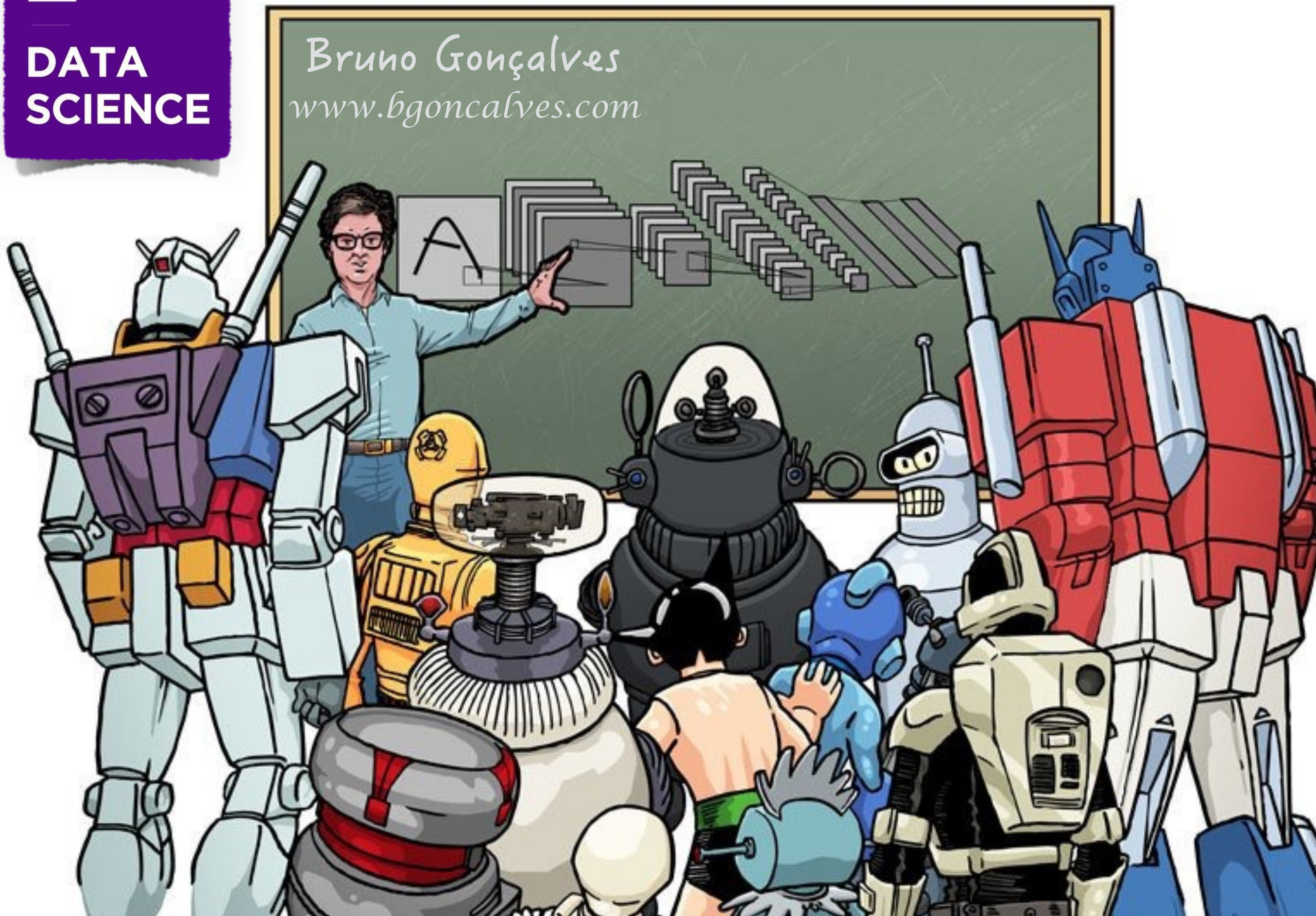
NYU

DATA SCIENCE

Machine(s) Learning and Data Science

Bruno Gonçalves
www.bgoncalves.com



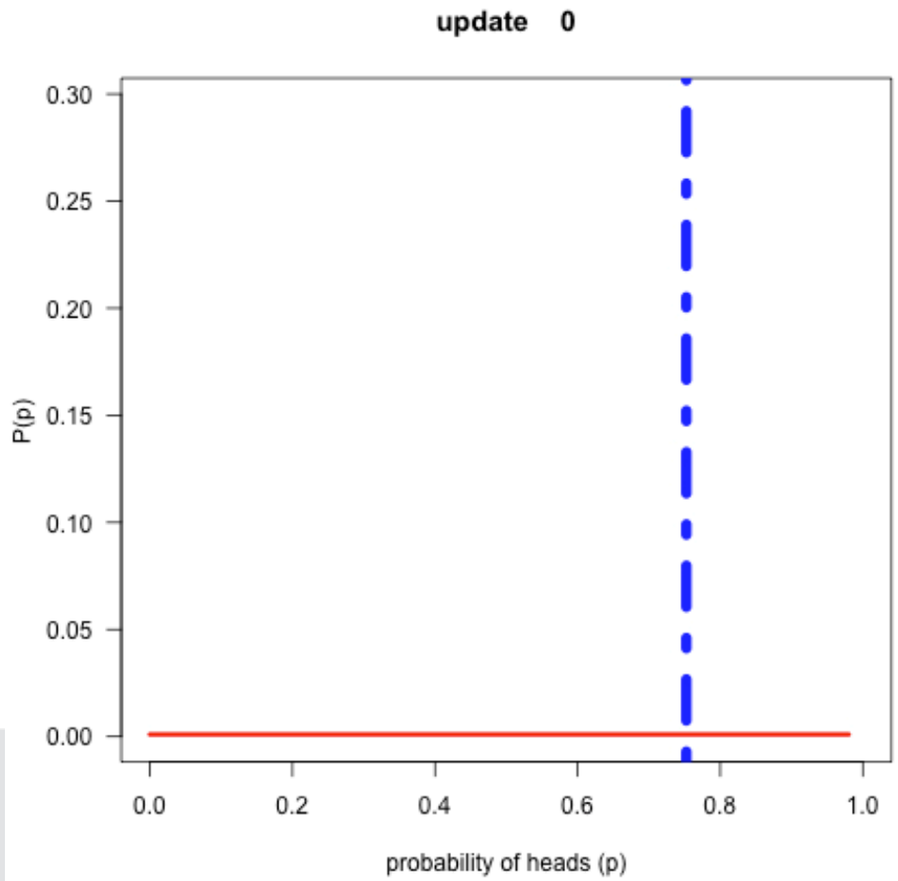
Bayesian Coin Flips

http://youtu.be/GTx0D8VY0CY

- Biased coin with unknown probability of heads (p)
- Perform N flips and update our belief after each flip using Bayes Theorem

$$P\left(p|heads\right) = \frac{P\left(heads|p\right)P\left(p\right)}{P\left(heads\right)}$$

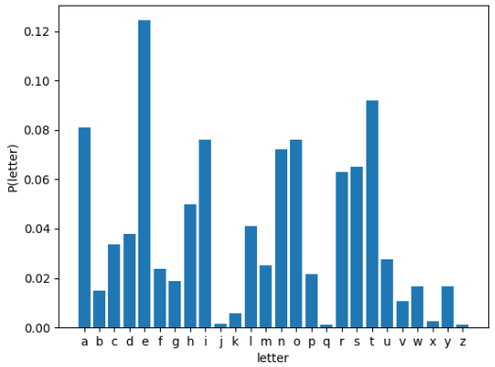
$$P\left(p|tails\right) = \frac{P\left(tails|p\right)P\left(p\right)}{P\left(tails\right)}$$




```
# Uninformative prior
prior = np.ones(bins, dtype='float')/bins
likelihood_heads = np.arange(bins)/float(bins)
likelihood_tails = 1-likelihood_heads
flips = np.random.choice(a=[True, False], size=flips, p=[0.75, 0.25])

for coin in flips:
    if coin:  # Heads
        posterior = prior * likelihood_heads
    else:  # Tails
        posterior = prior * likelihood_tails

    # Normalize
    posterior /= np.sum(posterior)

    # The posterior is now the new prior
    prior = posterior
```

coins.py

Language Detection



- Previously, we measured the probability distribution of letters in the english language. In effect, we calculated:

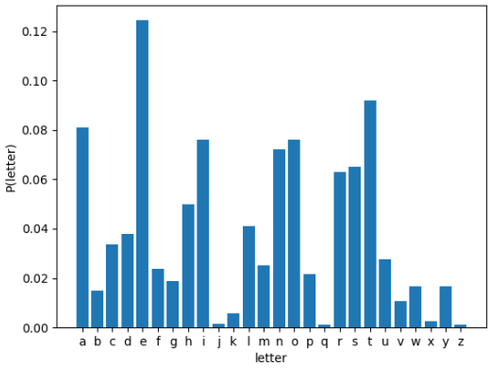
$$P\left(letter|english\right)$$

- The probability of seeing a specific letter given that the text is in English. If we do this for a few other languages we can have a table of the form:

$$P\left(letter|language\right)$$

- In the repository you'll find table_langs.dat where this conditional probability is calculated for 5 different languages: English, French, German, Italian and Spanish.

Language Detection



- Previously, we measured the probability distribution of letters in the english language. In effect, we c

- The probab

  a few other

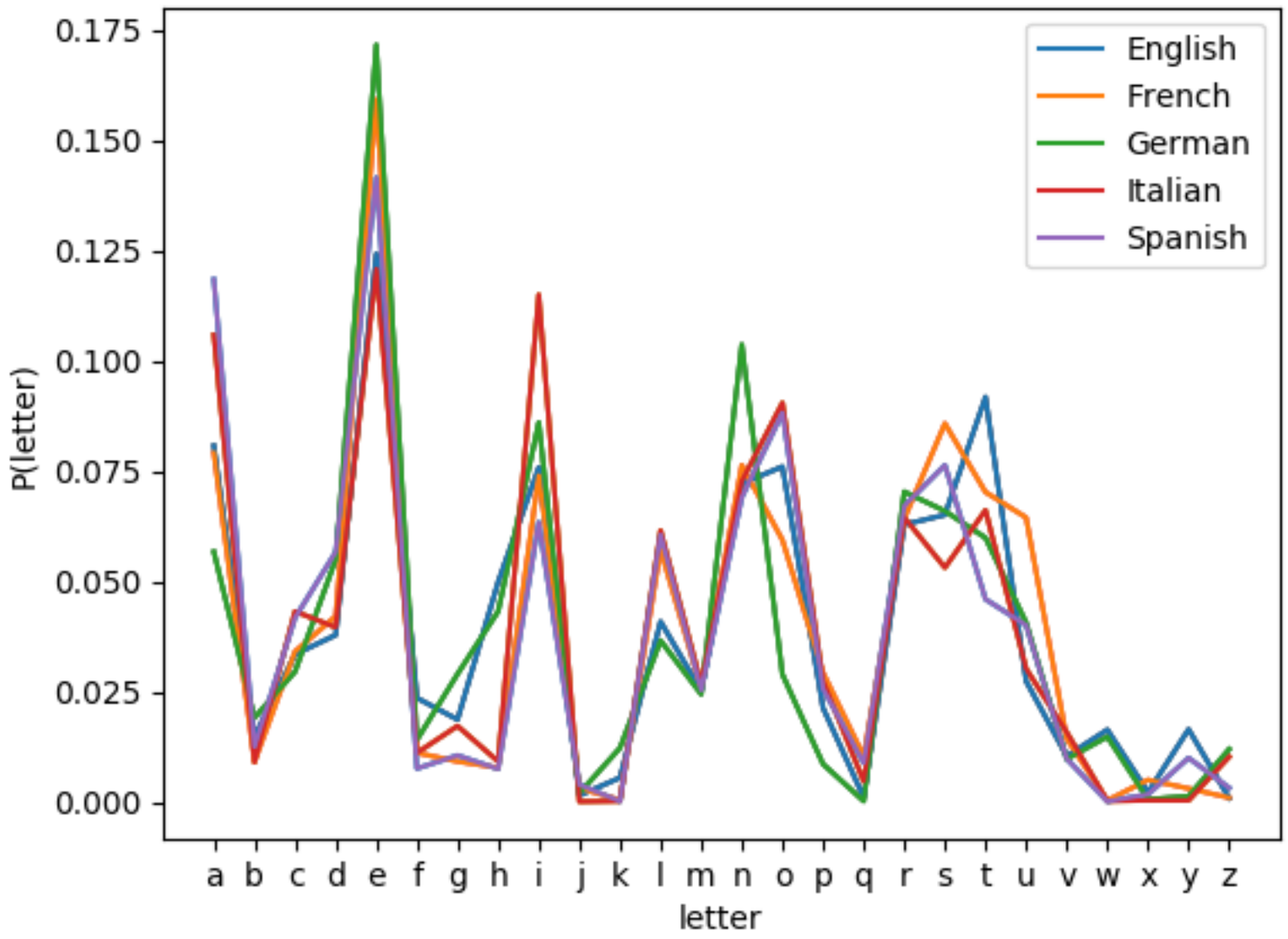
  ve do this for

- In the repos

  calculated fc

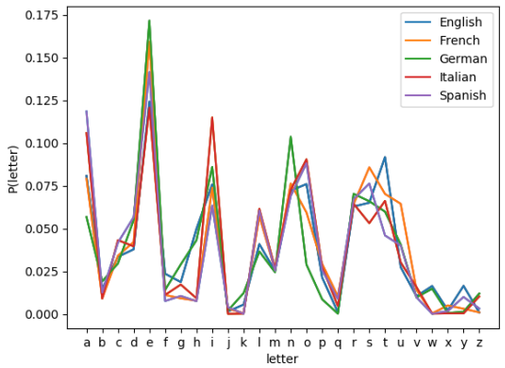
  ty is

  d Spanish.



plot_langs.py

Language Detection



- A few minutes ago we measured the probability distribution of letters in the english language. In effect, we calculated

$$P\left(letter|english\right)$$

- The probability of seeing a specific letter given that the text is in English. If we do this for a few other languages we can have a table of the form:

$$P\left(letter|language\right)$$

- In the repository you'll find table_langs.dat where this conditional probability is calculated for 5 different languages: English, French, German, Italian and Spanish.

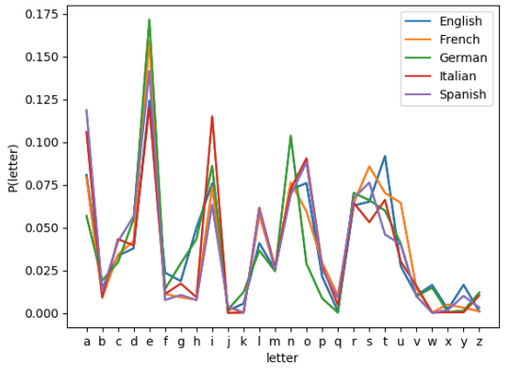
- Using these conditional probabilities, and Bayes Theorem, we can easily build a language detector. For that we just need to calculate:

$$P\left(language|text\right)$$

- We can rewrite it as a:

$$P\left(language|letter_1, letter_2, \cdots, letter_n\right)$$

Naive Bayes Classifier



$$P\left(language|letter_1, letter_2, \cdots, letter_n\right)$$

- If we treat each letter independently, we obtain:

$$P\left(language|letter_1, letter_2, \cdots, letter_n\right) = \prod_i P\left(language|letter_i\right)$$

- This is known as the Naive Bayes Approach and is an obvious oversimplification as it completely ignores correlations present in the sequence of letters.

- All we have to do is apply Bayes Theorem to our original table:

$$P\left(language|letter\right) = \frac{P\left(letter|language\right)P\left(language\right)}{P\left(letter\right)}$$

- And if we assume that all languages are equally probable (non-informative prior)

$$P\left(letter\right) = \frac{1}{N_{langs}}$$

Naive Bayes Classifier - Numerical Considerations

- Naive Bayes approaches (and many others) use terms of the form:

$$\prod_i P\left(A|B_i\right)$$

- which implies multiplying many small numbers. To avoid numerical complications, it is best to use, instead:

$$\sum_i \log P\left(A|B_i\right)$$

- Which is commonly referred to as the "Log-Likelihood". Our expression then becomes:

$$\mathcal{L}\left(language|letter_1, letter_2, \cdots, letter_n\right) = \sum_i \log\left[\frac{P\left(letter_i|language\right)P\left(language\right)}{P\left(letter_i\right)}\right]$$

- Or more simply:

$$\mathcal{L}\left(language|text\right) = \sum_i \log\left[\frac{P\left(letter_i|language\right)P\left(language\right)}{P\left(letter_i\right)}\right]$$

$$\mathcal{L}\left(language|text\right) = \sum_i \mathcal{L}\left(language|letter_i\right)$$

Naive Bayes Classifier - Normalization

- The final question to answer is, how can we compare the different results?

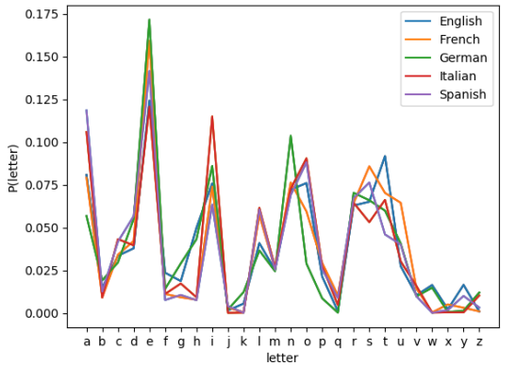
- Intuitively, we expect that the language with the highest likelihood is, well, the most likely one.

- Can we quantify how certain we are that we have the correct answer?

- We just need to remember that likelihoods actually represent probabilities and renormalize them!

$$P\left(language|text\right) = \frac{\exp\left[\mathcal{L}\left(language|text\right)\right]}{\sum_{language}\exp\left[\mathcal{L}\left(language|text\right)\right]}$$

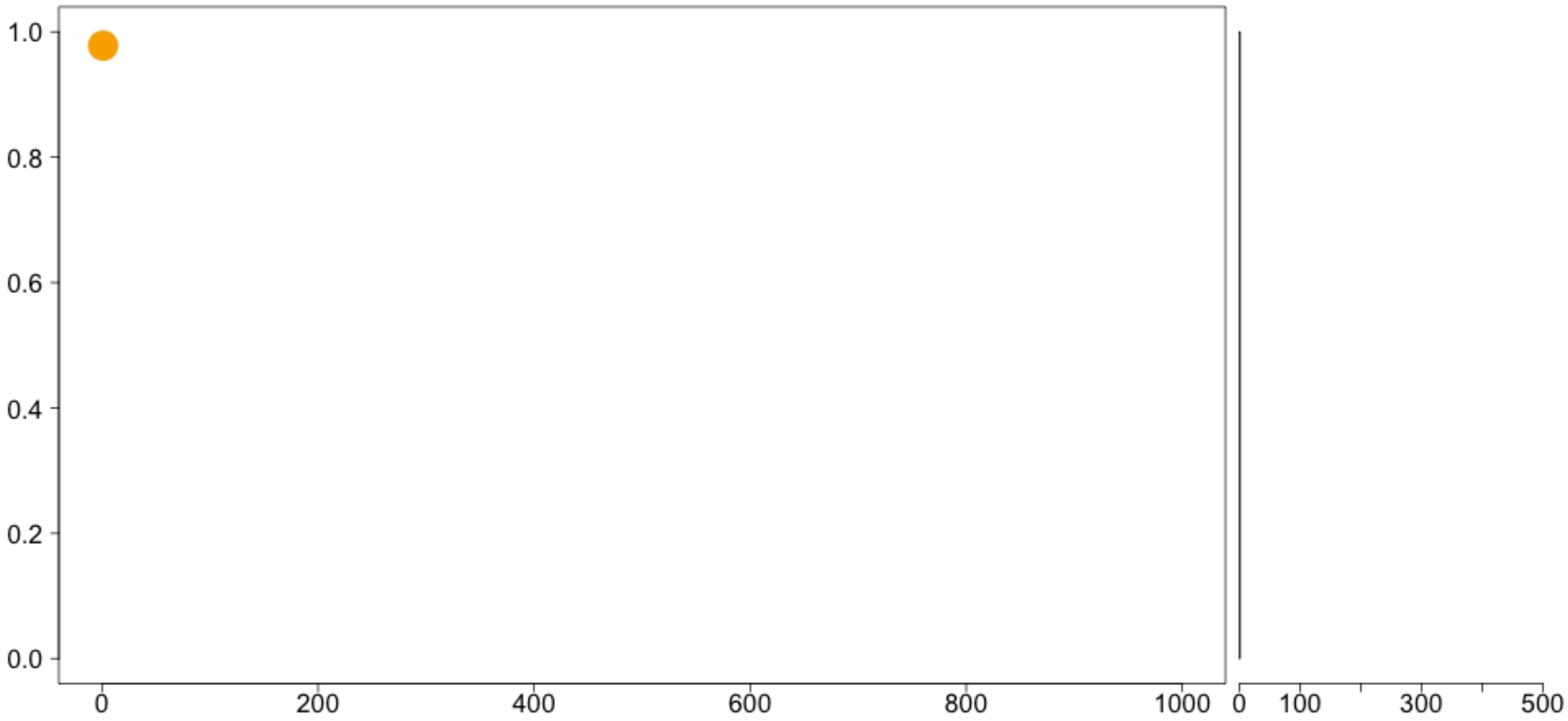
Language Detection




```python
import pandas as pd
import numpy as np
from collections import Counter

def load_data():
    P_letter_lang = pd.read_csv('table_langs.dat', sep=' ', header=0, index_col = 0)

    langs = list(P_letter_lang.columns)

    P_letter = P_letter_lang.mean(axis=1)
    P_letter /= P_letter.sum()

    P_lang_letter = np.array(P_letter_lang)/(P_letter_lang.shape[1]*P_letter.T[:,None])

    L_lang_letter = np.log(P_lang_letter.T)

    return langs, P_letter, L_lang_letter

def detect_lang(langs, P_letter, L_lang_letter, text):
    counts = np.zeros(26, dtype='int')
    pos = dict(zip(P_letter.index, range(26)))

    text_counts =  Counter(text).items()

    for letter, count in text_counts:
        if letter in pos:
            counts[pos[letter]] += count

    L_text = np.dot(L_lang_letter, counts)
    index = np.argmax(L_text)
    lang_text = langs[index]
    prob = np.exp(L_text[index])/np.sum(np.exp(L_text))*100

    return lang_text, prob, L_text
```

detect.py

Law of Large Numbers



Central Limit Theorem

- As $n \to \infty$ the random variables:

$$\sqrt{n}\left(S_n - \mu\right)$$

- with:

$$S_n = \frac{1}{n}\sum_i x_i$$

- converge to a normal distribution:

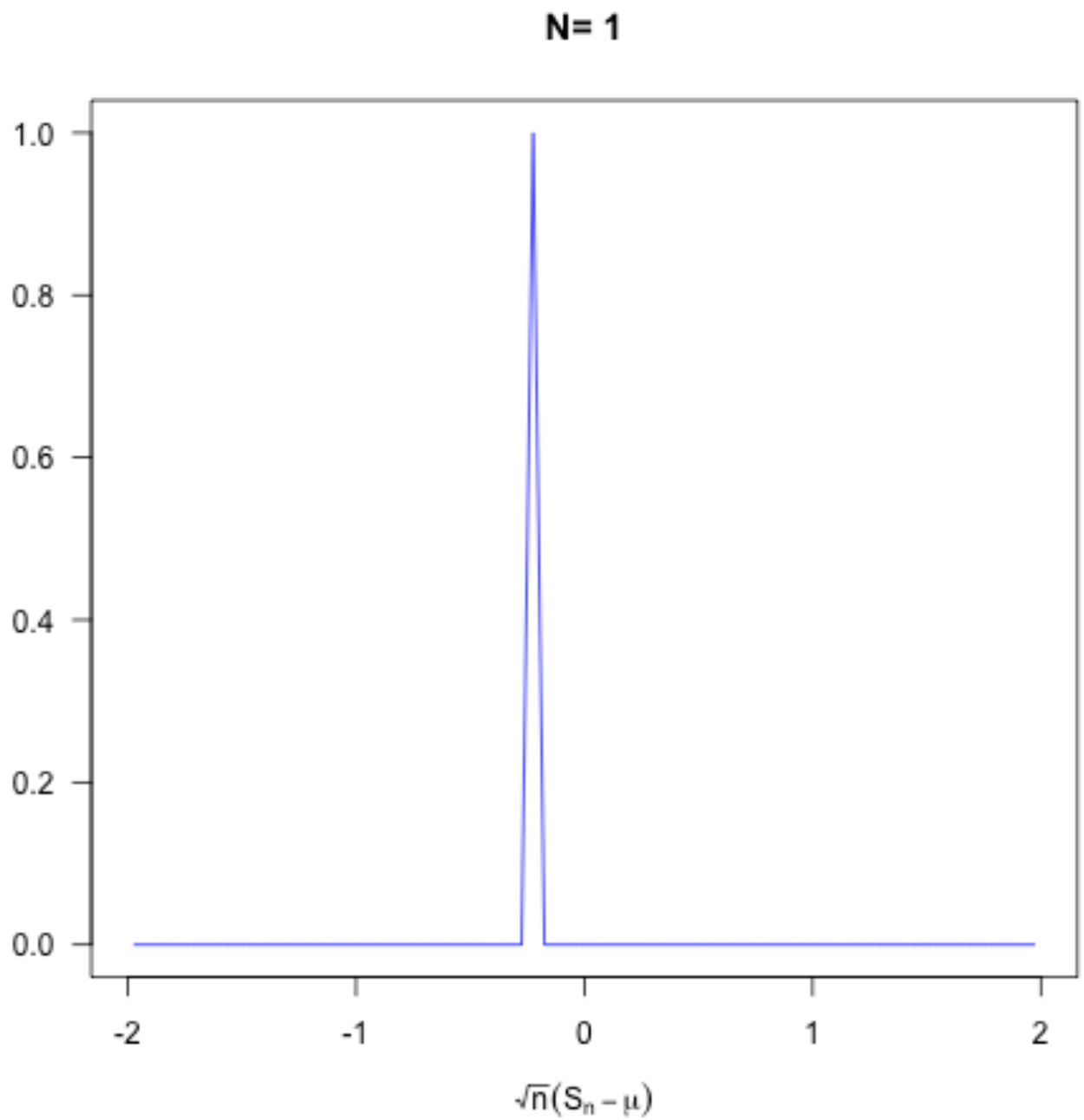
$$\mathcal{N}\left(0, \sigma^2\right)$$

- after some manipulations, we find:

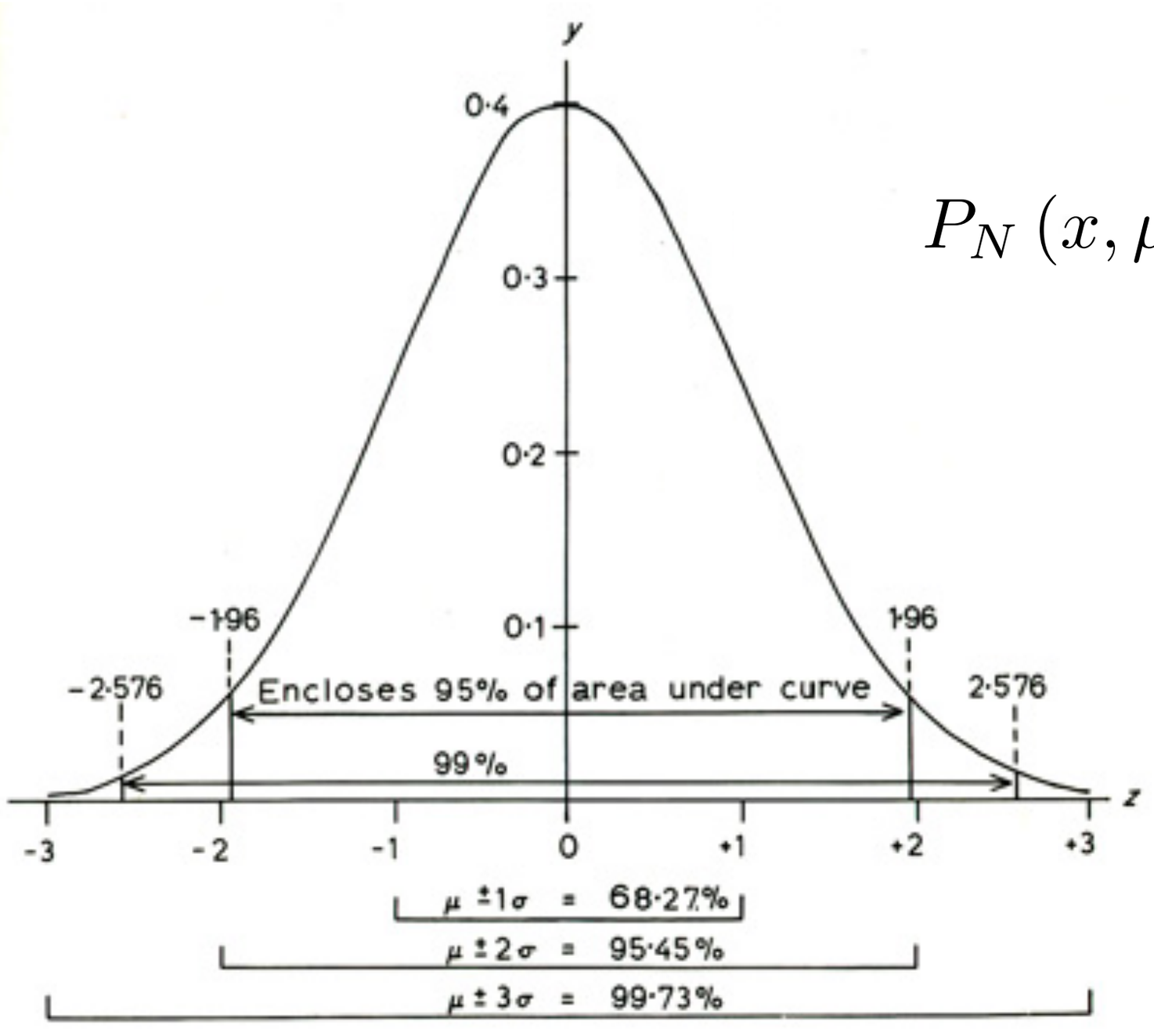
$$S_n \sim \mu + \frac{\mathcal{N}\left(0, \sigma^2\right)}{\sqrt{n}} \to SE = \frac{\sigma}{\sqrt{n}}$$

The estimation of the mean converges to the true mean with the square root of the number of samples

Central Limit Theorem



Gaussian Distribution



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

Experimental Measurements

- Experimental errors commonly assumed gaussian distributed

- Many experimental measurements are actually averages:

    - Instruments have a finite response time and the quantity of interest varies quickly over time

    - Stochastic Environmental factors

    - Etc

MLE - Fitting a theoretical function to experimental data

- In an experimental measurement, we expect (CLT) the experimental values to be normally distributed around the theoretical value with a certain variance. Mathematically, this means:

$$y - f\left(x\right) \approx \frac{1}{\sqrt{2\sigma^2}} \exp\left[-\frac{\left(y - f\left(x\right)\right)^2}{2\sigma^2}\right]$$

- where y are the experimental values and $f\left(x\right)$ the theoretical ones. The likelihood is then:

$$\mathcal{L} = -\frac{N}{2}\log\left[2\sigma^2\right] - \sum_i \left[\frac{\left(y_i - f\left(x_i\right)\right)^2}{2\sigma^2}\right]$$

- Where we see that to maximize the likelihood we must minimize the sum of squares

Least Squares Fitting

MLE - Linear Regression

- Let's say we want to fit a straight line to a set of points:

$$y = w \cdot x + b$$

- The Likelihood function then becomes:

$$\mathcal{L} = -\frac{N}{2}\log\left[2\sigma^2\right] - \sum_i \left[\frac{\left(y_i - w \cdot x_i - b\right)^2}{2\sigma^2}\right]$$

- With partial derivatives:

$$\frac{\partial \mathcal{L}}{\partial w} = \sum_i \left[2x_i\left(y_i - w \cdot x_i - b\right)\right]$$

$$\frac{\partial \mathcal{L}}{\partial b} = \sum_i \left[\left(y_i - w \cdot x_i - b\right)\right]$$

- Setting to zero and solving for $\hat{w}$ and $\hat{b}$:

$$\hat{w} = \frac{\sum_i \left(x_i - \langle x \rangle\right)\left(y_i - \langle y \rangle\right)}{\sum_i \left(x_i - \langle x \rangle\right)^2}$$

$$\hat{b} = \langle y \rangle - \hat{w}\langle x \rangle$$

MLE for Linear Regression

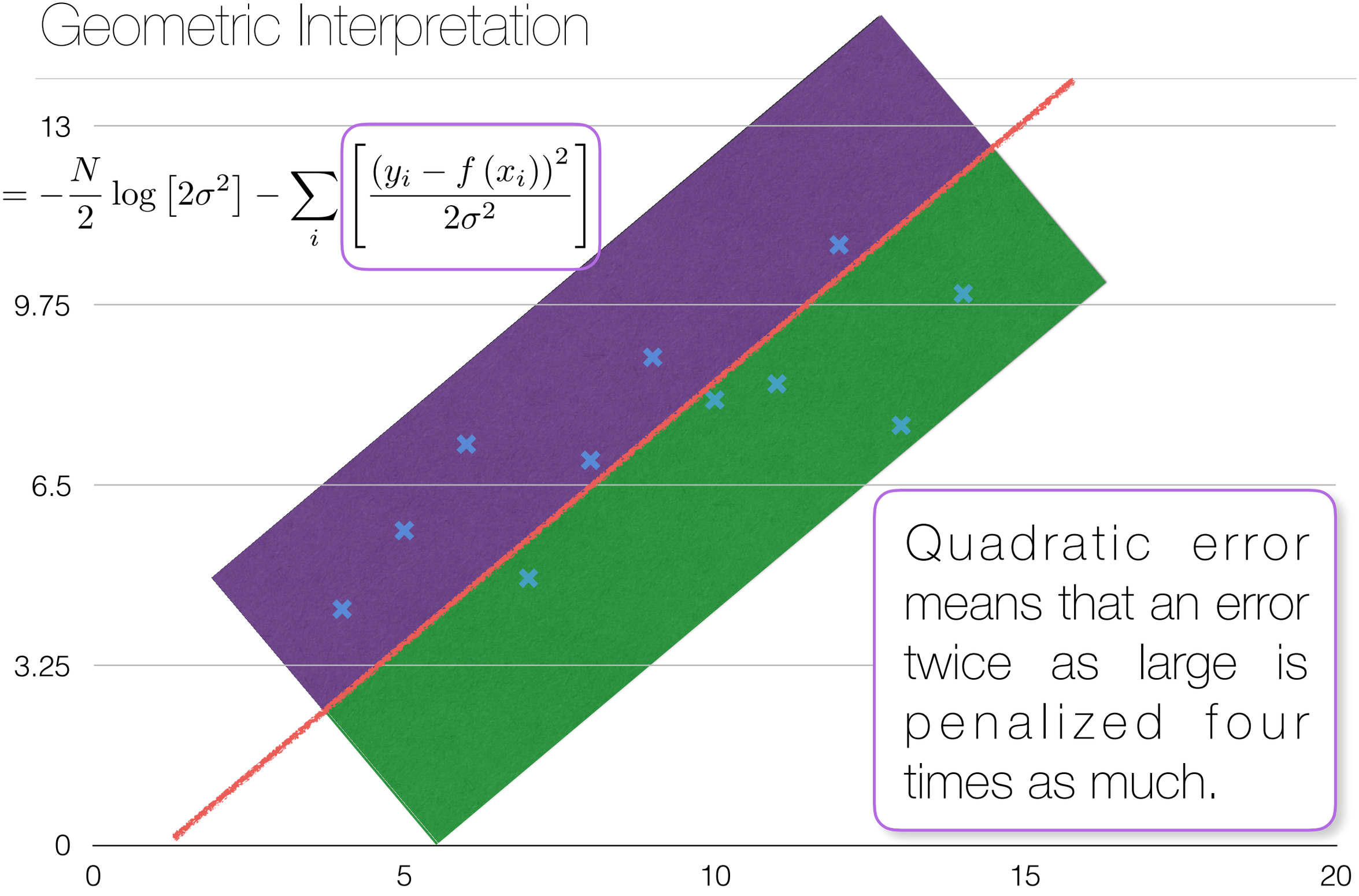
```python
from __future__ import print_function
import sys
import numpy as np
from scipy import optimize

data = np.loadtxt(sys.argv[1])

x = data.T[0]
y = data.T[1]

meanx = np.mean(x)
meany = np.mean(y)

w = np.sum((x-meanx)*(y-meany))/np.sum((x-meanx)**2)
b = meany-w*meanx

print(w, b)

#We can also optimize the Likelihood expression directly
def likelihood(w):
    global x, y
    sigma = 1.0
    w, b = w

    return np.sum((y-w*x-b)**2)/(2*sigma)

w, b = optimize.fmin_bfgs(likelihood, [1.0, 1.0])

print(w, b)
```

MLElinear.py

@bgoncalves

Geometric Interpretation

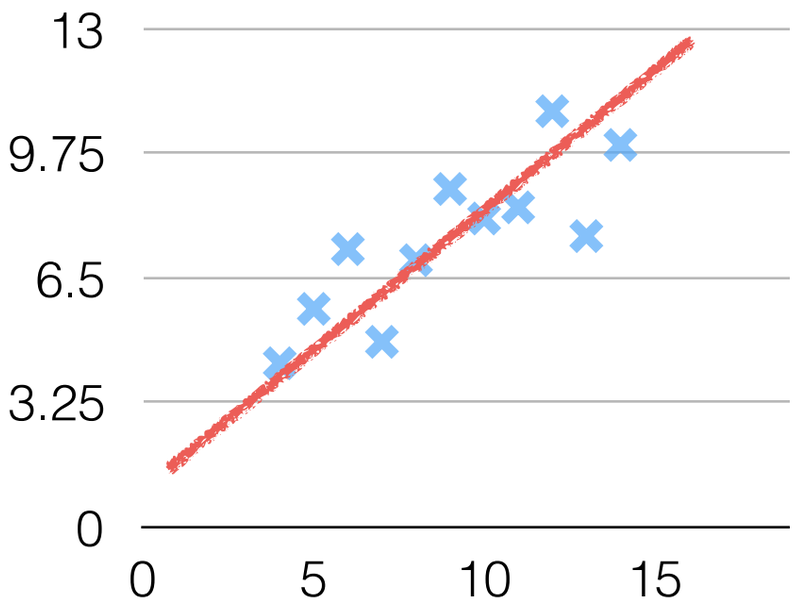
$$\mathcal{L} = -\frac{N}{2}\log\left[2\sigma^2\right] - \sum_i \left[\frac{\left(y_i - f\left(x_i\right)\right)^2}{2\sigma^2}\right]$$



Quadratic error means that an error twice as large is penalized four times as much.

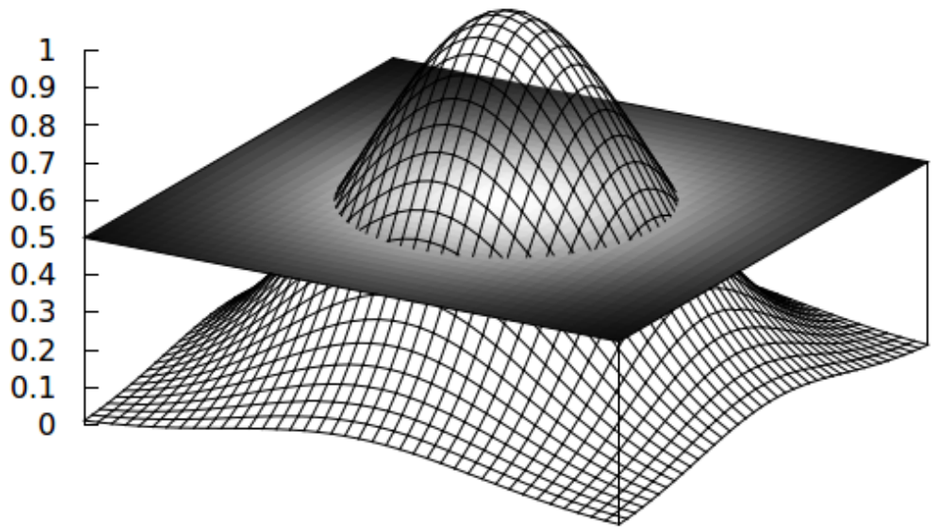
Geometric Interpretation

2D



$$y = \omega_0 + \omega_1 x_1$$

3D

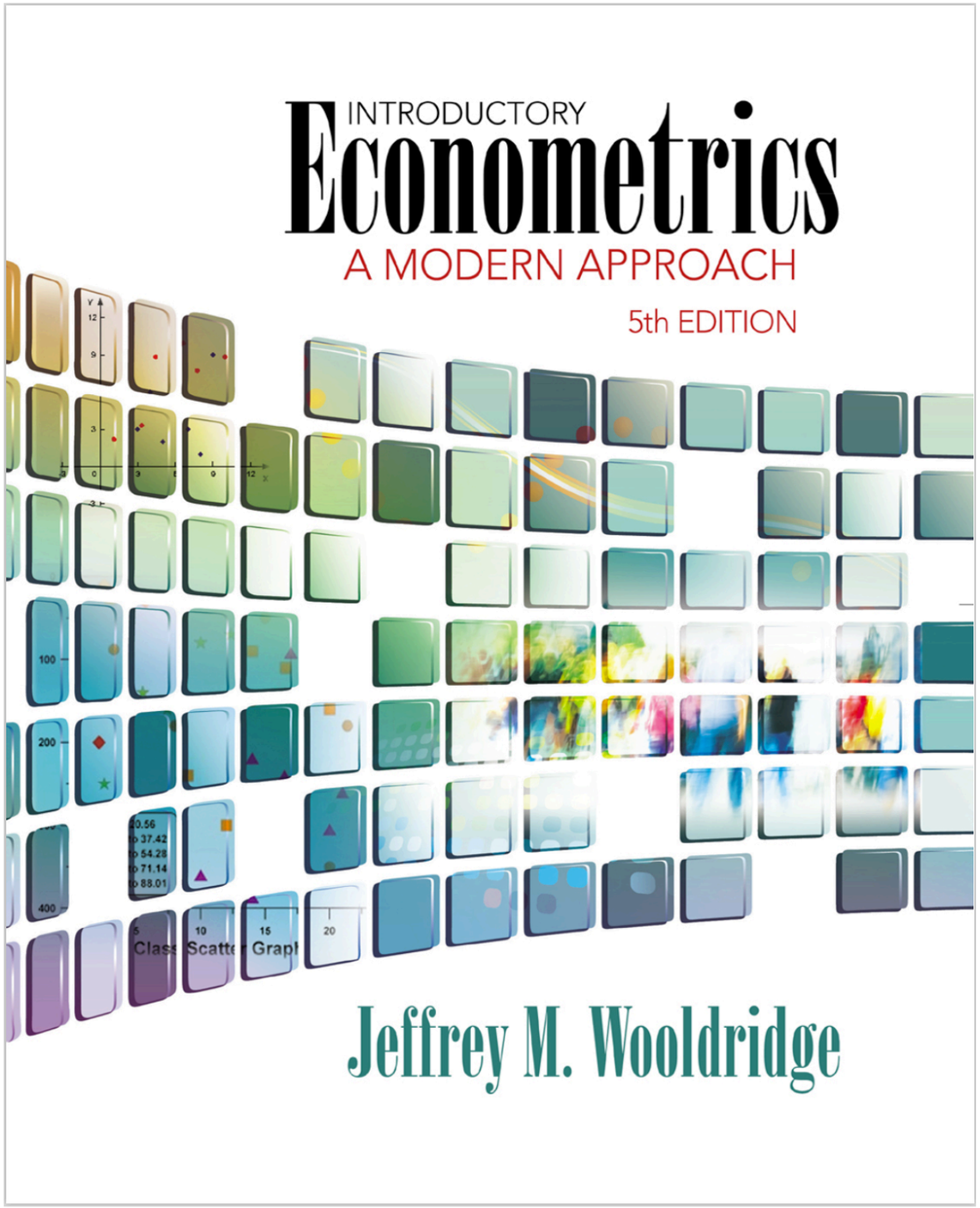


$$y = \omega_0 + \omega_1 x_1 + \omega_2 x_2$$

nD

$$y = \vec{\omega}^T \vec{x}$$

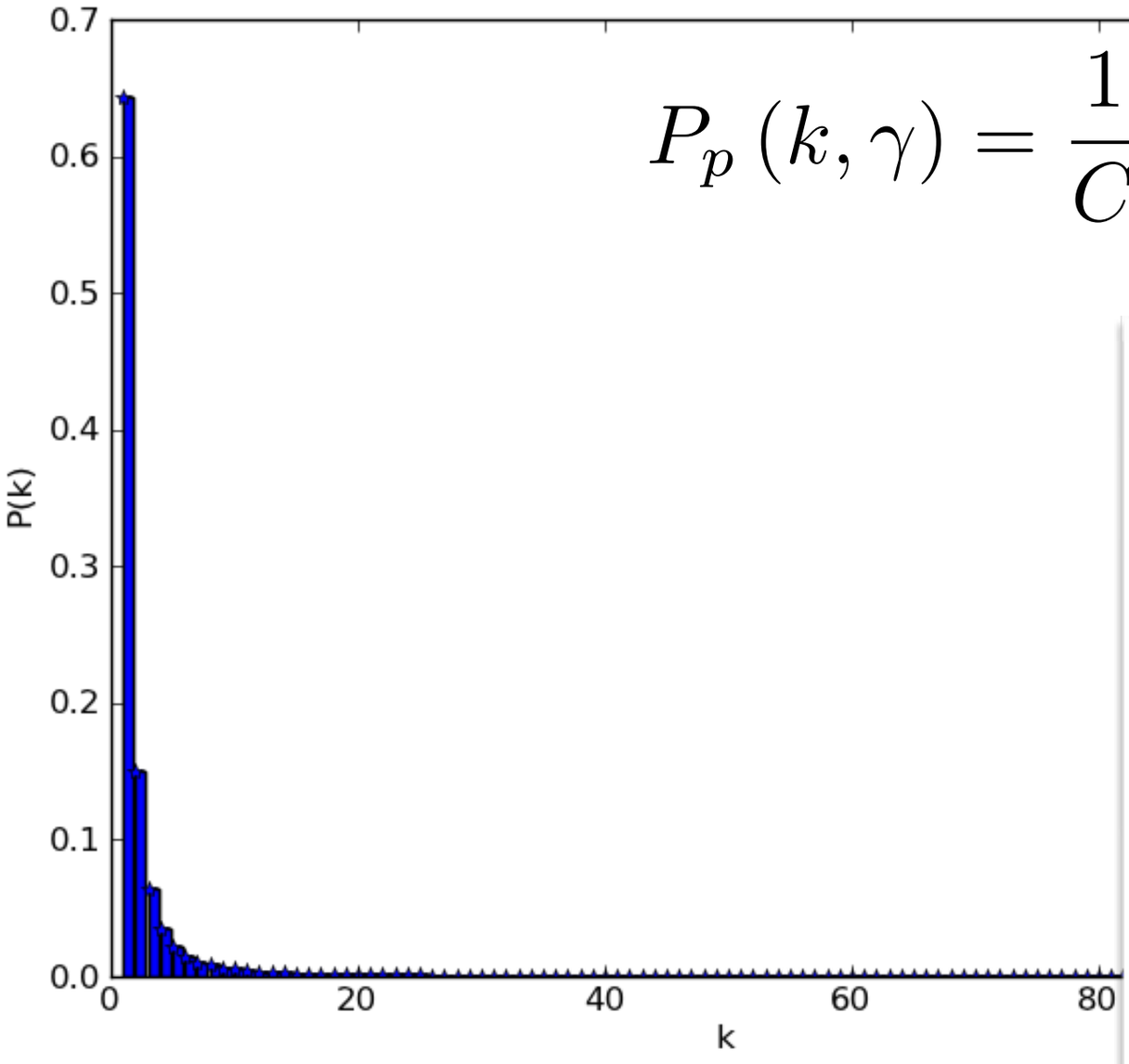
Multiple Regression...

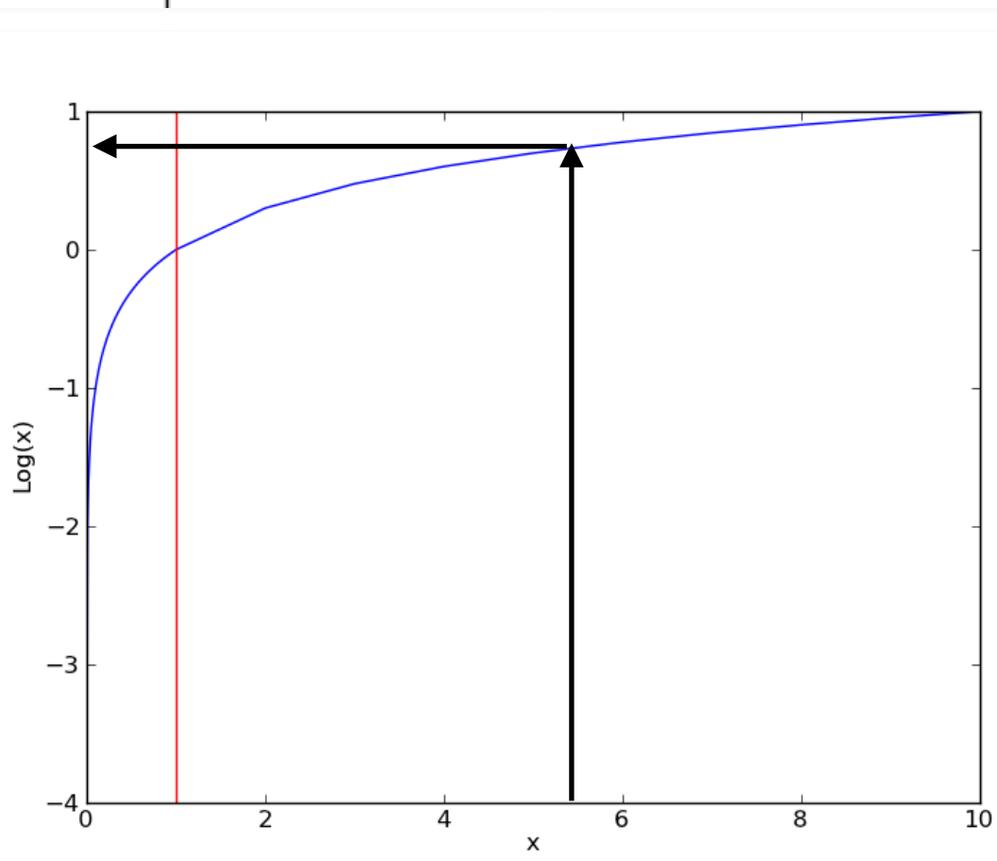


BRIEF CONTENTS

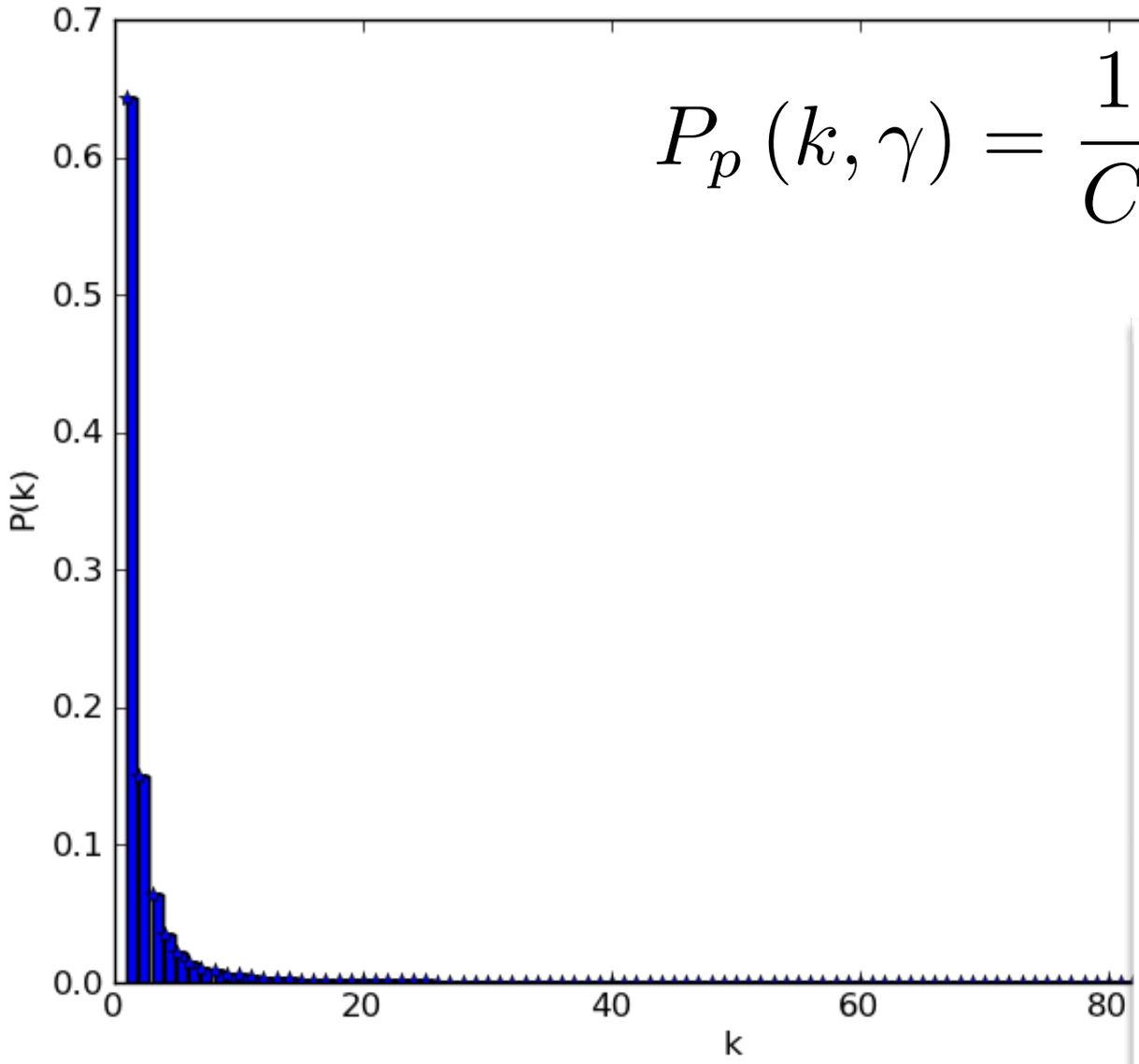
Broad-tailed distributions

$$P_p\left(k,\gamma\right) = \frac{1}{C}k^{-\gamma}$$

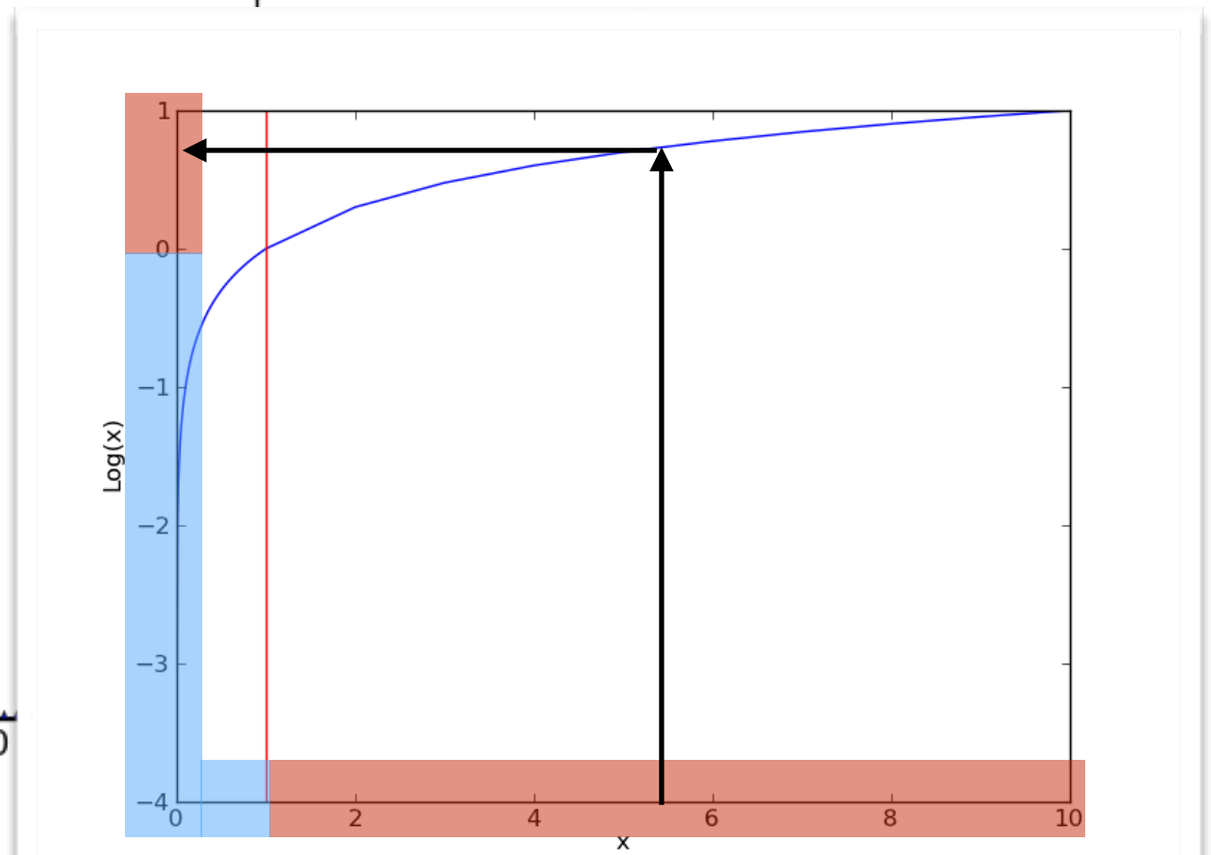




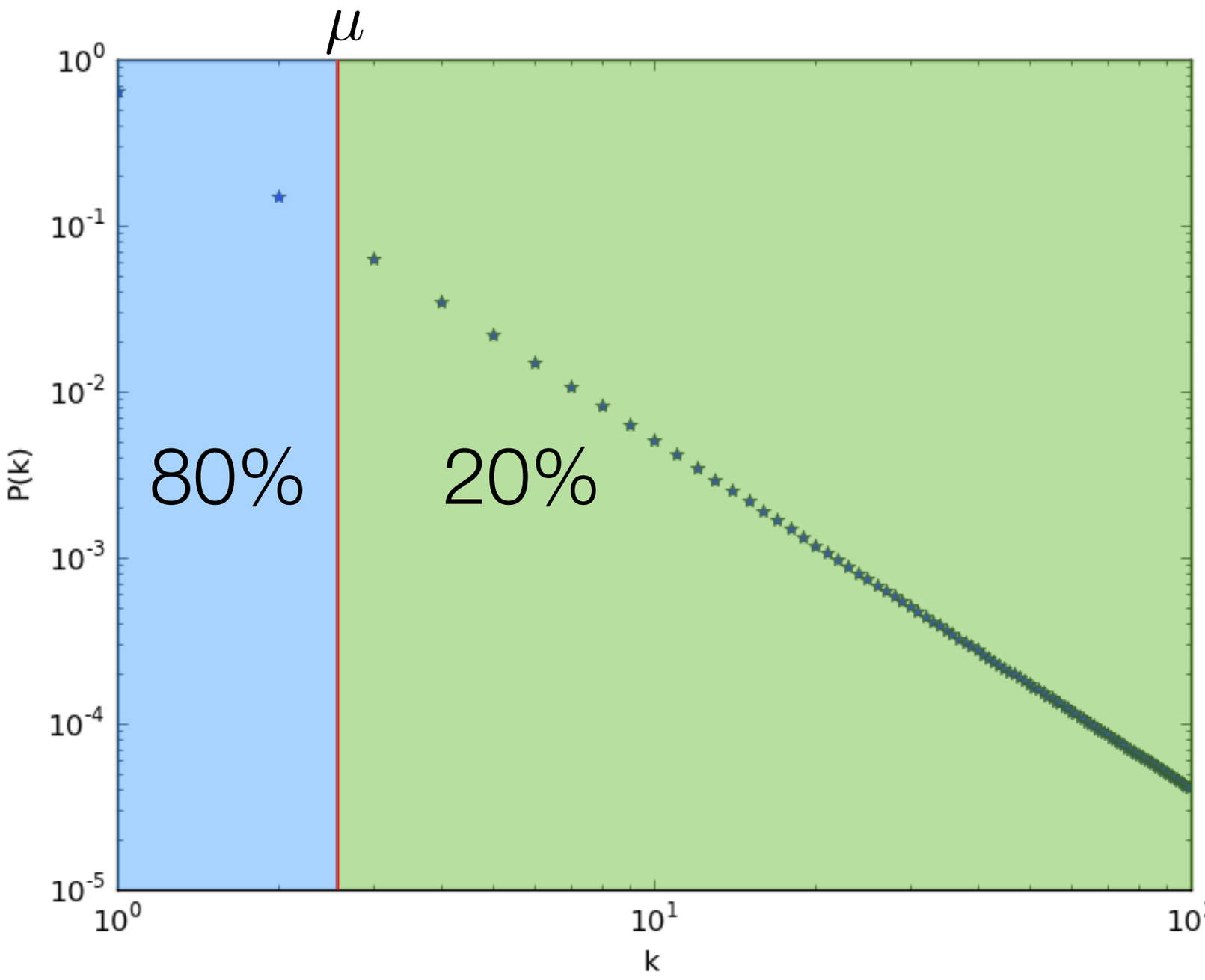
Broad-tailed distributions

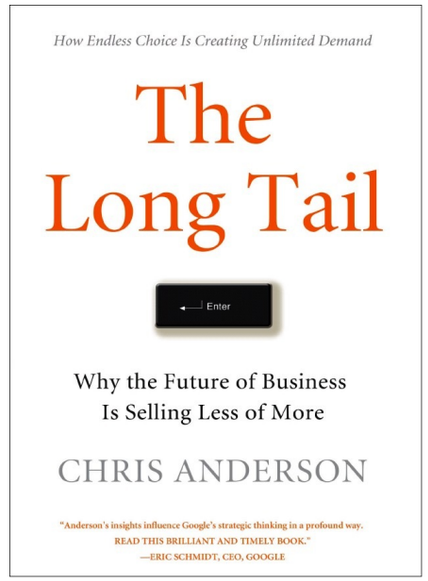


$$P_p\left(k,\gamma\right) = \frac{1}{C}k^{-\gamma}$$



(Almost) Everyone is below average!





MLE - Fitting a power-law to experimental data

SIAM Rev. 51, 661 (2009)

- We often find what look like power-law distributions in empirical data:

$$P(k) = \frac{\gamma - 1}{k_{min}} \left(\frac{k}{k_{min}} \right)^{-\gamma}$$

and we would like to find the right parameter values.

- The likelihood of any set of points is:

$$\mathcal{L} = \sum_i \log \left[\frac{\gamma - 1}{k_{min}} \left(\frac{k_i}{k_{min}} \right)^{-\gamma} \right]$$

- And maximizing, we find:

$$\gamma = 1 + n \left[\sum_i \log \left(\frac{k_i}{k_{min}} \right) \right]^{-1}$$

- with a standard error of:

$$SE = \frac{\gamma - 1}{\sqrt{n}}$$

MLE Coin Flips

- Biased coin with unknown probability of heads (p)
- In a sequence of N flips, the likelihood of N_h heads and N_t=N-N tails is:

$$\mathcal{L} = \log\left[p^{N_h}\left(1-p\right)^{N-N_h}\right]$$

Ignoring the combinatorial factor!

- or simply:

$$\mathcal{L} = N_h \log\left[p\right] + \left(N-N_h\right)\log\left[1-p\right]$$

- Taking the derivative:

$$\frac{\partial \mathcal{L}}{\partial p} = \frac{N_h}{p} - \frac{N-N_h}{1-p}$$

- Setting to zero and solving for p:

$$p = \frac{N_h}{N}$$

Binomial Distribution

- The probability of getting k successes with n trials of probability p (k heads in n coin flips):

$$P_B\left(k,n,p\right) = \frac{n!}{k!\left(n-k\right)!}p^k\left(1-p\right)^{n-k}$$

- The mean value is:

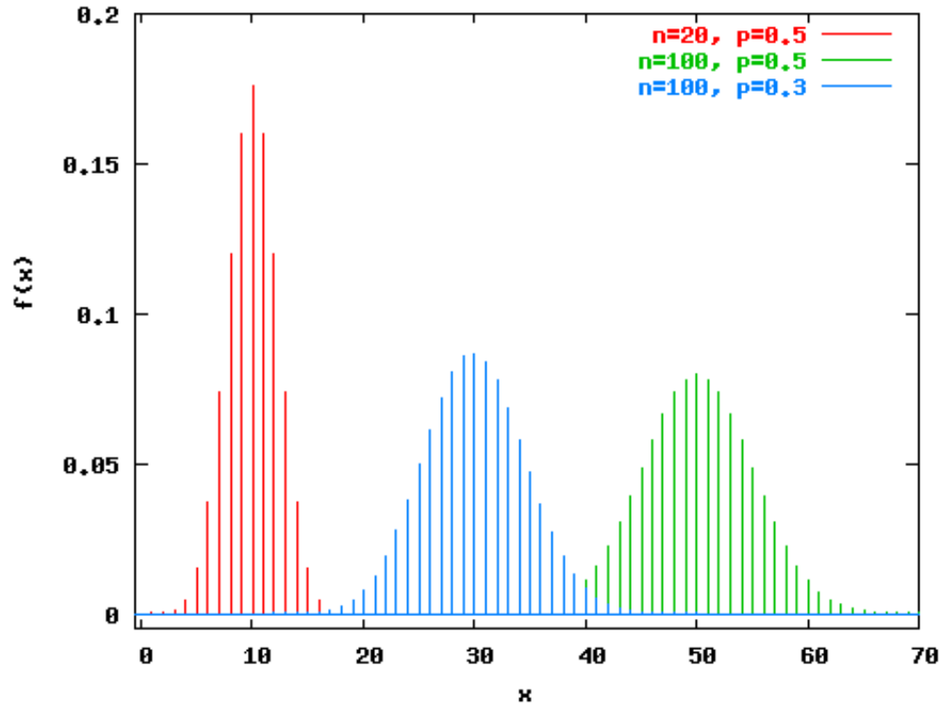
$$\mu = np$$

- and the variance:

$$\sigma = np\left(1-p\right)$$

- and for sufficiently large n:

$$P_B\left(k,n,p\right) \sim P_N\left(np, np\left(1-p\right)\right)$$



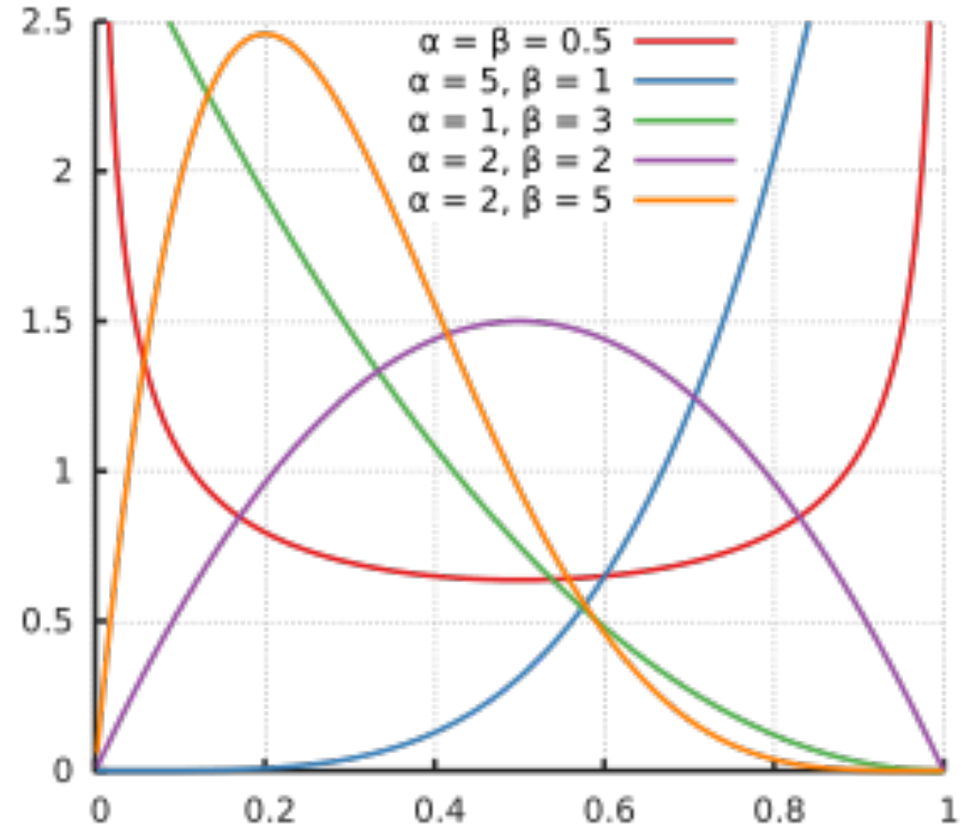
(Beta Distribution)

$$P_B\left(k,n,p\right)=\frac{n!}{k!\left(n-k\right)!}p^k\left(1-p\right)^{n-k}$$

- Related to Binomial and has a very similar form:

$$P_\beta\left(x,\alpha,\beta\right)=\frac{\Gamma\left(\alpha+\beta\right)}{\Gamma\left(\alpha\right)\Gamma\left(\beta\right)}x^{\alpha-1}\left(1-x\right)^{\beta-1}$$

- with $x\in[0,1]$ and $\alpha,\beta>0$.



- $\Gamma\left(a\right)$ is the continuous extension to $a!$

- The mean is:

$$\mu=\frac{\alpha}{\alpha+\beta}$$

- And the variance:

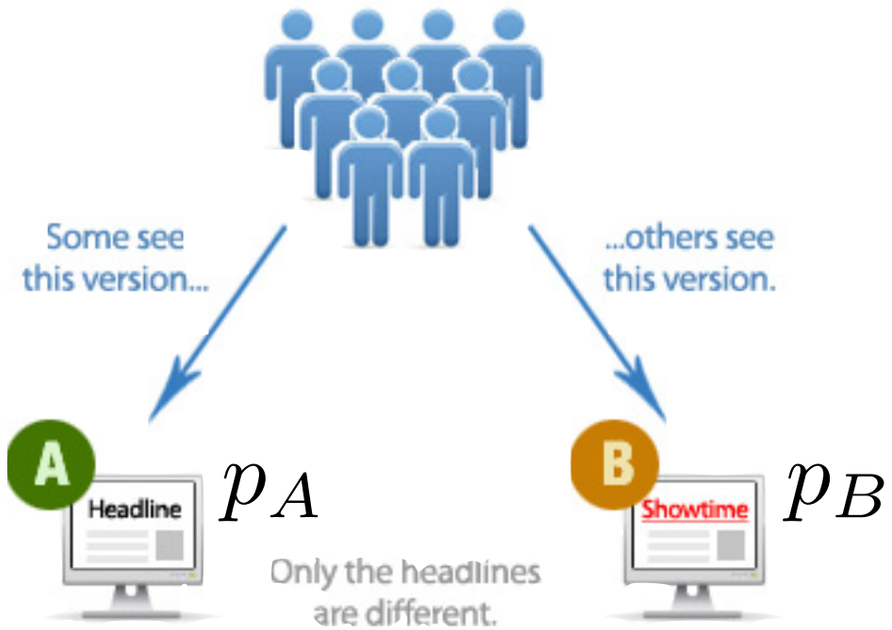
$$\sigma=\frac{\alpha\beta}{\left(\alpha+\beta\right)^2\left(\alpha+\beta+1\right)}$$

A/B Testing

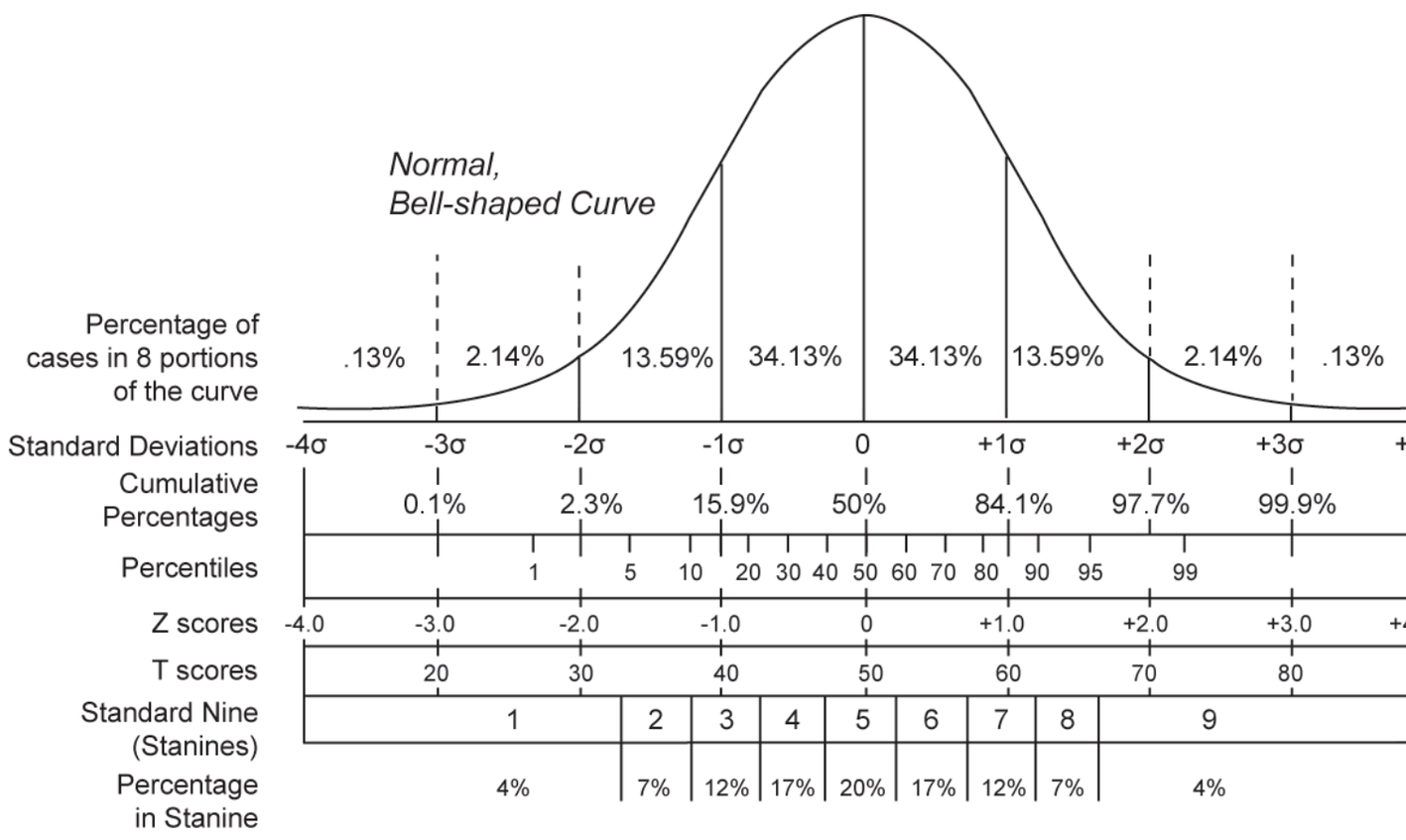
- Divide users into two groups A and B
- Measure some metric for each group (conversion probability, for example)

$$p_A \qquad p_B$$

A/B Testing



p_A p_B



- If conversion is a binomial process, then:
- Standard Error

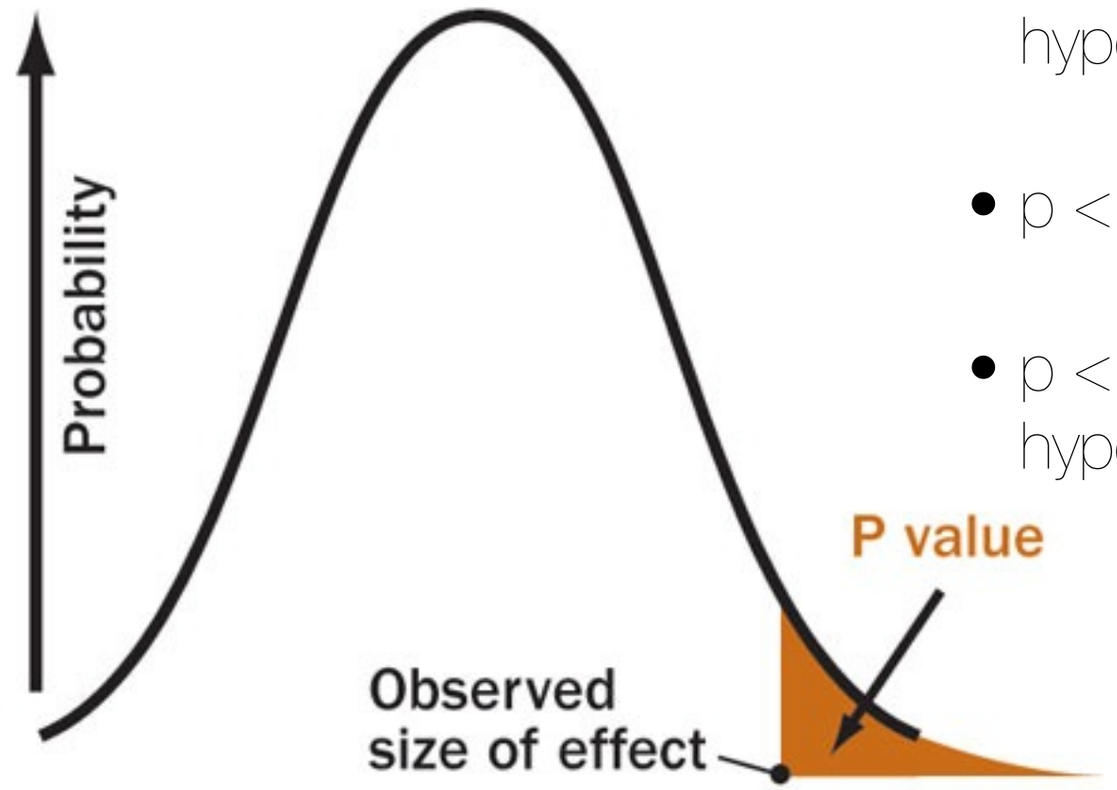
$$SE = \sqrt{\frac{p\left(1-p\right)}{N}}$$

- Z score

$$Z = \frac{p_A - p_B}{\sqrt{SE_A^2 + SE_B^2}}$$

p-value

- Calculate the probability of an event more extreme that the observation under the "null hypothesis"



- $p < 0.05$ Moderate evidence agains null-hypothesis
- $p < 0.01$ Strong evidence against null-hypothesis
- $p < 0.001$ Very strong evidence agains the null-hypothesis

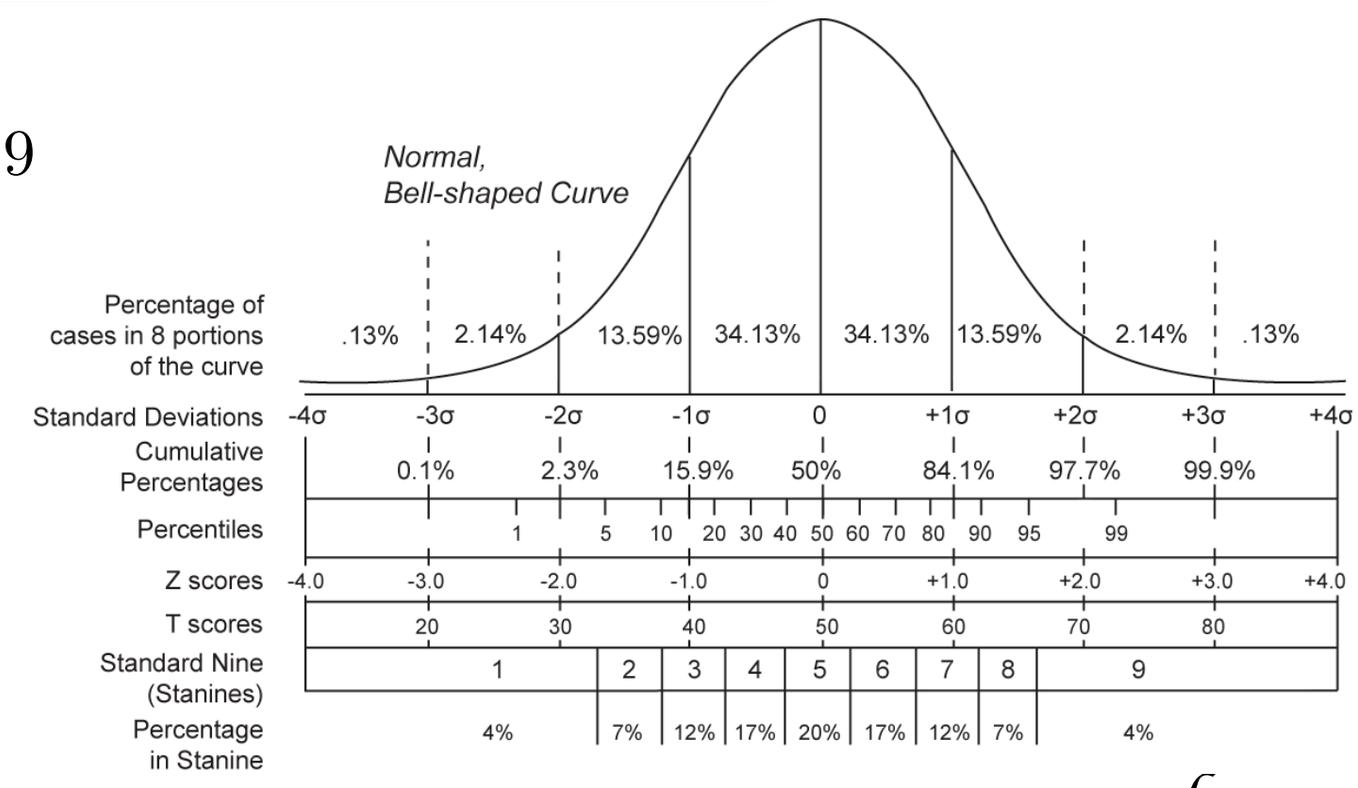
- The smaller the p-value the better.

Berkeley Discrimination Case Part I

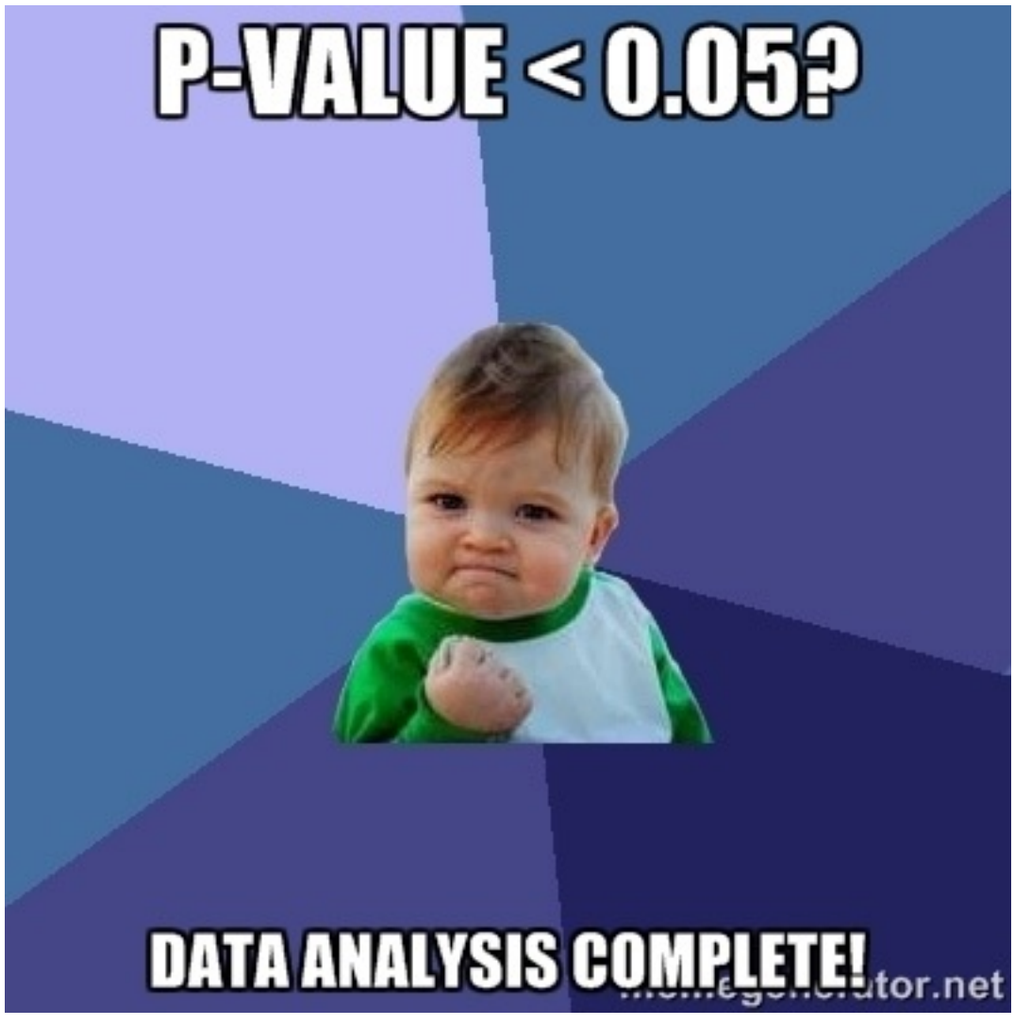
	Candidates	Acceptance Rate	SE
Men	8442	0.44	5.4×10^{-3}
Women	4321	0.35	7.2×10^{-3}

$$Z = \frac{p_A - p_B}{\sqrt{SE_A^2 + SE_B^2}} = 9.9$$

$$p \approx 10^{-23}$$

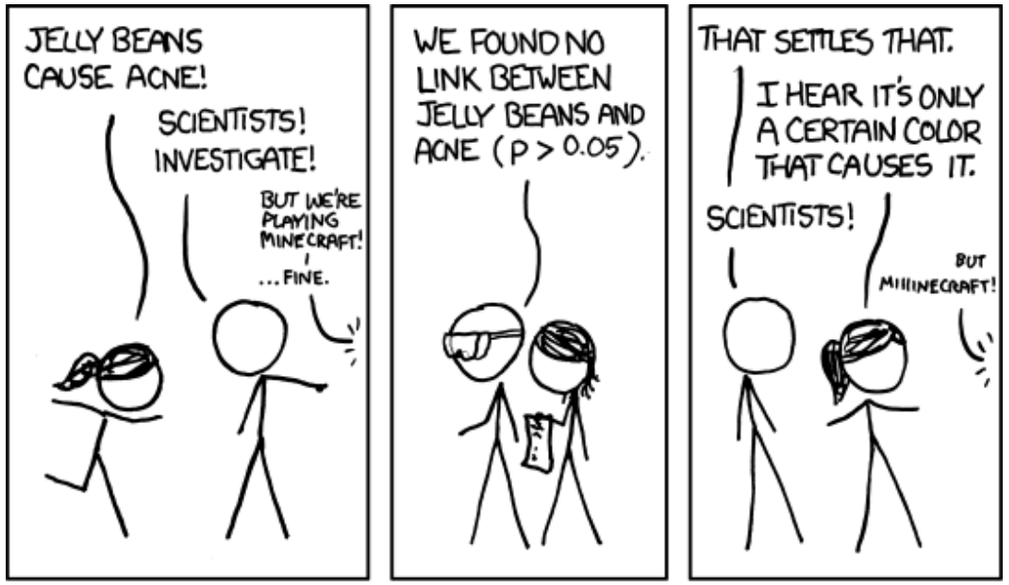


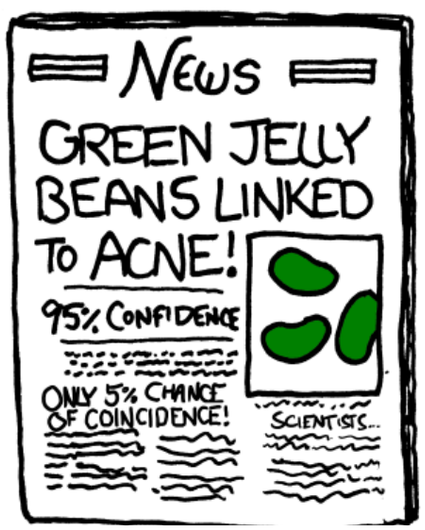
p-value

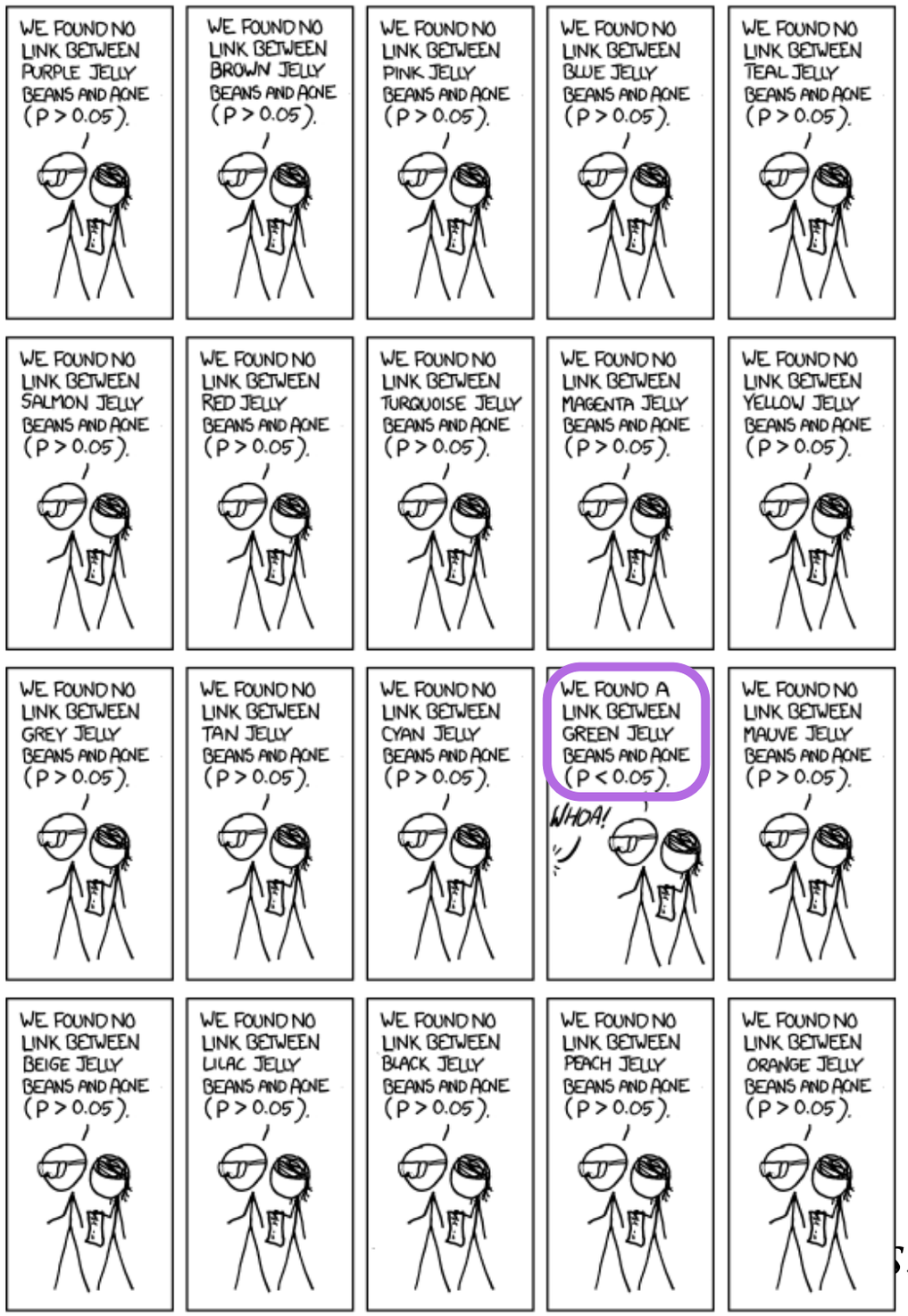


"Statistical significance does not imply scientific significance"

(Bonferoni Correction)







(Bonferoni Correction)

- You can think of p as the probability of observing a result as extreme by chance. With n comparisons, this probability becomes:

$$p_n = 1 - (1 - p)^n$$

  which quickly goes to 1 as n increases.

- However, by replacing p by p/n for each individual comparison, we obtain:

$$p_n = 1 - \left(1 - \frac{p}{n}\right)^n$$

- and for sufficiently large n:

$$p_n \approx 1 - e^{-p} \approx p$$

- allowing us to keep the probability of false results arbitrarily low even with arbitrarily large numbers of comparisons.

The Simpsons

@bgoncalves

Simpson's Paradox

Science 187, 398 (1975)

	Candidates	Acceptance Rate
Men	8442	0.44
Women	4321	0.35

Berkeley Discrimination Case Part II:
The statisticians strike back.

Simpson's Paradox

Science 187, 398 (1975)

	Candidates	Acceptance Rate
Men	8442	0.44
Women	4321	0.35

"aggregated data can appear
to reverse important trends
in the numbers being combined"
WSJ, Dec 2, 2009

Dept	Men		Women	
	Candidates	Acceptance	Candidates	Acceptance
A	825	0.62	108	**0.82**
B	560	0.63	25	**0.68**
C	325	0.37	594	0.34
D	417	0.33	375	0.35
E	191	**0.28**	393	0.24
F	272	**0.06**	341	0.07
	2590	**0.46**	**1835**	**0.30**