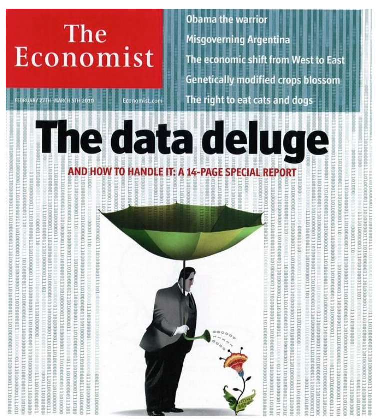
Big Data Analytics: What is Big Data?

Stony Brook University
CSE545, Fall 2016
“the inaugural edition”

What's the BIG deal?!

2008



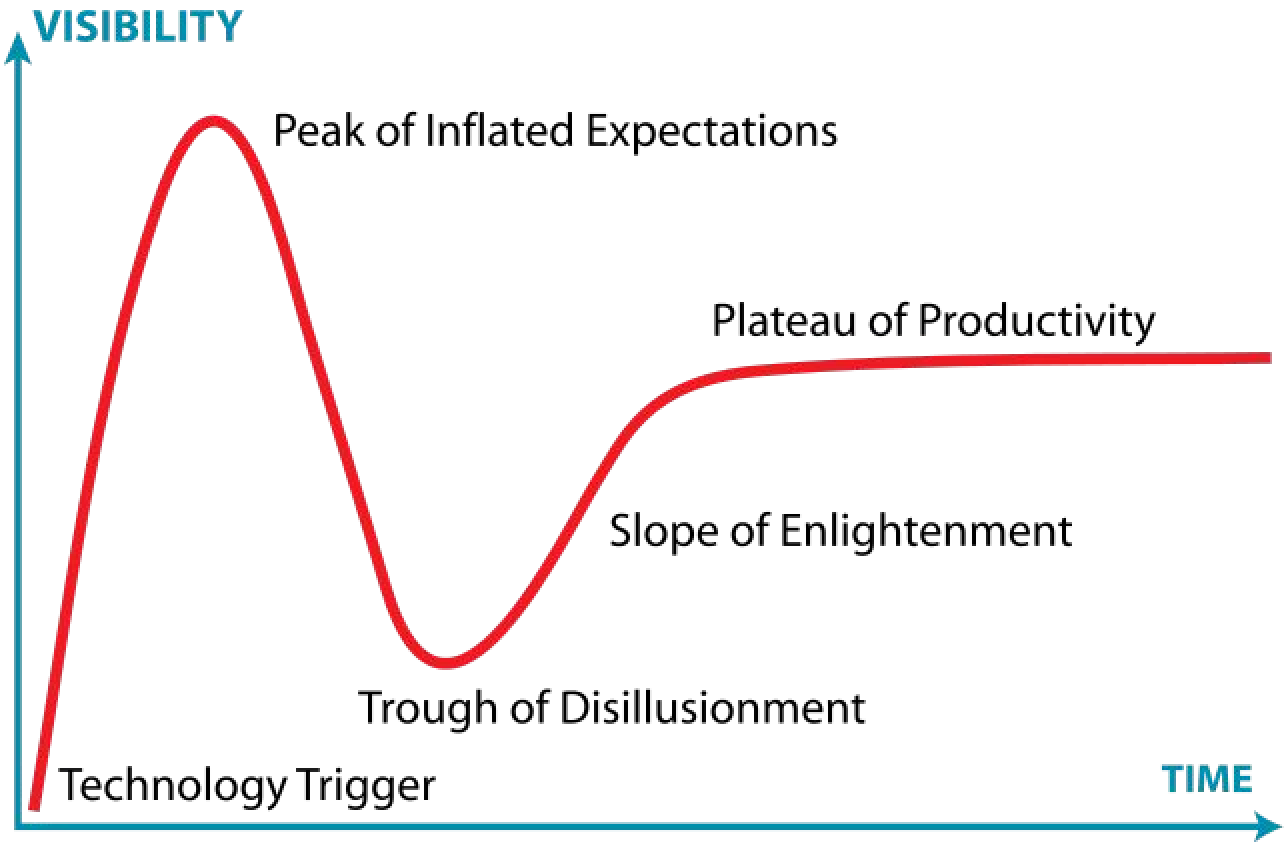
2010

2011

2011

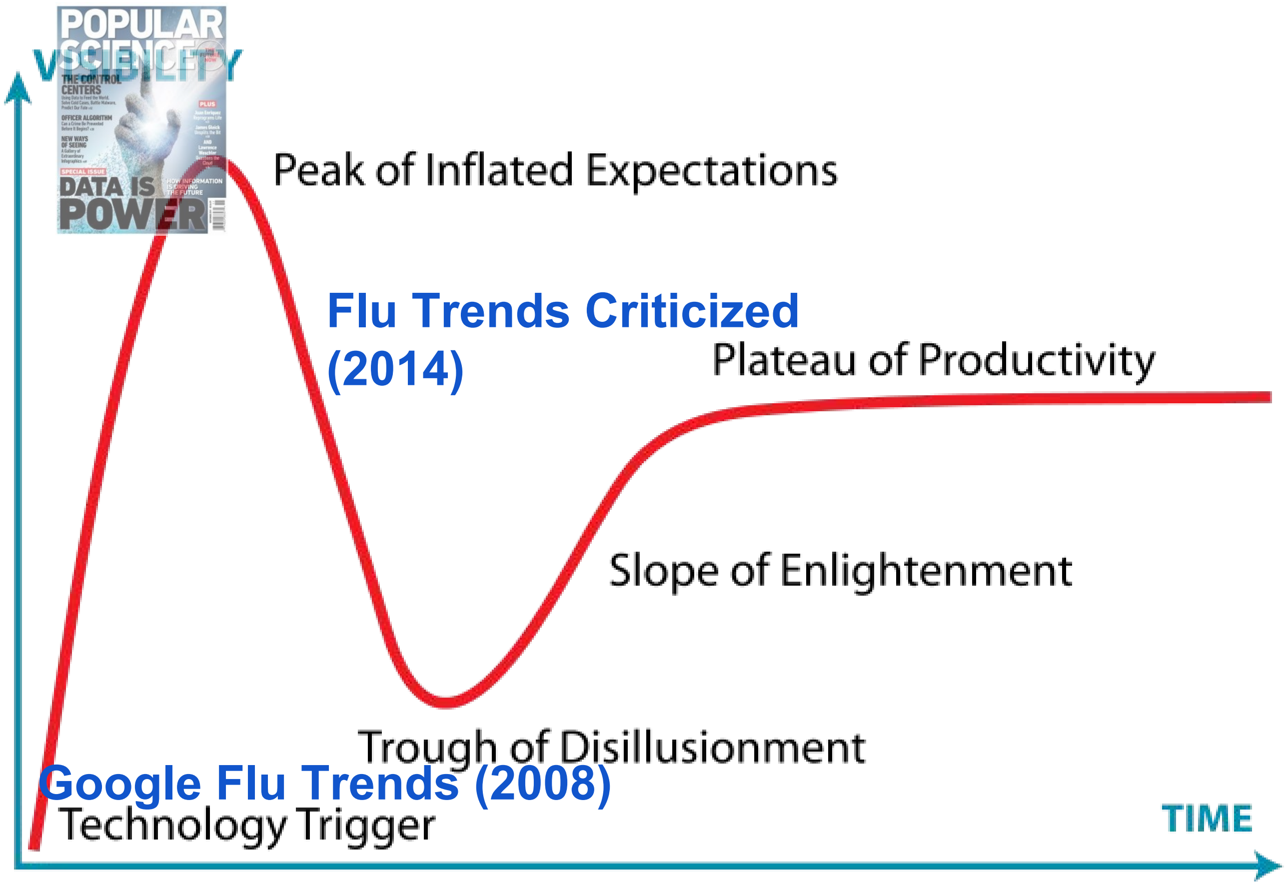
2012

What's the BIG deal?!



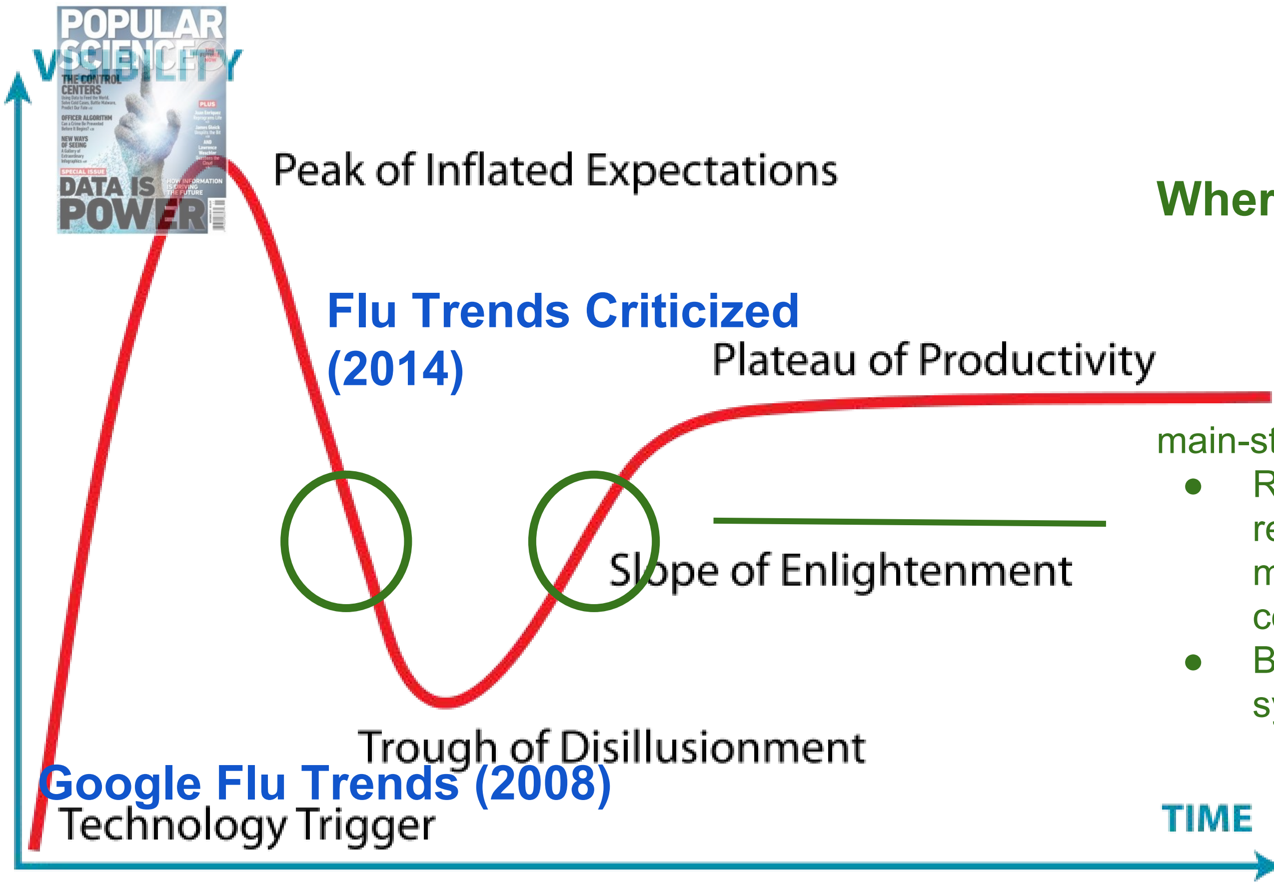
(Gartner Hype Cycle)

What’s the BIG deal?!



(Gartner Hype Cycle)

What’s the BIG deal?!



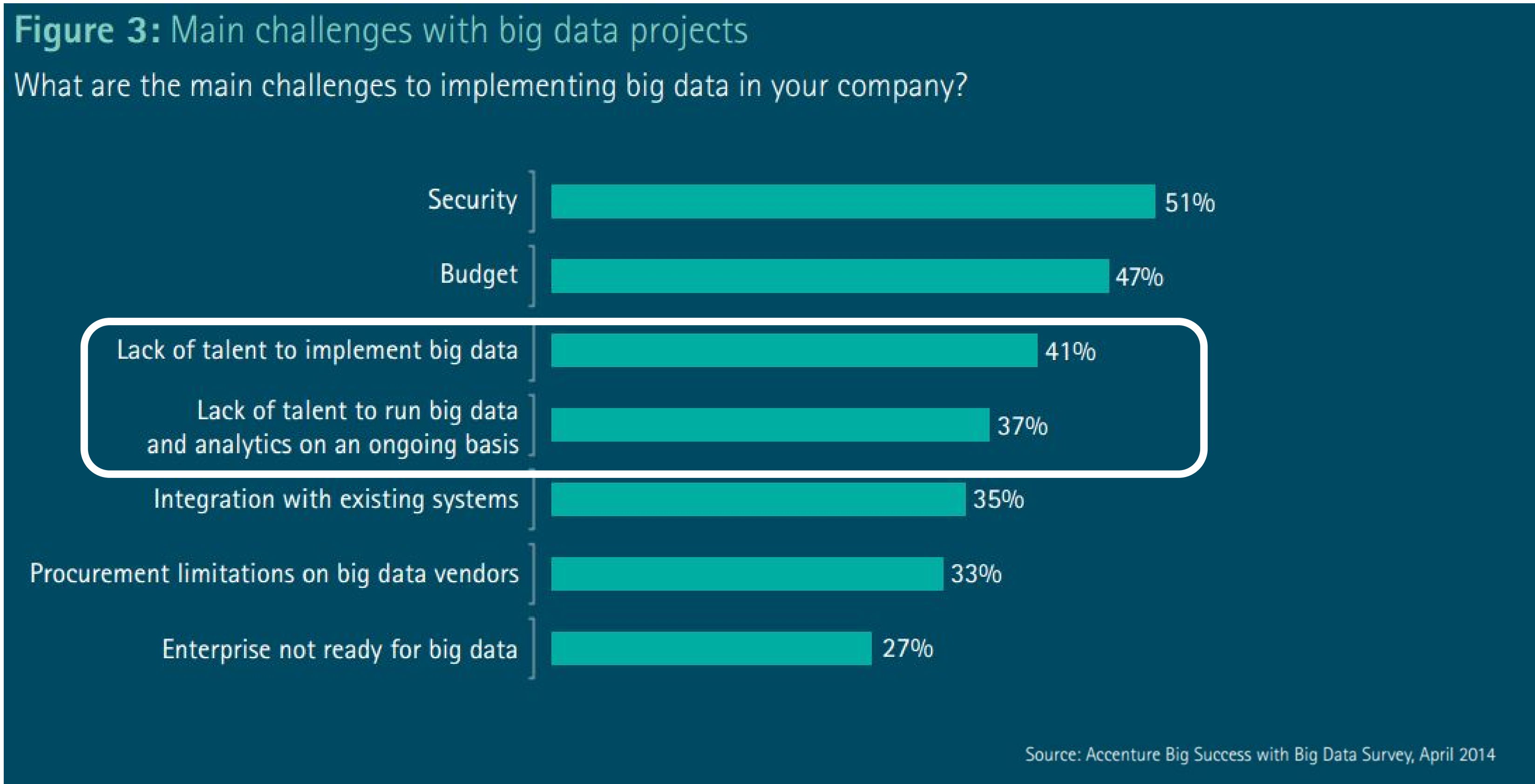
Where are we today?

main-stream study being established

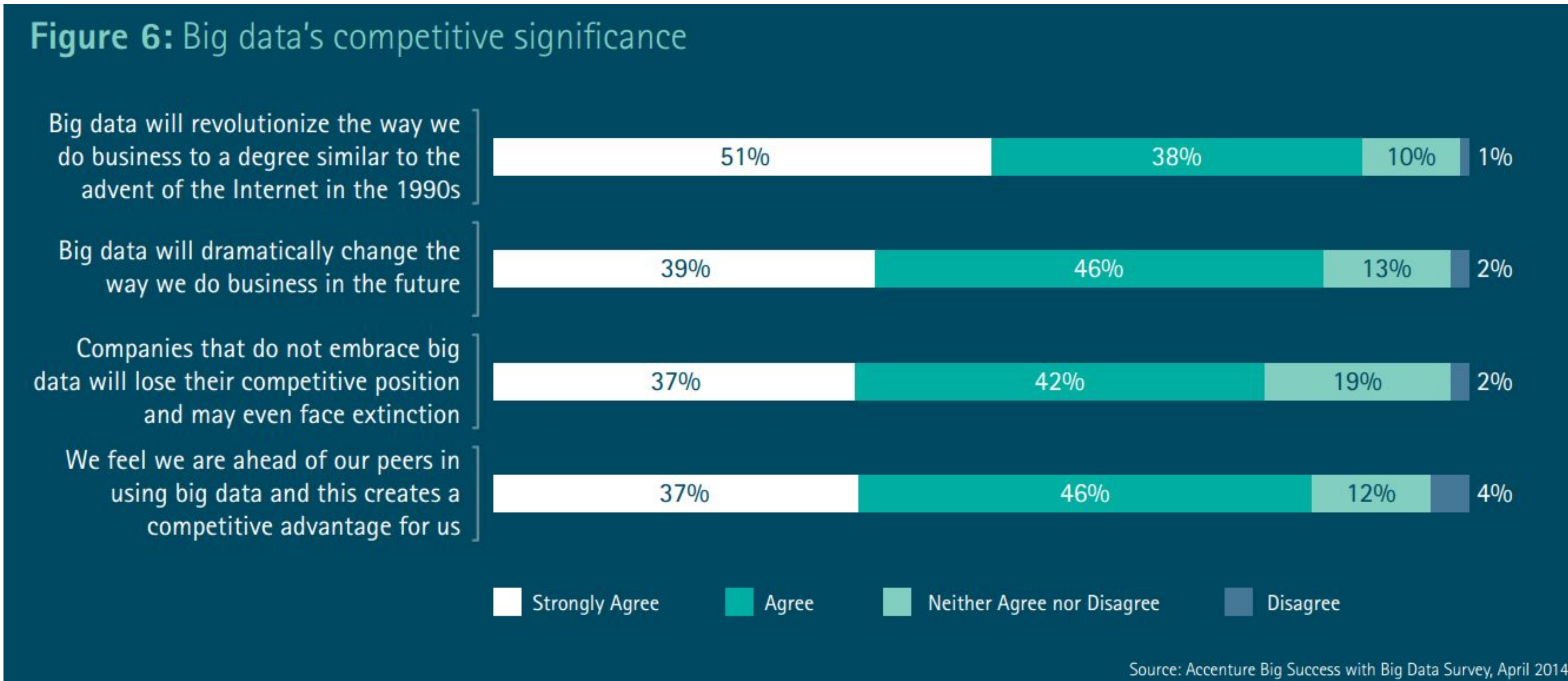
- Realization of what subfields are really doing “big data” (i.e. data mining, ML, Statistics, computational social sciences).
- Best practices being synthesized.

(Gartner Hype Cycle)

What’s the BIG deal?!



What's the BIG deal?!



What is Big Data?

What is Big Data?

traditional
computer scientists

data that will not fit
in main memory.

What is Big Data?

traditional
computer scientists

data that will not fit
in main memory.

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

statisticians

What is Big Data?

data that will not fit in main memory.

traditional computer scientists

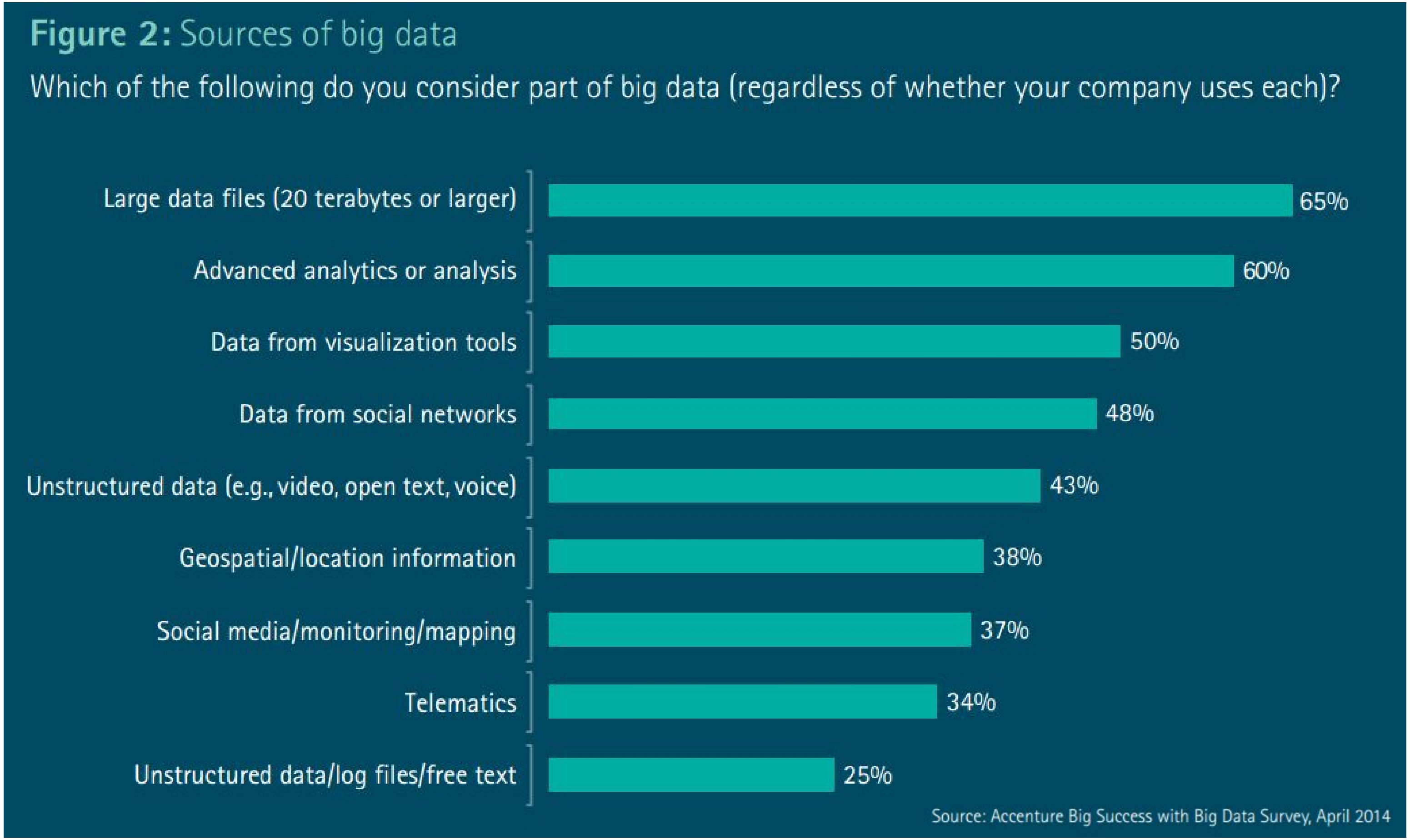
data with a *large* number of observations and/or features.

statisticians

other fields

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

What is Big Data? Industry view:



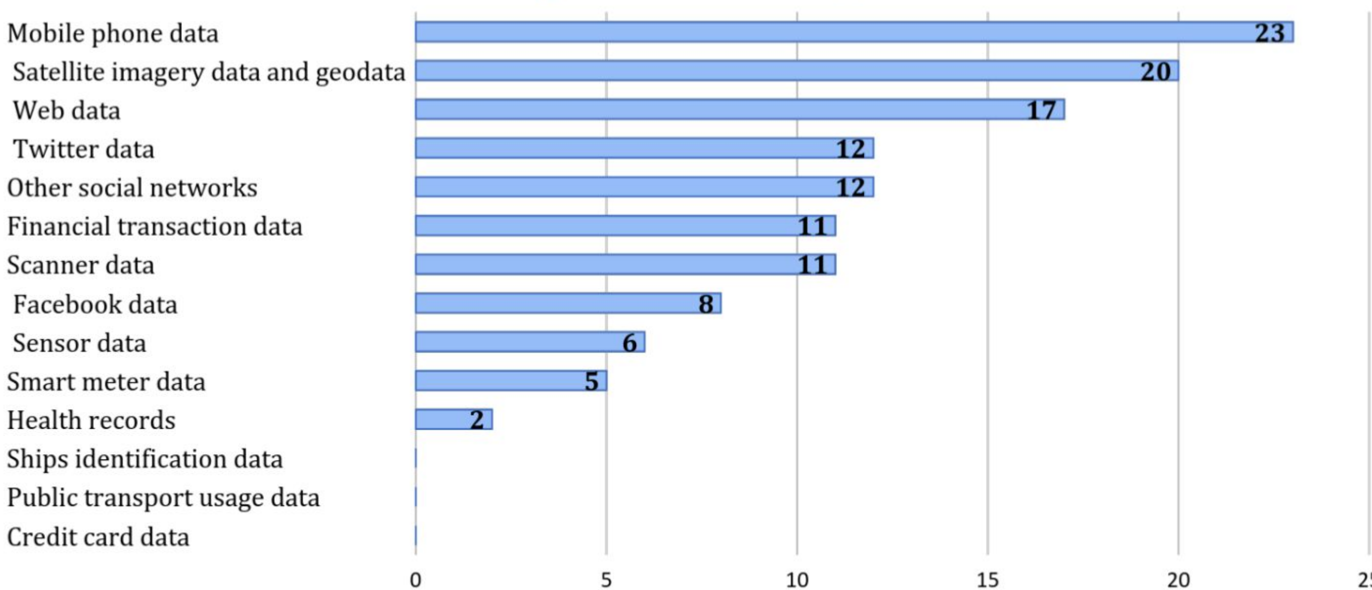
What is Big Data? Government view:

THE WORLD BANK
IBRD • IDA | WORLD BANK GROUP

BigData
UN Global Working Group

1. Survey of SDG-related Big Data projects

Type of data source(s)



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

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?

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

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?

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 (insights)

2. Predictive analytics

Big Data Analytics -- The Class

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

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*

Big Data Analytics -- The Class

We will learn:

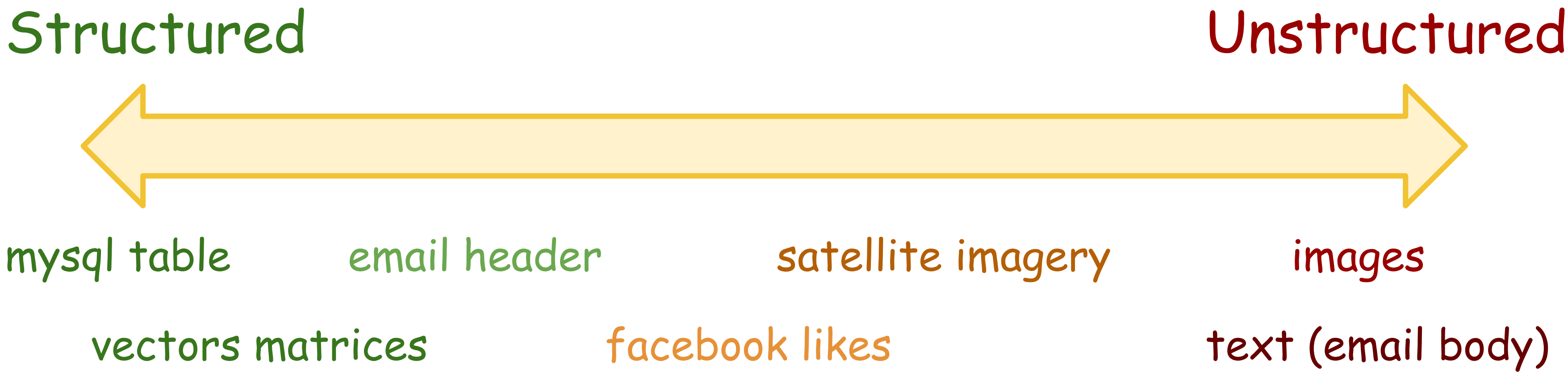
- to solve real-world problems
  - Recommendation systems
  - Market-basket analysis
  - Spam and duplicate document detection
  - *Geo-coding data*
  - *Estimating financial risk*
- uses of various “tools”:
  - linear algebra
  - optimization
  - dynamic programming
  - hashing
  - *Monte-Carlo simulations*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

Preliminaries

Ideas and methods that will repeatedly appear:

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

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

Statistical Limits

Bonferroni's Principle

Statistical Limits

Which iphone case will be least popular?

Bonferroni's Principle

Red

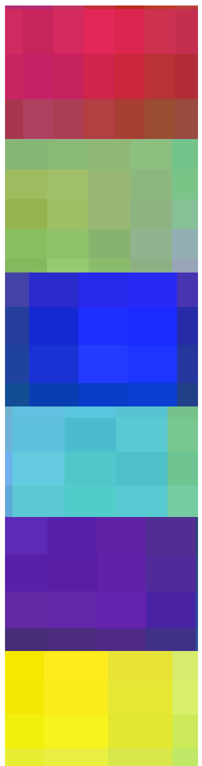
Green

Blue

Teal

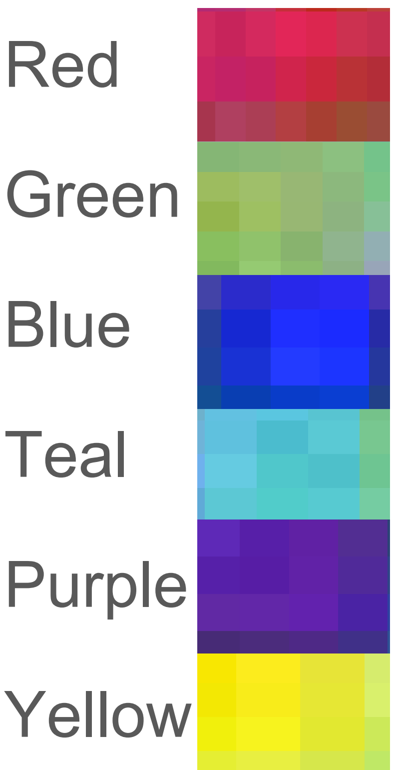
Purple

Yellow



Statistical Limits

Bonferroni's Principle



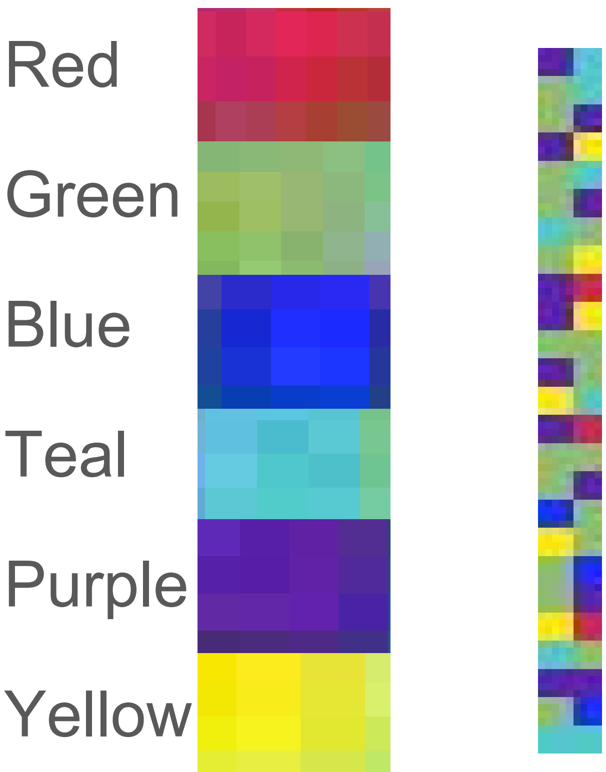
Which iphone case will be least popular?

First 10 sales come in:

Can you make any conclusions?

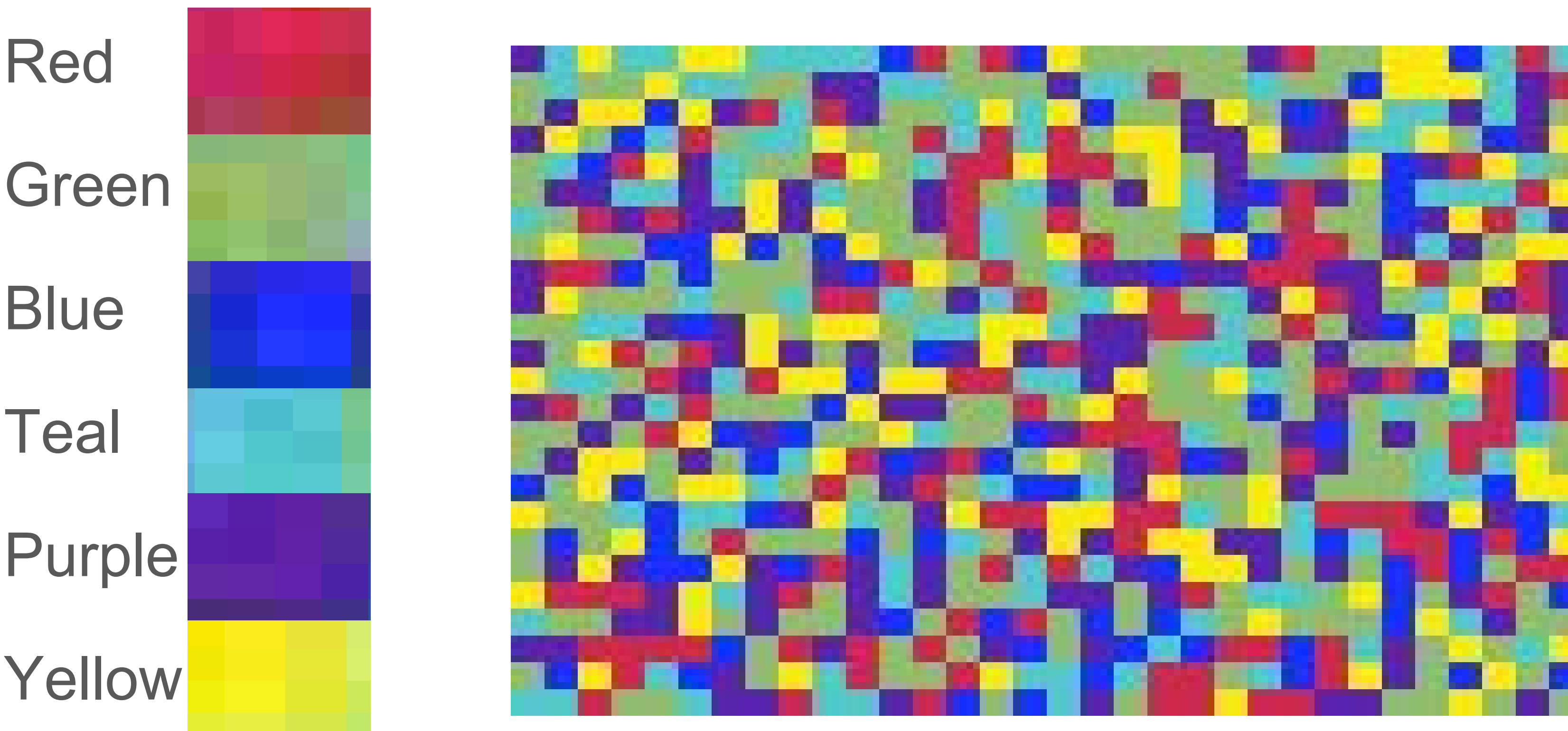
Statistical Limits

Bonferroni's Principle



Statistical Limits

Bonferroni's Principle



Statistical Limits

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 you find something just by chance.

"Data mining" was originally a bad word!

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:

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

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

Inverse Document Frequency:

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

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.

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

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

Hash Functions and Indexes

Review:

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

Objective: send the same number of expected hash-keys to each bucket

Example: storing word counts.

Hash Functions and Indexes

Review:

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

Objective: send the same number of expected hash-keys to each bucket

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: send the same number of expected hash-keys to each bucket

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: send the same number of expected hash-keys to each bucket

Example: storing word counts.

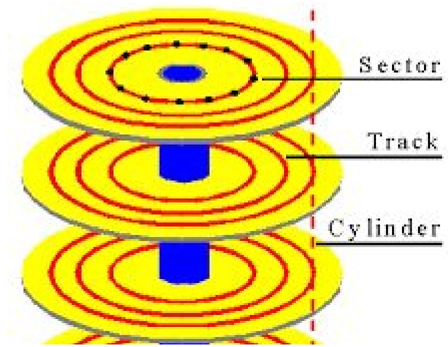
Indexes: Retrieve all records with a given *value.*
(also review if unfamiliar / forgot)

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).

Power Law

Many frequency patterns tend to follow a power law 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”]

(many popularity based statistics, especially without limits)

Power Law

Review

Power Law: $$\log y = b + a \log x$$

raising to the natural log:

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

where c is just a constant

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