EASC2410 Lecture 14

Statistics Basics: Understanding Distributions of Datasets using Histograms

Dr. Binzheng Zhang
Department of Earth Sciences

Review of Lecture 13:

- What is Statistics?
- What is the expectation from a probability?
- What is a probability distribution function?
  - Discrete distributions - Uniform, Binomial, Normal
  - Continuous distributions
- How to use Python to generate probability distributions
  - the **random** module
  - *Monte Carlo* simulations

In Lecture 14, you will learn:

- Properties of Normal distributions
  - Mode, Mean, Median, Standard Deviation
  - Continuous distributions
- How to test whether a data distribution is Normal
- How to compute and interpret the skewness of a data distribution
- How to use Seaborn to visualize distributions of a dataset

Understanding the distributions of your data sets

Histograms are graphs that display the distribution of your continuous data. They are fantastic exploratory tools because they reveal properties about your sample data in ways that summary statistics cannot. For instance, while the mean and standard deviation can numerically summarize your data, histograms bring your sample data to life.

Use histograms when you have continuous measurements and want to understand the distribution of values and look for outliers. These graphs take your continuous measurements and place them into ranges of values known as bins. Each bin has a bar that represents the count or percentage of observations that fall within that bin.

Problems with single-value descriptions

In the field of statistics, we often use summary statistics to describe an entire dataset. These statistics use a single number to quantify a characteristic of the sample. For example, a measure of central tendency is a single value that represents the center point or typical value of a dataset, such as the mean. A measure of variability is another type of summary statistic that describes how spread out the values are in your dataset. The standard deviation is a conventional measure of dispersion.

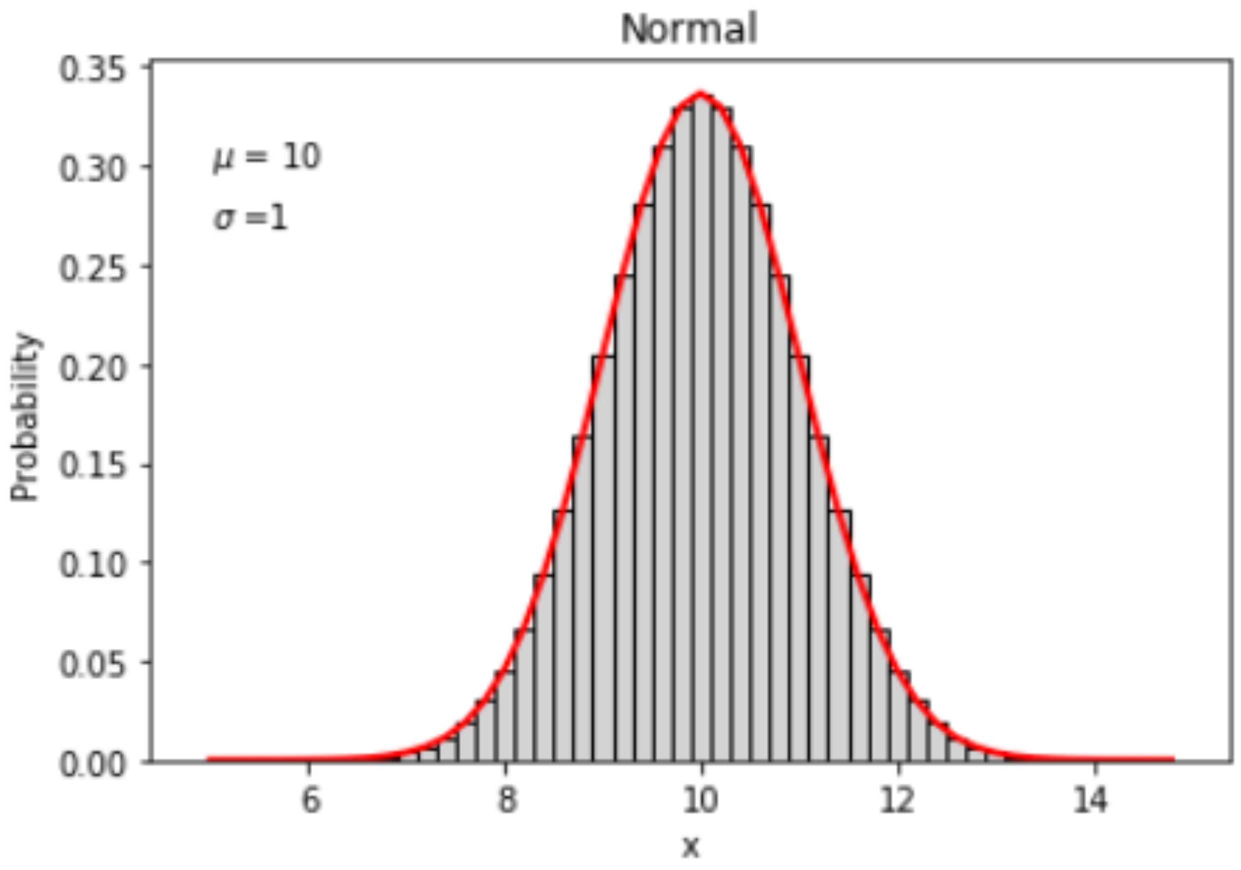
These summary statistics are crucial. How often have you heard that the mean of a group is a particular value? It provides meaningful information. However, these measures are simplifications of the dataset. Graphing the data brings it to life. Generally, I find that using graphs in conjunction with statistics provides the best of both worlds!

Recall the Normal distribution

Probably the most common distribution in real-life data is the so-called "normal" distribution (also known as a Gaussian distribution after the guy who thought it up).

For normal distributions, the **average** is the *arithmetic mean* μ and the **spread** is the *standard deviation* σ :

$$P(x) = f(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, -\infty < x < +\infty$$



A.K.A: a bell-curve
(because it is exactly what it looks like!)

Key parameters in a Normal distribution:

μ : Mean or Expectation

σ : Standard Deviation or Spread

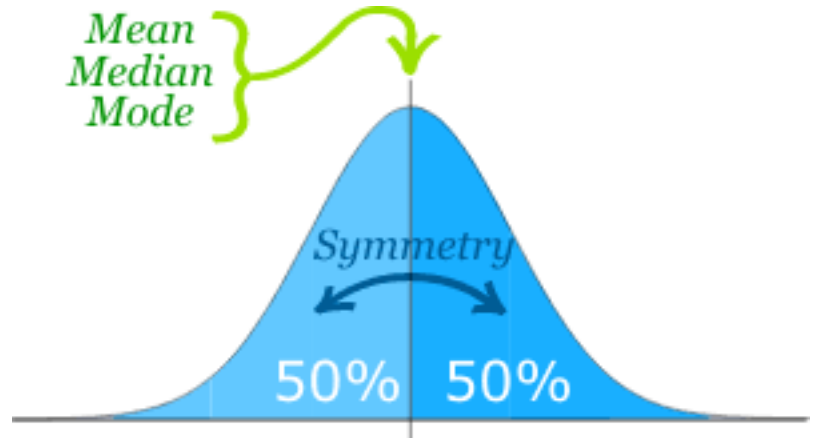
In a normal distribution, **median is the same as mean**. The deviation of median from mean is a measure of the **skewness** of the distribution

In a data set assumed to be Normal distributions:

$$\mu = \frac{1}{N}\Sigma_{i=1}^{N} x[i]$$
$$\sigma = \frac{1}{N}\Sigma_{i=1}^{N} (x[i] - \mu)^2$$

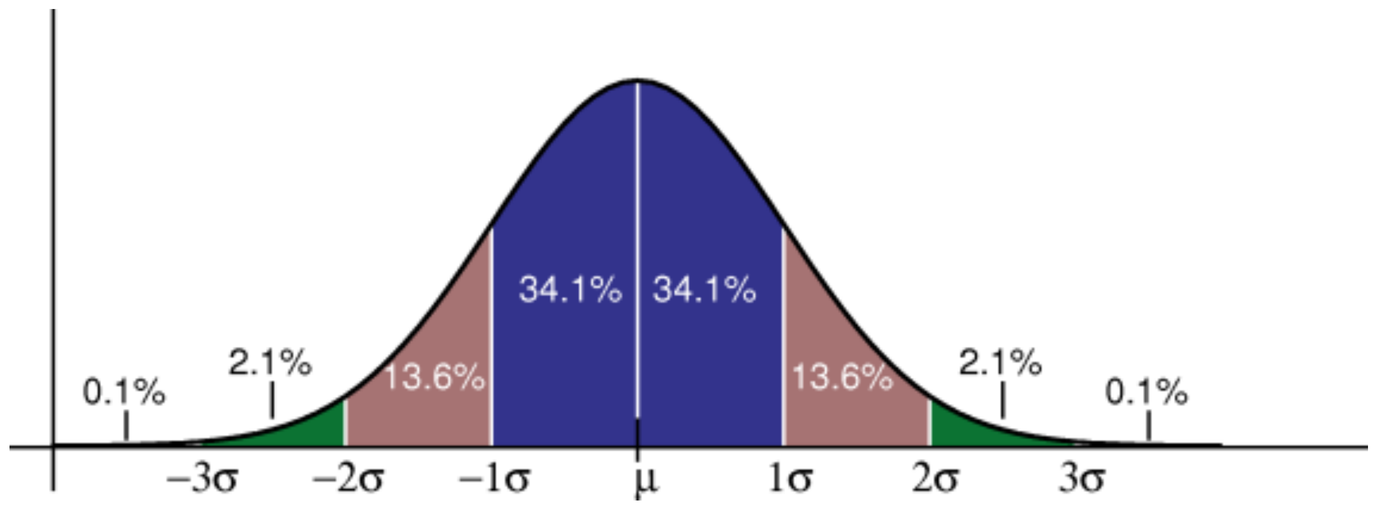
More properties of the Normal distribution

Symmetry:



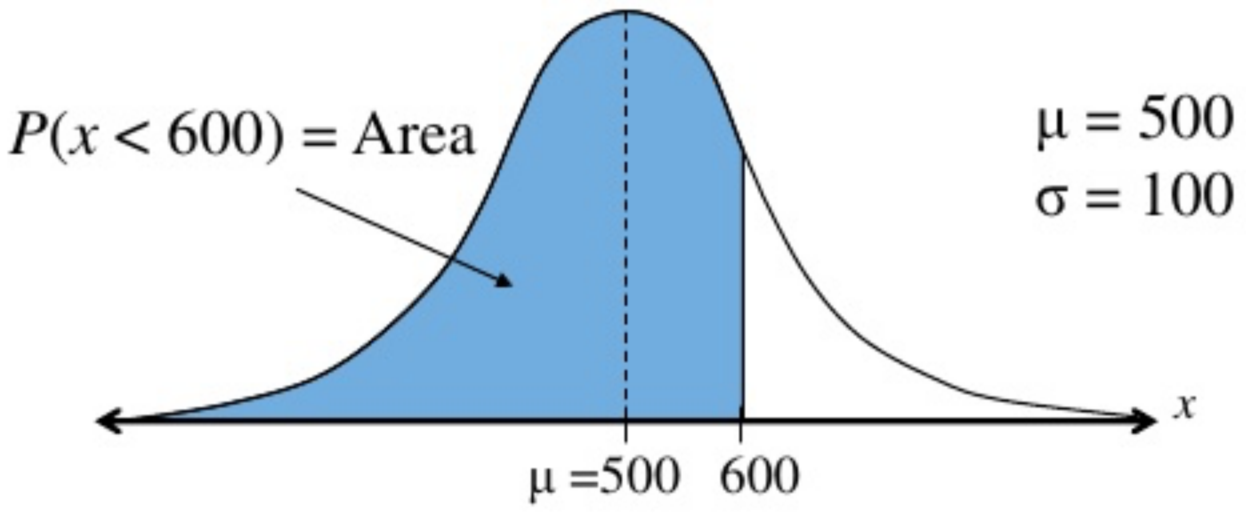
A Normal distribution is symmetric about Mean

Spread:



Any value that is outside 2sigma, it occurrence rate is ~ 4%

Probability Estimation

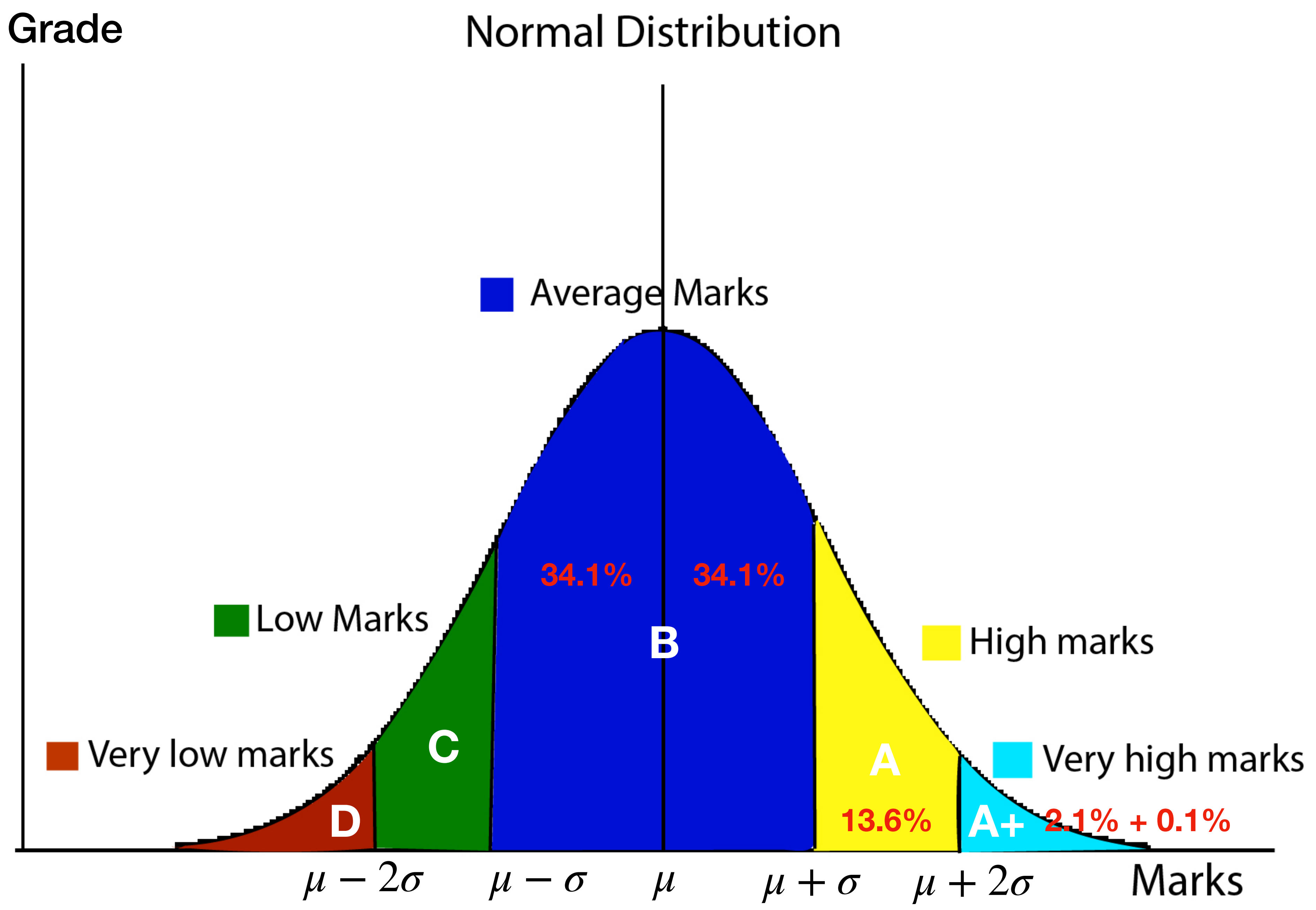


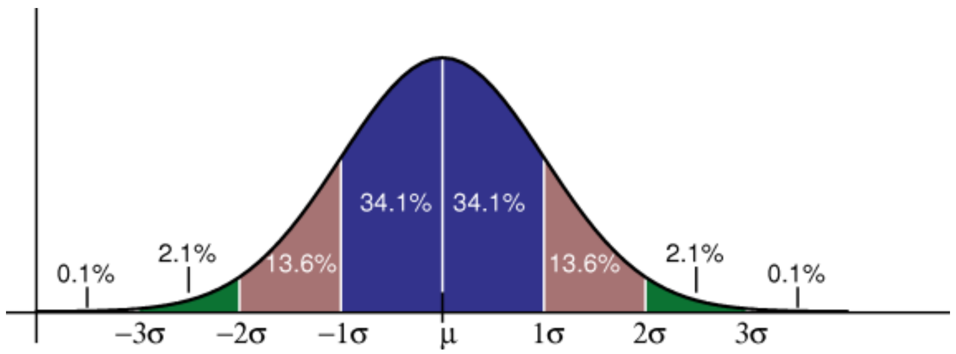
What does **P(x<600)** = **area** mean?

So if we know an event has a **normal** distribution, we can estimate the probability the occurrence of variables within a certain range by calculating the area of the assumed normal distribution function, this is basically a decision making process based on data science.

Let's take a look at some examples

Example of a Normal distribution

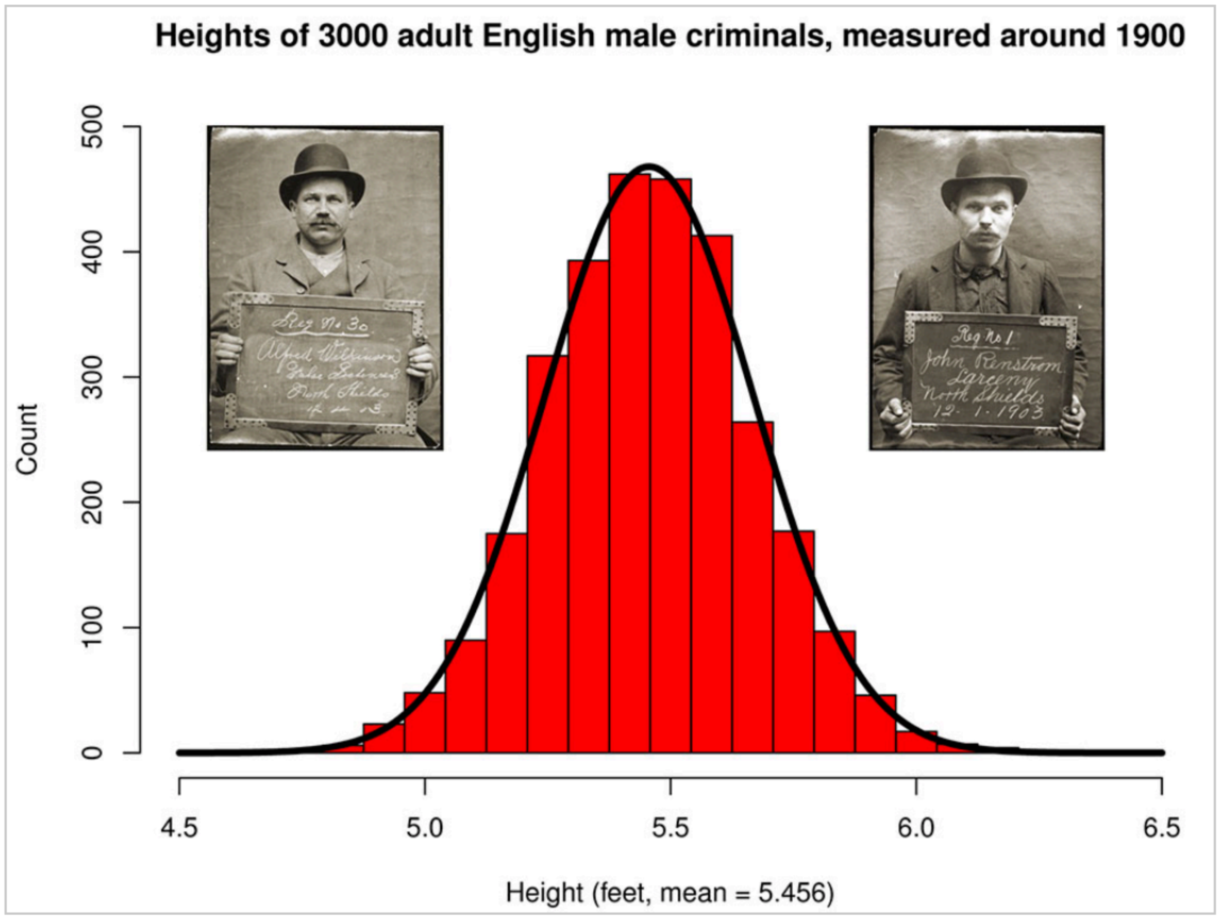




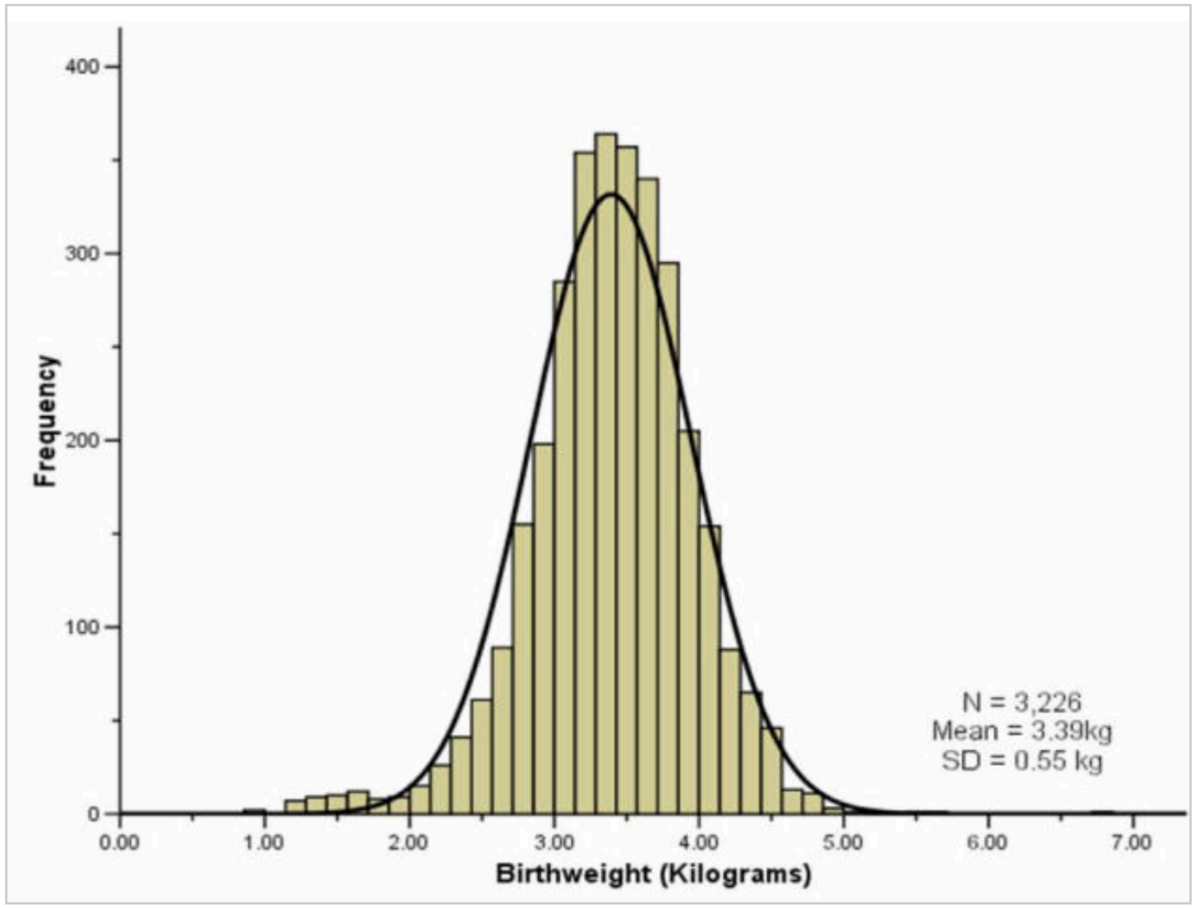
Question: Can you estimate how many students get As and Bs in total?

P(A and B) = 34.1% + 34.1% + 13.6% + 2.1% +0.1% = 84%

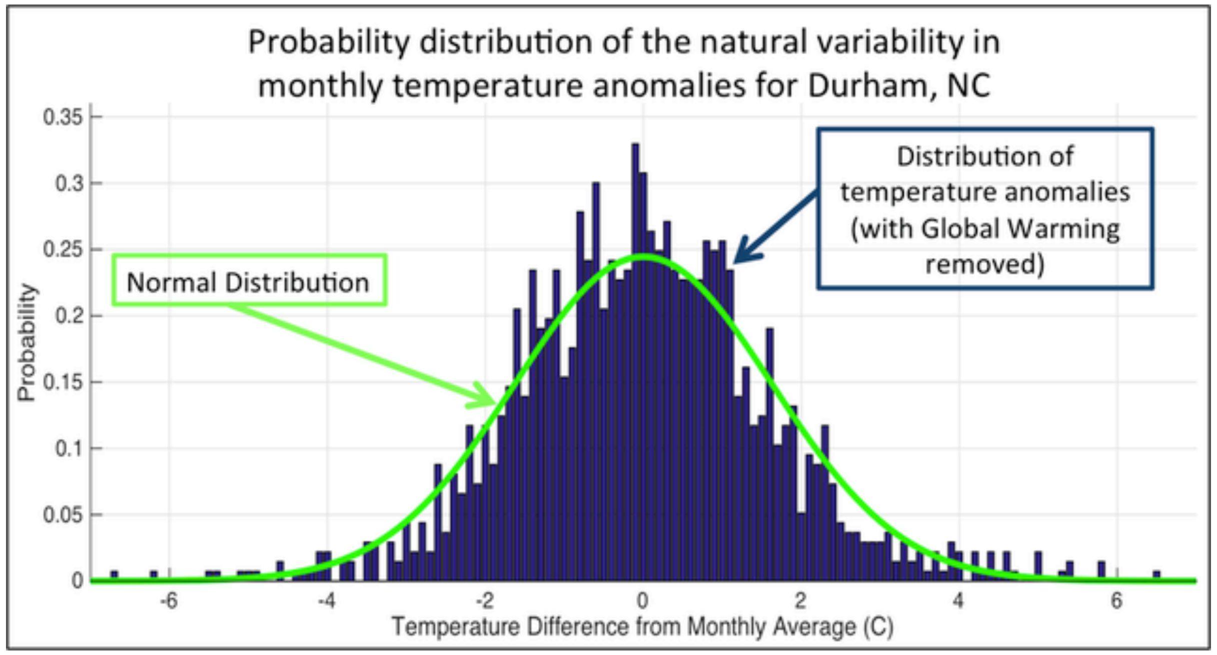
Height of English Criminals



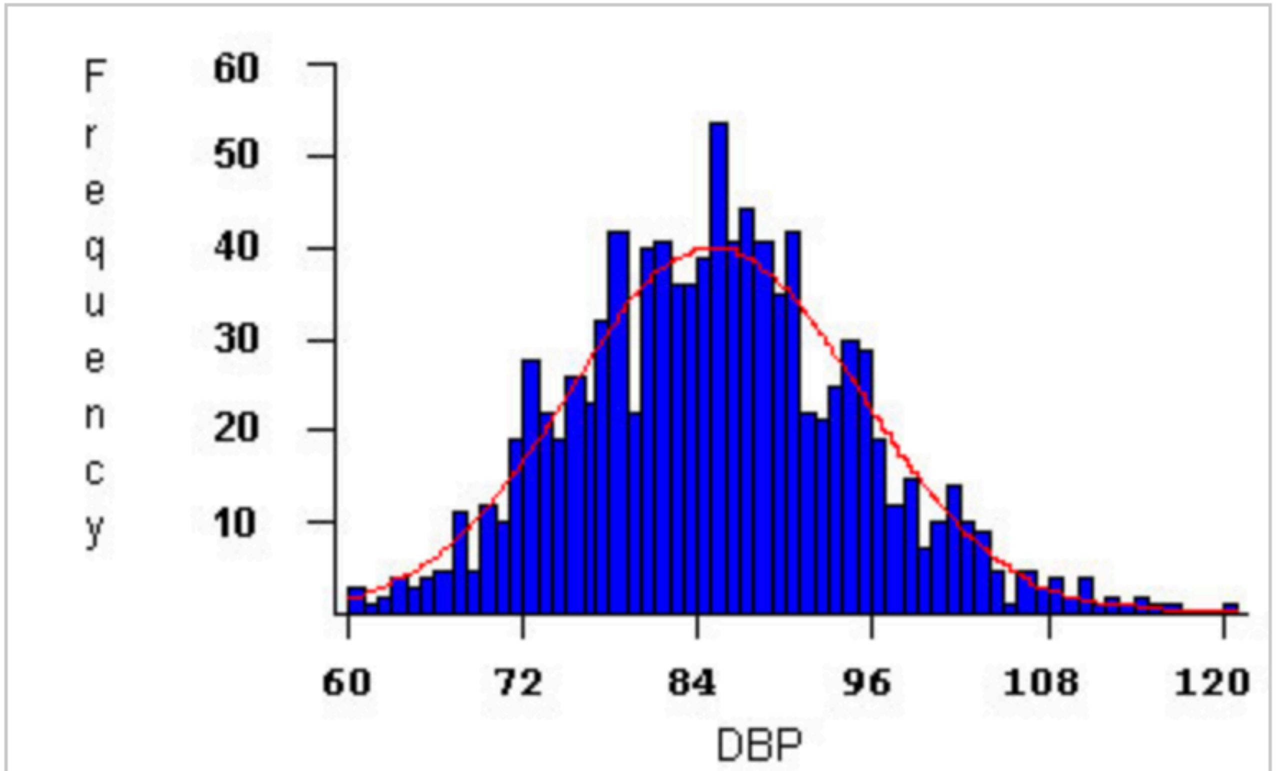
Weight of newborns



Anomalies of temperature



Human blood pressure



Tsinghua University (清華) admits approximately 128 undergraduate students in the Guangdong Province every year. Given that the total high-school graduates in Guangdong is about 750,000 per year, what's the odds of getting admitted by Tsinghua University (THU) in Guangdong?

清華大學

Tsinghua University

odds = 128/750,000 = 0.00017 (one out of 5880)

Ewww, it looks extremely tough to get in! **But**, the chance is **different** for an individual student. If we know the scores of mock examinations of a particular student, how do we estimate his/her chance of getting into THU in the coming year?

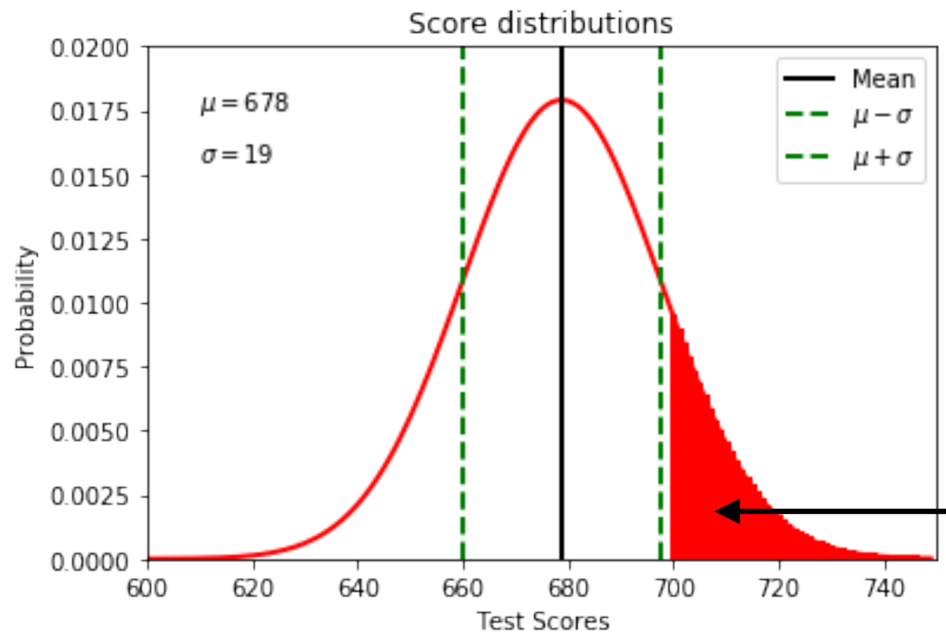
First, let's collect the data. Here's Jane's mock test scores in the past year:

[657, 677, 690, 634, 688, 703, 692, 665, 681, 691, 689] **out of 750**

Then, let's assume her test score follows a **Normal distribution**

Next, estimate mean and standard deviation: $\mu = \frac{1}{N}\Sigma_{i=1}^{N} x[i] \approx 678$ $\sigma = \frac{1}{N}\Sigma_{i=1}^{N}(x[i] - \mu)^2 \approx 19$

Now construct a normal distribution:



OK, then how do we estimate the odds based on Jane's mock score distribution?

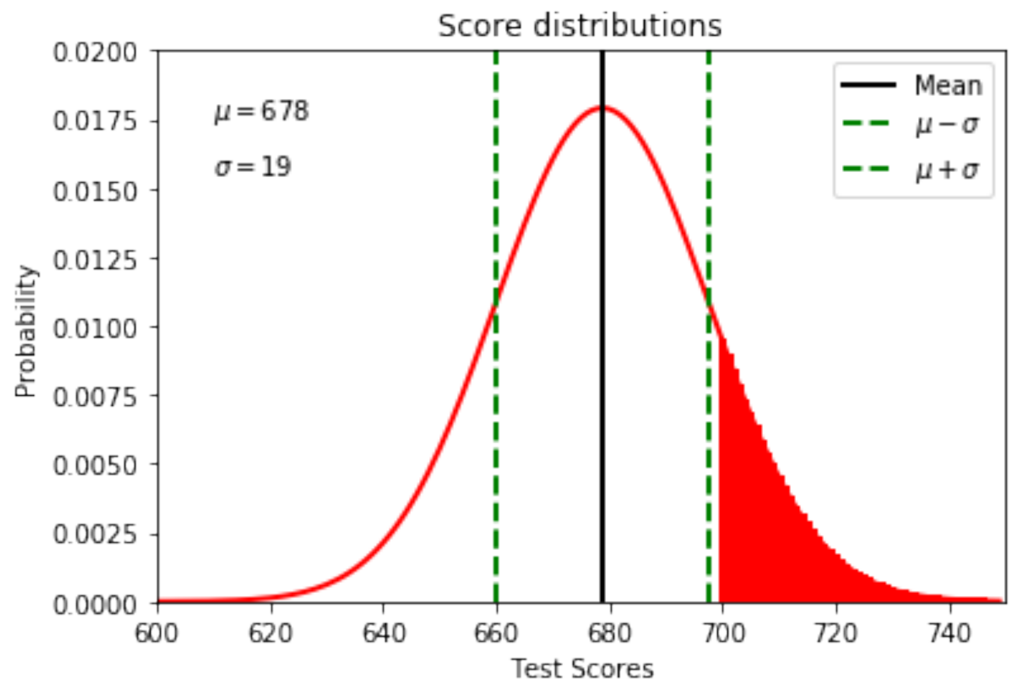
According to last year's statistics, you need **at least 700** to get admitted by Tsinghua University (really competitive!) So if we use 700 as the mock admission line, the odds of getting into Tsinghua University is P(score>700), which is denoted by the red area

$$P(score > 700) \approx 16\,\% \gg 0.017\,\%$$

More on Normal distributions

Two more things about Normal distributions of a data set:

1) the standard deviation includes ~67% of the data (not 95%), that would be 1.97σ (or 2-sigma, informally). The $\pm\sigma$ bounds are the dashed lines in a plot (histogram).



2) the mean of our sample is generally not the same as the mean of the distribution ($x \neq \mu$). In fact, the 95% confidence bounds for the MEAN is related to the 'standard error s_e', which is:

$$s_e = \frac{\sigma}{\sqrt{N}}$$

The 95% confidence bounds for the mean is given by 1.97s_e. This means in practice that the mean will be more than 1.97s_e away from true mean 5% of the time. We could test that statement with another little Monte Carlo type simulation, but I leave that your curiosity.

Important Measures of a data set

Central tendency

In statistics, a **central tendency** (or **measure of central tendency**) is a central or typical value for a probability distribution. It may also be called a **center** or **location** of the distribution. Colloquially, measures of central tendency are often called *averages.* The term *central tendency* dates from the late 1920s. - "*Oxford Dictionary of Statistics*"

How to measure the Central tendency of a data set

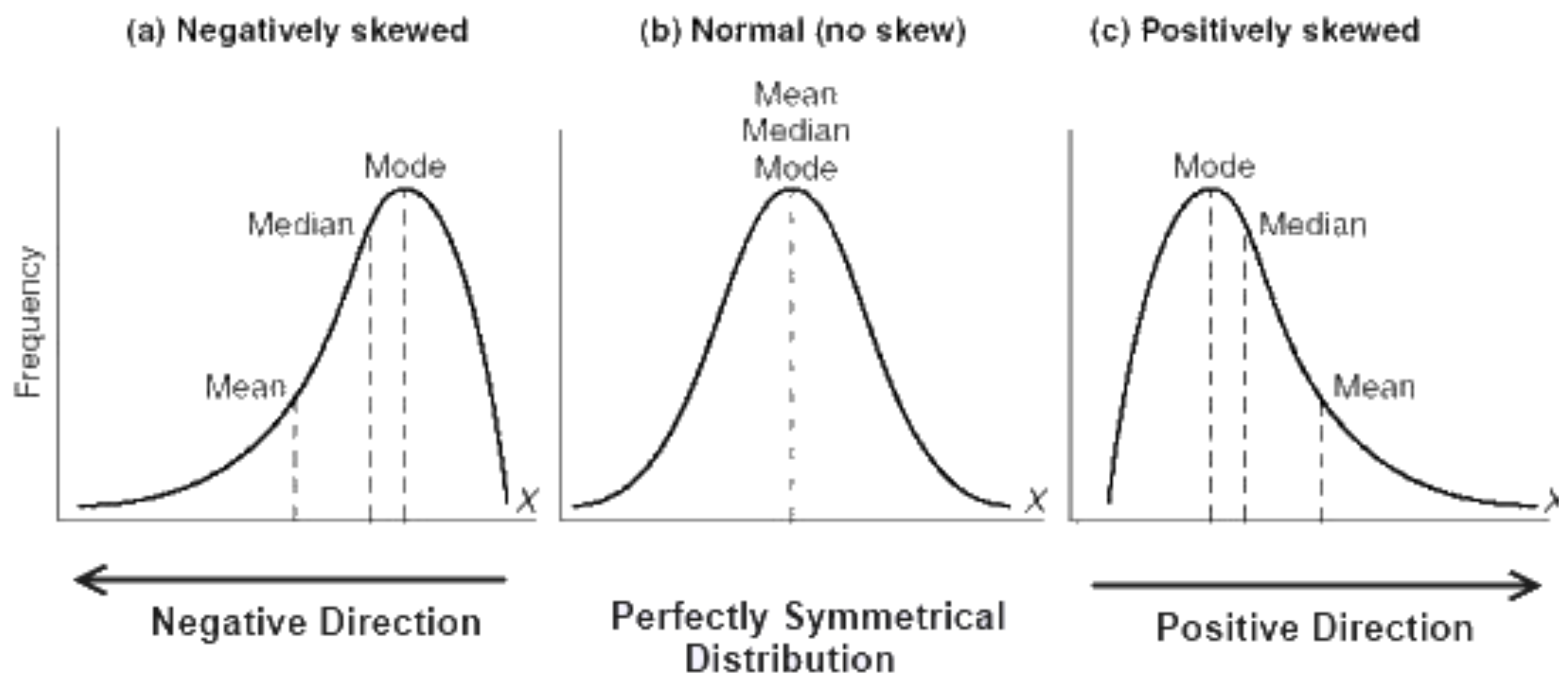
The most common measures of central tendency are the arithmetic mean, the median and the mode.

Arithmetic mean: the sum of all measurements divided by the number of observations in the data set.

Median: the middle value that separates the higher half from the lower half of the data set

Mode: the most frequent value in the data set.

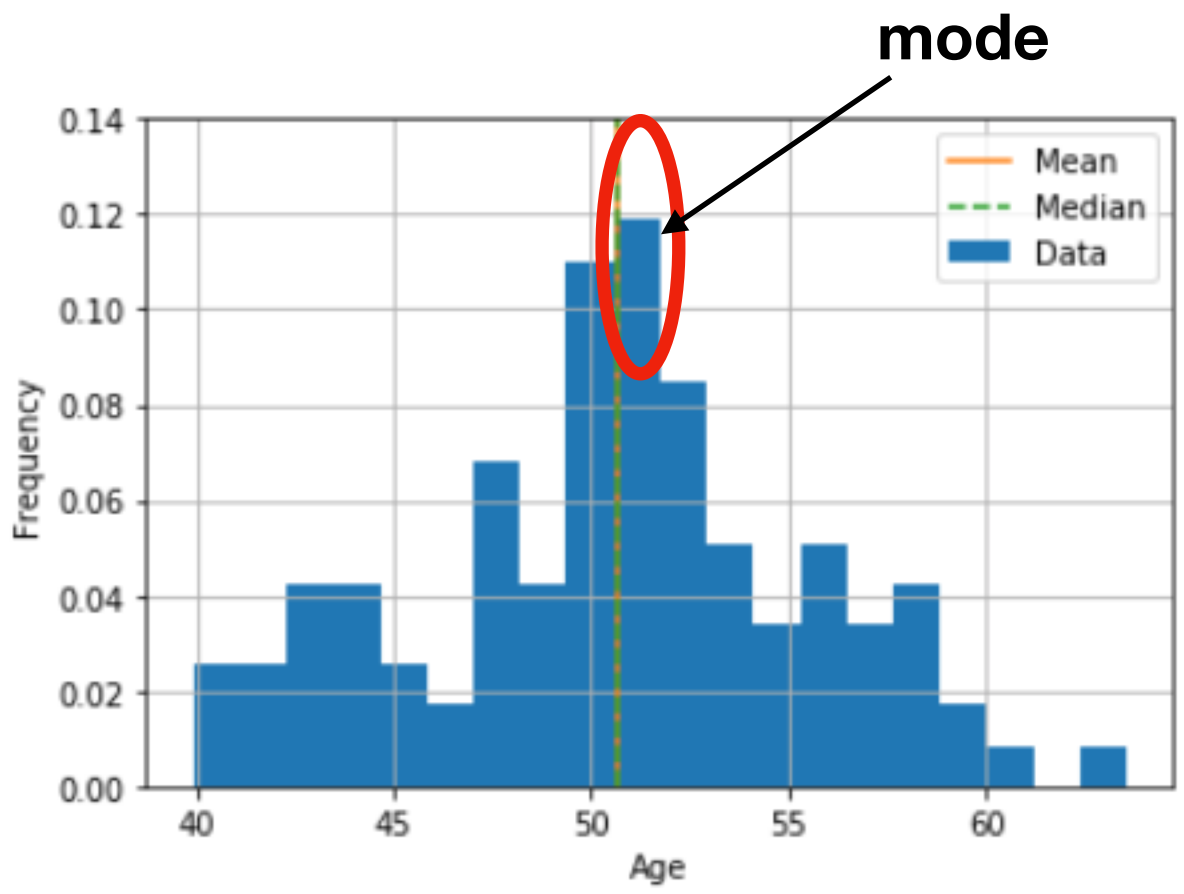
Skewness:



Understanding the Central Tendency of your data sets using Histograms

```python
df = pd.read_csv('datasets/histgr.csv') # load the data set

print(df.columns) # print out the column names - understanding your data set

df['A'].hist(bins=20,density=True,label='Data') # make a histogram with 20 bins

print('The mean of the dataset A is', '%3.1f'%(df.A.mean())) # print out mean
print('The median of the dataset A is', '%3.1f'%(df.A.median())) # print out median

plt.plot([df.A.mean(),df.A.mean()],[0,0.14],label='Mean')
plt.plot([df.A.median(),df.A.median()],[0,0.14],'--',label='Median')

plt.ylim([0,0.14])
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.legend()
```

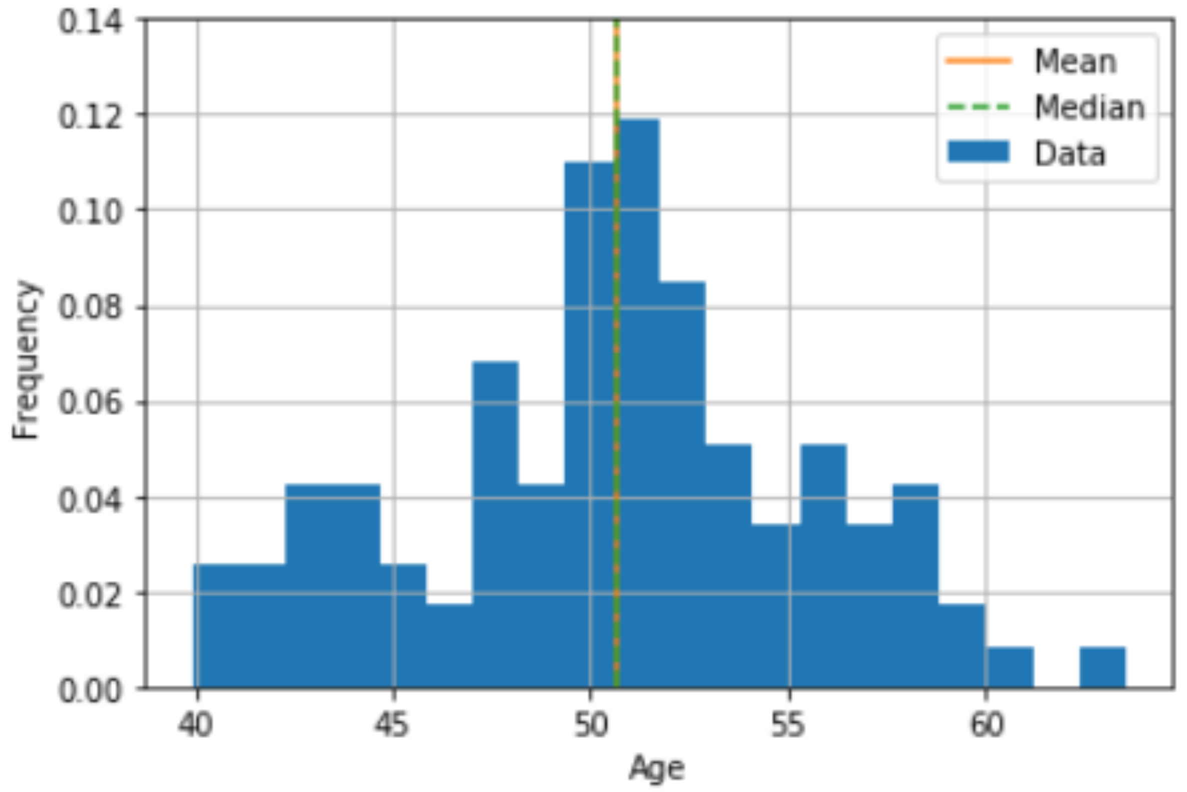
```
Index(['A', 'B', 'C', 'D', 'Left Skew', 'Multimodal', 'IQ20', 'IQ100'], dtype='object')
The mean of the dataset A is 50.6
The median of the dataset A is 50.7
```



Observations:

- does the data look like a Normal distribution?
- Mean = Median?
- What's the central tendency of the data?
- Which one is the mode?

Test whether your data set is a Normal distribution



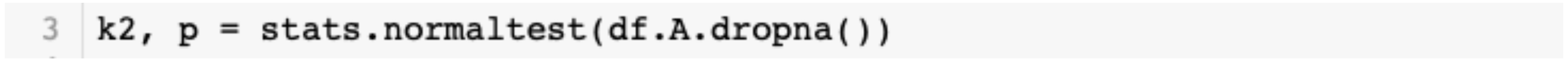
- does the data look like a Normal distribution?

It is sometimes very important to test whether a distribution of your data set is normal (or not). There're very rigorous mathematical definitions of how to compute such a test, but with Python you don't need to worry about those tedious math steps. Instead, we use the **normaltest**() function from the **stats** module of **scipy**:

```
from scipy import stats

k2, p = stats.normaltest(df.A.dropna())

print('For distributon A, the p value is ', p)

if p > 0.05:
    print('Sample looks Gaussian')
else:
    print('Sample does not look Gaussian')
```

```
For distributon A, the p value is  0.8328983767345228
Sample looks Gaussian
```

Interpret the results of a Normal test



```
k2, p = stats.normaltest(df.A.dropna())
```

Each test will return at least two things:

- Statistic (k2): A quantity calculated by the test that can be interpreted in the context of the test via comparing it to critical values from the distribution of the test statistic.
- p-value (p): Used to interpret the test, in this case whether the sample was drawn from a Gaussian distribution.

Each test calculates a *test-specific statistic*. This statistic can aid in the interpretation of the result, although it may require a deeper proficiency with statistics and a deeper knowledge of the specific statistical test. **Instead, the p-value can be used to quickly and accurately interpret the statistic in practical applications.**

The tests assume that that the sample was drawn from a Gaussian distribution. In the SciPy implementation of these tests, you can interpret the p value as follows.

$p <= 0.05$: reject, distribution is not normal.

$p > 0.05$: fail to reject, distribution is normal.

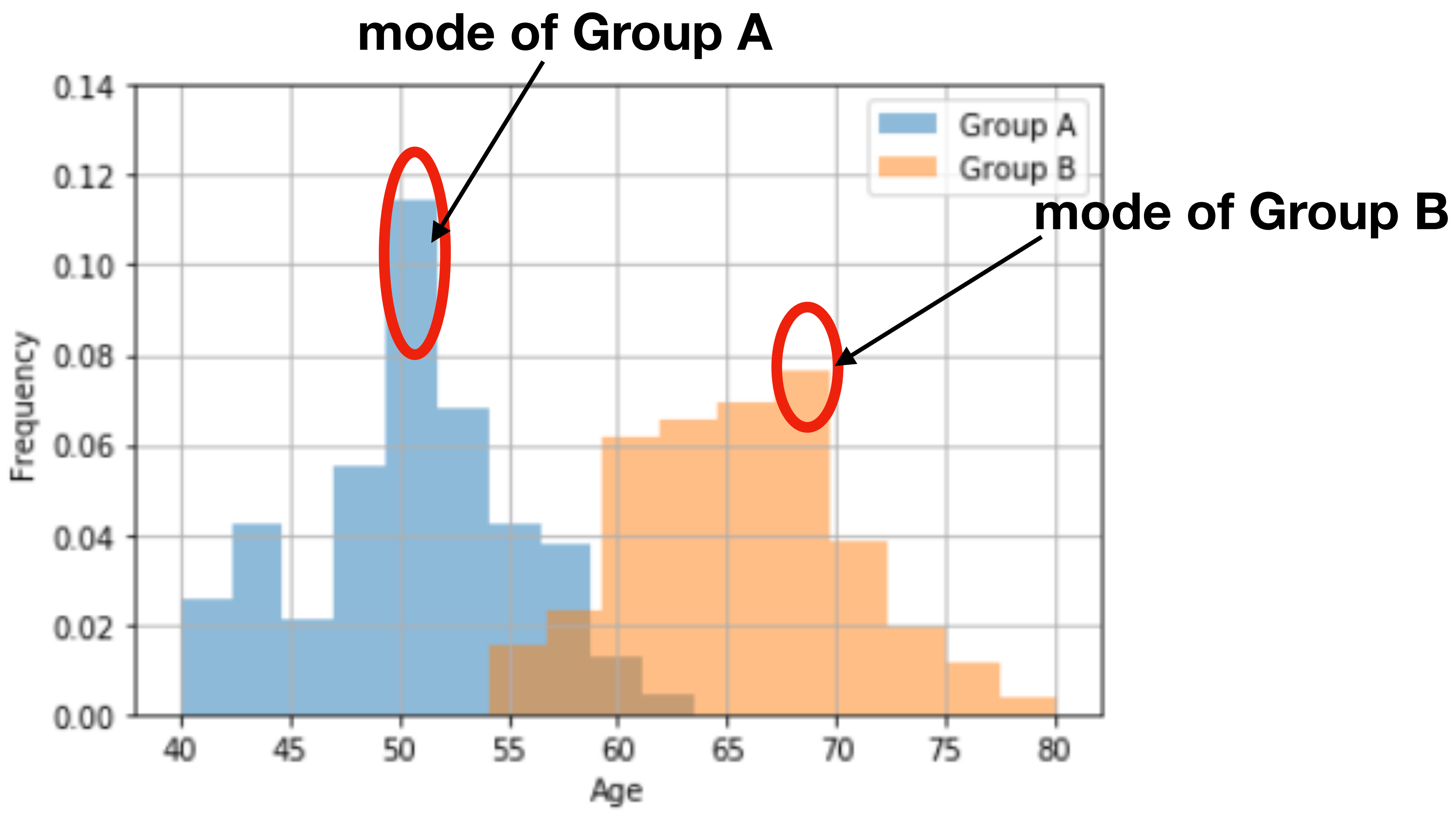
This means that, in general, we are seeking results with a **larger p-value to confirm** that our sample was likely drawn from a Gaussian distribution.

A result above 5% *does not* always mean that the normality (null) hypothesis is true. It means that it is very likely true given available evidence. The p-value is not the probability of the data fitting a Gaussian distribution; it can be thought of as a value that helps us interpret the statistical test.

Understanding the Central Tendency of your data sets using Histograms

A difference in means shifts the distributions horizontally along the X-axis (unless the histogram is rotated). In the histograms below, one group has a mean of 50 while the other has a mean of 65.

```
df = pd.read_csv('datasets/histgr.csv') # load the data set

df['A'].hist(bins=10,density=True, alpha = 0.5,label='Group A')
df['B'].hist(bins=10,density=True, alpha = 0.5,label='Group B')

plt.ylim([0,0.14])
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.legend()
```



The .describe( ) function

Pandas provides a very handy function, .describe( ), which gives you a quick statistical summary of your data frame. A full documentation of the .describe( ) function is here:

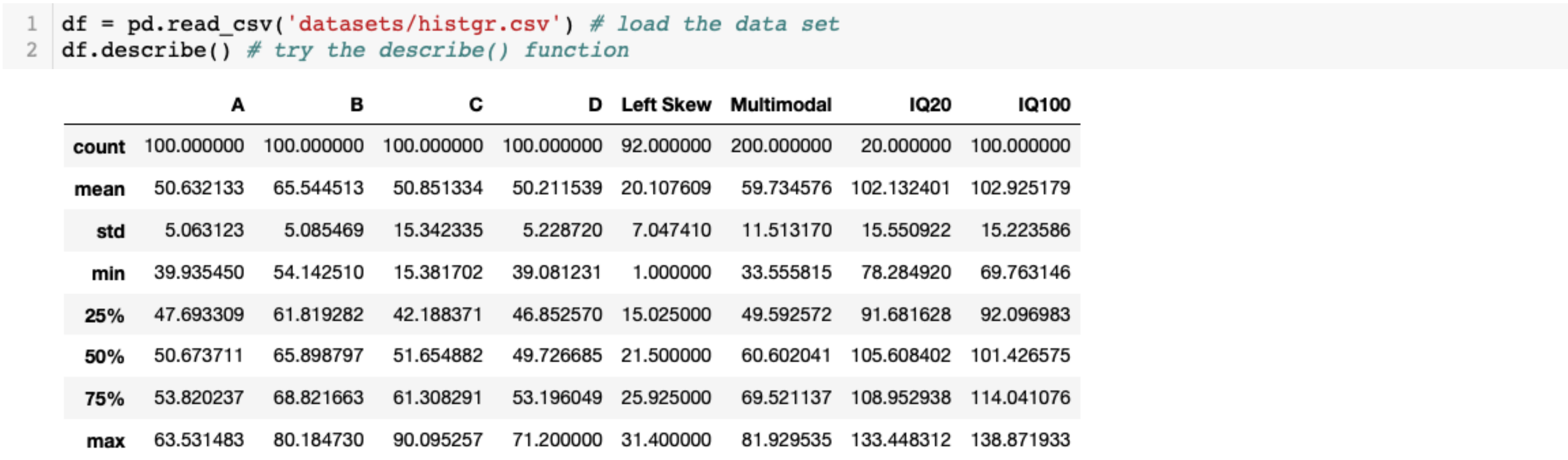
https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.describe.html

Basic syntax

`DataFrame.`**`describe`**(*percentiles=None*, *include=None*, *exclude=None)*

The describe( ) function analyzes both numeric and object series, as well as `DataFrame` column sets of mixed data types. The output will vary depending on what is provided. Refer to the Pandas documentation for more detail.

Quick Example:

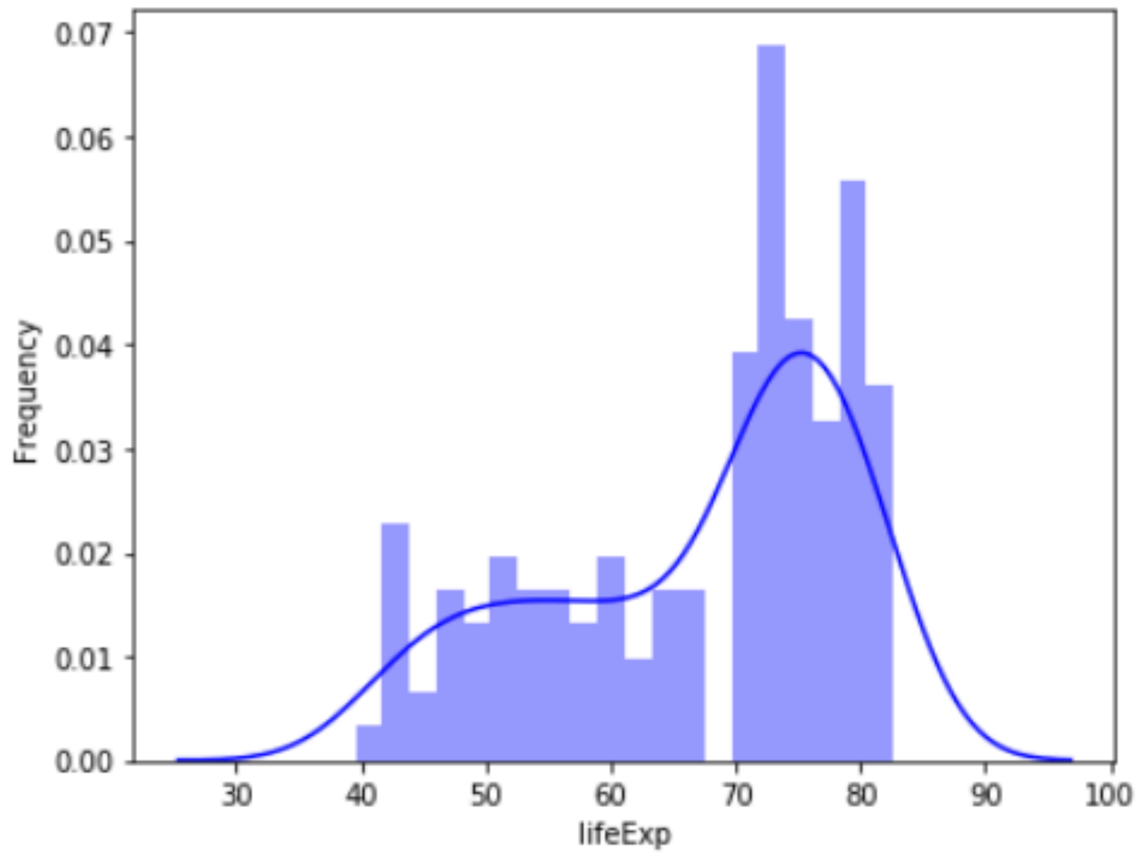


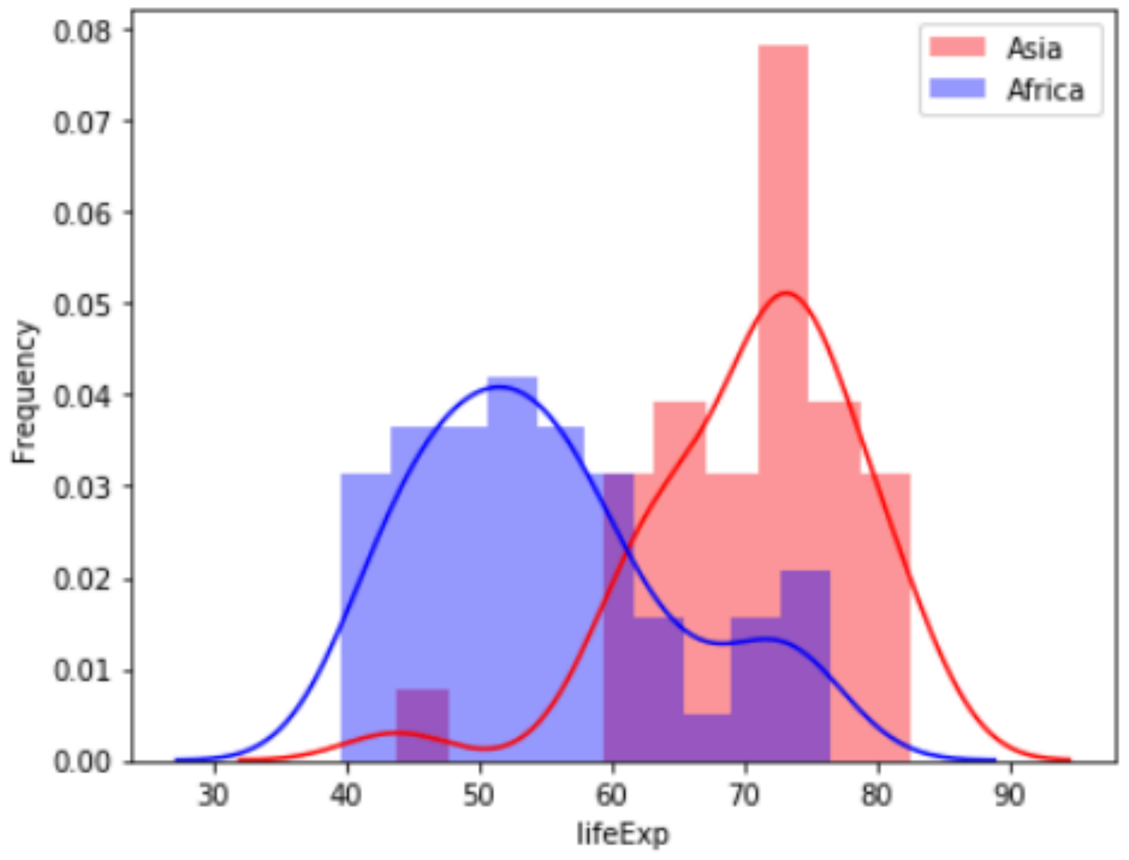
```python
df = pd.read_csv('datasets/histgr.csv') # load the data set
df.describe() # try the describe() function
```

	A	B	C	D	Left Skew	Multimodal	IQ20	IQ100
count	100.000000	100.000000	100.000000	100.000000	92.000000	200.000000	20.000000	100.000000
mean	50.632133	65.544513	50.851334	50.211539	20.107609	59.734576	102.132401	102.925179
std	5.063123	5.085469	15.342335	5.228720	7.047410	11.513170	15.550922	15.223586
min	39.935450	54.142510	15.381702	39.081231	1.000000	33.555815	78.284920	69.763146
25%	47.693309	61.819282	42.188371	46.852570	15.025000	49.592572	91.681628	92.096983
50%	50.673711	65.898797	51.654882	49.726685	21.500000	60.602041	105.608402	101.426575
75%	53.820237	68.821663	61.308291	53.196049	25.925000	69.521137	108.952938	114.041076
max	63.531483	80.184730	90.095257	71.200000	31.400000	81.929535	133.448312	138.871933

Multimodal Distributions

A **multimodal** distribution has more than one peak. It's easy to miss multimodal distributions when you focus on summary statistics, such as the mean and standard deviations. Again, histograms are the best method for detecting multimodal distributions.





Sometimes these multimodal distributions reflect the actual distribution of the phenomenon that you're studying. In other words, there are genuinely different peak values in the distribution of one population. However, in other cases, multimodal distributions indicate that you're combining subpopulations that have different characteristics. Histograms can help confirm the presence of these subpopulations and illustrate how they're different from each other.

The above figures were generated using the seaborn module

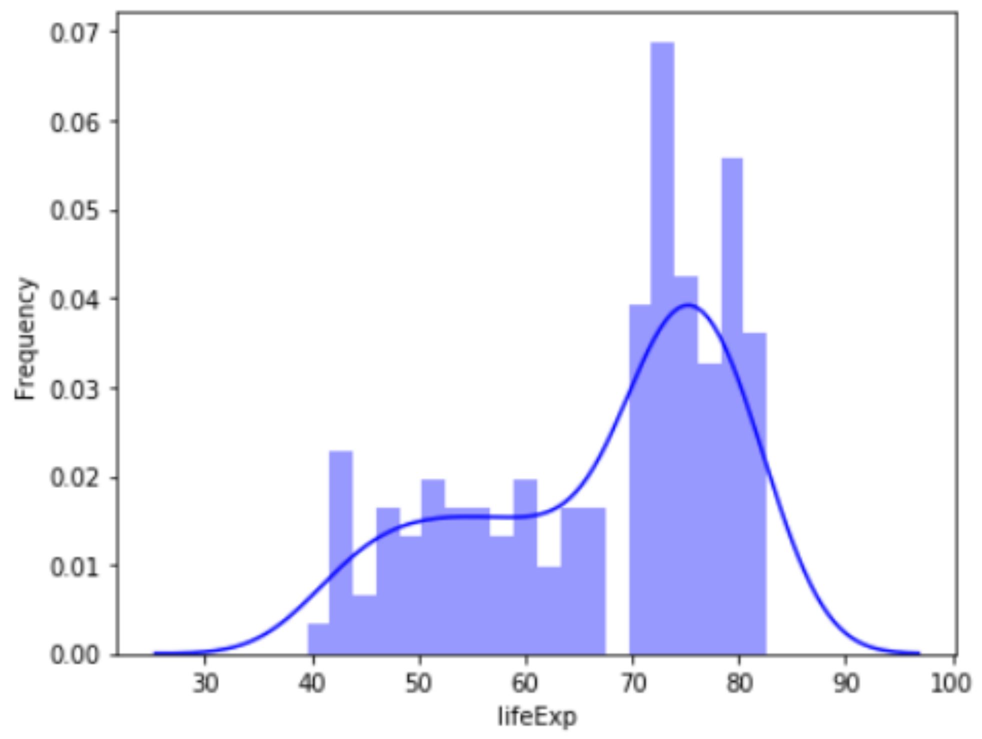
The seaborn module

Import the **seaborn** module, rename it as **sns**

```python
import numpy as np
import matplotlib.pyplot as plt
from IPython.display import Image
import pandas as pd
import seaborn as sns
```

Use **sns.distplot( )** function to generate histograms with, kde = True gives the Kernel Density Estimation

```python
file = "datasets/gdp_data.txt"
gdp = pd.read_csv(file)

plt.figure(figsize=(14,5))
plt.subplot(1,2,1)
gdpPC = gdp.lifeExp[(gdp.year==2007)]
sns.distplot(gdpPC,kde=True, color='blue', bins=20)
plt.ylabel('Frequency')
```



Syntax of the **seaborn.distplot( ) function:**

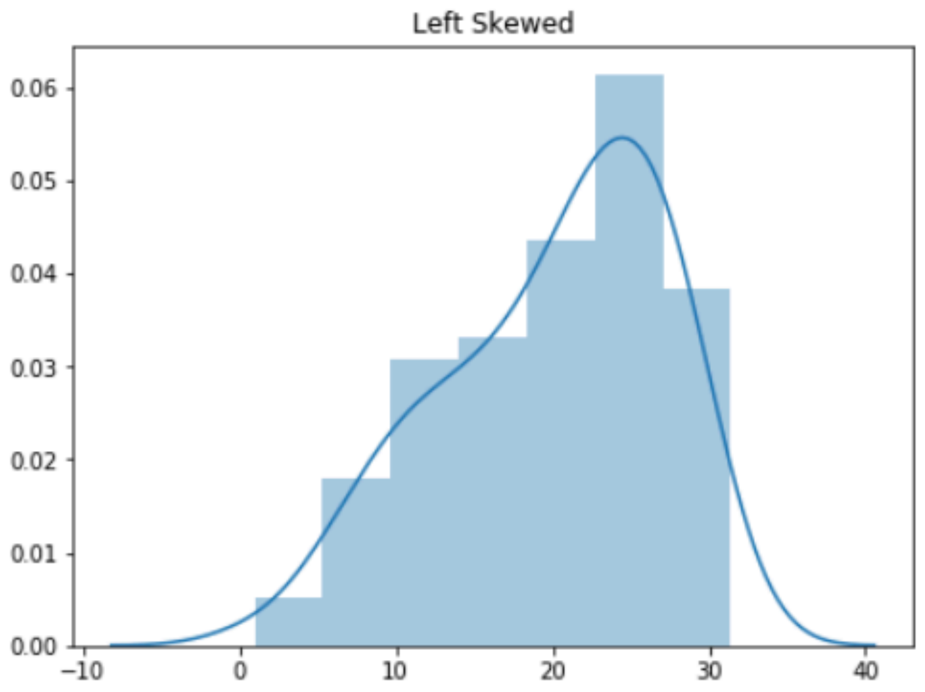
`seaborn.distplot`(a, bins=None, hist=True, kde=True, rug=False, fit=None, hist_kws=None, kde_kws=None, rug_kws=None, fit_kws=None, color=None, vertical=False, norm_hist=False, axlabel=None, label=None, ax=None)

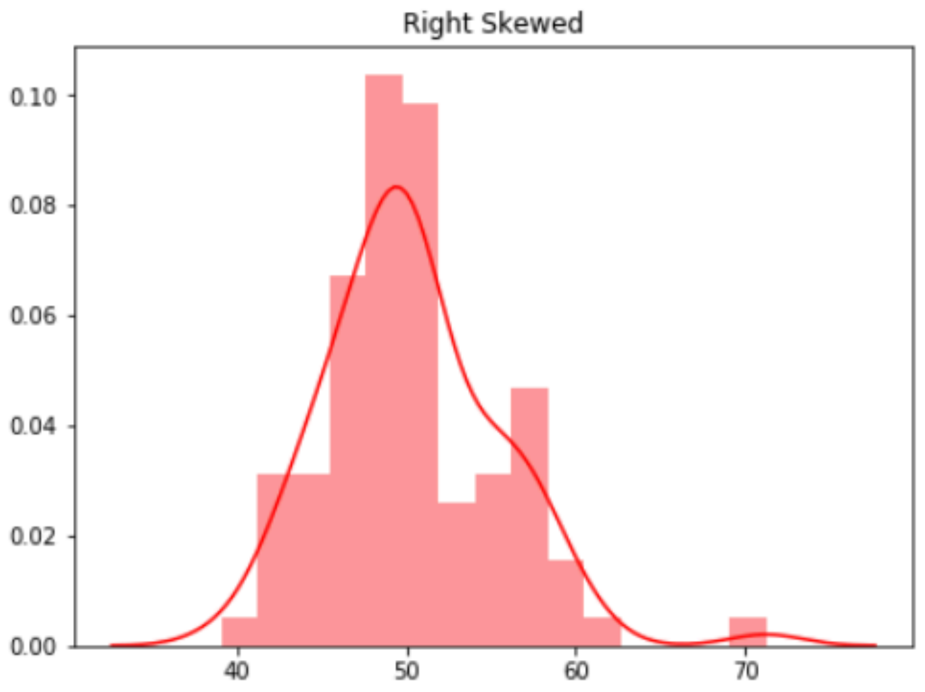
More details about the **seaborn** module can be found here:

https://seaborn.pydata.org/generated/seaborn.distplot.html

Not everyone is a Normal: Skewed distributions

Histograms are an excellent tool for identifying the shape of your distribution. So far, we've been looking at symmetric distributions, such as the normal distribution. However, not all distributions are symmetrical. You might have nonnormal data that are skewed.





The shape of the distribution is a fundamental characteristic of your sample that can determine which measure of central tendency best reflects the center of your data. Relatedly, the shape also impacts your choice between using a parametric or nonparametric hypothesis test. In this manner, histograms are informative about the summary statistics and hypothesis tests that are appropriate for your data.

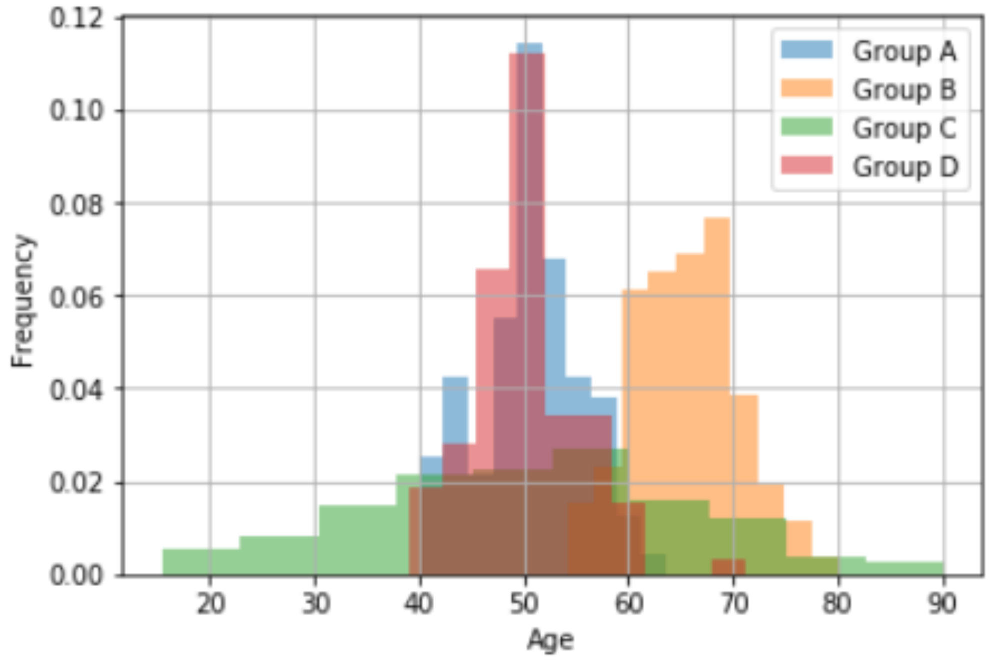
For right-skewed distributions, the long tail extends to the right while most values cluster on the left, as shown below.

```python
from scipy import stats

sk = stats.skew(df['Left Skew'].dropna())
print('The skewness of data set "Left Skew" is',sk)

sk = stats.skew(df['D'].dropna())
print('The skewness of data set "Right Skew" is',sk)
```

```
The skewness of data set "Left Skew" is -0.6052306911249611
The skewness of data set "Right Skew" is 0.6527984416681031
```

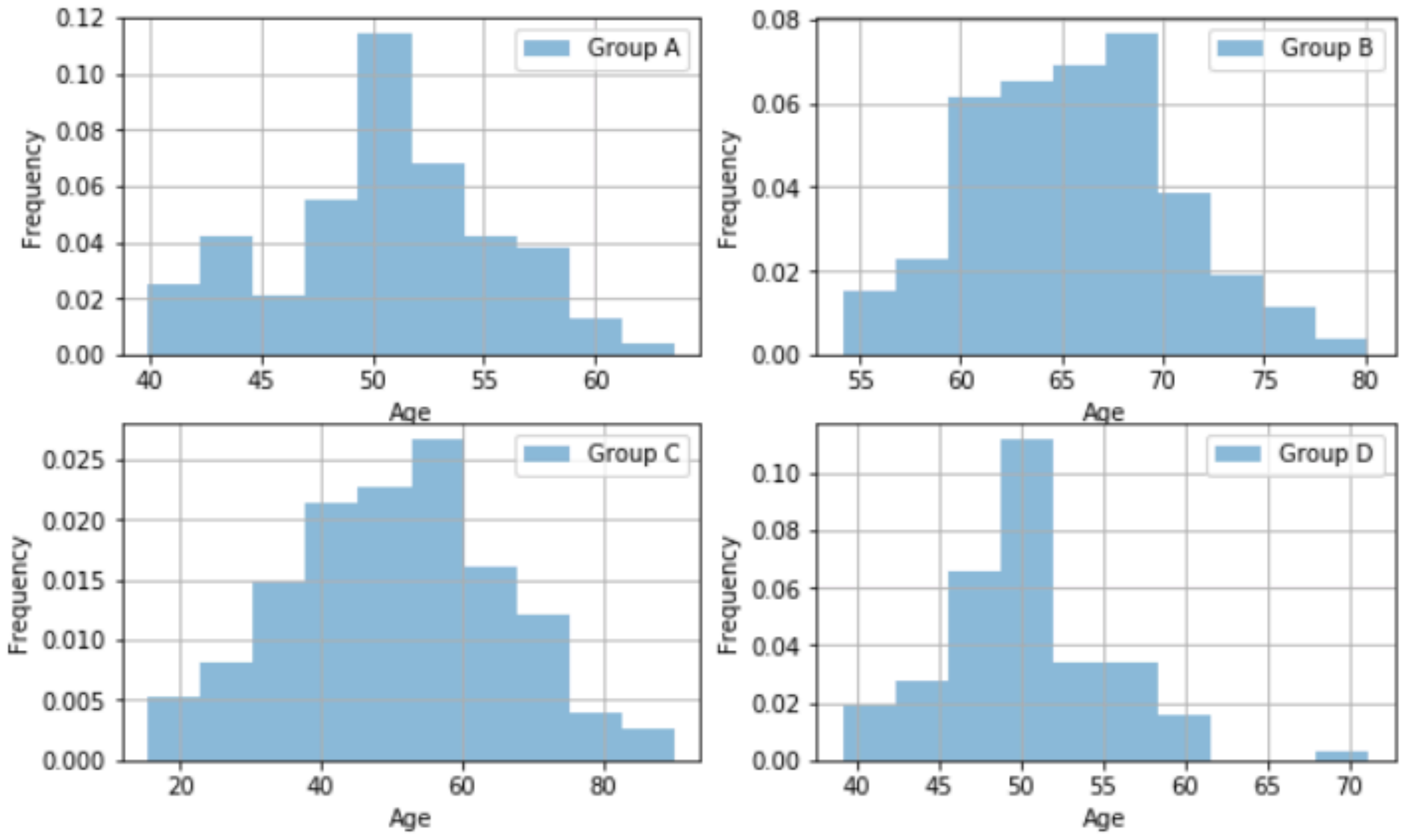
Compare distributions between groups

To compare distributions between groups using histograms, you'll need both a continuous variable and a categorical grouping variable. There are two common ways to display groups in histograms. You can either overlay the groups or graph them in different panels, as shown below.



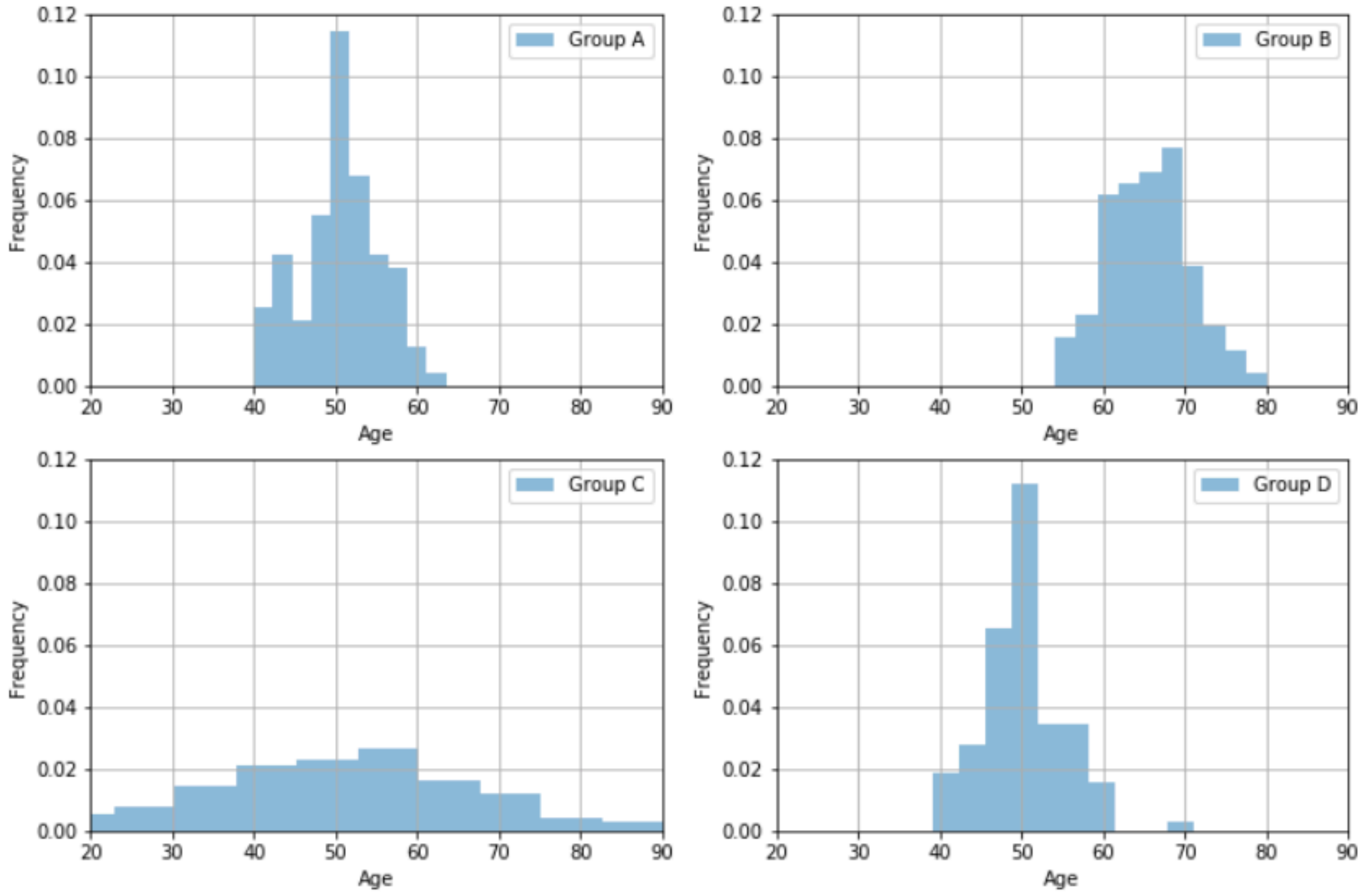
Method #1

Method #2



Compare distributions between groups

When comparing distributions across different groups, make sure adjusting the x-axis, y-axis so you're always compare apples to apples.

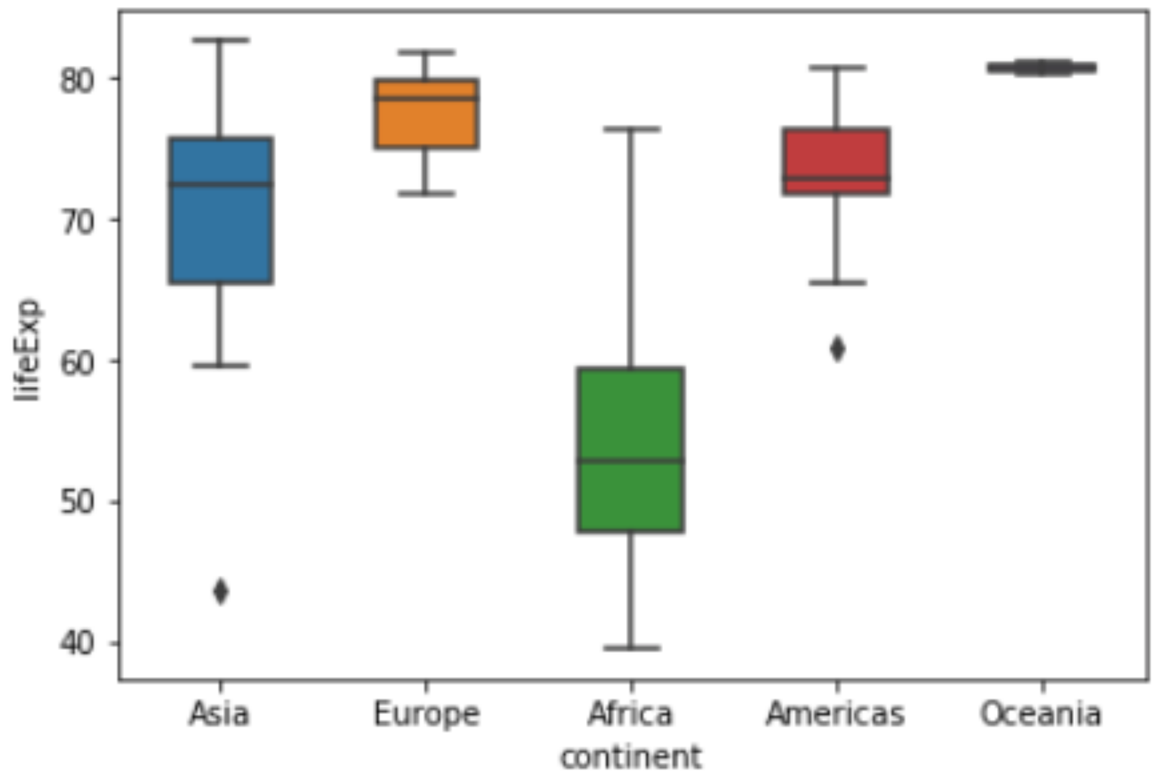


Understanding your data sets using Boxplots

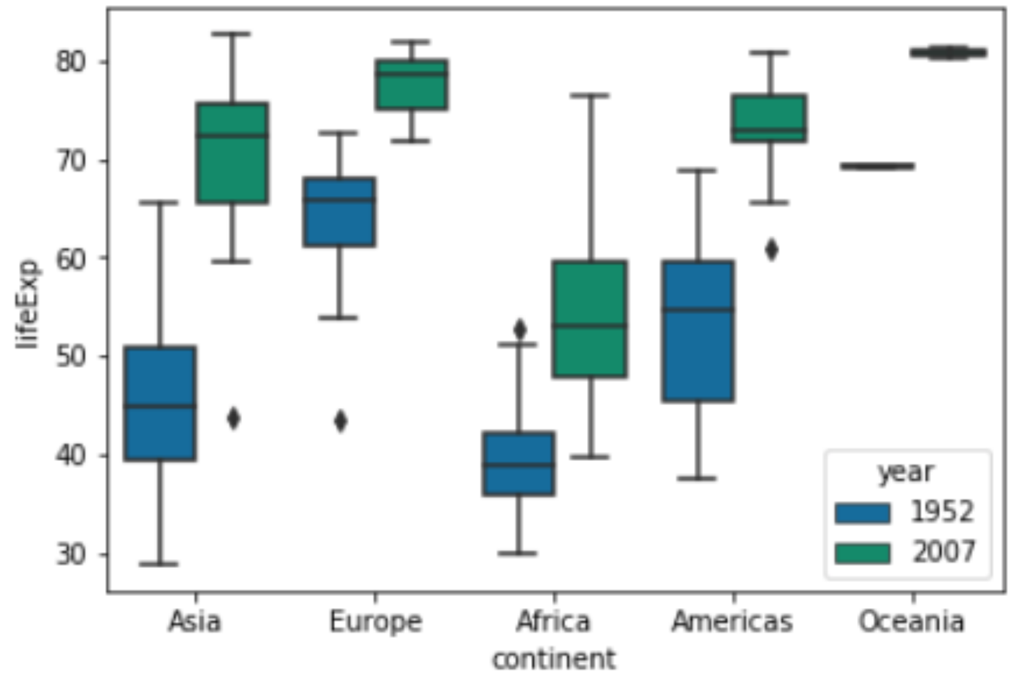
Boxplots are one of the most common ways to visualize data distributions from multiple groups. In Python, Seaborn potting library makes it easy to make boxplots and similar plots swarmplot and stripplot. Sometimes, your data might have multiple subgroups and you might want to visualize such data using grouped boxplots.

```python
file = "datasets/gdp_data.txt" # file location and name
gdp = pd.read_csv(file) # load data
print(gdp.columns) # print the columns of the data frame

# make a box plot
sns.boxplot(y='lifeExp',x='continent',data = gdp[(gdp.year==2007)], width=0.5)
plt.show()
```

```
Index(['country', 'year', 'pop', 'continent', 'lifeExp', 'gdpPercap'], dtype='object')
```



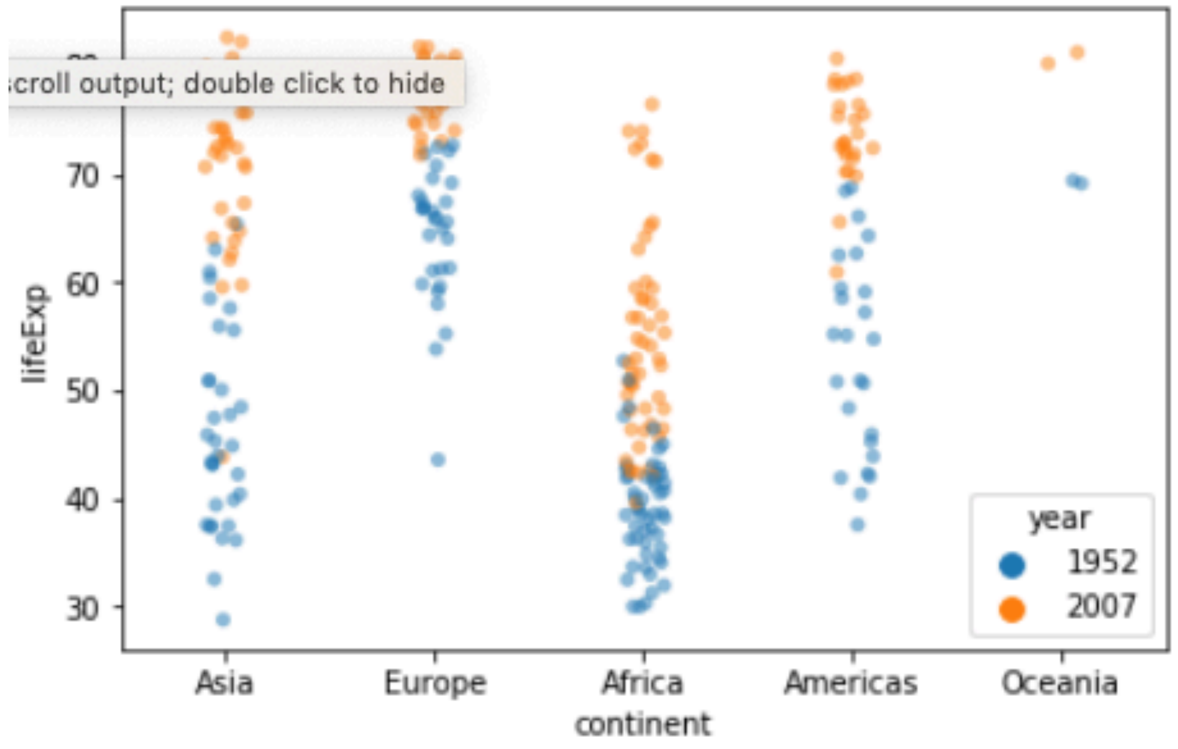
```python
df1 = gdp[gdp['year'].isin([1952,2007])]

df1.head()

sns.boxplot(y='lifeExp', x='continent',
            data=df1,
            palette="colorblind",
            hue='year')
```

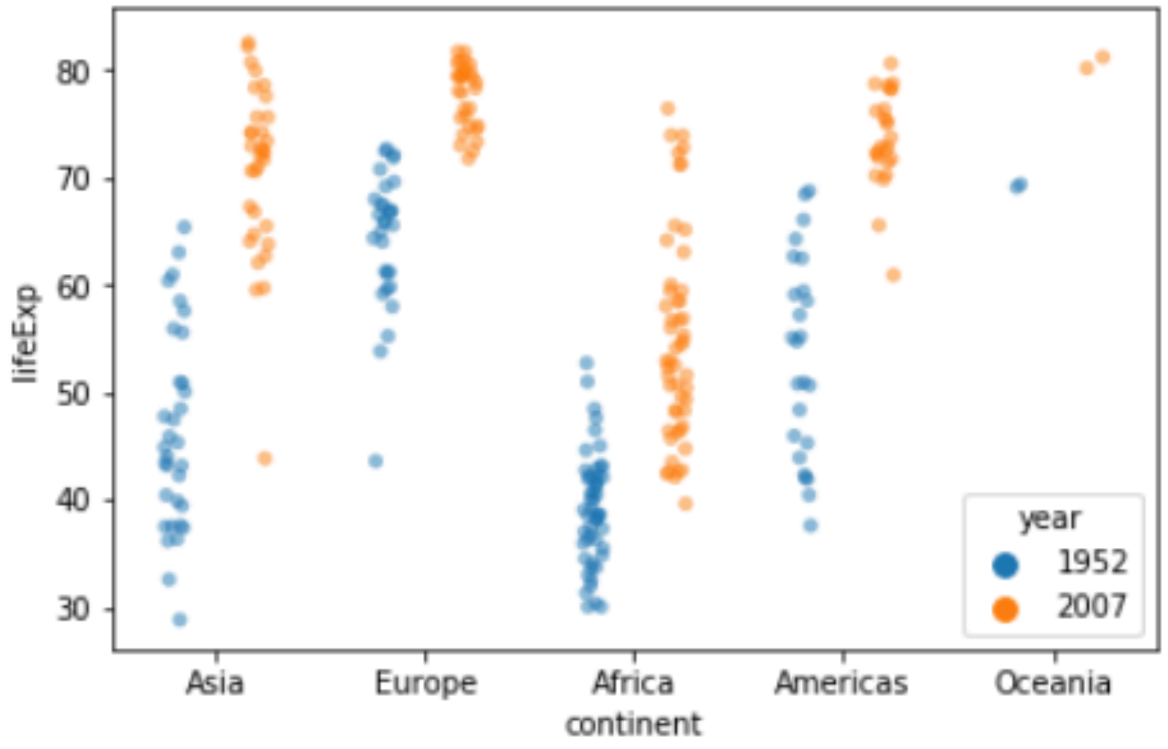


Understanding your data sets using Strip Plots

An alternative to boxplot in Python is simply plotting the original data points with jitter using Seaborn's stripplot. One of the biggest benefits of stripplot is we can actually see the original data and its distributions, instead of just the summary.

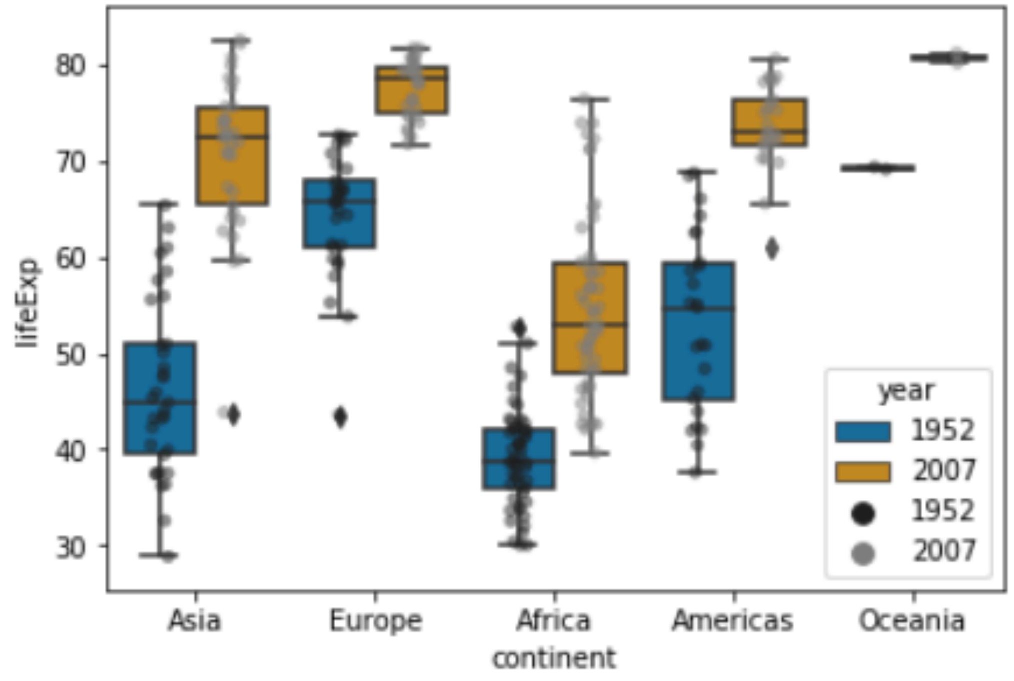
```
plt.figure(figsize=(14,4))
plt.subplot(1,2,1)
sns.stripplot(y='lifeExp', x='continent',
                    data=df1,
                    jitter=True,
                    marker='o',
                    alpha=0.5,
                   hue='year')

plt.subplot(1,2,2)
sns.stripplot(y='lifeExp', x='continent',
                   data=df1,
                   jitter=True,
                   dodge=True,
                   marker='o',
                   alpha=0.5,
                   hue='year')
plt.show()
```





Understanding your data sets using Strip Plots

```python
# make grouped boxplot
sns.boxplot(y='lifeExp', x='continent',
                 data=df1,
                 palette="colorblind",
                  hue='year')

# make grouped stripplot
sns.stripplot(y='lifeExp', x='continent',
                   data=df1,
                   jitter=True,
                   dodge=True,
                   marker='o',
                   alpha=0.5,
                   hue='year',
                   color='grey')
plt.show()
```



Put them together

Now let's try some real data!