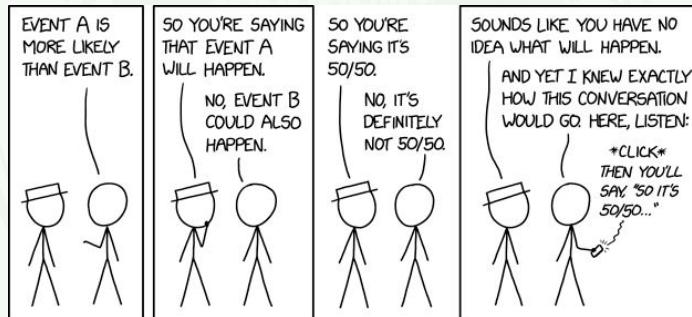


Statistics and Probability



David J Butts

Computational Mathematics Science and Engineering
Michigan State University





Stats



How much do summary statistics actually tell us?

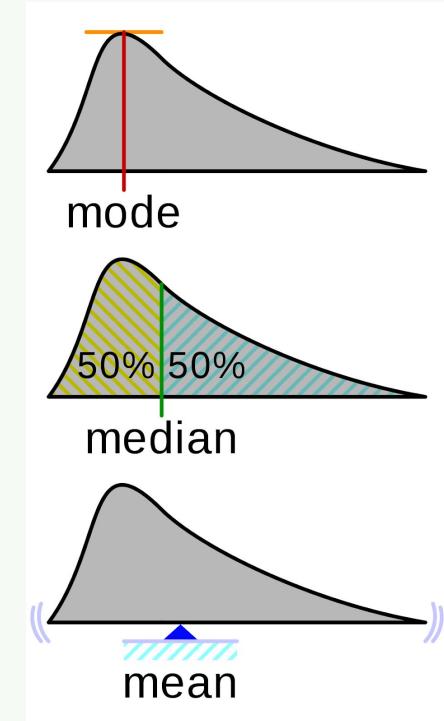
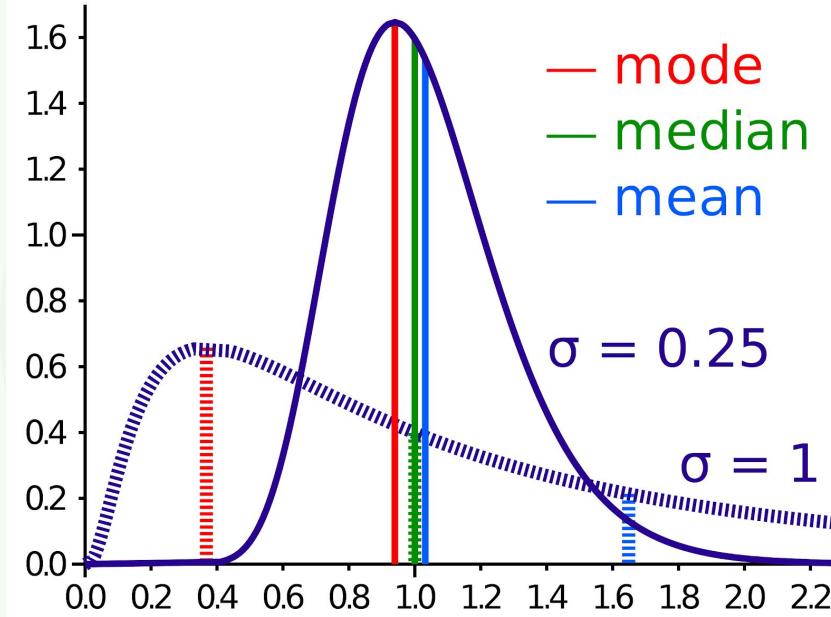
```
X Mean: 54.2659224  
Y Mean: 47.8313999  
X SD   : 16.7649829  
Y SD   : 26.9342120  
Corr.  : -0.0642526
```

For non-normally distributed data, summary statistics might be misleading



The average can *mean* different things

$$\begin{aligned}\bar{x} &= \langle x \rangle \\ &= E[x] \\ &= \mu \\ &= \frac{1}{n} \sum_{i=1}^n x_i\end{aligned}$$



Covariance

$$\begin{aligned} \text{Cov}(X, Y) &= E[(X - E[X])(Y - E[Y])] \\ &= E[XY - YE[X] - XE[Y] + E[X]E[Y]] \\ &= E[XY] - E[Y]E[X] - E[X]E[Y] + E[X]E[Y] \\ &= E[XY] - E[X]E[Y] \end{aligned}$$

Covariance measures how much two things vary with respect to each other.

We will see later how to quantify how to quantify the amount of variance



Variance

$$\begin{aligned}Var[X] &= Cov[X, X] \\&= \sigma_X^2 \\&= E[(X - E[X])^2] \\&= E[X^2] - E[X]^2\end{aligned}$$

$$Correlation = \frac{Cov(x, y)}{\sigma_x * \sigma_y}$$

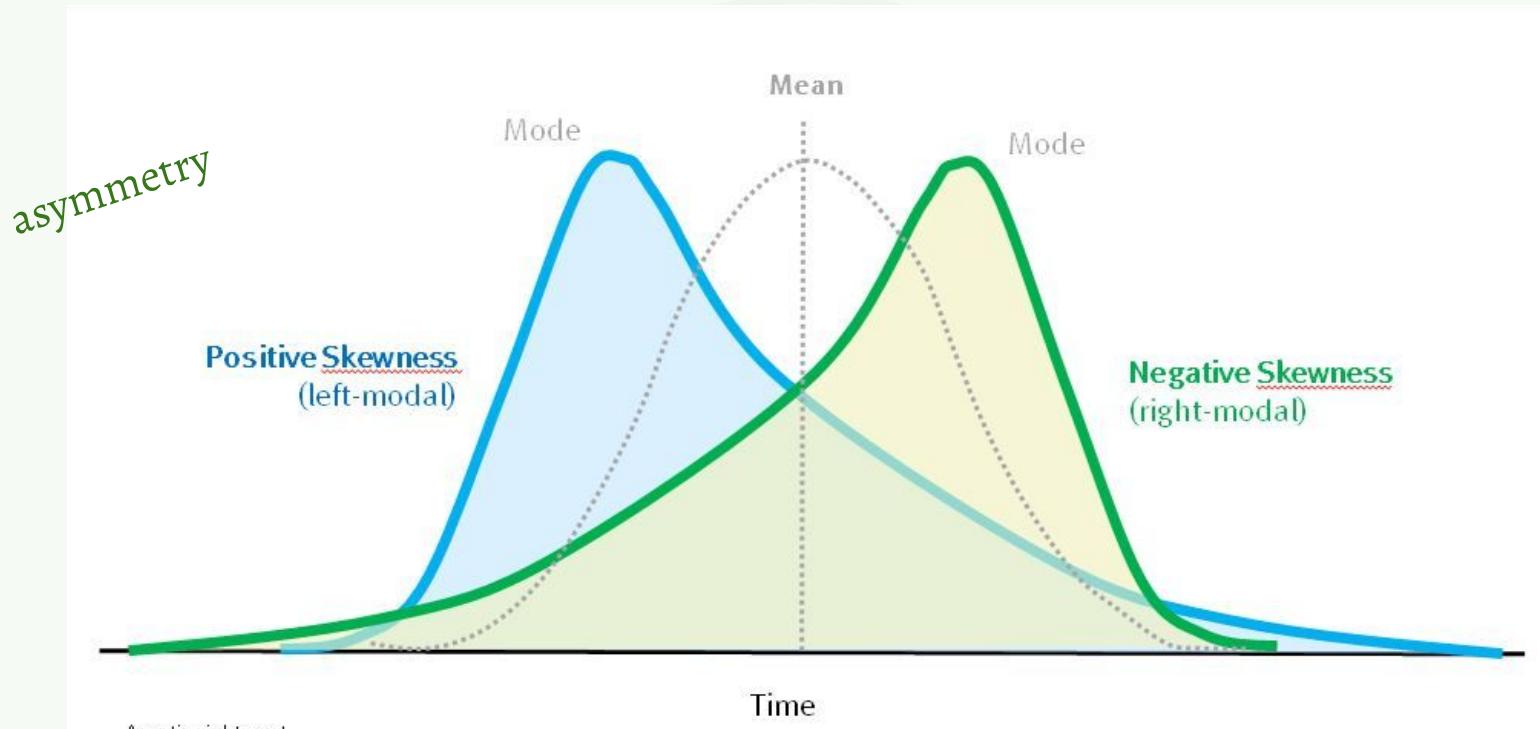
Variance measures, obviously, how much something varies.

An example from physics is this: if you knew the velocity distribution of the particles on our classroom, the variance in that distribution would be proportional to the temperature.

Quantities like this are sometimes called “autocorrelation functions”.

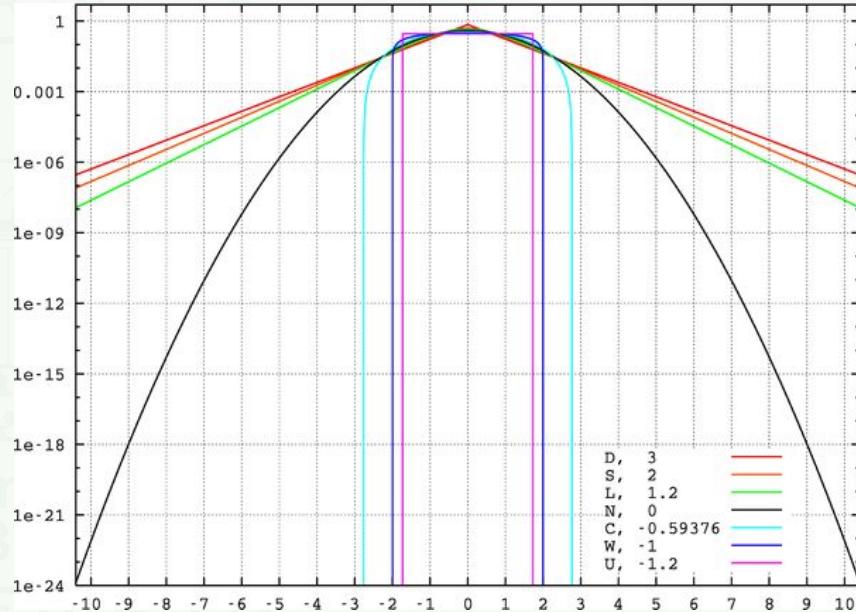
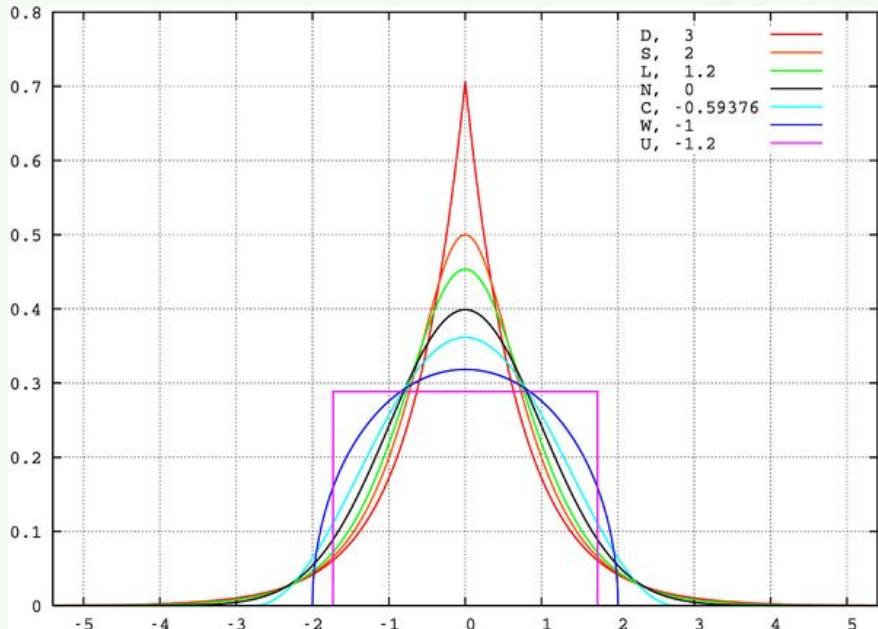


Skewness



Summary statistics: kurtosis

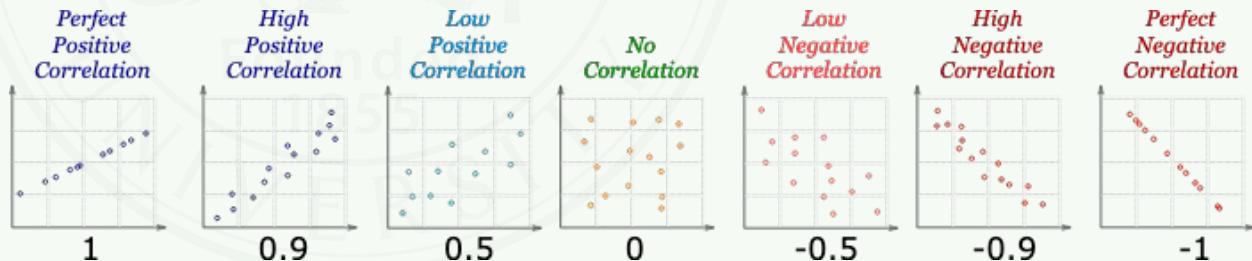
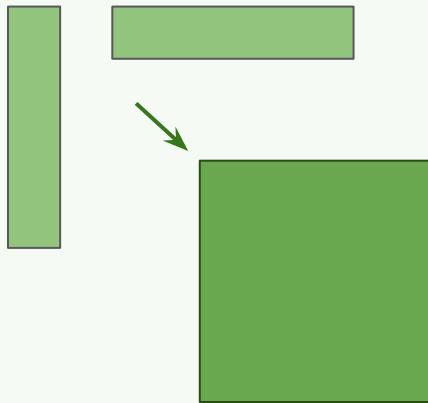
tailedness



Covariance Matrix

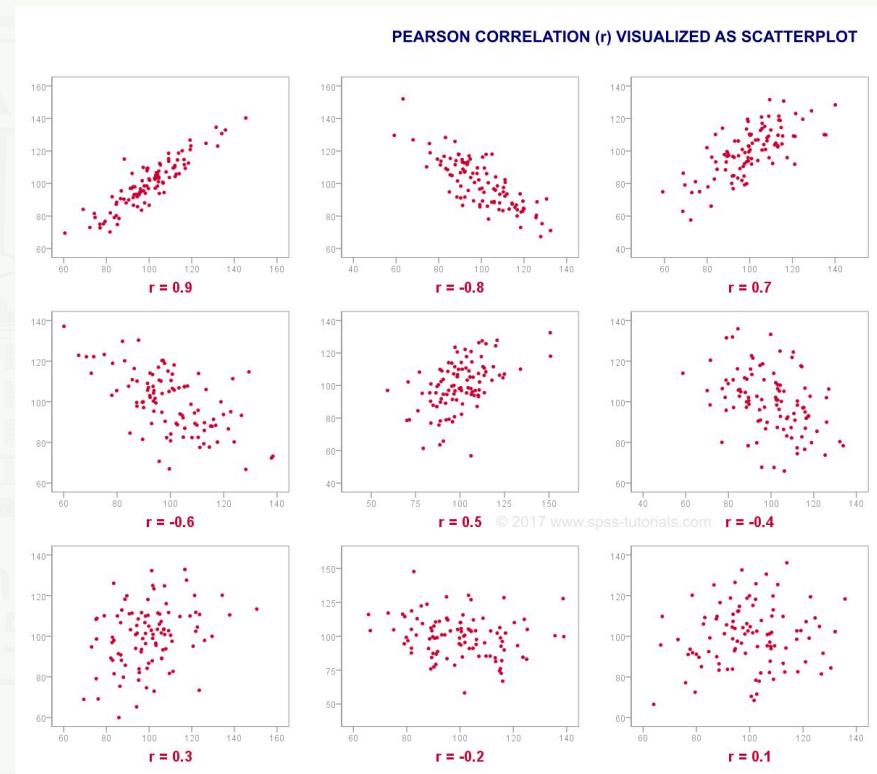
Covariance matrix: X is a column vector

$$E[(X - \mu)(X - \mu)^T] = \begin{pmatrix} E[(X_1 - \mu_1)^2] & E[(X_1 - \mu_1)(X_2 - \mu_2)] & \dots \\ E[(X_1 - \mu_1)(X_2 - \mu_2)] & E[(X_2 - \mu_2)^2] & \dots \\ \vdots & & \\ E[(X_n - \mu_n)(X_1 - \mu_1)] & E[(X_n - \mu_n)(X_2 - \mu_2)] & \dots \end{pmatrix}$$

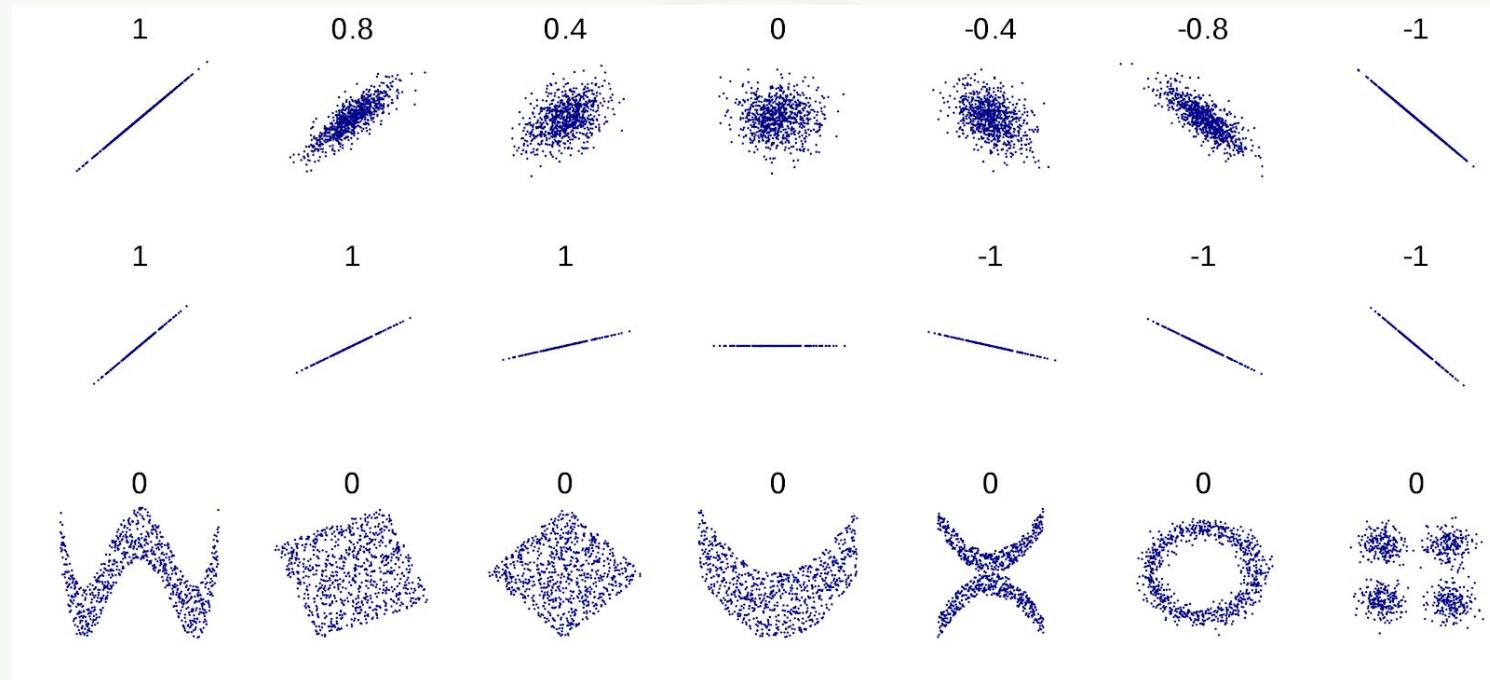


Pearson correlation

$$\rho_{X,Y} = \frac{Cov(X, Y)}{\sqrt{Cov(X, X)Cov(Y, Y)}}$$



Pearson does not reflect shape

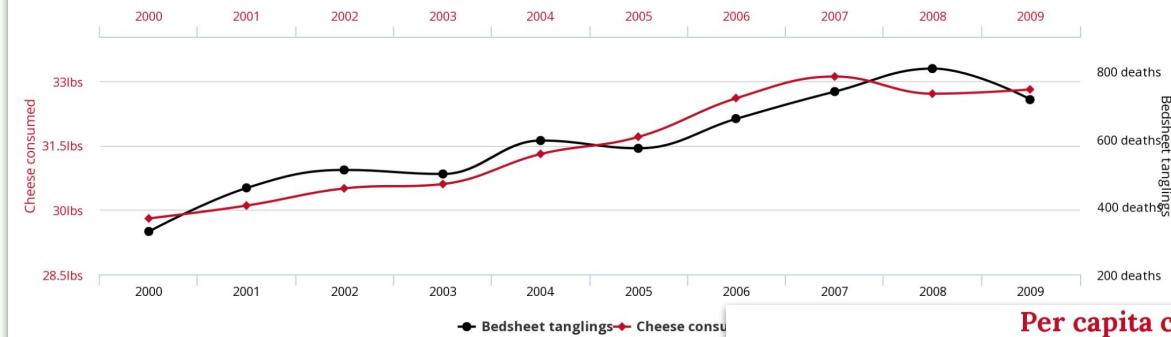


Should we risk eating cheese?

Per capita cheese consumption

correlates with

Number of people who died by becoming tangled in their bedsheets



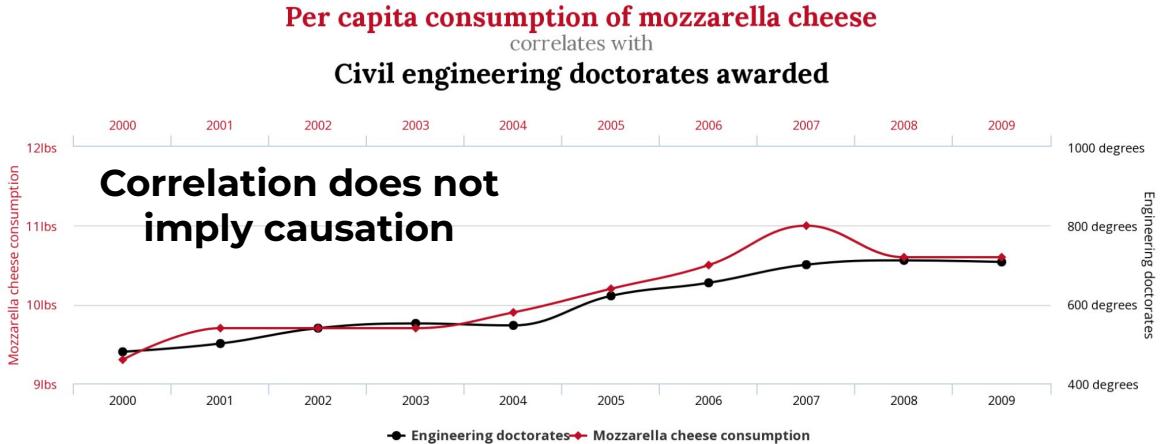
Does cheese kill?

Or, does cheese make us smarter?

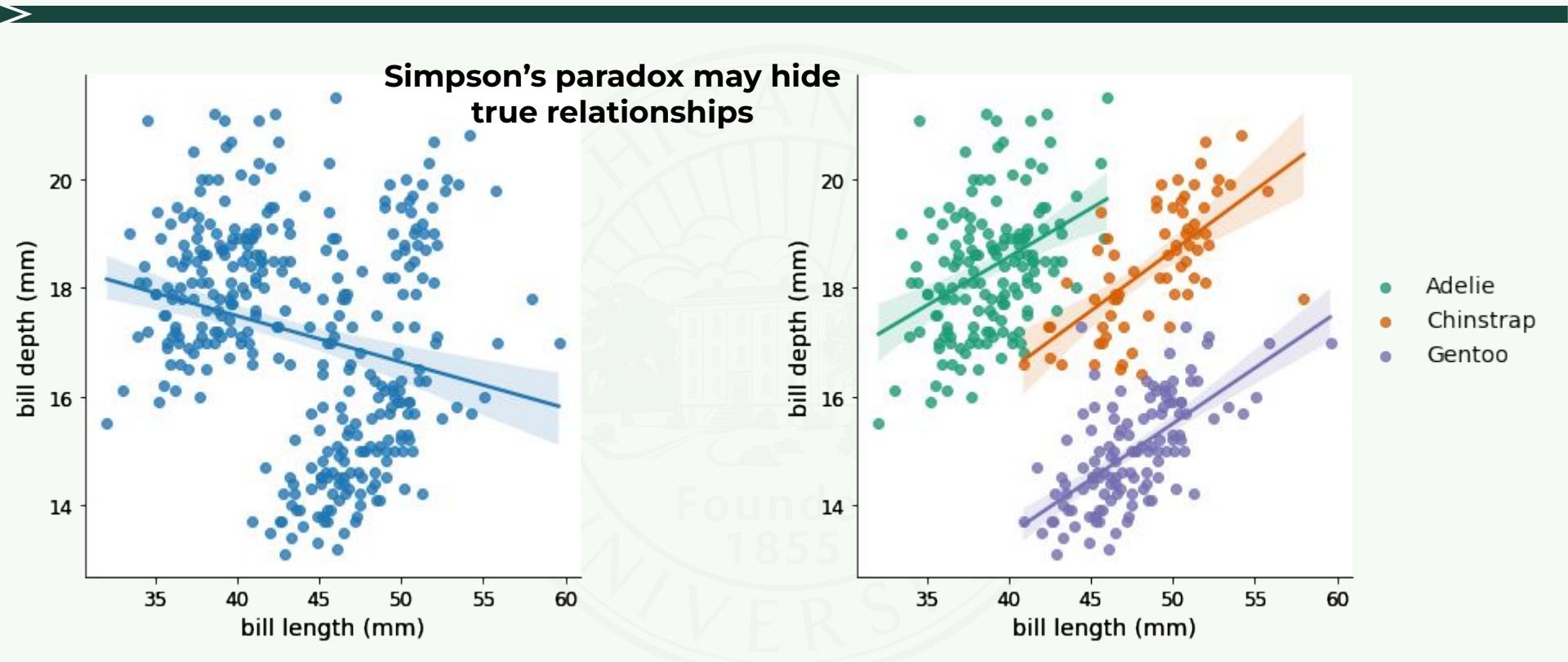
<https://www.tylervigen.com/spurious-correlations>



Correlation does not imply causation



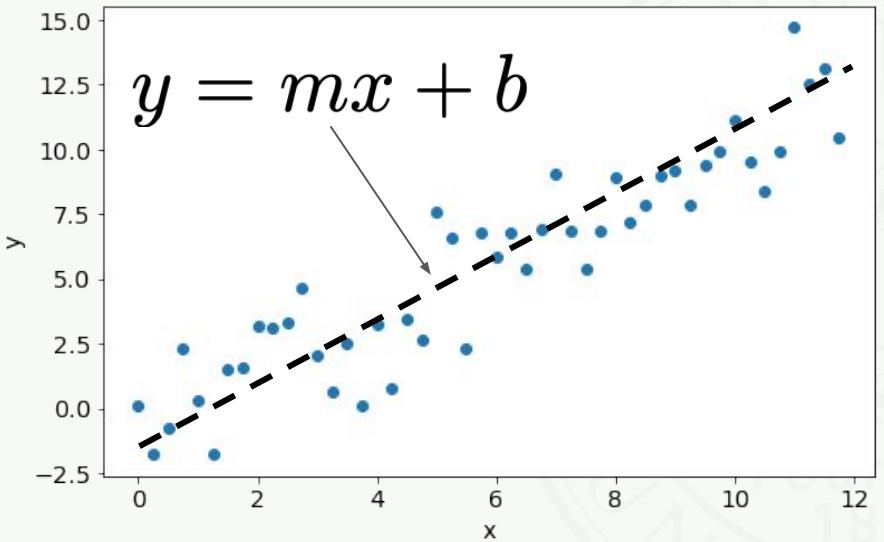
Am I picking the best fit to my data?



Some Examples



Fitting a line



$$L(m, b) = \sum_i^N (y_i - mx_i - b)^2$$

$$\frac{\partial L}{\partial m} = -2 \sum_i (y_i - mx_i - b)x_i = 0$$

$$\frac{\partial L}{\partial b} = -2 \sum_i (y_i - mx_i - b) = 0$$



Minimizing with respect to b

$$\frac{\partial L}{\partial b} = 0 = -2 \sum_i (y_i - mx_i - b)$$

$$0 = \sum_i y_i - m \sum_i x_i - \sum_i b$$

$$0 = N\bar{y} - mN\bar{x} - Nb$$

$$b = \bar{y} - m\bar{x}$$



Minimizing with respect to m

$$\frac{\partial L}{\partial m} = 0 = -2 \sum_i (y_i - mx_i - b)x_i$$

$$0 = \sum_i y_i x_i - m \sum_i x_i x_i - b \sum_i x_i$$

$$0 = \sum_i y_i x_i - m \sum_i x_i x_i - bN\bar{x}$$

$$0 = \sum_i y_i x_i - m \sum_i x_i x_i - N\bar{x}\bar{y} + mN\bar{x}\bar{x}$$

$$0 = \sum_i y_i x_i - N\bar{x}\bar{y} - m \left(\sum_i x_i x_i - N\bar{x}\bar{x} \right)$$

$$m = \frac{\sum_i y_i x_i - N\bar{x}\bar{y}}{\sum_i x_i x_i - N\bar{x}\bar{x}}$$

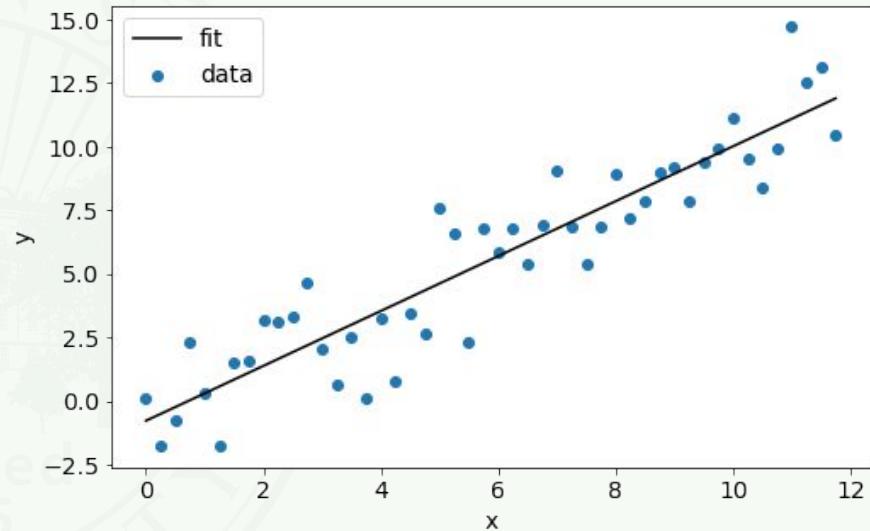
$$m = \frac{\frac{1}{N} \sum_i y_i x_i - \bar{x}\bar{y}}{\frac{1}{N} \sum_i x_i x_i - \bar{x}\bar{x}}$$

$$m = \frac{cov(x, y)}{var(x)}$$



The least squares fit is related to statistical fits

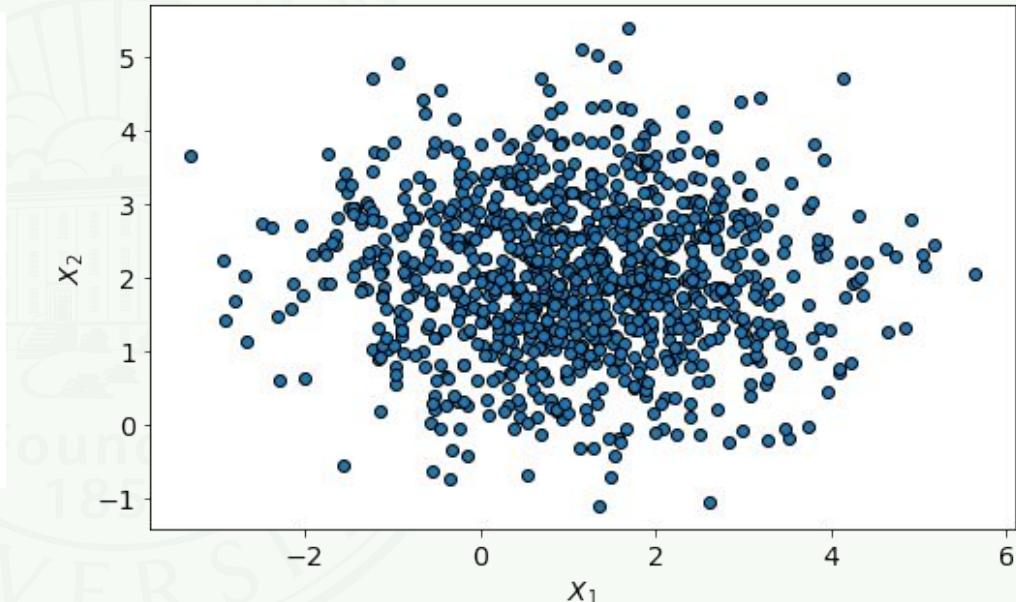
$$\begin{aligned}y &= \frac{\text{cov}(x, y)}{\text{var}(x)}x + \bar{y} - \frac{\text{cov}(x, y)}{\text{var}(x)}\bar{x} \\&= \frac{\text{cov}(x, y)}{\text{var}(x)}(x - \bar{x}) + \bar{y}\end{aligned}$$



Multidimensional data example

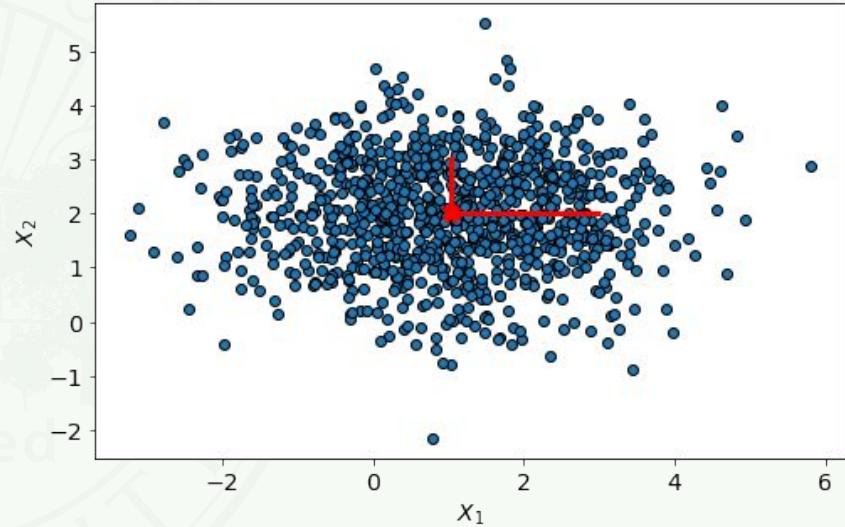
```
X =  
[[ 0.38849788  5.08330752]  
 [ 0.81034528  2.04741571]  
 [-1.56082481  1.31164164]  
 ...  
 [ 2.355576    0.53936631]  
 [ 1.52170224  2.02778523]  
 [ 0.43006137  1.65176376]]
```

1,000 by 2 numbers



```
print(f'E[X] = {np.mean(X, axis=0)}')
print(f'Var[X] = {np.var(X, axis=0)}')
print(f'Cov[X] =\n {np.cov(X.T)}')
```

```
E[X] = [1.02687839 1.98999062]
Var[X] = [2.02788973 1.0213675 ]
Cov[X] =
[[ 2.02991965 -0.01297718]
 [-0.01297718  1.02238989]]
```



Scipy.stats has many statistical functions

1. Know that there are many summary statistics for data.
2. Best to choose the one most useful to your particular problem. Their use is largely context dependent.
3. Even better, allow the context of the problem to reveal to you which is better.

Statistical functions

Several of these functions have a similar version in `scipy.stats.mstats` which work for masked arrays.

<code>describe(a[, axis])</code>	Computes the O'Brien transform on input data (any number of arrays).
<code>gmean(a[, axis, dtype])</code>	The signal-to-noise ratio of the input data.
<code>hmean(a[, axis, dtype])</code>	Bayesian confidence intervals for the mean, var, and std.
<code>kurtosis(a[, axis, fisher, bias])</code>	Calculates the standard error of the mean (or standard error of measurement) of t
<code>kurtosistest(a[, axis])</code>	Calculates the relative z-scores.
<code>mode(a[, axis])</code>	Calculates the z score of each value in the sample, relative to the sample mean a
<code>moment(a[, moment, axis])</code>	Iterative sigma-clipping of array elements.
<code>normaltest(a[, axis])</code>	Clip array to a given value.
<code>skew(a[, axis, bias])</code>	Slices off a proportion of items from both ends of an array.
<code>skewtest(a[, axis])</code>	Slices off a proportion of items from ONE end of the passed array distribu
<code>tmean(a[, limits, inclusive])</code>	Performs a 1-way ANOVA.
<code>tvar(a[, limits, inclusive])</code>	Calculates a Pearson correlation coefficient and the p-value for testing non-correlation.
<code>tmin(a[, lowerlimit, axis, inclusive])</code>	Calculates a Spearman rank-order correlation coefficient and the p-value to test for nor
<code>tmax(a[, upperlimit, axis, inclusive])</code>	Calculates a point biserial correlation coefficient and the associated p-value.
<code>tstd(a[, limits, inclusive])</code>	Calculates Kendall's tau, a correlation measure for ordinal data.
<code>tsem(a[, limits, inclusive])</code>	Calculate a regression line
<code>nanmean(x[, axis])</code>	Calculates the T-test for the mean of ONE group of scores.
<code>nanstd(x[, axis, bias])</code>	Calculates the T-test for the means of TWO INDEPENDENT samples
<code>nanmedian(x[, axis])</code>	Calculates the T-test on TWO RELATED samples of scores, a and b.
<code>variation(a[, axis])</code>	Perform the Kolmogorov-Smirnov test for goodness of fit.
<code>cumfreq(a[, numbins, defaultreallimits, histtype='bar', **kwargs])</code>	Calculates a one-way chi square test.
<code>histogram2(a, bins)</code>	Cressie-Read power divergence statistic and goodness of fit test.
<code>histogram(a[, numbins, defaultlimits, density=False, **kwargs])</code>	Computes the Kolmogorov-Smirnov statistic on 2 samples.
<code>itemfreq(a)</code>	Computes the Mann-Whitney rank test on samples x and y.
<code>percentileofscore(a, score[, kind])</code>	Tie correction factor for ties in the Mann-Whitney U and Kruskal-Wallis
<code>scoreatpercentile(a, per[, limit, ...])</code>	Assign ranks to data, dealing with ties appropriately.
<code>relfreq(a[, numbins, defaultreallimits, width])</code>	Compute the Wilcoxon rank-sum statistic for two samples.
<code>binned_statistic(x, values[, statistic, bins, range])</code>	Calculate the Wilcoxon signed-rank test.
<code>binned_statistic_2d(x, y, values[, bins, range])</code>	Compute the Kruskal-Wallis H-test for independent samples
<code>binned_statistic_dd(sample, values[, bins, range])</code>	Computes the Friedman test for repeated measurements
<code>ansari(x, y)</code>	Perform the Ansari-Bradley test for equal scale parameters
<code>bartlett(*args)</code>	Perform Bartlett's test for equal variances



Many Python libraries include stats capabilities

SciPy.org		Scipy.org	Scipy.org
SciPy.org		Scipy.org	Scipy.org
S	SciPy.org	E	ENTHOUGHT
SciPy.org	Docs	NumPy v1.13 Manual	NumPy Reference
Routines			
Statistics		Statistics	
Order statistics		Order statistics	
amin (a[, axis, out, keepdims])	Return the minimum of an array or minimum along an axis.	amin (a[, axis, out, keepdims, initial, where])	Return the minimum of an array or minimum along an axis.
amax (a[, axis, out, keepdims])	Return the maximum of an array or maximum along an axis.	amax (a[, axis, out, keepdims, initial, where])	Return the maximum of an array or maximum along an axis.
nanmin (a[, axis, out, keepdims])	Return minimum of an array or minimum along an axis, ignoring any NaNs.	nanmin (a[, axis, out, keepdims])	Return minimum of an array or minimum along an axis, ignoring any NaNs.
nanmax (a[, axis, out, keepdims])	Return the maximum of an array or maximum along an axis, ignoring any NaNs.	nanmax (a[, axis, out, keepdims])	Return the maximum of an array or maximum along an axis, ignoring any NaNs.
ptp (a[, axis, out])	Range of values (maximum - minimum).	ptp (a[, axis, out, keepdims])	Range of values (maximum - minimum) along an axis.
percentile (a, q[, axis, out, ...])	Compute the q-th percentile of the data along the specified axis.	percentile (a, q[, axis, out, ...])	Compute the q-th percentile of the data along the specified axis.
nanpercentile (a, q[, axis, out, ...])	Compute the q-th percentile of the data along the specified axis, while ignoring nan values.	nanpercentile (a, q[, axis, out, ...])	Compute the q-th percentile of the data along the specified axis, while ignoring nan values.
quantile (a, q[, axis, out, overwrite_input, ...])	Compute the q-th quantile of the data along the specified axis.	quantile (a, q[, axis, out, ...])	Compute the q-th quantile of the data along the specified axis.
nanquantile (a, q[, axis, out, ...])	Compute the q-th quantile of the data along the specified axis, while ignoring nan values.		
Averages and variances		Averages and variances	
median (a[, axis, out, overwrite_input, keepdims])	Compute the median along the specified axis.	median (a[, axis, out, overwrite_input, keepdims])	Compute the median along the specified axis.
average (a[, axis, weights, returned])	Compute the weighted average along the specified axis.	average (a[, axis, weights, returned])	Compute the weighted average along the specified axis.
mean (a[, axis, dtype, out, keepdims])	Compute the arithmetic mean along the specified axis.	mean (a[, axis, dtype, out, keepdims])	Compute the arithmetic mean along the specified axis.
std (a[, axis, dtype, out, ddof, keepdims])	Compute the standard deviation along the specified axis.	std (a[, axis, dtype, out, ddof, keepdims])	Compute the standard deviation along the specified axis.
var (a[, axis, dtype, out, ddof, keepdims])	Compute the variance along the specified axis.	var (a[, axis, dtype, out, ddof, keepdims])	Compute the variance along the specified axis.
nanmedian (a[, axis, out, overwrite_input, ...])	Compute the median along the specified axis.	nanmedian (a[, axis, out, overwrite_input, ...])	Compute the median along the specified axis, while ignoring NaNs.
nanmean (a[, axis, dtype, out, keepdims])	Compute the arithmetic mean along the specified axis, ignoring NaNs.	nanmean (a[, axis, dtype, out, keepdims])	Compute the arithmetic mean along the specified axis, ignoring NaNs.
nanstd (a[, axis, dtype, out, ddof, keepdims])	Compute the standard deviation along the specified axis, while ignoring NaNs.	nanstd (a[, axis, dtype, out, ddof, keepdims])	Compute the standard deviation along the specified axis, while ignoring NaNs.
nanvar (a[, axis, dtype, out, ddof, keepdims])	Compute the variance along the specified axis, while ignoring NaNs.	nanvar (a[, axis, dtype, out, ddof, keepdims])	Compute the variance along the specified axis, while ignoring NaNs.
Correlating		Correlating	
corccoeff (x[, y, rowvar, bias, ddof])	Return Pearson product-moment correlation coefficients.	corccoeff (x[, y, rowvar, bias, ddof])	Return Pearson product-moment correlation coefficients.
correlate (a, v[, mode])	Cross-correlation of two 1-dimensional sequences.	correlate (a, v[, mode])	Cross-correlation of two 1-dimensional sequences.
cov (m[, y, rowvar, bias, ddof, fweights, ...])	Estimate a covariance matrix, given m arrays of observations.		
Histograms			
histogram (a[, bins, range, normed, weights, ...])	Compute the histogram of a.		
histogram2d (x, y[, bins, range, normed, weights])	Compute the bi-dimensional histogram of 2D data.		
histogramdd (sample[, bins, range, normed, ...])	Compute the multidimensional histogram of sample.		
bincount (x[, weights, minlength])	Count number of occurrences of each value in array of non-negative ints.		
digitize (x, bins[, right])	Return the indices of the bins to which each value in input array belongs.		

models's Documentation

n vides classes and functions for the estimation of many different statistical tests, and statistical data exploration. An extensive list of n ator. The results are tested against existing statistical packages to s released under the open source Modified BSD (3-clause) license. mmodels.org.

describe

None, include=None, exclude=None)

summarize the central tendency, dispersion and shape of a dataset's

series, as well as DataFrame column sets of mixed data types. The output will refer to the notes below for more detail.

-like of numbers, optional
iles to include in the output. All should fall between 0 and 1. The default
.75], which returns the 25th, 50th, and 75th percentiles.



Probability



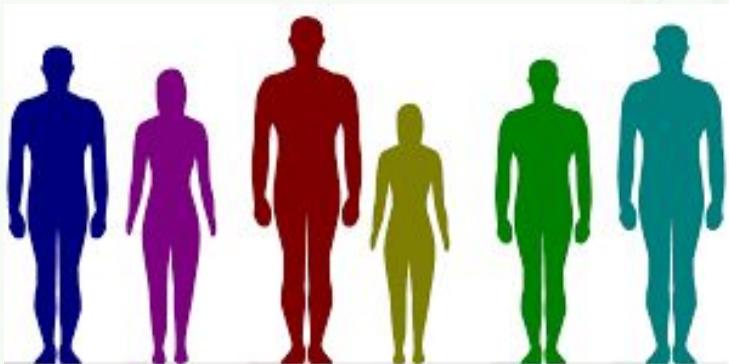
Example: flipping a fair coin

If I flip a coin, how likely is it
to land head or tails?



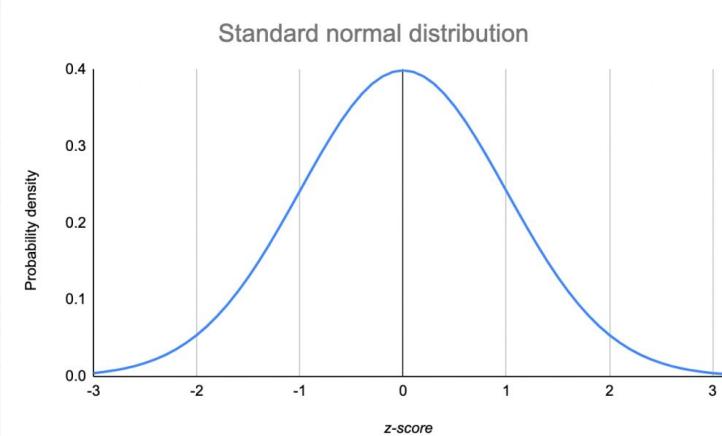
$$p(\text{heads}) = p(\text{tails}) = \frac{1}{2}$$

How are our heights distributed?



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

We will generalize this to more dimensions later



Properties of probability mass/density

Probability mass/density
are only zero when
something is impossible.

$$p(x) \geq 0$$

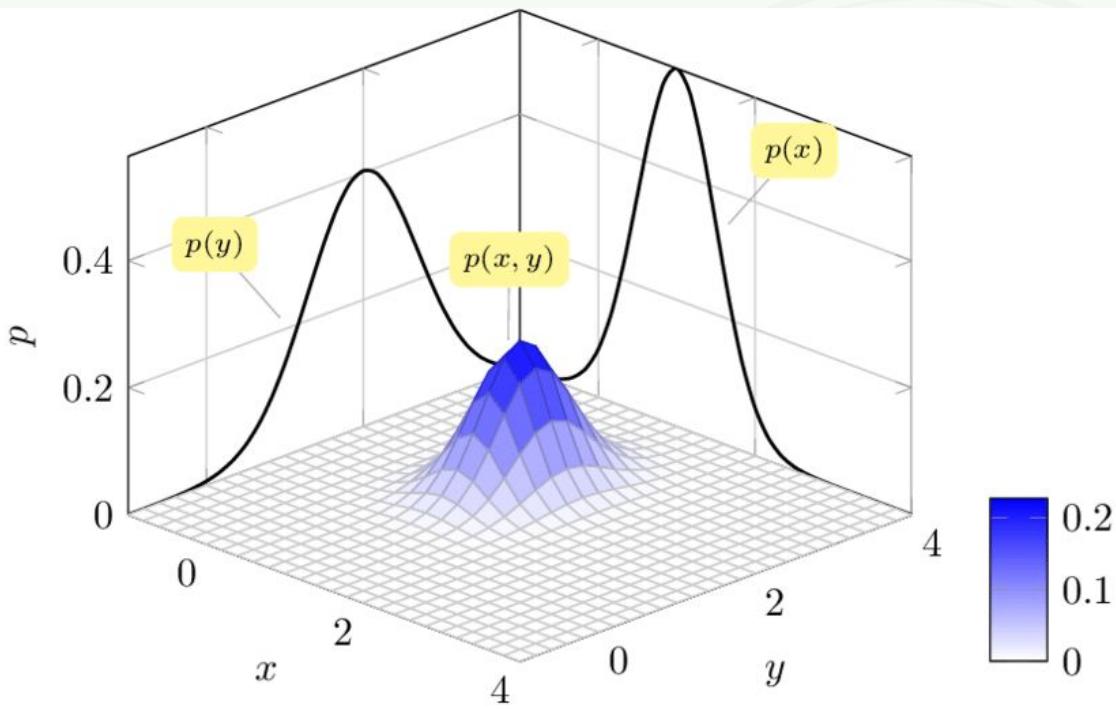
$$f(x) \geq 0$$

Adding (integrating) their
values sum to 1. There is 100%
of something happening.

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



Probabilities can be multidimensional



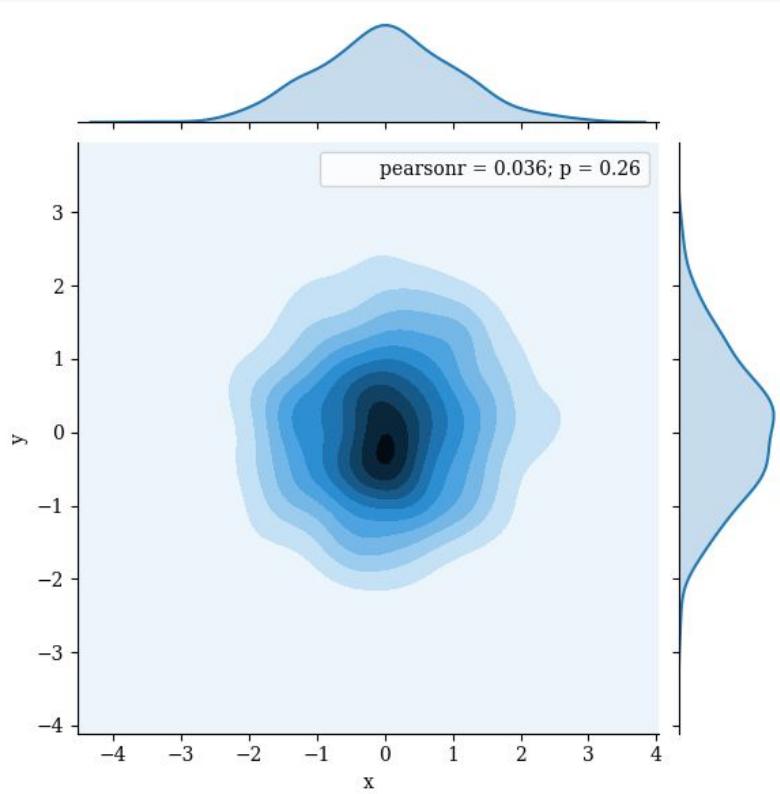
Marginals

$$p(y) = \int p(x, y) dx$$

$$p(x) = \int p(x, y) dy$$



Top-down view of a joint distribution



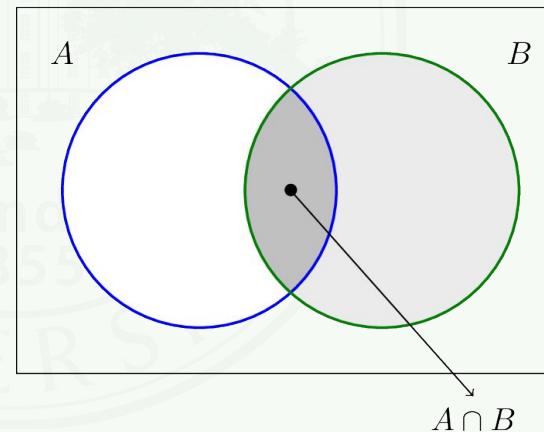
Probabilities can be conditional

The bar means
“given that”

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

What is the probability
of A happening, given
that B happened?

Probability of A and B
happening.
—————
Probability of B
happening.



Independent events



$$p(\text{heads}) = p(\text{tails}) = \frac{1}{2}$$

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

What is the probability that I flip a heads, given I just flipped a heads?

$$\begin{aligned} P(\text{heads}|\text{heads}) &= \frac{P(\text{heads} \cap \text{heads})}{P(\text{heads})} \\ &= \frac{1/2 \cdot 1/2}{1/2} = \frac{1}{2} \end{aligned}$$



Bayes' theorem switches the order of conditionals

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

$$P(\text{smoke}) = .1$$

If we see smoke, what's the probability of danger?

$$P(\text{danger}) = .01$$

$$P(\text{smoke}|\text{danger}) = .9$$

$$P(\text{danger}|\text{smoke}) = \frac{.9 \cdot .01}{.1} = .09$$

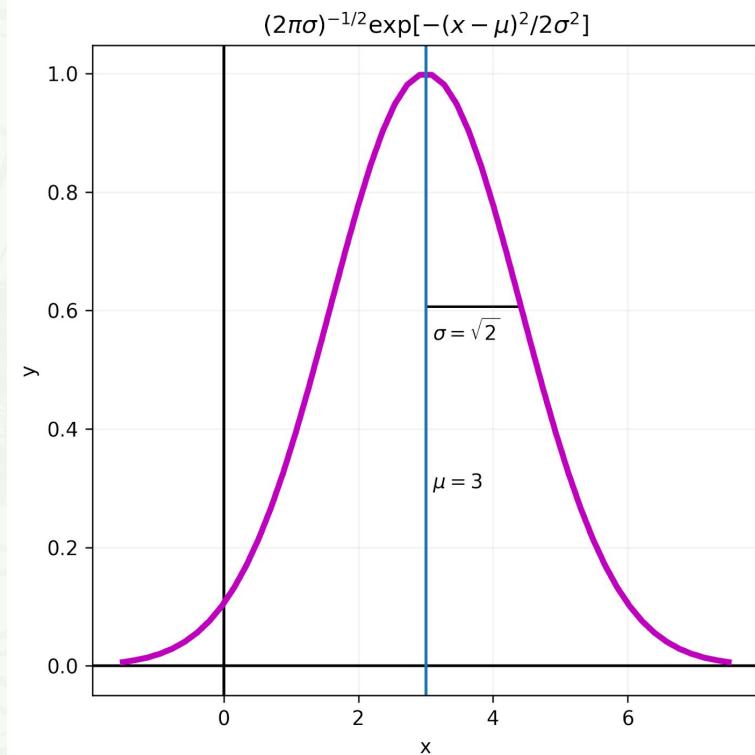


Properties of Gaussians

$$\mathcal{N}(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2}(x - \mu)^2/\sigma^2\right]$$

A quick summary:

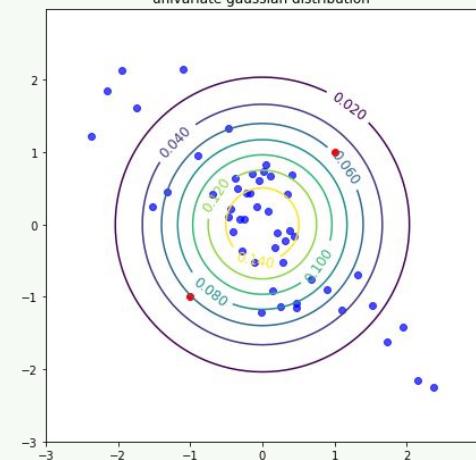
- Very commonly used in ML.
- Also called the “normal” distribution.
- Uses only the mean and the variance.
- Has many very nice mathematical properties:
 - Easy to generalize to many dimensions.
 - Integrals of Gaussians yield (lower-dimensional) Gaussians.



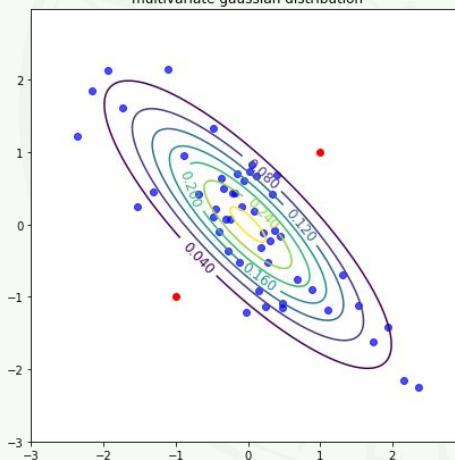
Multivariate Gaussians

$$G(x_1, \dots, x_n) = \frac{1}{\sqrt{(2\pi)^n \det(\Sigma)}} \exp \left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right)$$

univariate gaussian distribution



multivariate gaussian distribution



mean

covariance matrix

$$\Sigma_{ij} = E[(X_i - \mu_i)(X_j - \mu_j)]$$

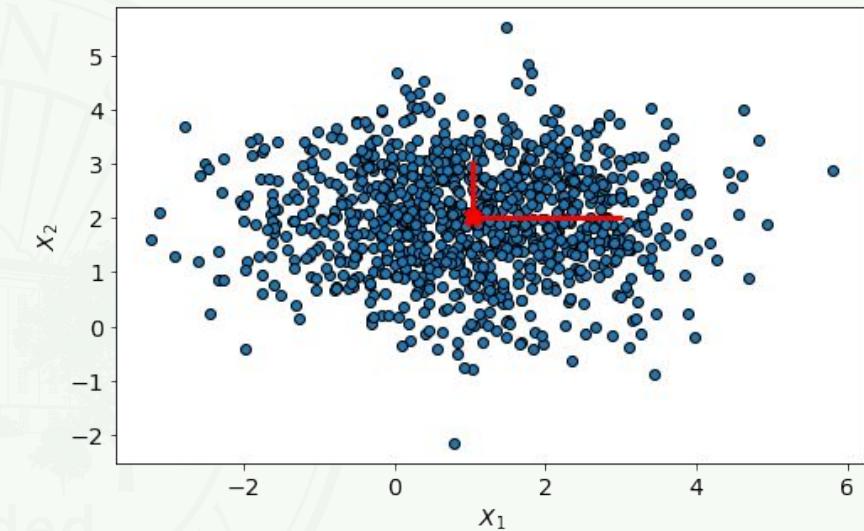
Can be used to describe patterns
in the data and/or describe noise
in the data.



Walkthrough multivariate Gaussians

```
print(f'E[X] = {np.mean(X, axis=0)}')
print(f'Var[X] = {np.var(X, axis=0)}')
print(f'Cov[X] =\n {np.cov(X.T)}')
```

```
E[X] = [1.02687839 1.98999062]
Var[X] = [2.02788973 1.0213675 ]
Cov[X] =
 [[ 2.02991965 -0.01297718]
 [-0.01297718  1.02238989]]
```

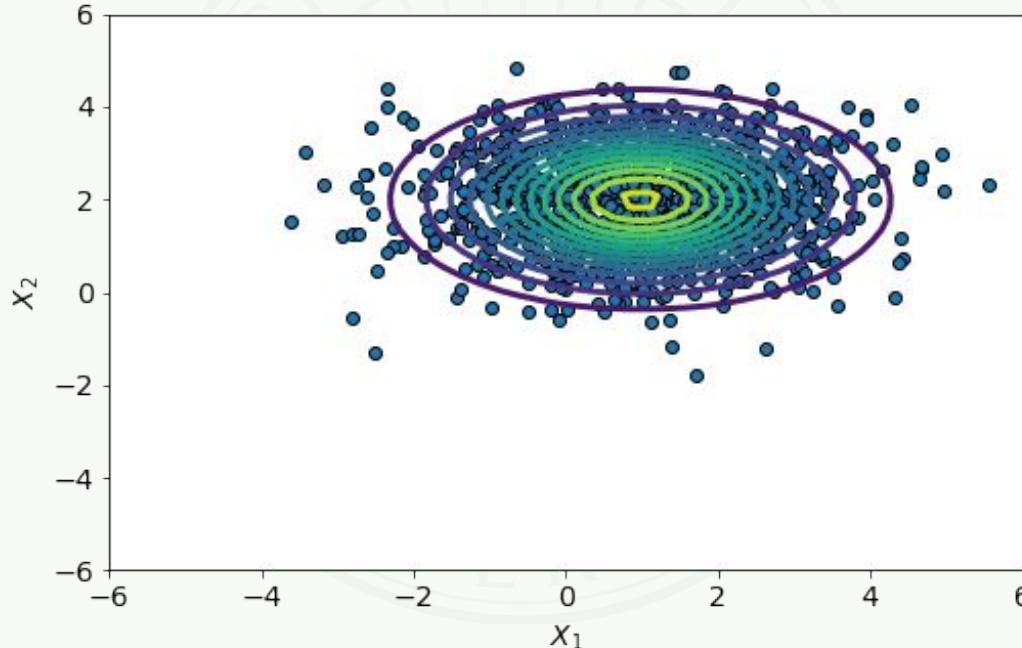


$$\mu_x = 1.02 \quad \mu_y = 1.98 \quad \Sigma = \begin{bmatrix} 2.02 & -0.01 \\ -0.01 & 1.02 \end{bmatrix}$$



Walkthrough multivariate Gaussians

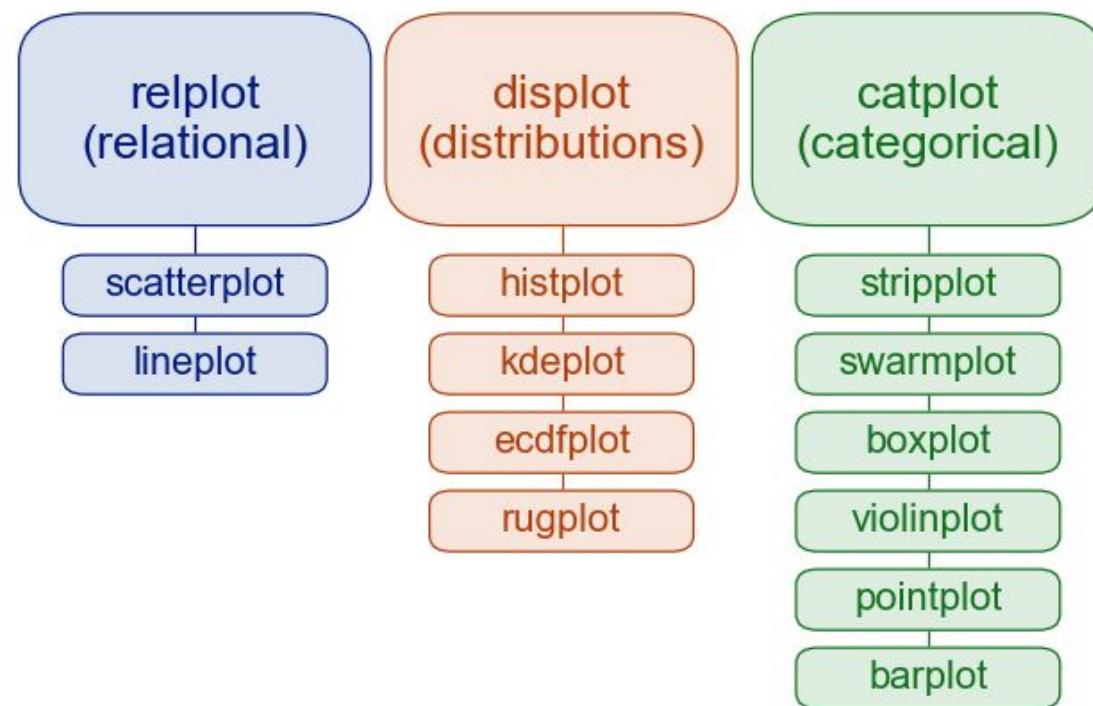
$$G(x, y) = \frac{1}{\sqrt{4\pi^2 \cdot 2.25}} \exp \left(-\frac{1}{2} \begin{bmatrix} x - \mu_x & y - \mu_y \end{bmatrix} \begin{bmatrix} 2.02 & -0.01 \\ -0.01 & 1.02 \end{bmatrix}^{-1} \begin{bmatrix} x - \mu_x \\ y - \mu_y \end{bmatrix} \right)$$



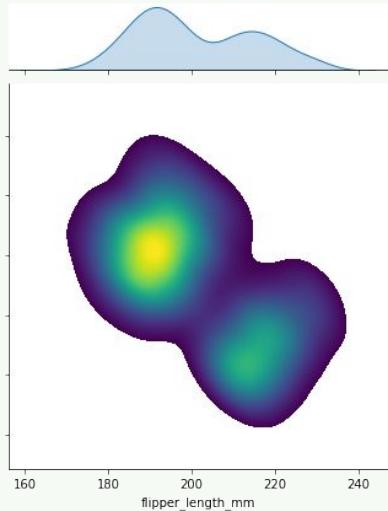
Homework and Inclass



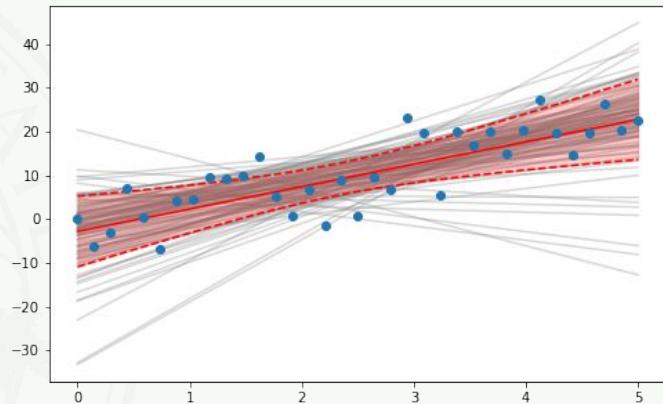
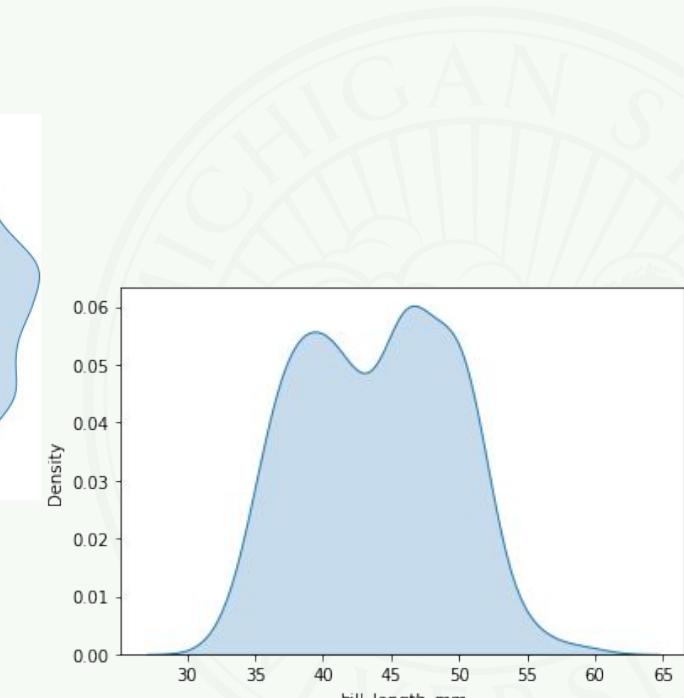
Useful plots in Seaborn



You will explore many Seaborn plots in the ICA and HW



Visualizing joint and
marginal probability
distributions



Visualizing uncertainty
in estimates

