



## DATA605: Big Data Systems

### Introduction

**Instructor:** Dr. GP Saggese - [gsaggese@umd.edu](mailto:gsaggese@umd.edu)

**References:**

- *Class Intro*
- Big Data
- Data Models

# Invariants of a Class Lecture

---

- **Invariants**

- Focus on intuition over math
- Emphasize realistic assumptions, numerical methods
  - Analytical solutions are outdated
- Interactive Jupyter notebook tutorials
  - Tutorials done at home
  - Videos added over time

- **Class flow**

- Alternate between slides, whiteboard, tutorials

- **Labs**

- Review complete class project examples
- Collaborate on class project



# Books of the Class

---

- Slides
  - Are extracted from books, technical articles, Internet
  - Should be self-sufficient

# Learning Outcomes

---

- Model and reason about data
- Process and manipulate data
  - E.g., Python, Pandas
- Introduce a variety of data models
  - E.g., relational, NoSQL, graph DBs
  - Decide appropriate data model for different applications
- Use data management systems
  - E.g., PostgreSQL, MongoDB, HBase
  - Decide appropriate system for scenarios
- Build data processing pipelines
  - E.g., Docker, Airflow
- Build a big-data system end-to-end
  - Class project
  - Contribute to an open-source project



# Tools We Will Learn To Use

---

- **Programming languages**
  - Python
- **Development tools**
  - Bash/Linux OS
  - Git: data model, branching
  - GitHub: Pull Requests (PR), issues
  - Jupyter notebooks
  - Docker
- **Big data tools**
  - Extract-Transform-Load (ETL) pipelines
  - Relational DBs (PostgreSQL)
  - NoSQL DBs (HBase, MongoDB, Couchbase, Redis)
  - Graph DBs (Neo4j, GraphX, Giraph)
  - Computing framework (Hadoop, Spark, Dask)
  - Workflow manager (Airflow)
  - Cloud services (AWS)
- **Tutorials** for tools used in the class projects

# Todos

---

- Study slides and materials
- DATA605 - ELMS/Canvas site
  - Enable notifications
  - Contact info for me/TAs
- DATA605 - Schedule
- DATA605 - GitHub repo
- Setup computing environment
  - Install Linux/VMware
  - Install Docker on laptop
  - Instructions in class repo
- Bring laptop to class
  - Quizzes at class start
- Lessons recorded
  - Attend class!

# Grading

---

- **Quizzes**
  - 40% of grade
  - Multi-choice on previous 2 lessons
  - 20 questions in 20 minutes
  - 4-5 quizzes to encourage study during semester
- **Final Project**
  - 60% of grade
  - Comprehensive application of course concepts
  - Big data project in Python from a list of topics
  - Individual or group

# Class Projects

---

- The project is “*Build X with Y*”, where *X* is a “use case” and *Y* is a “technology”
  - Study and describe technology *Y*
  - Implement use case *X* using technology *Y*
  - Create Jupyter notebooks to demo your project
  - Commit code to GitHub, contribute to open-source repo
  - Write a blog entry
  - Present your project in a video
- Choose from list of *X* and *Y*, e.g.,
  - Big data
  - Large language models
  - ...
- Each project:
  - Individual or group ( $n < 4$ )
  - Varying difficulty levels



# Soft Skills to Succeed in the Workplace

---

- **Goal:** model class project for workplace preparation
  - Work in a team
  - Design software architecture (OOP, Agile, Design Patterns)
  - Comment your code
  - Write external documentation (tutorials, manuals, how-tos)
  - Write understandable code (including for future-you)
  - Read others' code
  - Follow code conventions (PEP8, Google Code)
  - Communicate clearly (emails, Slack)
  - File a bug report
  - Reproduce a bug
  - Intuition of CS constants
  - Basic understanding of OS (virtual memory, processes)



# Links

---

- [ELMS](#)
- [Syllabus](#)
  - Schedule
  - GitHub project
  - Class FAQs
- [Project specs](#)

# Yours Truly

---

- **GP Saggese**
  - 2001-2006, PhD / Postdoc at the University of Illinois at Urbana-Champaign
  - [LinkedIn](#)
  - [gsaggese@umd.edu](mailto:gsaggese@umd.edu)
- **University of Maryland:**
  - 2023-, Lecturer for UMD DATA605: Big Data Systems
  - 2025-, Lecturer for UMD MSML610: Advanced Machine Learning
- **In the real-world**
  - Research scientist at NVIDIA, Synopsys, Teza, Engineers' Gate
  - 3x AI and fin-tech startup founder (ZeroSoft, June, Causify AI)
  - 20+ academic papers, 2 US patents



- Class Intro
- *Big Data*
- Data Models

# Data Science

---

- **Promises of data science**

- Give a competitive advantages
- Make better strategic and tactical business decisions
- Optimize business processes

- **Data science is not new**, it was called:

- Operation research (~1970-80s)
- Decision support, business intelligence (~1990s)
- Predictive analytics (Early 2010s)
- ...

- **What has changed**

- Now learning and applying data science is **easy**
  - No need for hiring a consulting company
- Tools are open-source
  - E.g., Python + pydata stack (numpy, scipy, Pandas, sklearn)
- Large data sets available
- Cheap computing
  - E.g., cloud computing (AWS, Google Cloud), GPUs



# Motivation: Data Overload

---

- “*Data science is the number one catalyst for economic growth*” (McKinsey, 2013)
- **Explosion of data in every domain**
  - Sensing devices/networks monitor processes 24/7
    - E.g., temperature of your room, your vital signs, pollution in the air
  - Sophisticated smart-phones
    - 80% of the world population has a smart-phone
  - Internet and social networks make it easy to publish data
  - Internet of Things (IoT): everything is connected to the internet
    - E.g., power supply, toasters
  - Datafication turns all aspects of life into data
    - E.g., what you like/enjoy turned into a stream of your “likes”
- **Challenges**
  - How to handle the increasing amount data?
  - How to extract actionable insights and scientific knowledge from data?

# Scale of Data Size

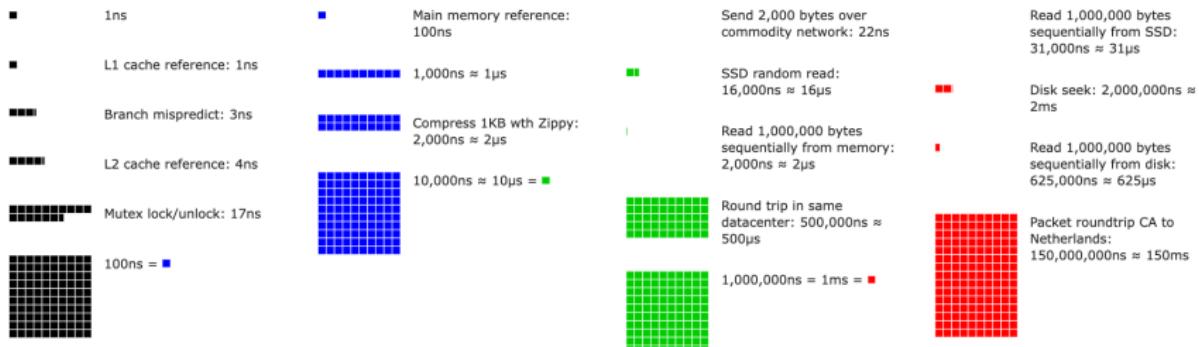
---

- **Megabyte** =  $2^{20} \approx 10^6$  bytes
  - Typical English book
- **Gigabyte** =  $10^9$  bytes = 1,000 MB
  - 1/2 hour of video
  - Wikipedia (compressed, no media) is 22GB
- **Terabyte** = 1 million MB
  - Human genome: ~1 TB
  - 100,000 photos
  - \$50 for 1TB HDD, \$23/mo on AWS S3
- **Petabyte** = 1000 TB
  - 13 years of HD video
  - \$250k/year on AWS S3
- **Exabyte** = 1M TB
  - Global yearly Internet traffic in 2004
- **Zetabyte** = 1B TB =  $10^{21}$  bytes
  - Global yearly Internet traffic in 2016
  - Fill 20% of Manhattan, New York with data centers
- **Yottabytes** =  $10^{24}$  bytes
  - Yottabyte costs \$100T
  - Fill Delaware and Rhode Island with a million data centers
- **Brontobytes** =  $10^{27}$  bytes



# Constants Everybody Should Know

- CPU at 3GHz: 0.3 ns per instruction
- L1 cache reference/register: 1 ns
- L2 cache reference: 4 ns
- Main memory reference: 100 ns
- Read 1MB from memory: 20-100 us
- SSD random read: 16 us
- Send 1KB over network: 1 ms
- Disk seek: 2 ms
- Packet round-trip CA to Netherlands: 150 ms



# Big Data Applications

---

- **Personalized marketing**
- Target each consumer individually
  - E.g., Amazon personalizes suggestions using:
    - Shopping history
    - Search, click, browse activity
    - Other consumers and trends
    - Reviews (NLP and sentiment analysis)
- Brands understand customer-product relationships
  - Use sentiment analysis from:
    - Social media, online reviews, blogs, surveys
    - Positive, negative, neutral sentiment
- E.g.,
  - In 2022, \$600B spent on digital marketing

# Big Data Applications

- **Mobile advertisement**
- Mobile phones are ubiquitous
  - 80% of world population has one
  - 6.5 billion smartphones
- Integrate online and offline databases,  
e.g.,
  - GPS location
  - Search history
  - Credit card transactions
- E.g.,
  - You've bought a new house
  - You google questions about house renovations
  - You watch shows about renovations
  - Your phone tracks where you are
  - Google sends you coupons for the closest Home Depot
  - *"I feel like Google is following me"*



# Big Data Applications

---

- **Biomedical data**
- Personalized medicine
  - Patients receive treatment tailored to them for efficacy
  - Genetics
  - Daily activities
  - Environment
  - Habits
- Genome sequencing
- Health tech
  - Personal health trackers (e.g., smart rings, phones)



# Big Data Applications

---

- **Smart cities**
- Interconnected mesh of sensors
  - E.g., traffic sensors, camera networks, satellites
- Goals:
  - Monitor air pollution
  - Minimize traffic congestion
  - Optimal urban services
  - Maximize energy savings



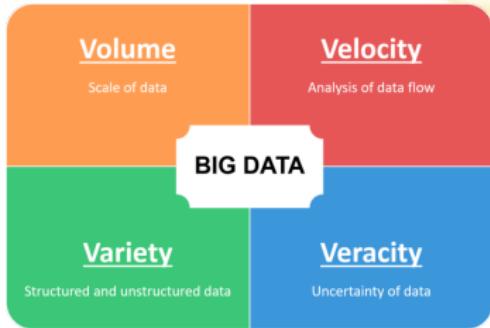
# Goal of Data Science

- **Goal:** from data to wisdom
  - Data (raw bytes)
  - Information (organized, structured)
  - Knowledge (learning)
  - Wisdom (understanding)
- Insights enable decisions and actions
- Combine streams of big data to generate new data
  - New data can be “big data” itself



# The Six V'S of Big Data

- **Volume**
  - Vast amount of data is generated
- **Variety**
  - Different forms
- **Velocity**
  - Speed of data generation
- **Veracity**
  - Biases, noise, abnormality in data
  - Uncertainty, trustworthiness
- **Valence**
  - Connectedness of data in the form of graphs
- **Value**
  - Data must be valuable
  - Benefit an organization



# The Six V's of Big Data

---

- **Volume**

- Exponentially increasing data
- 2.5 exabytes (1m TB) generated daily
  - 90% of data generated in last 2 years
  - Data doubles every 1.2 years
- Twitter/X: 500M tweets/day (2022)
- Google: 8.5B queries/day (2022)
- Meta: 4PB data/day (2022)
- Walmart: 2.5PB unstructured data/hour (2022)

- **Variety**

- Different data forms
  - Structured (e.g., spreadsheets, relational data)
  - Semi-structured (e.g., text, sales receipts, class notes)
  - Unstructured (e.g., photos, videos)
- Different formats (e.g., binary, CSV, XML, JSON)



# The Six V's of Big Data

---

- **Velocity**

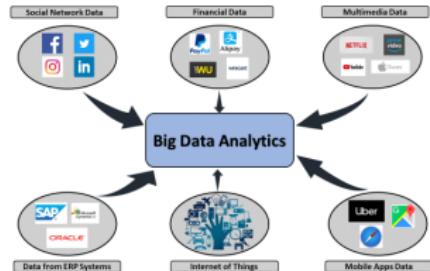
- Speed of data generation
  - E.g., sensors generate data streams
- Process data off-line or in real-time
- Real-time analytics: consume data as fast as generated

- **Veracity**

- Relates to data quality
- How to remove noise and bad data?
- How to fill in missing values?
- What is an outlier?
- How do you decide what data to trust?

# Sources of Big Data

- Distinguish Big Data by source
  - **Machines**
  - **People**
  - **Organizations**



# Sources of Big Data: Machines

---

- **Machines** generate data
  - Real-time sensors (e.g., sensors on Boeing 787)
  - Cars
  - Website tracking
  - Personal health trackers
  - Scientific experiments
- **Pros**
  - Highly structured
- **Cons**
  - Difficult to move, computed in-place or centralized
  - Streaming, not batch

# Sources of Big Data: People

- **People** and their activities generate data
    - Social media (Instagram, Twitter, LinkedIn)
    - Video sharing (YouTube, TikTok)
    - Blogging, website comments
    - Internet searches
    - Text messages (SMS, Whatsapp, Signal, Telegram)
    - Personal documents (Google Docs, emails)
  - **Pros**
    - Enable personalization
    - Valuable for business intelligence
  - **Cons**
    - Semi-structured or unstructured data
      - Text, images, movies
    - Requires investment to extract value
      - Acquire → Store → Clean → Retrieve → Process → Insights
- SCIENCE ACADEMY Surveillance capitalism



# Sources of Big Data: Orgs

---

- **Organizations** generate data

- Commercial transactions
- Credit cards
- E-commerce
- Banking
- Medical records
- Website clicks

- **Pros**

- Highly structured

- **Cons**

- Store every event to predict future
  - Miss opportunities
- Stored in “data silos” with different models
  - Each department has own system
  - Additional complexity
  - Data outdated/not visible
  - Cloud computing helps (e.g., data lakes, data warehouses)



# Is Data Science Just Hype?

- Big data (or data science)
  - “Any process where interesting information is inferred from data”
- Data scientist called the “sexiest job” of the 21st century
  - The term has become very muddled at this point
- **Is it all hype?**

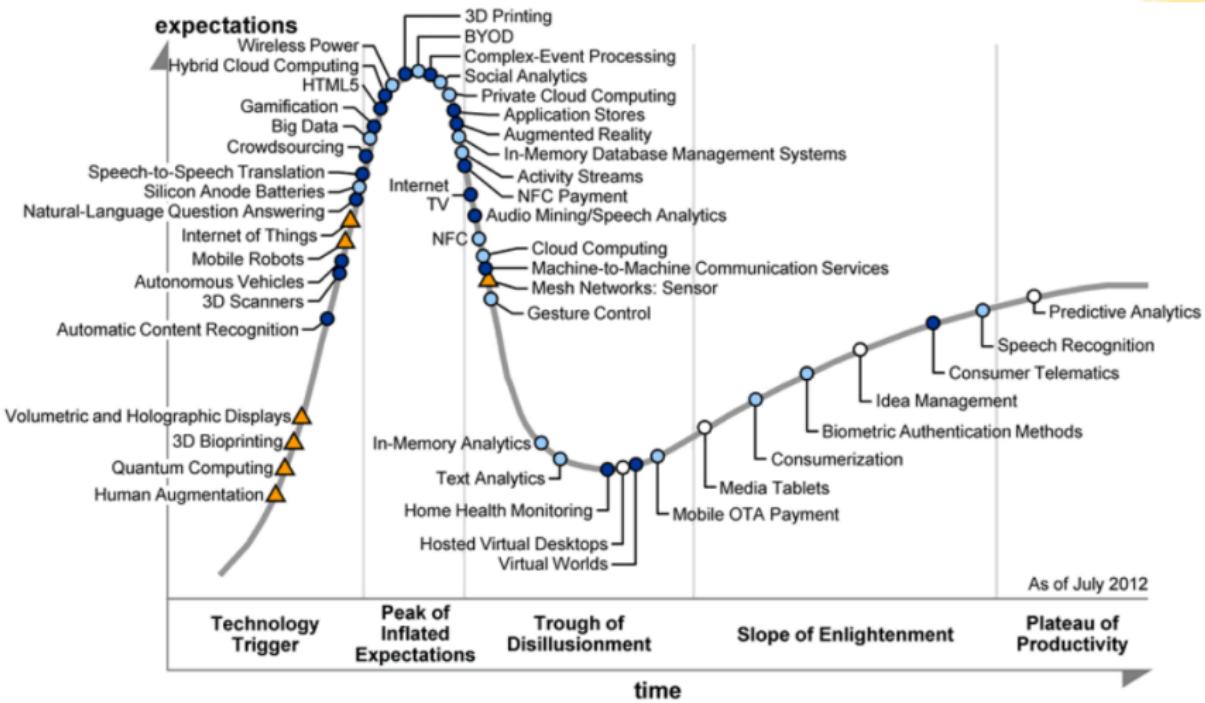


# Is Data Science Just Hype?

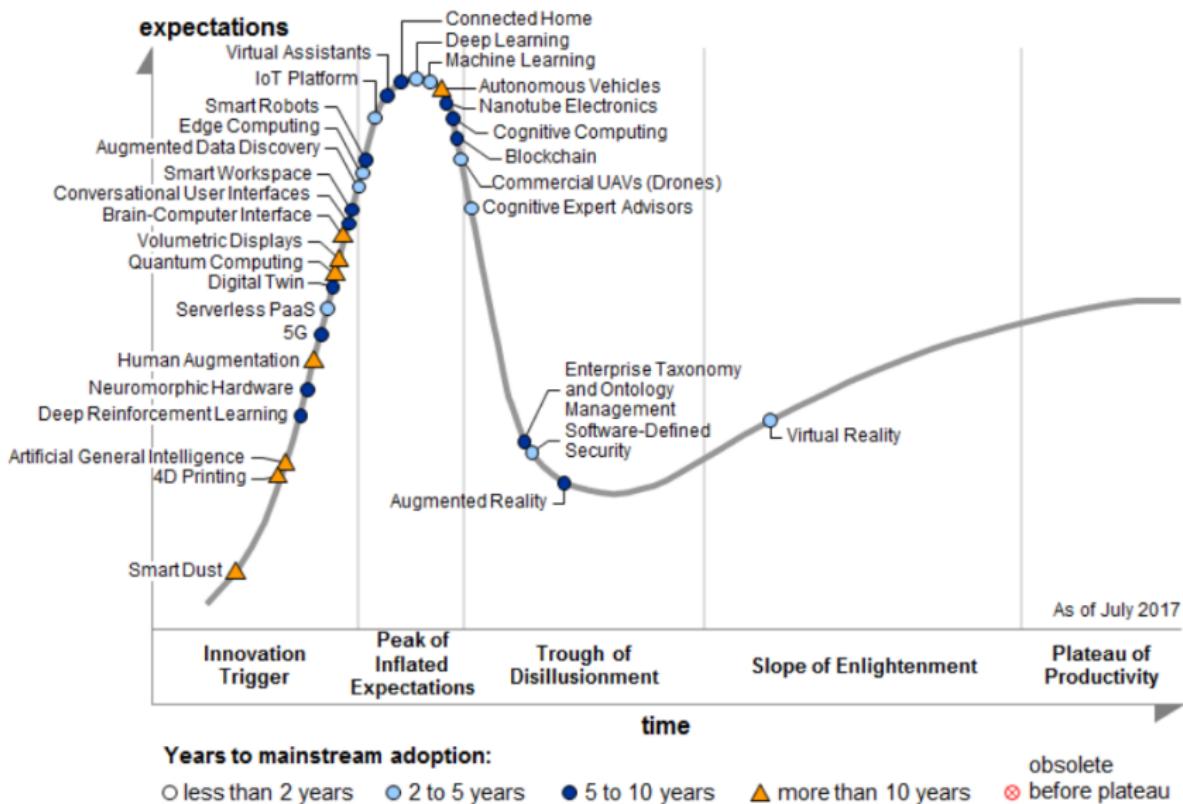
---

- **No**
  - Extract insights and knowledge from data
  - Big data techniques revolutionize many domains
    - E.g., education, food supply, disease epidemics
- **But**
  - Similar to what statisticians have done for years
- **What is different?**
  - More data is digitally available
  - Easy-to-use programming frameworks (e.g., Hadoop) simplify analysis
  - Cloud computing (e.g., AWS) reduces costs
  - Large-scale data + simple algorithms often outperform small data + complex algorithms

# What Was Cool in 2012?

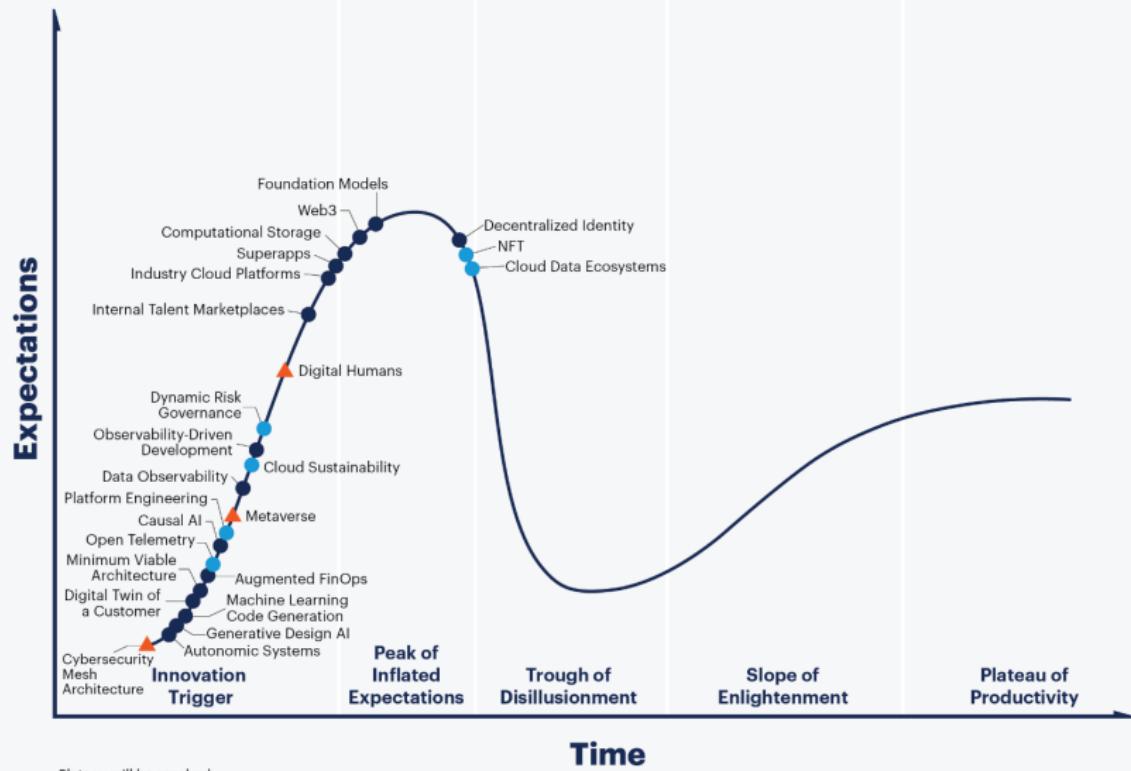


# What Was Cool in 2017?



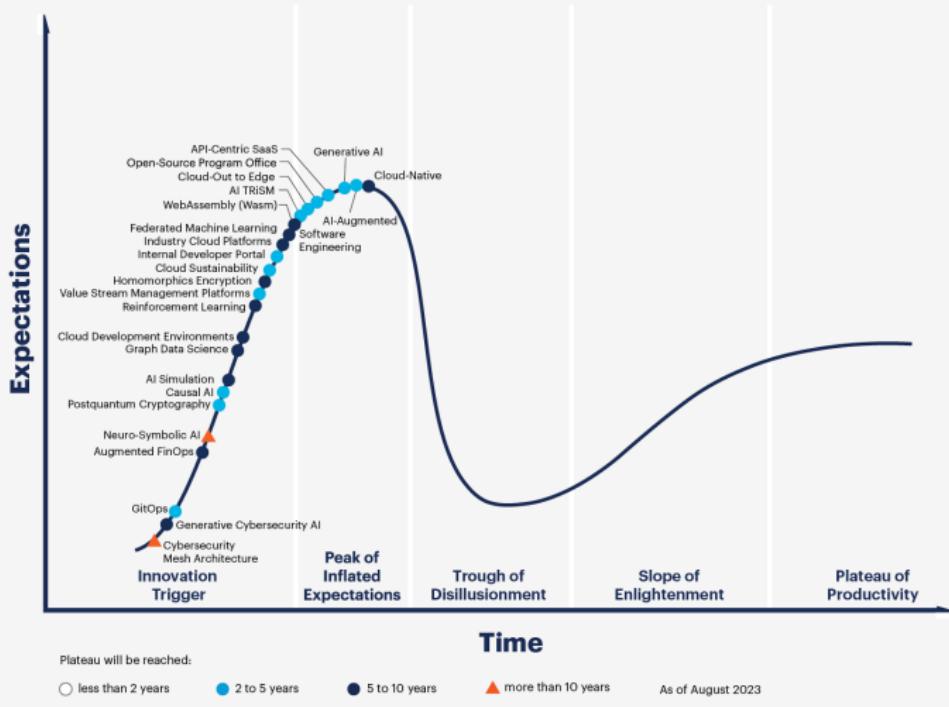
# What Was Cool in 2022?

## Hype Cycle for Emerging Tech, 2022



# What Was Cool in 2023?

## Hype Cycle for Emerging Technologies, 2023



# Key Shifts Before/After Big-Data

---

- **Datasets: small, curated, clean → large, uncurated, messy**
  - Before:
    - Statistics based on small, carefully collected random samples
    - Costly and careful planning for experiments
    - Hard to do fine-grained analysis
  - Today:
    - Easily collect huge data volumes
    - Feed into algorithms
    - Strong signal overcomes noise
- **Causation → Correlation**
  - Goal: determine cause and effect
  - Causation hard to determine → focus on correlation
    - Correlation is often sufficient
    - E.g., diapers and beer bought together
- **"Data-fication"**
  - = converting abstract concepts into data
  - E.g., "sitting posture" data-fied by sensors in your seat
  - Preferences data-fied into likes
- From: Rise of Big Data, 2013



# Examples: Election Prediction

- Nate Silver and the 2012 Elections
  - Predicted 49/50 states in 2008 US elections
  - Predicted 50/50 states in 2012 US elections
- Reasons for accuracy
  - Multiple data sources
  - Historical accuracy incorporation
  - Statistical models
  - Understanding correlations
  - Monte-Carlo simulations for electoral probabilities
  - Focus on probabilities
  - Effective communication



# Examples: Google Flu Trends

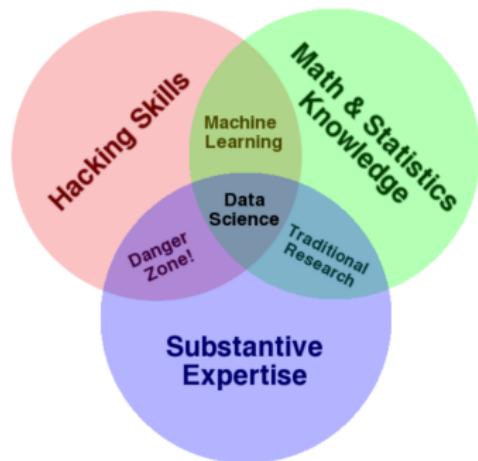
---

- 5% to 20% of US population contracts flu yearly; 40k deaths
- Early warnings enable prevention and control
- Google Flu Trends
  - Early flu outbreak warnings via search query analysis
    - 45 search terms analyzed
    - IP used to determine location
  - Predict regional flu outbreaks 1-2 weeks before CDC
  - Active from 2008 to 2015
- Caveat: accuracy declined
  - Claimed 97% accuracy
  - Out of sample accuracy lower (overshot CDC data by 30%)
  - People search about flu without knowing diagnosis
    - E.g., searching for “fever” and “cough”
  - Google Flu Trends: The Limits of Big Data

# Data Scientist

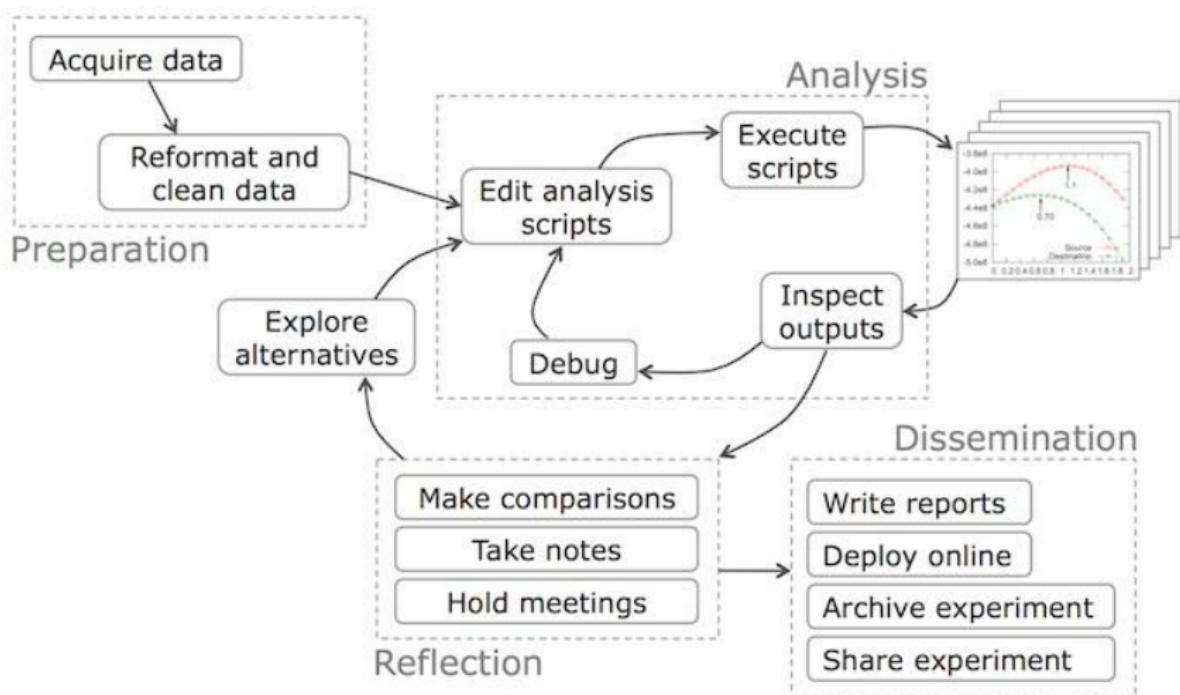
---

- Ambiguous, ill-defined term
- From Drew Conway's Venn Diagram



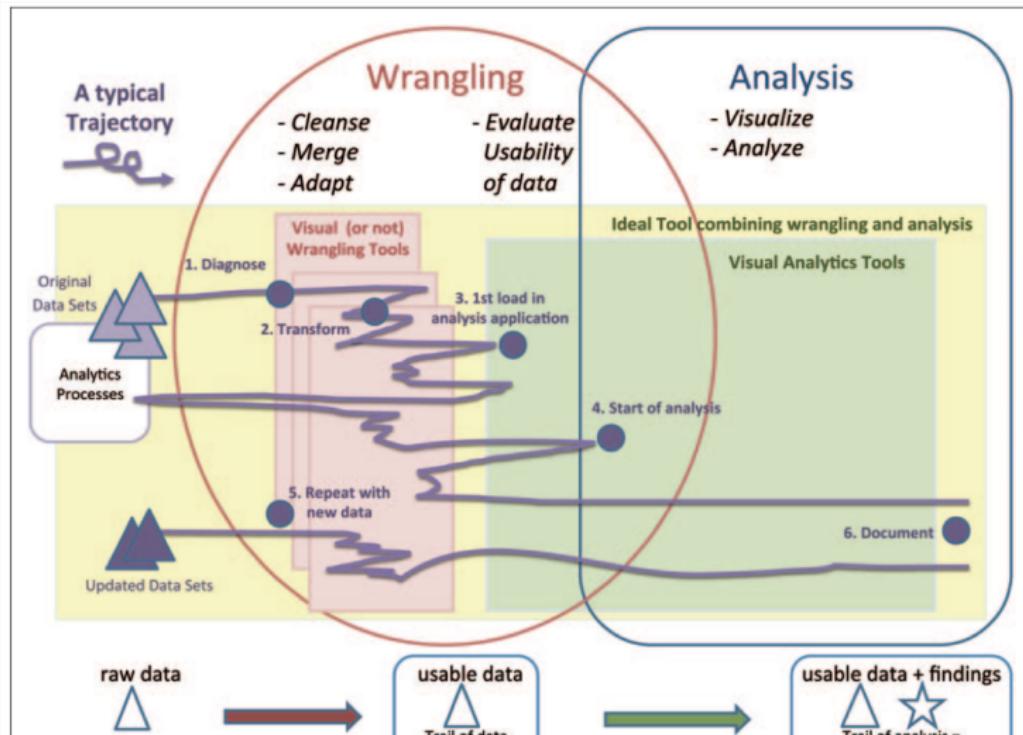
# Typical Data Scientist Workflow

- From Data Science Workflow



# Where Data Scientist Spends Most Time

- 80-90% of the work is data cleaning and wrangling
- “Janitor Work” in Data Science
- Research Directions in Data Wrangling



# What a Data Scientist Should Know

---

- From: How to hire a data scientist
- **Data grappling skills**
  - Move and manipulate data with programming
  - Scripting languages (e.g., Python)
  - Data storage tools: relational databases, key-value stores
  - Programming frameworks: SQL, Hadoop, Spark
- **Data visualization experience**
  - Draw informative data visuals
  - Tools: D3.js, plotting libraries
  - Know what to draw
- **Knowledge of statistics**
  - Error-bars, confidence intervals
  - Python libraries, Matlab, R
- **Experience with forecasting and prediction**
  - Basic machine learning techniques
- **Communication skills**
  - Tell the story, communicate findings



- Class Intro
- Big Data
- *Data Models*

# Data Models

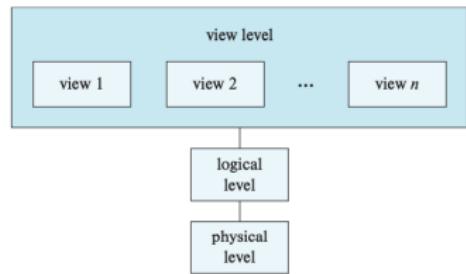
---

- **Data modeling**
  - Represents and captures structure and properties of real-world entities
  - Abstraction: real-world → **representation**
- **Data model**
  - Describes how data is *represented* (e.g., relational, key-value) and *accessed* (e.g., insert operations, query)
  - Schema in a DB describes a specific data collection using a data model
- **Why need data model?**
  - Know data structure to write general-purpose code
  - Share data across programs, organizations, systems
  - Integrate information from multiple sources
  - Preprocess data for efficient access (e.g., building an index)



# Multiple Layers of Data Modeling

- **Physical layer**
  - How is the data physically stored
  - How to represent complex data structures (e.g., B-trees for indexing)
- **Logical layer**
  - Entities
  - Attributes
  - Type of information stored
  - Relationships among the above
- **Views**
  - Restrict information flow
  - Security and/or ease-of-use



# Data Models: Logical Layer

- **Modeling constructs**

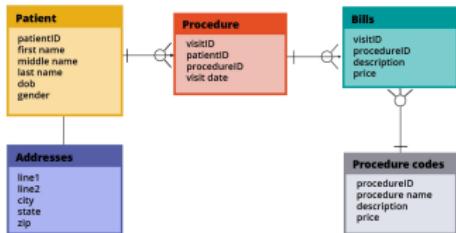
- Concepts to represent data structure
- E.g.,
  - Entity types
  - Entity attributes
  - Relationships between entities
  - Relationships between attributes

- **Integrity constraints**

- Ensure data integrity
  - Avoid errors and inconsistencies
  - E.g., field can't be empty, must be an integer

- **Manipulation constructs**

- E.g., insert, update, delete data



# Examples of Data Models

---

- We will cover:
  - Relational model (SQL)
  - Entity-relationship (ER) model
  - XML
  - Object-oriented (OO)
  - Object-relational
  - RDF
  - Property graph
- Serialization formats as data models
  - CSV
  - Parquet
  - JSON
  - Protocol Buffer
  - Avro/Thrift
  - Python Pickle

# Good Data Models

---

- Data model should be:
  - Expressive
    - Capture real-world data
  - Easy to use
  - Perform well
- Tension between characteristics
  - Powerful models
    - Represent more datasets
    - Harder to use/query
    - Less efficient (e.g., more memory, time)
- Evolution of data modeling tools captures data structure
  - Structured data → Relational DBs
  - Semi-structured web data → XML, JSON
  - Unstructured data → NoSQL DBs



# Data Independence

---

- **Logical data independence**

- Change data representation without altering programs
- E.g., API abstracting backend

- **Physical data independence**

- Change data layout on disk without altering programs
  - Index data
  - Partition/distribute/replicate data
  - Compress data
  - Sort data



# Databases: A Brief History

---

- **1960s: Early beginning**
  - Computers become attractive technology
  - Enterprises adopt computers
  - Applications use own data stores
    - Each application has its own format
    - Data unavailable to other programs
  - **Database:** term for “shared data banks” by multiple applications
    - Define data format
    - Store as “data dictionary” (schema)
    - Implement “database management” software to access data
  - Issues:
    - How to write data dictionaries?
    - How to access data?
    - Who controls the data?
    - E.g., integrity, security, privacy concerns

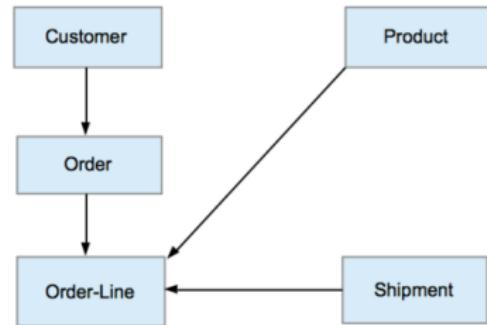
# Databases: A Brief History

---

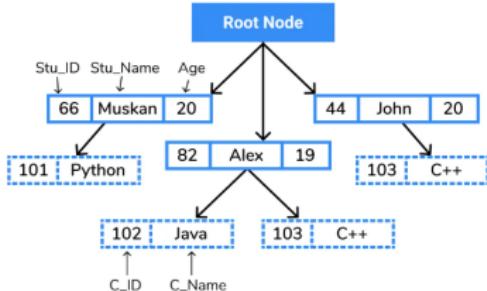
- **1960s, Hierarchical and Network Model**
  - Connect records of different types
  - Example: connect accounts with customers
  - Network model aimed for generality and flexibility
- IBM's IMS Hierarchical Database (1966)
  - Designed for Apollo space program
  - Predates hard disks
  - Used by over 95% of top Fortune 1000 companies
  - Processes 50 billion transactions daily, manages 15 million gigabytes of data
- Cons:
  - Exposed too much internal data (structures/pointers)
  - Leaky abstraction

# Relational, Hierarchical, Network Model

- **Relational model**
  - Data as tuples in relations
  - SQL
- **Hierarchical model**
  - Tree-like structure
    - One parent, many children
    - Connected through links
  - XML DBs resurgence in 1990s
- **Network model**
  - Graph organization
    - Multiple parents and children
  - Graph DBs resurgence in 2010s



Customer ID	Tax ID	Name	Address	[More fields...]
1234567890	555-5512222	Ramesh	323 Southern Avenue	...
2223344556	555-5523232	Adam	1200 Main Street	...
3334445563	555-5533323	Shweta	871 Rani Jhansi Road	...
4232342432	555-5325523	Sarfaraz	123 Maulana Azad Sarani	...



# Databases: A Brief History

---

- **1970s: Relational model**

- Set theory, first-order predicate logic
  - Ted Codd developed the Relational Model
- Elegant, formal model
  - Provided data independence
  - Users didn't worry about data storage, processing
- High-level query language
  - SQL based on relational algebra
- Notion of normal forms
  - Reason about data and relations
  - Remove redundancies

- Influential projects:

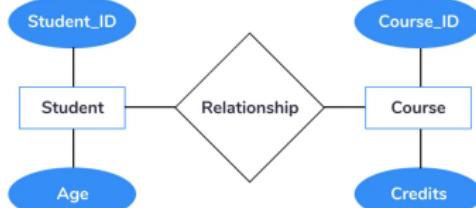
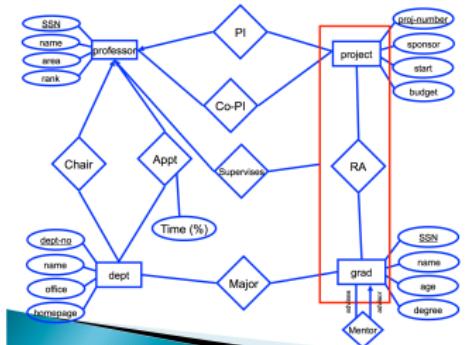
- INGRES (UC Berkeley), System R (IBM)
- Ignored IMS compatibility

- Debates: Relational Model vs Network Model proponents



# Entity-Relationship Model

- 1976: Peter Chen proposed “Entity-Relationship Model”
- Describes knowledge as entities and relationships
- **Entities:** Physical or logical objects, “Nouns”
- **Relationships:** Connections between entities, “Verbs”
- Map ER model to relational DB: Entities, relationships → tables



Attribute

Relationship

Entity

# Databases: A Brief History

---

- **1980s: Relational model acceptance**
  - SQL standard due to IBM's backing
  - Enhanced relational model
    - Set-valued attributes, aggregation
- Late 80's
  - Object-oriented DBs
    - Store objects, not tables
    - Overcome *impedance mismatch* between languages and databases
  - Object-relational DBs
    - User-defined types
    - Combine object-oriented benefits with relational model
  - No expressive difference from pure relational model

# Object-Oriented

---

- OOP is a data model
  - Object behavior described through data (fields) and code (methods)
- **Composition**
  - Aka has-a relationships
  - E.g., Employee class has an Address class
- **Inheritance**
  - Aka is-a relationships
  - E.g., Employee class derives from Person class
- **Polymorphism**
  - Code executed depends on the class of the object
  - One interface, many implementations
  - E.g., draw() method on a Circle vs Square object, both descending from Shape class
- **Encapsulation**
  - E.g., private vs public fields/members
  - Prevents external code from accessing inner workings of an object



# Databases: A Brief History

---

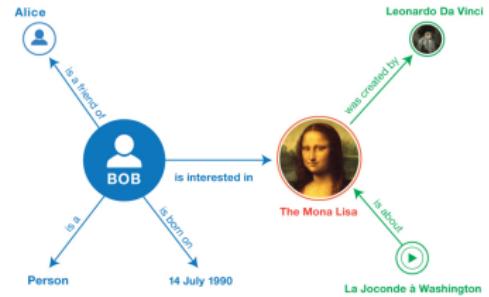
- Late 90's-today
- Web/Internet emerges
- XML: eXtensible Markup Language
  - For *semi-structured* data
  - Tree-like structure
  - Flexible schema

```
<?xml version="1.0" encoding="UTF-8"?>
<CATALOG>
    <CD>
        <TITLE>Empire Burlesque</TITLE>
        <ARTIST>Bob Dylan</ARTIST>
        <COUNTRY>USA</COUNTRY>
        <COMPANY>Columbia</COMPANY>
        <PRICE>10.90</PRICE>
        <YEAR>1985</YEAR>
    </CD>
    <CD>
        <TITLE>Hide your heart</TITLE>
        <ARTIST>Bonnie Tyler</ARTIST>
        <COUNTRY>UK</COUNTRY>
```



# Resource Description Framework

- Aka RDF
- Key construct:  
“subject-predicate-object” triple
  - Subject=sky
  - Predicate=has-the-color
  - Object=blue
- Maps to a labeled, directed multi-graph
  - More general than a tree
- Stored in:
  - Relational DBs
  - Dedicated “triple-stores” DBs



01 <<http://example.org/bob>>  
02 <<http://example.org/bob>>  
03 <<http://example.org/bob>>  
04 <<http://example.org/bob>>  
05 <<http://www.wikidata.org>>  
06 <<http://www.wikidata.org>>  
07 <<http://data.europeana.eu>>

# Property Graph Model

- Graph:
  - Vertices and edges
  - Properties for each edge and vertex
- Stored in:
  - Relational DBs
  - Graph DBs

