Tutorial 1

Statistical Computation and Analysis

Spring 2024

Course Information

- The course is in English.
- The homework assignments need to be handed in in English.
- The course is in Python.
  - The codes for the examples in the lectures and tutorials will be made available on Moodle.
- Course grade:
  - Pre-lecture questions (5%)
  - Homework assignments (15%)
    - 5 assignments, due two weeks after publication (3.6% per exercise)
  - Midterm quiz (10%, magen)
    - 10.6 – both groups together?
  - Exam (70%)
    - On computers on campus
    - Open moodle except previous tests

Tutorial Outline

- Python (libraries + google colab)
- Data and Population
- Central Tendency
- Variance
- Skewness
- Kurtosis

Practice Google Colab Notebook

- **Python basics**

https://colab.research.google.com/github/cs231n/cs231n.github.io/blob/master/python-colab.ipynb

- **Code for this tutorial:**

Moodle – Tutorial1_2024.ipynb

Useful Python Libraries

- **NumPy:** provides support for arrays, matrices, and mathematical operations on them.
- **Pandas:** provides tools for data manipulation and analysis, including data structures and functions for transforming, cleaning, and analyzing datasets.
- **SciPy:** builds on NumPy and provides additional functionality for scientific computing, including statistical functions.
- **Matplotlib:** data visualization.
- **Seaborn:** more data visualization.
- **Scikit-learn:** provides algorithms for data analysis, including statistical models such as linear regression.
- **Statsmodels:** also provides a wide range of statistical models and methods, including regression analysis and hypothesis testing.
- **Math:** math operations.

Data Types

- **Integers**
- **Floating-point numbers**
- **Strings**
- **Boolean**
- **Lists:** ordered sequences of elements. They can contain any combination of data types and are defined using square brackets, e.g., l = [1, 2, "three"].
- **Tuples:** similar to lists, but they are immutable (cannot be changed). They are defined using parentheses, e.g., t = (1, 2, "three").
- **Dictionaries:** used to store data values in key:value pairs. They are defined using curly brackets, e.g., d = {"brand": "Ford", "model":"Mustang“}.

Python

- **Import libraries and external functions:**
  - import numpy as np (np.array([1, 2, 3])).
  - from file_name import function_name
  - from numpy import * - imports all numpy functions, not recommended.
- **Functions:**
  - def function_name(input arguments)
  - Defined before function call / separate code and imported.
- **Strings:**
  - Concatenation: ‘hello’ + str(var) + ‘_’ + ‘world’
  - fstring: f‘hello{var }_world’

Python

- **Booleans:**

MATLAB	PYTHON
true	True
false	False
&	and
|	or
~	not
~=	!=

- **Indexing:**

MATLAB	PYTHON
Starts from 1	Starts from 0
array(1:3)	array[0:2]

Python

- **For loops:**
  - for i in range(5):

    print(i)

  Output:

  0

  1

  2

  3

  4

Python

- **For loops:**
  - animals = ['cat', 'dog', 'monkey’] *(list)*

```
for animal in animals:
   print(animal)
```

Output:

cat

dog

monkey

Python

- **For loops:**
  - for i, animal in enumerate(animals):

    print(f"Number {i}, animal {animal}")

  Output:

  Number 0, animal cat

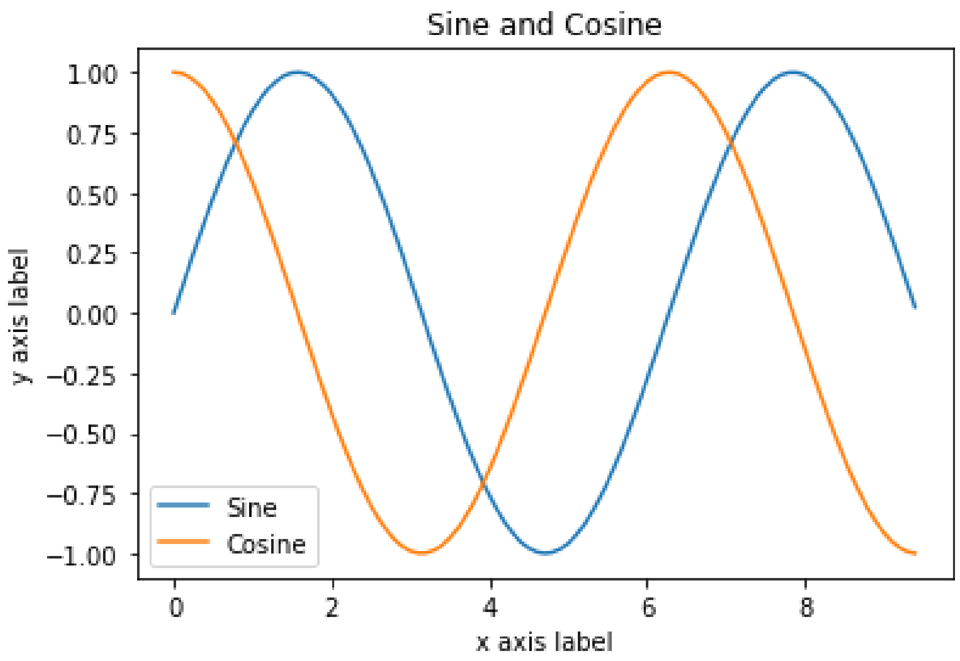
  Number 1, animal dog

  Number 2, animal monkey

Python

- **Plots:**

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

x = np.arange(0, 3 * np.pi, 0.1)
y_sin = np.sin(x)
y_cos = np.cos(x)

# Plot the points using matplotlib
plt.plot(x, y_sin)
plt.plot(x, y_cos)
plt.xlabel('x axis label')
plt.ylabel('y axis label')
plt.title('Sine and Cosine')
plt.legend(['Sine', 'Cosine'])
```



Data and Population

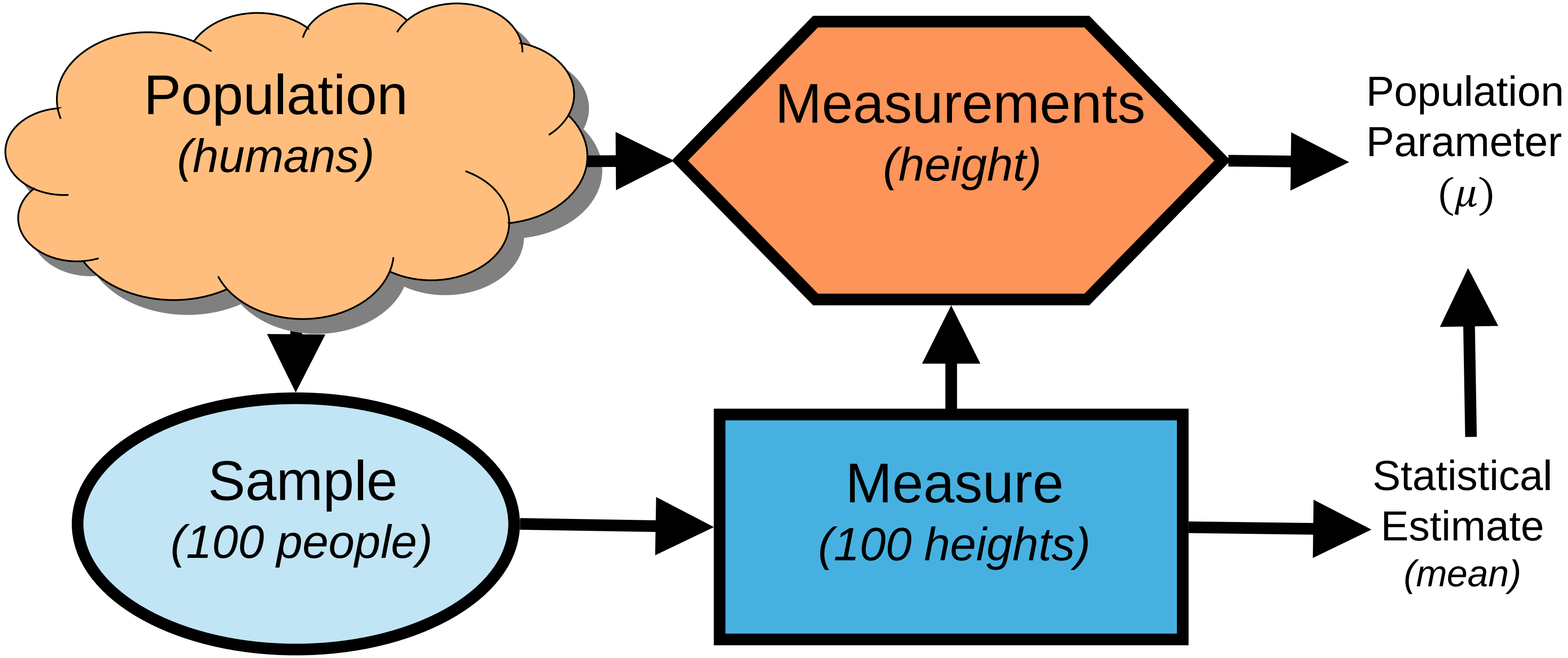
- Types of data:
  - Continuous (e.g., weight, height, time)
  - Categorical (e.g., gender, color)
  - Ordinal (e.g., morning, afternoon, night; baby, child, teenager, adult)
- Population
  - A “pool” from which we take a sample.
  - Depends on the question.
  - We have no access to the entire population but aim to describe it.
    - Can be described using the probability distribution or summarized using one or two numbers.
- Data
  - Samples from the distribution that describes the population
  - Finite

Data and Population

Goal: to infer the population parameter from the sampled data.

- **Parameter:** numerical value that describes a population. It is fixed and unknown but can be estimated.
- **Statistic:** numerical value that describes a sample. It varies from sample to sample and can be calculated from sample data.
- **Estimator:** a formula or rule that uses sample data to estimate an unknown parameter. Provides a best guess of the true value of the parameter, based on the available data.

Data and Population



Data and Population

Good sample:

- Representative
- Independent
- Identically Distributed

Example 1

A study conducted on 200 cats in Beer Sheva found that they sleep 14 hours a day.

- What is the sample in this study?
- What is the population?

Example 2

Are the following observations independent?

- Heights of 50 sixth grade students.

- If the population is heights of the elementary students in the school – observations are not independent.

- If the population is heights of the sixth grade students in the school – observations are independent.

Data and Population

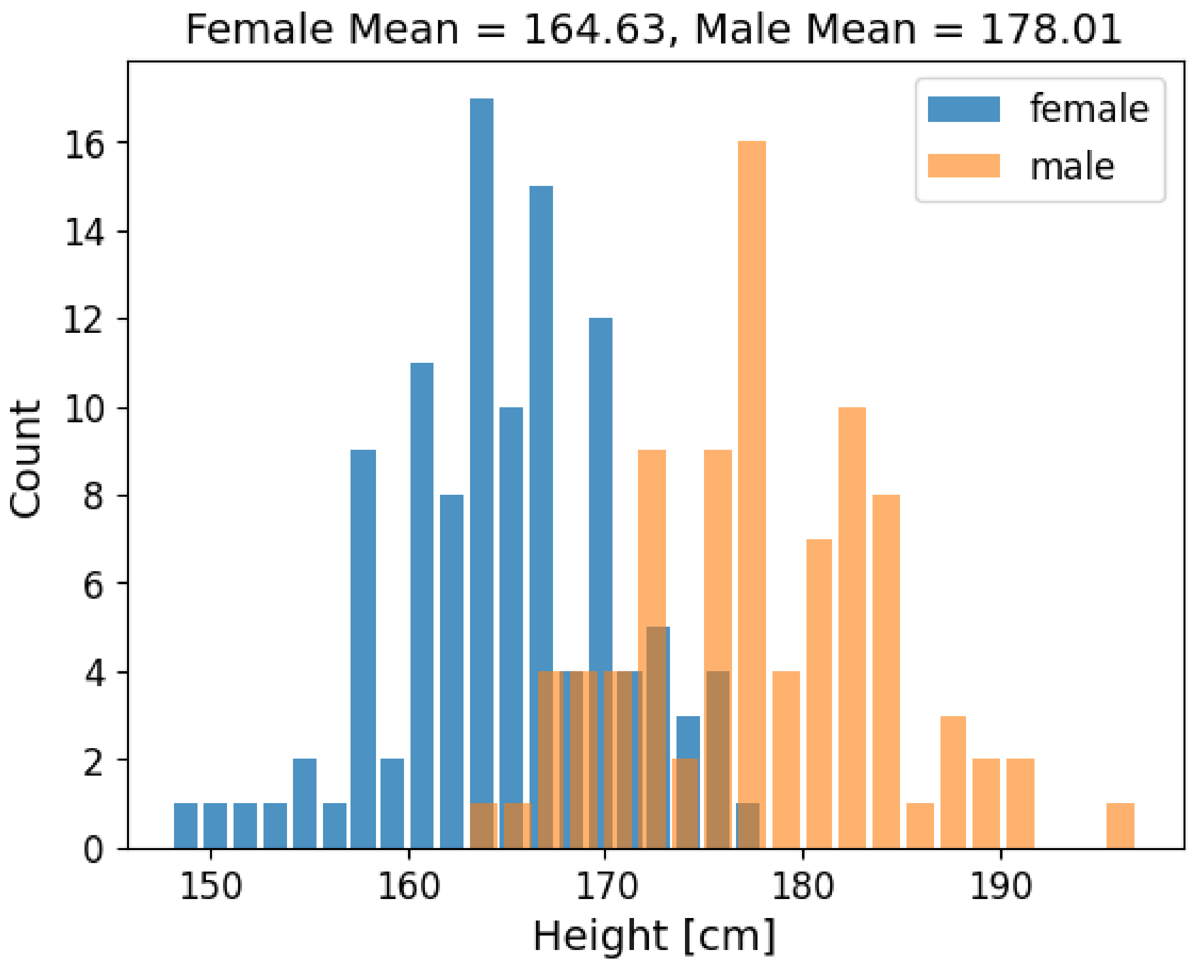
- Example: What is the average height of male and female university students?
  - Population – university students.
    - Parameter = real height average of university students.
  - Data – a sample of 200 students we collected.
    - Estimator – the average of the 100 heights for each gender.

```
     subject sex  height
0          1   M     182
1          2   F     161
2          3   F     161
3          4   M     177
4          5   F     157
..       ...  ..     ...
195      196   M     175
196      197   M     180
197      198   M     175
198      199   M     181
199      200   M     177

[200 rows x 6 columns]
```

- Load data:
- data = pd.read_csv(io.BytesIO(uploaded['Davis.csv']))

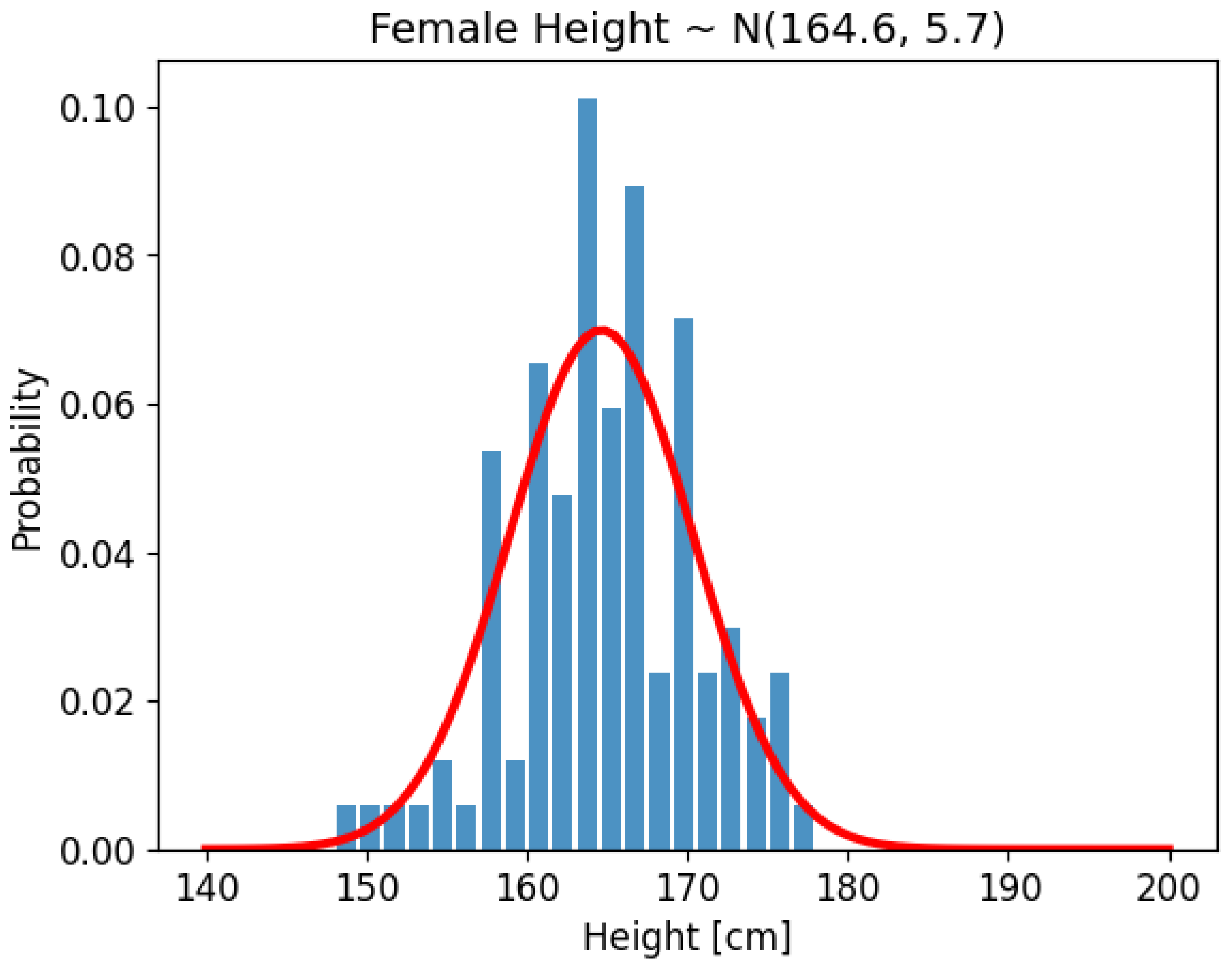
Data and Population

- Examine the data



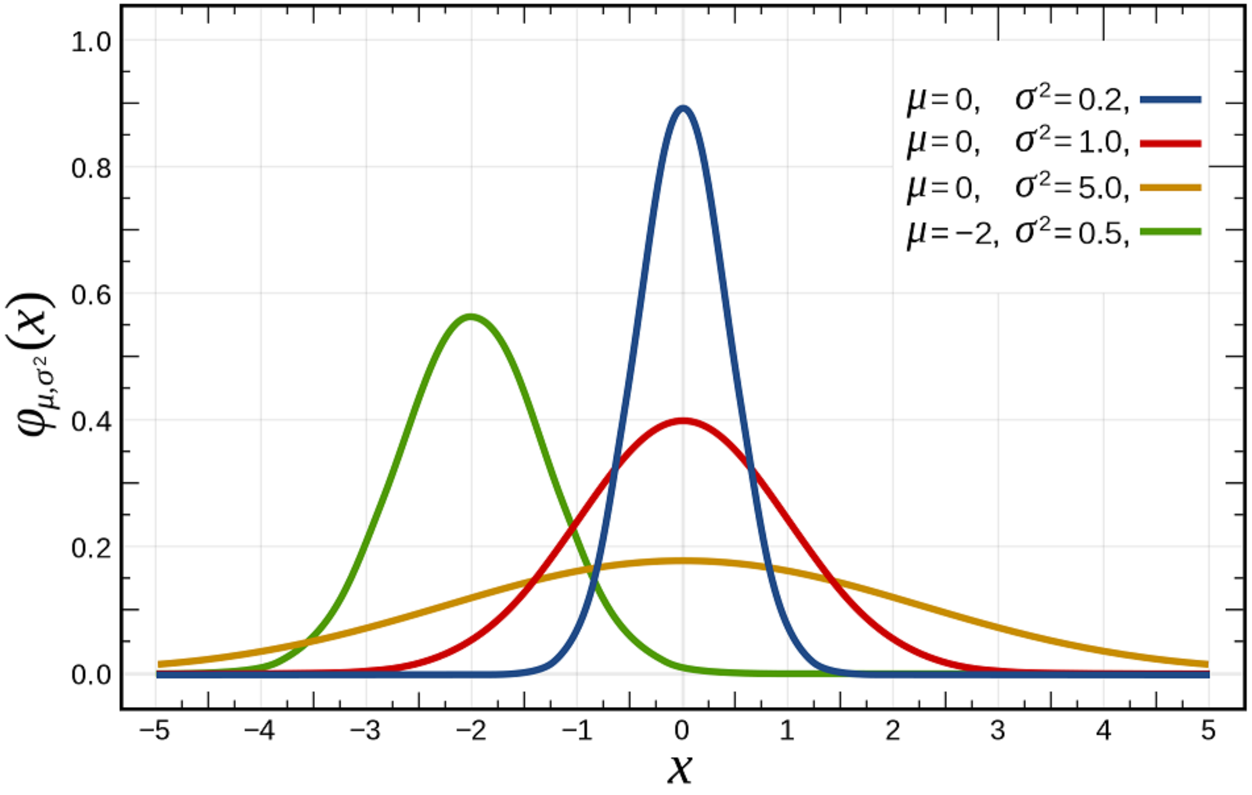
Data and Population

- Probability density function



Probability density function

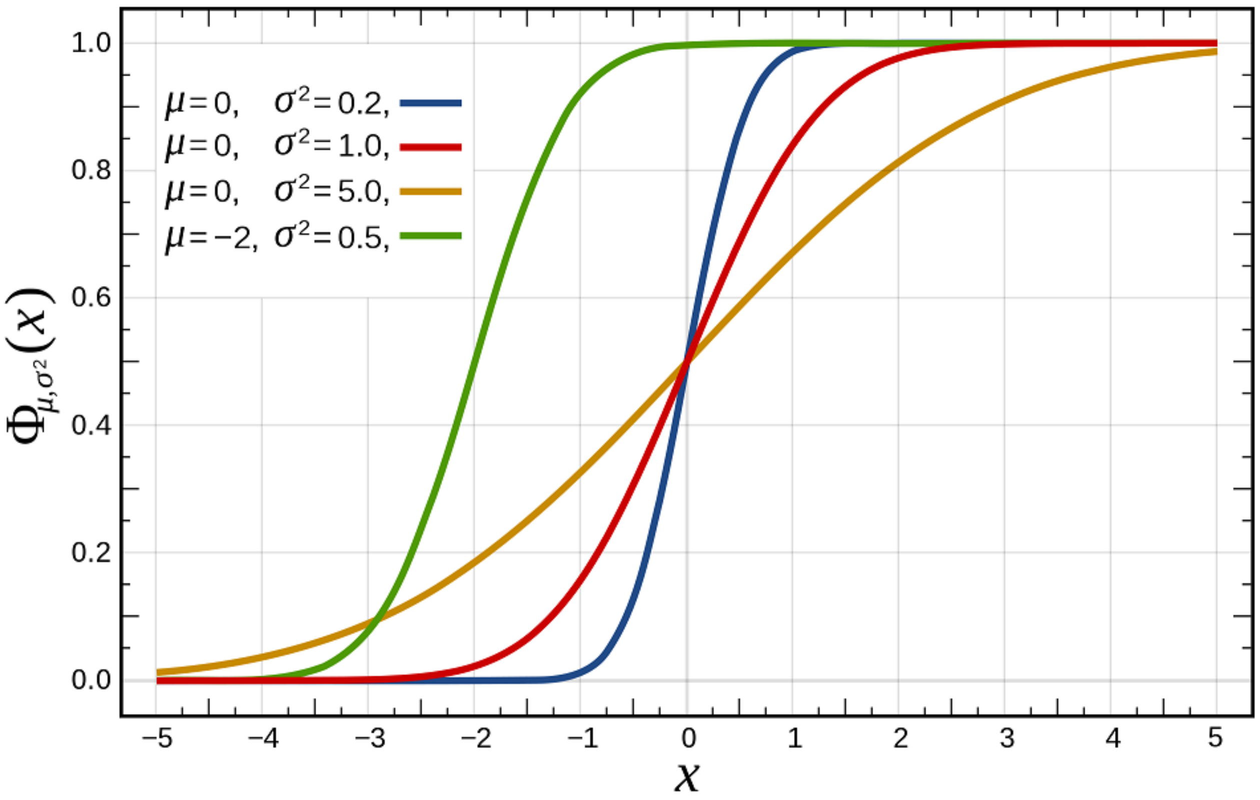
- Describes the probability that a random variable equals a certain value.



$$pdf_{N(\mu,\sigma^2)}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Cumulative distribution function

- Describes the probability that a random variable equals a certain value or less.



$$cdf_{N(\mu,\sigma^2)}(x) = \frac{1}{2}\left(1 + erf\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right)$$

$$\operatorname{erf} z = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2}\, dt.$$

Central Tendency

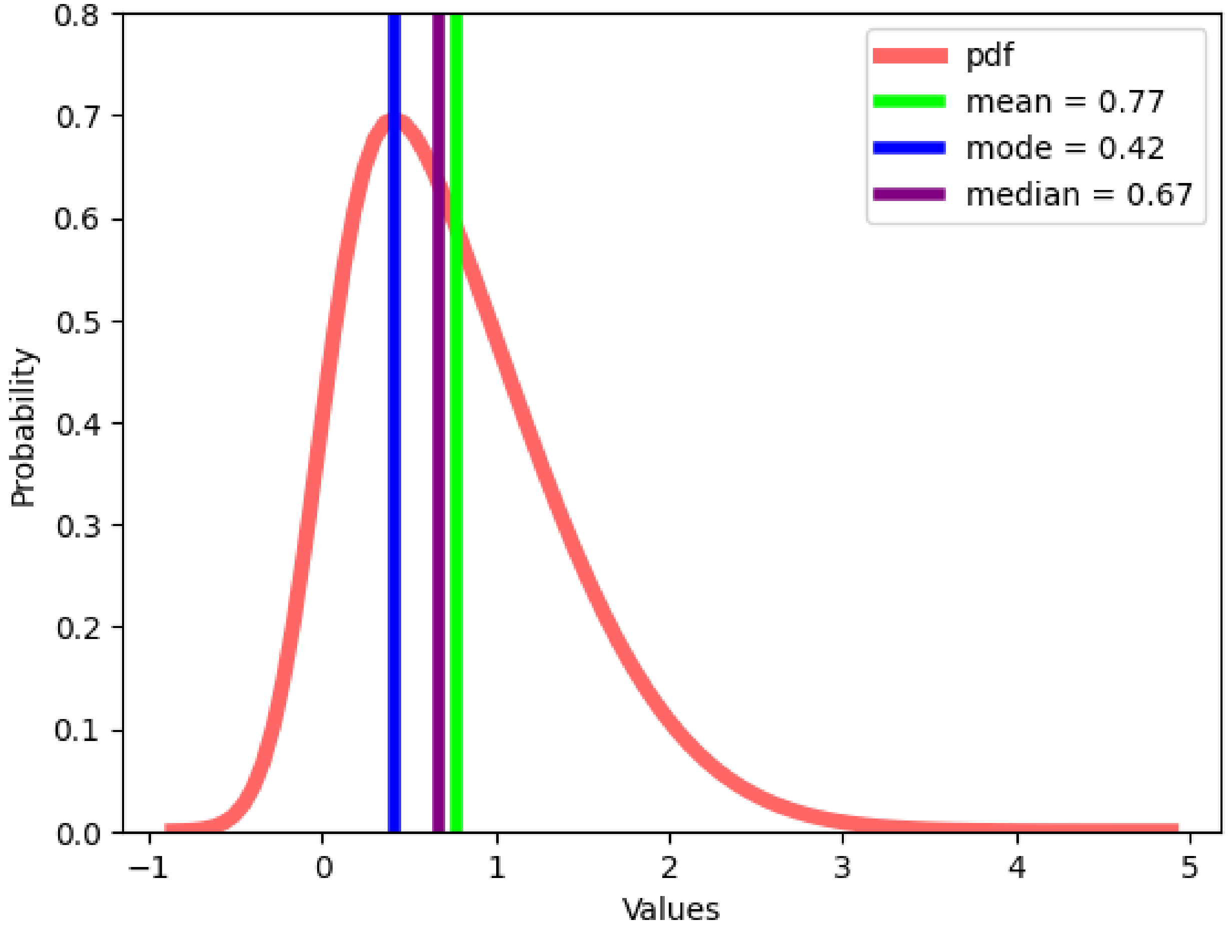
- Measure for middle of the data
  - Mean: $$\mu = \int_{-\infty}^{-\infty} xp\left(x\right)dx = E\left(x\right)$$
  - Median : $$\int_{-\infty}^{x_{med}} p\left(x\right)dx = 0.5$$
  - Mode – the most frequent value.

Central Tendency

- Measure for middle of the data
  - Mean – the sum of all values divided by the total number of values.
  - Median – the middle number in an ordered dataset.
  - Mode – the most frequent value.

Central Tendency

- Where is the middle?



Central Tendency

```
# generate population distribution
mean, var, skew, kurt = stats.skewnorm.stats(4, moments='mvsk')
x = np.linspace(stats.skewnorm.ppf(0.00001, 4), stats.skewnorm.ppf(0.999999, 4), 100)
plt.plot(x, stats.skewnorm.pdf(x, 4), 'r-', lw = 5, alpha = 0.6, label = 'pdf')
#get pdf for computing mode
t=stats.skewnorm.pdf(x, 4)

data_mode = x[np.argmax(t)] #mode is the value with the highest occurrence
r = np.array(stats.skewnorm.rvs(4, size = 10000)) #get some values
data_median = np.median(r)
```

Central Tendency

```
plt.plot([mean, mean] , [0, 0.8], color = 'lime', linewidth = 4,
           label = f'mean = {round(mean,2)}')

plt.plot([data_mode, data_mode] , [0, 0.8], color = 'blue', linewidth = 4,
label = f'mode = {round(data_mode,2)}')

plt.plot([data_median, data_median] , [0, 0.8], color = 'purple', linewidth
= 4,label = f'median = {round(data_median,2)}')

plt.xlabel('Values')
plt.ylabel('Probability')
plt.legend(loc = 'upper right')
plt.ylim([0, 0.8])
```

Central Tendency

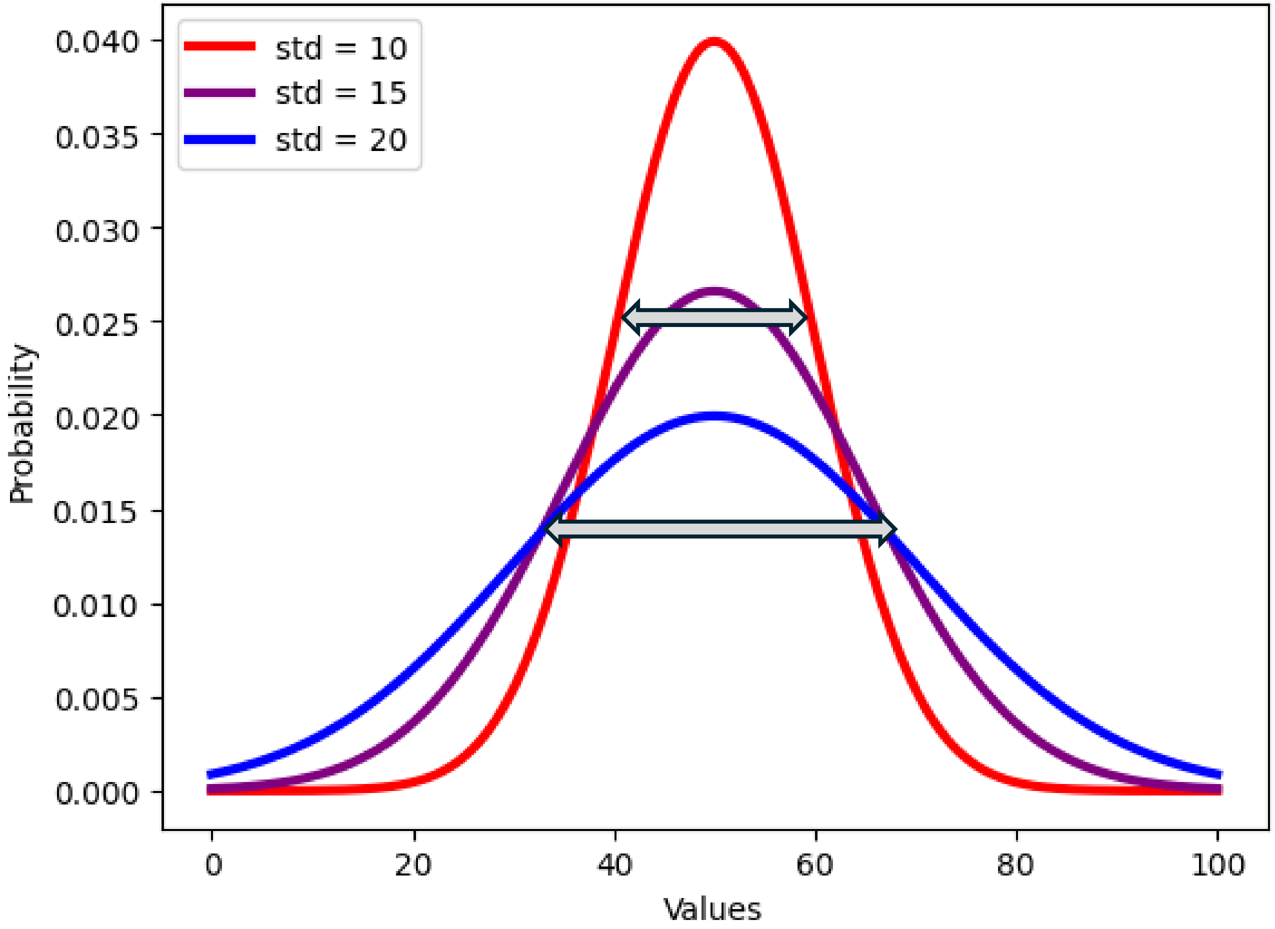
- Measure for middle of the data
  - Mean
    - Convenient to work with mathematically
    - Very effected by outliers
    - Can be used for continuous or ordinal data (for ordinal, we need to assign numbers to the categories)
  - Median
    - Less effected by outliers
    - Can be used for continuous or ordinal data
  - Mode
    - Less effected by outliers
    - The only central tendency measurement for categorical data

Spread

- How far are we from the middle?

$$Var(x) = E[(x-\mu)^2]$$

$$std(x) = \sqrt{Var(x)}$$



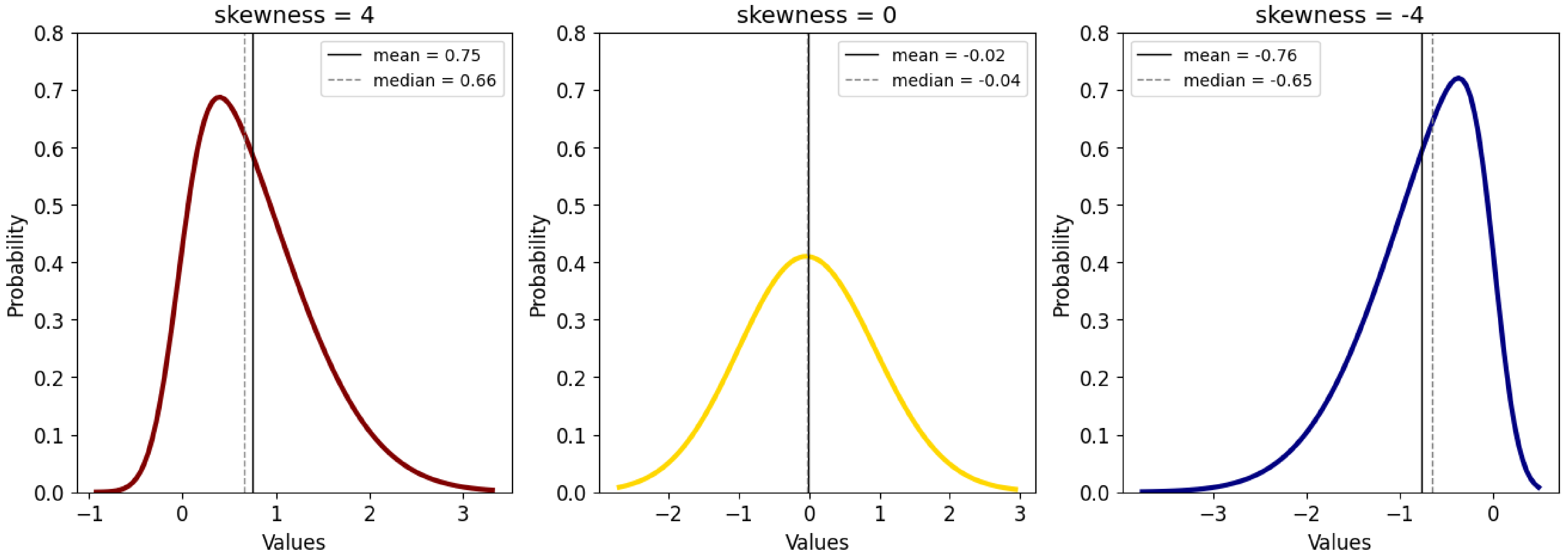
Spread

```
# examining normal distribution with different variances
x = np.linspace(0, 100, 1000)

#three different probability density functions
pdf1 = stats.norm.pdf(x, 50, 10)
pdf2 = stats.norm.pdf(x, 50, 15)
pdf3 = stats.norm.pdf(x, 50, 20)

plt.plot(x, pdf1, color = 'red', lw = 3, label = 'std = 10')
plt.plot(x, pdf2, color = 'purple', lw = 3, label = 'std = 15')
plt.plot(x, pdf3, color = 'blue', lw = 3, label = 'std = 20')

plt.xlabel('Values')
plt.ylabel('Probability')
plt.legend(loc = 'upper left')
```

Skewness

- How symmetric is the data?

$$\text{Skewness} = \frac{1}{\sigma^3} E\left(\left(x_i - \mu\right)^3\right)$$



Skewness

```
#creating distribution with different skewness values
data = stats.skewnorm(4, 0, 1).rvs(1000)
# estimate parameters from sample
ae, loce, scalee = stats.skewnorm.fit(data)
plt.subplot(1, 3, 1)
x = np.linspace(min(data), max(data), 100)
p = stats.skewnorm.pdf(x,ae, loce, scalee)
data_median = np.median(data)
data_mean = np.mean(data)
plt.plot(x, p, color = 'maroon', lw = 3)
plt.plot([data_mean, data_mean] , [0, 0.8], color = 'black', linewidth = 1,
           label = f'mean = {round(data_mean,2)}')
plt.plot([data_median, data_median] , [0, 0.8], color = 'gray', linewidth = 1,
           label = f'median = {round(data_median,2)}')
plt.legend(loc = 'upper right')
plt.title('skewness = 4')
plt.xlabel('Values')
plt.ylabel('Probability')
plt.ylim([0, 0.8])
```

Skewness

```
data = stats.skewnorm(0, 0, 1).rvs(1000)
# estimate parameters from sample
ae, loce, scalee = stats.skewnorm.fit(data)
plt.subplot(1, 3, 2)
x = np.linspace(min(data), max(data), 100)
p = stats.skewnorm.pdf(x,ae, loce, scalee)
data_median = np.median(data)
data_mean = np.mean(data)
plt.plot(x, p, color = 'gold', lw = 3)
plt.plot([data_mean, data_mean] , [0, 0.8], color = 'black', linewidth = 1,
           label = f'mean = {round(data_mean,2)}')
plt.plot([data_median, data_median] , [0, 0.8], color = 'gray', linewidth = 1,
           label = f'median = {round(data_median,2)}')
plt.legend(loc = 'upper right')
plt.title('skewness = 0')
plt.xlabel('Values')
plt.ylabel('Probability')
plt.ylim([0, 0.8])
```

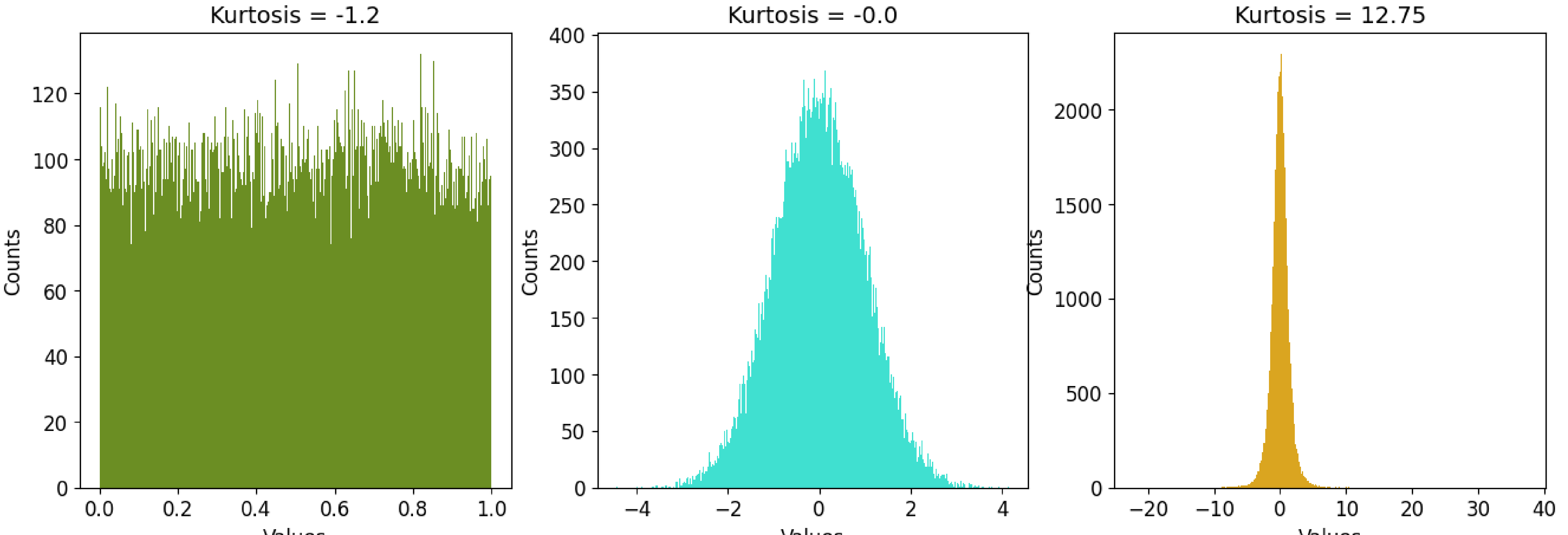
Skewness

```
data = stats.skewnorm(-4, 0, 1).rvs(1000)
# estimate parameters from sample
ae, loce, scalee = stats.skewnorm.fit(data)
plt.subplot(1, 3, 3)
x = np.linspace(min(data), max(data), 100)
p = stats.skewnorm.pdf(x,ae, loce, scalee)
data_median = np.median(data)
data_mean = np.mean(data)
plt.plot(x, p, color = 'navy', lw = 3)
plt.plot([data_mean, data_mean] , [0, 0.8], color = 'black', linewidth = 1,
           label = f'mean = {round(data_mean,2)}')
plt.plot([data_median, data_median] , [0, 0.8], color = 'gray', linewidth = 1,
           label = f'median = {round(data_median,2)}')
plt.legend(loc = 'upper left')
plt.title('skewness = -4')
plt.xlabel('Values')
plt.ylabel('Probability')
plt.ylim([0, 0.8])
```

Kurtosis

- Size of tails
- Pointiness of peak

$$\text{Kurtosis} = \frac{1}{\sigma^4} E\left(\left(x-\mu\right)^4\right) - 3$$



Kurtosis

```
#creating datasets with different kurtosis values
data = np.random.normal(0, 1, 100000)
plt.subplot(1, 3, 2)
plt.hist(data, bins = 1000, color = 'turquoise');
plt.title(f'Kurtosis = {round(stats.kurtosis(data), 2)}')
plt.xlabel('Values')
plt.ylabel('Counts')

data = np.random.uniform(0, 1, 100000)
plt.subplot(1, 3, 1)
plt.hist(data, bins = 1000, color = 'olivedrab');
plt.title(f'Kurtosis = {round(stats.kurtosis(data), 2)}')
plt.xlabel('Values')
plt.ylabel('Counts')
```

Kurtosis

```
data = np.random.standard_t(4, 100000)
plt.subplot(1, 3, 3)
plt.hist(data, bins = 1000, color =
'goldenrod');
plt.title(f'Kurtosis =
{round(stats.kurtosis(data), 2)}')
plt.xlabel('Values')
plt.ylabel('Counts')

fig = plt.gcf()
fig.set_size_inches(16, 5)
```

Recommended Homework (not for submission)

- Review google colab practice notebook in detail