ENG 10: Lecture 6

Spring 2021

Engineering of the week

1. Fernando Rico “Transparent solar panels”

2. Tiffany Soebroto “Drug delivery nanoparticles for cancer treatment”

Please, complete this feedback form before next class
(also posted on Canvas - Modules – Links, Forms, Quizzes)

Week 6 Feedback Form:

- https://forms.gle/HGGZa5ZRJ8oYd3DY7

Why Linear Algebra is important?

- **How many variables** does this landing system need?
- **How many equations** does this landing system need?
- All the computations have to be done **in real-time** for a high accuracy of landing.

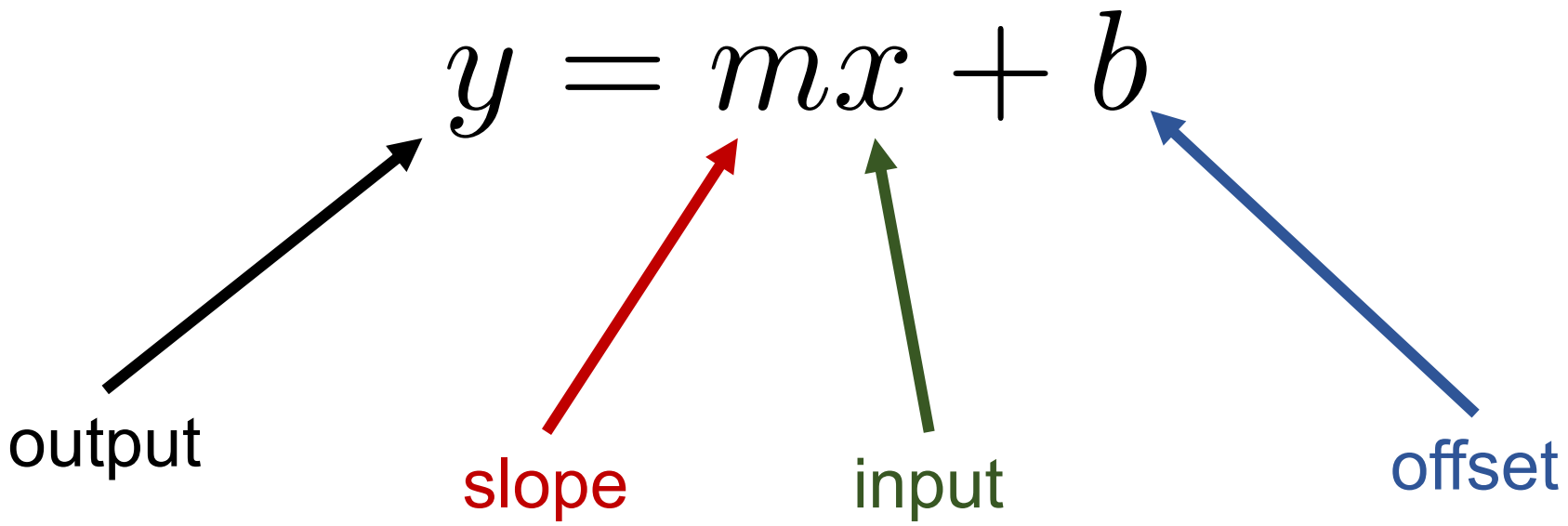
SPACEX

SPACEY

SPACEX^{2}=Y

Linear Equations

Start with a single linear equation:



3x3 System of Linear Equations

$$a_{1,1}x_1 + a_{1,2}x_2 + a_{1,3}x_3 = b_1$$
$$a_{2,1}x_1 + a_{2,2}x_2 + a_{2,3}x_3 = b_2$$
$$a_{3,1}x_1 + a_{3,2}x_2 + a_{3,3}x_3 = b_3$$

3x3 System of Linear Equations

Answer: YES!

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix} \quad \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

A **x** **b**

System of Linear Equations

What about bigger systems (e.g. 5x5, 10x10, 10^6x10^6)?

Same ideas apply:

1) Group the components

2) Rewrite as Ax = b

3) Solve

System of Linear Equations

- But how do you solve it???
- We are unable to calculate the values in real time.
- … We can quickly compute Ax = b problems in Python with the following NumPy Module: ``x = np.linalg.solve (A, b)``

Math: Probability and Statistics

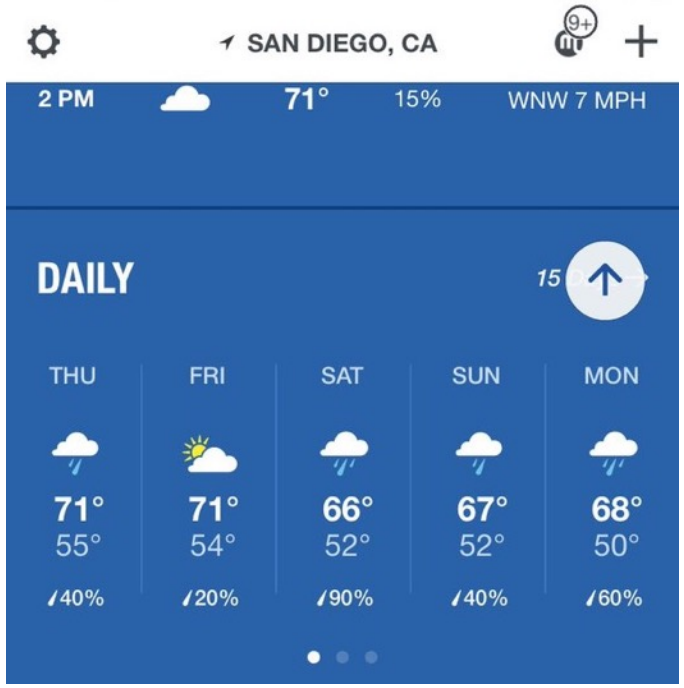
Probability

Probability: Introduction

- What is probability?

- How likely something is to happen.

① Chance of rain in San Diego ☺



Roughly 0 (usually)☺

② Chance of winning a lottery

Jackpot odds of average state lottery: 100,000,000 to 1

③ Chance of being struck by lightning

The estimated chance of an average person living in the US being struck by lightning a year: 960,000 to 1

Probability: Introduction

- Tossing a coin:

	Head	Tail
Probability	$\frac{1}{2}$	$\frac{1}{2}$

✓ Probability of Head or Tail: 1/2

- Throwing a dice:

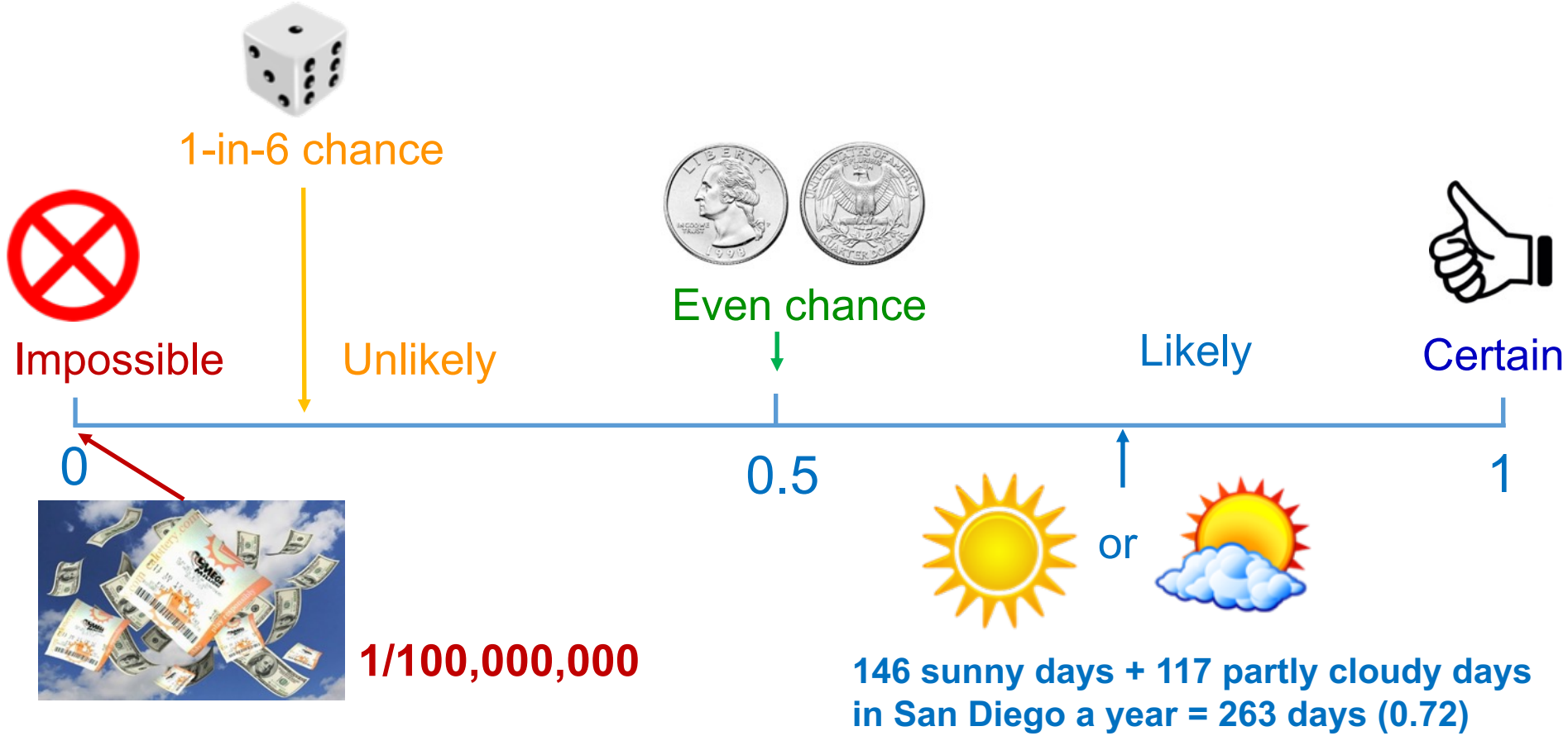
Probability	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$

✓ Probability of the any one of the faces: 1/6

Probability: Probability scale

- **Probabilities can be written as:**
  ① Fractions from 0 to 1
  ② Decimals from 0 to 1
  ③ Percentages from 0 % to 100 %

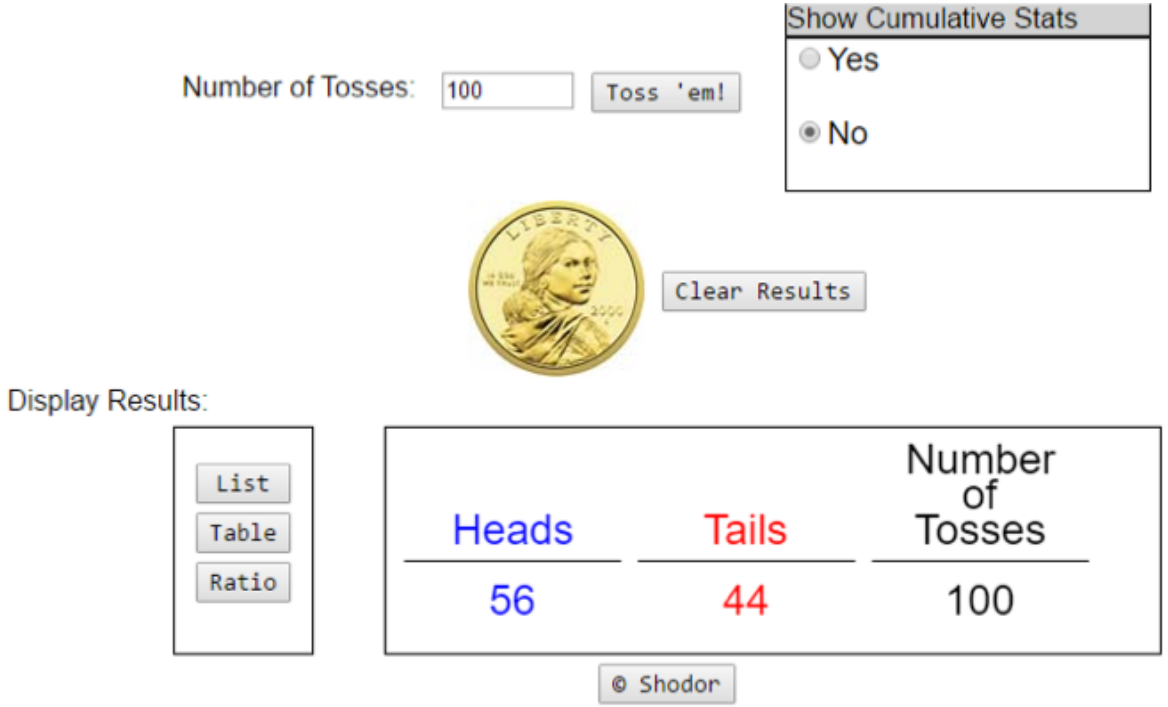
- **Probability Scale:**



Probability: Difference b/w theoretical value and actual result

- Interactive: Coin toss

http://www.shodor.org/interactivate/activities/Coin/



- Try tossing a coin **10** times
- Try tossing a coin **100** times
- Try tossing a coin **1000** times

Discussion about the results

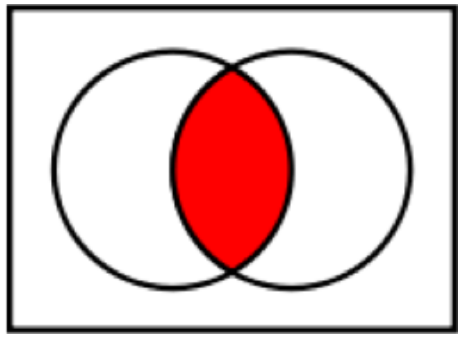
- What is the difference b/w the theoretical probability value and the real result? For 10, 100, and 1000 times
- How does the result change with the number of trials?
- Why is the experimental result different from the theoretical value?
- How do we narrow the gap b/w theoretical value and experimental results?

Probability Review: Venn diagram

- Diagram that shows **all possible logical relations** between a finite collection of different sets using circles/closed curves

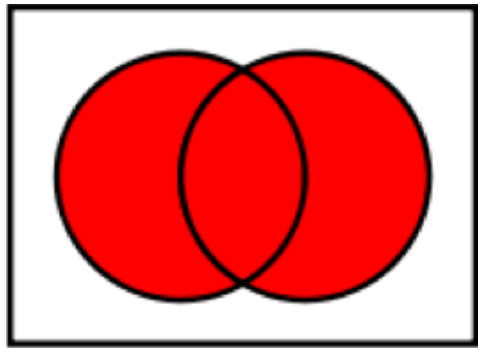
Intersection of two sets

$A \cap B$



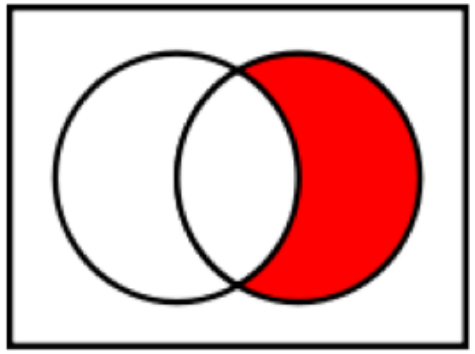
Union of two sets

$A \cup B$

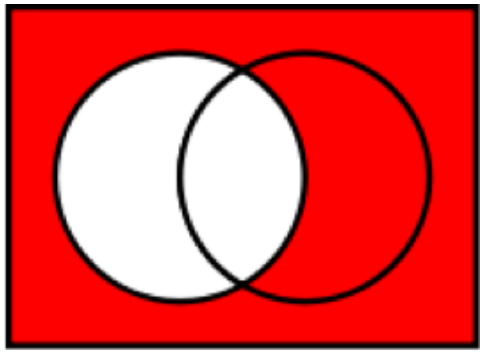


Relative complement of A in B

$A^c \cap B = B \setminus A$

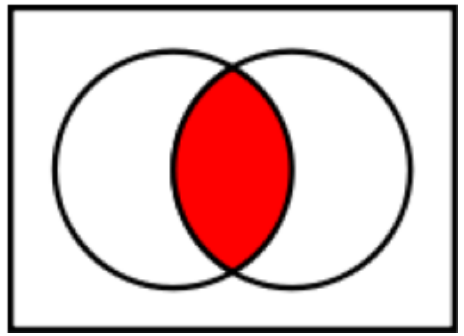


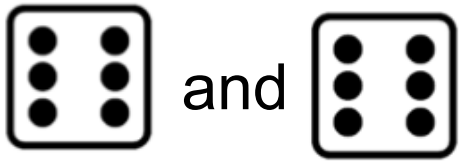
$A^c = U \setminus A$



Probability: Mathematical treatment

- **JOINT probability for Independent events:**
  - When two events A and B are independently occur on a single experiment
  - P(A and B) = P (A ∩ B) = P(A)·P(B)
  - *e.g. if a pair of dice are rolled, the chance of both being 6 is:*



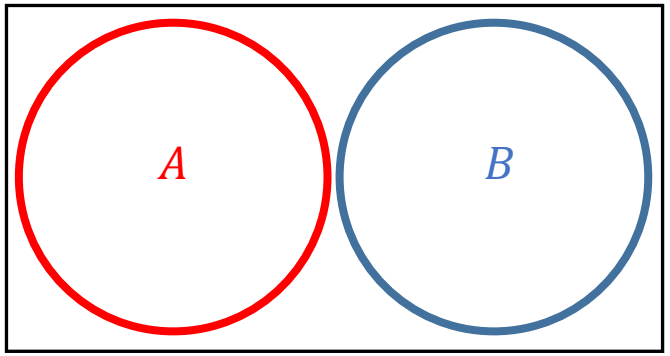
 and

$$\frac{1}{6} \times \frac{1}{6} = \frac{1}{36}$$

- **Mutually EXCLUSIVE events:**
  - When either event A or event B occurs on a single experiment
  - P(A or B) = P(A∪B) = P(A) + P(B)
  - *e.g. the chance of rolling a 3 or 6 is P(3 or 6):*

 or 

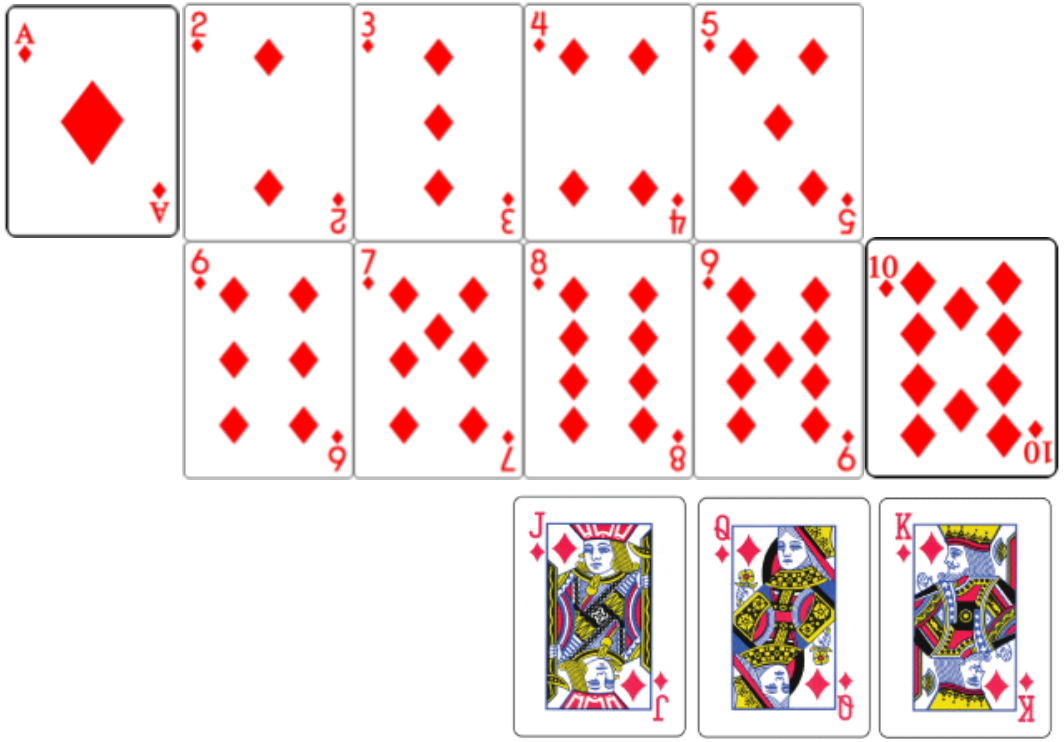
$$\frac{1}{6} + \frac{1}{6} = \frac{1}{3}$$



Probability: Mathematical treatment

- **NOT mutually EXCLUSIVE events**
  - **When the events are <u>not mutually exclusive</u>**
  - **P(A or B) = P(A) + P(B) – P(A and B)**

- *e.g. the chance of getting **a diamond or an alphabet card***



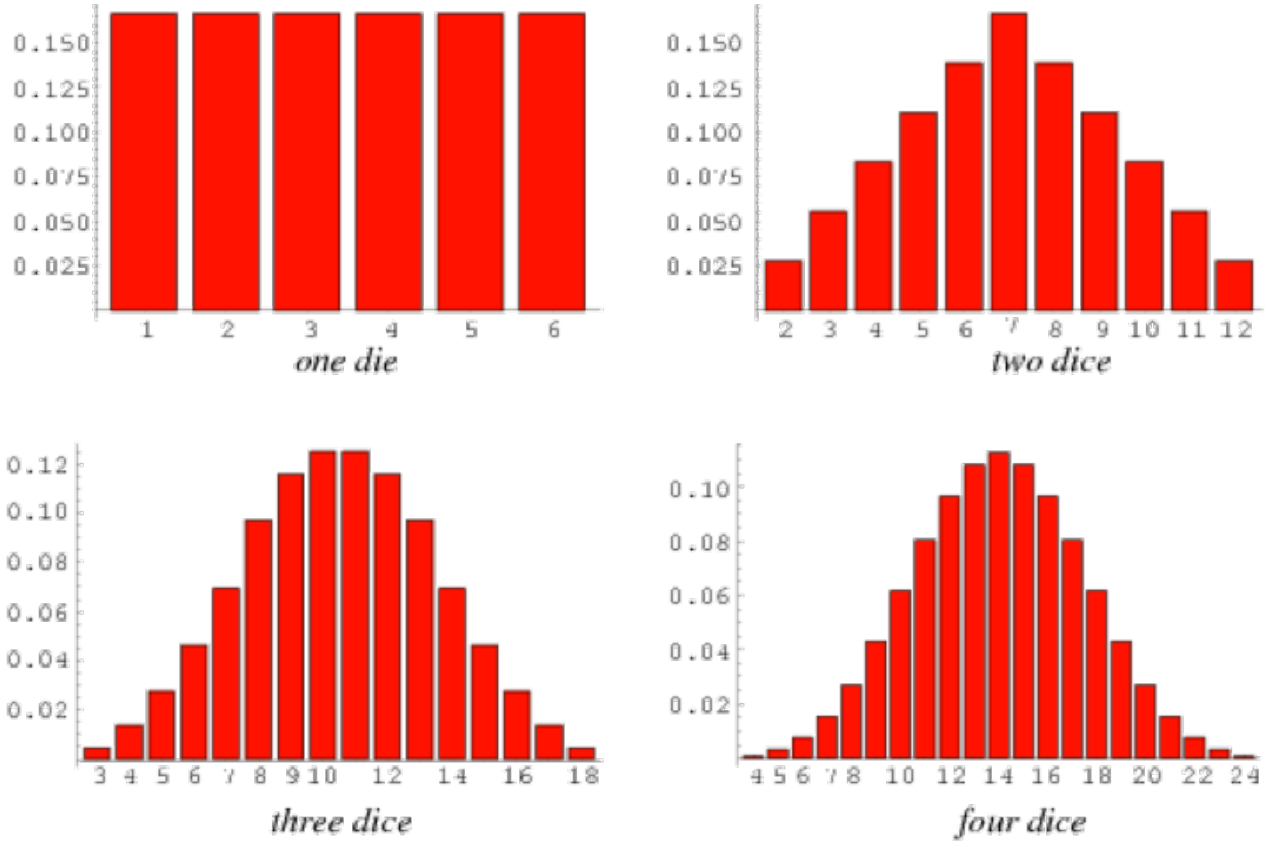
or

$$\frac{13}{52} + \frac{16}{52} - \frac{4}{52} = \frac{25}{52}$$

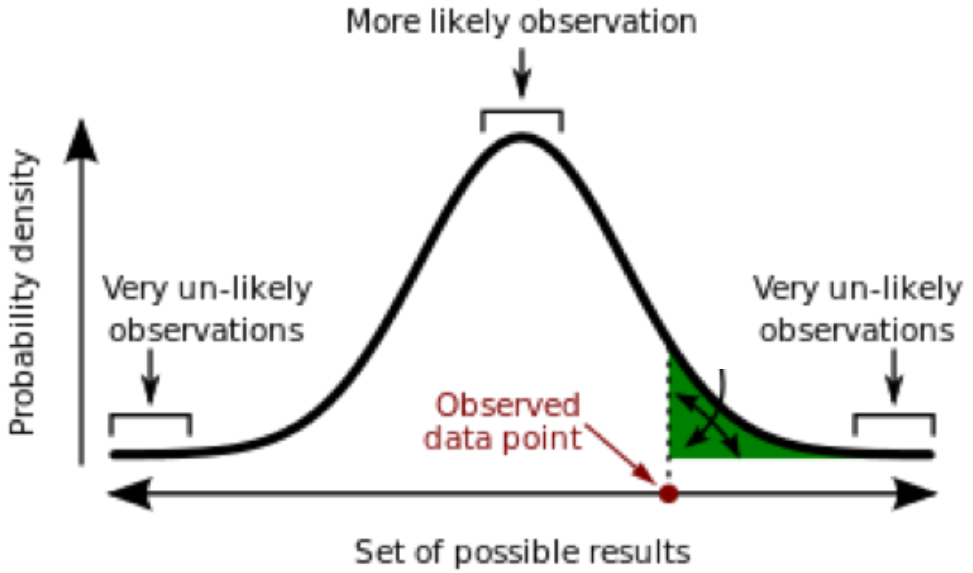
Probability: Probability distribution

- **The distribution of a random variable is a collection of possible outcomes along with their probabilities:**

- **<u>Binomial</u> Distribution (symmetric): Discrete probability distribution**



- **<u>Normal</u> (or Gaussian) distribution (symmetric): Very common continuous distribution**



Central limit theorem !

Weibull statistics for survival probability of a particular component as a function of its volume and the applied stress

$$P(V_o) = \exp\left\{-\left(\frac{\sigma - \sigma_u}{\sigma_o}\right)^m\right\}$$

Probability of survival

$$F(V_o) = 1 - \exp\left\{-\left(\frac{\sigma - \sigma_u}{\sigma_o}\right)^m\right\}$$

Probability of failure

σ = applied stress

σ_u = stress below which there is a zero probability of failure

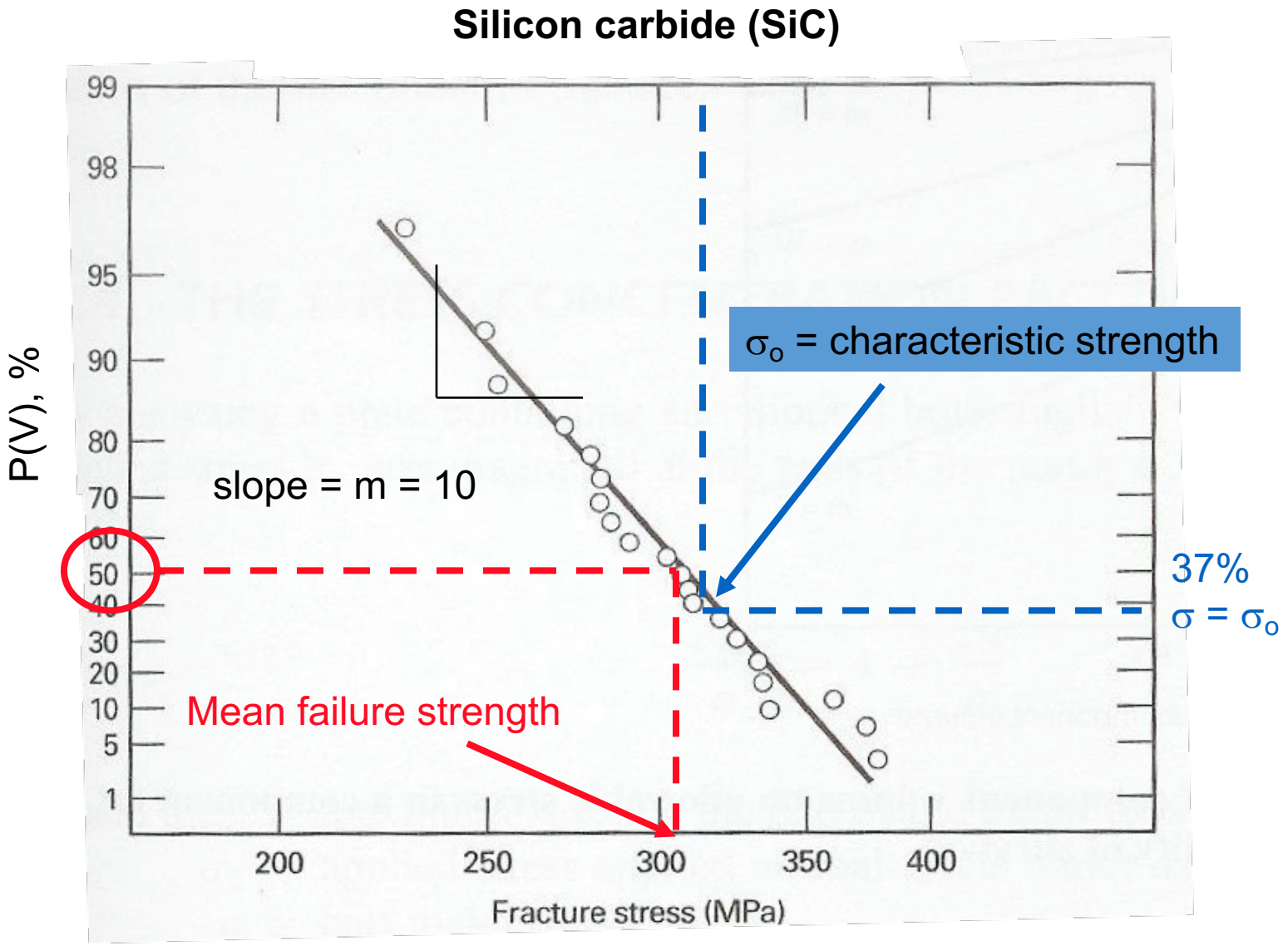
- this means there is an upper limit to flaw size
- for brittle materials $\sigma_u = 0$ (since any tensile stress might cause failure)

σ_o = characteristic strength ~ **mean strength** of the material

m = **Weibull** modulus

- characterizes the variability in strength ~ **standard deviation** of material's strength

Weibull plot



Weibull analysis

- **Increasing m values reflect** more homogeneous material behavior with **strength levels** for a given component being **more predictable**.
- In the limit where m approaches ∞, the probability of failure = 0 for all stress levels < σ_0.
- In the limit where m approaches 0, the probability of failure 1, and failure occurs with equal certainty at any stress level.


FORUM
SOCIAL NETWORK
CONTENT SHARING
CROBLOGS
SOCIAL MEDIA
VIDEO
COMMUNICATION
CHAT
PICTURE
OUR BUSINESS
OK
CLIENT 1
CLIENT 2
CLIENT n.
FIREWALL
SERVER FARM
INTERNET
VOIP
COMPANY HEADQUARTERS
GROWTH
DESIGN
OUR COMPANY
COMPETITOR
OTHER
35%
MONE
TIME
STATISTICS
500.000
LAST YEAR
1.000.000
THIS YEAR
QUALITY
STRATEGY
VISION
SMS

Statistics

What is Statistics?

Definition: The mathematical and scientific methodologies for collecting, organizing, analyzing, and drawing conclusions from data or information to:

- **Make decisions, predictions, or plans**
- **Solve problems**
- **Generalize uncertain phenomenon and events**

Salary Negotiation

Plan for College Applications

Global Warming

Data collection methods (**sampling**)

- **Surveys**

- **Interviews**

- **Observations**

- **Experiments**

Importance of accuracy: Watch out for **Faulty data**!

What issues might we encounter
when collecting data?

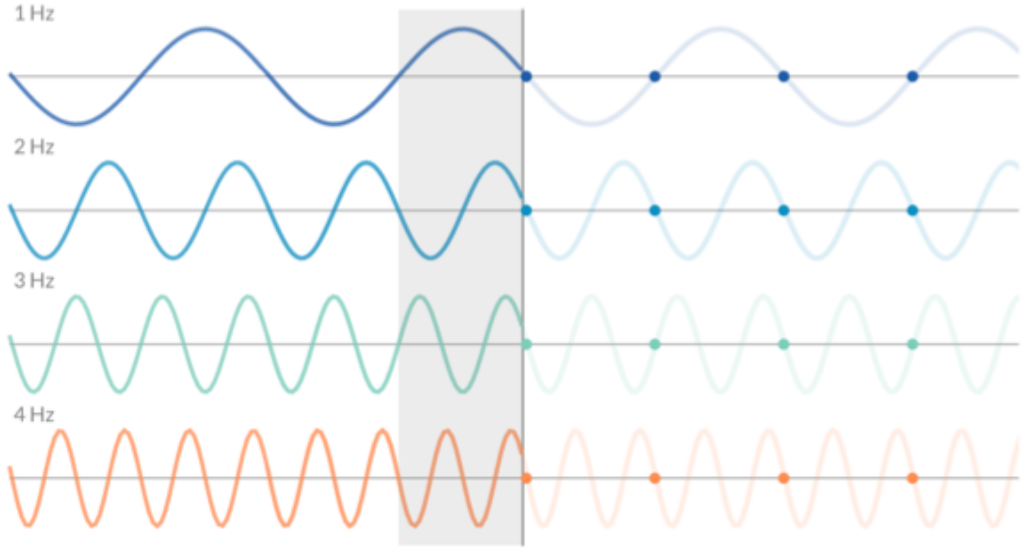
Importance of accuracy: Watch out for **Faulty** data!

- **Lies in survey**

- Samples do not represent the whole.

- **Failed in controlling any independent variables that affect results**

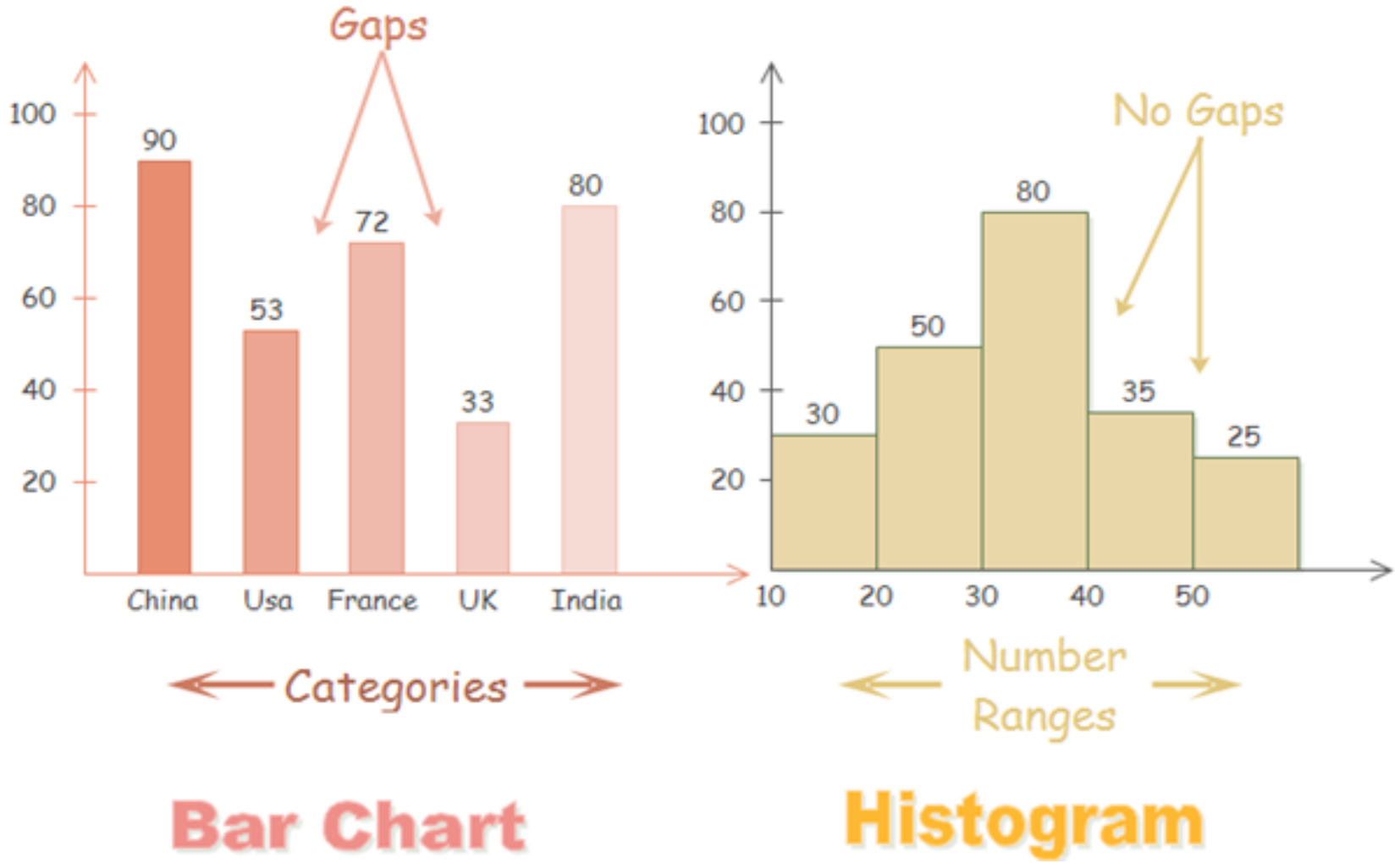
- **Not enough # of sampling**



- Accurate data collection is essential to minimize errors
- Otherwise, these unreliable data bring wrong decisions or distorted findings

How to represent data: Bar Chart and Histogram

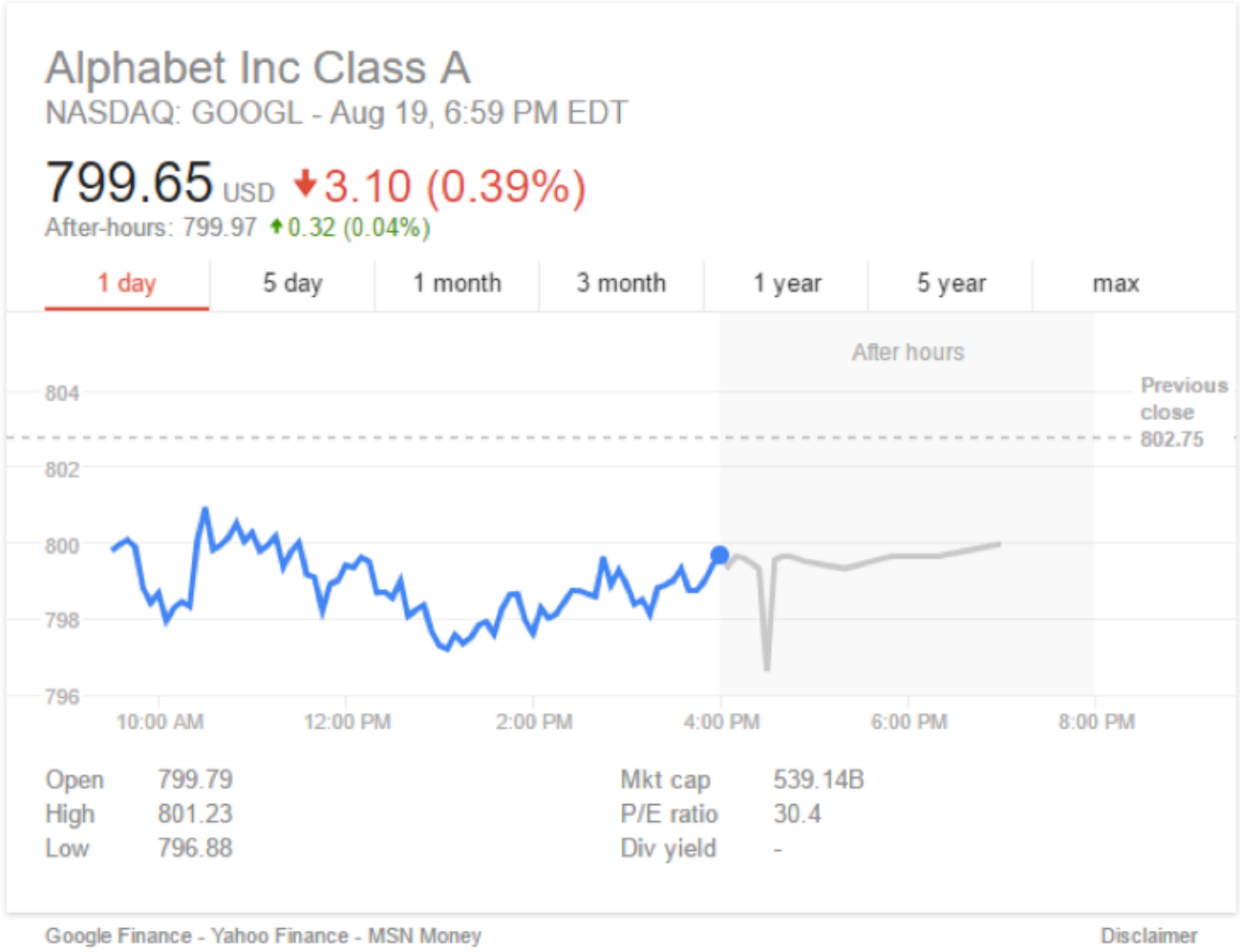
- Bar Chart (Discrete & Categories) - Histogram (Continuous & Ranges)



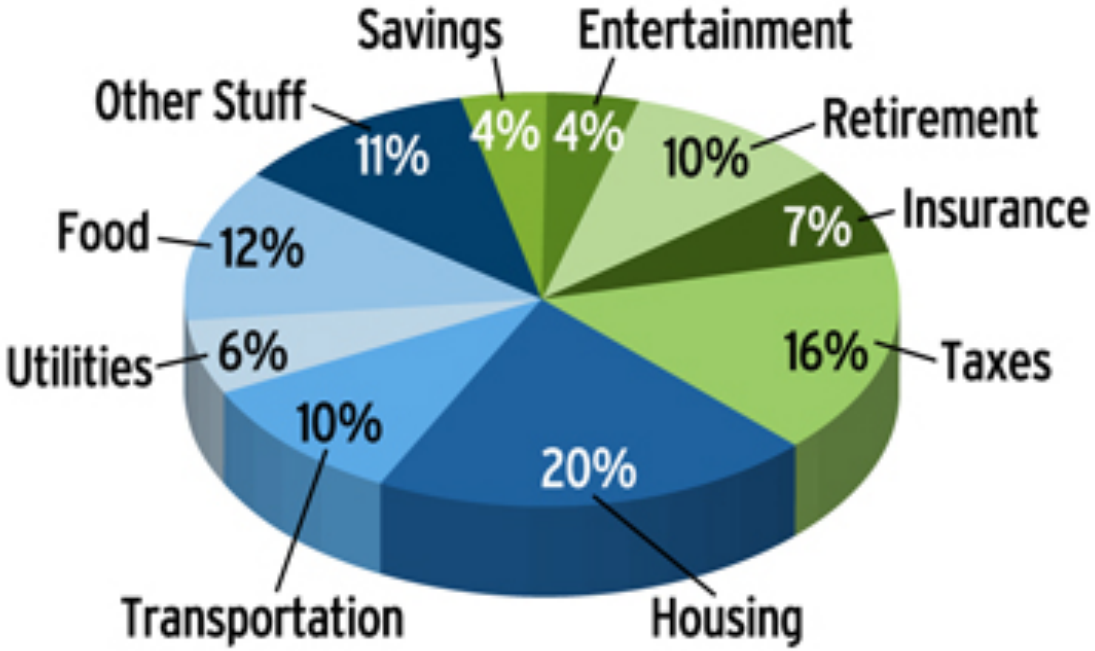
Histograms are a great way to show results of continuous data, such as: Weight, height, time…

How to represent data: Line Graphs and Pie Charts

- Line Graph: Good to observe **trends**

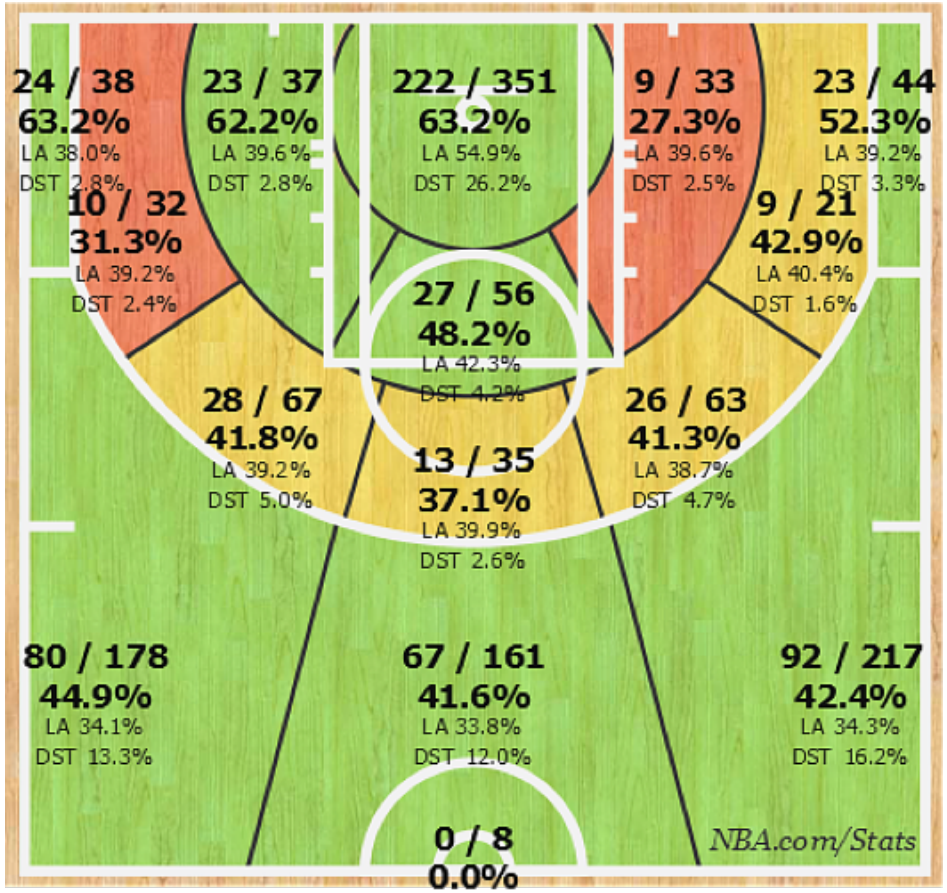


- Pie Chart: Good to describe **numerical proportion**



Descriptive statistics

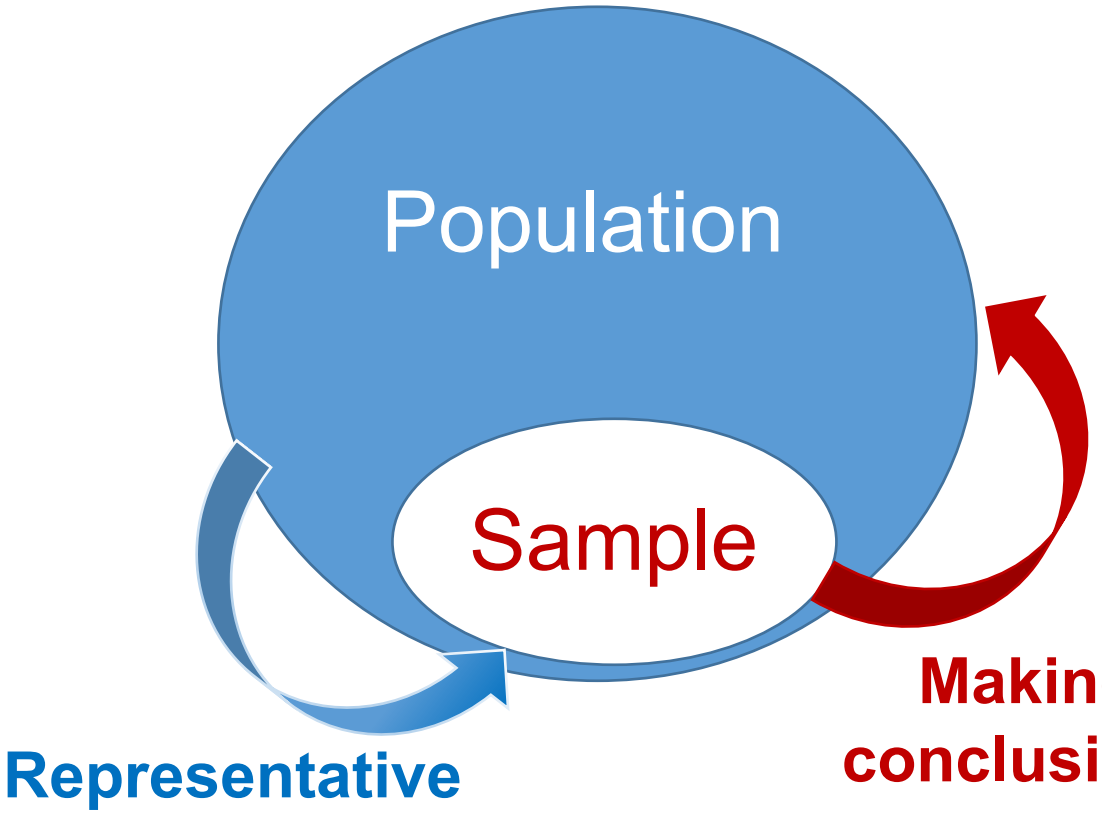
A way to provide simple and straightforward summaries of data from the sample and the measures in form of numbers or graphs.

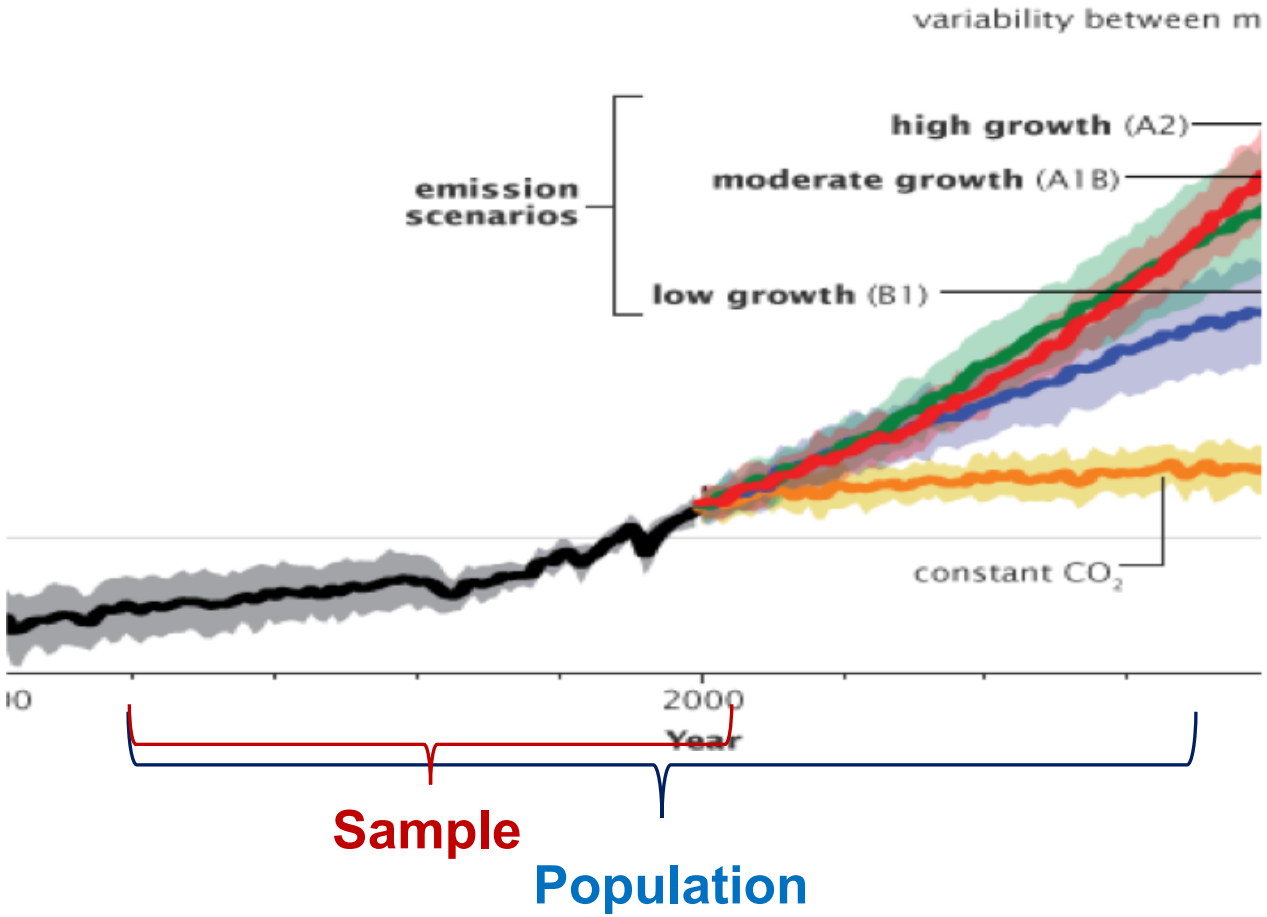


The shooting percentage in basketball is a descriptive statistic that summarizes the performance of a player or a team.

Inferential statistics

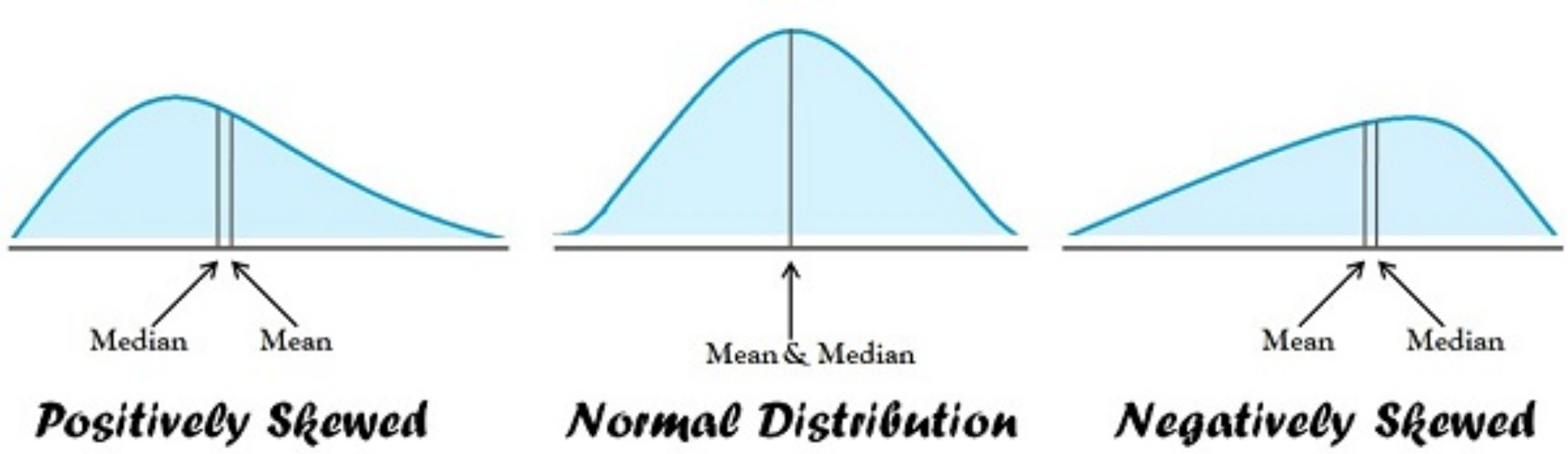
A procedure for inferring or predicting future or a population by using data from the population with some form of sampling (**Very Complicated**)





Central tendency

Definition: A measure of central tendency is **a representative value** to describe a set of data by identifying the central position within that set of data.



Representative values: **Arithmetic Mean, Median, and Mode**

Central tendency: Arithmetic mean

Arithmetic Mean: The mean is equal to the sum of all the values in the data set divided by the number of values in the data set.

$$\bar{x} = \frac{(x_1 + x_2 + \cdots + x_n)}{n}$$

Example:

Staff	1	2	3	4	5	6	7	8	9	10
Salary	15k	18k	16k	14k	15k	15k	12k	17k	90k	95k

$$\frac{15{,}000 + 18{,}000 + 16{,}000 + 14{,}000 + 15{,}000 + 15{,}000 + \ 12{,}000 + 17{,}000 + 90{,}000 + 95{,}000}{10}$$

$$= \$\,30{,}700$$

Disadvantage of Mean: Not the best way to reflect the population!!!

Central tendency: Median

Definition: The **middle score** for a set of data that has been arranged **in order of magnitude.**

For the set of salaries:

Salary	15k	18k	16k	14k	15k	15k	12k	17k	90k	95k

Rearrange from smallest to largest salary:

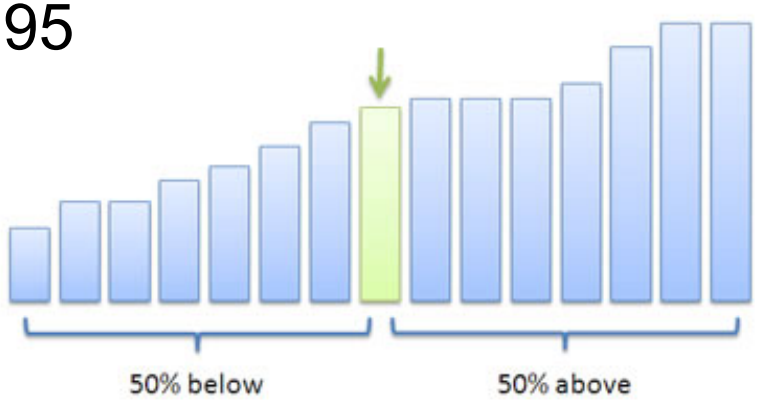
Salary	12k	14k	15k	15k	15k	16k	17k	18k	90k	95k

Median

Median = 12k, 14k, 15k, 15k, 15k, 16k, 17k, 18k, 90k, 95

Because there are an even number of data points,
Take the mean of the middle numbers…

Median = 15.5k

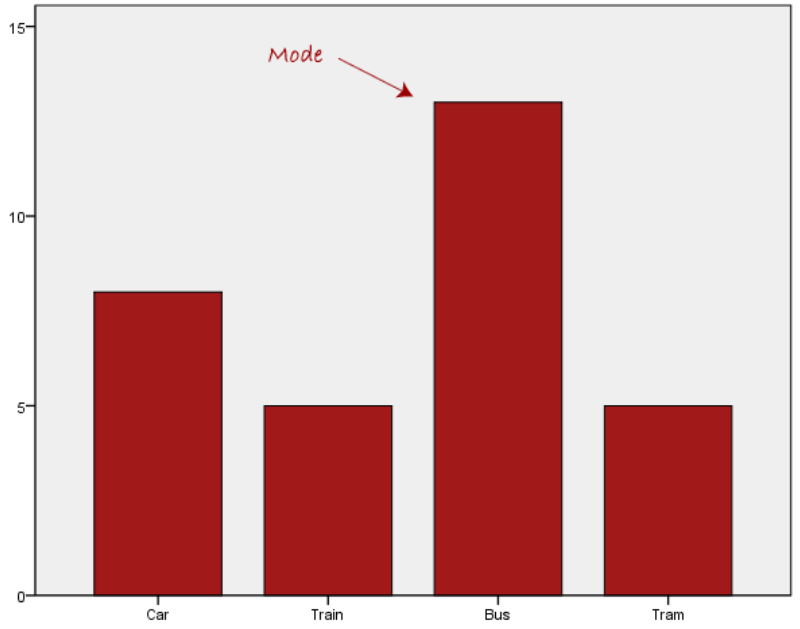


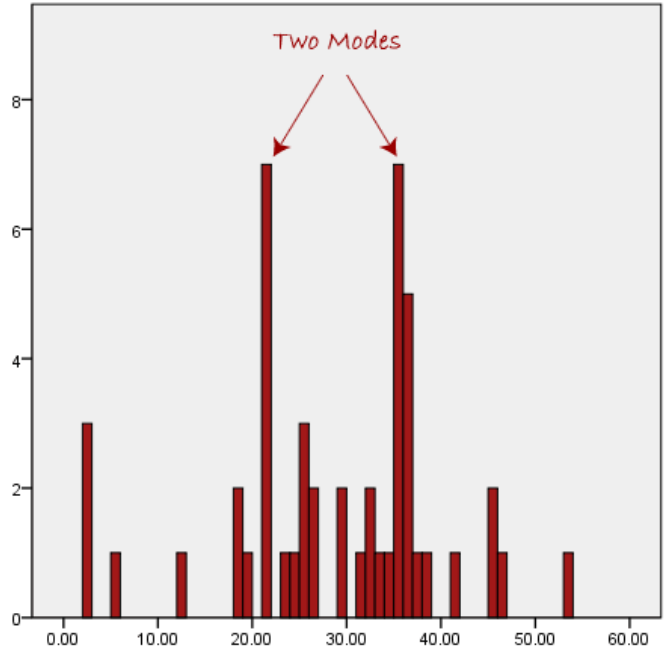
Central tendency: Mode

Definition: The mode is **the most frequent score** in our data set. It represents the highest bar in a bar chart or histogram. Gives an idea of the most dominant samples in the data set.

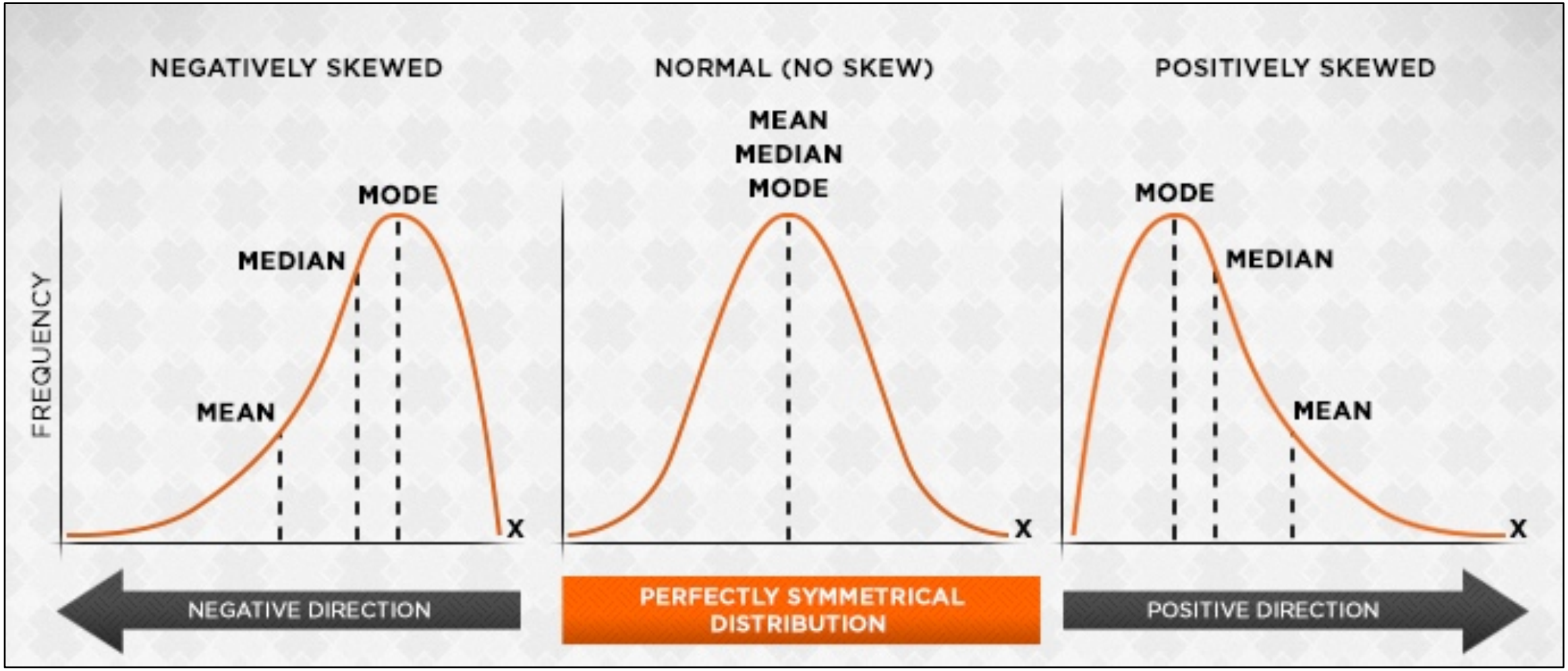
In our salary data set, the dominating salary amount is $15k, as it appears the most in the data set.

Salary	12k	14k	15k	15k	15k	16k	17k	18k	90k	95k





Central tendency:
Symmetric, positively and negatively skewed data



Mode < Median < Mean **Mode = Median = Mean** **Mean > Median > Mode**

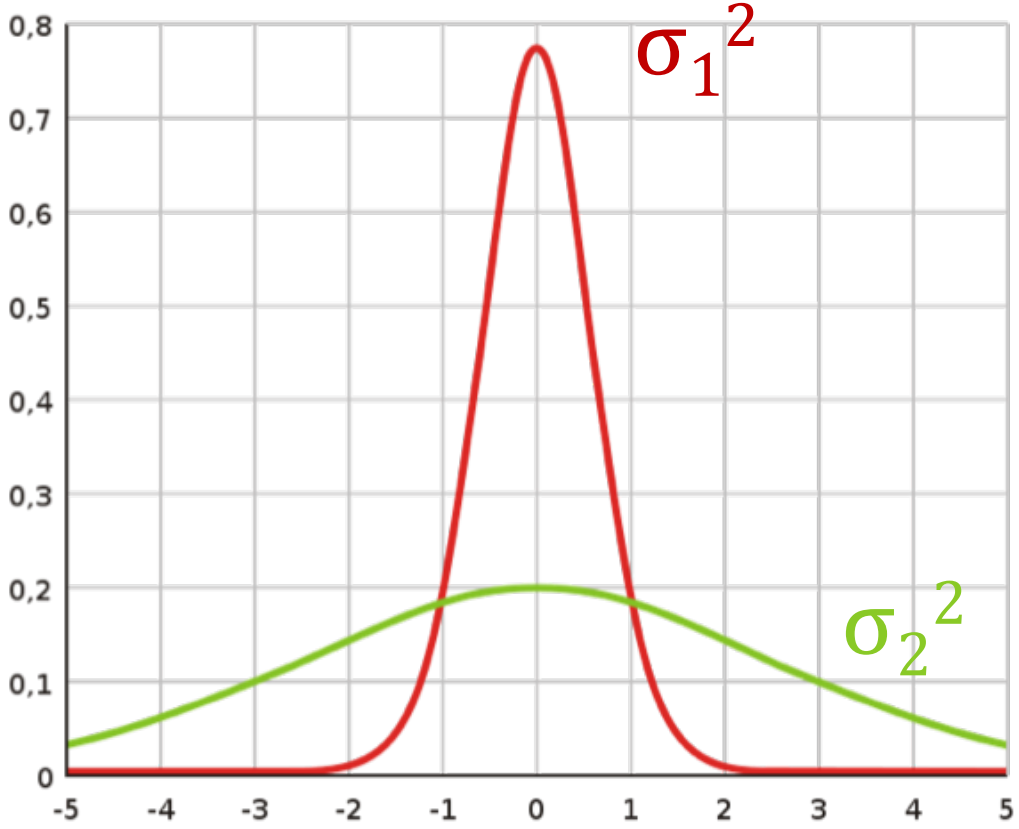
Variance

Definition: Variance is a measurement of the spread between numbers in a data set. The variance is a distance measure telling us **how far each number** in the set is from **the mean**.

Denoted as σ^2:

$$\sigma^2 = \frac{\sum(X - \mu)^2}{N}$$

X: individual data point
μ : mean of data points
N: total # of data points



$\sigma_1^2 < \sigma_2^2$

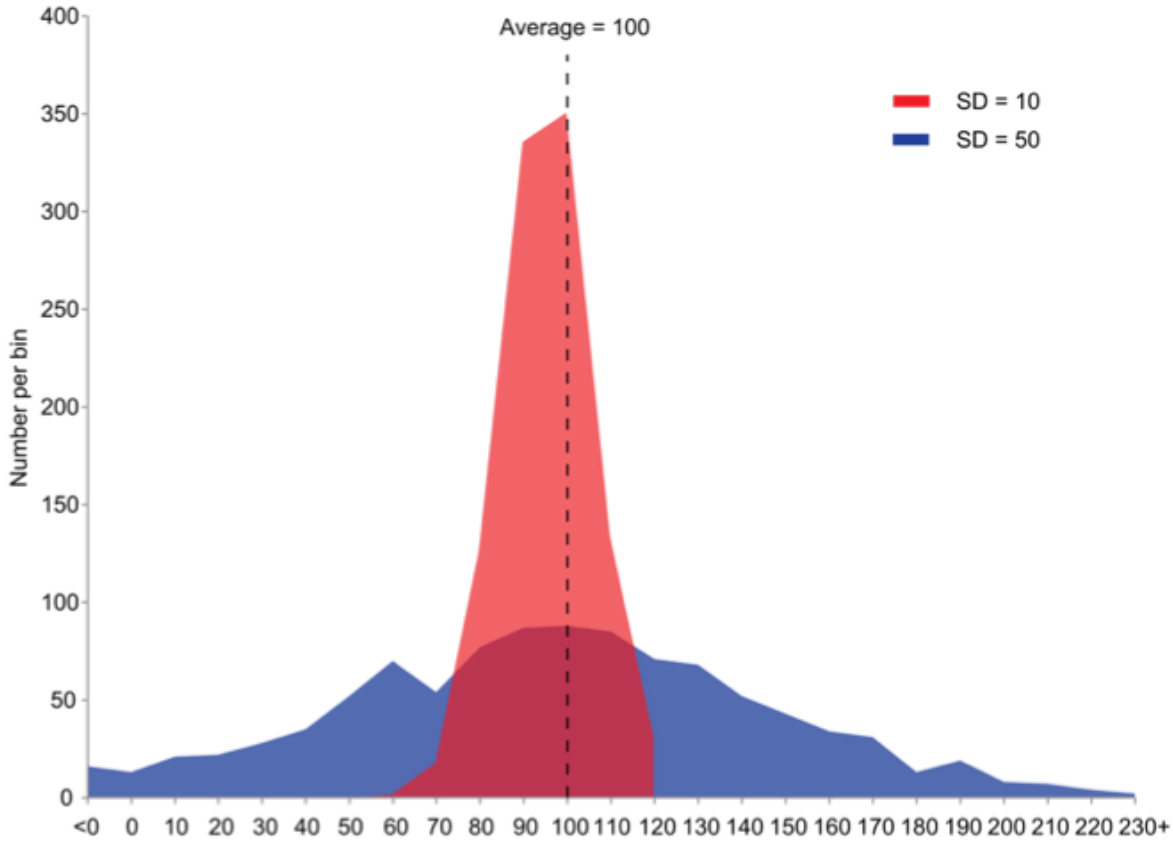
Standard deviation

Definition: A measure that is used to quantify the amount of variation or dispersion of a set of data values. This can also be considered as a distance measure.

Denoted as σ which is the square root of its **variance**:

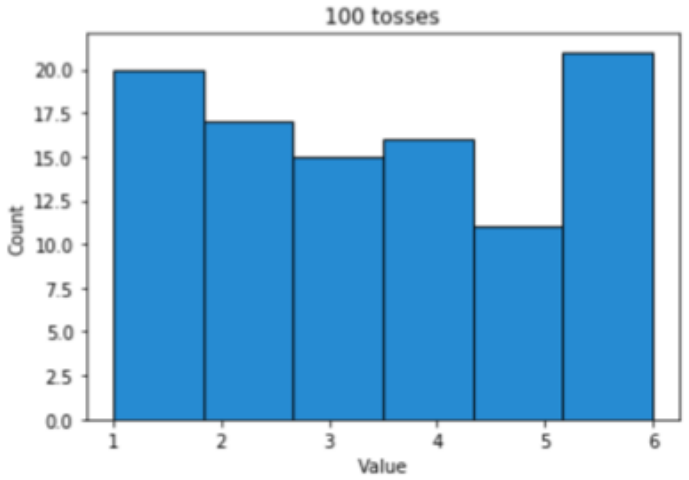
$$\sigma = \sqrt{\frac{\sum(X - \mu)^2}{N}}$$

X: individual data point
μ : mean of data points
N: total # of data points



Homework 5 (probability and statistics)

This week's homework will use Python to explore probability concepts through simulated dice rolls and to calculate statistical values from known data



	TOTAL MONS	GRASS	WETLAND	RIVERBANK	LAKE	SALTWTR BEACH	FARMLAND	UNIVERSITY	GOLF COURSE	PARK	PARKING LOT	INDUSTRIAL	RESIDENTIAL	MEADOW
NORMAL	14%	27%	9%	8%	25%	4%	29%	41%	6%	18%	43%	38%	37%	53%
BUG	11%	20%	3%	4%	25%	7%	14%	12%	0%	8%	15%	22%	16%	19%
POISON	11%	12%	3%	1%	15%	7%	6%	9%	6%	8%	8%	11%	12%	5%
WATER	10%	14%	61%	76%	35%	50%	6%	11%	6%	31%	7%	11%	5%	8%
FIRE	7%	6%	12%	2%	6%	7%	14%	2%	6%	8%	6%	7%	5%	3%
GROUND	6%	4%	6%	0%	6%	6%	14%	3%	12%	6%	4%	2%	7%	3%
GRASS	5%	6%	6%	3%	6%	11%	6%	3%	41%	4%	5%	6%	5%	8%
FIGHTING	4%	2%	6%	0%	6%	6%	6%	2%	6%	4%	3%	3%	3%	0%
ROCK	4%	3%	6%	0%	6%	4%	29%	2%	6%	4%	3%	4%	3%	3%
FAIRY	2%	3%	6%	2%	6%	11%	6%	6%	6%	3%	3%	3%	2%	3%
ELECTRIC	2%	2%	6%	0%	6%	6%	6%	10%	6%	2%	3%	3%	2%	3%
PSYCHIC	2%	2%	6%	0%	6%	6%	6%	2%	6%	3%	3%	3%	3%	0%
DRAGON	3%	2%	6%	5%	6%	6%	6%	6%	12%	3%	6%	6%	6%	3%
ICE	3%	6%	6%	6%	6%	6%	6%	3%	6%	2%	3%	3%	6%	6%
GHOST	6%	6%	6%	6%	6%	6%	6%	6%	6%	6%	6%	6%	6%	6%

1. Download the Homework 5 file from the Canvas website
2. Complete the homework by (1) filling in your own code where you see ??? and where asked to give a second example and (2) answering questions in "markdown" cells
3. Submit homework through Canvas using proper file names/types

HW 5

- **HW5 is due 11:59 pm on Wednesday (May 12th)**
- Complete all codes and answer questions (inside the notebook)
- Rename the notebook file with your name
- Submit via Canvas

Design Proposal
(several helpful thoughts)

Design under constraints

- You never have **unlimited**

  time, money, etc.

- Some constraints **overlap**

  (e.g. small and lightweight)

- Some constraints **conflict** with each other (e.g. low cost and low weight)

Mission statement

- Describe the **overarching** goal(s)
- Doesn't necessarily included quantified requirements

- Usually given (in part) by a client

Design requirements…

- Quantify your goals
- Guide you towards solutions
- Provide justifications for design decisions

Source of design requirements

- Sometimes provided by a client (directly or indirectly) or external source (e.g. government regulation)
- Usually requires work to get “hard” numbers
- Typically are “minimum” values (i.e. you would expect to do “better”)

Example 1: Sizing a pizza

- **Okay:** The pizza should be pizza size
- **Better:** The pizza should fit in the box
- **Better x2:** The pizza should have a diameter of 12” +/- 0.5” and height less than 2”

Example 2: Car frame

- **Okay:** The car frame should survive a crash
- **Better:** The car frame should not deform when hit by another car
- **Better x2:** The frame should deflect less than 1” when hit by a 1.5 ton vehicle traveling at 10 mph

Example 3: Storing medicine for shipping

- **Okay:** The medicine stays cool
- **Better:** The medicine stays below 50°F
- **Better x2**: The medicine's internal temperature stays in the range of 20–50°F, with a median temperature of 40°F

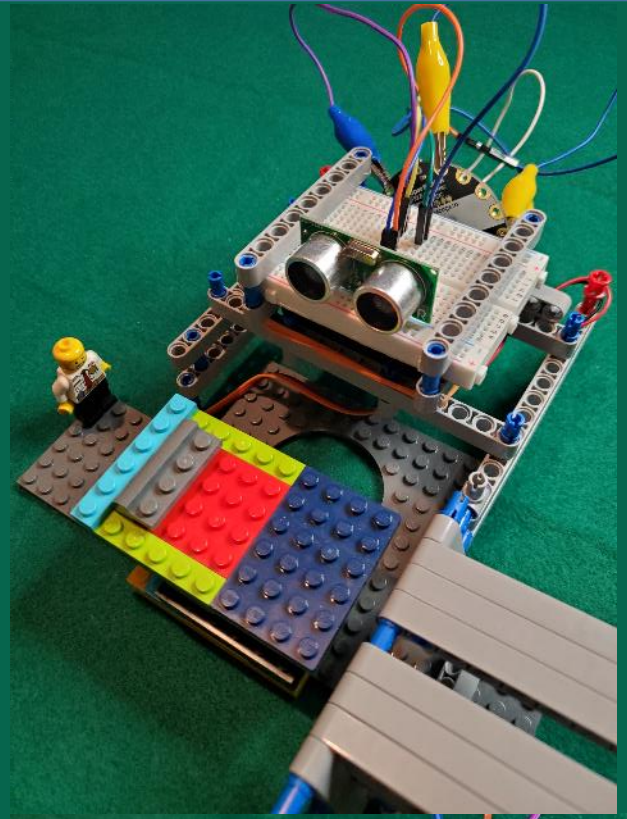
Design Challenge

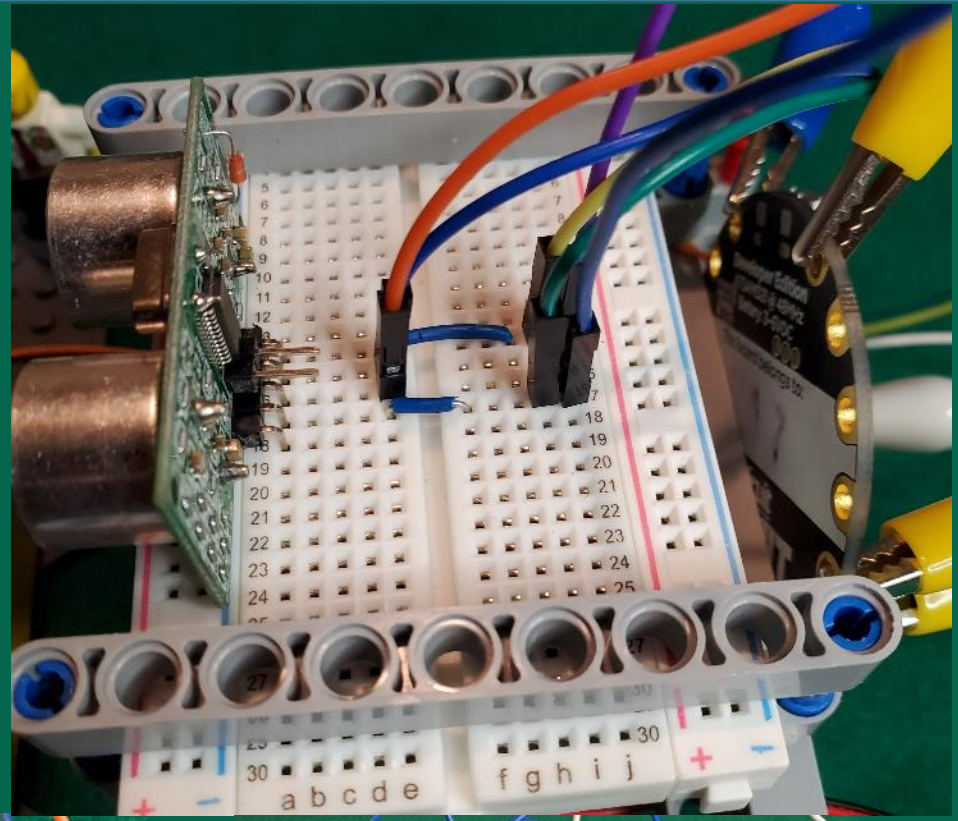
Example project ideas: Sound

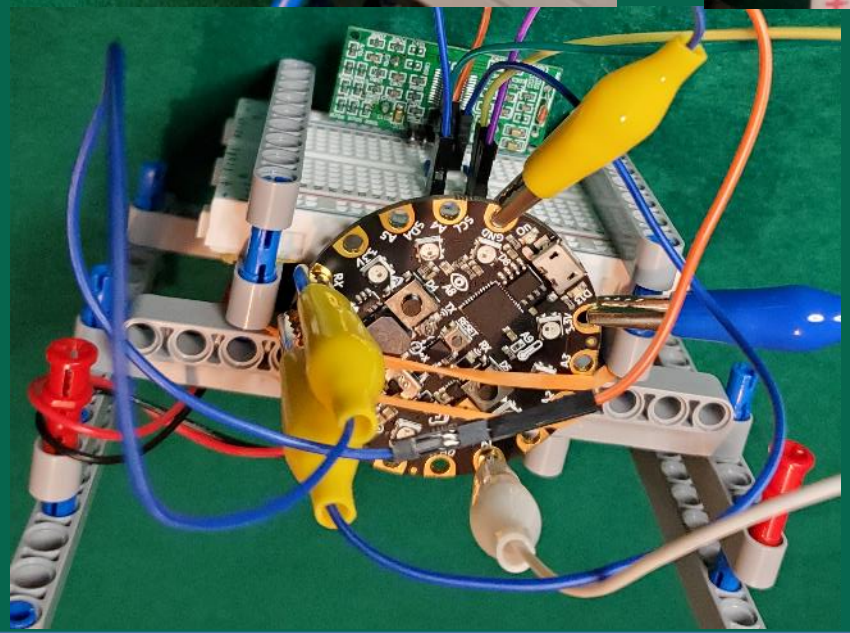
- **Idea:** Provide a non-verbal warning to runners with headphones if a loud car/truck is approaching
- **Input:** Sound sensor => Frequency data
- **Output:** Vibrating motor

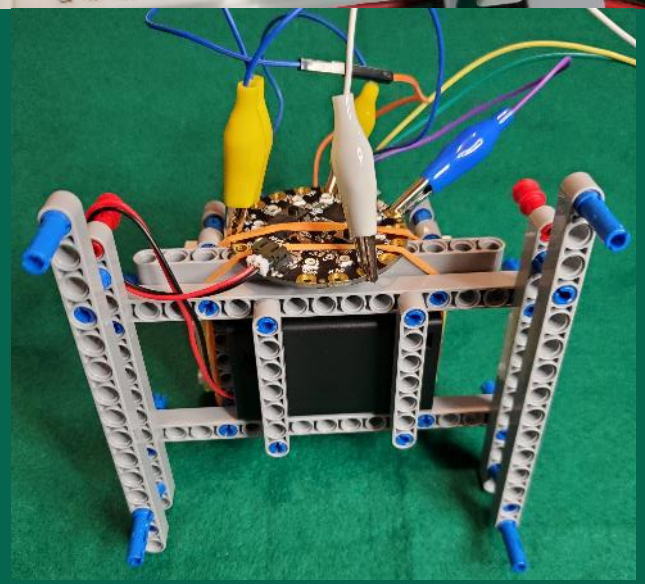
Example project ideas: Accelerometer

- **Idea:** Increase visibility (and fun) for snowboarders at night by adding a motion-controlled light display
- **Input:** Accelerometer => Detect motion
- **Output:** LEDs









1 Create the code

sketch_nov05a | Arduino 1.8.7 Hourly Build 2018/10/24 04:33

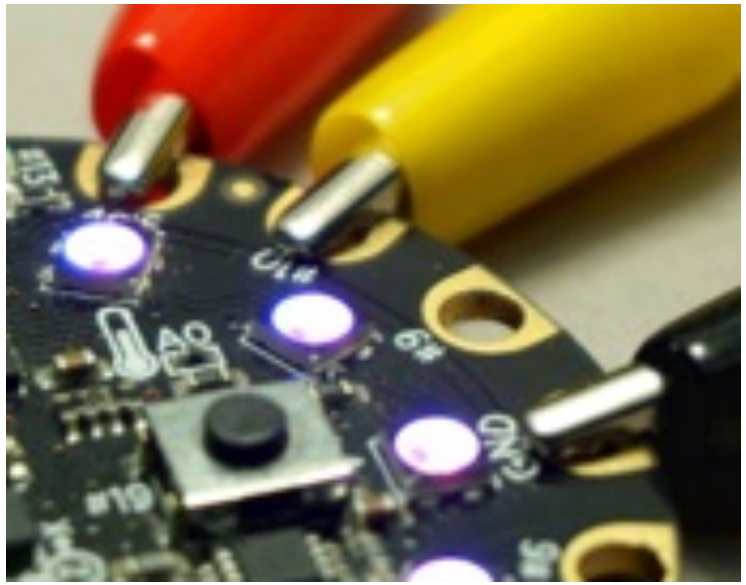
File Edit Sketch Tools Help

sketch_nov05a §

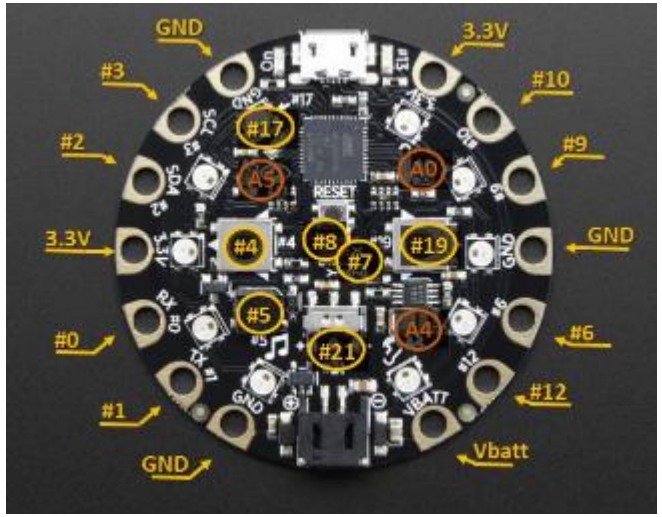
```
#include <Servo.h>              // Add library
Servo servo_ENG10;               // Define any servo name

void setup() {
  servo_ENG10.attach (9);          // Define the servo signal pins (PWM 3-5-6-9-10-11)
}
void loop() {
  servo_ENG10.write (90);          // Turned to 90 degrees
}
```

6 Adafruit Circuit Playground on COM4

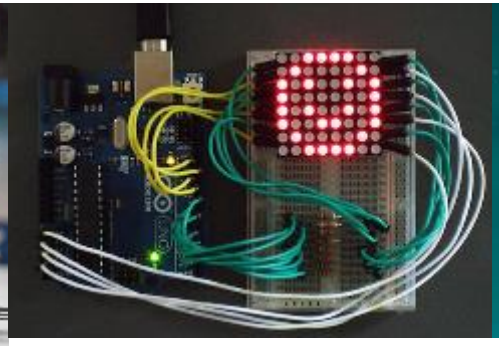
2 Make the connections

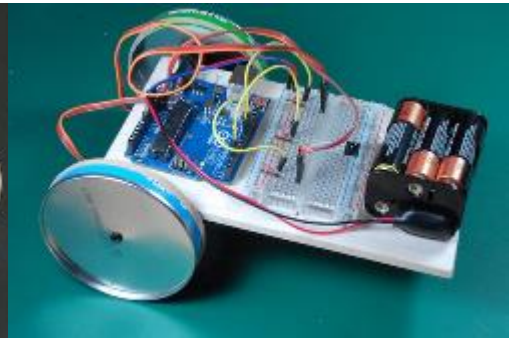


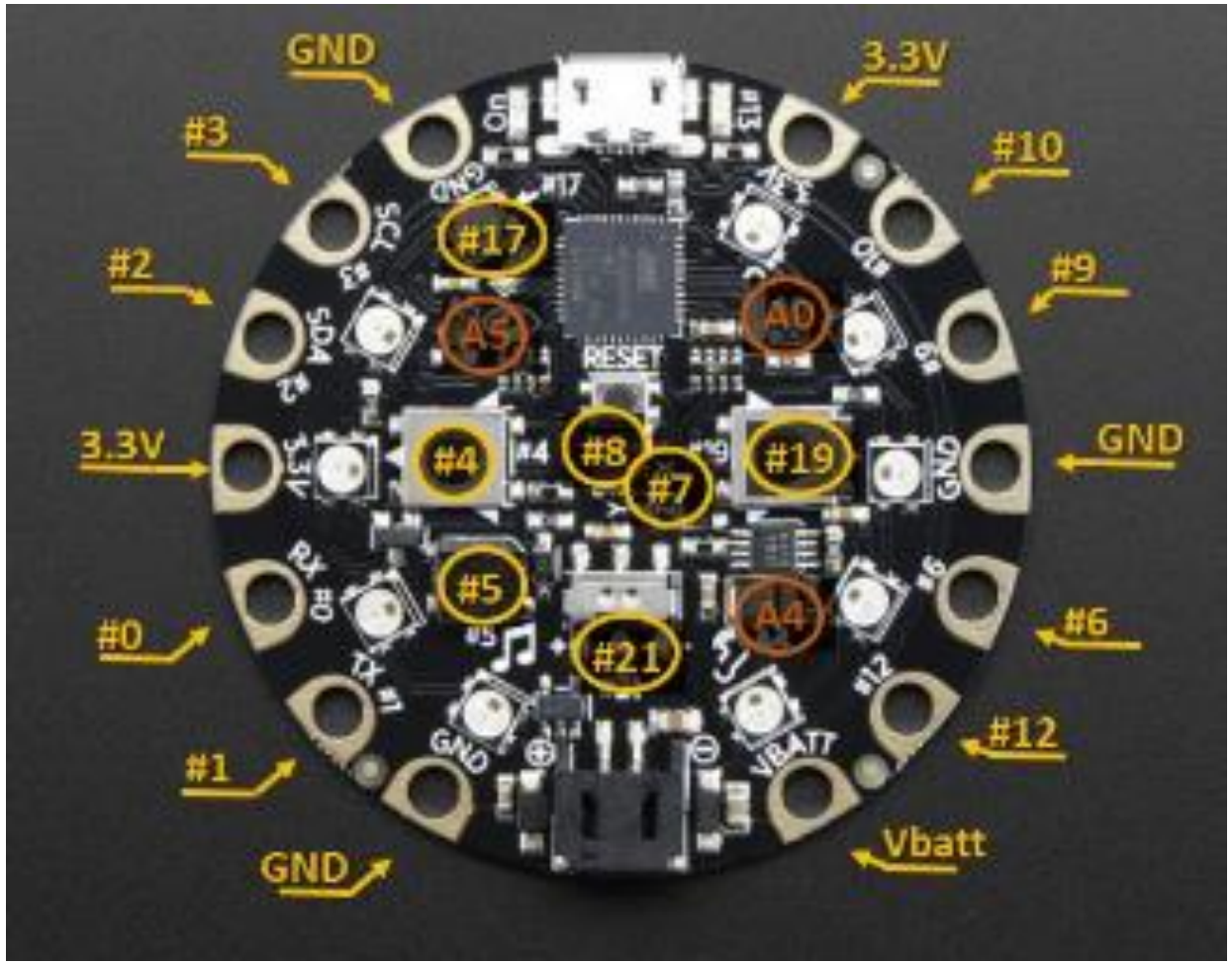
3 Upload the code



4 Test it

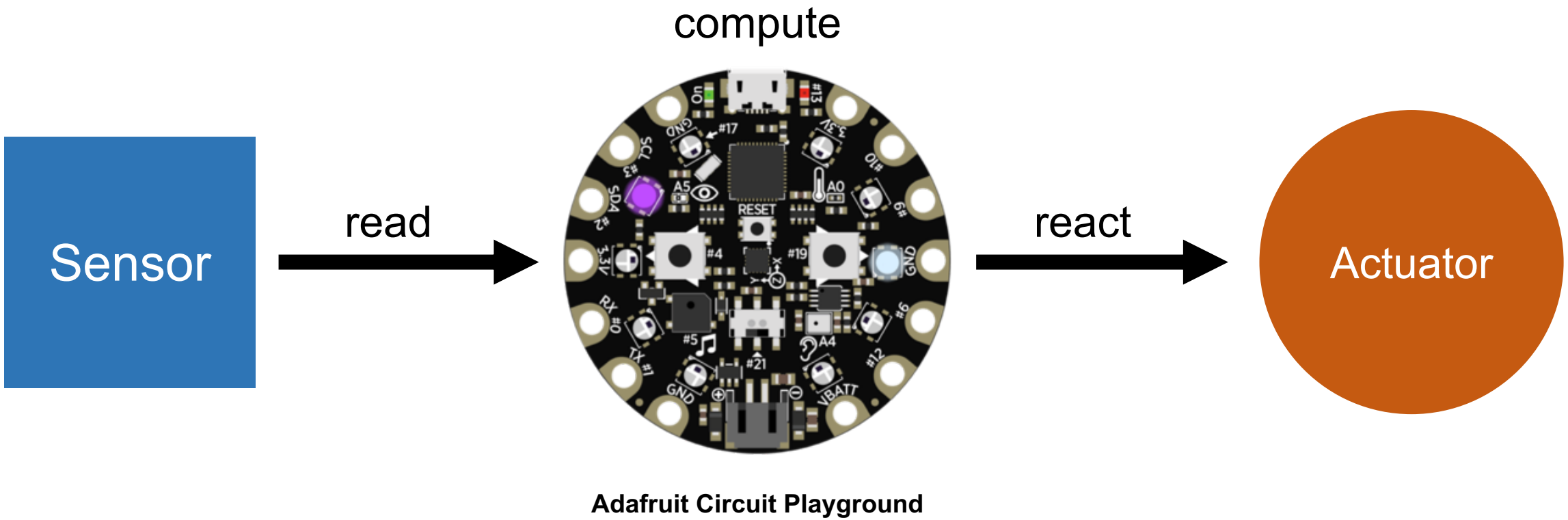






Arduino External Outputs

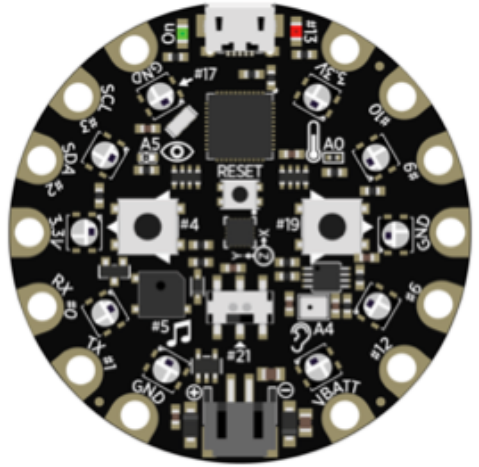
Arduino Microcontrollers



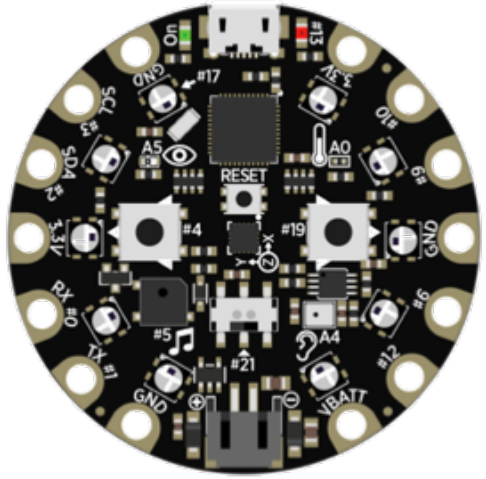
Adafruit Circuit Playground

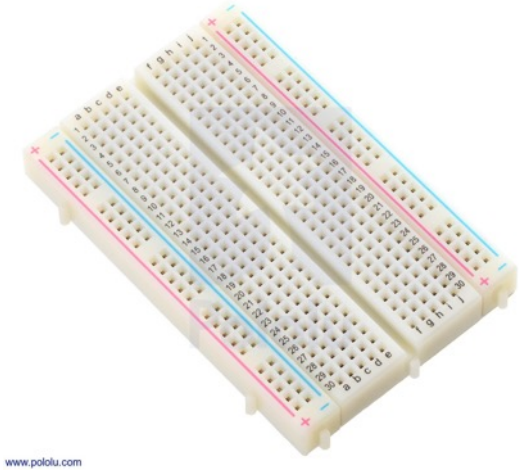
Output Options (Actuators)

- **Built-in devices on the Circuit Playground**
  - Speaker
  - LEDs
- **Today's external actuator**
  - Servo motor



Other Circuit Building **Tools**

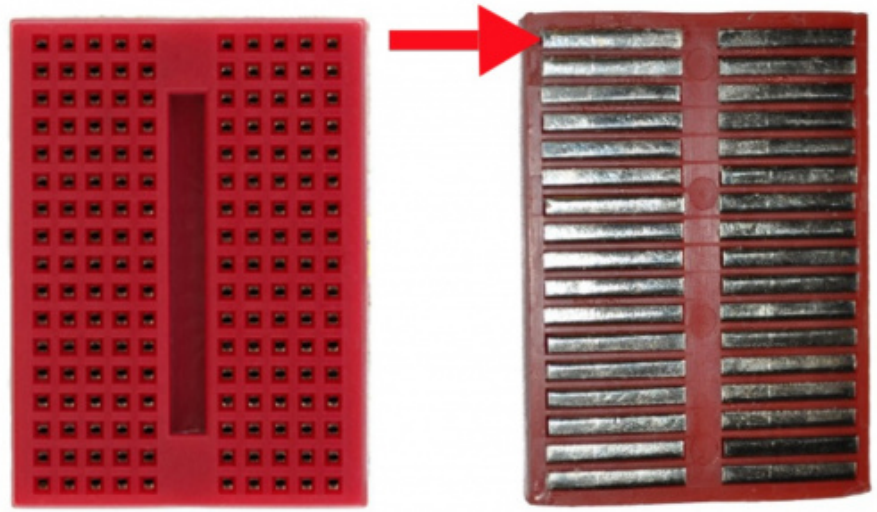




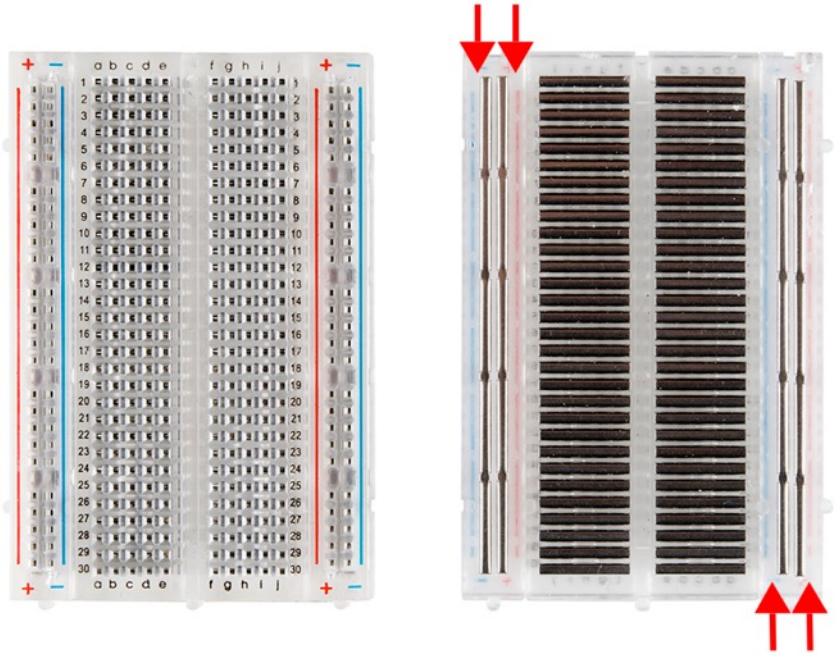
- CPX
- Breadboard
- Jumper wires and alligator clips

Breadboards

Fundamental tool to build circuits by allowing multiple connections between devices



Metal connects each row beneath the plastic which allows for connections between each element plugged into the ***terminal strips*** (rows)



Most breadboards also have ***power rails*** which are two columns on each end that are connected to each other

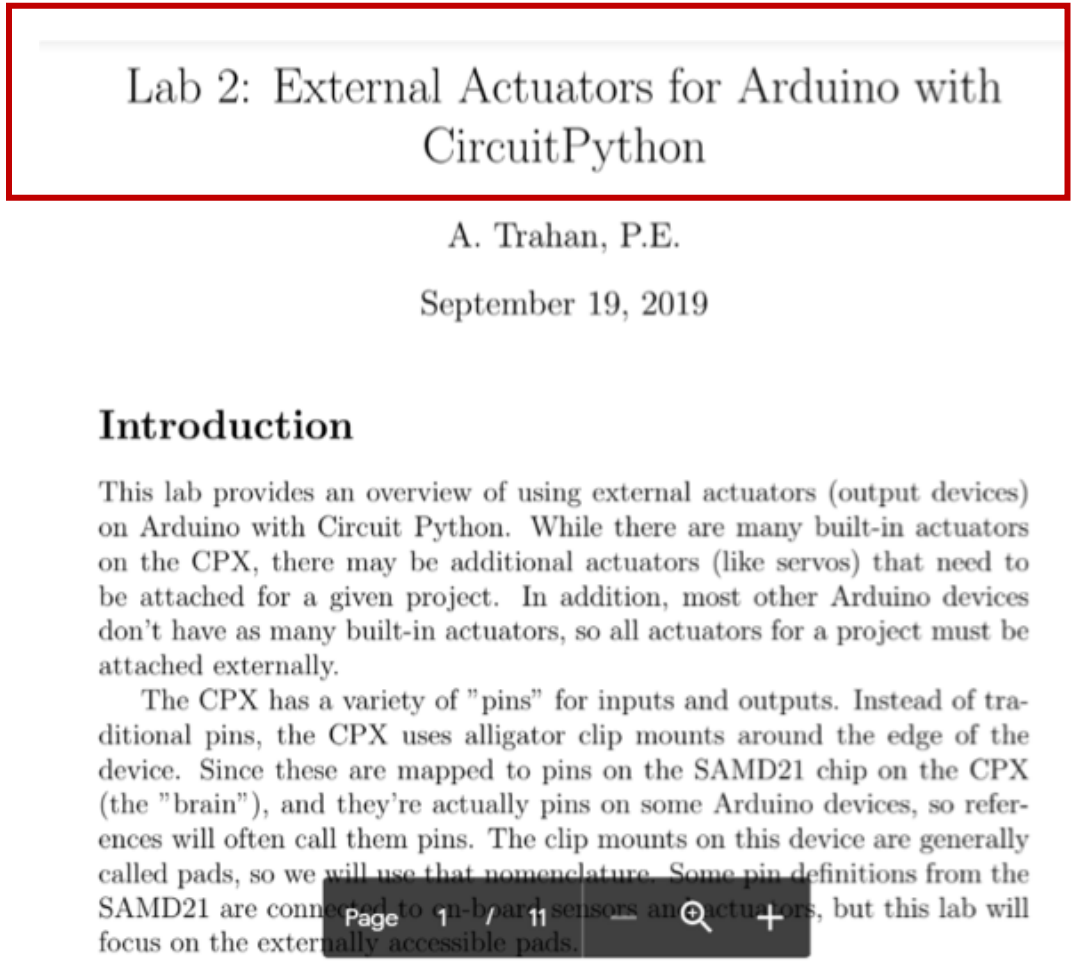
https://learn.sparkfun.com/tutorials/how-to-use-a-breadboard/all

Arduino Tutorial: External Actuators

We have to finish parts 2.3, 3.1 and 3.2 from this tutorial (**Lab 2**) begore we will move to **Lab 3**.

Main steps are:

1. **Download the library bundle (Bundle version 6.X, see next slide) and save it on your computer**
2. **Use certain files from this bundle when needed by imported those files directly to your CIRCUITPY (library)**



Lab 2: External Actuators for Arduino with CircuitPython

A. Trahan, P.E.

September 19, 2019

Introduction

This lab provides an overview of using external actuators (output devices) on Arduino with Circuit Python. While there are many built-in actuators on the CPX, there may be additional actuators (like servos) that need to be attached for a given project. In addition, most other Arduino devices don't have as many built-in actuators, so all actuators for a project must be attached externally.

The CPX has a variety of "pins" for inputs and outputs. Instead of traditional pins, the CPX uses alligator clip mounts around the edge of the device. Since these are mapped to pins on the SAMD21 chip on the CPX (the "brain"), and they're actually pins on some Arduino devices, so references will often call them pins. The clip mounts on this device are generally called pads, so we will use that nomenclature. Some pin definitions from the SAMD21 are connected to on-board sensors and actuators, but this lab will focus on the externally accessible pads.

Full Arduino tutorial is available on Canvas in Tutorials Module **"Arduino CPX Tutorials" zip** Folder

2.3 (Lab 2) Using a Buzzer via *SimpleIO (Optional)*

We can also use **a different library** to accomplish the same buzzer actuation

1. Open Serial and hit CTRL+C to find the version of your software (**see figure below**)
2. Download the corresponding ***bundle*** *(***Bundle version 6.X***)* from https://circuitpython.org/libraries
3. **Save bundle to desktop** then unzip folder (**extract all**)
   Note: You can put this folder in other areas but may run into "pathway too long" errors
4. In unzipped folder > lib > search: **simple.io** → copy and paste in **CIRCUITPY > lib**
5. Add new code from tutorial using **simple.io** and save as ***code.py***

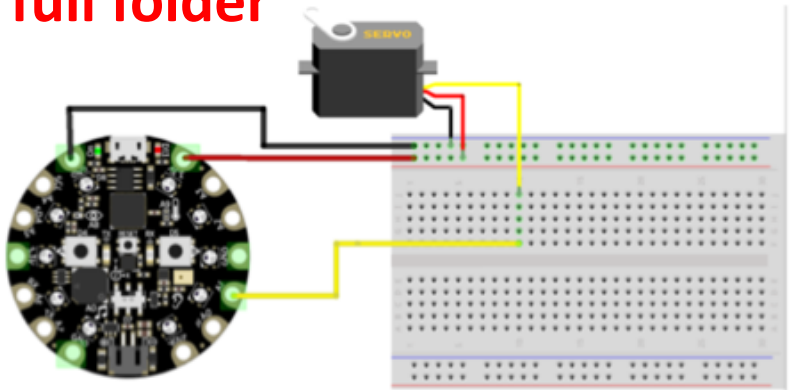
Figure 3: Finding the CircuitPython version with Mu

3.1 (Lab 2) Responding to Sensors on the CPX (Optional)

This example demonstrates how **the amount of light changes the position of servo**

Our first several examples relied on set schedules for actuator behavior. Here we will learn how we can use inputs from sensors to control external actuators

1. **First, we will need more software from the bundle—copy the full folder *adafruit_motor* into CIRCUITPY > lib**
2. Add **a fin** to the servo motor top
3. Connect the servo motor to CPX using the following:
    1. **Brown** is ground, **connect to GND**
    2. **Red** is power, connects to **3.3V**
    3. **Orange** is PWM (Pulse-Width Modulation, controls the motor), **connect to A1**
4. Input the new code from the tutorial and save as *code.py* on your CPX
    1. If not working, check Serial w/ Ctrl+D
    2. If it says not working because safe mode, eject CPX and reload



We can do this one without the breadboard!

https://learn.adafruit.com/adafruit-circuit-playground-express/circuitpython-servo

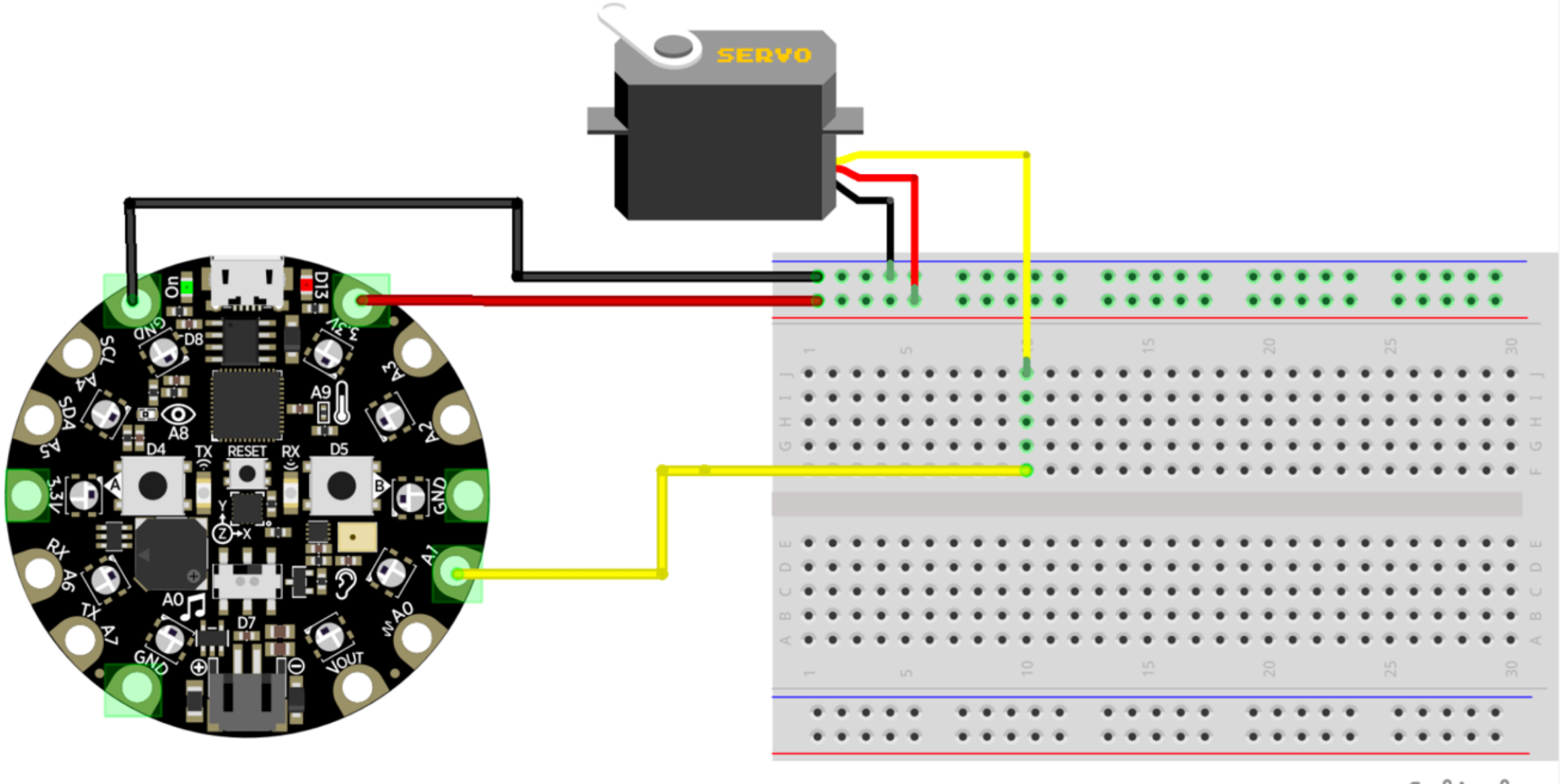
3.1 (Lab 2) Responding to Sensors on the CPX (Optional)

This example demonstrates how **the amount of light changes the position of servo. There are minor changes into the Tutorial program. Please consider to check this program!**

```
import board
import pulseio
from adafruit_circuitplayground.express import cpx
from adafruit_motor import servo
from time import sleep
# Setup Section
pwm = pulseio.PWMOut(board.A1, duty_cycle=2 ** 15, frequency=50)
servo = servo.Servo(pwm, min_pulse=750, max_pulse=2600)
# Function Section
def light_to_servo_pos(light):
    light_max = 320
    return 180 - ((light/light_max) * 180)
# Loop Section
while True:
    servo.angle = light_to_servo_pos(cpx.light)
    print(servo.angle)
    sleep(0.5)
```

3.1 (Lab 2) Responding to Sensors on the CPX (Optional)

This example demonstrates how **the amount of light changes the position of servo**



3.2 (Lab 2) Another example that shows how **servo rotates between 0 and 90 degrees**

Our first examples all relied on set schedules for actuator behavior. Here we will learn how we can use inputs from sensors to control external actuators

1. **First, we will need more software from the *bundle*—copy the full folder *adafruit_motor* into CIRCUITPY > lib**
2. Add **a fin** to the servo motor top
3. Connect the servo motor to CPX using the following:
    1. **Brown** is ground, **connect to GND**
    2. **Red** is power, connects to **Vout**
    3. **Orange** is PWM (Pulse-Width Modulation, controls the motor), **connect to A2**
4. Input the new code (**that Fabian sent, see next slide**) and save as *code.py on your CPX*
    1. If not working, check Serial w/ Ctrl+D
    2. If it says not working because safe mode, eject CPX and reload

We can do this one without the breadboard!

https://learn.adafruit.com/adafruit-circuit-playground-express/circuitpython-servo

3.2 (Lab 2) Program for Servo Motor to show how servo rotates between 0 and 90 degrees

```
import time
import board
import pulseio
from adafruit_motor import servo

# create a PWMOut object on Pin A2.
pwm = pulseio.PWMOut(board.A2, duty_cycle=2 ** 15, frequency=50)

# Create a servo object, my_servo.
my_servo = servo.Servo(pwm)

while True:
    for angle in range(0, 90, 5):  # 0 - 180 degrees, 5 degrees at a time.
        my_servo.angle = angle
        time.sleep(0.05)
    for angle in range(90, 0, -5): # 180 - 0 degrees, 5 degrees at a time.
        my_servo.angle = angle
        time.sleep(0.05)
```

Input Options (Sensors)

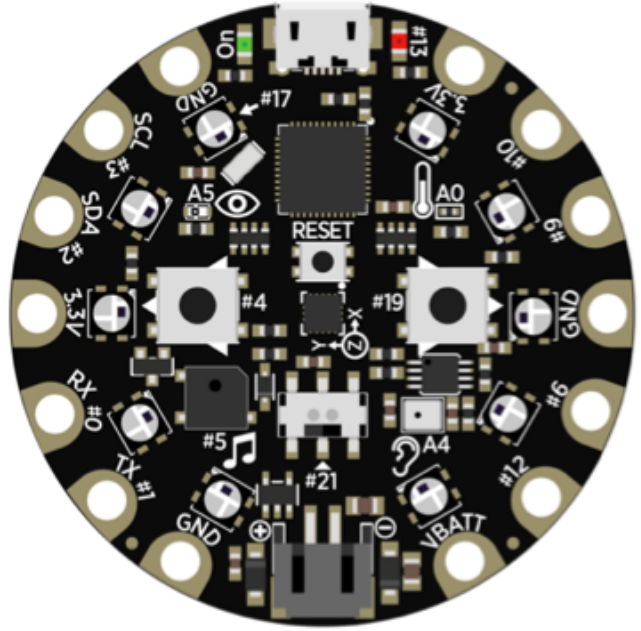
Built-in sensors on the Circuit Playground:

Sound: Noise levels, sound signals, frequencies

Temperature

Light: Brightness, color (using LEDs)

Accelerometer: Acceleration, tilt



Today's external sensor

Proximity: Measures distance of object from sensor by detecting ultrasonic wave reflection

Libraries for different sensors (example)

- In order to figure out everything regarding **proximity sensor (for example)** please go to:

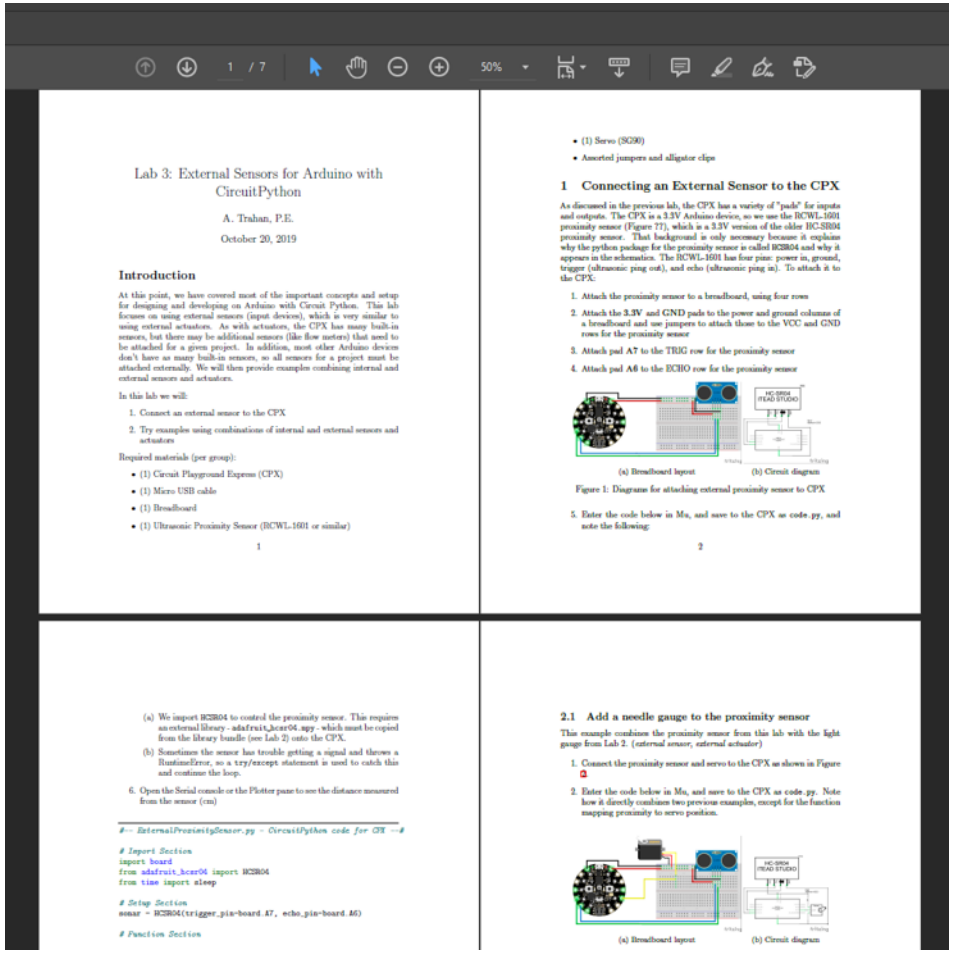
- https://learn.adafruit.com/using-vcnl4010-proximity-sensor/

- Read everything carefully, **press "Pinouts"** on the right bottom corner and again read everything carefully.

- Next, **press "Arduino"** on the right bottom corner, and check what kind of steps you have to do in order to install Adafruit_VCNL4010 library and a demo.

Arduino Tutorial: External Sensors

Lab 3: External Sensors for Arduino with CircuitPython



Lab 3: External Sensors for Arduino with CircuitPython

A. Trahan, P.E.

October 20, 2019

Introduction

At this point, we have covered most of the important concepts and setup for designing and developing on Arduino with Circuit Python. This lab focuses on using external sensors (input devices), which is very similar to using external actuators. As with actuators, the CPX has many built-in sensors, but there may be additional sensors (like flow meters) that need to be attached for a given project. In addition, most other Arduino devices don't have as many built-in sensors, so all sensors for a project must be attached externally. We will then provide examples combining internal and external sensors and actuators.

In this lab we will:

1. Connect an external sensor to the CPX
2. Try examples using combinations of internal and external sensors and actuators

Required materials (per group):

- (1) Circuit Playground Express (CPX)
- (1) Micro USB cable
- (1) Breadboard
- (1) Ultrasonic Proximity Sensor (RCWL-1601 or similar)
- (1) Servo (SG90)
- Assorted jumpers and alligator clips

1 Connecting an External Sensor to the CPX

As discussed in the previous lab, the CPX has a variety of "pads" for inputs and outputs. The CPX is a 3.3V Arduino device, so we use the RCWL-1601 proximity sensor (Figure ??), which is a 3.3V version of the older HC-SR04 proximity sensor. That background is only necessary because it explains why the python package for the proximity sensor is called HCSR04 and why it appears in the schematics. The RCWL-1601 has four pins: power in, ground, trigger (ultrasonic ping out), and echo (ultrasonic ping in). To attach it to the CPX:

1. Attach the proximity sensor to a breadboard, using four rows
2. Attach the **3.3V** and **GND** pads to the power and ground columns of a breadboard and use jumpers to attach those to the VCC and GND rows for the proximity sensor
3. Attach pad **A7** to the TRIG row for the proximity sensor
4. Attach pad **A6** to the ECHO row for the proximity sensor

(a) Breadboard layout (b) Circuit diagram

Figure 1: Diagrams for attaching external proximity sensor to CPX

5. Enter the code below in Mu, and save to the CPX as **code.py**, and note the following:

(a) We import HCSR04 to control the proximity sensor. This requires an external library - **adafruit_hcsr04.mpy** - which must be copied from the library bundle (see Lab 2) onto the CPX.

(b) Sometimes the sensor has trouble getting a signal and throws a RuntimeError, so a **try/except** statement is used to catch this and continue the loop.

6. Open the Serial console or the Plotter pane to see the distance measured from the sensor (cm)

```
# -- ExternalProximitySensor.py - CircuitPython code for CPX --#

# Import Section
import board
from adafruit_hcsr04 import HCSR04
from time import sleep

# Setup Section
sonar = HCSR04(trigger_pin=board.A7, echo_pin=board.A6)

# Function Section
```

2.1 Add a needle gauge to the proximity sensor

This example combines the proximity sensor from this lab with the light gauge from Lab 2. (*external sensor, external actuator*)

1. Connect the proximity sensor and servo to the CPX as shown in Figure ??
2. Enter the code below in Mu, and save to the CPX as **code.py**. Note how it directly combines two previous examples, except for the function mapping proximity to servo position.

(a) Breadboard layout (b) Circuit diagram

Introduction

At this point, we have covered most of the important concepts and setup for designing and developing on Arduino with Circuit Python. This lab focuses on using external sensors (input devices), which is very similar to using external actuators. As with actuators, the CPX has many built-in sensors, but there may be additional sensors (like ow meters) that need to be attached for a given project. In addition, most other Arduino devices don't have as many built-in sensors, so all sensors for a project must be attached externally. We will then provide examples combining internal and external sensors and actuators.

Full Arduino tutorial is available on Canvas in **Arduino CPX Tutorials.zip** folder

1. (Lab 3) Connecting an External Proximity Sensor (shows the distance from the external proximity sensor in cm)

1. Attach the proximity sensor to a breadboard, using four rows

2. Attach the **3.3V** and **GND** pads to the power and ground columns of a breadboard and use jumpers to attach those to the VCC and GND rows for the proximity sensor

3. Attach pad **A7** to the TRIG row for the proximity sensor

4. Attach pad **A6** to the ECHO row for the proximity sensor

5. Enter the code below in Mu, and save to the CPX as `code.py`, and note the following: Page 3

   (a) We import `HCSR04` to control the proximity sensor. This requires an external library - `adafruit_hcsr04.mpy` - which must be copied from the library bundle (see Lab 2) onto the CPX.

   (b) Sometimes the sensor has trouble getting a signal and throws a RuntimeError, so a `try/except` statement is used to catch this and continue the loop.

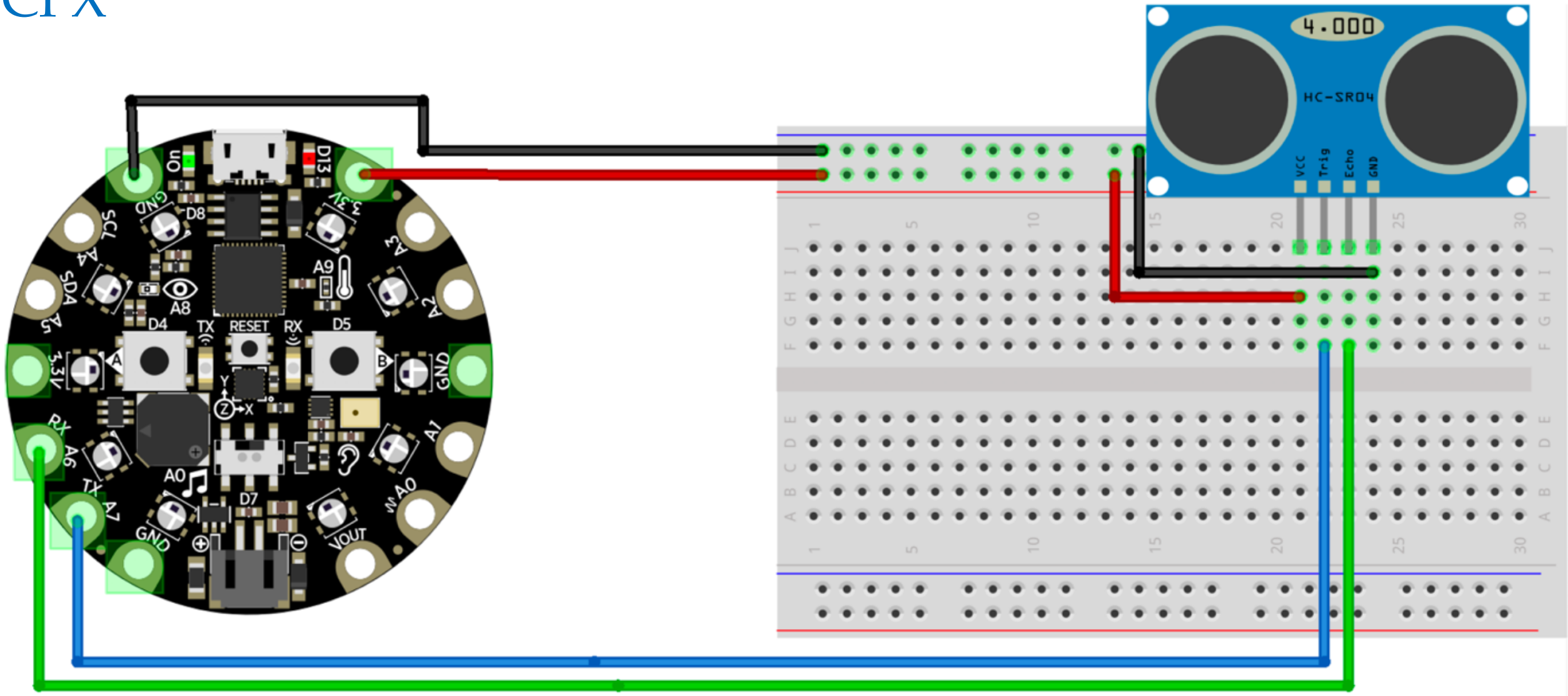
6. Open the Serial console or the Plotter pane to see the distance measured from the sensor (cm)

5. Enter the code below in Mu, and save to the CPX as **code.py**

```
#-- ExternalProximitySensor.py - CircuitPython code for CPX --#

# Import Section
import board
from adafruit_hcsr04 import HCSR04
from time import sleep

# Setup Section
sonar = HCSR04(trigger_pin=board.A7, echo_pin=board.A6)

# Function Section

# Loop Section
while True:
    try:
        print((sonar.distance,))
    except RuntimeError:
        print("Retrying!")
    sleep(0.5)
```

Import the external library "*adafruit hcsr04.mpy*" into the **lib** folder in **your CPX**

1. **(Lab 3)** Connecting an External Proximity Sensor to CPX

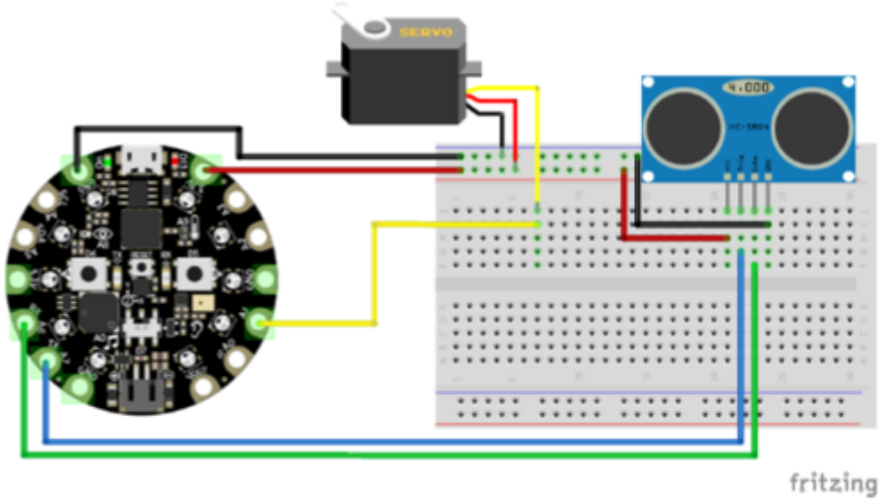


2.1 (Lab 3). Combining Ext. Sensors and Actuators "Add a needle gauge to the proximity sensor" for mapping proximity to servo position

No need to disassemble the proximity sensor setup!

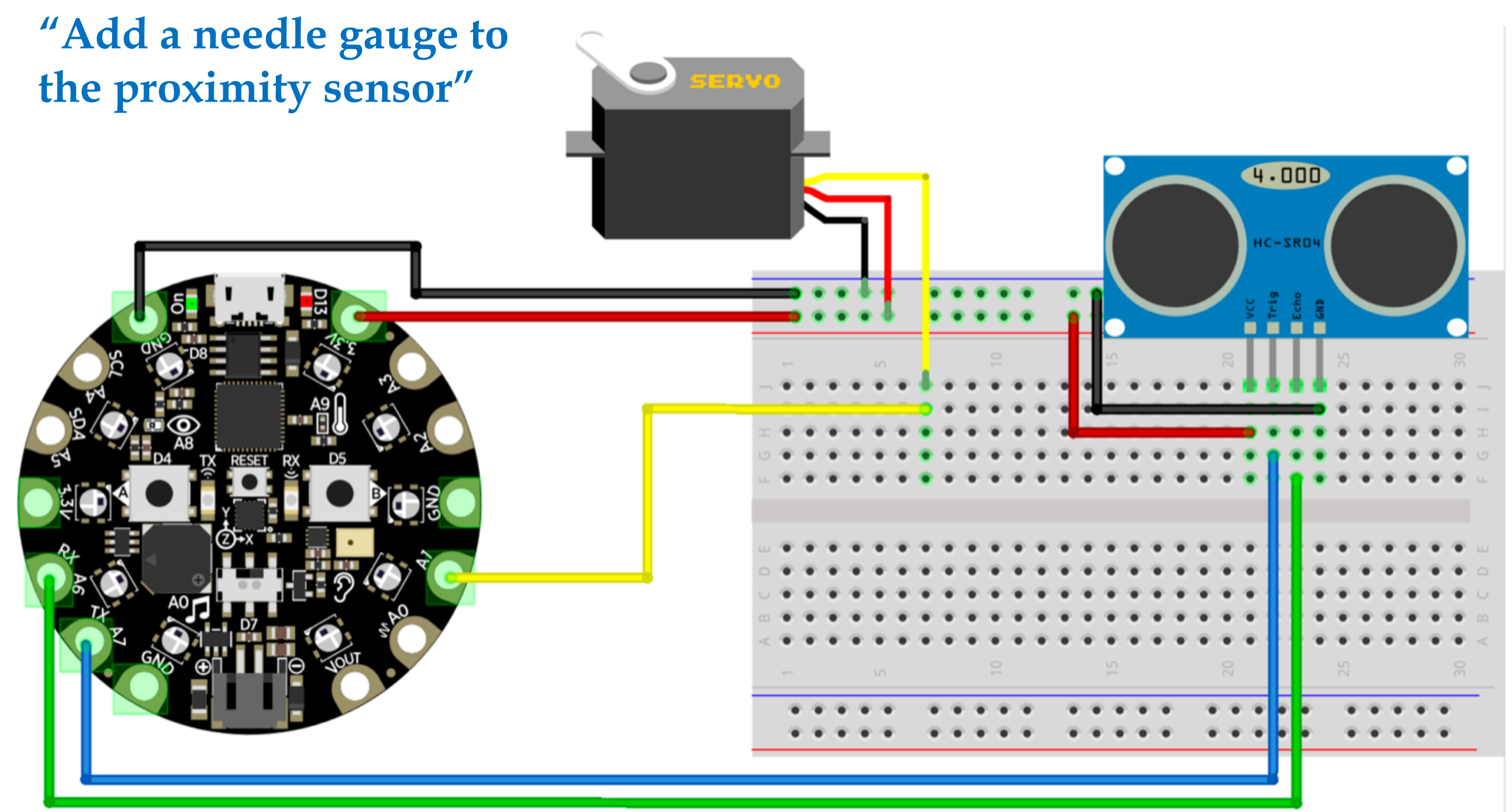
Just add the servo motor connections!

1. **Brown is ground**, connect to GND
2. **Red is power**, connects to 3.3V
3. **Yellow is control** (Pulse-Width Modulation (PWM)), connect to A1



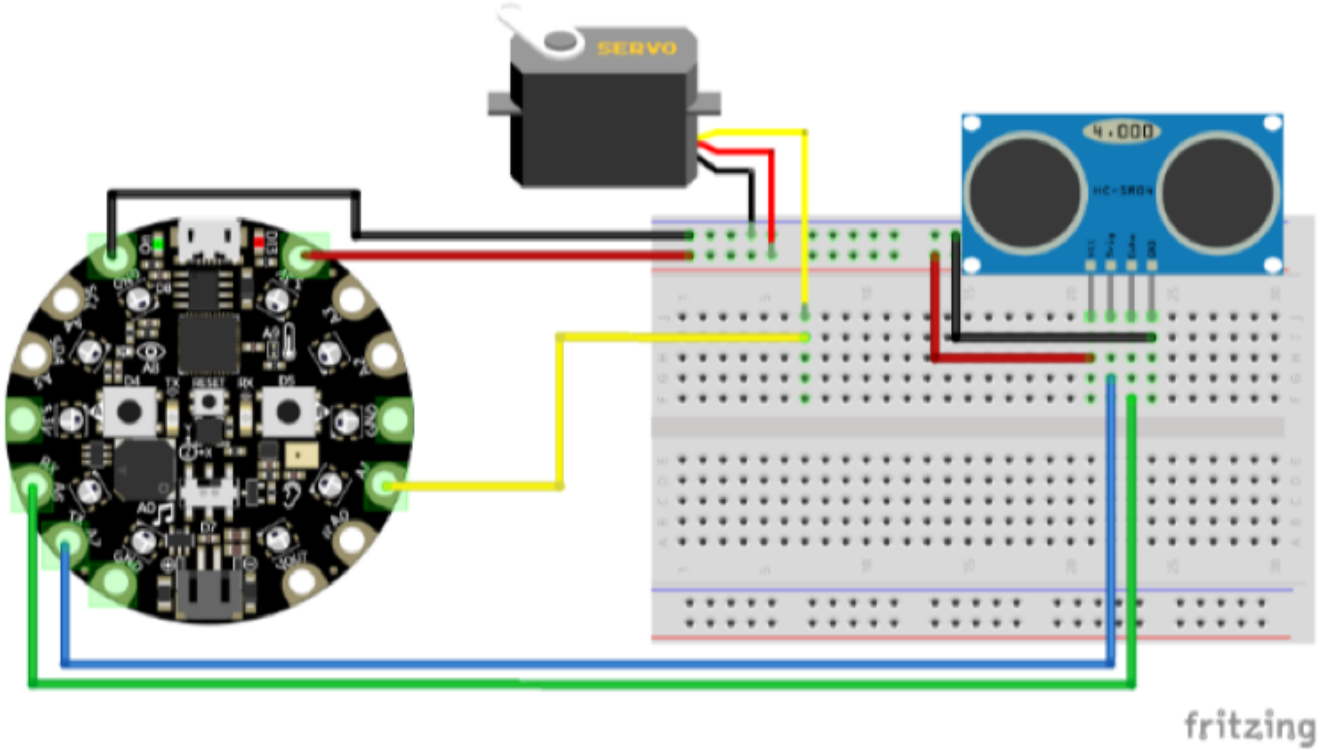
(a) Breadboard layout

“Add a needle gauge to the proximity sensor”



2.2 (Lab. 3). Multiple Actuation
"Add warning lights/sounds to the proximity sensor"

No need to disassemble the proximity sensor/servo motor setup!
Just change the code (**from pp. 6-7**) to add additional actuation of warning lights and sounds!



(a) Breadboard layout