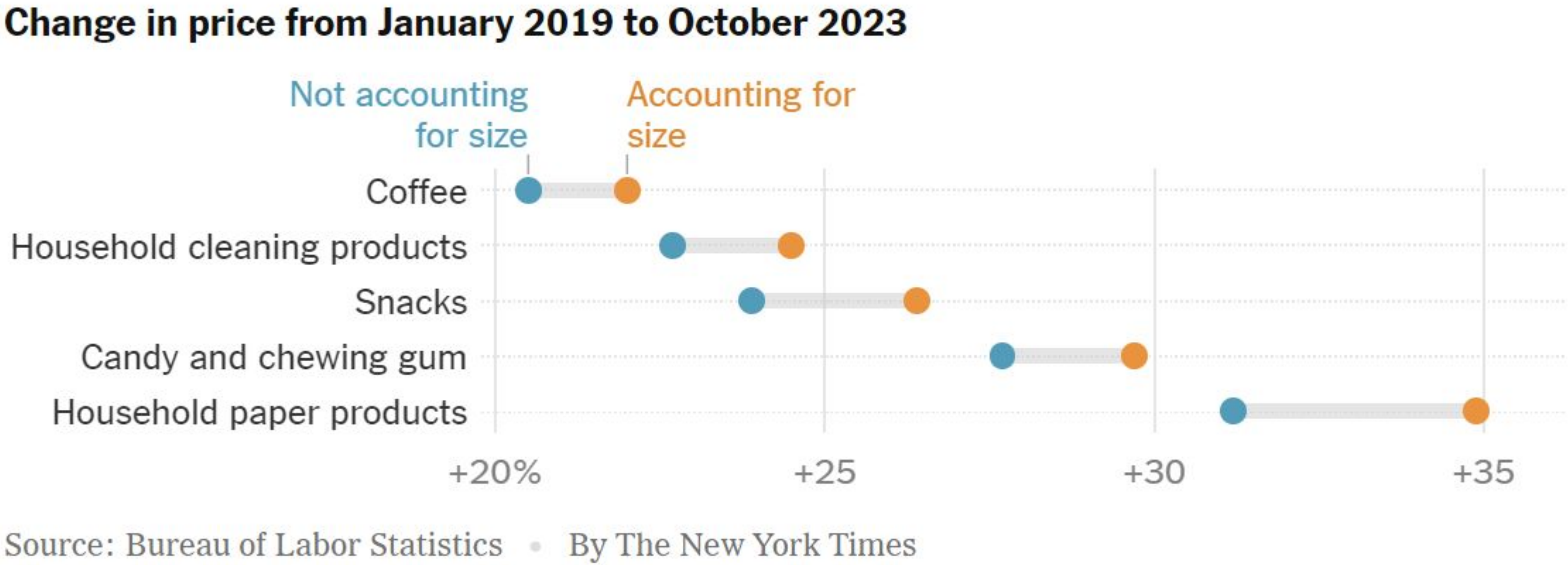
This lecture we will learn what we can **do** with that data once we have processed it.

We'll focus mainly on three steps:

1. Data Cleaning
2. **Exploration & Visualization**
3. **Statistics & Analysis**



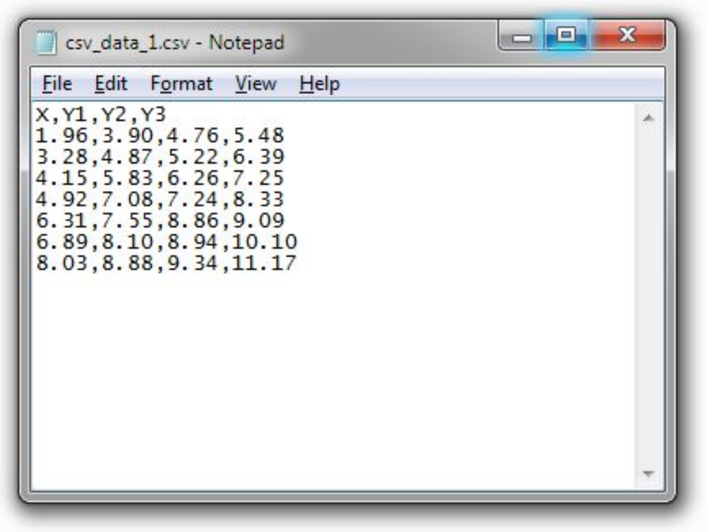
How are the prices and sizes of some consumer items changing? [reference]



Source: Bureau of Labor Statistics • By The New York Times

What’s something we can conclude from this graph?

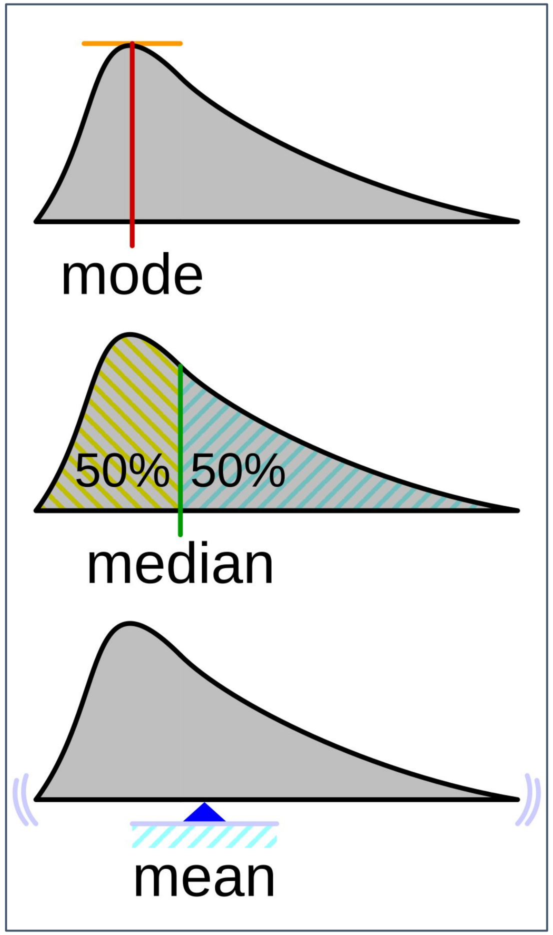
Topics covered today:



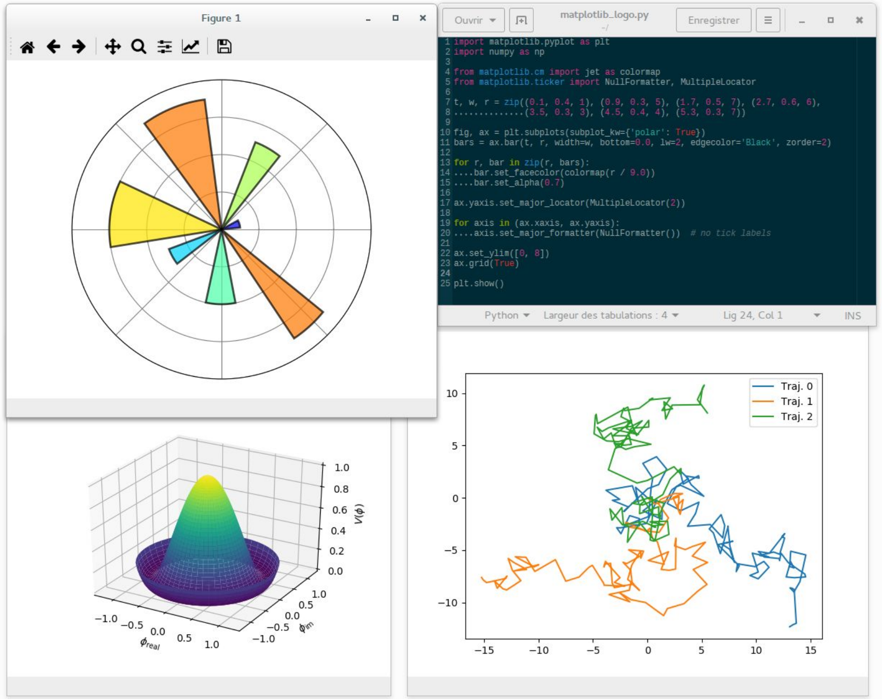
```
X,Y1,Y2,Y3
1.96,3.90,4.76,5.48
3.28,4.87,5.22,6.39
4.15,5.83,6.26,7.25
4.92,7.08,7.24,8.33
6.31,7.55,8.86,9.09
6.89,8.10,8.94,10.10
8.03,8.88,9.34,11.17
```

Refresher on data processing and CSVs

`statistics` library



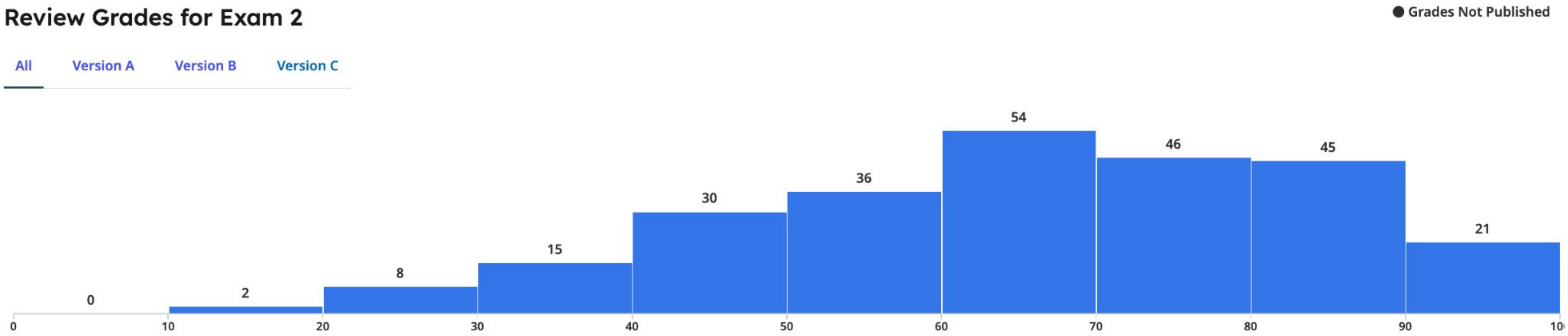
`matplotlib` library



```
import matplotlib.pyplot as plt
import numpy as np

from matplotlib.cm import jet as colormap
from matplotlib.ticker import NullFormatter, MultipleLocator

t, w, r = zip((0.1, 0.4, 1), (0.9, 0.3, 5), (1.7, 0.5, 7), (2.7, 0.6, 6),
..............(3.5, 0.3, 3), (4.5, 0.4, 4), (5.3, 0.3, 7))

fig, ax = plt.subplots(subplot_kw={'polar': True})
bars = ax.bar(t, r, width=w, bottom=0.0, lw=2, edgecolor='Black', zorder=2)

for r, bar in zip(r, bars):
....bar.set_facecolor(colormap(r / 9.0))
....bar.set_alpha(0.7)

ax.yaxis.set_major_locator(MultipleLocator(2))

for axis in (ax.xaxis, ax.yaxis):
....axis.set_major_formatter(NullFormatter())  # no tick labels

ax.set_ylim([0, 8])
ax.grid(True)

plt.show()
```

Refresher: CSVs

Exam 2 didn’t go great for many of you….



Our dataset for this lecture: Exam Reflections (anonymized)

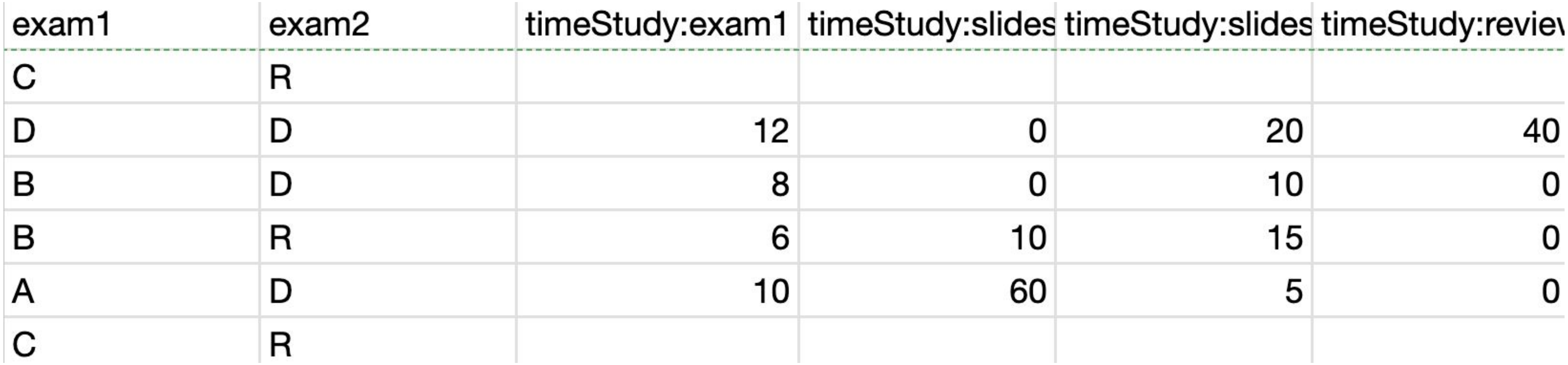
Exam reflections are a pedagogical tool for metacognition: helping students learn how to learn.

But are they actually helpful for students in this class to better prepare for exams?

Example questions:

- Did the students who got an A on Exam 2, complete the Exam 1 reflection?
- Did students who complete the Exam 1 reflection, do better on Exam 1 than Exam 2?
- What is the relationship between completing the Exam 1 reflection and Exam 2 grades?

The structure of the dataset (linked on the course schedule):



exam1	exam2	timeStudy:exam1	timeStudy:slides	timeStudy:slides	timeStudy:reviev
C	R				
D	D	12	0	20	40
B	D	8	0	10	0
B	R	6	10	15	0
A	D	10	60	5	0
C	R				

Some guarantees:

- Empty cells are empty strings
- Correctly formatted CSV (no extra commas)
- Every row has an exam1 and exam2 grade (not real grades)

Refresher: What type of data are letter grades A, B, C, D, R?

[1]: Categorial

[2]: Ordinal

[3]: Numerical

When reading CSVs and converting columns to numbers, we often need to decide what do with NaN (not-a-number) values.

For example: When you load data into a CSV, the cells with no data are empty strings.

Some options:

- Fill in NaN values with something else
- **Drop elements with NaN values**

```
def convertToFloat(columnData):
    floats = []
    for num in columnData:
        # need to deal with
        # empty strings
        if num != '':
            floats.append(float(num))
    return floats
```

Statistical Analysis

Python has a built-in the `statistics` library that we can use to compute some descriptive statistics.

```
import statistics


data = [41, 65, 64, 50, 45, 13, 29, 14, 7, 14]


statistics.mean(data) # 34.2
statistics.median(data) # 35.0
statistics.mode(data) # 14
```

Documentation: https://docs.python.org/3/library/statistics.html

We can answer some simple questions with `statistics` library:

Question: Did people spend more time studying for Exam 1 or Exam 2?

```
medianExam1 = statistics.median(timeStudyExam1)
medianExam2 = statistics.median(timeStudyExam2)
```

Question: Did students do better on Exam 1 or Exam 2?

```
modeExam1 = statistics.mode(exam1Grade)
modeExam2 = statistics.mode(exam2Grade)
```

We can also compute basic probabilities from our data sets.

Probability P(item): count of how often it occurred, divided by the total number of data points

```
lst.count(item) / len(lst)
```

Conditional Probability P(item | property):
create a modified version of the list that contains only those elements with that property, then you can use the same equation.

```
newLst = []
for x in lst:
    if meetsProperty(x):
        newLst.append(x)
newLst.count(item) / len(newLst)
```

Question: What is the probability that a student did Exam 2 Reflection, given they did Exam 1 Reflection?

```
# P (exam2 | exam1) = P (exam1 & exam2) / P (exam 1)
countBoth = 0
for i in range(len(didExam1Reflection)):
    if didExam1Reflection[i] == 'True' and didExam2Reflection[i] == 'True':
        countBoth += 1
condProb = countBoth/didExam1Reflection.count('True')
```

Question: What is the probability that a student scored an A on Exam 2, given they did the Exam 1 reflection?

You do: What is the code for this?

```
# P (exam2 A | exam1) = P (exam1 & exam2 A) / P (exam 1)
countBoth = 0
for i in range(len(didExam1Reflection)):
    if  BLANK 1  :
        countBoth += 1
condProb = countBoth / BLANK 2
```

There many other kinds of statistical analysis you can do that are out of scope for this lecture.

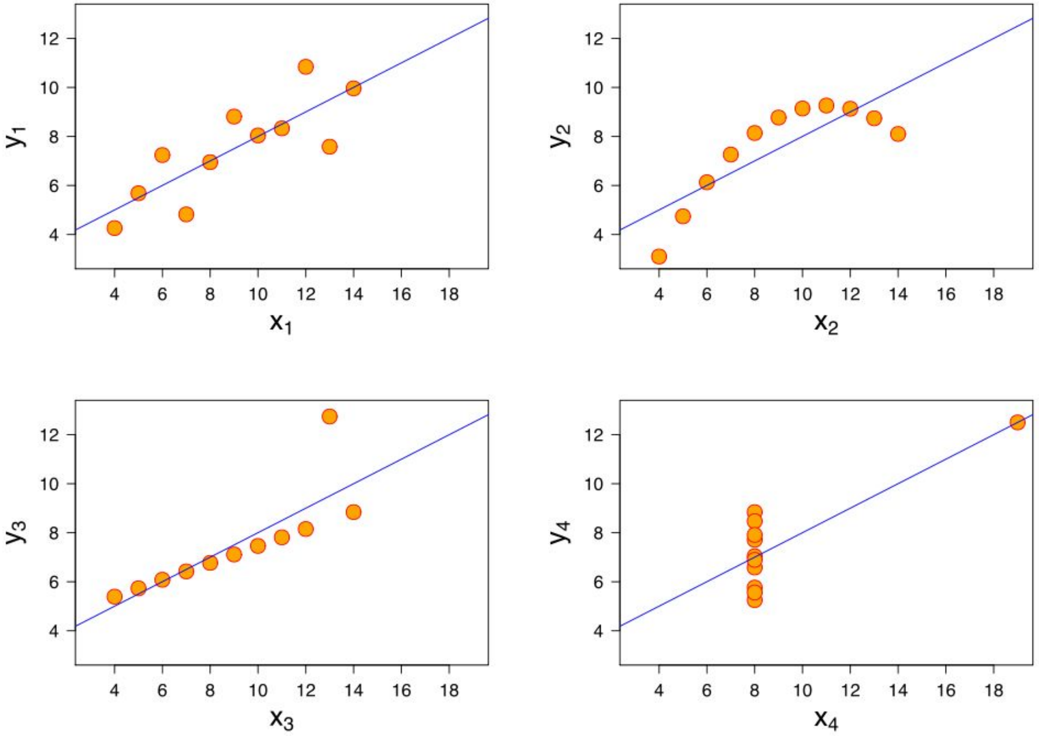
This lecture covers some basics from the `statistics` library, but there are all kinds of other analyses you will want to do depending on the context of the problem you are solving.

There are also external Python modules that have implemented many more algorithms like `numpy` [docs] and `scipy` [docs].

Visualization & Matplotlib

Data visualization is the process of taking a set of data and representing it in a visual format for exploration or presentation.

For example: The four graphs here all have the same mean and best-fit linear regression but all have different patterns.

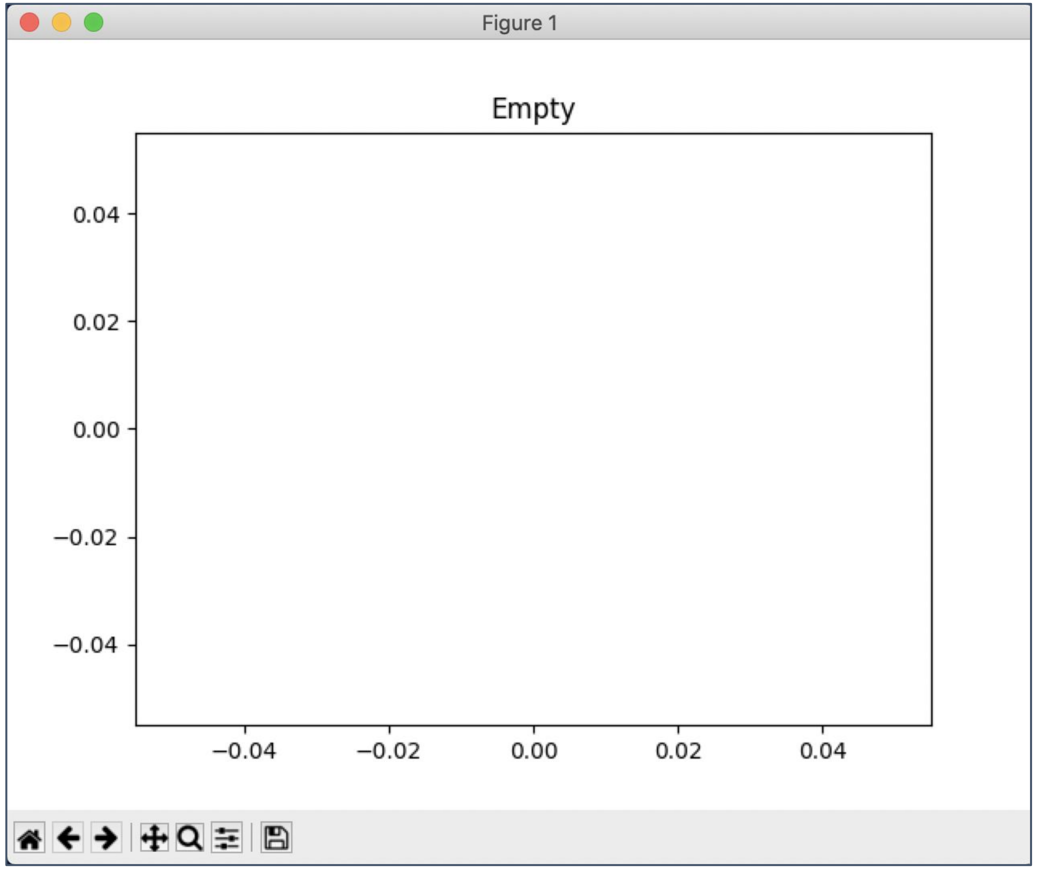


A good data visualization for presentation highlights a pattern or trend and tells a clear story.

In Python we can use the external* `matplotlib` library to draw a wide variety of visualizations.

We can make an empty plot like this:

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()
ax.set_title('Empty')
plt.show()
```

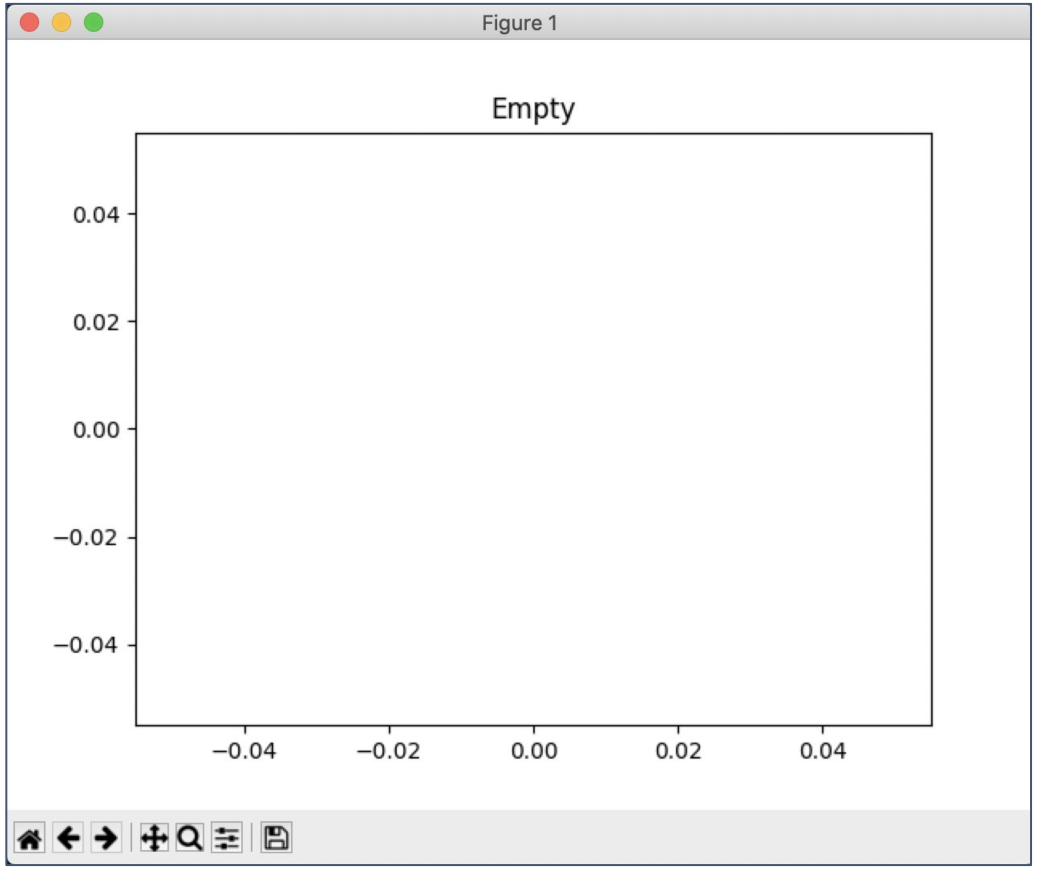


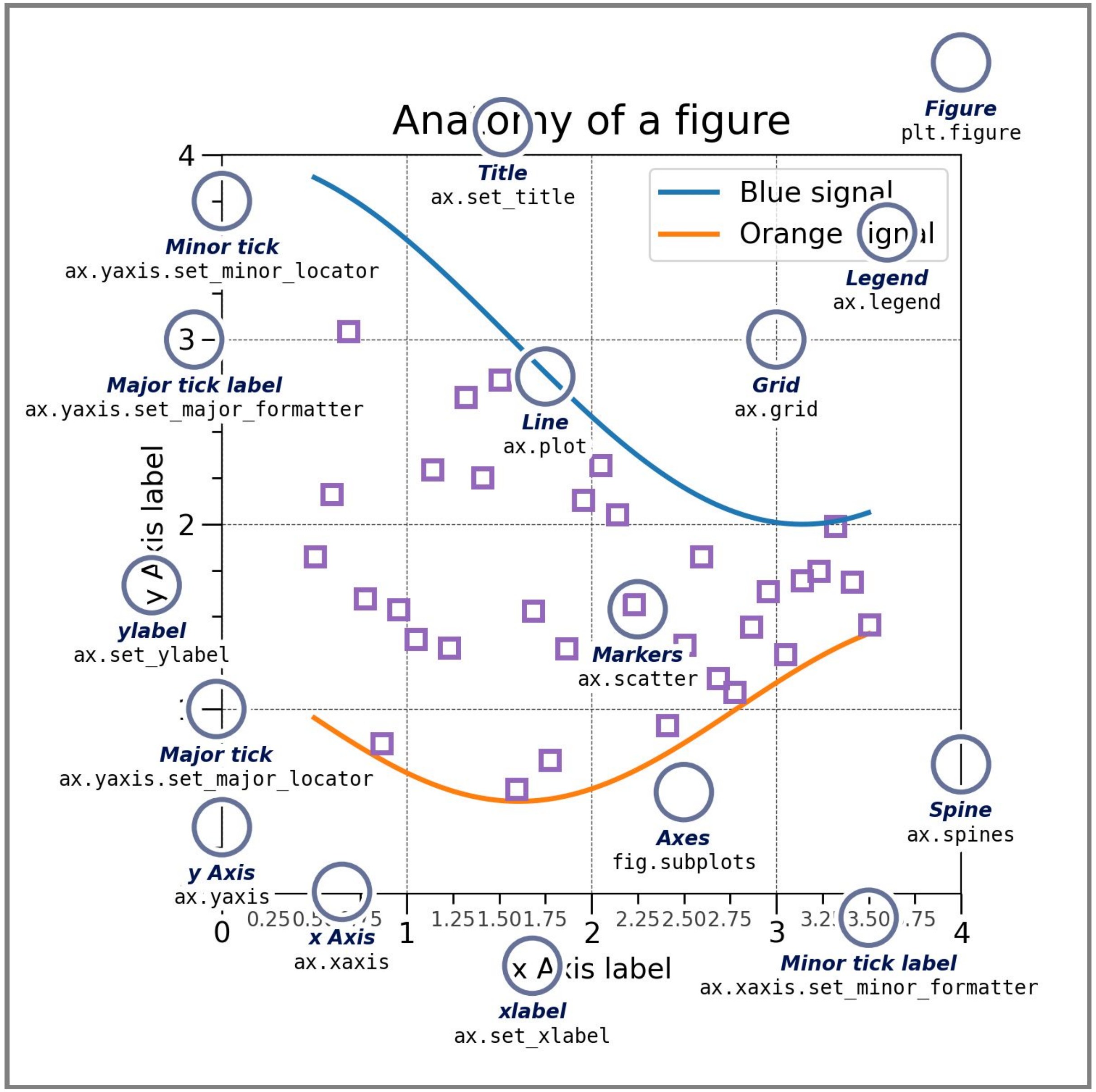
*See slides about installing external libraries.

In Python we can use the external `matplotlib` library to draw a wide variety of visualizations.

We can make an empty plot like this:

```
import matplotlib.pyplot as plt


# make a figure and axes objects
fig, ax = plt.subplots()
# add a title
ax.set_title('Empty')
# show plot
plt.show()
```





Anatomy of a figure
Figure
plt.figure
Title
ax.set_title
Blue signal
Orange signal
Legend
ax.legend
Minor tick
ax.yaxis.set_minor_locator
Major tick label
ax.yaxis.set_major_formatter
Line
ax.plot
Grid
ax.grid
y Axis label
ylabel
ax.set_ylabel
Markers
ax.scatter
Major tick
ax.yaxis.set_major_locator
Spine
ax.spines
Axes
fig.subplots
y Axis
ax.yaxis
x Axis
ax.xaxis
Minor tick label
ax.xaxis.set_minor_formatter
x Axis label
xlabel
ax.set_xlabel
4
3
2
0
1
2
3
4
0.25
0.50
0.75
1.25
1.50
1.75
2.25
2.50
2.75
3.25
3.50
3.75

The **kind of visualization** you want to draw will depend on the data.

When picking a visualization, consider the following questions:

- What is the **question** we want to answer?

- What are the **dimensions** of the data?
    - How many variables are we trying to visualize?
        - One, two, three or more?

- What are the **data types**?
    - Ordinal, categorical, or numerical?

A **one-dimensional visualization** only visualizes a single feature of a dataset.

We want to **visualize the distribution** of the data, which we cannot completely capture with a statistical number like median.

Example Questions:

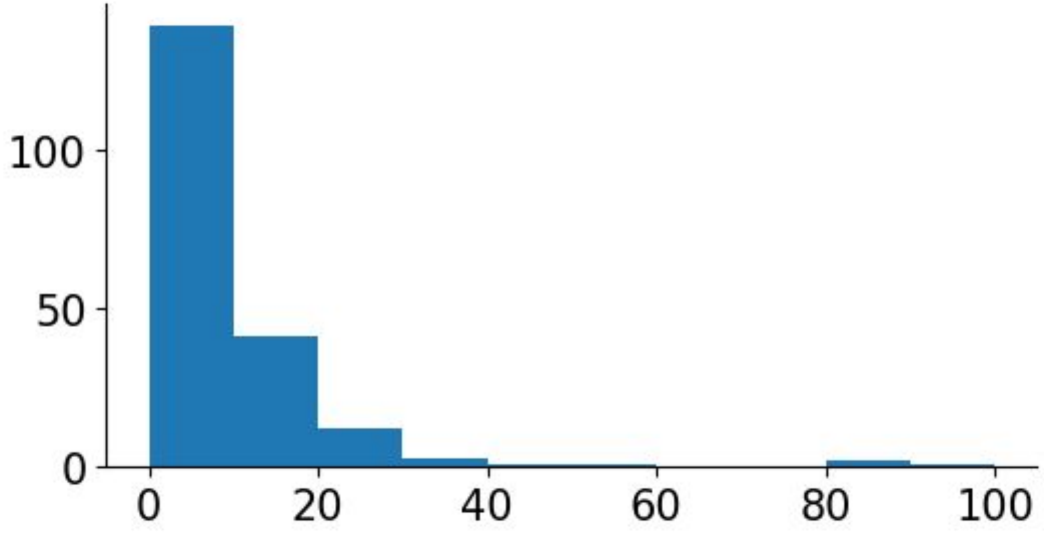
How much time did students spend on studying for Exam 1?
How many students scored each letter grade on Exam 1?
What fraction of students did Exam 1 Reflection?

We can visualize one-dimensional **numerical data** using a **histogram**.

Question:
How much time did students spend studying for Exam 1?

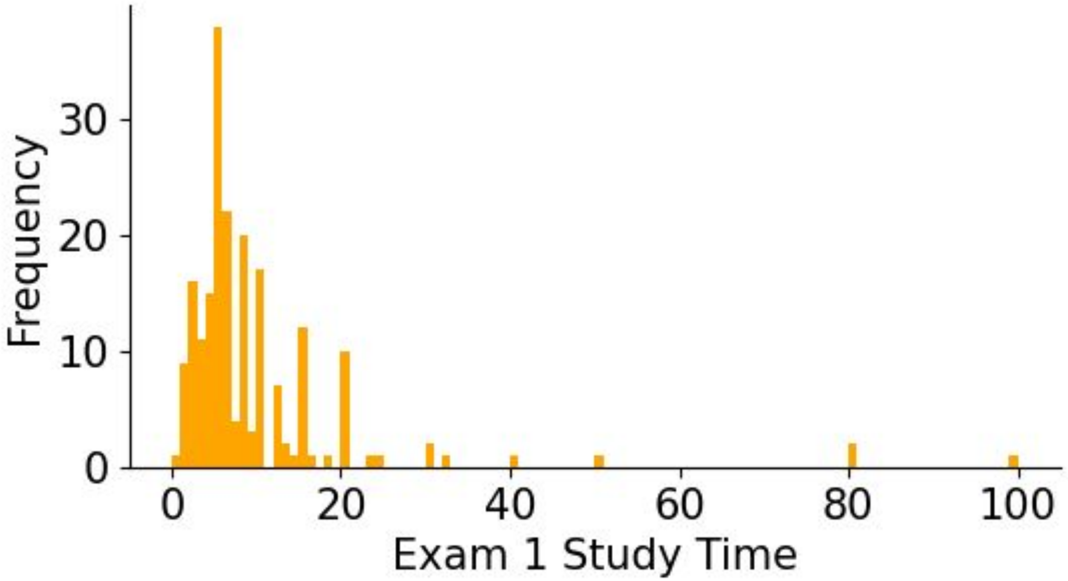
To make a histogram [docs]:

```
fig, ax = plt.subplots()
ax.hist(timeStudyExam1)
plt.show()
```



We can **customize graphs** with keyword arguments like `color` and axes functions like `set_xlabel`.

```
fig, ax = plt.subplots()
ax.hist(timeStudyExam1, bins=100, color='orange')
ax.set_xlabel('Exam 1 Study Time')
ax.set_ylabel('Frequency')
plt.show()
```



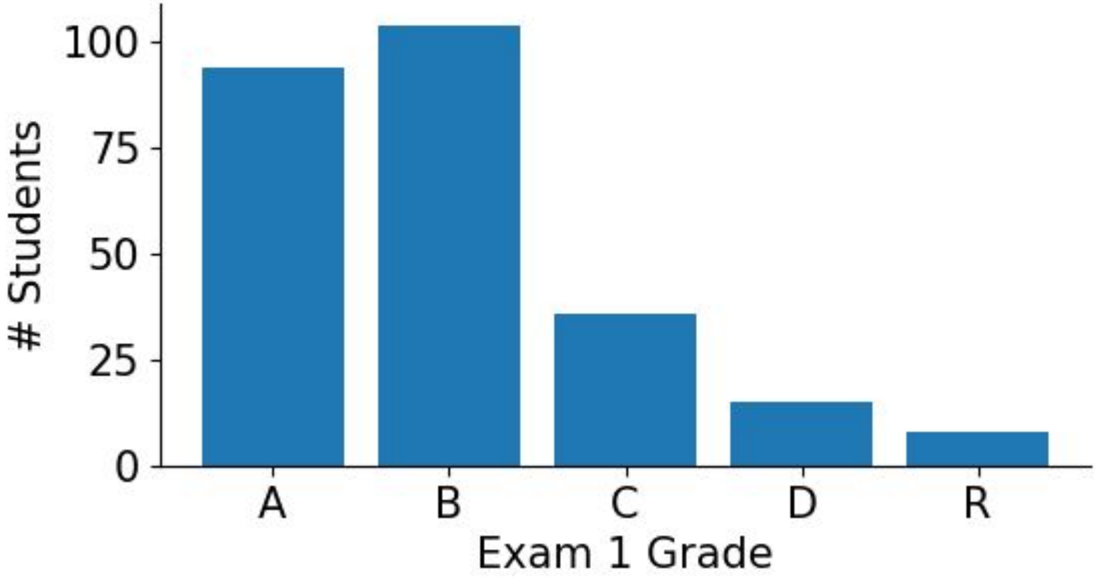
We can visualize one-dimensional **ordinal data** using a **bar graph**.

Question:

How many students scored each letter grade on Exam 1?

To make a bar chart [docs]:

```
fig, ax = plt.subplots()
ax.bar(gradeLabels, gradeCounts)
ax.set_xlabel('Exam 1 Grade')
ax.set_ylabel('# Students')
plt.show()
```

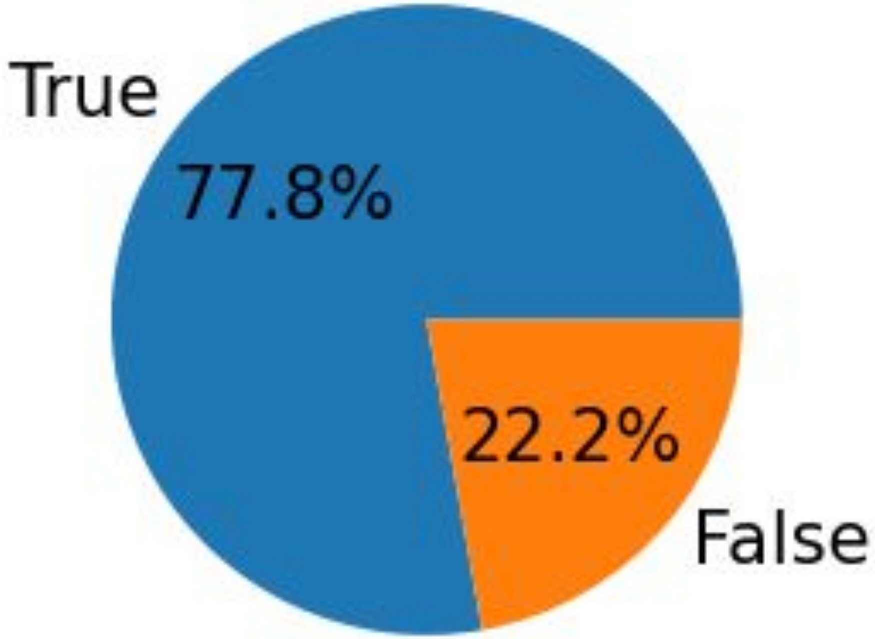


We can visualize one-dimensional **categorical data** using a **pie chart**.

Question:
What fraction of students did Exam 1 Reflection?

To make a pie chart [docs]:

```
fig, ax = plt.subplots()
ax.pie(reflectionCounts,
       labels=reflectionLabels,
        autopct='%d%%')
plt.show()
```



A **two-dimensional visualization** shows how two features in a dataset relate to each other.

We want to **visualize relationships,** which we cannot fully capture with a statistical number like conditional probabilities.

Example Questions:

How does the time studied for Exam 1 relate to the time studied for Exam 2?
How does Exam 2 grade relate to the time students studied for Exam 2?
How does Exam 2 grade relate to if students completed the Exam 1 reflection?

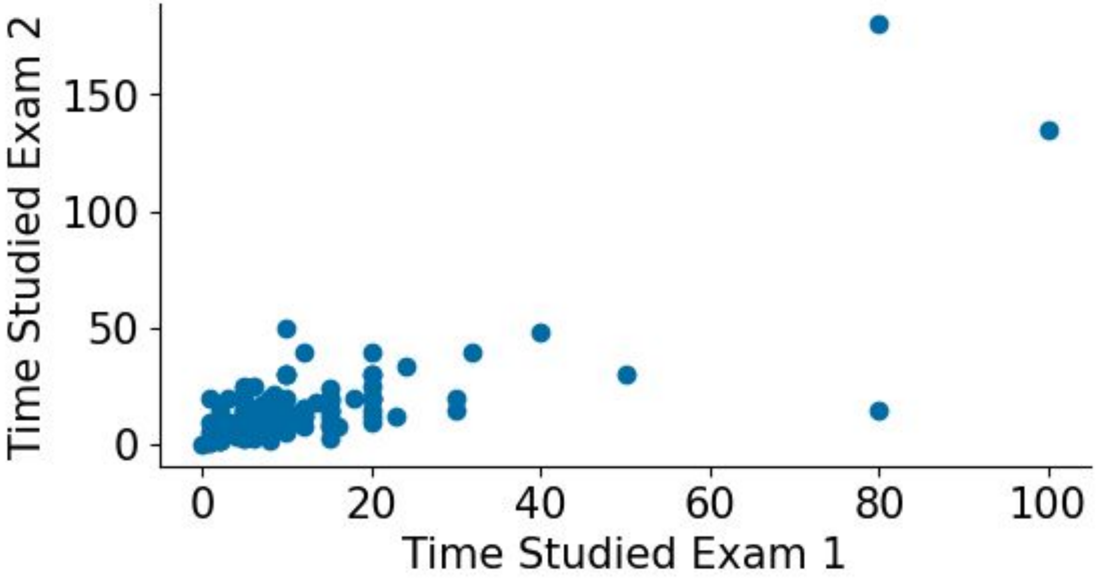
We can visualize two-dimensional **numerical data vs. numerical data** using a scatter plot.

Question:

How does the time studied for Exam 1 relate to the time studied for Exam 2?

To make a scatter plot [docs]:

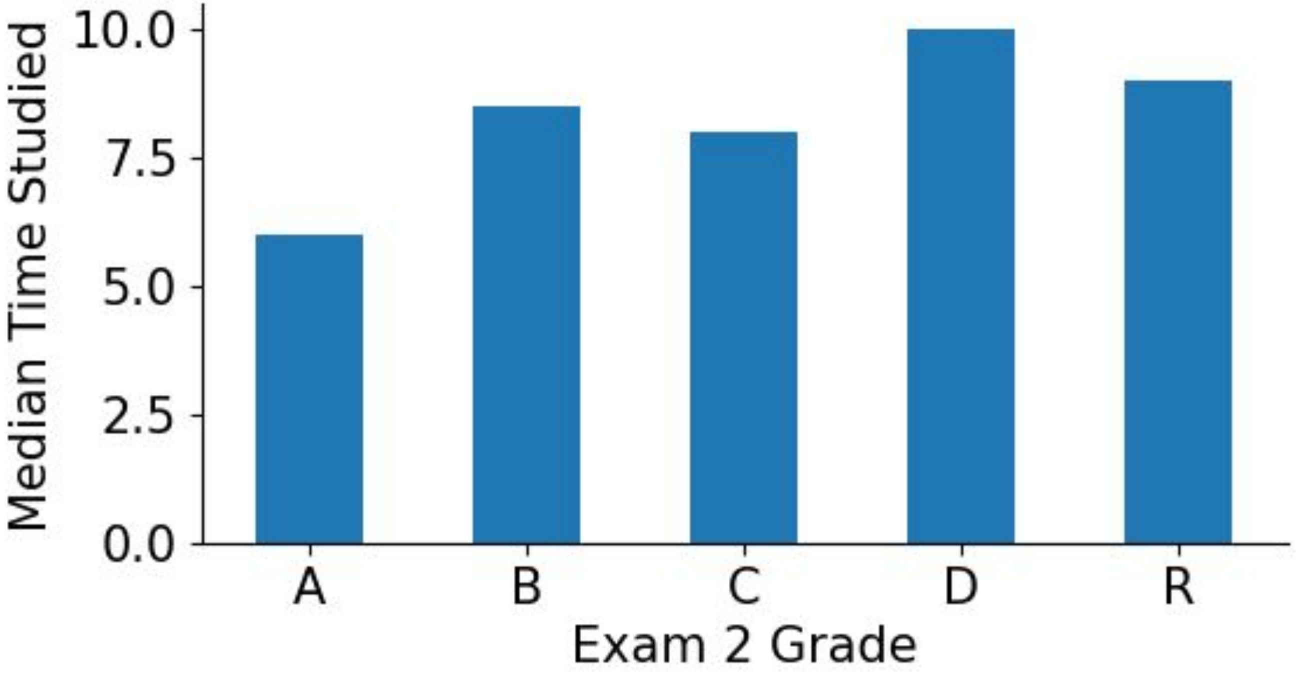
```
ax.scatter(timeStudyExam1,
           timeStudyExam2)
ax.set_xlabel('Time Studied Exam 1')
ax.set_ylabel('Time Studied Exam 2')
```



We can visualize two-dimensional **numerical vs. ordinal/categorical data** using bar plots of averages or box plots.

Question:

How does Exam 2 grade relate to the time students studied for Exam 2?



It can be hard to visualize **ordinal/categorical vs. ordinal/categorical data** depending on the question. Sometimes a table works well.

Question:

How does Exam 2 grade relate to if students completed the Exam 1 reflection? (from real data)

Exam 2 Grade	**Percentage Completed Exam 1 Reflection**
A	84 %
B	88 %
C	81 %
D	87 %
R	**63 %**

A **three-dimensional visualization** tries to show how three features in a data set relate to each other.

Visualizing data in three or more dimensions can be hard!

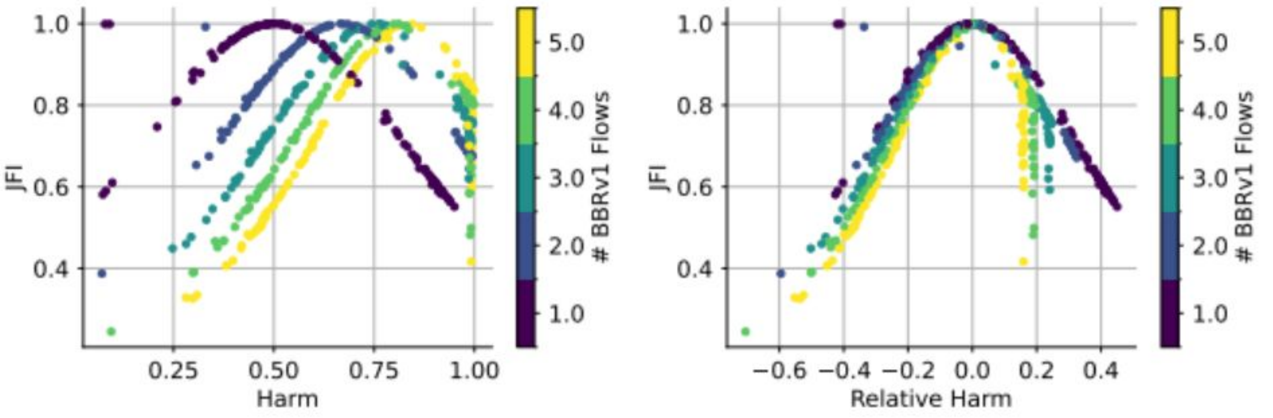
What are our options here if we want to plot this in 2 dimensions?

Activity: Draw a visualization to help show if Exam 2 reflection helped students better prepare for Exam 1.

Convert Exam 2 grades to numbers. How does Exam 2 grade relate to time studied for Exam 2 and completing the Exam 1 reflection?

Types of plots:
[1] histogram
[2] bar plot
[3] pie chart
[4] scatter plot
[5] I don't know

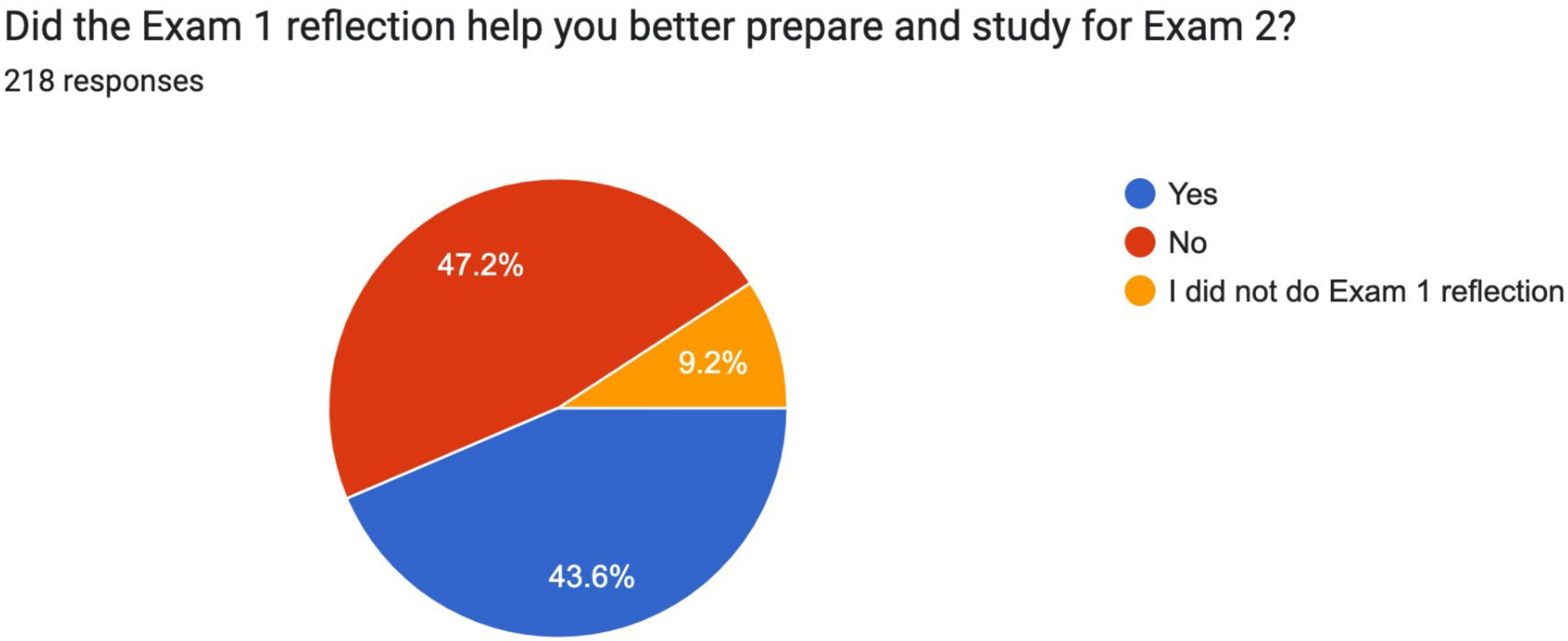
One example of multidimensional data from my recent paper



(a) Absolute harm **(b)** Relative harm

Figure 2: Comparison of absolute and relative harm for 1 Cubic flow vs. 1-5 BBR flows for varying network settings

Here’s what students self-reported saying if the Exam 2 reflection helped (from Google Form summary):



Did the Exam 1 reflection help you better prepare and study for Exam 2?

218 responses

Some other considerations for data visualization:

- Consider the audience
- Accessibility: Choosing a color blind friendly palette

  In matplotlib:

  ```
  plt.style.use('tableau-colorblind10')
  ```

- Informative axes labels
- And much more!

Learning Goals

- Perform **basic analyses** on data, including calculating **statistics** and **probabilities**, to answer simple questions

- Choose an appropriate **visualization** to create based on the number of **dimensions** and **data types**

- Create simple **matplotlib visualizations** that show the state of a dataset

Backup Slides

Some activities to try on your own:

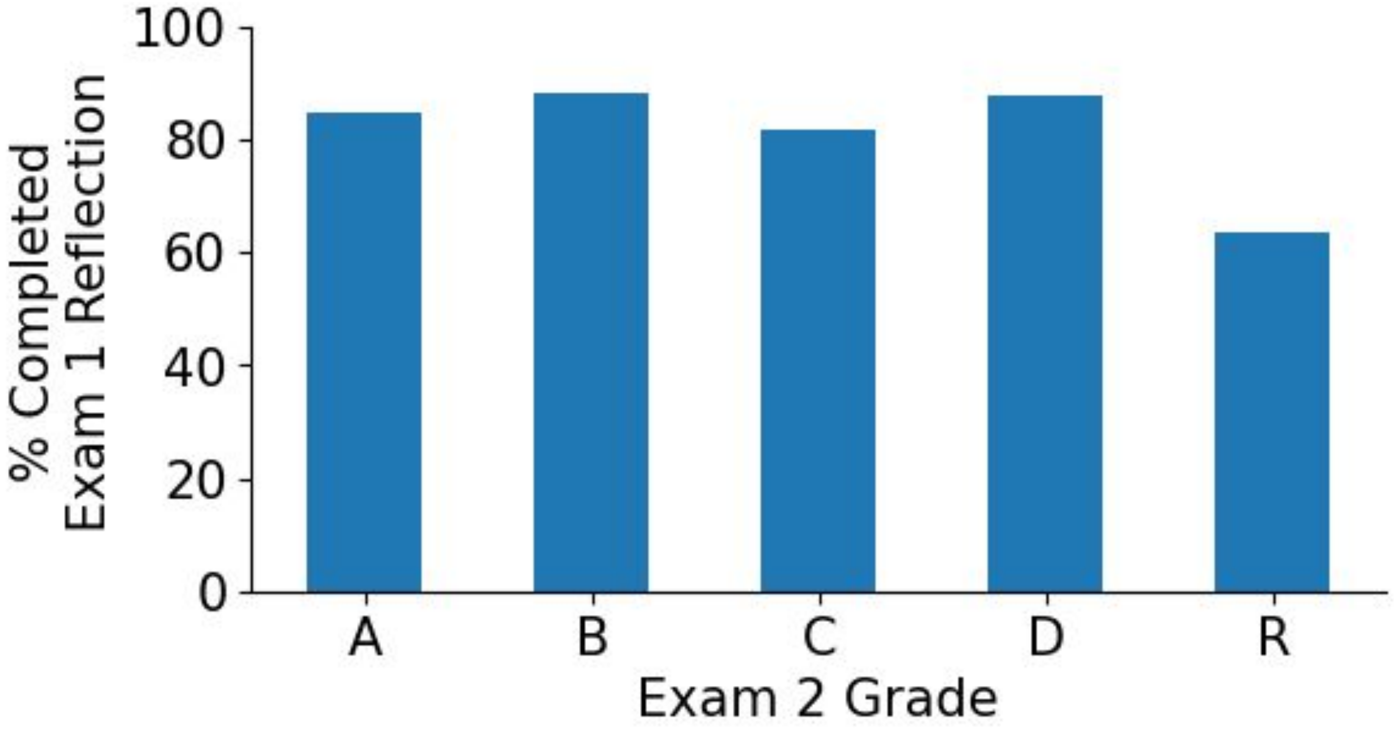
Write code to compare the relationship between grades and time spent:

- Reading notes
- Reading slides
- Doing practice problems

Write code to compute what was the most common way students spent studying.

Write code to compute what was the most common things students lost points for.

The pandas library [doc] is a convenient library for doing data processing, cleaning, analysis, and visualization!


% Completed
Exam 1 Reflection
A
B
C
D
R
Exam 2 Grade