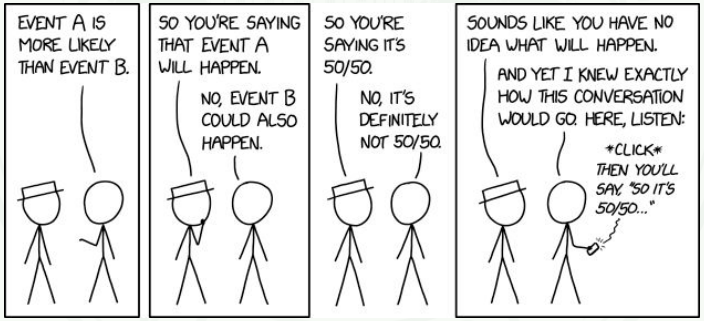
Statistics and Probability



David J Butts
Computational Mathematics Science and Engineering
Michigan State University

Stats

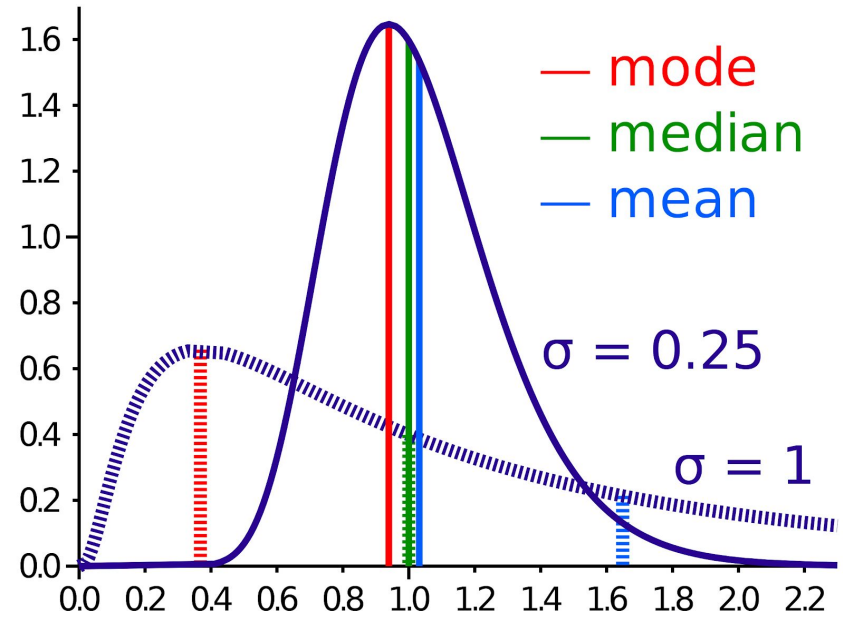
How much do summary statistics actually tell us?

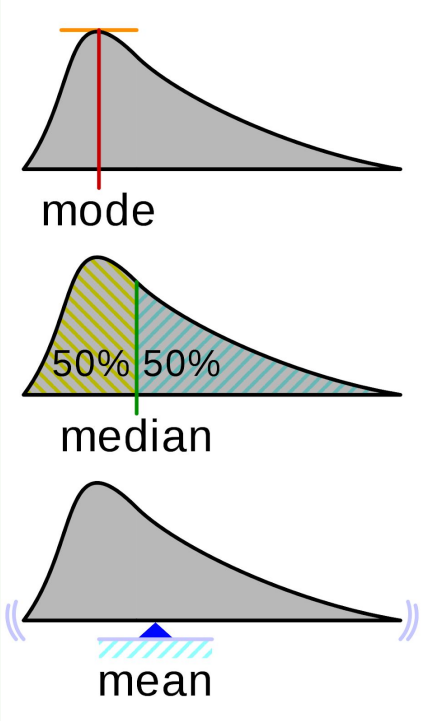
X Mean:	54.26	59224
Y Mean:	47.83	13999
X SD :	16.76	49829
Y SD :	26.93	42120
Corr. :	-0.06	42526

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

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

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

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

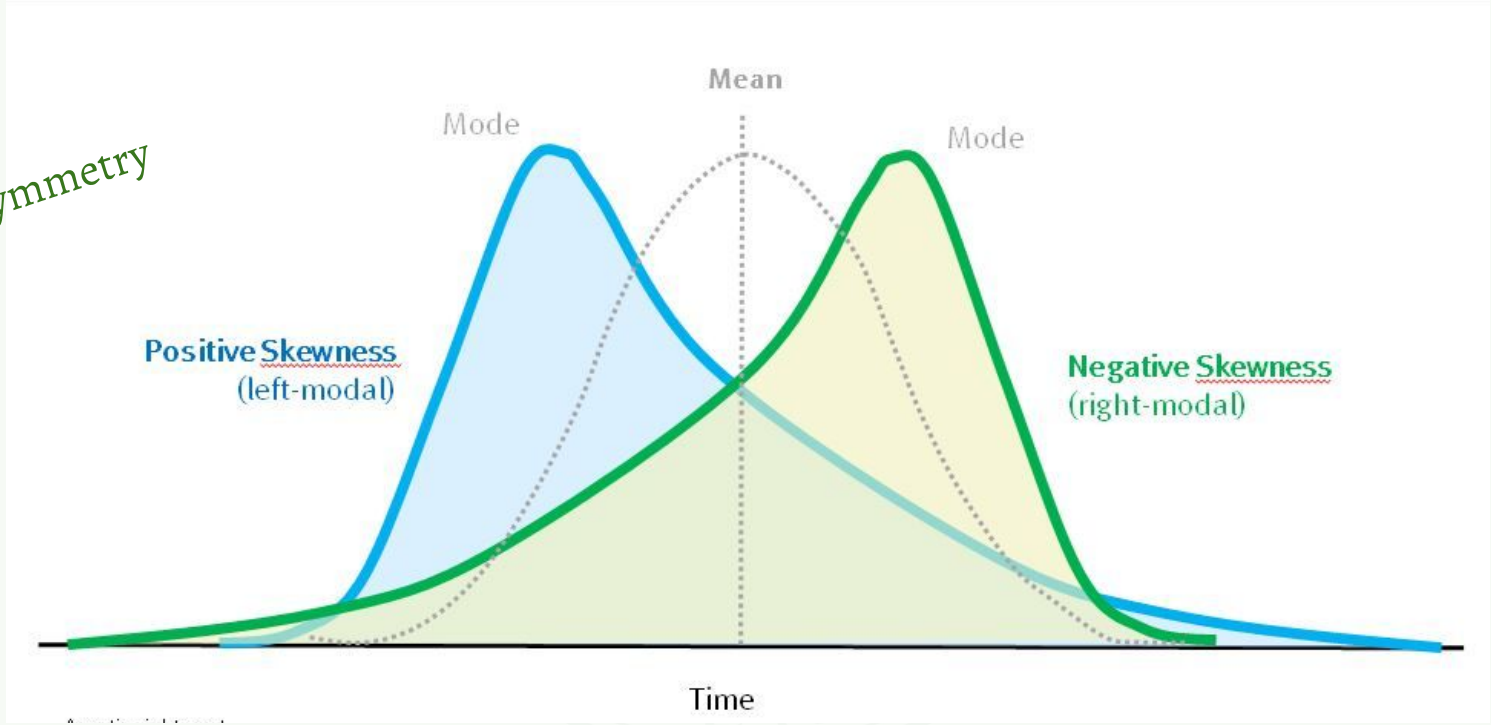
$$Correlation = \frac{Cov\,(x,y)}{\sigma x * \sigma y}$$

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.

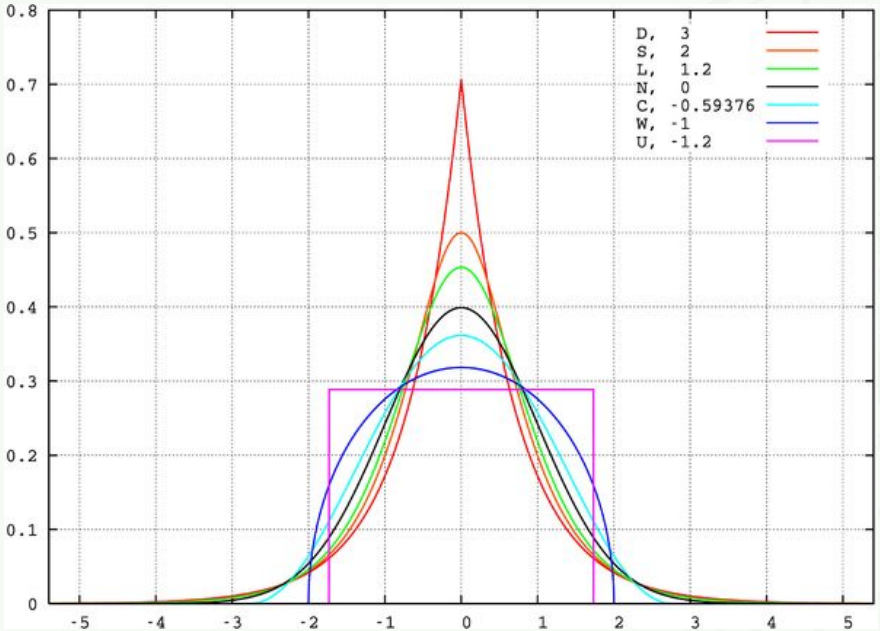
Quantities like this are sometimes called "autocorrelation functions".

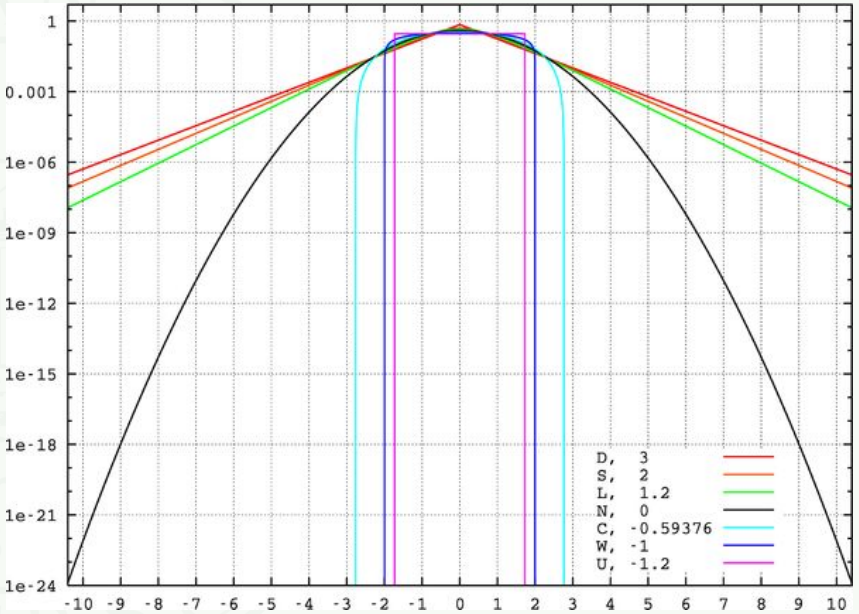
Skewness


asymmetry
Mean
Mode
Mode
Positive Skewness
(left-modal)
Negative Skewness
(right-modal)
Time

Summary statistics: kurtosis

tailedness

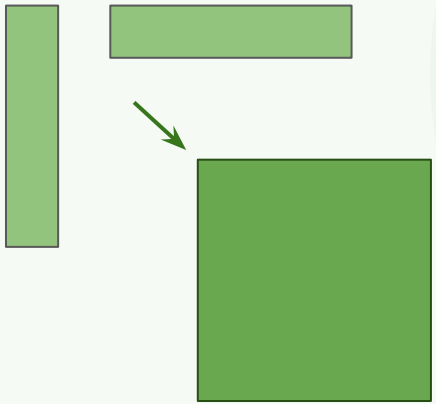


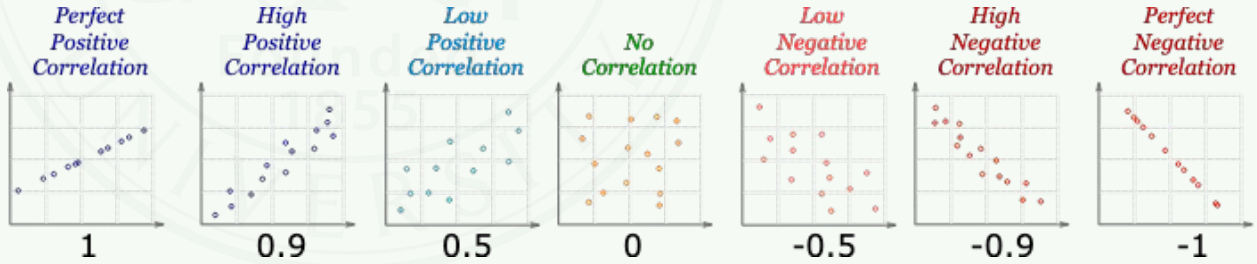


Covariance Matrix

Covariance matrix: *X* is a column vector

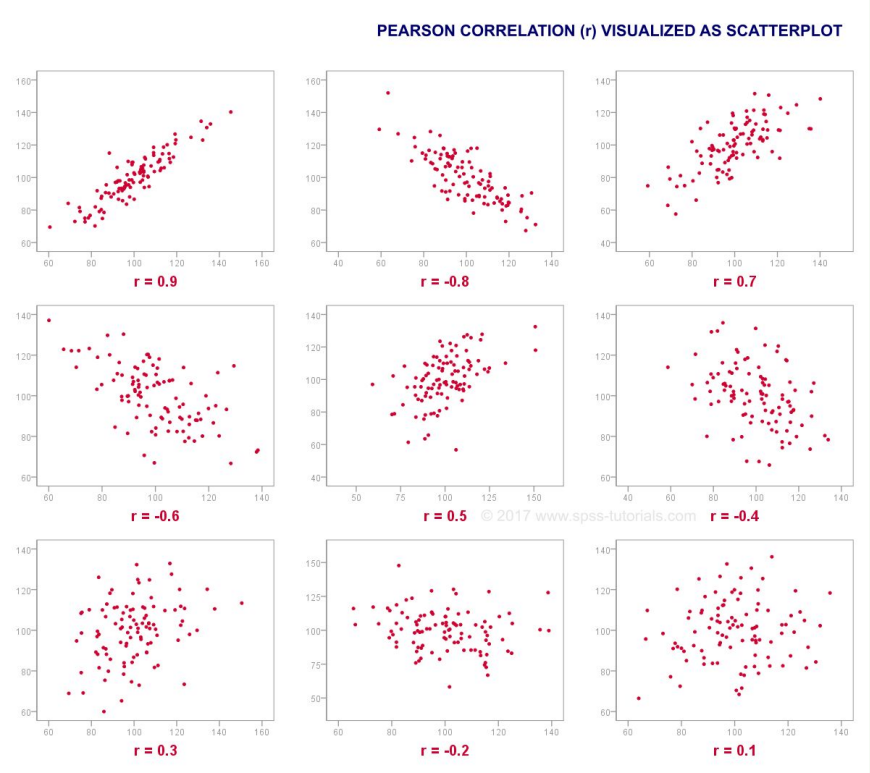
$$E[(X-\mu)(X-\mu)^T] = \begin{pmatrix} E[(X_1-\mu_1)^2] & E[(X_1-\mu_1)(X_2-\mu_2)] & \dots \\ E[(X_1-\mu_1)(X_2-\mu_2)] & E[(X_2-\mu_2)^2] & \dots \\ \vdots & & \\ E[(X_n-\mu_n)(X_1-\mu_1)] & E[(X_n-\mu_n)(X_2-\mu_2)] & \dots \end{pmatrix}$$



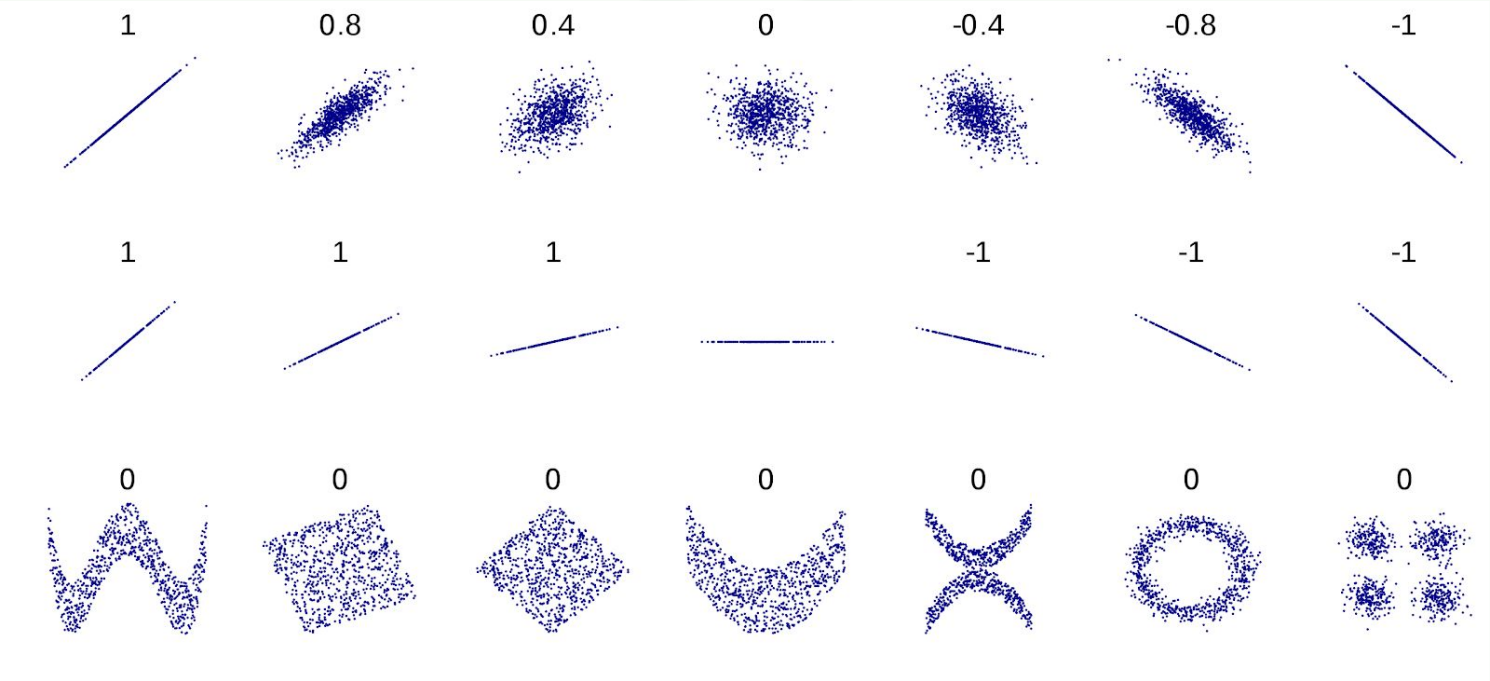


Pearson correlation

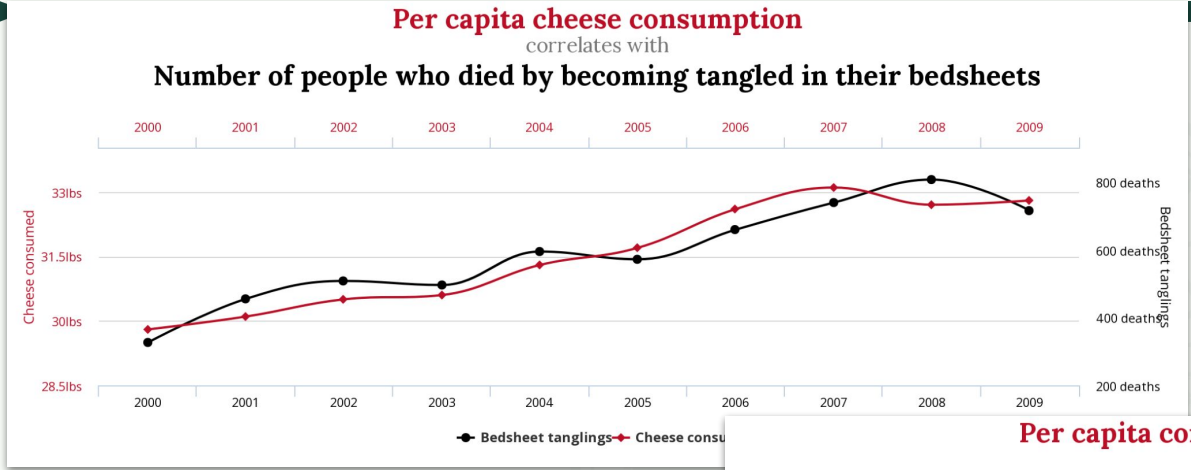
$$\rho_{X,Y} = \frac{Cov(X,Y)}{\sqrt{Cov(X,X)Cov(Y,Y)}}$$



Pearson does not reflect shape

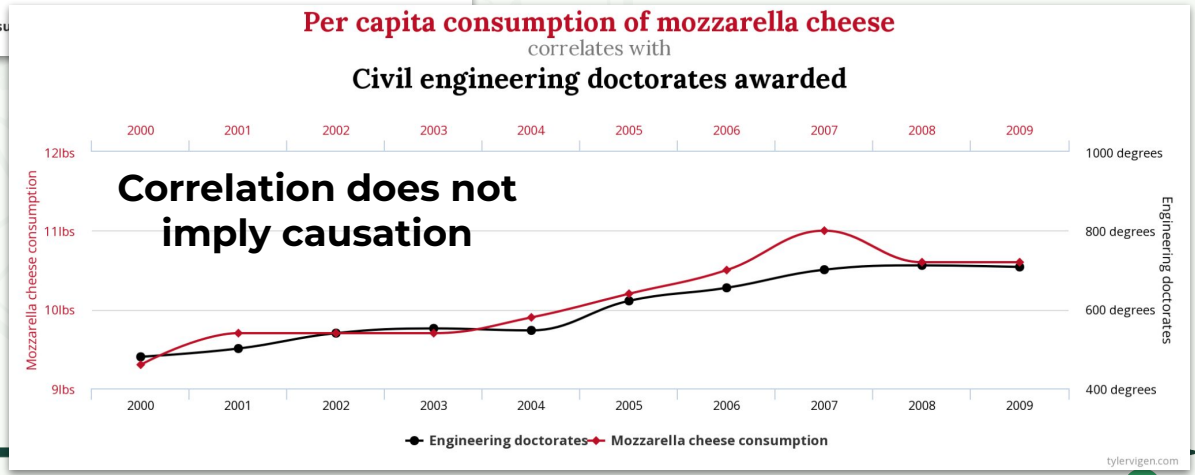


Should we risk eating cheese?



Does cheese kill?

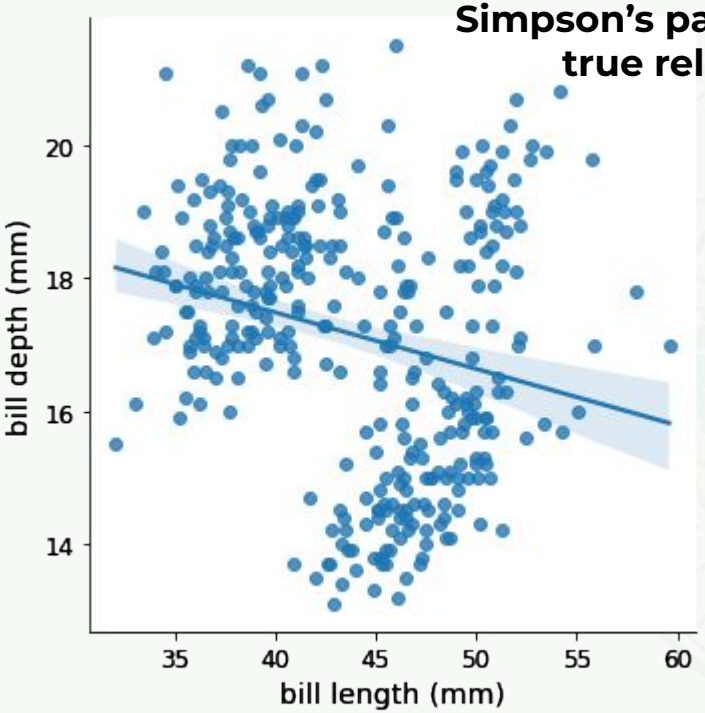
Or, does cheese make us smarter?

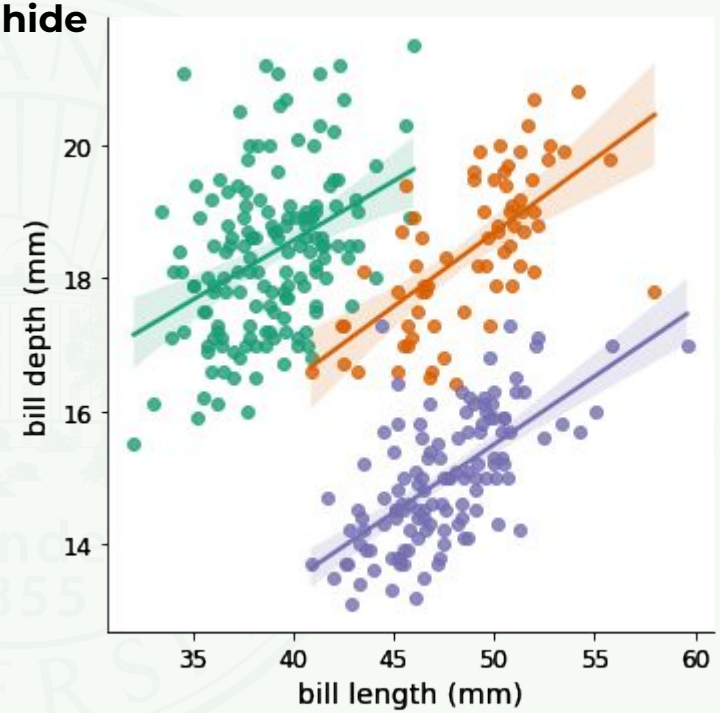


https://www.tylervigen.com/spurious-correlations

Am I picking the best fit to my data?

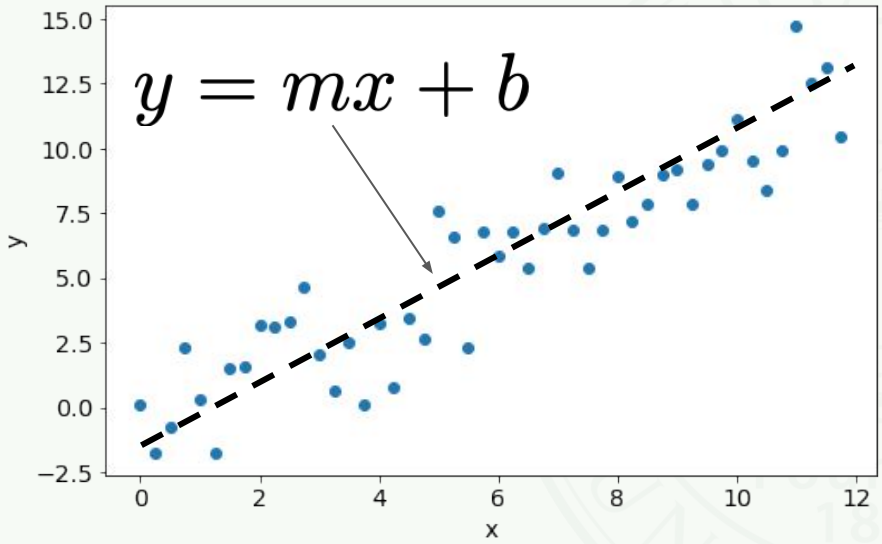
Simpson's paradox may hide true relationships





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

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

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

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

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

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

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

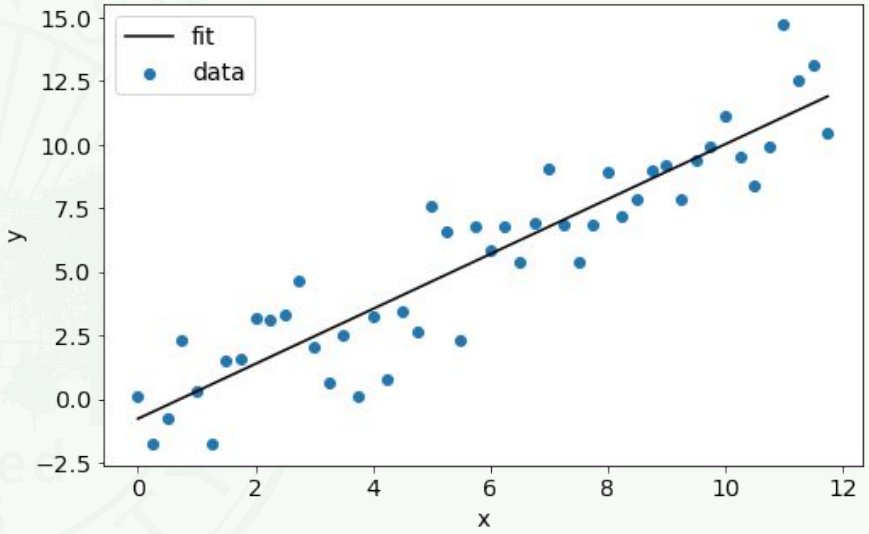
$$m = \frac{\frac{1}{N}\sum_i y_i x_i - \bar{x}\bar{y}}{\frac{1}{N}\sum_i x_i x_i - \bar{x}\bar{x}}$$

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

The least squares fit is related to statistical fits

$$y = \frac{cov(x,y)}{var(x)}x + \bar{y} - \frac{cov(x,y)}{var(x)}\bar{x}$$

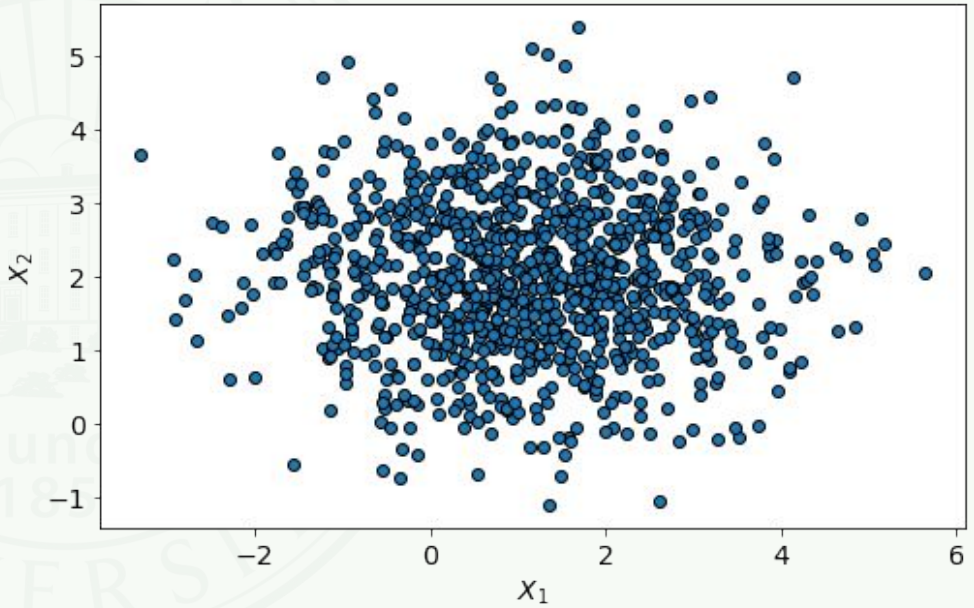
$$= \frac{cov(x,y)}{var(x)}(x - \bar{x}) + \bar{y}$$



Multidimensional data example

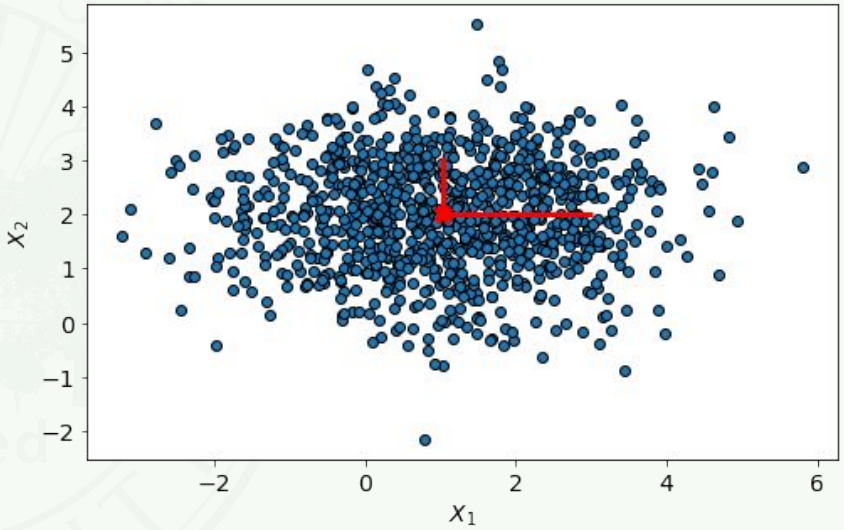
```
X =
[[ 0.38849788  5.08330752]
 [ 0.81034528  2.04741571]
 [-1.56082481  1.31164164]
 ...
 [ 2.355576    0.53936631]
 [ 1.52170224  2.02778523]
 [ 0.43006137  1.65176376]]
```

1,000 by 2 numbers



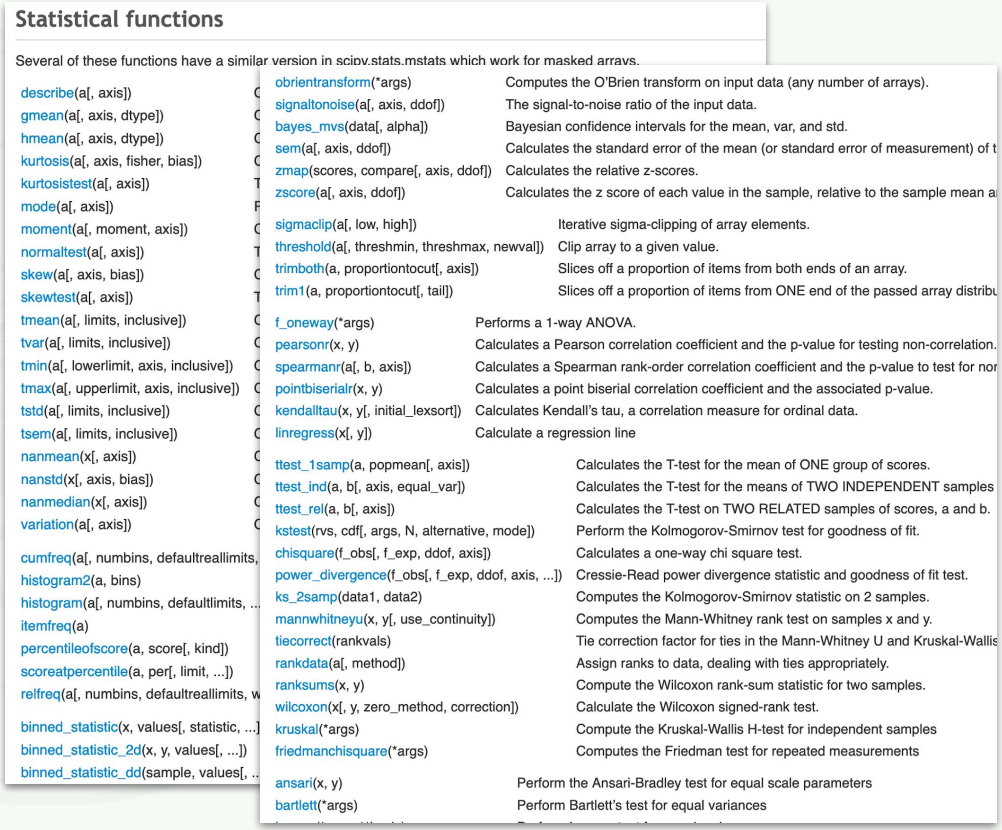
```
print(f'E[X] = {np.mean(X,axis=0)}')
print(f'Var[X] = {np.var(X,axis=0)}')
print(f'Cov[X] =\n {np.cov(X.T)}')
```

```
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.

describe(a[, axis])
gmean(a[, axis, dtype])
hmean(a[, axis, dtype])
kurtosis(a[, axis, fisher, bias])
kurtosistest(a[, axis])
mode(a[, axis])
moment(a[, moment, axis])
normaltest(a[, axis])
skew(a[, axis, bias])
skewtest(a[, axis])
tmean(a[, limits, inclusive])
tvar(a[, limits, inclusive])
tmin(a[, lowerlimit, axis, inclusive])
tmax(a[, upperlimit, axis, inclusive])
tstd(a[, limits, inclusive])
tsem(a[, limits, inclusive])
nanmean(x[, axis])
nanstd(x[, axis, bias])
nanmedian(x[, axis])
variation(a[, axis])

cumfreq(a[, numbins, defaultreallimits,
histogram2(a, bins)
histogram(a[, numbins, defaultlimits, ...
itemfreq(a)
percentileofscore(a, score[, kind])
scoreatpercentile(a, per[, limit, ...])
relfreq(a[, numbins, defaultreallimits, w

binned_statistic(x, values[, statistic, ...]
binned_statistic_2d(x, y, values[, ...])
binned_statistic_dd(sample, values[, ...

obrientransform(*args)	Computes the O'Brien transform on input data (any number of arrays).
signaltonoise(a[, axis, ddof])	The signal-to-noise ratio of the input data.
bayes_mvs(data[, alpha])	Bayesian confidence intervals for the mean, var, and std.
sem(a[, axis, ddof])	Calculates the standard error of the mean (or standard error of measurement) of t
zmap(scores, compare[, axis, ddof])	Calculates the relative z-scores.
zscore(a[, axis, ddof])	Calculates the z score of each value in the sample, relative to the sample mean a

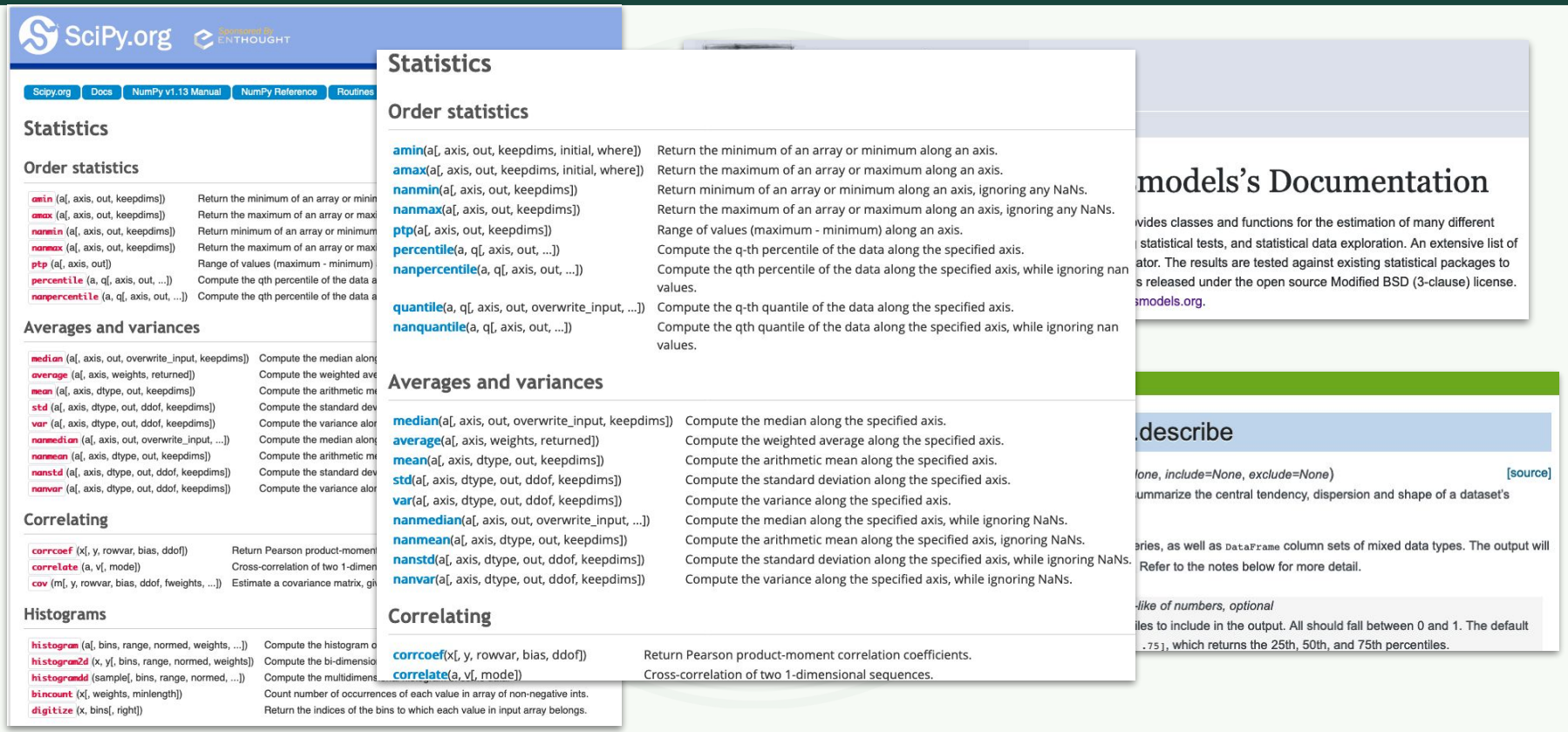
sigmaclip(a[, low, high])	Iterative sigma-clipping of array elements.
threshold(a[, threshmin, threshmax, newval])	Clip array to a given value.
trimboth(a, proportiontocut[, axis])	Slices off a proportion of items from both ends of an array.
trim1(a, proportiontocut[, tail])	Slices off a proportion of items from ONE end of the passed array distribu

f_oneway(*args)	Performs a 1-way ANOVA.
pearsonr(x, y)	Calculates a Pearson correlation coefficient and the p-value for testing non-correlation.
spearmanr(a[, b, axis])	Calculates a Spearman rank-order correlation coefficient and the p-value to test for nor
pointbiserialr(x, y)	Calculates a point biserial correlation coefficient and the associated p-value.
kendalltau(x, y[, initial_lexsort])	Calculates Kendall's tau, a correlation measure for ordinal data.
linregress(x[, y])	Calculate a regression line

ttest_1samp(a, popmean[, axis])	Calculates the T-test for the mean of ONE group of scores.
ttest_ind(a, b[, axis, equal_var])	Calculates the T-test for the means of TWO INDEPENDENT samples
ttest_rel(a, b[, axis])	Calculates the T-test on TWO RELATED samples of scores, a and b.
kstest(rvs, cdf[, args, N, alternative, mode])	Perform the Kolmogorov-Smirnov test for goodness of fit.
chisquare(f_obs[, f_exp, ddof, axis])	Calculates a one-way chi square test.
power_divergence(f_obs[, f_exp, ddof, axis, ...])	Cressie-Read power divergence statistic and goodness of fit test.
ks_2samp(data1, data2)	Computes the Kolmogorov-Smirnov statistic on 2 samples.
mannwhitneyu(x, y[, use_continuity])	Computes the Mann-Whitney rank test on samples x and y.
tiecorrect(rankvals)	Tie correction factor for ties in the Mann-Whitney U and Kruskal-Wallis
rankdata(a[, method])	Assign ranks to data, dealing with ties appropriately.
ranksums(x, y)	Compute the Wilcoxon rank-sum statistic for two samples.
wilcoxon(x[, y, zero_method, correction])	Calculate the Wilcoxon signed-rank test.
kruskal(*args)	Compute the Kruskal-Wallis H-test for independent samples
friedmanchisquare(*args)	Computes the Friedman test for repeated measurements

ansari(x, y)	Perform the Ansari-Bradley test for equal scale parameters
bartlett(*args)	Perform Bartlett's test for equal variances

Many Python libraries include stats capabilities



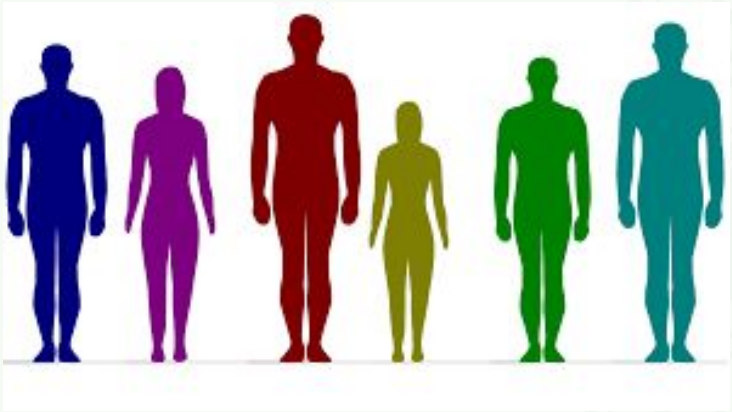
Probability

Example: flipping a fair coin

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

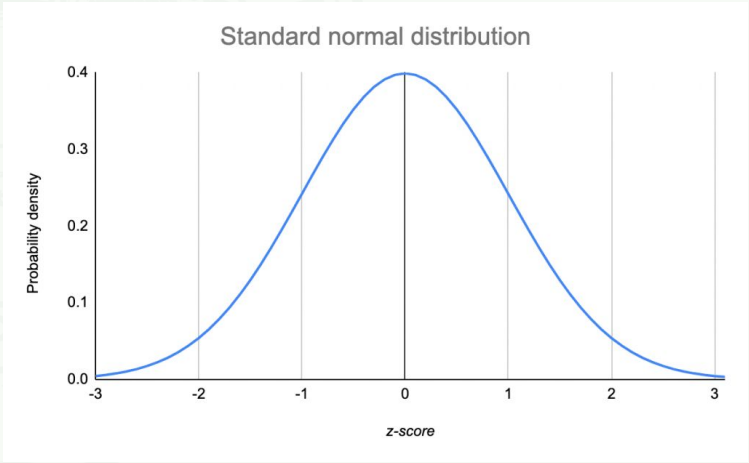
$$p(\text{heads}) = p(\text{tails}) = \frac{1}{2}$$

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.

$$p(x) \geq 0$$

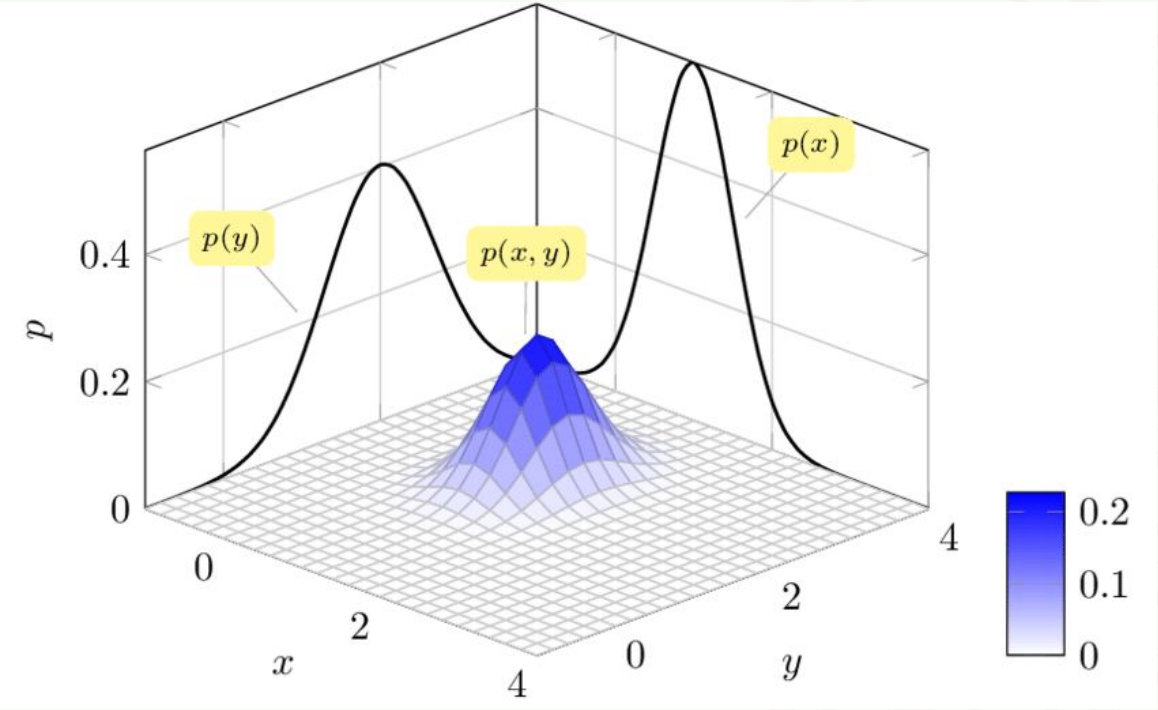
$$f(x) \geq 0$$

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

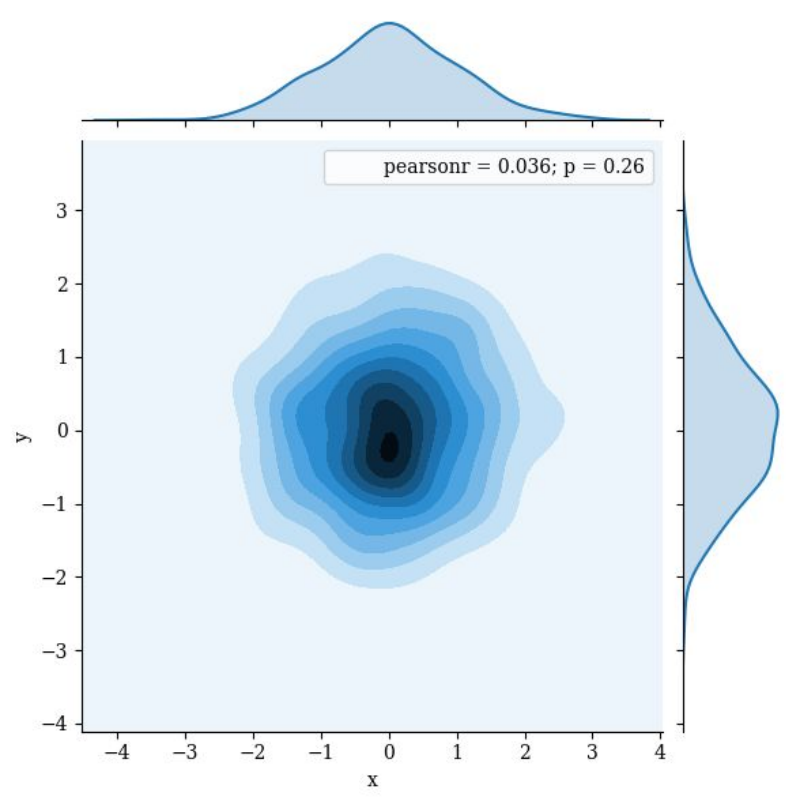


Marginals

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

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

Top-down view of a joint distribution



Probabilities can be conditional

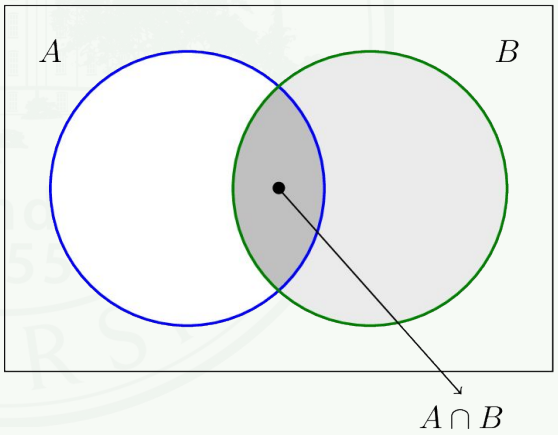
The bar means "given that"

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

Probability of A and B happening.

Probability of B happening.

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



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?

$$P(\text{heads}|\text{heads}) = \frac{P(\text{heads} \cap \text{heads})}{P(\text{heads})}$$

$$= \frac{1/2 \cdot 1/2}{1/2} = \frac{1}{2}$$

Bayes' theorem switches the order of conditionals

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

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

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

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

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

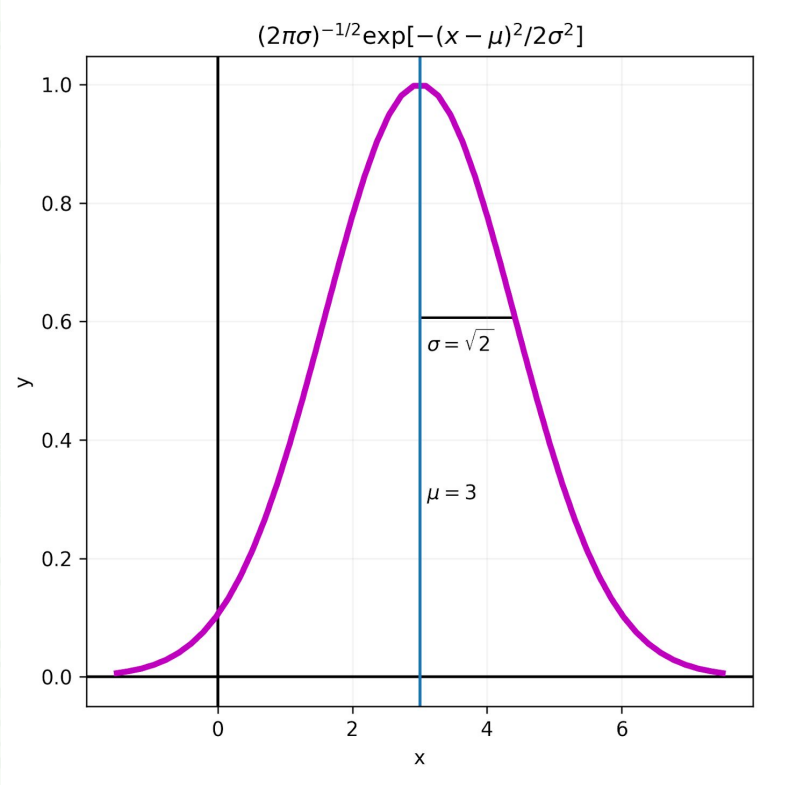
$$P(\text{danger}|\text{smoke}) = \frac{.9 \cdot .01}{.1} = .09$$

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.

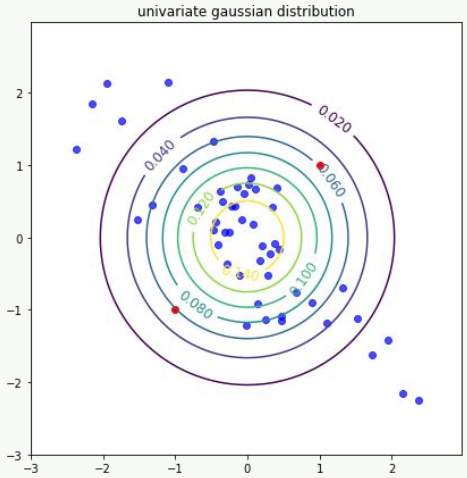


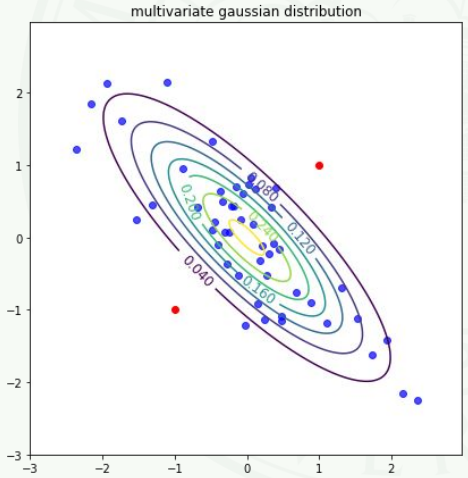
Multivariate Gaussians

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

mean

covariance matrix





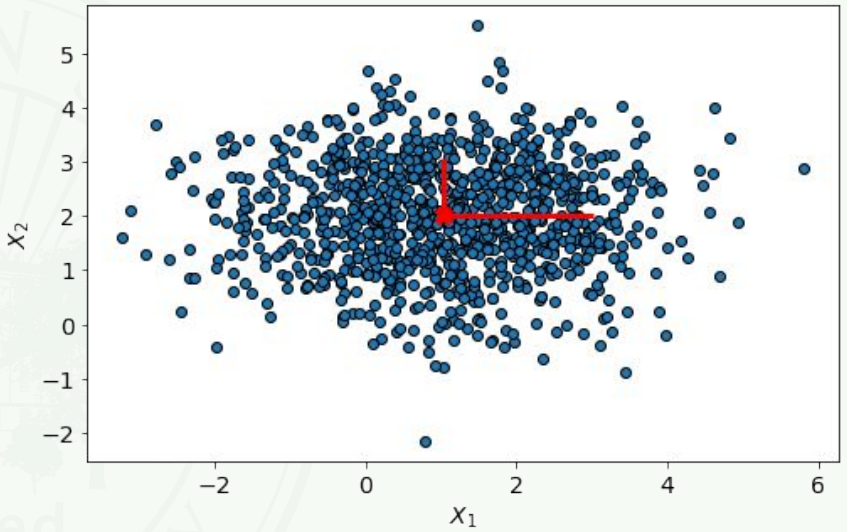
$$\Sigma_{ij} = E[(X_i - \mu_i)(X_j - \mu_j)]$$

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)}')
```

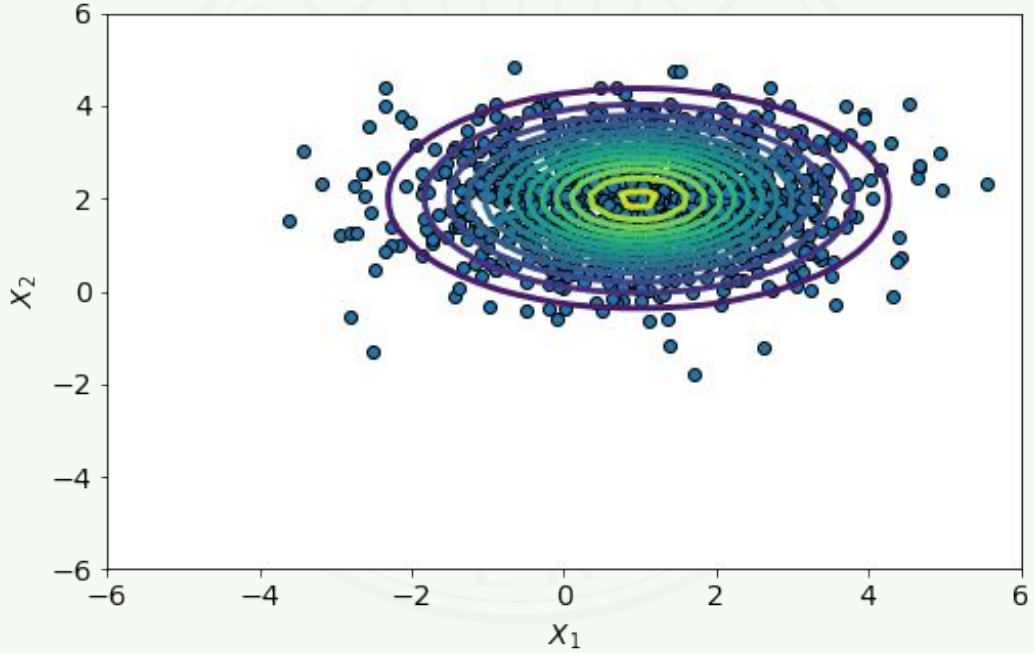
```
E[X] = [1.02687839 1.98999062]
Var[X] = [2.02788973 1.0213675 ]
Cov[X] =
 [[ 2.02991965 -0.01297718]
 [-0.01297718  1.02238989]]
```



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

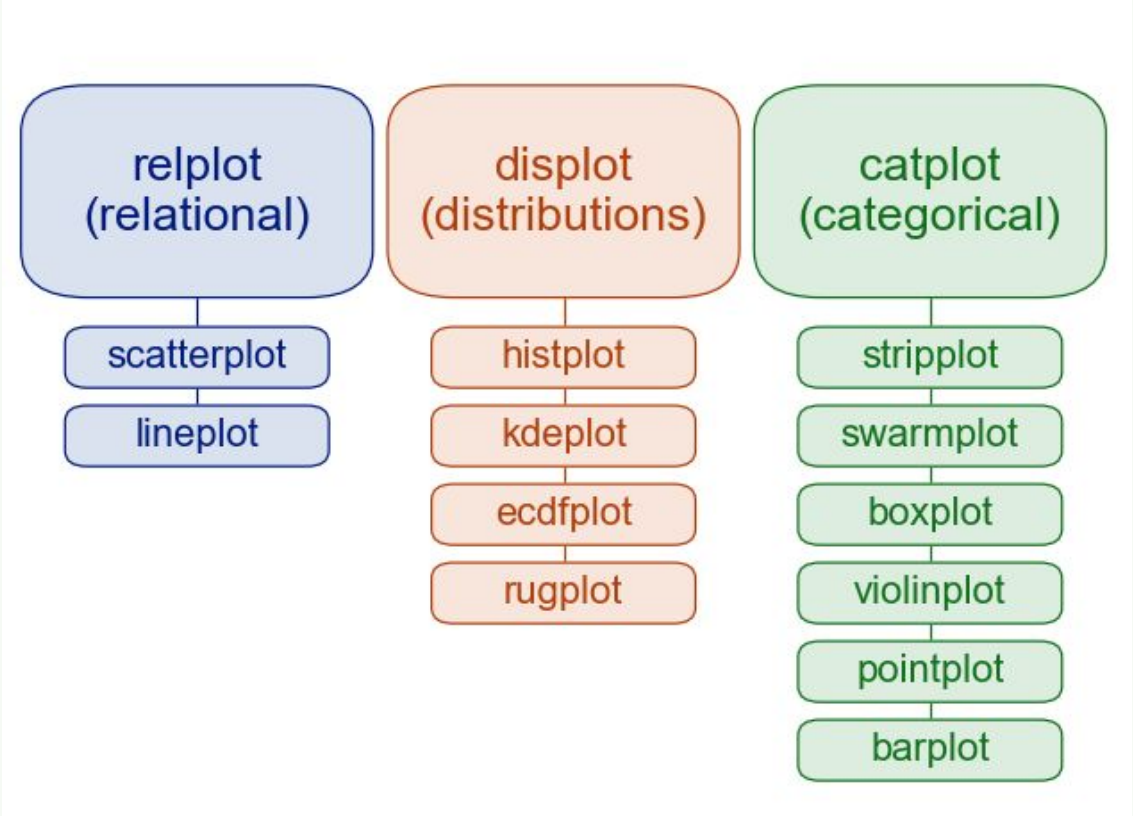
Walkthrough multivariate Gaussians

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

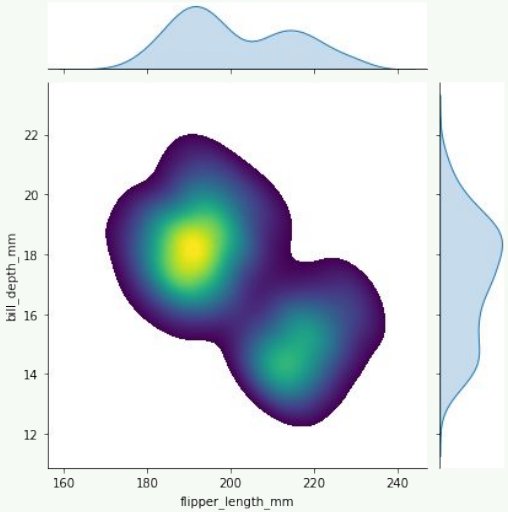


Homework and Inclass

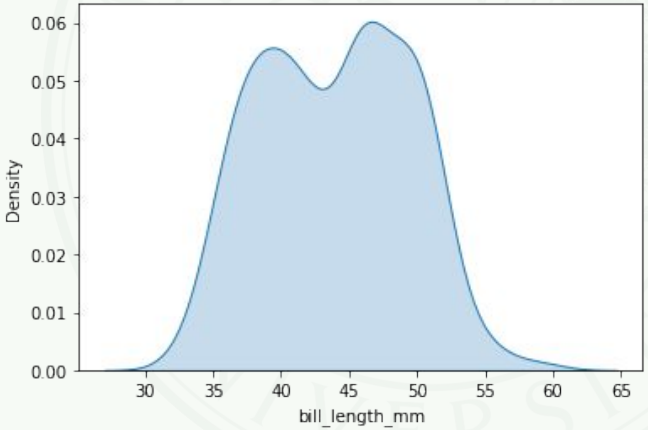
Useful plots in Seaborn

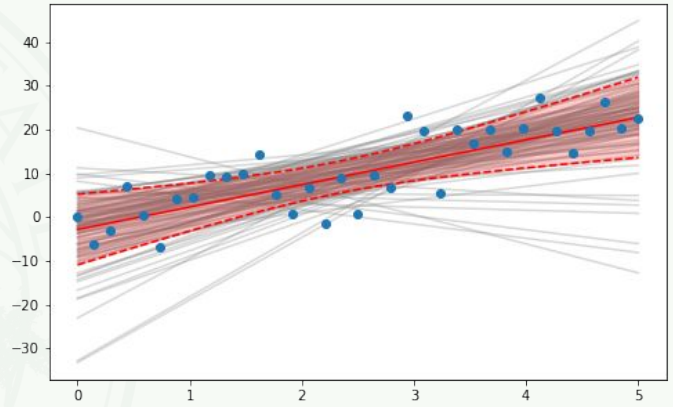


You will explore many Seaborn plots in the ICA and HW



Visualizing joint and marginal probability distributions





Visualizing uncertainty in estimates