

# Probability theory, statistics, simulation

---

EBA3500: Data analysis with programming

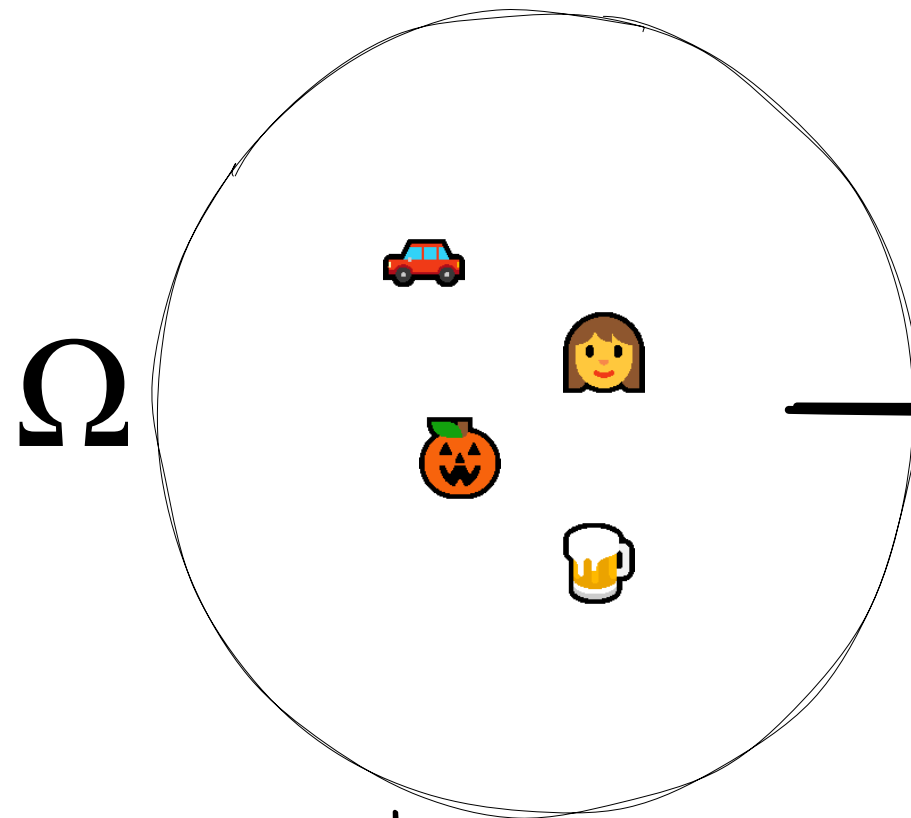
Lecturer: Jonas Moss

Contact: [jonas.moss@bi.no](mailto:jonas.moss@bi.no)

Office hours: Tuesdays, 10:00 - 11:00

What's a random variable  $X$  again?

sample space



the sample space  
contains WHATEVER

outcome space

$X$



$\dots, -2, -1, 0, 1, 2, 3, 4, \dots$

$E$

Some set, typically  
decimal numbers

What's a random variable again?

$$\underline{X : \Omega \rightarrow E}$$

$$x(\text{🚗}) = -2$$

$$x(\text{👩}) = 55$$

$$x(\text{🎃}) = 1$$

$$x(\text{🍺}) = 1000$$

Maps the sample space into the outcome space.

Why do we care? 🤔

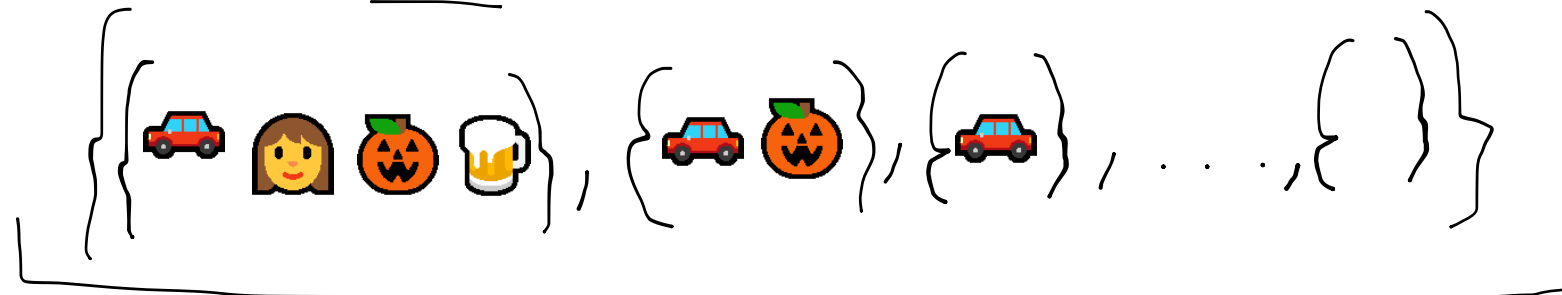
Because there's a PROBABILITY living on the sample space!

$$\underline{P : \mathcal{P}(\Omega) \rightarrow [0, 1]}$$

$$P : \mathcal{P}(\Omega) \rightarrow [0, 1]$$


---

$\mathcal{P}(\Omega)$  : The powerset of  $\Omega$ . Contains all subsets of  $\Omega$ .



Powerset of all four elements {     }

Exercise: Fill out the missing values.  
How many elements are there in a powerset of 4 elements? 5?  $N$ ?

$$\underline{P : \mathcal{P}(\Omega) \rightarrow [0, 1]}$$

The probability assigns a number to every member of the powerset. But it must obey Kolmogorov's axioms.

①  $P(\Omega) = 1, \quad P(\emptyset) = 0$

② Additivity.

$$P\left(\bigcup_{i=1}^n E_i\right) = \sum_{i=1}^n P(E_i), \quad E_i \text{ are disjoint sets}$$

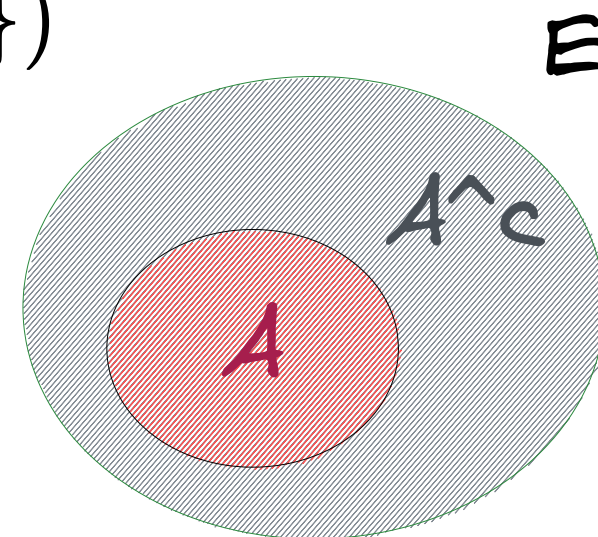
The random variable connects the probability space to the outcome space!

Suppose  $X$  is a random variable with outcome space  $E$ . Let  $A$  be a subset of  $E$ . Then we can define:

$$P(X \in A) = P(\{\omega \in \Omega \mid X(\omega) \in A\})$$

And, if  $E$  are numbers:

$$P(X \leq x) = P(\{\omega \in \Omega \mid X(\omega) \leq x\})$$



$$P(\text{🚗}) = 0.1$$

$$P(\text{👩}) = 0.5$$

$$P(\text{🎃}) = 0.2$$

$$P(\text{🍺}) = 0.2$$

$$x(\text{🚗}) = -2$$

$$x(\text{👩}) = 55$$

$$x(\text{🎃}) = 1$$

$$x(\text{🍺}) = 1000$$

$$P(x < 0) = 0.1$$

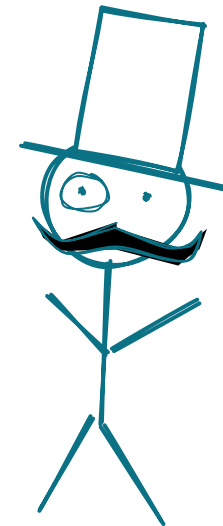
$$P(x < 2) = 0.3$$

$$P(x < 100) = 0.8$$

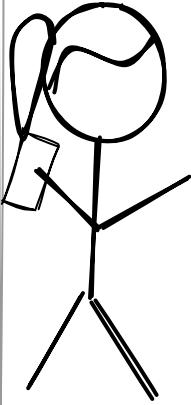
$$P(x < 1001) = 1$$

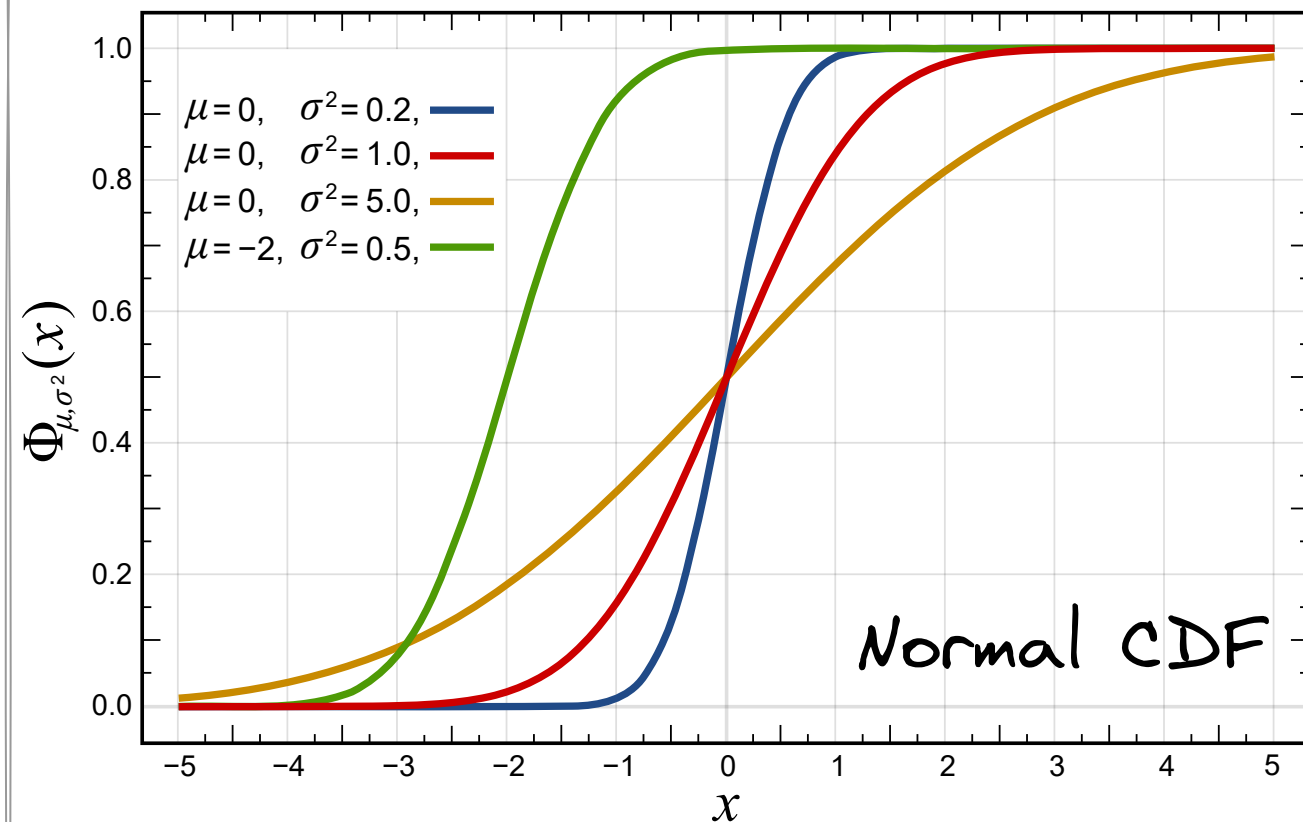


The probability space can be almost anything. But we can very very often ignore it. If we need it, we can *\*always\** make it equal to  $[0,1]$ .



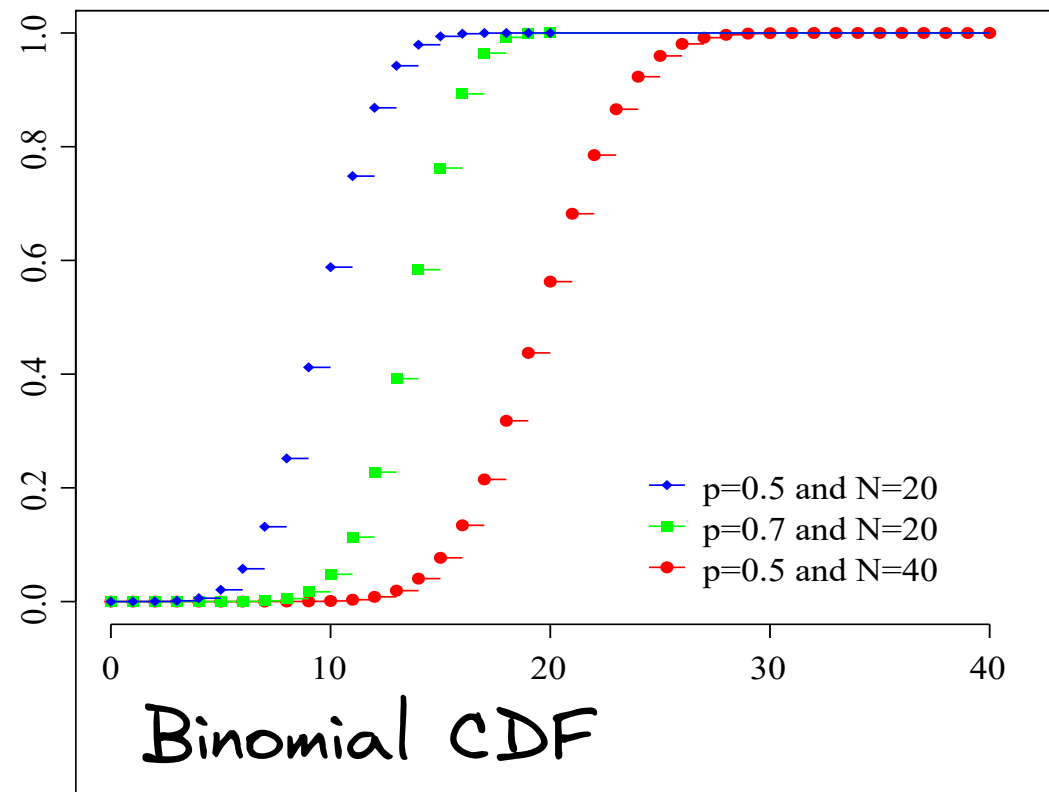
If  $E$  is a set of numbers,  
the probability  $P(X \text{ in } A)$ , for  $A$  a subset of  
 $E$  is completely described by the cumulative  
distribution function.


$$\begin{aligned} P(X \leq x) &= P(\{\omega \in \Omega \mid X(\omega) \leq x\}) \\ &= \underline{F(x)} \end{aligned}$$



- A CDF is non-decreasing.
- A CDF starts at 0.
- A CDF ends at 1.

Conversely, every non-decreasing function from 0 to 1 is a CDF!



• A CDF can have jumps!

• A CDF starts at 0.

• A CDF ends at 1.

• A CDF is non-decreasing.

$$\lim_{x \rightarrow \infty} F(x) = 1, \quad \lim_{x \rightarrow -\infty} F(x) = 0$$

(Formal definition of "ends at" and "starts at".)

Cumulative distribution functions (CDFs) are extremely useful since they describe everything about the relationship between the probability measure and the random variable!

## Probability density functions (PDF)

---

If the CDF has no jumps then

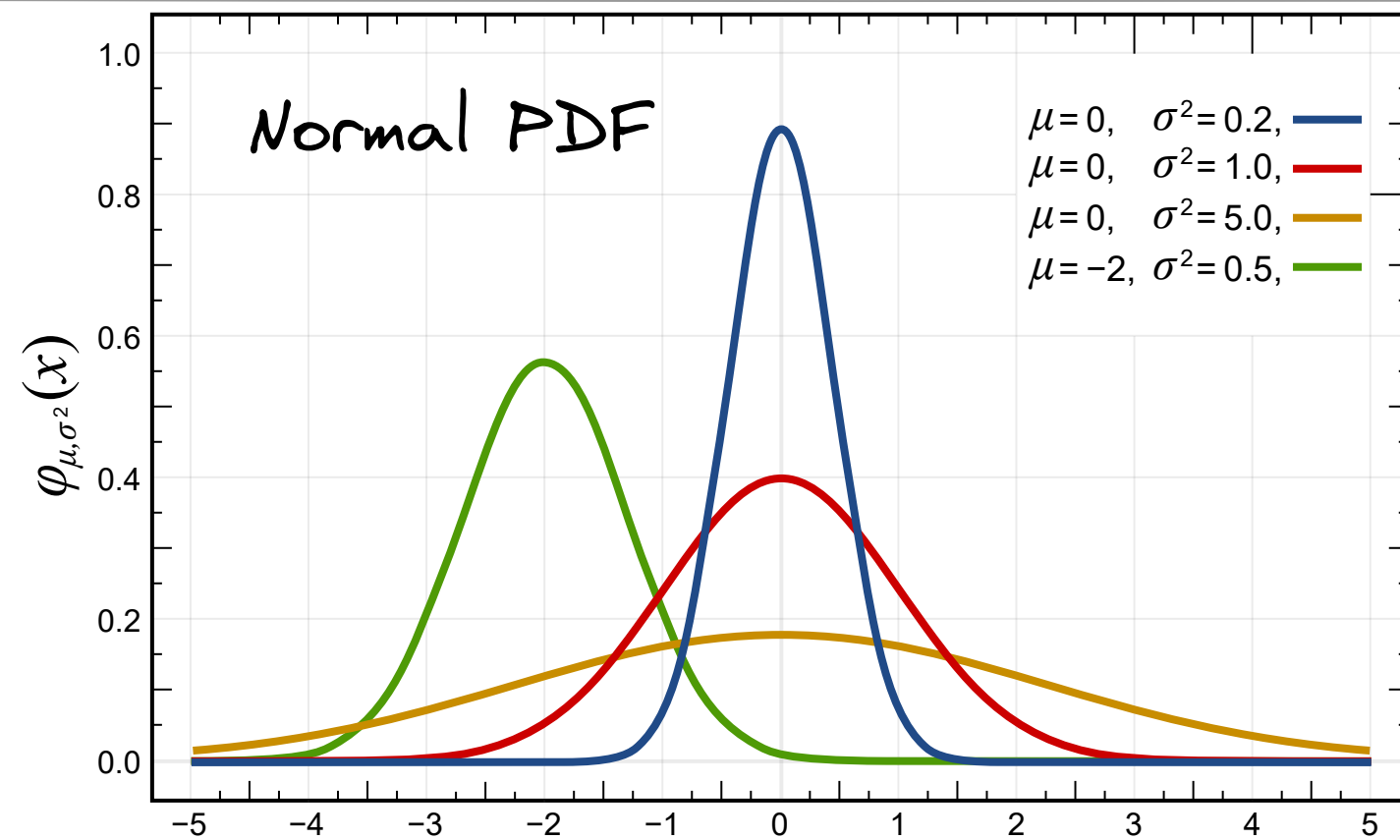
$$\frac{d}{dx}F(x) = f(x)$$

is the probability density function (PDF).

---

Equivalently, by the Fundamental Theorem of Calculus:

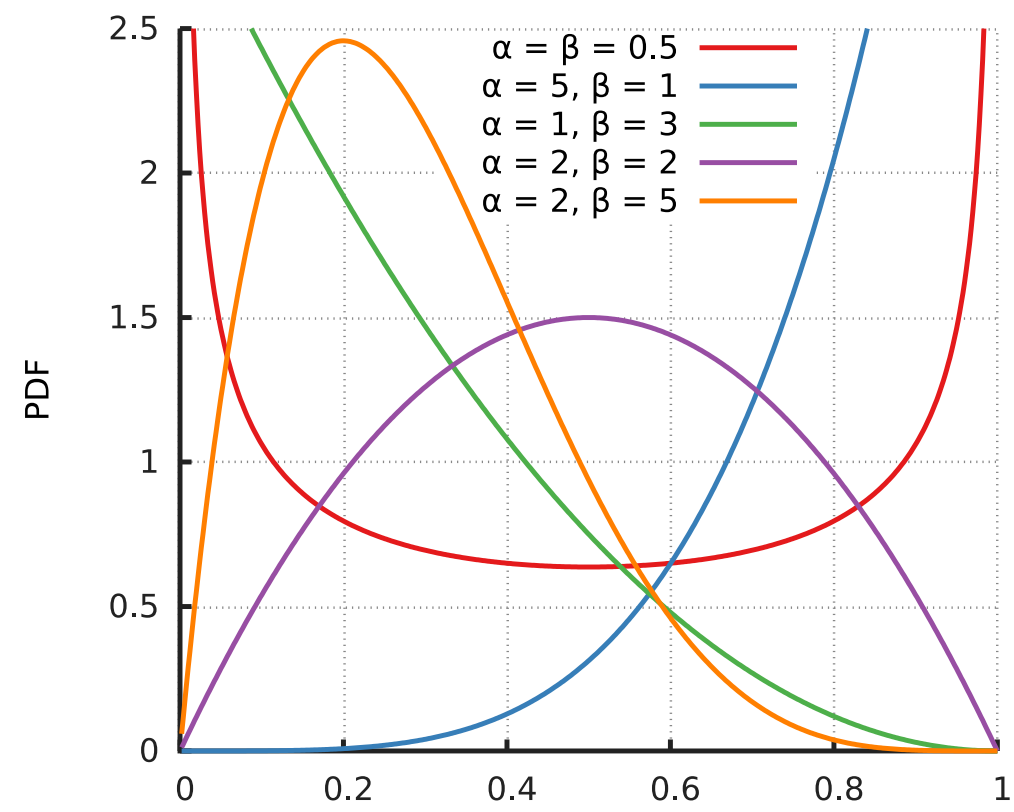
$$P(X \leq x) = F(x) = \int_{-\infty}^x f(x)dx$$



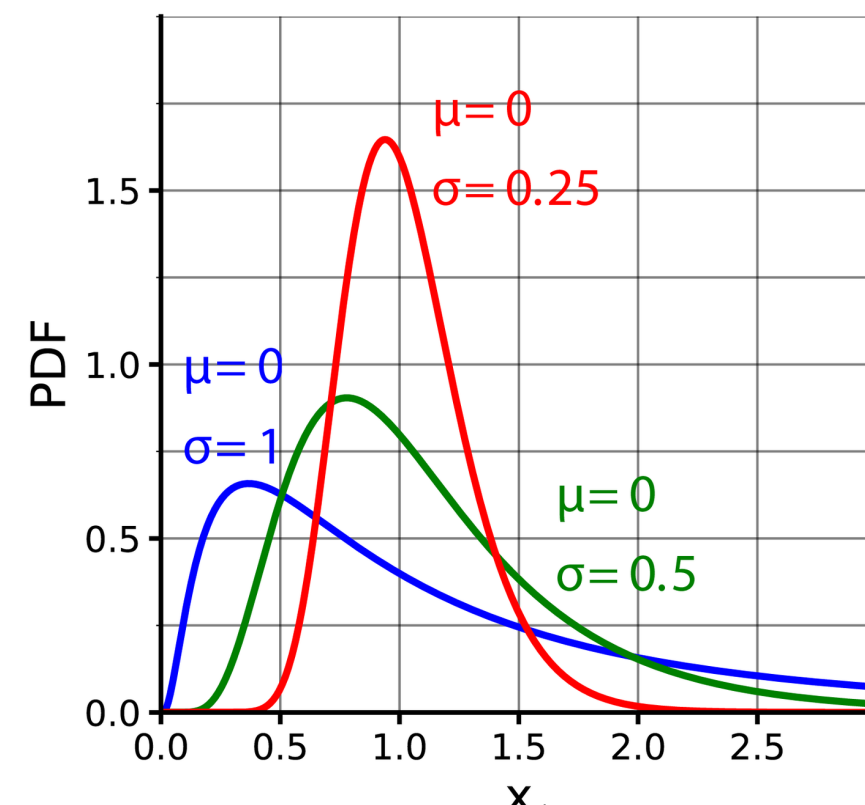
- A PDF is non-negative.
- A PDF integrates to 1.

$$\int_{-\infty}^{\infty} f(x) dx = 1$$

Conversely, any non-negative function integrating to 1 is a PDF of a random variable (whose CDF has no jumps.)



Beta PDF



Lognormal PDF

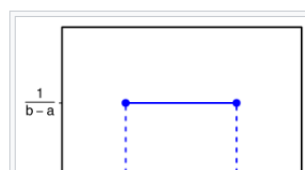
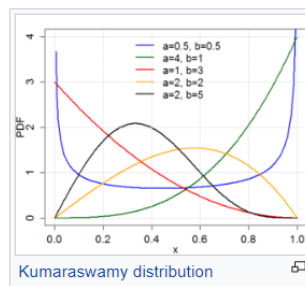
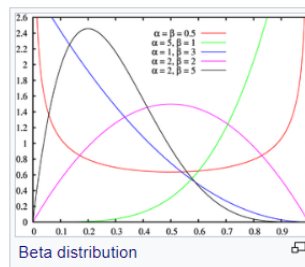


# Find more examples on Wikipedia!

Absolutely continuous distributions [edit]

Supported on a bounded interval [edit]

- The [Beta distribution](#) on  $[0,1]$ , a family of two-parameter distributions with one mode, of which the uniform distribution is a special case, and which is useful in estimating success probabilities.
- The [arcsine distribution](#) on  $[a,b]$ , which is a special case of the Beta distribution if  $\alpha = \beta = 1/2$ ,  $a = 0$ , and  $b = 1$ .
- The [PERT distribution](#) is a special case of the beta distribution
- The [uniform distribution](#) or rectangular distribution on  $[a,b]$ , where all points in a finite interval are equally likely.
- The [Irwin–Hall distribution](#) is the distribution of the sum of  $n$  independent random variables, each of which having the uniform distribution on  $[0,1]$ .
- The [Bates distribution](#) is the distribution of the mean of  $n$  independent random variables, each of which having the uniform distribution on  $[0,1]$ .
- The [logit-normal distribution](#) on  $(0,1)$ .
- The [Dirac delta function](#) although not strictly a probability distribution, is a limiting form of many continuous probability functions. It represents a *discrete* probability distribution concentrated at 0 — a *degenerate distribution* — it is a [Distribution](#)



Absolutely continuous means that a PDF exists.

Do you remember when a PDF exists? 🎅 🧐

🌐 [https://en.wikipedia.org/wiki/List\\_of\\_probability\\_distributions](https://en.wikipedia.org/wiki/List_of_probability_distributions)

Remember, the density is not a probability.

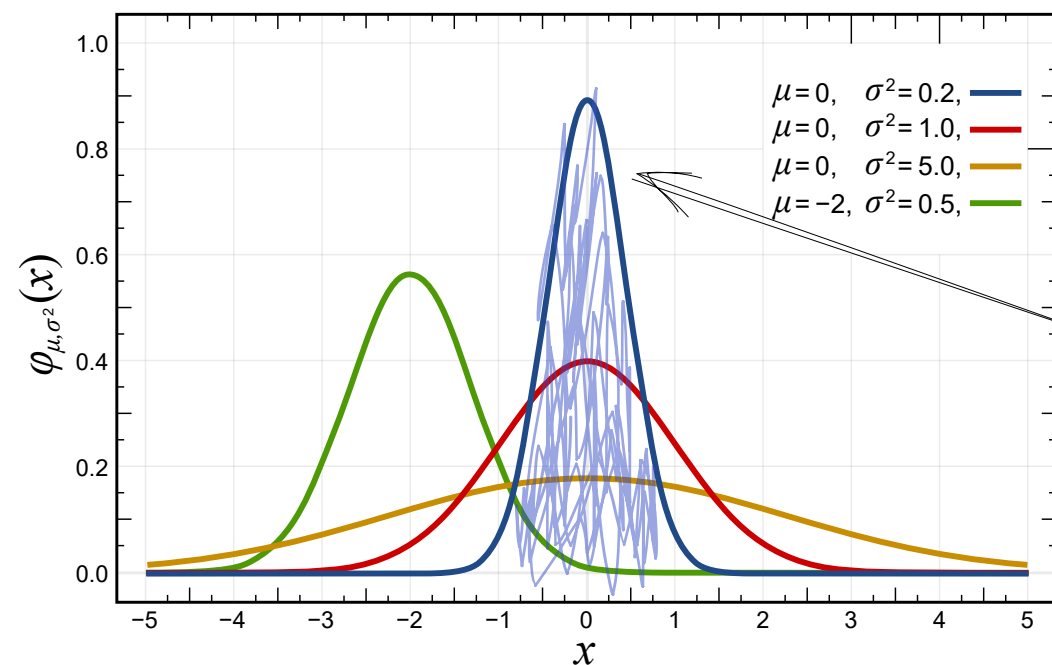
$$P(a \leq X \leq b) = F(b) - F(a) = \int_a^b f(x)dx$$

---

The probability of  $X = a$  when the CDF  
is without jumps is 0!

$$P(X = a) = P(a \leq X \leq a) = F(a) - F(a) = 0$$

But it's "sort of like" a probability.  
You can look at the curve and decide which  
values are most probable visually.



Remember that the integral  
is the area the curve. 😎

That's a very probable  
area! Probably about 95%?

Know your expected values! 

$$E(X) = \int_{-\infty}^{\infty} x f(x) dx = \mu$$

The expectation of  $X$ . This is often called  $\mu$ .

$$E(X^k) = \int_{-\infty}^{\infty} x^k f(x) dx$$

The  $k$ th moment of  $X$ .

$$E(g(X)) = \int_{-\infty}^{\infty} g(x) f(x) dx$$

The expectation of  $g(X)$ .

# More than two variables.

I'm going through this stuff fast! 

But don't worry - this course is about random variables with more than one dimensions! You'll understand soon enough.

We'll simulate a lot of this stuff.

## Two or more random variables?

$$\underbrace{(X, Y)} : \Omega \rightarrow \mathbb{R}^2$$

"Bivariate variable."

We now map the elements of the probability space to TWO random elements each!

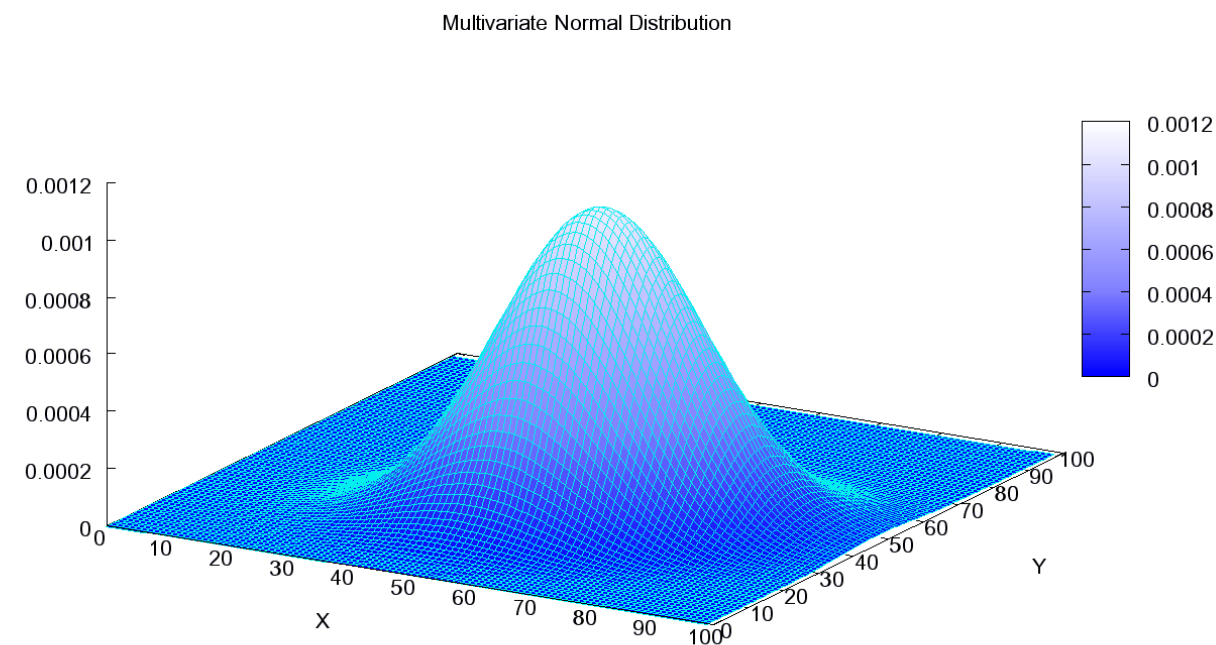
Joint CDF  $\leftrightarrow$  CDF

Joint PDF  $\leftrightarrow$  PDF

$$\underbrace{P(X \leq x, Y \leq y)} = \int_{-\infty}^x \int_{-\infty}^y \underbrace{f(x, y)} dx$$

# Joint densities

---



Same interpretation as an ordinary PDF: Area under curve is the probability.

The PDF of a bivariate normal.

Bivariate PDFs are not common to work with directly.

For one variable we have:

---

$$P(X \leq x) = \int_{-\infty}^x f(x) dx$$

For two, we have:

---

$$P(X \leq x, Y \leq y) = \int_{-\infty}^x \int_{-\infty}^y f(x, y) dx dy$$

In particular:

---

$$P(X < \infty, Y \leq y) = \int_{-\infty}^{\infty} \int_{-\infty}^y f(x, y) dx dy = P(Y \leq y)$$



## Joint densities

$$f(x | y) = \underbrace{f(x, y)}_{\text{A two-dimensional PDF! Not very different from a one-dimensional.}} / f(y)$$

X: The weight of a Norwegian male adult.

Y: The height of a Norwegian male adult.

---

$f(y | x = 60\text{kg})$  : The density of an adults height conditional on the weight being 60 kg!

## Conditional expectation

$$E(Y \mid X = x) = \int_{-\infty}^{\infty} y f(y \mid x) dy$$

$X$ : The weight of a Norwegian male adult.

$Y$ : The height of a Norwegian male adult.

$E(Y)$                       Mean height of an adult.

$E(Y \mid X = 99\text{kg})$       Mean height of an adult whose weight is 99kg.

## Iterated expectations

---

Also known as "law of total expectation".

$$E[E(Y \mid X)] = E[Y]$$

---

$E(Y)$

Mean height of an adult.

$E(Y \mid X = x)$

Mean height of an adult whose weight is  $x$  kg.

---

If you average out the  $E(Y \mid X = x)$  over  $x$ ,  
you obtain  $E(Y)$ !

## Iterated expectations

---

$$\begin{aligned} E[E(Y | X)] &= \int_{-\infty}^{\infty} \left( \int_{-\infty}^{\infty} y f(y | x) dy \right) f(x) dx && \text{Definition} \\ &= \int_{-\infty}^{\infty} \left( \int_{-\infty}^{\infty} y f(y | x) f(x) dy \right) dx && \text{Putting } f(x) \text{ inside.} \\ &= \int_{-\infty}^{\infty} \left( \int_{-\infty}^{\infty} y f(y, x) dy \right) dx && \text{Definition, } f(y|x) \\ &= \int_{-\infty}^{\infty} y \left( \int_{-\infty}^{\infty} f(y, x) dx \right) dy && \text{Move } y \text{ outside.} \\ &= \int_{-\infty}^{\infty} y f(y) dy && \text{Integrate out } x. \end{aligned}$$

## Central moments, variance, and kurtosis

$$E[(X - \mu)^k] = \int_{-\infty}^{\infty} (x - \mu)^k f(x) dx$$

The kth central moment.

$$\text{Var}(X) = E[(X - \mu)^2] = E(X^2) - E(X)^2 = \sigma^2$$

The variance

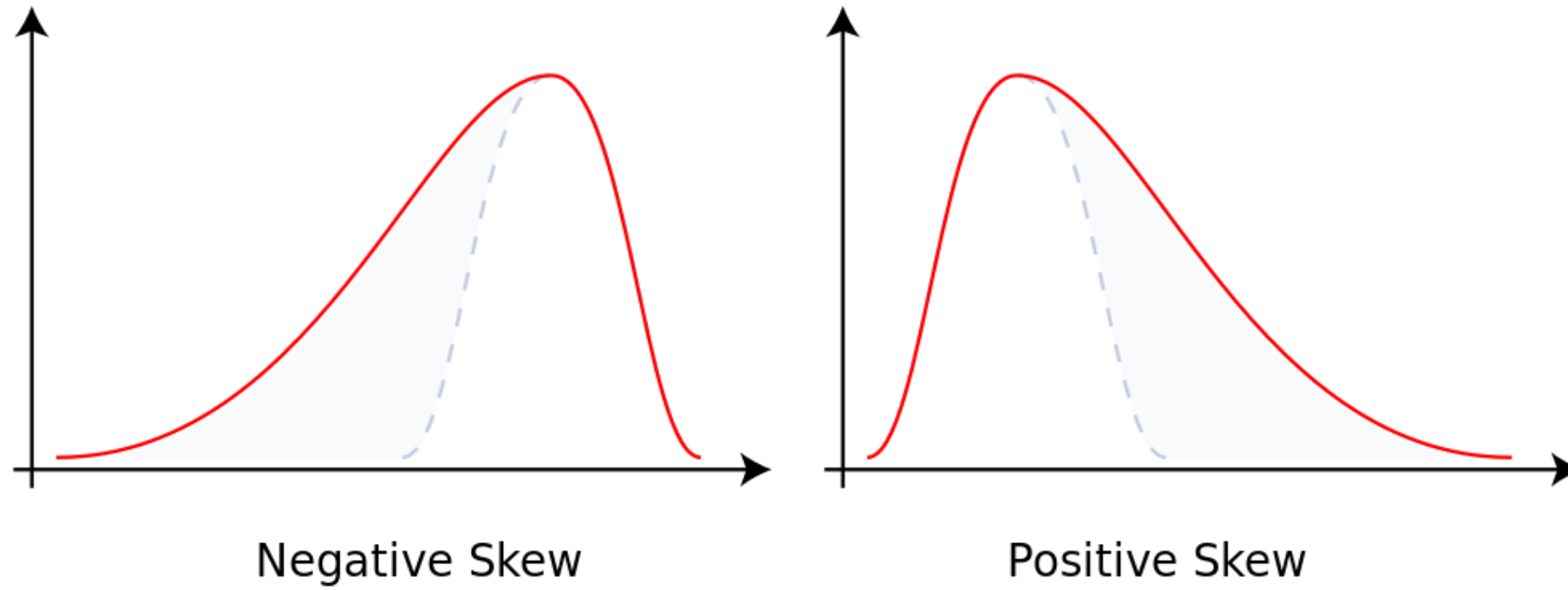
$$\text{Skew}(X) = \frac{E[(X - \mu)^3]}{\sigma^{3/2}}$$

The skewness of the distribution.  
Sort of measures asymmetry.

$$\text{Kurt}(X) = \frac{E[(X - \mu)^4]}{\sigma^2}$$

The kurtosis of the distribution.  
VERY "sort of" measures the  
thickness of the tails.

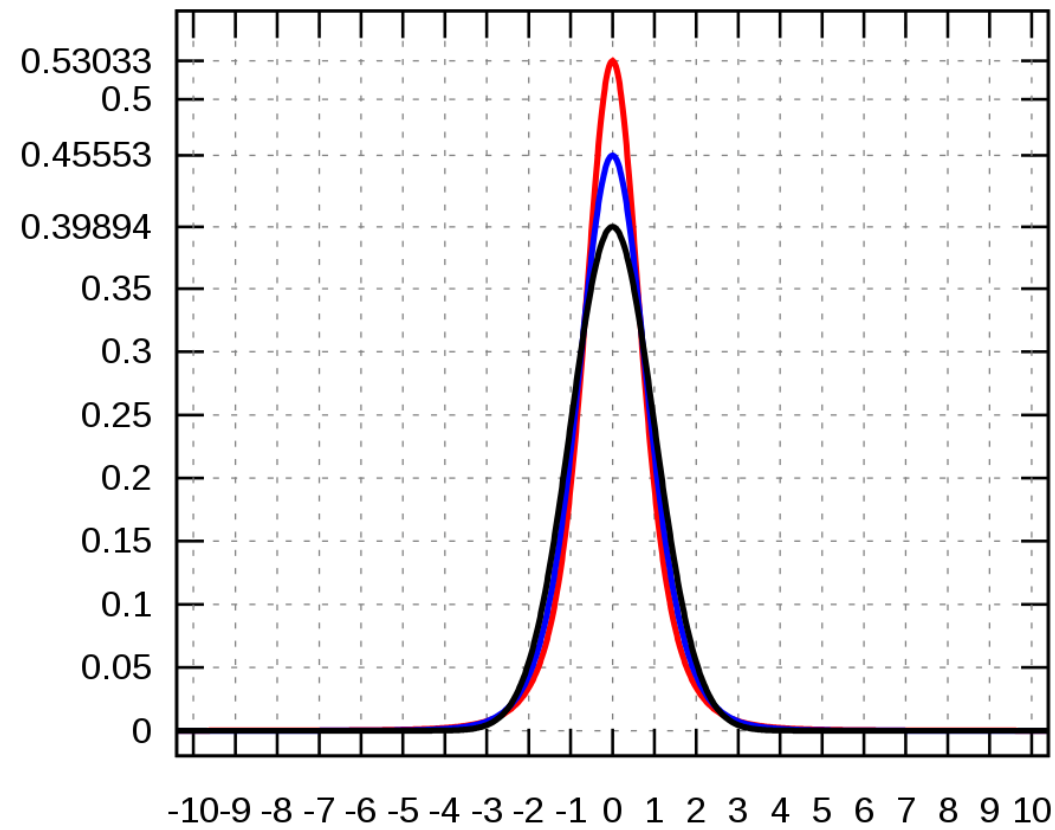
What skew means!



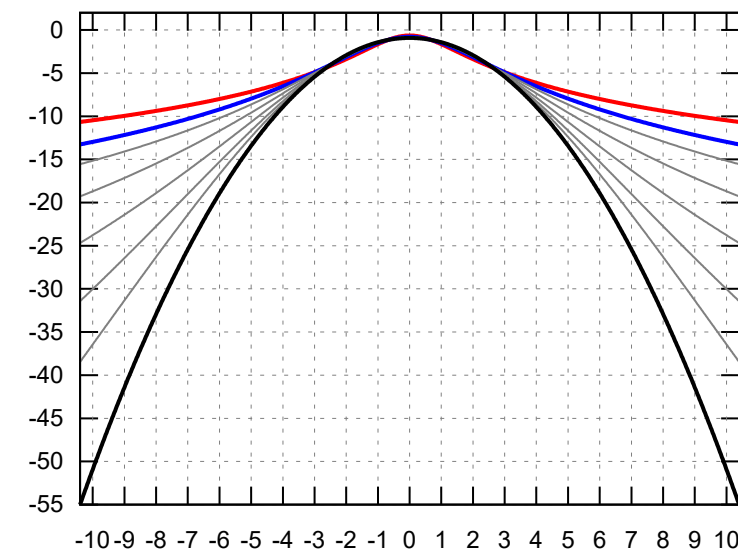
Often "excess kurtosis" is used.

The normal distribution has kurtosis 3, and the excess kurtosis measures how much more, or less, kurtosis the distribution has than the relatively thinly tailed normal distribution.

$$\text{ExcessKurt}(X) = \frac{E[(X - \mu)^4]}{\sigma^2} - 3$$



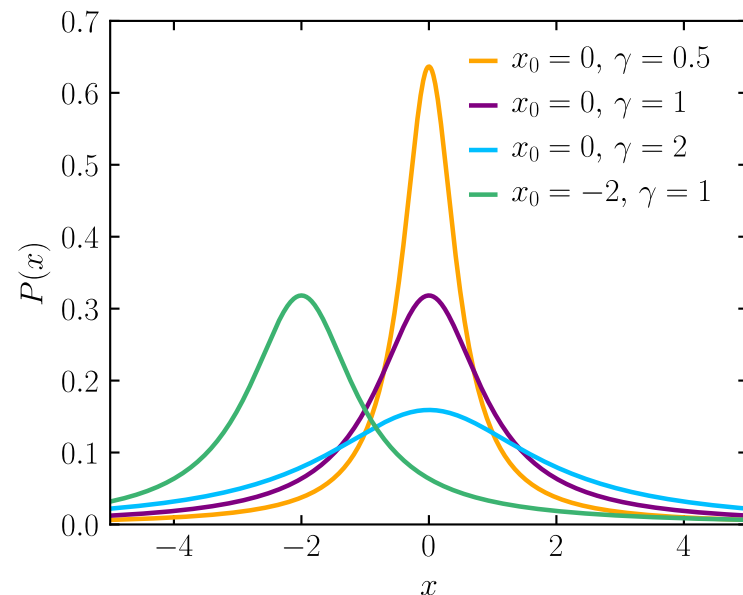
On the log-scale you  
can. The red is  
heaviest.



The Pearson type VII distribution with excess kurtosis of:  
infinity (red); 2 (blue); 1, 1/2, 1/4, 1/8, and 1/16 (gray);  
and 0 (black)



## (Excess) kurtosis is kinda stupid!



(This is a measure mostly used by non-statisticians. 😊)

The Cauchy distribution is extremely heavy-tailed. It has no mean, no variance, no skewness, and no kurtosis!

There are many distributions with fat tails with infinite kurtosis. And, of course, the kurtosis can't differentiate between them!

# Random sampling

---

- \* We still have some probability left. But we need to go through random sampling first!
- \* We'll go through:
  - convergence in probability
  - consistency
  - the central limit theorem
  - asymptotic normality.

```
# Exponential variables.  
import numpy as np  
import seaborn as sns  
import matplotlib.pyplot as plt  
from scipy import stats  
rng = np.random.default_rng(seed = 313)  
n_reps = 10 ** 5  
n = 100  
x = rng.standard_exponential((n_reps, n))
```

????

This is a random seed!

```
>>> x  
array([[0.97812657, 0.19291269, 3.6193487 , ..., 0.43550025, 1.99655493,  
        0.37468738],  
       [1.60635369, 1.79141188, 1.201482  , ..., 0.22350811, 3.66340103,  
        3.82593416],  
       [0.86975771, 0.40404461, 1.08379262, ..., 3.58329629, 0.79628372,  
        0.54135605],  
       ...,  
       [2.72648332, 1.40583745, 1.2140306 , ..., 0.67786698, 0.20422725,  
        1.52255718],  
       [1.97977493, 0.49726604, 0.01280329, ..., 1.67049025, 0.28688934,  
        0.79747027],  
       [1.12133612, 2.79753345, 0.72723982, ..., 2.93122518, 1.4128838 ,  
        1.53680554]])
```

True randomness  
does not exist on  
a computer!

```
# Exponential variables.  
import numpy as np  
import seaborn as sns  
import matplotlib.pyplot as plt  
from scipy import stats  
rng = np.random.default_rng(seed = 313)  
n_reps = 10 ** 5  
n = 100  
x = rng.standard_exponential((n_reps, n))
```

Notice that the random number generators are method of the rng object.

????

We use the seed for reproducibility. If you run the same code twice you get the same result!

Always use a seed! That way we can run the same code, on different machines, and get the same result!

Distributions

<code>beta(a, b[, size])</code>	Draw samples from a Beta distribution.
<code>binomial(n, p[, size])</code>	Draw samples from a binomial distribution.
<code>chisquare(df[, size])</code>	Draw samples from a chi-square distribution.
<code>dirichlet(alpha[, size])</code>	Draw samples from the Dirichlet distribution.
<code>exponential([scale, size])</code>	Draw samples from an exponential distribution.
<code>f(dfnum, dfden[, size])</code>	Draw samples from an F distribution.
<code>gamma(shape[, scale, size])</code>	Draw samples from a Gamma distribution.
<code>geometric(p[, size])</code>	Draw samples from the geometric distribution.
<code>gumbel([loc, scale, size])</code>	Draw samples from a Gumbel distribution.
<code>hypergeometric(ngood, nbad, nsample[, size])</code>	Draw samples from a Hypergeometric distribution.
<code>laplace([loc, scale, size])</code>	Draw samples from the Laplace or double exponential distribution with specified location (or mean) and scale (decay).
<code>logistic([loc, scale, size])</code>	Draw samples from a logistic distribution.
<code>lognormal(mean, sigma, size)</code>	Draw samples from a log-normal distribution.
<code>logseries(p[, size])</code>	Draw samples from a logarithmic series distribution.
<code>multinomial(n, pvals[, size])</code>	Draw samples from a multinomial distribution.
<code>multivariate_normal(mean, cov[, size, ...])</code>	Draw random samples from a multivariate normal distribution.
<code>negative_binomial(n, p[, size])</code>	Draw samples from a negative binomial distribution.
<code>noncentral_chisquare(df, nonc[, size])</code>	Draw samples from a noncentral chi-square distribution.
<code>noncentral_f(dfnum, dfden, nonc[, size])</code>	Draw samples from the noncentral F distribution.
<code>normal([loc, scale, size])</code>	Draw random samples from a normal (Gaussian) distribution.
<code>pareto(a[, size])</code>	Draw samples from a Pareto II or Lomax distribution with specified shape.
<code>poisson([lam, size])</code>	Draw samples from a Poisson distribution.
<code>power(a[, size])</code>	Draws samples in [0, 1] from a power distribution with positive exponent a - 1.
<code>rayleigh([scale, size])</code>	Draw samples from a Rayleigh distribution.
<code>standard_cauchy([size])</code>	Draw samples from a standard Cauchy distribution with mode = 0.
<code>standard_exponential([size])</code>	Draw samples from the standard exponential distribution.
<code>standard_gamma(shape[, size])</code>	Draw samples from a standard Gamma distribution.
<code>standard_normal([size])</code>	Draw samples from a standard Normal distribution (mean=0, stdev=1).
<code>standard_t(df[, size])</code>	Draw samples from a standard Student's t distribution with <i>df</i> degrees of freedom.
<code>triangular(left, mode, right[, size])</code>	Draw samples from the triangular distribution over the interval <code>[left, right]</code> .
<code>uniform([low, high, size])</code>	Draw samples from a uniform distribution.
<code>vonmises(mu, kappa[, size])</code>	Draw samples from a von Mises distribution.
<code>wald(mean, scale[, size])</code>	Draw samples from a Wald, or inverse Gaussian, distribution.
<code>weibull(a[, size])</code>	Draw samples from a Weibull distribution.
<code>zipf(a[, size])</code>	Draw samples from a Zipf distribution.

Random sampling (numpy.random)

Simple random data

<code>rand(d0, d1, ..., dn)</code>	Random values in a given shape.
<code>randn(d0, d1, ..., dn)</code>	Return a sample (or samples) from the “standard normal” distribution.
<code>randint(low[, high, size, dtype])</code>	Return random integers from <i>low</i> (inclusive) to <i>high</i> (exclusive).
<code>random_integers(low[, high, size])</code>	Random integers of type np.int between <i>low</i> and <i>high</i> , inclusive.
<code>random_sample([size])</code>	Return random floats in the half-open interval [0.0, 1.0).
<code>random([size])</code>	Return random floats in the half-open interval [0.0, 1.0).
<code>ranf([size])</code>	Return random floats in the half-open interval [0.0, 1.0).
<code>sample([size])</code>	Return random floats in the half-open interval [0.0, 1.0).
<code>choice(a[, size, replace, p])</code>	Generates a random sample from a given 1-D array
<code>bytes(length)</code>	Return random bytes.

Permutations

<code>shuffle(x)</code>	Modify a sequence in-place by shuffling its contents.
<code>permutation(x)</code>	Randomly permute a sequence, or return a permuted range.

Random generator

<code>RandomState([seed])</code>	Container for the Mersenne Twister pseudo-random number generator.
<code>seed([seed])</code>	Seed the generator.
<code>get_state()</code>	Return a tuple representing the internal state of the generator.
<code>set_state(state)</code>	Set the internal state of the generator from a tuple.

Not as much variety as Scipy,  
but usually enough

# Convergence in probability

$$\lim_{n \rightarrow \infty} P(|X_n - a| > \epsilon) = 0, \quad \text{for all } \epsilon > 0.$$

Means that the random variable  $X_n$  gets arbitrarily close to  $a$  as  $n$  goes to infinity.

$$X_n \xrightarrow{p} a$$

# Convergence in probability

---

$$X_n = 2 + \frac{1}{n}X$$

This is true no matter what  $X$  is!

$$X_n \xrightarrow{p} 2$$

---

$$Y_n = \frac{1}{n} \sum_{i=1}^n X_i, \quad X_i \stackrel{iid}{\sim} F$$

This is the law of large numbers.

$$Y_n \xrightarrow{p} E(X_i)$$

# Examples

## Law of large numbers



1. Let  $X_n$  be normal with standard deviation  $1/\sqrt{n}$  and mean 0. Then  $X_n$  converges in probability to 0.
2. Let  $X_n$  be a sequence of random variables that is eventually constant, i.e., there is an  $N$  and a  $c$  so that  $X_n = c$  for all  $n \geq N$ .
3. Let  $X_n = X(-1)^n$  for some non-constant random variable  $X$ . Then  $X_n$  will oscillate between  $-X$  and  $X$  indefinitely, and does not converge. When  $n$  is even,  $(-1)^n = 1$ , when  $n$  is odd,  $(-1)^n = -1$ .



# Consistency

---

An estimator is consistent for its population value if it converges in probability to its sample value.

$$\hat{\theta} \xrightarrow{p} \theta$$

# How to show an estimator is consistent?

---

*Assume that  $Y_n$  are random variables and  $a$  is a constant such that*

$$\begin{aligned} EY_n &\rightarrow a, \\ \text{Var}(Y_n) &\rightarrow 0, \end{aligned}$$

*as  $n \rightarrow \infty$ . Then*

$$Y_n \rightarrow a,$$

*as  $n \rightarrow \infty$ , in probability.*

# How to "show" an estimator is consistent

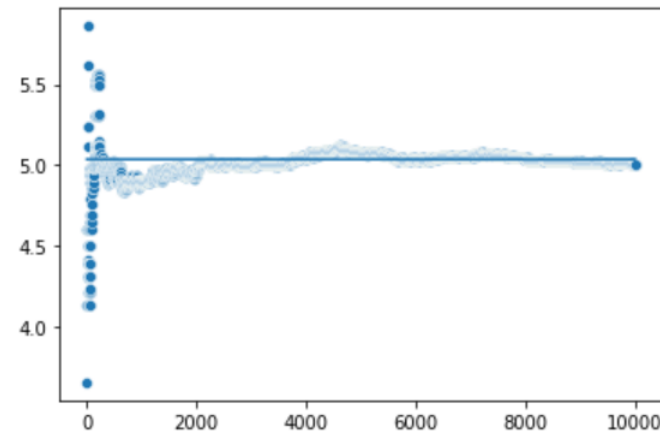
---

```
1 import numpy as np
2 import seaborn as sns
3 from scipy import stats
4
5 rng = np.random.default_rng(313)
6 x = rng.gamma(2, 3, 10000)
7 medians = [np.median(x[0:i]) for i in range(1, 10001)]
8 sns.scatterplot(np.arange(1, 10001), medians)
9 sns.lineplot(np.arange(1, 10001), stats.gamma(2, 0, 3).median())
10 # Different parameterization in Scipy!
```

---

This is not a mathematical proof, but  
enough almost all the time!

---



How long do we have to wait for the law of large numbers to kick in?

### The weak law of large numbers

If  $X_i$  are iid with finite expected value  $E(X_1)$ , its sample mean converges in probability to  $\bar{X}_n$

$$\bar{X}_n = \frac{1}{n} \sum_{i=1}^n X_i \xrightarrow{p} E(X_1)$$

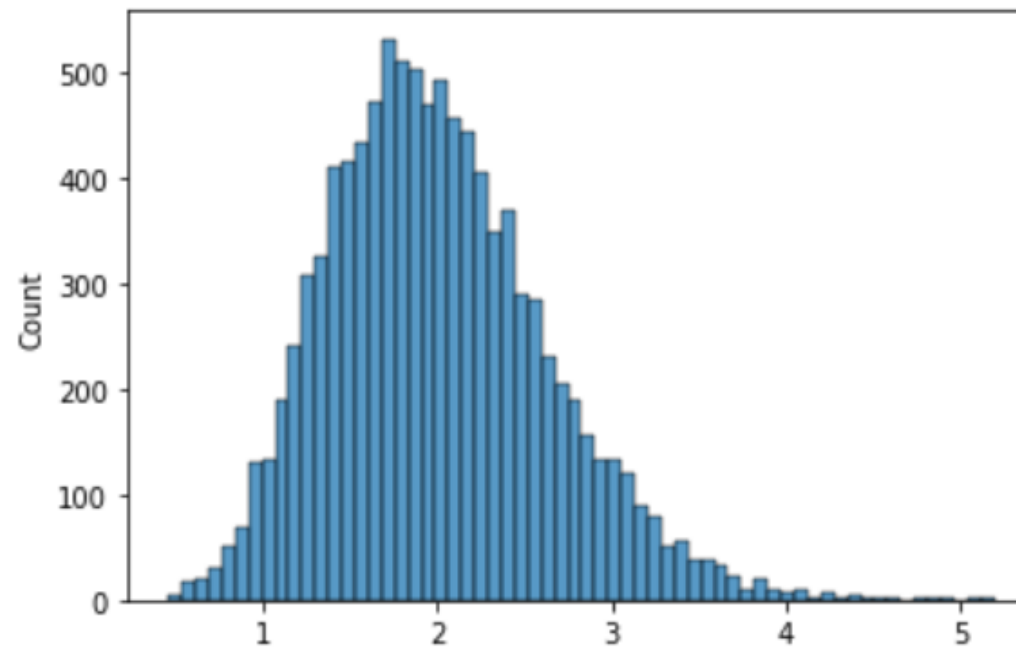
1. How fast can we expect this to work?
2. Is it equally fast for every density? If so, when can we expect it to be really slow?

Exercise!

# Convergence in distribution

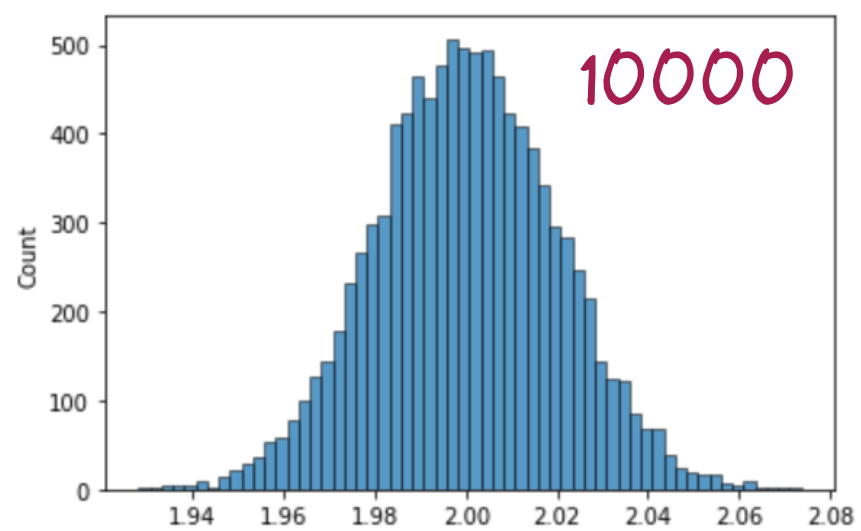
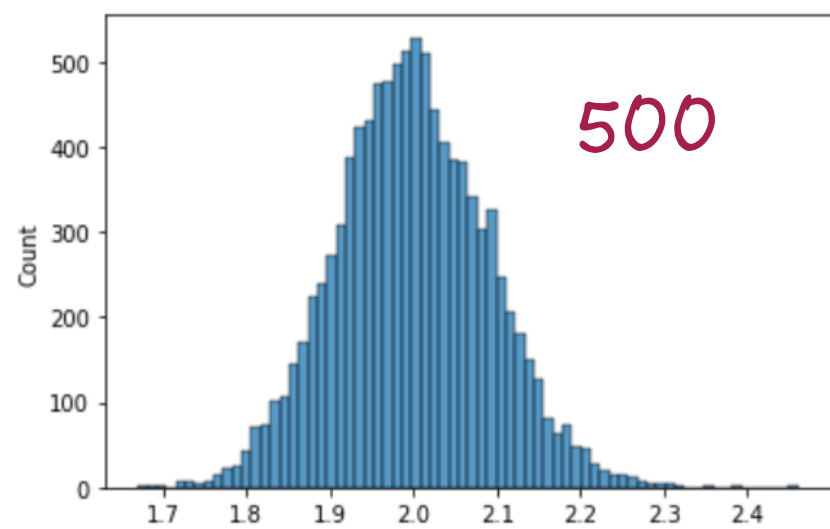
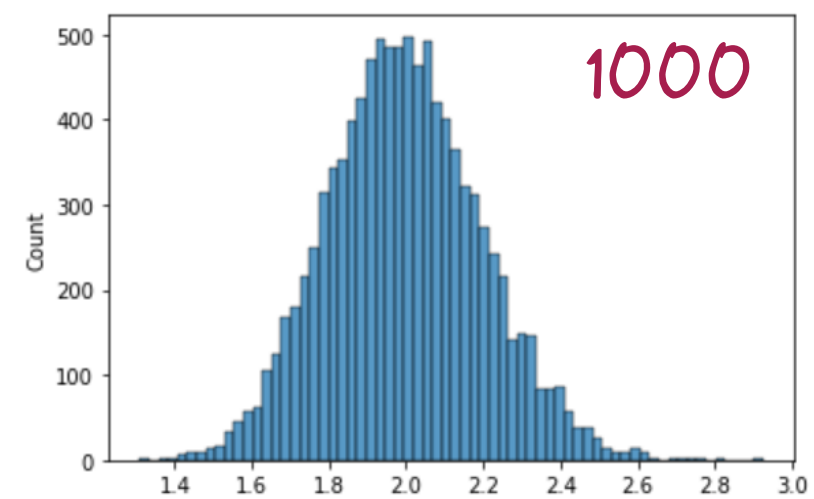
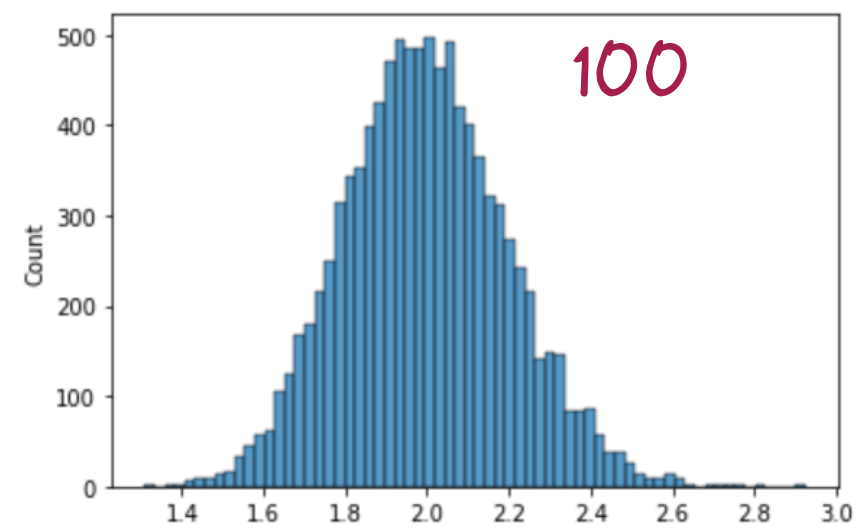
```
1 import numpy as np
2 import seaborn as sns
3 from scipy import stats
4
5 rng = np.random.default_rng(313)
6 x = rng.exponential(2, [10, 10000])
7 stats.expon(0, 3).mean() # Location 0 and scale 3.
8 y = x.mean(axis = 0)
9 # What is x.mean(axis = 0).size ?
10
11 sns.histplot(y)
```

Let's try  
different values than  
10!



# Convergence in distribution

```
7 def plotter(n, n_reps, rng):  
8     x = rng.exponential(2, [n, n_reps])  
9     stats.expon(0, 3).mean()  
10    y = x.mean(axis = 0)  
11    sns.histplot(y)
```



```
15 def calc_std(n, n_reps, rng):  
16     x = rng.exponential(2, [n, n_reps])  
17     stats.expon(0, 3).mean()  
18     return x.mean(axis = 0).std()
```

```
upper = np.array([calc_sd(i, 10000, rng) for i in [100, 400, 900, 1600, 10000]])  
lower = np.array([100, 400, 900, 1600, 10000]) ** 0.5  
upper * lower
```

```
array([2.00216082, 2.00890198, 1.99148659, 2.01103019, 2.00543746])
```

This means the standard deviations multiplied by the root of n stabilizes.

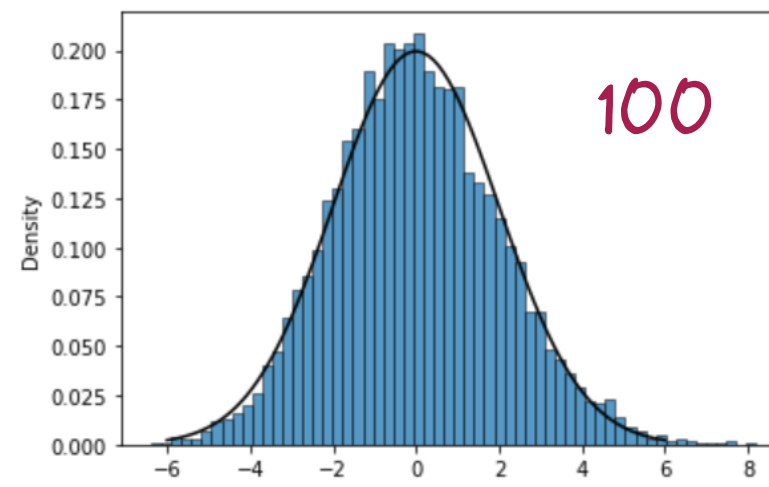
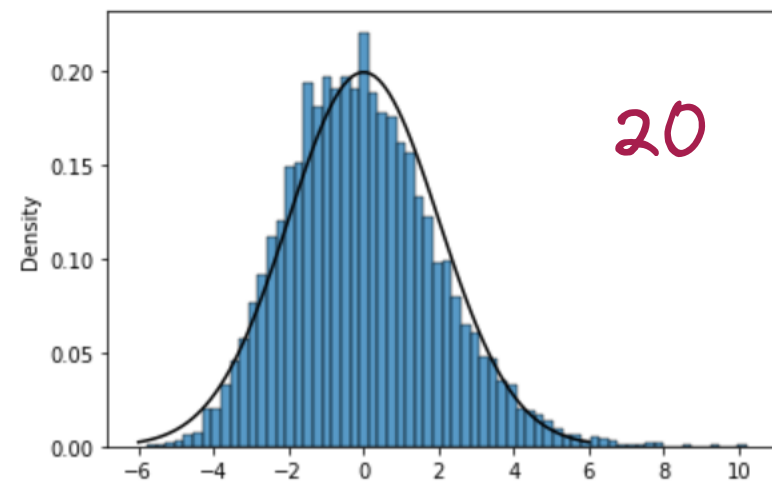
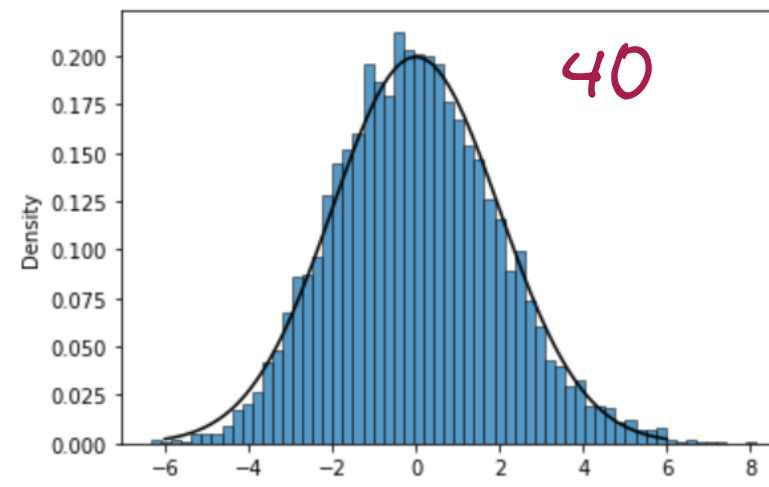
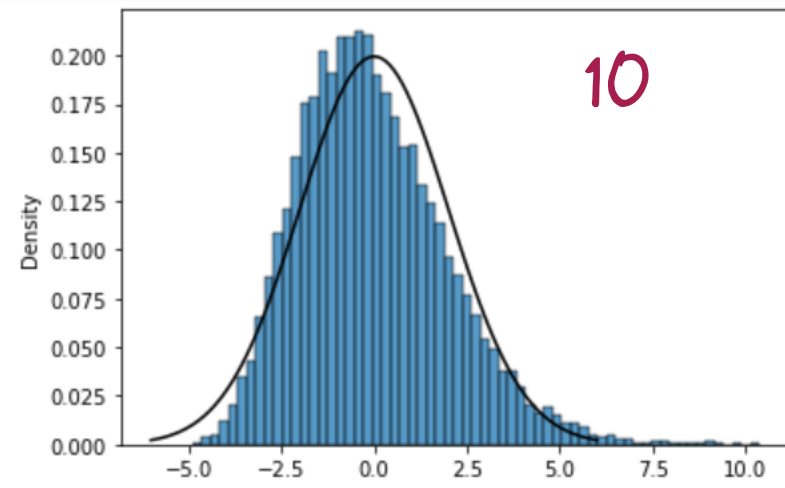


So let's rescale everything by the root of  $n$ !

```
1 import matplotlib.pyplot as plt
2
3 def plotter(n, n_reps, rng):
4     x = rng.exponential(2, [n, n_reps])
5     stats.expon(0, 3).mean()
6     y = x.mean(axis = 0)
7     sns.histplot((y - 2) * n ** 0.5, stat = "density")
8     z = np.linspace(-6, 6, 100)
9     plt.plot(z, stats.norm(0, 2).pdf(z), color = "black")
10
11 for n in [10, 20, 50, 100]:
12     plt.figure()
13     plotter(n, 10000, rng)
```

Rescaling and  
moving the mean to  
0.

Adding a normal  
PDF.



# Central limit theorem (CLT)

---

Let  $X_1, X_2, X_3, \dots$  be iid random variables with expectation  $E(X_1) = \mu$  and finite variance  $\text{Var}(X_1) = \sigma^2$ . Let  $\Phi$  denote the CDF of the standard normal distribution. Then

$$P\left(\sqrt{n}\frac{(\bar{X}_n - \mu)}{\sigma} \leq x\right) = \Phi(x).$$

In other words,

$$\sqrt{n}\frac{(\bar{X}_n - \mu)}{\sigma} \xrightarrow{d} N(0, 1),$$

where  $\xrightarrow{d}$  denotes convergence in distribution.

# Convergence in distribution

---

The sequence of random variables  $X_n$ , with cumulative distribution functions  $F_n$ , converge in distribution to a random variable with distribution function  $F$  if  $F_n(x) - F(x) \rightarrow 0$  as  $n \rightarrow \infty$  for all  $x$ .

# Convergence in distribution

---

You can also use density functions for the same purpose: You can (and should) think of convergence in distribution as convergence of histograms.

# Asymptotic normality

---

An estimator is *asymptotically normal* if

$$\sqrt{n}(\hat{\theta} - \theta) \xrightarrow{d} N(0, \sigma^2),$$

where  $\sigma^2$  is the *asymptotic variance* of the estimator.

If  $\hat{\theta}$  is a reasonable estimator of some parameter  $\theta$  it is almost always asymptotically normal.

Asymptotic normality is not just found in the central limit theorem, but everywhere!

The end

