Data Science Overview

Why, Where, What, How, Who

Assist.professor Dr.Manirath Wongsim

Outline

- Data Science -- Why all the excitement?
  - history
  - examples
- Where does data come from?
- What is Data Science?
- How to do Data Science?
- Who are Data Scientists?

Data Science – Why all the excitement?

Data Analysis Has Been Around for a While…

1935: "The Design of Experiments"

R.A. Fisher

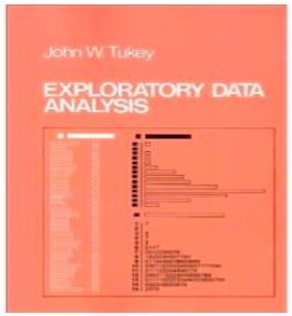
1939: "Quality Control"

W.E. Deming

1958: "A Business Intelligence System"

Peter Luhn

1977: "Exploratory Data Analysis"



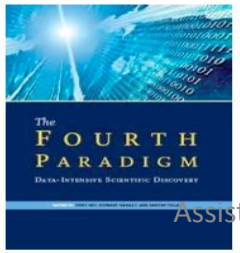
1989: "Business Intelligence"

Howard Dresner

1997: "Machine Learning"

2010:"The Data Deluge"

2007:"The Fourth Paradigm"

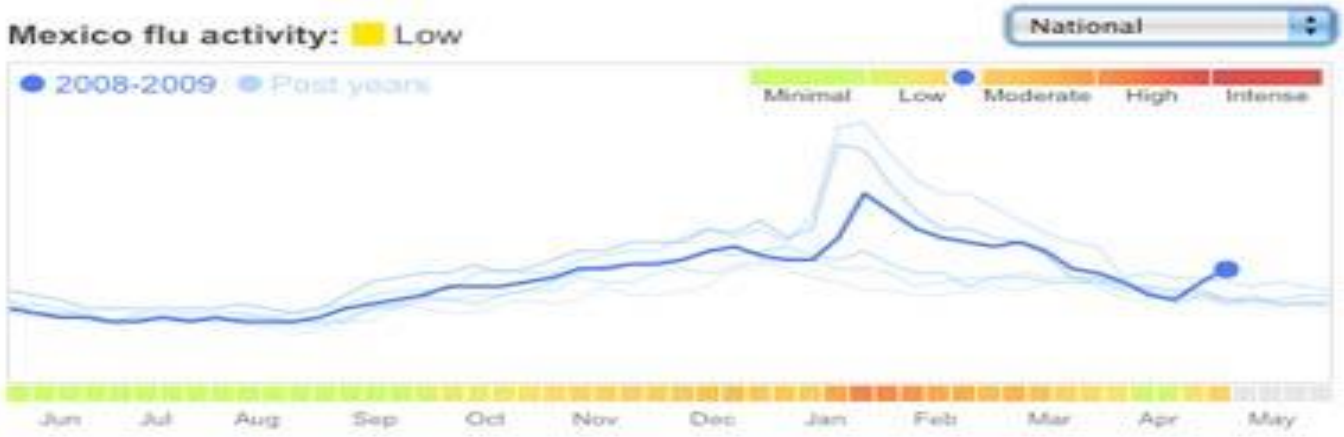


2009:"The Unreasonable Effectiveness of Data"

1996: Google

Google™

Data Science: Why all the Excitement?



Exciting new effective applications of data analytics

e.g.,
Google Flu Trends:

Detecting outbreaks two weeks ahead of CDC data

New models are estimating which cities are most at risk for spread of the Ebola virus.

Prediction model is built on Various data sources, types and analysis.

Why the all the Excitement?

elections2012

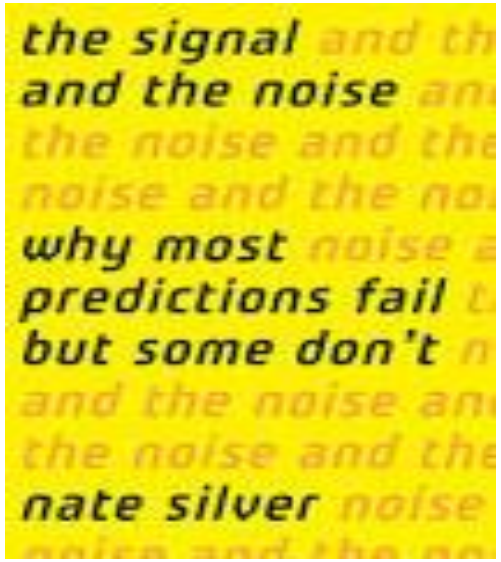
Live results | President | Senate | House | Governor | Choose your

Numbers nerd Nate Silver's forecasts prove all right on election night

FiveThirtyEight blogger predicted the outcome in all 50 states, assuming Barack Obama's Florida victory is confirmed

Luke Harding
guardian.co.uk, Wednesday 7 November 2012 10.45 EST

Predicting political champagne and election Outcome:



Data and Election 2012 (cont.)

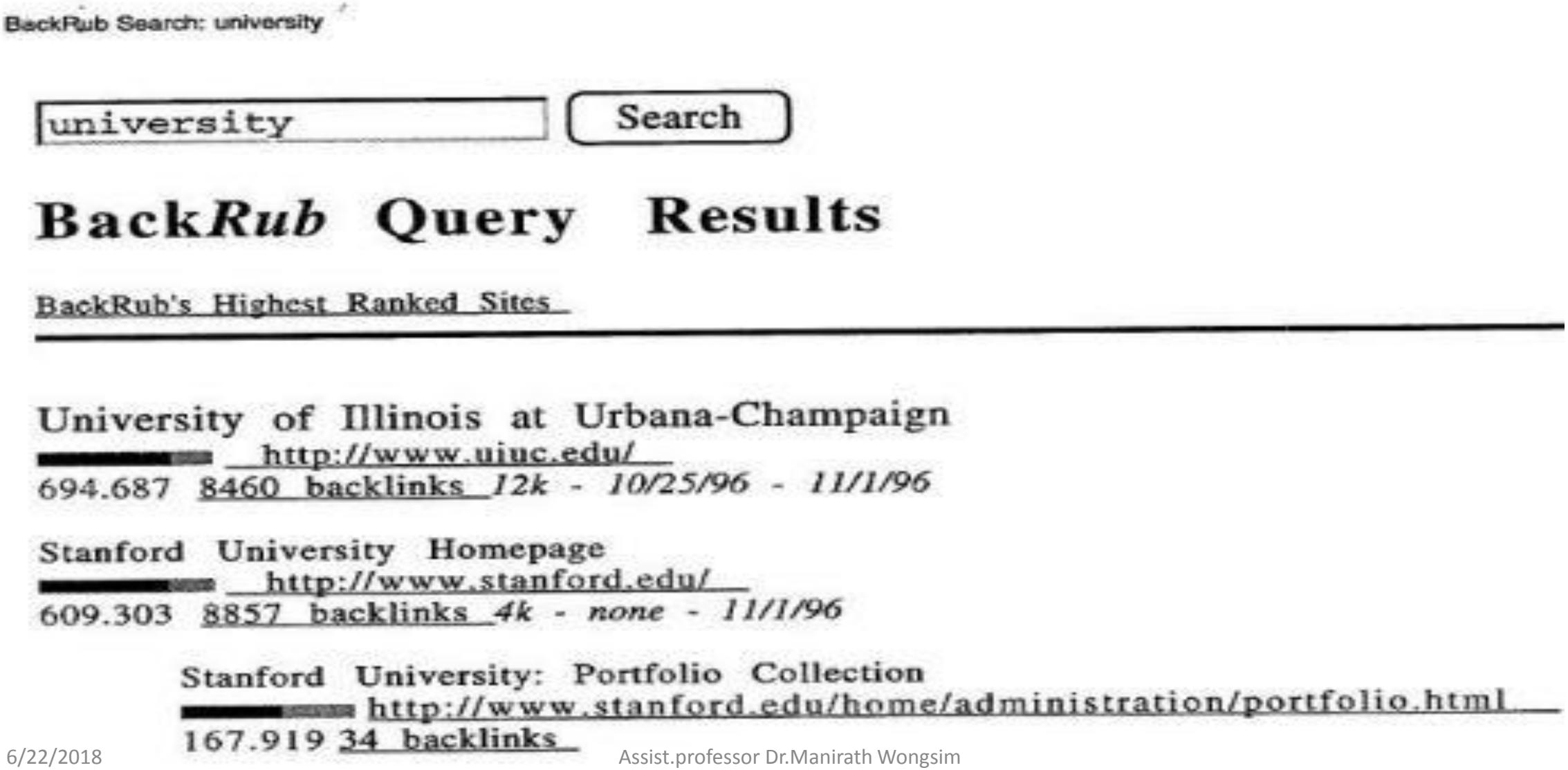
- …that was just one of several ways that Mr. Obama's campaign operations, some unnoticed by Mr. Romney's aides in Boston, helped save the president's candidacy. In Chicago, the campaign recruited a team of behavioral scientists to build an extraordinarily sophisticated database

- …that allowed the Obama campaign not only to alter the very nature of the electorate, making it younger and less white, but also to create a portrait of shifting voter allegiances. The power of this operation stunned Mr. Romney's aides on election night, as they saw voters they never even knew existed turn out in places like Osceola County, Fla.

  -- New York Times, Wed Nov 7, 2012

- The White House Names Dr. DJ Patil as the First U.S. Chief Data Scientist, Feb. 18th 2015

A history of the (Business) Internet: 1997



BackRub Search: university

university Search

BackRub Query Results

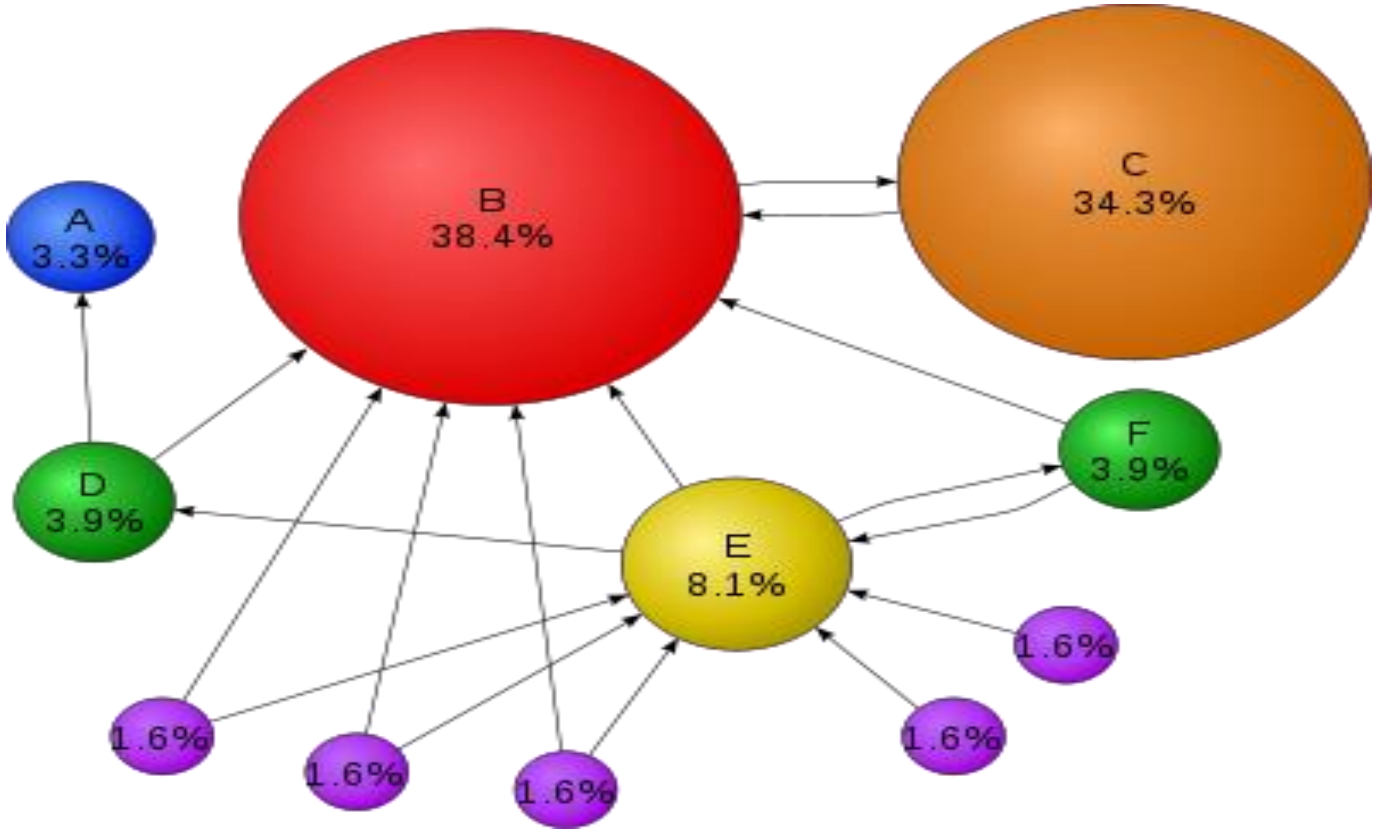
BackRub's Highest Ranked Sites

University of Illinois at Urbana-Champaign
http://www.uiuc.edu/
694.687 8460 backlinks 12k - 10/25/96 - 11/1/96

Stanford University Homepage
http://www.stanford.edu/
609.303 8857 backlinks 4k - none - 11/1/96

Stanford University: Portfolio Collection
http://www.stanford.edu/home/administration/portfolio.html
167.919 34 backlinks

PageRank: The web as a behavioral dataset



Sponsored search

Google gatco towel bars Search Advanced Search

Web Show options... Results 1 - 10 of about 661,000 for gatco towel bars. (0.33 seconds)

Gatco Towel Bars Sponsored Links
www.eFaucets.com/Gatco Low Price Guarantee, Free Shipping. 110% Price Match, No Tax, Shop Now!
Google Checkout

Gatco Towel Bars
www.AmericanHomePlus.com/Gatco 110% Low Price Guarantee. Buy Now. Free Shipping On **Gatco** Accessories.

Gatco Towel Bars
www.PlumberSurplus.com Free S/H Available. Large Selection **Gatco Towel Bars**, ready to ship! Google Checkout

Gatco Towel bars, Gatco toilet paper holders, **Gatco** Bathroom ...
Gatco Towel bars, **Gatco** toilet paper holders, **Gatco** Bathroom Accessories, **Gatco** robe hooks, **Gatco** bathroom shelves, **Gatco** hotel shelves, **Gatco** double **towel** ...
www.kitchensnbath.com/**gatco**-bath-accessories.html - Cached - Similar -

Discount **Towel Bars**, **Towel Bars**, **Towel** Racks
Discount **Towel Bars** offers only the highest quality bath hardware. We are factory direct distributors for Moen, Baldwin, Dynasty Hardware and **Gatco**. ...
www.discount**towelbars**.com/ - Cached - Similar -

Gatco at Lowe's: 24" Franciscan Chrome Double **Towel Bar**
24" Franciscan Chrome Double **Towel Bar** - 69656 5286.
www.lowes.com/lowes/lkn?action=productDetail... - Cached - Similar -

Shopping results for **gatco towel bars**

Gatco Bleu 18 in. **Towel Bar** - Polished Chrome
$39.99 new - Sears
Gatco 4240 24-Inch Latitude II **Towel Bar**, Chrome
$30.53 new - Amazon.com
4621 Camden **Towel Bar** 18 **Gatco** Inc

Sponsored Links

Gatco Bleu 18 in. **Towel** Bar - Polished Chrome
$39.99 - Sears

Gatco Spa **Towel** Rack, 3 Tier - Satin Nickel
$94.99 - Sears

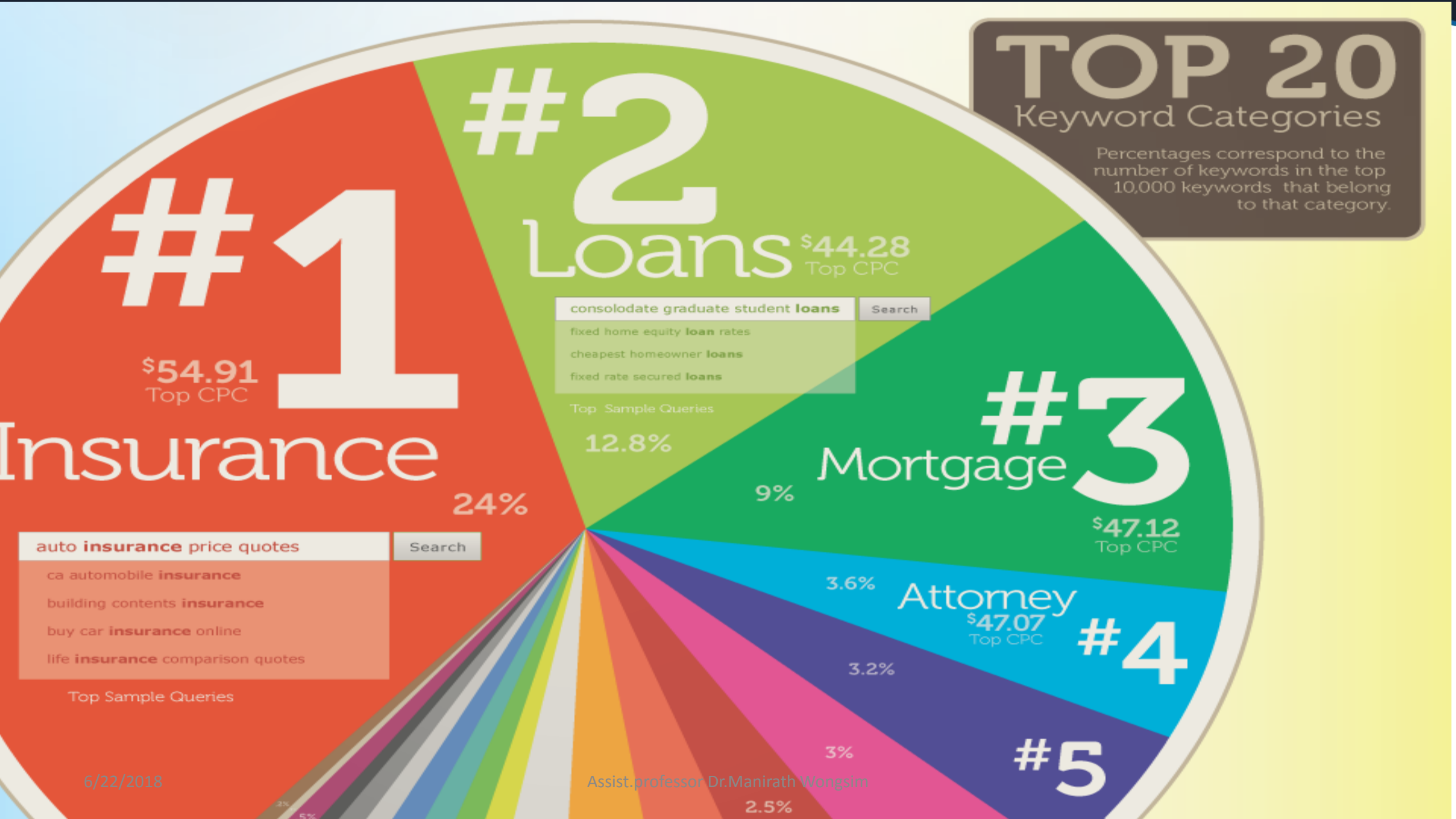
Gatco Chenille 18 in. **Towel** Bar - Vintage ...
$44.99 - Sears

Gatco Bath Accessories
Lowest prices.
All **Gatco** collections
www.TheHomeDecor.net

Towel Bars on Sale
Save 20%-50% Off List- New Styles
Lowest Prices + Free Shipping!
www.FixtureUniverse.com
Google Checkout

Sponsored search

- Google revenue around $50 bn/year from marketing, 97% of the companies revenue.
- Sponsored search uses an auction – a pure competition for marketers trying to win access to consumers.
- In other words, a competition for **models** of consumers – their likelihood of responding to the ad – and of determining the right bid for the item.
- There are around 30 billion search requests a month. Perhaps a **trillion events** of history between search providers.
- Google Adwords and Adsense


TOP 20
Keyword Categories
Percentages correspond to the number of keywords in the top 10,000 keywords that belong to that category.
#1
$54.91
Top CPC
Insurance
24%
auto insurance price quotes
Search
ca automobile insurance
building contents insurance
buy car insurance online
life insurance comparison quotes
Top Sample Queries
#2
Loans
$44.28
Top CPC
consolodate graduate student loans
Search
fixed home equity loan rates
cheapest homeowner loans
fixed rate secured loans
Top Sample Queries
12.8%
#3
Mortgage
9%
$47.12
Top CPC
3.6%
Attorney
$47.07
Top CPC
#4
3.2%
#5
3%
2.5%

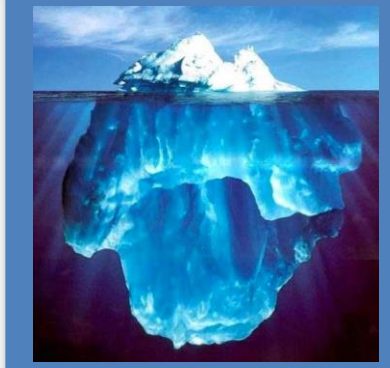
Other Data Science Applications

- Transaction Databases → Recommender systems (NetFlix), Fraud Detection (Security and Privacy)
- Wireless Sensor Data → Smart Home, Real-time Monitoring, Internet of Things
- Text Data, Social Media Data → Product Review and Consumer Satisfaction (Facebook, Twitter, LinkedIn), E-discovery
- Software Log Data → Automatic Trouble Shooting (Splunk)
- Genotype and Phenotype Data → Epic, 23andme, Patient-Centered Care, Personalized Medicine

Where does data come from?

“Big Data” Sources

It’s All Happening On-line



Every:
Click
Ad impression
Billing event
Fast Forward, pause,…
Server request
Transaction
Network message
Fault
…

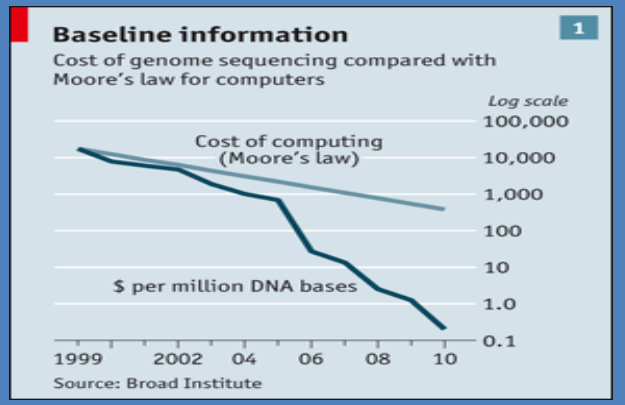
User Generated (Web & Mobile)



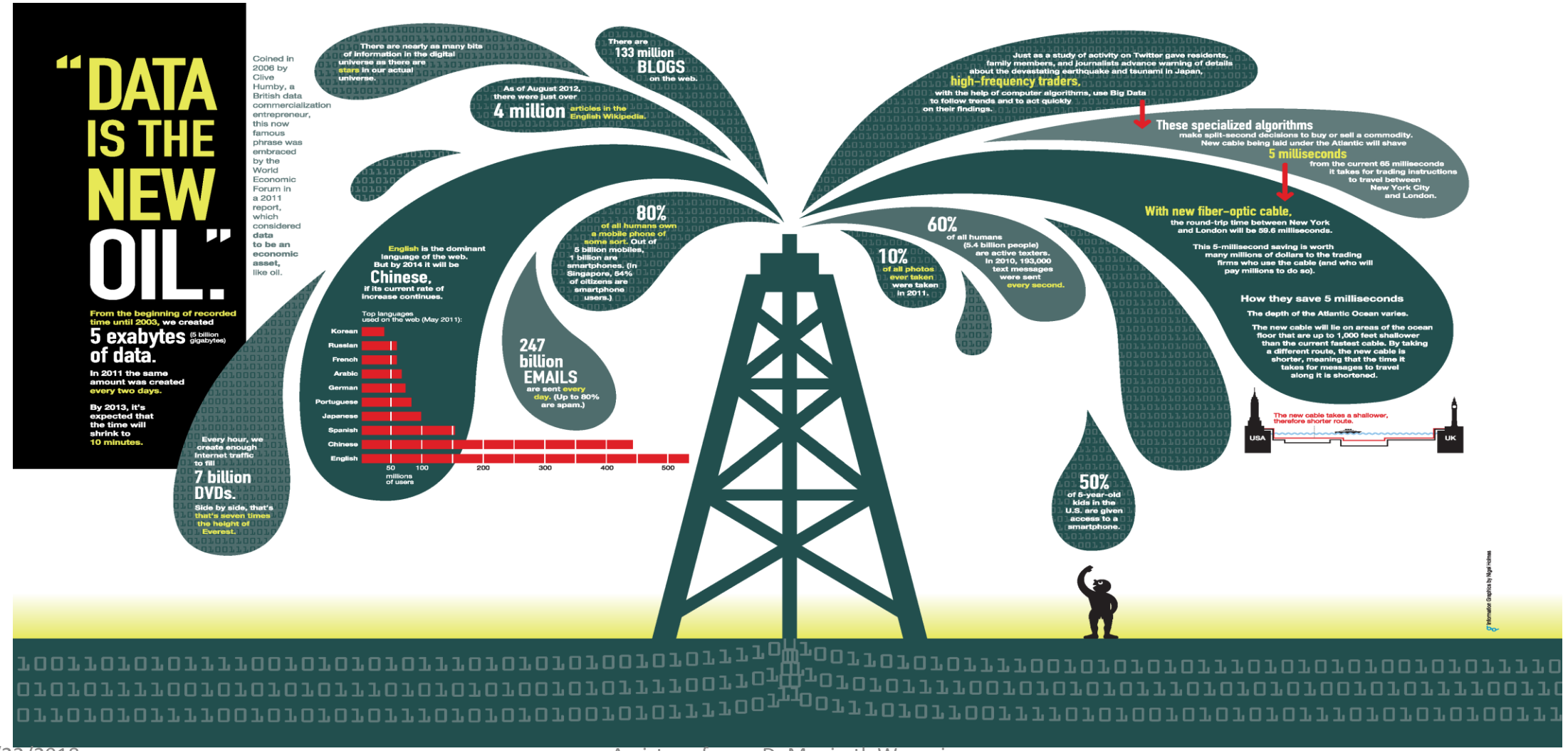

…..


Internet of Things / M2M

Health/Scientific Computing



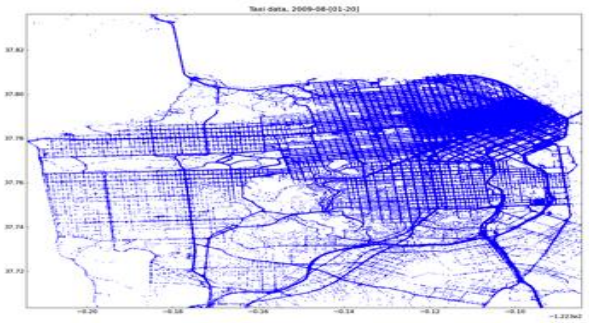
“Data is the New Oil” – World Economic Forum 2011

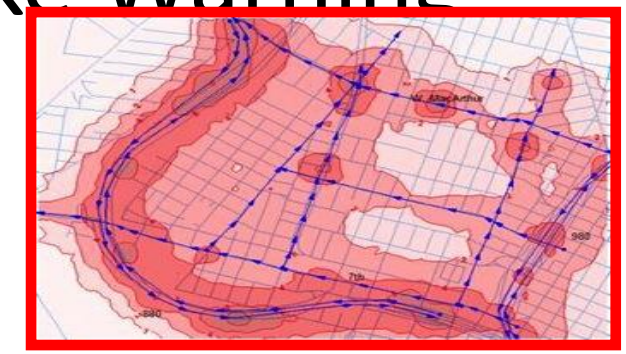


5 Vs of Big Data

- Raw Data: Volume
- Change over time: Velocity
- Data types: Variety
- Data Quality: Veracity
- Information for Decision Making: Value

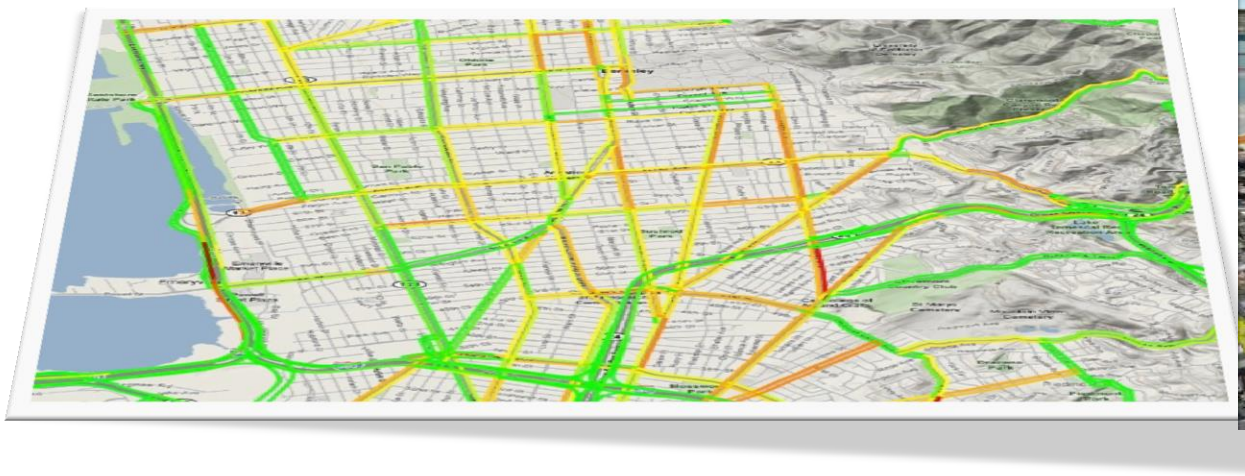
What can you do with the data?
Traffic Prediction and Earthquake Warning

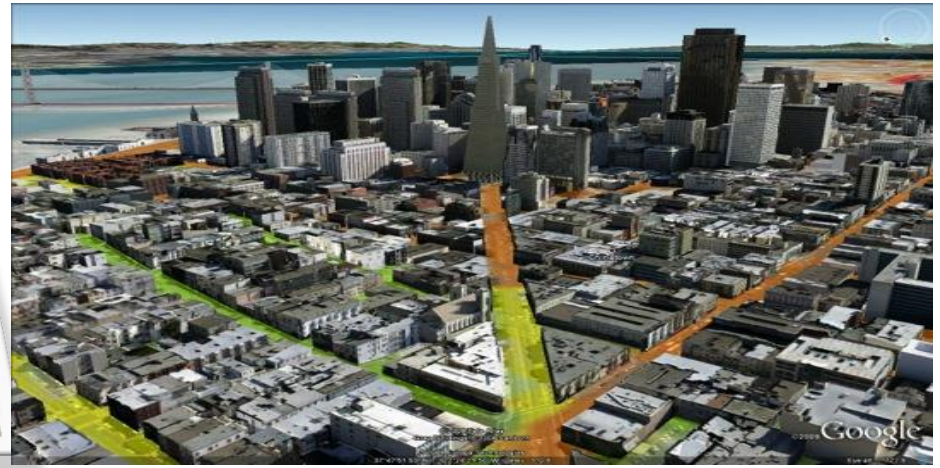




Crowdsourcing + physical modeling + sensing + data assimilation

to produce:





From Alex Bayen, UCB, Director, Institute for Transportation Studies

What is Data Science?

“Data Science” an Emerging Field

O’Reilly Radar report, 2011

Data Science – A Definition

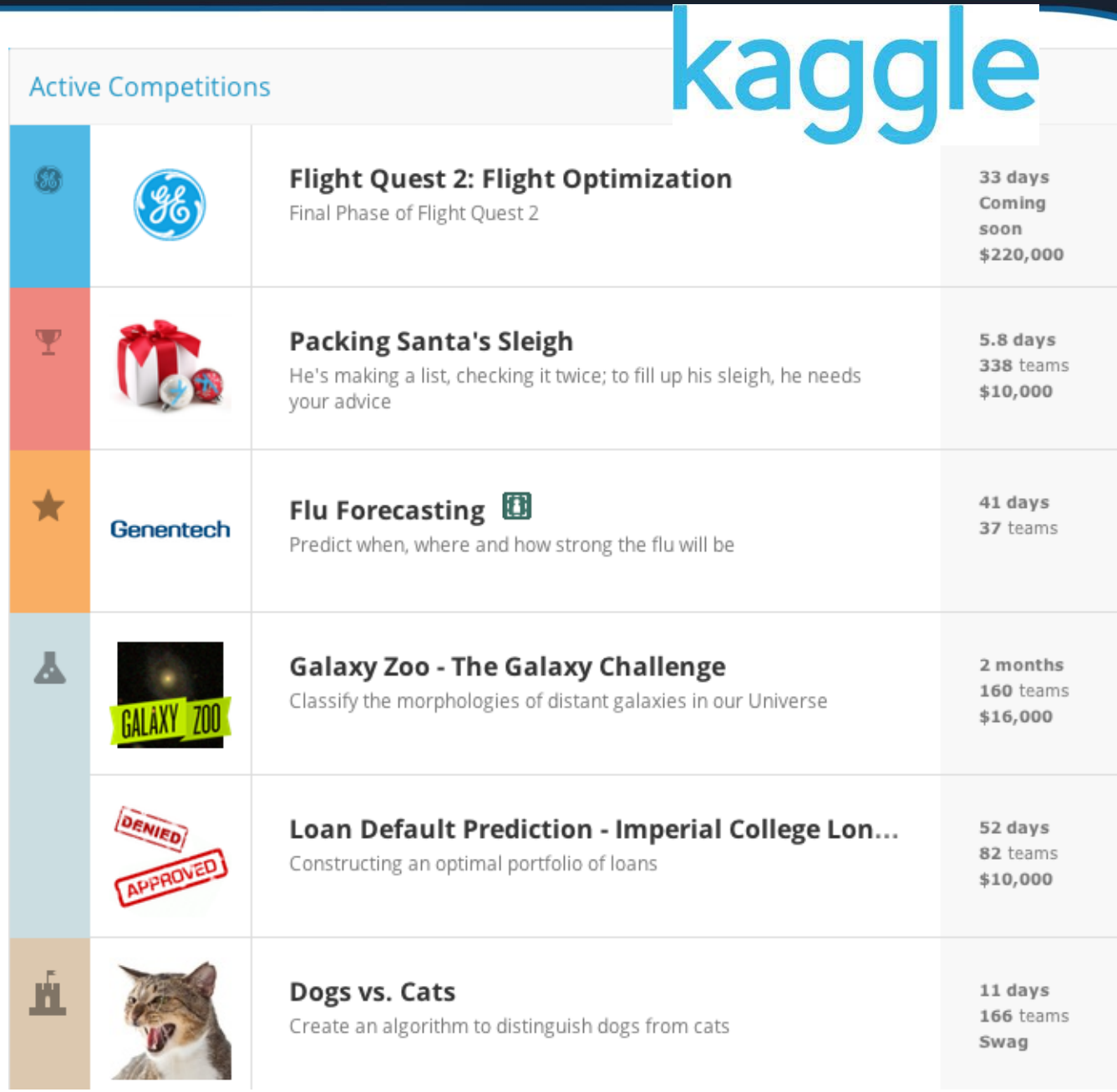
Data Science is the science which uses computer science, statistics and machine learning, visualization and human-computer interactions to collect, clean, integrate, analyze, visualize, interact with data to create data products.

Goal of Data Science

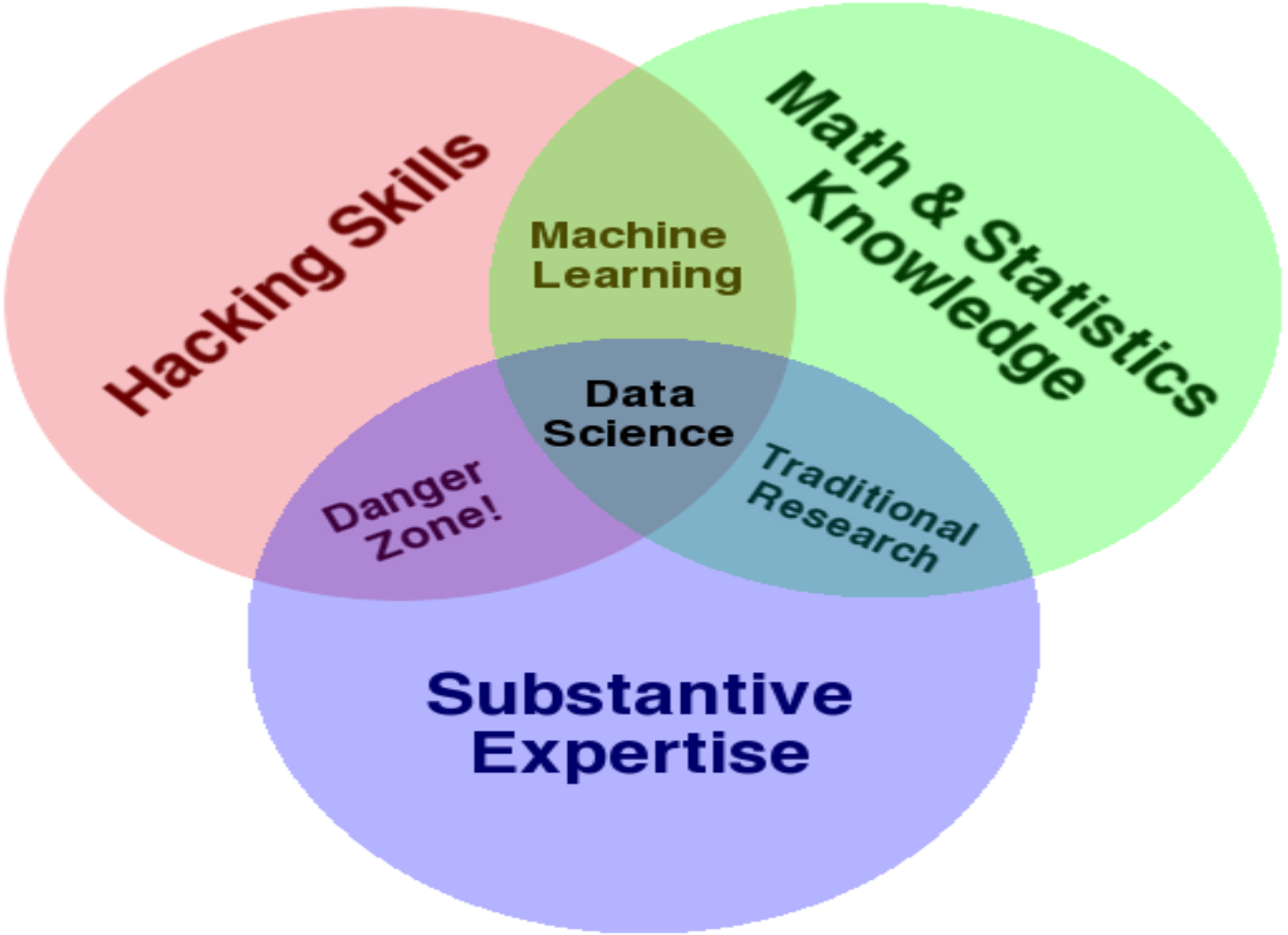
Turn data into data products.

Some recent ML Competitions at https://www.kaggle.com/

NIST Pre-Pilot Data Science Evaluation – likely to be incorporated to be part of Labs/Final project



Data Science – A Visual Definition



Contrast: Databases

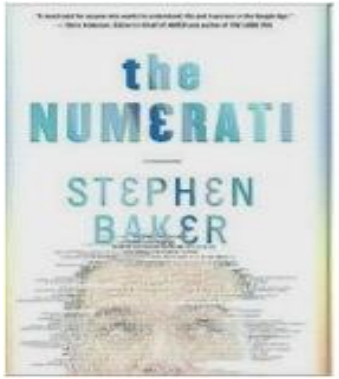
	Databases	Data Science
Data Value	“Precious”	“Cheap”
Data Volume	Modest	Massive
Examples	Bank records, Personnel records, Census, Medical records	Online clicks, GPS logs, Tweets, Building sensor readings
Priorities	Consistency, Error recovery, Auditability	Speed, Availability, Query richness
Structured	Strongly (Schema)	Weakly or none (Text)
Properties	Transactions, ACID*	CAP* theorem (2/3), eventual consistency
Realizations	SQL	NoSQL: MongoDB, CouchDB, Hbase, Cassandra, Riak, Memcached, Apache River, …

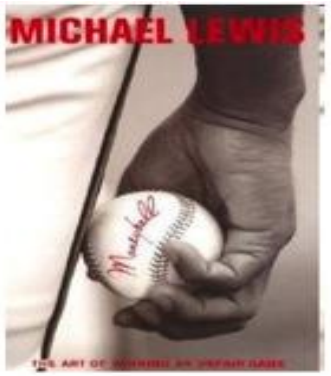
ACID = Atomicity, Consistency, Isolation and Durability

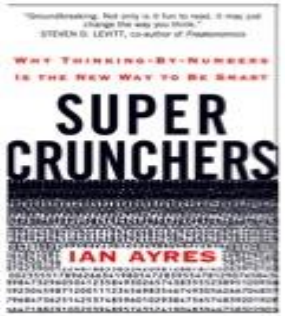
CAP = Consistency, Availability, Partition Tolerance

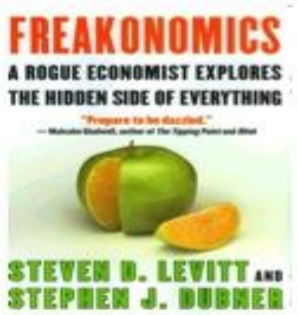
Contrast: Business Intelligence

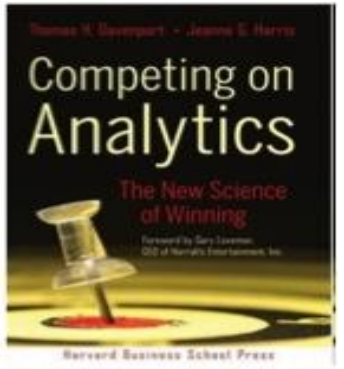
Business Intelligence	Data Science
Querying the past	Querying the past present and future











Contrast: Machine Learning

Machine Learning	Data Science
Develop new (individual) models	Explore many models, build and tune hybrids
Prove mathematical properties of models	Understand empirical properties of models
Improve/validate on a few, relatively clean, small datasets	Develop/use tools that can handle massive datasets
Publish a paper	Take action!

Introduction to Big Data :

Big Data

What is it?

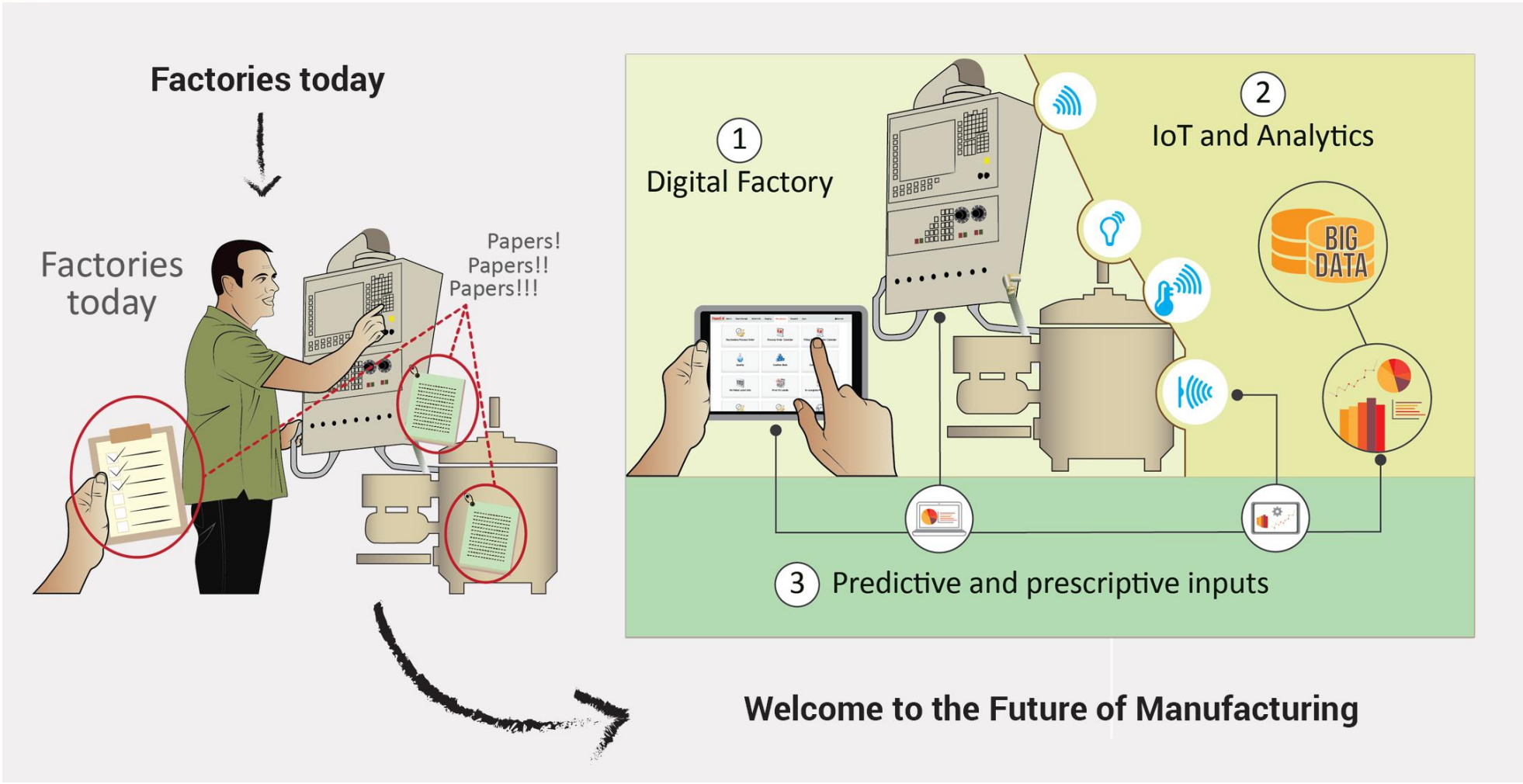
Question : BIG DATA

@ อะไรคือ BIG DATA

@ ทำไมเราต้องสนใจ BIG DATA

@ เราต้องการอะไรจาก BIG DATA

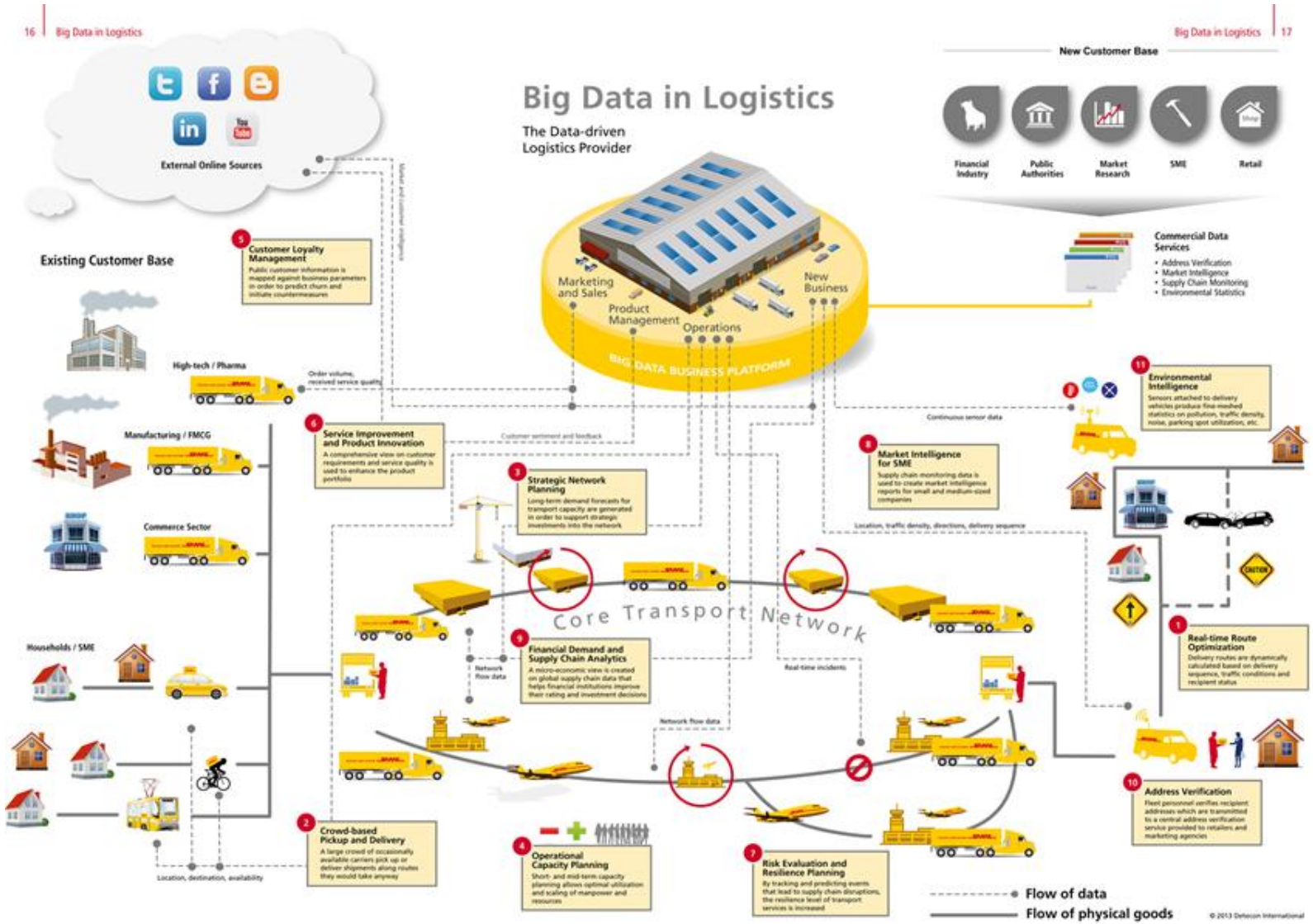
Why need BIG DATA : Industry 4.0



http://d3ugbpdwcj3u8l.cloudfront.net/wp-content/uploads/2016/07/01174310/Vegam_Digital_Dashboard_02.png

Why need BIG DATA : Modern Coffee Shop

Why need BIG DATA : Logistic



https://i0.wp.com/cerasis.com/wp-content/uploads/2016/11/2016-11-14.png

Why need BIG DATA : Traffic

Why need BIG DATA : wholesale

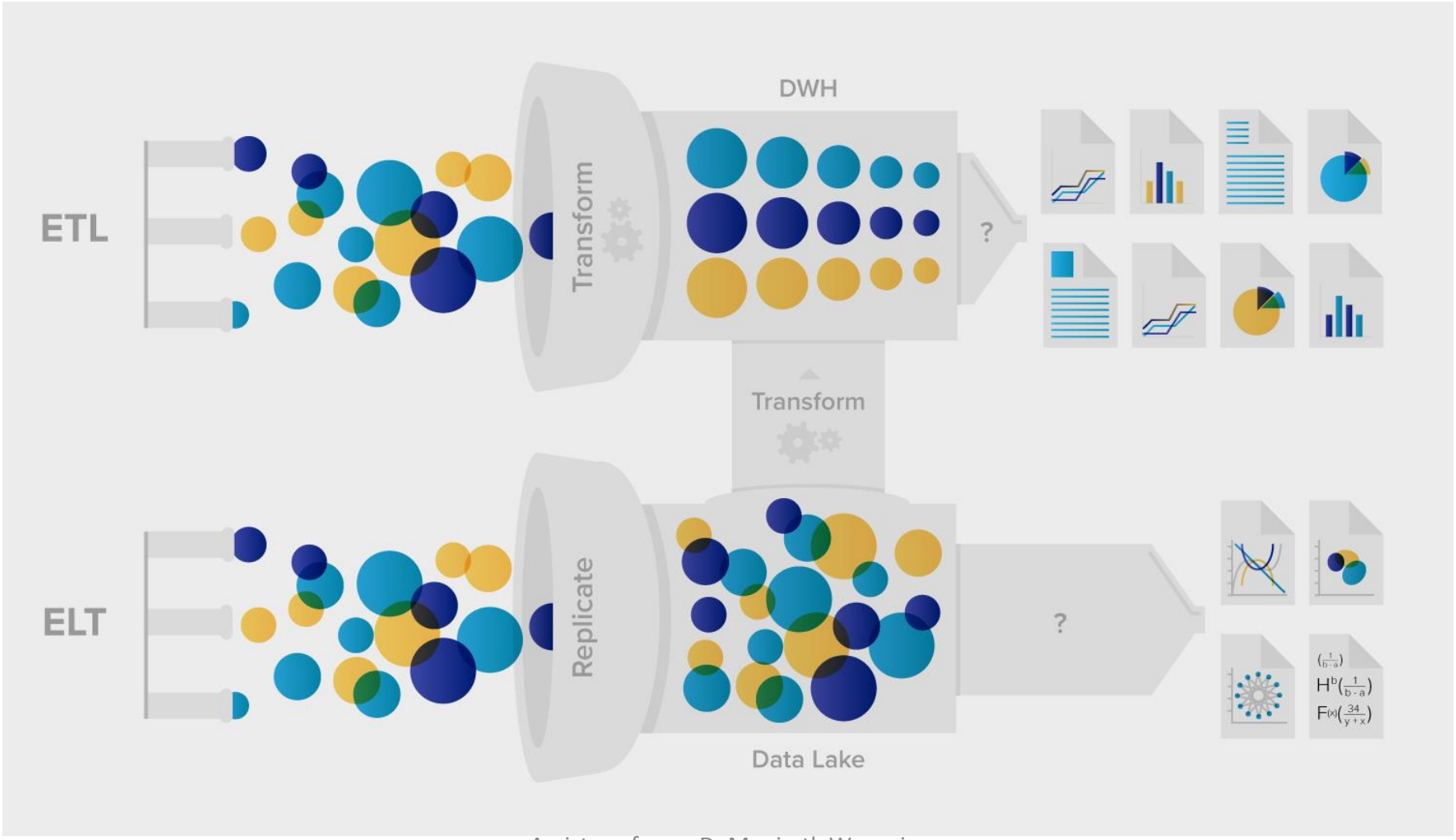
Why need BIG DATA : hospital

https://www.meed.com/pictures/1240x826/7/5/0/2003750_big-data-healthcaredreamstime_m_49010041.jpg

Why need BIG DATA : Banking

https://cdn.computerworlduk.com/cmsdata/slideshow/3657529/Bribe_money_cash_bribe_thumb800.jpg

Why need BIG DATA : Data Warehouse



https://cdn.computerworlduk.com/cmsdata/slideshow/3657529/Bribe_money_cash_bribe_thumb800.jpg

BIG DATA มาจากไหน

Activity Data

Conversation Data

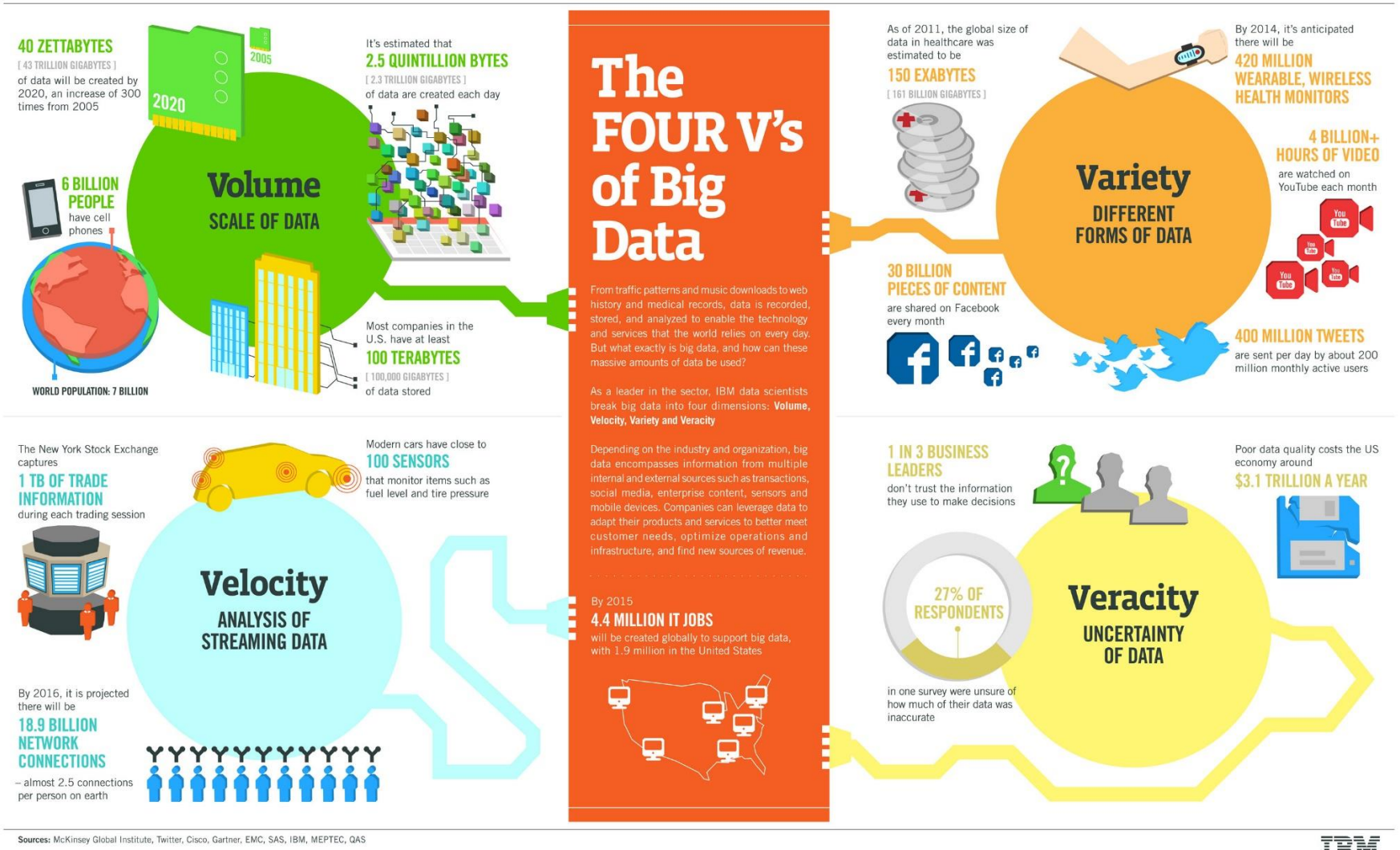
Photo and Video Image Data

Sensor Data

The Internet of Things Data

http://www.seekbig.in/wp-content/uploads/2016/02/tnpsc-economics-questions.png

BIG DATA Definition :

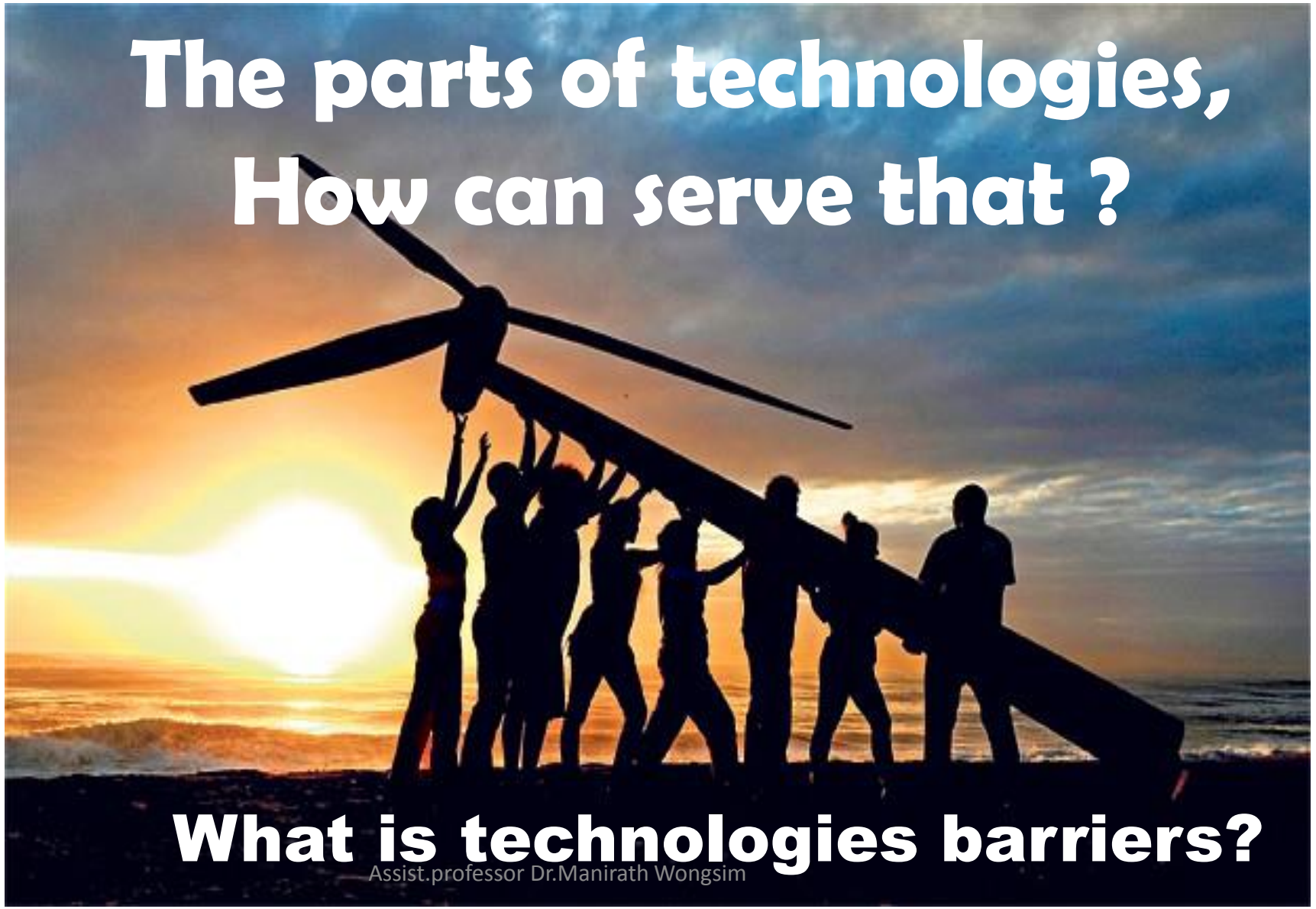


https://www-01.ibm.com/software/data/bigdata/

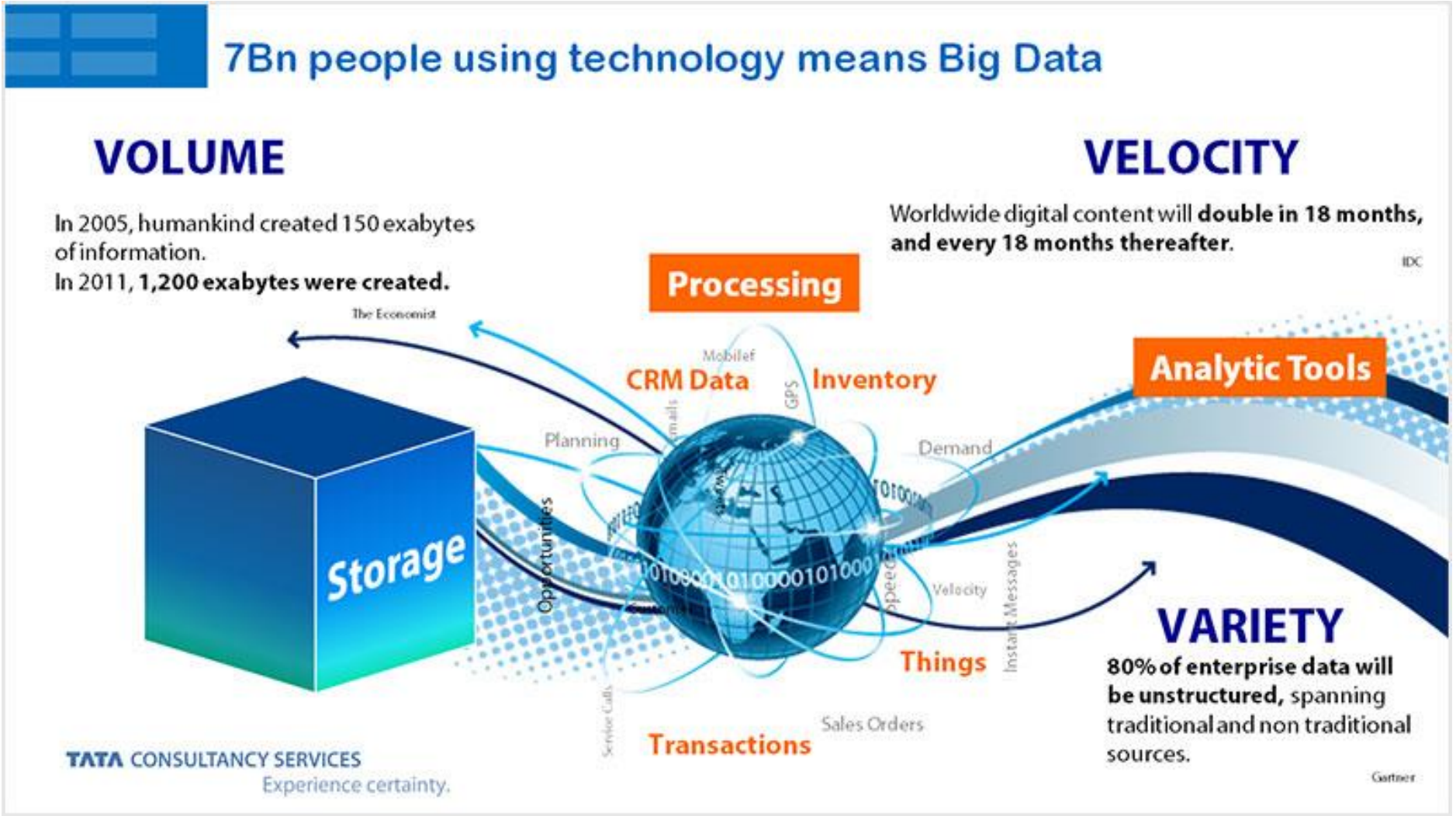
The Sizing of BIG DATA

http://images.rigzone.com/images/home/article/infog_140631.jpg

BIG DATA สร้างปัญหาต่อเนื่องอะไรให้บ้าง



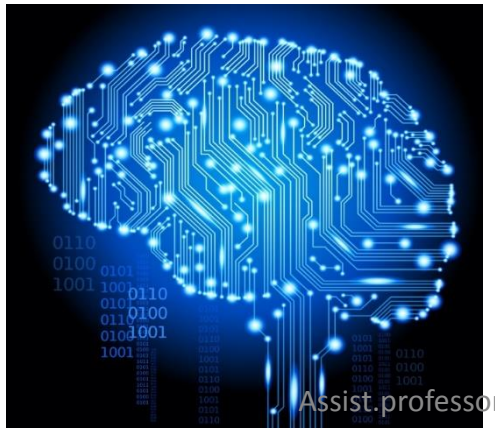
The challenge of BIG DATA :

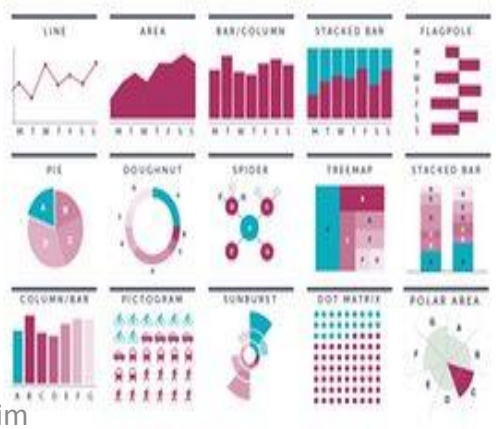


https://secure.business.nova.edu/marketing-blog/enclosures/Big%20Data%20Brotspies%20TATA%20Consulting%20Services.jpg

The things cover to BIG DATA Technology :

Big Data Technology



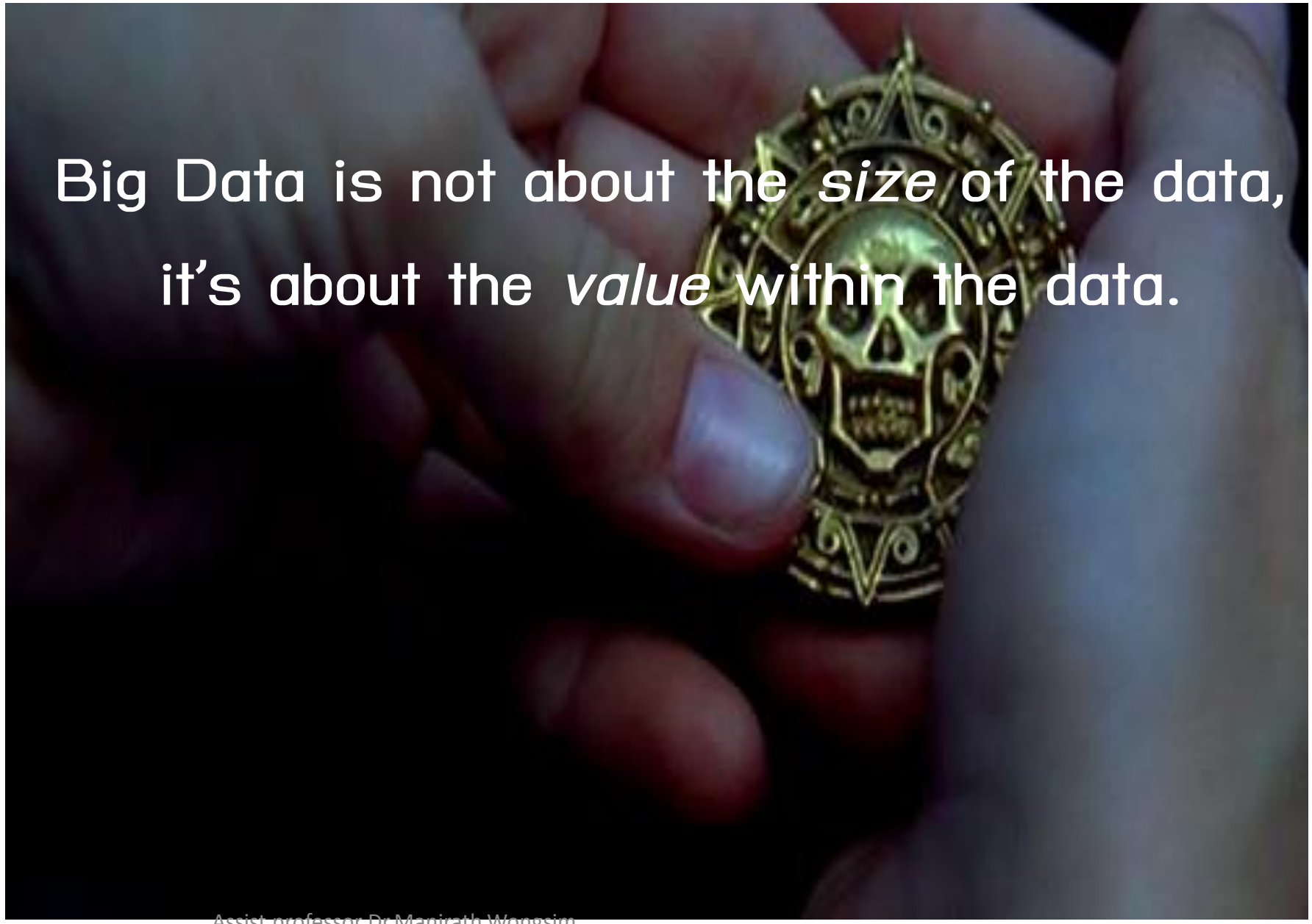


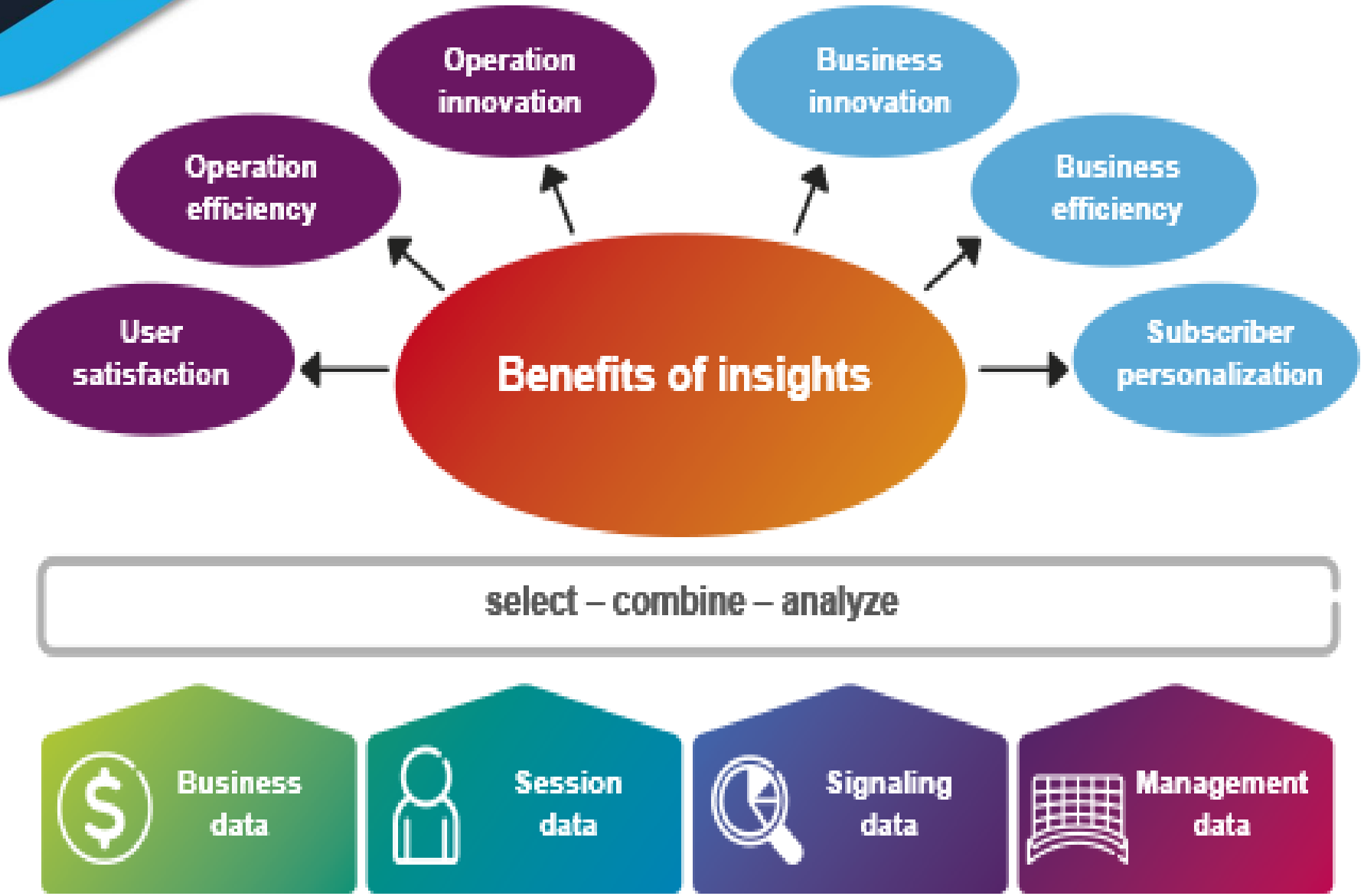
BIG DATA , What is it ?

Big Data = Big + Data - Big

So, what is

value?

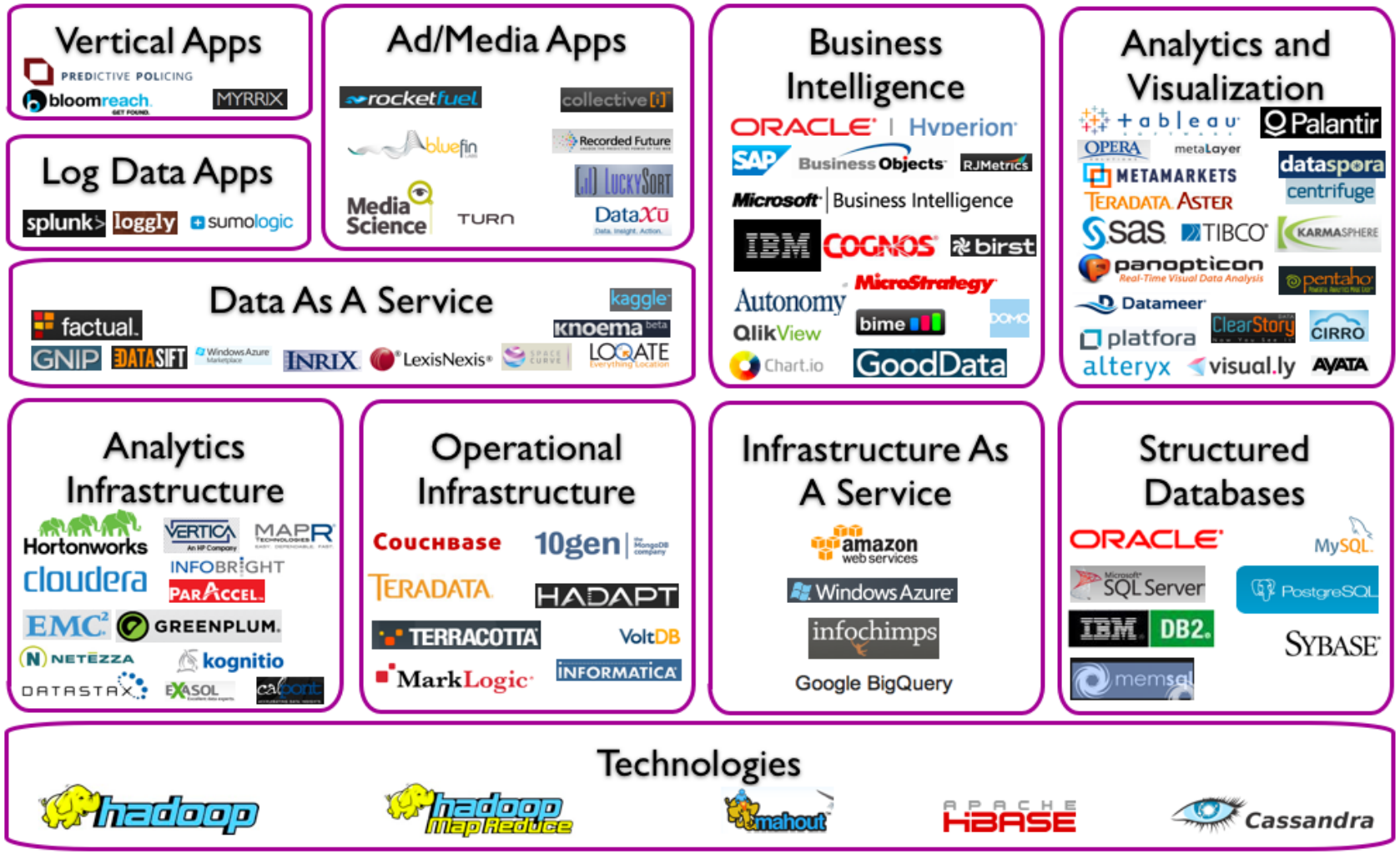




https://s-media-cache-ak0.pinimg.com/originals/02/e8/bd/02e8bd3d8adb71dcbdd7549da90fcbde.jpg


Market Intelligence?
Business Intelligence?
Operation Intelligence?
Social Intelligence?
Life's Intelligence?

Big Data Landscape



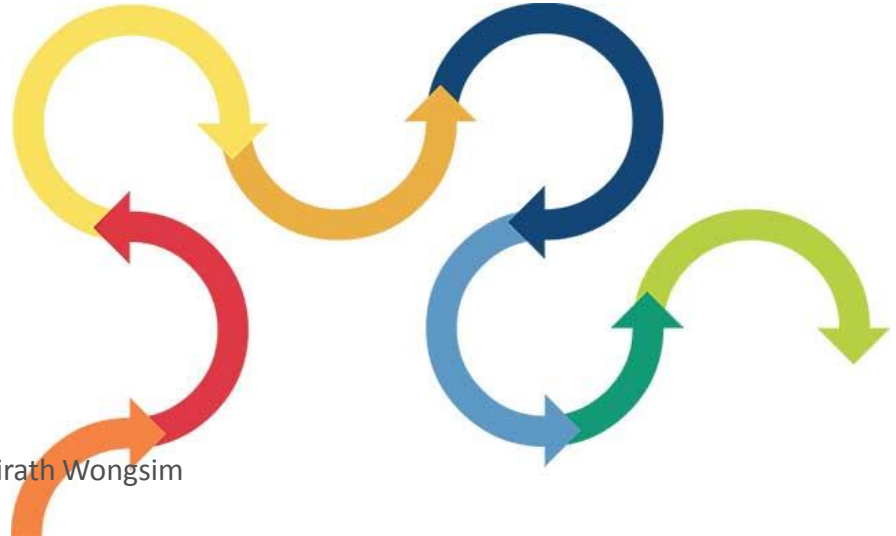
dave@vcdave.com

blogs.forbes.com/davefeinleib

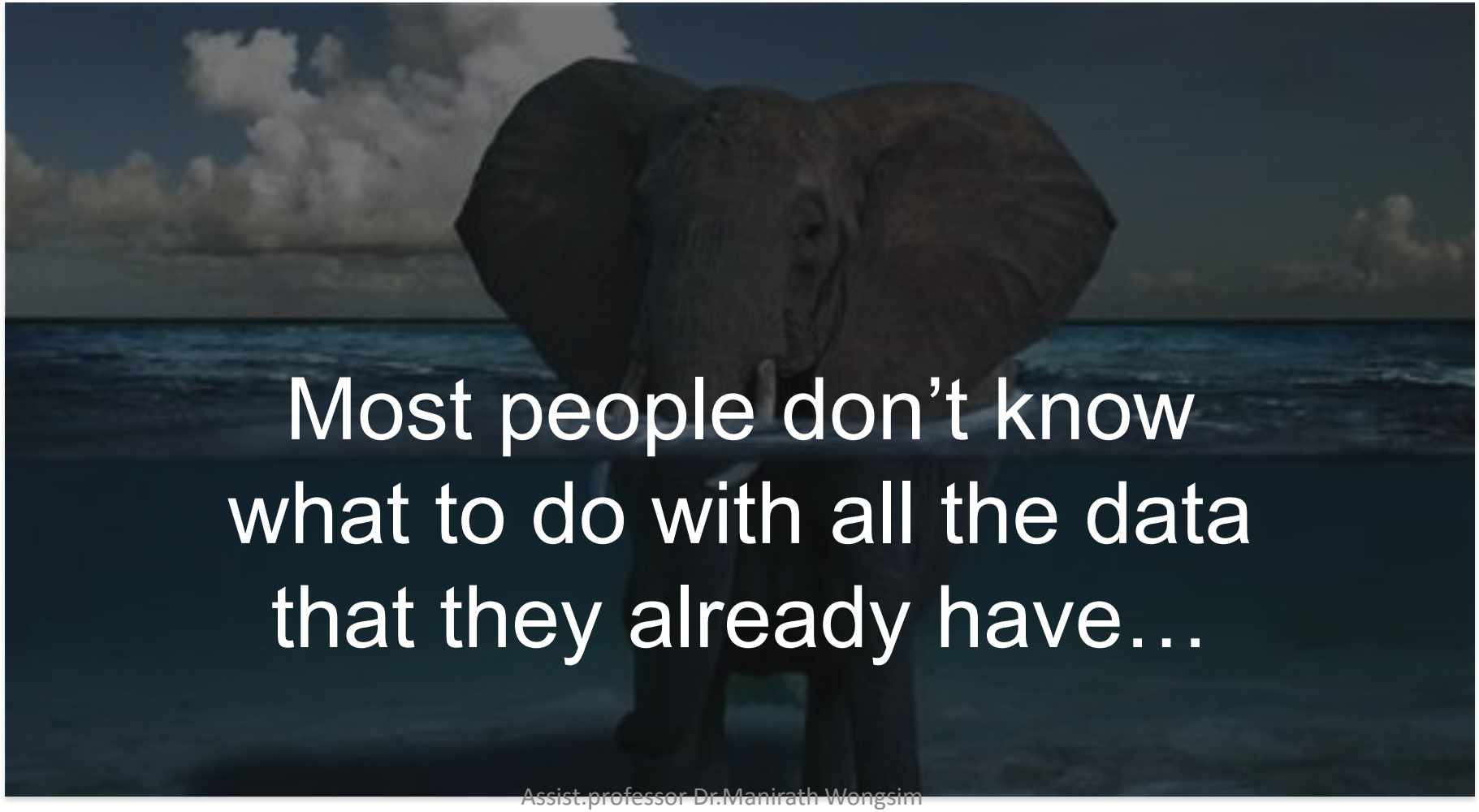
https://blogs-images.forbes.com/davefeinleib/files/2014/06/big-data-landscape-jul-4-2012-00111.png

How can we take Advantage from

Big Data Technology




Big Data isn't *big*,
if you *know* how to
use it.


Most people don’t know
what to do with all the data
that they already have…


Get Big
by starting
small

Focus on

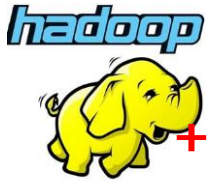

Business Impact.
START
TRY SOMETHING
DID IT WORK?
NO
YES
YES
STOP

Introduction to Big Data

Q & A

Hadoop Classification :

Open Source



+ Ecosystem

Distribution

cloudera MAPR TECHNOLOGIES

Hortonworks

TERADATA

ORACLE BIG DATA

IBM

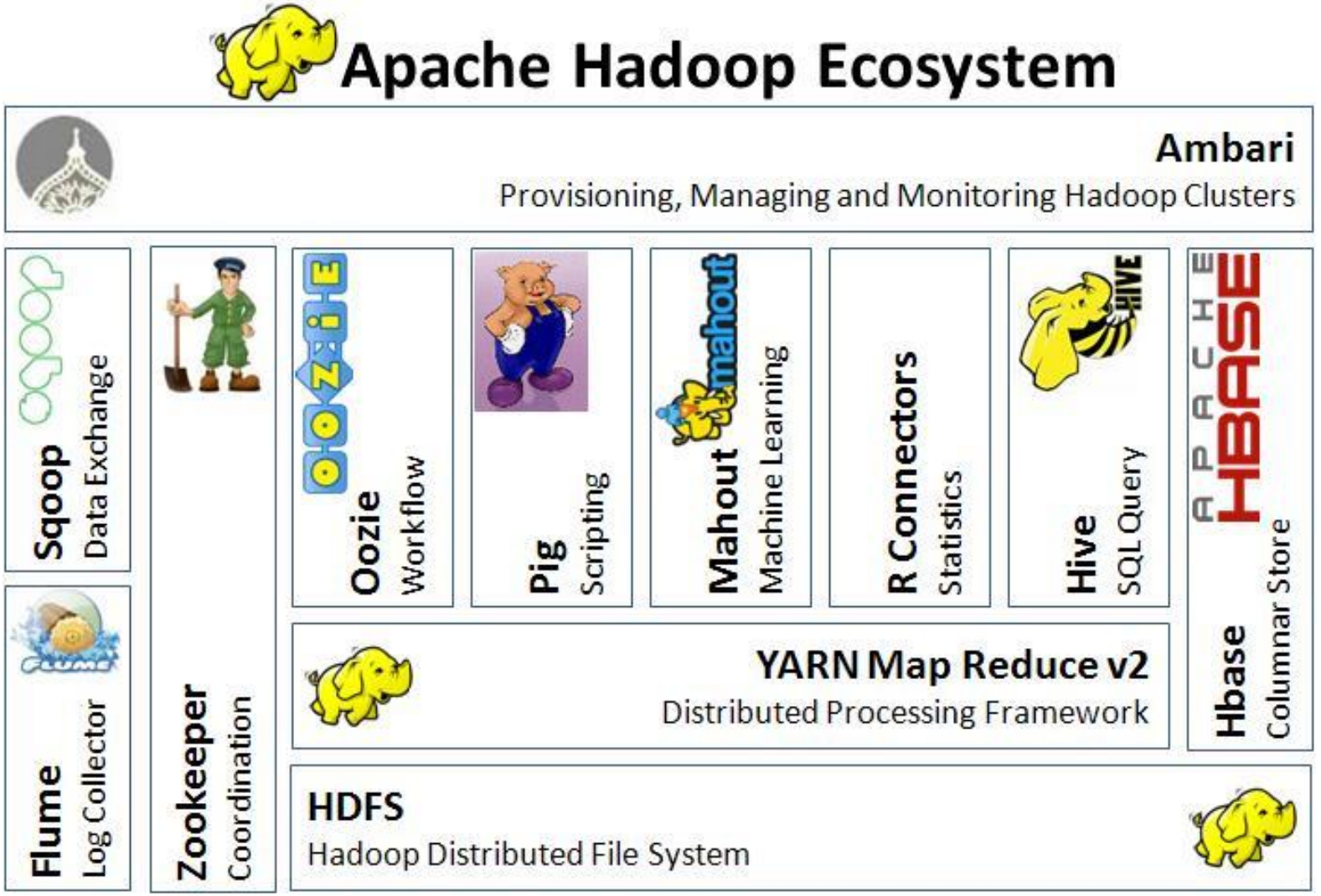
Appliance

Cloud

Hadoop Component :

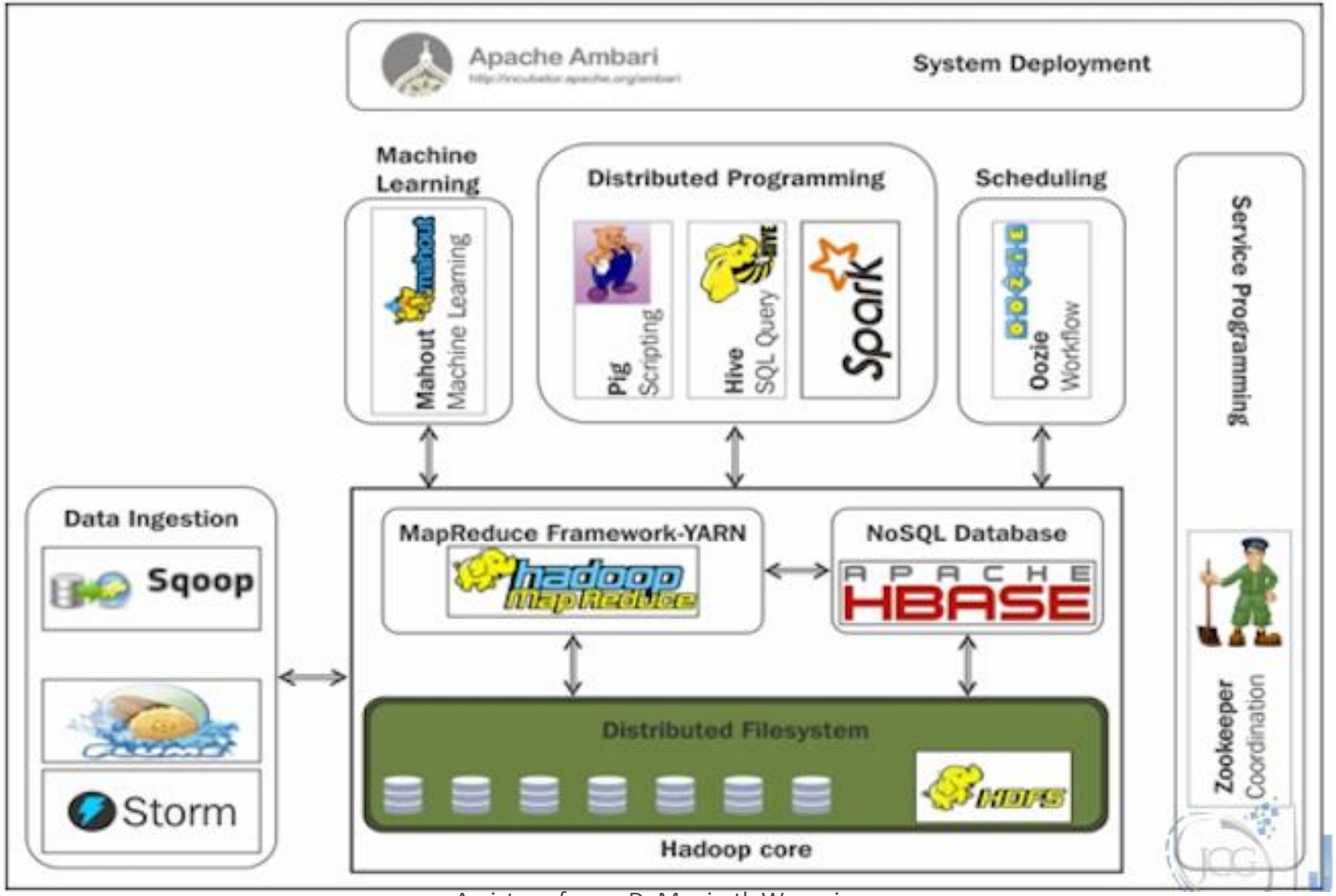
- **Hadoop Distributed File System (HDFS™)**: A distributed file system that provides high-throughput access to application data.
- **Hadoop YARN**: A framework for job scheduling and cluster resource management.
- **Hadoop MapReduce**: A YARN-based system for parallel processing of large data sets.

Hadoop Ecosystem :



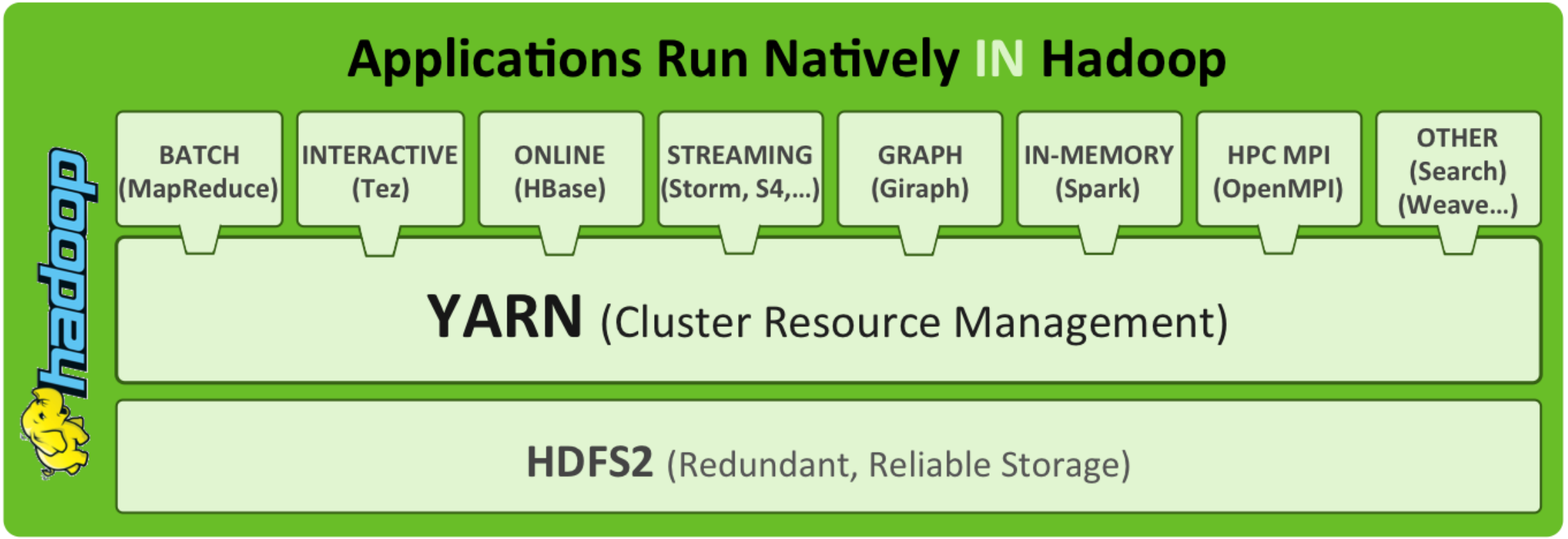
http://thebigdatablog.weebly.com/uploads/3/2/3/2/32326475/87375_orig.jpg

Hadoop Ecosystem :



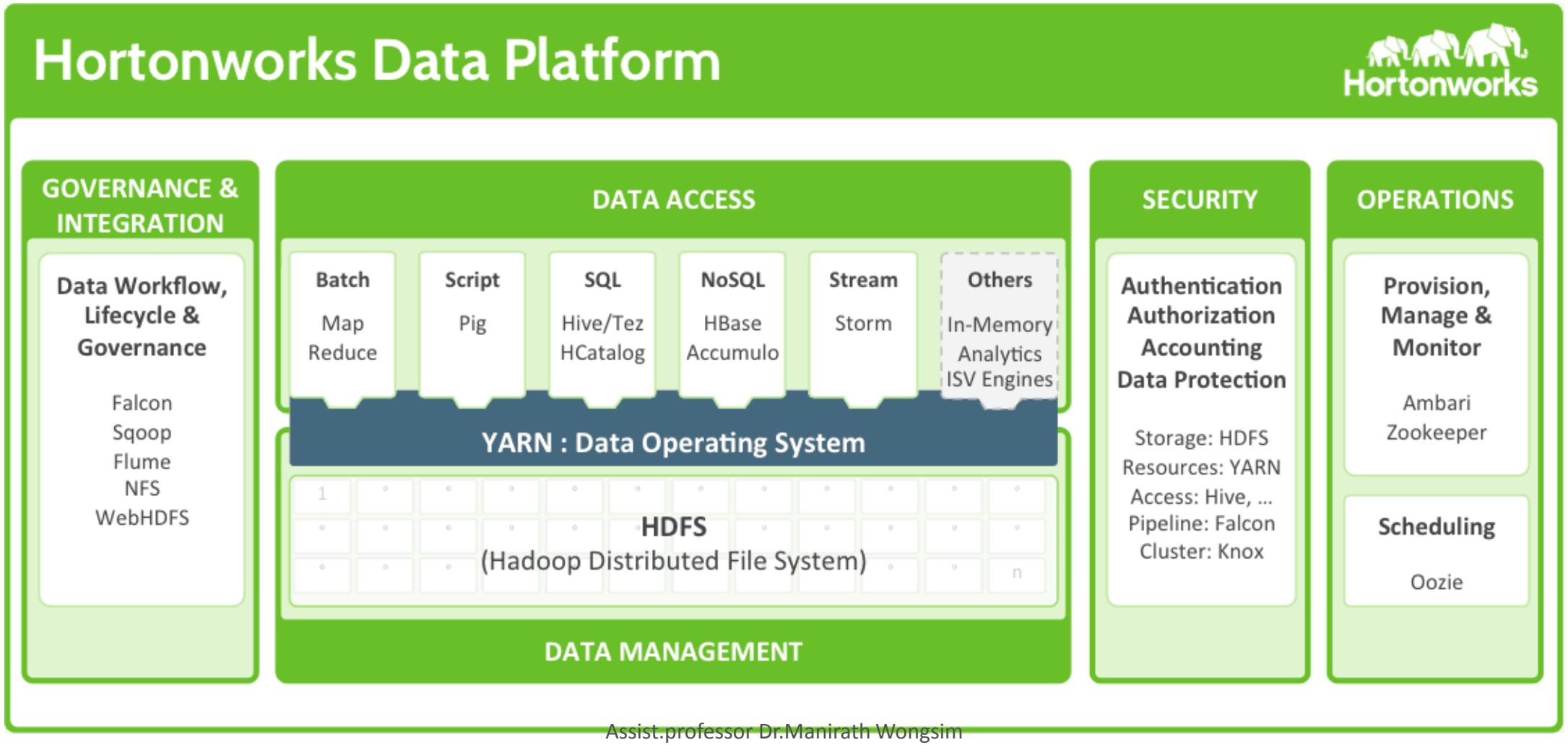
https://examples.javacodegeeks.com/wp-content/uploads/2016/08/Slide6.png

Multi-Purpose Hadoop Component Stack :

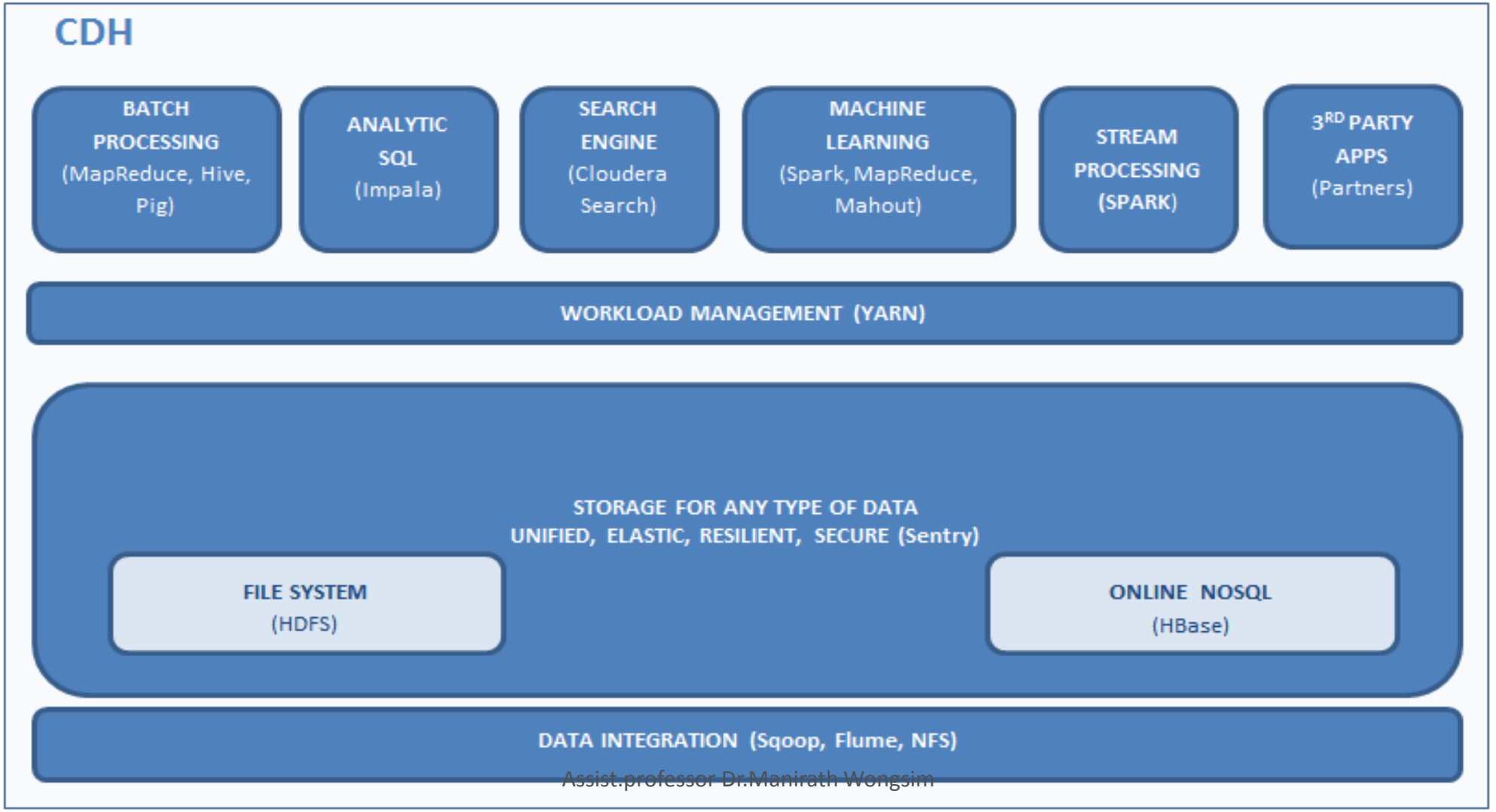


http://tm.durusau.net/wp-content/uploads/2013/06/YARN2.png

Enterprise Hadoop Component Stack :



Enterprise Hadoop Component Stack:



Hadoop Component & Terminology :

Master/Slave Architecture:

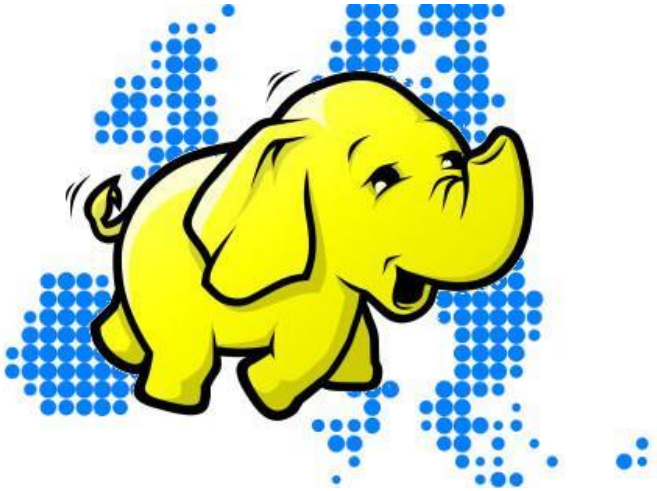
- Master / Secondary NameNode
- DataNodes

Data Management:

- HDFS
- Data Bock
- Replication Management
- Rack Awareness
- HDFS Read/Write – Behind the scenes

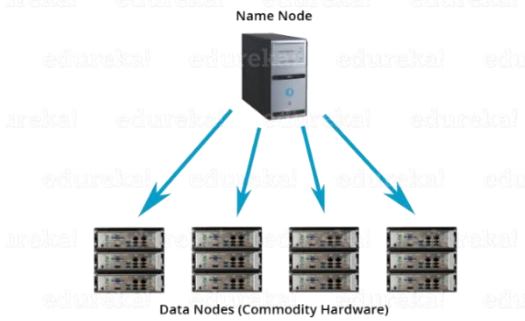
Process & Resource Management :

- YARN

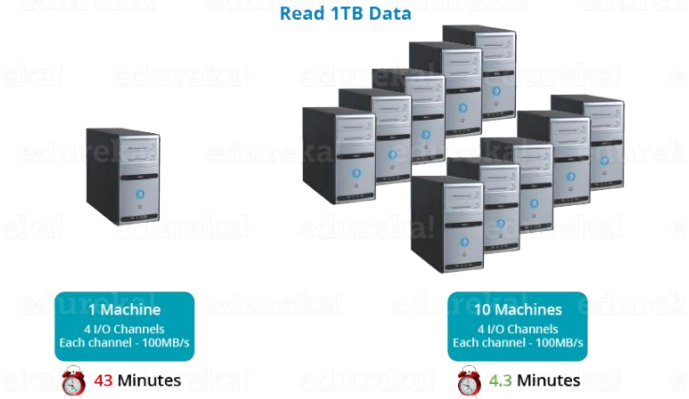


Hadoop Component & Terminology :

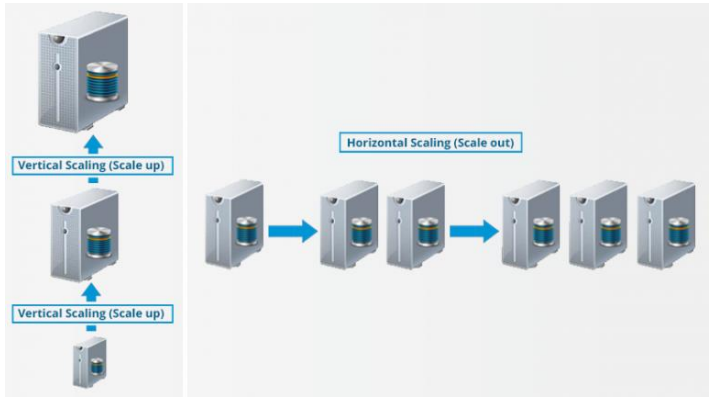
Hadoop is software framework with optimized for distributed processing of very large datasets. The main features are the Hadoop Distributed File System(HDFS) and MapReduce.



1. Distributed Storage:



2. Distributed & Parallel Computation:



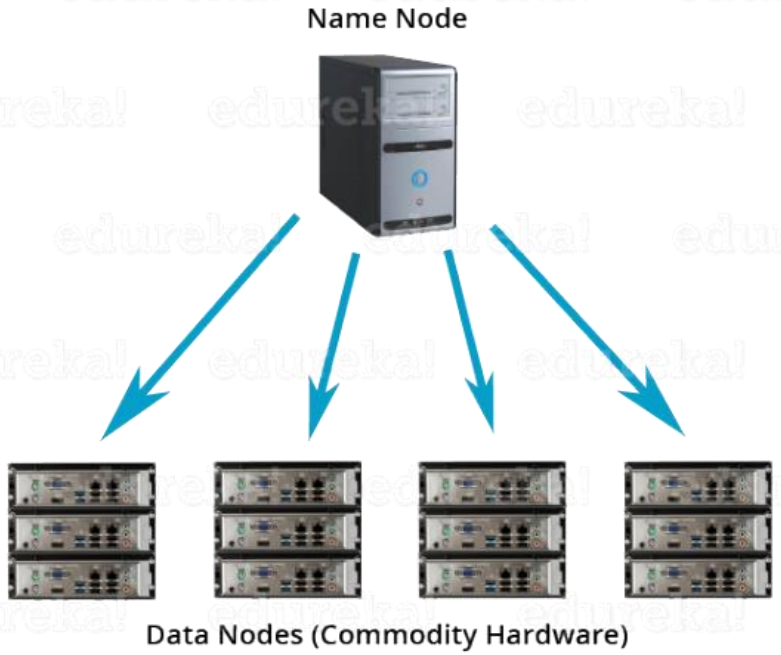
3. Horizontal Scalability:

Hadoop Component & Cluster :

Master/Slave Architecture:

Master / Secondary NameNode

DataNodes



NameNode (Master Node)

- Manage data block on DataNodes with Metadata , block location , file size , permission , hierarchy ,etc.
- Maintain DataNodes on cluster , ensure DNs are live

DataNode (Slave Node)

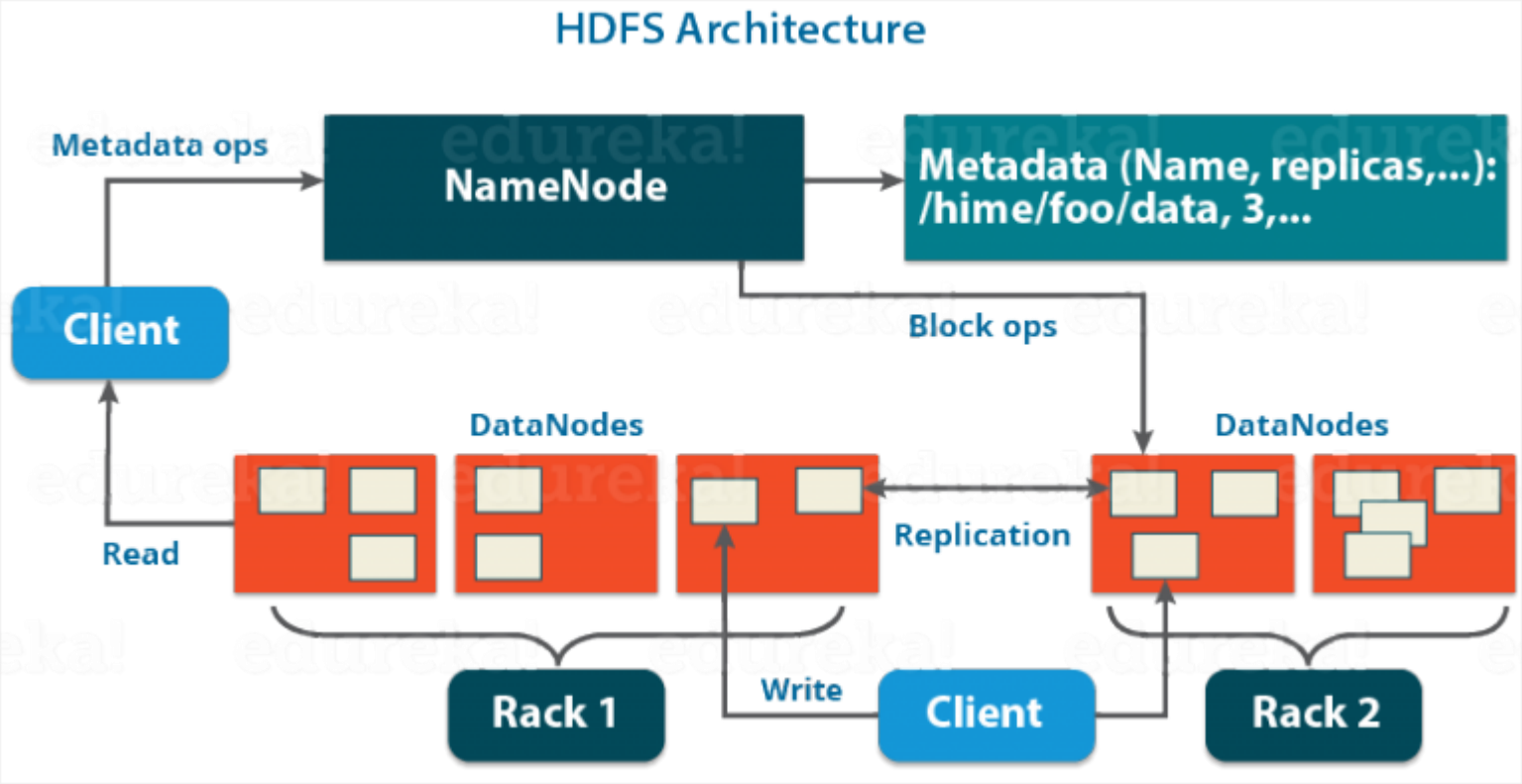
- Actual stored data
- Servicer read and write to Clients.
- Sent Heartbeats to NameNode

HDFS Architecture :

Data Management:

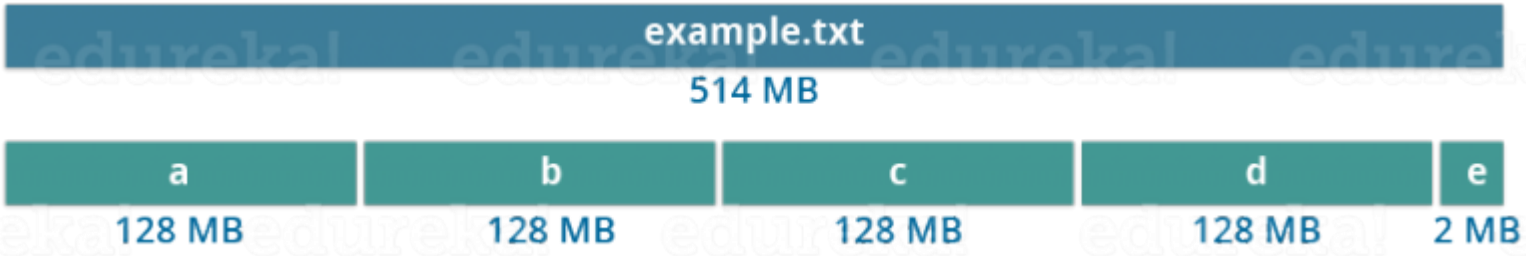
HDFS | Data Bock

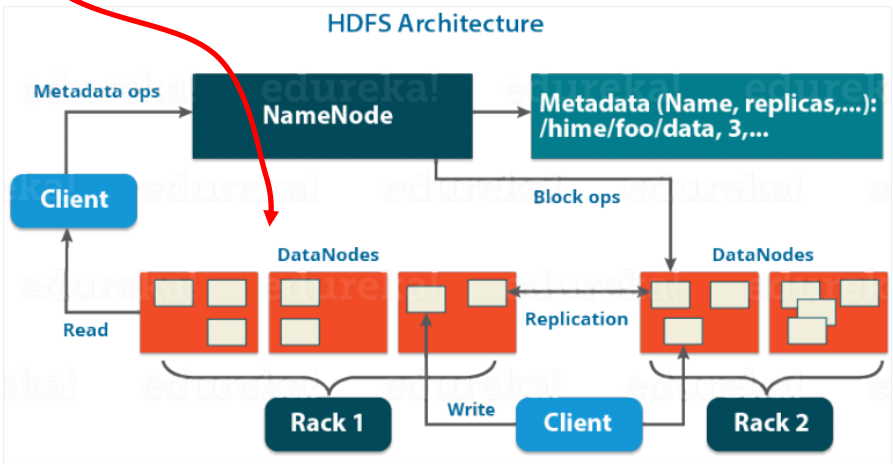
Replication | Rack Awareness



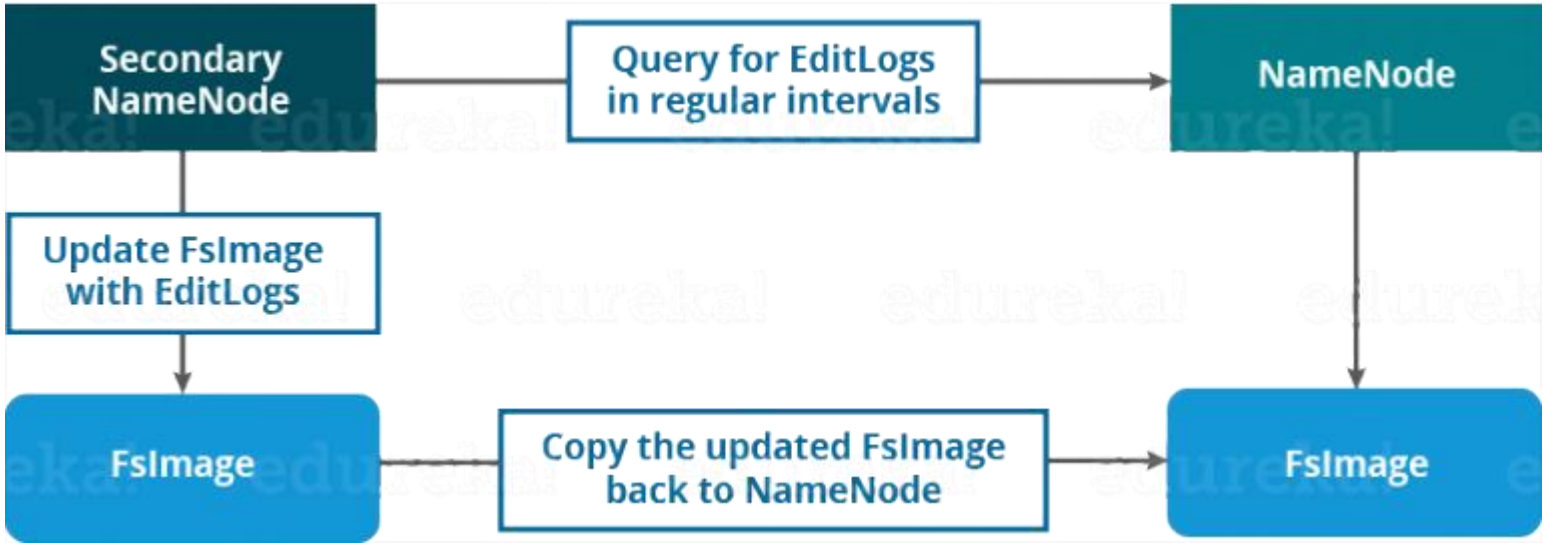
Data Block :

64MB , 128MB / Block





Secondary NaeNode :

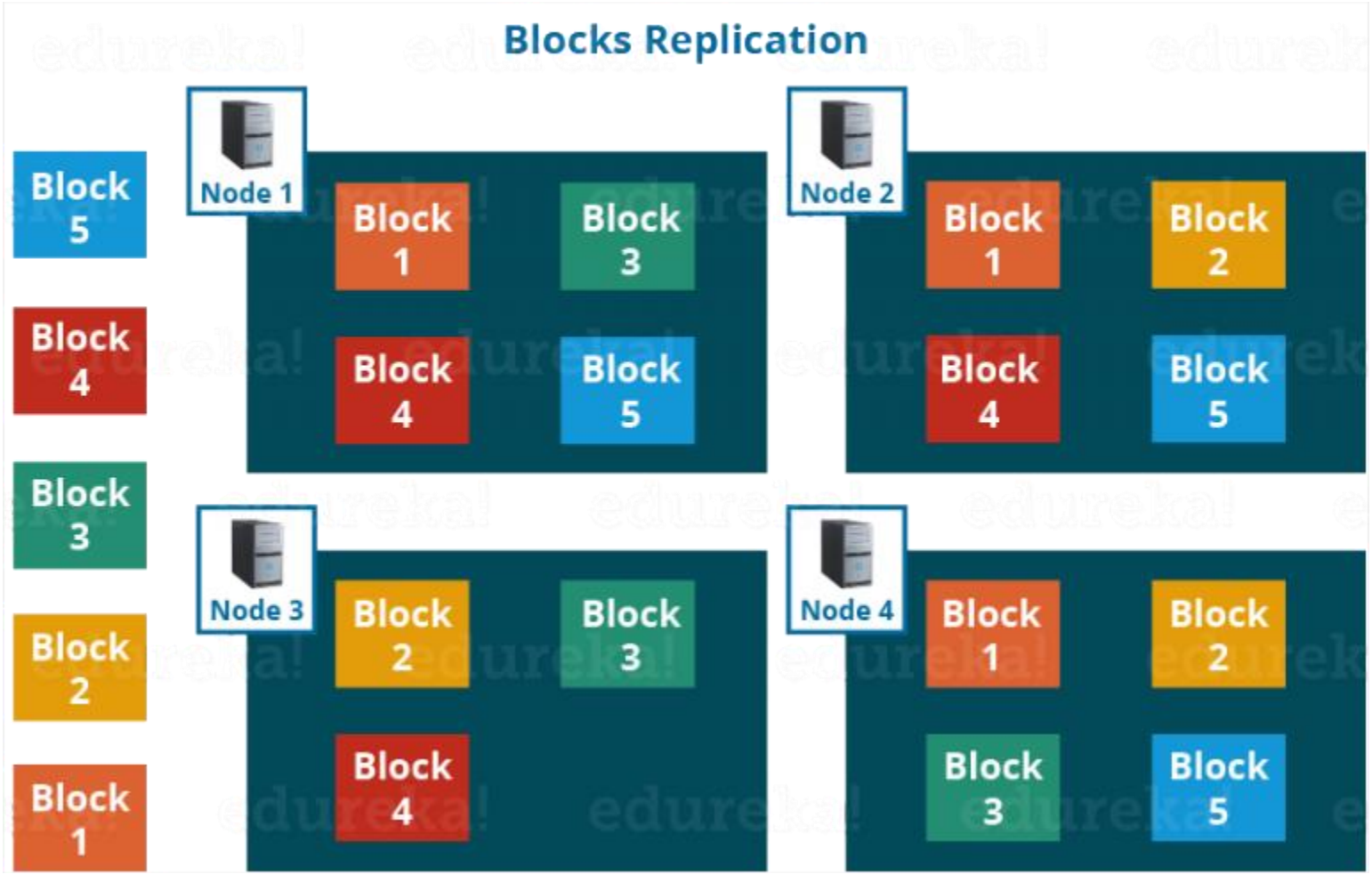


don't be confused about the Secondary NameNode being a **backup NameNode because it is not.**

Automatic backup Metadata of Master NameNode , Metadata , FsImage , EditLogs

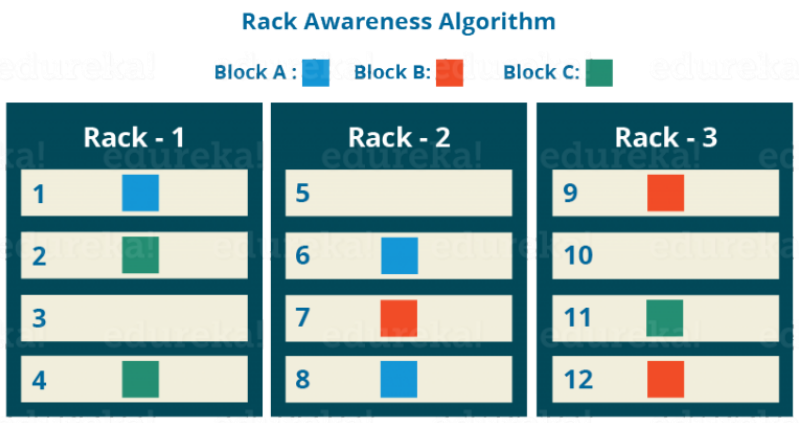
Secondary NameNode act as Checkpoint Node

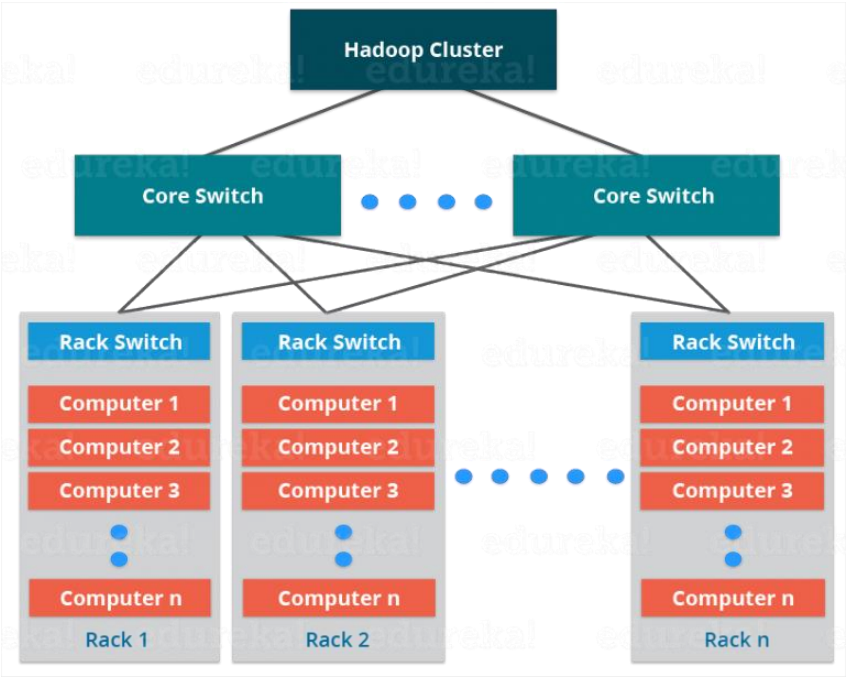
Replication Management :



Hadoop Topology :

Rack Awareness :



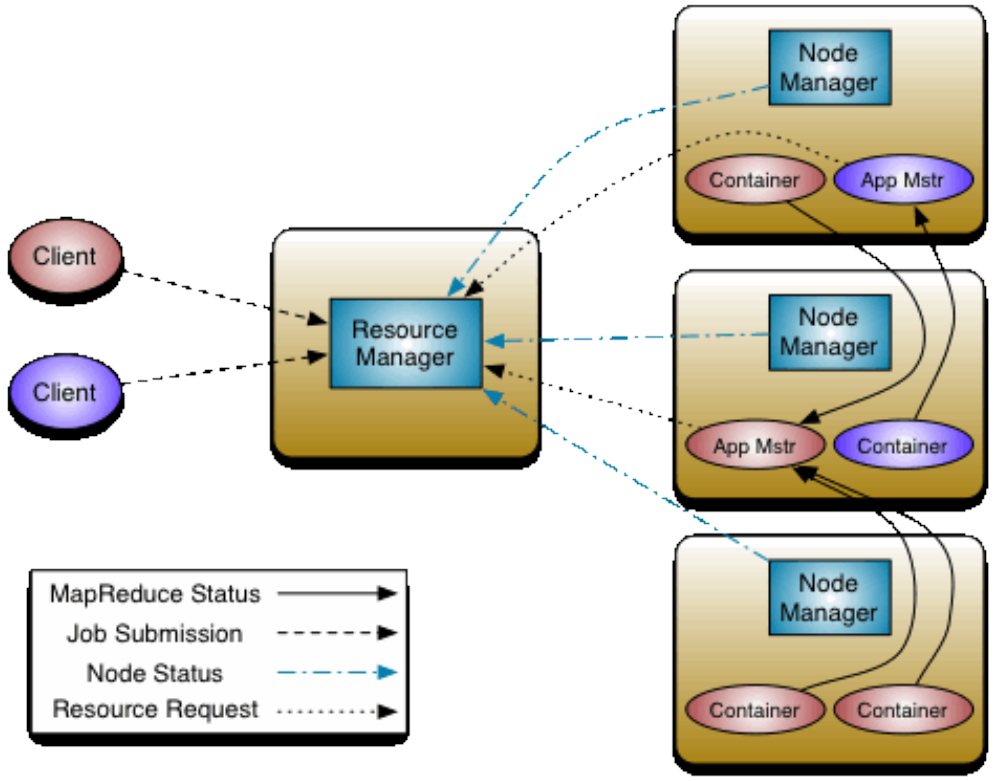


Balance the network traffic for reduce latency

YARN Architecture :

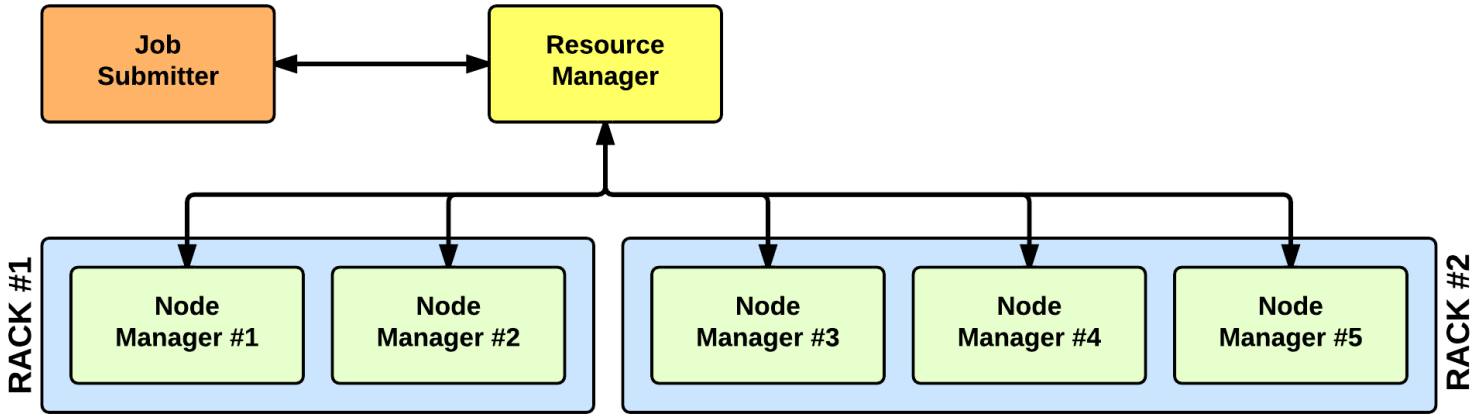
Process & Resource Management :

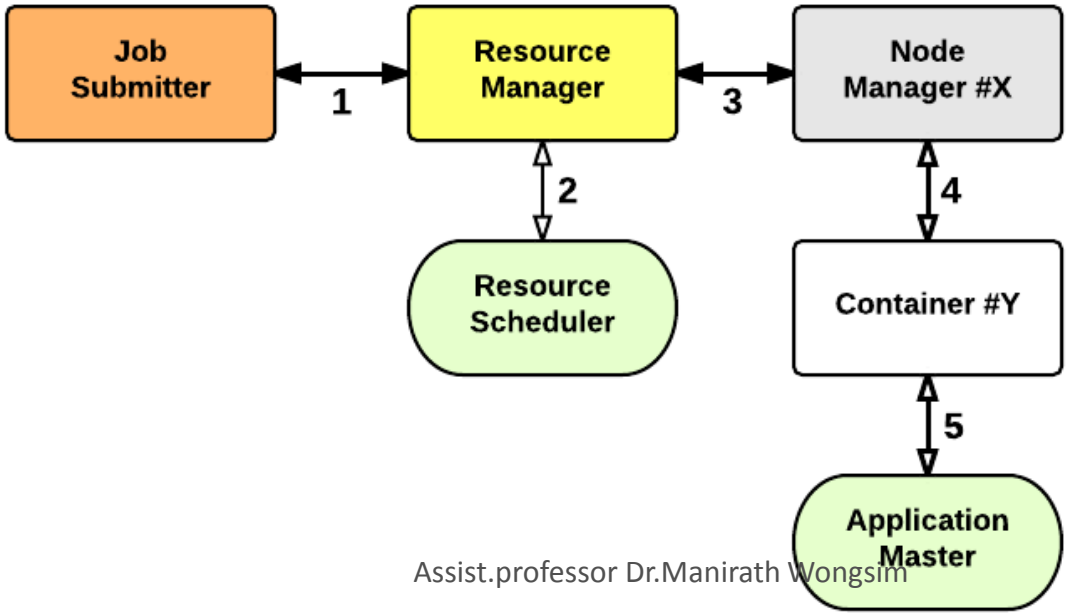
YARN



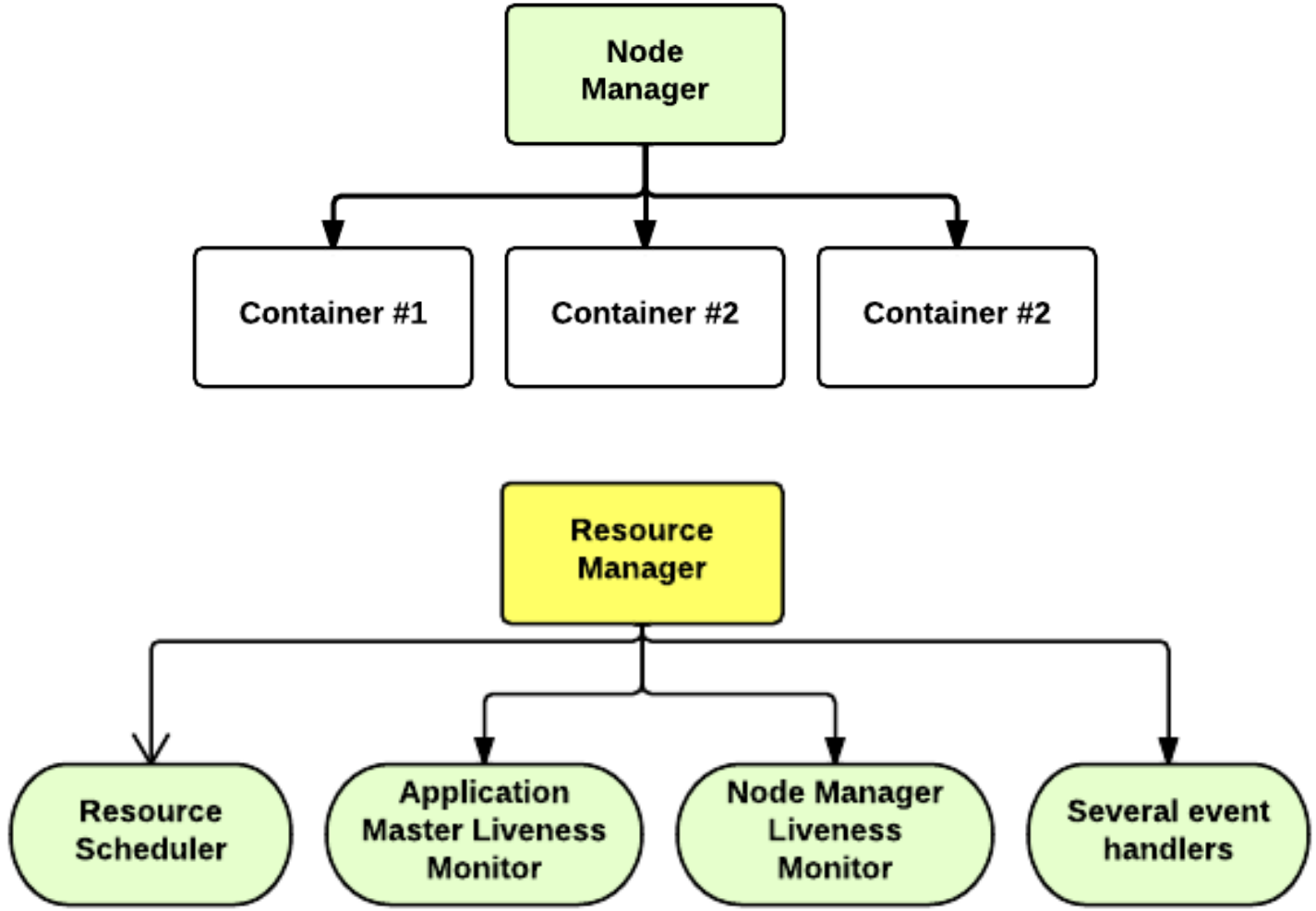
https://hadoop.apache.org/docs/r2.7.2/hadoop-yarn/hadoop-yarn-site/YARN.html

YARN Architecture :



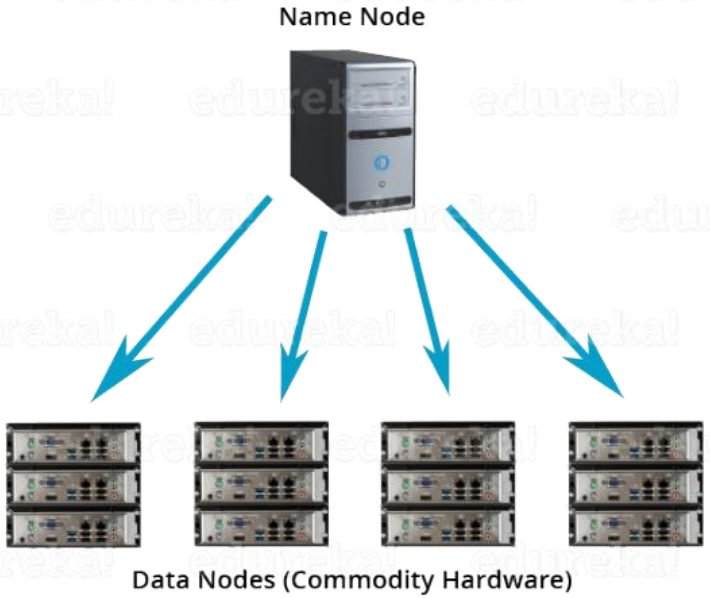


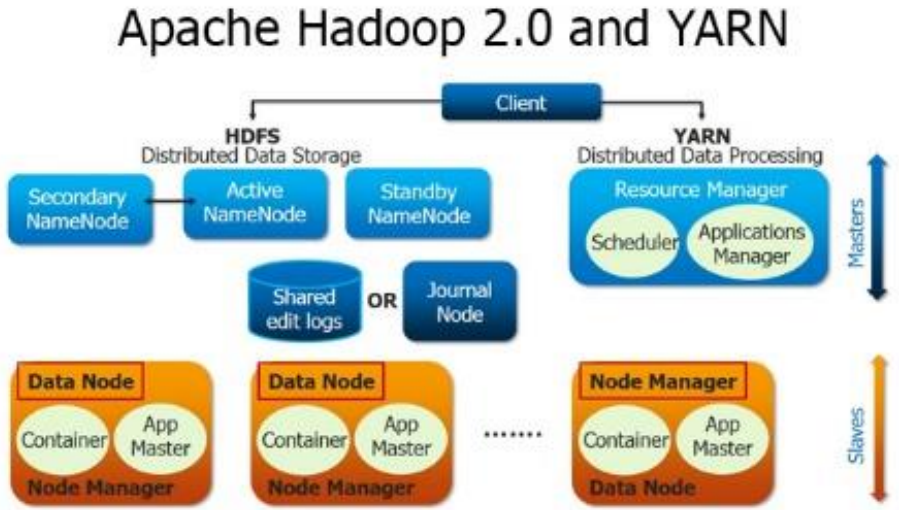
YARN Architecture :



https://hadoop.apache.org/docs/r2.7.2/hadoop-yarn/hadoop-yarn-site/YARN.html

HDFS & YARN Daemon Process :





MasterNode:

NameNode , ResourceManager

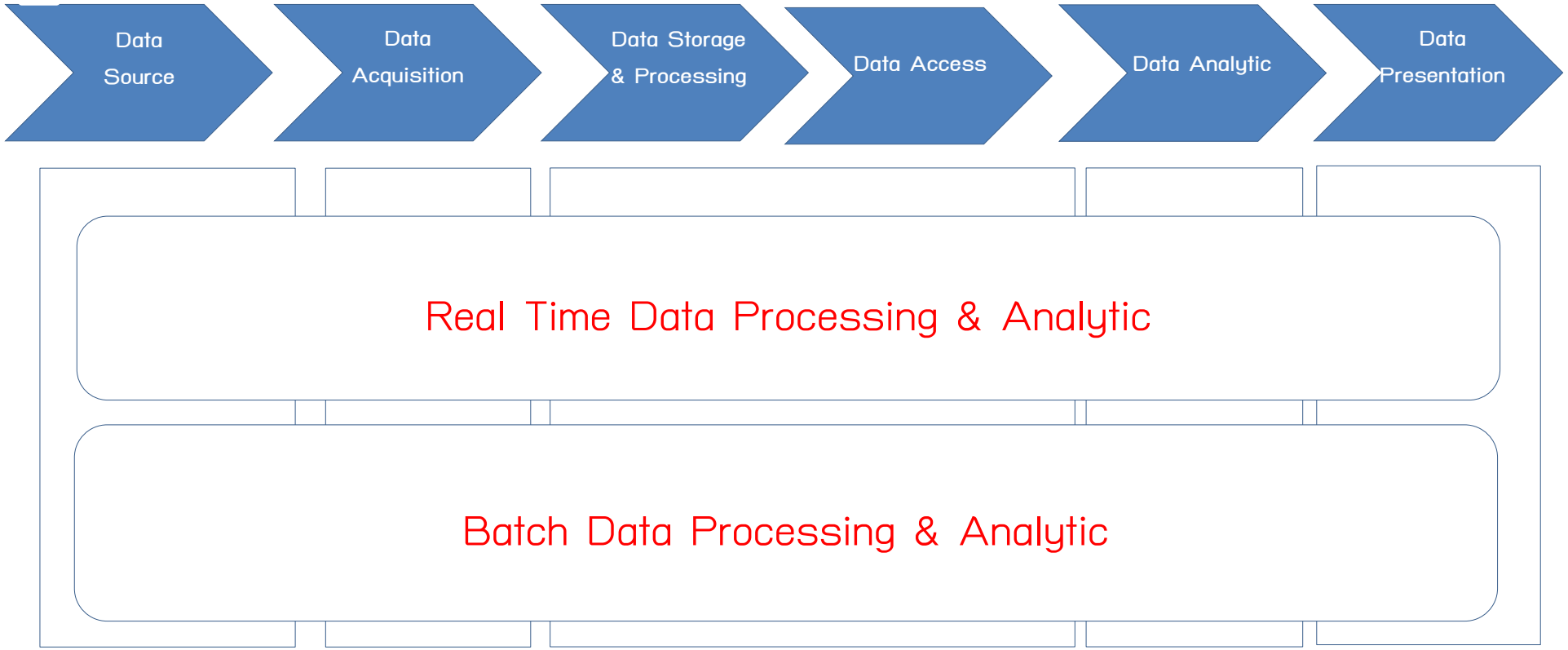
SlaveNode:

DataNode , NodeManager

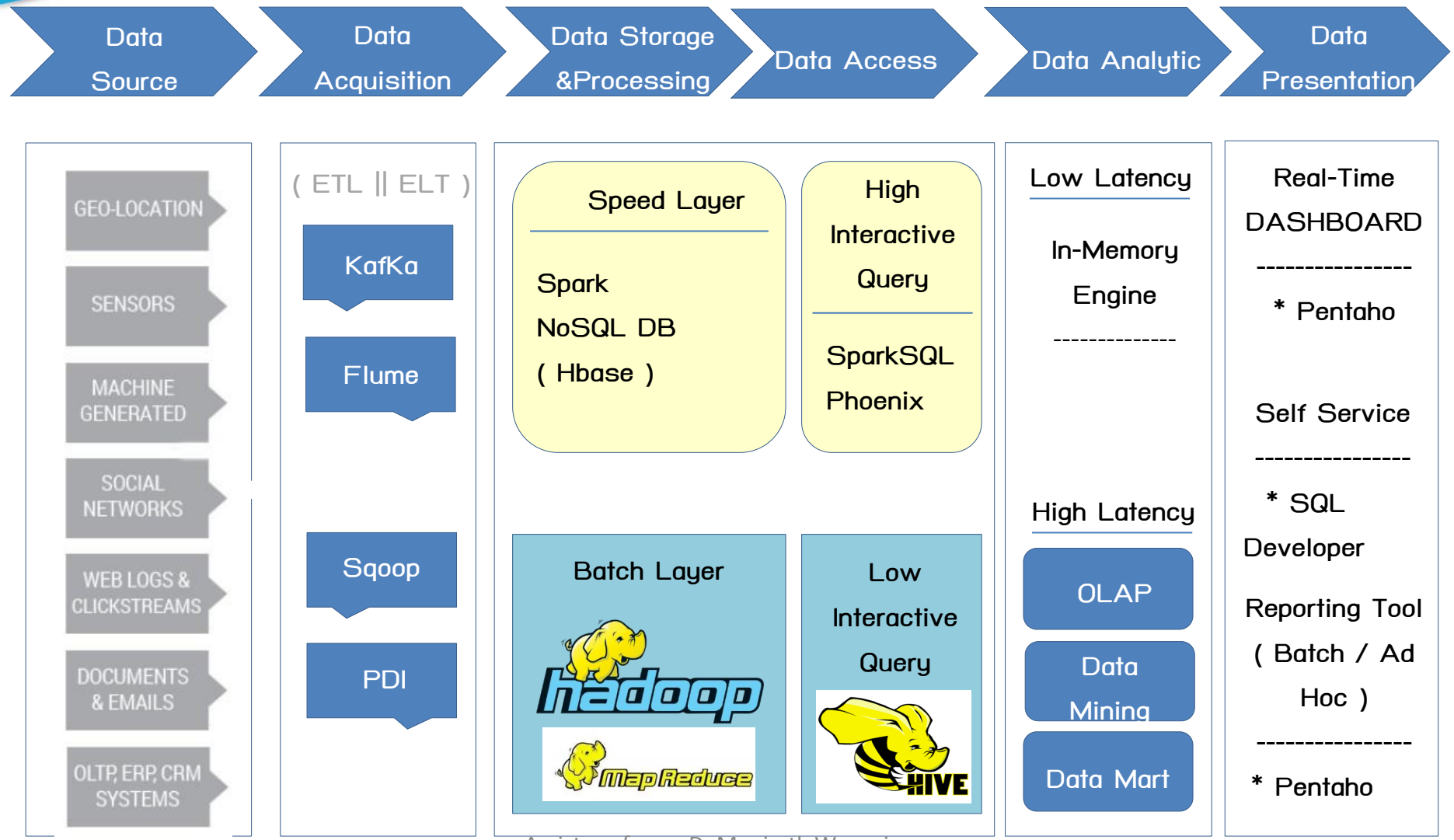
Big Data Architecture & Case :

Big Data Technology & System Tier :

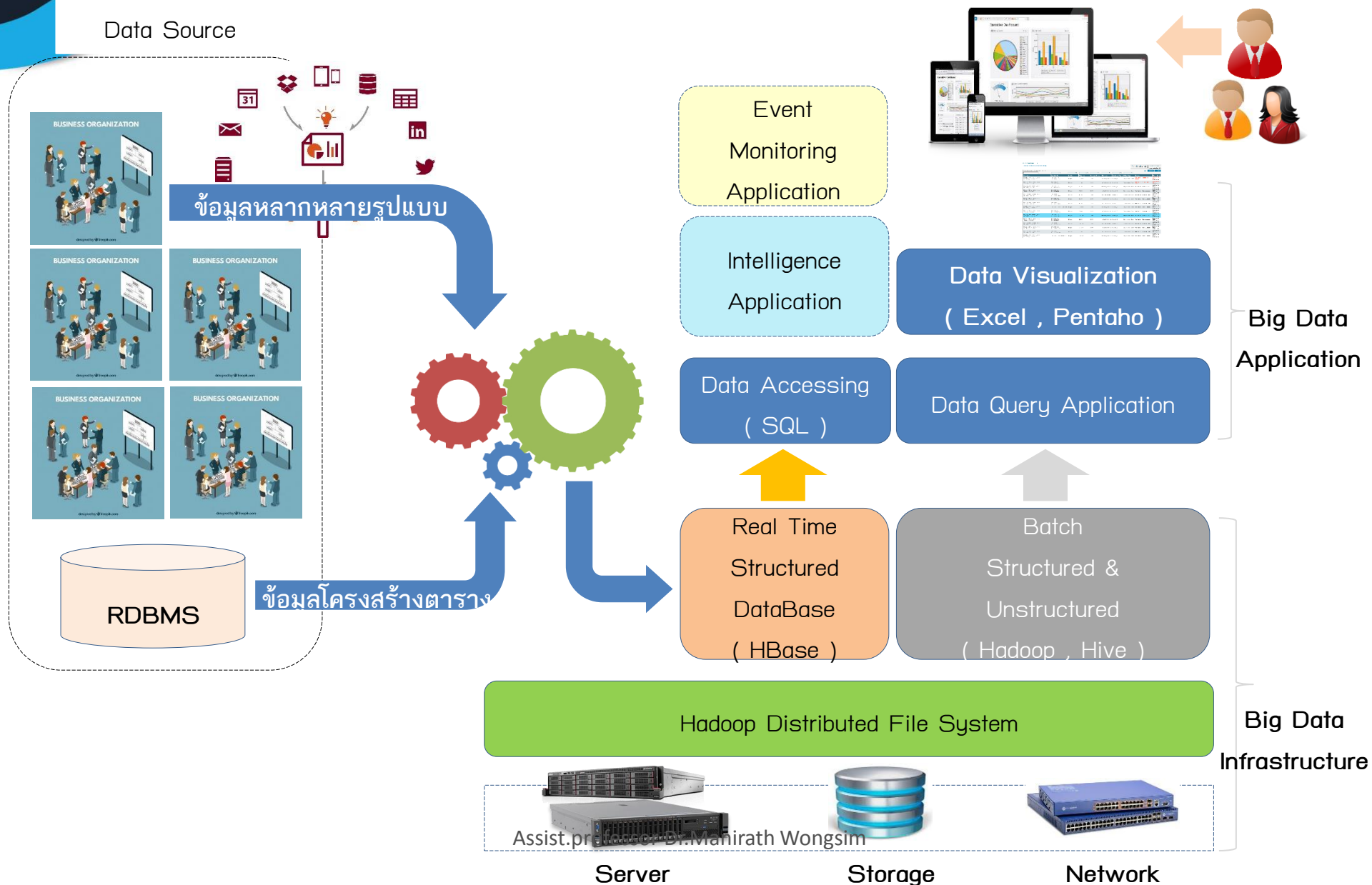
{ องค์กรมีข้อมูล } { เอาข้อมูลเข้า } { เก็บข้อมูล } { ประมวลผลข้อมูล } { เรียกข้อมูล } { วิเคราะห์ข้อมูล } { แสดงข้อมูล }



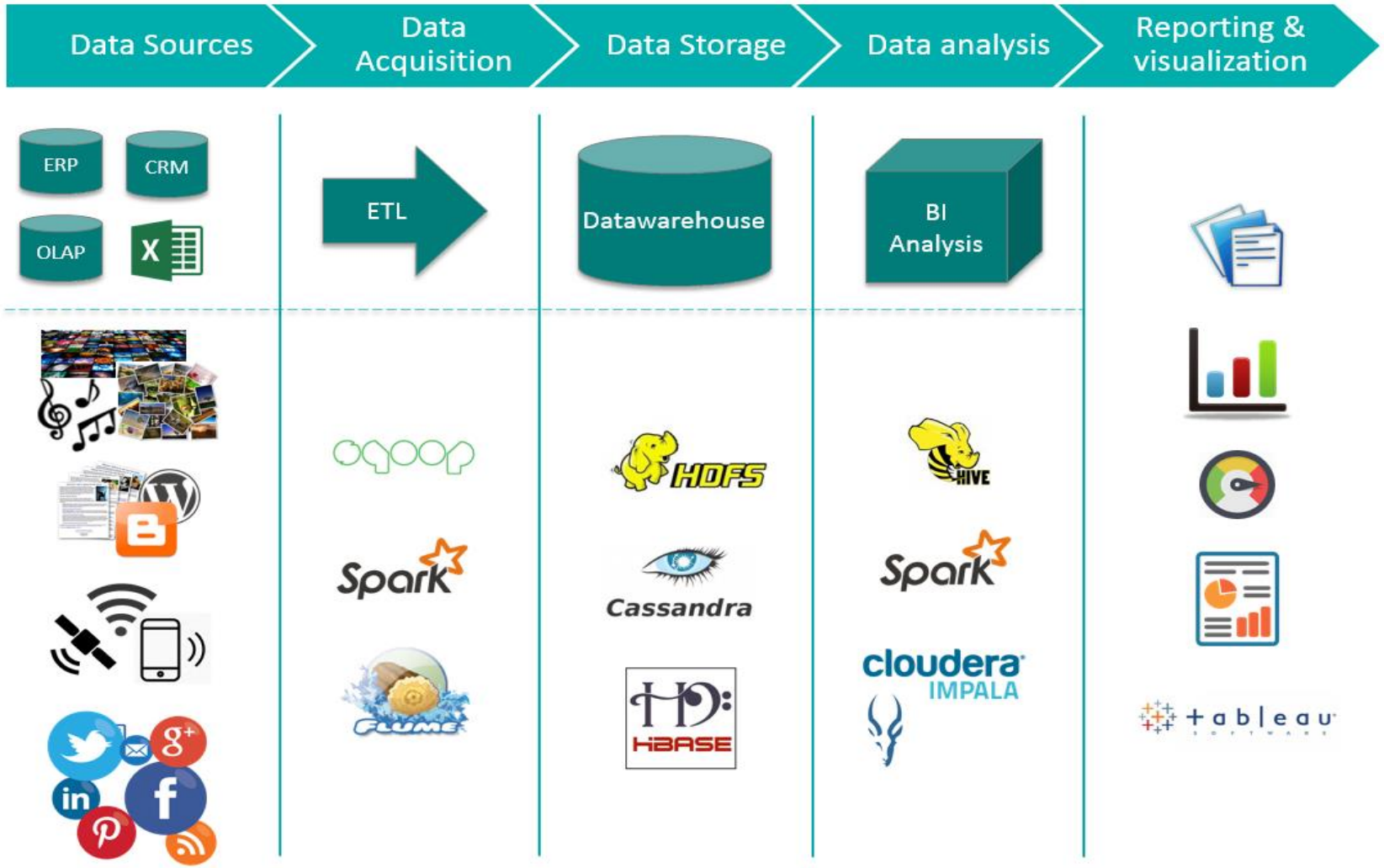
Big Data Technology & System Tier :



Big Data Technology & System Tier :

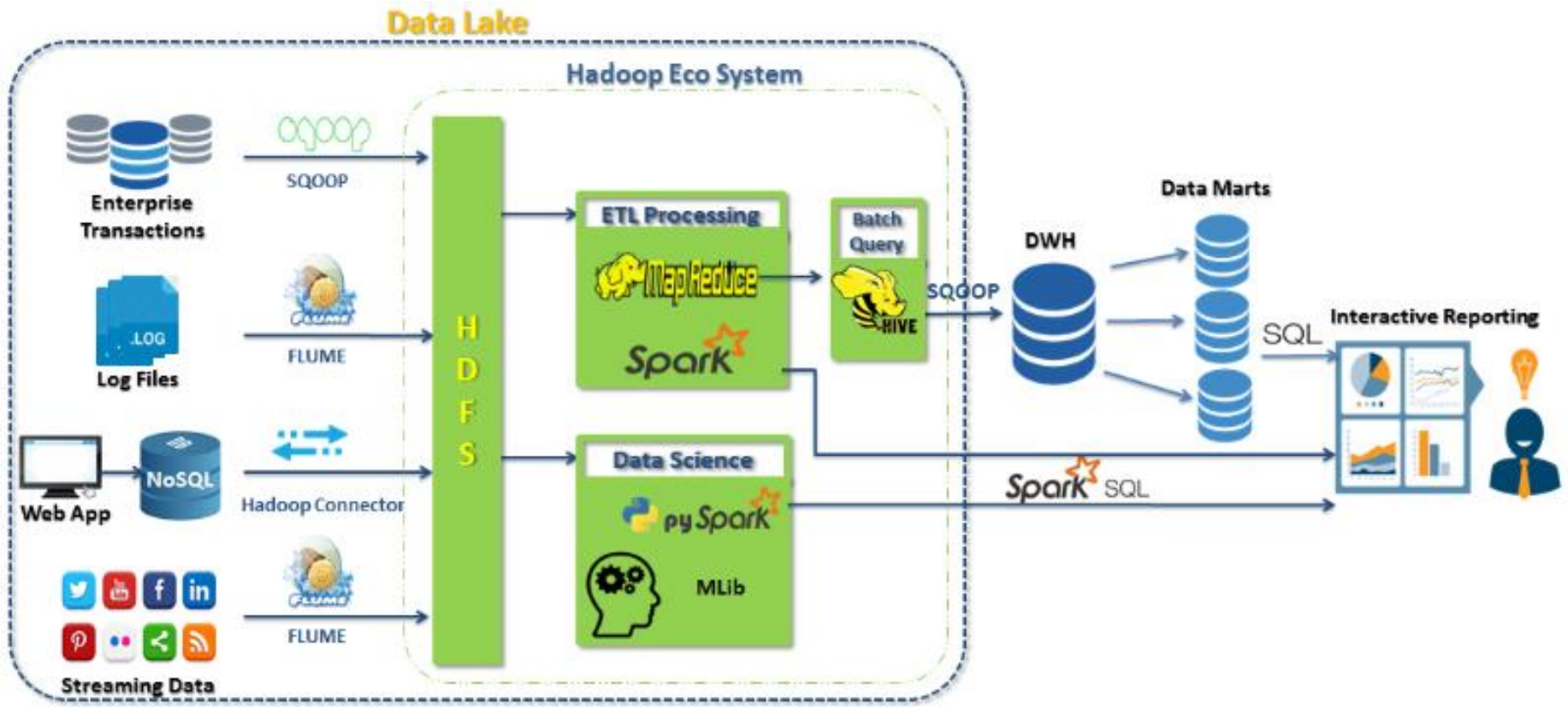


Big Data Technology & System Tier :



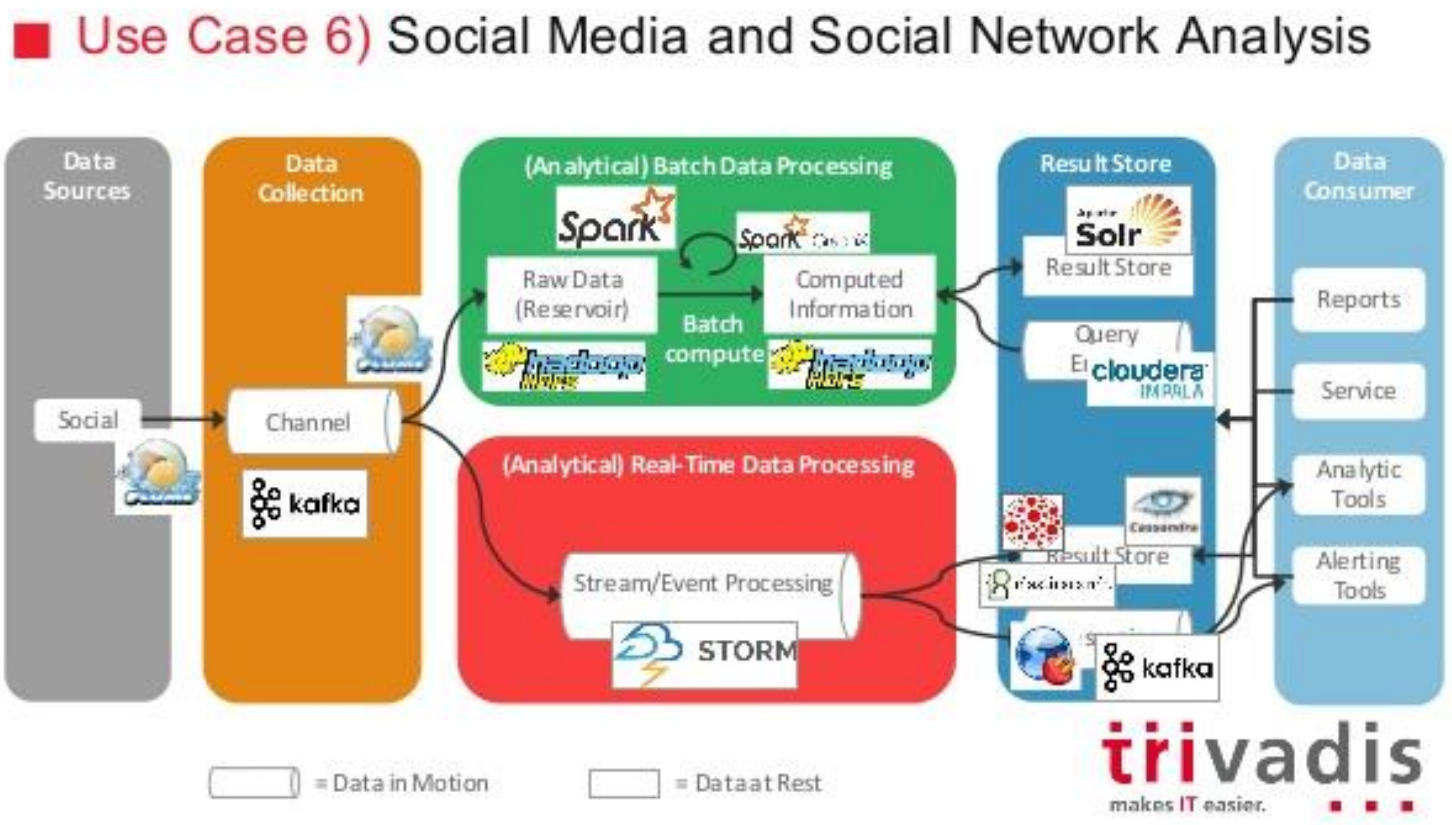
http://www.clearpeaks.com/blog/wp-content/uploads/2015/07/Image-1.png

Big Data Architecture#1:



https://www.linkedin.com/pulse/beyond-data-warehouse-lake-sirish-m-simha

Big Data Architecture#1:



https://image.slidesharecdn.com/bigdataarchitecture-v1-150926202752-lva1-app6892/95/big-data-architecture-30-638.jpg?cb=1449318045

Big Data Architecture#1 :

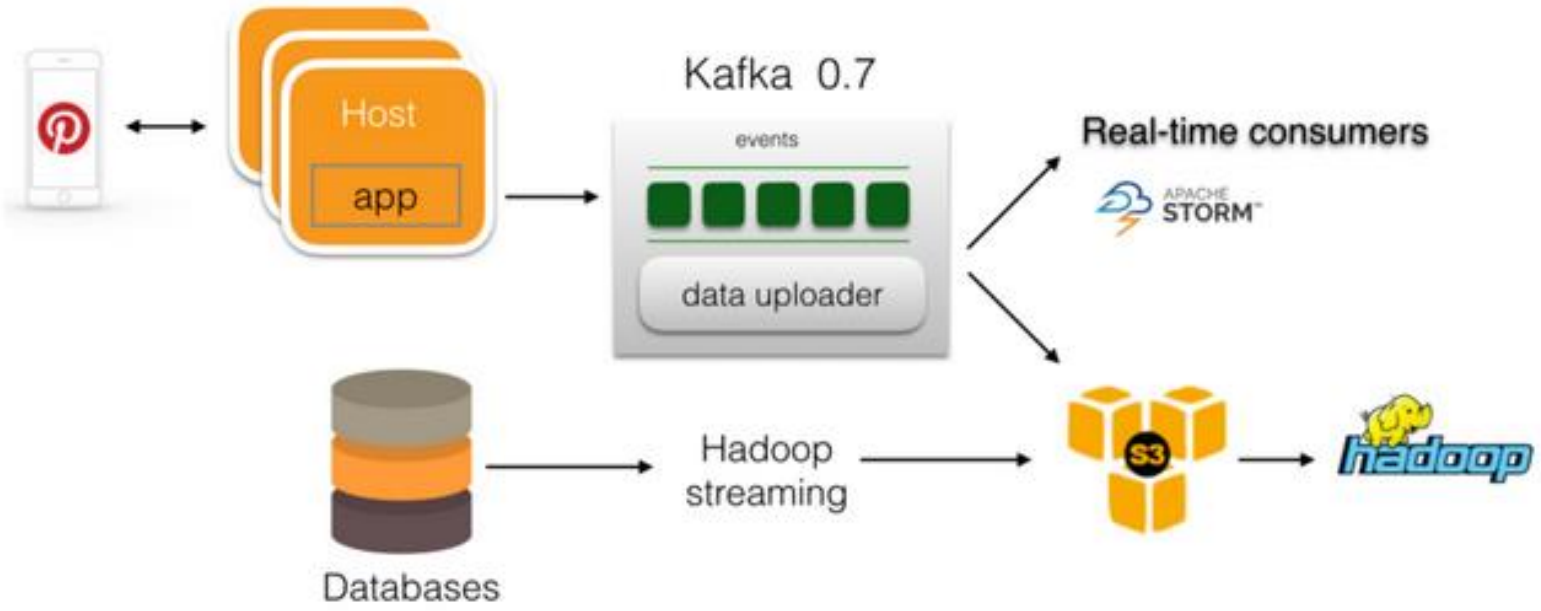
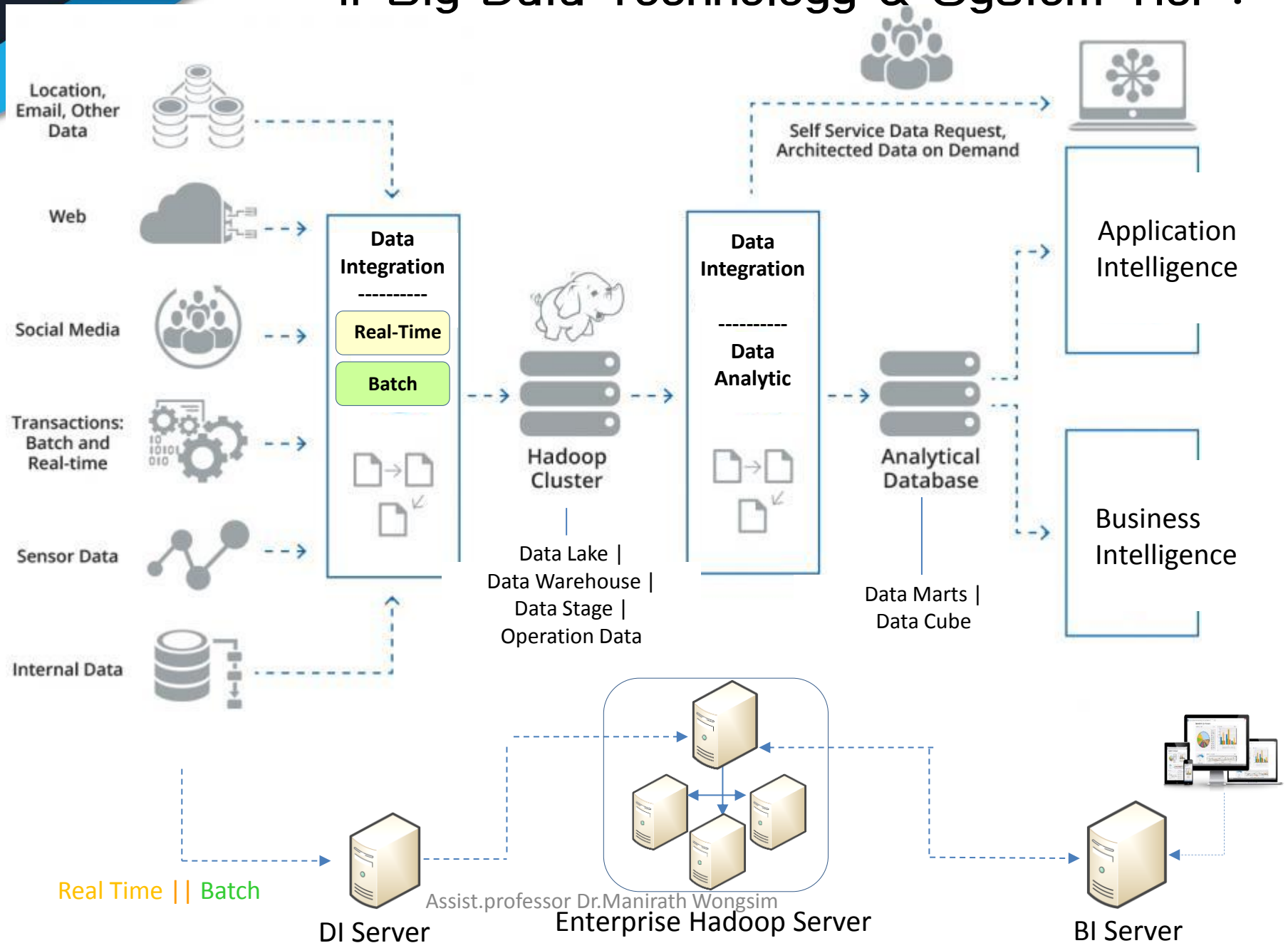
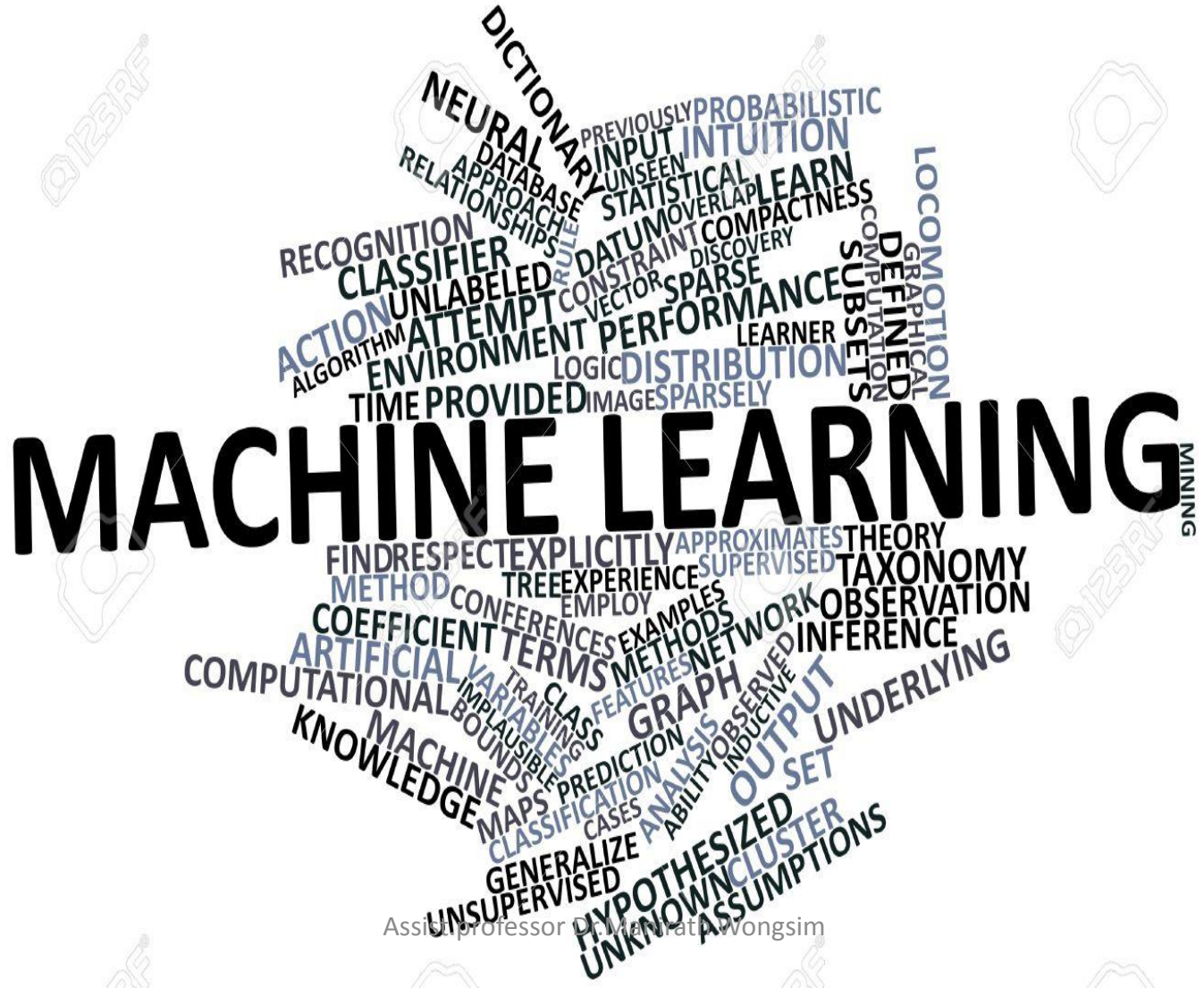


Figure 1. Data ingestion infrastructure in 2014

https://medium.com/@Pinterest_Engineering/scalable-and-reliable-data-ingestion-at-pinterest-b921c2ee8754

Big Data Technology & System Tier :




MACHINE LEARNING
DICTIONARY
NEURAL
PREVIOUSLY
PROBABILISTIC
INPUT
INTUITION
DATABASE
APPROACH
RELATIONSHIPS
UNSEEN
STATISTICAL
LEARN
RECOGNITION
CLASSIFIER
RULE
DATUM
OVERLAP
CONSTRAINT
COMPACTNESS
DISCOVERY
LOCOMOTION
ACTION
UNLABELED
ALGORITHM
ATTEMPT
VECTOR
SPARSE
ENVIRONMENT
PERFORMANCE
COMPUTATION
GRAPHICAL
DEFINED
SUBSETS
LEARNER
LOGIC
DISTRIBUTION
TIME
PROVIDED
IMAGE
SPARSELY
MINING
FINDRESPECTEXPLICITLY
APPROXIMATES
THEORY
SUPERVISED
TAXONOMY
TREE
EXPERIENCE
METHOD
CONFERENCES
EMPLOY
OBSERVATION
COEFFICIENT
TERMS
EXAMPLES
METHODS
NETWORK
INFERENCE
ARTIFICIAL
VARIABLE
TRAINING
FEATURES
COMPUTATIONAL
IMPLAUSIBLE
CLASS
BOUNDS
GRAPH
UNDERLYING
MACHINE
PREDICTION
OUTPUT
KNOWLEDGE
MAPS
CLASSIFICATION
ANALYSIS
INDUCTIVE
OBSERVED
ABILITY
SET
CASES
GENERALIZE
HYPOTHESIZED
UNSUPERVISED
CLUSTER
UNKNOWN
ASSUMPTIONS

Why "Learn"?

- Machine learning is programming computers to optimize a performance criterion using example data or past experience.
- There is no need to "learn" to calculate payroll
- Learning is used when:
  - Human expertise does not exist (navigating on Mars),
  - Humans are unable to explain their expertise (speech recognition)
  - Solution changes in time (routing on a computer network)
  - Solution needs to be adapted to particular cases (user biometrics)

What We Talk About When We Talk About"Learning"

- Learning general models from a data of particular examples
- Data is cheap and abundant (data warehouses, data marts); knowledge is expensive and scarce.
- Example in retail: Customer transactions to consumer behavior:

  People who bought "Da Vinci Code" also bought "The Five People You Meet in Heaven" (www.amazon.com)

- Build a model that is ***a good and useful approximation*** to the data.

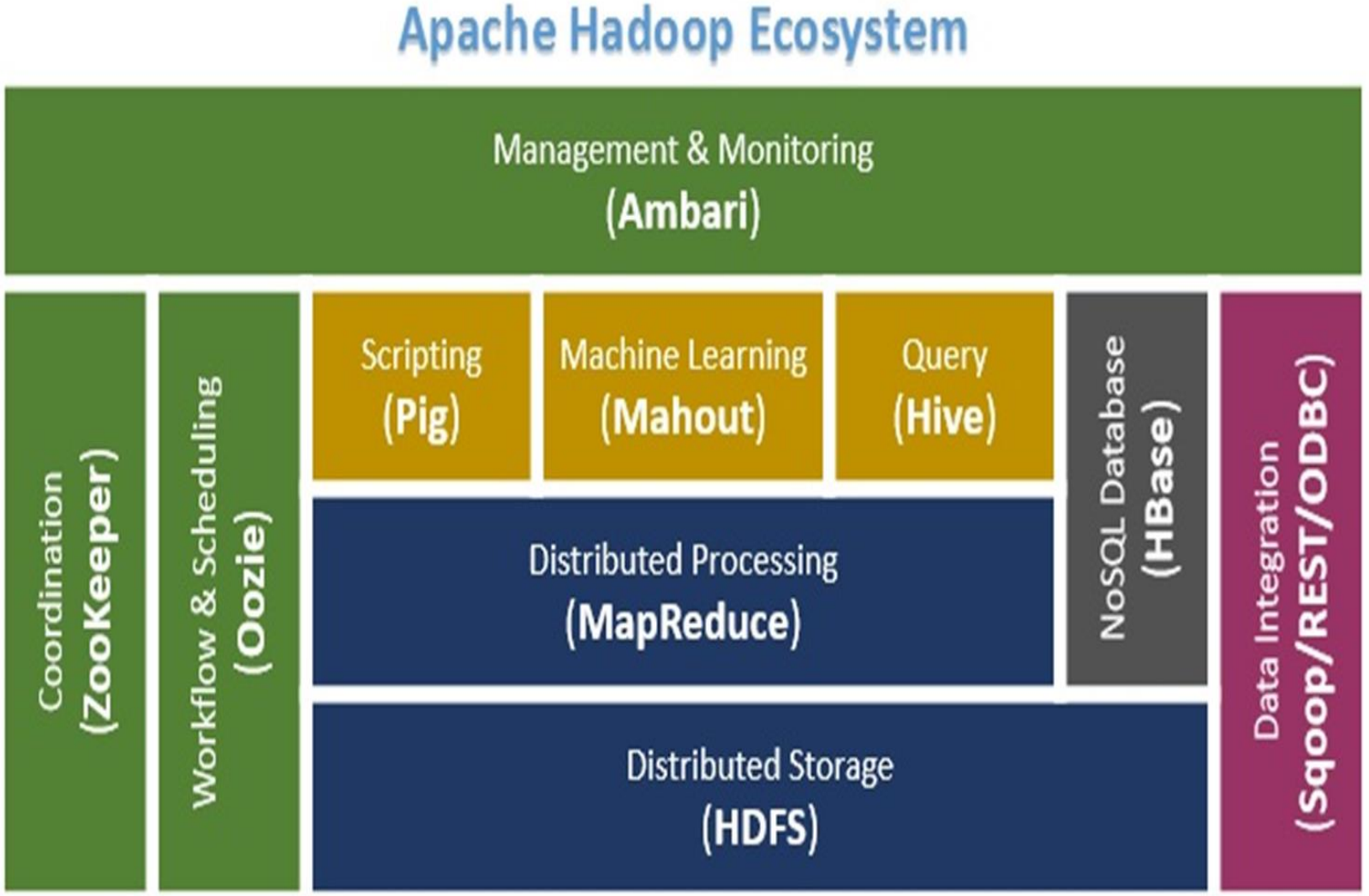
What is Machine Learning?

- Machine Learning
  - Study of algorithms that
  - improve their performance
  - at some task
  - with experience
- Optimize a performance criterion using example data or past experience.
- Role of Statistics: Inference from a sample
- Role of Computer science: Efficient algorithms to
  - Solve the optimization problem
  - Representing and evaluating the model for inference

Growth of Machine Learning

- Machine learning is preferred approach to
  - Speech recognition, Natural language processing
  - Computer vision
  - Medical outcomes analysis
  - Robot control
  - Computational biology
- This trend is accelerating
  - Improved machine learning algorithms
  - Improved data capture, networking, faster computers
  - Software too complex to write by hand
  - New sensors / IO devices
  - Demand for self-customization to user, environment
  - It turns out to be difficult to extract knowledge from human experts → *failure of expert systems in the 1980's.*

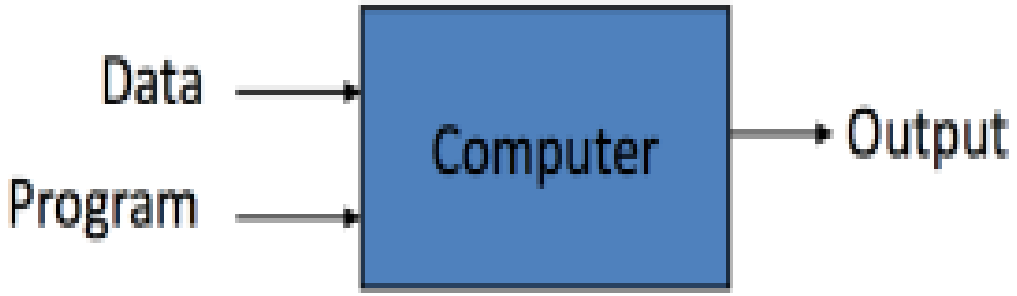
Machine Learning



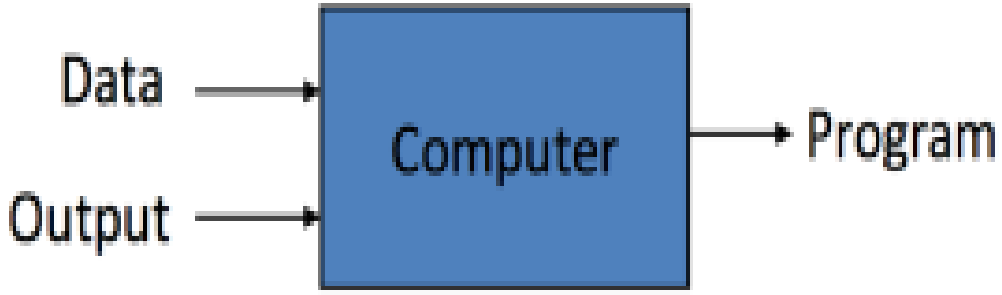
https://www.dezyre.com/article/hadoop-components-and-architecture-big-data-and-hadoop-training/114

Machine Learning

Traditional Programming

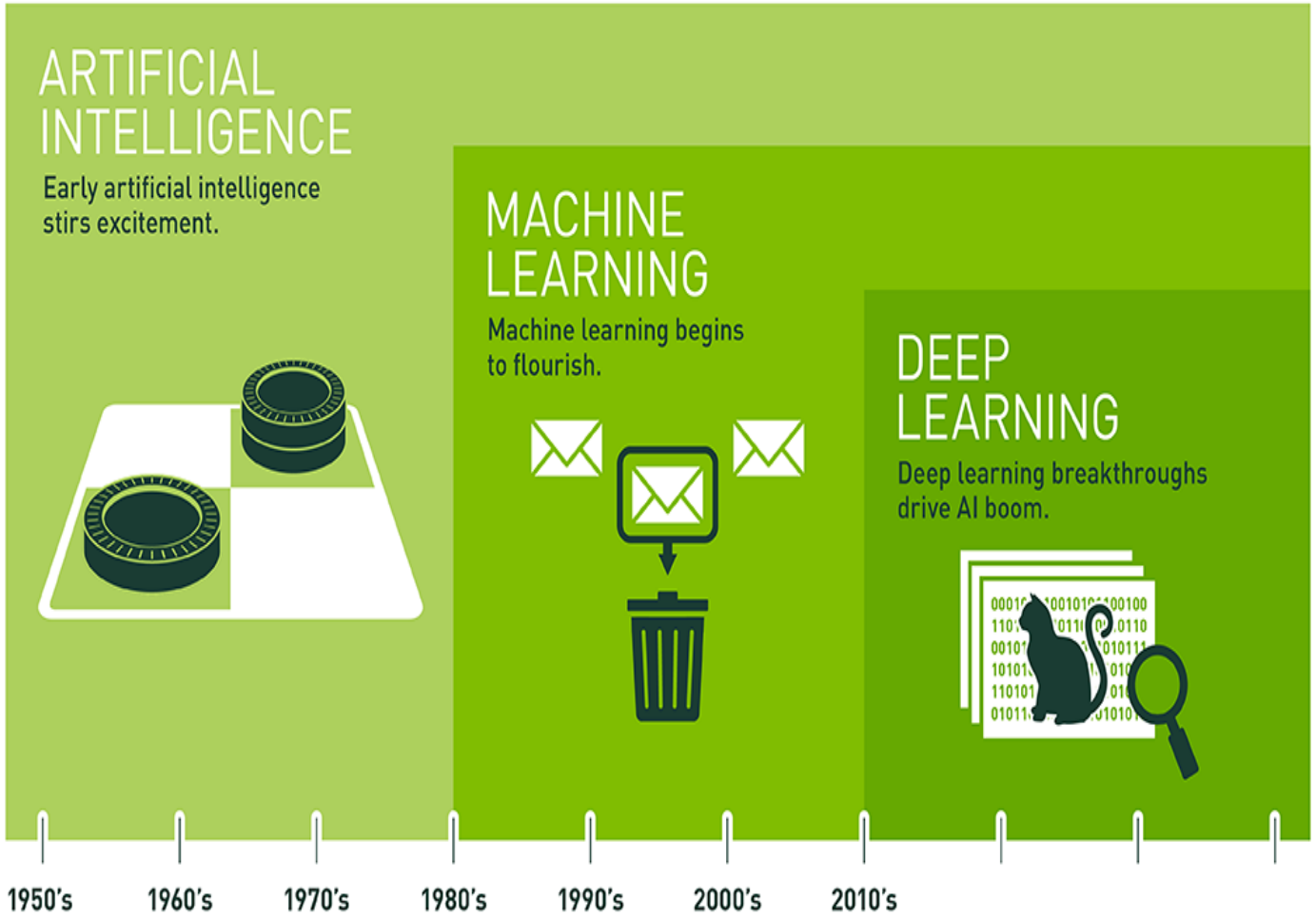


Machine Learning



http://machinelearningmastery.com/basic-concepts-in-machine-learning/

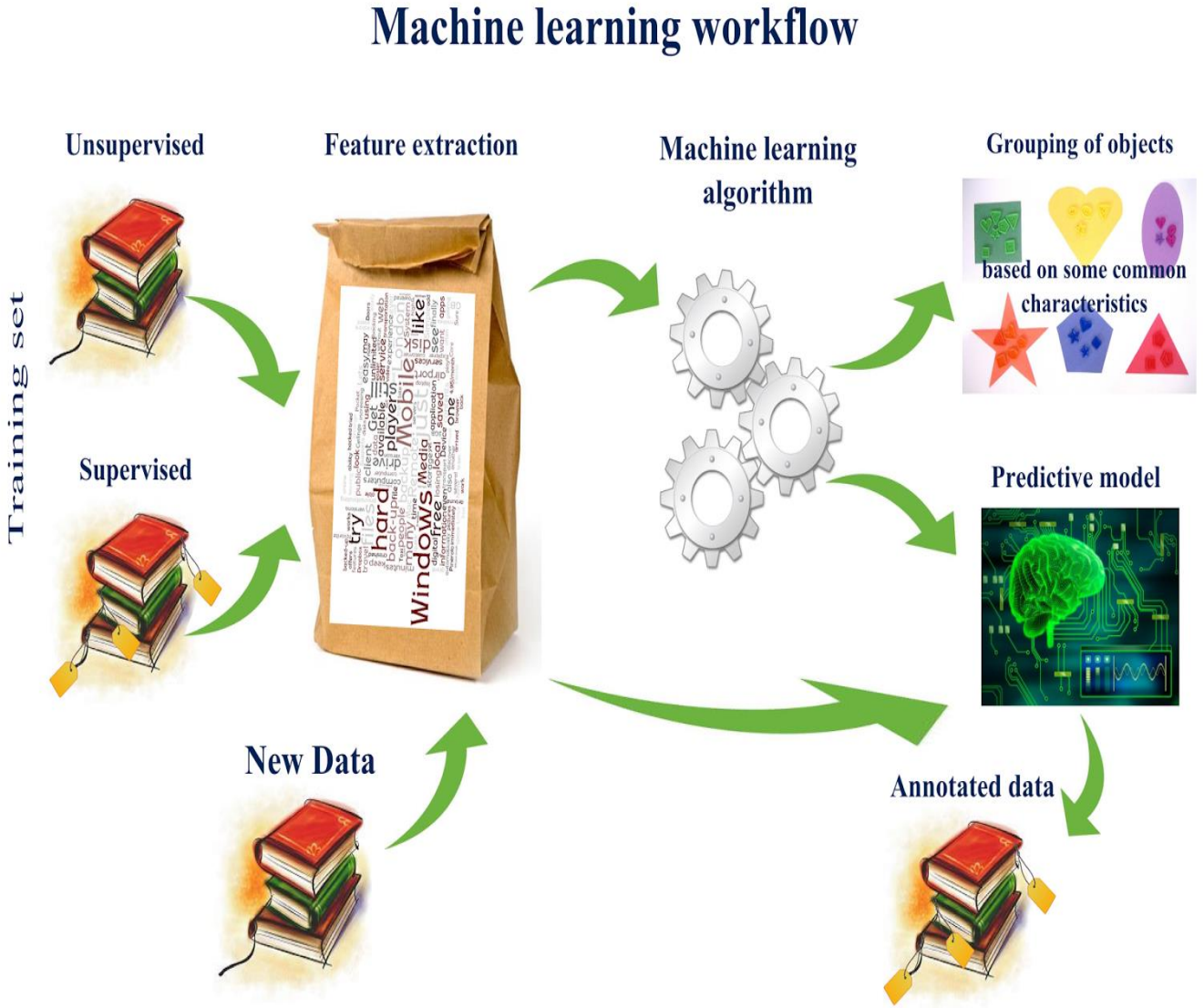
Machine Learning



Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

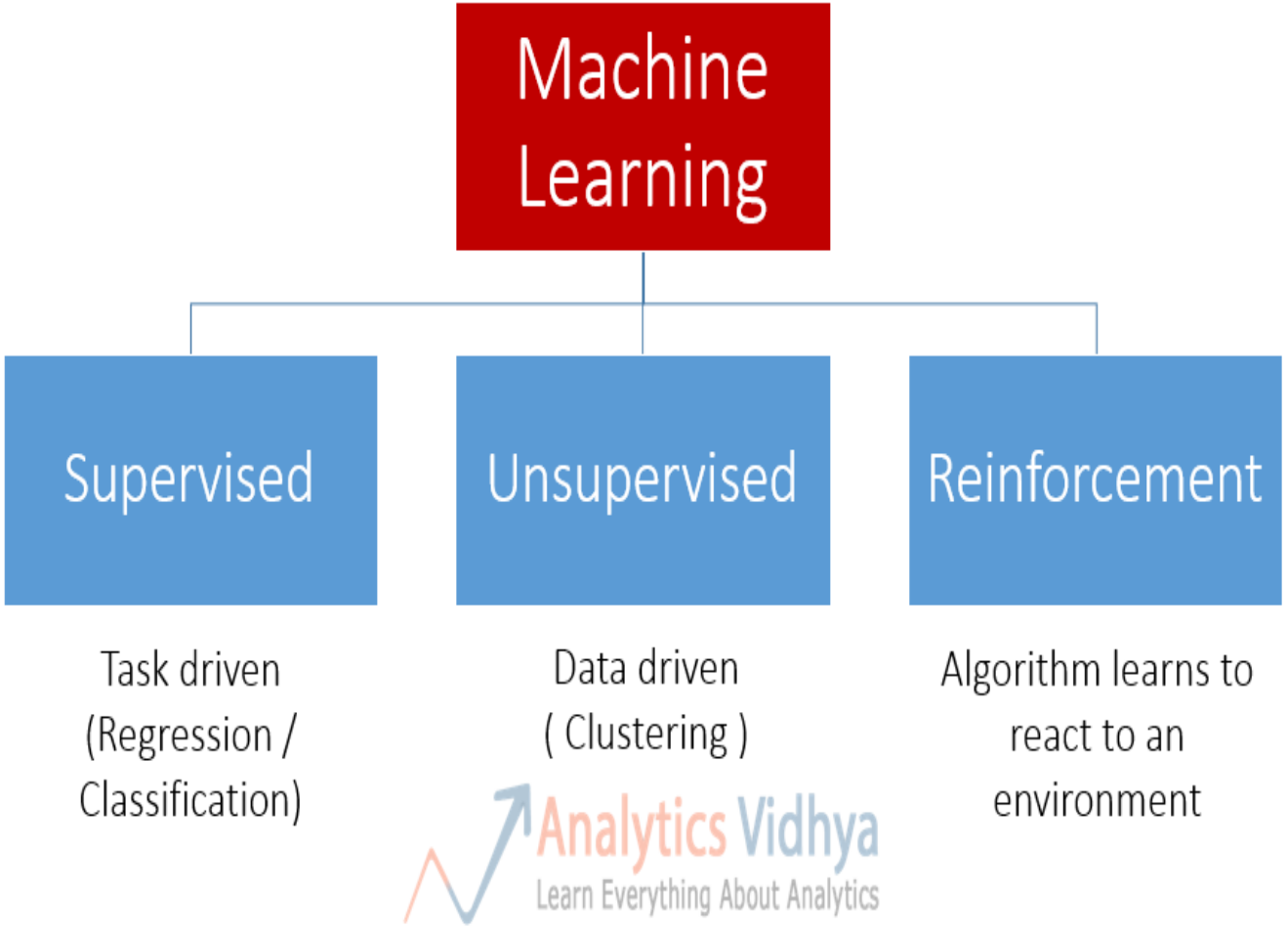
https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/

Machine Learning



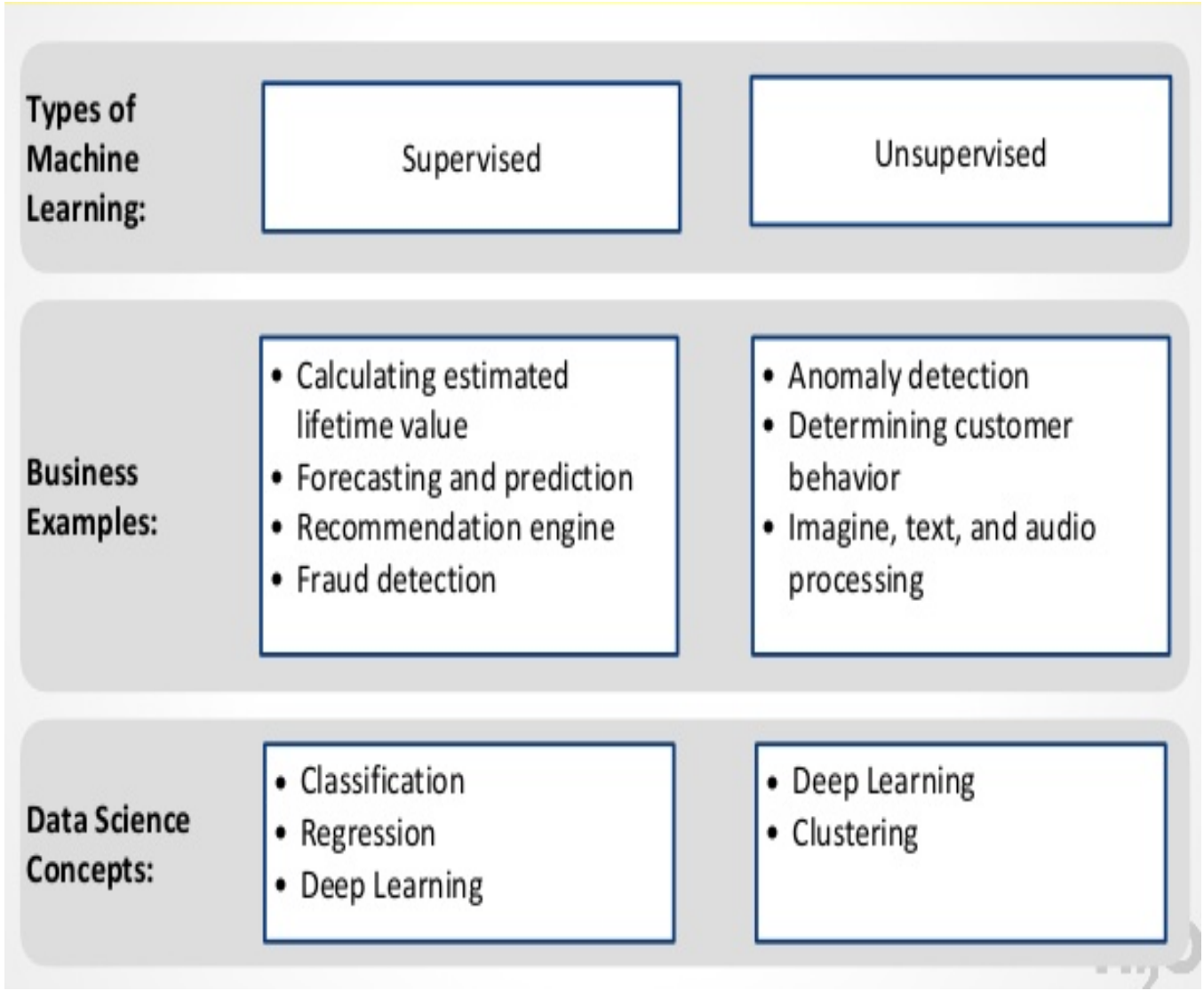
http://www.computervisionblog.com/2015/03/deep-learning-vs-machine-learning-vs.html

Types of Machine Learning



https://www.analyticsvidhya.com/blog/2015/06/machine-learning-basics/

Machine Learning Summary



https://www.slideshare.net/0xdata/h2o-world-machine-learning-for-nondata-scientists

Supervised Learning: Uses

Example: decision trees tools that create rules

- Prediction of future cases: Use the rule to predict the output for future inputs
- Knowledge extraction: The rule is easy to understand
- Compression: The rule is simpler than the data it explains
- Outlier detection: Exceptions that are not covered by the rule, e.g., fraud

Unsupervised Learning

- Learning “what normally happens”
- No output
- Clustering: Grouping similar instances
- Other applications: Summarization, Association Analysis
- Example applications
  - Customer segmentation in CRM
  - Image compression: Color quantization
  - Bioinformatics: Learning motifs

Reinforcement Learning

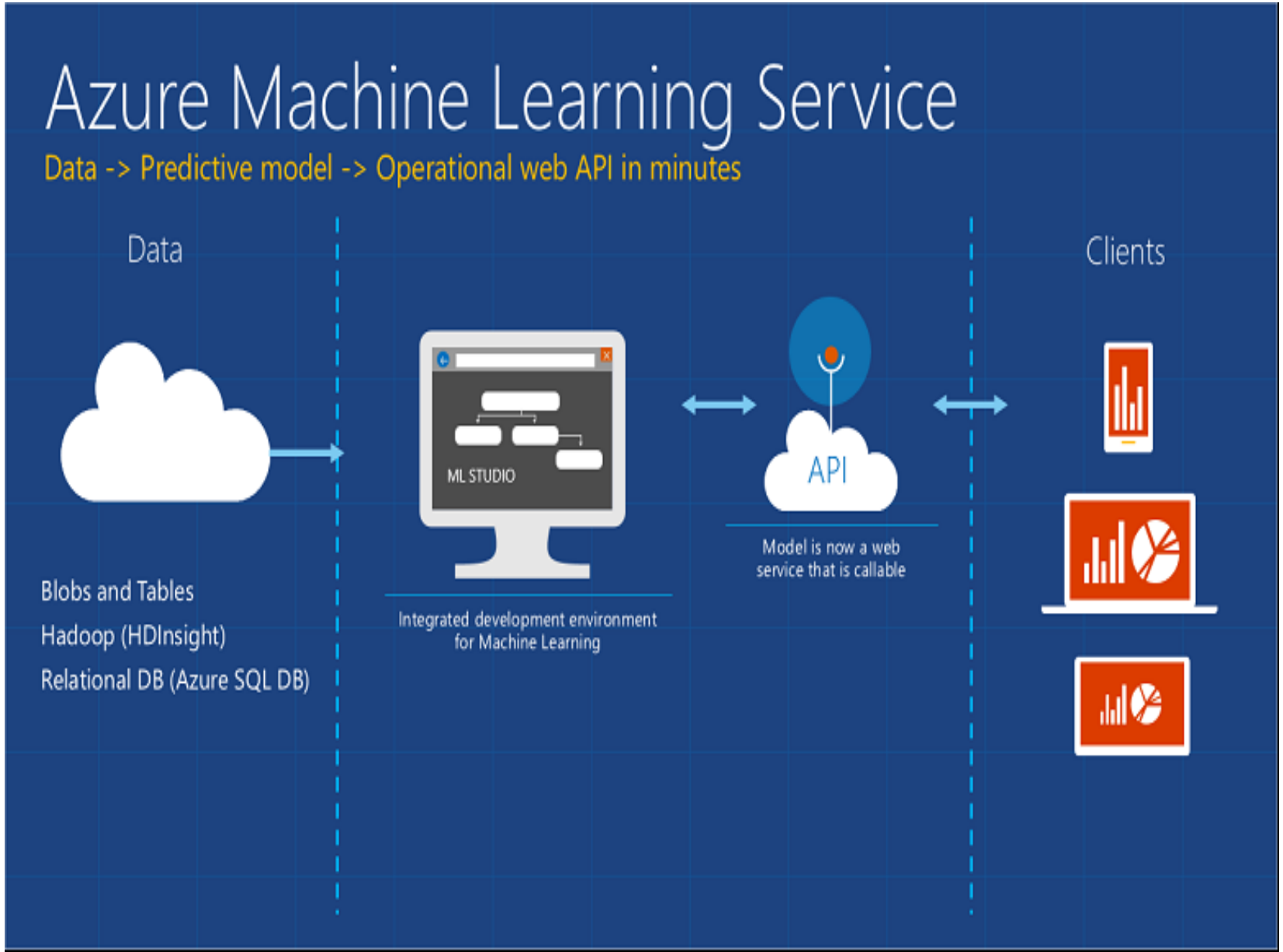
- Topics:
  - Policies: what actions should an agent take in a particular situation
  - Utility estimation: how good is a state (→used by policy)
- No supervised output but delayed reward
- Credit assignment problem (what was responsible for the outcome)
- Applications:
  - Game playing
  - Robot in a maze
  - Multiple agents, partial observability, ...

Machine Learning

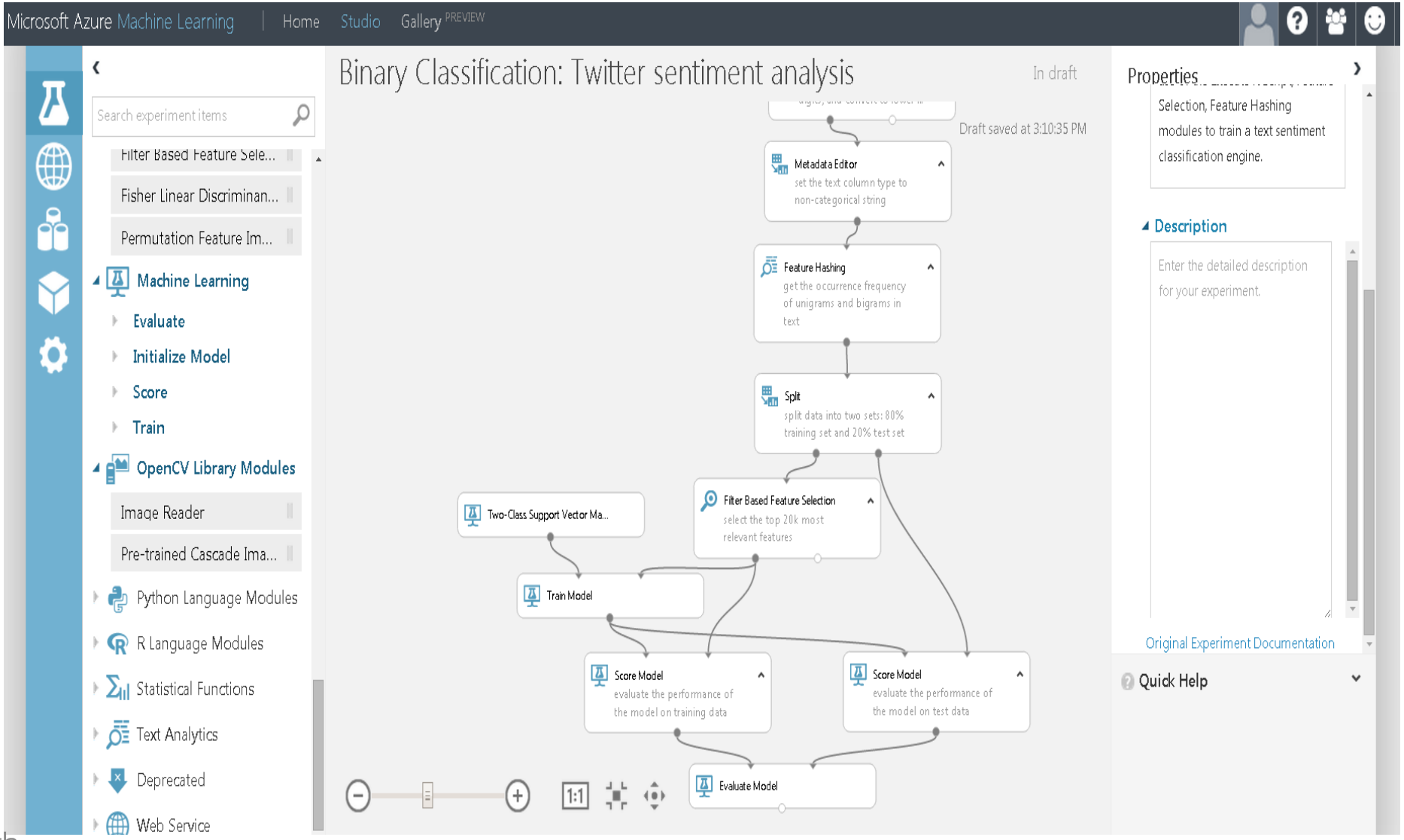
Frameworks for Machine Learning

https://www.linkedin.com/pulse/machine-learning-becomes-mainstream-how-increase-your-ronald-van-loon

Machine Learning

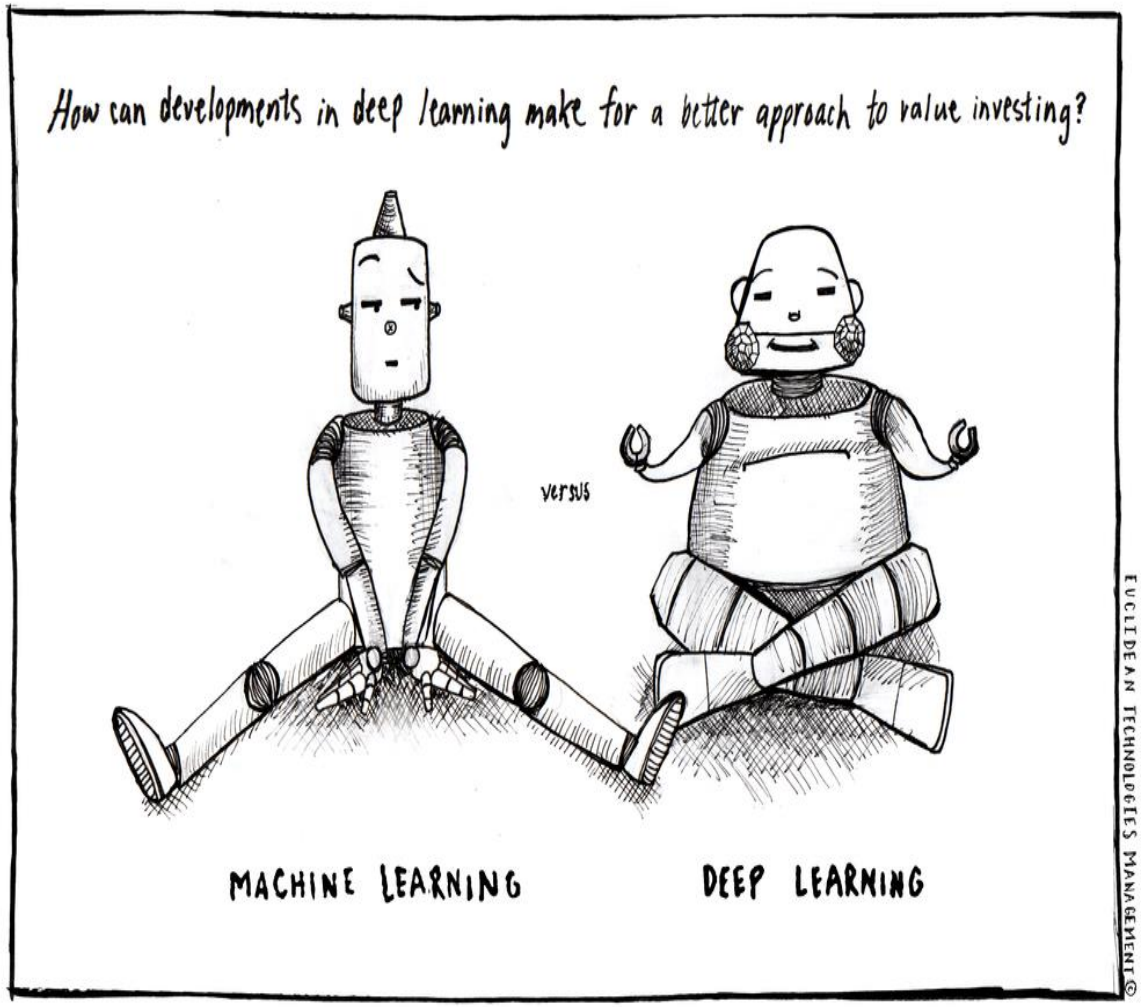


Machine Learning



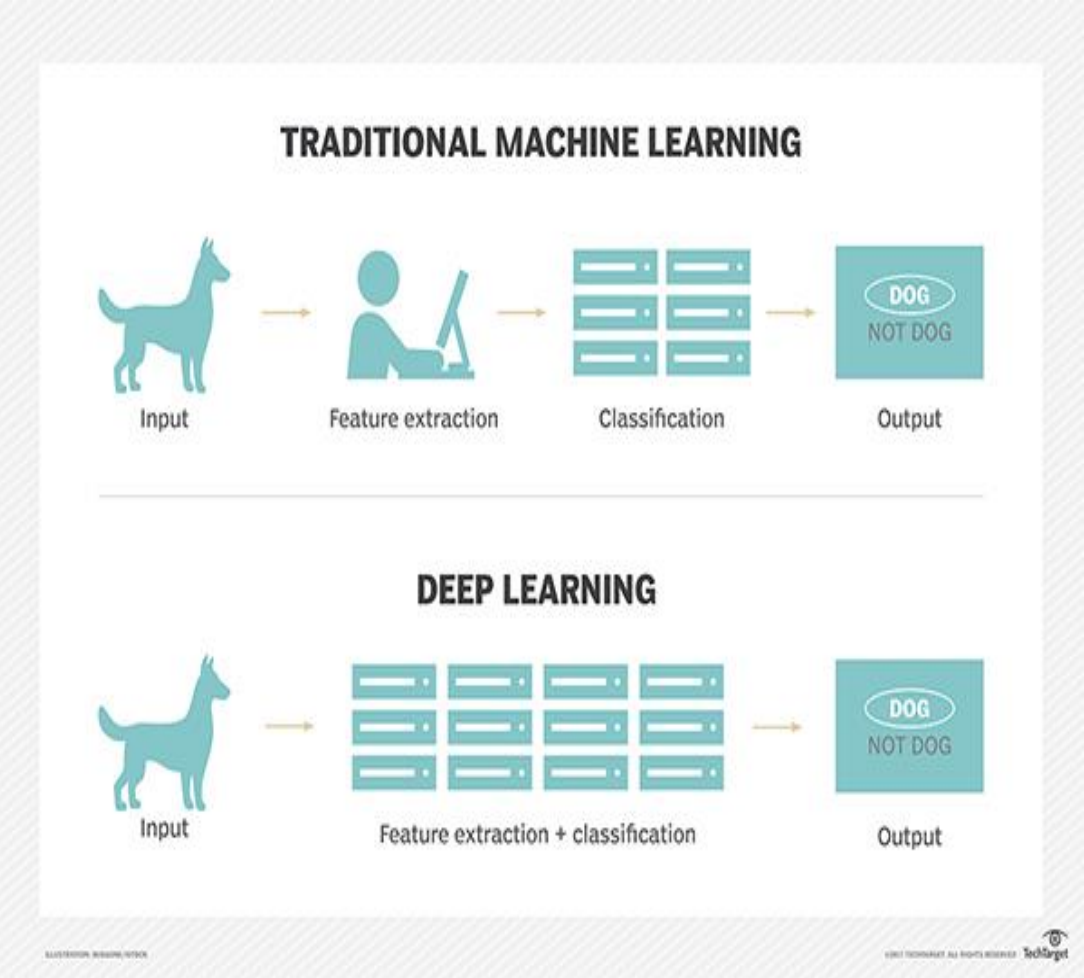
https://robusttechhouse.com/microsoft-azure-machine-learning-review/

Machine Learning



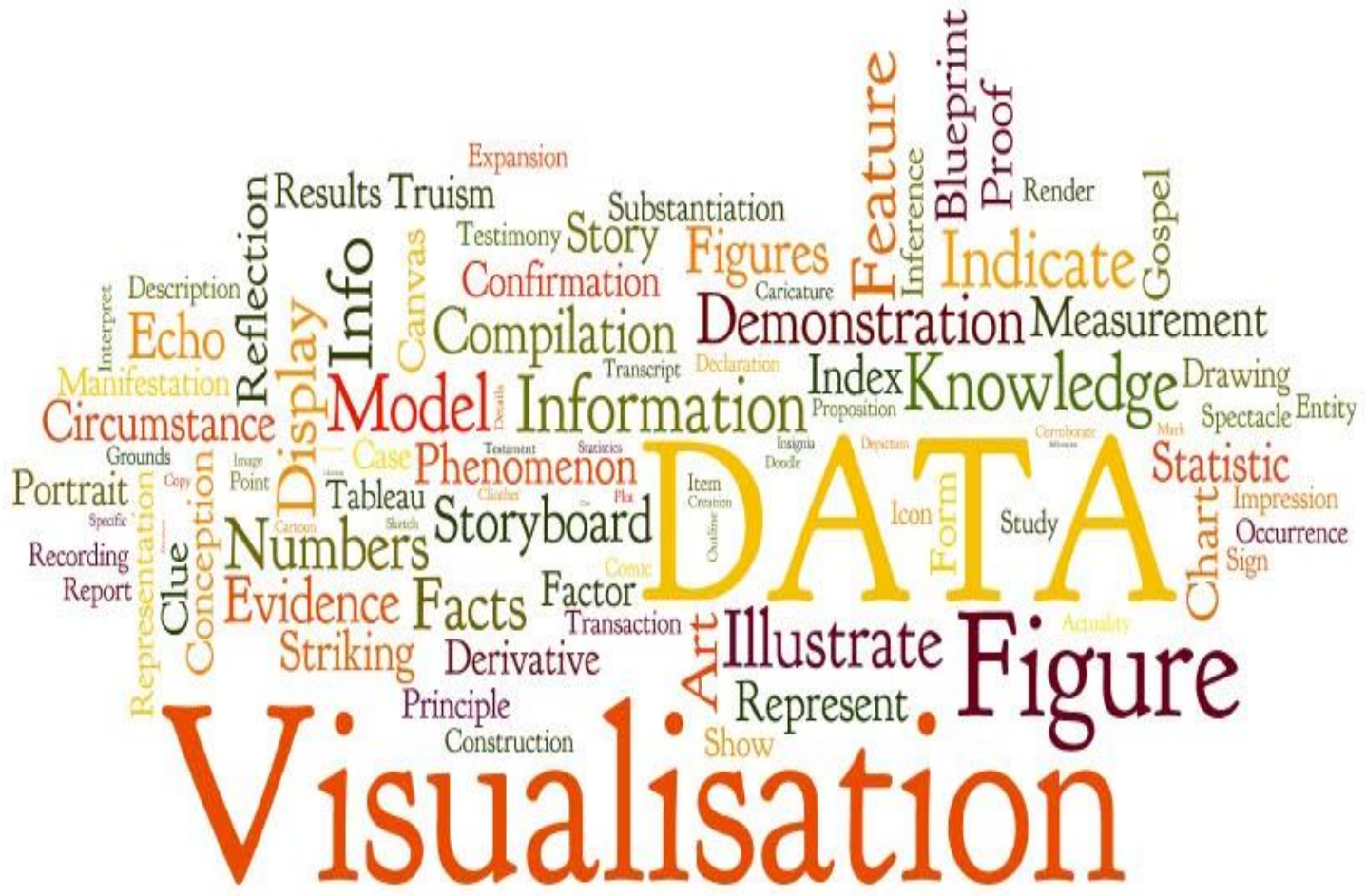
http://www.euclidean.com/deep-learning-and-value-investing/

Machine Learning



http://searchbusinessanalytics.techtarget.com/definition/deep-learning

Machine Learning

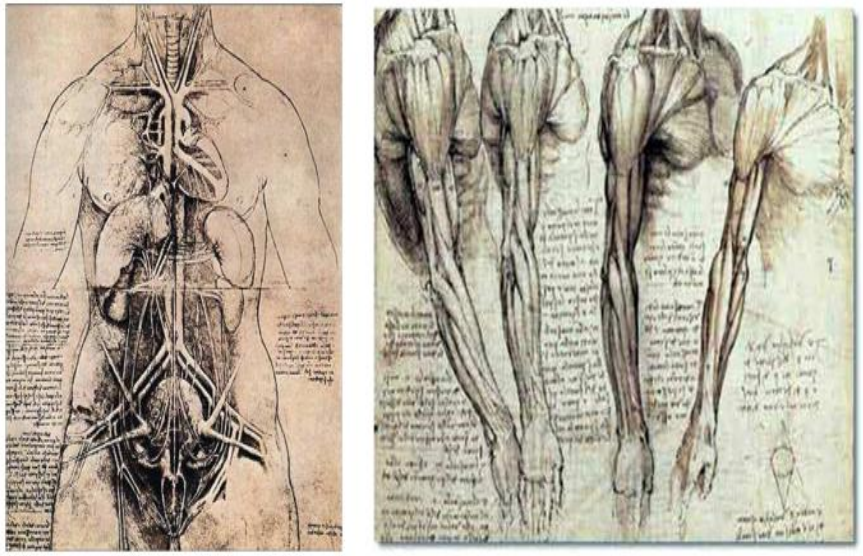


http://www.bigdatatraining.in/data-visualization-training-chennai/

History of Visualization

- Visualization = rather old

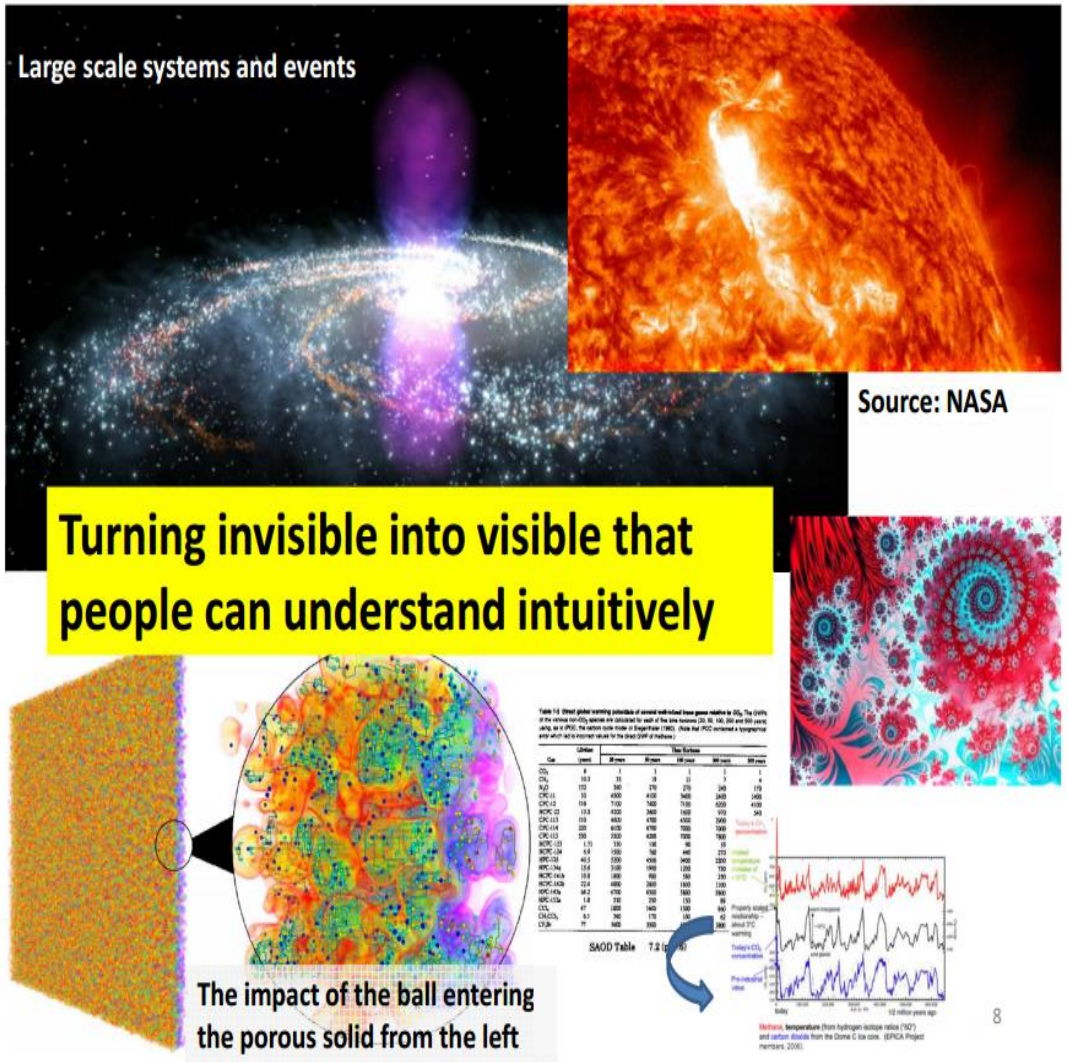
L. da Vinci (1452-1519)



- Often an intuitive step: graphical illustration

Image source: http://www.leonardo-da-vinci-biography.com/leonardo-da-vinci-anatomy.html

Data Visualization

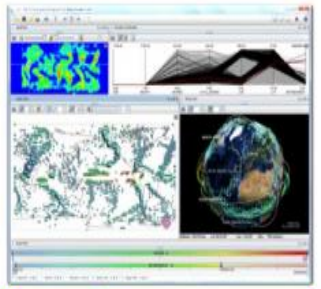


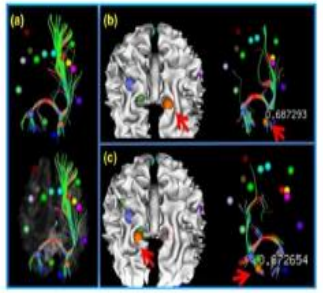
http://www2.cs.uh.edu/~gnawali/courses/cosc6397-f13

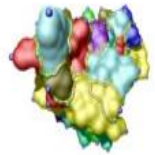
Data Visualization

What Does Visualization Do?

- Three types of goals for visualization
  - … to explore
    - Nothing is known,
    - Vis. used for data exploration
  - … to analyze
    - There are hypotheses,
    - Vis. used for Verification or Falsification
  - … to present
    - "everything" known about the data,
    - Vis. used for Communication of Results







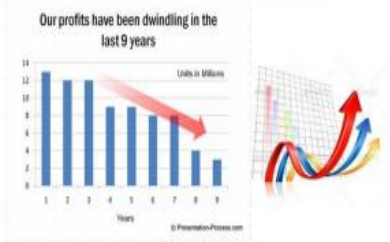


Image source: Google images

http://www2.cs.uh.edu/~gnawali/courses/cosc6397-f13

Examples of Pre-Established Color Meanings

Red	Green	Blue
Stop	On	Cool
Off	Plants	Safe
Dangerous	Carbon	Deep
Hot	Moving	Nitrogen
High stress	Money	
Oxygen		
Shallow		
Money loss		

http://www2.cs.uh.edu/~gnawali/courses/cosc6397-f13

Data Visualization

Improving the Understanding

- Principle 1: Provide explanations and draw conclusions
  - A graphical representation is often the means in which a hypothesis is confirmed or results are communicated.
  - Describe everything, draw attention to major features, describe conclusions

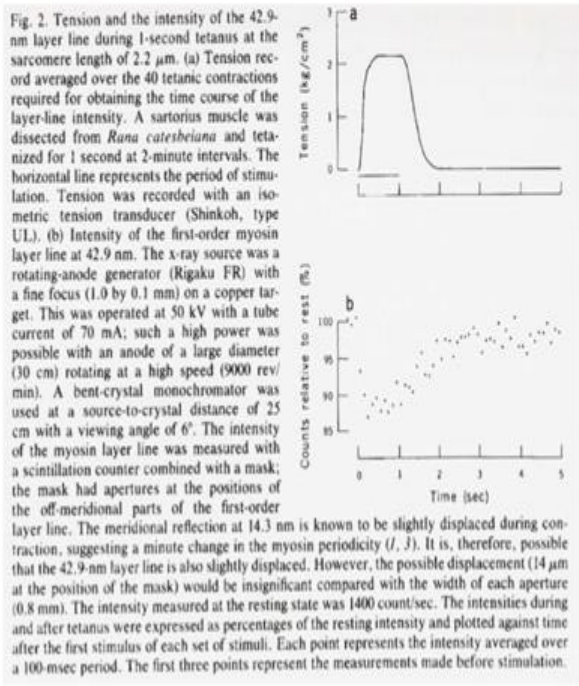
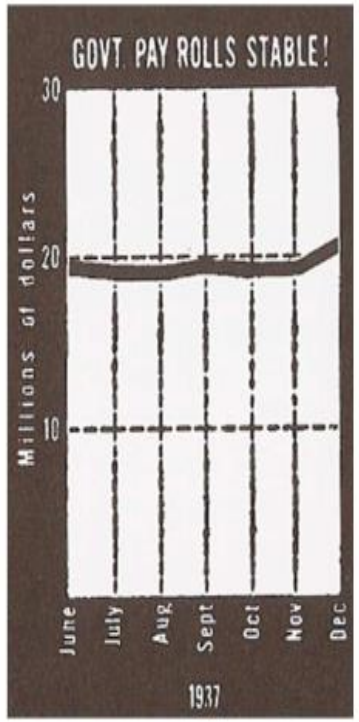


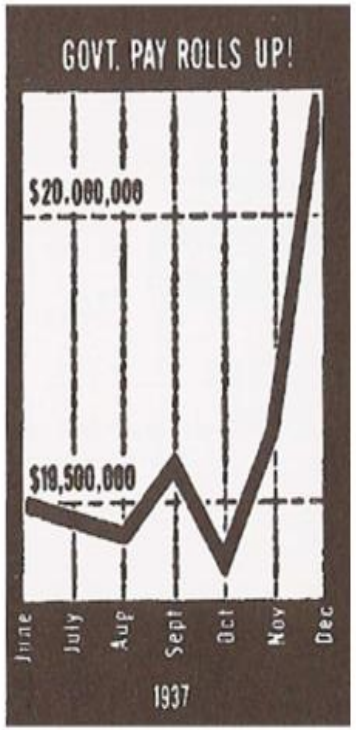
Fig. 2. Tension and the intensity of the 42.9-nm layer line during 1-second tetanus at the sarcomere length of 2.2 μm. (a) Tension record averaged over the 40 tetanic contractions required for obtaining the time course of the layer-line intensity. A sartorius muscle was dissected from *Rana catesbeiana* and tetanized for 1 second at 2-minute intervals. The horizontal line represents the period of stimulation. Tension was recorded with an isometric tension transducer (Shinkoh, type UL). (b) Intensity of the first-order myosin layer line at 42.9 nm. The x-ray source was a rotating-anode generator (Rigaku FR) with a fine focus (1.0 by 0.1 mm) on a copper target. This was operated at 50 kV with a tube current of 70 mA; such a high power was possible with an anode of a large diameter (30 cm) rotating at a high speed (9000 rev/min). A bent-crystal monochromator was used at a source-to-crystal distance of 25 cm with a viewing angle of 6°. The intensity of the myosin layer line was measured with a scintillation counter combined with a mask; the mask had apertures at the positions of the off-meridional parts of the first-order layer line. The meridional reflection at 14.3 nm is known to be slightly displaced during contraction, suggesting a minute change in the myosin periodicity (*1, 3*). It is, therefore, possible that the 42.9-nm layer line is also slightly displaced. However, the possible displacement (14 μm at the position of the mask) would be insignificant compared with the width of each aperture (0.8 mm). The intensity measured at the resting state was 1400 count/sec. The intensities during and after tetanus were expressed as percentages of the resting intensity and plotted against time after the first stimulus of each set of stimuli. Each point represents the intensity averaged over a 100-msec period. The first three points represent the measurements made before stimulation.

http://www2.cs.uh.edu/~gnawali/courses/cosc6397-f13

Improving the Understanding

- Principle 2: Use all available space.
  - Fill the data rectangle, only use zero if you need it



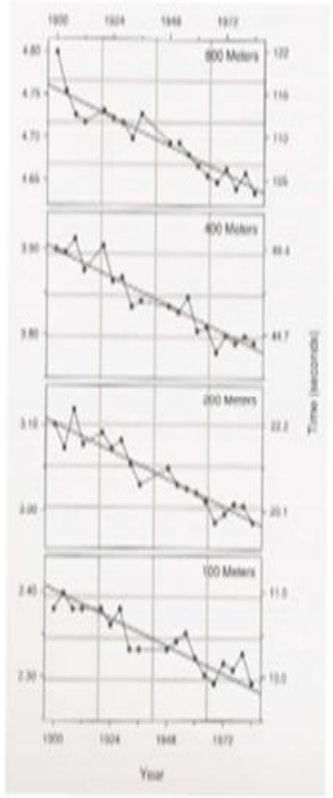


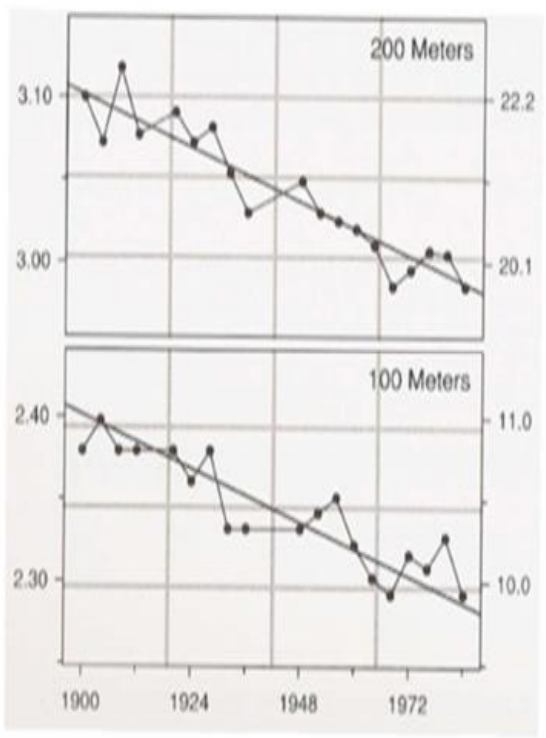
http://www2.cs.uh.edu/~gnawali/courses/cosc6397-f13

Data Visualization

Improving the Understanding

- Principle 3: Align juxtaposed plots
  - Make sure scales match and graphs are aligned



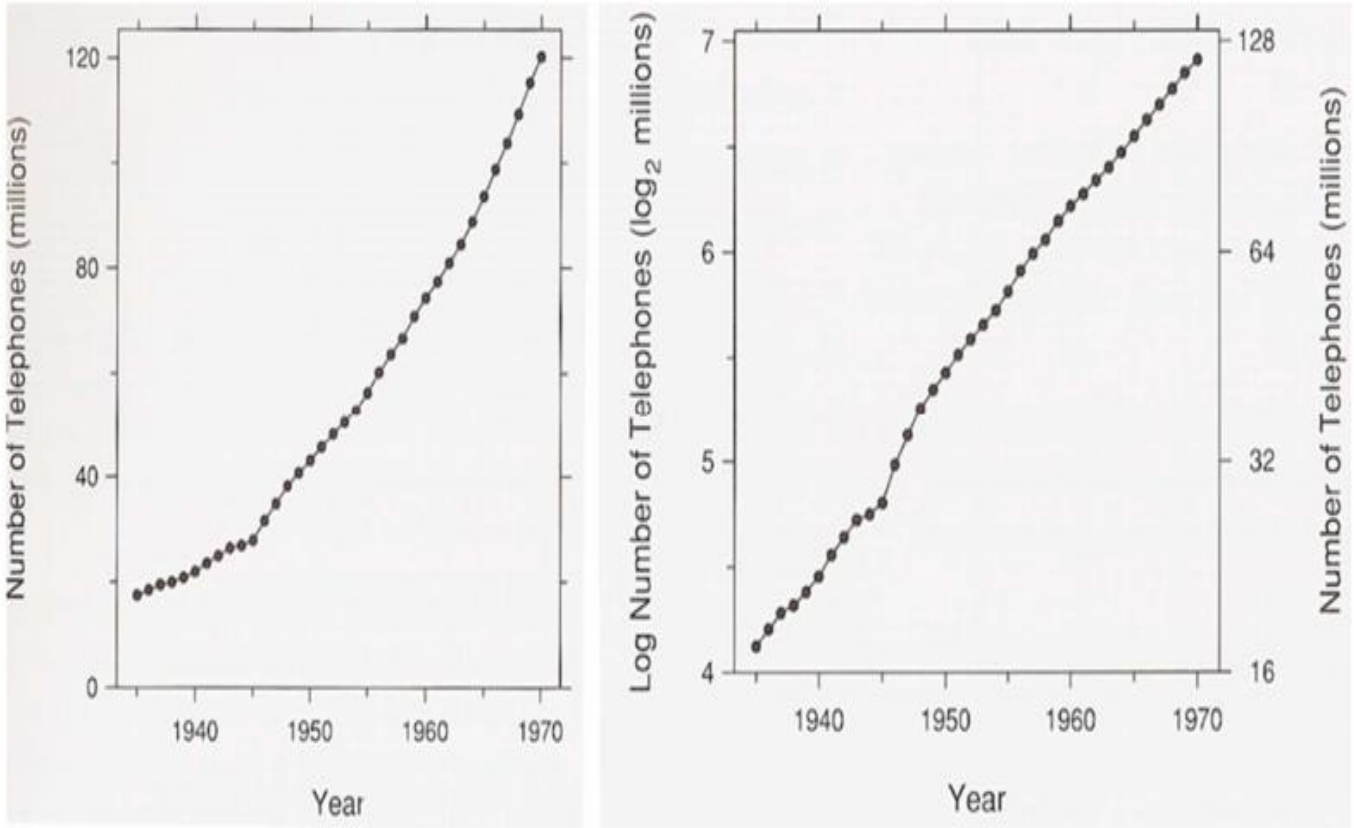


http://www2.cs.uh.edu/~gnawali/courses/cosc6397-f13

Data Visualization

Improving the Understanding

- Principle 4: Use log scales when appropriate
  - Used to show percentage change, multiplicative factors and skewness

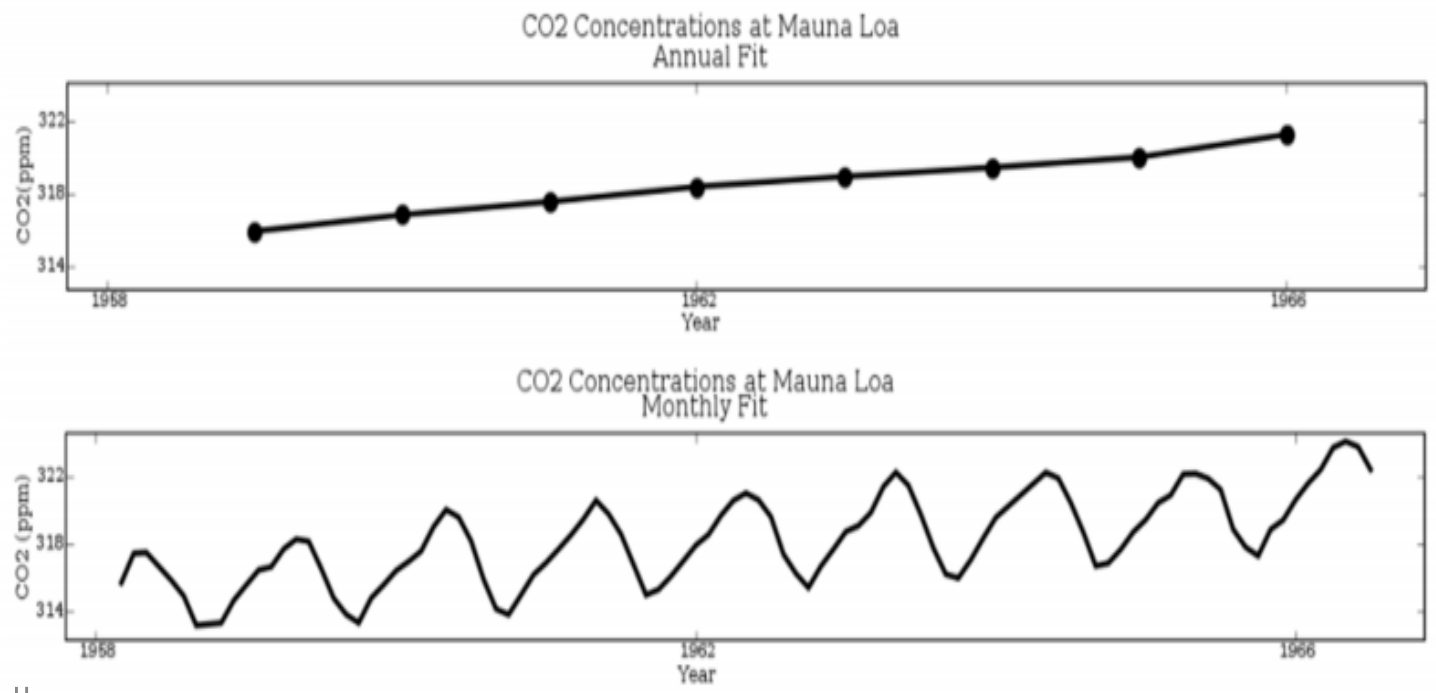


http://www2.cs.uh.edu/~gnawali/courses/cosc6397-f13

Data Visualization

Improving the Understanding

- Principle 5: Bank to 45^o
  - Optimize the aspect ratio of the plot



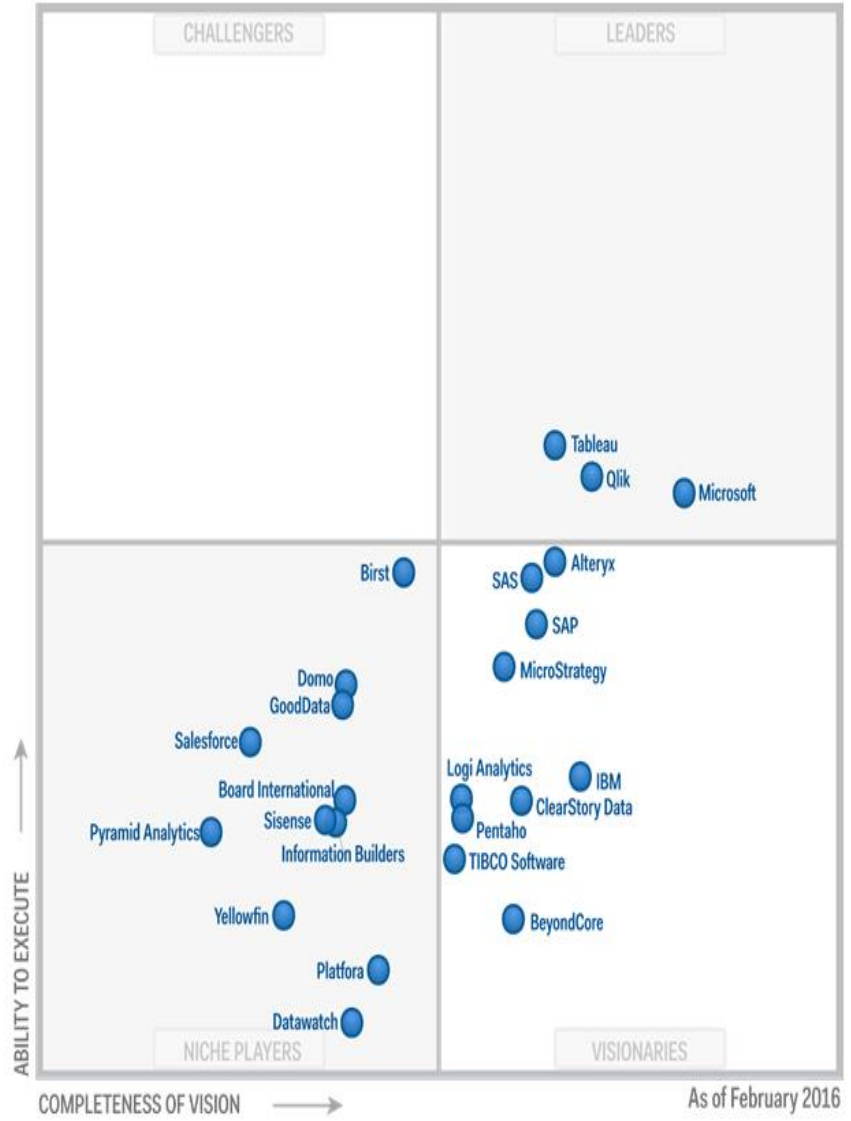
http://www2.cs.uh.edu/~gnawali/courses/cosc6397-f13

Summary of Principles

- Improve vision
  1. Reduced clutter, Make data stand out
  2. Use visually prominent graphical elements
  3. Use proper scale lines and a data rectangle
  4. Reference lines, labels, notes, and keys
  5. Superposed data set

- Improve understanding
  1. Provide explanations and draw conclusions
  2. Use all available space
  3. Align juxtaposed plots
  4. Use log scales when appropriate
  5. Bank to 45^o

Data Visualization

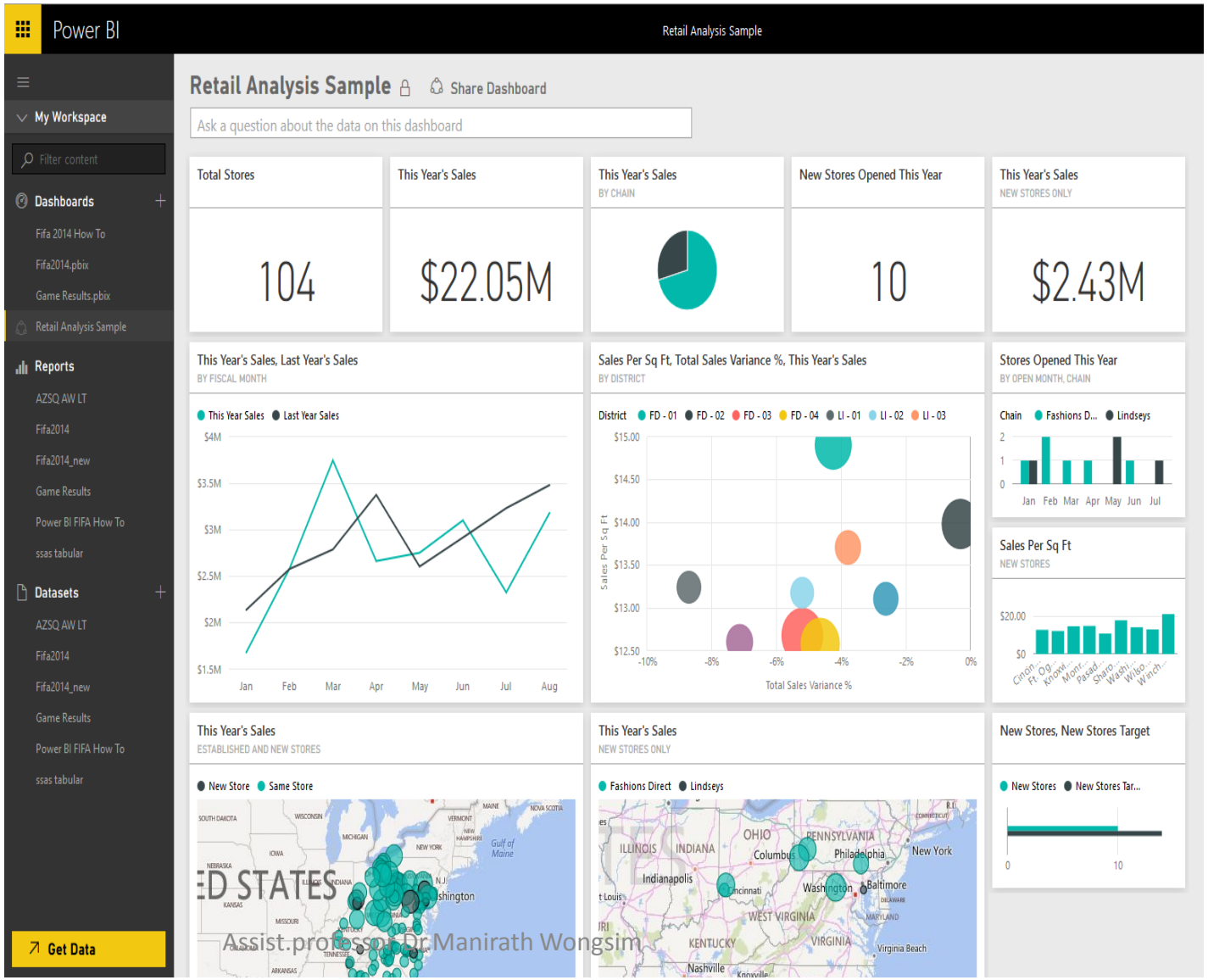


http://optimalbi.com/blog/2017/02/17/gartner-magic-quadrant-for-business-intelligence-2017-cloud-is-coming-slowly/

Data Visualization

Microsoft | Power BI

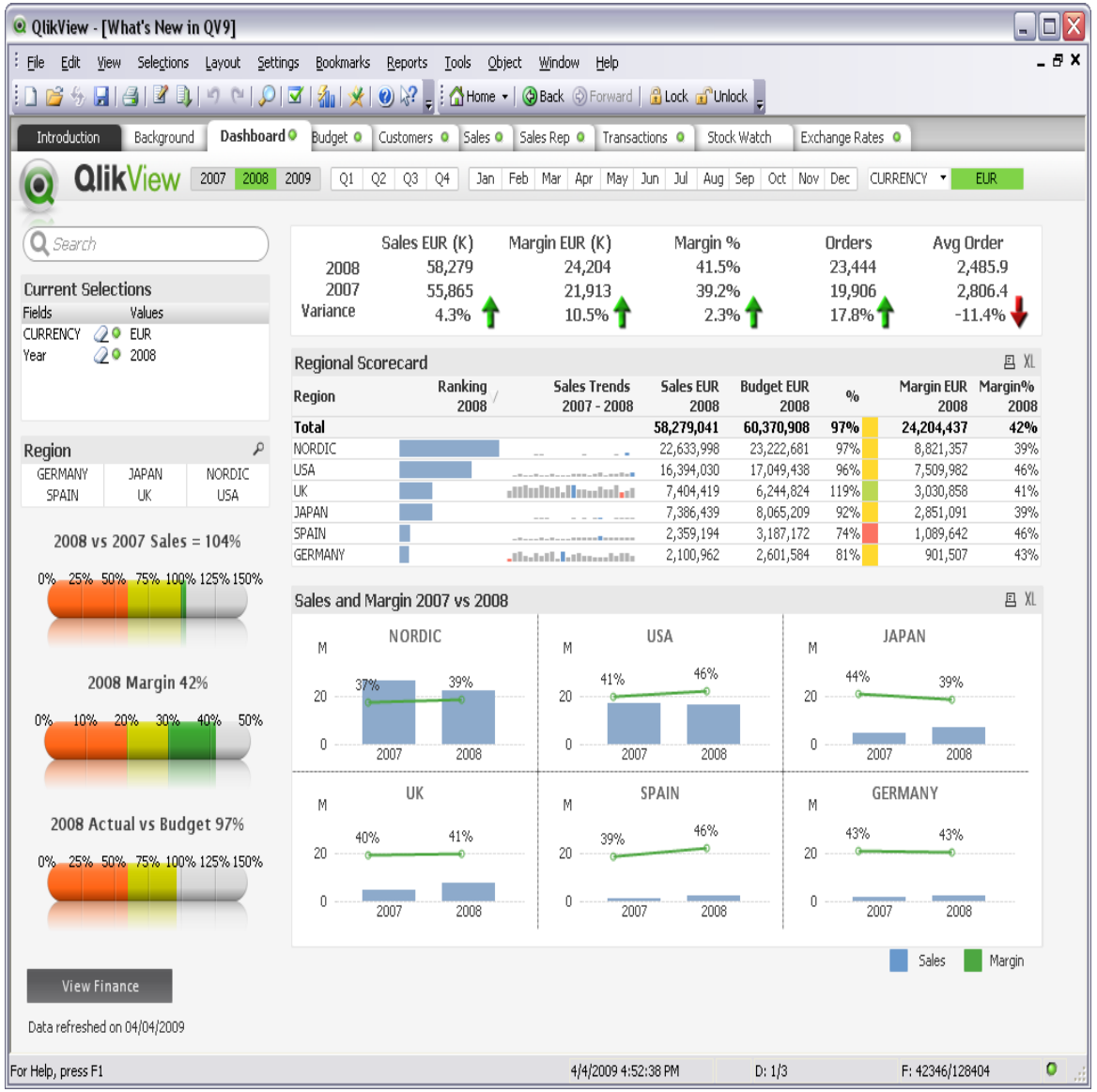
Power BI



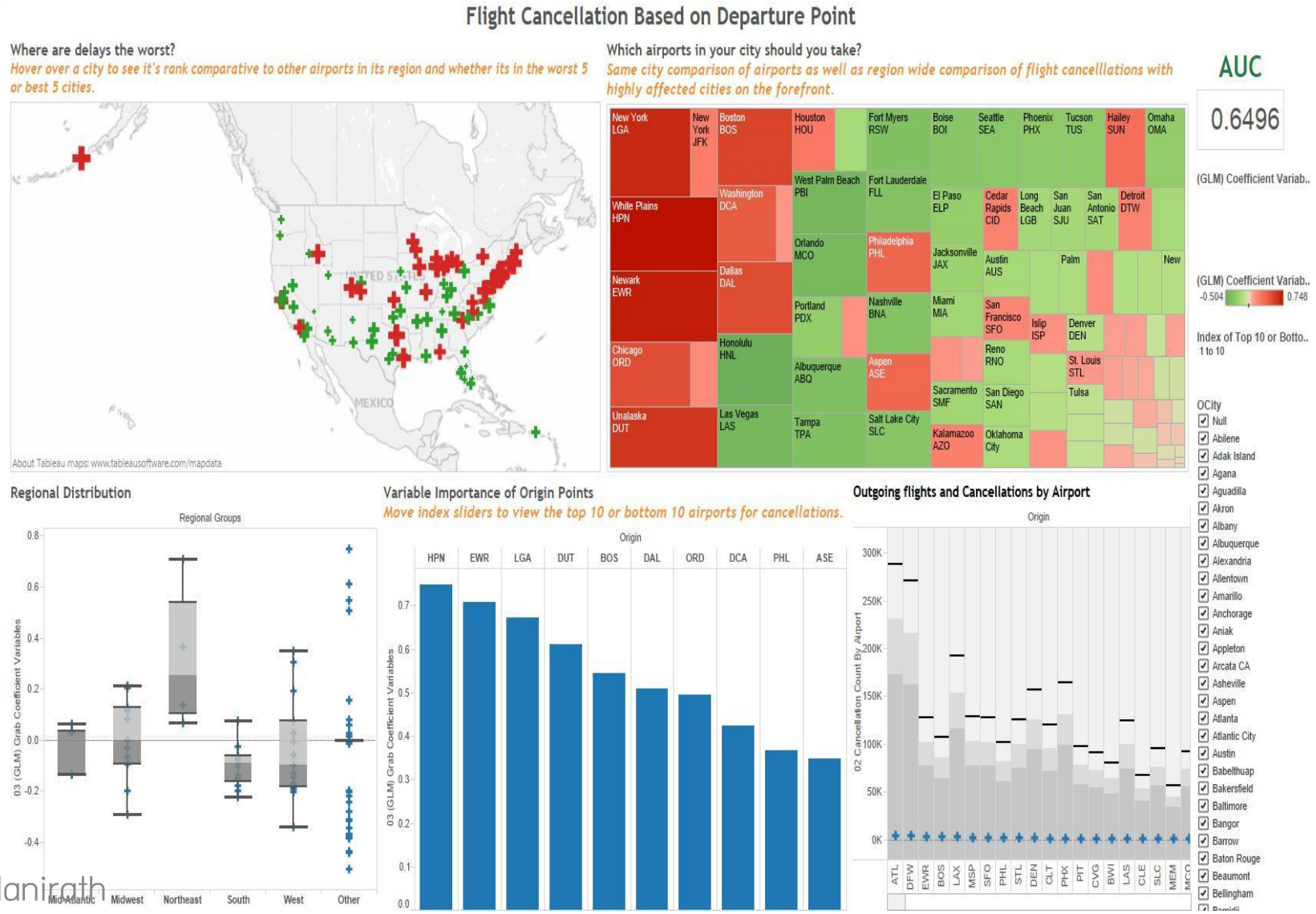
Data Visualization

Author	Share and collaborate	Scale large deployments
Power BI Desktop	Power BI Pro	Power BI Premium
Free	$9.99 per user per month	Capacity pricing per node per month
DOWNLOAD FREE >	TRY FREE >	PLAN YOUR COSTS >
Connect to hundreds of data sources	Build dashboards that deliver a 360-degree, real-time view of the business	Gain dedicated capacity you allocate, scale, and control
Clean and prepare data using visual tools	Keep data up-to-date automatically, including on-premises sources	Distribute and embed content without purchasing per-user licenses
Analyze and build stunning reports with custom visualizations	Collaborate on shared data	Publish reports on-premises with Power BI Report Server
Publish to the Power BI service	Audit and govern how data is accessed and used	Unlock more capacity and higher limits for your Pro users
Embed in public websites		

Data Visualization



Data Visualization



Data Visualization

Resources: Datasets

- UCI Repository: http://www.ics.uci.edu/~mlearn/MLRepository.html
- UCI KDD Archive: http://kdd.ics.uci.edu/summary.data.application.html
- Statlib: http://lib.stat.cmu.edu/
- Delve: http://www.cs.utoronto.ca/~delve/

- #This will launch R interpreter and you will get a prompt > where you can start typing your program as follows –
- myString <- "Hello, World!"
- myString <- "Hello, World!" print (myString)
- # My first program in R Programming myString
- <- "Hello, World!"
- print (myString)
- # My first program in R Programming
- if(FALSE) { "This is a demo for multi-line comments and it should be put inside either a single OR double quote" } myString <- "Hello, World!" print (myString) # Create a vector. apple <- c('red','green',"yellow") print(apple) # Get the class of the vector. print(class(apple)) # Create a list. list1 <- list(c(2,5,3),21.3,sin) # Print the list. print(list1)

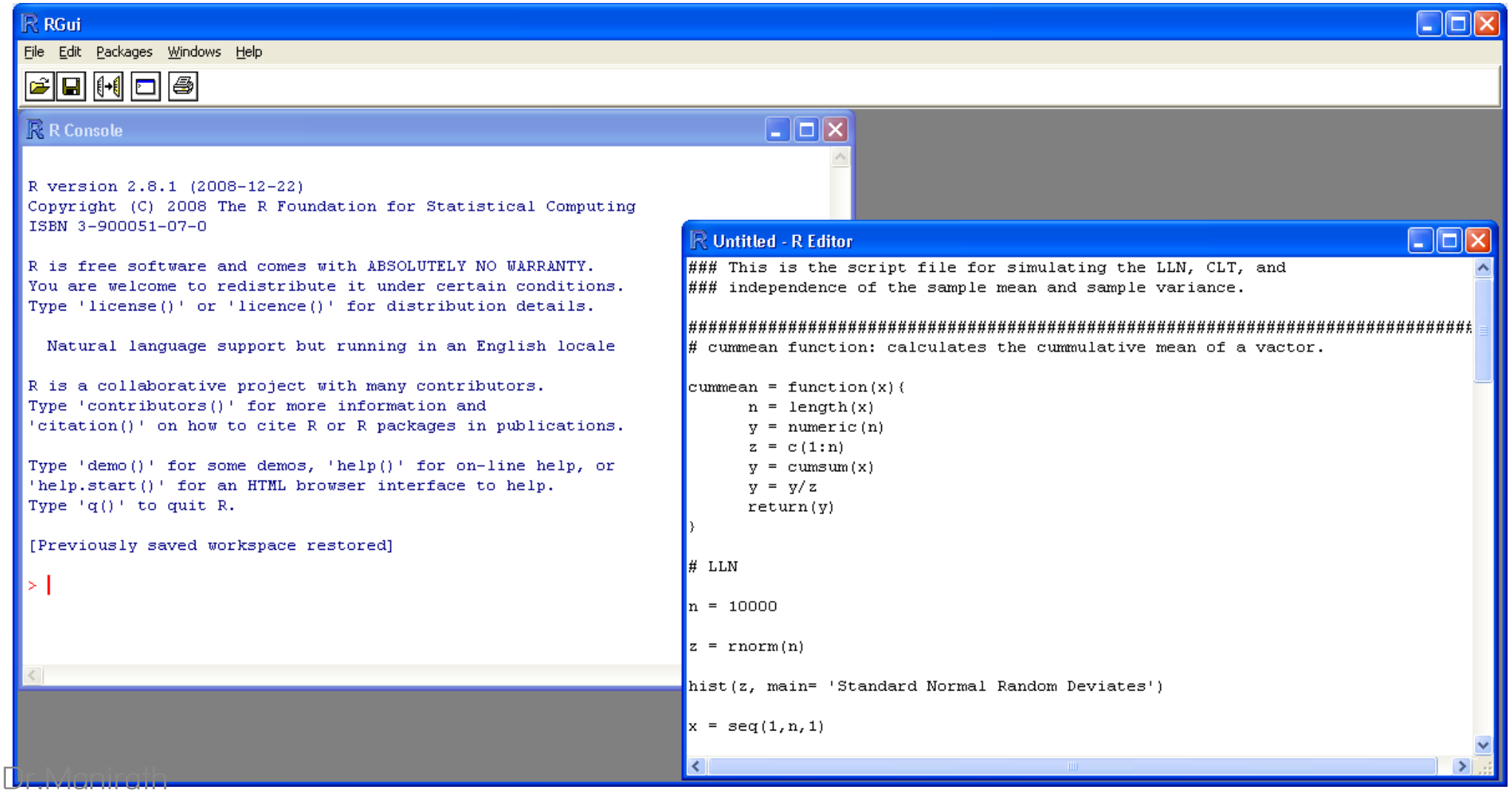
Communication with R

- In my opinion, the R/S language has become the most common language for communication in the fields of Statistics and and Data Analysis.
- Books are being written now with R presented directly placed within the text.
- SV use R, for example
- Excellent for teaching.

R Software

- To download R
- http://www.r-project.org/
- CRAN
- Manuals
- The R Journal
- Books

R Software


RGui
File Edit Packages Windows Help
R Console
R version 2.8.1 (2008-12-22)
Copyright (C) 2008 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
Natural language support but running in an English locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
[Previously saved workspace restored]
>
Untitled - R Editor
This is the script file for simulating the LLN, CLT, and
independence of the sample mean and sample variance.
# cummean function: calculates the cummulative mean of a vactor.
cummean = function(x){
n = length(x)
y = numeric(n)
z = c(1:n)
y = cumsum(x)
y = y/z
return(y)
}
# LLN
n = 10000
z = rnorm(n)
hist(z, main= 'Standard Normal Random Deviates')
x = seq(1,n,1)

R Interfaces

- RWinEdt
- Tinn-R
- JGR (Java Gui for R)
- Emacs + ESS
- Rattle
- AKward
- Playwith (for graphics)

R code

```
> 2+2
[1] 4
> 2+2^2
[1] 6
> (2+2)^2
[1] 16
```

```
> sqrt(2)
[1] 1.414214
> log(2)
[1] 0.6931472
> x = 5
> y = 10
> z <- x+y
> z
[1] 15
```

How To Set Up R on Ubuntu

R packages for Ubuntu on i386 and amd64 are available for all stable Desktop releases of Ubuntu until their official end of life date. However, only the latest Long Term Support (LTS) release is fully supported. As of May 3, 2017 the supported releases are Zesty Zapus (17.04), Yakkety Yak (16.10), Xenial Xerus (16.04; LTS), and Trusty Tahr (14.04; LTS). R 3.3.0 and above is not supported for Precise Pangolin (12.04; LTS), but previous builds will remain here until end of life for 12.04.

See https://wiki.ubuntu.com/Releases for details.

Installation

To obtain the latest R packages, add an entry like

deb https://<my.favorite.cran.mirror>/bin/linux/ubuntu zesty/

or

deb https://<my.favorite.cran.mirror>/bin/linux/ubuntu yakkety/

or

deb https://<my.favorite.cran.mirror>/bin/linux/ubuntu xenial/

or

deb https://<my.favorite.cran.mirror>/bin/linux/ubuntu trusty/

or

deb https://<my.favorite.cran.mirror>/bin/linux/ubuntu precise/

or

in your /etc/apt/sources.list file, replacing <my.favorite.cran.mirror> by the actual URL of your favorite CRAN mirror. See https://cran.r-project.org/mirrors.html for the list of CRAN mirrors. To install the complete R system, use

```
sudo apt-get update

sudo apt-get install r-base
```

Users who need to compile R packages from source [e.g. package maintainers, or anyone installing packages with install.packages()] should also install the r-base-dev package:

```
sudo apt-get install r-base-dev
```

The R packages for Ubuntu otherwise behave like the Debian ones. One may find additional information in the Debian README file located at https://cran.R-project.org/bin/linux/debian/.

Installation and compilation of R or some of its packages may require Ubuntu packages from the "backports" repositories. Therefore, it is suggested to activate the backports repositories with an entry like

```
deb https://<my.favorite.ubuntu.mirror>/ trusty-backports main restricted universe
```

in your /etc/apt/sources.list file. See https://launchpad.net/ubuntu/+archivemirrors for the list of Ubuntu mirrors.

Supported Packages

sudo apt-get build-dep r-cran-foo

Secure APT

The Ubuntu archives on CRAN are signed with the key of "Michael Rutter <marutter@gmail.com>" with key ID E084DAB9. To add the key to your system with one command use (thanks to Brett Presnell for the tip):

sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys E084DAB9

An alternate method can be used by retrieving the key with

gpg --keyserver keyserver.ubuntu.com --recv-key E084DAB9

and then feed it to apt-key with

gpg -a --export E084DAB9 | sudo apt-key add -

Some people have reported difficulties using this approach. The issue is usually related to a firewall blocking port 11371. If the first gpg command fails, you may want to try (thanks to Mischan Toosarani for the tip):

gpg --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys E084DAB9

and then feed it to apt-key with

gpg -a --export E084DAB9 | sudo apt-key add -

Another alternative approach is to search for the key at http://keyserver.ubuntu.com:11371/ and copy the key to a plain text file, say key.txt. Then, feed the key to apt-key with

sudo apt-key add key.txt

Administration and Maintances of R Packages

The R packages part of the Ubuntu r-base and r-recommended packages are installed into the directory /usr/lib/R/library. These can be updated using apt-get with

sudo apt-get update

sudo apt-get upgrade

The other r-cran-* packages shipped with Ubuntu are installed into the directory /usr/lib/R/site-library.

Installing R packages not provided with Ubuntu first requires tools to compile the packages from source. These tools are installed via the R development package with

sudo apt-get install r-base-dev

Then a site administrator can install R packages into the directory /usr/local/lib/R/site-library by running R as root and using the

> install.packages()

R Command Prompt

Once you have R environment setup, then it's easy to start your R command prompt by just typing the following command at your command prompt −

```
$ R
```

This will launch R interpreter and you will get a prompt > where you can start typing your program as follows −

```
> myString <- "Hello, World!"
> print ( myString)
[1] "Hello, World!"
```

Here first statement defines a string variable myString, where we assign a string "Hello, World!" and then next statement print() is being used to print the value stored in variable myString.

R Script File

Usually, you will do your programming by writing your programs in script files and then you execute those scripts at your command prompt with the help of R interpreter called **Rscript**. So let's start with writing following code in a text file called test.R as under −

```
# My first program in R Programming

myString <- "Hello, World!"

print ( myString)
```

Save the above code in a file test.R and execute it at Linux command prompt as given below. Even if you are using Windows or other system, syntax will remain same.

```
$ Rscript test.R
```

When we run the above program, it produces the following result.

```
[1] "Hello, World!"
```

Comments

Comments are like helping text in your R program and they are ignored by the interpreter while executing your actual program. Single comment is written using # in the beginning of the statement as follows −

```
# My first program in R Programming
```

R does not support multi-line comments but you can perform a trick which is something as follows −

```
if(FALSE) {
    "This is a demo for multi-line comments and it should be put inside either a single
        OR double quote"
}
```

Data Type	Example	Verify
Logical	TRUE, FALSE	v <- TRUE print(class(v)) it produces the following result – [1] "logical"
Numeric	12.3, 5, 999	v <- 23.5print(class(v)) it produces the following result – [1] "numeric"
Integer	2L, 34L, 0L	v <- 2Lprint(class(v)) it produces the following result – [1] "integer"

Data Type	Example	Verify
Complex	3 + 2i	v <- 2+5iprint(class(v)) it produces the following result – [1] "complex"
Character	'a' , '"good", "TRUE", '23.4'	v <- "TRUE"print(class(v)) it produces the following result – [1] "character"
Raw	"Hello" is stored as 48 65 6c 6c 6f	v <- charToRaw("Hello")print(class(v)) it produces the following result – [1] "raw"

Vectors

When you want to create vector with more than one element, you should use **c()** function which means to combine the elements into a vector.

```
# Create a vector.

apple <- c('red','green',"yellow")print(apple)

# Get the class of the vector.print(class(apple))
```

When we execute the above code, it produces the following result –

```
[1] "red"    "green"  "yellow"[1] "character"
```

Using the c() function

The non-character values are coerced to character type if one of the elements is a character.

The logical and numeric values are converted to characters.

```
s <- c('apple','red',5,TRUE)

print(s)
```

When we execute the above code, it produces the following result –

```
[1] "apple" "red"   "5"     "TRUE"
```

Multiple Elements Vector

Using colon operator with numeric data

Creating a sequence from 5 to 13.v <- 5:13print(v) # Multiple Elements Vector

Using colon operator with numeric data

Creating a sequence from 5 to 13.

v <- 5:13

print(v)

Creating a sequence from 6.6 to 12.6.

v <- 6.6:12.6

print(v)

If the final element specified does not belong to the sequence then it is discarded.

v <- 3.8:11.4

print(v)

When we execute the above code, it produces the following result –

[1] 5 6 7 8 9 10 11 12 13

[1] 6.6 7.6 8.6 9.6 10.6 11.6 12.6

[1] 3.8 4.8 5.8 6.8 7.8 8.8 9.8 10.8

Creating a sequence from 6.6 to 12.6.v <- 6.6:12.6print(v) # If the final element specified does not belong to the sequence then it is discarded.v <- 3.8:11.4print(v)

When we execute the above code, it produces the following result –

[1] 5 6 7 8 9 10 11 12 13[1] 6.6 7.6 8.6 9.6 10.6 11.6 12.6[1] 3.8 4.8 5.8 6.8 7.8 8.8 9.8 10.8

Accessing Vector Elements

Elements of a Vector are accessed using indexing. The [ ] brackets are used for indexing. Indexing starts with position 1. Giving a negative value in the index drops that element from result.TRUE, FALSE or 0 and 1 can also be used for indexing.

```
# Accessing vector elements using position.
t <- c("Sun","Mon","Tue","Wed","Thurs","Fri","Sat")
u <- t[c(2,3,6)]
print(u)
# Accessing vector elements using logical indexing.
v <- t[c(TRUE,FALSE,FALSE,FALSE,FALSE,TRUE,FALSE)]
print(v)
# Accessing vector elements using negative indexing.
x <- t[c(-2,-5)]
print(x)
# Accessing vector elements using 0/1 indexing.
y <- t[c(0,0,0,0,0,0,1)]
print(y)
```

When we execute the above code, it produces the following result −

```
[1] "Mon" "Tue" "Fri"
[1] "Sun" "Fri"
[1] "Sun" "Tue" "Wed" "Fri" "Sat"
[1] "Sun"
```

Vector Manipulation

Vector Manipulation

Vector arithmetic

Two vectors of same length can be added, subtracted, multiplied or divided giving the result as a vector output.

```
# Create two vectors.
v1 <- c(3,8,4,5,0,11)
v2 <- c(4,11,0,8,1,2)
# Vector addition.
add.result <- v1+v2
print(add.result)
# Vector substraction.
sub.result <- v1-v2
print(sub.result)
# Vector multiplication.
multi.result <- v1*v2
print(multi.result)
# Vector division.
divi.result <- v1/v2
print(divi.result)
```

When we execute the above code, it produces the following result –

```
[1]  7 19  4 13  1 13
[1] -1 -3  4 -3 -1  9
[1] 12 88  0 40  0 22
[1] 0.7500000 0.7272727       Inf 0.6250000 0.0000000 5.5000000
```

Vector element recycling

- If we apply arithmetic operations to two vectors of unequal length, then the elements of the shorter vector are recycled to complete the operations.
- v1 <- c(3,8,4,5,0,11)
- v2 <- c(4,11)
- # V2 becomes c(4,11,4,11,4,11)
- add.result <- v1+v2
- print(add.result)
- sub.result <- v1-v2
- print(sub.result)
- When we execute the above code, it produces the following result –
- [1] 7 19 8 16 4 22
- [1] -1 -3 0 -6 -4 0

Assist.professor Dr.Manirath Wongsim

Vector Element Sorting

Elements in a vector can be sorted using the sort() function.

```
v <- c(3,8,4,5,0,11, -9, 304)
# Sort the elements of the vector.
sort.result <- sort(v)
print(sort.result)
# Sort the elements in the reverse order.
revsort.result <- sort(v, decreasing = TRUE)
print(revsort.result)
# Sorting character vectors.
v <- c("Red","Blue","yellow","violet")
sort.result <- sort(v)
print(sort.result)
# Sorting character vectors in reverse order.
revsort.result <- sort(v, decreasing = TRUE)
print(revsort.result)
```

When we execute the above code, it produces the following result –

```
[1]  -9   0   3   4   5   8  11 304
[1] 304  11   8   5   4   3   0  -9
[1] "Blue"   "Red"    "violet" "yellow"
[1] "yellow" "violet" "Red"    "Blue"
```

Lists

A list is an R-object which can contain many different types of elements inside it like vectors, functions and even another list inside it.

```
# Create a list containing a vector, a matrix and a list.
list_data <- list(c("Jan","Feb","Mar"), matrix(c(3,9,5,1,-2,8), nrow = 2),
  list("green",12.3))

# Give names to the elements in the list.
names(list_data) <- c("1st Quarter", "A_Matrix", "A Inner list")

# Show the list.
print(list_data)
```

Lists

```
# Create a list containing a vector, a matrix and a list.
list_data <- list(c("Jan","Feb","Mar"), matrix(c(3,9,5,1,-2,8), nrow = 2),
   list("green",12.3))

# Give names to the elements in the list.
names(list_data) <- c("1st Quarter", "A_Matrix", "A Inner list")

# Access the first element of the list.
print(list_data[1])

# Access the thrid element. As it is also a list, all its elements will be printed.
print(list_data[3])

# Access the list element using the name of the element.
print(list_data$A_Matrix)
```

Matrices

A matrix is a two-dimensional rectangular data set. It can be created using a vector input to the matrix function.

Create a matrix.M = matrix(c('a','a','b','c','b','a'), nrow = 2, ncol = 3, byrow = TRUE)print(M)

When we execute the above code, it produces the following result –

[,1] [,2] [,3]

[1,] "a" "a" "b"

[2,] "c" "b" "a"

Syntax

The basic syntax for creating a matrix in R is –

matrix(data, nrow, ncol, byrow, dimnames)

Following is the description of the parameters used –

- data is the input vector which becomes the data elements of the matrix.
- nrow is the number of rows to be created.
- ncol is the number of columns to be created.
- byrow is a logical clue. If TRUE then the input vector elements are arranged by row.
- dimname is the names assigned to the rows and columns.

Create a matrix taking a vector of numbers as input

```
# Elements are arranged sequentially by row.
M <- matrix(c(3:14), nrow = 4, byrow = TRUE)
print(M)

# Elements are arranged sequentially by column.
N <- matrix(c(3:14), nrow = 4, byrow = FALSE)
print(N)

# Define the column and row names.
rownames = c("row1", "row2", "row3", "row4")
colnames = c("col1", "col2", "col3")

P <- matrix(c(3:14), nrow = 4, byrow = TRUE, dimnames = list(rownames, colnames))
print(P)
```

When we execute the above code, it produces the following result −

```
     [,1] [,2] [,3]
[1,]    3    4    5
[2,]    6    7    8
[3,]    9   10   11
[4,]   12   13   14
     [,1] [,2] [,3]
[1,]    3    7   11
[2,]    4    8   12
[3,]    5    9   13
[4,]    6   10   14
```

Accessing Elements of a Matrix

Elements of a matrix can be accessed by using the column and row index of the element. We consider the matrix P above to find the specific elements below.

```
# Define the column and row names.
rownames = c("row1", "row2", "row3", "row4")
colnames = c("col1", "col2", "col3")

# Create the matrix.
P <- matrix(c(3:14), nrow = 4, byrow = TRUE, dimnames = list(rownames, colnames))
# Access the element at 3rd column and 1st row.
print(P[1,3])
# Access the element at 2nd column and 4th row.
print(P[4,2])
# Access only the  2nd row.
print(P[2,])
# Access only the 3rd column.
print(P[,3])
```

When we execute the above code, it produces the following result −

```
[1] 5
[1] 13
col1 col2 col3
   6    7    8
row1 row2 row3 row4
   5    8   11   14
```

Matrix Computations

Various mathematical operations are performed on the matrices using the R operators. The result of the operation is also a matrix.

The dimensions (number of rows and columns) should be same for the matrices involved in the operation.

Matrix Addition & Subtraction

```
# Create two 2x3 matrices.
matrix1 <- matrix(c(3, 9, -1, 4, 2, 6), nrow = 2)
print(matrix1)
matrix2 <- matrix(c(5, 2, 0, 9, 3, 4), nrow = 2)
print(matrix2)
# Add the matrices.
result <- matrix1 + matrix2
cat("Result of addition","\n")
print(result)
# Subtract the matrices
result <- matrix1 - matrix2
cat("Result of subtraction","\n")
print(result)
```

When we execute the above code, it produces the following result –

```
     [,1] [,2] [,3]
[1,]    3   -1    2
[2,]    9    4    6
     [,1] [,2] [,3]
[1,]    5    0    3
[2,]    2    9    4
Result of addition
     [,1] [,2] [,3]
[1,]    8   -1    5
[2,]   11   13   10
Result of subtraction
     [,1] [,2] [,3]
[1,]   -2   -1   -1
[2,]    7   -5    2
```

Matrix Multiplication & Division

```
# Create two 2x3 matrices.
matrix1 <- matrix(c(3, 9, -1, 4, 2, 6), nrow = 2)
print(matrix1)
matrix2 <- matrix(c(5, 2, 0, 9, 3, 4), nrow = 2)
print(matrix2)
# Multiply the matrices.
result <- matrix1 * matrix2
cat("Result of multiplication","\n")
print(result)
# Divide the matrices
result <- matrix1 / matrix2
cat("Result of division","\n")
print(result)
```

When we execute the above code, it produces the following result −

```
     [,1] [,2] [,3]
[1,]    3   -1    2
[2,]    9    4    6
     [,1] [,2] [,3]
[1,]    5    0    3
[2,]    2    9    4
Result of multiplication
     [,1] [,2] [,3]
[1,]   15    0    6
[2,]   18   36   24
Result of division
     [,1]      [,2]      [,3]
[1,]  0.6      -Inf 0.6666667
[2,]  4.5 0.4444444 1.5000000
```

Arrays

While matrices are confined to two dimensions, arrays can be of any number of dimensions. The array function takes a dim attribute which creates the required number of dimension. In the below example we create an array with two elements which are 3x3 matrices each.

```
# Create an array.
a <- array(c('green','yellow'),dim = c(3,3,2))
print(a)
```

When we execute the above code, it produces the following result –

```
, , 1

     [,1]     [,2]     [,3]
[1,] "green"  "yellow" "green"
[2,] "yellow" "green"  "yellow"
[3,] "green"  "yellow" "green"

, , 2

     [,1]     [,2]     [,3]
[1,] "yellow" "green"  "yellow"
[2,] "green"  "yellow" "green"
[3,] "yellow" "green"  "yellow"
```

Arrays are the R data objects which can store data in more than two dimensions. For example – If we create an array of dimension (2, 3, 4) then it creates 4 rectangular matrices each with 2 rows and 3 columns. Arrays can store only data type.

An array is created using the array() function. It takes vectors as input and uses the values in the dim parameter to create an array.

Example

The following example creates an array of two 3x3 matrices each with 3 rows and 3 columns.

```
# Create two vectors of different lengths.

vector1 <- c(5,9,3)

vector2 <- c(10,11,12,13,14,15)

# Take these vectors as input to the array.

result <- array(c(vector1,vector2),dim = c(3,3,2))

print(result)
```

Naming Columns and Rows

We can give names to the rows, columns and matrices in the array by using the dimnames parameter.

```
# Create two vectors of different lengths.
vector1 <- c(5,9,3)
vector2 <- c(10,11,12,13,14,15)
column.names <- c("COL1","COL2","COL3")
row.names <- c("ROW1","ROW2","ROW3")
matrix.names <- c("Matrix1","Matrix2")
# Take these vectors as input to the array.
result <- array(c(vector1,vector2),dim = c(3,3,2),dimnames = list(row.names,column.names,
   matrix.names))
print(result)
```

When we execute the above code, it produces the following result –

```
, , Matrix1

     COL1 COL2 COL3
ROW1    5   10   13
ROW2    9   11   14
ROW3    3   12   15

, , Matrix2

     COL1 COL2 COL3
ROW1    5   10   13
ROW2    9   11   14
ROW3    3   12   15
```

Accessing Array Elements

```
# Create two vectors of different lengths.
vector1 <- c(5,9,3)
vector2 <- c(10,11,12,13,14,15)
column.names <- c("COL1","COL2","COL3")
row.names <- c("ROW1","ROW2","ROW3")
matrix.names <- c("Matrix1","Matrix2")

# Take these vectors as input to the array.
result <- array(c(vector1,vector2),dim = c(3,3,2),dimnames = list(row.names,
    column.names, matrix.names))

# Print the third row of the second matrix of the array.
print(result[3,,2])

# Print the element in the 1st row and 3rd column of the 1st matrix.
print(result[1,3,1])

# Print the 2nd Matrix.
print(result[,,2])
```

Manipulating Array Elements

- As array is made up matrices in multiple dimensions, the operations on elements of array are carried out by accessing elements of the matrices.
- # Create two vectors of different lengths.
- vector1 <- c(5,9,3)
- vector2 <- c(10,11,12,13,14,15)

- # Take these vectors as input to the array.
- array1 <- array(c(vector1,vector2),dim = c(3,3,2))

- # Create two vectors of different lengths.
- vector3 <- c(9,1,0)
- vector4 <- c(6,0,11,3,14,1,2,6,9)
- array2 <- array(c(vector1,vector2),dim = c(3,3,2))

- # create matrices from these arrays.
- matrix1 <- array1[,,2]
- matrix2 <- array2[,,2]

- # Add the matrices.
- result <- matrix1+matrix2
- print(result)

Factors

Factors are the r-objects which are created using a vector. It stores the vector along with the distinct values of the elements in the vector as labels. The labels are always character irrespective of whether it is numeric or character or Boolean etc. in the input vector. They are useful in statistical modeling.

Factors are created using the factor() function.The nlevels functions gives the count of levels.

```
# Create a vector.
apple_colors <- c('green','green','yellow','red','red','red','green')

# Create a factor object.
factor_apple <- factor(apple_colors)

# Print the factor.
print(factor_apple)
print(nlevels(factor_apple))
```

When we execute the above code, it produces the following result –

```
[1] green  green  yellow red    red    red    green
Levels: green red yellow
# applying the nlevels function we can know the number of distinct values
[1] 3
```

Data Frames

first column can be numeric while the second column can be character and third column can be logical. It is a list of vectors of equal length.

Data Frames are created using the data.frame() function.

```
# Create the data frame.
BMI <-   data.frame(
    gender = c("Male", "Male","Female"),
    height = c(152, 171.5, 165),
    weight = c(81,93, 78),
    Age = c(42,38,26)
)
print(BMI)
```

When we execute the above code, it produces the following result –

```
  gender height weight Age
1   Male  152.0     81  42
2   Male  171.5     93  38
3 Female  165.0     78  26
```

Factors in Data Frame

On creating any data frame with a column of text data, R treats the text column as categorical data and creates factors on it.

```
# Create the vectors for data frame.
height <- c(132,151,162,139,166,147,122)
weight <- c(48,49,66,53,67,52,40)
gender <- c("male","male","female","female","male","female","male")

# Create the data frame.
input_data <- data.frame(height,weight,gender)
print(input_data)

# Test if the gender column is a factor.
print(is.factor(input_data$gender))

# Print the gender column so see the levels.
print(input_data$gender)
```

Changing the Order of Levels

The order of the levels in a factor can be changed by applying the factor function again with new order of the levels.

```
data <-
c("East","West","East","North","North","East","West","West","West","East","North")

# Create the factors

factor_data <- factor(data)

print(factor_data)


# Apply the factor function with required order of the level.

new_order_data <- factor(factor_data,levels =
c("East","West","North"))

print(new_order_data)
```

Generating Factor Levels

We can generate factor levels by using the gl() function. It takes two integers as input which indicates how many levels and how many times each level.

Syntax

gl(n, k, labels)

Following is the description of the parameters used −

- n is a integer giving the number of levels.
- k is a integer giving the number of replications.
- labels is a vector of labels for the resulting factor levels.

Example

```
v <- gl(3, 4, labels = c("Tampa", "Seattle","Boston"))
print(v)
```

When we execute the above code, it produces the following result −

```
Tampa   Tampa   Tampa   Tampa   Seattle Seattle Seattle Seattle Boston
[10] Boston  Boston  Boston
Levels: Tampa Seattle Boston
```

Create Data Frame

```
# Create the data frame.
emp.data <- data.frame(
   emp_id = c (1:5),
   emp_name = c("Rick","Dan","Michelle","Ryan","Gary"),
   salary = c(623.3,515.2,611.0,729.0,843.25),

   start_date = as.Date(c("2012-01-01", "2013-09-23", "2014-11-15", "2014-05-11",
      "2015-03-27")),
   stringsAsFactors = FALSE
)
# Print the data frame.
print(emp.data)
```

When we execute the above code, it produces the following result –

	emp_id	emp_name	salary	start_date
1	1	Rick	623.30	2012-01-01
2	2	Dan	515.20	2013-09-23
3	3	Michelle	611.00	2014-11-15
4	4	Ryan	729.00	2014-05-11
5	5	Gary	843.25	2015-03-27

Get the Structure of the Data Frame

The structure of the data frame can be seen by using str() function.

```
# Create the data frame.
emp.data <- data.frame(
   emp_id = c (1:5),
   emp_name = c("Rick","Dan","Michelle","Ryan","Gary"),
   salary = c(623.3,515.2,611.0,729.0,843.25),

   start_date = as.Date(c("2012-01-01", "2013-09-23", "2014-11-15", "2014-05-11",
      "2015-03-27")),
   stringsAsFactors = FALSE
)
# Get the structure of the data frame.
str(emp.data)
```

Summary of Data in Data Frame

The statistical summary and nature of the data can be obtained by applying summary() function.

```
# Create the data frame.
emp.data <- data.frame(
    emp_id = c (1:5),
    emp_name = c("Rick","Dan","Michelle","Ryan","Gary"),
    salary = c(623.3,515.2,611.0,729.0,843.25),

    start_date = as.Date(c("2012-01-01", "2013-09-23", "2014-11-15", "2014-05-11",
        "2015-03-27")),
    stringsAsFactors = FALSE
)
# Print the summary.
print(summary(emp.data))
```

Extract Data from Data Frame

Extract specific column from a data frame using column name.

```
# Create the data frame.
emp.data <- data.frame(
    emp_id = c (1:5),
    emp_name = c("Rick","Dan","Michelle","Ryan","Gary"),
    salary = c(623.3,515.2,611.0,729.0,843.25),

    start_date = as.Date(c("2012-01-01","2013-09-23","2014-11-15","2014-05-11",
        "2015-03-27")),
    stringsAsFactors = FALSE
)
# Extract Specific columns.
result <- data.frame(emp.data$emp_name,emp.data$salary)
print(result)
```

Extract the first two rows and then all columns

```
# Create the data frame.
emp.data <- data.frame(
   emp_id = c (1:5),
   emp_name = c("Rick","Dan","Michelle","Ryan","Gary"),
   salary = c(623.3,515.2,611.0,729.0,843.25),
   start_date = as.Date(c("2012-01-01", "2013-09-23", "2014-11-15", "2014-05-11",
      "2015-03-27")),
   stringsAsFactors = FALSE
)
# Extract first two rows.
result <- emp.data[1:2,]
print(result)
```

Expand Data Frame

A data frame can be expanded by adding columns and rows.

Add Column

Just add the column vector using a new column name.

```
# Create the data frame.
emp.data <- data.frame(
    emp_id = c (1:5),
    emp_name = c("Rick","Dan","Michelle","Ryan","Gary"),
    salary = c(623.3,515.2,611.0,729.0,843.25),

    start_date = as.Date(c("2012-01-01", "2013-09-23", "2014-11-15", "2014-05-11",
        "2015-03-27")),
    stringsAsFactors = FALSE
)

# Add the "dept" coulmn.
emp.data$dept <- c("IT","Operations","IT","HR","Finance")
v <- emp.data
print(v)
```

Variable Assignment

The variables can be assigned values using leftward, rightward and equal to operator. The values of the variables can be printed using print() or cat()function. The cat() function combines multiple items into a continuous print output.

```
# Assignment using equal operator.
var.1 = c(0,1,2,3)

# Assignment using leftward operator.
var.2 <- c("learn","R")

# Assignment using rightward operator.
c(TRUE,1) -> var.3

print(var.1)
cat ("var.1 is ", var.1 ,"\n")
cat ("var.2 is ", var.2 ,"\n")
cat ("var.3 is ", var.3 ,"\n")
```

When we execute the above code, it produces the following result −

```
[1] 0 1 2 3
var.1 is  0 1 2 3
var.2 is  learn R
var.3 is  1 1
```

Data Type of a Variable

In R, a variable itself is not declared of any data type, rather it gets the data type of the R - object assigned to it. So R is called a dynamically typed language, which means that we can change a variable's data type of the same variable again and again when using it in a program.

```
var_x <- "Hello"
cat("The class of var_x is ",class(var_x),"\n")

var_x <- 34.5
cat("  Now the class of var_x is ",class(var_x),"\n")

var_x <- 27L
cat("   Next the class of var_x becomes ",class(var_x),"\n")
```

When we execute the above code, it produces the following result –

```
The class of var_x is  character
   Now the class of var_x is  numeric
      Next the class of var_x becomes  integer
```

Finding Variables

- To know all the variables currently available in the workspace we use the ls() function. Also the ls() function can use patterns to match the variable names.
- print(ls())
- When we execute the above code, it produces the following result –
- [1] "my var" "my_new_var" "my_var" "var.1"
- [5] "var.2" "var.3" "var.name" "var_name2."
- [9] "var_x" "varname"
- Note – It is a sample output depending on what variables are declared in your environment.
- The ls() function can use patterns to match the variable names.
- # List the variables starting with the pattern "var".
- print(ls(pattern = "var"))
- When we execute the above code, it produces the following result –
- [1] "my var" "my_new_var" "my_var" "var.1"
- [5] "var.2" "var.3" "var.name" "var_name2."
- [9] "var_x" "varname"

The variables starting with dot(.) are hidden, they can be listed using "all.names = TRUE" argument to ls() function.

```
print(ls(all.name = TRUE))
```

When we execute the above code, it produces the following result –

```
[1] ".cars"          ".Random.seed" ".var_name"     ".varname"      ".varname2"
[6] "my var"         "my_new_var"   "my_var"        "var.1"         "var.2"
[11]"var.3"          "var.name"     "var_name2."    "var_x"
```

Deleting Variables

Variables can be deleted by using the rm() function. Below we delete the variable var.3. On printing the value of the variable error is thrown.

```
rm(var.3)

print(var.3)
```

When we execute the above code, it produces the following result –

```
[1] "var.3"

Error in print(var.3) : object 'var.3' not found
```

All the variables can be deleted by using the rm() and ls() function together.

```
rm(list = ls())

print(ls())
```

When we execute the above code, it produces the following result –

```
character(0)
```

Operator	Description	Example
+	Adds two vectors	v <- c(2,5.5,6)t <- c(8, 3, 4)print(v+t) it produces the following result – [1] 10.0 8.5 10.0
–	Subtracts second vector from the first	v <- c(2,5.5,6)t <- c(8, 3, 4)print(v-t) it produces the following result – [1] -6.0 2.5 2.0
*	Multiplies both vectors	v <- c(2,5.5,6)t <- c(8, 3, 4)print(v*t) it produces the following result – [1] 16.0 16.5 24.0
/	Divide the first vector with the second	v <- c(2,5.5,6)t <- c(8, 3, 4)print(v/t) When we execute the above code, it produces the following result – [1] 0.250000 1.833333 1.500000

Operator	Description	Example
%%	Give the remainder of the first vector with the second	v <- c(2,5.5,6)t <- c(8, 3, 4)print(v%%t) it produces the following result – [1] 2.0 2.5 2.0
%/%	The result of division of first vector with second (quotient)	v <- c(2,5.5,6)t <- c(8, 3, 4)print(v%/%t) it produces the following result – [1] 0 1 1
^	The first vector raised to the exponent of second vector	v <- c(2,5.5,6)t <- c(8, 3, 4)print(v^t) it produces the following result – [1] 256.000 166.375 1296.000

Decision making structures

Sr.No.	Statement & Description
1	if statement An if statement consists of a Boolean expression followed by one or more statements.
2	if...else statement An if statement can be followed by an optional else statement, which executes when the Boolean expression is false.
3	switch statement A switch statement allows a variable to be tested for equality against a list of values.

Built-in Function

- Simple examples of in-built functions are seq(), mean(), max(), sum(x) and paste(...) etc. They are directly called by user written programs. You can refer most widely used R functions.

```
# Create a function to print squares of numbers in sequence.
new.function <- function(a) {
   for(i in 1:a) {
      b <- i^2
      print(b)
   } }
```

```
new.function <- function(a) {
    for(i in 1:a) {
        b <- i^2
        print(b)
    }
}
```

```
# Call the function new.function supplying 6 as an argument.
new.function(6)
```

User-defined Function

We can create user-defined functions in R. They are specific to what a user wants and once created they can be used like the built-in functions. Below is an example of how a function is created and used.

```
# Create a function to print squares of numbers in sequence.
new.function <- function(a) {
  for(i in 1:a) {
    b <- i^2
    print(b)
  }
}
```

Calling a Function

```
# Create a function to print squares of numbers in sequence.
new.function <- function(a) {
  for(i in 1:a) {
    b <- i^2
    print(b)
  }
}

# Call the function new.function supplying 6 as an argument.
new.function(6)
```

When we execute the above code, it produces the following result −

```
[1] 1
[1] 4
[1] 9
[1] 16
[1] 25
[1] 36
```

R packages

R packages are a collection of R functions, complied code and sample data. They are stored under a directory called "library" in the R environment. By default, R installs a set of packages during installation. More packages are added later, when they are needed for some specific purpose. When we start the R console, only the default packages are available by default. Other packages which are already installed have to be loaded explicitly to be used by the R program that is going to use them.

All the packages available in R language are listed at R Packages.

Below is a list of commands to be used to check, verify and use the R packages.

Check Available R Packages

- Get library locations containing R packages
- .libPaths()
- When we execute the above code, it produces the following result. It may vary depending on the local settings of your pc.
- [2] "C:/Program Files/R/R-3.2.2/library"
- Get the list of all the packages installed
- library()
- When we execute the above code, it produces the following result. It may vary depending on the local settings of your pc.
- Packages in library 'C:/Program Files/R/R-3.2.2/library':

- base The R Base Package
- boot Bootstrap Functions (Originally by Angelo Canty
- for S)
- class Functions for Classification
- cluster "Finding Groups in Data": Cluster Analysis
- Extended Rousseeuw et al.
- codetools Code Analysis Tools for R
- compiler

Install a New Package

There are two ways to add new R packages. One is installing directly from the CRAN directory and another is downloading the package to your local system and installing it manually.

Install directly from CRAN

The following command gets the packages directly from CRAN webpage and installs the package in the R environment. You may be prompted to choose a nearest mirror. Choose the one appropriate to your location.

```
install.packages("Package Name")


# Install the package named "XML".
install.packages("XML")
```

Install package manually

Go to the link R Packages to download the package needed. Save the package as a .zip file in a suitable location in the local system.

Now you can run the following command to install this package in the R environment.

install.packages(file_name_with_path, repos = NULL, type = "source")

Install the package named "XML"

install.packages("E:/XML_3.98-1.3.zip", repos = NULL, type = "source")

Load Package to Library

Before a package can be used in the code, it must be loaded to the current R environment. You also need to load a package that is already installed previously but not available in the current environment.

A package is loaded using the following command –

library("package Name", lib.loc = "path to library")

Load the package named "XML"

install.packages("E:/XML_3.98-1.3.zip", repos = NULL, type = "source")

Joining Columns and Rows in a Data Frame

We can join multiple vectors to create a data frame using the cbind()function. Also we can merge two data frames using rbind() function.

```
# Create vector objects.
city <- c("Tampa","Seattle","Hartford","Denver")
state <- c("FL","WA","CT","CO")
zipcode <- c(33602,98104,06161,80294)
# Combine above three vectors into one data frame.
addresses <- cbind(city,state,zipcode)
# Print a header.
cat("# # # # The First data frame\n")

# Print the data frame.
print(addresses)
# Create another data frame with similar columns
new.address <- data.frame(
   city = c("Lowry","Charlotte"),
   state = c("CO","FL"),
   zipcode = c("80230","33949"),
   stringsAsFactors = FALSE
)
# Print a header.
cat("# # # The Second data frame\n")

# Print the data frame.
print(new.address)
# Combine rows form both the data frames.
all.addresses <- rbind(addresses,new.address)

# Print a header.
cat("# # # The combined data frame\n")
# Print the result.
print(all.addresses)
```

Merging Data Frames

- We can merge two data frames by using the merge() function. The data frames must have same column names on which the merging happens.
- In the example below, we consider the data sets about Diabetes in Pima Indian Women available in the library names "MASS". we merge the two data sets based on the values of blood pressure("bp") and body mass index("bmi"). On choosing these two columns for merging, the records where values of these two variables match in both data sets are combined together to form a single data frame.

Merging Data Frames

```
library(MASS)

merged.Pima <- merge(x = Pima.te, y = Pima.tr,

    by.x = c("bp", "bmi"),

    by.y = c("bp", "bmi")

)

print(merged.Pima)

nrow(merged.Pima)
```

When we execute the above code, it produces the following result

```
bp  bmi npreg.x glu.x skin.x ped.x age.x type.x npreg.y glu.y skin.y ped.y
1  60 33.8      1   117     23 0.466    27     No      2   125     20 0.088
2  64 29.7      2    75     24 0.370    33     No      2   100     23 0.368
3  64 31.2      5   189     33 0.583    29    Yes      3   158     13 0.295
4  64 33.2      4   117     27 0.230    24     No      1    96     27 0.289
5  66 38.1      3   115     39 0.150    28     No      1   114     36 0.289
6  68 38.5      2   100     25 0.324    26     No      7   129     49 0.439
7  70 27.4      1   116     28 0.204    21     No      0   124     20 0.254
8  70 33.1      4    91     32 0.446    22     No      9   123     44 0.374
9  70 35.4      9   124     33 0.282    34     No      6   134     23 0.542
10 72 25.6      1   157     21 0.123    24     No      4    99     17 0.294
11 72 37.7      5    95     33 0.370    27     No      6   103     32 0.324
12 74 25.9      9   134     33 0.460    81     No      8   126     38 0.162
13 74 25.9      1    95     21 0.673    36     No      8   126     38 0.162
14 78 27.6      5    88     30 0.258    37     No      6   125     31 0.565
15 78 27.6     10   122     31 0.512    45     No      6   125     31 0.565
16 78 39.4      2   112     50 0.175    24     No      4   112     40 0.236
17 88 34.5      1   117     24 0.403    40    Yes      4   127     11 0.598
   age.y type.y
1     31     No
2     21     No
3     24     No
4     21     No
5     21     No
6     43    Yes
7     36    Yes
8     40     No
9     29    Yes
10    28     No
11    55     No
12    39     No
13    39     No
14    49    Yes
15    49    Yes
16    38     No
17    28     No
[1] 17
```

Melting and Casting

One of the most interesting aspects of R programming is about changing the shape of the data in multiple steps to get a desired shape. The functions used to do this are called melt() and cast().

We consider the dataset called ships present in the library called "MASS".

```
library(MASS)

print(ships)
```

Melt the Data

Now we melt the data to organize it, converting all columns other than type and year into multiple rows.

molten.ships <- melt(ships, id = c("type","year"))

print(molten.ships)

- recasted.ship <- cast(molten.ships, type+year~variable,sum)print(recasted.ship)

R - CSV Files

- In R, we can read data from files stored outside the R environment. We can also write data into files which will be stored and accessed by the operating system. R can read and write into various file formats like csv, excel, xml etc.
- In this chapter we will learn to read data from a csv file and then write data into a csv file. The file should be present in current working directory so that R can read it. Of course we can also set our own directory and read files from there.

Getting and Setting the Working Directory

- # Get and print current working directory.
- print(getwd())

- # Set current working directory.
- setwd("/web/com")

- # Get and print current working directory.
- print(getwd())

Input as CSV File

id,name,salary,start_date,dept

1,Rick,623.3,2012-01-01,IT

2,Dan,515.2,2013-09-23,Operations

3,Michelle,611,2014-11-15,IT

4,Ryan,729,2014-05-11,HR

,Gary,843.25,2015-03-27,Finance

6,Nina,578,2013-05-21,IT

7,Simon,632.8,2013-07-30,Operations

8,Guru,722.5,2014-06-17,Finance

Reading a CSV File

- survey <- read.table("survey_data.csv", header=TRUE, sep=",")
- survey <- read.table("survey_mess.csv", header=TRUE, sep=",")

(a) How many survey respondents are from MISM or Other?

```
sum(survey[["Program"]] == "MISM" | survey[["Program"]] == "Other")
```

(b) What % of survey respondents are from PPM?

```
100 * sum(survey[["Program"]] == "PPM") / nrow(survey)
```

2. Index practice

(a) Use $ notation to pull the OperatingSystem column from the survey data

```
survey$OperatingSystem
```

(b) Do the same thing with [,] notation, referring to OperatingSystem by name

```
survey[, "OperatingSystem"]
```

(c) Repeat part (b), this time referring to OperatingSystem by column number

```
survey[, 4]
```

R - Web Data

- Many websites provide data for consumption by its users. For example the World Health Organization(WHO) provides reports on health and medical information in the form of CSV, txt and XML files. Using R programs, we can programmatically extract specific data from such websites. Some packages in R which are used to scrap data form the web are – "RCurl",XML", and "stringr". They are used to connect to the URL's, identify required links for the files and download them to the local environment.

Install R Packages

- The following packages are required for processing the URL's and links to the files. If they are not available in your R Environment, you can install them using following commands.
- install.packages("RCurl")
- install.packages("XML")
- install.packages("stringr")
- install.packages("plyr")

Input Data

- We will visit the URL weather data and download the CSV files using R for the year 2015.
- Example
- We will use the function getHTMLLinks() to gather the URLs of the files. Then we will use the function downlaod.file() to save the files to the local system. As we will be applying the same code again and again for multiple files, we will create a function to be called multiple times. The filenames are passed as parameters in form of a R list object to this function.

Read the URL.

```
# Read the URL.
url <- "http://www.geos.ed.ac.uk/~weather/jcmb_ws/"

# Gather the html links present in the webpage.
links <- getHTMLLinks(url)

# Identify only the links which point to the JCMB 2015 files.
filenames <- links[str_detect(links, "JCMB_2015")]

# Store the file names as a list.
filenames_list <- as.list(filenames)

# Create a function to download the files by passing the URL and filename list.
downloadcsv <- function (mainurl,filename) {
    filedetails <- str_c(mainurl,filename)
    download.file(filedetails,filename)
}

# Now apply the l_ply function and save the files into the current R working directory.
l_ply(filenames,downloadcsv,mainurl = "http://www.geos.ed.ac.uk/~weather/jcmb_ws/")
```

Verify the File Download

- After running the above code, you can locate the following files in the current R working directory.
- "JCMB_2015.csv" "JCMB_2015_Apr.csv" "JCMB_2015_Feb.csv" "JCMB_2015_Jan.csv"
- "JCMB_2015_Mar.csv"

R - Databases

- The data is Relational database systems are stored in a normalized format. So, to carry out statistical computing we will need very advanced and complex Sql queries. But R can connect easily to many relational databases like MySql, Oracle, Sql server etc. and fetch records from them as a data frame. Once the data is available in the R environment, it becomes a normal R data set and can be manipulated or analyzed using all the powerful packages and functions.
- In this tutorial we will be using MySql as our reference database for connecting to R.

R MySQL Package

R has a built-in package named "RMySQL" which provides native connectivity between with MySql database. You can install this package in the R environment using the following command.

install.packages("RMySQL") R MySQL Package

R has a built-in package named "RMySQL" which provides native connectivity between with MySql database. You can install this package in the R environment using the following command.

install.packages("RMySQL")

Connecting R to MySql

Once the package is installed we create a connection object in R to connect to the database. It takes the username, password, database name and host name as input.

```
# Create a connection Object to MySQL database.
# We will connect to the sampel database named "sakila" that comes with MySql installation.
mysqlconnection = dbConnect(MySQL(), user = 'root', password = '', dbname = 'sakila',
    host = 'localhost')


# List the tables available in this database.
 dbListTables(mysqlconnection)
```

Connecting R to MySql

Once the package is installed we create a connection object in R to connect to the database. It takes the username, password, database name and host name as input.

Create a connection Object to MySQL database.# We will connect to the sampel database named "sakila" that comes with MySql installation.mysqlconnection = dbConnect(MySQL(), user = 'root', password = '', dbname = 'sakila', host = 'localhost') # List the tables available in this database. dbListTables(mysqlconnection)

Querying the Tables

We can query the database tables in MySql using the function dbSendQuery(). The query gets executed in MySql and the result set is returned using the R fetch() function. Finally it is stored as a data frame in R.

```
# Query the "actor" tables to get all the rows.

result = dbSendQuery(mysqlconnection, "select * from actor")


# Store the result in a R data frame object. n = 5 is used to fetch first 5 rows.

data.frame = fetch(result, n = 5)

print(data.fame)
```

Query with Filter Clause

We can pass any valid select query to get the result.

result = dbSendQuery(mysqlconnection, "select * from actor where last_name = 'TORN"')

Fetch all the records(with n = -1) and store it as a data frame.

data.frame = fetch(result, n = -1)

print(data)

Updating Rows in the Tables

We can update the rows in a Mysql table by passing the update query to the dbSendQuery() function.

```
dbSendQuery(mysqlconnection, "update mtcars set disp = 168.5 where hp = 110")
```

After executing the above code we can see the table updated in the MySql Environment.

Inserting Data into the Tables

```
dbSendQuery(mysqlconnection,
    "insert into mtcars(row_names, mpg, cyl, disp, hp, drat, wt, qsec, vs, am, gear, carb)
    values('New Mazda RX4 Wag', 21, 6, 168.5, 110, 3.9, 2.875, 17.02, 0, 1, 4, 4)"
)
```

After executing the above code we can see the row inserted into the table in the MySql Environment.

Creating Tables in MySql

We can create tables in the MySql using the function dbWriteTable(). It overwrites the table if it already exists and takes a data frame as input.

```
# Create the connection object to the database where we want to create the table.
mysqlconnection = dbConnect(MySQL(), user = 'root', password = '', dbname = 'sakila',
    host = 'localhost')


# Use the R data frame "mtcars" to create the table in MySql.
# All the rows of mtcars are taken inot MySql.
dbWriteTable(mysqlconnection, "mtcars", mtcars[, ], overwrite = TRUE)
```

Dropping Tables in MySql

We can drop the tables in MySql database passing the drop table statement into the dbSendQuery() in the same way we used it for querying data from tables.

dbSendQuery(mysqlconnection, 'drop table if exists mtcars')

After executing the above code we can see the table is dropped in the MySql Environment.

JSON file stores data as text in human-readable format. Json stands for JavaScript Object Notation. R can read JSON files using the rjson package.

Install rjson Package

In the R console, you can issue the following command to install the rjson package.

install.packages("rjson")

Input Data

Create a JSON file by copying the below data into a text editor like notepad. Save the file with a .json extension and choosing the file type as all files(*.*).

```
{
    "ID":["1","2","3","4","5","6","7","8" ],
    "Name":["Rick","Dan","Michelle","Ryan","Gary","Nina","Simon","Guru" ],
    "Salary":["623.3","515.2","611","729","843.25","578","632.8","722.5" ],

    "StartDate":[
"1/1/2012","9/23/2013","11/15/2014","5/11/2014","3/27/2015","5/21/2013",
        "7/30/2013","6/17/2014"],
    "Dept":[
"IT","Operations","IT","HR","Finance","IT","Operations","Finance"]
}
```

Read the JSON File

The JSON file is read by R using the function from JSON(). It is stored as a list in R.

```
# Load the package required to read JSON files.
library("rjson")


# Give the input file name to the function.
result <- fromJSON(file = "input.json")


# Print the result.
print(result)
```

Convert JSON to a Data Frame

We can convert the extracted data above to a R data frame for further analysis using the as.data.frame() function.

```
# Load the package required to read JSON files.
library("rjson")

# Give the input file name to the function.
result <- fromJSON(file = "input.json")

# Convert JSON file to a data frame.
json_data_frame <- as.data.frame(result)

print(json_data_frame)
```

R - Mean, Median & Mode

Statistical analysis in R is performed by using many in-built functions. Most of these functions are part of the R base package. These functions take R vector as an input along with the arguments and give the result.

The functions we are discussing in this chapter are mean, median and mode.

Mean

It is calculated by taking the sum of the values and dividing with the number of values in a data series.

The function mean() is used to calculate this in R.

Syntax

The basic syntax for calculating mean in R is −

```
mean(x, trim = 0, na.rm = FALSE, ...)
```

Following is the description of the parameters used −

x is the input vector.

trim is used to drop some observations from both end of the sorted vector.

na.rm is used to remove the missing values from the input vector.

Example

```
# Create a vector.
x <- c(12,7,3,4.2,18,2,54,-21,8,-5)

# Find Mean.
result.mean <- mean(x)
print(result.mean)
```

applying Trim Option

- When trim parameter is supplied, the values in the vector get sorted and then the required numbers of observations are dropped from calculating the mean.
- When trim = 0.3, 3 values from each end will be dropped from the calculations to find mean.
- In this case the sorted vector is (−21, −5, 2, 3, 4.2, 7, 8, 12, 18, 54) and the values removed from the vector for calculating mean are (−21,−5,2) from left and (12,18,54) from right.

```
# Create a vector.
x <- c(12,7,3,4.2,18,2,54,-21,8,-5)

# Find Mean.
result.mean <-  mean(x,trim = 0.3)
print(result.mean)
```

Median

The middle most value in a data series is called the median. The median() function is used in R to calculate this value.

Syntax

The basic syntax for calculating median in R is −

```
median(x, na.rm = FALSE)
```

Following is the description of the parameters used −

x is the input vector.

na.rm is used to remove the missing values from the input vector.

Example

```
# Create the vector.
x <- c(12,7,3,4.2,18,2,54,-21,8,-5)

# Find the median.
median.result <- median(x)
print(median.result)
```

Mode

The mode is the value that has highest number of occurrences in a set of data. Unike mean and median, mode can have both numeric and character data.

R does not have a standard in-built function to calculate mode. So we create a user function to calculate mode of a data set in R. This function takes the vector as input and gives the mode value as output.

Example

```
# Create the function.
getmode <- function(v) {
   uniqv <- unique(v)
   uniqv[which.max(tabulate(match(v, uniqv)))]
}
# Create the vector with numbers.
v <- c(2,1,2,3,1,2,3,4,1,5,5,3,2,3)

# Calculate the mode using the user function.
result <- getmode(v)
print(result)
# Create the vector with characters.
charv <- c("o","it","the","it","it")
# Calculate the mode using the user function.
result <- getmode(charv)
print(result)
```

Linear Regression

- Regression analysis is a very widely used statistical tool to establish a relationship model between two variables. One of these variable is called predictor variable whose value is gathered through experiments. The other variable is called response variable whose value is derived from the predictor variable.

- In Linear Regression these two variables are related through an equation, where exponent (power) of both these variables is 1. Mathematically a linear relationship represents a straight line when plotted as a graph. A non-linear relationship where the exponent of any variable is not equal to 1 creates a curve.

- The general mathematical equation for a linear regression is –

- $y = ax + b$

Linear Regression

```
library(MASS)
library(plyr)
Interaction terms in regression
# Building up the familiar birthwt data...

# Rename the columns to have more descriptive names
colnames(birthwt) <- c("birthwt.below.2500", "mother.age", "mother.weight",
    "race", "mother.smokes", "previous.prem.labor", "hypertension", "uterine.irr",
    "physician.visits", "birthwt.grams")

# Transform variables to factors with descriptive levels
birthwt <- transform(birthwt,
            race = as.factor(mapvalues(race, c(1, 2, 3),
                        c("white","black", "other"))),
            mother.smokes = as.factor(mapvalues(mother.smokes,
                        c(0,1), c("no", "yes"))),
            hypertension = as.factor(mapvalues(hypertension,
                        c(0,1), c("no", "yes"))),
            uterine.irr = as.factor(mapvalues(uterine.irr,
                        c(0,1), c("no", "yes")))
            )
```

lm()

Function

(a) Run a linear regression to better understand how birthweight varies with the mother's age and smoking status (do not include interaction terms).

```
# Run regression model
birthwt.lm <- lm(birthwt.grams ~ mother.age + mother.smokes, data = birthwt)
# Output coefficients table
summary(birthwt.lm)
```

(b) What is the coefficient of mother.age in your regression? How do you interpret this coefficient?

```
coef(birthwt.lm)["mother.age"]
age.coef <- round(coef(birthwt.lm)["mother.age"], 1)
```

(c) How many coefficients are estimated for the mother's smoking status variable? How do you interpret these coefficients?

```
coef(birthwt.lm)["mother.smokesyes"]
smoke.coef <- abs(round(coef(birthwt.lm)["mother.smokesyes"], 1))
```

predict() Function

The basic syntax for predict() in linear regression is –

predict(object, newdata)

Following is the description of the parameters used –

object is the formula which is already created using the lm() function.

newdata is the vector containing the new value for predictor variable.

Predict the weight of new persons

```
# The predictor vector.
x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)

# The resposne vector.
y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)

# Apply the lm() function.
relation <- lm(y~x)

# Find weight of a person with height 170.
a <- data.frame(x = 170)
result <-  predict(relation,a)
print(result)
```

Visualize the Regression Graphically

```
# Create the predictor and response variable.
x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)
y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)
relation <- lm(y~x)

# Give the chart file a name.
png(file = "linearregression.png")

# Plot the chart.
plot(y,x,col = "blue",main = "Height & Weight Regression",
abline(lm(x~y)),cex = 1.3,pch = 16,xlab = "Weight in Kg",ylab = "Height in cm")

# Save the file.
dev.off()
```

Multiple Regression

Multiple regression is an extension of linear regression into relationship between more than two variables. In simple linear relation we have one predictor and one response variable, but in multiple regression we have more than one predictor variable and one response variable.

The general mathematical equation for multiple regression is −

y = a + b1x1 + b2x2 +...bnxn

Following is the description of the parameters used −

y is the response variable.

a, b1, b2...bn are the coefficients.

x1, x2, ...xn are the predictor variables.

We create the regression model using the lm() function in R. The model determines the value of the coefficients using the input data. Next we can predict the value of the response variable for a given set of predictor variables using these coefficients.

lm() Function

This function creates the relationship model between the predictor and the response variable.

Syntax

The basic syntax for lm() function in multiple regression is –

lm(y ~ x1+x2+x3...,data)

Following is the description of the parameters used –

formula is a symbol presenting the relation between the response variable and predictor variables.

data is the vector on which the formula will be applied.

```
phys.visit.binned <- birthwt$physician.visits

phys.visit.binned[phys.visit.binned >= 3] <- "3.or.more"

birthwt <- transform(birthwt, phys.visit.binned = as.factor(phys.visit.binned))

birthwt$phys.visit.binned


summary(aov(birthwt.grams ~ phys.visit.binned, data = birthwt))
```

Anova

Chi Square Test

Chi-Square test is a statistical method to determine if two categorical variables have a significant correlation between them. Both those variables should be from same population and they should be categorical like – Yes/No, Male/Female, Red/Green etc.

For example, we can build a data set with observations on people's ice-cream buying pattern and try to correlate the gender of a person with the flavor of the ice-cream they prefer. If a correlation is found we can plan for appropriate stock of flavors by knowing the number of gender of people visiting.

Syntax

The function used for performing chi-Square test is chisq.test().

The basic syntax for creating a chi-square test in R is –

chisq.test(data)

Chi Square Test

Following is the description of the parameters used −

data is the data in form of a table containing the count value of the variables in the observation.

Example

We will take the Cars93 data in the "MASS" library which represents the sales of different models of car in the year 1993.

```
library("MASS")

print(str(Cars93))
```

Load the library.

```
# Load the library.
library("MASS")

# Create a data frame from the main data set.
car.data <- data.frame(Cars93$AirBags, Cars93$Type)

# Create a table with the needed variables.
car.data = table(Cars93$AirBags, Cars93$Type)
print(car.data)

# Perform the Chi-Square test.
print(chisq.test(car.data))
```

Pie Charts

R Programming language has numerous libraries to create charts and graphs. A pie-chart is a representation of values as slices of a circle with different colors. The slices are labeled and the numbers corresponding to each slice is also represented in the chart.

In R the pie chart is created using the pie() function which takes positive numbers as a vector input. The additional parameters are used to control labels, color, title etc.

Syntax

The basic syntax for creating a pie-chart using the R is –

pie(x, labels, radius, main, col, clockwise)

Following is the description of the parameters used −

x is a vector containing the numeric values used in the pie chart.

labels is used to give description to the slices.

radius indicates the radius of the circle of the pie chart.(value between −1 and +1).

main indicates the title of the chart.

col indicates the color palette.

clockwise is a logical value indicating if the slices are drawn clockwise or anti clockwise.

Example

A very simple pie-chart is created using just the input vector and labels. The below script will create and save the pie chart in the current R working directory.

Create data for the graph.

```
# Create data for the graph.
x <- c(21, 62, 10, 53)
labels <- c("London", "New York", "Singapore", "Mumbai")

# Give the chart file a name.
png(file = "city.jpg")

# Plot the chart.
pie(x,labels)

# Save the file.
dev.off()
```

3D Pie Chart

```
# Get the library.

library(plotrix)

# Create data for the graph.

x <-  c(21, 62, 10,53)

lbl <-  c("London","New York","Singapore","Mumbai")

# Give the chart file a name.

png(file = "3d_pie_chart.jpg")

# Plot the chart.

pie3D(x,labels = lbl,explode = 0.1, main = "Pie Chart of Countries ")

# Save the file.

dev.off()
```

Graphics with ggplot2

- In R, type install.packages("ggplot2") to install the ggplot2 package.
- plot()
- The qplot() function can be used to create the most common graph types. While it does not expose ggplot's full power, it can create a very wide range of useful plots. The format is:
- qplot(x, y, data=, color=, shape=, size=, alpha=, geom=, method=, formula=, facets=, xlim=, ylim= xlab=, ylab=, main=, sub=)

- Everyone: Download workshop materials:
- Download materials from http://tutorials.iq.harvard.edu/R/Rgraphics.zip

- Extract the zip file containing the materials to your desktop

  Workshop notes are available in .hmtl format. Open a file browser, navigate to your desktop and open Rgraphics.html

- **plot vs qplot**

Here's how the default scatterplots look in ggplot compared to the base graphics. We'll illustrate things using the Cars93 data from the MASS library.

```
with(Cars93, plot(EngineSize, MPG.city)) # Base graphics qplot(x=EngineSize, y=MPG.city, data=Cars93) # using qplot from ggplot2
```

Remember how it took us some effort last time to add color coding, use different plotting characters, and add a legend? Here's the qplot call that does it all in one simple line.

```
qplot(x=EngineSize, y=MPG.city, data=Cars93, colour=Cylinders, shape=Cylinders, xlab = "Engine size (litres)", ylab = "Fuel economy (MPG city)" )
```

This way you won't run into problems of accidentally producing the wrong legend. The legend is produced based on thecolour and shape argument that you pass in. (Note: color and colour have the same effect.)

Graphics with ggplot2

Plotting the Cars93 data

ggplot function

The ggplot2 library comes with a dataset called diamonds. Let's look at it

```
dim(diamonds)
head(diamonds)
library(maps)
head(USArrests)
diamond.plot <- ggplot(data=diamonds, aes(x=carat, y=price))
diamond.plot + geom_point()
diamond.plot + geom_point(size = 0.7, alpha = 0.3)
```

If we wanted to let point color depend on the color indicator of the diamond, we could do so in the following way.

- diamond.plot <- ggplot(data=diamonds, aes(x=carat, y=price, colour = color))
- diamond.plot + geom_point()

es the Cars93 dataset from the MASS package.

(a) Use qplot to create a scatterplot with Price on the y-axis and EngineSize on the x-axis.

```
qplot(x = EngineSize, y = Price, data = Cars93)
```

(b) Repeat part (a) using the ggplot function and geom_point() layer.

```
ggplot(Cars93, aes(x = EngineSize, y = Price)) + geom_point()
```

(c) Repeat part (b), but this time specifying that the color mapping should depend on Type and the shape mapping should depend on DriveTrain.

```
ggplot(Cars93, aes(x = EngineSize, y = Price, colour = Type, shape = DriveTrain)) + geom_point()
```

Graphics with ggplot2

- Here's how we can get a headmap of Murder rates (per 100,000 population) on a map of the US.
- # Create data frame for map data (US states)
- states <- map_data("state")
- # Here's what the states data frame looks like str(states)
- # Make a copy of the data frame to manipulate

  arrests <- USArrests
- Convert everything to lower case

  names(arrests) <- tolower(names(arrests)) arrests$region <- tolower(rownames(USArrests))

Merge the map data with the arrests data based on region

choro <- merge(states, arrests, sort = FALSE, by = "region")

choro <- choro[order(choro$order),]

Plot a map, filling in the states based on murder rate

qplot(long, lat, data = choro, group = group, fill = murder, geom = "polygon")

WorkShop2

- **(a)** Construct histograms of MPG.highway, one plot for each Origin category.
- **(b)** Using the Cars93 data and the t.test() function, run a t-test to see if average MPG.highway is different between US and non-US vehicles.

Workshop3

- **ANOVA with birthwt data**
- **(a)** Create a new factor that categorizes the number of physician visits into three levels: 0, 1, 2, 3 or more.
- **(b)** Run an ANOVA to determine whether the average birth weight varies across number of physician visits.
- The p-value is greater than 0.05, so the variation in birthweight across number of physician visits is not statistically significant.

การอ่านไฟล์จาก **xml** ไฟล์

- การอ่านไฟล์จาก **xml** ไฟล์
- ```{r}
- library(xml2)
- library(rvest)
- webtest<-read_html("http://www.imdb.com/title/tt1961175/?ref_=inth_ov_tt")

library(ggplot2)

- library(ggplot2)
- library(MASS) # Useful for data sets
- library(plyr) # We'll need mapvalues
- qplot(x=EngineSize, y=MPG.city, data=Cars93,
- colour=Cylinders,
- shape=Cylinders,
- xlab = "Engine size (litres)",
- ylab = "Fuel economy (MPG city)"
-)

Example

- qplot(x =Max.Price, data = Cars93, facets = ~Origin, geom = "histogram", fill = Origin, binwidth = 2)
- diamond.plot2 <- ggplot(data=diamonds, aes(x=carat, y=price))
- diamond.plot2 + geom_line()
- diamond.plot + geom_point(size = 0.7, alpha = 0.3)
- library(ggplot2)
- diamond.plot <- ggplot(data=diamonds, aes(x=carat, y=price, colour = color))
- diamond.plot + geom_point()
- diamond.plot <- ggplot(data=diamonds, aes(x=carat, y=price, colour = color))
- diamond.plot + geom_point() + facet_wrap(~cut)

Decision Tree

- library(party)
- print(head(readingSkills))

```
input.dat <- readingSkills[c(1:105),]

# Give the chart file a name.
png(file = "decision_tree.png")

# Create the tree.
  output.tree <- ctree(
  nativeSpeaker ~ age + shoeSize + score,
  data = input.dat)

# Plot the tree.
plot(output.tree)

# Save the file.
dev.off()
```

K-Nearest

- # Read data
- data1 = read.csv('USPresidentialData.csv')
- View(data1)

```
 # load library
   library(caret)
   library(e1071)

# Transforming the dependent variable to a factor
   data1$Win.Loss = as.factor(data1$Win.Loss)
```

```
#Partitioning the data into training and validation data
set.seed(101)
index = createDataPartition(data1$Win.Loss, p = 0.7, list = F )
train = data1[index,]
validation = data1[-index,]
# Explore data
dim(train)
dim(validation)
names(train)
head(train)
head(validation)
 # Setting up train controls
   repeats = 3
   numbers = 10
   tunel = 10
   set.seed(1234)
   x = trainControl(method = “repeatedcv”,
                    number = numbers,
                    repeats = repeats,
                    classProbs = TRUE,
                    summaryFunction = twoClassSummary)
```

```
model1 <- train(Win.Loss~. , data = train, method = “knn”,
                preProcess = c(“center”,”scale”),
                trControl = x,
                metric = “ROC”,
                tuneLength = tunel)
  # Summary of model
  model1
  plot(model1)
# Validation
  valid_pred <- predict(model1,validation, type = “prob”)

  #Storing Model Performance Scores
  library(ROCR)
  pred_val <-prediction(valid_pred[,2],validation$Win.Loss)

  # Calculating Area under Curve (AUC)
  perf_val <- performance(pred_val,”auc”)
  perf_val

  # Plot AUC
  perf_val <- performance(pred_val, “tpr”, “fpr”)
  plot(perf_val, col = “green”, lwd = 1.5)

  #Calculating KS statistics
  ks <- max(attr(perf_val, “y.values”)[[1]] – (attr(perf_val, “x.values”)[[1]]))
  ks
```

```
newiris <- iris
newiris$Species <- NULL
(kc <- kmeans(newiris, 3))
K-means clustering with 3 clusters of sizes 38, 50, 62
table(iris$Species, kc$cluster)
plot(newiris[c("Sepal.Length", "Sepal.Width")], col=kc$cluster)
points(kc$centers[,c("Sepal.Length", "Sepal.Width")], col=1:3,
pch=8, cex=2)
```

```
library(lattice)
library(ggplot2)
library(caret)
set.seed(7267166)
trainIndex=createDataPartition(mydata$prog,p=0.7)$Resample1
train=mydata[trainIndex, ]
test=mydata[-trainIndex, ]

## check the balance
print(table(mydata$prog))
```

```
#Getting started with Naive Bayes in mlr
#Install the package
#install.packages(“mlr”)
#Loading the library
library(mlr)

#Create a classification task for learning on Titanic Dataset and specify the target feature
task = makeClassifTask(data = Titanic_dataset, target = "Survived")

#Initialize the Naive Bayes classifier
selected_model = makeLearner("classif.naiveBayes")

#Train the model
NB_mlr = train(selected_model, task)

#Read the model learned
NB_mlr$learner.model

#Predict on the dataset without passing the target feature
predictions_mlr = as.data.frame(predict(NB_mlr, newdata = Titanic_dataset[,1:3]))

table(predictions_mlr[,1],Titanic_dataset$Survived)<span style="font-family: 'Noto Serif', serif"><span></span></span>
```

Q/A