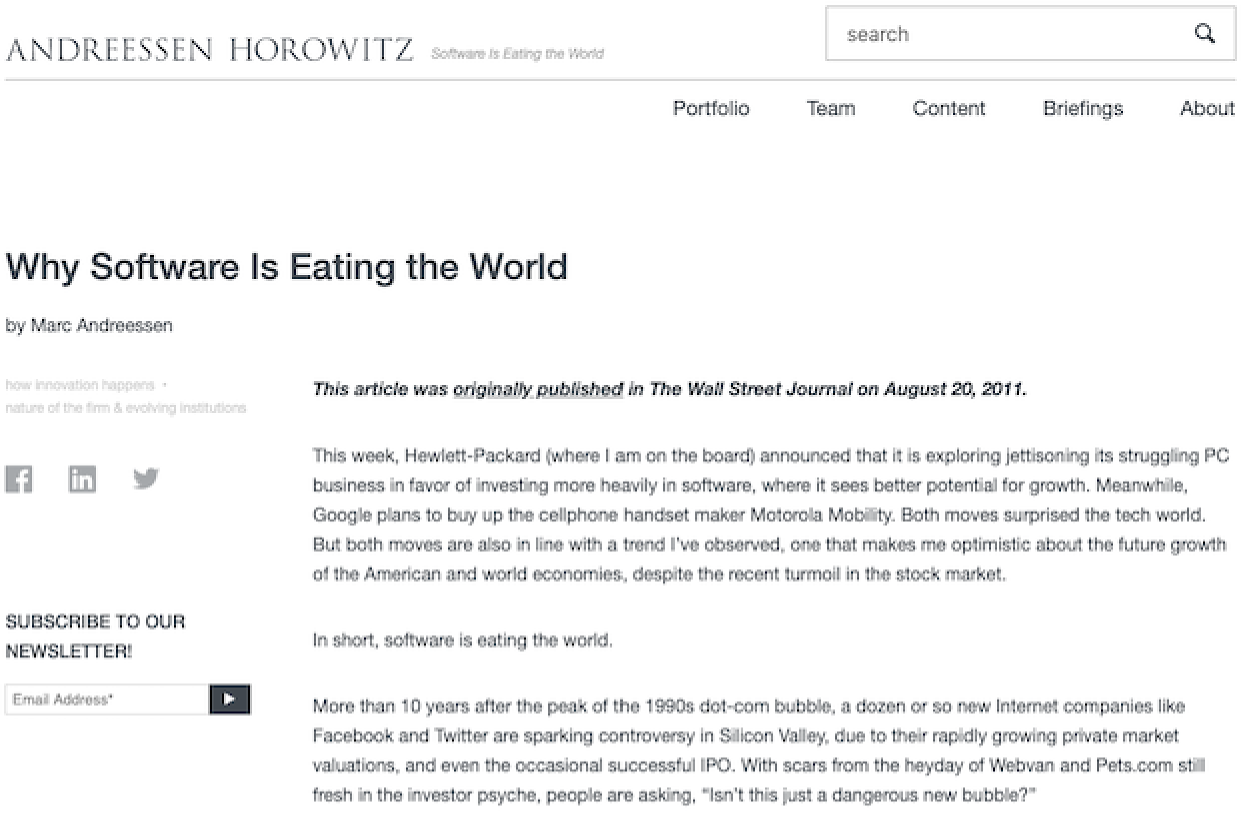
Introduction

CS534 - Machine Learning

Yubin Park, PhD


ANDREESSEN HOROWITZ Software Is Eating the World

search

Portfolio Team Content Briefings About

Why Software Is Eating the World

by Marc Andreessen

how innovation happens · nature of the firm & evolving institutions

SUBSCRIBE TO OUR NEWSLETTER!

Email Address*

This article was originally published in The Wall Street Journal on August 20, 2011.

This week, Hewlett-Packard (where I am on the board) announced that it is exploring jettisoning its struggling PC business in favor of investing more heavily in software, where it sees better potential for growth. Meanwhile, Google plans to buy up the cellphone handset maker Motorola Mobility. Both moves surprised the tech world. But both moves are also in line with a trend I've observed, one that makes me optimistic about the future growth of the American and world economies, despite the recent turmoil in the stock market.

In short, software is eating the world.

More than 10 years after the peak of the 1990s dot-com bubble, a dozen or so new Internet companies like Facebook and Twitter are sparking controversy in Silicon Valley, due to their rapidly growing private market valuations, and even the occasional successful IPO. With scars from the heyday of Webvan and Pets.com still fresh in the investor psyche, people are asking, "Isn't this just a dangerous new bubble?"

https://a16z.com/2011/08/20/why-software-is-eating-the-world/

What he meant by then...

Transforming the retail industry

amazon

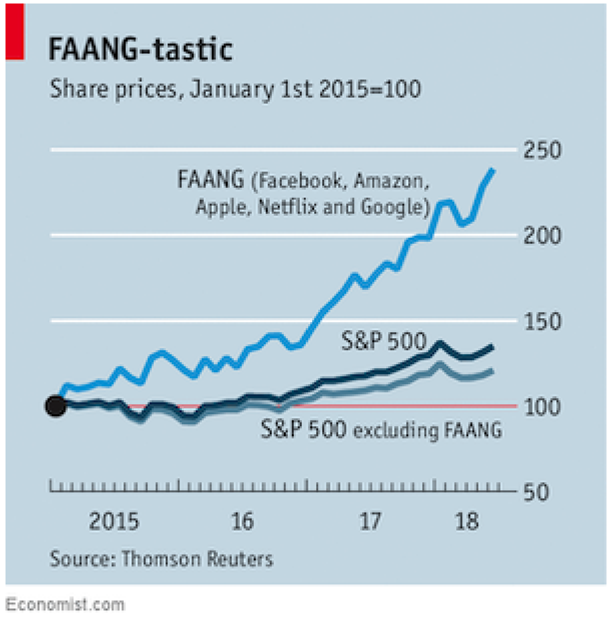
Transforming the entertainment industry

NETFLIX

Transforming the advertisement industry

Google

Seems software has been really eating the world...



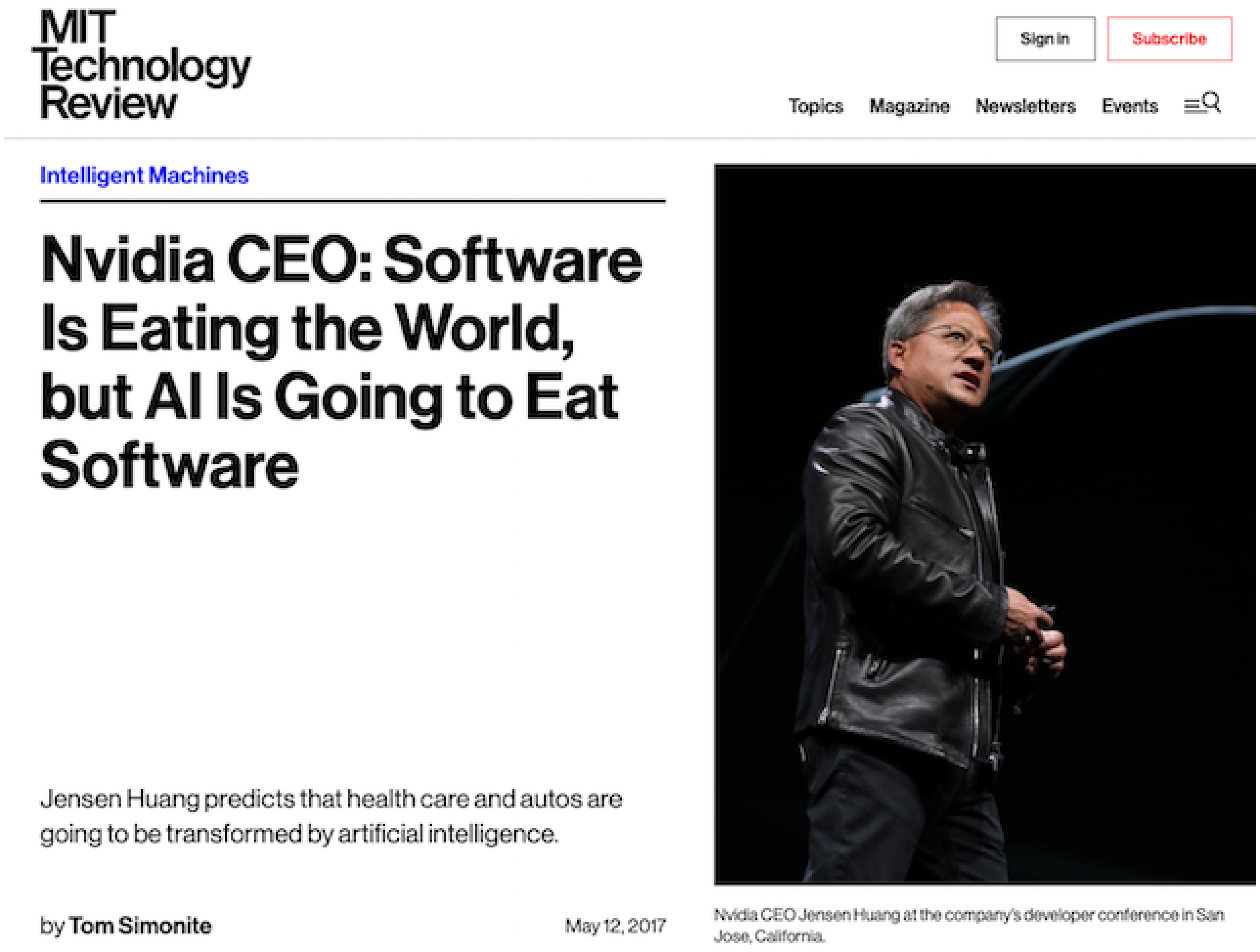
https://www.economist.com/finance-and-economics/2018/06/23/most-stockmarket-returns-come-from-a-tiny-fraction-of-shares

How did they become so successful?

Many reasons maybe.

However, one commonality remarkably stands out:

Their love for **data**



MIT Technology Review

Intelligent Machines

Nvidia CEO: Software Is Eating the World, but AI Is Going to Eat Software

Jensen Huang predicts that health care and autos are going to be transformed by artificial intelligence.

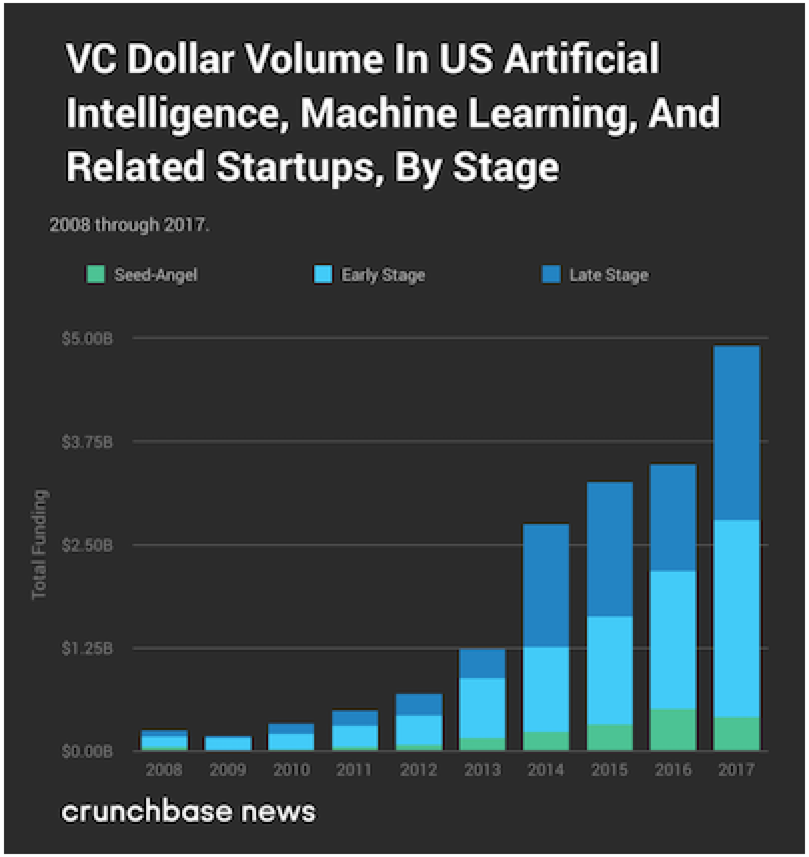
by Tom Simonite

May 12, 2017

Nvidia CEO Jensen Huang at the company's developer conference in San Jose, California.

https://www.technologyreview.com/s/607831/nvidia-ceo-software-is-eating-the-world-but-ai-is-going-to-eat-software/

Money is flowing into AI



https://news.crunchbase.com/news/venture-funding-ai-machine-learning-levels-off-tech-matures/

Machine Learning?

My Definition

Machine learning is a study of **algorithmic and statistical** techniques to perform **outcome-measurable** tasks with **minimally** using explicit instructions.

Machine learning is one of the key engines for artificial intelligence to work.

Just FYI, what Wikipedia says

> Machine learning (ML) is the scientific study of algorithms and statistical models that computer systems use in order to perform a specific task effectively without using explicit instructions, relying on patterns and inference instead.

About Me

Currently, I am

- Adjunct Professor at Emory University
- Principal Scientist (part-time) at WithMe Health, Inc.
- Principal at Bonsai Research, LLC.
- Advisor at Tensility Venture Partners

Most recently, I worked for 7-Eleven, Inc. leading a team of data scientists to automate/analyze their inventory management and consumer behaviors. Prior to that, I worked for Evolent Health, Inc. as Managing Director of AI Solutions. I joined Evolent through the acquisition of my start-up, a healthcare analytics start-up, Accordion Health, Inc. I received my Ph.D. degree from the University of Texas at Austin by studying various algorithms in machine learning and their healthcare applications.

Course Schedule & Logisitcs

Machine learning is a rapidly evoling research discipline.

We will cover:

- Time-tested models and algorithms
  - Linear models
  - Decision trees
  - Support vector machines
  - Boosting and bagging
  - (and some others)
- Evaluation strategies and methodologies
- Recently popularized models and algorithms
  - Deep learning
  - (and real world applications)

The course schedule and course logistics are available on the course website: https://github.com/yubin-park/cs534

Things Not Covered

- Bayesian Models
- Unsupervised Learning
  - Clustering
  - Topic Modeling
- Time Series Models
- Learning Theory
- Decision Theory
- Reinforcement Learning

If you want to learn about the topics not covered in this course, please take a look at these courses: CS570 Data Mining and CS557 Artificial Intelligence.

Homework #0

- https://github.com/yubin-park/cs534/blob/master/homework/hw0.md
- Should take an hour to three hours (max)
- This is a Computer Science course, so you need to "code"
- The homework is to test your knowledge with reading data files, slicing the data, visualizing, and find any patterns
- Due 9/2 at 11:59 PM — NO LATE DAYS ALLOWED
- Meant to be an assessment of your preparation — all students MUST submit this
- Strongly encourage you to drop this course if any of these conditions apply:
    - Difficulty completing it in less than 6 hours
    - Difficulty with more than 1 problem

Statistics Crash Course

- Distributions
- Summary Statistics
- Hypothesis Testing

Let's take a look at some data

Boston House-Price Data

The Boston house-price data of Harrison, D. and Rubinfeld, D.L. 'Hedonic prices and the demand for clean air', J. Environ. Economics & Management, vol.5, 81-102, 1978.

The original data is archived at http://lib.stat.cmu.edu/datasets/boston

Scikit-learn provides the data through one of its modules:

```python
from sklearn.datasets import load_boston
import pandas as pd

data = load_boston()
X = data.data
y = data.target
df = pd.DataFrame(X, columns = data.feature_names)
df["MEDV"] = y
n, m = df.shape # n: number of rows, m: number of columns
```

https://scikit-learn.org/stable/datasets/index.html#boston-dataset and https://pandas.pydata.org/

To see how the data looks like:

```
print(df.head())
```

```
      CRIM    ZN  INDUS  CHAS    NOX  ...    TAX  PTRATIO       B  LSTAT   MEDV
0  0.00632  18.0   2.31   0.0  0.538  ...  296.0     15.3  396.90   4.98   24.0
1  0.02731   0.0   7.07   0.0  0.469  ...  242.0     17.8  396.90   9.14   21.6
2  0.02729   0.0   7.07   0.0  0.469  ...  242.0     17.8  392.83   4.03   34.7
3  0.03237   0.0   2.18   0.0  0.458  ...  222.0     18.7  394.63   2.94   33.4
4  0.06905   0.0   2.18   0.0  0.458  ...  222.0     18.7  396.90   5.33   36.2
```

If you want to take a look at a specific column e.g. `RM` which stands for the average number of rooms per dwelling

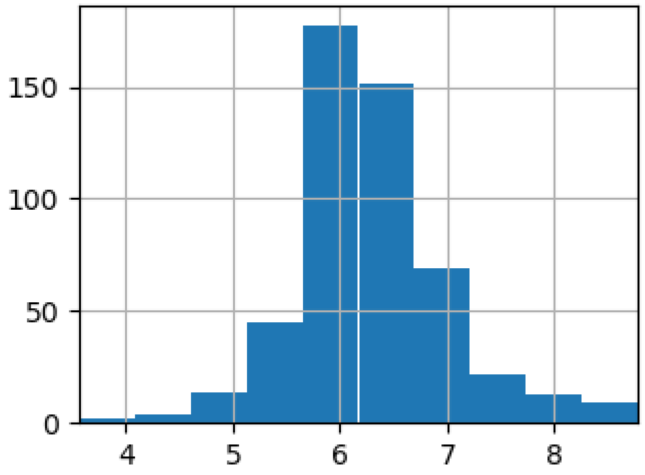
```
print(df["RM"].values[:100])
```

```
array([6.575 6.421 7.185 6.998 7.147 6.43  6.012 6.172 5.631 6.004 6.377 6.009
       5.889 5.949 6.096 5.834 5.935 5.99  5.456 5.727 5.57  5.965 6.142 5.813
       5.924 5.599 5.813 6.047 6.495 6.674 5.713 6.072 5.95  5.701 6.096 5.933
       5.841 5.85  5.966 6.595 7.024 6.77  6.169 6.211 6.069 5.682 5.786 6.03
       5.399 5.602 5.963 6.115 6.511 5.998 5.888 7.249 6.383 6.816 6.145 5.927
       5.741 5.966 6.456 6.762 7.104 6.29  5.787 5.878 5.594 5.885 6.417 5.961
       6.065 6.245 6.273 6.286 6.279 6.14  6.232 5.874 6.727 6.619 6.302 6.167
       6.389 6.63  6.015 6.121 7.007 7.079 6.417 6.405 6.442 6.211 6.249 6.625
       6.163 8.069 7.82  7.416])
```

Not so easy to understand a long array of numbers. Another way is:

```
col = "RM" # the number of rooms
xmin = df[col].min()
xmax = df[col].max()
figsize = (3.6, 2.7)

ax = df[col].hist(bins=10, figsize=figsize) # histogram
ax.set_xlim(xmin, xmax) # setting the range of x-axis
fig = ax.get_figure()
fig.savefig("img/histogram_{}.png".format(col))
ax.clear()
```

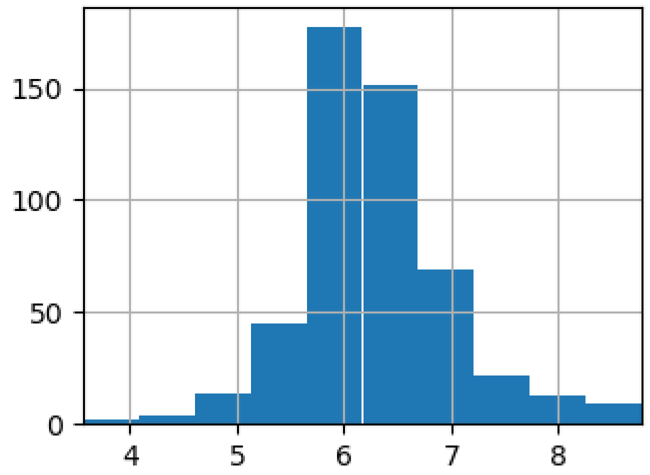


https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.hist.html

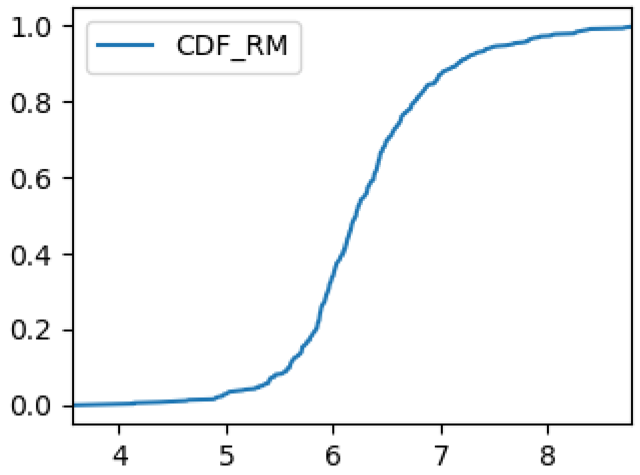
Visualizing a cumulative index

```
col_cdf = "CDF_{}".format(col) # a new variable name
n, m = df.shape # n: number of rows, m: number of columns
df = df.sort_values(by=[col])
df = df.reset_index() # reindex from the smallest RM to largest RM
df[col_cdf] = df.index / n # normalize by n
ax = df.plot.line(x=col, y=col_cdf, figsize=(3.6, 2.7))
fig = ax.get_figure()
fig.savefig("img/cdf_{}.png".format(col))
ax.clear()
```

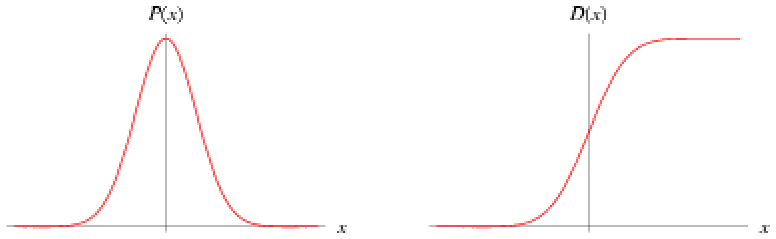
histogram



cumulative index



Do you find any similarities?



Note that **only** two parameters are needed to characterize the normal distribution: mean (`mu`) and variance (`sigma2`).

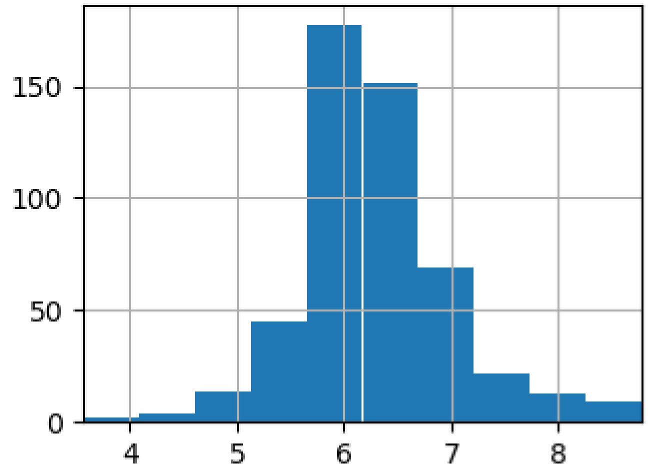
http://mathworld.wolfram.com/NormalDistribution.html

Simulating the number of rooms in the data.

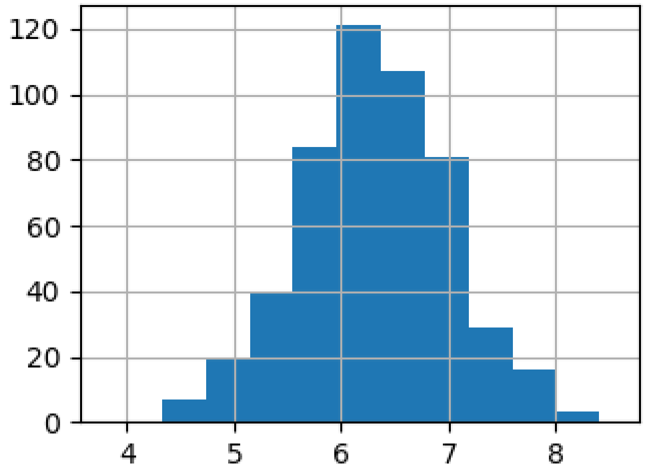
```
mu = np.mean(df[col]) # mean
sigma2 = np.var(df[col]) # variance
sigma = np.sqrt(sigma2) # standard deviation

col_simul = "{}_simul".format(col)
df[col_simul] = sigma * np.random.randn(n) + mu # random samples
ax = df[col_simul].hist(bins=10, figsize=(3.6,2.7))
ax.set_xlim(xmin, xmax)
fig = ax.get_figure()
fig.savefig("img/histogram_{}.png".format(col_simul))
ax.clear()
```

actual



simulated



Why do we calculate mean and variance?

Because these statistics can "summarize" the data.

They are called **summary statistics**.

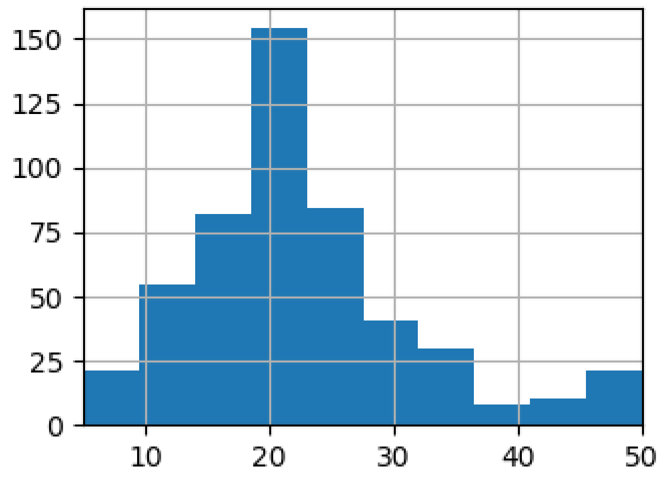
https://en.wikipedia.org/wiki/Summary_statistics

What about other variables?

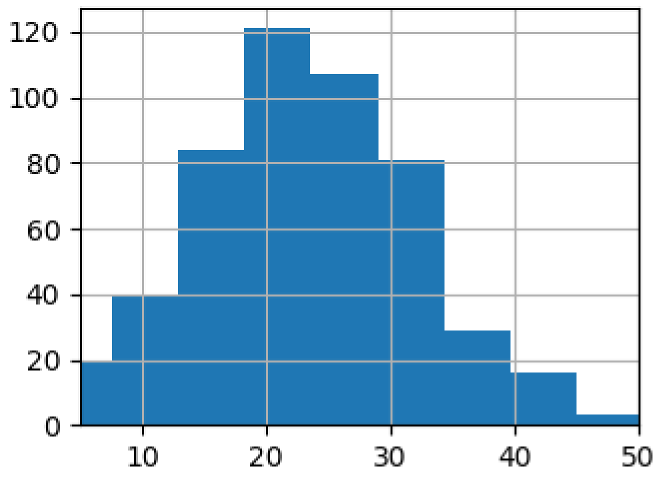
MEDV: Median value of owner-occupied homes in $1000's

```
col = "MEDV" # then repeat the same process
```

actual



simulated



The MEDV variable seems like a mixture of two normal distributions. Thus, the simulated histogram does not look similar with the original as well as in the RM variable. Note that the normality assumption does not hold always.

Your friend, Bob, says

"Hey, I am a real estate expert.

Houses near a river, in this case Charles River,

are more expensive than the others.

You should put that in your model."

How would you know Bob's statement is true?

On the other hand,

I think the prices are not affected by the river.

I want to prove that I am right.

One approach would be to calculate

the sample means of the two groups

and see if they are the same? i.e.

$$\frac{\sum x_i}{n} = \frac{\sum z_j}{m}?$$

where x_i and z_j represent house prices

with/without river nearby respectively.

Unfortunately, in this setup,

I would look **almost always wrong** i.e. $\Pr(\frac{\sum x_i}{n} \neq \frac{\sum z_j}{m}) \approx 1$

Note that we are dealing with **continuous random variables**.

Instead, what we can say is:

"If I am right, their sample means should be close enough."

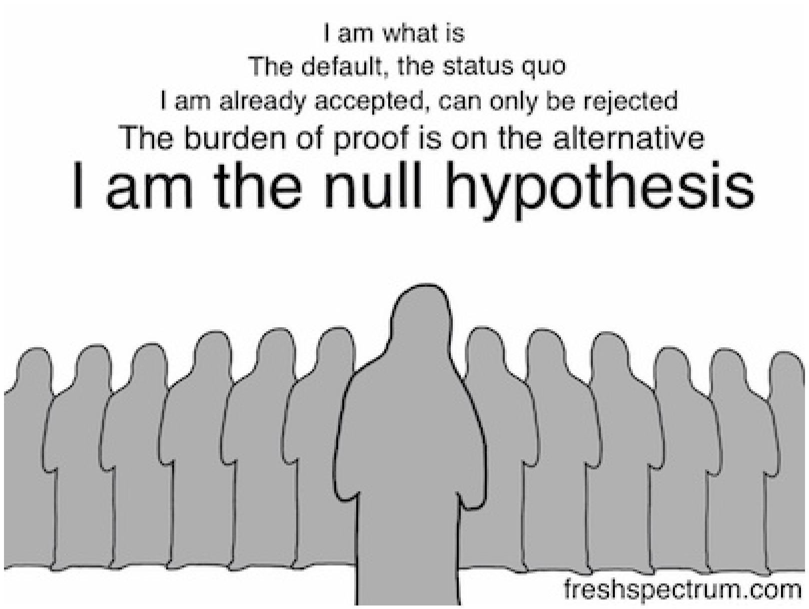
But how much close?

I will calculate the likelihood of observing

the difference between the sample means

assuming my theory, *the null hypothesis*, holds.

If the probability is low, then Bob may be right?


I am what is
The default, the status quo
I am already accepted, can only be rejected
The burden of proof is on the alternative
I am the null hypothesis
freshspectrum.com

Hypothesis Testing

Calculate a statistically meaningful number using the samples:

$$t = f([x_0, x_1, \cdots, x_n], [z_0, z_1, \cdots, z_m])$$

If you assume the equal variances between X and Z, then Student's t-test.

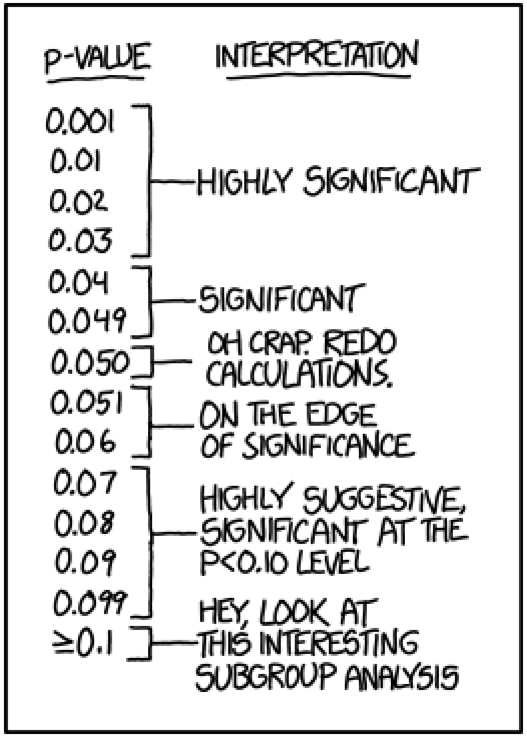
Or, if you think their variances are different, then Welch's t-test.

It is always about what assumptions you make.

Many other variants based on different assumptions.

The probability of this t-statistics, *which you may end up using/hearing a lot of times in your professinoal career*, is the infamous "**p-value**".

Just for fun.


P-VALUE
INTERPRETATION
0.001
0.01
0.02
0.03
HIGHLY SIGNIFICANT
0.04
0.049
SIGNIFICANT
0.050
OH CRAP. REDO CALCULATIONS.
0.051
0.06
ON THE EDGE OF SIGNIFICANCE
0.07
0.08
0.09
0.099
HIGHLY SUGGESTIVE, SIGNIFICANT AT THE P<0.10 LEVEL
≥0.1
HEY, LOOK AT THIS INTERESTING SUBGROUP ANALYSIS

https://xkcd.com/1478/

Let's run some tests for real.

```
from scipy import stats
houses_noriver = df.loc[df["CHAS"]==0,"MEDV"].values
houses_wriver = df.loc[df["CHAS"]==1,"MEDV"].values

print("Avg(Prices w/ River): ${0:.2f}".format(np.mean(houses_wriver)*1000))
print("Std(Prices w/ River): ${0:.2f}".format(np.std(houses_wriver)*1000))
print("Avg(Prices w/o River): ${0:.2f}".format(np.mean(houses_noriver)*1000))
print("Std(Prices w/o River): ${0:.2f}".format(np.std(houses_noriver)*1000))

results = stats.ttest_ind(houses_wriver, houses_noriver, equal_var=True)
print(results)

results = stats.ttest_ind(houses_wriver, houses_noriver, equal_var=False)
print(results)
```

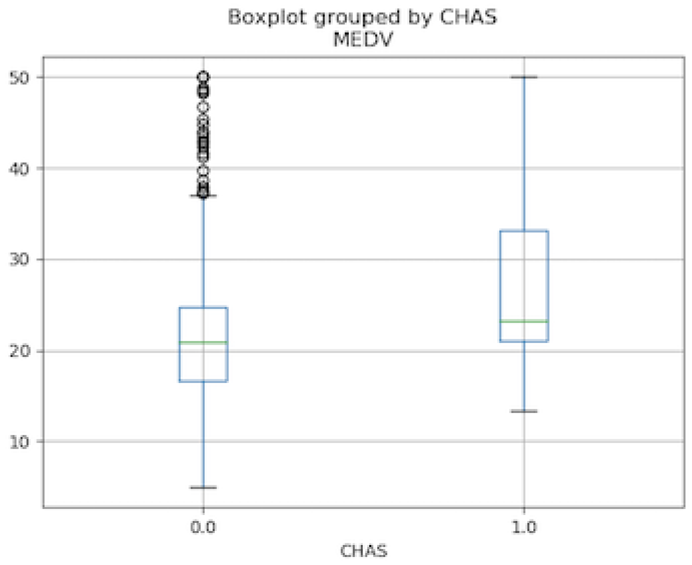
```
Avg(Prices w/ River): $28440.00
Std(Prices w/ River): $11646.61
Avg(Prices w/o River): $22093.84
Std(Prices w/o River): $8821.98
Ttest_indResult(statistic=3.996437466090508, pvalue=7.390623170519937e-05)
Ttest_indResult(statistic=3.113291312794837, pvalue=0.003567170098137517)
```

Looks like your friend, Bob, may be right.

https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_ind.html

Or, if you are lazy like me:

```
ax = df.boxplot(column="MEDV", by="CHAS")
fig = ax.get_figure()
fig.savefig("img/boxplot_CHAS_MEDV.png")
ax.clear()
```



Sometimes, it is more obvious in a graph than a number.

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

Please explore the other variables in the dataset **by yourself**.

What other distributions do you see?

What other hypotheses can you come up and how would you validate?

How would you visualize the data to support your claims?

Questions?