Big Data Analytics: What is Big Data?

H. Andrew Schwartz
Stony Brook University
CSE545, Spring 2019

What’s the BIG deal?!

2010

2011

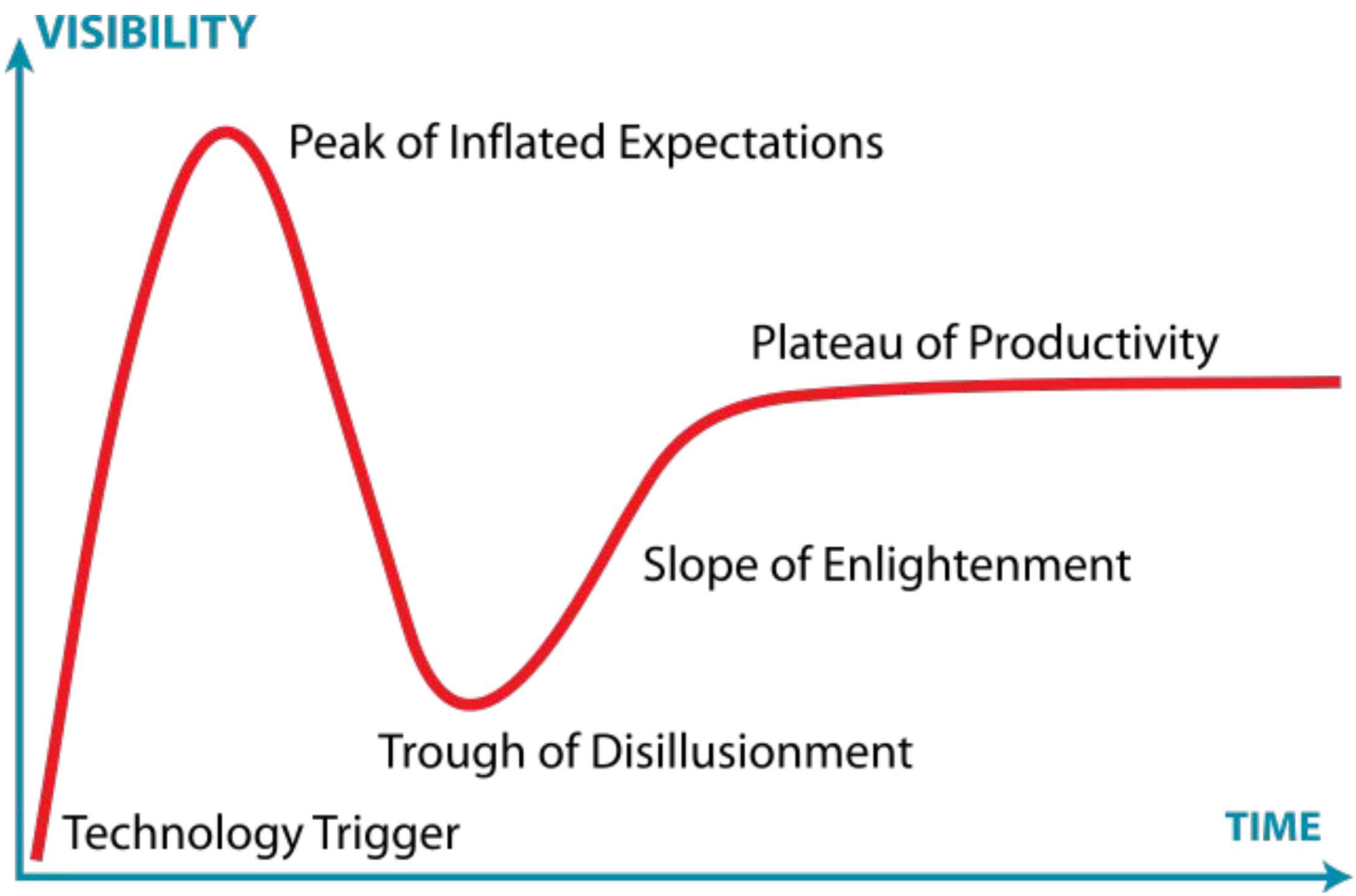
2008

2011

2012

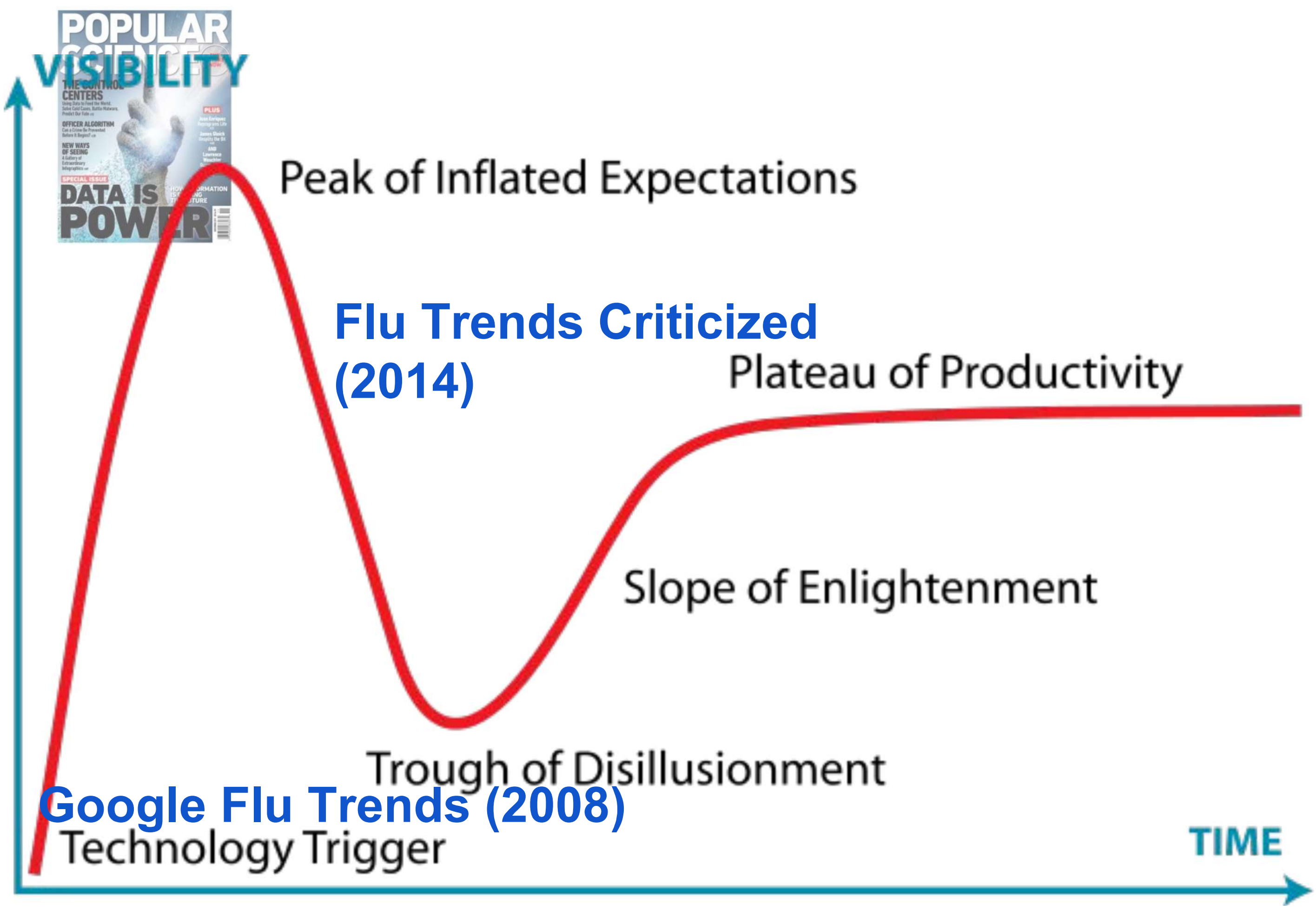
2018

What's the BIG deal?!



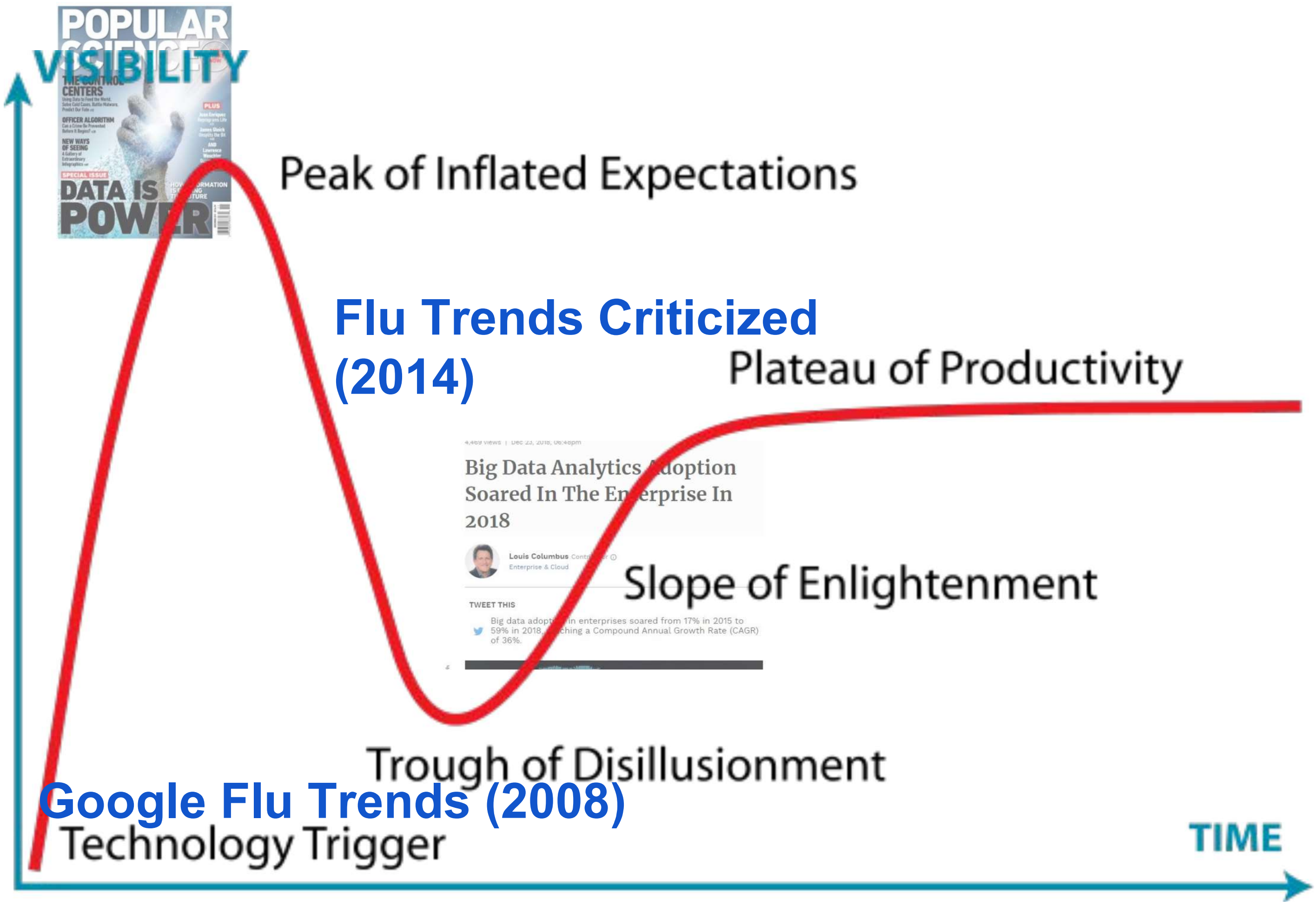
(Gartner Hype Cycle)

What’s the BIG deal?!



(Gartner Hype Cycle)

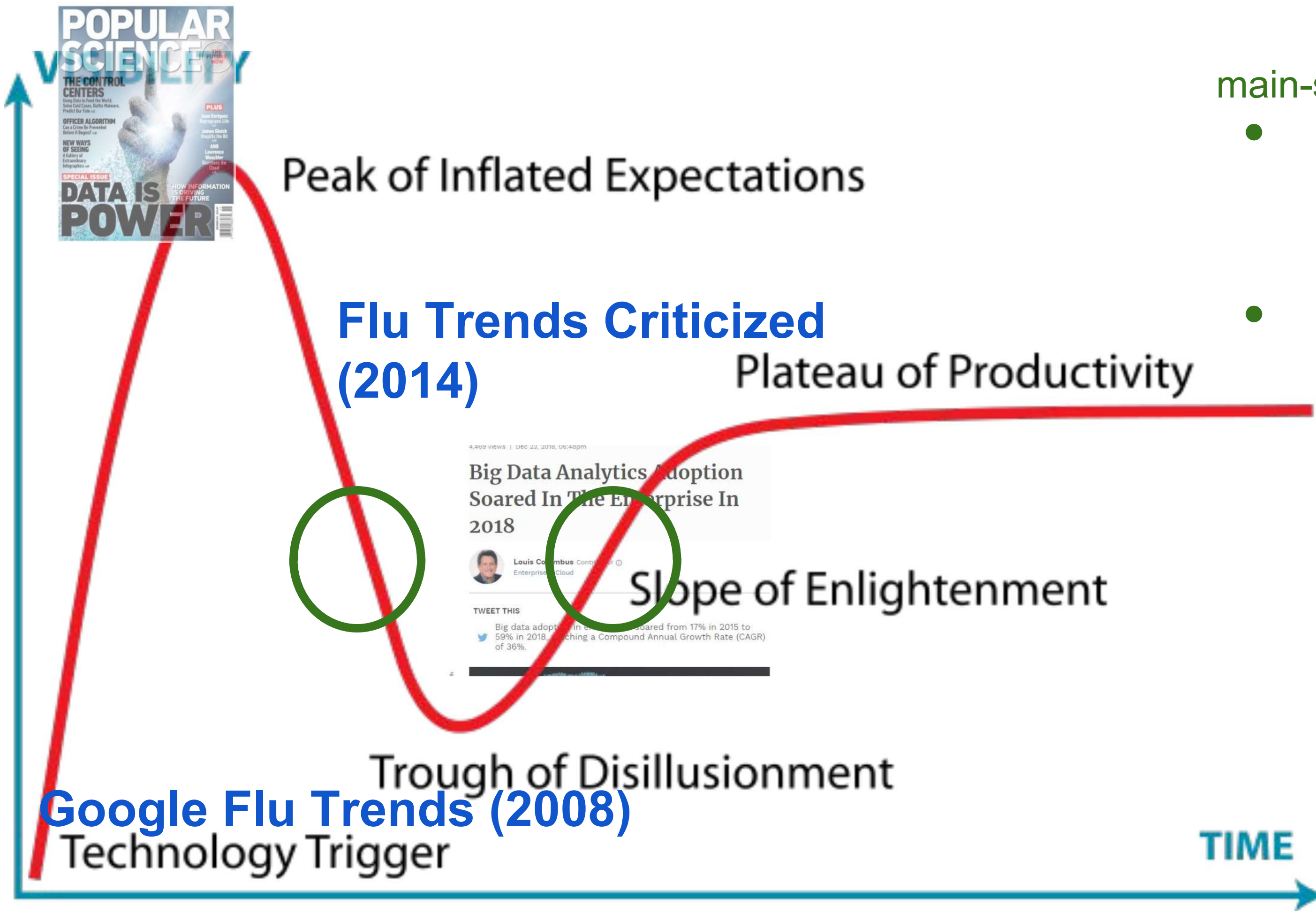
What's the BIG deal?!



(Gartner Hype Cycle)

Where are we today?

What’s the BIG deal?!

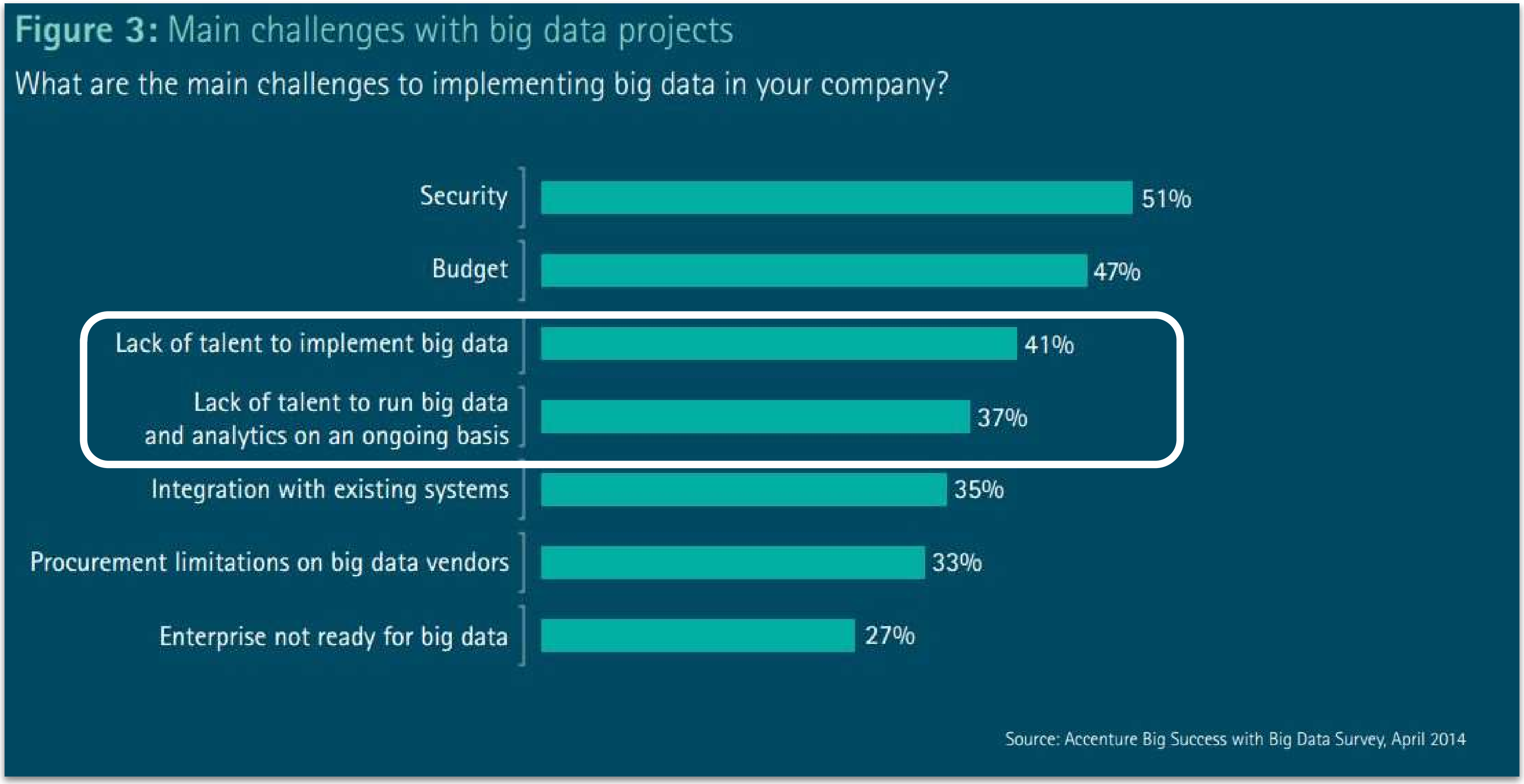


main-stream study being established

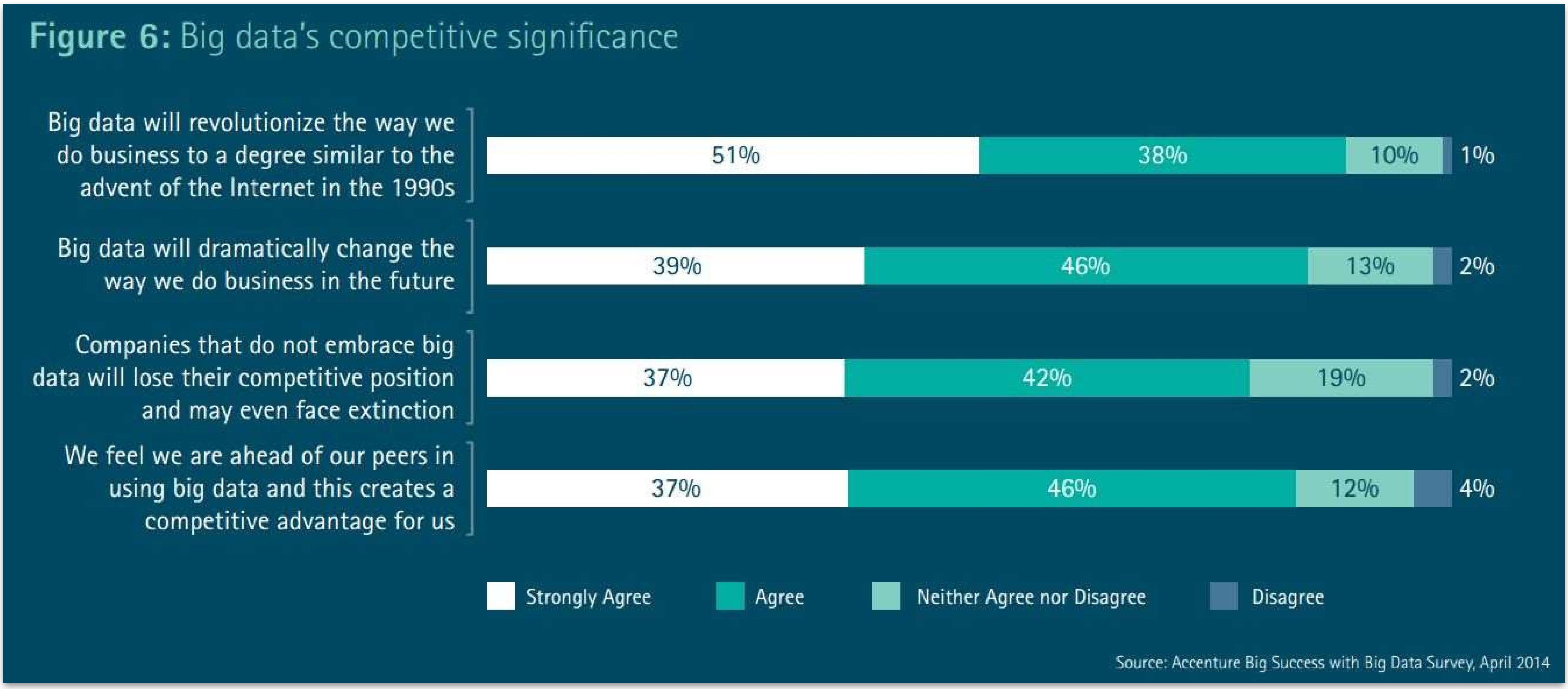
- Realization of what subfields are really doing “big data” (i.e. data mining, ML, Statistics, computational social sciences).
- Best practices being established.

(Gartner Hype Cycle)

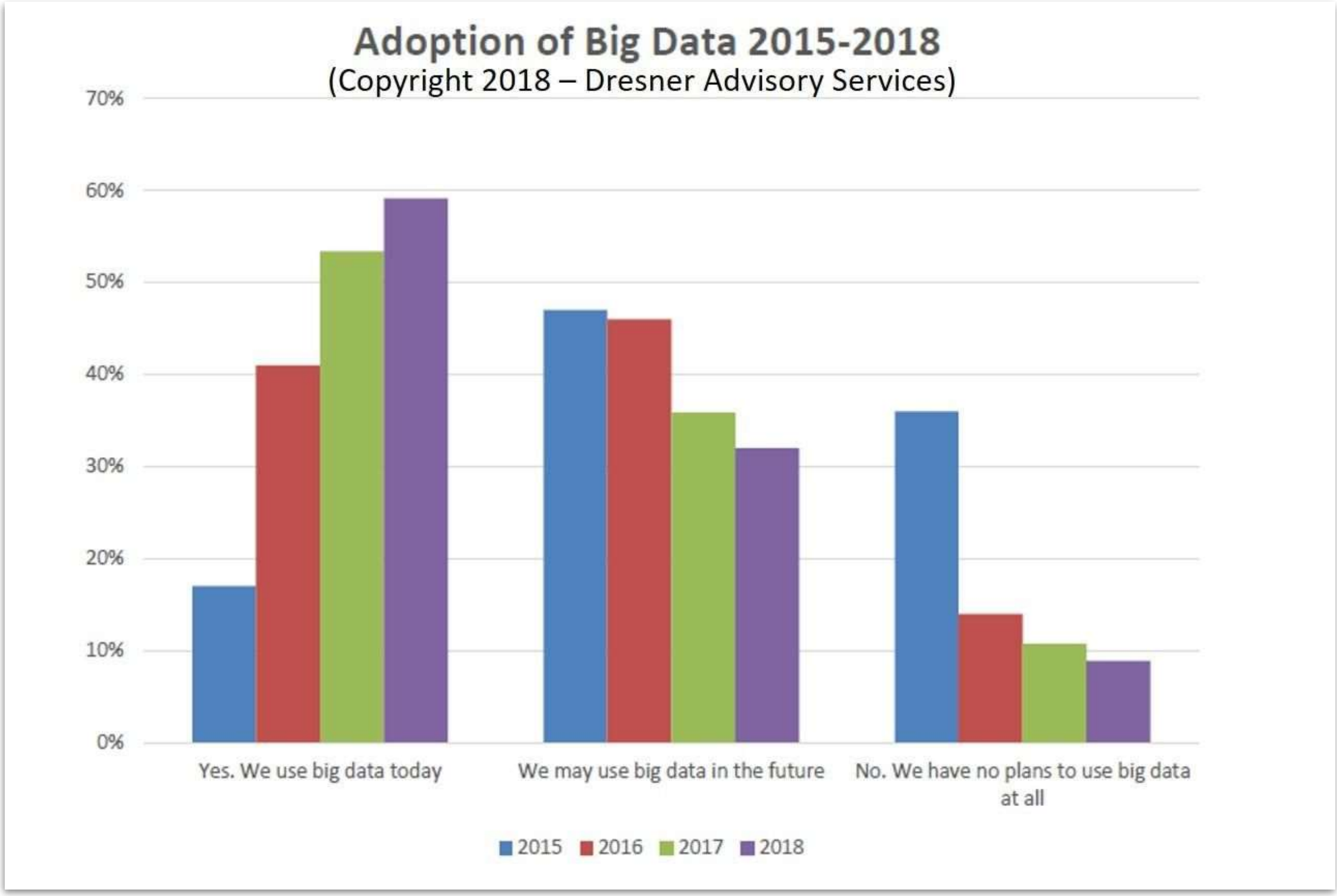
What's the BIG deal?!



What's the BIG deal?!

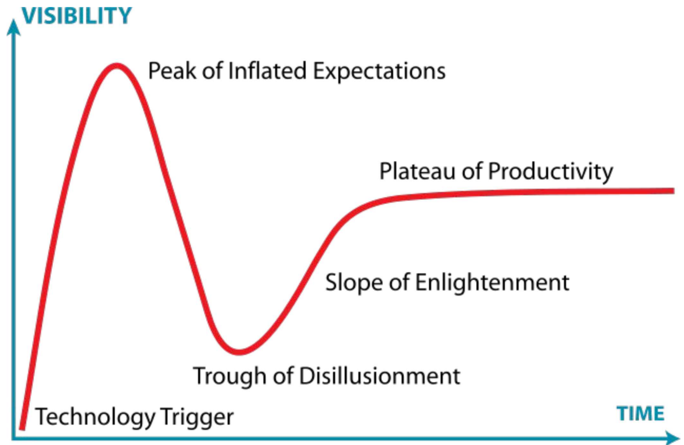


What's the BIG deal?!



https://www.forbes.com/sites/louiscolumbus/2018/12/23/big-data-analytics-adoption-soared-in-the-enterprise-in-2018/

What's the BIG deal?!



Reasons to be skeptical

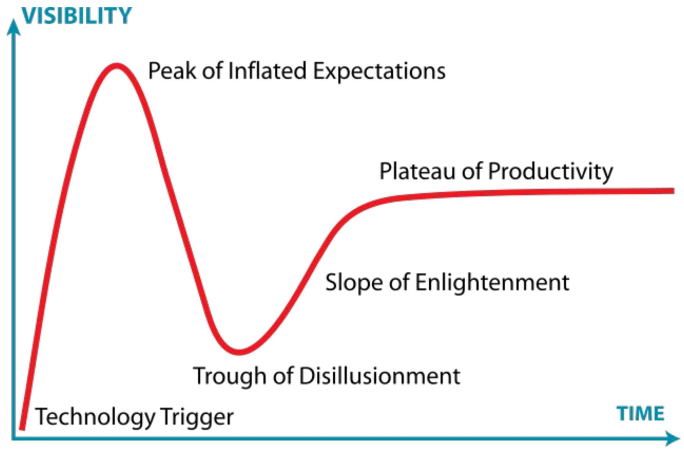
- Hype machine
- Downside of many tools:
  - Creates obfuscation: encourages seeing as magic black boxes
  - Less "standards": difficult to translate between, understand results
- Downside of large amounts of data:
  - Harder to "view"
  - Training takes longer
  - More prone to errors: rounding; heterogeneity

amazon web services™

Microsoft Azure

Google Cloud

What's the BIG deal?!



amazon web services™

Microsoft Azure

Google Cloud

Reasons to be skeptical

- Hype machine
- Downside of many tools:
  - Creates obfuscation: encourages seeing as magic black boxes
  - Less "standards": difficult to translate between, understand results
- Downside of large amounts of data:
  - Harder to "view"
  - Training takes longer
  - More prone to errors: rounding; heterogeneity

Combat with:

- **Understanding *how* it works (theory)**
- ***When/where it works* (applied; experience)**

What is Big Data?

Combat with:

- **Understanding *how* it works (theory)**
- ***When/where it works* (applied; experience)**

What is Big Data?

traditional
computer science

data that will not fit
in main memory.

What is Big Data?

traditional
computer science

data that will not fit
in main memory.

data with a *large*
number of observations
and/or features.

statistics

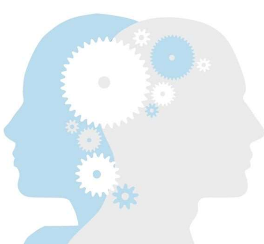
What is Big Data?

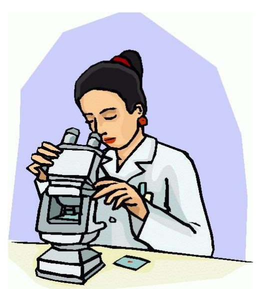
traditional
computer science

data that will not fit
in main memory.

data with a *large*
number of observations
and/or features.

statistics

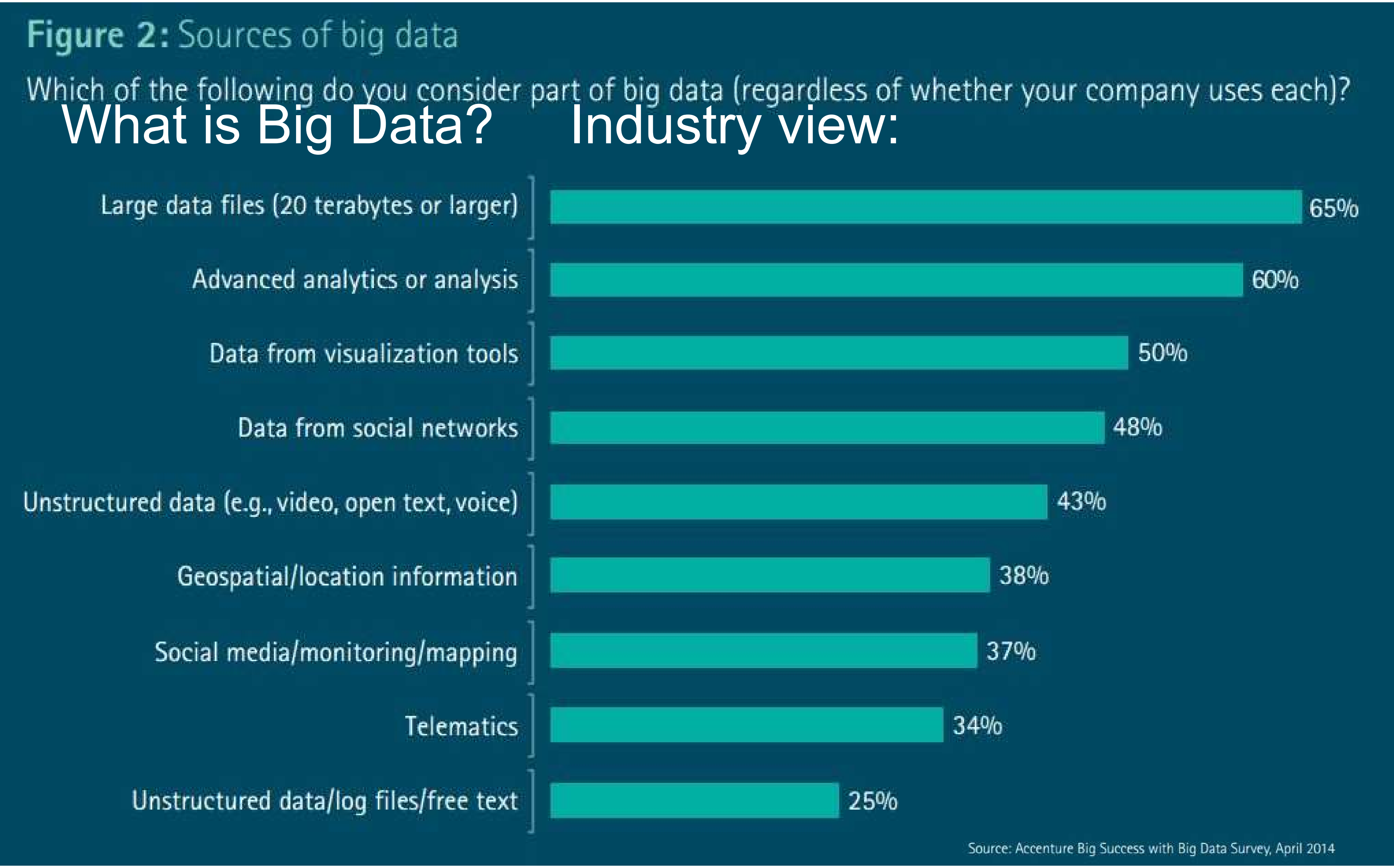




other fields

non-traditional sample size
(i.e. > 100 subjects); can't
analyze in stats tools (Excel).

What is Big Data? Industry view:



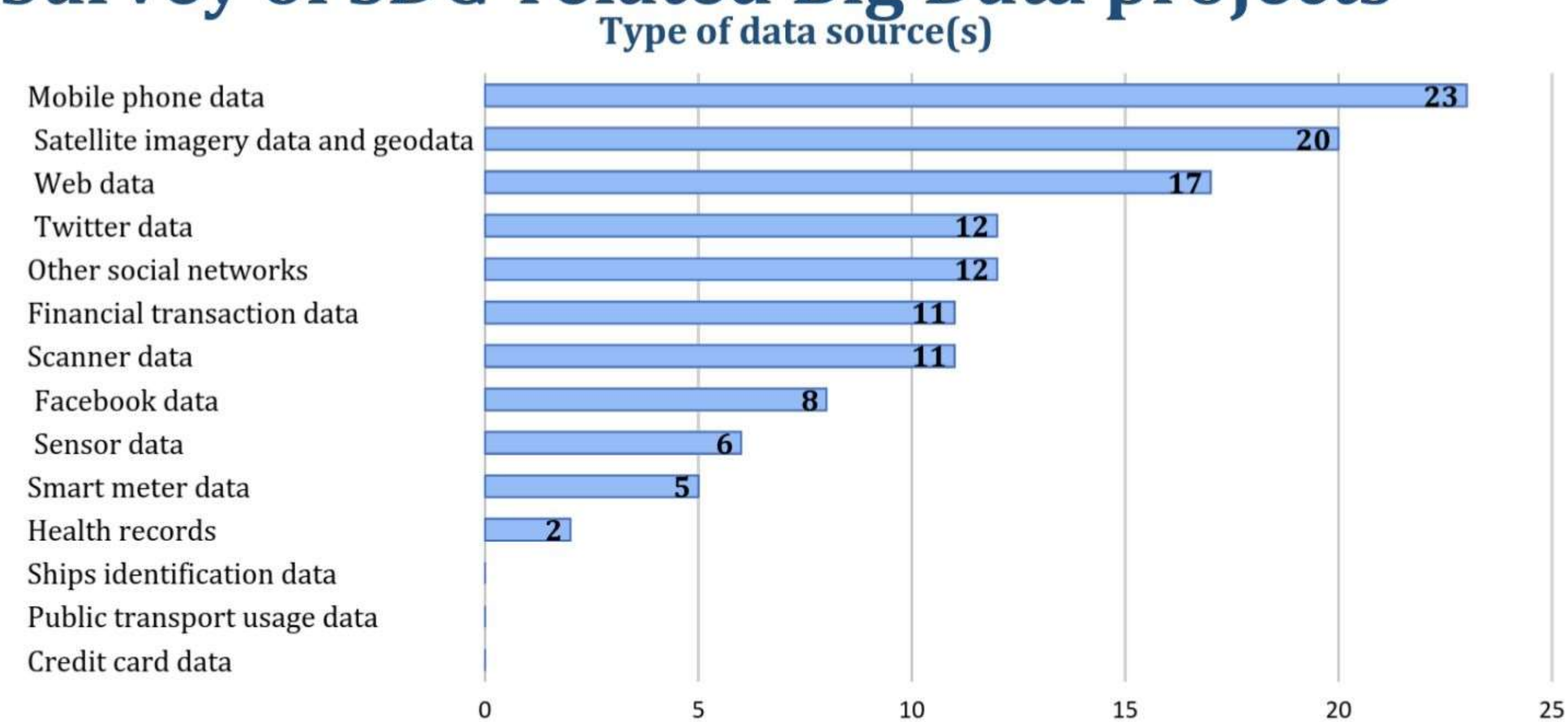
What is Big Data? Government view:

THE WORLD BANK
IBRD • IDA | WORLD BANK GROUP

(2016)

BigData
UN Global Working Group

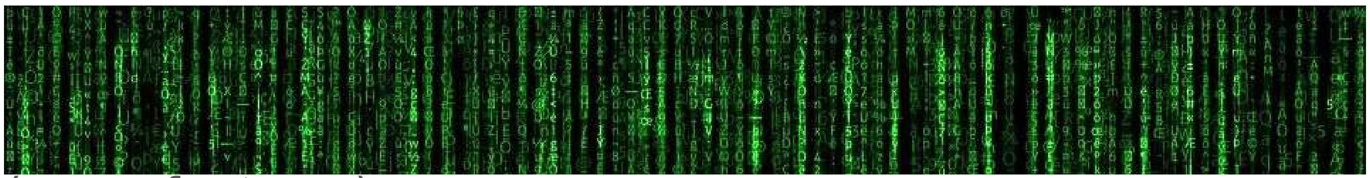
1. Survey of SDG-related Big Data projects



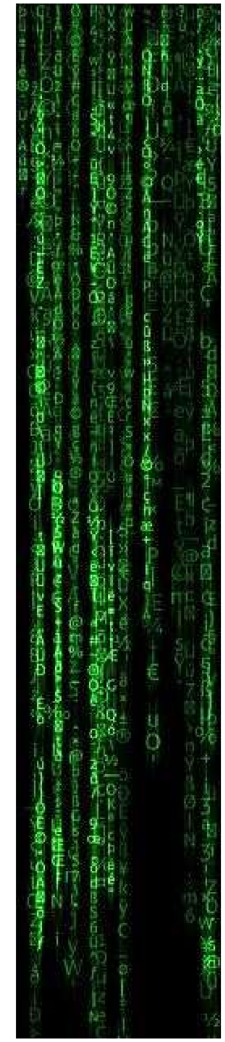
- Mobile (23), Satellite imagery (20) and social media (12+12+8) are the most prominent sources

What is Big Data?

Wide



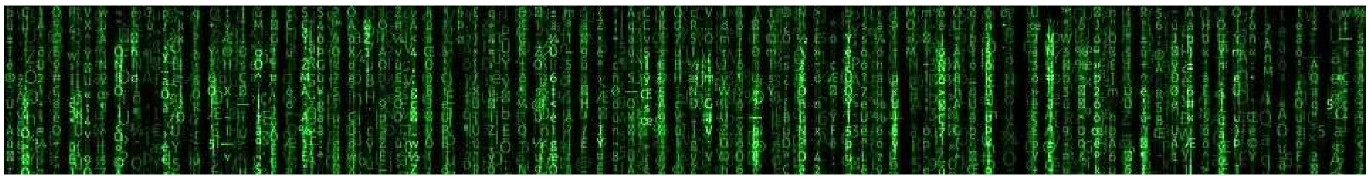
(many features)



Tall

(many records)

What is Big Data?

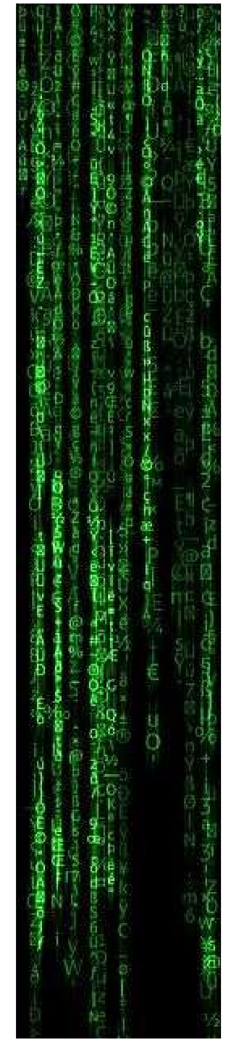


data that will not fit in main memory.

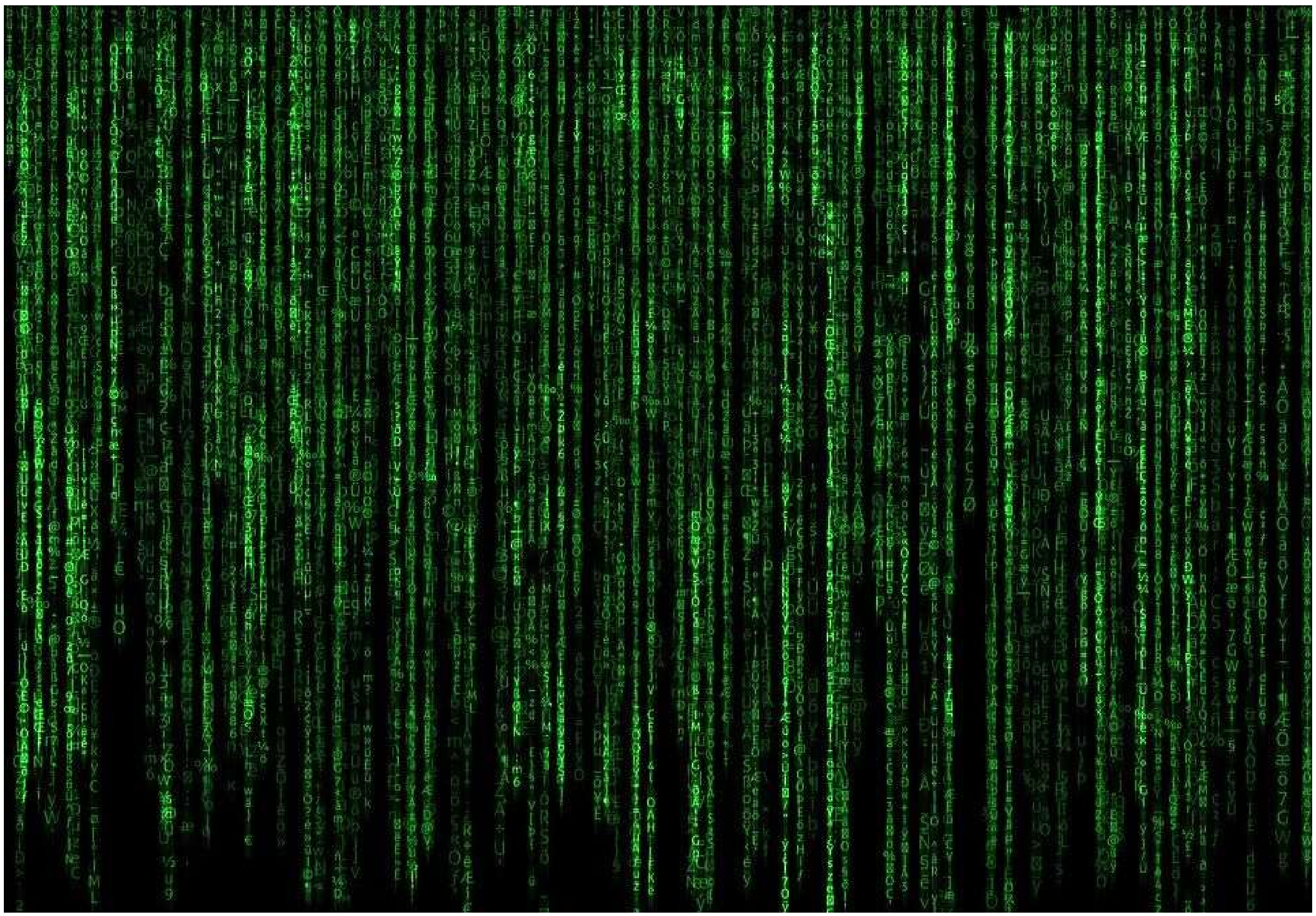
data with a *large* number of observations and/or features.

Tall

(many records)



What is Big Data?



What is Big Data?

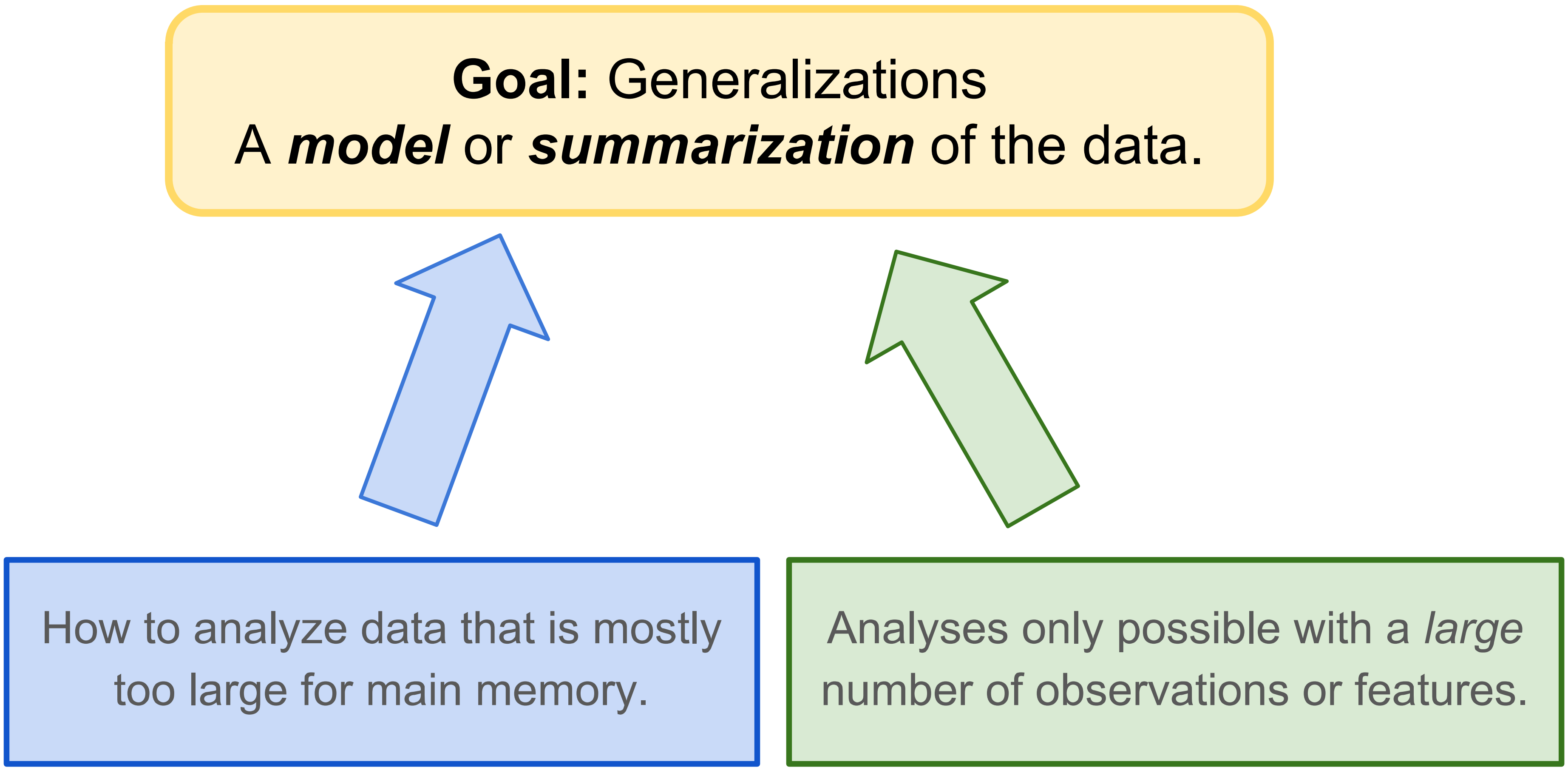
Short Answer:

Big Data ≈ Data Mining ≈ Predictive Analytics ≈ Data Science (Leskovec et al., 2014)

This Class:

How to analyze data that is mostly too large for main memory.	Analyses only possible with a *large* number of observations or features.

What is Big Data?



What is Big Data?

Goal: Generalizations
A *model* or *summarization* of the data.

E.g.

- Google's PageRank: *summarizes* web pages by a single number.
- Twitter financial market predictions: *Models* the stock market according to shifts in sentiment in Twitter.
- Distinguish tissue type in medical images: *Summarizes* millions of pixels into clusters.
- Mental health diagnosis in social media: *Models* presence of diagnosis as a distribution (a summary) of linguistic patterns.
- Frequent co-occurring purchases: *Summarize* billions of purchases as items that frequently are bought together.

What is Big Data?

Goal: Generalizations
A *model* or *summarization* of the data.

1. Descriptive analytics
   Describe (*generalizes*) the data itself

2. Predictive analytics
   Create something *generalizeable* to new data

Big Data Analytics -- The Class

Core Data Science Courses

CSE 519: Data Science Fundamentals

CSE 544: Prob/Stat for Data Scientists

CSE 545: Big Data Analytics

CSE 512: Machine Learning

CSE 537: Artificial Intelligence

CSE 548: Analysis of Algorithms

CSE 564: Visualization

Applications of Data Science

CSE 507:
Computational Linguistics

CSE 527:
Computer Vision

CSE 549:
Computational Biology

Big Data Analytics -- The Class

Core Data Science Courses

CSE 519: Data Science Fundamentals

CSE 544: Prob/Stat for Data Scientists

CSE 545: Big Data Analytics

CSE 512: Machine Learning

CSE 537: Artificial Intelligence

CSE 548: Analysis of Algorithms

CSE 564: Visualization

Applications of Data Science

CSE 507:
Computational Linguistics

CSE 527:
Computer Vision

CSE 549:
Computational Biology

Key Distinction:
Focus on scalability and algorithms / analyses not possible without large data.

Big Data Analytics -- The Class

We will learn:

- to analyze different types of data:
  - high dimensional
  - graphs
  - infinite/never-ending
  - labeled
- to use different models of computation:
  - MapReduce
  - streams and online algorithms
  - single machine in-memory
  - *Spark, Tensorflow*

Big Data Analytics -- The Class

We will learn:

- to solve real-world problems
  - Recommendation systems
  - Market-basket analysis
  - Spam and duplicate document detection
  - *Geo-coding data*
- uses of various “tools”:
  - linear algebra
  - optimization
  - dynamic programming
  - hashing
  - *functional programming*
  - *tensorflow*

Big Data Analytics -- The Class

http://www3.cs.stonybrook.edu/~has/CSE545/

Preliminaries

Ideas and methods that will repeatedly appear:

- Bonferroni's Principle
- Normalization (TF.IDF)
- Power Laws
- Hash functions
- IO Bounded (Secondary Storage)
- Unstructured Data

- *Parallelism*
- *Functional Programming*

Statistical Limits.

Goal: **Generalization**

Bonferroni's Principle

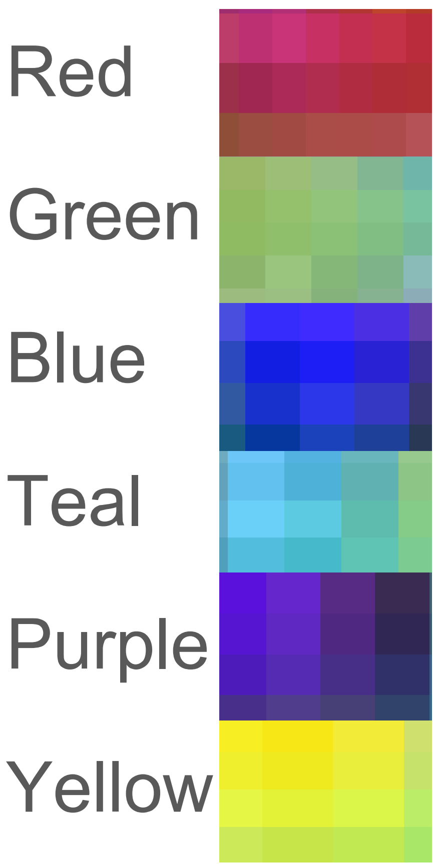
Statistical Limits. Goal: **Generalization**

Bonferroni's Principle

Statistical Limits

Which iphone case will be least popular?

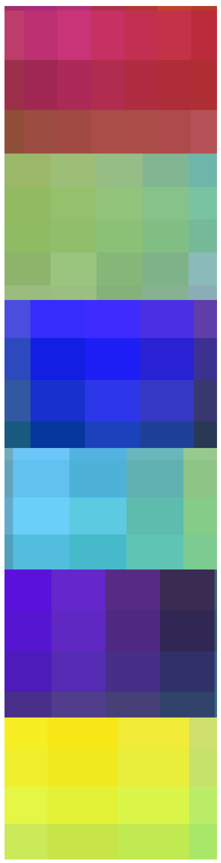
Bonferroni's Principle



Statistical Limits

Bonferroni's Principle

Red
Green
Blue
Teal
Purple
Yellow

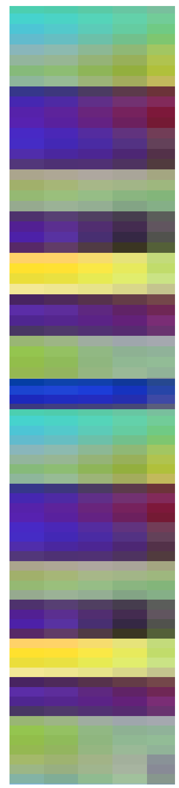


Which iphone case will be least popular?

First 20 sales come in:

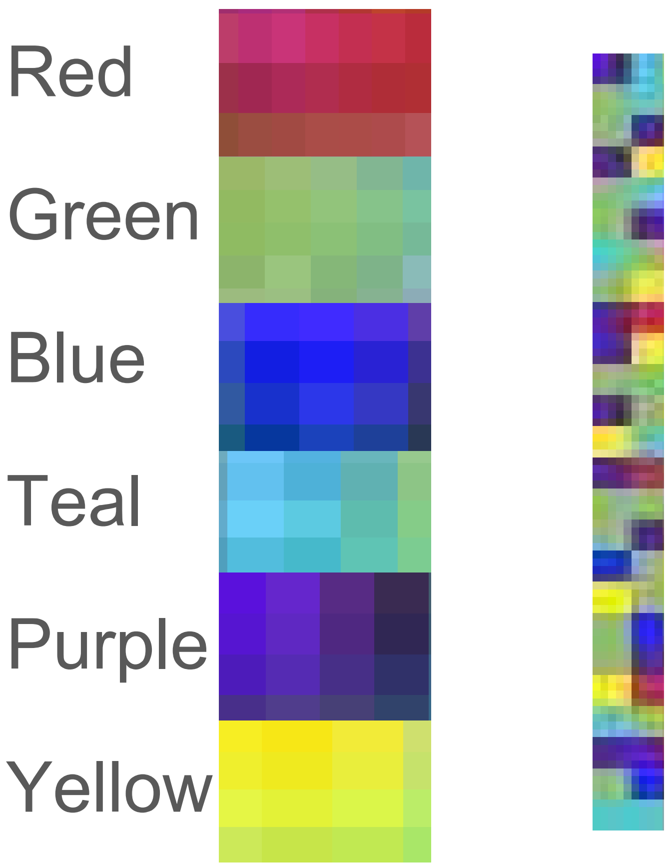
Can you make any conclusions?

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20



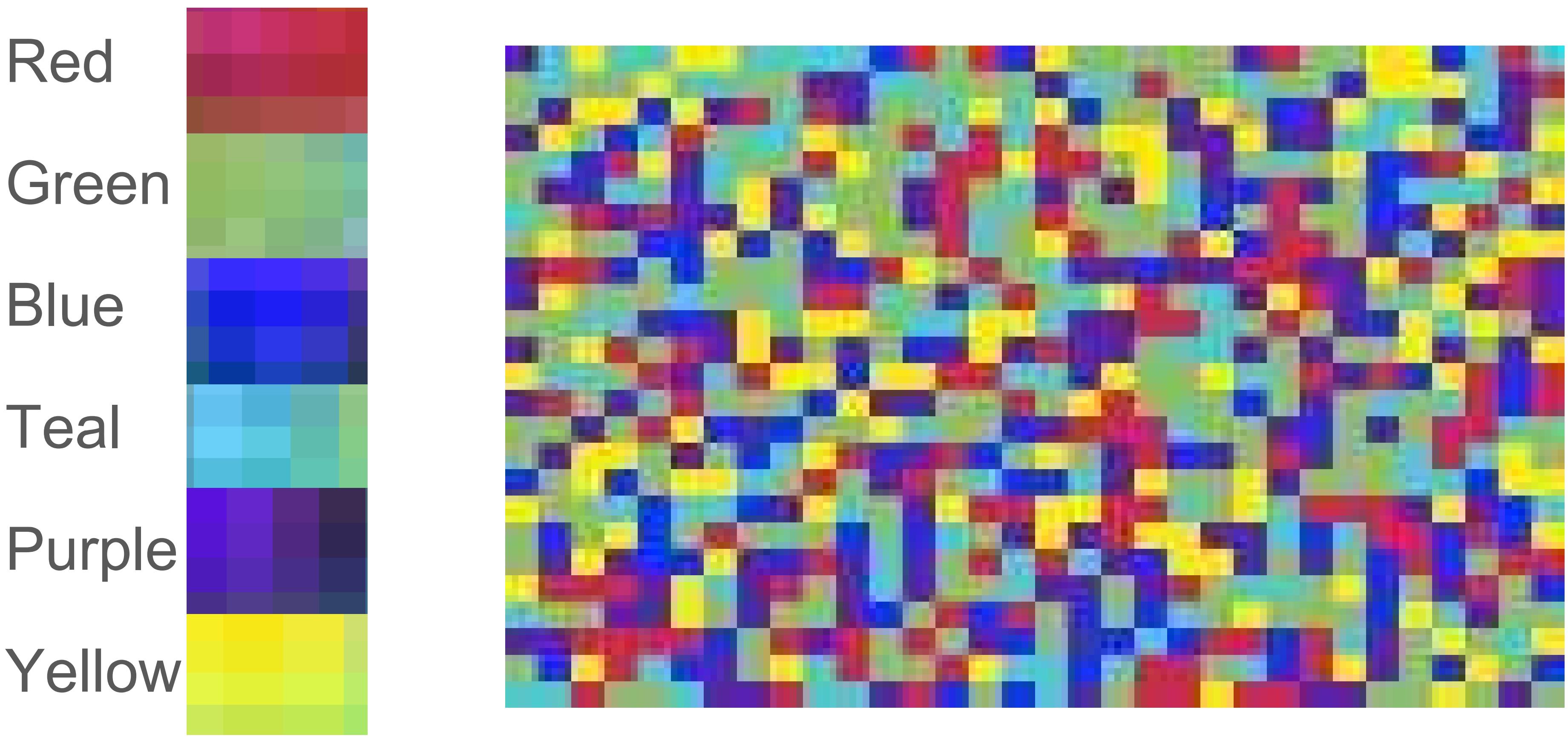
Statistical Limits

Bonferroni's Principle



Statistical Limits

Bonferroni's Principle



Statistical Limits. Goal: **Generalization**

Bonferroni's Principle

Roughly, calculating the probability of any of n *findings* being true requires n times the probability as testing for 1 finding.

https://xkcd.com/882/

In brief, one can only look for so many patterns (i.e. features) in the data before one finds something just by chance (i.e. finding something that does **not** generalize).

"Data mining" is a bad word in some communities!

Normalizing

Count data often need *normalizing* -- putting the numbers on the same "scale".

Prototypical example: TF.IDF

Normalizing

Count data often need *normalizing* -- putting the numbers on the same “scale”.

Prototypical example: TF.IDF of word *i* in document *j:*

Term Frequency:

$$tf_{ij} = \frac{count_{ij}}{\max_k count_{kj}}$$

$$tf.idf_{ij} = tf_{ij} \times idf_i$$

Inverse Document Frequency:

$$idf_i = log_2(\frac{docs_*}{docs_i}) \propto \frac{1}{\frac{docs_i}{docs_*}}$$

where docs is the number of documents containing word *i*.

Normalizing

Count data often need *normalizing* -- putting the numbers on the same "scale".

Prototypical example: TF.IDF of word *i* in document *j:*

Term Frequency:

$$tf_{ij} = \frac{count_{ij}}{\max_k \, count_{kj}}$$

$$tf.idf_{ij} = tf_{ij} \times idf_i$$

Inverse Document Frequency:

$$idf_i = log_2(\frac{docs_*}{docs_i}) \propto \frac{1}{\frac{docs_i}{docs_*}}$$

where docs is the number of documents containing word *i*.

Normalizing

Standardize: puts different sets of data (typically vectors or random variables) on the same scale with the same center.

- Subtract the mean (i.e. "mean center")
- Divide by standard deviation

$$z_i = \frac{x_i - \bar{x}}{s_x}$$

Power Law

Characterized many frequency patterns when ordered from most to least:

County Populations [r-bloggers.com]

links into webpages [Broader et al., 2000]

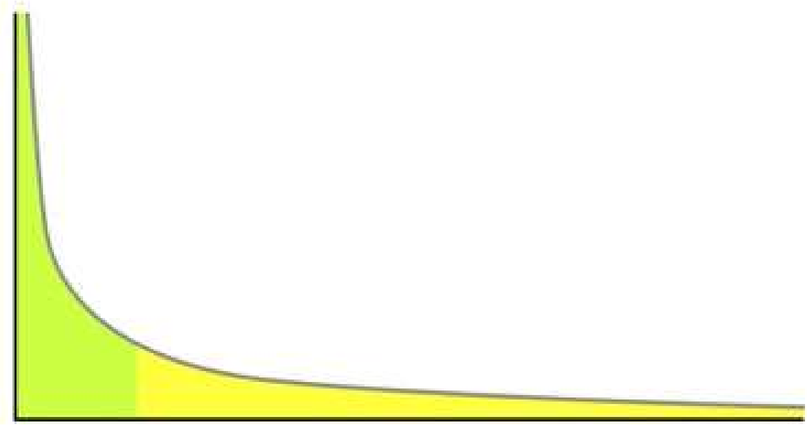
Sales of products [see book]

Frequency of words [Wikipedia, “Zipf’s Law”]

(“popularity” based statistics, especially without limits)

Power Law

$$\log y = b + a \log x$$



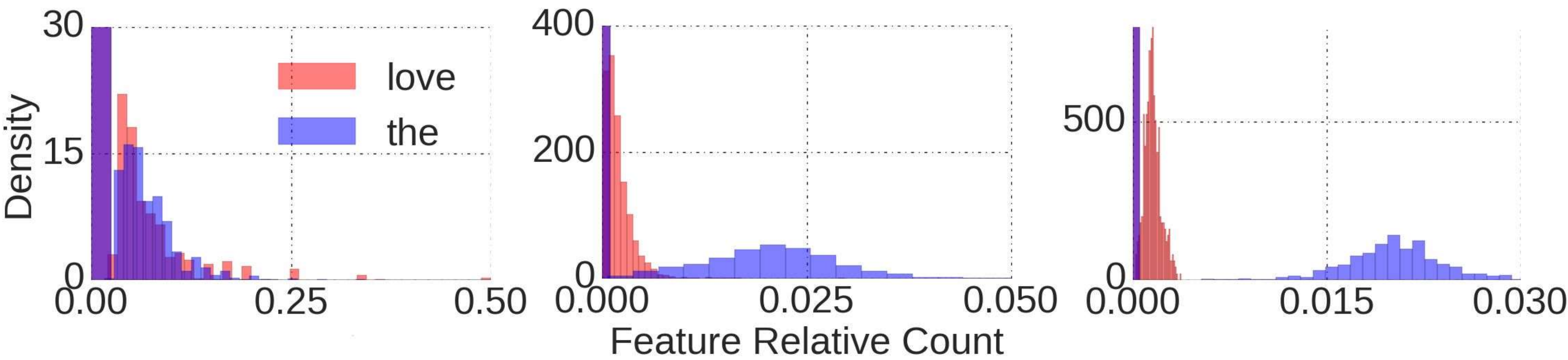
raising to the natural log:

$$y = e^b e^{a \log x} = e^b x^a = c x^a$$

where c is just a constant

Characterizes “the Matthew Effect” -- the rich get richer

Power Law



message-level

user-level

county-level

Almodaresi, F., Ungar, L., Kulkarni, V., Zakeri, M., Giorgi, S., & Schwartz, H. A. (2017). On the Distribution of Lexical Features at Multiple Levels of Analysis. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics* (pp. 79-84).

Hash Functions and Indexes

Review:

h: *hash-key -> bucket-number*

Objective: uniformly distribute hash-keys across buckets.

Example: storing word counts.

Hash Functions and Indexes

Review:

h: *hash-key -> bucket-number*

Objective: uniformly distribute hash-keys across buckets.

Example: storing word counts.

$$h(word) = \left(\sum_{char \in word} ascii(char) \right) \% \#buckets$$

Hash Functions and Indexes

Review:

h: *hash-key -> bucket-number*

Objective: uniformly distribute hash-keys across buckets.

Example: storing word counts.

$$h(word) = \left(\sum_{char \in word} ascii(char) \right) \% \#buckets$$

Data structures utilizing hash-tables (i.e. O(1) lookup; dictionaries, sets in python) are a friend of big data algorithms! Review further if needed.

Hash Functions and Indexes

Review:

h: *hash-key -> bucket-number*

Objective: uniformly distribute hash-keys across buckets.

Example: storing word counts.

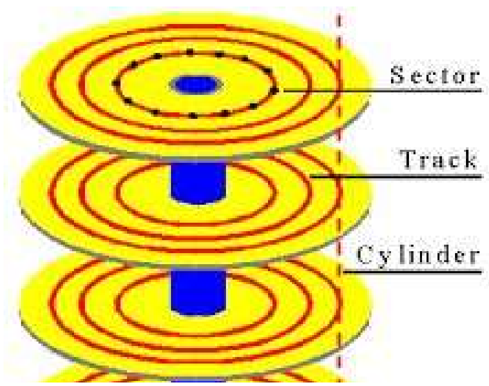
Database Indexes: Retrieve all records with a given *value.* (also review if unfamiliar / forgot)

Data structures utilizing hash-tables (i.e. O(1) lookup; dictionaries, sets in python) are a friend of big data algorithms! Review further if needed.

IO Bounded

Reading a word from disk versus main memory: 10^5 slower!

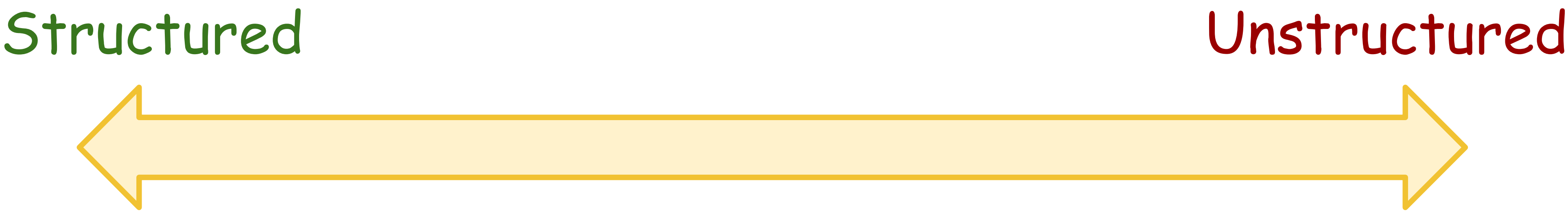
Reading many contiguously stored words is faster per word, but fast modern disks still only reach 150MB/s for sequential reads.



IO Bound: biggest performance bottleneck is reading / writing to disk.

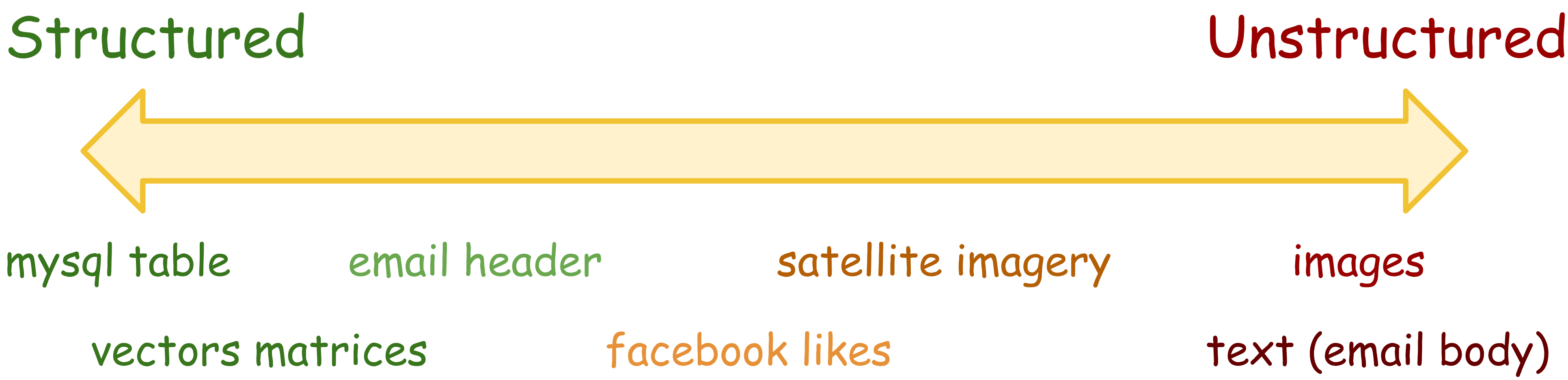
(starts around 100 GBs; ~10 minutes just to read).

Data



- Unstructured ≈ requires processing to get what is of interest
- Feature extraction used to turn unstructured into structured
- Near infinite amounts of potential features in unstructured data

Data



- Unstructured ≈ requires processing to get what is of interest
- Feature extraction used to turn unstructured into structured
- Near infinite amounts of potential features in unstructured data